Security Engineering - Lab 1

Harris A. Ransom

February 21, 2025

Introduction

A key aspect of security engineering is understanding the "security mindset" - that is, thinking like an attacker and understanding the assets, vulnerabilities, risks, and mitigations present in a given system. This lab report will demonstrate these concepts, along with security goals like the CIA triad, in the context of a Linux environment. **Note:** A full listing of the commands run during this lab can be found in this text file.

Step 1: Acquire Your Weapon (Docker Image)

```
harris gaming-pc ../Lab01 docker pull dlambros/labs:lab1
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock:

harris gaming-pc ../Lab01 sudo !!
sudo docker pull dlambros/labs:lab1
[sudo] password for harris:
lab1: Pulling from dlambros/labs
1f3e46996e29: Pull complete
b256024cad64: Pull complete
7fcd542b154c: Pull complete
c05ecd34ca12: Pull complete
e3dc232d3007: Pull complete
9d80dfc3f984: Pull complete
1ccaf9de5370: Pull complete
416135818691: Pull complete
7dd1d215b257: Pull complete
d7f4a5f9f0d8: Pull complete
91eab41e4559: Pull complete
31a975dee62f: Pull complete
405cfbb05094: Pull complete
4f4fb700ef54: Pull complete
9240dd240202: Pull complete
ee475210e086: Pull complete
Digest: sha256:e024dfa9c379117a106b1d1f38b7ee4faf63aaeb1eb183d7f7b2fdbf1fe545a0
Status: Downloaded newer image for dlambros/labs:lab1
docker.io/dlambros/labs:lab1
```

Figure 1: Docker Pull Command Output

```
harris gaming-pc ../Lab01 docker images
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock:

harris gaming-pc ../Lab01 sudo !!
sudo docker images
REPOSITORY          TAG         IMAGE ID       CREATED         SIZE
dlambros/labs       lab1        043596fdeff2   35 hours ago    11.1MB
```

Figure 2: Docker Images Command Output

Step 2: Enter the Sandbox (Run the Container)

Figure 3: Docker Run Command Output

Step 3: Linux Kung Fu (Basic Commands)

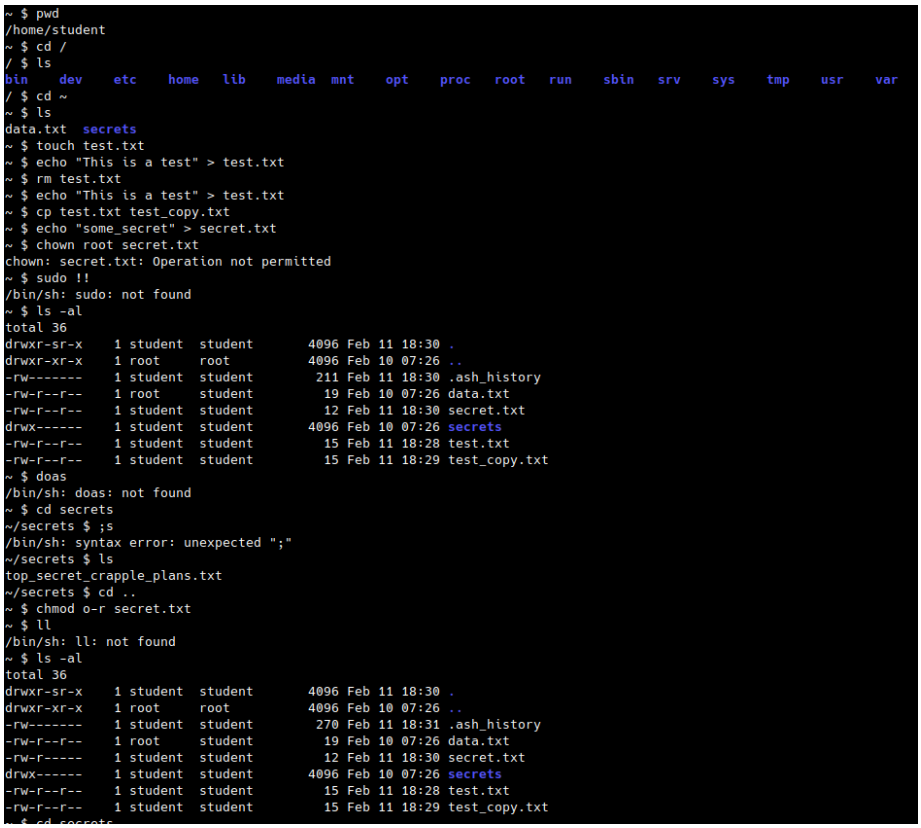

Figure 4: Basic Linux Commands Output

Step 4: The CIA Triad and Other Spooky Stuff

1. **Confidentiality:**

```
~/secrets $ ls -al
total 12
drwx------    1 student  student       4096 Feb 10 07:26 .
drwxr-sr-x    1 student  student       4096 Feb 11 18:30 ..
-rw-r--r--    1 root     root            22 Feb 10 07:26 top_secret_crapple_plans.txt
~/secrets $ cd ..
~ $ ls
data.txt       secret.txt     secrets        test.txt       test_copy.txt
~ $ touch top_secret_crapple_plans.txt
~ $ echo "Top sneaky stuff inside" >> top_secret_crapple_plans.txt
~ $ ls -al
total 40
drwxr-sr-x    1 student  student       4096 Feb 11 18:41 .
drwxr-xr-x    1 root     root          4096 Feb 10 07:26 ..
-rw-------    1 student  student        507 Feb 11 18:41 .ash_history
-rw-r--r--    1 root     student         19 Feb 10 07:26 data.txt
-rw-r-----    1 student  student         12 Feb 11 18:30 secret.txt
drwx------    1 student  student       4096 Feb 10 07:26 secrets
-rw-r--r--    1 student  student         15 Feb 11 18:28 test.txt
-rw-r--r--    1 student  student         15 Feb 11 18:29 test_copy.txt
-rw-r--r--    1 student  student         24 Feb 11 18:41 top_secret_crapple_plans.txt
~ $ chmod og-r
BusyBox v1.37.0 (2024-12-13 21:18:49 UTC) multi-call binary.

Usage: chmod [-Rcvf] MODE[,MODE]... FILE...

MODE is octal number (bit pattern sstrwxrwxrwx) or [ugoa]{+|-|=}[rwxXst]

        -R      Recurse
        -c      List changed files
        -v      Verbose
        -f      Hide errors
~ $ chmod og-r top_secret_crapple_plans.txt
~ $ ls -al
total 40
drwxr-sr-x    1 student  student       4096 Feb 11 18:41 .
drwxr-xr-x    1 root     root          4096 Feb 10 07:26 ..
-rw-------    1 student  student        566 Feb 11 18:42 .ash_history
-rw-r--r--    1 root     student         19 Feb 10 07:26 data.txt
-rw-r-----    1 student  student         12 Feb 11 18:30 secret.txt
drwx------    1 student  student       4096 Feb 10 07:26 secrets
-rw-r--r--    1 student  student         15 Feb 11 18:28 test.txt
-rw-r--r--    1 student  student         15 Feb 11 18:29 test_copy.txt
-rw-------    1 student  student         24 Feb 11 18:41 top_secret_crapple_plans.txt
```

Figure 5: Confidentiality Action (chmod)

From Fig. 5, we can see that the file `top_secret_crapple_plans.txt` that was created has read and write permissions for the owner and read permissions for the group and others. We can use the `chmod` command to remove read permissions from both the group and others, which allows only the owner of the file (student) to read and write to it.

In addition to the built-in Linux file permissions, Linux also provides a set of commands for handling Access Control Lists (ACLs) in the form of Set File ACL (`setfacl`) and Get File ACL (`getfacl`). These ACLs can provide more granular control over file permissions by allocating permissions for specific users

outside of the usual Owner-Group-Other structure. This can be particularly useful for scenarios where you normally want only the owner/group to have access, but want to grant exceptions to specific users. Software modules like Security Enhanced Linux (SELinux) also extend the traditional discretionary access controls of a Linux system to include additional policy and control mechanisms, particularly mandatory access controls.

It is important to understand the potential consequences of a breach of confidentiality for files like `top_secret_crapple_plans.txt` in order to implement effective and efficient security controls. Files like these can contain trade secrets (i.e. secret avocado toast recipe), important operational plans/procedures (i.e. coffee machine launch codes), or other types of sensitive information like Personally Identifiable Information (PII) or Personal Health Information (PHI). For any kind of sensitive data, it is important to protect the data's confidentiality to avoid the consequences of a data breach. These consequences can range from compliance and regulatory investigations to loss of revenue, additional security breaches, and loss of trust in the organization.

2. **Integrity:**

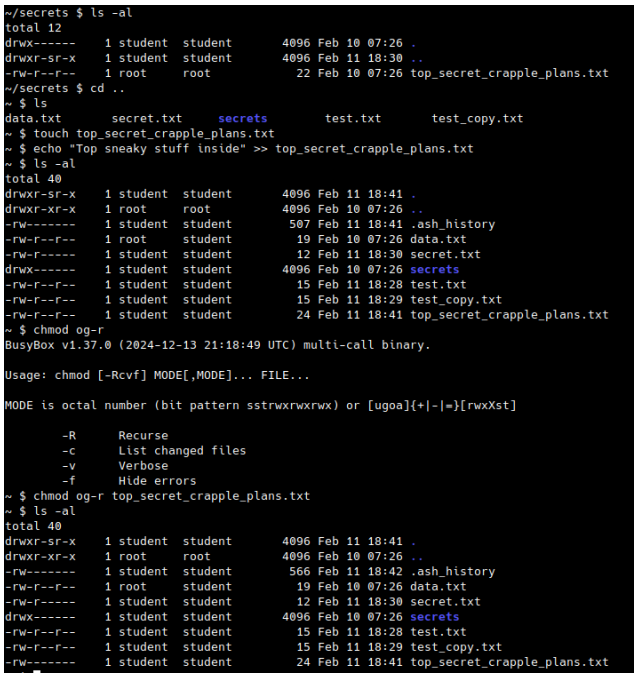

Figure 6: Integrity Action (hashing)

Out of the listed hashing algorithms (MD5, SHA-256, SHA-512), SHA-512 is generally considered the most secure. MD5 has been deprecated for its cryptographic flaws, namely its lack of collision resistance. That is, it is easier to find two inputs that produce the same (or similar) hash with MD5 than it is with other hashing algorithms. Between the SHA-2 hash family algorithms listed (SHA-256 and SHA-512), SHA-512 is generally considered more secure due to its longer length/larger size, which, combined with an increased number of hashing rounds, increases its collision resistance when compared to its peers.

Digital signatures and checksums can also be used to verify the integrity (and authenticity) of data and users. Digital signatures take advantage of the properties of asymmetric cryptosystems and are often used in conjunction with hashes when performing data transmission and user authentication. Typically, a message is hashed and then that hash is "signed" by a user's private key. This ensures that a.) the message was created/verified by the intended user and b.) if any changes are made to the data, the hashes will be different and cannot be modified without knowledge of the private key. Thus, digital signatures and hashes can be used to verify the integrity and authenticity of data. Checksums also verify the integrity of data (but not the authenticity of the data) by deriving another small block of data from a given block of data. This small checksum block is then appended to the end of the provided data packet and sent together. For example, a parity bit or word can be computed and appended to the end of a data packet for error correction purposes. Layer 2 network protocols like Ethernet also use checksums in the form of a 32-bit Cyclic Redundancy Check (CRC) for error-correction purposes, ensuring the integrity of the data sent across a data link.

Checksums can also be used to verify the integrity of software downloads. A common example of this is the process of downloading a Linux ISO. Software repositories for Linux distributions typically contain both the distribution's ISO file and the checksum (or hash) associated with that ISO file. The user can then download both the ISO file and its corresponding checksum and perform the calculation locally. If the calculated checksum matches the provided checksum, then the user can verify that the file has not been modified through either network errors or malicious modification. Of course, if an attacker is able to modify both the Linux ISO and the checksum file in the online repository, then it is possible that malicious modification could be left undetected as a user would see that the maliciously modified ISO and modified checksum match. Therefore, checksums are typically only used to verify software integrity against

non-malicious errors (e.g. network errors), and are not effective against remote modification.

3. **Availability:**

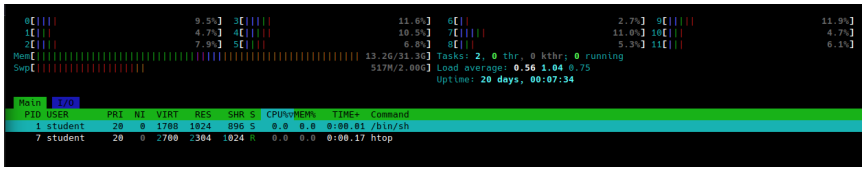

Figure 7: Availability Baseline Resource Usage

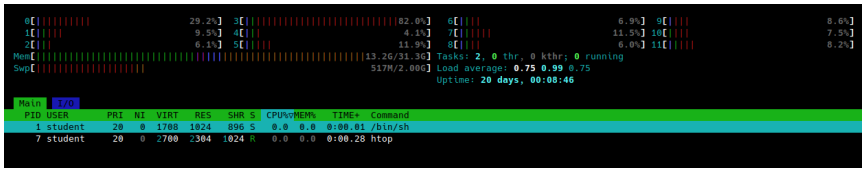

Figure 8: DOS Resource Usage

System downtime can directly impact availability by making the services running on said system unavailable to the intended users. For example, if a mail server goes down due to scheduled maintenance or a remote attack, all of the users that want to use that service will not be able to. Systems going down can also have a cascading effect, leading to a loss of service across parts of network (or even the entire networks) if appropriate safeguards and backups are not put into place. To simulate a Denial of Service (DoS) attack against a Crapple server, a shell script was created that generates a large number of files with random data.

On modern computer systems, this script will not have a significant impact on system availability. Due to multi-core architectures, smart process scheduling, and memory paging, the effects of even a larger DoS attack could be mitigated. If, instead, a number of distributed systems were sending/receiving large network packets or executing many remote commands, this Distributed Denial of Service (DDoS) could have a severe impact on system availability and even cause the system to crash. We can see that, when comparing Figs. 7 and 8, the simulated DoS attack only had a minimal impact on processor and memory resources. This kind of attack simulates a *flooding* or brute-force DoS attack,

in which an attempt is made to overwhelm a system by "flooding" it with data/packets and thus reduce the available system resources. This contrasts with an *amplification*-style attack (commonly used for networked services), in which another network service is used to amplify the total bandwidth sent to a victim machine. Common examples of amplifier services are NTP and DNS.

Businesses like Crapple can be severely impacted by DoS and DDoS attacks. Impacts can include, but are certainly not limited to, loss of revenue, loss of customer trust, and disabling of essential services. Mitigations like load balancers, firewalls, instrusion prevention systems (IPS), and resource monitoring software can all help to reduce the impact and risk of DoS-style attacks.

4. **Least Privilege:**

```
/home $ cd /home/sensitive_data/
/bin/sh: cd: can't cd to /home/sensitive_data/: Permission denied
/home $ ls -al
total 32
drwxr-xr-x    1 root     root          4096 Feb 10 07:26 .
drwxr-xr-x    1 root     root          4096 Feb 11 18:26 ..
drwx------    1 user1    user1         4096 Feb 10 07:26 sensitive_data
drwxr-xr-x    1 user2    user2         4096 Feb 10 07:26 shared_data
drwxr-sr-x    1 student  student       4096 Feb 12 03:33 student
drwxr-sr-x    2 user1    user1         4096 Feb 10 07:25 user1
drwxr-sr-x    2 user2    user2         4096 Feb 10 07:25 user2
/home $ cd /home/shared_data/
/home/shared_data $ ls -al
total 12
drwxr-xr-x    1 user2    user2         4096 Feb 10 07:26 .
drwxr-xr-x    1 root     root          4096 Feb 10 07:26 ..
/home/shared_data $ cd ..
/home $ whoami
student
/home $ groups
student
/home $ cd shared_data/
/home/shared_data $ touch test.txt
touch: test.txt: Permission denied
/home/shared_data $ cd ..
```

Figure 9: Least Privilege Action (chmod)

From Fig. 9, it can be seen that the `sensitive_data` and `shared_data` folders are owned by user1 and user2, respectively. We can also see that the `sensitive_data` directory only allows read/write/execute access to the owner of the directory (i.e. user1). Therefore, our user (student) cannot access that folder. Likewise, for the `shared_data`, `user1`, and `user2` folders, the "other" user only has read and execute permissions (no write access, since our student user is not part of the same group for user1/user2). Since we do not have write access in any of these folders besides our own home directory, we cannot `touch` files to create new ones. Therefore, the student user can only read/execute files

in the `shared_data`, `user1`, and `user2` directories, and they cannot access the `sensitive_data` directory.

These privileges effectively demonstrate the principle of least privilege. The student user is only able to view files in non-sensitive folders, and can only write to folders they have ownership of. This reduces the chance of a privilege escalation attack succeeding since it limits the potential vulnerabilities available to exploit due to improper access control.

5. **Attacker Mindset:** To attempt to steal the avocado toast recipe, some kind of privilege escalation would be required. We want to move from our *student* user to a more privileged user like *user1* or *user2*, since they are the owners of the *shared_data* and *sensitive_data* directories.

   For example, a possible privilege escalation attack using other binaries (e.g. SUID/SGID binaries) on the system could give us initial access as user1. From there, we would have access to the sensitive_data directory. We can also consider a *social engineering* attack, where, for example, a phishing email or link could be sent to the owner of the user1 account in an attempt to get the credentials/access to their data. A social engineering attack could also be performed on the system administrator (root), which would bypass all of the directory permission restrictions if granted access. There is also the possibility of finding a vulnerable network service tied to the root user or user1, allowing us to gain a shell as one of those users and access the sensitive data.

Step 5: Secrecy, Trust, Threats, and Vulnerabilities:

1. **Secrecy and Trust:** First, we will define "secrecy" as the ability for two or more parties to communicate information/data secretly (confidentially). We will also define "trust" as achieving the security goal of authenticity. From these definitions, it can be said that it is not necessary to have secrecy in order to have trust (i.e. authenticate users). For example, a physical access control mechanism like a gate or barrier can trust that a user is who they say they are based on a fingerprint and RFID badge without needing to keep that information confidential. In this case, the lack of confidentiality as a requirement stems from the fact that the combination of the RFID badge and fingerprint is not easily duplicatable. However, to have secrecy there is msot likely some degree of trust/authenticity required. Even if two parties communicate their

information in an encrypted manner, it is entirely possible for one party to lie about their identity (e.g. in a Man in the Middle attack) and breach the confidentiality of the encrypted information. This is where cryptographic authentication and encryption protocols like Kerberos and PKI are particularly useful. In these protocols, trust is placed in a known Trusted Third Party (TTP), so even if trust is low between two entities (e.g. a server and a user), they can still establish secrecy through the aid of this TTP.

2. **Threats and Vulnerabilities:** A *threat* is any potential actor or event that may exploit a vulnerability (or vulnerabilities) to compromise a system's security (i.e. confidentiality, integrity, and availability). On the other hand, a *vulnerability* is a flaw or weakness in a system that may be exploited by a threat to cause harm. These definitions are broad in order to encompass the various types of threats and vulnerabilities present in security engineering.

   For this lab scenario, we can perform a basic risk assessment of Crapple's systems by inventorying threats, vulnerabilities, and threat likelihoods, along with the potential impact of a given risk occurring.

   Threats:

   - Password Brute-Forcing
   - Denial of Service (DoS) Attack
   - Social Engineering/Phishing
   - Insider threat

   Vulnerabilities:

   - Broken/improper file access control
   - Lack of load balancing/DoS mitigations
   - Privilege escalation (e.g. SUID/SGID binaries)

   Threat Likelihoods:

   - Password Brute-Forcing: Medium
   - Denial of Service (DoS) AttacK: High
   - Social Engineering/Phishing: High
   - Insider Threat: Low

Potential Impacts:

- Loss of revenue
- Loss of trust/reputation
- Information leakage
- Destruction of essential business data
- Disabled systems/services

Step 6: Reflection Time (aka "Thinky Thoughts"):

1. The principle of **least privilege** is a helpful guide to achieve the goals outlined in the CIA Triad, as it attempts to reduce the exposed attack surface provided by users and services of a given system. By giving every user/service the lowest amount of privilege required to do its job, by definition it ensures that each principal can only affect the entities it has access to, thus ensuring the confidentiality and integrity of the entities that should not be exposed to principals that do not have the necessary permissions. In providing these security controls, availability is still ensured since each principal at a minimum has enough access to perform their role successfully.

2. A few examples of real-world security breaches caused by poor access controls include:

   - The 2013 Target data breach, in which a third-party vendor was compromised and allowed attackers to gain a foothold into Target's network. This is an example of a *supply-chain attack*, in which a separate dependency is compromised, ultimately leading to the compromise of the entire system. Better access control would have limited the privileges given to the third-party vendor, preventing the attackers from pivoting into Target's network and gaining admin access.
   - Kevin Mitnick's attack against Motorola in which he used social engineering to get a receptionist to send the source code for a new mobile phone over to a remote FTP server. Better access control would have a.) prevented the receptionist from accessing the source code, b.) only allowed access to the source code after external/secondary identity verification, and c.) blocked the source code from being sent to an anonymous FTP server.

- The leaking of classified offensive hacking tools and software from the NSA by The Shadow Brokers in 2017. While the exact cause of the incident is unknown, the two commonly supported theories are an insider threat or a foreign APT (most likely Russia). Ultimately, a lack of access control lead to the files being extracted and leaked. Even if an account that has permission to access the files was compromised, additional security controls in the form of secondary approval for access along with monitoring and logging software should have been in place to reduce the chance of this kind of attack succeeding.

3. We explored above how vulnerabilities like SUID and SGID binaries can lead to an attacker executing binaries with the permissions of the owner of the file rather than their own permissions, leading to elevated privileges. Attackers can also exploit weak file permissions on files that contain sensitive data (e.g. /etc/passwd and /etc/shadow) to elevate their privileges. For example, if there is write access to users other than root for /etc/passwd or /etc/shadow, a malicious user can create a new user entry in those files that has root-level access with a password known to them, allowing the malicious user to gain root access. Weak file permissions on files that contain data such as cryptographic keys (e.g. SSH private keys) can also allow privilege escalation and loss of confidentiality/integrity for the information said cryptographic secrets protect.