Ransomware Implementation

HPY 413

January 5, 2026 – January 14, 2026

Overview

Ransomware is a form of malware used for extortion: attackers deny access to systems or data—most commonly by encrypting files—and demand payment (often in crypto-currencies) in exchange for a decryption key or recovery instructions. Many modern campaigns also steal data before encryption and threaten to leak it if the victim refuses to pay, increasing pressure even when backups exist. In organizational intrusions, ransomware is often the end-stage payload after initial access (e.g., phishing, stolen credentials, or exploited vulnerabilities), followed by privilege escalation and lateral movement to maximize impact. Defense typically focuses on preventing initial compromise, limiting spread, and ensuring reliable recovery via well-tested, protected backups, and incident response procedures.

1 Prerequisites

- Make sure you have completed the prerequisite Access Control Logging assignment, since the access-logging tool you implemented there is required for this assignment.
- Docker installed on your computer

2 Getting Started

2.1 Setting Up Docker

Docker is an open-source platform that automates the deployment of applications within lightweight and portable containers. Unlike traditional virtual machines, containers share the host operating system's kernel but run in isolated user spaces. This means that in a containerized application, you will build/work exactly the same way on your laptop as it does on a production server, eliminating the "it works on my machine" problem

Installation Guides

For detailed, step-by-step installation instructions specific to your operating system, please refer to the following links:

Docker Desktop for Windows:
https://docs.docker.com/desktop/install/windows-install/

Docker Desktop for Mac:
https://docs.docker.com/desktop/install/mac-install/

Docker Engine for Linux (Ubuntu):
https://docs.docker.com/engine/install/ubuntu/

General Installation Overview:
https://docs.docker.com/get-docker/

Implementation

3.0 Environment Setup

A dedicated Docker image has been prepared for this course and is available on Docker Hub here. This image provides a controlled Ubuntu-based lab environment with the required tooling preinstalled, so that you can complete the assignment without installing dependencies on your host system. Working inside the container also reduces the risk of accidentally modifying or encrypting important files on your machine, since your work can be confined to explicitly mounted lab directories (bind mounts).

1) Download the image

Run:

```
docker pull kostasamplia/ransom-lab-ubuntu:2025 # or kostasamplia/ransom-lab-ubuntu:latest
```

.
or through the DockerHub and Docker Desktop:

`https://hub.docker.com/r/kostasamplia/ransom-lab-ubuntu`

2) Prepare local (host) directories

From an empty working directory (this will be your lab workspace), run:

```
mkdir -p assignment data target
```

Use the directories as follows:

- `./assignment`: Your source code (including the access-logging tool from the prerequisite assignment).
- `./data`: Logs and other generated artifacts that must persist on the host.
- `./target`: Output directory for generated/encrypted test files.

  Use **./target** directory only as a safe directory while developing and debugging your ransomware logic. This will save you some frustration when trying to restore your files after an unsuccessful encryption-decryption sequence.

  In other words, the final implementation must encrypt the files **in place**. **Any try that will encrypt files in target directory as an end product is considered wrong**.

3) Run the container with bind mounts

From the same directory, start the container and mount the host folders into it:

```
docker run --rm -it \
  -v "$(pwd)/assignment:/home/student/assignment" \
  -v "$(pwd)/data:/home/student/data" \
  -v "$(pwd)/target:/home/student/target" \
  kostasamplia/ransom-lab-ubuntu:2025 \
  bash
```

You can put several test files **that are not important to you** in the ***/home/student/data*** directory and via the mount option. The files will appear in your container environment directly.

3.1 Ransomware Implementation

In this assignment you will implement a Bash script named **ransomware.sh** that simulates a simple ransomware workflow within the ***assignment*** directory. The script must do the following:

1. create or select all the files inside ***/home/student/data*** directory
2. generate corresponding encrypted versions of those files
3. delete the original, unencrypted files once encryption has completed.

The script must also support generating a large volume of files on demand. Given a target directory and an integer as command-line arguments, ransomware.sh should create the respective files in that directory before performing the encryption and deletion steps described above.

It's important for the bash script to be able to get a target directory (for testing purposes).

For the encryption you can use any cryptographic algorithm of your choice. Openssl is already installed in the given image. In any case, do not forget to mention the algorithm you chose and how you have integrated it to the ransomware file.

Bonus: You can try to use your RSA implementation in order to encrypt decrypt the files. If you do so, include how the implementation have been imported to the container and what were the changes you've made in order for the implementation to work (if any).

3.2 Run the ransomware file in the /data directory

After completing the **ransomware.sh** try to run it in the **/data** directory. What is the outcome of this action and why ? Does this remind you of a specific principle that companies apply ?

3.3 Access Control Logging tool

Your ransomware simulation must use the shared library from Assignment 3 ("Access Control Logging"), named **audit_logger.so**, so that each relevant file operation is recorded in a system-wide log file called **access_audit.log**, every intercepted operation must generate a new log entry in append mode, using the exact same schema as Assignment 3.

Each log entry must include:

- the user ID (UID via getuid())
- the process ID (PID via getpid())
- the absolute filename (meaning that the path
- the event date and time in UTC, an operation code (0 = create, 1 = open, 2 = write, 3 = close)
- a denied flag (1 if denied due to insufficient privileges, otherwise 0)
- a SHA-256 hash of the file contents (via OpenSSL EVP)

3.4 Detect the ransomware by enriching the Access Control Log Monitoring tool

Extend your Access Control Monitoring tool (`audit_monitor.c`) from the previous assignment by implementing additional detection logic aimed at identifying behavioral indicators consistent with ransomware activity. In this phase, the monitor should leverage the audit trail produced by the previously developed shared library (`audit_logger.so`) and flag patterns that suggest large-scale file staging and encryption.

Required detection capabilities

1. **High-volume file creation (burst activity)** Ransomware commonly attempts to conceal malicious artifacts by operating in directories that already contain a large number of files. A typical precursor behavior is the rapid creation of many new files. Your monitor must detect whether at least X files were created within the last 20 minutes, where X is a threshold value defined by the assignment (or provided as an input parameter).

2. **Encryption-and-delete workflow detection.** Another characteristic pattern is opening an unencrypted file, producing an encrypted counterpart, and then removing the original file. Your monitor must identify and report all log events indicating that an unencrypted file was opened and an encrypted version was created. For the purposes of this assignment, encrypted files are identified by the suffix ***.enc***.

3.5 Theoretical Questions

1. Why can "least privilege" and role separation reduce ransomware blast radius, even if the initial infection happens on a user workstation? Give a concrete example involving shared drives or administrative credentials.

2. Why do "offline" or "immutable" backups reduce ransomware risk compared to ordinary network-accessible backups?

3. Why do many ransomware attacks try to delete backups and recovery points before encrypting files, and what is one simple defense strategy that specifically targets this behavior?

Notes

After the assignment is complete you're required to submit the following:

1. A **ransomware.sh** file that will contain the encryption/decryption options as well as the generation of multiple large files in bash.

2. The **Makefile**, **audit_logger.c**, **audit_logger.so** and **audit_monitor.c** with the changes made to fulfill the assignment's requests.

3. A **pdf** report of your course of action including your approach to every step of the respective sections as well as the answers to the theoretical questions. You can include screenshots or pieces of code if you think such cases will make it clearer to understand your implementation.

Use of AI-based tools (e.g. ChatGPT or other agentic systems) is permitted (yet not suggested), provided that you fully understand and can explain your work, as your understanding on the specifics will be examined thoroughly.

Questions

Due to extensive load of work, it could be cases that I miss some threads in "$\Sigma\upsilon\zeta\acute{\eta}\tau\eta\sigma\eta$". In such a case, if you have unresponded threads for more than a few hours, do not hesitate to send me a personal message at 'kamplianitis@tuc.gr'.

Please, upon finding solutions to possible issues with any of the assignments, use the "$\Sigma\upsilon\zeta\acute{\eta}\tau\eta\sigma\eta$" to open a thread describing the issue as well as a reply to that thread with the way of resolving it. It is very important for us in order to maintain the assignment through the years and become better. Every comment is much appreciated.