UNIVERSITY OF AMSTERDAM

Malware Aquarium: A virtualized infrastructure where malware resides and is being monitored

December 23, 2022

Authors:
Rio Kierkels

Rareș Brătean

Master of Science:
Network & Security Engineering

Abstract

Malware evolves in unison with technology. Malicious software is constantly changing and developing capabilities, with lateral movement being the most prevalent in recent years. Security experts analyze their behaviour using premium, freemium or open-source sandboxes. Current free online sandboxes have restrictions that prevent analysts from seeing the entire infection process. This paper introduces a solution to this problem: the malware aquarium. It is a bounded space for malware to live its life, possibly indefinitely, without escaping its boundaries, while the environment observes its behaviour. The aquarium is made up of a fully accessible flat network with interconnected devices and monitoring systems. This research implements such a concept by applying a methodology that combines known requirements of sandbox technology and secure architecture with an analysis of two malware types: Trojans and Worms. This concept is validated by comparing the malware aquarium's analysis results with reports provided by two free sandboxes. The resulting malware aquarium proof of concept shows more details than the used free sandboxes while paving the way for further development.

1 Introduction

Over the years, malware developed extensively, enhancing its capabilities and behaviour. One of the most prominent capabilities seen in recent years is lateral movement, which can be seen even in current Ransomware strains (Ryuk [1]), Trojans (TrickBot [2]) or Wipers (HermeticWiper [3]). Currently, the most popular method of investigating malware is to perform dynamic analysis within sandbox technologies such as Cuckoo Online[4], Anyrun [5], or other free online alternatives. These sandboxes are largely focused on executing the malware on a single system in a limited time window, which may result in parts of the deployment and capabilities of the malware being missed. Furthermore, the single-machine sandbox might not observe the malware behaviour towards the wider infrastructure.

This research takes the initial steps towards tackling these problems by creating a virtualized infrastructure. In this environment, one or multiple malware can be deployed and analyzed simultaneously for a longer period. Moreover, they can be analyzed either together with other malware or individually. The infrastructure allows for a multitude of machines, with different operating systems, to be connected in a flat network composed of private Internet Protocol (IP) address ranges. The architecture's design and security are built on past research on malware sandboxing and safe architecture. Monitoring techniques were researched as well and implemented inside the newly created infrastructure, which was derived from the modus operandi of the chosen malware types. The layout of this system is extensible, flexible and testable.

To demonstrate the utility of such a design, this research provides a comparison between the analysis of two selected malware samples generated by the newly constructed infrastructure and well-known online, free sandbox solutions.

1.1 Contributions

In this research, the following contributions are made:

- A structured methodology towards creating a secure, monitored and flexible sandbox environment with the needed requirements and components.
- A proof of concept of an open source and extensible malware analysis infrastructure where multiple malware can be deployed for longer periods of time.
- A comparison between the analysis of two samples using the created design and free online sandbox solutions.

2 Related Work

When it comes to virtualized infrastructure for analyzing malware, we could not find any previous work detailing specifically on this topic in the open source community or publicly available. However, there is previous research on malware sandbox design requirements, secure architecture and what they refer to as a Honeynet [6]. A concept close to what this research tries to achieve.

In the paper "Honeynets Applied to the CSIRT Scenario" [7] made by Cristine Hoepers et. al. a Honeynet architecture is designed and implemented. The architecture was divided into two components: the Honeynet made out of Honeypots and the administrative network. The administrative network was composed of a file server, an Intrusion Detection System (IDS), a forensic machine and Hogwash which were in charge of containing the Honeynet traffic. The honeynet was monitored using the network information within the administrative infrastructure and the logs within the honeypots that were exfiltrated using the Syslog protocol to the file system machine. Additionally, this paper shows the value such a network could bring by revealing what are the current malicious activities in the wild and what are the most common cyber attacks within such an infrastructure.

In terms of malware sandbox environment, the United Nation's International Telecommunication Union (ITU) has published a paper ITU-T X.1218 [8] in their Series X. This paper covers some recommendations they make with regards to malware sandbox environments and is thus aptly called "Requirements and guidelines for dynamic malware analysis in a sandbox environment". Moreover, Adrian Sanabria's Global Information Assurance Certification (GIAC) paper [9] explores and outlines how to build a malware analysis lab environment from components to a whole architecture.

When it comes to security architecture principles, the National Cyber Security Center (NCSC) of the United Kingdom (UK) provides multiple secure principles regarding architecture and virtualization designs [10]. Fur-

thermore, Christopher C. Lamb's paper, "A Survey of Secure Architectural Principles" [11], summarizes a multitude of secure architecture principles from different sources giving an overview of what secure architecture entails.

3 Problem Statement

As technology develops at an alarming rate, the number of malicious attempts to use it grows as well. Thus, malware becomes more complex and advanced, continuously changing its behaviour. One of the most common techniques used by malware in recent years is propagation with the purpose of expanding its infection as much as possible. This can be seen in recent Ransomware variants (Ryuk [1]), Trojans (TrickBot [2]) or Wipers (HermeticWiper [3]). To combat the malware techniques, security experts use dynamic analysis within sandboxes to investigate their behaviour. The paper focuses on the most used free online sandboxes which would be: Joe Sandbox Basic Cloud [12], Triage [13], AnyRun [5], Cuckoo Online [4] and Hybrid Analysis [14].

The current free online sandboxes can possibly detect just the initial step towards propagation, not the complete process, which could result in missing important details of the malware's behaviour and actions. Furthermore, based on a manual review of the platform's given capability, the analyzed free online sandboxes have the following additional restrictions:

- **Provide short-term analysis** - the purpose of the machine is to run an analysis in a specific time frame and then deliver an analysis report. (All)
- **Do not provide live monitoring while the analysis is performed** - the actions taken by the malware can be seen just after the sandbox finished its report generation. The user cannot monitor the malware's actions during the infection. (Cuckoo Online [4])
- **Restriction on the number of deployed malware** - the number of malware analyses are restricted to a specific number. (Joe Sandbox Basic [12], Triage [13])
- **Time restrictions** - time constraints exist on how long the malware analysis can be performed. (AnyRun [5], Joe Sandbox Basic [12], Hybrid Analysis [14], Triage [13])
- **Its main design is to run on a single machine with one malware deployed at a time** - the malware is analyzed within a single machine and not a network of machines. Thus, it is restricted to what behaviour takes place on that single machine. Moreover, a user can deploy only one malware in a machine and not multiple ones to verify their interaction. (All)

Therefore, a malware aquarium solution in which malware may dwell for extended periods and flourish will give a comprehensive view of malware collaboration, behaviour, and spreading activities.

3.1 Scope

The project is limited to a proof of concept for a malware aquarium, a modular and user-friendly malware analysis network. The project excludes any malware Tactics, Techniques, and Procedures (TTP) that are not examined in this study. Additionally, only free online sandboxes are used for validating the analysis of the obtained proof of concept. The focus of the paper is on the results of the analysis of the two components.

3.2 Research Questions

This research attempts to answer the following main question and is split up into the below research sub-questions.

What are the added benefits of a malware aquarium approach compared to existing malware sandboxing methods?

3.2.1 Subquestions

- What is the behaviour of the targeted malware types?
- What detection methods can be implemented inside the infrastructure to securely monitor the malware's behaviour?
- What is a secure and extensible way to manage and deploy the infrastructure required for malware deployment and analysis?

4 Theoretical Framework

4.1 Malware Aquarium

A malware aquarium is a bounded space for malware to live its life, possibly indefinitely, without escaping its boundaries, while the environment observes its behaviour. Translating this into computer systems, in the context of this research, the malware aquarium is a fully reachable network with interconnected machines that have extensible and controllable visibility paths. From the malware's perspective, it is living in an actual network with multiple machines allowing it to propagate and possibly coexist with other malware strains within the same systems.

4.2 Secure Architecture

The malware aquarium's infrastructure entails a secure design which does not allow the malware to propagate outside the designated analysis environment. Christopher C. Lamb's paper, "A Survey of Secure Architectural Principles" [11], summarizes a multitude of secure architecture principles from different sources giving an overview of what secure architecture entails. Additionally, the NCSC of the UK provides a guide for the design of cyber secure systems [10]. From both sources the following key principles were distilled for use in this research project:

- **Simplicity** - also known as Economy of Mechanism, entails that the concepts and functionality provided by the architecture should be easy to understand and as simple as possible while maintaining product relevance.
- **Open Design** - the design should be open source to offer the possibility of verifying the code. Therefore, the security of the system should be visible and should not be based on obscurity but rather it should allow full inspection and verification of the system akin to Kerckhoffs's principle [15].
- **Design for Iteration** - the architecture should allow for fast iteration and easy extensibility. This is essential in the fast-paced development of new technologies and malware threats.
- **Redundancy** - the overall mechanism should be able to have a fast recovery process which would not intervene with its continuity. Self-healing capabilities should be preferred to manual intervention whenever possible.
- **Segmentation** - having a segmented approach to one's design can reduce compromise and increase simplicity and traceability in a system. This principle can be applied in multiple areas such as resource namespacing, networking and access control. Deny-by-default policies can allow for precise flow control over any information transmitted through the system, gradually layering more lenient policies on top.
- **Defense in Depth** - the implemented security mechanism should not be a single subsystem that is relied upon. Rather, it should be spread throughout the system in different layers of the architecture. This moves the design away from a single point of failure, aiding in resiliency and limiting the effects of a single security incident.
- **Monitoring** - a secure architecture provides ways to track what happens within and between its systems at multiple levels. Visibility in all layers such as network, process and hardware, is extremely useful. Having access to this aggregated information allows for threat analysis to be performed during or post-infection.
- **Accountability** - each change introduced into the system should be traceable to a unique identifier which can match the source of the action, be it introduced by a person or a system.

4.3 Sandbox Requirements

While our research concept of a sandbox environment differs from a traditional one [16], multiple design principles can still be applied to it. The main goals remain the same; to deploy and analyze malware. It is worthwhile to port these requirements over to our environment. According to the ITU-T X paper [8] and Adrian Sanabria's GIAC paper [9] these are the main requirements that a sandbox environment should have:

- **Isolation** - restrictions applied to the environment separating it from other systems. This should avoid attacks both from outside and inside the environment and should control the spread of the malware to unwanted locations.
- **Anti-Evasion** - methods to counter anti-sandbox and anti-analysis techniques which could result in the malware executing, but laying dormant essentially making it a "useless" execution.
- **Scalability** - provide the possibility to scale the solution and to be able to handle larger volumes of traffic and more machines of possibly different configurations.
- **Granular Behaviour Capture** - a sandbox should be able to record a wide range of behaviours, from system or application programming interface calls (API) to network traffic and process-related events.
- **Adjustability To Malware Needs** - the environment should have the possibility to adjust to the malware needs by creating an environment in which it can execute and proliferate. This could include services, operating systems, machine configuration and other variables.
- **Rapid Recovery** - quick recovery from errors or even the complete removal of the sandbox environment. This allows for so-called (automated) panic buttons and fast iteration.
- **Secure Malware Deployment** - reliable and secure deployment of malware inside the enclosed environment should be provided for.

5 Methodology

The following section discusses the approach of the research and goes into detail on each of the phases of this study. First, the testing environment used is described, followed by the approach to this research that introduces the sequential research phases which are further explained to provide a standardized approach. This chapter is divided into the following subsections:

1. Test Environment
2. Research Approach
3. Malware Type & Sample Analysis
4. Mapped Detections
5. Architecture Design and Implementation
6. Malware Deployment Analysis
7. Result Comparison

5.1 Test Environment

The utilized environment in this research was composed of a Dell server with 128 GiB of RAM, 56 CPUs and approximately 900 GiB of Solid-State Drive (SSD) storage with hardware accelerated virtualization support. The operating system used on the server is Ubuntu 22.04. Furthermore, an additional layer of security is added between the used machine and the internet by using a software-based firewall, named pfSense [17]. It resides on a different machine and acts as its only gateway to the internet. The version used in this research was 2.6.0 and it allowed established egress traffic using Domain Name System (DNS), Hypertext Transfer Protocol (HTTP) and Hypertext Transfer Protocol Secure (HTTPS). When it comes to ingress, Secure Shell (SSH) connections with the required key pair are allowed to connect to the machine. For both ingress and egress, bandwidth management with a restriction to 5 Mb/s was used to make sure that Denial of Service attacks cannot be performed from within the network.

5.2 Research Approach

As this paper focuses on validating the usability of a malware aquarium, it is composed of two general steps: building a malware aquarium and comparing the dynamic analysis results with free online sandbox solutions. The construction phase of the malware aquarium represents a complex and large process. Thus, it is divided in sequential steps which each help form the implementation of the malware aquarium.

The initial part focuses on the analysis of malware types to deploy and the modus operandi used by them. The second step uses the selected malware to map detections for those types of malware. The third part revolves around designing the architecture and implementing the proof of concept based on the previously researched requirements, secure architecture and sandbox, and newly analyzed malware behaviour and mapped detections.

After the malware aquarium is built, the next step is to deploy the malware samples from the investigated malware types that were researched in-depth and analyze their behaviour. Additionally, the same samples are analyzed in two free online sandboxes for comparison.

Lastly, utilizing the captured data from the malware aquarium and online sandbox deployments, a comparison between the discoveries of each tool is drawn.

The specified methodology was chosen because it provides the project with a structured perspective of each component required to develop such a project, as well as the ability to replicate and extend it by modifying the mentioned phases and technology choices. By following this approach where the malware type behaviour is analyzed, the research can uncover any potential gaps left in the sandbox and secure architecture requirements. If implemented correctly, the architecture's design should allow for rapid iteration implementing new technology and techniques that might help close these gaps.

5.3 Malware Type & Sample Analysis

For the initial design of the malware aquarium, a full analysis of the behaviour of malware types should be performed to better understand what initial security measures have to be in place to contain them with their modus operandi. This is done by taking multiple malware representatives from each type and mapping their behaviour on the Tactics, Techniques and Procedures (TTP) from the MITRE ATT&CK framework [18] which give an insightful overview of the capabilities that malware has. This is performed by using the MITRE Navigator [19] online application which allows its users to easily translate malware behaviour on the ATT&CK framework using layers [20] for each malware. After each malware was mapped on the specified MITRE framework, the resulting layers are collapsed into one whole for each malware type layer which contain all the techniques that those malware types use. Finally, all of the examined malware type layers are integrated to provide the full range of known techniques that the malware types can utilize.

Due to time constraints, only two malware types are examined in this study. For each kind, three examples are chosen, and a subset of the generated techniques are employed. The samples are picked based on the following criteria in this research:

- **Relevance** - the malware has to be relevant and current at the time of the research.
- **Level of Research** - the chosen sample must have been previously researched to a degree that allows us to have some certainty around its capabilities. This requirement is present to protect the environment directly outside of the test setup while the initial designs and implementation are laid down.
- **Accessibility** - the sample is available on an open repository or can be provided by the involved parties.

5.4 Mapped Detection

Based on the TTP extracted and mapped previously, detection methods are researched and proposed. This is done by creating a layer of abstraction from the newly obtained techniques in components that could be found in a network or host machine. An example of such a component would be network traffic. If a technique could be detected by inspecting network traffic, then that technique is placed in the network traffic category. The components that this research uses are based on dynamic analysis and revolve around: processes, network traffic, native API/syscalls, account actions and system/application modifications. Technologies that cover these areas are selected and implemented in the architecture part of the methodology, based on their various strengths, weaknesses and effectiveness.

5.5 Architecture Design and Implementation

The implementation of the malware aquarium's proof of concept is the amalgamation of all the requirements, sample analysis and design research done previously. Following the parameters outlined in the theoretical framework, two layers are being applied, identifying technical concepts and implementing technologies that best meet these needs within the available time.

The first two layers, Secure Architecture and Sandbox Environment, lay down the foundation of the initial implementation. This foundation enables the level of iteration, experimentation and adaptability required to apply any findings from the third layer Malware Behaviour & Detection Mapping mentioned previously in 5.2. With the malware aquarium's core idea involving networked machines, we start from the premise that at least two interconnected machines are available in the first implementation. Each of the layers may have separate needs, but it is possible that the technology choices may cross layer boundaries in terms of functionality or interactions, and so appear several times in the results.

5.6 Malware Deployment & Analysis

This phase is divided in two parts: the deployment of the malware and the resulting analysis from it. Deployment includes the transfer and detonation of the research samples inside the newly created malware aquarium. These samples are also detonated in two free online sandboxes with the maximum analysis time allowed. For this research these consist of:

- JoeSandbox Basic Cloud [12]
- Triage [13]

These solutions were chosen as they represent some of the most known sandbox technologies and are widely used by the cyber security community while utilizing the same operating system as the malware aquarium. If many pieces of malware are detonated, they are sent sequentially into the aquarium. Where the first sample is launched and executed in a clear environment, followed by the other. This allows us to better track the individual malware based on deployment times and analyze their behaviour both independently or together.

5.7 Result Comparison

The final part of the methodology is comparing the dynamic analysis results from the deployment of the malware aquarium with the results from the two chosen free online sandboxes. This is done by first evaluating each malware sample results, with the online sandboxes reports and then, investigating any deltas from their analysis. This should verify the effectiveness and usability of the malware aquarium design. From this phase, it is expected that the malware aquarium provides additional information when it comes to Command and Control and Lateral Movement actions.

6 Results

6.1 Malware Types & Sample Analysis

The selected and analyzed malware types in this study are: Trojans and Worms. Trojans were chosen because they are some of the most common types seen in the wild [21] and commonly used as a stepping stone for a further infection. Moreover, they provide a large number of capabilities. This results in a vast coverage of techniques used. The second type, Worms, are chosen for their self-propagation capability that is seen even in recent malware such as HermeticWiper [3].

6.1.1 Trojans

According to David Laka's paper [22], Trojan malware or Trojan Horse is software that is designed to look like a legitimate and harmless program to avoid detection. Additionally, there are multiple types of Trojans [23]. Banking Trojans were analyzed in this paper because of their capability to exfiltrate data and their heavy usage in recent years by email, according to the 2022 threat review of MalwareBytes [24]. Based on yearly reports from MalwareBytes [24] and BeyondTrust [25] the following three Trojans were the most active in the past 3-4 years.

Emotet Emotet [26] is a Windows-based malware known for its modularity and rapid evolution. It first appeared in 2014, as a banking Trojan and then continuously developed, containing capabilities such as information stealing, spam-bot activity and loading other malware. The malware's primary aim is to obtain access to the infected device, gather various information about the victim from the device, and download payload modules from the Command & Control (C2) system. It is designed to take advantage of the machine's profile and steal credentials. Moreover, it is common to drop other malware in the infected machine with the most common being other banking Trojans such as TrickBot, IceID and Ransomware. The targets of this Trojan are financial sectors or larger companies and it spreads by utilizing spam and phishing emails. The emails themselves contain Word and Excel files with malicious macros that download Emotet on the victim's machine. The tactics and techniques used by Emotet were analyzed by the MITRE corporation [18] and the full list of the techniques used by this malware can be found on its page in MITRE ATT&CK [27].

TrickBot TrickBot [2], similar to Emotet, is a banking Trojan that targets Windows machines and was created throughout the years. It first appeared in 2016, being one of the more recent banking Trojans with the initial goal of stealing financial information and evolved into a complex malware with multiple functionalities. These functionalities or modules are delivered via Dynamic-Link Library (DLL) files that each have their purpose, from gathering information to the exploitation of vulnerabilities and lateral movement. The modules are activated by contacting a C2 address from which the functionalities are chosen. The full spectrum of modules used by TrickBot is enumerated and described by SecureList [28]. Like Emotet, TrickBot spreads via phishing and spam emails, however, it can spread even through service vulnerability exploitation such as Server Message Block (SMB) vulnerabilities [29]. The TTP (Tactics, Techniques and Procedures) are monitored by the MITRE community on its malware page [30].

Dridex Dridex [31], like the previous two, is a banking Trojan with the main purpose of extracting information from its targets. It appeared in 2014 and has the same structure as TrickBot by containing a multitude of modules each with its tasks of downloading payloads or harvesting information. Dridex spreads via phishing and spam email and it is downloaded on the machine of the victim by using malicious macros within Word and Excel documents. Additionally, it tries to hide its digital footprint by encrypting and encoding its files. The TTP are monitored by the MITRE community on its analysis page [32].

6.1.2 Worms

According to David Laka's paper [22], Worms are self-replicating and active malicious programs that can spread over the network using vulnerabilities or common propagation commands. This allows the Worms to spread through a network without human interaction. Worms with network propagation capabilities, known as Net-Worms, are of main interest in this research. The reason behind it is because they were used in the most known malware attacks such as WannaCrypt [33] or Conficker [34] and the network spreading capability can be seen even in different malware TrickBot[29] and HermeticWiper [3]. The three representatives of Net-Worms that are researched in this paper were chosen for their actuality, capability and popularity.

WannaCrypt WannaCrypt [33], also known as Wannacry or Wannacrypt0r, is a notorious cryptoworm which created havoc globally in 2017. The malware spread via a SMB version 1 vulnerability titled EternalBlue and tracked as MS17-010, which can be exploited with a specially crafted SMB packet sent to ports 139 and 445. This allowed the malware to spread to over 300000 machines at an alarming speed. After the malware reaches the new machine, it starts encrypting multiple files. Additionally, the malware drops a backdoor, called DoublePulsar, inside the victim's machine for persistent access to the machine. Even if EternalBlue was initially affecting only Windows versions below Windows 10 and Windows Server 2019, there is a possibility of porting this vulnerability to even Windows 10 operating systems increasing the relevance of such a Worm and vulnerability. The full spectrum of TTP used by Wannacry is tracked by MITRE Community on its page [35].

HermeticWizard HermeticWizard [3] is one of the components of HermeticWiper, seen in February 2022 and it is used in the cyber-warfare that happens between Russia and Ukraine. The malware is used to propagate laterally in a network using two modules romance.dll and exec_32.dll. The romance.dll module uses SMB to propagate via known credentials, while the exec_32.dll uses Windows Management Interface (WMI) commands to perform lateral movement. After moving laterally in the network, the HermeticWizard launches the HermeticWiper, a Wiper software that swiftly corrupts the first 512 bytes of the Master Boot Record

(MBR) for each Physical Drive, rendering the computer inoperable. The TTP used by this Worm are tracked by MITRE [36].

EthernalRocks EternalRocks [37] is a Net-Worm that appeared in 2017 that spreads using all the NSA capabilities leaked by the ShadowBrokers [38] including multiple SMB vulnerabilities (EternalBlue, EternalRomance, EternalChampion, EternalSynergy) and three SMB scanning tools used for reconnaissance (SMBTouch, ArchTouch, DoublePulsar). Once in a system, it downloads the Tor private browser and then waits for 24 hours before dropping the NSA tools, which then are used to spread itself on the Internet. The payloads downloaded are usually hidden as normal Windows processes such as svchost.exe. Furthermore, EternalRocks does not have a specific motive besides spreading as much as possible, thus, being marked as harmless malware. Utilizing the analysis done by a researcher named Miroslav Stampar, the TTPs used by this Worm were mapped on the MITRE ATT&CK framework which can be seen in the project GitHub page [39].

6.1.3 Overall TTP

The resulting collapse of TTP used by the representatives of each analyzed malware type shows the full spectrum of techniques that these malware could perform according to MITRE ATT&CK and can be seen in the project GitHub repository [40]. Out of the overall represented tactics, four tactics are the main focus of this research because they are the tactics with the largest number of techniques and the most overlap between the malware types. The four tactics, including the reason they were chosen, are the following: Execution for its most common techniques used in both malware, C2 for its extensive usage by the banking Trojans, Lateral Movement which represents the essence behind Net-Worm behaviour and Defense Evasion to be able to understand the steps taken by the malware to avoid monitoring and triggering. The chosen tactics with their techniques and subtechniques can be seen in Appendix A.

6.1.4 Malware Sample

The deployed and analyzed samples within this paper are TrickBot for the Trojan type and HermeticWizard for the Worm Types. TrickBot was chosen because of its extensive capabilities from extracting data to moving laterally within the network, and being one of the most used malware in 2021 [41]. HermeticWizard was picked for its time relevance and capability to propagate via two media, WMI and SMB. Additionally, both were chosen because they fulfill the requirements mentioned in Methodology 5.2.

The sample for TrickBot is a recent one from 2022 chosen for time relevance with the name of sample2.exe [42] with the hash seen in Table 1. The sample was researched by Netsecurity [43], VirusTotal [44] and Aliaksandr Trafimchuk et al. [45]. Also, the current behaviour of Trickbot is shown in [28].

For the HermeticWiper the sample present in MalwareBazaar [46] with the executable details seen in Table 1 was picked. It was researched by ESET [3] and the Cybersecurity and Infrastructure Security Agency (CISA) [47] which give in-depth insights in its behaviour.

HermeticWiper Sample Details	
File Name	a259e9b0acf375a8bef8dbc27a8a1996ee02a56889cba07ef58c49185ab033ec.dll
SHA256	a259e9b0acf375a8bef8dbc27a8a1996ee02a56889cba07ef58c49185ab033ec
TrickBot Sample Details	
File Name	sample2.exe
SHA256	236f4e149402cba69141e6055a113a68f2bd86539365210afb9861f4e2d3ad5f

Table 1: Malware Sample Details

6.2 Mapped Detection

The evaluated tactics and techniques seen in Appendix A were translated to a level of abstraction for essaying the detection mapping process. The translation contains the following main components: process, network, account, native API, system modification and booting-related actions. The system modification component contains any modification made to registry keys, files, directories, accounts or the system itself. Each subtechnique or technique, if a subtechnique is not available, represented by its code, is translated into one of the components. The result of this mapping can be seen in Table 2.

Related Actions With Their Subtechniques	
Related Actions	**Technique & Subtechniques**
Process	T1059.001, T1059.005, T1059.006, T1559.001, T1053.005, T1569.02, T1204.002, T1047, T1036.004, T1036.005, T1055.001, T1055.012, T1218.010, T1218.011,T1553.002. T1497.003
Network	T1204.001, T1210, T1570, T1563.002, T1021.003, T1021.002, T1021.005,T1071.004,T1072.001, T1132.001, T1573.002, T1573.001, T1008, T1105, T1571, T1090.002, T1090.003, T1219
Native API	T1106
System Modification	T1140, T1222.001, T1564.001, T1112, T1070.001, T1562.001, T1027.002, T1078.003
Booting	T1542.003

Table 2: Mapped Detections

6.3 Architecture Design and Implementation

6.3.1 Secure Architecture Design

The given design adheres to the methodology and considers thoroughly the sandbox requirements, as well as the above-mentioned demands for a secure architecture. The trade-offs of the design might come as a consequence of incompatibility, lack of previous implementations or incomplete coverage of the principles employed. All of them influence the resulting architecture of which many variants could potentially fit the requirements. The architecture is described from a higher functional level down to lower levels, listing which technology implements what principles and meets which of the gathered criteria as shown in Sections 4.2 and 4.3.

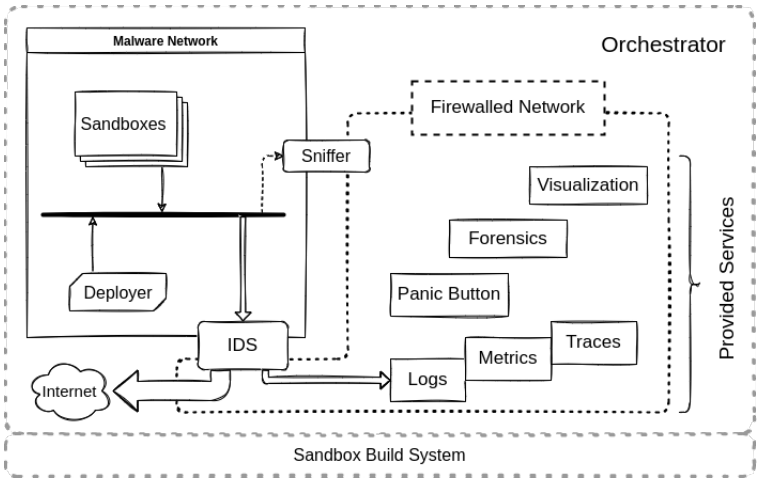

Figure 1: Architecture Overview

The high-level overview shown in Figure 1 consists of several different parts: the sandbox and its build system, the monitoring services, the network they preside on and the orchestrator managing all of these components. More detail concerning the specific technologies used is discussed in the following subsections. First, this section focuses on how the different parts of the architecture conform to the sandbox and secure architecture requirements.

6.3.2 Sandboxes & Build System

A core component of the architecture is the sandboxes themselves of which a more detailed view is shown in Figure 2. The technology used must have proper isolation controls and properties built in. The de facto technological concept used here is virtual machines. They provide process, memory, storage and network isolation with various levels of configurability. The choice alone of using virtualization technology already satisfies many sandbox requirements, namely Isolation, Anti-Evasion and Adjustability To Malware Needs.

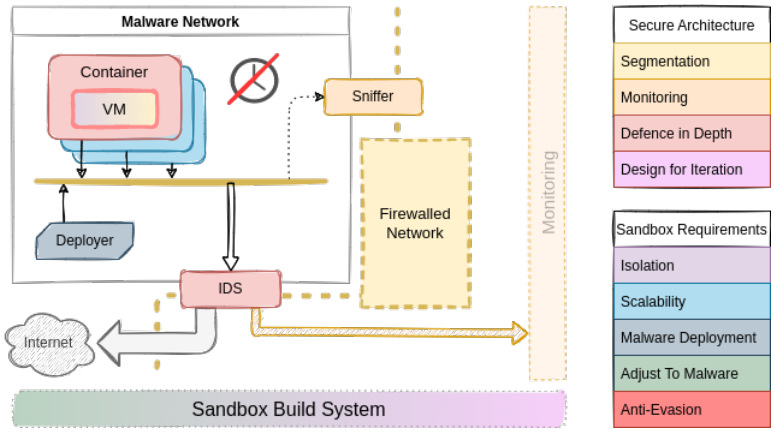

Figure 2: Sandbox & Build System

Virtual Machines A virtual machine has the benefit of allowing for emulation, meaning that various components of a computer system can be mimicked to appear and behave like specific real-world hardware or entirely new models. Additionally, an emulator can make use of hypervisors to provide different levels of hardware acceleration or even hardware pass-through, such as Graphical Processing Unit (GPU) or a specialized Network Interface Card (NIC). This all aids in tricking the malware into thinking that it is running in a genuine system. This applies even in the case where malware decides to lay dormant for an extended period of time to evade time limited sandbox environments.

Allowing for different kernels to be run inside of virtual machines increases the level of isolation between it and the surrounding systems. For example, a kernel vulnerability in a guest operating system does not immediately expose the host machine. Moreover, it enables entirely different classes of operating systems to run within the same networked environment. For example, Linux, Windows or even mobile operating systems like Android running side by side.

Containers These virtual machines are wrapped in containers. This has numerous advantages that involve the orchestrator, mainly pertaining to the architecture's scalability. From an architecture and sandbox perspective, while it does not provide the same level of isolation as virtual machines, it does add an extra layer of defence that threats need to overcome. If anything breaks out of the virtual machine's emulation layer and it ends up in the container, it will still have a hard time moving around as it has to break out of the additional layer the container provides. The tools installed in the container are very sparse and mainly pertain to running the virtual machine emulation. Tools to manipulate the networking stack, while there, require elevated privileges that the container simply does not have. Allowing these containers to provide the tools required for running virtual machines, as opposed to the host operating system providing them, creates opportunities to fine-tune them. They can provide different features without having to rebuild the entire underlying operating systems or virtual machine images.

Build System Any virtual machine images deployed in this setup can come directly from their vendors or respective open-source projects without any customization. However, the ability to very specifically have images contain features, vulnerabilities or other specific patches has proven crucial for experimentation. Configuring operating systems can be time intensive and might not be easily recreated if there is no standard way for executing the required customization. Design for Iteration and Adjustability for Malware Needs are both items in the secure architecture and sandbox requirements. Thus, providing build systems that can reliably produce ready-to-use images in any desired configuration was essential.

Network Connectivity The machines deployed in the malware network need to have full connectivity between each other to allow internal malware propagation. This is represented by the ochre colored horizontal bar in Figure 2. In the current design this is implemented using a standard layer 2 Linux bridge. While this does not allow the virtual machines to be deployed across multiple hosts, it does make it possible to see any network traffic that crosses it by using a sniffer. Some malware requires internet connectivity which we provide in the form of a gateway. This gateway has an IDS installed which is used as an early warning system for any

malware outbreak towards the internet. It also runs a Domain Name System (DNS) resolver that acts as a filter and routes queries meant for the monitoring systems differently while it directs all other queries to the default upstream resolvers.

While the internal malware network is open, any outbound network traffic should be controlled and restricted according to the parameters of the experiment and to satisfy the Segmentation property of the secure architecture property list. For this reason, the network immediately beyond the gateway is tightly controlled by the underlying network provider. Towards the internet it only allows traffic conforming to the Hypertext Transfer Protocols both secured and unsecured (HTTP/HTTPS) and DNS queries. Event logs are exported from any running virtual machine so specific connectivity from the malware network into the monitoring systems is allowed. The logging system is HTTP based so constraints are placed on the use of a specific port number, HTTP POST method and HTTP path. When it comes to other information like performance metrics from the virtual machines, these are provided by the orchestrator out of band and are not crossing the malware network boundary.

6.3.3 Data & Analysis

Secure architecture requires Monitoring to be part of its properties and the sandbox requirements call for Granular Behaviour Capture as stated before in Sections 4.2 and 4.3. With the sandboxes and a build system in place and a basic way to extract information regarding traffic generated by the systems, the focus can shift towards extracting information from the virtual machines.

Monitoring As said before the orchestrator can provide us with basic metrics surrounding virtual machine performance like CPU and memory usage, disk performance and network activity. This covers the basics of what could be called black-box monitoring where only externally visible signals are used to try and infer a system's internal state. In Figure 3 this black box monitoring is represented by both the orchestrator bar at the bottom as well as the firewalled network on the left.

To allow for more detailed analysis white-box monitoring capabilities coming from within the virtual machine are also required. The Google SRE Book [48] describes it as follows:

> Monitoring based on metrics exposed by the internals of the system, including logs, interfaces like the Java Virtual Machine Profiling Interface, or an HTTP handler that emits internal statistics.

The guest operating system has all of the information required to enable white-box monitoring. It needs to be extracted and transformed in a way where it can be treated as any other part of the system's observability traits. For this research in particular, extracting and sending out the Windows Event Logs to the logging system was crucial in understanding what is happening inside of the virtual machine. Multiple agents were installed in them to make it happen.

All of this telemetry is shipped to their respective collectors and aggregators using standard protocols and formats. It makes switching out components or even deploy and test multiple components with overlapping functionality, relatively simple. This helps with the architectures Design for Iteration property when searching for components that better fit the requirements. Including how these monitoring logs and metrics are consumed by visualisation, analysis and automation software.

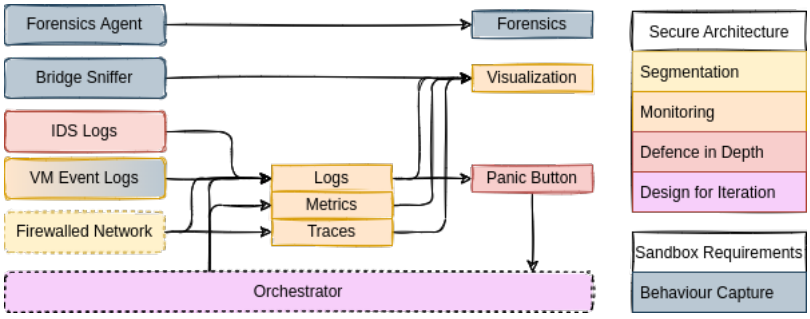

Figure 3: Data & Analysis

Defence in Depth Security actuators, like the panic button shown in Figure 3 in red on the right, can also make use of the monitoring information. It is fed pattern matched classification logs the IDS submitted to the logging system with the panic button feeding from that stream through the standard logging system's query API. This panic button is watching for specific patterns in those classifications and triggers, if any of them is found. That trigger requests the orchestrator to remove all virtual machines in the system with no graceful shutdown periods. This, again, is only there to satisfy the secure architecture's requirements of Defence in Depth. But it also provides a basic example for other automated workflows and notification systems tailored to the environment and malware.

Forensics A final component that was introduced pertaining to Monitoring and Granular Behaviour Capture is a forensics agent and service. This agent is installed through the build system in all the deployed images. This agent connects to the provided forensics service, pushing some basic identifying information and awaiting any commands that the service needs to execute. This allows for ad hoc querying for anything that the agent provides, from basic downloading of files to full-on memory analysis, on the entire fleet of deployed virtual machines at once. With experiments potentially going on for weeks, doing ad hoc forensics in between start and end can give unique insights into any behaviour that malware could exhibit.

6.3.4 Orchestrator & Network

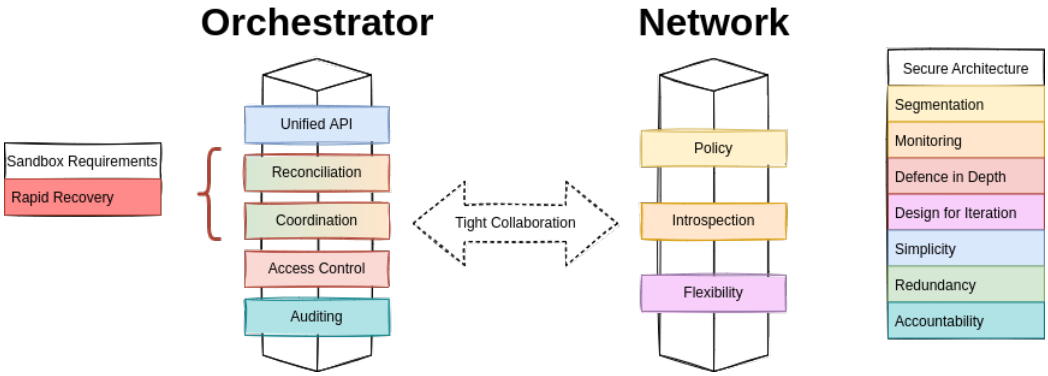

Figure 4: Orchestrator & Network

With so many moving components, points for configuration and dynamism, having a system in place that can take commands and ensures everything is in the desired state becomes a necessity. This orchestrator's job is manifold. First and foremost, it needs to run and manage all the sandboxes including all its auxiliary services. Secondly, it tightly collaborates with the network to very specifically shape network traffic. Next to that, it allows for auditing and access control, all while presenting and controlling resources through a unified API significantly simplifying the system as a whole.

Combining the capabilities of the orchestrator and network covers nearly all of the secure architecture requirements. On top of that, state reconciliation and coordination play a large role in the Rapid Recovery of the sandbox environment as well.

- **Segmentation** using access control and network policy enforcement
- **Monitoring** of workload status and traffic
- **Defence in Depth** using access control over all managed resources
- **Design for Iteration** by easily allowing for resource and network modification
- **Simplicity** by abstracting almost all of the functionality behind a unified API
- **Redundancy** by providing coordination and reconciliation of desired system state
- **Accountability** by audit logging any action executed

6.3.5 Implementation

Implementing an architecture is an iterative process and since these technologies are often interdependent or interact in subtle ways, switching out one component might require switching or adjusting another. An interesting side effect of iterating is that as technologies get swapped, replacements which are modular and behave well with other systems are favoured over ones that do not. If a choice is made that requires more lock-in or is more rigid in nature then that choice is made very deliberately. The technologies listed here are a result of this iteration that attempts to fill the sandbox requirements as best as possible while also adhering to the secure architecture guidelines. These technologies can also be seen in Figure 5, Table 3 and all of the used configuration manifests can be found on GitHub [40].

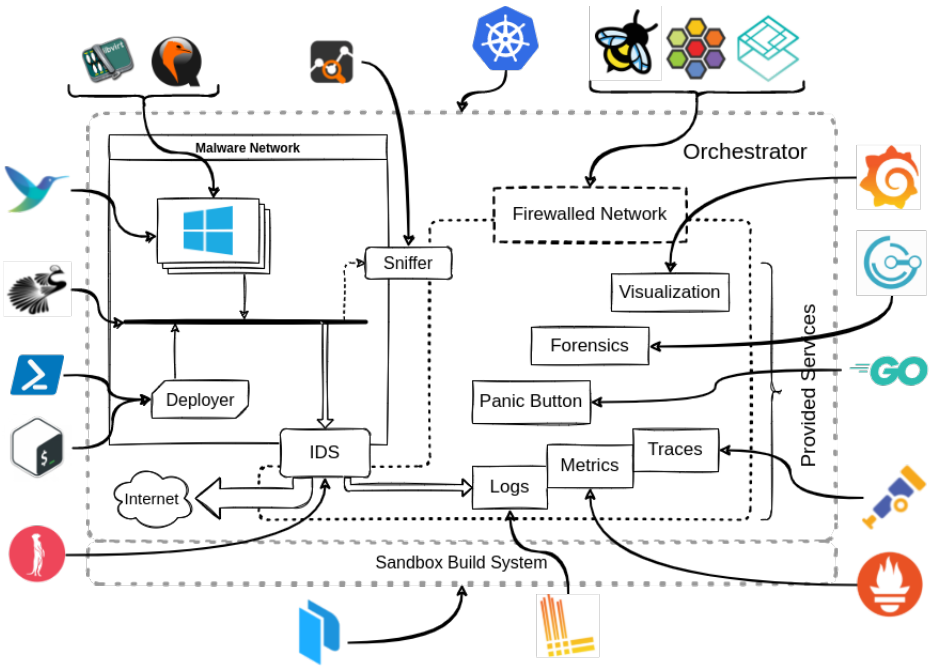

Figure 5: Technology Choices

6.3.6 Sandbox Technology Choices

Virtualization & Containerization The core of our sandbox is the concept of a virtual machine. QEMU [49] was chosen as the basis for providing the virtual machine emulation functionality which in turn can use the acceleration functionality provided by the Kernel-based Virtual Machine (KVM) [50] built into modern Linux kernels. These machines are managed using the libvirt virtualization API [51] providing a standardized way of interacting with different virtualization technologies. The optional KVM acceleration can provide speedups to machine execution or can simply be turned off to trade in speed for increased security. Thus, parts of the machine are no longer executed in kernel space, but fully in user space. One downside to using QEMU is the relatively large size of the project. Breakouts through devices it emulates are not unheard of, such as the VENOM floppy disk vulnerability [52] first found by Jason Geffner. Since most of the analyzed malware types operate on Windows, namely Windows 10, this operating system was chosen as the basis for our virtual machines. The container surrounding the virtual machine was implemented using containerd [53].

Build System In order to get the sandbox images in the desired shape, the Packer [54] build system by HashiCorp was introduced. This project enables building different types of images including VirtualBox, LXC, Docker, Hyper-V and more, courtesy of its plugin system. The underlying technologies used to provision said images are very configurable from simple shell script execution to full on configuration management tools like Ansible and Puppet. This builder proved invaluable in providing consistent and repeatable builds for the base experimentation images used within the sandbox.

Malware Deployment Each image that was deployed had a Powershell script watching a specific folder. Whenever an event signalled the creation of a file in that folder it would simply execute it. For this research, we limited ourselves to `.exe` and `.dll` files, which are the file types used by the selected malware samples. These files would be transferred to the target machine using the already present SMB shares. To actually deploy those files a short-lived container would be started that only had a network connection inside of the malware network and thus is subject to the same security measures the rest of the machines are subject to. This container would have a special in-memory file system mount which would be used as a working directory. A Bash [55] shell script is executed that uses basic tools to download (curl [56]), extract (7-zip [57]) and transfer (smbclient [58]) the malware after which the container terminates automatically, removing the in-memory file system. Keeping this part simple and easily extensible is quite crucial depending on the desired type of malware to be deployed.

Windows Event Logs Extracting information from the Windows 10 systems is done using the built-in Windows Event API [59]. These are not all enabled by default, hence, some configuration was required. This system was used as the primary way of gathering information and was augmented using the Sysmon [60] system service provided by the Windows Sysinternals project. The configuration for each one of them is based on the abstraction layers shown in Detection Technologies 6.3.9. Shipping these logs was done using a commonly used log collector called Fluent Bit [61] which provides a multitude of output formats such as Syslog [62], Elastic Search [63] and, the chosen format, Loki [64].

Gateway The gateway bordering the sandbox environment and the rest of the system is implemented using three containers that share the same network namespace. Moreover, the gateway has two interfaces configured: one for the malware network and another inside of the firewalled network. The first container only exists briefly before the other two are started and contains tools to manipulate iptables [65] for the container's network namespace. It setups Network Address Translation (NAT) rules to masquerade traffic entering the malware network interface and exiting the internet-facing interface. The main container runs Suricata [66] used for sniffing the malware network-facing interface and classifying traffic on the fly. The other one runs a CoreDNS [67] server which has the primary goal of providing a recursive resolver for the systems inside of the malware network. Suricata is used because it provides a readily usable large community library of classification rules. It outputs any classifications to standard out which are automatically picked up by the log aggregation systems. CoreDNS was used because it is simple to configure and provides a large number of plugins to adjust its behaviour. For example, for catching domains specifically meant for the monitoring systems and resolving them with a different upstream resolver. This is a key point to also log DNS queries, do domain rewrites or implement other custom DNS-based behaviour using the extensible plugin system CoreDNS provides.

Out of Band Sniffing One final thing to mention that lives at the perimeter of the malware network is another network sniffer called ntopng [68]. It is technically not part of the malware network, as it has a read-only view of the bridge in the host namespace connecting all the virtual machines. This system provides live insights about all the traffic inside the malware contrary to only the peripheral view Suricata gets at the edge gateway. It is more meant as an ad hoc inspection system for the users instead of a structural analysis system like the event logs exported from the virtual machines. This component is currently the only thing locking the implementation to a single machine, as the bridge only connects virtual machines executing on the same host machine. It is possible to use the orchestrator's underlying network to create highly shielded and controlled sub-networks or isolated endpoints. This unlocks the scaling properties for the virtual machines even more while also making use of the network insights that the underlying network provides.

6.3.7 Monitoring Technology Choices

Logs Outside of the sandbox perimeter, within the firewalled network, reside the monitoring services. One of which is Loki [64], the log aggregator and query service. It is based on an HTTP/gRPC [69] push model and stands out because of its ability to use a multitude of storage systems, while remaining light weight. In combination with Fluent Bit, used inside of the virtual machines, it can easily store any events emitted by them. Furthermore, it can query them based on either orchestrator labels assigned to the log streams or other labels assigned manually during export from Fluent Bit. A distinct advantage of this system is that it can feed any logs produced into the system and query it using the same tools. Logs produced by the network, orchestrator or any service running on the orchestrator are all accessible through Loki.

Metrics Another major component that is part of the monitoring system is Prometheus [70], a combination of a pull-based metrics gathering system and time series database. All components of the orchestrator and most of the deployed services emit metrics in the OpenMetrics standard format, ingested by Prometheus. While it monitors all of the systems, the main use case within the context of the experiments is tracking virtual machine telemetry like CPU, memory and disk activity.

Traces Finally, tracing has also been implemented, a way of tracking units of work across multiple systems. This is harder to implement because traces generally need to traverse systems in order to be useful which means that systems need to adapt their programming to it. Nonetheless, there was still room to make use of them here as the network provider emitted traces for the application and network layers. This shows the communication paths that for example a DNS request took, including other information that is attached as attributes to the trace. For this, two projects were put in place: the OpenTelemetry Collector [71] and Tempo [72]. The OpenTelemetry standard is used to standardize telemetry collection, this includes logs, traces and metrics, including the OpenMetrics standard. Traces would be ingested by the collector and exported to the Tempo system. This system would act as a storage and query layer much like its sibling Loki.

Visualisation All of the data collected should be accessible for analysis, for both people and machines. The choice fell on Grafana [73], a web-based service with a wide range of supported backends, including all of the used systems. It allows for visualisation of all types of telemetry covered previously. Dashboards can be created that display any of these sources and can even link between them, if the source supports it. During the course of this research, various dashboards were created with panels representing the most often used queries like listing DNS requests for a specific host or anything that triggered a warning from the Suricata IDS.

Panic Button Having our telemetry based on standards, allows for deployment of multiple types of visualisation and analysis systems that use the same APIs. One of which is our *"Panic Button"*, a small Go [74] program that uses Loki's gRPC [69] API to watch for any hits on a particular log query and requests the orchestrator to remove all the virtual machines if it encounters one. This can also be triggered manually using an HTTP POST request.

6.3.8 Orchestrator Technology Choices

Orchestrator Having covered nearly all of the services, the orchestrator that manages all these systems and services should be mentioned. Kubernetes [75] became the orchestrator of choice for a number of reasons. Firstly, it provides a single unified, access-controlled API, allowing configuration of almost all of the systems. This made building the Panic Button's functionality relatively straightforward. Because the Kubernetes API is designed in a declarative manner, the entire configuration of the architecture can be stored in a version control system and can be recreated relatively easily on different hardware. Kubernetes' ability to be extended through what it calls Custom Resources [76] was key to integrating systems into Kubernetes. For example, integrating the concept of a virtual machine through the KubeVirt [77] project. This allows it to manage the sandboxes, attach and detach drives to it and even provide Virtual Network Computing (VNC) connectivity. In its core, the Kubernetes design automatically increases resiliency for anything deployed using it. It could be any of the monitoring services like Loki, the Suricata/CoreDNS gateway combination or the virtual machines themselves. By continuously reconciling current state to desired state, it can make sure that systems quickly recover from transient failures. Kubernetes is provided using different distributions, not unlike the way Linux operating systems come in different distributions. The k3s [78] distribution was chosen because of its small footprint, easy installation and bundling of a container runtime all in a single binary. This simplicity reduced the requirements on the host operating system Kubernetes is running on.

Network Kubernetes requires a network plugin adhering to the Container Network Interface (CNI) specification, enabling extensible network management delegation. The Cilium project was used to power the network. Being eBPF- based, it can hook into specific locations in the Linux kernel to provide basic network functions. An example of such functionality is routing, but it can also provide logs, metrics, and traces along with enforcing network policies. All of these were vital for the sandbox requirements and secure architecture. Appendix refappendix:int$_f irewallshowsanexampleofthenetworkpolicythatgovernedtheingressandegressoutsideofthemalwarenet$
$bridgewiththehost-localIPaddressmanagement(IPAM)citecnicni-host-localmethodtosetupthemalwarenetwork.Int$

Forensics Lastly, a forensics system was selected. This system existed to easily execute commands and gather information from inside the virtual machines. For this GRR Rapid Response Citation, an incident response framework developed by Google was used. The documentation describes its goal as:

> The goal of GRR is to support forensics and investigations in a fast and scalable manner to allow analysts to quickly triage attacks and perform analysis remotely.

This gives any person running experiments almost unlimited access to the infected machines without requiring ingress. After installing the agent in our Windows 10 image, it connects back to the GRR server and awaits further instructions. This makes downloading files, running memory analysis operations, or other forensic tasks across all the machines relatively painless and is an ideal addition to the already-emitted event logs and metrics.

Technology List	
Fields	**Technologies**
Virtualization & Containerization	QEMU with KVM and libvirt & containerd
Build System	Packer
Malware Deployment	Powershell, Bash and Go
Logs Extraction	Windows Event Logs, Sysmon, FluentBit
Telemetry & Visualization	Loki, Prometheus, OpenTelemetry and Grafana
Network	Cilium, Multus and CoreDNS
Network Security	Suricata and Ntopng
Orchestration	Kubernetes
Forensics	GRR Rapid Response

Table 3: Used Technologies

6.3.9 Detection Technologies

Having enumerated these technologies, it can be assessed that their coverage overlaps with multiple other technologies meant to detect tactics and techniques employed by the selected malware types. They all do it with different approaches and with varying levels of detail increasing the chance of detection. This mapping between technology and detection method can be seen in Table "monitormonitor-table," where the abstraction level described in Section "sec:dec-abs" was used. However, only the most prominently related actions were taken into consideration because they covered the majority of the techniques and subtechniques.detail,detect the

Mapped Technologies On The Abstraction Level	
Related Actions	**Technologies**
Process	Event Logs, GRR, Sysmon
Network	Cilium, GRR, ntopng, Suricata, Sysmon
System Modification	Event Logs, GRR, Sysmon

Table 4: Detection Technologies

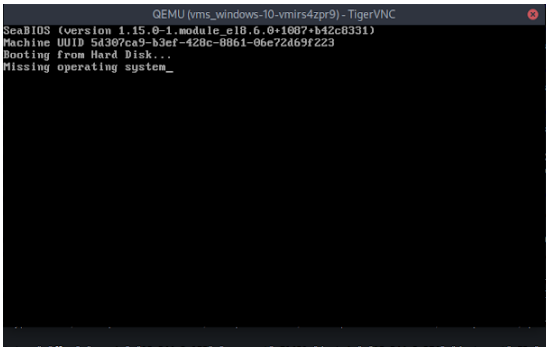

Figure 6: No Operating System detected

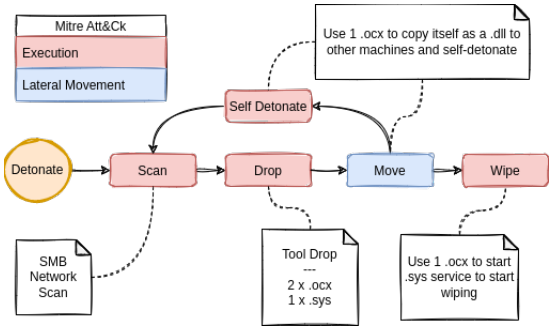

Figure 7: Flow Chart HermeticWiper Analysis in Malware Aquarium

6.4 Malware Deployment & Analysis

The HermeticWizard malware was initially distributed, followed by the TrickBot sample, which explains how the findings are structured. When it comes to the setup, two machines were used in the malware aquarium, as a starting point in each experiment, except the HermeticWiper where a four-machine experiment was also performed. Additionally, the analysis of the malware was done manually with self-made queries and dashboards. The configuration used for the free online sandboxes was set to the maximum allowed analysis time while keeping the interaction with it to a minimum.

6.4.1 HermeticWizard

Malware Aquarium The HermeticWizard run took between 10 and 12 minutes to erase a machine to the point that the operating system could no longer be identified, as shown in Figure 6. The steps taken by the malware are fully described in the created timeline in Appendix C where the time, general action and source of the information are mentioned. An overview of the main steps are given in this subsection following the flow diagram from Figure 7 that contains the used MITRE techniques as well.

The initial step after the malicious DLL file is executed, is to scan the internal network for opened Server Message Block protocol (SMB) services on port 445 with the purpose of discovering any neighbouring machines. After the scan is done, a file containing the Wiper, with the format .ocx [79] and a 6 random character name is dropped. In this case the file was named `cehwda.ocx`. Immediately after executing the Wiper, a driver file is created within "C:\Windows\System32\drivers\" with the name `pvdr.sys` where the two first characters are randomly generated. Then, this service is registered and starts immediately, which executes the wiping process in memory according to the CISA article [47]. Shortly after, a new .ocx file is created that is then used to propagate to the found SMB neighbours.

From the perspective of the second machine, as soon as the propagation starts, an SMB connection is identified and a new DLL file is dropped within "C:\Windows\" folder. Afterwards, a remote command using Windows Management Interface (WMI) is used to execute the DLL which restarts the Wiper process in this host as described above.

To verify if the propagation of the Wiper is sequential, or if it spreads to all devices at once, four machines were deployed. This has shown that the propagation of the HermeticWizard is done at the same time for all the virtual machines. This can be seen in Figure 8, CPU usage and Figure 9, disk throughput. The bottom bar in both graphs have increased activity before the other three machines simultaneously have a spike in usage.

Joe Sandbox Cloud Basic The full analysis of the Joe Sandbox was 12 minutes overall, with 3 minutes of dynamic analysis, which is the maximum time the free version allows. In these minutes of analysis, the Wiper [80] managed to deploy a .ocx file with the .sys driver and perform a network scan, which resulted in the detection of itself. Hence, the analysis was not complete as it can be seen in the flow chart from Figure 10.

Triage Even if the malware analysis took longer than the Joe Sandbox detonation, there were no results regarding the virus deployed after 30 minutes. As a result, the findings [81] revealed no HermeticWizard-related behaviours.

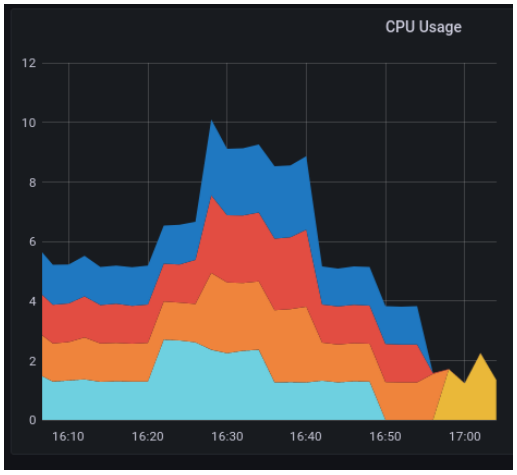

Figure 8: Four Machine CPU Usage HermeticWizard

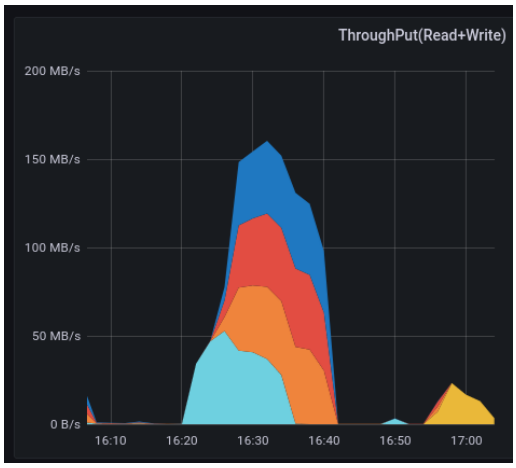

Figure 9: Four Machine Disk Throughput HermeticWizard

6.4.2 TrickBot

Malware Aquarium The TrickBot Trojan was subjected to a two-hour study session to ensure that its behaviour was thoroughly examined. The two-hour time range was also chosen due to time constraints especially with how lengthy manual examination of these data can be. The full analysis can be found in Appendix D in the form of a two hour event timeline where the repeated actions are deduplicated. The essential steps taken by TrickBot revolve around hiding the executable and continuous communication with the Command and Control server.

The first step that the Trojan takes is to copy itself into the "C:\Users\admin\AppData\Roaming\" directory using the svchost.exe process and then again in the "C:\Windows\System32\config\systemprofile\AppData\Roaming\". Additionally, it registers the executable in the Registry Key related to application compatibility to make sure the executable is compatible with the operating system. Afterwards, using the svchost.exe, the TrickBot executable found in the "C:\Users\admin\AppData\Roaming\" contacted the domain "myexternalip[.]com". Then, the svchost.exe process, that is started continuously by the executable, contacts multiple malicious TrickBot IPs using HTTPS.

The overall number of unique IPs addressed was 160, however, these include false positives since the Windows workstations generated network traffic to Microsoft servers and other services that Windows uses. After filtering out known IPs, around 90 IPs that might potentially be associated with TrickBot were left out, although false positives may still exist because not all IPs were identified as being related to TrickBot. Additionally, the task schedule application was opened using the mmc.exe executable, however, the reason why this happened could not be found in the logs. Furthermore, no lateral movement was seen.

One action that the malware took was not present in the logs and it was only detected via a VNC connection to the machine. This action was the creation of a scheduled task that occurred immediately after the first execution of the malware. The task was named Bot which was to run at 12:00 AM every day and executed the AppData executable every second.

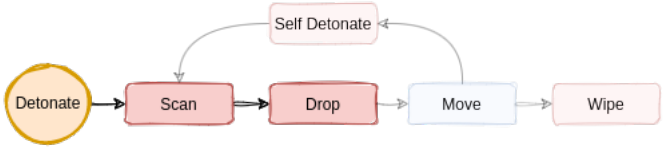

Figure 10: Flow Chart HermeticWiper Analysis in Joe Sandbox

Joe Sandbox Cloud Basic The Joe Sandbox detonation [82] revealed that the executable started a svchost.exe which generates Command and Control traffic, with 12 IPs being contacted using HTTPS. The same domain was contacted as well, "myexternalip[.]com". The full analysis lasted 6 minutes and 37 seconds with the dynamic analysis being approximately 3 minutes. Additionally, the sandbox shows proof of a new scheduled task being created with the name Bot, without requiring any user interaction to discover it. However, it was detected using Windows API tracking which is outside of the current capabilities of the malware aquarium.

Triage The Triage sandbox dynamic analysis [83] revealed that the executable was copied in both directories "systemprofile" and "<Username>\AppData", that the domain "myexternalip[.]com" was contacted, that mmc.exe activity was detected, and that svchost.exe was used to contact 50 IPs via HTTPS. Furthermore, the creation of scheduled tasks for the Bot operation was not stated in the report. Nevertheless, it can be observed if the user interacts with the sandbox and searches for it.

6.5 Result Comparison

6.5.1 HermeticWizard

The malware aquarium displayed the whole chain of propagation, from the scan to the remote copy of the Wiper and its remote execution. Furthermore, it confirmed with four target hosts that the virus spreads practically simultaneously inside the network rather than sequentially. Aside from that, it got to the stage where the wiping started and the victim machines were erased.

On the other hand, the Triage sandbox did not identify any of the Wiper actions and the Joe Sandbox detonation was able to detect until the initial step towards the propagation to a new host. This step was the command used with the *.ocx* file to the neighbour IP, where in this case it connected to itself. Thus, excluding the wiping activity and the full propagation process. The discrepancies in the outcomes of these three analysis may be observed in Table 5, where a green plus indicates that it had detected that behaviour and a red minus indicates that it had not.

Analysis Comparison HermeticWizard		
Malware Aquarium	**Joe Sandbox**	**Triage**
+ Execution of DLL	+ Execution of DLL	+ Execution of DLL
+ SMB Scan on port 445	+ SMB Scan on port 445	- Nothing was detected
+ .ocx files created	- .ocx file created	
+ .sys file created	+ .sys file created	
+ Add driver in Registry Key	- No driver in Registry Key	
+ Command to move laterally using the .ocx file	+ Command to move laterally using the .ocx file	
+ Dropping DLL to other machine using SMB	- Dropping DLL to itself using SMB	
+ Remote execute DLL file using WMI	- No remote execute DLL file using WMI	
+ Files created on the second machine	- No files created on the second machine	
+ Wiping started	- No wiping started	
+ Proof of wiping of both machines	- No proof of wiping of both machines	

Table 5: Analysis Comparison HermeticWizard

6.5.2 TrickBot

The largest difference between the malware aquarium and the rest of the sandboxes is the time frame in which TrickBot was analyzed. This can be seen in the amount of IPs contacted: 90 for malware aquarium, 50 for Triage and 12 for Joe Sandbox. However, the Joe Sandbox contains only the initial executable with a few svchost.exe processes compared to the other analysis, but the task schedule new creation action was detected in Joe Sandbox using an API monitoring solution. These differences are highlighted in Table 6.

Analysis Comparison TrickBot		
Malware Aquarium	**Joe Sandbox**	**Triage**
+ Start of executable file	+ Start of executable file	+ Start of executable file
+ Multiple copies of the executable discovered	- No copies of the executable discovered	+ Multiple copies of the executable discovered
+ Registry entry found	- No registry entry found	+ Registry entry found
+ Multiple svchost.exe discovered	- One svchost.exe discovered	+ Multiple svchost.exe discovered
- No scheduled task creation proof	+ Scheduled task creation proof (API)	- No scheduled task creation proof
+ 90 IPs contacted	- 12 IPs contacted	- 50 IPs contacted

Table 6: Analysis Comparison TrickBot

7 Discussion

7.1 Malware Aquarium vs Compared Free Online Sandboxes

In both malware analysis cases, the malware aquarium provided a deeper insight in the modus operandi of the malware and the used indicators of compromise (IoC).

In the HermeticWiper analysis, the malware aquarium provided the full trace of the propagation alongside with the second host infection logs and proof of wiping. As a result, the malware's whole infection process and behaviours are displayed. This is critical for delivering countermeasures and defending against malware since a detection or countermeasure may be built for each step that the infection takes. Furthermore, due to its extendable and continuous deployment, little aspects such as the infection occurring concurrently rather than sequentially and the infection phase lasting around 10 minutes might be identified. When it comes to blocking malware and infection countermeasures, these data may be crucial in a business infrastructure.

In the TrickBot analysis the other benefit of the malware aquarium can be seen. For malware which focuses on stealth and long term development a prolonged analysis is required to fully understand its behaviour. Moreover, malware such as TrickBot which receives commands from a C2 server can modify its behaviour continuously depending on the environment it is present in. Hence, having the ability to readily adjust one's surroundings to the infection's benefits would result in greater understanding of what the malware may accomplish. The malware aquarium fulfills the description and criteria in both circumstances.

7.2 Research in Perspective

The malware aquarium concept can definitely contribute to the malware analysis field. The implementation's extensible architecture proved itself invaluable when adding and changing components to fill in the blind spots discovered during research. This is quite welcome, since continuing to use this proof of concept for researching other malware types, requires it to be adaptable. Building it from exclusively open-source software allows anyone to customize it for their use case. Furthermore, it can quickly incorporate new technology which continues to develop at a rapid pace, not to mention the new malware discovered daily.

Not only does the quick and easy adaptability of the system helps speed up feedback cycles, but it also helps with the execution of the experiments themselves. Resetting an environment of potentially tens of machines can take just minutes and is fully automated. The standardized build system enables automated rebuilds of images whenever a new vulnerability is found or a new set of patches is released. This creates an entirely new generation of images ready for testing with minimal manual intervention. All of this results in saving time, allowing for more experiments and possibly increasing their diversity. Alternatively, that extra time can be used to simply extend the run length of the experiments as the sandboxes are no longer time-bound.

Combining these properties with the ability for deploying multiple systems and multiple types of malware simultaneously can potentially uncover unseen interactions between them. This potential grows if additional research is done to extend the current detection methods for different TTP or experiment with new approaches for the current ones. Allowing the malware to really flourish, propagate and develop may very well be the only way to research specific malware types.

The provided detection methods can be considered a subset on their own because they each provide a large number of capabilities which can be combined. An example of such a possibility is the capability of Sysmon [60] to save deleted files within the machine which can be combined with the GRR [84] feature to extract data. This allows the analyst to further investigate the deleted file that the malware removed without redeploying

and interacting with the machine itself.

Therefore, a malware aquarium offers the possibility for different malware types to flourish while being analyzed and investigated. The current proof of concept, even if still limited, shows these capabilities during its completed experiments. Thus, it can be used as an initial step towards building a full-fledged malware aquarium which provides the entire TTP coverage and maintains its flexibility and iterability.

7.3 Challenges

Taking the project's complexity into consideration, multiple challenges were encountered while building the malware aquarium. Out of all of them, the following two stood out the most.

Unexpected New Default Behaviour The first challenge was related to the Ubuntu 22.04 operating system that the malware aquarium's host machine was utilizing. This new version of the operating system introduces two new behavioural defaults within the systemd-networkd components when compared to the 20.04 version. These new components are `ManageForeignRoutingPolicyRules` and `ManageForeignRoutes` which remove any routing rules or routes that are not managed by that system whenever an interface reconfiguration was triggered. There are multiple reasons for these to happen, with some examples being the execution of `netplan apply` and even when a link would go down. This was a problem in this research because of Kubernetes-specific rules and routes that were installed in the host networking namespace which are not managed by systemd-networkd. Therefore, network connectivity was seemingly lost at random. The solution to the problem was to disable the two newly introduced options.

Windows Familiarity Another challenge was our unfamiliarity with the Windows 10 environment. The trial image used for Windows 10 [85] generated a large amount of ambiguous logs from contacting the Bank of America website, DigiCert and other random domain names such as "wpad". This made it hard to distinguish between normal and abnormal behaviour. Additionally, the inability to fully disable Windows updates meant that the machines could introduce unwanted system changes. Not only was it a challenge to let Windows 10 behave in a predictable manner but the mechanism Windows provides to prepare these changes was complicated and time consuming. With each added change our build times would increase, sometimes seemingly disproportionate to the complexity of the change we applied. This made the feedback cycle loop tedious which was amplified by the difficulty of debugging this process. All of this could be attributed to our unfamiliarity with Windows and its processes. None the less, it proved a time drain and slowed us down significantly reducing the overall time we could spend on the actual experiments.

8 Limitations and Future Work

8.1 Limitations

There are three significant constraints which can be identified within the project. The first one relates to the fact that the resulting product is in the early stages of development. Thus, the implemented malware aquarium still requires development before being considered a full-fledged product. It is still limited to the detection methods presented and the behaviour resulted from the analysis of the chosen malware types with their picked tactics.

The second present restriction is sample availability. As the current product goes by the concept of open source, it utilizes malware repositories which are open to the public. Such an example is the MalwareBazaar [86] platform. Thus, the extraction functionality is limited to the resources that these platforms provide.

Lastly, the current solution was tested only on a desktop solution, however, it can be extended to headless servers and even to solutions such as Internet of Things [87], industrial control systems [88] or mobile devices [89].

8.2 Future Work

Aside from the work that might be done by expanding on the mentioned restrictions, there are several additional parts of the research that can be investigated more thoroughly. A first development will be the extension of the malware aquarium coverage when it comes to malware types and TTP. This will help better secure and structure the aquarium turning it into a full-fledged solution. Additionally, a deep dive into anti-evasion or

anti-virtualization techniques used by malware and the implementation of these techniques in the current solution, will make sure that malware will not be aware that it is in an analysis environment. A starting point for the described development can be the "Malware Analyst CookBook" by Matthew Richard et.al. [90] where multiple malware analysis known techniques are described in detail.

The operating system support is another addition to the specified model. The present solution was created for Windows 10, but the same concept may be adapted for a variety of other operating systems. Older versions of Windows (XP, 7), server versions, or even non-Windows operating systems such as Linux-based ones are examples of such operating systems (Ubuntu, Fedora). The only aspects that are lacking from these sections are the porting of the Windows 10 version to the other operating system and ensuring that all of the existing components are available.

The product's analysis part may be improved in terms of usability and user experience (UX). There are now various tools where a user may look to acquire a complete picture of the infection. As a result, having a system with a single pane of glass would facilitate the analysis of massive log volumes. Furthermore, the product's speed may be increased to allow for quicker image creation, boot time, and handling of massive volumes of log data.

When it comes to comparing the current proof of concept with other products, a more elaborate analysis could be performed to include premium and open-source solutions. An example could be to investigate how the Cuckoo open-source sandbox [91] compares to the malware aquarium from an architecture and even analysis point of view.

Finally, integrating with other tools such as VirusTotal [92] or Open-Source Intelligence Technology (OSINT) to enrich the information discovered within the aquarium can substantially improve the attribution and quicker assessment of threats and their behaviours within the aquarium.

9 Conclusion

Malicious software's capabilities and features are becoming increasingly complex to combat using current security measures. The most prominent property is lateral movement which is used to increase coverage and spread the infection. Recent cases of Ransomware (Ryuk [1]), Trojans (TrickBot [2]) and Wipers (HermeticWiper [3]) all attest to this. To analyze malware behaviour, security experts use sandboxes, ranging from premium tiers to freemium and open source. The current free online sandboxes contain limitations which do not reveal the entire infection chain. Time constraints, single machine design, limited number of simultaneous deployments and other factors all hamper malware research in some form. This project takes the initial steps towards a solution for these problems by implementing a concept called the malware aquarium. This is defined as a reachable network with interconnected machines where malware can flourish while it is being monitored.

This research implements a proof of concept of a malware aquarium based on sandbox and secure architecture requirements enhanced by an analysis of two malware types, namely Trojans and Worms, with a focus on four MITRE ATT&CK [18] tactics. The aquarium's design and implementation builds on an open-source design philosophy and have a high degree of flexibility and adaptability. Virtualization and containerization are the core technologies for providing sandbox functionality. Additionally, monitoring capabilities for network, processes and system modification detection are provided.

To validate the completed product, an analytical comparison was performed between the malware aquarium and two free online sandboxes, Joe Sandbox [12] and Triage [13]. The malware samples utilized were among the malware categories specified, in this case, TrickBot and HermeticWizard, which also fit the sample criteria. In both situations, the malware aquarium offered more information about the malware's behaviour and actions. In the case of the HermeticWiper, the entire propagation process and proof of wiping were visible, while the free online sandboxes provided partial detection of the malware behaviour or none at all. More importantly, the aquarium's extensibility allowed the researchers to add components and assess even minor parameters such as CPU, disk consumption, and infection order. The TrickBot study surfaced more indicators of compromise and allowed for a lengthier analysis period when compared to the other sandboxes, which is critical in investigating a C2-based and stealth malware like TrickBot.

Therefore, the malware aquarium proved to be an improvement towards the analysis of malicious software compared to the free used online sandboxes. This may be observed in the benefits it provided by leveraging a network of machines for analysis, being easily scalable and expandable, the removal of time constraints and to allow ad hoc introspection throughout the investigation. However, the supplied solution is still a proof of concept that may be further expanded to include broader malware type analysis, wider operating system coverage, and a better user experience when it comes to malware behaviour analysis. Moreover, the entire

development process was based on a self-created methodology, from the initial exploration of the requirements to the point where the implementation is validated. This resulted in the successful production of a solution that can be easily replicated and iterated upon adapting to newly emerging technology and malware.

10 Acknowledgements

We wish to extend our gratitude to our teachers and lecturers from the University of Amsterdam for their guidance, patience and insight throughout the courses. We would also like to sincerely thank our supervisors for allowing us to build off of their knowledge and for providing us with valuable feedback during our research.

Finally, special thanks to Randall Munroe whose xkcd comic *Network* [93] inspired this paper's subject. A reproduction can be found in Appendix E.

References

[1] Marc EliasDelPozzo. "Ryuk Ransomware Now Targeting Webservers". In: (2021). URL: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-ryuk-ransomware-targeting-webservers.pdf.

[2] Ruveyda Celik, Ali Gezer. "Behavioral Analysis of Trickbot Banking Trojan with its New Tricks". In: (2019). URL: http://kkgpublications.com/wp-content/uploads/2019/12/ijtes.5.10004-3.pdf/.

[3] ESET Research. "IsaacWiper and HermeticWizard: New wiper and worm targeting Ukraine". In: (2022). URL: https://www.welivesecurity.com/2022/03/01/isaacwiper-hermeticwizard-wiper-worm-targeting-ukraine/.

[4] Cuckoo. "Cuckoo Sandbox". In: (2018). URL: https://cuckoo.cert.ee/.

[5] AnyRun. "AnyRun interactive malware Hunting service". In: (2022). URL: https://any.run/.

[6] ScienceDirect. "Honeynets". In: (2010). URL: https://www.sciencedirect.com/topics/computer-science/honeynets.

[7] Cristine Hoepers et. al. "Honeynets Applied to the CSIRT Scenario". In: (2003). URL: https://www.cert.br/docs/papers/hnbr-first2003.pdf.

[8] International Telecommunication Union. "SERIES X: DATA NETWORKS, OPEN SYSTEM COMMUNICATIONS AND SECURITY". In: (2020). URL: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.1218-202010-I!!PDF-E&type=items.

[9] Adrian Sanabria. "Malware Analysis: Environment Design and Artitecture". In: (2007). URL: https://www.giac.org/paper/gcih/1202/malware-analysis-environment-design-artitecture/109281.

[10] NCSC. "Secure design principles". In: (2019). URL: https://www.ncsc.gov.uk/collection/cyber-security-design-principles.

[11] Christopher C. "A Survey of Secure Architectural Principles". In: (2015). URL: https://www.osti.gov/biblio/1504842.

[12] JoeSandbox Cloud Basic. "Deep Malware Analysis". In: (2022). URL: https://www.joesandbox.com/.

[13] Hatching. "Triage". In: (2022). URL: https://tria.ge/.

[14] CrowdStrike. "Hybrid Analysis". In: (2022). URL: https://www.hybrid-analysis.com/.

[15] "Kerckhoffs's principle". In: (2022). URL: https://en.wikipedia.org/wiki/Kerckhoffs%27s_principle.

[16] Tim van Dijk. "Stealthy and in-depth behavioral malware analysis with Zandbak". In: (2019). URL: https://www.ru.nl/publish/pages/769526/z6_timvandijk_masterthesis.pdf.

[17] Deciso B.V. "Secure Your Network with ease". In: (2022). URL: https://opnsense.org/.

[18] MITRE. "MITRE ATTCK". In: (2022). URL: https://attack.mitre.org/.

[19] MITRE. "MITRE ATTCK® Navigators". In: (2022). URL: https://mitre-attack.github.io/attack-navigator/.

[20] MITRE. "Comparing Layers in ATTCK Navigator". In: (2021). URL: https://attack.mitre.org/docs/training-cti/Comparing%5C%20Layers%5C%20in%5C%20Navigator.pdf/.

[21] Puplesec. "The Ultimate List Of Stats Data, Trends For 2022". In: (2022). URL: https://purplesec.us/resources/cyber-security-statistics/#Malware.

[22] Dawid Laka. "Malware: Types, Analysis and Classification". In: (2022). URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4036836.

[23] Alison Grace Johansen. "What is a Trojan? Is it a virus or is it malware?" In: (2020). URL: https://us.norton.com/internetsecurity-malware-what-is-a-trojan.html.

[24] Malwarebytes. "2022 Threat Review". In: (2022).
URL: https://www.malwarebytes.com/resources/malwarebytes-threat-review-2022/mwb_threatreview_2022_ss_v1.pdf.

[25] James Maude. "Malware Threat Report 2021". In: (2021).
URL: https://assets.beyondtrust.com/assets/documents/BT_WhitePaper-Malware-Threat-Report-2021.pdf.

[26] Brad Duncan. "Emotet Summary: November 2021 Through January 2022". In: (2022).
URL: https://unit42.paloaltonetworks.com/emotet-malware-summary-epoch-4-5/.

[27] Att&CK MITRe. "Emotet". In: (2020). URL: https://attack.mitre.org/software/S0367/.

[28] OLEG KUPREEV. "Trickbot module descriptions". In: (2021).
URL: https://securelist.com/trickbot-module-descriptions/104603/.

[29] MalwareBytes. "Trojan.TrickBot". In: (2021).
URL: https://blog.malwarebytes.com/detections/trojan-trickbot/.

[30] MITRE ATTCK. "TrickBot". In: (2021). URL: https://attack.mitre.org/software/S0266/.

[31] Saray Mh. Doktor Adnan. "DRIDEX BANKING TROJAN". In: (2020).
URL: https://gaissecurity.com/assets/uploads/EN-Dridex-banking-trojan.pdf/.

[32] MITRE ATTCK. "Dridex". In: (2021). URL: https://attack.mitre.org/software/S0384/.

[33] Maxat Akbanov et al. "WannaCry Ransomware:Analysis of Infection, Persistence, Recovery Prevention and Propagation Mechanisms". In: (2019).
URL: https://www.il-pib.pl/czasopisma/JTIT/2019/1/113.pdf.

[34] Wikipedia. "Conficker". In: (2022). URL: https://en.wikipedia.org/wiki/Conficker.

[35] MITRE ATTCK. "Wannacry". In: (2019). URL: https://attack.mitre.org/software/S0366/.

[36] MITRE ATTCK. "HermeticWizard". In: (2022).
URL: https://attack.mitre.org/software/S0698/.

[37] Alfred Ng. "'Doomsday' worm uses seven NSA exploits (WannaCry used two)". In: (2017).
URL: https://www.cnet.com/news/privacy/doomsday-worm-eternalrocks-seven-nsa-exploits-wannacry-ransomware/.

[38] Wikipedia. "The Shadow Brokers". In: (2022).
URL: https://en.wikipedia.org/wiki/The_Shadow_Brokers.

[39] Rares Bratean and Rio Kierkels. "EthernalRocks". In: (2022). URL: https://github.com/rio/malware-aquarium/blob/main/MITRE%5C%20TTPS/EternalRocks.json.

[40] Rio Kierkels and Rares Bratean. "Malware Aquarium". In: (2022).
URL: https://github.com/rio/malware-aquarium.

[41] Check Point Blog. "September 2021's Most Wanted Malware: Trickbot Once Again Tops the List". In: (2021). URL: https://blog.checkpoint.com/2021/10/08/september-2021s-most-wanted-malware-trickbot-once-again-tops-the-list/.

[42] Nolind Raymond. "MalwareBazaar Database". In: (2022). URL: https://bazaar.abuse.ch/sample/236f4e149402cba69141e6055a113a68f2bd86539365210afb9861f4e2d3ad5f/.

[43] Anila Nadella, Inno Eroraha. "Trickbot Malware Analysis". In: (2022).
URL: https://blog.netsecurity.com/trickbot-malware-analysis/.

[44] Virus Total. "Trickbot". In: (2022). URL: https://www.virustotal.com/gui/file/236f4e149402cba69141e6055a113a68f2bd86539365210afb9861f4e2d3ad5f/detection.

[45] Aliaksandr Trafimchuk, Raman Ladutska.
"A Modern Ninja: Evasive Trickbot Attacks Customers of 60 High-Profile Companies". In: (2022).
URL: https://research.checkpoint.com/2022/a-modern-ninja-evasive-trickbot-attacks-customers-of-60-high-profile-companies/.

[46] vxunderground. "MalwareBazaar Database". In: (2022). URL: https://bazaar.abuse.ch/sample/a259e9b0acf375a8bef8dbc27a8a1996ee02a56889cba07ef58c49185ab033ec/.

[47] CISA. "Malware Analysis Report (AR22-115B)". In: (2022).
URL: https://www.cisa.gov/uscert/ncas/analysis-reports/ar22-115b.

[48] Rob Ewaschuk. "SRE Book Chapter 6 - Monitoring Distributed Systems". In: (2017). URL: https://sre.google/sre-book/monitoring-distributed-systems/.

[49] "QEMU: A generic and open source machine emulator and virtualizer". In: (2022). URL: https://www.qemu.org/.

[50] "Kernel Virtual Machine". In: (2022). URL: https://www.linux-kvm.org/.

[51] "libvirt: The virtualzation API". In: (2022). URL: https://libvirt.org/.

[52] Jason Geffner. "CVE-2015-3456". In: (2015). URL: https://www.cve.org/CVERecord?id=CVE-2015-3456.

[53] "containerd: An industry-standard container runtime". In: (2022). URL: https://containerd.io/.

[54] HashiCorp. "Packer". In: (2022). URL: https://www.packer.io/.

[55] "GNU Bash". In: (2014). URL: https://www.gnu.org/software/bash/.

[56] "curl: command line tool and library for transferring data with URLs (since 1998)". In: (2022). URL: https://curl.se/.

[57] "7-Zip". In: (2022). URL: https://www.7-zip.org/.

[58] "Samba: the standard Windows interoperability suite of programs for Linux and Unix". In: (2022). URL: https://www.samba.org/.

[59] Microsoft. "Windows Event Logs". In: (2022). URL: https://docs.microsoft.com/en-us/windows/win32/wec/windows-event-collector?redirectedfrom=MSDN.

[60] Microsoft. "Sysmon". In: (2022). URL: https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon.

[61] "Fluent Bit: An End to End Observability Pipeline". In: (2022). URL: https://fluentbit.io/.

[62] Rainer Gerhards. *The Syslog Protocol*. RFC 5424. Mar. 2009. DOI: 10.17487/RFC5424. URL: https://www.rfc-editor.org/info/rfc5424.

[63] "Elasticsearch". In: (2022). URL: https://www.elastic.co/elasticsearch/.

[64] "Loki". In: (2022). URL: https://grafana.com/oss/loki/.

[65] "iptables". In: (2022). URL: http://git.netfilter.org/iptables/.

[66] "Suricata". In: (2022). URL: https://suricata.io/.

[67] "CoreDNS". In: (2022). URL: https://coredns.io/.

[68] "ntopng: High-Speed Web-based Traffic Analysis and Flow Collection". In: (2022). URL: https://www.ntop.org/products/traffic-analysis/ntop/.

[69] "gRPC: A high performance, open source universal RPC framework". In: (2022). URL: https://grpc.io/.

[70] "Prometheus: From metrics to insight". In: (2022). URL: https://prometheus.io/.

[71] "OpenTelemetry: High-quality, ubiquitous, and portable telemetry to enable effective observability". In: (2022). URL: https://opentelemetry.io/.

[72] "Tempo". In: (2022). URL: https://grafana.com/oss/tempo/.

[73] "Grafana". In: (2022). URL: https://grafana.com/oss/grafana/.

[74] "Go: Build fast, reliable, and efficient software at scale". In: (2022). URL: https://go.dev/.

[75] "Kubernetes". In: (2022). URL: https://kubernetes.io/.

[76] "Kubernetes Custom Resources". In: (2022). URL: https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/.

[77] "KubeVirt". In: (2022). URL: https://kubevirt.io/.

[78] "k3s: Lightweight Kubernetes". In: (2021). URL: https://k3s.io/.

[79] FileInfo. "ActiveX Control". In: (2019). URL: https://fileinfo.com/extension/ocx#:~:text=An%5C%20OCX%5C%20file%5C%20contains%5C%20a,online%5C%20games%5C%2C%5C%20and%5C%20multimedia%5C%20viewers..

[80] Joe Sandbox. "HermeticWizard". In: (2022). URL: https://www.joesandbox.com/analysis/1011255.

[81] Triage. "HermeticWizard". In: (2022). URL: https://tria.ge/220704-xphxascgf8.

[82] Joe Sandbox. "TrickBot". In: (2022). URL: https://www.joesandbox.com/analysis/1024632.

[83] Triage. "TrickBot". In: (2022). URL: https://tria.ge/220705-gjwyzaeeaq.

[84] Google. "GRR Rapid Response". In: (2022). URL: https://github.com/google/grr.

[85] Microsoft. "Windows 10 21H2". In: (2021). URL: https://software-download.microsoft.com/download/sg/444969d5-f34g-4e03-ac9d-1f9786c69161/19044.1288.211006-0501.21h2_release_svc_refresh_CLIENTENTERPRISEEVAL_OEMRET_x64FRE_en-us.iso.

[86] Abuse CH. "MalwareBazaar". In: (2022). URL: https://bazaar.abuse.ch/.

[87] IoT-LAB. "IoT-LAB". In: (2022). URL: https://iot-lab.github.io/docs/getting-started/virtual-machine/.

[88] Peter Maynard and Kieran McLaughlin and Sakir Sezer. "ICS TestBed Framework". In: ().

[89] Google. "Android Emulation using QEMU". In: ().

[90] Michael Ligh, Steven Adair, Blake Hartstein, Matthew Richard. "Malware Analyst's Cookbook and DVD: Tools and Techniques for Fighting Malicious Code". In: (2010). URL: https://www.wiley.com/en-us/Malware+Analyst%5C%27s+Cookbook+and+DVD%5C%3A+Tools+and+Techniques+for+Fighting+Malicious+Code-p-9780470613030.

[91] Cuckoo. "Cuckoo Automated Malware Analysis". In: (2019). URL: https://cuckoosandbox.org/.

[92] VirusTotal. "VirusTotal". In: (2022). URL: https://www.virustotal.com/gui/home/upload.

[93] Randall Munroe. "Network". In: (Nov. 2007). URL: https://xkcd.com/350/.

11 Appendix

A Analyzed Tactics and Techniques

Analyzed Tactics and Techniques		
Tactic	**Technique**	**Subtechnique**
Execution	Command and Scripting Interpreter (T1059)	PowerShell (T1059.001), Visual Basic (T1059.005), Windows Command Shell (T1059.006)
	Native API (T1106)	
	Inter-Process Communication (T1559)	Component Object Model (T1559.001)
	Schedule Task/Job (T1053)	Schedule Task (T1053.005)
	System Services (T1569)	Service Execution (T1569.02)
	User Execution (T1204)	Malicious File (T1204.002), Malicious Link (T1204.001)
	Windows Management Instrumentation (T1047)	
Defense Evasion	Deobfuscate/Decode Files/Data (T1140)	
	File and Directory Permission Modification (T1222)	Windows (T1222.001)
	Hide Artifacts (T1564)	Hide Files/Directories (T1564.001)
	Impair Defenses (T1562)	Disable or modify tools (T1562.001)
	Indicator Removal on Host (T1070)	Clear Windows Event Logs (T1070.001)
	Masquerade (T1036)	Masquerade Task or Service (T1036.004), Match Legitimate Name or Location (T1036.005)
	Modify Registry (T1112)	
	Obfuscate Files of Information (T1027)	Software Packing (T1027.002)
	Pre-OS Boot (T1542)	Bootkit (T1542.003)
	Process Injection (T1055)	Dynamic-link Library Injection (T1055.001), Process Hollowing (T1055.012)
	Subvert Trust Controls (T1553)	Code Signing (T1553.002)
	System Binary Proxy execution (T1218)	Regsvr32 (T1218.010), Rundll32 (T1218.011)
	Valid Accounts (T1078)	Local Accounts (T1078.003)
	Virtualization/Sandbox Evasion (T1497)	Time Based Evasion (T1497.003)
Lateral Movement	Exploitation of Remote Services (T1210)	
	Lateral Tool Transfer (T1570)	
	Remote Service Session Hijacking (T1563)	RDP Hijacking (T1563.002)
	Remote Services (T1021)	Distributed Component Object Model(T1021.003), SMB/Windows Admin Shares (T1021.002), VNC (T1021.005)
Command and Control	Application Layer Protocol (T1071)	DNS (T1071.004), Web Protocols (T1072.001)
	Data Encoding (T1132)	Standard Encoding (T1132.001)
	Encrypted Channel (T1573)	Asymmetric Cryptography (T1573.002), Symmetric Cryptography (T1573.001)
	Fallback Channels (T1008)	
	Ingress Tool Transfer (T1105)	
	Non-Standard Port (T1571)	
	Proxy (T1090)	External Proxy (T1090.002), Multi-hop Proxy (T1090.003)
	Remote Access Software (T1219)	

B Internal Firewall Rules

```
apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
metadata:
  name: suricata-egress
  namespace: suricata
spec:
  # select all endpoints in this namespace
  endpointSelector: {}
  # ingress blocked
  ingress:
    - {}
  egress:
    # allow L3/L4 to the world
    - toEntities:
      - world
      toPorts:
      - ports:
        - port: "53"
          protocol: ANY
        rules:
          dns:
            - matchPattern: "*"
      - ports:
        - port: "443"
          protocol: ANY
      - ports:
        - port: "80"
          protocol: ANY
        rules:
          http:
            - method: ".*"
    # allow L3/L4 to the cluster DNS specifically to resolve monitoring domains
    - toEndpoints:
      - matchLabels:
          k8s:io.kubernetes.pod.namespace: kube-system
          k8s:k8s-app: kube-dns
      toPorts:
      - ports:
        - port: "53"
          protocol: ANY
        rules:
          dns:
            - matchPattern: "*.loki.svc.cluster.local"
            - matchPattern: "*.grr.svc.cluster.local"
    # allow sending logs to loki
    - toEndpoints:
      - matchLabels:
          k8s:io.kubernetes.pod.namespace: loki
          k8s:app: loki
      toPorts:
      - ports:
        - port: "3100"
          protocol: TCP
        rules:
          http:
            - method: "POST"
              path: "/loki/api/v1/push"
    # allow grr connectivity
    - toEndpoints:
      - matchLabels:
          k8s:io.kubernetes.pod.namespace: grr
          k8s:app: grr
      toPorts:
      - ports:
        - port: "8080"
          protocol: TCP
```

C Timeline HermeticWiper

Malware Aquarium Timeline - HermeticWiper		
HOST A		
Time	**Action**	**Log Source**
18:30:29	Execute the DLL file using "regsvr32.exe /i /s" command	Sysmon Event ID 1
18:30:32	SMB Scan on port 445	Sysmon Event ID 3
18:30:36	New file creation cehwda.ocx	Sysmon EventID 11
18:30:36	Execute cehwda.ocx	Sysmon Event ID 1
18:30:36	Driver created pvdr.sys	Sysmon Event ID 11
18:30:37	Add driver in Registry Key related to services	Sysmon Event ID 13
18:31:13	SMB Scan on port 445	Sysmon Event ID 3
18:35:34	New .ocx file dropped, cchebs.ocx	Sysmon Event ID 11
18:35:35	Execute cchebs.ocx with the IP of the neighbour "-i (IP address)"	Sysmon Event ID 1
18:35:37	DNS query towards the hostname of the second host	Sysmon Event ID 22
18:37:04	Wiping started by privileged service	Security Logs
18:40:43	Network connection on SMB from host B to host A	Sysmon Event ID 3
HOST B		
Time	**Action**	**Log Source**
18:35:35	DLL File dropped in C\Windows using SMB	Security Logs
18:35:35	Remove execution of the DLL file using WMI	Sysmon Event ID 1
18:35:41	OCX file created, cwhshg.ocx	Sysmon Event ID 11
Start repeating process described for Host A		

D Timeline TrickBot

Malware Aquarium Timeline - TrickBot		
Time	**Action**	**Log Source**
21:37:58	Start malicious executable	Sysmon Event ID 1
21:38:13	Start svchost.exe process by the malicious file	Sysmon Event ID 1
21:38:13	Stop the malicious executable process	Sysmon Event ID 1
21:38:13	New file created in C:\Users\¡User¿\AppData \Roaming	Sysmon Event ID 11
21:38:14	Execute newly created file and start process svchost.exe	Sysmon Event ID 1
21:38:15	Register new executable in Registry Key for compatibility	Sysmon Event ID 13
21:38:15	New file created in config \systemprofile \AppData \Roaming	Sysmon Event ID 11
21:39:03	DNS query towards myexternalip[.]com	Sysmon Event ID 22
21:45:30	SVChost.exe contacts C2 IP	Sysmon Event ID 3
21:47:42	MMC start task scheduler	Security log
21:47:50	Svchost.exe contacts C2 IP	Sysmon Event ID 3
21:47:56	Svchost.exe contacts C2 IP	Sysmon Event ID 3
...		
23:53:03	Svchost.exe contacts C2 IP	Sysmon Event ID 3

E XKCD 350: Network by Randall Munroe

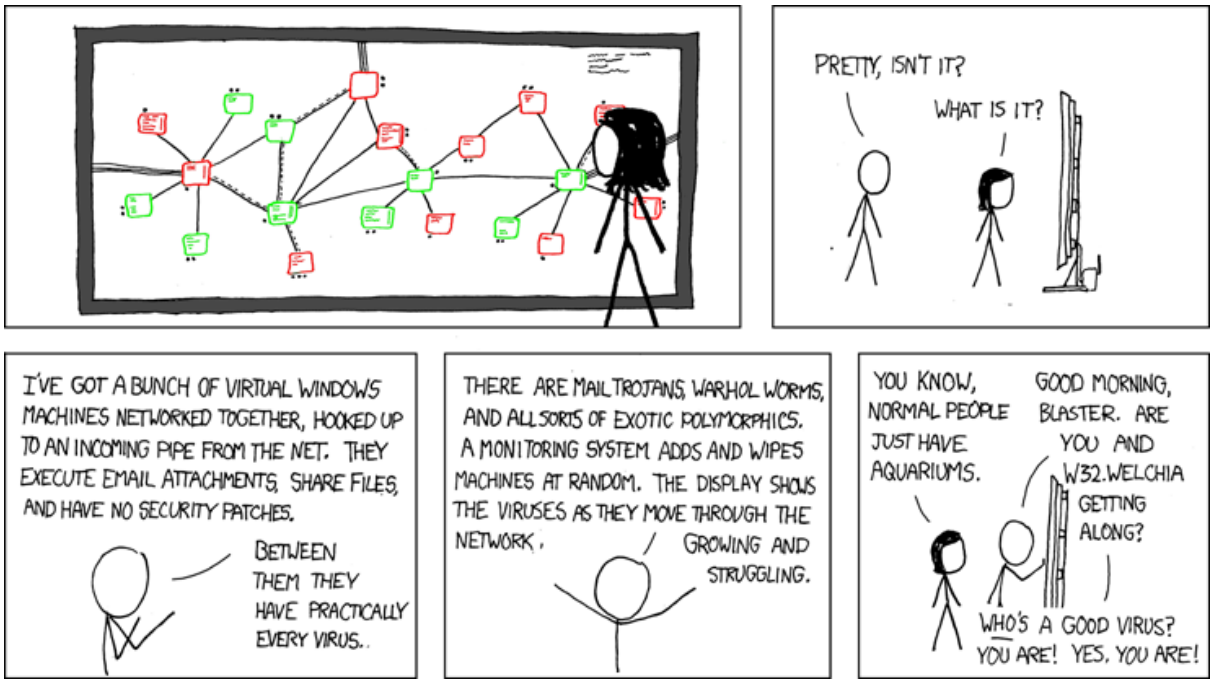

Figure 11: https://xkcd.com/350/ - Creative Commons Attribution-NonCommercial 2.5 License