

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING

SANDBOX GENERATOR

ERKAN YILMAZ

SUPERVISOR
PROF. IBRAHIM SOGUKPINAR

GEBZE
2022

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

SANDBOX GENERATOR

ERKAN YILMAZ

SUPERVISOR
PROF. IBRAHIM SOGUKPINAR

2022
GEBZE

 <p>GEBZE TECHNICAL UNIVERSITY</p>	<p>GRADUATION PROJECT JURY APPROVAL FORM</p>
--	--

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 31/08/2021 by the following jury.

JURY

Member

(Supervisor) : Prof. Ibrahim Sogukpınar

Member : Doç. Dr. Mehmet Göktürk

Member : M.Sc Sibel Gülmez

ABSTRACT

The goal of this research was to create a malware sandbox that work like real environment so observes malwares actual behavior.

Due to the increase of the malwares that spread many ways like USB or phishing mail attacks against the enterprise environments or even targeting the individuals, you will hope to test every file you suspect on SandBox to analyze the file before running it on a real environment to make sure that this file is not malicious or harmful.

Keywords: malware, sandbox.

ÖZET

Bu araştırmanın amacı, gerçek ortam gibi çalışan ve böylece kötü amaçlı yazılımın gerçek davranışını gözlemleyen bir kötü amaçlı yazılım sanal alanı oluşturmaktır.

USB veya kimlik avı postası gibi birçok yolla yayılan kötü amaçlı yazılımların artması nedeniyle kurumsal ortamlara veya bireylere yönelik saldırılar artmıştır, Çalıştırmadan önce şüphelenen dosyayı analiz etmek için SandBox kullanmak bu tarz kötü amaçlı yazılımlara karşı elimizde olan araçlardan biridir. Kum havuzları bu dosyaların kötü niyetli veya zararlı olmadığından emin olmak için gerekli bir ortamlardır.

Anahtar Kelimeler: kötü amaçlı yazılım, Kum havuzu.

ACKNOWLEDGEMENT

I'd like to thank everyone who helped with the project's preparation, especially my instructor Prof. Dr. İbrahim SOGUKPINAR, who steered the project to completion, and Gebze Technical University for their supporting this study.

In addition, I'd want to convey my gratitude and affection to my family, who have always been supportive of me during my schooling, and to all of my instructors, who have led by example with their lives.

Erkan Yılmaz

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol or
Abbreviation : Explanation

CONTENTS

Abstract	iv
Özet	v
Acknowledgement	vi
List of Symbols and Abbreviations	vii
Contents	ix
List of Figures	x
List of Tables	xi
1 Introduction	1
2 Method and Material	2
2.1 A STUDY OF ANTI-VIRUS PROGRAM and SANDBOX SYSTEMS’ PRINCIPLES OF OPERATIONS	2
2.1.1 DETECTION BASED ON SIGNATURE	2
2.1.2 HEURISTIC DETECTION	2
2.2 ANTIVIRUS BYPASS TECHNIQUES	2
2.2.1 TECHNIQUE FOR FILE COMPRESSION (PACKERS) . . .	3
2.2.2 MALWARE ENCRYPTION (CRYPTER)	3
2.3 Sandbox detection and evasion techniques	3
2.3.1 WMI queries	4
2.3.2 Other environment checks	6
2.3.3 Does a sandbox have to detect all evasion techniques	7
3 Project Design	8
3.1 Requirements	8
3.2 Design Graph	8
4 Tests and Results	9
4.1 Test case 1	9
4.2 Test case 2	10

4.3 Test case 3	11
5 Conclusions	12
Bibliography	13
Appendices	13

LIST OF FIGURES

2.1	Output for the query <code>SELECT * FROM SAcpi ThermalZoneTemperature</code> on a physical machine	4
2.2	Output for the query <code>SELECT * FROM MSAcpi ThermalZoneTemperature</code> in a virtual environment	4
2.3	Output for the query <code>SELECT * FROM Win32 Fan</code> on a physical machine	5
2.4	Output for the query <code>SELECT * FROM Win32 Fan</code> in a virtual environment	5
3.1	Design Model of Sandbox Generator Project	8
4.1	Test case 1	9
4.2	Test case 2	10
4.3	Test case 3	11

LIST OF TABLES

1. INTRODUCTION

Due to the increase of the malwares that spread many ways like USB or phishing mail attacks against the enterprise environments or even targeting the individuals, you will hope to test every file you suspect on SandBox to analyze the file before running it on a real environment to make sure that this file is not malicious or harmful.

- Examining the anti-virus and sandbox systems' principles of operation.
- Study on Sandbox detection and evasion techniques
- A study of virtualization technologies
- Project planning
- Implementation of project
- Performing Tests
- Report Preparation
- Conclusion

2. METHOD AND MATERIAL

2.1. A STUDY OF ANTI-VIRUS PROGRAM and SAND-BOX SYSTEMS' PRINCIPLES OF OPERATIONS

Antiviruses must be understood in order to detect dangerous software, the number and variety of which is growing by the day, and to safeguard the system in which they are installed. To begin, dangerous software is detected using two methods: "Signature-Based Detection" and "Heuristic Detection."

2.1.1. DETECTION BASED ON SIGNATURE

Virus signatures are stored in the databases of all antivirus programs. During scanning, if the antivirus software encounters the hex codes (signatures), md5 values of the viruses recorded in the database, it recognizes the malware and quarantines it. This method provides protection against known viruses. Antiviruses have the problem of being one step behind when it comes to newly published infections. Because antiviruses are unable to distinguish viruses that have not been recorded in a database, they are unable to detect them. Antiviruses are constantly one step ahead of viruses, as seen by this.

2.1.2. HEURISTIC DETECTION

Because a virus that is not listed in the database may be present in the system, antivirus software tries to identify software that may access vital system files, make modifications, and display suspicious behavior. Heuristic detection is the term for this sort of pest detection.

2.2. ANTIVIRUS BYPASS TECHNIQUES

Various methods can be used to avoid getting stuck with antiviruses. Let's talk about these techniques one by one.

2.2.1. TECHNIQUE FOR FILE COMPRESSION (PACKERS)

To disguise its look, malware is compressed. When it starts to execute, the compressed malware unpacks itself from memory and begins executing the dangerous code. The pest's appearance will alter in this situation, but its function will not. Because it is not stripped of its package before execution, the compressed malicious file cannot be detected using signature-based detection.

2.2.2. MALWARE ENCRYPTION (CRYPTER)

The malware is encrypted in this method so that it cannot be decoded by antivirus software and hence cannot be identified by antivirus software. Crypters are made up of two pieces. The "Stub" feature of these pieces allows the infection to be hidden. The "client" section conceals the bug behind the stub. However, malware that uses this approach to encrypt data decrypts it within the target to complete its mission. As a result, antiviruses with dynamic analysis sensitive to change can identify the infection. Antiviruses that do static analysis, on the other hand, can be disregarded because they are unaffected by future alterations.

2.3. Sandbox detection and evasion techniques

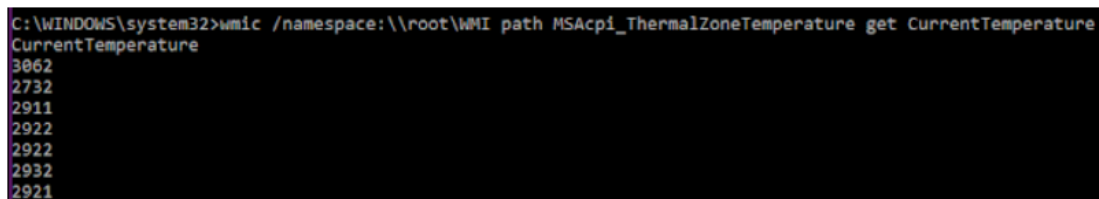
- Checking whether the SbieDll.dll, Dbghelp.dll, Apilog.dll, or Dirwatch.dll libraries are loaded,
- Obtaining the value of the SystemBiosVersion key of the HARDWARE DESCRIPTION System registry branch,
- Calling the GetTickCount function twice to check for step-through execution,
- Checking for the file C:FilesTools.exe,
- Sending WMI queries to get the BIOS version and manufacturer,
- Sending WMI queries to check the number of CPU cores; the value must exceed 1
- Sending WMI queries to check the amount of physical memory; the value must be at least 2,900,000,000 bytes
- Checking the number of running processes for Wireshark and Sysinternals
- Checking for obsolete SbieDll.dll system artifact

- Checking active processes for vmtoolsd.exe and vbox.exe
- Checking system time

2.3.1. WMI queries

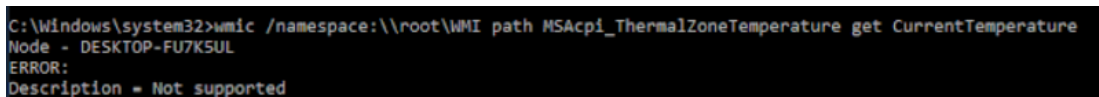
Since 2016, malware developers have been actively using WMI queries. Windows Management Instrumentation (WMI) is a technology for centralized management of Windows-based infrastructures. to access devices, accounts, services, processes, network interfaces, and other programs. Of the malware in question, 25 percent makes use of them. In most cases, the attackers are trying to find out the model of hard drive or motherboard, as well as OS and BIOS versions.

GravityRAT uses an interesting method to detect virtual environments. By sending the `SELECT * FROM MSAcpiThermalZoneTemperature` WMI query, it checks the CPU temperature: if the malware is being run on a physical machine, the temperature value will be returned. But if the system responds with "ERROR" or "Not Supported," this means that the malware is running in a virtual environment.



```
C:\WINDOWS\system32>wmic /namespace:\\root\WMI path MSAcpi_ThermalZoneTemperature get CurrentTemperature
CurrentTemperature
3062
2732
2911
2922
2922
2932
2921
```

Figure 2.1: Output for the query `SELECT * FROM SAcpi ThermalZoneTemperature` on a physical machine

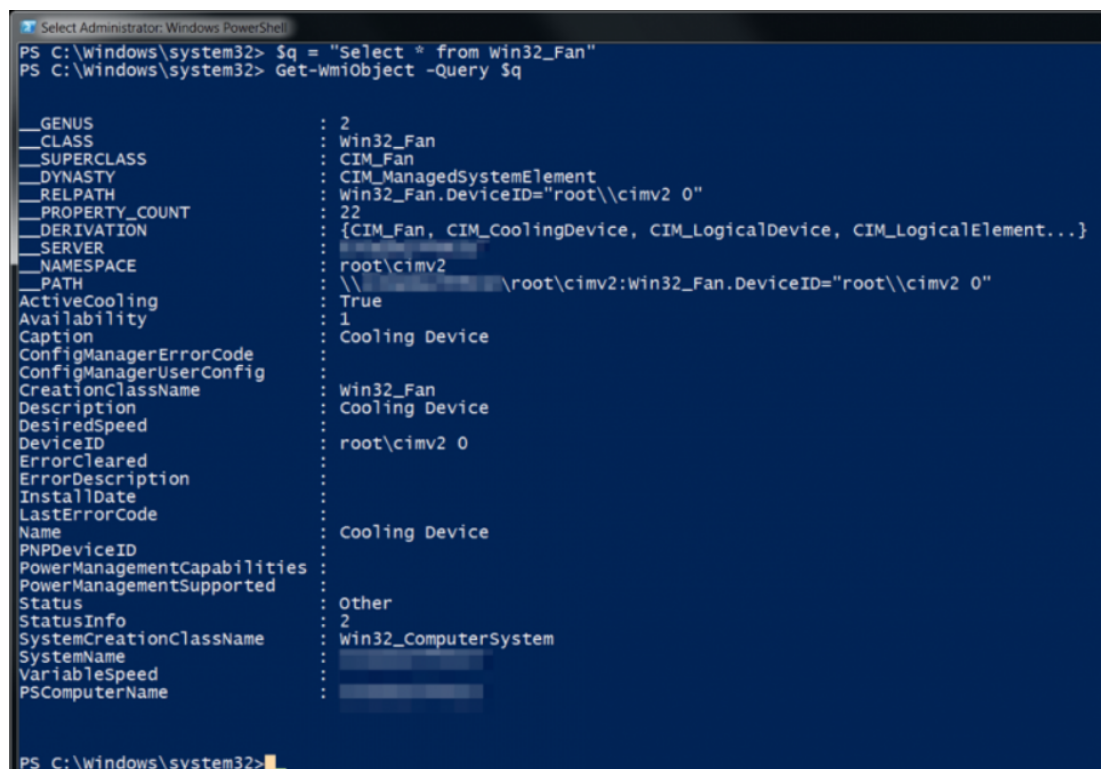


```
C:\Windows\system32>wmic /namespace:\\root\WMI path MSAcpi_ThermalZoneTemperature get CurrentTemperature
Node - DESKTOP-FU7K5UL
ERROR:
Description = Not supported
```

Figure 2.2: Output for the query `SELECT * FROM MSAcpi ThermalZoneTemperature` in a virtual environment

WMI queries have also been used by the OilRig group (APT34, Helix Kitten), which for more than five years has targeted a variety of industries, including government, finance, energy, and telecommunications, primarily in the Middle East. The group's backdoor OopsIE sends the WMI query `SELECT * FROM Win32 Fan` to check the state of the CPU fan. This query should return a class that provides statistics on the CPU fan. The backdoor checks whether the response is empty, which would indicate a

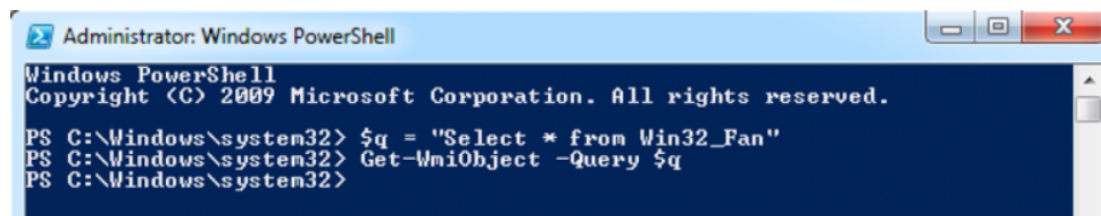
virtual environment.



```
Select Administrator: Windows PowerShell
PS C:\windows\system32> $q = "Select * from Win32_Fan"
PS C:\windows\system32> Get-WmiObject -Query $q

__GENUS                : 2
__CLASS                 : Win32_Fan
__SUPERCLASS            : CIM_Fan
__DYNASTY                : CIM_ManagedSystemElement
__RELPATH                : Win32_Fan.DeviceID="root\\cimv2 0"
__PROPERTY_COUNT        : 22
__DERIVATION             : {CIM_Fan, CIM_CoolingDevice, CIM_LogicalDevice, CIM_LogicalElement...}
__SERVER                : \\.\
__NAMESPACE              : root\\cimv2
__PATH                  : \\.\root\\cimv2:Win32_Fan.DeviceID="root\\cimv2 0"
ActiveCooling            : True
Availability              : 1
Caption                  : Cooling Device
ConfigManagerErrorCode    : 
ConfigManagerUserConfig   : 
CreationClassName        : Win32_Fan
Description               : Cooling Device
DesiredSpeed              : 
DeviceID                  : root\\cimv2 0
ErrorCleared              : 
ErrorDescription          : 
InstallDate               : 
LastErrorCode             : 
Name                      : Cooling Device
PNPDeviceID              : 
PowerManagementCapabilities : 
PowerManagementSupported  : 
Status                    : Other
StatusInfo                : 2
SystemCreationClassName   : Win32_ComputerSystem
SystemName                : 
VariableSpeed             : 
PSComputerName            :
```

Figure 2.3: Output for the query SELECT * FROM Win32 Fan on a physical machine



```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> $q = "Select * from Win32_Fan"
PS C:\Windows\system32> Get-WmiObject -Query $q
PS C:\Windows\system32>
```

Figure 2.4: Output for the query SELECT * FROM Win32 Fan in a virtual environment

2.3.2. Other environment checks

In addition to checking running processes, registry key values, and sending WMI queries, malefactors can check the environment in other ways. For example, the RTM (Redaman) banking trojan checks for the following files and directories on C: and D: drives:

- cuckoo,
- fake drive,
- perl,
- strawberry,
- targets.xls,
- tsl,
- wget.exe,
- *python*.

The existence of any of these files or directories indicates that the malware is running in a sandbox or a code analyzer.

APT37 (also known as ScarCruft, Group123, and TEMP.Reaper) has modified its ROKRAT backdoor over the last several years. In addition to checking registry key values, this malware also checks whether the file C:FilesTools.exe exists and whether the following code analyzer and debugger DLLs have been loaded:

- SbieDll.dll,
- Dbghelp.dll,
- Apilog.dll, Dirwatch.dll.

Correctly configuring virtual machines is enough to stop the following attacker technique. PoetRAT, remote access malware, used in targeted attacks against ICS and SCADA systems in the energy sector, checks the hard disk size to determine whether it is running in a sandbox environment. Since the malware assumes that sandboxes have hard drives of less than 62 GB, it can be tricked by allocating more space for the virtual machine.

2.3.3. Does a sandbox have to detect all evasion techniques

Not all sandbox evasion methods are easy to detect. Some checks—such as file path, MAC address, date and time, and operation execution time—strongly resemble legitimate actions. Detection may generate a large number of false positives and interfere with proper functioning of other programs. This, however, does not mean that malware will remain totally invisible. Sandboxes do not have to catch each and every evasion technique, since malware has many other attributes that can be detected at other stages of operation. That said, the more techniques the sandbox sees, the greater the chances of detecting new malware samples and applying this information to counter cyberthreats.

3. PROJECT DESIGN

3.1. Requirements

- Docker must be installed on the computer.
- Windows 10 operating system must be installed on the host machine.

3.2. Design Graph

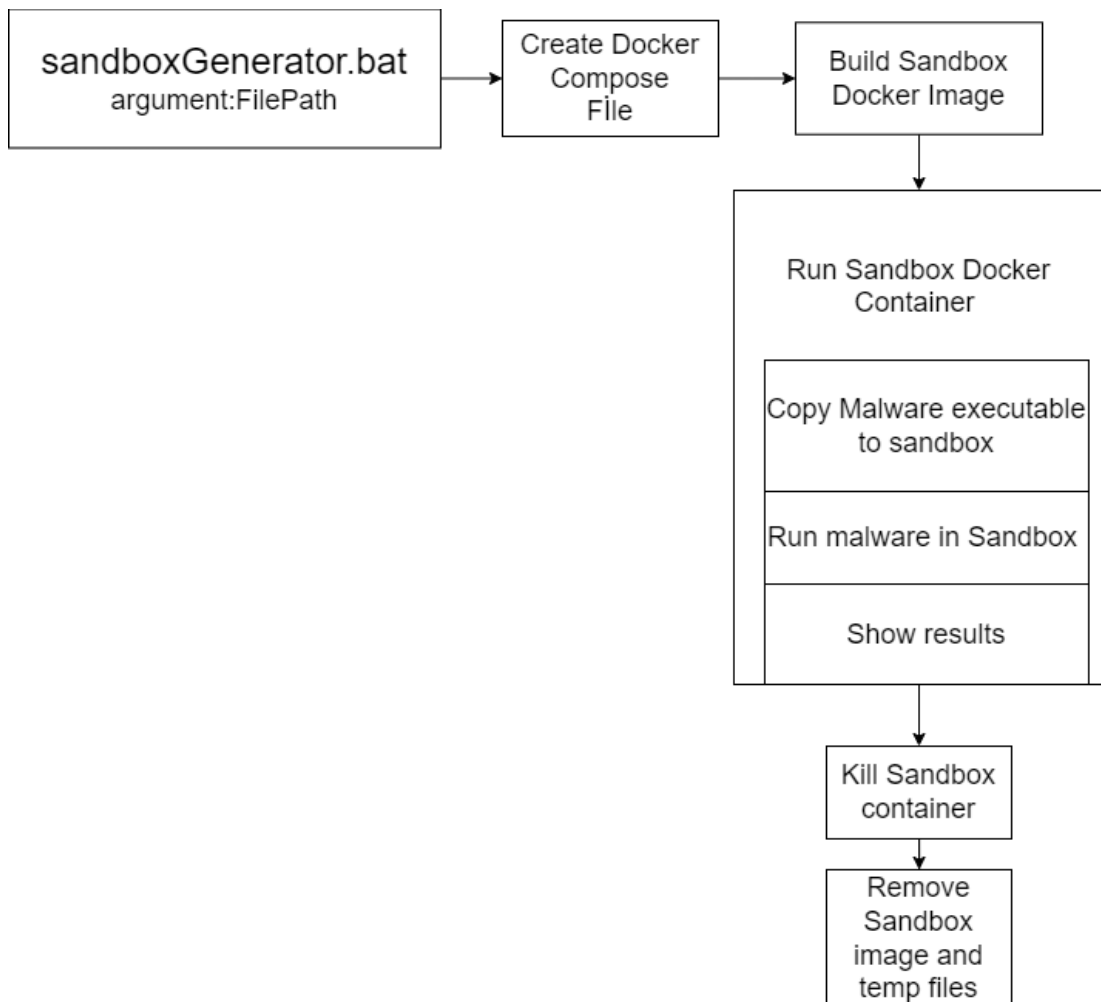


Figure 3.1: Design Model of Sandbox Generator Project

4. TESTS AND RESULTS

4.1. Test case 1

```
PS C:\Users\erkan\OneDrive\Masaüstü\bitirme\composetest> .\run.bat main
main.txt
1 file(s) copied.
REPOSITORY TAG IMAGE ID CREATED SIZE
[+] Building 3.1s (12/12) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 32B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/ubuntu:latest
=> [auth] library/ubuntu:pull token for registry-1.docker.io
=> [internal] load build context
=> => transferring context: 26B
=> [1/6] FROM docker.io/library/ubuntu:latest@sha256:26c68657ccce2cb0a31b330cb0be2b5e188d467f641c62e13ab40cbec258c68d
=> CACHED [2/6] RUN apt update && apt install gcc -y
=> CACHED [3/6] RUN apt install -y clamav
=> CACHED [4/6] RUN freshclam
=> CACHED [5/6] RUN echo 'echo 'program stating!!!' && ./temp && echo '\nprogram end!!!' && clamscan temp">>run.sh
=> CACHED [6/6] COPY temp /
=> exporting to image
=> => exporting layers
=> => writing image sha256:c7c4acffbc0114bb7418f909ca581fc71f9452ea618290dafa6c18cfd5a63951
=> => naming to docker.io/library/sandbox
program stating!!!
hello world
program end!!!
/temp: OK

----- SCAN SUMMARY -----
Known viruses: 8617620
Engine version: 0.103.6
Scanned directories: 0
Scanned files: 1
Infected files: 0
Data scanned: 0.01 MB
Data read: 0.01 MB (ratio 1.00:1)
Time: 23.993 sec (0 m 23 s)
Start Date: 2022-06-04 12:03:20
End Date: 2022-06-04 12:03:44
Untagged: sandbox:latest
Deleted: sha256:c7c4acffbc0114bb7418f909ca581fc71f9452ea618290dafa6c18cfd5a63951
PS C:\Users\erkan\OneDrive\Masaüstü\bitirme\composetest>
```

Figure 4.1: Test case 1

4.2. Test case 2

```
PS C:\Users\erkan\OneDrive\Masaüstü\bitirme\composetest> .\run.bat main
main.txt
1 file(s) copied.
REPOSITORY TAG IMAGE ID CREATED SIZE
[+] Building 3.0s (12/12) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 32B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/ubuntu:latest
=> [auth] library/ubuntu:pull token for registry-1.docker.io
=> [1/6] FROM docker.io/library/ubuntu:latest@sha256:26c68657ccce2cb0a31b330cb0be2b5e108d467f641c62e13ab40cbec258c68d
=> [internal] load build context
=> => transferring context: 26B
=> CACHED [2/6] RUN apt update && apt install gcc -y
=> CACHED [3/6] RUN apt install -y clamav
=> CACHED [4/6] RUN freshclam
=> CACHED [5/6] RUN echo "echo 'program stating!!!" && ./temp && echo '\nprogram end!!!" && clamscan temp">>run.sh
=> CACHED [6/6] COPY temp /
=> exporting to image
=> => exporting layers
=> => writing image sha256:45215ac74f2b2897ff41eacddc761450a25810249968754fec006e89630da58d
=> => naming to docker.io/library/sandbox
program stating!!!
this is a test program!!!
program end!!!
/temp: OK

----- SCAN SUMMARY -----
Known viruses: 8617620
Engine version: 0.103.6
Scanned directories: 0
Scanned files: 1
Infected files: 0
Data scanned: 0.01 MB
Data read: 0.01 MB (ratio 1.00:1)
Time: 24.021 sec (0 m 24 s)
Start Date: 2022:06:04 12:08:19
End Date: 2022:06:04 12:08:43
Untagged: sandbox:latest
Deleted: sha256:45215ac74f2b2897ff41eacddc761450a25810249968754fec006e89630da58d
PS C:\Users\erkan\OneDrive\Masaüstü\bitirme\composetest>
```

Figure 4.2: Test case 2

4.3. Test case 3

```
PS C:\Users\erkan\OneDrive\MasaÅüstÅ\bitirme\composetest> .\run.bat shell.elf
shell.elf.txt
1 file(s) copied.
REPOSITORY TAG IMAGE ID CREATED SIZE
[+] Building 2.8s (12/12) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 32B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/ubuntu:latest
=> [auth] library/ubuntu:pull token for registry-1.docker.io
=> [internal] load build context
=> => transferring context: 26B
=> [1/6] FROM docker.io/library/ubuntu:latest@sha256:26c68657ccce2cb0a31b330cb0be2b5e108d467f641c62e13ab40cbec258c68d
=> CACHED [2/6] RUN apt update && apt install gcc -y
=> CACHED [3/6] RUN apt install -y clamav
=> CACHED [4/6] RUN freshclam
=> CACHED [5/6] RUN echo "echo 'program stating!!!' &&echo '\nprogram end!!!' &&clamscan temp">>run.sh
=> CACHED [6/6] COPY temp /
=> exporting to image
=> => exporting layers
=> => writing image sha256:d4d0693ecfae03298e63813da4d4780956235ff4cc54ef7170cc4a23f6ca970
=> => naming to docker.io/library/sandbox
program stating!!!

program end!!!
/temp: OK

----- SCAN SUMMARY -----
Known viruses: 8617620
Engine version: 0.103.6
Scanned directories: 0
Scanned files: 1
Infected files: 1
Data scanned: 1.50 MB
Data read: 0.00 MB (ratio 0.00:1)
Time: 18.681 sec (0 m 18 s)
Start Date: 2022:06:04 12:29:28
End Date: 2022:06:04 12:29:47
Untagged: sandbox:latest
Deleted: sha256:d4d0693ecfae03298e63813da4d4780956235ff4cc54ef7170cc4a23f6ca970
PS C:\Users\erkan\OneDrive\MasaÅüstÅ\bitirme\composetest>
```

Figure 4.3: Test case 3

5. CONCLUSIONS

Sandbox is a virtual environment that is unaffected by your computer or network. It's essentially a closed-off testing environment.

Sandbox, for example, is used by software developers to test new scripts and eliminate programming problems. Sandboxes are used in information security to test and execute harmful applications.

In practice, a sandbox is used to test files, attachments, URLs, and applications to see if they are harmful or not.

Simply described, a sandbox is a secure environment where files, attachments, URLs, and applications may be tested before being sent to the end user. This verification might take anything from a few seconds to many minutes.

The key benefit is that whatever occurs in the sandbox remains in the sandbox. Because of this, as well as its high analytical capacity, the sandbox proves to be an effective weapon against zero-day exploits, which are dangerous threats that have yet to be detected by security software.

Investing in a sandbox to increase the security of your systems and business comes with a lot of benefits. Take a look at the most crucial ones.

- Block malicious links.
- Fight malicious attachments.
- Avoid zero-day threats.
- Keep information and data secure.
- Prevent data breaches.

As a result, sandboxes are a critical tool for defending against cyber-attacks. It can stop a lot of assaults if utilized appropriately. Sandboxes provide us a leg up when it comes to the security of our important systems, especially when signature-based end point products aren't enough.

APPENDICES

Appendix 1: Some publications

[1]Brushi, D., Martignoni, L., and Monga, M. Using Code Normalization for Fighting Self-Mutating Malware. In Proceedings of the International Symposium of Secure Software Engineering (Arlington, VA, 2006).

[2]Chess, D. M., and White, S. R. An Undetectable Computer Virus. In Proceedings of Virus Bulletin Conference (2000)

[3]Cohen, F. Computational Aspects of Computer Viruses. Computers Security, 8 (1989), 325—344

[4]Virus total[Online], <https://www.virustotal.com/> , [Visited 13 january 2022]