CS5231 System Security

Semester 1 AY2022/2023

System Audit Analysis Report

Project Report

Team 12

Team Member	Matric Number
Neo Ken Hong Kelvin	A0154598M
Tan Jia Le	A0159052H
Tay Zheng Yao Schuyler	A0154604L
Lee Kai Wen Aloysius	A0154597N

Table of Contents

Introduction

The world today is much more interconnected than it was a decade ago, with more devices from internet-of-things sensors to smartphones and desktops connected to the internet. As such, the consequences of cyber-attacks and espionage are greater as adversaries can affect industrial sectors and government agencies, causing data breaches or denial-of-service to critical infrastructure. Incidents such as the Stuxnet virus and WannaCry ransomware have shown that there is a lack of system security protection within many critical areas around the world, and attacks will only continue to increase as the incentives for successful attacks grow more attractive. Therefore, there is a need for monitoring and detecting malicious activities early within the system to mitigate the attacks from occurring.

Objectives

The objective of the project is to perform an analysis on the provided malicious program given in VM based on the logs obtained from auditd. The analysis would be to identify the root cause of the malware, processes created, and files interacted with during the runtime of the program. To better understand the malicious program provided, the team has built a program to visualise the processes created by other process, file accesses by the programs and the syscalls made during its runtime. The goal of the program is to aid analysts in identifying key information needed during the early stages of the analysis.

Project Approach

As specified in the project brief, logs from auditd were collected when the ./run-attacker.sh was ran. The auditd service was started before the execution and stopped after the script was run, and the logs was collected and sent to Elasticsearch for parsing. After sending the logs through to Elasticsearch, the parsed logs can be queried, and analysis can be performed to identify the indicators of compromise performed by the malicious program. After analysing, a python-based program is developed, to parse the logs and perform early-stage analysis on the logs to visualise key information that helps with understanding the processes actions and interactions with the system. This is to make early-stage analysis easier and less tedious as compared to performing analysis using Elasticsearch.

Malicious Program Analysis

Access of Secret File

@timestamp	process.pid	process.parent.pid	process.name	auditd.data.syscall	process.executable	file.path
Oct 13, 2022 @ 16:27:42.263	10113	10112	program11	openat	/home/student /Downloads/program11	/home/student/secret /secret.txt

Figure 1: Event Log of Access to secret.txt

As shown in Figure 1 above, the program11 executable was observed to have used the “openat” syscall to open secret.txt in the student’s secret directory.

@timestamp	process....	process.name	auditd.data.syscall	file.path	auditd.sequence	auditd.data.exit
Oct 13, 2022 @ 16:27:42.263	10113	program11	openat	/home/student /secret/secret.txt	3,054,737	3
Oct 13, 2022 @ 16:27:42.263	10113	program11	fstat	-	3,054,738	0
Oct 13, 2022 @ 16:27:42.263	10113	program11	read	-	3,054,739	23

Figure 2: File Interaction Syscalls with Secret File

Immediately after the "openat" syscall are a "fstat" and a "read" syscall. In particular, the "read" syscall resulted in a return value of 23, which was the length of the contents in secret.txt. With these logs, it was indicative that the program11 executable had successfully read the contents of secret.txt.

@timestamp	auditd.sequence	process.name	process.pid	process.parent.pid	auditd.data.syscall	auditd.data.exit	file.path
Oct 13, 2022 @ 16:27:42.263	3,054,743	program11	10113	10112	openat	4	/home/attacker /output.txt
Oct 13, 2022 @ 16:27:42.263	3,054,744	program11	10113	10112	fstat	0	-
Oct 13, 2022 @ 16:27:42.263	3,054,745	program11	10113	10112	write	200	-

Figure 3: File Interaction Syscalls to Output File

Finally, the program11 executable opened the output.txt file in the attacker's home directory before performing a "write" syscall. This indicated that the attacker was successful in extracting the contents of secret.txt.

Origin of Program11

@timestamp	process.pid	process.parent.pid	process.name	auditd.data.syscall	process.executable	file.path
Oct 13, 2022 @ 16:27:42.259	10112	9976	sudo	stat	/usr/bin/sudo	/home/student /Downloads/program11
Oct 13, 2022 @ 16:27:42.263	10113	10112	program11	execve	/home/student /Downloads/program11	/home/student /Downloads/program11

Figure 4: Program execution Syscall of malicious program

When a program is executed, an "execve" syscall replaces the caller process' image with the image of it. As shown in Figure 3 above, program11 was spawned from a sudo process, based on the process ID and parent process ID.

@timestamp	auditd.sequence	process.name	process.parent.pid	process.pid	auditd.data.syscall	process.args
Oct 13, 2022 @ 16:27:31.031	3,034,758	curl	9976	10072	execve	[curl, localhost:8080/programs/program11, -o, /home/student/Downloads /program11]
Oct 13, 2022 @ 16:27:31.043	3,035,858	chmod	9976	10074	execve	[chmod, 777, /home/student/Downloads/program11]
Oct 13, 2022 @ 16:27:42.251	3,053,420	sudo	9976	10112	execve	[sudo, /home/student/Downloads/program11]

Figure 5: Multiple preceding program execution Syscalls

Preceding "execve" syscalls showed that the program11 executable was downloaded from a web service via curl and its permission was changed to allow reads, writes and executions by any user on the system. These actions had set up the program11 executable to be executed by the attacker to exfiltrate the contents of secret.txt.

Provenance Graph Visualisation

Development

Python 3 and Graphviz, a graph visualisation software, were used to develop the process tree script, the process-file access script and the syscall graph script. These three scripts use auditbeat logs to visualise the behaviour and file accesses by processes in a tree or graph structure and would assist analysts by providing a high-level visualisation of program behaviours and interactions during system investigations.

Process Tree Graph

The process tree script generates a tree structure image that displays the association between parent processes and any forked child processes, as well as any in-process replacement such as by the `execve` syscall. Additional information displayed in the visualisation include the process ID, the program name, the path of the executable, the user that initiated the process and any command line arguments passed to the program. Such a graph would provide system analysts with a high-level visual overview of process behaviours with some basic details and allow them to quickly identify suspicious processes based on their child processes and program arguments.

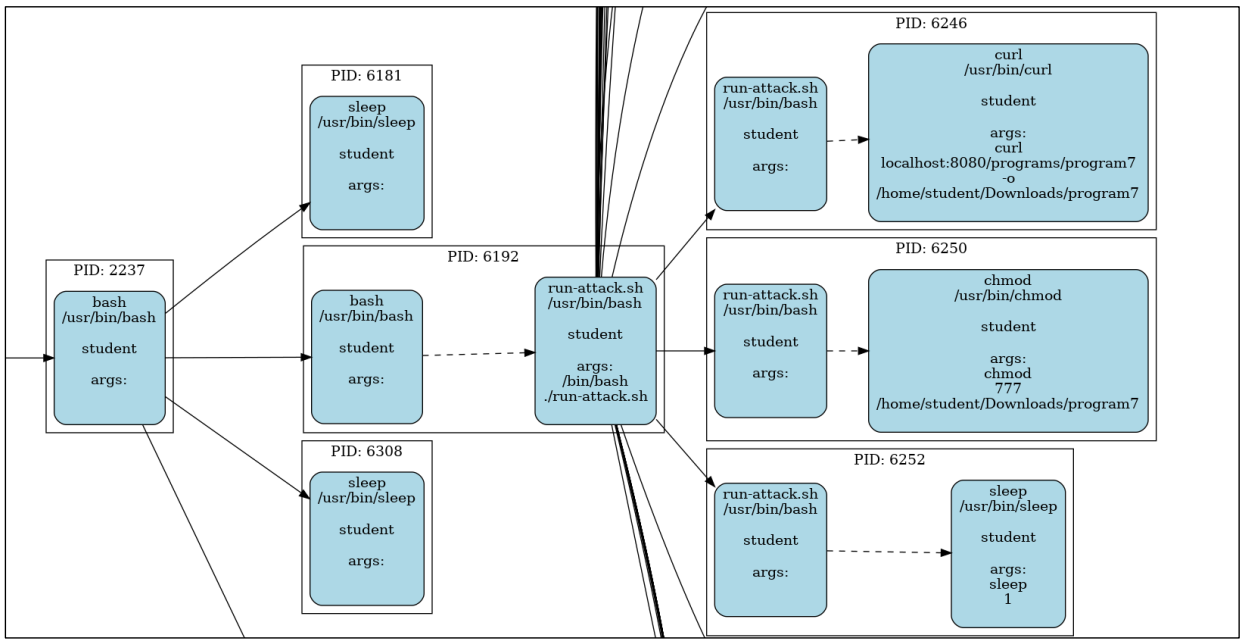

Figure 6: Process Tree Graph

Process-File Access Graph

The process-file access script generates a graph image that displays processes and the files each process had accessed. The information provided for each process include the program name, the process ID, the executable path, the user that initiated the process and the number of files opened by that process while the information for each file opened consists of the file path and the times opened by various processes. The directed edge from a process to a file also indicates the number of times that process has opened that file. The process-file access graph provides an overview of the files opened by processes and would allow system analysts to identify suspicious programs based on the files that they have opened.

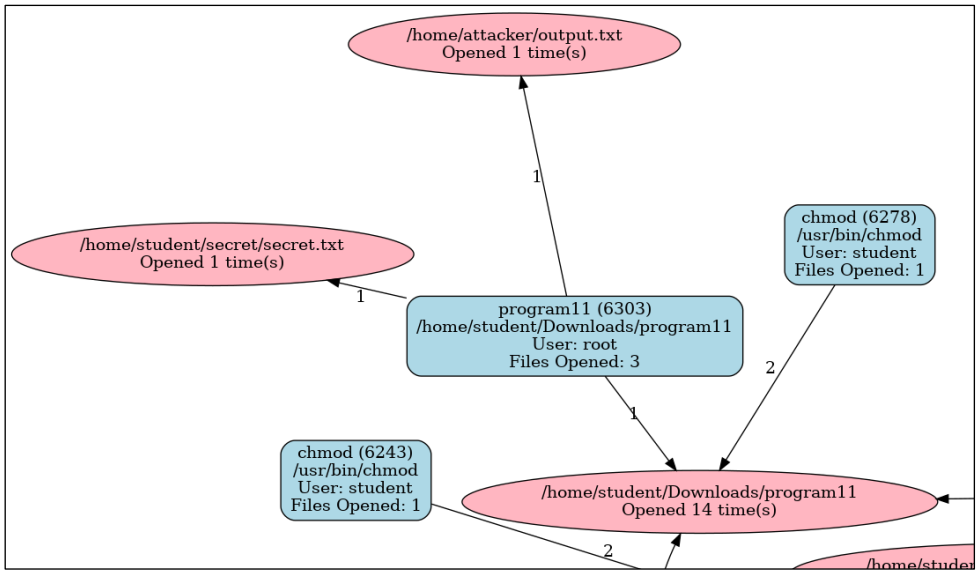

Figure 7: Process-File Access Graph

Syscall Graph

The syscall graph script generates a graph image that displays the syscalls within a selected program's execution and their respective interaction with a file. The information provided for each syscall include the auditbeat sequence number, the syscall, the process ID, the program name, the result of the syscall and the exit status. Certain syscalls such as "openat", "read" and "write" is also shown associated with a file descriptor, either a file, standard input or output, via a directed edge. The syscall graph provides a graphical view of a program's syscalls and their respective interaction with a file, thus allowing system analysts quickly gain an understanding of a program's execution behaviour.

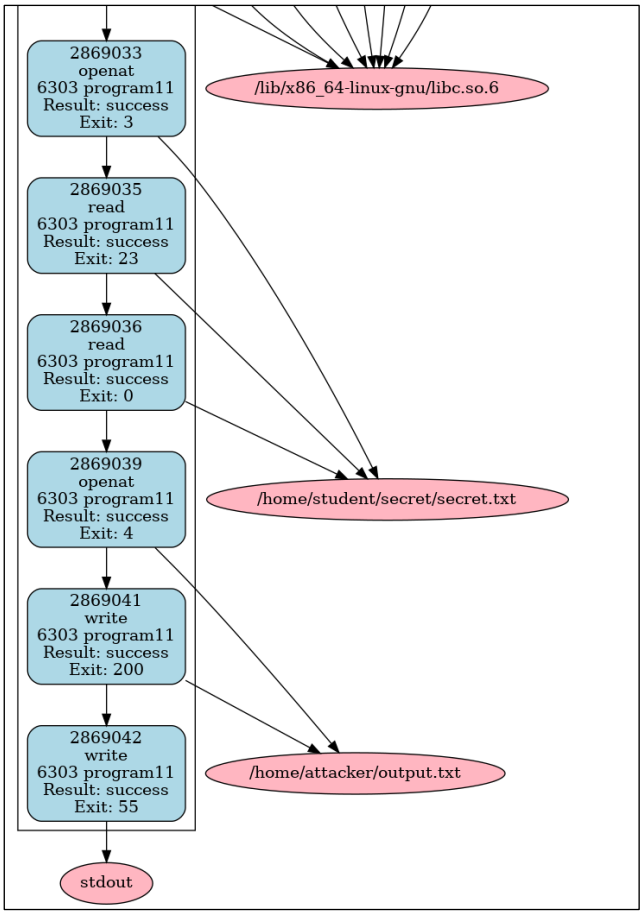

Figure 8: Syscall Graph

Difficulties

Inconsistent Collection of Logs from Auditbeat

In the early stages of the project, the team noticed that certain actions executed by the binaries were not appearing within the collected Auditbeat logs. After performing repeated runs and cross-analysis between the logs, the team identified that Auditbeat is unable to keep up with the amount of activity generated by the operating system and programs running. As such, Auditbeat would drop the log entry, causing certain activities such as the opening of file to not be recorded in only certain instances of the repeated runs.

To solve the issue, the team employed three approaches to reduce the chance of logs being dropped. Firstly, the virtual machine's resources were increased. Specifically, the CPU core count and RAM allocation was doubled to ensure parallelization was not bottlenecked. Secondly, the addition of filters to the configuration file of Auditbeat will reduce the disk utilization as unnecessary logs are removed from the final collection. Lastly, the attack script was changed to include longer sleep between execution to ensure that critical activities can be captured by Auditbeat. After performing the changes, the amount of log activities dropped decreased significantly and the team was able to perform analysis and create new visualisations based on the logs collected.

Formulating Simple and Meaningful Visualisations

As one of the main objectives of the project was to create an intuitive overview of events within the logs, it is important that the tool generates simple graphs for analysts to quickly understand the

workflow. As such, one main challenge in the project was reading the raw log files and extracting the key elements of the activity to be used to draw the graphs.

Conclusion

To achieve the objective of understanding the malicious behaviour of the programs found in the virtual machines, initial analysis must be done on the raw logs, which could then be used to identify the tools required to visualise the behaviour of the malicious program and perform the necessary corrective actions to prevent the attack. From the analysis of raw logs, the team has developed three programs that visualises the Auditbeat logs collected in the machine. The programs can be used to visualise the files opened by the various processes in the operating system, as well as to understand the execution flow of the processes. These tools will greatly aid the analysts with the initial understanding of potential malicious activities within the system as they can identify the processes that accessed sensitive files within the system and understand their behaviour with the syscalls they performed.

References

Elasticsearch: The Official Distributed Search. (n.d.). Retrieved from Elastic: https://www.elastic.co/elasticsearch/

Graphviz. (n.d.). Retrieved from Graphviz: https://graphviz.org/