Antivirus Detection of Containerized Malware in Linux Distributions

Author: Emily Stevenson, stevenson.j.emily@gmail.com
Advisor: *Tanya Baccam*

Accepted: 12/04/2023

Abstract

The use of containers exploded in popularity over the past decade and has exponential growth forecasted for years to come. There is a significant shortage of current research and documentation regarding antivirus detection of malware within container architectures, especially considering their prevalence in today's market. Understanding current capabilities is critical in network defense and building defense-in-depth strategies, as is understanding existing vulnerabilities and security blind spots. The research presented in this paper explores the effectiveness of antivirus software at detecting a malicious payload obfuscated only by its presence within container architecture.

1. Introduction

The use of containers in enterprise environments is a concept that has been around for many years. Some trace the origins of containers back to the 1960s with the advent of VM partitioning or back to the late 1970s with the development of chroot in Unix Version 7. Regardless of how far back the technology goes, containers took significant developmental strides in the early 2000s through 2013, when Docker first hit the market (Mell, 2023). Since then, its technology has only progressed in complexity and popularity. Today, projections estimate that the container market share will experience compound annual growth in the coming decade. According to a 2019 survey, the use of containers in production rose by 15% in a year, from 59% to 84% of respondents using containers ("CNCF Survey 2019", n.d.). Containers are not limited to a specific operating system (OS). They are already prevalent in Linux and Windows on-premises environments as well as in off-premises environments such as public cloud providers. As more and more applications move to containers — a behavior seen in organizations looking to conserve resources and increase scalability — it is reasonable to assume that the attack surface for containerized malware will also increase. With an estimated two-thirds of the market at risk, the consumer must understand the risks associated with container use and current antivirus software capabilities in safeguarding vulnerabilities and detecting compromises in a worst-case scenario. This paper will explore to what extent popular open-source and subscription antivirus software can detect malware-compromised containers.

1.1. Container Architecture

Container popularity is primarily due to its functionality — its ability to scale, migrate, and clone, as well as its ability to reduce strain on computing resources. Containers possess these features as a result of design. Additionally, the containers can easily be versioned, backed up, and duplicated, directly feeding into their functionality (Malik, 2019). Effectively, they are software programs built by a container engine that sits on top of a host OS. By not duplicating the OS, containers can remain lightweight,

requiring fewer computing resources. Once built, “containers encapsulate an application as a single executable package of software that bundles application code together with all of the related configuration files, libraries, and dependencies required for it to run” (“What is Containerization”, n.d.). The result of this is that an application can run inside of a container in isolation away from other applications and systems. While this application isolation is the goal of containerization, it is also the mechanism by which malware attempts to hide within a container to evade antivirus detection.

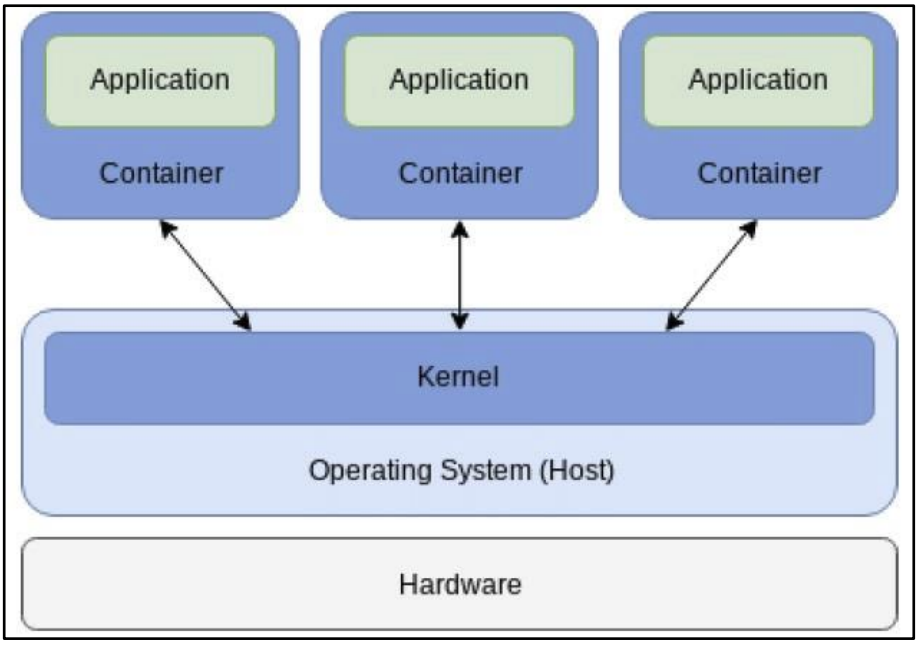


Figure 1: Container Architecture (Scolati, Fronza, El Ioini, Samir, Pahl, 2019)

1.2. Antivirus Software

With the increasingly dire need to detect malware infections in the shortest possible time, multiple security avenues are available to the network defender — one of which is endpoint security in the form of antivirus software. There are many different versions of commercially available antivirus software, plenty of which are free to download and install. The different versions have varying capabilities, from preventing ransomware and data loss to application control and web threat protection. Several factors, including OS, protection priorities, and budget, determine the specific flavor of

antivirus software chosen for a network or endpoint. Once software is selected and installed, "[it] works on one of two principles: either it scans programs and files as they enter your device and compares them to known viruses, or it scans programs already on your device, looking for suspicious behavior" (Vigderman, 2023). Given that a containerized application is isolated from the rest of the operating system, it stands to reason that antivirus software would not be able to scan inside of an existing container. To reinforce this point, scanning within containers is generally not listed as a capability of antivirus software. The mention of containers is often left out entirely in the overview of the antivirus capabilities.

2. Research Method

2.1. Hypothesis

Due to the nature of container operation in the modern computing environment and the scanning methodology employed by antivirus software, it is assumed that any malware installed and running inside a container will evade antivirus detection on Linux distributions.

2.2. Test Variables

In order to test this hypothesis, controlled experiments will run against two virtual machines (VMs), VM A and VM B. As the intended research is to assess the current detection capabilities of antivirus software concerning whether a container will successfully obfuscate malware from antivirus scans, VM A will have a Docker container installed. A malicious binary is then installed on both VMs. On VM A, the binary will be installed directly inside the Docker container, and on VM B, the virtual machine's home directory will host the malware. As the container is the variable of the experiment, the results will determine if the presence of the container prevents the antivirus software from detecting the malicious binary.

2.3. Scope

There are many ways malware might attempt obfuscation. This paper does not explore any external capabilities, methodologies, or code malware might employ to avoid detection. It also does not examine malicious actor's tactics to inject malware into a container and how such activity might be detected. The research presented in this paper solely focuses on the capability of antivirus software to detect malicious binaries, both at rest and running, within a container.

An extremely common binary was also selected for this research: Meterpreter, a Metasploit attack payload and remote access trojan. Due to the prevalence and popularity of Meterpreter, it is often used as the baseline for malware detection as it is so heavily signatured. Selecting such prevalent malware ensures the antivirus software will detect the malware. The experiment will focus solely on the container's role in obfuscating a malicious binary by eliminating this possibility.

2.4. Virtual Environment

The virtual environment built to conduct the experiments consisted of two Ubuntu 23.10.1 VMs, VM A and VM B, built in VMWare Workstation Pro. On VM A, Docker Desktop 23.0.5 was installed. Docker's use dominates approximately 80% of the market share; therefore, it is the selected container engine for the experiments ("Market Share of Docker"). As one of the most popular container engines, using Docker makes the findings of the experiments relevant to the majority of container users. After the completed container setup on VM A, the malware intended for the experiment was configured.

As Meterpreter is a Metasploit module, it was necessary first to build the payload using msfvenom, which was done on a Kali Linux VM. The following command was run: msfvenom -a x64 --payload linux/x64/meterpreter_reverse_tcp --platform linux --format elf LHOST=192.168.80.80 LPORT=8080 > docker_payload_01.elf to generate the payload.

Figure 2: Creation of the Meterpreter Payload

After the malicious payload generation, it was copied to both VM A and VM B. On VM A, the Meterpreter payload was saved within the preconfigured container. On VM B, the payload was saved on the Desktop. At this point, a snapshot was taken of both VMs so they could be reverted after each iteration of the experiment.

Six different types of antivirus software were installed separately and used to run the individual experiments. ClamAV, Sophos, ESET, Rootkit Hunter, Bitdefender GravityZone, and Kaspersky were used to test the hypothesis. ClamAV, Sophos, ESET, and Rootkit Hunter were chosen because they are free, open-source, and widely available for download by any user. Additionally, they do not have any verbiage indicating or acknowledging an ability (or inability) to scan inside a container. In effect, this makes internal container detection a blind spot for the customer. Bitdefender GravityZone and Kaspersky were selected as paid-for antivirus subscription models advertised predominantly as enterprise service solutions. Each antivirus software was installed on both VM A and VM B. The output of the antivirus scans contributed to the determination of hypothesis accuracy.

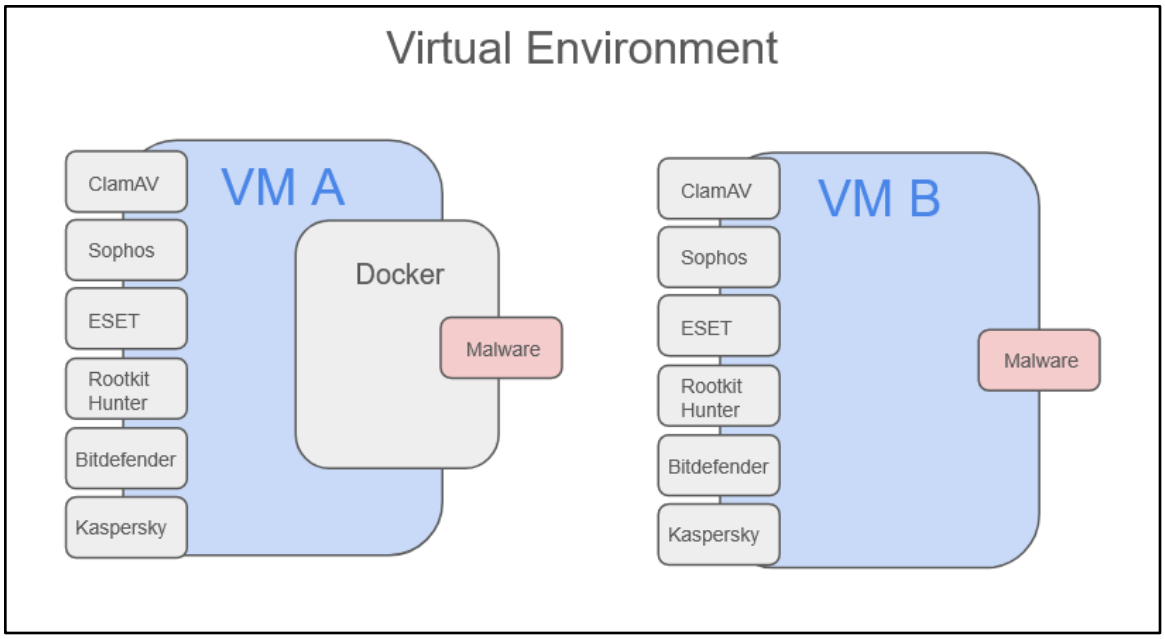


Figure 3: Virtual Environment

2.5. Research Execution

The research experiments were conducted with both the control (non-container) and variable (container) VMs. After the environment setup and the antivirus software installation was completed, the first scan was run, and the findings documented. The next round of experimenting required the execution of the malicious binary to verify if the antivirus software could detect the executed payload. At this point, the binary was executed, and the second scan was initiated, the findings of which were documented upon completion. Once the experiments were completed, the VMs were returned to the snapshot taken during the environment setup. The process was repeated for each of the remaining antivirus software.

3. Findings

3.1. ClamAV

3.1.1. VM A Container Scans

The first scan was run after the download and installation of Clam AV on VM A. For this first scan, the Meterpreter payload was loaded into the home directory of the running container. To differentiate between detecting a binary at rest and a running binary, the first AV scan is run before executing the Meterpreter payload.

The ClamAV scan was recursive of the entire root directory. After completing, it was found that out of the 279,034 files scanned, there was one malicious file detected in /var/lib/docker/overlay2/5e373ca3b8cf058b8e5ac94457783e650bbbd62e42e2180592322 6b4ca1ae1fd/diff/home/docker_payload.01.elf. This first finding is contrary to the hypothesis that modern antivirus software is not capable of detecting malware inside a container. Figure 4 shows the 'virus found' message as well as the data output from the completed scan:

```
LibClamAV Warning: fmap_readpage: pread fail: asked for 4072 bytes @ offset 24, got 0
LibClamAV Warning: fmap_readpage: pread fail: asked for 4094 bytes @ offset 2, got 0
LibClamAV Warning: fmap_readpage: pread fail: asked for 4089 bytes @ offset 7, got 0
LibClamAV Warning: fmap_readpage: pread fail: asked for 4091 bytes @ offset 5, got 0
/var/lib/docker/overlay2/5e373ca3b8cf058b0e5ac94457783e650bbbd62e42e21805923226b4ca1ae1fd/diff/home/docker_pa
yload_01.elf: Unix.Trojan.Generic-9908886-0 FOUND
```

Figure 4: ClamAV Scan #1 Virus Found Output on VM A

```
----------- SCAN SUMMARY -----------
Known viruses: 8678669
Engine version: 0.103.9
Scanned directories: 63121
Scanned files: 279034
Infected files: 1
Total errors: 42915
Data scanned: 11495.06 MB
Data read: 25891.25 MB (ratio 0.44:1)
Time: 4908.351 sec (81 m 48 s)
Start Date: 2023:11:18 15:27:02
End Date:   2023:11:18 16:48:50
```

Figure 5: ClamAV Scan #1 Output on VM A

The second scan conducted against VM A was after the Meterpreter payload was executed. The scan was initiated once it was verified that the payload was running. Similar to the previous scan, the AV identified the malicious file. The following figures

show the ClamAV output of the 'virus found' message as well as the completed scan results:

```
LibClamAV Warning: fmap_readpage: pread fail: asked for 4072 bytes @ offset 24, got 0
LibClamAV Warning: fmap_readpage: pread fail: asked for 4094 bytes @ offset 2, got 0
LibClamAV Warning: fmap_readpage: pread fail: asked for 4089 bytes @ offset 7, got 0
LibClamAV Warning: fmap_readpage: pread fail: asked for 4091 bytes @ offset 5, got 0
/var/lib/docker/overlay2/5e373ca3b8cf058b0e5ac94457783e650bbbd62e42e21805923226b4ca1ae1fd/diff/home/docker_pa
yload_01.elf: Unix.Trojan.Generic-9908886-0 FOUND
/var/lib/docker/overlay2/5e373ca3b8cf058b0e5ac94457783e650bbbd62e42e21805923226b4ca1ae1fd/merged/home/docker_
payload_01.elf: Unix.Trojan.Generic-9908886-0 FOUND
```

Figure 6: ClamAV Scan#2 Virus Found Output on VM A

```
----------- SCAN SUMMARY -----------
Known viruses: 8678773
Engine version: 0.103.9
Scanned directories: 63761
Scanned files: 279627
Infected files: 2
Total errors: 43024
Data scanned: 11573.61 MB
Data read: 25967.11 MB (ratio 0.45:1)
Time: 4504.865 sec (75 m 4 s)
Start Date: 2023:11:19 07:29:11
End Date:   2023:11:19 08:44:16
```

Figure 7: ClamAV Scan #2 Output on VM A

3.1.2. VM B Non-Container Scans

The same version of the antivirus software, ClamAV, that was downloaded and installed on VM A was also downloaded and installed on VM B. As this VM is our control for the experiment, no container is present. Instead, the Meterpreter payload was saved directly on the Desktop of the VM. The first scan is of the binary at rest, prior to execution. Shortly after the scan started, ClamAV printed its virus FOUND message to stdout (see Figure 8). That ClamAV found the malware on VM B is in line with expectations, as there was no attempt to obfuscate the malicious payload.

```
LibClamAV Warning: fmap_readpage: pread fail: asked for 4091 bytes @ offset 5, got 0
LibClamAV Warning: fmap_readpage: pread fail: asked for 4072 bytes @ offset 24, got 0
LibClamAV Warning: fmap_readpage: pread fail: asked for 4094 bytes @ offset 2, got 0
LibClamAV Warning: fmap_readpage: pread fail: asked for 4089 bytes @ offset 7, got 0
LibClamAV Warning: fmap_readpage: pread fail: asked for 4091 bytes @ offset 5, got 0
/home/emily/Desktop/docker_payload_01.elf: Unix.Trojan.Generic-9908886-0 FOUND
```

Figure 8: ClamAV Scan #1 Virus FOUND Output on VM B

```
----------- SCAN SUMMARY -----------
Known viruses: 8678669
Engine version: 0.103.9
Scanned directories: 60988
Scanned files: 266220
Infected files: 1
Total errors: 42468
Data scanned: 11785.38 MB
Data read: 25115.46 MB (ratio 0.47:1)
Time: 4768.149 sec (79 m 28 s)
Start Date: 2023:11:18 15:27:31
End Date:   2023:11:18 16:47:00
```

Figure 9: ClamAV Scan #1 Output on VM B

The second scan run on VM B was initiated after executing the Meterpreter payload. As evidenced by the first scan, ClamAV had no difficulty identifying the malicious binary in this secondary scan. Whether the binary was running or at rest had no perceivable impact on the antivirus scan results. The ClamAV output from the second scan is shown in Figures 10 and 11.

```
LibClamAV Warning: fmap_readpage: pread fail: asked for 4091 bytes @ offset 5, got 0
LibClamAV Warning: fmap_readpage: pread fail: asked for 4072 bytes @ offset 24, got 0
LibClamAV Warning: fmap_readpage: pread fail: asked for 4094 bytes @ offset 2, got 0
LibClamAV Warning: fmap_readpage: pread fail: asked for 4089 bytes @ offset 7, got 0
LibClamAV Warning: fmap_readpage: pread fail: asked for 4091 bytes @ offset 5, got 0
/home/emily/Desktop/docker_payload_01.elf: Unix.Trojan.Generic-9908886-0 FOUND
```

Figure 10: ClamAV Scan #2 Virus Found Output on VM B

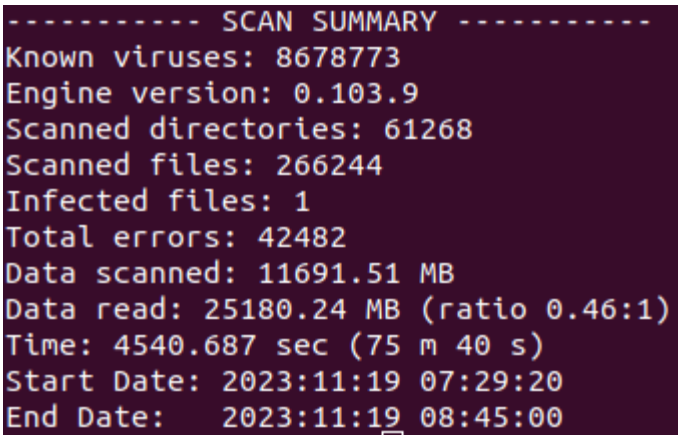


Figure 11: ClamAV Scan #2 Output on VM B

After completing all scanning in VMs A and B, both machines reverted to their original snapshots.

3.2. Sophos

3.2.1. VM A Container Scans

Sophos was downloaded and installed onto VM A when the first manual scan began. The VM had previously reverted to its snapshot, so the malicious binary was located in the same Docker directory as in the experiment's previous iteration. Additionally, the Docker container was running throughout the scan.

Upon completing the scan, the antivirus returned that no viruses were discovered on the system. It did not alert that there were any threats on the system, or do anything to quarantine the binary. The results of this scan, while in direct contrast with the findings from the ClamAV scan, align with the earlier stated hypothesis. In this instance, the malicious binary being saved within the container was enough to obfuscate it from antivirus detection. Figure 12 displays the output from this first scan:

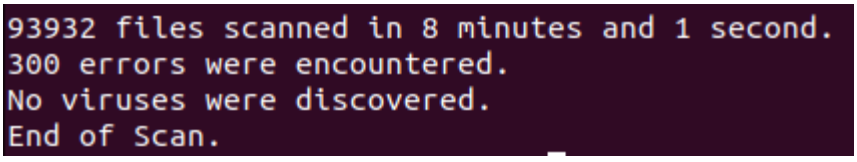


Figure 12: Sophos Scan #1 Output on VM A

For the next step in the experiment the Meterpreter payload was executed inside the container prior to running the second scan. Despite this, Sophos was still unable to detect the payload and reported no virus detection.

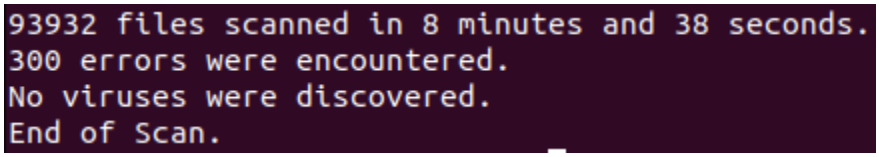


Figure 13: Sophos Scan #2 Output on VM A

3.2.2. VM B Non-Container Scans

After installing Sophos on VM B, it became immediately apparent that the experiment's outcome would be different. As soon as Sophos was running on the system, an alert message popped up on the screen stating that there was a detected threat. Sophos instantly detected the Meterpreter payload and immediately eliminated the infected binary. Figure 14 displays the threat alert message. Behind the alert message, the output from the first scan displays that no viruses were discovered. This lack of discovery by the scan is because the threat had been mitigated by the time the antivirus scanned the original location of the malicious binary.

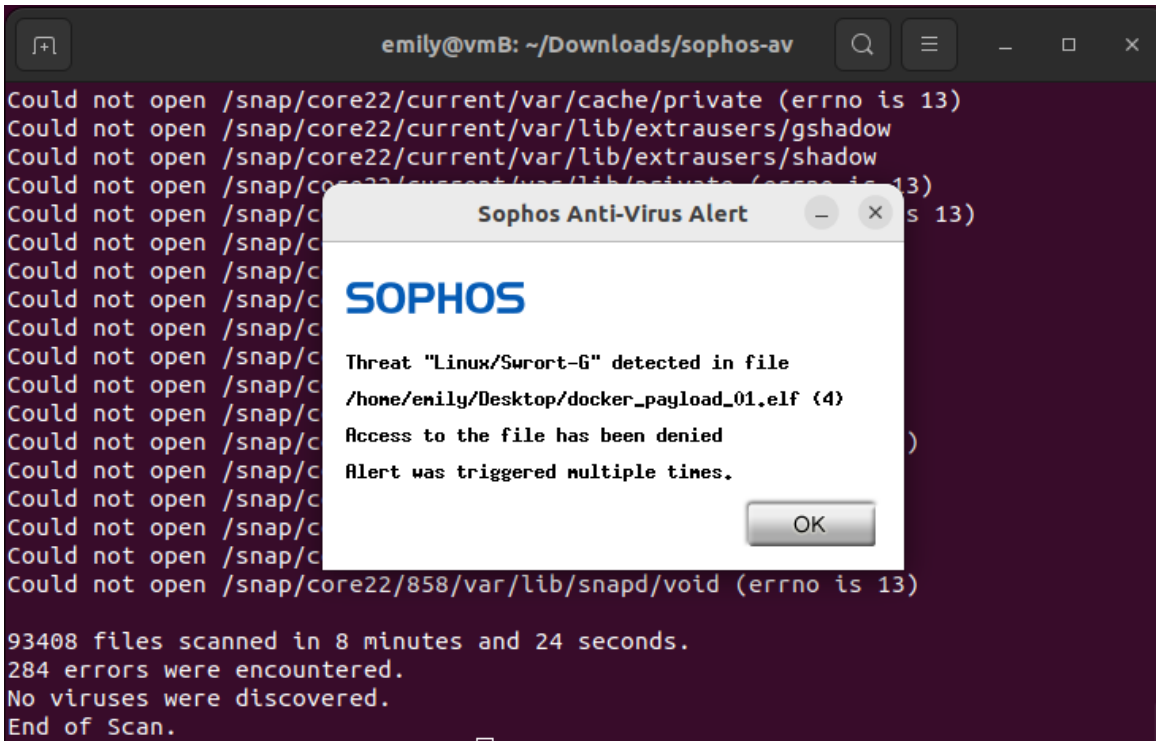


Figure 14: Sophos Scan #1 Output on VM B

Sophos was highly effective at identifying and mitigating the threat on VM B. So much so that immediately after Sophos' installation, the malicious binary was eliminated. Due to this efficiency, the version of the experiment where the antivirus scans for an executed binary could not be performed.

Upon completion of the Sophos scans, the VMs reverted to their snapshots in preparation for the installation of the following antivirus.

3.3. ESET

3.3.1. VM A Container Scans

The next antivirus examined was ESET for Ubuntu. This software only supported specific versions of Ubuntu, namely Ubuntu 20.04 - so VM A and VM B were rebuilt using the earlier version of the OS. Once the environment was replicated with the new OS version, ESET was installed on both VMs. A custom scan of the entire root directory

was started on VM A. Similar to the previous iterations of the experiment, this first scan was completed prior to binary execution. Upon scan completion, the details showed that ESET did not detect the binary at rest within the container.

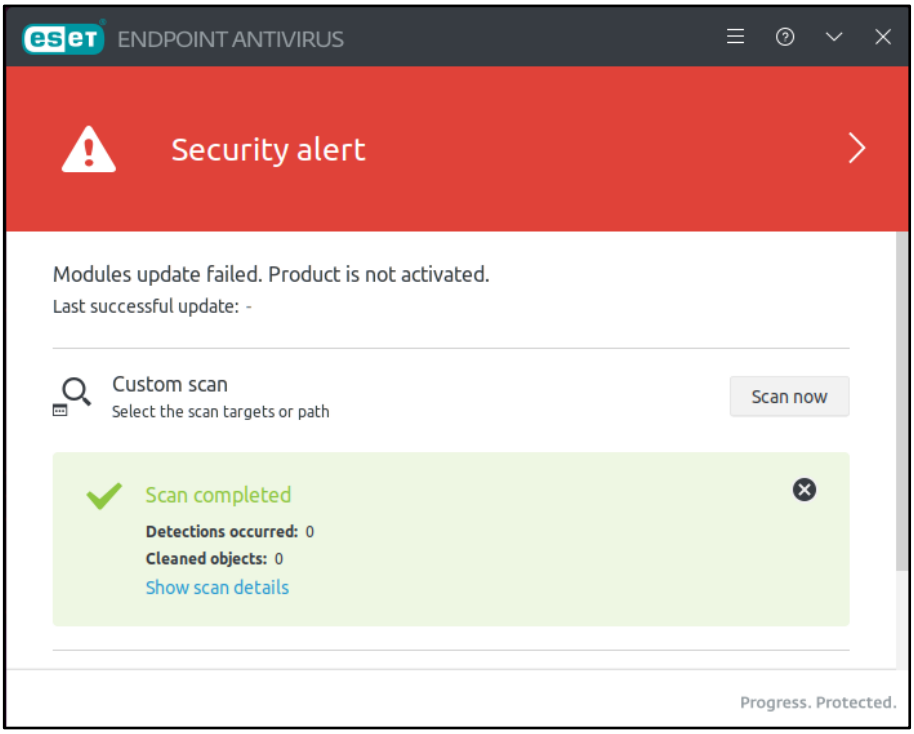


Figure 15: ESET Scan #1 Output on VM A

For the second round of the experiment, the Meterpreter payload was executed within the container. Interestingly, as soon as the binary was executed, a ‘Threat Found’ popup appeared on the screen of the VM (see Figure 16). ESET was able to identify the binary upon execution even though it did not identify it at rest. Figure 17 shows the series of commands executed in the container that were the catalyst to the ‘Threat Found’ notification. Additionally, it should be noted that although ESET identified the threat and prevented the binary execution, it did not quarantine the malware.

Figure 16: ESET Threat Found Notification on VM A

```
emily@ubuntu:~$ sudo docker attach whitepaper
/home # ls
docker_payload_01.elf
/home # chmod +x docker_payload_01.elf
/home # ./docker_payload_01.elf
/bin/sh: ./docker_payload_01.elf: Operation not permitted
/home # ls
docker_payload_01.elf
/home #
```

Figure 17: Docker Command to Execute Meterpreter on VM A

3.3.2. VM B Non-Container Scans

ESET's detection of the malicious payload was very different on VM B, where there was no container to obfuscate the binary. On this VM, the malware was saved to the Desktop, and as soon as ESET was running on the machine, it detected the malicious binary and removed the threat from the system in its entirety. Figure 18 displays the 'Threat Removed' message on the VM. Due to the removal of the malicious binary by ESET, it is no surprise that the scan detected no remaining threats on the system.

Figure 18: ESET Threat Found Notification on VM B

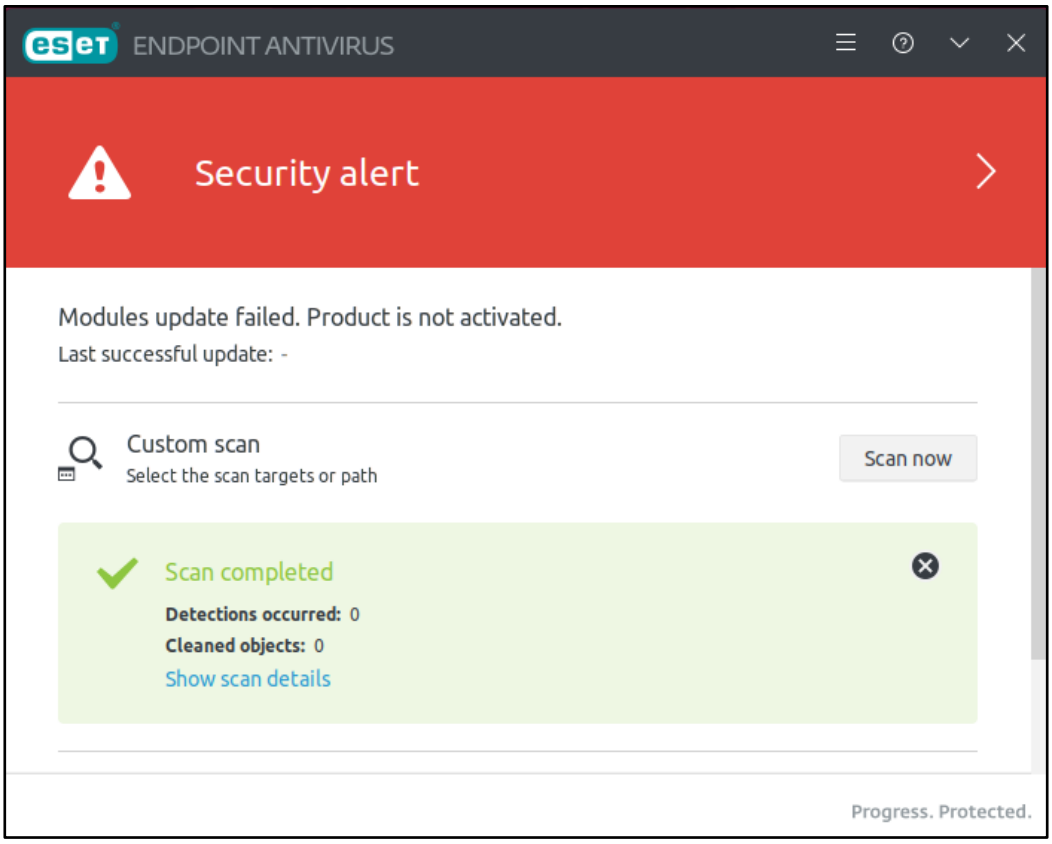


Figure 19: ESET Scan #2 Output on VM B

After the conclusion of the ESET experiments, the VMs reverted to their original snapshot in preparation for the next round of experimentation.

3.4. Rootkit Hunter

The last free version of software used for the experiments was Rootkit Hunter. This software failed to detect the malicious payload in both the control and variable experiments, regardless of whether the binary was running or at rest. To force the detection of the malware, the binary was moved to a new directory on VM B, /usr/bin, as this directory was prevalent in many of Rootkit Hunter's scans. The theory tested was that the Desktop was an area that the antivirus did not scan. Despite verifying that the usr/bin was a scanned directory in the rkhunter.conf file, the payload still failed to be detected. Figures 20 and 21 show the scan outputs from VM A and VM B. The scans all

identified suspect files; however, when reviewing rkhunter.log it became clear that the suspect files were false positives. None of the scans correctly identified the Meterpreter payload in the container, Desktop, or usr directory.

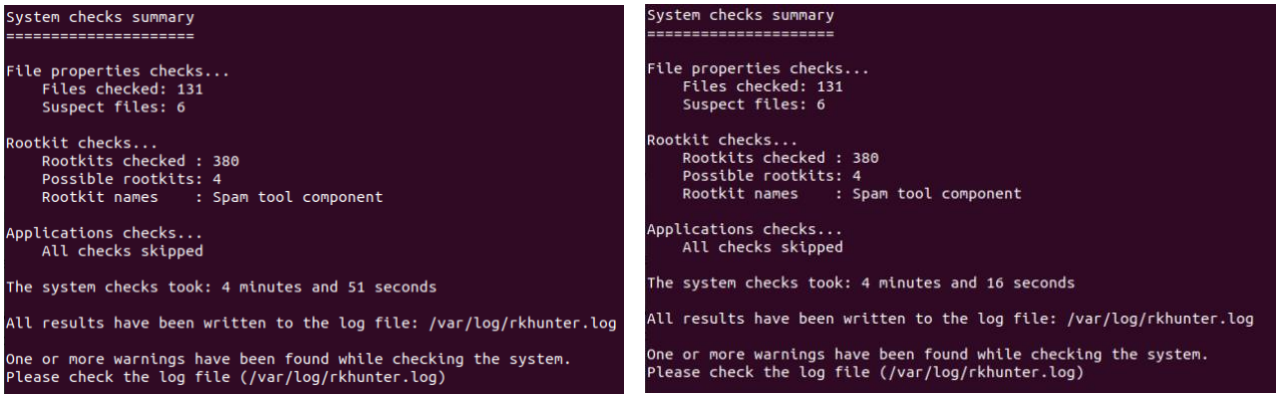


Figure 20: Rootkit Hunter Scan #1 (Left) and Scan #2 (Right) Output on VM A

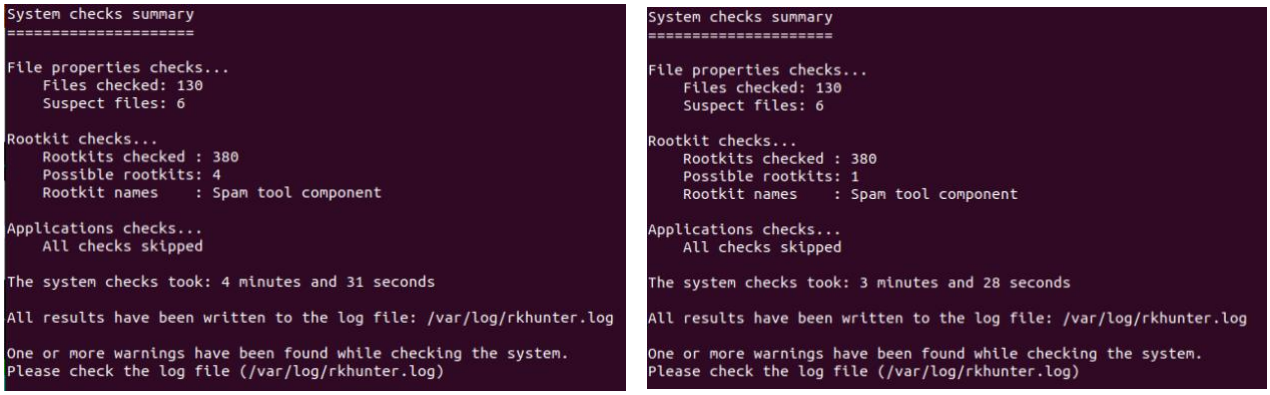


Figure 21: Rootkit Hunter Scan #1 (Left) and Scan #2 (Right) Output on VM B

Rootkit Hunter is advertised as “an easy-to-use tool which checks computers running UNIX (clones) for the presence of rootkits and other unwanted tools” (Horne, n.d.). Its primary function is the detection of rootkits. Although it does advertise that can

detect other tools, the Meterpreter payload was not one of them. While these findings do not necessarily contribute to the analysis of the earlier stated hypothesis, it is worth noting that different software offers different kinds of threat protection and that no one solution offers complete protection.

3.5. Bitdefender GravityZone

3.5.1. VM A Container Scans

Bitdefender antivirus software differs slightly from those previously tested in that it requires a paid-for subscription to access the product. It is a top-of-the-line antivirus software option for Linux distributions, and an assigned account manager with personalized support is available after purchase. With all the features built into this product, it was unsurprising when, on the first scan of VM A, Bitdefender identified the Meterpreter payload. Figure 22 shows the scan output identifying that one item was resolved. The scan log was where the resolved item details were listed; it can be seen that the identified file was the Meterpreter payload.

```
Scan started with Bitdefender Security Tools V7 engines 7.95611
TaskUID: dcf483c4-26d0-4e6f-ba28-6a53a00adae1
Last scan was on: 2023-11-20 at 12:26:21 GMT-5
Scan type: full scan
Scanned paths:
        /
Total scanned items: 307866
Resolved items: 1
Unresolved items: 0
Full Report Available: /opt/bitdefender-security-tools/var/log/dcf483c4-26d0-4e6
f-ba28-6a53a00adae1/1700501181_1700502091948_1_2.xml
```

Figure 22: Bitdefender Scan #1 Output on VM A

```
                <ResolvedDetails>
                        <Item type="0" objectType="0" path="/var/lib/d
ocker/overlay2/5e373ca3b8cf058b0e5ac94457783e650bbbd62e42e21805923226b
4ca1ae1fd/diff/home/docker_payload_01.elf" threatType="6" threatName="
Gen:Variant.Application.Linux.Mettle.2" action="3" initialStatus="3" f
inalStatus="5" failReason="0" flags="0" quarId="" itemHash="a1d485b554
ae2fc7d69aafcbca968c8d94b490f961060d3b741615d9046da5fc" chainHash="a1d
485b554ae2fc7d69aafcbca968c8d94b490f961060d3b741615d9046da5fc" nameCha
in="/var/lib/docker/overlay2/5e373ca3b8cf058b0e5ac94457783e650bbbd62e4
2e21805923226b4ca1ae1fd/diff/home/docker_payload_01.elf" >
</Item>
```

Figure 23: Bitdefender Scan #1 Log Details on VM A

The second scan run on VM A was after executing the malicious binary within the container. In a similar fashion to the first scan, Bitdefender identified the malicious payload within the container and immediately put the file in quarantine. Figures 24, 25, and 26 show the detection of the executed binary in the scan output, log details, and quarantine.

```
Scan started with Bitdefender Security Tools V7 engines 7.95611
TaskUID: dcf483c4-26d0-4e6f-ba28-6a53a00adae1.
Last scan was on: 2023-11-20 at 13:11:05 GMT-5
Scan type: full scan
Scanned paths:
        /
Total scanned items: 308005
Resolved items: 1
Unresolved items: 1
Full Report Available: /opt/bitdefender-security-tools/var/log/dcf483c4-26d0-4e6f-ba28
-6a53a00adae1/1700503865_1700504822983_1_3.xml
```

Figure 24: Bitdefender Scan #2 Output on VM A

```
        </ScanSummary>

        <ScanDetails>
                <UnresolvedDetails>
                        <Item type="0" objectType="0" path="/var/lib/docker/overlay2
/5e373ca3b8cf058b0e5ac94457783e650bbbd62e42e21805923226b4ca1ae1fd/merged/home/docker
_payload_01.elf" threatType="6" threatName="Gen:Variant.Application.Linux.Mettle.2"
action="1" initialStatus="3" finalStatus="3" failReason="2" flags="1" itemHash="a1d4
85b554ae2fc7d69aafcbca968c8d94b490f961060d3b741615d9046da5fc" chainHash="a1d485b554a
e2fc7d69aafcbca968c8d94b490f961060d3b741615d9046da5fc" nameChain="/var/lib/docker/ov
erlay2/5e373ca3b8cf058b0e5ac94457783e650bbbd62e42e21805923226b4ca1ae1fd/merged/home/
docker_payload_01.elf" >
</Item>
                </UnresolvedDetails>

                <ResolvedDetails>
                        <Item type="0" objectType="0" path="/var/lib/docker/overlay2
/5e373ca3b8cf058b0e5ac94457783e650bbbd62e42e21805923226b4ca1ae1fd/diff/home/docker_p
ayload_01.elf" threatType="6" threatName="Gen:Variant.Application.Linux.Mettle.2" ac
tion="3" initialStatus="3" finalStatus="5" failReason="0" flags="1" quarId="" itemHa
sh="a1d485b554ae2fc7d69aafcbca968c8d94b490f961060d3b741615d9046da5fc" chainHash="a1d
485b554ae2fc7d69aafcbca968c8d94b490f961060d3b741615d9046da5fc" nameChain="/var/lib/d
ocker/overlay2/5e373ca3b8cf058b0e5ac94457783e650bbbd62e42e21805923226b4ca1ae1fd/diff
/home/docker_payload_01.elf" >
</Item>
```

Figure 25: Bitdefender Scan #2 Log Details on VM A

```
  # | Threat name | Quarantined time | Path
------------------------------------------------------------------
  1 Gen:Variant.Application.Linux.Mettle.2 2023-11-20-13:18:50 /var/lib/docker/overl
ay2/5e373ca3b8cf058b0e5ac94457783e650bbbd62e42e21805923226b4ca1ae1fd/merged/home/doc
ker_payload_01.elf
  2 Gen:Variant.Application.Linux.Mettle.2 2023-11-20-13:18:50 /var/lib/docker/overl
ay2/5e373ca3b8cf058b0e5ac94457783e650bbbd62e42e21805923226b4ca1ae1fd/diff/home/docke
r_payload_01.elf
```

Figure 26: Bitdefender Scan #2 Quarantine Details on VM A

3.5.2. VM B Non-Container Scans

Given the effectiveness of Bitdefender in detecting the Meterpreter payload in the container, it was no surprise that the antivirus was even more effective at discovering and mitigating the threat when the binary was simply on the VM Desktop. The scan output was perplexing to understand at first glance. Figure 27 illustrates that the scan output listed the number of ‘resolved items’ as zero, meaning no malicious files were found. However, when querying the quarantined items, as is illustrated in Figure 28, the

Meterpreter payload is listed. Once Bitdefender was installed and running on the VM, it immediately detected and quarantined the malicious binary. These actions were taken before the initialization of the system scan, which is why the scan returned zero results — at that point, the threat was already mitigated.

```
Scan started with Bitdefender Security Tools V7 engines 7.95613
TaskUID: dcf483c4-26d0-4e6f-ba28-6a53a00adae1
Last scan was on: 2023-11-20 at 12:32:53 GMT-5
Scan type: full scan
Scanned paths:
        /
Total scanned items: 304415
Resolved items: 0
Unresolved items: 0
Full Report Available: /opt/bitdefender-security-tools/var/log/dcf483c4-26d0-4e6
f-ba28-6a53a00adae1/1700501573_1700502473608_1_1.xml
```

Figure 27: Bitdefender Scan #1 Output on VM B

```
  # | Threat name | Quarantined time | Path
-----------------------------------------------------------------
  1 Gen:Variant.Application.Linux.Mettle.2 2023-11-20-12:21:05 /home/emily/Deskt
op/docker_payload_01.elf
  2 Gen:Variant.Application.Linux.Mettle.2 2023-11-20-12:21:05 /home/emily/Deskt
op/docker_payload_01.elf
  3 Gen:Variant.Application.Linux.Mettle.2 2023-11-20-13:05:35 /home/emily/Deskt
op/docker_payload_01.elf
  4 Gen:Variant.Application.Linux.Mettle.2 2023-11-20-13:05:35 /home/emily/Deskt
op/docker_payload_01.elf
```

Figure 28: Bitdefender Scan #1 Quarantine Details on VM B

As Bitdefender immediately mitigated the malicious payload upon installation, executing the binary for the second scan was impossible as it was already in quarantine.

3.6. Kaspersky

3.6.1. VM A Container Scans

Kaspersky is similar to Bitdefender in that it is also a paid-for, subscription-based antivirus software. There are many variations of the Kaspersky product to include cloud,

endpoint, and enterprise solutions. For the purposes of this experiment, the Linux endpoint solution was tested. After the installation of the product the first malware scan was initiated. As is evident in Figure 29, Kaspersky was successfully able to identify the malicious payload even as it was a binary at rest within the container. In addition to identification, the antivirus also removed the malware moments after identification which is seen in Figure 30.

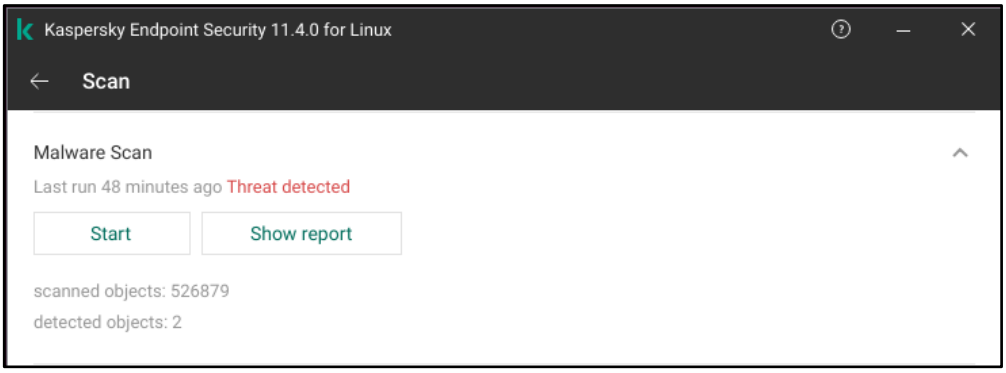
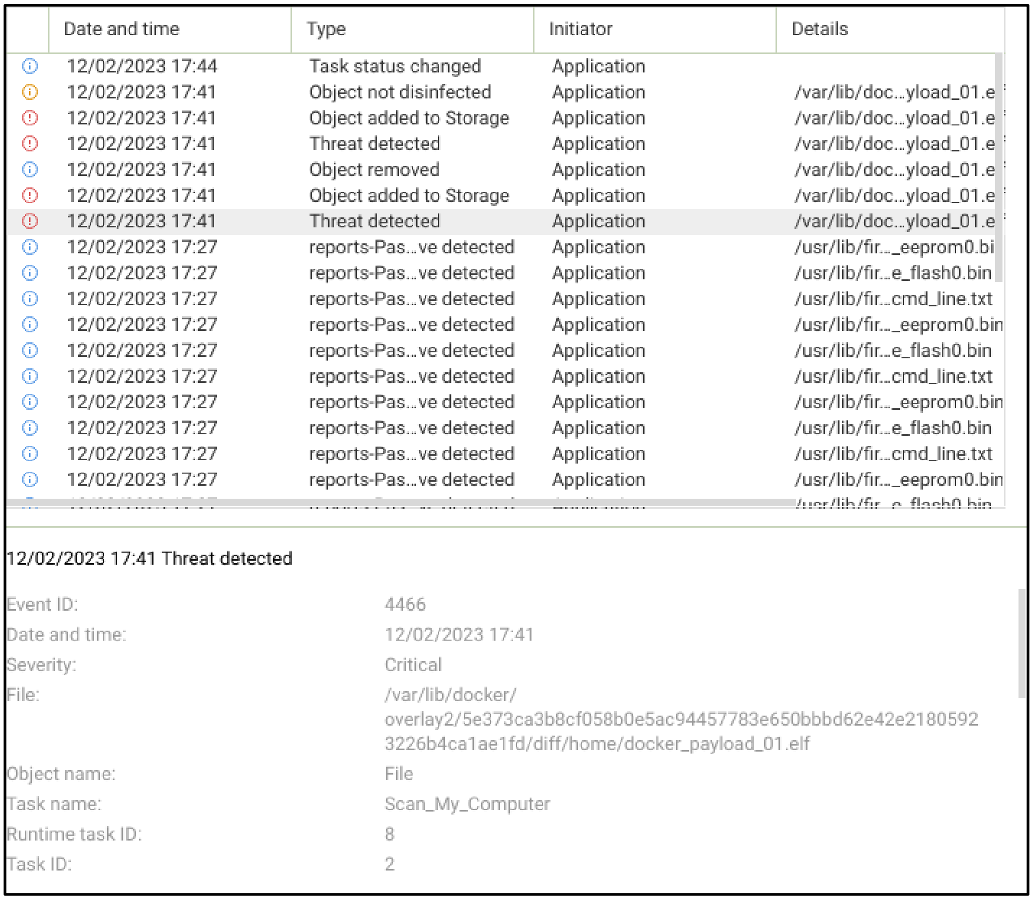


Figure 29: Kaspersky Scan #1 Results on VM A



Date and time	Type	Initiator	Details
12/02/2023 17:44	Task status changed	Application	
12/02/2023 17:41	Object not disinfected	Application	/var/lib/doc...yload_01.e
12/02/2023 17:41	Object added to Storage	Application	/var/lib/doc...yload_01.e
12/02/2023 17:41	Threat detected	Application	/var/lib/doc...yload_01.e
12/02/2023 17:41	Object removed	Application	/var/lib/doc...yload_01.e
12/02/2023 17:41	Object added to Storage	Application	/var/lib/doc...yload_01.e
12/02/2023 17:41	Threat detected	Application	/var/lib/doc...yload_01.e
12/02/2023 17:27	reports-Pas...ve detected	Application	/usr/lib/fir..._eeprom0.bi
12/02/2023 17:27	reports-Pas...ve detected	Application	/usr/lib/fir...e_flash0.bin
12/02/2023 17:27	reports-Pas...ve detected	Application	/usr/lib/fir...cmd_line.txt
12/02/2023 17:27	reports-Pas...ve detected	Application	/usr/lib/fir..._eeprom0.bin
12/02/2023 17:27	reports-Pas...ve detected	Application	/usr/lib/fir...e_flash0.bin
12/02/2023 17:27	reports-Pas...ve detected	Application	/usr/lib/fir...cmd_line.txt
12/02/2023 17:27	reports-Pas...ve detected	Application	/usr/lib/fir..._eeprom0.bin
12/02/2023 17:27	reports-Pas...ve detected	Application	/usr/lib/fir...e_flash0.bin
12/02/2023 17:27	reports-Pas...ve detected	Application	/usr/lib/fir...cmd_line.txt
12/02/2023 17:27	reports-Pas...ve detected	Application	/usr/lib/fir..._eeprom0.bin

12/02/2023 17:41 Threat detected

Event ID:	4466
Date and time:	12/02/2023 17:41
Severity:	Critical
File:	/var/lib/docker/overlay2/5e373ca3b8cf058b0e5ac94457783e650bbbd62e42e2180592 3226b4ca1ae1fd/diff/home/docker_payload_01.elf
Object name:	File
Task name:	Scan_My_Computer
Runtime task ID:	8
Task ID:	2

Figure 30: Kaspersky Scan #1 Report on VM A

The execution of the malware within the container did yield slightly different results from those that we saw in the first scan. Immediately after executing the binary there was a small popup on the desktop of the host computer. The popup, shown in Figure 31, indicated that a threat was detected. Upon further investigation, the removal of the malware was also confirmed. Given the immediate nature of the detection and removal upon execution, it was not possible to run a second scan.

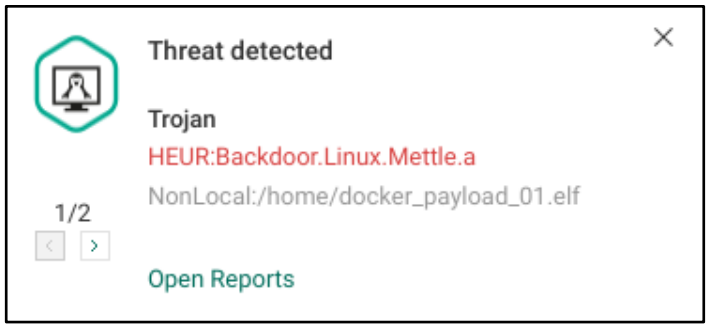
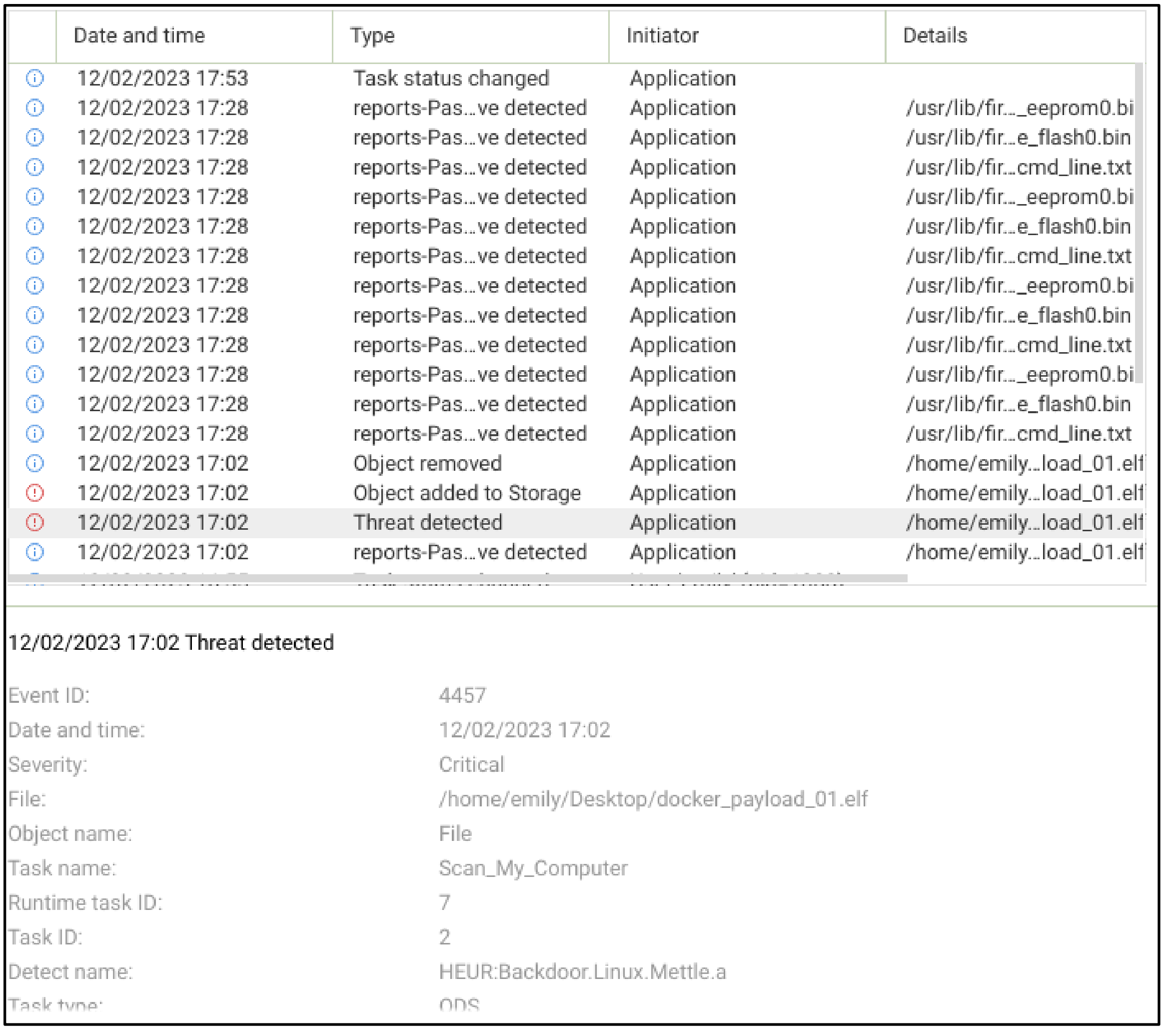


Figure 31: Kaspersky Detection and Removal of Executed Malware on VM A

3.6.2. VM B Non-Container Scans

Unsurprisingly, the results of the scan on VM B were not only similar to the results on VM A, but they were also similar to the Bitdefender results. The first scan discovered and removed the malicious binary, residing on the computer's desktop. This can be seen in Figures 32 and 33.

Figure 32: Kaspersky Scan #1 Results on VM B



Date and time	Type	Initiator	Details
12/02/2023 17:53	Task status changed	Application	
12/02/2023 17:28	reports-Pas...ve detected	Application	/usr/lib/fir..._eeprom0.bi
12/02/2023 17:28	reports-Pas...ve detected	Application	/usr/lib/fir...e_flash0.bin
12/02/2023 17:28	reports-Pas...ve detected	Application	/usr/lib/fir...cmd_line.txt
12/02/2023 17:28	reports-Pas...ve detected	Application	/usr/lib/fir..._eeprom0.bi
12/02/2023 17:28	reports-Pas...ve detected	Application	/usr/lib/fir...e_flash0.bin
12/02/2023 17:28	reports-Pas...ve detected	Application	/usr/lib/fir...cmd_line.txt
12/02/2023 17:28	reports-Pas...ve detected	Application	/usr/lib/fir..._eeprom0.bi
12/02/2023 17:28	reports-Pas...ve detected	Application	/usr/lib/fir...e_flash0.bin
12/02/2023 17:28	reports-Pas...ve detected	Application	/usr/lib/fir...cmd_line.txt
12/02/2023 17:28	reports-Pas...ve detected	Application	/usr/lib/fir..._eeprom0.bi
12/02/2023 17:28	reports-Pas...ve detected	Application	/usr/lib/fir...e_flash0.bin
12/02/2023 17:28	reports-Pas...ve detected	Application	/usr/lib/fir...cmd_line.txt
12/02/2023 17:02	Object removed	Application	/home/emily...load_01.elf
12/02/2023 17:02	Object added to Storage	Application	/home/emily...load_01.elf
12/02/2023 17:02	Threat detected	Application	/home/emily...load_01.elf
12/02/2023 17:02	reports-Pas...ve detected	Application	/home/emily...load_01.elf

12/02/2023 17:02 Threat detected

Event ID:	4457
Date and time:	12/02/2023 17:02
Severity:	Critical
File:	/home/emily/Desktop/docker_payload_01.elf
Object name:	File
Task name:	Scan_My_Computer
Runtime task ID:	7
Task ID:	2
Detect name:	HEUR:Backdoor.Linux.Mettle.a
Task type:	ODS

Figure 33: Kaspersky Scan #1 Report on VM B

As with Bitdefender, a second scan on VM B was impossible. This is because upon execution of the malicious payload, Kaspersky immediately removed the threat; it did not wait for the initiation of a scan for identification and removal.

3.7. Summary of Findings

As is shown in Figure 34, the results of the experiments are all across the board. While obfuscating the malware in a container was effective against specific antivirus software, it was not effective in all cases. The antivirus software most effective at detecting the malicious binary were ClamAV, an open-source file scanning software, as

well as Bitdefender and Kaspersky, both paid-for subscription services geared towards enterprise deployments. The binary execution only made a difference in detection within the container for ESET; in all other cases the binary detection (or lack thereof) was execution agnostic. The only software to fail the control was Rootkit Hunter. Although this software cannot contribute to proving or disproving the hypothesis, it does illustrate that not all antivirus software is intended to find every kind of malware and emphasizes the need to select the appropriate software for the intended environment.

	VM A		VM B	
	Binary At Rest	Executed Binary	Binary At Rest	Executed Binary
ClamAV	✓	✓	✓	✓
Sophos	✗	✗	✓	✓
Rootkit Hunter	✗	✗	✗	✗
ESET	✗	✓	✓	✓
Bitdefender	✓	✓	✓	✓
Kaspersky	✓	✓	✓	✓

Figure 34: Table of Findings

The hypothesis initially stated was that “malware that is installed in and run inside of a container will evade antivirus detection.” The results of the experiments did not definitively prove this true, nor did they prove it false. Rather, it can be determined that antivirus software selection for the system plays a monumental role in whether containerized malware will be detected.

4. Recommendations

As the containerization of applications and services grows, their security should be of utmost concern to the organization that deploys them. While their functionality provides many benefits to the organization, they also provide an increased attack surface - as was evidenced by the conducted experiments. An infected container could precipitate a much larger compromise, so their security is important. One key recommendation gleaned from this exercise is for each organization to have a keen understanding of the

antivirus used within their networks. Not only was it proven that certain antivirus software is more effective at detecting malware within a container, but it was also proven that certain antivirus is incapable of detecting the chosen type of malware. As such, the organization must understand what their chosen software can detect and whether it can scan within the container architecture. Additionally, regardless of whatever flavor of antivirus is installed in the network, regular scans should be run, and the results of those scans reviewed and addressed.

Another recommendation to mitigate the threat of container-obfuscated malware would be to run an instance of antivirus software within the container instance itself. While not all antivirus software advertises that this is possible, during the research for suitable software for the experiments presented in this paper, multiple versions of antivirus were discovered that advertise they can run within a container; ClamAV was one to assert that claim.

Lastly, if an organization acknowledges that containers are a security blind spot and that their antivirus software is incapable of detection within a container architecture, the organization could, and arguably should, regularly delete and rebuild their containers from known-good images. Doing so would prevent the long-term occupation of malware within the network. It would also baseline the network for the administrators in that they would have a solid understanding of 'clean' infrastructure.

5. Conclusion

Containers are not a new computing phenomenon. They are, however, experiencing intense growth as their functionality continues to impart impressive benefits to the organizations that adopt them. With continued growth inevitably comes an increase in risk that must be addressed. The first step in addressing the vulnerability is understanding already established tools and their capabilities. This thought led to the hypothesis that containerized malware would be completely obfuscated from antivirus scans.

Experimenting and testing different kinds of antivirus software made it clear that there was no one-size-fits-all solution to the previously stated hypothesis. The degree of success the antivirus had in detecting the containerized malware varied wildly from software to software. The antivirus software's follow-on actions after malware identification were also drastically different. Given these findings, recommendations were provided on how a network administrator might address containers' increased risk. Understanding the capabilities of the antivirus software at use within the network is imperative.

As with any new technology, necessity drives innovation and it is no different in the case of containers. Fortunately, there are already tools on the market that have kept pace with technological development and can protect against and mitigate the ever-present threats to networks everywhere. The research presented in this paper shows that the key to container security lies not only in the software chosen to protect the network but also in the knowledge held by administrators of their chosen tool's capabilities.

References

CNCF Survey 2019. (n.d.). https://www.cncf.io/wp-content/uploads/2020/08/CNCF_Survey_Report.pdf

Containerization explained. IBM. (n.d.). https://www.ibm.com/topics/containerization

Docker - market share, competitor insights in Containerization. 6sense. (n.d.). https://6sense.com/tech/containerization/docker-market-share

Horne, j. (n.d.). *Rootkit hunter*. Rootkit Hunter / Code / [016a77] /files/FAQ. https://sourceforge.net/p/rkhunter/rkh_code/ci/master/tree/files/FAQ

Malik, F. (2022, December 10). *What is containers architecture?*. Medium. https://medium.com/p/54826e93fc18

Mell, E. (2023, January 20). *The evolution of containers: Docker, Kubernetes and the future: TechTarget*. IT Operations. https://www.techtarget.com/searchitoperations/feature/Dive-into-the-decades-long-history-of-container-technology

Scolati, Remo & Fronza, Ilenia & El Ioini, Nabil & Samir, Areeg & Pahl, Claus. (2019). *A Containerized Big Data Streaming Architecture for Edge Cloud Computing on Clustered Single-Board Devices*. https://doi.org/10.5220/0007695000680080.

Vigderman, A. (2023, September 20). *How does antivirus software work?*. Security.org. https://www.security.org/antivirus/how-does-antivirus-work/