Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Dynamic Network Twins to Improve Threat Intelligence

Author:
Sergio Miguez Aparicio

Supervisor:
Sergio Maffeis

Submitted in partial fulfillment of the requirements for the MSc degree in MSc Computing (Security and Reliability) of Imperial College London

September 2024

Abstract

This project introduces an innovative approach to cybersecurity defence and threat intelligence by integrating multiple techniques to create a system that dynamically isolates attackers within a replicated network environment and facilitates their tracking. Unlike conventional methods, which rely on static honeypots or simple network segmentation, this prototype offers real-time replication of physical devices connected to the attacker's network, generating a honeynet of Linux Container (LXC) hosts. This setup effectively isolates and redirects the attacker's malicious activity to the replicated network through VLAN segmentation, ensuring minimal disruption to other devices.

Additionally, the system employs a secure Software-Defined Network (SDN) channel within the honeynet, preventing attackers from detecting real-time tracking and data collection. Combining these technologies and adequate design decisions results in a resource-efficient, adaptable solution that can be customised to various network topologies. The project highlights its potential to enhance threat intelligence and gather critical insights into attacker behaviour while addressing the challenges of scalability and resource demands in network virtualisation.

Key Words: Cybersecurity, Honeynet, Network Virtualisation, Attacker Isolation, Threat Intelligence.

Acknowledgements

I want to express my deepest gratitude to my supervisor, Sergio Maffeis, for his trust and continuous support throughout this project. His willingness to allow me the freedom and total flexibility to explore a wide range of topics of personal interest was invaluable. Moreover, his guidance was essential in shaping my research and in the successful development of this prototype. I am deeply thankful for his mentorship and encouragement.

Abbreviations

APT: Advanced Persistent Threat
DoS: Denial of Service
DPI: Deep Packet Inspection
EDR: Endpoint Detection and Response
ISP: Internet Service Provider
IDS: Intrusion Detection System
NGFW: Next Generation Firewall
SDN: Software Defined Network
OSI: Open System Interconnection
TTPs: Tactics, Techniques & Procedures
UTM: Unified Threat Management
VLAN: Virtual Local Area Network
WAF: Web-Application Firewall
ZTA: Zero Trust Architecture

Contents

Chapter 1

Introduction

In the contemporary digital landscape, the proliferation of cyber threats has made cybersecurity one of the most relevant areas in computing. The profound impact of cyber-attacks on individuals, businesses, and nations has driven this field to evolve rapidly and adapt, responding to daily emerging hazards as society increasingly becomes dependent on interconnected systems. The attack surface has expanded dramatically, making protecting confidential information from customers and companies a complex yet vital task for the survival and reputation of any business. Furthermore, critical infrastructure has emerged as a prime target, especially given the current geopolitical climate, where attacks on these strategic assets can devastate entire regions or countries. Consequently, investing in robust cyber-defences to protect individuals, businesses, and infrastructure ensures stability, trust, and continued development.

Firewalls have traditionally been one of the primary defence mechanisms for preventing unauthorised incoming and outgoing connections within a network. These systems employ a set of rules and policies to filter network packets based on their source and destination addresses and connection ports (1), ensuring that only legitimate or approved connections are allowed. By serving as a barrier between trusted internal networks and untrusted external networks, firewalls play a crucial role in maintaining network security. However, the sharp increase in cyberattacks in recent decades has driven the evolution and enhancement of these ”simple” filters. For instance, Cloudflare reported a 117% year-over-year increase (2). They also observed a 3370% growth of attacks against Taiwan compared to the previous year, which they related to the upcoming general elections and conflicts with China. CrowdStrike 2024 Global Threat Report (3) states that adversaries are now aware and target cloud businesses, where the detected intrusions are increasing by 75% year-over-year. Moreover, they also reported that attackers are moving into faster and more effective tactics than the “traditional” malware approach to gain access to targets. In 2023, 75% of attacks were malware-free, while in 2019, it represented 40%. Identity attacks, such as social engineering or phishing strategies, or the active exploitation of vulnerabilities, drastically reduce the time per attack and allow them to perform more attacks. Finally, cybercrime costs were estimated to be $6 trillion in 2021 and anticipated to increase to $10.5 trillion annually worldwide by 2025

(4). This has driven the development and deployment of advanced cybersecurity solutions.

In addition to firewalls, other security systems, such as Endpoint Detection and Response (EDR) tools and decoy honeypots, have become integral in modern cybersecurity strategies. Most systems are designed to isolate and disconnect the compromised devices when suspicious behaviour or a breach within a network is detected (5). Other services like CrowdStrike's Falcon Adversary Intelligence (6) detect and automatically change the credentials to prevent future access (3). This approach effectively limits the damage. However, it may miss valuable opportunities for deeper data gathering about attackers' tactics, techniques, and procedures (TTPs) and doing better threat intelligence.

This project proposes an alternative approach, focusing on extending the interaction period with the attacker while maintaining the security of the rest of the network. The primary aim of this project is to create a robust and automated infrastructure that serves as the foundation for advanced threat intelligence tools to analyse these adversarial techniques effectively. By creating dynamically a virtual replica of the network and isolating the attacker within it, this system aims to deceive adversaries into continuing their operations, making them believe they have not been detected. Thereby providing a unique opportunity to study their behaviour in a controlled and isolated environment.

The system uses Proxmox as its virtualisation environment and specifically employs virtual machines (VMs) and Linux Containers (LXCs) to recreate the compromised network environment efficiently. LXCs provide lightweight virtualisation, allowing the creation of virtual instances of hosts at a minimal hardware and time demand. The attacker is then isolated through VLAN segmentation, forcing the adversary's interactions to be redirected to the new LXC instances. This procedure allows flexible adaptation to different network topologies, and scalability can be achieved by adapting the prototype to support Q-in-Q or VXLAN. Proxmox's internal Software-Defined Network (SDN) capabilities can create a communication network between the LXCs, allowing tracking and sharing of the data gathered without alerting the attacker. This could allow tools to perform in-depth analysis and capture important details of the attack in real time without compromising the honeynet.

By combining a wide range of technologies, the prototype establishes a resilient, adaptable infrastructure upon which future threat intelligence tools can operate. This allows better real-time cyberattack analysis while maintaining network integrity and security.

Chapter 2

Background

2.1 Next Generation Firewalls

The rapidly evolving landscape of threats has shown traditional firewalls, which were primarily focused on filtering traffic based on a set of policies or rules depending on IP addresses, ports, and protocols, to be insufficient.

In response, firewalls have undergone significant enhancements, incorporating advanced features like packet payload inspection and machine learning, which are crucial for detecting sophisticated threats (7). These new capabilities extend far beyond traditional firewall filtering strategies. Gartner coined Next-Generation Firewalls (NGFWs) around 2009 to describe these advanced tools (8). Current NGFWs typically respond to detected threats by quarantining the malicious or infected devices within the network. According to Palo Alto Networks (5), these devices are isolated by blocking their communications with the internet and the rest of the network, notifying the administrator, and keeping the device isolated until forensic engineers can investigate.

NGFWs are a relevant evolution forward, incorporating multiple layers of security and sophisticated intelligence to provide comprehensive protection against modern cyber threats.

2.1.1 Evolution and Types of Firewalls

The Digital Equipment Cooperation (DEC) developed the first firewall in 1988 (9), designed to filter and protect networks from unwanted packets from certain source IP addresses. The system followed the Open System Interconnection (OSI) model (10) to inspect and filter traffic, focusing primarily on the Network Layer (Layer 3) for IP addresses and the Transport Layer (Layer 4) for TCP/UDP inspection. AT&T Bell Labs introduced the stateful firewall to improve filtering efficiency in 1990 (11) (12). In 1991, DEC advanced firewall technology by developing the application gateway firewall to detect network attacks. The concept of combining all firewalls into a single system was proposed in 2004 and named Unified Thread Management

(UTM). These would later, in 2009, evolve into the concept of Next-Generation Firewalls (NGFWs).

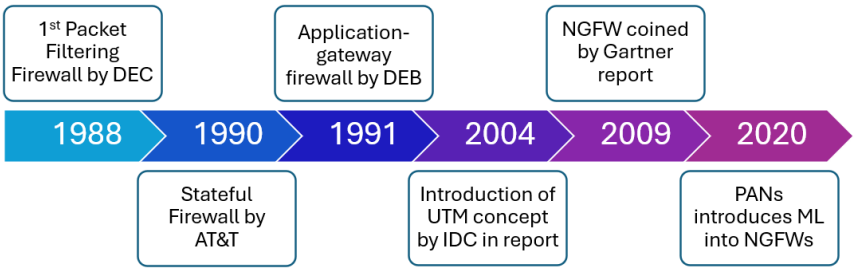

Figure 2.1: Evolution of Firewall Technology

Packet Filtering Firewalls

These simple firewalls filter incoming and outgoing packets based on criteria such as IP address, protocol, and port numbers, which correspond to OSI Layers 3 and 4. Each packet is inspected to determine and apply the appropriate filtering by following a set of rules and policies (13). Due to the simplicity of these firewalls, they do not inspect the payload at OSI Layer 7 and are vulnerable to IP spoofing (14), which can lead to incorrect filtering policies being applied (15). Additionally, they cannot authenticate users.

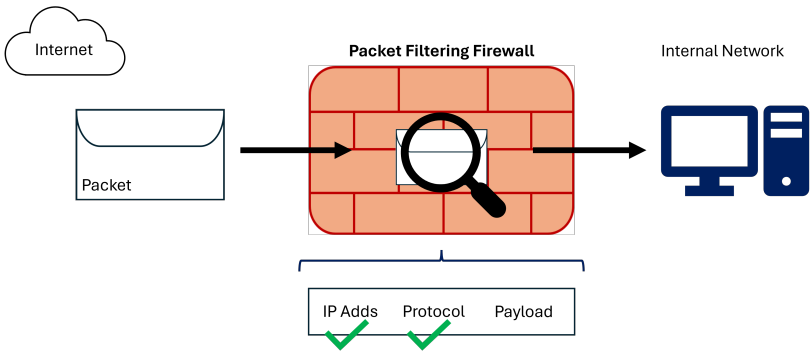

Figure 2.2: Packet Filtering Firewall

Stateful Firewalls

Unlike simple packet-filtering firewalls, stateful firewalls maintain a record of established communication channels between internal and external devices on the network (16). They review all traffic flowing in both directions and rely on the three-way TCP handshake to validate and track the channels. This is achieved through

an internal cache that stores information on every traffic flow. When a new connection is initiated and the initial SYN packet passes through, the firewall creates a new entry in the cache with the header information, such as the IP addresses and port numbers. Subsequent communication packets are checked against this firewall cache; the packets are allowed if the connection exists. Otherwise, the firewall will evaluate the new connection using the basic packet filtering rules to decide whether to add it to the cache. If a FIN packet terminates a connection or remains inactive for too long, the firewall removes the entry from the cache and further communication is blocked (14). In this way, stateful firewalls can ensure that only active and legitimate connections are maintained.

As a result, stateful firewalls are more secure and, especially, more efficient than simple packet filtering firewalls, as consecutive communication packets will not require the same inspection as the first one. However, this approach requires firewalls to include a cache, which limits the number of connections that can be tracked. This might become a problem if the firewall receives a Denial of Service (DoS) attack (17), as its table could become full. Like the packet filtering firewall, they do not inspect the payload or authenticate users.

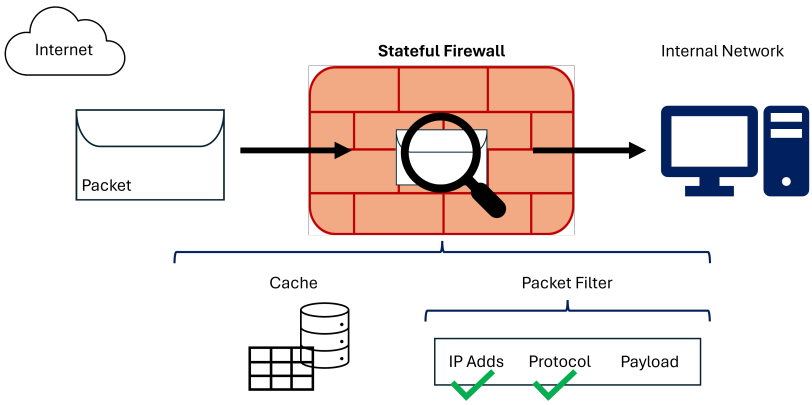

Figure 2.3: Stateful Firewall

Application-Gateway Firewalls

This firewall controls information flow at the application layer (Layer 7). In this case, the system does not filter packets based on rules; instead, it functions like a gateway that proxies the network traffic between clients and external resources. In this way, clients are prevented from directly accessing these and protecting them from potential malicious packets as these affect the gateway first rather than reaching the client(18).

However, there are drawbacks to using an application gateway firewall. Traffic throughput is notably penalised, as every packet is received and processed within

the gateway before being forwarded to the client. Furthermore, scalability is limited. Nevertheless, compared to the previous firewalls, it can authenticate users by receiving the clients' credentials (13).

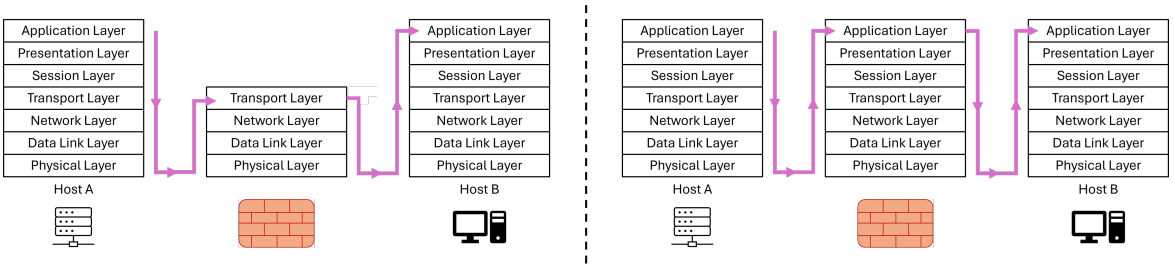

Figure 2.4: Comparing Stateless & Stateful Firewalls vs. Application-Gateway Firewalls

Circuit-Gateway Firewalls

These are similar to application-gateway firewalls as they operate as "gateways" but, in contrast, they operate at the Session Layer (Layer 5) of the OSI model and focus primarily on the connections between a client and an external resource without checking the payload (19). Circuit-gateway firewalls run as a proxy and forward TCP connections between devices. In this way, this type of firewall ensures that only legitimate connections are allowed to pass through. Moreover, it checks that responses match requests, improving the network's security.

Since the payloads are not being checked, circuit-gateway firewalls are said to be less secure than application-gateway firewalls, but this dramatically improves the traffic speed (8). Additionally, it is often used together with other firewalls.

Web Application Firewalls

A web application firewall (WAF) is similar to an application gateway firewall but is specifically designed to protect back-end web applications. It inspects the application layer (Layer 7) HTTP/HTTPS request packets and network traffic patterns. If it finds a suspicious packet or pattern, the WAF will block the connections between client and server (20).

As it is highly specialised, WAFs have an advantage over other more general-purpose firewalls, making them highly "effective" against web attacks such as cross-site scripting (XSS), SQL injection, and DDoS attacks (21). However, they respond poorly to unknown vulnerabilities due to their lack of flexibility and reliance on pattern recognition. Despite zero-day vulnerabilities and never-seen attacks easily going unnoticed, these are still a minority of the attacks web servers receive, as most are still vulnerable to more accessible, known attacks. Including Machine Learning has helped mitigate these issues, as proven by Kumar H (22).

Unified Threat Management

Unified Threat Management (UTM) solutions have surged to integrate different security technologies into a cohesive solution. According to Charles Kolodgy from International Data Corporation (IDC), UTMs must include Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS) and anti-virus functionalities in addition to traditional firewall capabilities (23). Other publications suggest that an effective UTM should incorporate an anti-bot system (24).

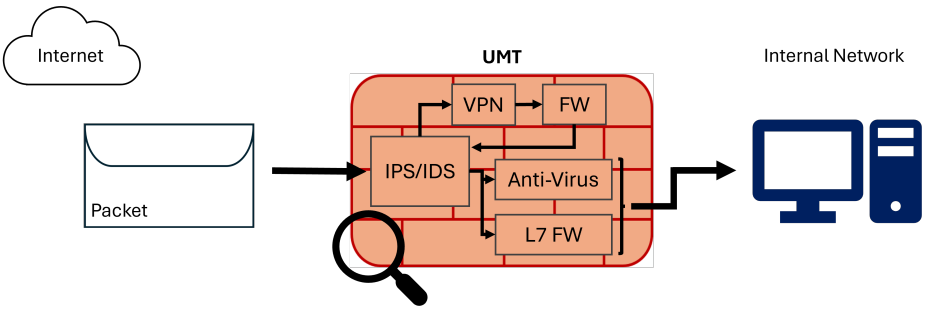

Figure 2.5: UTM Internals

By consolidating multiple security features into a single platform, UTMs aim to reduce the complexity of managing the tools separately while achieving better coordination and efficacy. Modern UTMs also often include additional security features such as virtual private networks (VPNs), content filtering, and spam filtering. While increasing the number of tools can enhance security, the UTM and the network's throughput performance can be negatively affected as more tools are included in the analysis. According to Synder (23), performance depends on the enabled features and the data type that is analysed, making it hard to report benchmarks.

Next Generation Firewalls

UTMs evolved and became a more sophisticated product. An NGFW has more advanced features and can identify and control traffic at all the OSI model layers, including tools specifically focused on deep packet inspection (7). As a result, they can identify which applications are sending packets regardless of port number and protocol used, identify the user and the user's identity, and recognise the real intentions of packets.

Despite being similar products to UTMs, NGFWs kept evolving. Currently, both products coexist in the market. UTMs focus on smaller and home networks, where performance is not critical, while NGFWs aim for enterprise-level environments.

2.1.2 Key Features of NGFW

Deep Packet Inspection

Next-Generation Firewalls innovated and focused in this area to scan and better assess the traffic flowing through them. Raza et al. (25) describe DPI as performing an X-ray scan of network packets to ensure safety and detect the application protocols being used. These methods involve analysing all layers above the transport layer (Layer 4) (7). While traditional firewalls can only filter packets based on rules using the header data, DPI systems used in NGFWs and industrial firewalls can inspect encrypted TLS traffic using man-in-the-middle (MitM) techniques and classify payloads.

The two major approaches to packet inspection are Signature-based DPI, which is effective but has issues detecting new threats (26), and Anomaly or Behaviour-based DPI, which checks for deviations from normal traffic patterns.

According to Heino et al. (7), each protocol has its properties, requiring specific analysis strategies that result in different false positive and negative results. They also point out that vendor-specific protocols are not openly disclosed, making it more difficult to analyse and draw conclusions about the intentions of the packets, especially when these packets are sending files that could be potentially malicious. As each protocol and application has its particularities, signature-based identification techniques can be used to recognise the different protocols sent across the network. On the other hand, the increase in application signatures in the last years caused by the rapid development of new systems and applications has added difficulties as DPI systems now must be compared against millions of application signatures or rules (27), drastically penalising DPI's performance.

DPI systems contain different modules that allow them to operate effectively (25). Initially, packets must be captured from the Network Interface Card (NIC) to prepare them for processing. This module must be an effective, lossless packet-capturing solution. A Data Plane Development Kit (DPDK) has been proposed as one of the best options. It is a set of open-source libraries and drivers which facilitate high-speed packet processing (28). While capturing packets, it generates and fills a table-like data structure that will be utilised for storage and subsequent processing by inspecting the packet headers and payload for patterns that could identify the application sending that data. The packet is then classified by protocol, and the traffic metadata is extracted. This is saved in a log file using the Internet Protocol Data Record (IPDR) form. The logs are then stored in a database, and after some processing, the data is displayed on a dashboard.

Hash fingerprinting involves extracting insights from packets and identifying the applications sending the information through hashing algorithms. Each protocol and application has its particularities reflected in the resulting hash. This mechanism is also widely used in endpoint security (29).

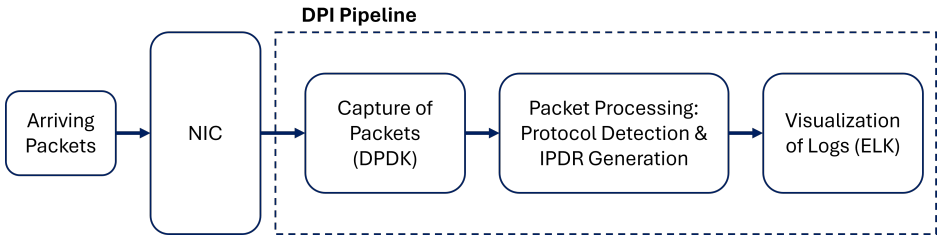

Figure 2.6: DPI packet processing pipeline

Heino et al. (7) discovered fingerprinting techniques for SSH, gQUIC, RDP and SMB protocols. They also mention that, due to the cutting-edge of the methods and protocols, no further research has been done regarding their identification accuracy. Moreover, they especially urge further study of server application protocols like JA3S or HASSHServer. These have not been adequately studied in the current literature, and NGFW-protecting server installations might be less effective in preventing attacks.

Sandboxing

Current NGFWs include sandboxing systems which allow the inspection of files and programs in isolated and controlled environments where potential damages can be reverted. This will enable them to detect advanced threats that traditional systems might miss. Cisco's NGFW, FirePOWER (30), and Forcepoint (31) incorporate this strategy to perform advanced malware scans. Forcepoint explains that its NGFW sends the hash of the file or zip to their sandbox servers for analysis. If the hash matches a previously analysed file, the sandbox server will immediately report the analysis results to the NGFW. Depending on its policies, the NGFW might block the file or let it through. If the hash is not found, the NGFW uploads a copy of the file or zips it to the sandbox server for analysis. At the end of the study, the NGFW receives the results and decides the appropriate actions based on its policies.

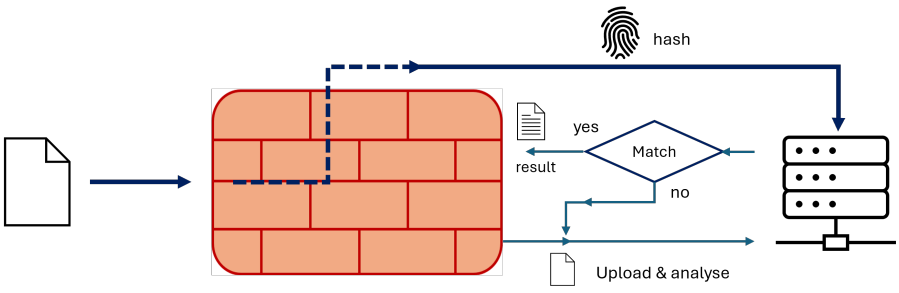

Figure 2.7: Sandbox analysis

Quarantining

If a malicious actor is detected in the network, the firewall must respond and prevent further damage. Quarantining involves isolating the affected device or segment to

prevent the spread of the threat while allowing for further analysis, which is crucial for a correct threat response. However, the IDS analysis can often flag unfamiliar behaviours as malicious, generating many false positives. Hooper (32) proposed using Network Quarantine Channels (NQCs), which, in essence, are a system that temporally handles suspicious packets from hosts without shutting down all network traffic from that device until it has been found malicious. This approach helps balance security and usability by avoiding unnecessary disruptions in network services. Further studies of this approach have been shown in (33) and (34).

Recent advancements in artificial intelligence have further improved the effectiveness of such quarantining strategies, boosted accuracy and reduced false alarms. Industrial providers also offer quarantining solutions as part of their NGFWs or as easy-to-incorporate products into existing network infrastructures. Currently, Cisco commercialises Software-Defined Access (SDA) (35) and Palo Alto Networks its Cortex XSOAR platform (5). Both solutions enable real-time threat response and quarantining of hosts.

2.2 Network Isolation and Segmentation

Network segmentation or compartmentalisation is crucial for having a secure network. Stawowski (36) proposed isolating resources according to their different "sensitivity" levels and distributing them into separate security zones.

2.2.1 Techniques for Network Segmentation

There are multiple methods to achieve compartmentalisation of networks. Among these, the most relevant to the project are:

Firewall policies

The network can be split into smaller segments by setting the correct rules in the firewall. However, the difficulty of this approach lies in writing and managing the rules, especially as these increase in number and complexity. Misconfigurations can easily occur, and extensive research has been conducted to automate and optimise rule development and orchestration (37) (38).

VLAN Segmentation

Virtual Local Area Network (VLAN) technology allows the identification of groups of hosts or applications regardless of their physical location in the LAN. The grouping can even connect devices that are placed under different network switches (39). VLANs are widely used to isolate network packets as they provide logical network segmentation, behaving as it was separate LANs (40).

Traditionally, routers create broadcast domains and forward packets to all hosts in a shared LAN segment, regardless of whether the device is the intended receiver. This broadcasting creates a lot of traffic, especially if many devices are connected. VLANs allow for network segmentation and broadcast containment (41). Moreover, using the abstraction layer that VLAN creates, network traffic between VLAN groups can be minimised and controlled (42), improving network security and simplifying firewall rules. Despite all this, the initial developer of Zero Trust Architecture, John Kindervag, states that VLANs are still highly insecure (43) as this network segmentation does not prevent attackers from moving between VLANs and gaining access to privileged information.

Due to misconfiguration or poor VLAN implementation, an attacker can bypass layer 2 restrictions that segment the network, gaining access to devices outside their VLAN. This type of exploit is known as VLAN hopping (44). To prevent this attack, it is essential to ensure that the trunk ports (those on a switch that connect to another switch or router) are correctly configured, limit the number of VLANs, and implement Access Control Lists (ACLs) to restrict access to specific VLANs.

Micro-segmentation

According to Cloudflare (45), through this technique, the network is highly divided into small sections, each with its security policies. The aim is to reduce the network's attack surface, preventing threads and breaches from compromising a large part of a network and preventing lateral movements. This strategy has become crucial for Zero Trust architecture solutions. Usually, micro-segmentation relies on Software Define Networks (SDN) to achieve the network granularity intended. This segmentation technology differs from VLANs in providing context regarding the applications communicating data and how the network traffic flows.

It was only recently that Mhaskar (46) formally defined network segmentation and proposed novel algorithms to build robust networks.

Zero Trust Architecture

Zero Trust Architecture (ZTA) is a security model that enforces security by default, assuming that any packet sent comes from an untrusted host. It eliminates the idea of having separate trusted (internal) and untrusted (external) networks (43), requiring authentication, authorisation and continuous validation before granting access to applications and data. ZTA denies by default, applies least privileged policy, and role-based access control (47). Compared to other network segmentation solutions, access to resources in ZTA is managed through credentials instead of having complex policies and rules. Depending on the level of security and number of hosts, zero-trust environments can be divided through micro-segmentation into smaller independent zero-trust networks, achieving highly granular control. However, according to John Kindervag (48), implementing ZTA is the starting point to achieving proper security in a network.

2.3 VLANs

After exploring the multiple network segmentation methodologies, VLANs were found to best suit the project's purposes. Therefore, further understanding this technique is convenient for the later proposed technical solution.

2.3.1 VLAN Port Tagging

Within a VLAN-enabled network, traffic can be either tagged or untagged.

- **Tagged VLANs** are used primarily on trunk ports, where each data packet is marked with a VLAN identifier, ensuring that it is routed to the correct VLAN as it traverses the network infrastructure. This tagging allows for seamless communication across multiple VLANs over a single physical link.

- **Untagged VLANs** operate without VLAN identifiers in their Ethernet frames and are typically used for ports that connect directly to end devices. Frames entering through these ports are automatically linked to a specific VLAN, as defined by the port configuration, allowing for straightforward network segmentation at the network's edge.

The Port VLAN IDs (PVID) set the default VLAN for any untagged traffic entering a port, ensuring that the packets can be returned without losing them.

To implement VLAN segmentation, the network interface on end devices must be configured to receive and send packets with a specific VLAN tag. Otsuka et al. (49) proposed a solution where VLAN tagging is handled by the switch itself, enabling the connection of devices that, at the time, could not use VLANs. This concept of enforcing isolation from a higher-level device in the network topology has been adapted and automated in this project to facilitate the seamless transition of an attacker from one network segment to another.

2.3.2 Native VLAN

A native VLAN is a particular VLAN on a switch that handles untagged traffic on a trunk port, as specified by the IEEE 802.1Q standard (50). When traffic is sent across a trunk link, the native VLAN is the VLAN that does not receive a VLAN tag. Typically, the native VLAN is set to VLAN 1 by default, though it can be configured to any other VLAN number according to network requirements.

The use of a native VLAN introduces several security risks. A primary concern is VLAN hopping, a technique where an attacker sends traffic that can trick a switch into forwarding frames to a different VLAN. This occurs because untagged traffic on a trunk link is automatically assigned to the native VLAN, potentially allowing malicious traffic to bypass VLAN boundaries and access other network parts (51).

Additionally, suppose the native VLAN is set to the default VLAN (often VLAN 1). In that case, there is an increased risk of exposing network management traffic to potential threats, as VLAN 1 is commonly reserved for network management protocols.

To mitigate the risks associated with native VLANs and following Cisco's VLANs best practices (52), it is advisable not to use the default VLAN as the native VLAN. Instead, network administrators should configure a separate, unused VLAN to serve as the native VLAN. This practice reduces the risk of VLAN hopping by isolating untagged traffic on a dedicated VLAN that does not carry sensitive data. Moreover, ensuring the native VLAN does not have management traffic can safeguard critical network operations against potential attacks.

2.3.3 Queue in Queue and VXLANs

Queue-in-Queue (Q-in-Q) and Virtual Extensible LANs (VXLANs) are advanced networking techniques extending traditional VLANs capabilities, providing enhanced scalability, flexibility, and traffic management in complex network environments.

Q-in-Q is a technique known as VLAN stacking that allows the encapsulation of a VLAN within another. Service providers use this double-tagging mechanism to transport customer VLANs over a shared network infrastructure (53). Commonly, the service provider sets the external VLAN tag to route the packages. In contrast, the customer uses the inner tag, enabling a clear separation between the provider's and the customer's network domains.

Double-tagging introduces the risk of double-tagging attacks (54), a VLAN hopping technique in which an attacker crafts packets with two VLAN tags. The first tag corresponds to a segment accessible by the attacker, while the second tag is for a restricted network. During routing, if a networking device, such as a router, removes the first tag, the packet may be forwarded to a second networking device. At this point, the packet could be misinterpreted as legitimate for the targeted VLAN. To prevent such attacks, meticulous configuration of network devices is crucial.

Virtual Extensible LAN (VXLAN) is a network virtualisation technique introduced in 2012 by a consortium of companies, including VMware and Cisco, to extend the capabilities of traditional VLANs and enable the creation of larger and more flexible virtual networks. VXLAN operates by encapsulating Layer 2 Ethernet frames within Layer 3 UDP packets, allowing for the extension of VLANs across multiple networks or data centres (55) (56). The encapsulation process in VXLAN enables seamless network segmentation, even across geographically dispersed locations, providing a high level of flexibility and scalability supporting up to 16 million unique network identifiers (VXLAN Network Identifiers or VNIs), compared to the 4,094 VLANs available with standard VLAN technology.

However, using VXLAN also brings challenges, particularly in the areas of net-

work performance (55), where encapsulation can increase the latency, and security (57), as it has not been fully tested or documented.

2.4 Software-Defined Networking

The installation and configuration of network devices require highly skilled engineers, particularly considering each device must be individually configured and is highly likely to have its interface and dashboard. This increases the complexity and the chances of human error (58). Network devices such as routers and switches traditionally handle data forwarding (data plane) and the decision-making process about where that data should go (control plane). Software Defined Networks (SDNs) is a technology that, by separating the control plane from the data plane in the network, allows for greater flexibility, easier management, and a reduction of operational costs, especially in large-scale and dynamic networks (59).

2.4.1 Benefits of SDN in Network Management

As explained by the Open Network Foundation (60), SDN centralises network control in a controller, which allows administrators to have a global view of the state of the network. The primary advantage of having a controller that manages the control plane is that the behaviour of the network can be directly programmed through a single software application. The SDN controller establishes communication with switches through a secure control plane using the OpenFlow (OF) protocol (61). This enables automation and faster deployment of new services and policies. Moreover, improved network administration efficiency and consistency leads to better policy enforcement and enhanced security (62). SDNs also allow administrators to selectively forward legitimate traffic to their destination while filtering anomalies and attacks in the network.

2.4.2 SDN in Network Security

Despite SDNs being said to improve security through their granular decision capabilities and improved administration, the technology only separates control from data. Therefore, previous security solutions must be adapted and integrated into this architecture.

Zerkane et al. (63) proposed a stateful firewall solution designed to be reactive, reducing the number of OpenFlow rules in data plane devices and mitigating some Denial of Service (DoS) attacks.

FLOWGUARD was proposed as an SDN-based framework that checks the network flow path for conflicts between firewalls and flow rules (64).

Kim et al. (62) proposed a distributed and flexible software-defined firewall system that dynamically performs firewall operations using SDN functionalities. They state that individual switches cannot make packet filtering decisions on their own in an SDN; however, this can be solved by deploying firewall rules as SDN flow rules and optimising the aggregate data traffic volume, given the limited capacity of the switches.

Rietz et al. (65) explain that ARP spoofing and poisoning can be prevented if the packets are no longer broadcasted but instead routed to the controller to detect attacks. The same method applies to DHCP attacks. Since the attacker does not receive the DHCP packets, they would not have the chance to send malicious replies. This approach requires integrating an ARP and DHCP server in the controller as OpenFlow applications. The server should answer ARP requests with virtual, non-existent MAC addresses so that each client only knows its own MAC and IP addresses. This way, other hosts' addresses and the network topology remain hidden.

Having a firewall at the network's entry point, which must process all communication packets, can become a performance bottleneck. In SDNs, the firewall could be centralised in the controller. However, Wang (66) studied the throughput difference between a traditional firewall and a distributed firewall (67) in the SDN architecture and demonstrated that distributed firewalls are still more efficient in this context. Interestingly, the study found that only the distributed firewall effectively intercepts abnormal data from the internal network. In addition, Ayodele et al. (68) further gather other related SDN solutions in their recent survey.

Finally, despite SDNs improving security, they are still vulnerable, as shown by Cherednichenko et al. (69). Specifically, SDNs can fail to maintain consistency of flow rules and are vulnerable to malicious software and vulnerabilities in the controller.

2.4.3 Dynamic Network Segmentation with SDN

The Robust Network and Segmentation (RNS) algorithm is widely used to generate a specific network topology as well as the firewall policies that each segment should implement in each switch to reconfigure the network dynamically (46). Modern networks are constantly changing as hosts are connected and disconnected frequently. With the topology changes, previous policies might no longer be applicable. Therefore, the RNS algorithm is one of the best solutions to control the network and ensure correct compartmentalisation dynamically. Alabbad et al. (70) studied three different implementations of the RNS in the SDN architecture and the viability of using it together with the Dynamic Configuration and Governance (DCG) plan. This layer is placed at the same level as the application plane. It enables dynamic creation and modification of network configurations and ensures compliance with network policies and security requirements.

Through traffic patterns, Dong et al. (71) could recognise user identities and automatically deploy management and network security policies in an SDN architecture. This shows that the dynamicity and security enforcement required to complete this prototype is possible.

2.5 Proxmox and Virtualisation

Proxmox Virtual Environment (Proxmox VE or PVE) is an open-source platform designed for managing virtualisation (72). It is a Type-1 hypervisor that runs directly on hardware based on Debian Linux and can run both Linux and Windows operating systems on x64 architecture. It allows combining two virtualisation technologies:

- KVM (Kernel-based Virtual Machine)
- LXC (Linux Containers)

Proxmox can be deployed on a single server or configured as a cluster with multiple nodes, enabling high availability (HA) of virtual servers (73). This HA is achieved using well-established Linux HA technologies, ensuring robust and reliable HA services. Virtual machines (VMs), containers, and other resources are managed through a web-based interface.

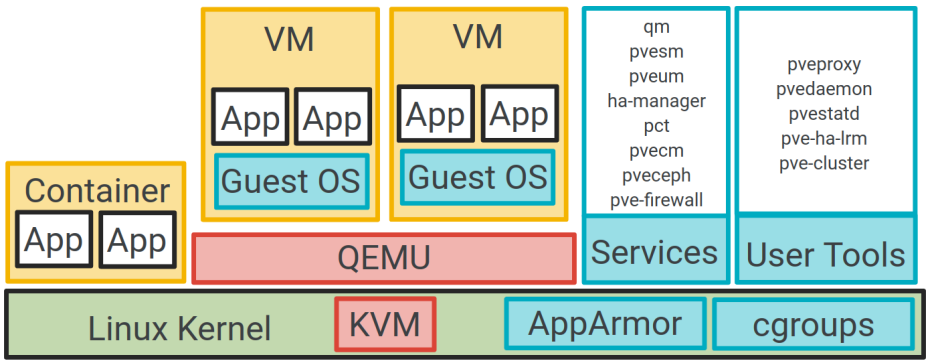

Figure 2.8: Proxmox architecture (72).

Managing Virtual machines (VMs), containers, and other resources through a user-friendly web interface.

Proxmox can be deployed on a single server or configured as a cluster with multiple nodes, enabling high availability (HA) of virtual servers. This HA is achieved using well-established Linux HA technologies, ensuring robust and reliable HA services as shown in Ljubojevic et al. (74)

It includes a firewall that enables packet filtering on any VM or container interface, with rules that can be conveniently grouped into “security groups”, IP sets, aliases and firewall macros. It has full support for IPv4 and IPv6 protocols, and IPv6

works transparently, making the firewall easier to maintain as there is no need for a different set of rules depending on the protocol. Proxmox offers a straightforward method for deploying and managing Software-Defined Networks (SDNs). Creating Virtual Zones and Networks (VNets) and subnets allows the network to be separated and finely-grained controlled through a software-controlled configuration.

An alternative to using Proxmox would be to use KVM directly. However, as shown by Dordevic et al. (75), KVM would only achieve better performance if the node were dedicated to performing web server workloads. As this is not the case, Proxmox offers more significant advantages and is one of the best options for implementing the current project.

2.5.1 LXCs and VMs

The two virtualisation technologies available in Proxmox are:

- **Linux Containers (LXCs)** are lightweight virtualisation environments which allow running multiple isolated Linux systems on a single host as if these were containers. They use the host's kernel to optimise resources and reduce the deployment time. LXCs are commonly used in environments where high performance, low latency and efficient resource utilisation are critical.

Linux Containers enforce security measurements using namespaces and control groups (cgroups), which provide isolation and resource management. However, as these share the host's kernel, they are considered to have a higher risk when compared with full virtualised machines (VMs) (76).

- **Virtual Machines (VMs)** are full virtualisation solutions where the hardware is emulated and runs an independent (and possibly different) operating system from the host. This approach gives them better isolation at the expense of consuming more resources and having a higher overhead than running LXCs. As a result, VMs have longer deployment times and lower performance.

Lightweight VMs, also known as micro VMs (77), have been proposed as a middle-ground solution between full VMs and containers. These intend to combine more robust isolation from the VMs while achieving efficiency and speed from containers. They reduce their code surface while keeping their independent kernel. It includes the essential components required to run a specific application (78), reducing the needed resources and being more secure as there are fewer chances for vulnerabilities (79). On the other side, as their footprint has been reduced, they have quicker boot times and are more efficient. Micro VMs have not been directly used for this prototype, but it might be an interesting solution to explore to ensure high performance while reducing security risks.

Security Comparison

Commonly, containers rely on creating new namespaces to achieve their isolation. As a result, root privileges are required to create a container. This can be a big security risk if an attacker successfully escapes from the container. As a result, the adversary would immediately have root permissions at the host (80). Fortunately, Proxmox includes a configuration option to create unprivileged containers, which allows the container to be mapped to an unprivileged user on the host.

Other container solutions (81) also allow further isolation features to reduce the risks and improve the isolation by using hypervisors isolation (82) and kernel isolation (83). However, these techniques are not supported by the Proxmox LXCs version.

LXCs as Cyber Defence Tools

Linux Containers (LXCs) have been widely used for various purposes within the cybersecurity field.

LXCs are commonly used to recreate environments for practising cybersecurity procedures under realistic conditions (84) (85), to run honeypots that detect and analyse malware (86), and more recently, as decoy honeypots. In this decoy honeypot setup, LXCs are the front-end that interacts with attackers, while centralised servers handle resource-intensive processes (87) (88). LXCs have also been employed to create honeypots in IoT environments (89).

The idea of deploying honeynets using LXCs has been around for some time. Memary et al. published (90), which was further explored in later studies (91). These papers inspired Kong et al. (92) to develop an automated honeynet deployment strategy, successfully reducing attacks by 83% in container-based cloud environments.

2.6 Dynamic Network Replication

Dynamic Network Cloning is a technique which involves the real-time creation and deployment of identical or near-identical network environments. The network's configuration, topology and behaviour are replicated in this process. Its main purpose is to create an isolated network replica that can be used for various purposes without affecting the live environment.

On the other side, Digital Twins are the digital equivalent of a physical (real) product. Grieves introduced this concept in (93) and can be brought into this context by trying to create a virtual version of the network and its devices. Network twins are currently under research by standard organisations such as IETF (94), and ITU

(95), among others. Springer has recently (2024) published about Digital Twins (96). However, it is not focused on replicating computer networks.

2.6.1 Methods for Network Replication

Most existing publications in this area focus on modelling networks rather than creating an exact "clone" of a real-world network. Peuster et al. (97) adapted the Mininet (98) platform to include Docker containers as hosts (99), enabling the recreation of network topologies. While tools are designed to virtualise physical devices (100) (101), these are primarily used for disaster recovery or cloning virtual hosts. However, the full virtual replication of existing physical networks, including hosts and processes, is not widely documented. This may be due to the high computational requirements and the perceived lack of practical applications.

Recently, Charan et al. (102) demonstrated that creating authentic organisational environments using virtual machines can effectively mislead attackers, suggesting this area could attract more research attention. A significant contribution to the concept of digital network twins comes from Kim Gyuyeong, who proposed a dynamic request cloning system that decides when to duplicate requests based on server conditions (103).

2.7 Threat Intelligence and Analysis

Threat intelligence is crucial in modern cybersecurity to get insights into adversarial tactics, techniques, and procedures (TTPs). It helps anticipate, detect and respond rapidly and effectively to threats (104). Using honeypots or performing network traffic analysis makes it possible to understand the threats better and develop solutions to mitigate the vulnerabilities (105). Moreover, in recent years, data collection has evolved from passive to active and dynamic gathering, where security teams include threat intelligence feeds to further prepare against adversities (106).

2.7.1 Techniques for Gathering and Analysing Threat Data

Honeypots are used as "traps" and defence mechanisms to detect attackers. These are fake systems that monitor how other hosts interact with them. They use fake data and represent no value for an attacker (107). There are multiple levels of interaction with the honeypot. Initially, researchers used low-interaction honeypots that emulated important services such as SSH, HTTP, and FTP. Still, these were shown to be inefficient for gaining a deep understanding of the attackers' capabilities and strategies. Additionally, studies have shown that attackers have learned how to identify static honeypots that have been deployed for a long period, leading to a significant reduction of attacks and data gathering (108).

According to Bartwal et al. (109), attackers do not interact with honeypots in most current cases unless they are dynamically deployed. As a countermeasure, Bartwal suggests a new Security Orchestration, Automation, and Response (SOAR) engine capable of dynamically deploying custom honeypots inside an internal network based on the attacker's behaviour. The dynamic deployment of honeypots increased the average attacker's engagement time from 102 to 3148 seconds.

As hardware capabilities evolve and virtualisation has become more affordable, launching multiple honeypots at once has become possible. This network of connected honeypots is known as honeynets.

Well-designed honeynets can achieve high attack engagement over a long period. This was demonstrated by Charan et al. (102), who tailored a network of honeypots to recreate the entire operational mode of an organisation, specifically targeting APT-40, APT-28 and Lazarus (APT-38). After leaving the honeynet running for 100 days, the results showed that sophisticated attacks were performed in long cyber-attack campaigns, proving that this methodology can keep engagement and record valuable information.

Alani (110) used a similar method to encapsulate malicious actors inside a honeynet while making them think they are connecting to the correct network and integrating this procedure with SDNs.

Chapter 3

Contribution

This chapter provides an in-depth description of the development process to complete this prototype. It is structured to cover the motivations and technical configurations. Each section offers a comprehensive understanding of how the system was implemented. Figure 3.1 shows a roadmap of the different sections covered in this chapter.

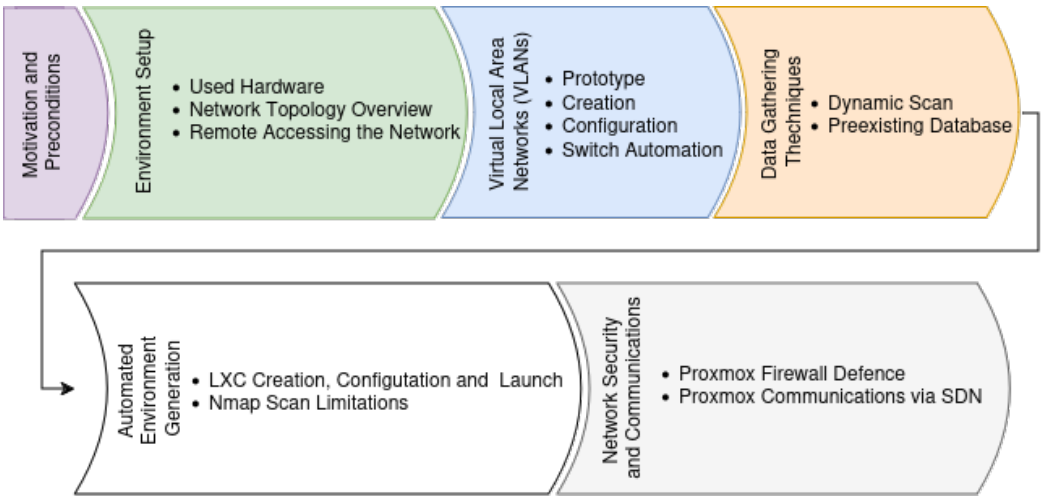

Figure 3.1: Contributions roadmap.

The first section, 3.1 Motivations and Preconditions, outlines the driving factors behind the project and covers the Motivation, where the rationale and goals of the project are stated, followed by the Assumptions and Permissions, which establish the scope, clear boundaries and necessary conditions for the project's successful development and deployment.

The 3.2 Environment Setup section describes the infrastructure used and the environment in which our prototype runs. It starts by detailing the Used Hardware and follows with a Network Topology Overview, which describes how the network components are organised. The Accessing the Network section explains our approach to

connecting securely to the remote network where the prototype is running.

In the 3.3 Virtual Local Area Networks (VLANs) section, the design and configuration of VLANs are discussed. These represent a vital role in the network segmentation and isolation used in the prototype. This includes the VLANs Prototype Overview and the steps involved in VLAN Creation and VLAN Configuration. Moreover, the automation of VLAN Switch Configuration addresses how our switch can be automated to change the VLAN segmentation automatically, rerouting the attacker's activities to the isolated network.

3.4 Data Gathering Techniques are explored next, focusing on how data about the network's topology is collected. Dynamic Topology Scan discusses an approach to scan dynamically and get live data, while the Preexisting Database subsection proposes an alternative which relies on a static database.

The 3.5 Automated Environment Generation section focuses on how the prototype dynamically replicates the network. It details LXC Creation, Configuration, and Launch, supported by a General Clone Script and a solution following the Database procedure. Moreover, the latter solution is further optimised by applying Parallelisation. Additionally, this section examines Nmap Scanning Limitations and Suggested Solutions, addressing some challenges encountered during scanning.

The last section, 3.6 Network Security and Communications, addresses how the prototype ensures network security. The Proxmox Firewall Defence explains how Proxmox's firewall is utilised to defend the Proxmox server. Proxmox Communication via SDN proposes using an internal Software-Defined Network (SDN) to maintain secure communications within the honeynet environment.

Two generic schematics of the prototype are presented to provide a final overview before delving into the contribution details. The first schematic, 3.2, illustrates the structure of the prototype before any malicious activity is detected. It shows the network layout, where the hosts, the attacker, and the prototype are all connected through a switch, with the devices segmented into two distinct VLANs: General and ProxIso. The second schematic, 3.3, demonstrates the system in action after the prototype is activated. Here, the Proxmox server automatically generates virtual clones (covered in sections 3.4, 3.5) of the actual hosts, which are isolated within a separate VLAN (ProxIso3) (covered in section 3.3). The attacker's interactions are then securely rerouted or tunnelled into this virtual replica, effectively containing the malicious activities and preventing any damage to the actual network infrastructure. Moreover, an internal SDN communication channel is created to allow secure communication to track the adversary (covered in section 3.6).

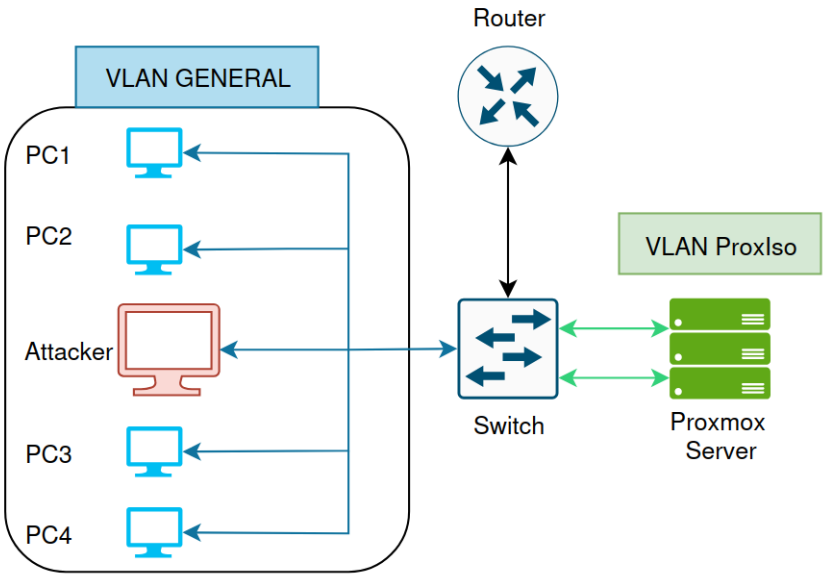

Figure 3.2: Prototype with attacker isolation not activated.

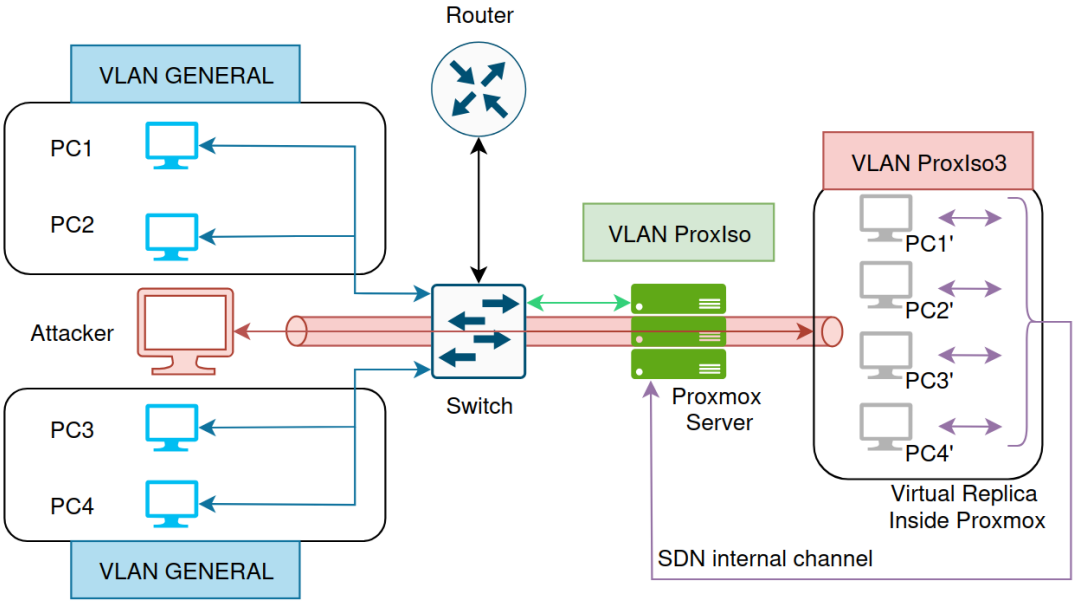

Figure 3.3: Prototype with attacker isolation activated.

3.1 Motivations and Preconditions

3.1.1 Motivation

As previously stated, the primary aim of this project is to enhance current methodologies in threat intelligence by creating and isolating the attacker in a virtualised network. The virtual network will recreate the original by spawning virtual machines and Linux containers (LXCs) with the same operating system and IP address as the original. Traditionally, compromised systems under attack are quarantined and dis-

connected from the network; however, the attacker's machine and the dummy network can be isolated to gather better data on the incident. This self-proposed project seeks to innovate response procedures by exploring a novel approach to threat intelligence.

The concept most analogous to the network this project aims to recreate is that of a honeynet, a network composed of honeypots. Honeypots have evolved and specialised to mimic the behaviour of genuine network elements. In essence, these are programs designed to respond to attackers with expected vulnerability messages, thereby deceiving the attacker into believing that the machine is compromised. However, Advanced Persistent Threats (APTs) have demonstrated the capability to detect and circumvent these traps. Recent research by Charan et al. (102) has shown that strategically using virtual machines (VMs) and configuring them correctly can effectively mislead attackers, causing them to engage in prolonged attack campaigns.

This project aims not only to trick the attackers into believing that the virtual systems are the original ones but also to automate the creation of all necessary hosts, the generation of the network, and the isolation of the attacker. Furthermore, the virtual devices must be capable of sharing data and communicating the nature of the attacks to other network elements without arousing the attacker's suspicion. To achieve this, a separate communication tunnel is established. If executed correctly, the attacker should remain unaware that their machine has been relocated to a different network.

This project is highly ambitious and requires sophisticated techniques in multiple areas. However, a successful implementation could significantly enhance threat intelligence capabilities.

3.1.2 Assumptions

Given the extensive scope of this objective, it is essential to delineate specific project boundaries and assumptions to develop a proper demonstration that can effectively showcase the project's potential:

- Due to the limitations in hardware resources allocated for the current project, the demonstration will simulate a small business or startup network where a malicious actor infiltrates (either remotely or physically) and gains control of a company machine. Simulating a smaller network reduces hardware demands and reflects environments where resources are typically constrained, ensuring the project remains feasible within its hardware limits.

- It is assumed that the attacker will be detected either by the network's Next-Generation Firewall (NGFW) of the network or by the machine's Endpoint Detection and Response (EDR) system. These advanced security products are standard in businesses prioritising cybersecurity, as they are highly effective in identifying breaches. By assuming their presence, the demonstration can focus

on how the prototype performs and interacts with the attacker rather than the process of detecting the intrusion.

- It is assumed that the devices used by the attacker, which are to be isolated, will all be connected via wired networking through Ethernet cabling. Wired networks are more secure and stable and are commonly used in office environments where VLAN segmentation is employed to control network traffic efficiently. By focusing on wired Ethernet connections, the project can leverage VLAN segmentation to isolate the attacker's traffic and ensure robust control over the network environment, a typical setup in many businesses.

- A limited number of operating systems will be utilised in this demonstration. It is assumed that a business will have machines serving different purposes, leading to the use of various operating systems. However, only Ubuntu, Debian, Fedora, and Alpine will be supported for this demonstration. These distributions are commonly used in enterprise environments and are available as LXCs in Proxmox. This ensures the virtualisation is feasible within its hardware limits.

- The project must easily integrate into the business network regardless of its topology. This ensures the prototype remains versatile and applicable across various network configurations, reflecting that each business may have a unique setup. Flexibility in integration is crucial for real-world deployment.

- The costs associated with hardware must be minimised. Design decisions must prioritise the development of a prototype that can run within the current hardware limits. Small and medium-sized enterprises operate under tight budgets, making low-cost solutions essential for widespread adoption. Minimising hardware costs ensures the prototype is accessible to organisations without substantial financial resources.

- The prototype has permission to change packets' routing by changing the switches' configuration. Network switches are typically managed by administrators, and adjusting packet routing is standard practice for managing traffic in many networks. This assumption ensures that the project can operate effectively within existing administrative controls.

- In an enterprise setting, it is presumed that a network engineer with foundational expertise is employed to tailor the prototype's configuration to align with the organisation's network topology. This is essential to ensure the correct installation of the system, minimising potential risks and security flaws.

3.1.3 Permissions

To effectively implement, develop, and evaluate the prototype, it is essential to have complete control over network devices, including routers and switches, and manage both Proxmox and attacker machines. These permissions are necessary to configure

and modify network settings, perform VLAN segmentation, and simulate attacks in a controlled environment. Without them, it would be impossible to implement the rerouting of packets and isolation required for the prototype. Consequently, the project is conducted within a home infrastructure to ensure unrestricted control over these elements. This setup allows for flexibility in experimentation and ensures that all critical security measures can be thoroughly tested and validated.

3.2 Environment Setup

3.2.1 Used Hardware

The hardware utilised for this demonstration includes several key components, including a router, a network switch, and two personal computers (PCs), each with specific roles and configurations:

- **Router**: The Askey RTF8225VW Router Smart WiFi 6 Go, provided by the Internet Service Provider (ISP) Movistar/O2, is the standard home router for most O2 clients in Spain unless the client privately purchases an alternative. Despite being the latest model, the ISP has significantly restricted its functionalities, limiting its capabilities. However, this limitation requires the prototype's design not to rely directly on the router. At a business level, the firewall can be positioned externally. This constraint imposed by the RTF8225VW Router encourages the development of a flexible prototype.

- **Switch**: The TP-Link TL-SG105E (111) is an intelligent switch that is easily accessible due to its low price (24 USD). It allows basic VLAN tagging and network segmentation, which is fundamental for the prototype as one of the aims is isolating devices and separating networks. However, this switch can only be configured through a graphical interface, imposing a significant limitation for automation. Any business that utilises a wired network requires switches.

- **PC 1**: This machine is used to spawn virtual machines (VMs) and Linux and replicate networks. Despite the demanding task, the machine utilises components that are approximately ten years old (2015): Intel Core i7-6700 (112), Corsair Dominator Platinum 16GB 3200Hz (113), Asrock Z270 Taichi (114), and no GPU.

- **PC 2**: This machine is used as a normal business device under the attacker's control. The machine is an HP Pavilion 500-220ex (2014) (115) which contains: Intel Core i5-4440 (116), and 16GB of RAM.

PC 1 - Proxmox Server

As previously mentioned, PC 1 is used as a server to deploy virtual machines (VMs) and Linux containers (LXCs) to recreate the subnetwork where the attacker machine is situated. Therefore, employing the appropriate operating systems to handle this

task is crucial to maximise the available hardware's potential. The ideal solution is a lightweight, type 1, open-source hypervisor that facilitates easy management, provides full control of virtual environments, and is cost-free.

The selected operating system that forms the foundation of this prototype is the Proxmox Virtual Environment. Proxmox has been employed in large-scale production and server infrastructures and meets all the previously outlined ideal requirements for the project.

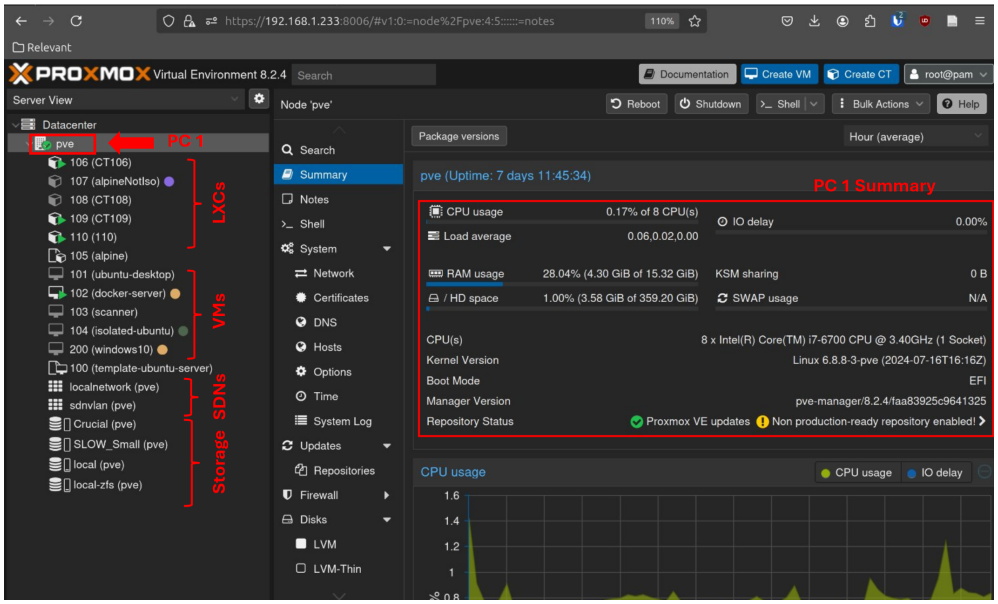

Figure 3.4: Proxmox GUI

It is important to note that multiple alternatives could have been used, as they meet some or all of the requirements mentioned above. Among these alternatives are products such as:

- **VMware vSphere**: VMware vSphere, despite its widespread use, requires expensive licenses. Additionally, vSphere focuses solely on virtual machines (VMs), whereas Proxmox natively supports KVM and Linux Containers (LXCs).
- **VirtualBox**: The commonly used VirtualBox is a Type 2 hypervisor, which results in significantly poorer performance compared to type 1 hypervisors, especially when launching VMs at scale. Therefore, this alternative would not be viable. Furthermore, VirtualBox offers limited capacity to manage virtual environments.
- **XCP-ng**: XCP-ng is an open-source virtualisation platform based on XenServer. While it could be a good alternative to Proxmox, it has a less intuitive interface and only supports VMs.
- **Microsoft Hyper-V**: The type 1 hypervisor from Microsoft primarily focuses on Windows environments and can incur additional costs.

- **oVirt**: oVirt is also an open-source virtualisation management platform that uses KVM. However, Proxmox is considered to be more user-friendly and easier to manage and set up. Moreover, the Proxmox community and documentation are larger and more accessible than those of oVirt.

While some of these products are suitable alternatives to Proxmox for this project, Proxmox excels by offering one of the most comprehensive virtualisation solutions. It provides superior flexibility for the development and maintenance of various virtualisation projects. Additionally, when multiple machines have Proxmox installed (nodes), they can be easily connected to form a cluster. In this way, Proxmox not only creates the environment but also allows for high availability natively, enabling VMs and LXCs to be moved to different nodes with no downtime for VMs and minimal downtime for LXCs.

This feature, which allows the prototype to grow efficiently and securely according to demand, makes Proxmox the optimal operating system and foundation for the project.

PC 2 - Controlled by Attacker

This machine represents a standard business device under the control of an attacker, either remotely or physically. Consequently, the operating system installed should be commonly used. The prototype is designed to be independent of the operating system selected for this device. Subsequently, this device evaluates network isolation and connection to the recreated virtual network. Given the hardware limitations, a widely-used Linux distribution is deemed appropriate, with Ubuntu 24.04 LTS being the selected operating system.

Other alternatives, such as Debian, Fedora, or Windows, could have been equally viable.

3.2.2 Network Topology Overview

The network topology for the prototype is centred around the Askey RTF8225VW Router Smart WiFi 6 Go, which is the primary gateway for all connected devices. This router is directly connected to Port 1 of a TP-Link TL-SG105E managed switch. The managed switch efficiently distributes network traffic to various devices within the network, including the prototype.

Port 2 of the TP-Link TL-SG105E is connected to PC 2, designated as the attacker device. In an enterprise environment, most machines are connected via Ethernet, ensuring a stable connection for each workstation. Ethernet is preferred in such settings due to its superior security. Wired connections are generally more secure than wireless ones, as they are less susceptible to unauthorised access and interference.

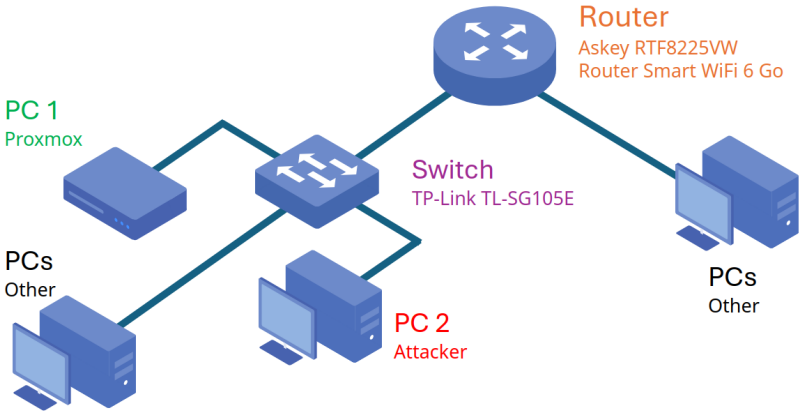

Figure 3.5: Network topology overview.

Ports 3 and 4 connect to the Proxmox server, designated as PC 1 and its inner VMs and LXCs through a trunk port. The dual-port connection is employed to ensure redundancy and create a fail-over system. By connecting the ports through an active backup bond, the Proxmox server enhances its reliability. Although using a Proxmox Link Aggregation Group (LAG) bond would increase the server's capacity, it was decided to proceed with a standard fail-over strategy for this prototype due to the anticipated low traffic load. In a real enterprise environment, the LAG option should be utilised. However, it is required to ensure that the switch can handle LAG settings.

Port 5 of the TP-Link TL-SG105E extends the network to additional PCs. The Askey RTF8225VW router also connects to other machines beyond the switch, simulating a larger network similar to an enterprise environment. The attacker device (PC 2) and the Proxmox server (PC 1) are connected to the same switch in the current implementation. However, in a larger network, this configuration would be uncommon. Typically, the Proxmox server would be positioned near the gateway, with multiple cascaded switches providing network access to each device in a large office.

3.2.3 Accessing the Network

Given that the project is conducted in a home environment, it was necessary to establish two Virtual Private Networks (VPN) servers to facilitate remote access and ensure reliability. One VPN server is configured within the Proxmox environment, while the other is on the attacker's machine. This configuration enables us to access the network as if physically present on-site, thereby allowing for remote modifications to the router, switch, and other machines.

To achieve this, we used WireGuard (117), a modern VPN protocol known for its simplicity, speed, and security. WireGuard is an open-source communication pro-

tocol that encrypts and securely transmits data between devices. Unlike traditional VPN protocols such as OpenVPN (118) and IPsec (45), WireGuard is designed to be more efficient, using fewer lines of code, which enhances its performance and reduces the potential for vulnerabilities. Through WireGuard, we ensured a robust and efficient remote access solution, facilitating seamless network infrastructure management.

Additionally, to enable the VPN connections, we had to open specific ports on the router. This step was crucial to allow the VPN traffic to pass through the router, ensuring that the remote access could be established and maintained effectively.

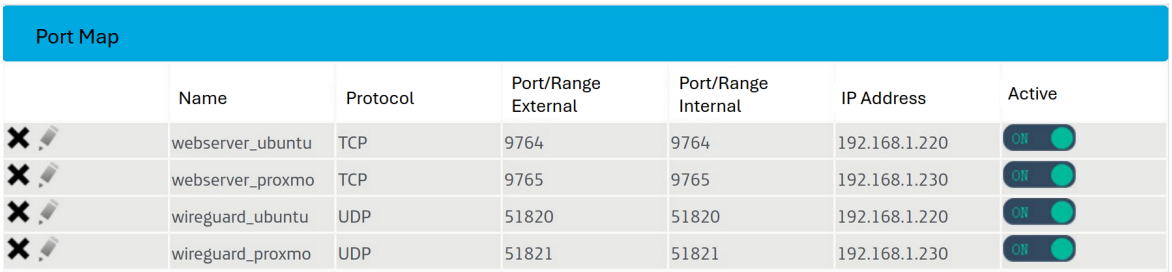

Figure 3.6: Router open ports.

Furthermore, we used DuckDNS, a dynamic DNS service hosted in AWS, to manage the challenge of not having a static IP address. DuckDNS allows the creation of a domain name that points to the current IP address of the router. Since our IP address is dynamic and can change frequently, a script container was used to query DuckDNS and update the router's IP address continuously. This setup ensures that WireGuard can query the domain with the specific port and successfully establish the VPN connection by performing the necessary handshake.

Alternatives to WireGuard

While WireGuard is an excellent choice for its efficiency and security, other alternatives can also be considered:

- OpenVPN (118): OpenVPN is a widely used open-source VPN protocol known for its robust security. It supports various encryption algorithms and is highly configurable. However, its performance is generally slower than WireGuard due to its complex code base and higher overhead. Additionally, OpenVPN setup and maintenance can be challenging for users without technical expertise.

- SoftEther VPN (119): SoftEther VPN is a multi-protocol VPN software supporting SSL-VPN, L2TP, and IPsec, among others. It is noted for its high performance and user-friendliness, offering a versatile alternative to WireGuard. Despite this, SoftEther VPN VPN is not as lightweight and fast as WireGuard.

Its extensive features may also introduce complexity and potential security vulnerabilities.

- ZeroTier (120): ZeroTier enables the creation of universally operable virtual Ethernet networks, providing a straightforward and efficient method to establish secure, private networks. However, it is not explicitly designed as a VPN protocol and may not match WireGuard's performance and security. Additionally, ZeroTier 's proprietary protocol can be less transparent and more challenging to audit compared to WireGuard's open-source design.

Alternatives to DuckDNS

Other dynamic DNS services can be considered:

- No-IP (121): No-IP offers free and paid dynamic DNS services, allowing users to create up to three free dynamic DNS hosts. It is a reliable alternative with a user-friendly interface. However, DuckDNS provides a more straightforward setup process and does not require monthly confirmation to maintain free accounts, enhancing user convenience.
- Dynu (122): Dynu provides both free and paid dynamic DNS services. It supports top-level domains and offers a client that automatically updates the IP address, ensuring continuous synchronisation. In contrast, DuckDNS offers seamless integration with various platforms and applications, making it a more versatile choice for diverse user needs.
- FreeDNS (123): FreeDNS offers a wide range of domain options and allows users to create multiple subdomains. It is a robust and flexible alternative to DuckDNS. Nevertheless, DuckDNS's straightforward and ad-free service model ensures a more streamlined and distraction-free user experience.

3.3 Virtual Local Area Networks (VLANs)

The use of VLANs allows the prototype to reuse existing IP addresses. This is done intentionally to deceive attackers into thinking they communicate with real devices in the subnetwork, which would be detected during a normal network scan. In reality, they interact with VMs and LXCs within the Proxmox server. The Proxmox server attempts to recreate the original subnetwork to which the attacker's device is connected.

3.3.1 VLANs Prototype Overview

The TP-Link TL-SG105E switch is pre-configured with a default VLAN tag of 1, facilitating initial setup and management. By default, all ports on the switch are assigned to VLAN 1, enabling seamless communication among connected devices without

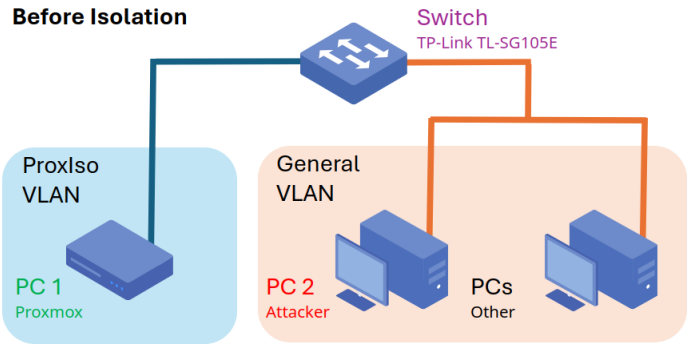

Figure 3.7: Network before isolation.

requiring additional configuration. This default setting ensures that the switch is operational immediately upon deployment and can be easily integrated into most network environments.

Under VLAN security guidelines (see Chapter 2.3.2), using the native/default VLAN tagged 1 is discouraged to minimise security risks. Consequently, a new Internet VLAN with tag 2 has been created to provide internet access to all the devices added to it, and a General VLAN with tag 3 has been created as a network segment to contain all non-isolated devices within the network area.

Additionally, as a design decision, VLANs 4 and 5 have been chosen to represent other department's networks that could be present in a realistic environment. After detecting suspicious behaviour in a device, it is isolated by changing the switch VLAN configuration, forcing it into a new VLAN, and redirecting its packets to VMs or LXCs inside the Proxmox server. For those purposes, VLANs 13 to 15 are used. It is in this way that the prototype is capable of dynamically and simultaneously isolating multiple attackers. Adding on, we made the design decision to isolate attackers operating in the same subnetwork together. This is because the same attacker might highly likely control them. In that case, the two (or more) machines should be able to communicate. These VLANs used for these purposes are named following the pattern ProxIso<Number>, where the <Number> identifies the original VLAN is recreating, and the tag would be original plus a shift of 10. Therefore, VLAN 3 becomes VLAN ProxIso3 and will have tag 13. Moreover, the Proxmox server has been isolated as a design decision in VLAN 11, named ProxIso, to avoid being directly exposed to potential attackers. This approach ensures proper network segmentation and enhances overall security.

In the ideal situation of having a router that allows tagging, the correct approach would be to use port 1 as tagged to provide internet separately between General and ProxIso, isolating the devices correctly. An even further solution would be to give each device its own Internet VLAN connecting to the router with the devices. However, due to the hardware limitations, all the devices would be connected via the Internet VLAN, and another solution via firewall must be found for this prototype.

To make this setup work with Proxmox, the server has been configured to use tagged packets by adding network and VLANs (11 and 13 to 15) interfaces. This can be seen in figure 3.8. In this way, each VM and LXC can directly specify the VLAN to which it should be added in its network interface configuration. This results in a clean solution to keep its networking separate from the rest through the switch's trunk port.

Name	Type ↓	Active	Autostart	VLAN ...	Ports/Slaves ↓	Bon...	CIDR	Gateway	Comment
vmbr0.14	Linux VLAN	Yes	Yes	No					ProxIso4
vmbr0.2	Linux VLAN	Yes	Yes	No			192.168.1.233/24	192.168.1.1	Internet
vmbr0.15	Linux VLAN	Yes	Yes	No					ProxIso5
vmbr0.11	Linux VLAN	Yes	Yes	No					ProxIso
vmbr0.13	Linux VLAN	Yes	Yes	No					ProxIso3 - Demo Active Prototype
enp6s0	Network Device	Yes	No	No					
enp0s31f6	Network Device	Yes	No	No					
vmbr0	Linux Bridge	Yes	Yes	Yes	enp0s31f6				
vmbr1	Linux Bridge	Yes	Yes	Yes	enp6s0				

Figure 3.8: Proxmox network setup to support VLANs 2 (Internet), 11 (ProxIso), 13 (ProxIso3 - Used for this demo), 14 (ProxIso4) & 15 (ProxIso5).

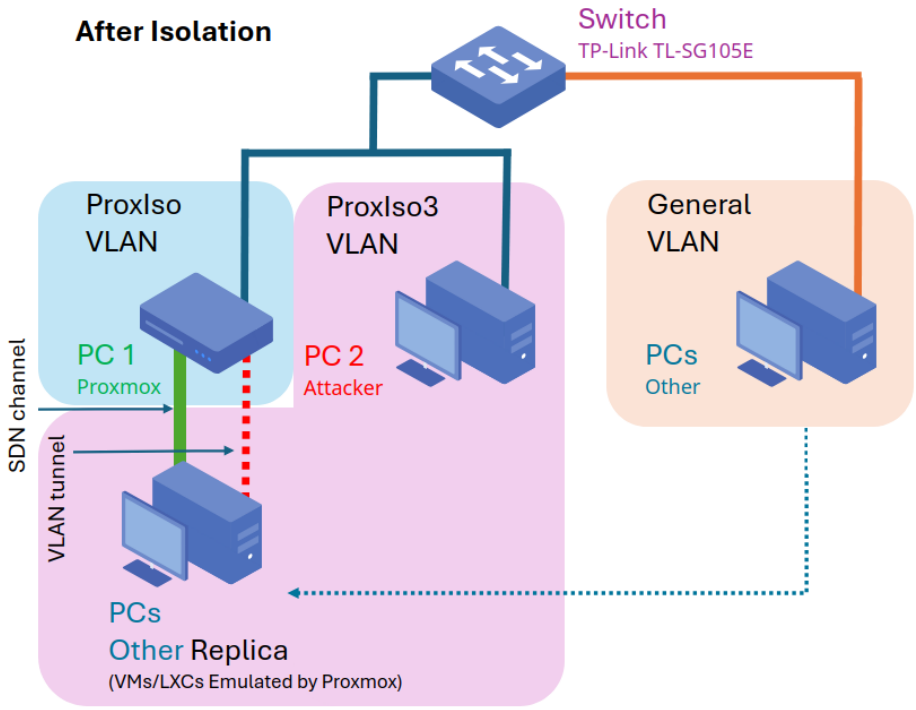

Figure 3.9: Network after isolation.

Given the prototype's small scale and the limited capabilities of the home switch, it is logical to use VLANs 12 to 15 for isolation, as a switch typically supports up to 4094 VLANs. However, these VLANs may be unavailable or significantly limited in a large enterprise environment. The on-demand creation of isolation VLANs could potentially conflict with existing VLANs, leading to networking issues and the unintended connection of devices that should remain segregated. Implementing Q-in-Q

(refer to Chapter 2.3.3) is advisable in such scenarios. Double tagging (VLAN tag with two layers) allows each 4094 VLAN to contain an additional 4094 VLANs. Q-in-Q presents a practical solution, as a specific VLAN layer can manage all isolation VLANs, thereby enabling 4094 isolation VLANs. This approach offers a straightforward and elegant solution for managing switches at multiple levels and configuring their trunk ports.

3.3.2 VLAN Creation

The article by TP-Link, which provides practical examples of how to create and manage VLANs using its devices, significantly aided our understanding of the network segmentation process through VLANs. It is crucial to comprehend when ports must be tagged and untagged to correctly configure the VLANs 2.3.1.

The initial creation and configuration of the VLANs were performed manually. However, after ensuring the correct setup of the prototype, the entire process was automated.

The TP-Link TL-SG105E switch only offers a web GUI where the VLAN configuration can be set.

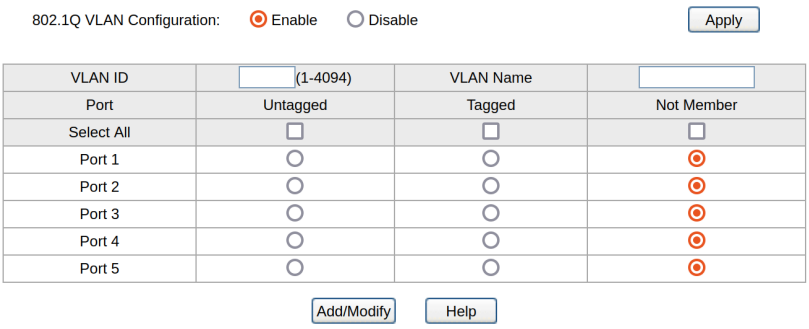

Figure 3.10: VLAN GUI

As shown in figure 3.10, an ID must be chosen to create new VLANs, and each port that forms part of the network segment must be correctly set.

3.3.3 VLANs Configuration

For the initial VLAN implementation, the following VLANs were set:

- Default - Tag ID 1: The default VLAN is not used for security reasons.
- Internet - Tag ID 2: Used to allow internet connection. All the devices are connected to this VLAN, and the ports are all untagged except for the ones for Proxmox, which must be tagged. Ideally, this VLAN would be removed if the correct hardware was avaiblable. Instead, port 1 would be added to General and ProxIso as tagged, allowing the correct segmentation, as explained before.

- General - Tag ID 3: In this case, all the ports are untagged except the Proxmox server's ports (3 & 4) as they must remain isolated, and port 1 of the router (in the ideal case of providing the internet).
- Department A and B - Tag ID 4 & 5: No specific port is assigned.
- ProxIso - Tag ID 11: This is the Proxmox server's isolated VLAN. It only includes ports 3 and 4 as tagged. Port 1 would be tagged in the ideal case to provide the Proxmox server internet directly.
- ProxIso3/4/5- Tag ID 13/15/16: These VLANs dynamically recreate VLANs 3/4/5 and initially have ports 1, 3 and 4 set as tagged.

VLAN ID	VLAN Name	Member Ports	Tagged Ports	Untagged Ports	Delete
1	Default				☐
2	Internet	1-5	3-4	1-2,5	☐
3	General	1-2,5	1	2,5	☐
11	ProxIso	1,3-4	1,3-4		☐

Figure 3.11: VLAN initial setup with attacker not isolated.

VLAN ID	VLAN Name	Member Ports	Tagged Ports	Untagged Ports	Delete
1	Default				☐
2	Internet	1-5	3-4	1-2,5	☐
3	General	1,5	1	5	☐
11	ProxIso	1,3-4	1,3-4		☐
13	ProxIso3	1-4	1,3-4	2	☐

Figure 3.12: VLAN setup with ProxIso3 and attacker isolated.

It is essential to highlight that configuring the correct Port VLAN Identifier (PVID) for each port is critical to facilitate communication when the connected devices do not inherently send VLAN-tagged packets. The PVID is the default tag assigned to all packets arriving at the switch through a specific port. Subsequently, the switch recognises the assigned VLAN tag and can appropriately forward the packet to devices within the same VLAN.

The PVIDs are illustrated in Figure 3.13 for the previous VLAN configurations. Notably, ports not used for the prototype are configured with PVID 2 to maintain connectivity without disrupting the experimental environment. Furthermore, in the event of a new attacker at Port 5, the port would be immediately assigned to its corresponding ProxIso<Current tag>.

When dealing with complex networks with multiple switch levels or when your Proxmox server is in a different network area, it's crucial to configure all VLAN ports that send packets to other network devices (like switches or routers) to use the

Select	Port	PVID
☐		
☐	Port 1	2
☐	Port 2	3
☐	Port 3	11
☐	Port 4	11
☐	Port 5	2

Figure 3.13: PVID Configuration

tagged option mode. This setup turns these ports into trunk ports, ensuring that VLAN tags are preserved in the network packets as they travel to other networking components.

Doing this allows you to assign the port connected to an attacker's device to a specific VLAN meant for isolation. This VLAN can then be routed to the Proxmox server, no matter where it is located in the network.

3.3.4 Switch Automation - VLANs

Given the limited capabilities of the TP-Link TL-SG105E, configuration modifications are restricted to the web interface. Consequently, we initially used tools such as Netcat and Burp Suite to inspect packets and reverse engineer the connection. After developing scripts to interact with the switch without success, we discovered a publicly available project (124) specifically designed to alter the TL-SG105E switch configuration via scripting. However, despite using this project, the switch's configuration remained unchanged after executing the scripts. Through debugging, we discerned that the project was three years old and that newer firmware versions might have patched the method the scripts used to send commands to the device without the graphical user interface (GUI).

After downgrading the firmware by one release, the published project managed to work, make a simple change in the VLAN configuration and reflect it in the GUI after refreshing the GUI website. The techniques used in the project are similar to the initially reverse-engineered scripts we created. Consequently, these methods might be compatible with the downgraded firmware, although this has not been verified.

The project employs a straightforward YAML file to define the VLAN configuration for the switch. Within this YAML file, users must specify port by port, the tag and the PVID to be assigned during runtime. A notable limitation identified in the original project is the inability to simultaneously assign multiple VLANs to a single port. This poses a significant challenge, particularly for port 1, which is connected to the router and requires concurrently assignment to all VLANs. Similarly, when using the dynamic ProxIso<tag> VLAN, the ports on the Proxmox server must also

be assigned multiple VLAN tags.

We implemented the necessary modifications within the public project to make it capable of setting the configurations outlined in 3.3.3.

The following short YAML example 3.14 illustrates the essence of these types of files. A YAML file must be created for each switch VLAN configuration. By default, three initial YAMLs are created:

- Remove all VLAN settings and reset the switch (see appendix A.1).
- Create the Internet, General, and ProxIso VLANs. In this configuration, the Proxmox server is isolated (see appendix A.2).
- Create a default defence VLAN configuration where the attacker device is connected to ProxIso3 (see appendix A.3).

These YAML files must include the IP address of the switch, its credentials, and the port configuration.

Figure 3.14: YAML Example

```yaml
default-VLAN: Internet
switches:
  Switch-Main:
    ip: 192.168.1.43
    type: "TLSG105E"
    username: admin
    password: *****
    description: "Switch-Main"
    default-vlan: Internet
    ports:
      port1:
        description: "Router"
        vlan: Internet
        type: untagged
        untagged-vlan-id: 2
        pvid: 2
      port3:
        description: "Proxmox Tower"
        type: trunk
        vlan: ProxIso

vlans:
  Default:
    vlan-id: 1
  Internet:
    vlan-id: 2
```

```
27   ProxIso:
28     vlan-id: 11
29
```

3.4 Data Gathering Techniques

To create a digital replica within the Proxmox server of the machines that form the subnetwork to which the attacker is connected, it is essential to have detailed knowledge of the number of devices within that subnetwork and their respective operating systems. This information is crucial to ensure that the replicated environment generates identical responses, particularly in scenarios where the attacker has previously interacted with these machines, thereby avoiding any suspicion that the environment has been altered.

Two approaches can be employed in this context:

- Using an automated script dynamically scans the network to collect the necessary data.
- Using a preexisting database created by the network engineers containing all the required information.

Each approach presents different advantages and disadvantages. Depending on the prototype's specific implementation context, the engineering team must carefully evaluate these factors to determine the most suitable method for their requirements.

3.4.1 Dynamic Topology Scan

This method is one of the most straightforward and convenient, as it eliminates the need to monitor which devices are connected and through which subnetwork. We used the Nmap tool (125) as the primary means to collect information from the subnetwork. Additionally, Nmap facilitates the scanning of devices to determine their operating systems. Following a TCP scan, Nmap compares the results for each device against its extensive database of over 2600 operating system fingerprints. If a match is found, Nmap displays the device's operating system.

Unfortunately, the fingerprint database is both small and limited. After conducting multiple experiments with various configurations, it has been demonstrated that Nmap possesses a limited capacity to identify the operating systems of devices within a home environment characterised by diverse devices running different operating systems. Specifically, in this context, Nmap failed to identify the operating systems of 35% of the devices within a network of 20 devices. This could be due to the absence of many fingerprinted IoT devices. Although these figures are not substantial, they suggest that this method could prove unreliable for generating a digital replica

when scaled up.

Other alternatives to Nmap can be used. However, after testing some of them, they have shown worse accuracy while requiring longer scan times. Examples of these are:

- p0f (126): a passive OS fingerprinting tool which analyses network traffic to determine the operating system of devices without actively scanning. The main issue is that it takes a long time and relies on devices which generate traffic.
- Xprobe2 (127): through fuzzy logic for operating system detection, this tool may perform better in complex network environments. Nevertheless, it demands extensive configuration and requires a user with significant expertise to operate it effectively.
- OpenVAS (128): this is a suite of tools designed for vulnerability scanning, including an operating system detector. However, it can be resource-intensive and requires a more extensive setup compared to Nmap.

Recent research has highlighted innovative projects integrating fingerprinting with machine learning, demonstrating impressive results. For instance, Song et al. (129) achieved a 94% accuracy rate in identifying operating systems. If resources are not a limitation, as would be the case in the case of a big enterprise, this type of solution could potentially be the most effective option available.

It is imperative to highlight that a tool must demonstrate high accuracy and perform scans at high speed. Detecting an attacker within the network indicates that suspicious activities have already occurred. Consequently, rapid scanning and deployment are essential, as they limit the information available to the attacker and significantly enhance security measures.

Another crucial factor to consider is the scanner's placement, which significantly affects the feasibility of this method. If the scan is conducted directly on the Proxmox server and this server is isolated within a different VLAN (ProxIso3), as is currently designed, the scanner would be unable to detect any devices. It would require temporally exposing the Proxmox server to the attacker. One potential solution involves exposing a Linux container (LXC) with appropriate VLAN tagging. In this way, the scanner would only be capable of detecting devices within the specific VLAN to which it is assigned. This approach was initially implemented in the prototype. Nevertheless, an alternative method was adopted due to the scanner's previously mentioned limitations, which implies using a preexisting database of devices within each subnetwork.

Alternatively, the scan could be conducted externally to the Proxmox server, such as on a device higher in the network topology or directly at the Next Generation Firewall. This method would require transmitting the necessary data to the Proxmox server to deploy the environment later correctly.

In conclusion, it is essential to note that Nmap, along with other similar tools, has a significant limitation that adversely impacts the creation and deployment of the environment. Although Nmap can detect the operating system and potentially identify the kernel version in use, it cannot distinguish between different Linux distributions. This limitation is discussed in greater detail in 3.5.2, where a potential solution is proposed.

Scan Script design and Implementation

The prototype's proposed script has been designed to perform the fastest possible scan. It assumes that Nmap has been installed, and its commands have been specifically tailored to minimise both delay and number of retries. The script can be divided into three stages, each with a different focus:

1. The simplest and quickest Nmap command is used to identify the devices on the subnetwork.
2. A parallelised stage where each detected device is scanned as rapidly as possible to determine its operating system and version.
3. The results are parsed, and the IP addresses, possible operating systems, and their respective probabilities are exported to a JSON file for the automated environment launch.

The key commands used in the script are as follows:

Figure 3.15: First stage general Nmap scan

```
1 sudo nmap -sn -n -T5 --host-timeout 15m --scan-delay 0 {subnetwork}
```

- **"-sn"**: This command indicates Nmap to perform a ping scan to discover devices without scanning ports.
- **"-n"**: This option is used to skip DNS resolution, which can speed up the scan.
- **"-T5"**: This sets the performance template to "insane," configuring Nmap for its fastest possible operation.
- **"–host-timeout"**: This option sets a maximum time limit for scanning each host, in this case, 15 milliseconds.
- **"–scan-delay"**: This flag controls the pacing of the scan. In this instance, it eliminates any delay, ensuring the scan proceeds quickly.

Figure 3.16: Second stage OS scan

```
sudo nmap -F -O -T5 -n --max-retries 2 --host-timeout 15m \\
    --scan-delay 0 {ip} --version-light --osscan-limit    \\
    --max-os-tries 1 -oN ./Scans/{ip}
```

The second Nmap command uses additional flags to detect and optimise the OS fingerprinting:

- **"-F"**: Performs a fast scan of the 100 most common ports, which is necessary for OS fingerprinting.
- **"-O"**: Enables operating system fingerprinting.
- **"–max-retries"**: Prevents the scan from becoming stuck on unreachable hosts.
- **"–version-light"**: Sets a lighter version scan, which is faster but less comprehensive.
- **"–osscan-limit"**: Restricts OS detection to only the most promising targets to accelerate the scan.
- **"–max-os-tries"**: Prevents the scan from getting stuck during OS fingerprinting.

The complete script is in the Appendix B.1.

3.4.2 Preexisting Database Approach

An alternative method to provide the Proxmox server with the necessary data to recreate the environment is maintaining a database that tracks which devices are present in each subnetwork. While this approach requires an initial investment of human resources and time, particularly depending on the organisation's size, it would only need periodic maintenance afterwards. The primary advantage of this method is that, if the database is accurate, the replicated environment would perfectly match the original, with no discrepancies. Additionally, this approach eliminates the challenge of identifying Linux distributions, a problem highlighted in the automatic scanning method.

A simple SQLite database was created for this prototype to store the IP addresses, VLAN information, devices' operating systems, and Linux distributions in the home network. To gather the necessary data, we accessed the local network map of the Askey RTF8225VW router, where all connected devices are listed. Each device was manually identified, categorised, and entered into the database.

It is important to note that many IoT and other devices use Linux distributions that are not identifiable. In such cases, manually checking and searching through the device's MAC address can assist in identifying it. When identification remains unclear, or the prototype does not support the distribution, it defaults to launching Alpine Linux containers (LXCs) to optimise hardware resources while effectively

recreating the device.

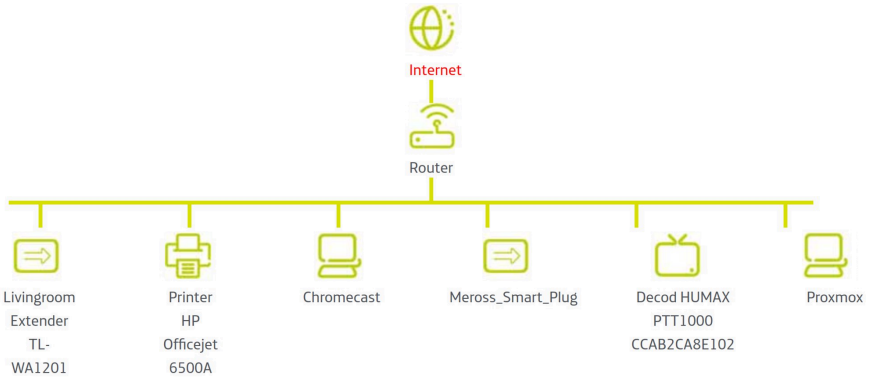

Figure 3.17: Part of Local Network Map

After creating the table of devices (Table 3.1), 21 devices were added to the database. The complete table is provided in Appendix B.2.

IP Address	Device Name	VLAN	OS	Distribution
192.168.1.1	Router Askey RTF8225VW	1	Linux	
192.168.1.43	Switch TP-Link TL-SG105E	1	Linux	
192.168.1.236	Proxmox Scanner LXC	2	Linux	Ubuntu
192.168.1.233	PC 1 - Proxmox Server	3	Linux	
192.168.1.220	PC 2 - Attacker Device	2	Linux	Ubuntu
192.168.1.230	Docker Server	1	Linux	Ubuntu
192.168.1.221	Debian Server	2	Linux	Debian
192.168.1.104	Person A Desktop 2	2	Linux	Ubuntu
192.168.1.46	Person D Desktop	2	Windows	10
192.168.0.51	Printer HP	2	Linux	

Table 3.1: Key Devices for the Prototype Demo

The primary advantage of this method is that, since the database is stored directly on the Proxmox server, the automated scripts used to launch the environment have direct access to the necessary data. In contrast, even in the best-case scenario where the scanner runs in an LXC within Proxmox, a communication channel would still need to be established to transmit the OS fingerprinting results to the server.

3.5 Automated Environment Generation

In this section, we use the data gathered in Section 3.4, whether obtained through automatic scanning or manual entry, to recreate the environment in which the attacker's device (PC 2) is located.

Fortunately, Proxmox provides a command-line tool called pct for managing Linux containers (LXCs) (130), as well as a REST API that allows for the creation and management of the Proxmox environment and virtual machines (131) through command-line commands. These commands can be executed via bash scripts, enabling the automation of these processes.

Since many devices use the same operating system and distribution, this characteristic can be leveraged to optimise resource usage. A single LXC can be configured with multiple network interfaces, allowing multiple IP addresses to be assigned simultaneously. This capability enables the deployment of just one LXC per operating system and distribution, effectively grouping all the corresponding IPs. As a result, this approach significantly reduces the resources needed to recreate the environment.

However, for VMs, these network changes must be manually configured within the VM itself, typically through files such as "/etc/network/interfaces" or similar configurations. This manual requirement, combined with the higher resource demands of VMs, made using Windows a non-viable option for this initial prototype. Nevertheless, with more powerful hardware and appropriate automation using the netsh tool (132), deploying Windows in this manner could become feasible.

3.5.1 LXC Creation, Configuration and Launch

As mentioned, the prototype can automatically spawn ready-to-use LXCs with Ubuntu, Debian, Fedora, or Alpine distributions in seconds. To achieve this, we manually created LXC templates for each Linux distribution and developed several scripts to automate and streamline the process. Since the prototype relies on the preexisting database approach, a script is required to query the database and prepare the data, including IP addresses and distributions, to be passed to the LXC creator.

The scripts developed are as follows:

1. A generic script to clone, configure and launch LXCs.
2. A script to get the correct data from the database and run the script (1), passing the correct arguments needed.
3. Further scripts to gather data for benchmarking

General Clone Script

The "launch_lxc.sh" bash script takes information about the template tag to clone, the new LXC ID, the current gateway, the distribution name to be set as the hostname, the group of IP addresses that use the distribution and the IP address assigned to the LXC for the internal SDN communication.

Figure 3.18: Key commands

```
# 1
pct clone $TEMPLATE_ID $CT_ID
# 2
pct set $CT_ID --hostname $DISTRO --cores 1 --cpulimit 2 --memory 512
 ↪  --swap 512
# 3
pct set $CT_ID -net$index
 ↪  name=eth$index,bridge=vmbr0,firewall=1,gw=$GW,ip=$ip/24,
 ↪  type=veth,tag=13
```

An alternative approach is to use the "pct create" command. However, using the clone method allows Proxmox to create a linked clone. This lightweight copy of the original template shares its disk storage with the parent template rather than duplicating the entire template. Only the differences between the template and the cloned LXC are stored, significantly reducing the time and storage space required for LXC creation. As a result, this linked clone method offers a substantial advantage in speeding up the deployment process.

The "pct set" commands are then used to configure the LXC, such as setting hardware limitations and adding additional network interfaces with static IP addresses, as shown in the third command.

The complete script can be found in the Appendix C.1.

Database Data Extraction and Parallelisation

To accurately recreate the target environment, data from the relevant VLAN and network sections must be extracted, organised, and prepared from the database. To accomplish this, the generic script is executed by a main script "db_script.py". The advantage of this nested LXC creation script is that it allows for further optimisation of the generic script.

The "db_script.py" script has undergone several iterations to minimise execution time, which is crucial since the environment needs to be available as quickly as possible. The generic script can be divided into two parallel phases: the cloning and configuration phase and the launch phase for the new LXCs. Due to Proxmox's architecture, the creation and startup of each LXC are treated as independent jobs, allowing multiple processes to run concurrently. The following code snippet demonstrates how to clone the LXCs in parallel in "db_script.py", while the complete version can be found in Appendix C.2:

Figure 3.19: Parallel LXC cloning

```
def create_container(distro, ct_id, distro_template, gateway, sdn_ip,
    sdn_end, ips):
        ips = " ".join(ips)
        cmd = f"./launch_lxcs.sh {distro} {ct_id}
        {distro_template[distro]} {gateway} {sdn_ip}{sdn_end}
        {ips}"
        print(cmd)
        subprocess.run(cmd.split(" "), stdout=subprocess.DEVNULL)
        ct_list.append(ct_id)

    # Create the LXC containers in parallel
    with concurrent.futures.ThreadPoolExecutor() as executor:
        # Submit the create_container function to the executor for each
        distro
        futures = [executor.submit(create_container, distro, ct_id + i,
        distro_template, gateway, sdn_ip, sdn_end + i,
        linux_dict[distro]) for i, distro in enumerate(linux_dict)]

        # Wait for all the futures to complete
        concurrent.futures.wait(futures)
```

3.5.2 Nmap Scanning Limitations and Suggested Solutions

As mentioned, the scanning approach has a significant limitation: Nmap does not provide detailed information about the specific Linux distribution. While an attacker might initially face the same limitation when using similar tools, a skilled attacker could deduce the distribution through machine interactions. This presents a critical challenge that needs to be addressed to ensure the security and accuracy of the environment.

Implementing a hybrid approach is the most effective way to address this issue. Proxmox could maintain a database of MAC addresses and their corresponding operating systems or Linux distributions. This allows the Nmap scanner to detect the devices currently active in the attacker's VLAN and launch a more accurate environment, replicating only the devices present at the time of isolation. This methodology could also be integrated into the existing database approach explained in section 3.4.2, where a simple ping scan could produce similar benefits by ensuring that only active devices are included in the recreated environment.

3.6 Network Security and Communications

3.6.1 Proxmox Firewall Defence

Proxmox includes a firewall system (72), which can protect both the server and the docker services (VPN, DNS, web server) from malicious actors. Despite being isolated, new rules are added automatically to the Proxmox server's firewall for security

purposes, ensuring that any packet from an attacker's IP is dropped when its destination is the Proxmox server directly.

This security measure ensures the attacker can never communicate with the prototype's core.

Figure 3.20: Firewall rule, preventing direct Attacker and Proxmox server connection

```
[ALIASES]

attacker 192.168.1.220
proxmox_server 192.168.1.233

[RULES]

|IN DROP -source dc/attacker -dest dc/proxmox_server -log emerg
```

In addition, the replicated LXCs use two communication channels. The one connected to the isolated VLAN is exposed to the attacker and should never be used to communicate with the server. Therefore, further rules to drop incoming packets from VLANs 13 to 15 could be optionally added.

3.6.2 Proxmox Communication via SDN

Proxmox supports the use of Software-Defined Networks (SDN) to manage cluster networks and facilitate internal communication between VMs, LXCs, and other Proxmox nodes. This powerful functionality can create a private communication channel within the recreated environment. Such a channel would be invisible to the attacker, preventing them from detecting or sniffing packets, thereby enhancing the environment's security. Moreover, the further intention of using this technology in the prototype is its capability of decoupling the infrastructure, control and application layers. These characteristics make it perfect for the intended purpose of the prototype, allowing cutting-edge threat intelligence tools to deeply inspect the communications happening and attack techniques used through this channel.

The first step in creating an SDN in Proxmox is establishing a network zone. Proxmox offers several types of zones, but the VLAN zone is the most suitable choice for this prototype. If a larger cluster is being used or access to Proxmox resources is needed from outside the current network, Q-in-Q and VXLAN zones are available as alternatives, depending on the specific requirements of the SDN.

Afterwards, a VNet must be created to assign the VLAN tag as the internal SDN VLAN identifier.

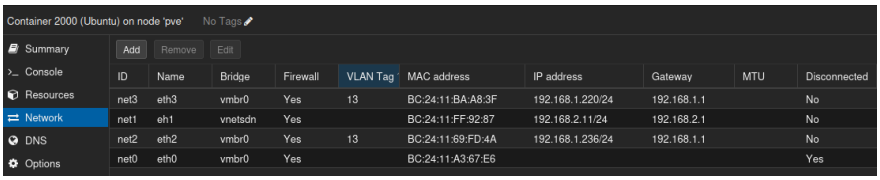

Figure 3.21: Network interfaces of generated Ubuntu LXC

The final step involves adding a network interface to each LXC in the environment replica, using the VNet as its bridge. This operation is performed using the pct set command, as discussed in Section 3.5, and is exemplified in line 20 of the launch_lxc.sh script (see Appendix C.1). Figure 3.21 illustrates the resulting configuration of the launch_lxc.sh script, where all network interfaces use vmbr0 except for the one designated for the SDN, which is set to vnetsdn.

Chapter 4

Evaluation

This section assesses and discusses the prototype's security, viability, and effectiveness. We performed benchmarks, which gave an idea of the performance and the time needed to launch an environment replica of a real network.

4.1 Security Evaluation

As this project aims to propose an initial system upon which more advanced threat intelligence strategies can be developed, ensuring that the system meets all security requirements and does not accidentally introduce new vulnerabilities to the network it is intended to protect is crucial. The well-known MITRE ATT&CK® (133) framework is used to achieve a comprehensive evaluation. The MITRE ATT&CK® (Adversarial Tactics, Techniques, and Common Knowledge) framework is a comprehensive database of attack procedures and cyber adversary behaviours organised according to the different phases of the attack lifecycle. These are represented in a matrix format (see figure 4.1), highlighting common attack strategies and their target platforms. Additionally, the framework outlines offensive tactics and offers guidance on defensive measures necessary to mitigate these attacks.

The MITRE ATT&CK® framework is extensive, so to effectively apply it to this project, it is essential to focus on the specific phases within the matrix most relevant to the project's objectives. The framework covers 14 phases or topics: Reconnaissance, Resource Development, Initial Access, Execution, Persistence, Privilege Escalation, Defence Evasion, Credential Access, Discovery, Lateral Movement, Collection, Command and Control, Exfiltration, and Impact.

Given the nature of this project, it is essential to identify the most pertinent tactics and techniques. In this context, the Reconnaissance, Discovery, Defence Evasion, Lateral Movement, and Exfiltration phases are especially critical. These phases directly relate to how an adversary might attempt to breach, maintain access, and move within the network. By concentrating on these areas, the evaluation can more effectively address potential vulnerabilities and ensure the system is robust against the most relevant threats.

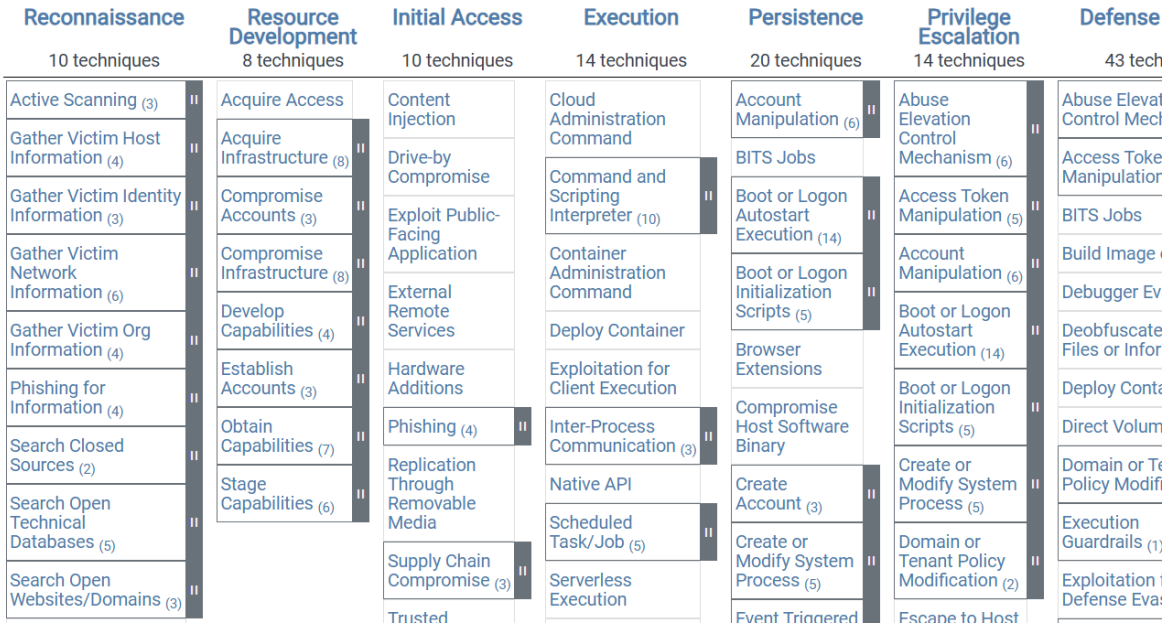

Figure 4.1: Part of MITR ATT&CK® matrix.

4.1.1 Reconnaissance and Discovery

In these two phases, the attacker tries to gather data to be used later in future operations. Reconnaissance is primarily associated with pre-attack activities to gain initial access to an organisation. At the same time, discovery takes place after the attacker has infiltrated the network to explore the internal environment. This enables them to identify additional systems that could be compromised, refine their strategy for further attacks and their scope, and prioritise specific targets.

Reconnaissance

Attackers can use various mechanisms during the reconnaissance process, commonly gathering information about the victim's network, such as topology, IP addresses, and security appliances.

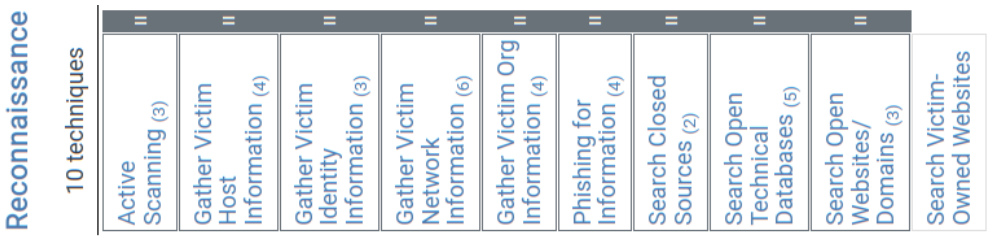

Figure 4.2: Reconnaissance MITR ATT&CK® column.

According to MITRE, these techniques are difficult to mitigate through standard controls because they often involve behaviours like Active Scanning or Phishing,

which can occur outside the organisation's defences and controls. Furthermore, if the isolation and environment replication strategy becomes widely known, attackers may seek information through Public or Private Sources to determine if this technology has been implemented. This awareness could undermine the method's effectiveness, potentially allowing attackers to evade the isolated environment and weakening the overall defence strategy.

Once the attacker has infiltrated, evading and countering the discovery process is crucial for successfully gathering threat intelligence in this prototype. The system must remain unnoticed, ensuring the environment is generated without alerting the adversary and minimising the chances of the attacker realising they have been isolated in a replica. This stealth approach is essential for effectively collecting valuable data on the adversary's tactics and behaviour without supposing a significant risk to the business.

Discovery

All network discovery strategies can be seen in figure 4.3. The prototype is designed to avoid any further service or process installation to prevent being discovered. In this way, Device Driver Discovery, Files and Directory Discovery or Groups Policy can not reveal any information about the defence strategy. Moreover, those related to the Cloud (Cloud Infrastructure Discovery, Service Discovery, or Storage Discovery) or Local System information (System Owner/User Discovery, System Time Discovery or Peripheral Device Discovery) have no effect. The relevant techniques which could potentially affect negatively and against which the prototype must be designed to avoid being discovered are those related to Remote Discovery of Systems such as Remover System Discovery, Network Sniffing, System Network Connections Discovery, System Network Configuration Discovery, Security Software Discovery and Process Discovery; which are discussed next.

Figure 4.3: Discovery MITR ATT&CK® column.

- **Remote System Discovery**: The adversaries attempt to identify other systems on a network by gathering information such as IP addresses, host names, or other identifiers. This information is crucial for understanding the environment and facilitating Lateral Movement, which is the process of moving from one compromised system to another within a network.

If an attacker successfully performs a remote system discovery while outside the isolated environment, they will gather data on devices within the same VLAN. The prototype anticipates that the attacker will employ such techniques and respond by creating a replica of the environment when the isolation process is triggered.

However, if an attacker successfully performs remote system discovery within the isolated environment, they may realise that they are not in the genuine network, particularly if they spot discrepancies. It is important to note that the current prototype replicates environments by launching hosts with the same IP addresses and operating systems as the real ones. Realistically, this version serves as a proof of concept, and more advanced procedures are necessary to create a more convincing environment.

Firstly, the machines in the generated environment do not replicate the open ports and services of the actual devices, which could immediately raise suspicion that it is a fake environment. Secondly, no communication occurs between the dummy devices or the gateway, making it a “dead” network. If the attacker attempts techniques such as network sniffing, they would quickly notice the unusual behaviour of the network. These limitations have yet to be resolved, and further improvements are discussed in section 5.2. Without addressing these issues, the attacker may alter their behaviour, compromising the intelligence-gathering objectives of the prototype.

MITRE suggests several methods for detecting these discovery procedures. One approach is to monitor the execution of commands that might allow attackers to identify other devices on the network. This includes tracking the establishment of new connections via tools like ping and other network scanners. The security systems could also monitor for processes such as ‘ping.exe’ or ‘tracert.exe’, commonly used in network discovery.

To enhance overall security, it is recommended to implement proper network segmentation, limiting the subset of devices an attacker could detect. Additionally, restricting certain network protocols, such as ICMP (used by ping) and NetBIOS (used by ‘net view‘), can prevent an attacker’s ability to perform network discovery.

Robust monitoring and alerting mechanisms should also be in place to detect unusual network discovery activities. Early detection of such activities is crucial, as it can significantly improve the effectiveness of the prototype by allowing for a quicker response to potential threats.

- **Network Sniffing**: An attacker may passively sniff network traffic to gather information about the environment. This methodology could impact the prototype in two significant ways. First, the attacker might obtain data that the isolated environment cannot replicate, such as specific network patterns or confidential information that would be absent after isolation. Second, if the attacker conducts sniffing within the replicated environment, they might detect the lack of expected traffic, raising suspicion about the authenticity of the environment.

Mitigation strategies against these techniques primarily include segmenting the network and using encrypted connections to transmit sensitive information. Detection methodologies focus on monitoring command execution and process creation, which can help identify attempts to sniff network traffic and take timely action.

- **System Network Connections Discovery**: Attackers may attempt to discover active network connections on a system to gather detailed information about the network environment and identify potential targets for further exploitation. This technique lets the attacker see which hosts a system communicates with, what services are running, and how data flows within the network. Such information is crucial for understanding the network's architecture and identifying critical systems.

In the context of this prototype, if an attacker successfully performs system network connection discovery, they might notice that connections to previously active hosts are suddenly interrupted after the isolation. These discrepancies between the expected and actual network network connections could reveal to the attacker that they are not operating within the genuine network, thus compromising the effectiveness of the isolation strategy.

According to MITRE, mitigating this type of attack is challenging, as it often involves exploiting standard system features. However, detection strategies can include monitoring the execution of commands such as 'netstat' or 'show tcp brief', observing OS API calls, and tracking process creation activity that could list network connections.

- **System Network Configuration Discovery**: Attackers may seek detailed information about the network configuration and settings of the systems they access. This includes gathering IP and MAC addresses, network routes, and other configuration details that could be crucial for further exploitation. To acquire this information, attackers often utilise standard operating system utilities such as arp, ipconfig/ifconfig, netstat, and route, typically used for legitimate network administration.

The prototype relies on modifying the switch's configuration to enforce the correct VLAN tagging and isolate the attacker within an emulated environment. Importantly, the available IPs and MACs of the networking devices remain unchanged, reducing the likelihood of raising suspicion. Currently, the launched environment does not copy the MAC addresses. However, Proxmox allows setting it when creat-

ing new VMs and LXCs. Therefore, if MAC cloning is required, this would be possible by setting the MAC address in the 'pct set' command.

After the attacker has been isolated, the device is intended to be connected to the internet. In most cases, if the attacker is connecting remotely, we do not want to immediately block his access, preventing the gathering of valuable threat intelligence data. Due to the current hardware limitations in the router, it must receive all packets untagged from the switch. This means the attacker still has access to the switch while, for security reasons, it should not. If the switch becomes compromised, the attacker could revert the imposed isolation. A future prototype with better hardware could solve this issue.

MITRE suggests that mitigating these discovery attempts is difficult, as they often involve the legitimate use of built-in system features. Detection, however, can be enhanced by monitoring the execution of commands associated with network configuration discovery, such as arp or ipconfig/ifconfig. Like Systems Network Connections Discovery, tracking OS API calls and checking for process creation are the most common approaches to minimise these threats.

- **Security Software Discovery**: An attacker will try identifying secure software, configurations, defensive tools, and sensors installed on a system. Among these, they may focus on antivirus programs, intrusion detection systems (IDS) or endpoint detection and response. This information can tailor the attacker's tactics to avoid detection, remove security measures, or exploit vulnerabilities.

As the device the attacker controls does not change, the security software remains unchanged. However, the prototype relies on the network and host's security tools to identify suspicious behaviour and trigger an attacker's isolation. Therefore, minimising the attacker's capability to gather data from these tools is important for the prototype's performance.

Mitigating these techniques is challenging because it often involves the legitimate querying of system information. However, detection can be improved by monitoring the execution of commands that may reveal security software, such as 'tasklist' and 'wmic', monitoring for firewalls enumeration and their setting or rules, and process creation, which could list security software and defences.

- **Process Discovery**: Like Security Software Discovery, Process Discovery involves a set of strategies that an attacker might employ to gather information about the running processes on a system. This information can be particularly valuable as it enables attackers to understand the software and applications being used on systems within the network.

As the prototype is launching empty LXCs, if an attacker performs process discovery, they may quickly realise that the devices are not running any significant

processes. This discrepancy between the processes observed before and after the isolation could alert the attacker that they are not interacting with real devices. The lack of active processes, or the presence of only generic or minimal ones, might raise suspicion and lead the attacker to conclude that they are scanning an artificial environment rather than real devices. This issue highlights the limitation of the prototype in its current state. While the environment might be convincingly replicated regarding IP address and operating systems, the absence of realistic running processes could easily expose the isolation. However, the simulation of more authentic process activity could improve this in future iterations.

The risks associated with process discovery are challenging to mitigate, as the technique relies on built-in system features. Detection efforts should focus on monitoring command execution, OS API interaction, and process creation activities. Commands such as 'ps', 'tasklist', 'hostname', 'systeminfo', and 'qwinsta' are commonly used for process discovery, and they should be carefully logged and analysed to identify potential threats.

4.1.2 Defence Evasion

The prototype relies on the network's tools and systems to identify suspicious behaviour and trigger the isolation of an attacker. The network's security would be significantly compromised if the adversary finds ways to evade these defences, whether by avoiding the detection or escaping from the isolated environment. Therefore, understanding and mitigating defence evasion techniques shown in figure 4.4 is crucial to maintaining the network's integrity, ensuring robust security, and enabling the successful operation of the proposed solution.

From the long set of techniques (43), those that can have an impact in evading the defences set by the prototype affect the network segmentation and take over the control of the prototype to disable the isolation. Evaluating and checking whether the prototype would be vulnerable to any of the techniques is crucial. Among these, the most relevant procedures are those inside Network Boundary Bridging, Impair Defences, Abuse Elevation Control Mechanism and Exploitation for Defence Evasion. On the other side, there are others like Modify Registry, Modifying System Image, Impersonation, Hiding Artifacts or Hijacking Execution Flow, among others, which do not represent a threat due to the nature of the prototype's design.

Network Boundary Bridging

These techniques try to bypass the restriction on network segmentation by compromising the perimeter or internal network devices. The developed prototype relies on VLAN-based isolation enforced at the switch level to keep the adversaries contained. An incorrect mitigation of these methodologies could strongly affect the prototype's efficacy and decrease the security. Some of the most effective mitigation strategies are setting multi-factor authentication, privileged account management and filtering network traffic. These attacks can be detected by monitoring the network traffic

Figure 4.4: Defence Evasion MITR ATT&CK® column.

content through packet inspection and traffic pattern analysis.

Another relevant methodology to bypass network segmentation is VLAN hopping (51), an attack where the adversary customises the sending packets to include an additional VLAN tag in the Ethernet frame. To succeed in this process, the attacker must know the destination's VLAN tag, and the attacker's device must be connected to the switch through a trunk port. In the current implementation, the switch uses an access port, which should prevent the attacker from successfully performing these attacks.

- **Network Address Translation Traversal**: Network Address Translation (NAT) Traversal refers to attackers modifying a network device's NAT configuration to bypass traffic routing restrictions between trusted and untrusted networks. Ensuring that NAT configurations are secure and cannot be altered by attackers is essential to keep the attacker inside the prototype's isolation environment.

Impair Defences

The adversaries may deliberately modify components of the network environment to reduce or disable the defensive mechanisms. This can include preventative measures like firewalls or detection systems such as antivirus software, which are critical

for identifying and stopping malicious activity. Even though the prototype is not responsible for detecting threats, the success of its isolation strategy depends on maintaining effective defences.

- **Downgrade Attack**: In a similar way, we downgraded the firmware version to enable the switch automation scripts to function. An adversary could also maliciously downgrade a system to an outdated or vulnerable version. Such attacks typically exploit the system's backward compatibility, often forcing it into a less secure mode of operation. If an attacker successfully downgrades the switch firmware, it could make the isolation environment more susceptible to evasion. To mitigate these attacks, it is essential to ensure that systems are up-to-date and cannot revert to insecure versions. The firmware downgrade on the domestic switch was necessary because it required manual operation through a graphical user interface (GUI). This increased the difficulty of automating VLAN configuration changes and isolating the attacker. These limitations could be addressed by using appropriate hardware.

Abuse Elevation Control Mechanism

Abuse Elevation Control Mechanism refers to the tactics employed by adversaries to bypass privilege elevation controls and gain unauthorised higher-level permissions. This involves exploiting native control mechanisms designed to limit user privileges on a system. If an attacker succeeds in gaining administrative privileges, they could potentially take over the environment and disable the prototype. Mitigating these threats involves several key strategies: configuring the operating system to launch processes with known vulnerabilities without the setuid or setgid bits, regularly updating software to patch vulnerabilities, and using strict user account management to limit the distribution of high-level privileges. Command execution monitoring, OS API requests, and analysis of process creations are some of the key detection methods.

Exploitation for Defence Evasion

It involves attackers taking advantage of vulnerabilities in software or systems to bypass security features. This could include vulnerabilities in defensive tools, allowing attackers to disable or circumvent them. In the prototype's case, if the attacker exploits vulnerabilities in the detection tools, it could take control of them and evade the defences and prototype. These methodologies can be mitigated by isolation and application sandboxing, keeping the software updated and using exploit protection programs. These attacks are hard to detect and must be done by analysing the application's log content and creating abnormal processes.

4.1.3 Lateral Movement

Lateral Movement refers to the strategies attackers use to gain access to and remotely control systems within a network. This often involves navigating multiple

systems and accounts, potentially using legitimate credentials or deploying remote access tools to achieve their objectives. Figure 4.5 shows the different techniques gathered in MITRE. Internal Spearphising, Replication Through Removable Media, Taint Shared Content and Use of Alternative Authentication Material do not represent a threat to the prototype as the device is isolated with LXCs. However, this subsection discusses techniques related to the Exploitation of Remote Services and taking control of the virtualised environment.

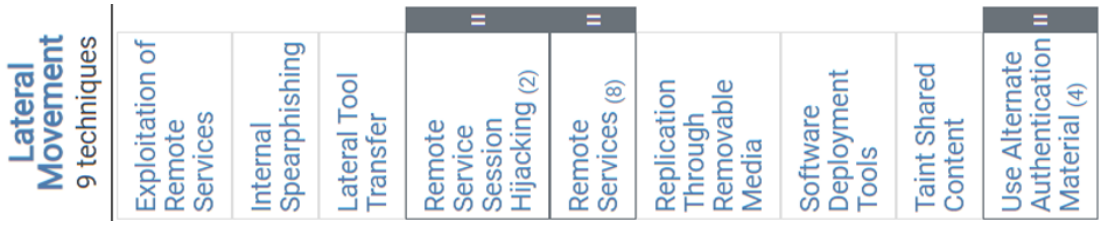

Figure 4.5: Lateral Movement MITR ATT&CK® column.

The risks of lateral movement when the attacker has been isolated in the environment replica are still present. Despite these being minimised, as the LXCs are currently not running any services or processes the attacker could exploit, it could still happen. In the case of running processes and services inside the LXCs, a thorough evaluation should be done to estimate the potential for an attacker to gain access by exploiting remote services (MITRE: Exploitation of Remote Services). To further reduce the risk, the LXCs have been configured to run as unprivileged containers and are protected by a firewall.

If an attacker gains access to an LXC, it would have a greater chance of escaping the container and compromising the Proxmox server. This situation is considered unlikely in the current state of the prototype as the LXCs are being launched with the most updated version of the distribution, and the adversary would likely require a zero-day vulnerability to succeed. Moreover, if the attacker uses unknown techniques to infiltrate these machines, the prototype would capture this information, thereby achieving the primary goal of gathering intelligence on novel attack techniques.

While LXCs are less resource-hungry than VMs, they have weaker isolation, as discussed in section 2.5.1. In the case of having a powerful Proxmox server, using virtual machines could be an alternative at the cost of increasing the launch time of the environment. Further discussion is covered in section 5.3 to use VMs with negligible time costs.

4.1.4 Exfiltration

Exfiltration involves attackers' strategies to transfer data from a compromised system to a location outside the targeted network while avoiding being detected. Figure

4.6 lists the techniques used for exfiltration. This process typically includes the prior data preparation and encryption for safe extraction. Detecting and mitigating these techniques is crucial, as the attacker controls the compromised device. The connection should be blocked immediately to prevent further damage if these operations are detected. Although the data on the compromised device may already be accessible to the attacker, preventing its exfiltration after the isolation remains a priority.

Figure 4.6: Exfiltration MITR ATT&CK® column.

While the project's scope includes creating the virtualised infrastructure and setting the SDN channel for the LXCs, it does not extend to monitoring the attacker's activities within the compromised device. Security tools, such as EDR systems or antivirus software, are responsible for checking the inner activities and blocking exfiltration attempts by the attacker.

If an adversary successfully moves laterally to an LXC or a Virtual Machine within the Proxmox server, these environments could potentially be used to exfiltrate data. Since their primary purpose is to log interactions with the attacker into a centralised database through the SDN channel, these hosts currently do not run their security processes. Consequently, if an attacker manages to transfer data from the compromised device to an LXC or VM, these could serve as intermediary storage with weaker security measures. Although these virtualised environments are isolated via VLANs and lack direct internet connectivity, if an attacker bypasses the isolation, they could establish a connection to the external network, enabling its exfiltration. This situation would make the exfiltration process entirely dependent on monitoring network traffic and its contents.

4.2 Environment Replication

This prototype intends to replicate virtually a physical environment. Therefore, the best way to evaluate this section is by comparing the Nmap scan the attacker can perform before and after the isolation.

To achieve more meaningful results, the five switch ports were set as follows:

- Port 1: Surface Pro 7 (running Ubuntu) (192.168.1.104)
- Port 2: Attacker device (running Ubuntu) (192.168.1.220)
- Port 3 & 4: Proxmox server (192.168.1.233)

- Port 5: HP printer (192.168.1.51)

In this way, more than one extra device shares the network segment with the attacker and will later be cloned. The port dedicated to the router would add noise to the current scan from other devices in the network as the router does not support using VLAN tagging itself.

The relevant fragments from the previous (Figure 4.7) and post (Figure 4.8) isolation Nmap scans focus on addresses 192.168.1.51 and 192.168.1.104, which are the hosts that have been cloned into the Alpine and Ubuntu LXCs respectively using the preexisting database approach. In the case of using the Nmap scanner, as previously mentioned, it will not be capable of distinguishing the distributions or even operating systems of the devices. In this case, it would launch them in the same Alpine LXC. As the MAC address is currently not being customised when creating the network interface, if this happens, Nmap would report the same MAC address for both IPs after the scan. This could catch the attacker's attention and raise the suspicion of interacting with a virtualised environment.

Figure 4.7: Nmap scan from Attackers perspective while not being isolated, the complete scan can be found in Appendix D.1.

```
# Nmap 7.94SVN scan initiated Sun Aug 25 05:10:19 2024 as: nmap -F -O
↪  -T5 -n --max-retries 5 --host-timeout 15m --scan-delay 0 -oN
↪  not_isolated 192.168.1.0/24

[...]

Nmap scan report for 192.168.1.51
Host is up (0.0024s latency).
Not shown: 93 closed tcp ports (reset)
PORT     STATE SERVICE
80/tcp   open  http
139/tcp  open  netbios-ssn
443/tcp  open  https
445/tcp  open  microsoft-ds
631/tcp  open  ipp
8080/tcp open  http-proxy
9100/tcp open  jetdirect
MAC Address: 98:4B:E1:85:4B:8A (Hewlett Packard)
Device type: general purpose|printer|VoIP adapter
Running: Wind River VxWorks, HP embedded, Vocality embedded
OS CPE: cpe:/o:windriver:vxworks
OS details: VxWorks: HP printer or Vocality BASICS Four Wire VoIP
↪  gateway
Network Distance: 1 hop

```

```
Nmap scan report for 192.168.1.104
Host is up (0.00060s latency).
Not shown: 99 closed tcp ports (reset)
PORT   STATE SERVICE
80/tcp open  http
MAC Address: A0:4A:5E:E2:36:A1 (Microsoft)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.8
Network Distance: 1 hop

[...]

OS detection performed. Please report any incorrect results at
↪  https://nmap.org/submit/ .
# Nmap done at Sun Aug 25 05:10:28 2024 -- 256 IP addresses (5 hosts
↪  up) scanned in 9.84 seconds
```

Figure 4.8: Nmap scan from Attackers perspective while being isolated, the complete scan can be found in Appendix D.2.

```
# Nmap 7.94SVN scan initiated Sun Aug 25 04:55:26 2024 as: nmap -F -O
↪  -T5 -n --max-retries 5 --host-timeout 15m --scan-delay 0 -oN
↪  isolated 192.168.1.0/24

[...]

Nmap scan report for 192.168.1.51
Host is up (0.00022s latency).
All 100 scanned ports on 192.168.1.51 are in ignored states.
Not shown: 100 closed tcp ports (reset)
MAC Address: BC:24:11:60:3D:86 (Unknown)
Too many fingerprints match this host to give specific OS details
Network Distance: 1 hop

Nmap scan report for 192.168.1.104
Host is up (0.00044s latency).
All 100 scanned ports on 192.168.1.104 are in ignored states.
Not shown: 100 filtered tcp ports (no-response)
MAC Address: BC:24:11:60:3D:86 (Unknown)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.8
Network Distance: 1 hop
```

```
23
24 [...]
25
26 OS detection performed. Please report any incorrect results at
   ↪ https://nmap.org/submit/ .
27 # Nmap done at Sun Aug 25 04:55:36 2024 -- 256 IP addresses (5 hosts
   ↪ up) scanned in 10.40 seconds
28
```

As it can be seen, the hosts have been perfectly cloned, and at the time of performing an SSH connection, it reports that the stored SSH fingerprint does not match, indicating that the SSH communication is being done to a new host (Figure 4.9).

```
udesk@udesk:~$ ssh ubuntusurface@192.168.1.104
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ED25519 key sent by the remote host is
SHA256:PqQKWtZqAvs37s3RtF8KIFECVUrVGKhBiGjEofMBSoc.
Please contact your system administrator.
Add correct host key in /home/udesk/.ssh/known_hosts to get rid of this message.
Offending ECDSA key in /home/udesk/.ssh/known_hosts:6
  remove with:
  ssh-keygen -f '/home/udesk/.ssh/known_hosts' -R '192.168.1.104'
Host key for 192.168.1.104 has changed and you have requested strict checking.
Host key verification failed.
```

Figure 4.9: Allowed Communications

4.2.1 Benchmarks

This section focuses especially on recording the launch time of various environments depending on the needs. Minimising the time taken to have a fully running isolated system is crucial to make this prototype feasible for incident response procedures to stop the risks the business is undergoing.

The following experiment was designed to record a close-to-reality situation where the three most common LXC types (Ubuntu, Debian & Fedora) are needed in an environment replication. It starts by querying the SQLite database and manipulating data to prepare the machine's information before deployment. Therefore, the runtime is expected to be slightly longer than deploying the LXCs exclusively, giving a more accurate idea of a realistic environment deployment from the prototype. The initial LXC deployment implementation used a sequential procedure to launch the replica. After fifty deployments of the three most common LXCs (Ubuntu, Debian & Fedora), the runtime was, on average, 19.271 ± 0.216 (95% CI). This implementation would make this prototype not viable as waiting almost 20 seconds while being attacked can be devastating with really talented or automated attackers.

Further strategies were developed and tested to reduce the deployment time. The process can speed up drastically by parallelising different parts of the deployment. The first optimisation implied applying parallelisation to the launch of the LXCs after they were sequentially cloned. After 50 deployments of the previous setup, this new strategy resulted in an average of 12.326 ± 0.727 seconds (95% CI). This was already a 36% improvement over the original time. Finally, the cloning process was also parallelised, reducing the same environment setup to 8.116 ± 0.189 seconds (95% CI) on average after 50 deployments, achieving another 34% improvement. Figure 4.10 shows the results where 'S' stands for sequential and 'P' for parallel.

Focusing exclusively on the LXC launch time, after recording the deployment of 350 environments and 650 LXCs with different configurations, figure 4.11 shows the times recorded.

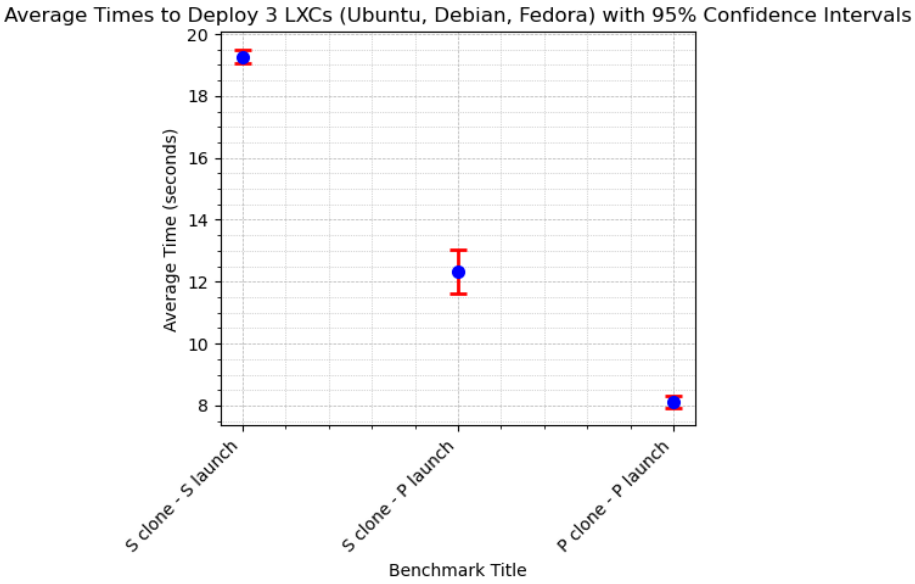

Figure 4.10: Improved deployment of LXCs through parallelisation

As it can be seen in figure 4.11, booting an LXC, regardless of its distribution, takes on average 5.974 ± 0.431 (95% CI) seconds, being 5.980 ± 0.0544 for Ubuntu, 5.975 ± 0.056 for Alpine, 5.961 ± 0.043 for Debian, and 5.979 ± 0.038 for Fedora. Considering that different LXCs distributions differ on the root file system, being close to base installation, while using the host's kernel. Therefore, they should not differ much in their launch time. These launching results are probably limited directly by the Proxmox LXCs' resource allocation and deployment process. Qubes OS (134), another Linux distribution which proposes achieving better security by isolating each process in VMs, has the same boot up limitations which can be bypassed by making some changes to the launching process.

Launching more LXCs comes at a launch time cost. However, parallelising the

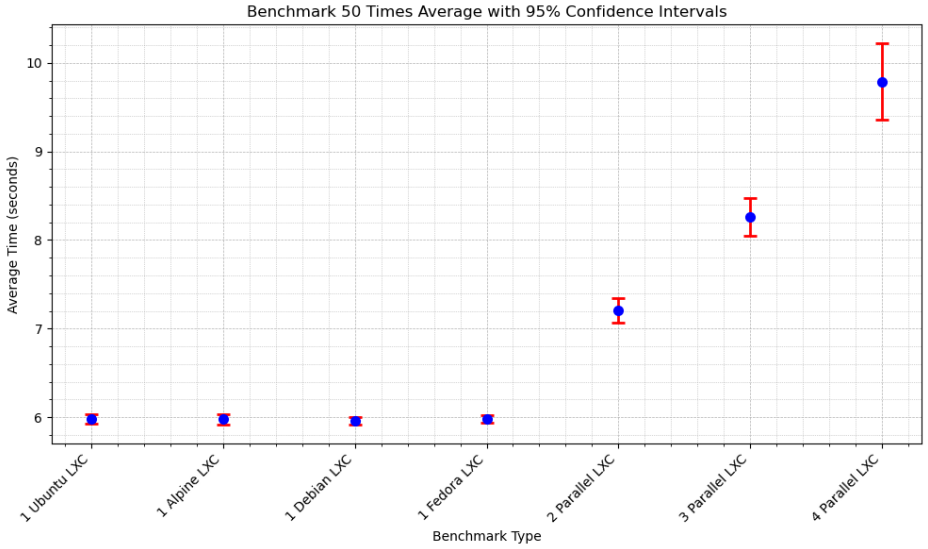

Figure 4.11: Deployment time while increasing number of LXCs

process drastically reduces boot up times compared to the sequential alternative. Launching two LXCs take on average 7.203 ± 0.138 (95% CI) seconds, while larger environments with three and four LXCs take 8.262 ± 0.212 (95% CI) and 9.787 ± 0.431 (95% CI) seconds respectively.

Comparing the recorded results from this experiment with the previous one, if launching an LXC takes on average 5.974 ± 0.431 (95% CI), then launching three LXCs would take $5.974 \times 3 = 17.922$, $\pm 0.431 \times 3 = \pm 1.293$, which in the worst case is 19.215, a pretty close value to the 19.271 ± 0.216 (95% CI) seconds on average recorded in the first experiment. Adding on, comparing the results for three LXCs cloned and launched in parallel with the previous experimental result, including the database querying runtime, both show close figures (clean) 8.262 ± 0.212 (95% CI), (with database) 8.116 ± 0.189 seconds (95% CI).

4.3 Network Segmentation

Using the switch proposed configurations, the attacker is effectively isolated in a network segment with the virtualised environment while keeping internet access. The following table represents a simplified network setup to evaluate the allowed communications:

In the beginning, the Proxmox should be in a separate network segment (Figure 4.12), preventing the attacker from communicating with it, neither with any internal LXC running (Figure 4.13). Moreover, it should not be able to interact with devices through the SDN channel (Figure 4.14).

The following figures are taken from the Attacker's machine (192.168.1.220)

IP Address	Host Name	Virtualised	Type	SDN IP	Isolated
192.168.1.233	PC 1 Prox Server	No	-	-	Yes
192.168.1.220	PC 2 Attacker	No	-	-	No/Yes
192.168.1.104	Testing Host 1	Yes	LXC	192.168.2.104	Yes
192.168.1.106	Testing Host 2	Yes	LXC	192.168.2.106	Yes
192.168.1.51	External Host 3	No	-	-	No

Table 4.1: Hosts Setup for Network Connectivity Evaluation

while not being isolated:

```
udesk@udesk:~$ ping -c 3 192.168.1.233
PING 192.168.1.233 (192.168.1.233) 56(84) bytes of data.
From 192.168.1.220 icmp_seq=1 Destination Host Unreachable
From 192.168.1.220 icmp_seq=2 Destination Host Unreachable
From 192.168.1.220 icmp_seq=3 Destination Host Unreachable

--- 192.168.1.233 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2055ms
pipe 2
```

Figure 4.12: Attacker & Proxmox server cannot communicate while being in different Network Segments

```
udesk@udesk:~$ ping -c 3 192.168.1.104
PING 192.168.1.104 (192.168.1.104) 56(84) bytes of data.
From 192.168.1.220 icmp_seq=1 Destination Host Unreachable
From 192.168.1.220 icmp_seq=2 Destination Host Unreachable
From 192.168.1.220 icmp_seq=3 Destination Host Unreachable

--- 192.168.1.104 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2047ms
pipe 3
```

Figure 4.13: Attacker & LXCs cannot communicate while being in different Network Segments

```
udesk@udesk:~$ ping -c 3 192.168.2.104
PING 192.168.2.104 (192.168.2.104) 56(84) bytes of data.

--- 192.168.2.104 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2077ms
```

Figure 4.14: Attacker & LXCs cannot communicate through the SDN while being in different Network Segments

To enforce better security, the LXCs should not be able to connect directly to the internet or through the SDN (Figure 4.15).

LXCs can communicate through the channel shared with the adversary or through the SDN channel (Figure 4.16).

After the attacker has been isolated, it should be able to communicate to the LXCs through the available channel (Figure 4.17), but it should not through the

```
root@CT104:~# ping -c 3 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
From 192.168.2.104 icmp_seq=1 Destination Host Unreachable
From 192.168.2.104 icmp_seq=2 Destination Host Unreachable
From 192.168.2.104 icmp_seq=3 Destination Host Unreachable

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2031ms
pipe 3
root@CT104:~# wget google.com
--2024-08-27 00:29:59--  http://google.com/
Resolving google.com (google.com)... failed: Temporary failure in name resolution.
wget: unable to resolve host address 'google.com'
```

Figure 4.15: LXCs have no internet access

```
root@CT104:~# ip route get 192.168.1.106
192.168.1.106 dev eth0 src 192.168.1.104 uid 0
    cache
root@CT104:~# ip route get 192.168.2.106
192.168.2.106 dev eth1 src 192.168.2.104 uid 0
    cache
```

Figure 4.16: LXCs use the separated channels to communicate

```
udesk@udesk:~$ ping -c 3 192.168.1.104
PING 192.168.1.104 (192.168.1.104) 56(84) bytes of data.
64 bytes from 192.168.1.104: icmp_seq=1 ttl=64 time=0.399 ms
64 bytes from 192.168.1.104: icmp_seq=2 ttl=64 time=0.224 ms
64 bytes from 192.168.1.104: icmp_seq=3 ttl=64 time=0.326 ms

--- 192.168.1.104 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2068ms
rtt min/avg/max/mdev = 0.224/0.316/0.399/0.071 ms
```

Figure 4.17: Attacker can communicate to LXCs through the normal channel

```
udesk@udesk:~$ ping -c 3 192.168.2.104
PING 192.168.2.104 (192.168.2.104) 56(84) bytes of data.

--- 192.168.2.104 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2025ms
```

Figure 4.18: Attacker cannot communicate to LXCs through the SDN channel

SDN channel (Figure 4.18).

The attacker must not be able to communicate with a device external to the isolated environment (Figure 4.19):

```
udesk@udesk:~$ ping -c 3 192.168.1.51
PING 192.168.1.51 (192.168.1.51) 56(84) bytes of data.

--- 192.168.1.51 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2075ms
```

Figure 4.19: Attacker cannot communicate outside the isolated network segment to external devices.

Moreover, the attacker should never be able to connect to the Proxmox server or the docker server running the VPN, DNS and web server. To prevent this, rules in the Proxmox server's firewall were added to drop any packets incoming packets from the attacker's device. This process can be seen in the Proxmox server live firewall logs (Figure 4.20).

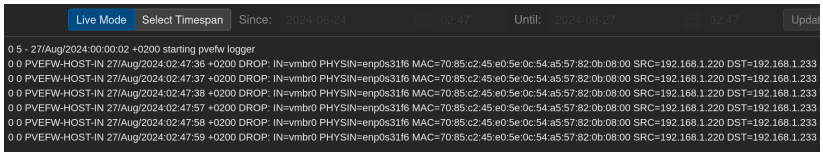

Figure 4.20: Firewall logs showing attacker packets being dropped if they are in the same VLAN.

Various measures were implemented and evaluated to ensure that an attacker cannot sniff SDN channel packets using tools like Wireshark. During the evaluation process, conducted via SSH, we used Tshark, the command-line version of Wireshark, alongside tcpdump to monitor packet transmissions.

However, it was observed that when communication occurs between hosts within the Proxmox environment, the routing is internal, which prevents the attacker's machine from capturing these packets. This effectively means the attacker could sniff any interaction between an LXC (Linux Container) and other hosts external to the Proxmox environment. In contrast, communications within the Proxmox environment remain inaccessible to the attacker's packet-sniffing attempts.

The attacker is expected to perceive network behaviour similar to the original segment before isolation. This was quickly verified using Wireshark. However, suppose the environment replica depends on misleading the attacker by routing packets solely between LXCs. In that case, all communications remain internal, and thus, the attacker would not receive any packets to sniff.

To address this issue, we attempted to set the network interface to promiscuous mode to force packet forwarding to the attacker's machine. Unfortunately, these efforts were unsuccessful. Consequently, further research is necessary to develop a method that would allow the attacker to engage in packet sniffing, thereby enhancing the realism of the isolated environment and better simulating a real-world scenario.

The currently developed mechanism to change the domestic switch (TP-Link TL-SG105E) is a fix and not preferable. It is obvious that to make the prototype work. This specific approach was the closest way to recreate the automation process required in this piece of hardware. However, this functionality is commonly built-in in enterprise switches, allowing remote and automated configuration via SSH (135) or SNMP (136). This would not require lowering the security of the hardware and exposing it to a higher risk. Other more recent switches allow network automation through the use of Ansible (137).

Finally, table 4.21 summarises the allowed connections.

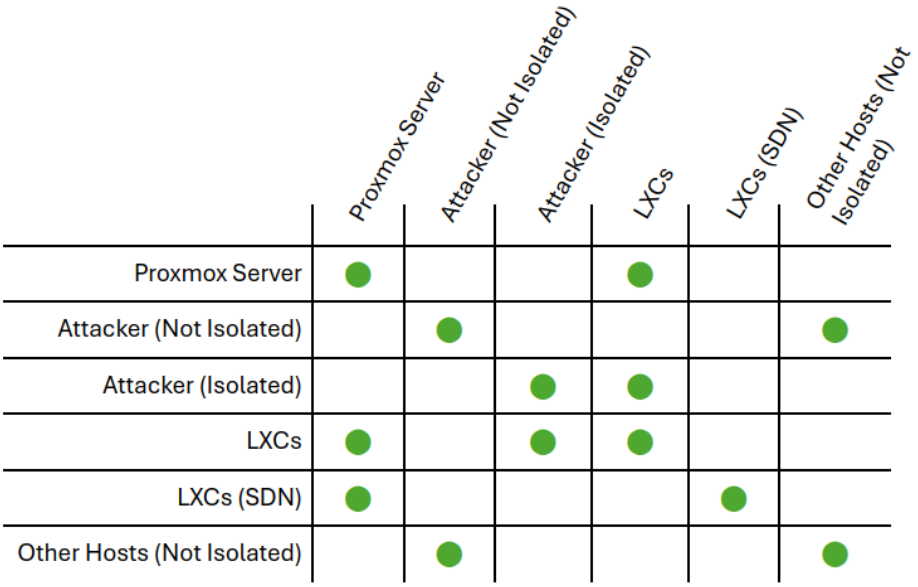

Figure 4.21: Allowed communications.

4.3.1 Benchmarks

Unfortunately, making the configuration changes at the TP-Link TL-SG105E through the scripts is not an immediate process. After running over 200 configuration changes in the switch, the average runtime to make the VLAN updates is 5.602 $\pm$0.040 seconds (95% CI).

This long runtime could be the result of the current code implementation. It takes time to start the session and must wait for packet confirmations, which are useful to know whether the switch has successfully changed its VLAN settings and the attacker has been correctly isolated.

Considering that the environment's creation is not immediate, especially with the current hardware, the current runtime with the domestic switch is relatively enough to prove the concept. The adequate enterprise switch with the built-in functionality would drastically reduce the runtime and be recommended for future, more advanced prototypes.

4.4 Multiple simultaneous attackers

The current project proposes a solution when considering having multiple simultaneous attackers. As IP addresses are different in each network segment and, in most cases, use a different subnet, the generated ProxIso3 VLAN can be reused for multiple attackers. If attackers are in the same network segment and subnetwork, the same attacker will likely control both devices simultaneously. Therefore, if the attacker wants to send packets between these devices to make an advanced attack,

the attacker should see the expected interaction. Otherwise, it would be obvious that the attacker is interacting with a virtual environment. In the case of attackers sharing the same subnetwork but being in a different network segment, a new VLAN would be required to keep the original environment of each attacker independent and realistic.

This approach allows machines to be reused whenever different attackers exist in separate subnetworks. The machines would require updating their network configuration to include a new network interface with adequate IP (and MAC). In the worst-case scenario, new LXCs would be launched to cover distributions that are not already running. As a result, the solution drastically minimises the hardware requirements and allows for the ability to respond to threats at scale.

However, a limit must be set to the number of compromised devices that can share a networking segment. The solution assumes that the attacker might take control of a few devices in the network, but allowing the attacker to keep operating while already having a large number of devices is a concern from a security point of view. The intention is to create the infrastructure to perform threat analysis, not allowing attackers to operate and control all the devices in the isolated environment. If the attacker compromises multiple machines in the same segment and is isolated, the attacker might take advantage of this, extracting all the possible valuable data from these machines. Therefore, a limit decided by the business security team must be set, considering this solution's potential risks and benefits.

4.5 Current Alternatives

The current prototype is innovative as it introduces a bottom-up solution that isolates attackers by dividing the network and also recreates the original network in a virtual environment. In contrast, Memary et al. (90) used LXCs to create a honeynet but used two LXC layers to improve scalability. Our project tackles scalability by setting up network bridges within a single LXC, reducing hardware requirements.

Moreover, SDNs have been used to counter cyberattacks, with traffic redirection being a common strategy (68). Beigi-Mohammadi et al. (138) employed this method to redirect suspicious DDoS traffic to a virtual copy of the application in an isolated environment. Our project does something similar, but we believe the specific use of SDNs in our prototype to create a secure communication channel to prevent packet sniffing by the attacker is novel or has not been documented. In other works, such as Charan et al. (102), this issue has been addressed with a "Chatterbot" application that camouflaged and sent the information gathered about the attacker's behaviour.

This solution also explores alternatives and challenges in cloning an active physical network with "real" hosts. In contrast, projects by Peuster (99), Kong et al. (92), Charan et al. (102), and Memary et al. (90) provide network models or artificial

replications.

Like Kong et al. (92), this project builds on previous solutions to create an automated method, aiming to reduce cyberattacks, as shown by Charan et al. (102).

This project acknowledges that network topologies vary across organisations and offers a flexible solution that does not rely on the latest hardware. It is designed to be flexible and suitable for any scale and requirement.

Finally, similar to Charan et al. (102), this approach aims to improve the quality of threat intelligence and enhance the data collected from attackers.

Chapter 5

Further Work

5.1 Enhancing Proxmox Security

While Proxmox includes a built-in firewall, it may be more effective to implement additional security measures by installing pfSense or OPNsense. This enhancement would offer better protection for the Proxmox server when an attacker interacts with the isolated environment, preventing any interaction between the adversary and the Proxmox server.

5.2 Improving Environment Replica

The current approach to the isolated environment minimises the number of LXCs launched to reduce the time required for environment creation and to conserve hardware resources. However, this strategy has certain limitations. A potential improvement could involve re-configuring the environment to group IP addresses not only by the operating system and distribution but also by open ports before generating LXCs and VMs. This adjustment would enable the LXCs and VMs to replicate the same open ports as the original environment, resulting in a more convincing replica.

5.3 Enhancing Environment Launch Time

An improvement to the prototype could involve pre-configuring a set of VMs and LXCs and keeping them running inside the Proxmox server. When isolation is triggered, only those VMs and LXCs that match the attacker's environments will change their network interface configuration to match the appropriate IP addresses. At the same time, the rest would turn off their network interface. This approach would substantially reduce the time needed to clone and launch the environment, making the process instantaneous and achieving an immediate security response system.

5.4 Adequate Hardware

The use of appropriate and higher-quality hardware is crucial for the optimal functioning of this approach. Upgrading to more powerful hardware could significantly enhance the Proxmox server's ability to run multiple VMs and LXCs simultaneously without compromising performance, particularly if virtualised Windows instances are required. The switch used in this prototype is a domestic model with limitations not found in enterprise-level devices. Investing in the appropriate hardware would create a more enterprise-like environment, facilitating better development and rigorous evaluation of the tools and the proposed strategy.

In addition, the current switch configuration is limited by the router's limitations to using VLANs. This has prevented the prototype from showing a complete solution where the devices are correctly isolated while keeping internet access.

Chapter 6

Conclusion

Traditional honeypots are often easily identified and bypassed by attackers. However, recent research by Charan et al. (102) has demonstrated that using virtual machines (VMs) and Linux containers (LXCs) with the appropriate configuration can effectively deceive attackers, encouraging prolonged engagement. This extended interaction gives defenders valuable intelligence on the attacker's behaviour and methods. Building on these insights, this project has designed and implemented an automated infrastructure to exploit such capabilities.

Typically, when a threat is detected within a network, the standard response is to disconnect and quarantine compromised devices or network segments, effectively preventing further damage (5). However, this project proposes an alternative approach by isolating the attacker within a recreated virtual. The goal is to prolong the duration of the attacker's engagement, enabling the collection of more detailed data on their techniques, tactics, and procedures (TTPs).

The core solution involves dynamically replicating the compromised network environment with its hosts while reconfiguring VLAN segmentation at the switch level to isolate the attacker in a secure virtual space. An internal Software-Defined Network (SDN) channel, designed to be undetectable and non-sniffable by the attacker, enables a secure monitoring and real-time recording of all activities within the virtualised environment. Although SDNs have been used in cybersecurity to improve network management and security, their application in creating a secure, real-time monitoring channel for observing attackers in a virtual replica has not been documented, presenting a novel contribution to the field. By capturing the attacker's behaviour in this isolated environment, defenders can extract critical intelligence, which could be used to refine defensive strategies and enhance future threat mitigation techniques. The infrastructure established here serves as a foundation for integrating more advanced threat intelligence tools, enhancing future security while minimising risks to the actual network.

This prototype successfully demonstrates the feasibility of dynamically recreating physical networks within an isolated virtual environment, proposing an innovative use and deployment of honeynets in cybersecurity. The ability to rapidly deploy

and configure LXCs and VMs in response to detected attacks ensures a flexible and scalable system capable of addressing modern cybersecurity challenges as networks become more complex and threats more sophisticated.

Additionally, this project lays the basis for further research and development, particularly in real-time duplication and virtualising physical network infrastructure. This area is relatively undocumented, leaving significant opportunities to enhance the prototype, refine its capabilities, and explore further use cases within enterprise environments.

Finally, by integrating automation with dynamic isolation, the proposed system not only advances current honeypot technology but also provides a versatile and scalable tool that has the potential to significantly enhance threat intelligence gathering and network security in the ongoing fight against evolving cyber threats.

Appendix A

Switch LAN Configuration Files

A.1 VLAN Reset

Figure A.1: YAML file with all ports set to VLAN 1

```yaml
default-vlan: Main
switches:
  Switch-Main:
    ip: 192.168.1.43
    type: "TLSG105E"
    username: admin
    password: *****
    description: "Switch-Main"
    default-vlan: Default
    ports:
      port1:
      port2:
      port3:
      port4:
      port5:

vlans:
  Default:
    vlan-id: 1
```

A.2 VLAN Initial Setup

Figure A.2: YAML file with Proxmox server isolated and the Attacker in General

```yaml
    default-vlan: General
    switches:
      Switch-Main:
        ip: 192.168.1.43
        type: "TLSG105E"
        username: admin
        password: *****
        description: "Switch-Main"
        default-vlan: General
        ports:
          port1:
            description: "Router"
            vlan: Default
            type: untagged
            untagged-vlan-id: 2
            pvid: 2
        # In the ideal case the following configuration would be used
            # vlan General
            # type: tagged
            # tagged-vlan-ids: [3, 11]
            # pvid: 3
          port2:
            description: "Ubuntu Tower"
            vlan: General
            type: untagged
            untagged-vlan-id: 2
            pvid: 2
          port3:
            description: "Proxmox Tower"
            vlan: ProxIso
            type: trunk
          port4:
            description: "Additional RJ45"
            vlan: ProxIso
            type: tagged
            tagged-vlan-id: [2, 11] #13, 14, 15
            pvid: 11
          port5:
            description: "Printer"
            vlan: General
            type: untagged
            untagged-vlan-id: 3
            pvid: 3

    vlans:
      # Not in use
      Default:
        vlan-id: 1
      # Provides internet
      Internet:
        vlan-id: 2
```

```
52        # Contain all devices
53        General:
54          vlan-id: 3
55        # Other departments
56        # DepartmentA:
57        #   vlan-id: 4
58        # DepartmentB:
59        #   vlan-id: 5
60
61        # Proxmox isolation
62        ProxIso:
63          vlan-id: 11
```

A.3 VLAN Attacker Isolation

Figure A.3: YAML file where Attacker is isolated with Proxmox server

```
1     default-vlan: General
2     switches:
3       Switch-Main:
4         ip: 192.168.1.43
5         type: "TLSG105E"
6         username: admin
7         password: *****
8         description: "Switch-Main"
9         default-vlan: General
10        ports:
11          port1:
12            description: "Router"
13            vlan: Default
14            type: untagged
15            untagged-vlan-id: 2
16            pvid: 2
17        # In the ideal case the following configuration would be used
18            # vlan General
19            # type: tagged
20            # tagged-vlan-ids: [3, 11, 13]
21            # pvid: 3
22          port2:
23            description: "Ubuntu Tower"
24            vlan: ProxIso3
25            type: untagged
26            untagged-vlan-id: 13
27            pvid: 13
28          port3:
```

```
            description: "Proxmox Tower"
            vlan: ProxIso
            type: trunk
          port4:
            description: "Additional RJ45"
            vlan: ProxIso
            type: tagged
            tagged-vlan-id: [2, 11, 13] #14, 15
            pvid: 11
          port5:
            description: "Printer"
            vlan: General
            type: untagged
            untagged-vlan-id: 3
            pvid: 3

    vlans:
      # Not in use
      Default:
        vlan-id: 1
      # Provides internet
      Internet:
        vlan-id: 2
      # Contain all devices
      General:
        vlan-id: 3
      # Other departments
      # DepartmentA:
      #   vlan-id: 4
      # DepartmentB:
      #   vlan-id: 5

      # Proxmox isolation
      ProxIso:
        vlan-id: 11

      # Isolation of General and Departments
      ProxIso3:
        vlan-id: 13
      # ProxIso4:
      #   vlan-id: 14
      # ProxIso5:
      #   vlan-id: 15

```

Appendix B

Environment's Operating System

B.1 Automated Scanning Script

Figure B.1: Script used to perform operating system fingerprinting to generate an approximate accurate environment

```
import os
import re
import sys
import json
import subprocess
import concurrent.futures

def raw_scan(subnetwork:str = None):
    """
    This is a fast scan to gather general information about the devices
    ↪  in the network.
    """

    try:
        # Creating Nmap command
        nmap_cmd = f"sudo nmap -sn -n -T5 --host-timeout 15m
        ↪  --scan-delay 0 {subnetwork}"
        nmap_cmd = nmap_cmd.split(" ")
        # Running Nmap and capturing the output
        return subprocess.run(nmap_cmd, capture_output=True,
        ↪  text=True).stdout

    except Exception as e:
        print(f"Error: {e}")


def scan_ip_OS(ip:str = None):
```

```
    """
    Scanning the IP addresses with more detailed information about the
    ↪  OS and version
    """

    print(f"Scanning {ip}")
    try:
        # Creating nmap command
        nmap_cmd = f"sudo nmap -F -O -T5 -n --max-retries 2
        ↪  --host-timeout 15m --scan-delay 0 {ip} --version-light
        ↪  --osscan-limit --max-os-tries 1 -oN ./Scans/{ip}"
        nmap_cmd = nmap_cmd.split(" ")
        # Running Nmap
        subprocess.run(nmap_cmd, stdout=subprocess.DEVNULL)
        # subprocess.run(nmap_cmd)

    except Exception as e:
        print(f"Error: {e}")


def get_OS(file_path:str = None):
    """
    Parsing the scan results to get OS information
    """

    try:
        with open(file_path) as f:
            content = f.read()
            # Parsing aggressive OS guesses
            guesses = re.findall(r"Aggressive OS guesses: (.+)",
            ↪  content)
            if guesses:
                #\(
                guesses = re.split(r"\%\)\,", guesses[0])
                os_list = []
                for guess in guesses:

                    os_info = re.split(r"\((?=\d)", guess) # \)
                    # os_info = guess.split(" (")
                    os_name = os_info[0].strip()
                    probability = os_info[1].strip().replace("%)", "")
                    os_list.append({
                        "name": os_name,
                        "probability": probability
                    })
                return os_list
            else:
                return None
```

```
69
70      except Exception as e:
71          print(f"Error: {e}")
72
73
74  def main(subnetwork):
75      print(f"Scanning subnetwork {subnetwork}")
76
77      # Performing a fast ping scan to know which devices are in the
        ↪  network
78      content = raw_scan(subnetwork)
79
80      # regex to get IP addresses
81      ip_addresses =
        ↪  re.findall(r"\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b", content)
82
83      # Using parallel processing to scan the IP addresses further and
        ↪  get OS information
84      with concurrent.futures.ThreadPoolExecutor() as executor:
85          executor.map(scan_ip_OS, ip_addresses)
86
87      # Create a results.json file to store the results
88      # Create a dictionary to store the results
89      results = {}
90
91      # parsing scan results to get IP addresses with OS information
92      scan_dir = "./Scans"
93      devices = []
94      for file_name in os.listdir(scan_dir):
95          file_path = os.path.join(scan_dir, file_name)
96          if os.path.isfile(file_path):
97              ip = file_name
98              # os_list = check_file_OS(file_path)
99              os_list = get_OS(file_path)
100             if os_list:
101                 device = {
102                     "ip": ip,
103                     "os": []
104                 }
105                 for os_name in os_list:
106                     os_info = {
107                         "name": os_name.get("name"),
108                         "probability": os_name.get("probability")
109                     }
110                     device["os"].append(os_info)
111                 devices.append(device)
112             else:
113                 continue
```

```
114
115     results = {
116         "Devices": devices
117     }
118
119     # Write the results to data.json
120     with open("data.json", "w") as f:
121         json.dump(results, f)
122
123    # Ensure that the data.json file is accessible to all users
124     try:
125         chmod_cmd = "chmod 666 data.json"
126         subprocess.run(chmod_cmd.split(" "))
127     except Exception as e:
128         print(f"Error: {e}")
129
130
131 def cleanScans():
132     """
133     Clean the Scans directory
134     """
135     for file_name in os.listdir("./Scans"):
136         file_path = os.path.join("./Scans", file_name)
137         if os.path.isfile(file_path):
138             os.remove(file_path)
139
140
141 if __name__ == "__main__":
142     # Getting the subnetwork and credentials to execute the scan
143     try:
144         subnetwork = sys.argv[1] if len(sys.argv) > 1 else None
145     except Exception as e:
146         print(f"Subnetwork Error: {e}")
147         sys.exit(1)
148
149     # Running script
150     main(subnetwork)
151     cleanScans()
```

B.2 Operating System Database

IP Address	Device Name	VLAN	OS	Distribution
192.168.1.1	Router Askey RTF8225VW	1	Linux	
192.168.1.43	Switch TP-Link TL-SG105E	1	Linux	
192.168.1.236	Proxmox Scanner LXC	2	Linux	Ubuntu
192.168.1.233	PC 1 - Proxmox Server	3	Linux	
192.168.1.220	PC 2 - Attacker Device	2	Linux	Ubuntu
192.168.1.230	Docker Server	1	Linux	Ubuntu
192.168.1.221	Debian Server	2	Linux	Debian
192.168.1.104	Person A Desktop 2	2	Linux	Ubuntu
192.168.1.37	Person A Desktop 1	1	Windows	11
192.168.1.54	Person A Phone	1	Android	13
192.168.1.139	Person B Desktop	1	Windows	11
192.168.1.32	Person B Phone	1	IOS	17
192.168.1.50	Person C Desktop	1	Windows	11
192.168.1.49	Person C Phone	1	Android	12
192.168.1.46	Person D Desktop	2	Windows	10
192.168.1.42	Person D Phone	1	Android	14
192.168.1.44	Chromecast	1	Linux	
192.168.1.53	TP-Link Extender	1	Linux	
192.168.1.34	Smart Plug	1	Linux	
192.168.1.200	TV Decoder	1	Linux	
192.168.1.51	Printer HP	2	Linux	

Table B.1: Complete Table of Devices in Network

Appendix C

Environment Creation

C.1 LXC Cloning Script

Figure C.1: Generic Script to clone and configure LXCs

```
#!/bin/bash
# run example: ./launch_lxcs.sh Ubuntu 112 1002 192.168.1.1
↪  192.168.2.12 192.168.1.207 192.168.1.208 192.168.1.209
↪  192.168.1.210 192.168.1.211

DISTRO="$1"
CT_ID="$2"
TEMPLATE_ID="$3"
GW="$4"
SDN_IP="$5"
# Additional IPs
shift 5
ADDITIONAL_IPS="$@"

# Cloning the LXC container
pct clone $TEMPLATE_ID $CT_ID
# Setting hardware limits
pct set $CT_ID --hostname $DISTRO --cores 1 --cpulimit 2 --memory 512
↪  --swap 512

index=1
# Setting up additional IPs
for ip in $ADDITIONAL_IPS; do
    echo "Adding: $ip"
    pct set $CT_ID -net$index
    ↪  name=eth$index,bridge=vmbr0,firewall=1,gw=$GW,ip=$ip/24,
    ↪  type=veth,tag=13
    index=$((index+1))
```

```
done

# Adding to SDN network
pct set $CT_ID -net$index
↪  name=eth$index,bridge=vnetsdn,firewall=1,gw=192.168.2.1,ip=$SDN_IP/24,
↪  type=veth
```

C.2 Database and Parallelisation of LXC operations

Figure C.2: Script to extract data from database and run LXCs operations in parallel

```
import sys
import sqlite3
import subprocess
import concurrent.futures

def get_all_devices_in_VLAN(db_path:str=None):
    try:

        if db_path is None:
            db_path = "networkDevices.db"

        conn = sqlite3.connect(db_path)
        cursor = conn.cursor()

        sql_cmd = f"SELECT ip_address, operating_system, distribution
        ↪  FROM devices WHERE vlan = {vlan};"
        cursor.execute(sql_cmd)

        rows = cursor.fetchall()

        # Close the cursor and connection
        cursor.close()
        conn.close()

        dict_rows = []
        for row in rows:
            ip = row[0]
            os = row[1]
            distro = row[2]

            # Those not supported or not identified, set to Alpine
            ↪  Linux
            if os != "Linux" and os != "Windows" or distro == "":
                os = "Linux"
```

```
33                  distro = "Alpine"
34
35              # Only Windows 10 is available
36              elif os == "Windows" and distro != "10":
37                  distro = "10"
38
39              dict_row = {
40              'ip': ip,
41              'os': os,
42              'distro': distro
43              }
44              dict_rows.append(dict_row)
45
46          return dict_rows
47
48      except Exception as e:
49          print(f"Error: {e}")
50          return []
51
52
53  def main():
54      if vlan == 0:
55          print("VLAN is required")
56          sys.exit(1)
57
58      # Call the function to get all devices in VLAN
59      devices = get_all_devices_in_VLAN(db_path)
60
61      linux_dict = {}
62      windows_dict = {}
63      other_dict = {}
64
65      # Classify devices by distro
66      for device in devices:
67          os = device.get('os')
68          distro = device.get('distro')
69
70          if os == "Linux":
71              if distro == "Ubuntu":
72                  if 'Ubuntu' not in linux_dict:
73                      linux_dict['Ubuntu'] = []
74                  linux_dict['Ubuntu'].append(device.get('ip'))
75              elif distro == "Debian":
76                  if 'Debian' not in linux_dict:
77                      linux_dict['Debian'] = []
78                  linux_dict['Debian'].append(device.get('ip'))
79              elif distro == "Fedora":
80                  if 'Fedora' not in linux_dict:
```

```
81                      linux_dict['Fedora'] = []
82                  linux_dict['Fedora'].append(device.get('ip'))
83              elif distro == "Alpine":
84                  if 'Alpine' not in linux_dict:
85                      linux_dict['Alpine'] = []
86                  linux_dict['Alpine'].append(device.get('ip'))
87              else:
88                  if 'Other' not in linux_dict:
89                      linux_dict['Other'] = []
90                  linux_dict['Other'].append(device.get('ip'))
91          elif os == "Windows":
92              if distro == "10":
93                  if 'Windows 10' not in windows_dict:
94                      windows_dict['Windows 10'] = []
95                  windows_dict['Windows 10'].append(device.get('ip'))
96              else:
97                  if 'Other' not in windows_dict:
98                      linux_dict['Other'] = []
99                  windows_dict['Other'].append(device.get('ip'))
100         else:
101             if 'Other' not in other_dict:
102                 other_dict['Other']
103             other_dict['Other'].append(device.get('ip'))
104
105     # Set the CT IDs of the Templates for each distro
106     distro_template = {
107         "Alpine": "1001",
108         "Ubuntu": "1002",
109         "Debian": "1003",
110         "Fedora": "1004"
111     }
112
113     ct_id = 2000
114     ct_list = []
115     gateway = "192.168.1.1"
116     sdn_end = 11
117     sdn_ip = "192.168.2."
118
119     # Create the LXC containers
120     def create_container(distro, ct_id, distro_template, gateway,
        ↪  sdn_ip, sdn_end, ips):
121         ips = " ".join(ips)
122         cmd = f"./launch_lxcs.sh {distro} {ct_id}
            ↪  {distro_template[distro]} {gateway} {sdn_ip}{sdn_end}
            ↪  {ips}"
123         print(cmd)
124         subprocess.run(cmd.split(" "), stdout=subprocess.DEVNULL)
125         ct_list.append(ct_id)
```

```
126
127     # Create the LXC containers in parallel
128     with concurrent.futures.ThreadPoolExecutor() as executor:
129         futures = [executor.submit(create_container, distro, ct_id + i,
            ↪  distro_template, gateway, sdn_ip, sdn_end + i,
            ↪  linux_dict[distro]) for i, distro in enumerate(linux_dict)]
130         concurrent.futures.wait(futures)
131
132     def start_containers(ct):
133         cmd = f"pct start {ct}"
134         subprocess.run(cmd.split(" "), stdout=subprocess.DEVNULL)
135
136     with
        ↪  concurrent.futures.ThreadPoolExecutor(max_workers=len(ct_list))
        ↪  as executor:
137         futures = [executor.submit(start_containers, ct) for ct in
            ↪  ct_list]
138         concurrent.futures.wait(futures)
139
140
141 if __name__ == "__main__":
142     # Getting the vlan and database path
143     try:
144         vlan = sys.argv[1] if len(sys.argv) > 1 else None
145     except Exception as e:
146         print(f"VLAN Error: {e}")
147         sys.exit(1)
148
149     try:
150         db_path = sys.argv[2] if len(sys.argv) > 2 else None
151     except Exception as e:
152         print(f"DB Path Error: {e}")
153         sys.exit(1)
154
155     # Running script
156     main()
```

Appendix D

Environment Cloning Results

D.1 Attacker Not Isolated Nmap Scan

Figure D.1: NMap scan from Attackers perspective while not being isolated, Network further than switch blocked

```
# Nmap 7.94SVN scan initiated Sun Aug 25 05:10:19 2024 as: nmap -F -O -T5 -n --max-retries 5 --host-timeout 15m --scan-delay 0 -oN not_isolated 192.168.1.0/24
Nmap scan report for 192.168.1.43
Host is up (0.0040s latency).
Not shown: 99 filtered tcp ports (no-response)
PORT   STATE SERVICE
80/tcp open  http
MAC Address: 28:EE:52:E4:25:46 (TP-Link Technologies)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: router|printer
Running (JUST GUESSING): HP embedded (91%), Ricoh embedded (91%), Brother embedded (89%), Adtran embedded (85%)
OS CPE: cpe:/h:hp:procurve_7102dl cpe:/h:ricoh:aficio_sp_c240sf cpe:/h:brother:mfc-7820n cpe:/h:adtran:netvanta_3430 cpe:/h:adtran:netvanta_1224r
Aggressive OS guesses: HP ProCurve Secure Router 7102dl (91%), Ricoh Aficio SP C240SF printer (91%), Brother MFC-7820N printer (89%), Adtran NetVanta 1224R or 3430 router (85%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop

Nmap scan report for 192.168.1.51
Host is up (0.0024s latency).
Not shown: 93 closed tcp ports (reset)
PORT     STATE SERVICE
```

```
80/tcp   open  http
139/tcp  open  netbios-ssn
443/tcp  open  https
445/tcp  open  microsoft-ds
631/tcp  open  ipp
8080/tcp open  http-proxy
9100/tcp open  jetdirect
MAC Address: 98:4B:E1:85:4B:8A (Hewlett Packard)
Device type: general purpose|printer|VoIP adapter
Running: Wind River VxWorks, HP embedded, Vocality embedded
OS CPE: cpe:/o:windriver:vxworks
OS details: VxWorks: HP printer or Vocality BASICS Four Wire VoIP
↪  gateway
Network Distance: 1 hop

Nmap scan report for 192.168.1.104
Host is up (0.00060s latency).
Not shown: 99 closed tcp ports (reset)
PORT   STATE SERVICE
80/tcp open  http
MAC Address: A0:4A:5E:E2:36:A1 (Microsoft)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.8
Network Distance: 1 hop

Nmap scan report for 192.168.1.220
Host is up (0.00011s latency).
Not shown: 98 closed tcp ports (reset)
PORT     STATE    SERVICE
22/tcp   open     ssh
8000/tcp filtered http-alt
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6.32
OS details: Linux 2.6.32
Network Distance: 0 hops

OS detection performed. Please report any incorrect results at
↪  https://nmap.org/submit/ .
# Nmap done at Sun Aug 25 05:10:28 2024 -- 256 IP addresses (5 hosts
↪  up) scanned in 9.84 seconds

```

D.2 Attacker Isolated Nmap Scan

Figure D.2: NMap scan from Attackers perspective while being isolated, Network further than switch blocked

```
# Nmap 7.94SVN scan initiated Sun Aug 25 04:55:26 2024 as: nmap -F -O -T5 -n --max-retries 5 --host-timeout 15m --scan-delay 0 -oN isolated 192.168.1.0/24
Nmap scan report for 192.168.1.43
Host is up (0.0039s latency).
Not shown: 99 filtered tcp ports (no-response)
PORT   STATE SERVICE
80/tcp open  http
MAC Address: 28:EE:52:E4:25:46 (TP-Link Technologies)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: router|printer
Running (JUST GUESSING): HP embedded (91%), Ricoh embedded (91%), Brother embedded (89%), Adtran embedded (85%)
OS CPE: cpe:/h:hp:procurve_7102dl cpe:/h:ricoh:aficio_sp_c240sf cpe:/h:brother:mfc-7820n cpe:/h:adtran:netvanta_3430 cpe:/h:adtran:netvanta_1224r
Aggressive OS guesses: HP ProCurve Secure Router 7102dl (91%), Ricoh Aficio SP C240SF printer (91%), Brother MFC-7820N printer (89%), Adtran NetVanta 1224R or 3430 router (85%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop

Nmap scan report for 192.168.1.51
Host is up (0.00022s latency).
All 100 scanned ports on 192.168.1.51 are in ignored states.
Not shown: 100 closed tcp ports (reset)
MAC Address: BC:24:11:60:3D:86 (Unknown)
Too many fingerprints match this host to give specific OS details
Network Distance: 1 hop

Nmap scan report for 192.168.1.104
Host is up (0.00044s latency).
All 100 scanned ports on 192.168.1.104 are in ignored states.
Not shown: 100 filtered tcp ports (no-response)
MAC Address: BC:24:11:60:3D:86 (Unknown)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.8
Network Distance: 1 hop
```

```
34
35 Nmap scan report for 192.168.1.220
36 Host is up (0.00011s latency).
37 Not shown: 98 closed tcp ports (reset)
38 PORT     STATE    SERVICE
39 22/tcp   open     ssh
40 8000/tcp filtered http-alt
41 Device type: general purpose
42 Running: Linux 2.6.X
43 OS CPE: cpe:/o:linux:linux_kernel:2.6.32
44 OS details: Linux 2.6.32
45 Network Distance: 0 hops
46
47 OS detection performed. Please report any incorrect results at
   ↪  https://nmap.org/submit/ .
48 # Nmap done at Sun Aug 25 04:55:36 2024 -- 256 IP addresses (5 hosts
   ↪  up) scanned in 10.40 seconds
49
```

Bibliography

[1] Makhdoomi A, Jan N, Palak, Goel N. Conventional and next generation firewalls in network security and its applications. In: 2022 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS); 2022. p. 964-9. Available from: https://ieeexplore.ieee.org/document/10037674. pages 1

[2] Yoachimik O, Pacheco J. DDoS threat report for 2023 Q4; 2024. Available from: https://blog.cloudflare.com/ddos-threat-report-2023-q4. pages 1

[3] Global Threat Report 2024. Crowdstrike; 2024. Available from: https://go.crowdstrike.com/rs/281-OBQ-266/images/GlobalThreatReport2024.pdf. pages 1, 2

[4] Sharif MHU, Mohammed MA, Sharif MHU, Mohammed MA. A literature review of financial losses statistics for cyber security and future trend. World Journal of Advanced Research and Reviews. 2022;15(1):138-56. Last Modified: 2022-07-09T23:56+05:30 Number: 1 Publisher: World Journal of Advanced Research and Reviews. Available from: https://doi.org/10.30574/wjarr.2022.15.1.0573. pages 2

[5] Quarantine Devices Using Host Information; 2024. Available from: https://docs.paloaltonetworks.com/globalprotect/10-1/globalprotect-admin/host-information/quarantine-devices-using-host-information. pages 2, 3, 10, 72

[6] CrowdStrike Falcon® Adversary Intelligence; 2024. Available from: https://www.crowdstrike.com/products/threat-intelligence/adversary-intelligence/. pages 2

[7] Heino J, Hakkala A, Virtanen S. Study of methods for endpoint aware inspection in a next generation firewall. Cybersecurity. 2022 Sep;5(1):25. Available from: https://doi.org/10.1186/s42400-022-00127-8. pages 3, 7, 8, 9

[8] Liang J, Kim Y. Evolution of Firewalls - Toward Securer Network Using Next Generation Firewall. In: 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC); 2022. p. 0752-9. Available from: https://ieeexplore.ieee.org/document/9720435. pages 3, 6

[9] Rajkumar B, Arunakranthi G. Evolution for a secured path using NexGen firewalls. In: 2022 OPJU International Technology Conference on Emerging Technologies for Sustainable Development (OTCON); 2023. p. 1-6. Available from: https://ieeexplore.ieee.org/abstract/document/10113935. pages 3

[10] Alani MM. OSI Model. In: Alani MM, editor. Guide to OSI and TCP/IP Models. Cham: Springer International Publishing; 2014. p. 5-17. Available from: https://doi.org/10.1007/978-3-319-05152-9_2. pages 3

[11] Ingham K. A History and Survey of Network Firewalls. 2002. pages 3

[12] W C. The Design of a Secure Internet Gateway. Proc of Summer Usenix Conf, Anaheim. 1990. Available from: https://cir.nii.ac.jp/crid/1573105973876825984. pages 3

[13] Abie H, Box PO. An Overview of Firewall Technologies. Telektronikk. 2000;96.3:47-52. pages 4, 6

[14] Wool A. Packet filtering and stateful firewalls. Handbook of Information Security; 2006. Available from: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=1e3cdf72351748e00e45d8fcbd157e1c66cd8fca. pages 4, 5

[15] Ehrenkranz T, Li J. On the state of IP spoofing defense. ACM Transactions on Internet Technology. 2009 May;9(2):1-29. Available from: https://dl.acm.org/doi/10.1145/1516539.1516541. pages 4

[16] Gouda MG, Liu AX. A model of stateful firewalls and its properties. In: 2005 International Conference on Dependable Systems and Networks (DSN'05); 2005. p. 128-37. ISSN: 2158-3927. Available from: https://ieeexplore.ieee.org/abstract/document/1467787. pages 4

[17] Trabelsi Z, Zeidan S, Shuaib K, Salah K. Improved Session Table Architecture for Denial of Stateful Firewall Attacks. IEEE Access. 2018;6:35528-43. Available from: https://ieeexplore.ieee.org/document/8395423/. pages 5

[18] Yavwa Y. The Firewall Technology; 2000. p. 18. Available from: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=e49a3c81e95ee3dda875fcd42a07c889af9aa42f#page=5.75. pages 5

[19] Kaplesh P, Goel A. Firewalls: A study on Techniques, Security and Threats;(2249). pages 6

[20] Tekerek A, Bay O. Design And Implementation Of An Artificial Intelligence-Based Web Application Firewall Model. Neural Network World. 2019 Jan;29:189-206. pages 6

[21] Appelt D, Panichella A, Briand L. Automatically Repairing Web Application Firewalls Based on Successful SQL Injection Attacks. In: 2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE). Toulouse: IEEE; 2017. p. 339-50. Available from: http://ieeexplore.ieee.org/document/8109099/. pages 6

[22] J HK, J GP. Securing Web Application using Web Application Firewall (WAF) and Machine Learning. In: 2023 First International Conference on Advances in Electrical, Electronics and Computational Intelligence (ICAEECI). Tiruchengode, India: IEEE; 2023. p. 1-8. Available from: https://ieeexplore.ieee.org/document/10370872/. pages 6

[23] Snyder J. Evaluating Unified Threat Performance Management Products for Requirements Enterprise Networks. 2006 Aug. pages 7

[24] Siddiqui A, Rimal BP, Reisslein M, Wang Y. Survey on Unified Threat Management (UTM) Systems for Home Networks. IEEE Communications Surveys & Tutorials. 2024:1-1. Conference Name: IEEE Communications Surveys & Tutorials. Available from: https://ieeexplore.ieee.org/abstract/document/10480701. pages 7

[25] Raza MS, Kazmi SBA, Ali R, Naqvi MM, Fiaz H, Akram A. High Performance DPI Engine Design for Network Traffic Classification, Metadata Extraction and Data Visualization. In: 2024 5th International Conference on Advancements in Computational Sciences (ICACS); 2024. p. 1-6. Available from: https://ieeexplore.ieee.org/document/10473274. pages 8

[26] Nyasore ON, Zavarsky P, Swar B, Naiyeju R, Dabra S. Deep Packet Inspection in Industrial Automation Control System to Mitigate Attacks Exploiting Modbus_TCP Vulnerabilities. In: 2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS); 2020. p. 241-5. Available from: https://ieeexplore.ieee.org/document/9123061. pages 8

[27] El-Maghraby RT, Abd Elazim NM, Bahaa-Eldin AM. A survey on deep packet inspection. In: 2017 12th International Conference on Computer Engineering and Systems (ICCES); 2017. p. 188-97. Available from: https://ieeexplore.ieee.org/abstract/document/8275301. pages 8

[28] Zhu W, Li P, Luo B, Xu H, Zhang Y. Research and Implementation of High Performance Traffic Processing Based on Intel DPDK. In: 2018 9th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP); 2018. p. 62-8. Available from: https://ieeexplore.ieee.org/document/8701793. pages 8

[29] Heino J, Jalio C, Hakkala A, Virtanen S. A Method for Endpoint Aware Inspection in a Network Security Solution. IEEE Access. 2022 Apr;10:44517-30.

Conference Name: IEEE Access. Available from: https://ieeexplore.ieee.org/abstract/document/9762961. pages 8

[30] Cisco NGFW webinar. Germany; 2016. Available from: https://www.cisco.com/c/dam/m/en_uk/events/2016/securityexperts/pdf/ngfw-webinar_german_oct_25th.pdf. pages 9

[31] Forcepoint. Forcepoint Advanced Malware Detection and how it works [concept];. Archive Location: Product Guide. Available from: https://help.forcepoint.com/ngfw/en-us/7.0.0/GUID-EECA15DA-9B8A-4C2F-9C42-D53516ADC19E.html. pages 9

[32] Hooper E. An intelligent detection and response strategy to false positives and network attacks. In: Fourth IEEE International Workshop on Information Assurance (IWIA'06); 2006. p. 20 pp.-31. Available from: https://ieeexplore.ieee.org/abstract/document/1609997. pages 10

[33] Hooper E. An Intelligent Intrusion Detection and Response System Using Network Quarantine Channels: Adaptive Policies and Alert Filters. In: 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology Workshops; 2006. p. 45-8. Available from: https://ieeexplore.ieee.org/document/4053201. pages 10

[34] Hooper E. An Intelligent Intrusion Detection and Response System Using Network Quarantine Channels: Firewalls and Packet Filters. In: 2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07); 2007. p. 1193-8. Available from: https://ieeexplore.ieee.org/document/4197441. pages 10

[35] Software-Defined Access;. Available from: https://www.cisco.com/c/en_uk/solutions/enterprise-networks/software-defined-access/index.html. pages 10

[36] Stawowski M. The Principles of Network Security Design. The Global Voice of Information Security. 2007 Oct. pages 10

[37] Castiglione A, De Santis A, Fiore U, Palmieri F. An Enhanced Firewall Scheme for Dynamic and Adaptive Containment of Emerging Security Threats. In: 2010 International Conference on Broadband, Wireless Computing, Communication and Applications; 2010. p. 475-81. Available from: https://ieeexplore.ieee.org/document/5633666. pages 10

[38] Rovniagin D, Wool A. The Geometric Efficient Matching Algorithm for Firewalls. IEEE Transactions on Dependable and Secure Computing. 2011 Jan;8(1):147-59. Conference Name: IEEE Transactions on Dependable and Secure Computing. Available from: https://ieeexplore.ieee.org/document/5184850. pages 10

[39] Wang X, Zhao H, Guan M, Guo C, Wang J. Research and implementation of VLAN based on service. In: GLOBECOM '03. IEEE Global Telecommunications Conference (IEEE Cat. No.03CH37489). vol. 5; 2003. p. 2932-6 vol.5. Available from: https://ieeexplore.ieee.org/abstract/document/1258771. pages 10

[40] Jiajia Liu, Wuwen Lai. Security analysis of VLAN-based Virtual Desktop Infrastructure. In: 2010 International Conference on Educational and Network Technology. Qinhuangdao, China: IEEE; 2010. p. 301-4. Available from: http://ieeexplore.ieee.org/document/5532167/. pages 10

[41] Hameed A, Mian AN. Finding efficient VLAN topology for better broadcast containment. In: 2012 Third International Conference on The Network of the Future (NOF); 2012. p. 1-6. Available from: https://ieeexplore.ieee.org/document/6464001. pages 11

[42] Bosa PA, Greenwald J, Buia C, Pantelis TS, Ball S. (54) SYSTEMIS AND METHODS FOR SOLATING FAULTS IN COMPUTER NETWORKS. pages 11

[43] Kindervag J. Build Security Into Your Network's DNA: The Zero Trust Network Architecture. 2010. pages 11

[44] Pam. VLAN Hopping: How to Mitigate an Attack; 2024. Available from: https://cybersecurity.att.com/blogs/security-essentials/vlan-hopping-and-mitigation. pages 11

[45] What is IPsec? | How IPsec VPNs work;. Available from: https://www.cloudflare.com/learning/network-layer/what-is-ipsec/. pages 11, 30

[46] Mhaskar N, Alabbad M, Khedri R. A Formal Approach to Network Segmentation. Computers & Security. 2021 Apr;103:102162. Available from: https://linkinghub.elsevier.com/retrieve/pii/S0167404820304351. pages 11, 15

[47] Sheikh N, Pawar M, Lawrence V. Zero trust using Network Micro Segmentation. In: IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). Vancouver, BC, Canada: IEEE; 2021. p. 1-6. Available from: https://ieeexplore.ieee.org/document/9484645/. pages 11

[48] Kindervag J. You Want Network Segmentation, But You Need Zero Trust; 2019. Available from: https://www.paloaltonetworks.com/blog/2019/01/you-want-network-segmentation-but-you-need-zero-trust/. pages 11

[49] Otsuka T, Koibuchi M, Kudoh T, Amano H. Switch-tagged VLAN Routing Methodology for PC Clusters with Ethernet. In: 2006 International Conference on Parallel Processing (ICPP'06); 2006. p. 479-86. ISSN: 2332-5690. Available from: https://ieeexplore.ieee.org/document/1690652/?arnumber=1690652. pages 12

[50] Mehdizadeh A, Suinggi K, Mohammadpoor M, Harun H. Virtual Local Area Network (VLAN): Segmentation and Security. 2017. pages 12

[51] Kim K, Lee M. SNMP-Based Detection of VLAN Hopping Attack Risk. In: Kim KJ, Baek N, editors. Information Science and Applications 2018. Singapore: Springer; 2019. p. 267-72. pages 12, 55

[52] VLAN Best Practices and Security Tips for Cisco Business Routers;. Available from: https://www.cisco.com/c/en/us/support/docs/smb/routers/cisco-rv-series-small-business-routers/1778-tz-VLAN-Best-Practices-and-Security-Tips-for-Cisco-Business-Routers.html. pages 13

[53] Chuanqing C, Li W. A Flexible Method to Support VLAN Stacking in Ethernet Network. In: 2009 Second Pacific-Asia Conference on Web Mining and Web-based Application; 2009. p. 134-7. Available from: https://ieeexplore.ieee.org/abstract/document/5232485. pages 13

[54] Karpesky: Double tagging attack;. Available from: https://encyclopedia.kaspersky.com/glossary/double-tagging-attack/. pages 13

[55] Muhammad T. Overlay Network Technologies in SDN: Evaluating Performance and Scalability of VXLAN and GENEVE. 2021;5(1). pages 13, 14

[56] Dr A SHAJI GEORGE, A S HOVAN GEORGE. A Brief Overview of VXLAN EVPN. 2021 Jul. Publisher: Zenodo. Available from: https://zenodo.org/record/7027361. pages 13

[57] Guido Pineda R. Security assessment on a VXLAN-based network; 2014. Available from: https://www.os3.nl/_media/2013-2014/courses/rp1/p57_report.pdf. pages 14

[58] Sezer S, Scott-Hayward S, Chouhan PK, Fraser B, Lake D, Finnegan J, et al. Are we ready for SDN? Implementation challenges for software-defined networks. IEEE Communications Magazine. 2013 Jul;51(7):36-43. Conference Name: IEEE Communications Magazine. Available from: https://ieeexplore.ieee.org/abstract/document/6553676. pages 14

[59] Bringhenti D, Marchetto G, Sisto R, Valenza F, Yusupov J. Automated optimal firewall orchestration and configuration in virtualized networks. In: NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium; 2020. p. 1-7. ISSN: 2374-9709. Available from: https://ieeexplore.ieee.org/document/9110402. pages 14

[60] Sloane T. Software-Defined Networking: The New Norm for Networks; 2013. Available from: https://opennetworking.org/sdn-resources/whitepapers/software-defined-networking-the-new-norm-for-networks/. pages 14

[61] Agarwal S, Kodialam M, Lakshman TV. Traffic engineering in software defined networks. In: 2013 Proceedings IEEE INFOCOM; 2013. p. 2211-9. ISSN: 0743-166X. Available from: https://ieeexplore.ieee.org/document/6567024. pages 14

[62] Kim S, Yoon S, Narantuya J, Lim H. Secure Collecting, Optimizing, and Deploying of Firewall Rules in Software-Defined Networks. IEEE Access. 2020;8:15166-77. Conference Name: IEEE Access. Available from: https://ieeexplore.ieee.org/document/8962096. pages 14, 15

[63] Zerkane S, Espes D, Le Parc P, Cuppens F. Software Defined Networking Reactive Stateful Firewall. In: Hoepman JH, Katzenbeisser S, editors. ICT Systems Security and Privacy Protection. Cham: Springer International Publishing; 2016. p. 119-32. pages 14

[64] Hu H, Han W, Ahn GJ, Zhao Z. FLOWGUARD: Building robust firewalls for software-defined networks; 2014. . pages 14

[65] Rietz R, Cwalinski R, König H, Brinner A. An SDN-Based Approach to Ward Off LAN Attacks. Journal of Computer Networks and Communications. 2018 Nov;2018:e4127487. Publisher: Hindawi. Available from: https://www.hindawi.com/journals/jcnc/2018/4127487/. pages 15

[66] Wang C. Construction and Deployment of a Distributed Firewall-based Computer Security Defense Network. 2023. Available from: http://ijns.jalaxy.com.tw/contents/ijns-v25-n1/ijns-2023-v25-n1-p89-94.pdf. pages 15

[67] Tudosi AD, Graur A, Balan DG, Dan Potorac A, Tarabuta R. Distributed Firewall Traffic Filtering and Intrusion Detection Using Snort on pfSense Firewalls with Random Forest Classification. In: 2023 46th International Conference on Telecommunications and Signal Processing (TSP); 2023. p. 101-4. ISSN: 2768-3311. Available from: https://ieeexplore.ieee.org/abstract/document/10197784. pages 15

[68] Ayodele B, Buttigieg V. SDN as a defence mechanism: a comprehensive survey. International Journal of Information Security. 2024 Feb;23(1):141-85. Available from: https://doi.org/10.1007/s10207-023-00764-1. pages 15, 68

[69] Cherednichenko O, Sharonova N, Pliekhova G, Babkova N. Intelligent Methods of Secure Routing in Software Defined Networks. 2024. pages 15

[70] Alabbad M, Khedri R. Configuration and Governance of Dynamic Secure SDN. Procedia Computer Science. 2021;184:131-9. Available from: https://linkinghub.elsevier.com/retrieve/pii/S1877050921006463. pages 15

[71] Dong L, Chen L, Zhang Y, He B, Zhou J, Wang W, et al. Dynamic Policy Deployment in SDN Switch Based on Monitoring and Analysis of User Behaviors. In: 2018 27th International Conference on Computer Communication and Networks (ICCCN); 2018. p. 1-8. ISSN: 1095-2055. Available from: https://ieeexplore.ieee.org/document/8487412. pages 16

[72] Proxmox VE Firewall; 2024. Available from: https://pve.proxmox.com/pve-docs/chapter-pve-firewall.html. pages 16, 45

[73] Proxmox High Availability [PDF] [5o348l0spnr0];. Available from: https://vdoc.pub/documents/proxmox-high-availability-5o348l0spnr0. pages 16

[74] Ljubojević M, Bajić A, Mijić D. Implementation of High-Availability Server Cluster by Using Fencing Concept. In: 2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH); 2019. p. 1-5. Available from: https://ieeexplore.ieee.org/document/8717752. pages 16

[75] Đorđević B, Timčenko V, Kraljević N, Jovičić N. Performance comparison of KVM and Proxmox Type-1 Hypervisors. In: 2022 30th Telecommunications Forum (FOR); 2022. p. 1-4. Available from: https://ieeexplore.ieee.org/document/9983666. pages 17

[76] Flauzac O, Mauhourat F, Nolot F. A review of native container security for running applications. Procedia Computer Science. 2020 Jan;175:157-64. Available from: https://www.sciencedirect.com/science/article/pii/S187705092031704X. pages 17

[77] Mehrab AKMF, Nikolaev R, Ravindran B. Kite: lightweight critical service domains. In: Proceedings of the Seventeenth European Conference on Computer Systems. EuroSys '22. New York, NY, USA: Association for Computing Machinery; 2022. p. 384-401. Available from: https://dl.acm.org/doi/10.1145/3492321.3519586. pages 17

[78] Kuenzer S, Bădoiu VA, Lefeuvre H, Santhanam S, Jung A, Gain G, et al. Unikraft: fast, specialized unikernels the easy way. In: Proceedings of the Sixteenth European Conference on Computer Systems. EuroSys '21. New York, NY, USA: Association for Computing Machinery; 2021. p. 376-94. Available from: https://dl.acm.org/doi/10.1145/3447786.3456248. pages 17

[79] Manco F, Lupu C, Schmidt F, Mendes J, Kuenzer S, Sati S, et al. My VM is Lighter (and Safer) than your Container. In: Proceedings of the 26th Symposium on Operating Systems Principles. SOSP '17. New York, NY, USA: Association for Computing Machinery; 2017. p. 218-33. Available from: https://dl.acm.org/doi/10.1145/3132747.3132763. pages 17

[80] Hertz J. Abusing Privileged and Unprivileged Linux Containers (NCC group); 2016. Available from: http://asm.rajiska.fr/misc/container_security.pdf. pages 18

[81] Randazzo A, Tinnirello I. Kata Containers: An Emerging Architecture for Enabling MEC Services in Fast and Secure Way. In: 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS); 2019. p. 209-14. Available from: https://ieeexplore.ieee.org/document/8939164/similar#similar. pages 18

[82] Babu SA, Hareesh MJ, Martin JP, Cherian S, Sastri Y. System Performance Evaluation of Para Virtualization, Container Virtualization, and Full Virtualization Using Xen, OpenVZ, and XenServer. In: 2014 Fourth International Conference on Advances in Computing and Communications; 2014. p. 247-50. Available from: https://ieeexplore.ieee.org/document/6906035. pages 18

[83] Kovács Comparison of different Linux containers. In: 2017 40th International Conference on Telecommunications and Signal Processing (TSP); 2017. p. 47-51. Available from: https://ieeexplore.ieee.org/document/8075934. pages 18

[84] Lesueur F, Noûs C. MI-LXC: A Small-Scale Internet-Like Environment for Network Security Teaching. In: Proceedings of the 16th International Conference on Availability, Reliability and Security. ARES '21. New York, NY, USA: Association for Computing Machinery; 2021. p. 1-6. Available from: https://dl.acm.org/doi/10.1145/3465481.3469181. pages 18

[85] Karagiannis S, Ntantogian C, Magkos E, Ribeiro LL, Campos L. PocketCTF: A Fully Featured Approach for Hosting Portable Attack and Defense Cybersecurity Exercises. Information. 2021 Aug;12(8):318. Number: 8 Publisher: Multidisciplinary Digital Publishing Institute. Available from: https://www.mdpi.com/2078-2489/12/8/318. pages 18

[86] Kedrowitsch A, Yao DD, Wang G, Cameron K. A First Look: Using Linux Containers for Deceptive Honeypots. In: Proceedings of the 2017 Workshop on Automated Decision Making for Active Cyber Defense. SafeConfig '17. New York, NY, USA: Association for Computing Machinery; 2017. p. 15-22. Available from: https://dl.acm.org/doi/10.1145/3140368.3140371. pages 18

[87] Sun J, Liu S, Sun K. A Scalable High Fidelity Decoy Framework against Sophisticated Cyber Attacks. In: Proceedings of the 6th ACM Workshop on Moving Target Defense. MTD'19. New York, NY, USA: Association for Computing Machinery; 2019. p. 37-46. Available from: https://dl.acm.org/doi/10.1145/3338468.3356826. pages 18

[88] Serem EK. DECEPTIVE DECOYS: COMBINING BELIEVABLE USER AND NETWORK ACTIVITIES AND DECEPTIVE NETWORK SETUP IN ENHANCING EFFECTIVENESS. 2021 Jun. pages 18

[89] Silva S, Sousa PR, Resende JS, Antunes L. Threat Detection and Mitigation with Honeypots: A Modular Approach for IoT. In: Katsikas S, Furnell S,

editors. Trust, Privacy and Security in Digital Business. Cham: Springer International Publishing; 2022. p. 66-80. pages 18

[90] Memari N, Hashim SJ, Samsudin K. Design of a virtual hybrid honeynet based on LXC virtualisation for enhanced network security. 2014 Jan;7:120-9. pages 18, 68

[91] Memari N, Hashim SJB, Samsudin KB. Towards virtual honeynet based on LXC virtualization. In: 2014 IEEE REGION 10 SYMPOSIUM; 2014. p. 496-501. Available from: https://ieeexplore.ieee.org/abstract/document/6863084/citations?tabFilter=papers#citations. pages 18

[92] Kong T, Wang L, Ma D, Xu Z, Yang Q, Lu Z, et al. Automated Honeynet Deployment Strategy for Active Defense in Container-based Cloud. In: 2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS); 2020. p. 483-90. Available from: https://ieeexplore.ieee.org/document/9407968. pages 18, 68, 69

[93] Grieves M. Digital Twin: Manufacturing Excellence through Virtual Factory Replication. 2015 Mar. pages 18

[94] Zhou C, Yang H, Duan X, Lopez D, Pastor A, Wu Q, et al. Network Digital Twin: Concepts and Reference Architecture. Internet Engineering Task Force; 2024. draft-irtf-nmrg-network-digital-twin-arch-06. Num Pages: 29. Available from: https://datatracker.ietf.org/doc/draft-irtf-nmrg-network-digital-twin-arch. pages 18

[95] Digital twin network – Requirements and architecture (ITU); 2022. Available from: https://www.itu.int/rec/T-REC-Y.3090-202202-I/en. pages 19

[96] Zhang Y. Digital Twin: Architectures, Networks, and Applications. vol. 16 of Simula SpringerBriefs on Computing. Cham: Springer Nature Switzerland; 2024. Available from: https://link.springer.com/10.1007/978-3-031-51819-5. pages 19

[97] Peuster M, Karl H, van Rossem S. MeDICINE: Rapid prototyping of production-ready network services in multi-PoP environments. In: 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN); 2016. p. 148-53. Available from: https://ieeexplore.ieee.org/document/7919490. pages 19

[98] Mininet: An Instant Virtual Network on Your Laptop (or Other PC) - Mininet;. Available from: http://mininet.org/. pages 19

[99] Peuster M. Containernet; 2016. Available from: https://containernet.github.io/. pages 19, 68

[100] robinharwood. Set up Hyper-V Replica; 2024. Available from: https://learn.microsoft.com/en-us/windows-server/virtualization/hyper-v/manage/set-up-hyper-v-replica. pages 19

[101] Virtualize a Physical Machine; 2019. Available from: https://docs.vmware.com/en/VMware-Workstation-Pro/16.0/com.vmware.ws.using.doc/GUID-87B43C18-06A0-4055-B82D-EA0E81047B45.html. pages 19

[102] Charan PVS, Mukhopadhyay S, Manna S, Rani N, Vaid A, Chunduri H, et al. ADAPT Adaptive Camouflage Based Deception Orchestration For Trapping Advanced Persistent Threats. Digital Threats: Research and Practice. 2024 Mar. Just Accepted. Available from: https://dl.acm.org/doi/10.1145/3651991. pages 19, 20, 24, 68, 69, 72

[103] Kim G. NetClone: Fast, Scalable, and Dynamic Request Cloning for Microsecond-Scale RPCs. In: Proceedings of the ACM SIGCOMM 2023 Conference; 2023. p. 195-207. ArXiv:2307.13285 [cs]. Available from: http://arxiv.org/abs/2307.13285. pages 19

[104] What is Threat Intelligence? (IBM); 2022. Available from: https://www.ibm.com/topics/threat-intelligence. pages 19

[105] Sanjeev K, Janet B, Eswari R. Automated Cyber Threat Intelligence Generation from Honeypot Data. In: Ranganathan G, Chen J, Rocha editors. Inventive Communication and Computational Technologies. Singapore: Springer; 2020. p. 591-8. pages 19

[106] Slatman H. hslatman/awesome-threat-intelligence; 2024. Original-date: 2015-12-21T11:31:04Z. Available from: https://github.com/hslatman/awesome-threat-intelligence. pages 19

[107] Lihet MA, Dadarlat V. How to build a honeypot System in the cloud. In: 2015 14th RoEduNet International Conference - Networking in Education and Research (RoEduNet NER); 2015. p. 190-4. ISSN: 2247-5443. Available from: https://ieeexplore.ieee.org/document/7311992. pages 19

[108] Sehgal R, Majithia N, Singh S, Sharma S, Mukhopadhyay S, Handa A, et al. Honeypot Deployment Experience at IIT Kanpur. In: Shukla SK, Agrawal M, editors. Cyber Security in India: Education, Research and Training. Singapore: Springer; 2020. p. 49-63. Available from: https://doi.org/10.1007/978-981-15-1675-7_6. pages 19

[109] Bartwal U, Mukhopadhyay S, Negi R, Shukla S. Security Orchestration, Automation, and Response Engine for Deployment of Behavioural Honeypots. In: 2022 IEEE Conference on Dependable and Secure Computing (DSC); 2022. p. 1-8. Available from: https://ieeexplore.ieee.org/document/9888808. pages 20

[110] Alani MM. HoneyTwin: Securing smart cities with machine learning-enabled SDN edge and cloud-based honeypots. Journal of Parallel and Distributed Computing. 2024 Jun;188:104866. Available from: https://www.sciencedirect.com/science/article/pii/S0743731524000303. pages 20

[111] 5-Port Gigabit Easy Smart Switch;. Available from: https://www.tp-link.com/uk/business-networking/easy-smart-switch/tl-sg105e/. pages 26

[112] Intel® Core™ i7-6700 Processor (8M Cache, up to 4.00 GHz) - Product Specifications;. Available from: https://www.intel.com/content/www/us/en/products/sku/88196/intel-core-i76700-processor-8m-cache-up-to-4-00-ghz/specifications.html. pages 26

[113] DOMINATOR® PLATINUM 16GB (2 x 8GB) DDR4 DRAM 3200MHz C16 Memory Kit;. Available from: https://www.corsair.com/us/en/p/memory/cmd16gx4m2b3200c16/dominatora-platinum-16gb-2-x-8gb-ddr4-dram-3200mhz-c16-memory-kit-cmd16gx4m2b pages 26

[114] Asrock. ASRock Z270 Taichi;. Available from: https://www.asrock.com/mb/intel/Z270Taichi/. pages 26

[115] HP Pavilion 500-220ex Desktop PC Product Specifications | HP® Support;. Available from: https://support.hp.com/us-en/document/c04084441. pages 26

[116] Intel® Core™ i5-4440 Processor (6M Cache, up to 3.30 GHz) - Product Specifications;. Available from: https://www.intel.com/content/www/us/en/products/sku/75038/intel-core-i54440-processor-6m-cache-up-to-3-30-ghz/specifications.html. pages 26

[117] Donenfeld JA. WireGuard: fast, modern, secure VPN tunnel;. Available from: https://www.wireguard.com/. pages 29

[118] Business VPN For Secure Networking;. Available from: https://openvpn.net/. pages 30

[119] SoftEther VPN Project - SoftEther VPN Project;. Available from: https://www.softether.org/. pages 30

[120] ZeroTier | Global Networking Solution for IoT, SD-WAN, and VPN;. Available from: https://www.zerotier.com/. pages 31

[121] Free Dynamic DNS - Managed DNS - Managed Email - Domain Registration - No-IP;. Available from: https://www.noip.com/. pages 31

[122] Dynu;. Available from: http://www.dynu.com. pages 31

[123] FreeDNS - Free DNS - Dynamic DNS - Static DNS subdomain and domain hosting;. Available from: https://freedns.afraid.org/. pages 31

[124] Manawyrm. Manawyrm/switchconf; 2023. Original-date: 2021-10-10T21:23:41Z. Available from: https://github.com/Manawyrm/switchconf. pages 36

[125] Nmap, the Network Mapper, Free Security Scanner;. Available from: https://nmap.org/. pages 38

[126] p0f | Kali Linux Tools;. Available from: https://www.kali.org/tools/p0f/. pages 39

[127] Arkin O, Yarochkin F, Kydyraliev M. Xprobe2; 2024. Original-date: 2020-02-24T21:58:26Z. Available from: https://github.com/binarytrails/xprobe2. pages 39

[128] OpenVAS - Open Vulnerability Assessment Scanner;. Available from: https://www.openvas.org/. pages 39

[129] Song J, Cho C, Won Y. Analysis of operating system identification via fingerprinting and machine learning. Computers & Electrical Engineering. 2019 Sep;78:1-10. Available from: https://www.sciencedirect.com/science/article/pii/S0045790618324765. pages 39

[130] pct - tool to manage LXC in Proxmox VE; 2024. Available from: https://pve.proxmox.com/pve-docs/pct.1.html. pages 43

[131] Proxmox VE API Documentation;. Available from: https://pve.proxmox.com/pve-docs/api-viewer/index.html. pages 43

[132] Netsh commands for Interface IP; 2009. Available from: https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-xp/bb490943(v=technet.10). pages 43

[133] MITRE ATT&CK®;. Available from: https://attack.mitre.org/. pages 48

[134] Qubes OS: A reasonably secure operating system;. Available from: https://www.qubes-os.org/. pages 62

[135] Dal JF, Jimenez J. Configure SSH on Routers and Switches;. Available from: https://www.cisco.com/c/en/us/support/docs/security-vpn/secure-shell-ssh/4145-ssh.html. pages 66

[136] Key Points of Managing the Switch via SNMP;. Available from: https://www.tp-link.com/us/configuration-guides/key_points_of_managing_the_switch_via_snmp/?configurationId=20745#_idTextAnchor000. pages 66

[137] Ansible;. Available from: https://www.ansible.com/. pages 66

[138] Beigi-Mohammadi N, Barna C, Shtern M, Khazaei H, Litoiu M. CAAMP: Completely automated DDoS attack mitigation platform in hybrid clouds. In: 2016 12th International Conference on Network and Service Management (CNSM); 2016. p. 136-43. ISSN: 2165-963X. Available from: `https://ieeexplore.ieee.org/document/7818409`. pages 68