System Security Lab Report

Submitted By: Shudarsan Regmi
CH.SC.U4CYS23055

Lab Exercise 1

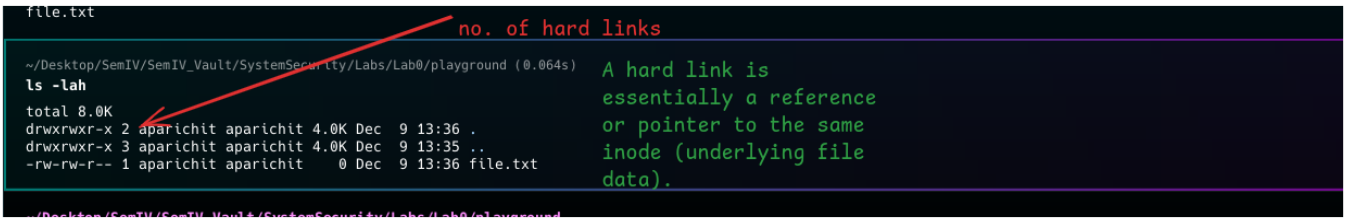
Implementing the discretionary access control (DAC) mechanism in operating Systems (Linux)

Checking for a file Permission

```
ls -lah
```

```
~/Desktop/SemIV/SemIV_Vault/SystemSecurity/Labs/Lab0/playground (0.064s)
ls -lah file.txt
-rw-rw-r-- 1 aparichit aparichit 0 Dec  9 13:36 file.txt
```

**No. of hard links **



> Directories typically have multiple links because of their inherent structure (. and ..), whereas regular files usually have one unless explicitly linked using the ln command.

Checking for parent directory permission

```
ls -ld
```

```
~/Desktop/SemIV/SemIV_Vault/SystemSecurity/Labs/Lab0/playground (0.063s)
ls -ld
drwxrwxr-x 2 aparichit aparichit 4096 Dec  9 13:36 .
```

Changing the permission of a file

- u -> Users
- o -> Owner
- g -> Group
- a -> All

Changing file permissions

```
chmod u+x file.txt
```

```
chmod g+x file2.txt
```

```
chmod o+x file2.txt
chmod o-x file2.txt
chmod o+rwx file.txt
chmod o-rwx file.txt
```

```
chmod a-rwx file.txt
```

```
~/Desktop/SemIV/SemIV_Vault/SystemSecurity/Labs/Lab0/playground (0.064s)
ls -lah
total 8.0K
drwxrwxr-x 2 aparichit aparichit 4.0K Dec  9 13:36 .
drwxrwxr-x 3 aparichit aparichit 4.0K Dec  9 13:35 ..
-rwxrw-r-- 1 aparichit aparichit    0 Dec  9 13:36 file.txt

~/Desktop/SemIV/SemIV_Vault/SystemSecurity/Labs/Lab0/playground (0.061s)
chmod o+x file.txt

~/Desktop/SemIV/SemIV_Vault/SystemSecurity/Labs/Lab0/playground (0.064s)
chmod g+x file.txt

~/Desktop/SemIV/SemIV_Vault/SystemSecurity/Labs/Lab0/playground (0.065s)
ls -lah
total 8.0K
drwxrwxr-x 2 aparichit aparichit 4.0K Dec  9 13:36 .
drwxrwxr-x 3 aparichit aparichit 4.0K Dec  9 13:35 ..
-rwxrwxr-x 1 aparichit aparichit    0 Dec  9 13:36 file.txt

~/Desktop/SemIV/SemIV_Vault/SystemSecurity/Labs/Lab0/playground (0.061s)
chmod g-x file.txt

~/Desktop/SemIV/SemIV_Vault/SystemSecurity/Labs/Lab0/playground
```

```
~/Desktop/SemIV/SemIV_Vault/SystemSecurity/Labs/Lab0/playground (0.063s)
ls -lah
total 8.0K
drwxrwxr-x 2 aparichit aparichit 4.0K Dec  9 13:36 .
drwxrwxr-x 3 aparichit aparichit 4.0K Dec  9 13:35 ..
---------- 1 aparichit aparichit    0 Dec  9 13:36 file.txt
```

Using chmod number with permission

```
chmod uog file.txt
u = rwx bits (111) = 7
o = rwx bits (111) = 7
g = rwx bits (111) = 7
chmod 777
```

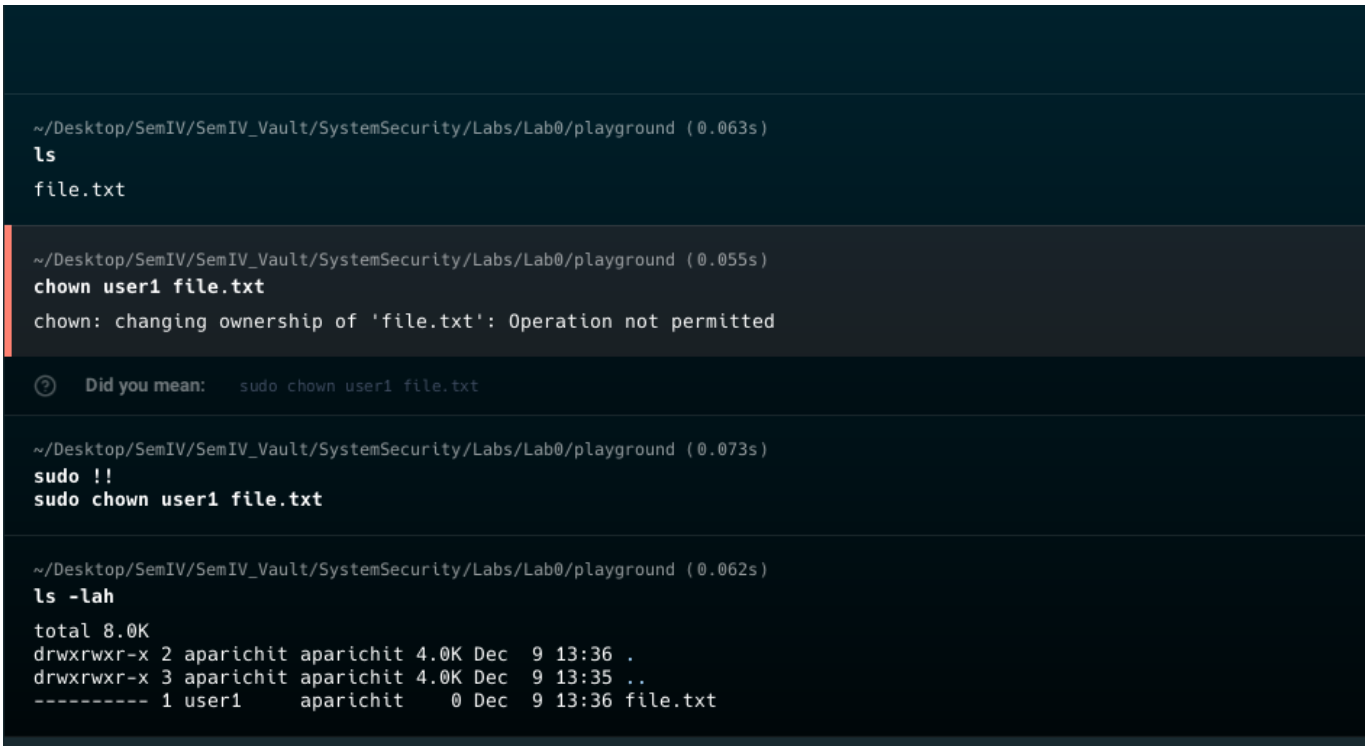
Changing permission recursively

```
chmod -R uog file.txt
chmod -R 444 file.txt
```

Changing the Ownership of a file

Changing the user

```
sudo chown user file.txt
```



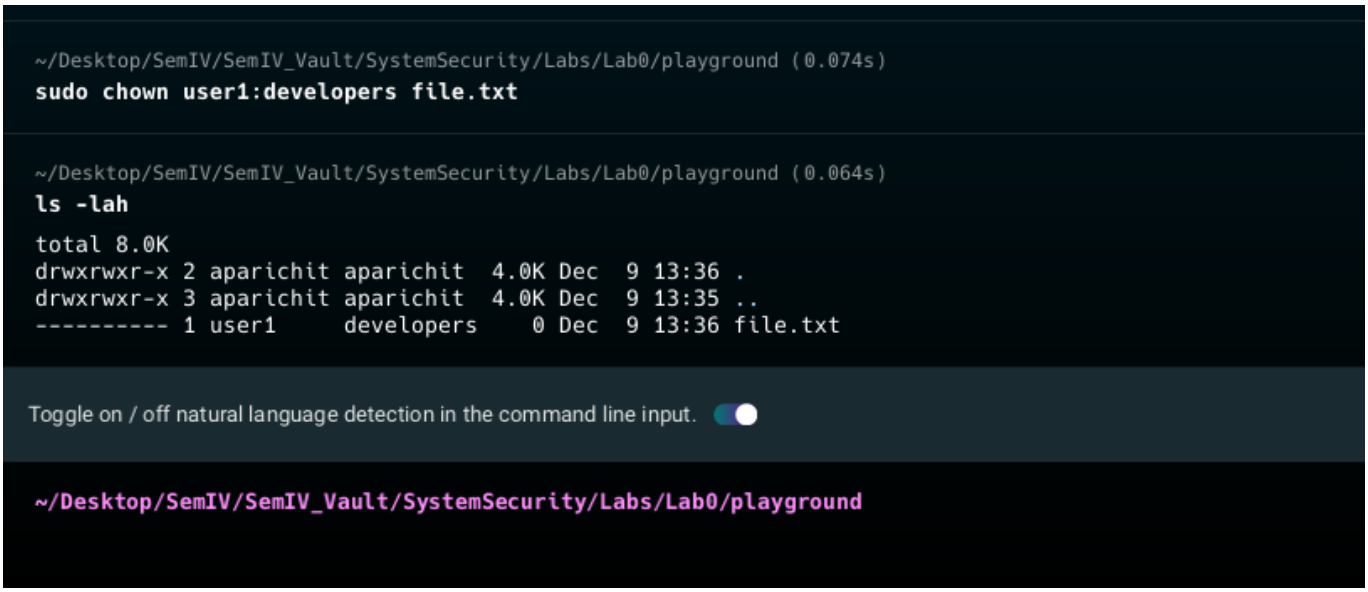
Changing the grop

groups aparichit

```
sudo chgrp developers flie.txt
```

changing both the user and group

```
sudo chown user:developers file.txt
```



SUID, SGID and Sticky Bits Summary

Summary of Symbols in File Permissions

Permission	Symbol in `ls -l`	Effect
SUID	`s` in owner exec (`rws`)	File executes with owner's privileges.
SGID	`s` in group exec (`r-s`)	File executes with group's privileges; directory ensures group consistency.
Sticky Bit	`t` in others exec (`rwt`)	Only owner can delete/modify their files in the directory.

Special permissions in Linux (SUID, SGID, sticky)

Understanding SUID, SGID, and the Sticky Bit in Linux

In Linux, **file permissions** determine who can read, write, or execute a file or directory. However, there are three **special permission bits** that modify how execution or access is handled:

1. **SUID (Set User ID)**
2. **SGID (Set Group ID)**
3. **Sticky Bit**

These special bits are mainly used for **security and access control**. Let's dive into each in detail.

---

1. SUID (Set User ID)

SUID (Set User ID) is a special permission that allows a user to execute a file with the permissions of the file's owner.
This is useful when a normal user needs to execute a program that requires **elevated privileges**, such as accessing hardware or modifying system files.

How It Works

- Normally, when you execute a file, it runs with **your** privileges.
- If the **SUID bit** is set, the file runs with the **owner's** privileges instead of the user's.
- The owner is usually **root** for system binaries.

Example Use Case

A classic example is the `/usr/bin/passwd` command, which allows users to change their passwords.

```
ls -l /usr/bin/passwd
```

Output:

```
-rwsr-xr-x 1 root root 54256 Feb  5  2024 /usr/bin/passwd
```

- The `s` in `rwsr-xr-x` means the SUID bit is set.
- The file is owned by **root**, but any user can execute it with **root privileges**.
- This allows non-root users to change their passwords, even though password files are system-protected.

Setting the SUID Bit

You can set the **SUID** bit using the `chmod` command.

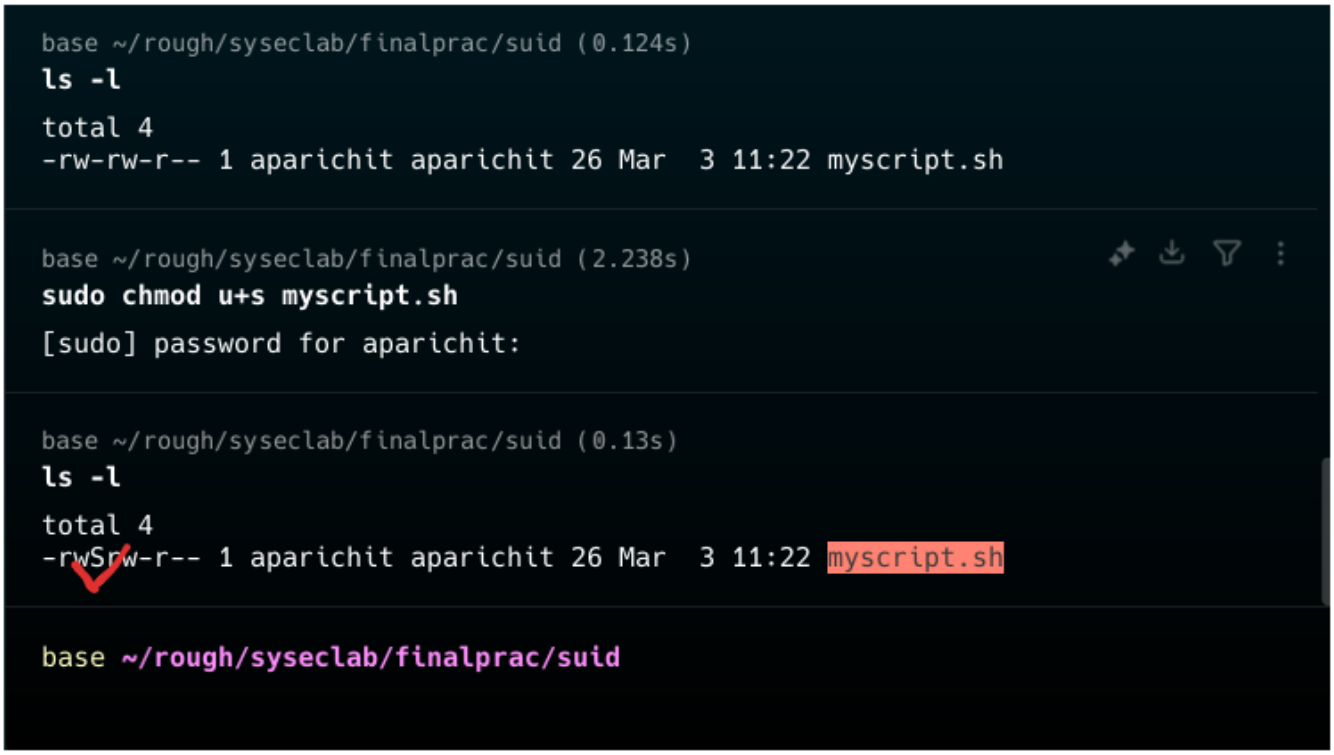
```
chmod u+s filename
```

Example:

```
chmod u+s myscript.sh
```

To remove it:

```
chmod u-s myscript.sh
```

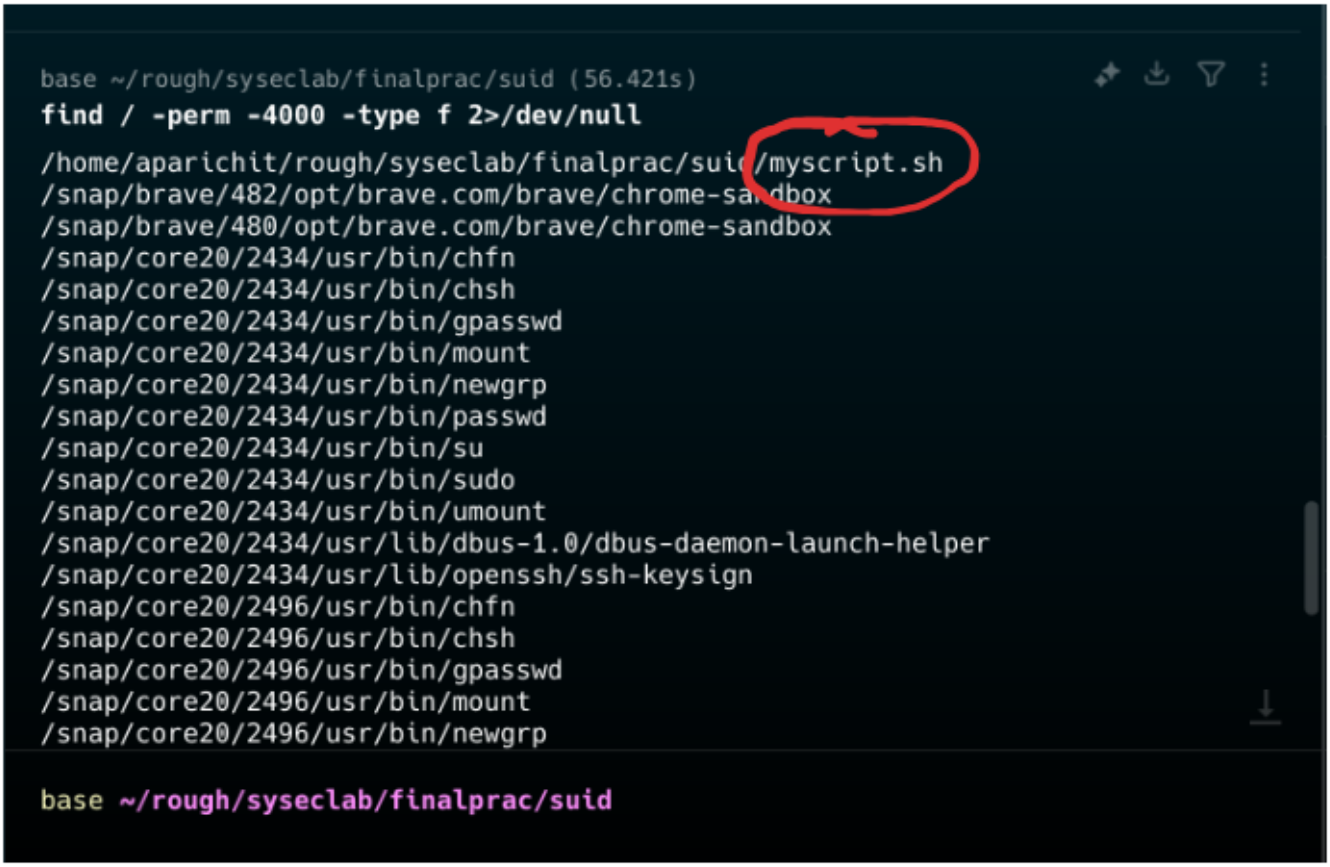


Security Concerns

- SUID can be a **security risk** if applied to insecure binaries.
- Attackers can exploit vulnerable SUID binaries to escalate privileges.

To find all SUID files on your system:

```
find / -perm -4000 -type f 2>/dev/null
```



---

2. SGID (Set Group ID)

SGID (Set Group ID) works similarly to SUID but for groups.

- When applied to **files**, it allows execution with the **group's** permissions.
- When applied to **directories**, it ensures that all newly created files **inherit the directory's group**.

SGID on Files

Example: `/usr/bin/locate`

```
ls -l /usr/bin/locate
```

Output:

```
-rwx--s--x 1 root locate 43656 Feb  5  2024 /usr/bin/locate
```

- The `s` in `--s--x` indicates SGID is set.
- The file is owned by the **root** user but belongs to the `locate` group.
- Any user executing this file will run it with the **locate group's** permissions.

Setting SGID on Files

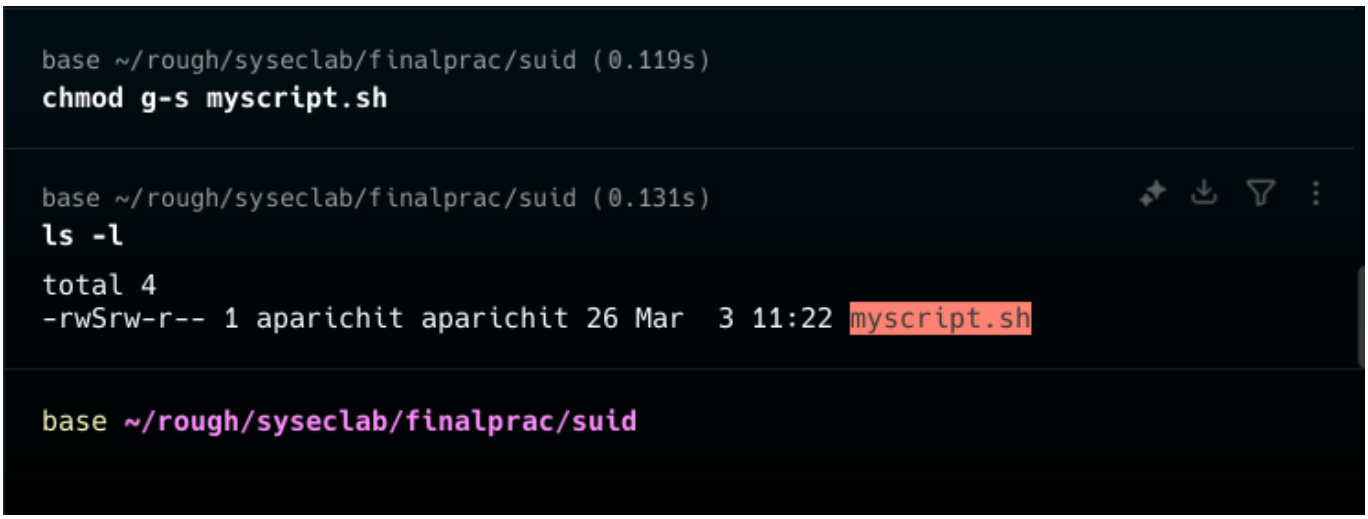
```
chmod g+s filename
```

Example:

```
chmod g+s myscript.sh
```

To remove it:

```
chmod g-s myscript.sh
```



SGID on Directories

When SGID is set on a directory, **all new files created inside will inherit the directory's group**, rather than the creator's default group.

Example Use Case

A shared group directory where multiple users collaborate.

```
mkdir /shared
chmod 2775 /shared
chown :developers /shared
```

- The **2** in `2775` sets the **SGID bit**.
- Now, any file created in `/shared` will **inherit the group 'developers'**, ensuring consistent access.

Finding SGID Files

```
find / -perm -2000 -type f 2>/dev/null
```

---

3. Sticky Bit

The **Sticky Bit** is used **only on directories**.
It ensures that **only the owner of a file (or root) can delete or modify it**, even if the directory is world-writable.

How It Works

- Normally, in a world-writable directory (`/tmp`), **any user** can delete any file.
- If the **Sticky Bit** is set, only the file **owner** (or root) can delete it.

Example: `/tmp` Directory

```
ls -ld /tmp
```

Output:

```
drwxrwxrwt 10 root root 4096 Feb  5  2024 /tmp
```

- The `t` in `rwxrwxrwt` indicates the **Sticky Bit is set**.
- This prevents users from deleting files that **they don't own**.

Setting the Sticky Bit

```
chmod +t directory_name
```

Example:

```
chmod +t /public
```

To remove it:

```
chmod -t /public
```

Finding Directories with Sticky Bit

```
find / -perm -1000 -type d 2>/dev/null
```

Summary Table

Bit	Symbol	Applied To	Effect
SUID	s (User)	Executable files	Runs with the **owner's** privileges instead of the user's.
SGID	s (Group)	Files & Directories	Files run with the **group's** permissions; directories inherit the group.
Sticky Bit	t	Directories	Prevents users from deleting files they don't own.

Practical Examples

Example 1: Creating an SUID Program

Let's create a simple **C program** that prints the `/etc/shadow` file (which stores encrypted passwords and is only readable by root).

Step 1: Create the C File

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    system("cat /etc/shadow");
    return 0;
}
```

Step 2: Compile and Set SUID

```
gcc suid_example.c -o suid_example
chmod u+s suid_example
sudo chown root suid_example
```

Now, even if a **normal user** runs `./suid_example`, it will execute as **root**.

Example 2: Setting SGID on a Shared Directory

If multiple users need access to a project folder:

```
sudo mkdir /project
sudo chown :developers /project
sudo chmod 2775 /project
```

- New files inside `/project` will inherit the **developers** group.

Example 3: Secure Temporary Directory with Sticky Bit

To create a shared directory where users **cannot delete each other's files**:

```
sudo mkdir /public
sudo chmod 1777 /public
```

Now, `/public` is writable by everyone, but users **can only delete their own files**.

Conclusion

- **SUID** allows **privilege escalation** for executables (e.g., `passwd`).
- **SGID** is useful for **group collaboration** (e.g., shared directories).
- **Sticky Bit** is essential for **public directories** (e.g., `/tmp`).

Each of these **special permission bits** improves **security and usability** but must be **used carefully** to avoid security risks.

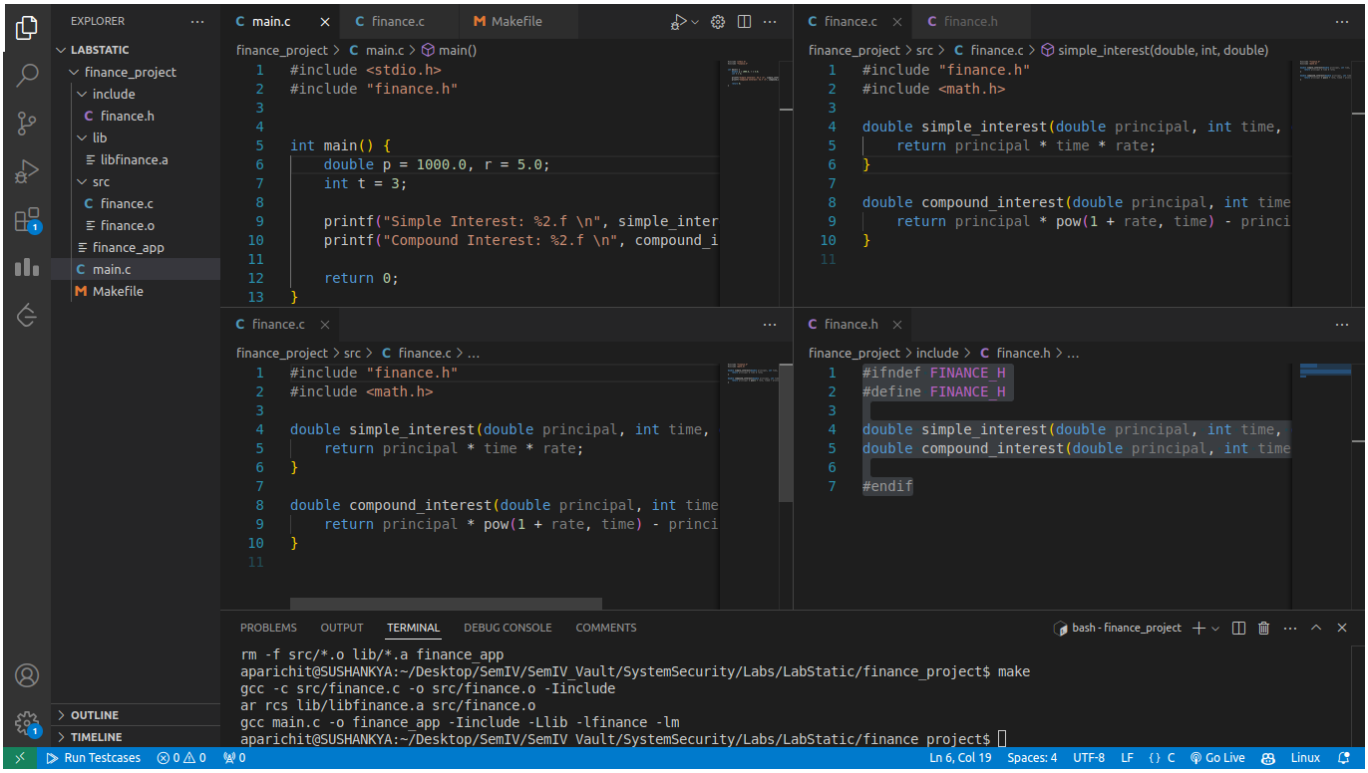
Creating Static Libraries in C - Lab 2

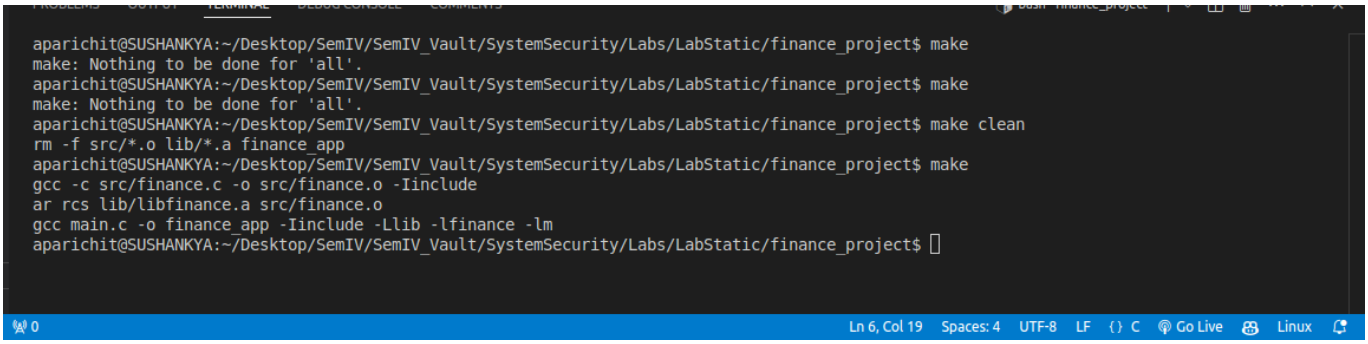
What is a Static Library?

A static library is a collection of object files (.o) bundled together in a single archive (.a) that can be linked into executables. Unlike dynamic libraries (.so or .dll), static libraries are copied into the final binary, making them self-contained.

```
finance_project/
├── include/
│   ├── finance.h
├── src/
│   ├── finance.c
├── lib/
├── main.c
├── Makefile
```

Codes





main.c

```c
#include <stdio.h>
#include "finance.h"


int main() {
    double p = 1000.0, r = 5.0;
    int t = 3;

    printf("Simple Interest: %2.f \n", simple_interest(p, t, r));
    printf("Compound Interest: %2.f \n", compound_interest(p, t, r));

    return 0;
}
```

finance.h

```c
#ifndef FINANCE_H
#define FINANCE_H

double simple_interest(double principal, int time, double rate);
double compound_interest(double principal, int time, double rate);

#endif
```

finance.c

```c
#include "finance.h"
#include <math.h>

double simple_interest(double principal, int time, double rate) {
    return principal * time * rate;
}

double compound_interest(double principal, int time, double rate) {
    return principal * pow(1 + rate, time) - principal;
```

```
}
```

```
# Compiler
CC = gcc

# Directories
SRC_DIR = src
INCLUDE_DIR = include
LIB_DIR = lib

# Source and Object Files
SRC_FILES = $(SRC_DIR)/finance.c
OBJ_FILES = $(SRC_DIR)/finance.o

# Static Library Name
LIB_NAME = libfinance.a

# Executable Name
TARGET = finance_app

# Compiler and Linker Flags
CFLAGS = -I$(INCLUDE_DIR)
LDFLAGS = -L$(LIB_DIR) -lfinance -lm

# Default target
all: $(TARGET)

# Build the Static Library
$(LIB_DIR)/$(LIB_NAME): $(OBJ_FILES)
    ar rcs $@ $^

# Compile Object Files
$(SRC_DIR)/finance.o: $(SRC_DIR)/finance.c $(INCLUDE_DIR)/finance.h
    $(CC) -c $< -o $@ $(CFLAGS)

# Build the Executable
$(TARGET): main.c $(LIB_DIR)/$(LIB_NAME)
    $(CC) main.c -o $(TARGET) $(CFLAGS) $(LDFLAGS)

# Force re-linking every time
# .PHONY: all clean

# Clean up generated files
clean:
    rm -f $(SRC_DIR)/*.o $(LIB_DIR)/*.a $(TARGET)
```

Compiling the library

```
gcc -c src/finance.c -o src/finance.o -Iinclude
```

Creating .a archive from object files

```
ar rcs lib/libfinance.a src/finance.o # use this
ar rcs lib/finance.a src/finance.o # this had created problem, i.e. giving
non-standard name to the library file
```

Checking what object files does the archive contains

```
ar -t lib/libfinance.a
```

Final compilation command

```
gcc main.c -Llib -l:finance.a -lm -Iinclude -o finance_app # if the library
name is non standard
gcc main.c -Llib -lfinance -lm -Iinclude -o finance_app # if follows
standard naming of static file like. libfinance, libcalculator, etc.
```

> Static libraries in Unix-like systems follow the lib.a convention because the linker (ld) automatically searches for files prefixed with lib when using -l, and omitting this prefix requires explicitly specifying the filename with -l:.a.

Creating Dynamic Libraries - Lab 3

Creating and Using a Dynamic Library in C

The process of creating and utilizing a **dynamic library (shared library)** in C involves several steps. The development takes place in a **Linux** environment, where the shared object file (`.so`) is compiled and linked with a program.

1. Development of the Library

A dynamic library typically consists of reusable functions that can be shared across multiple programs. In this case, a **mathematical library** was developed.

1.1. Implementation of the Library (`mymath.c`)

A source file named `mymath.c` was created to define mathematical functions. The functions included in the library perform basic arithmetic operations such as addition, subtraction, multiplication, and division. The implementation is as follows:

```c
#include <stdio.h>

int add(int a, int b) {
    return a + b;
}

int subtract(int a, int b) {
    return a - b;
}

int multiply(int a, int b) {
    return a * b;
}

double divide(int a, int b) {
    if (b == 0) {
        printf("Error: Division by zero!\n");
        return 0.0;
    }
    return (double)a / b;
}
```

1.2. Declaration of Function Prototypes (`mymath.h`)

A header file (`mymath.h`) was created to declare the function prototypes. This header file allows other programs to access the functions contained within the library.

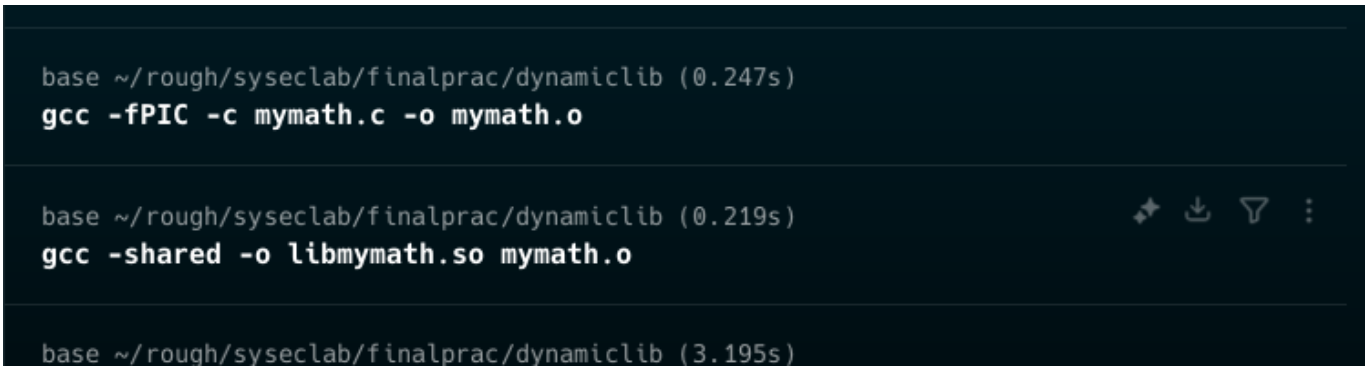
```
#ifndef MYMATH_H
#define MYMATH_H

int add(int a, int b);
int subtract(int a, int b);
int multiply(int a, int b);
double divide(int a, int b);

#endif
```

---

2. Compilation of the Shared Library

To generate the shared object file, the source code was compiled using **GCC**. The following commands were used:

```
gcc -fPIC -c mymath.c -o mymath.o
gcc -shared -o libmymath.so mymath.o
```



- The `-fPIC` flag was used to generate **position-independent code**, which is necessary for shared libraries.
- The `-c` flag compiled the source file into an object file without linking.
- The `-shared` flag was used to create a **shared object file (`.so`)**.

As a result of this process, a file named `libmymath.so` was successfully generated.

---

3. Utilization of the Shared Library

To verify the functionality of the newly created library, a **C program (`main.c`)** was written.

3.1. Implementation of the Main Program (`main.c`)

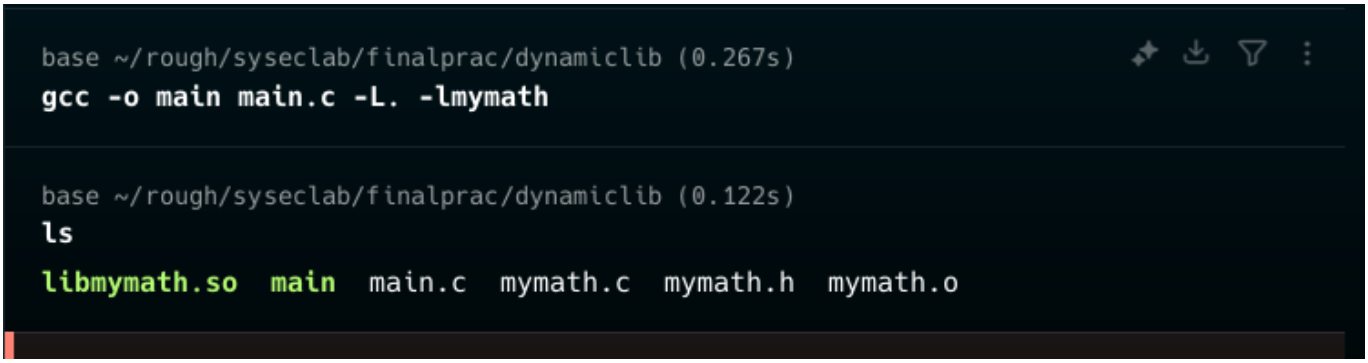
A program was created to utilize the functions from `libmymath.so`. The corresponding implementation is shown below:

```
#include <stdio.h>
#include "mymath.h"

int main() {
    int a = 10, b = 5;

    printf("Addition: %d\n", add(a, b));
    printf("Subtraction: %d\n", subtract(a, b));
    printf("Multiplication: %d\n", multiply(a, b));
    printf("Division: %.2f\n", divide(a, b));

    return 0;
}
```

3.2. Compilation of the Main Program

The main program was compiled with the following command:

```
gcc -o main main.c -L. -lmymath
```



- The `-L.` option specified the current directory as the location of the library.
- The `-lmymath` option linked the program against `libmymath.so`.

---

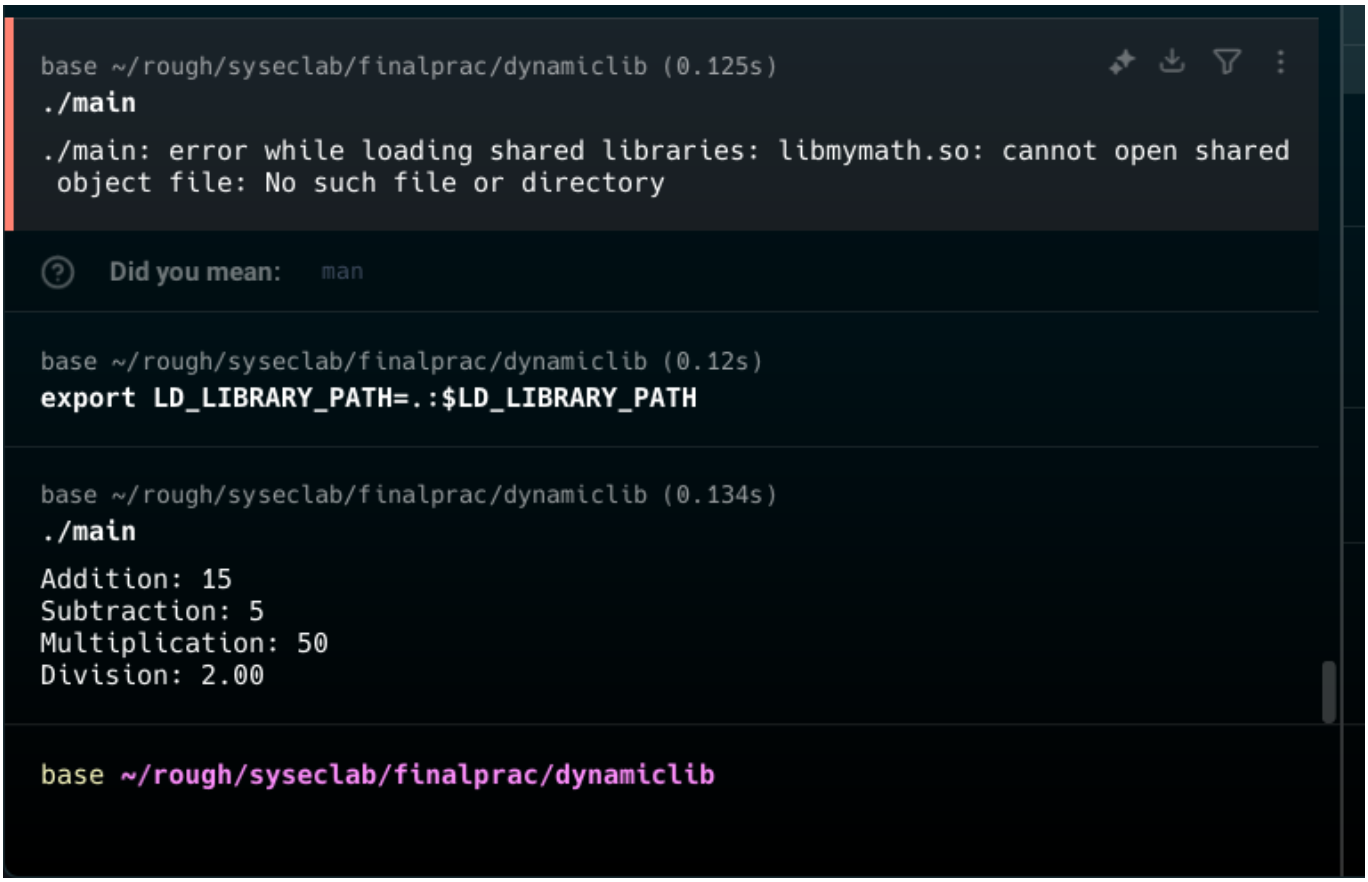
4. Execution and Library Path Configuration

During execution, an error was encountered due to the **shared library not being found** by the system. The following message was displayed:

```
error while loading shared libraries: libmymath.so: cannot open shared
object file: No such file or directory
```

To resolve this issue, the **LD_LIBRARY_PATH** environment variable was modified temporarily:

```
export LD_LIBRARY_PATH=.:$LD_LIBRARY_PATH
./main
```



Alternatively, to make the library accessible system-wide, it was moved to `/usr/local/lib` and registered with the system:

```
sudo cp libmymath.so /usr/local/lib
sudo ldconfig
```

Following this step, the program was executed without requiring additional environment variables.

---

5. Output Verification

After successful execution, the following output was produced:

```
Addition: 15
Subtraction: 5
Multiplication: 50
Division: 2.00
```

The expected results were observed, confirming the successful creation and integration of the **dynamic library** in C.

Conclusion

A **shared library** (`.so` file) was successfully developed, compiled, and integrated into a program. The necessary steps included defining functions, generating the library, linking it with a program, and ensuring proper execution by configuring the shared library path. The approach demonstrated the modular nature of dynamic libraries, allowing multiple programs to use the same functions without requiring static linking.

To make use of make file for compilation - Lab 4

Theory

When working on a C project with multiple source files, manually compiling each file can be inefficient. The `make` tool helps automate this process by only recompiling files that have changed, using timestamps.

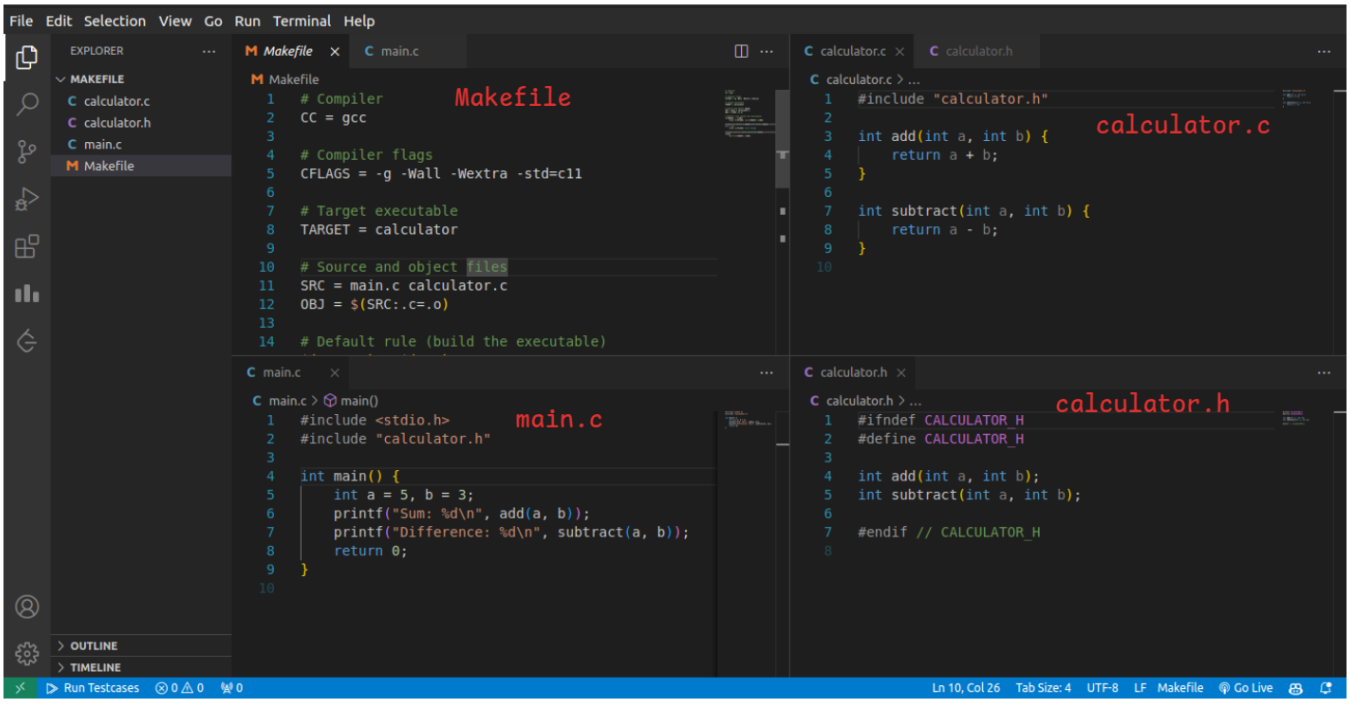
How Does `make` Track Changes?

- `make` checks **timestamps** of files.
- If a `.c` file has been **modified after** its corresponding `.o` file was last compiled, `make` recompiles only that file.
- This helps in **incremental compilation**, making the build process faster.

Project Directory Structure

```
project/
|── Makefile
```

```
|── main.c
|── calculator.c
|── calculator.h
```



Makefile

```
# Compiler
CC = gcc

# Compiler flags
CFLAGS = -g -Wall -Wextra -std=c11

# Target executable
TARGET = calculator

# Source and object files
SRC = main.c calculator.c
OBJ = $(SRC:.c=.o)

# Default rule (build the executable)
$(TARGET) : $(OBJ)
    $(CC) $(CFLAGS) -o $(TARGET) $(OBJ)

# Rule to compile C files into object files
%.o : %.c
    $(CC) $(CFLAGS) -c $< -o $@

# Clean rule to remove generated files
clean:
    rm -f $(TARGET) $(OBJ)
```

Source Code

main.c

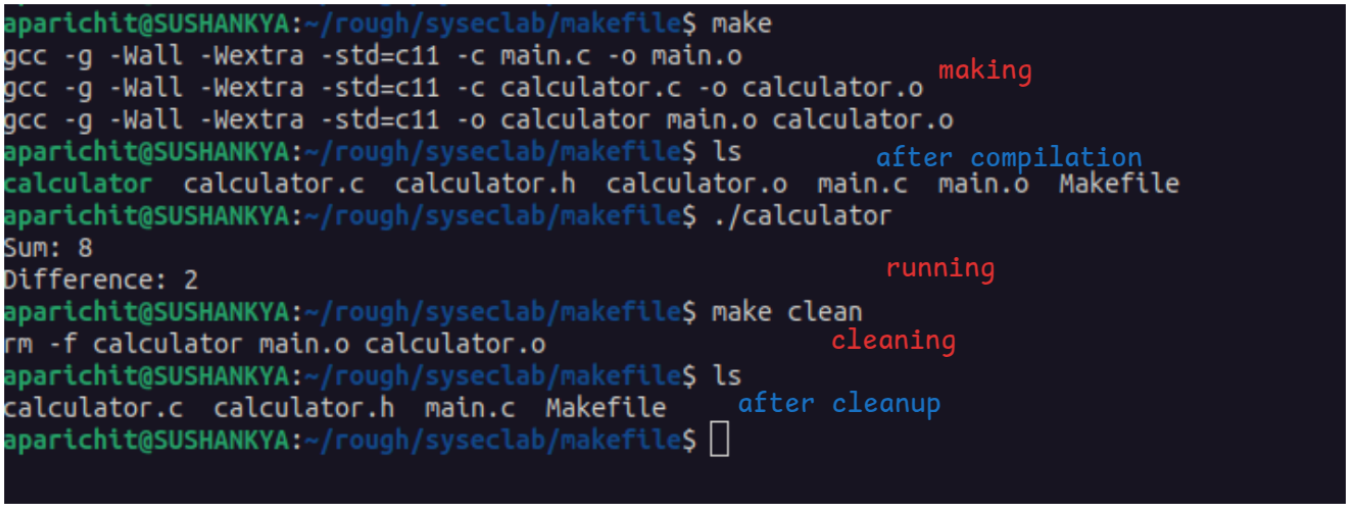
```c
#include <stdio.h>
#include "calculator.h"

int main() {
    int a = 5, b = 3;
    printf("Sum: %d\n", add(a, b));
    printf("Difference: %d\n", subtract(a, b));
    return 0;
}
```

calculator.c

```c
#include "calculator.h"

int add(int a, int b) {
    return a + b;
}

int subtract(int a, int b) {
    return a - b;
}
```

calculator.h

```c
#ifndef CALCULATOR_H
#define CALCULATOR_H

int add(int a, int b);
int subtract(int a, int b);

#endif // CALCULATOR_H
```



Building the Project

1. Open a terminal and navigate to the project directory:

   ```
   cd project
   ```

2. Run `make`:

   ```
   make
   ```

   - This will compile `main.c` and `calculator.c`, producing the `calculator` executable.

3. Execute the program:

   ```
   ./calculator
   ```

   - Expected output:

     ```
     Sum: 8
     Difference: 2
     ```

4. To clean up the compiled files:

   ```
   make clean
   ```

To create isolated environment in Linux using chroot - Lab 5

Create the directory setup

```
mkdir -p newroot/{bin,lib,lib64}
```

Copy the required binaries

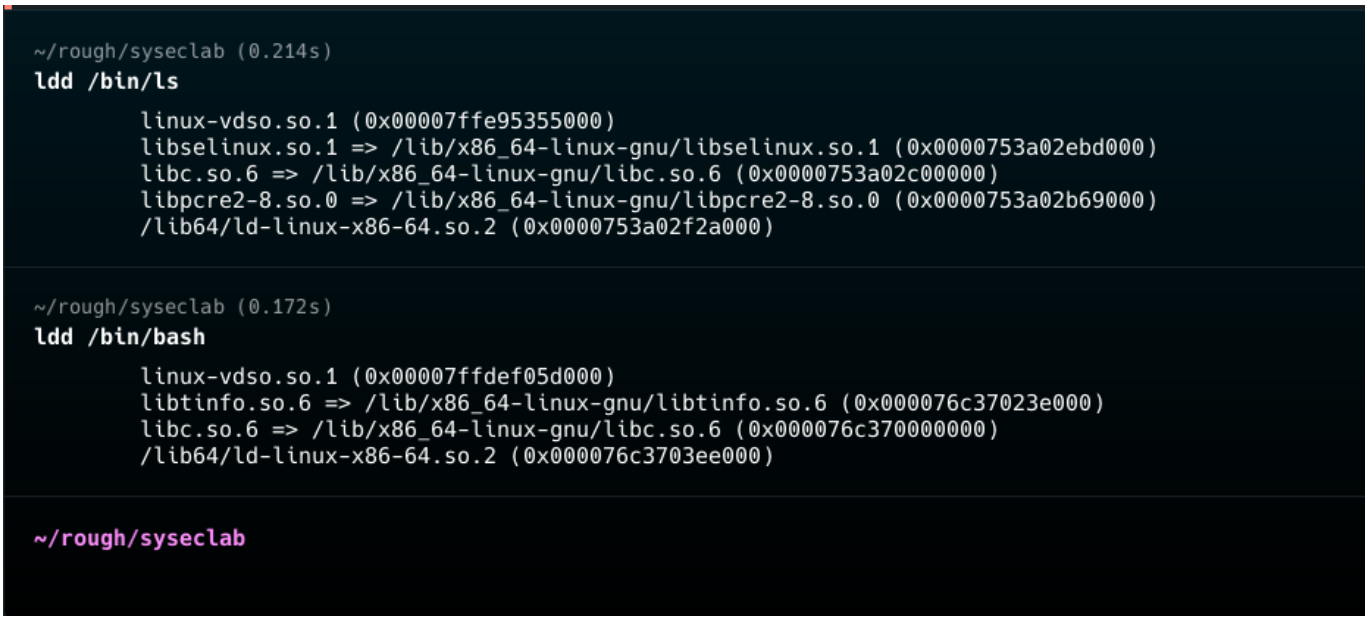
```
cp -v /bin/{bash,ls,cp} /newroot/bin
```

Find the the dynamic dependencies for each of above binaries

```
ldd /bin/bash
ldd /bin/ls
ldd /bin/pwd
```

Output

```
ldd /bin/bash
    linux-vdso.so.1 (0x00007ffd8a7d8000) # no need to add this
    libtinfo.so.6 => /lib/x86_64-linux-gnu/libtinfo.so.6
(0x00007ec9dcddc000)
    libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007ec9dca00000)
    /lib64/ld-linux-x86-64.so.2 (0x00007ec9dcf8c000)

ldd /bin/ls
    linux-vdso.so.1 (0x00007ffe95355000)
    libselinux.so.1 => /lib/x86_64-linux-gnu/libselinux.so.1
(0x0000753a02ebd000)
    libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x0000753a02c00000)
    libpcre2-8.so.0 => /lib/x86_64-linux-gnu/libpcre2-8.so.0
(0x0000753a02b69000)
    /lib64/ld-linux-x86-64.so.2 (0x0000753a02f2a000)
```



Copy all the respective files with proper directory structures

```
cp --parents  /lib/x86_64-linux-gnu/libselinux.so.1 newroot/
cp --parents /lib/x86_64-linux-gnu/libc.so.6 newroot/
# copy all the dependencies like this
```

Finally run the chroot command

```
sudo chroot newroot/ /bin/bash
```

Output Gives the bash shell spawned

```
~/rough/syseclab (0.201s)
sudo chroot newroot /bin/ls
bin  lib  lib64

~/rough/syseclab
sudo chroot newroot /bin/bash
bash-5.1# ls
bin  lib  lib64
bash-5.1#
```

Apparmor - Lab 6

Apparmor

Creating bash script

Choosing I

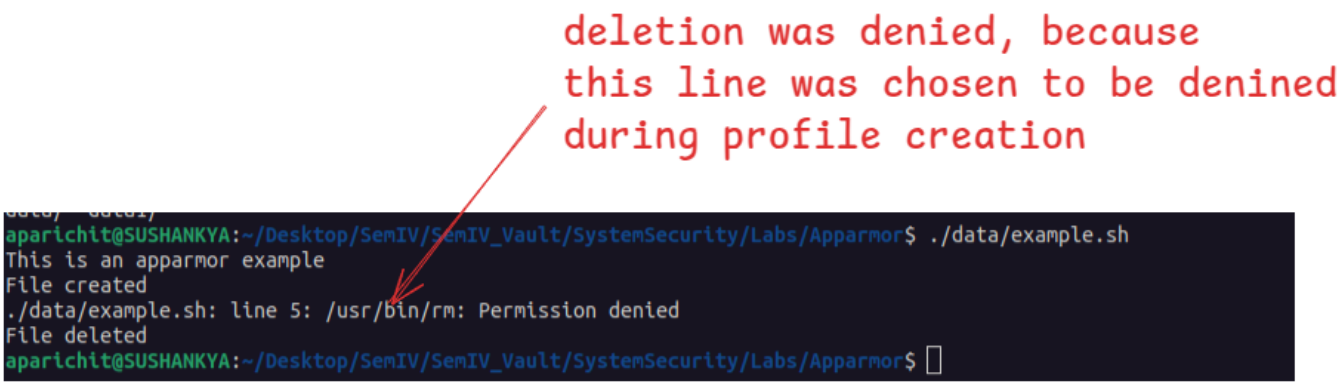
```
Reading log entries from /var/log/syslog.

Profiling: /home/aparichit/Desktop/SemIV/SemIV_Vault/SystemSecurity/Labs/Apparmor/data/example.sh

Please start the application to be profiled in
another window and exercise its functionality now.

Once completed, select the "Scan" option below in
order to scan the system logs for AppArmor events.

For each AppArmor event, you will be given the
opportunity to choose whether the access should be
allowed or denied.

[(S)can system log for AppArmor events] / (F)inish
Reading log entries from /var/log/syslog.

Profile:  /home/aparichit/Desktop/SemIV/SemIV_Vault/SystemSecurity/Labs/Apparmor/data/example.sh
Execute:  /usr/bin/touch
Severity: 3

(I)nherit / (C)hild / (N)amed / (X) ix On / (D)eny / Abo(r)t / (F)inish
```

choosing I

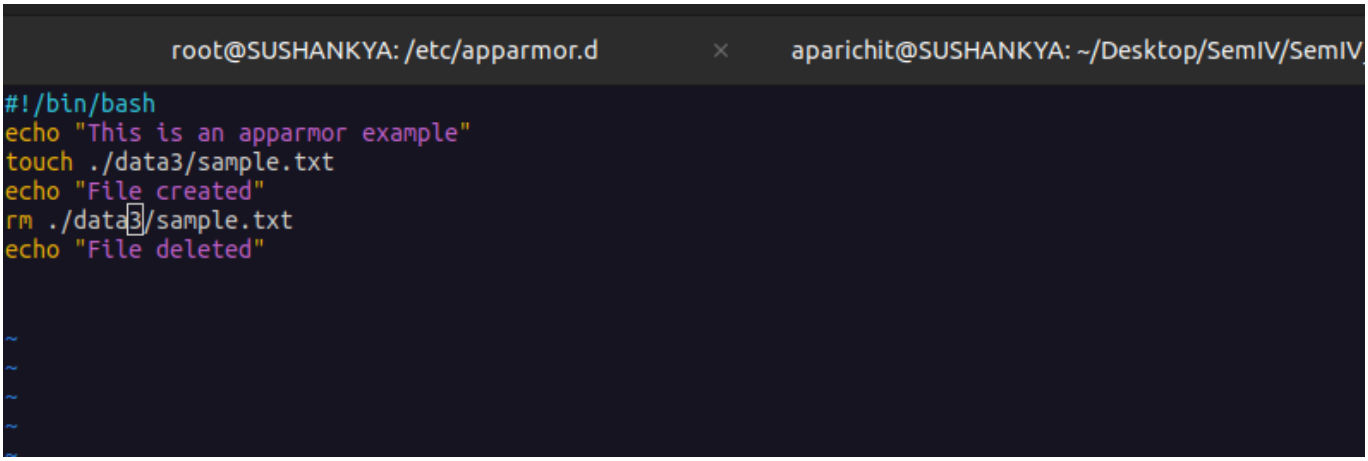
Creating and saving profiles

```
 [1 - owner /home/*/Desktop/SemIV/SemIV_Vault/SystemSecurity/Labs/Apparmor/data1/sample.txt w,]
  2 - owner /home/aparichit/Desktop/SemIV/SemIV_Vault/SystemSecurity/Labs/Apparmor/data1/sample.txt w,
(A)llow / [(D)eny] / (I)gnore / (G)lob / Glob with (E)xtension / (N)ew / Audi(t) / (O)wner permissions off / Abo(r)t / (F)inish
Adding owner /home/*/Desktop/SemIV/SemIV_Vault/SystemSecurity/Labs/Apparmor/data1/sample.txt w, to profile.

= Changed Local Profiles =

The following local profiles were changed. Would you like to save them?

 [1 - /home/aparichit/Desktop/SemIV/SemIV_Vault/SystemSecurity/Labs/Apparmor/data/example.sh]
(S)ave Changes / Save Selec(t)ed Profile / [(V)iew Changes] / View Changes b/w (C)lean profiles / Abo(r)t
Writing updated profile for /home/aparichit/Desktop/SemIV/SemIV_Vault/SystemSecurity/Labs/Apparmor/data/example.sh.

Profiling: /home/aparichit/Desktop/SemIV/SemIV_Vault/SystemSecurity/Labs/Apparmor/data/example.sh

Please start the application to be profiled in
another window and exercise its functionality now.

Once completed, select the "Scan" option below in
order to scan the system logs for AppArmor events.

For each AppArmor event, you will be given the
opportunity to choose whether the access should be
allowed or denied.

[(S)can system log for AppArmor events] / (F)inish
Setting /home/aparichit/Desktop/SemIV/SemIV_Vault/SystemSecurity/Labs/Apparmor/data/example.sh to enforce mode.

Reloaded AppArmor profiles in enforce mode.

Please consider contributing your new profile!
See the following wiki page for more information:
https://gitlab.com/apparmor/apparmor/wikis/Profiles

Finished generating profile for /home/aparichit/Desktop/SemIV/SemIV_Vault/SystemSecurity/Labs/Apparmor/data/example.sh.
aparichit@SUSHANKYA:~/Desktop/SemIV/SemIV_Vault/SystemSecurity/Labs/Apparmor$
```

Checking for restriction

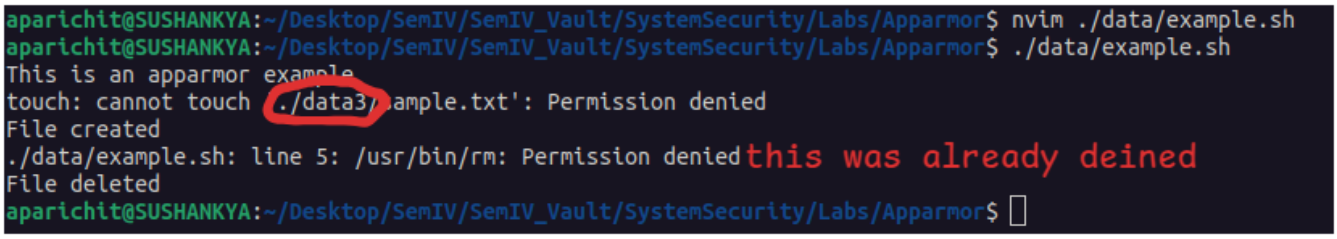


Changing the path of the script



ntries from /var/log/syslog.

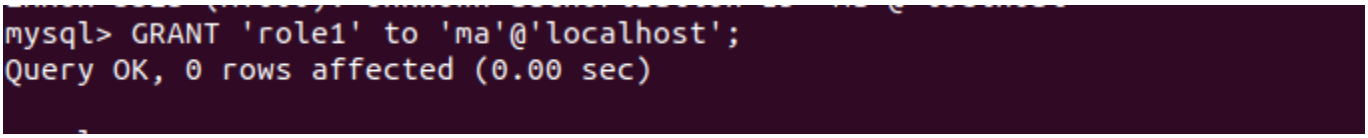
It was blocked

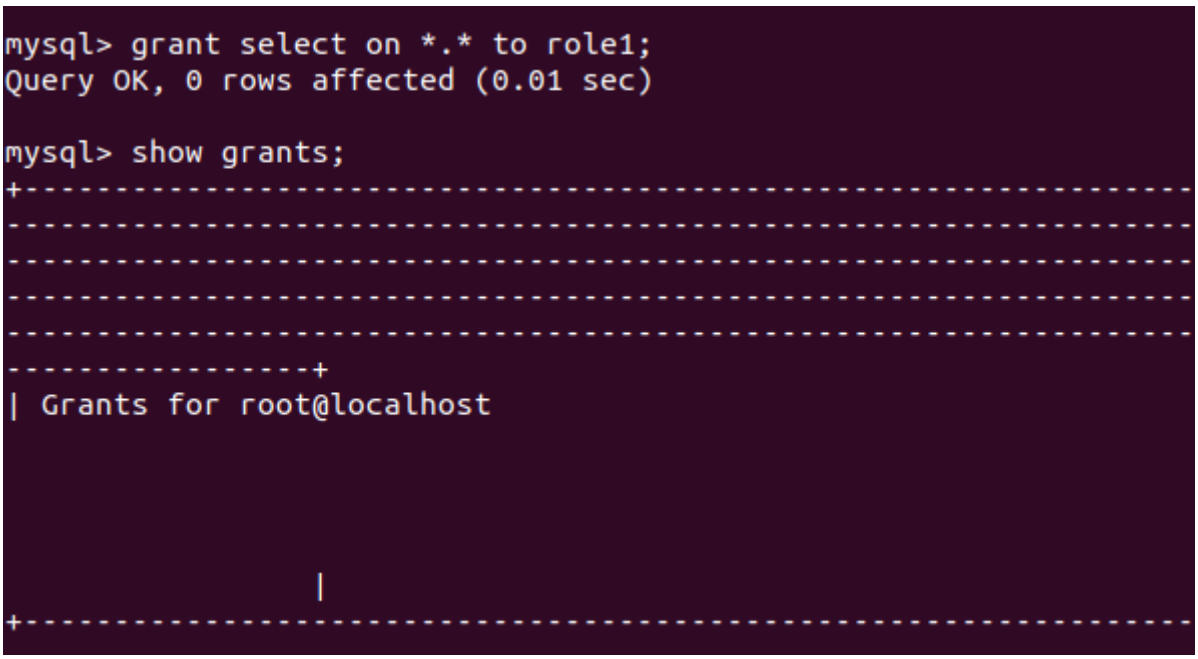


Conclusion:

Apparmor can be used to impose mandator access control to an applicaton or program.

Discretionay Access Control in Database : MySQL - Lab 7





Login as root user

```
sudo mysql
```

Showing the current user

```
SELECT CURRENT_USER();
```

Creating user

```
CREATE USER 'ma'@'localhost' identified by 'ma@localhost';
```

```
mysql> CREATE USER 'ma'@'localhost' identified by 'ma@localhost';
Query OK, 0 rows affected (0.02 sec)
```

Showing the grants for the current user

```
CREATE USER 'ma'@'localhost' identified by 'ma@localhost';
```

```
mysql> show grants for 'ma'@'localhost';
+----------------------------------------+
| Grants for ma@localhost                |
+----------------------------------------+
| GRANT USAGE ON *.* TO `ma`@`localhost` |
+----------------------------------------+
1 row in set (0.00 sec)
```

Creating database by newly created user

```
create database ma_ko_database
```

```
mysql> create database ma_ko_database;
ERROR 1044 (42000): Access denied for user 'ma'@'localhost' to database 'ma_ko_database'
mysql>
```

Granting Privileges from root user

```
1 row in set (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON *.* TO 'ma'@'localhost';
Query OK, 0 rows affected (0.01 sec)

mysql> show grants for 'ma'@'localhost';
+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| Grants for ma@localhost                                                                                                                                                    |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, SHUTDOWN, PROCESS, FILE, REFERENCES, INDEX, ALTER, SHOW DATABASES, SUPER, CREATE TEMPORA
RY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRI
GGER, CREATE TABLESPACE, CREATE ROLE, DROP ROLE ON *.* TO `ma`@`localhost`                                                                                |
| GRANT APPLICATION_PASSWORD_ADMIN,AUDIT_ABORT_EXEMPT,AUDIT_ADMIN,AUTHENTICATION_POLICY_ADMIN,BACKUP_ADMIN,BINLOG_ADMIN,BINLOG_ENCRYPTION_ADMIN,CLONE_
ADMIN,CONNECTION_ADMIN,ENCRYPTION_KEY_ADMIN,FIREWALL_EXEMPT,FLUSH_OPTIMIZER_COSTS,FLUSH_STATUS,FLUSH_TABLES,FLUSH_USER_RESOURCES,GROUP_REPLICATION_ADM
IN,GROUP_REPLICATION_STREAM,INNODB_REDO_LOG_ARCHIVE,INNODB_REDO_LOG_ENABLE,PASSWORDLESS_USER_ADMIN,PERSIST_RO_VARIABLES_ADMIN,REPLICATION_APPLIER,REPL
ICATION_SLAVE_ADMIN,RESOURCE_GROUP_ADMIN,RESOURCE_GROUP_USER,ROLE_ADMIN,SENSITIVE_VARIABLES_OBSERVER,SERVICE_CONNECTION_ADMIN,SESSION_VARIABLES_ADMIN,
SET_USER_ID,SHOW_ROUTINE,SYSTEM_USER,SYSTEM_VARIABLES_ADMIN,TABLE_ENCRYPTION_ADMIN,TELEMETRY_LOG_ADMIN,XA_RECOVER_ADMIN ON *.* TO `ma`@`localhost` |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
```

Creating database after the permission has been granted

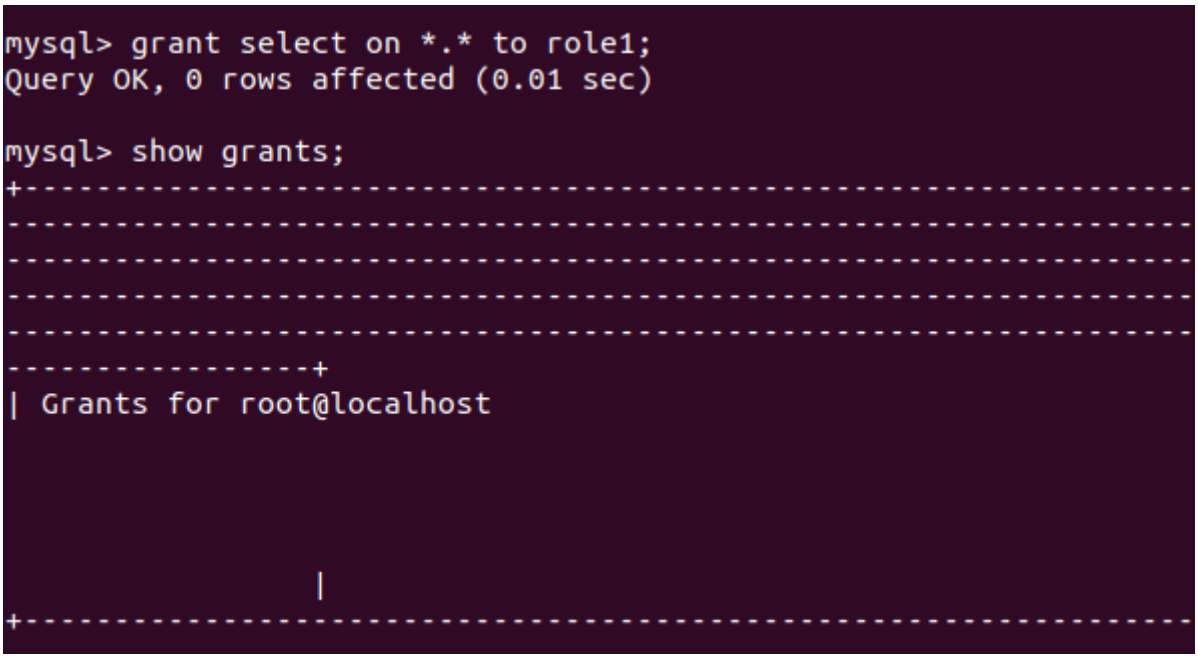
```
mysql> create database ma_ko_database;
Query OK, 1 row affected (0.01 sec)

mysql>
```

Creating role

```
CREATE ROLE 'role1';
```

Granting privileges to a rle

```
GRANT SELECT ON *.* TO role1;
```

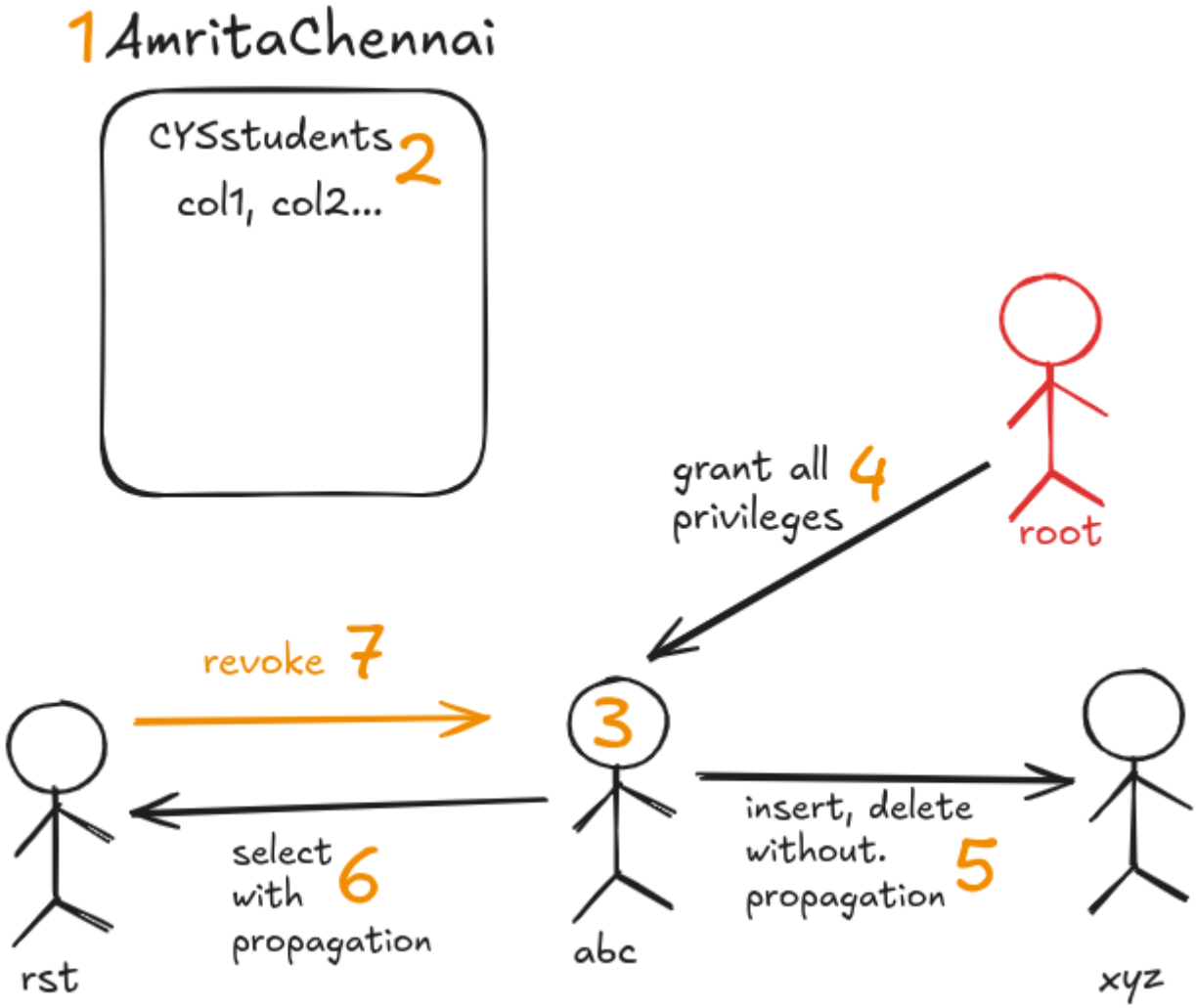


Assigning role to a particular user

```
GRANT 'role1' to 'ma'@'localhost';
```

Exercise

Exercise: Create a database “AmritaChennai”. Create a table “CYSStudents” with any number of fields of your choice. Create a user account “abc” with a password. Grant all privileges to the account “abc”. “abc” user account wants to grant to account “xyz” the privilege to insert and delete tuples in the table “CYSStudents”. However, “abc” does not want “xyz” to be able to propagate these privileges to additional accounts. Write the command for this and demonstrate. Suppose that “abc” wants to allow account “rst’ to retrieve information from the table “CYSStudents” and also to be able to propagate the SELECT privilege to other accounts. Write the command for this and demonstrate. Suppose that “abc” decides to revoke the SELECT privilege on the “CYSStudents” table from “rst”. Write the command for this and demonstrate.



Step 1: Create a Database

```
CREATE DATABASE AmritaChennai;
```

Step 2: Create a Table

```
USE AmritaChennai;

CREATE TABLE CYSStudents (
    StudentID INT PRIMARY KEY,
    StudentName VARCHAR(50),
    Course VARCHAR(30),
    Year INT
);
```

Step 3: Create a User Account abc with a Password

```
CREATE USER 'abc'@'localhost' IDENTIFIED BY 'abc_password';
```

Step 4: Grant All Privileges to User `abc`

```
GRANT ALL PRIVILEGES ON AmritaChennai.* TO 'abc'@'localhost';
```

Step 5: Grant Insert and Delete Privileges from `abc` to `xyz` Without Grant Option

User `abc` logs in and executes:

```
CREATE USER 'xyz'@'localhost' IDENTIFIED BY 'xyz_password';

GRANT INSERT, DELETE ON AmritaChennai.CYSStudents TO 'xyz'@'localhost';
```

- The absence of `WITH GRANT OPTION` ensures `xyz` cannot propagate these privileges.

Step 6: Grant SELECT Privilege from `abc` to `rst` With Grant Option

User `abc` logs in and executes:

```
CREATE USER 'rst'@'localhost' IDENTIFIED BY 'rst_password';

GRANT SELECT ON AmritaChennai.CYSStudents TO 'rst'@'localhost' WITH GRANT
OPTION;
```

- `WITH GRANT OPTION` allows `rst` to propagate the `SELECT` privilege to other accounts.

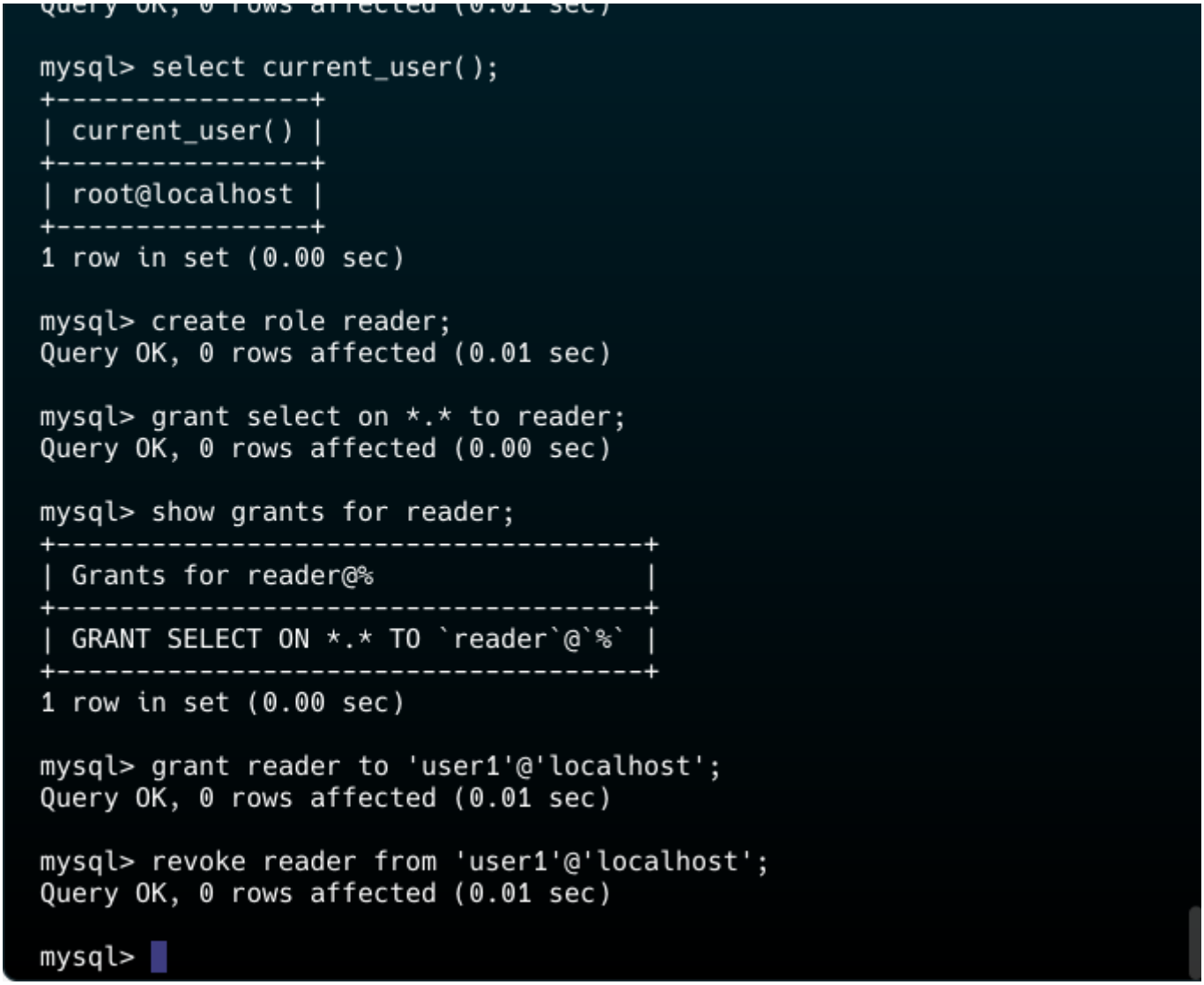
Step 7: Revoke SELECT Privilege from `rst`

User `abc` logs in and executes:

```
REVOKE GRANT OPTION FOR SELECT ON AmritaChennai.CYSStudents FROM
'rst'@'localhost';
REVOKE SELECT ON AmritaChennai.CYSStudents FROM 'rst'@'localhost';
```

- The first command removes the ability of `rst` to propagate the `SELECT` privilege.
- The second command completely revokes the `SELECT` privilege from `rst`.

Roles in mysql




```
create role reader;
```

```
grant select on *.* to reader;
```

```
show grants for reader;
```

```
grant reader to 'user1'@'localhost';
```

```
 revoke reader from 'user1'@'localhost';
```

Virtual Private Database : Implementing Row Level Security in mysql - Lab 8

Implementing Row-Level Security in MySQL

🔍 Goal

We want to store **all users' data in a single table** but **ensure that each user can only see, modify, or delete their own data**.
To achieve this, we will:

- Use **a single `books` table** with an `owner` column.
- Restrict access using **MySQL Views**.
- Enforce ownership using **triggers**.

🛠️ Step 1: Clean Up (If Needed)

If you attempted this before, **delete any previous data**:

```
DROP DATABASE IF EXISTS rls_practice;
DROP USER IF EXISTS 'user1'@'localhost';
DROP USER IF EXISTS 'user2'@'localhost';
FLUSH PRIVILEGES;
```

📂 Step 2: Create a New Database

```
CREATE DATABASE rls_practice;
USE rls_practice;
```

Output

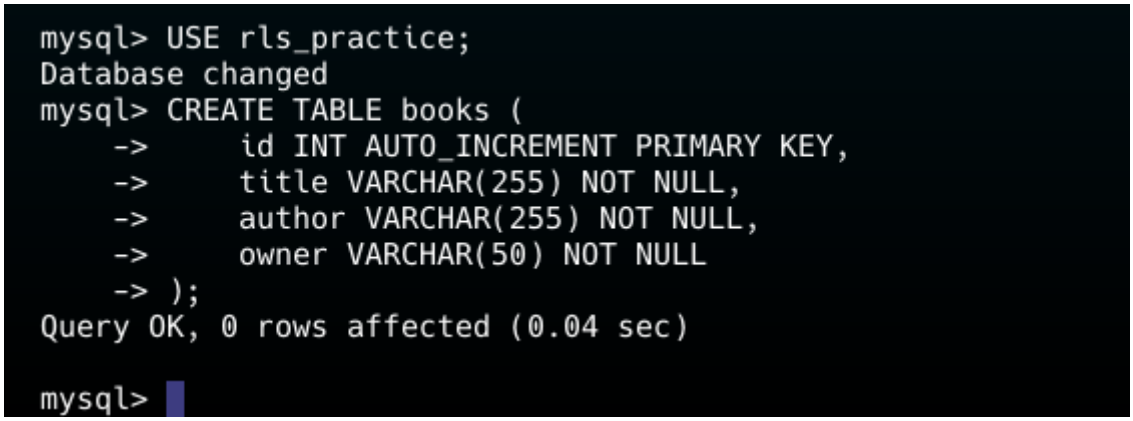
```
mysql> CREATE DATABASE rls_practice;
Query OK, 1 row affected (0.01 sec)

mysql> USE rls_practice;
Database changed
mysql>
```

Step 3: Create the `books` Table

This table stores all books **with an owner field** that records who inserted the book.

```
CREATE TABLE books (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    author VARCHAR(255) NOT NULL,
    owner VARCHAR(50) NOT NULL
);
```

Output

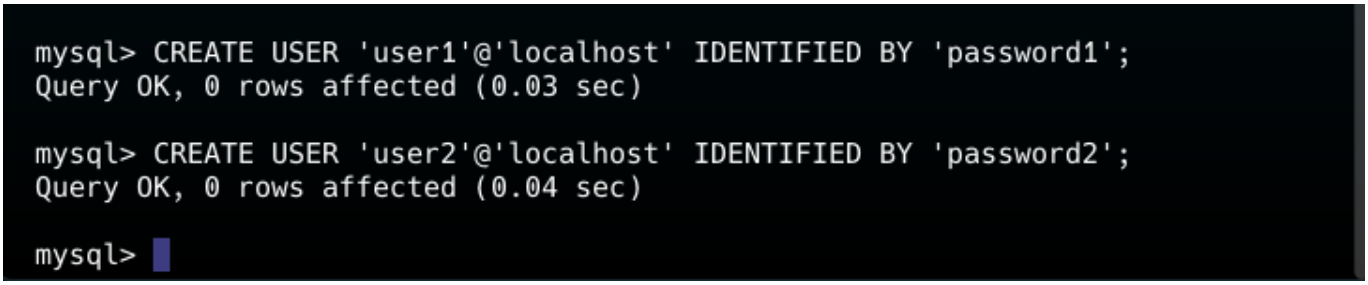


Step 4: Create Users

We will create **two users (`user1` and `user2`)**.

```
CREATE USER 'user1'@'localhost' IDENTIFIED BY 'password1';
CREATE USER 'user2'@'localhost' IDENTIFIED BY 'password2';
```

Output



Step 5: Enforce Ownership Using a Trigger

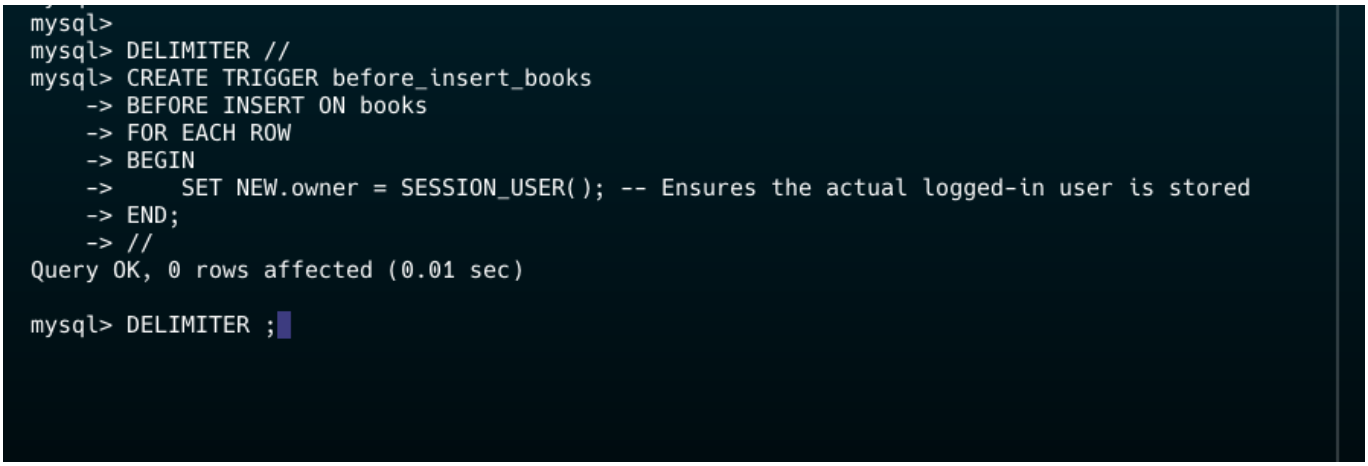
We create a **before-insert trigger** to **automatically store the username** when a book is inserted.

```
DELIMITER //
CREATE TRIGGER before_insert_books
BEFORE INSERT ON books
FOR EACH ROW
BEGIN
    SET NEW.owner = SESSION_USER(); -- Ensures the actual logged-in user is
stored
```

```
END;
//
DELIMITER ;
```

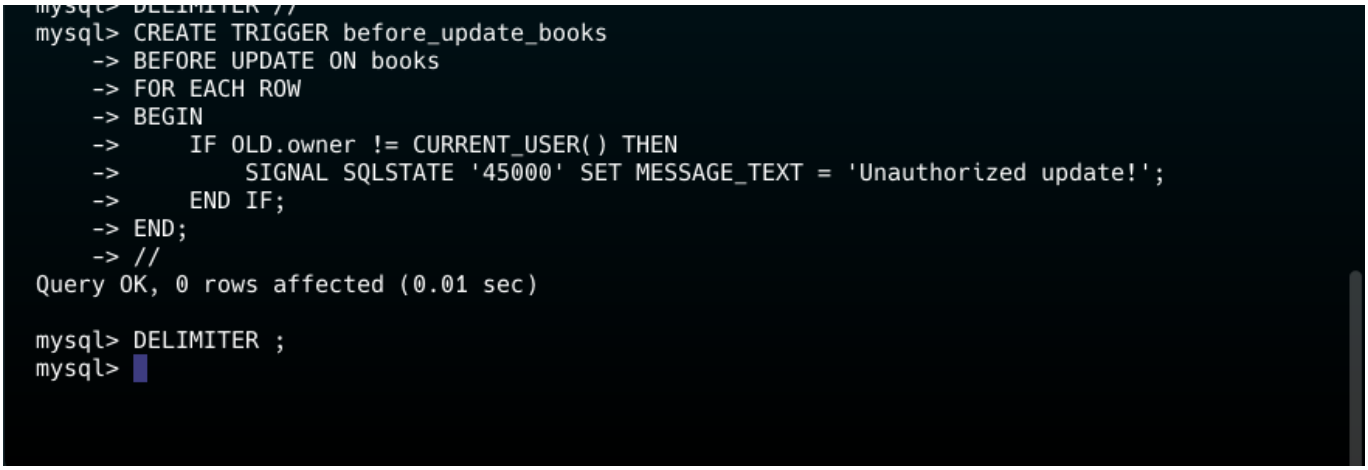
```
DELIMITER ;
```

Output




```
DELIMITER //
CREATE TRIGGER before_update_books
BEFORE UPDATE ON books
FOR EACH ROW
BEGIN
    IF OLD.owner != CURRENT_USER() THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Unauthorized update!';
    END IF;
END;
//
DELIMITER ;
```

Output

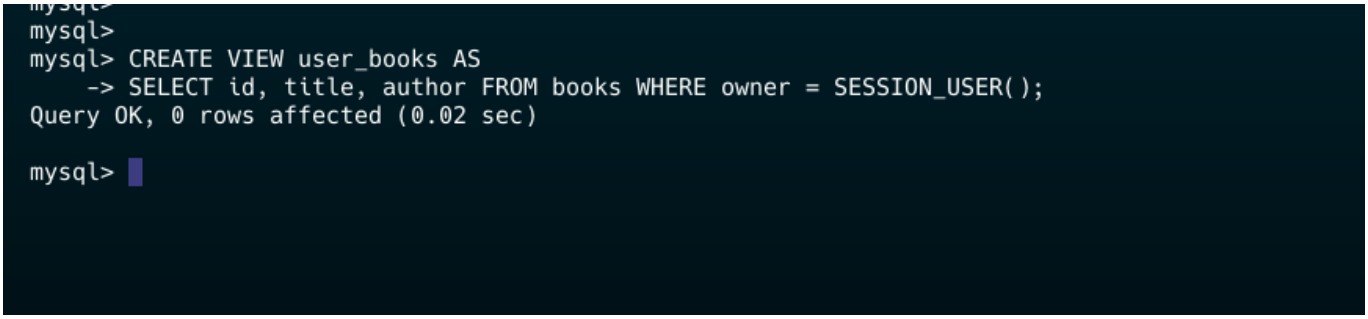


🔍 Step 6: Create a View for Row-Level Security

Since users should **only see their own books**, we create a **view** that filters by `owner`.

```
CREATE VIEW user_books AS
SELECT id, title, author FROM books WHERE owner = SESSION_USER();
```

Output



Step 7: Grant Permissions to Users

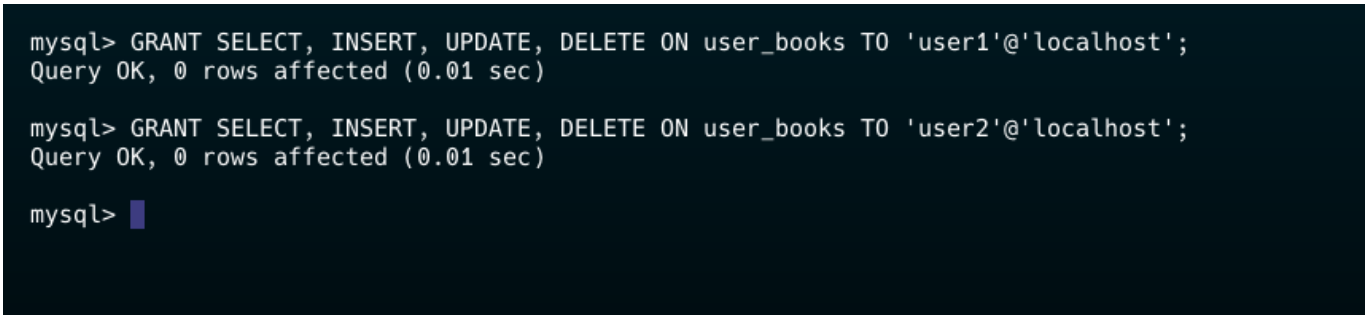
1. **Revoke direct table access** (so users cannot see everything).

```
REVOKE ALL PRIVILEGES ON books FROM 'user1'@'localhost';
REVOKE ALL PRIVILEGES ON books FROM 'user2'@'localhost';
```

2. **Grant access to the `user_books` view** (so they only see their data).

```
GRANT SELECT, INSERT, UPDATE, DELETE ON user_books TO 'user1'@'localhost';
GRANT SELECT, INSERT, UPDATE, DELETE ON user_books TO 'user2'@'localhost';
```

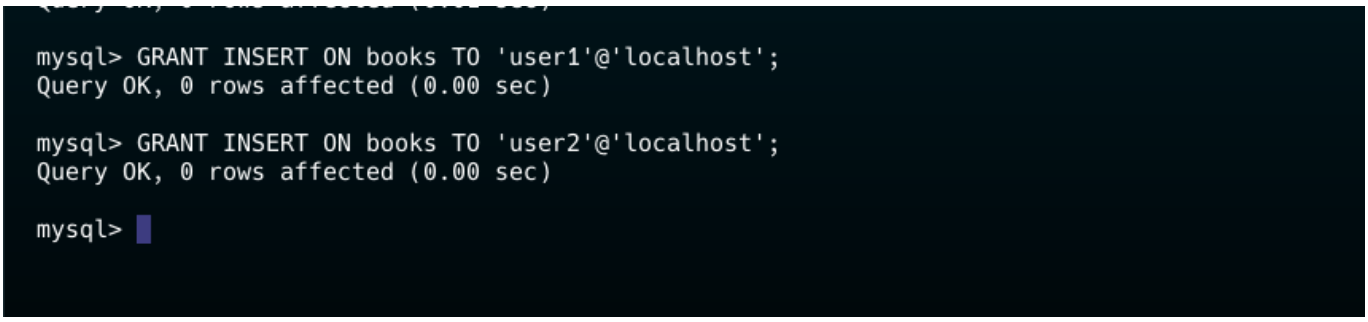
Output



3. **Allow inserting books** (users can insert, but ownership is handled by the trigger).

```
GRANT INSERT ON books TO 'user1'@'localhost';
GRANT INSERT ON books TO 'user2'@'localhost';
```

Output



Step 8: Testing

Test as `user1`

Login as `user1`

```
mysql -u user1 -p
USE rls_practice;
```

Insert a book

```
INSERT INTO books (title, author) VALUES ('User1 Book', 'Author A');
```

Check books (should only see their own)

```
SELECT * FROM user_books;
```

Output should be:

```
+----+------------+----------+
| id | title      | author   |
+----+------------+----------+
|  1 | User1 Book | Author A |
+----+------------+----------+
```

Test as `user2`

Login as `user2`

```
mysql -u user2 -p
USE rls_practice;
```

Insert a book

```
INSERT INTO books (title, author) VALUES ('User2 Book', 'Author B');
```

Check books (should only see their own)

```
SELECT * FROM user_books;
```

Output should be:

```
+----+------------+----------+
| id | title      | author   |
+----+------------+----------+
|  2 | User2 Book | Author B |
+----+------------+----------+
```

🔎 Step 9: Verify Security

Login as `user1` and try to access `user2`'s data:

```
SELECT * FROM books;
```

Access Denied!
Security is working!

Final Summary

What We Achieved

✔ A **single table (`books`)** that stores all users' books.
✔ Users **can only see their own data** (enforced via a `VIEW`).
✔ **A trigger automatically assigns ownership** during insertion.
✔ Users **cannot view/edit/delete books they do not own**.

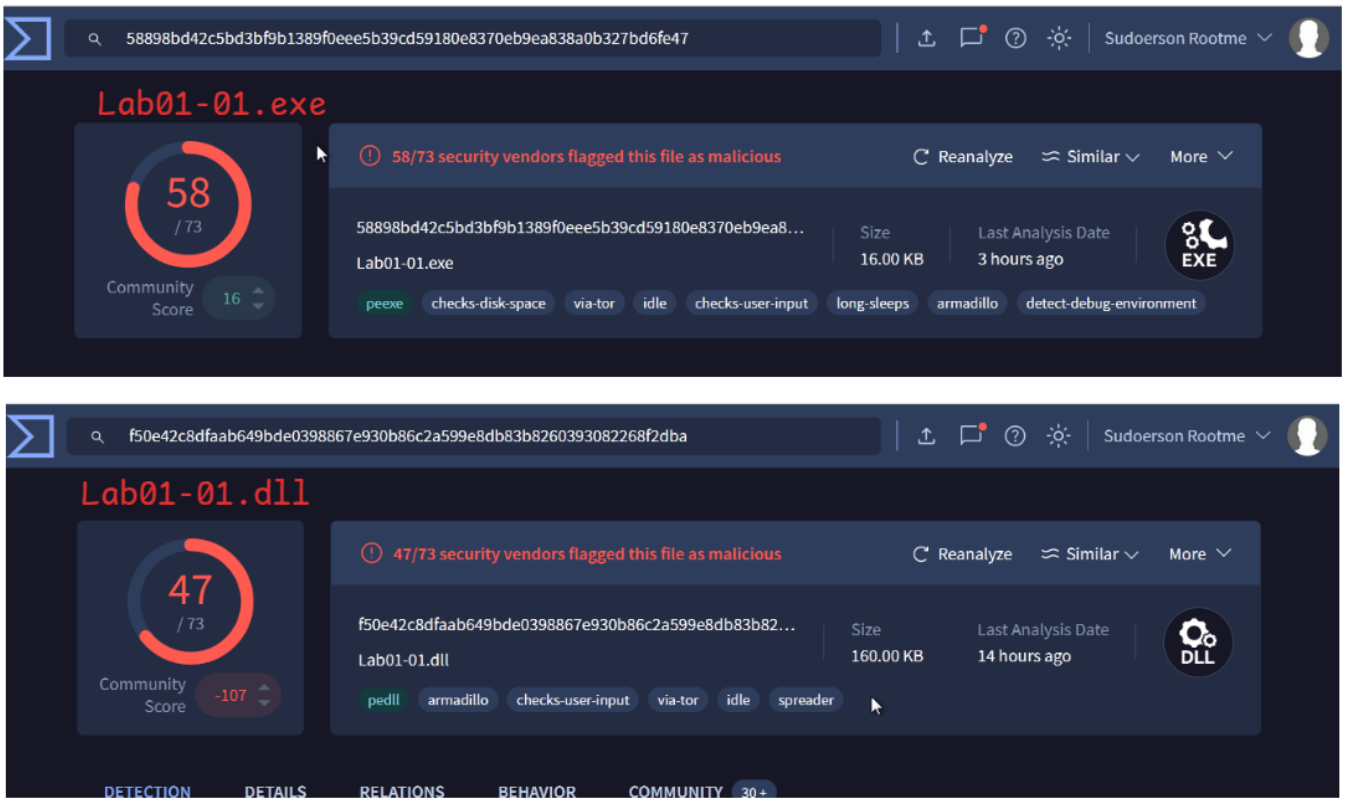
Static Analysis - Lab 9

Lab - 1-1

Questions

1. Upload the files to http://www.VirusTotal.com/ and view the reports. Does either file match any existing antivirus signatures?

Answer

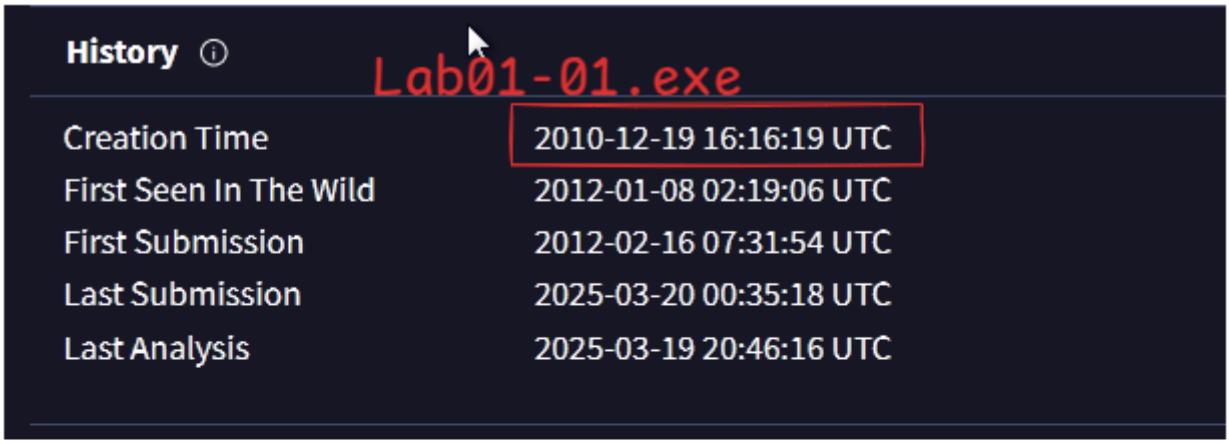


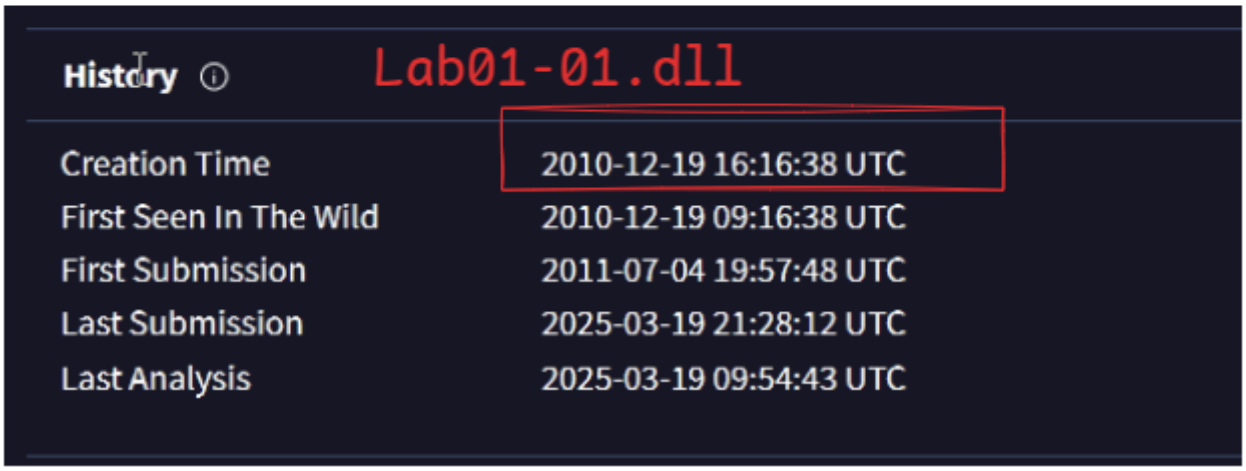
> Yes, 58/73 vendors flagged .exe as malware Yes, 47/73 vendors flagged .dll as malware

---

2. When were these files compiled

Answer



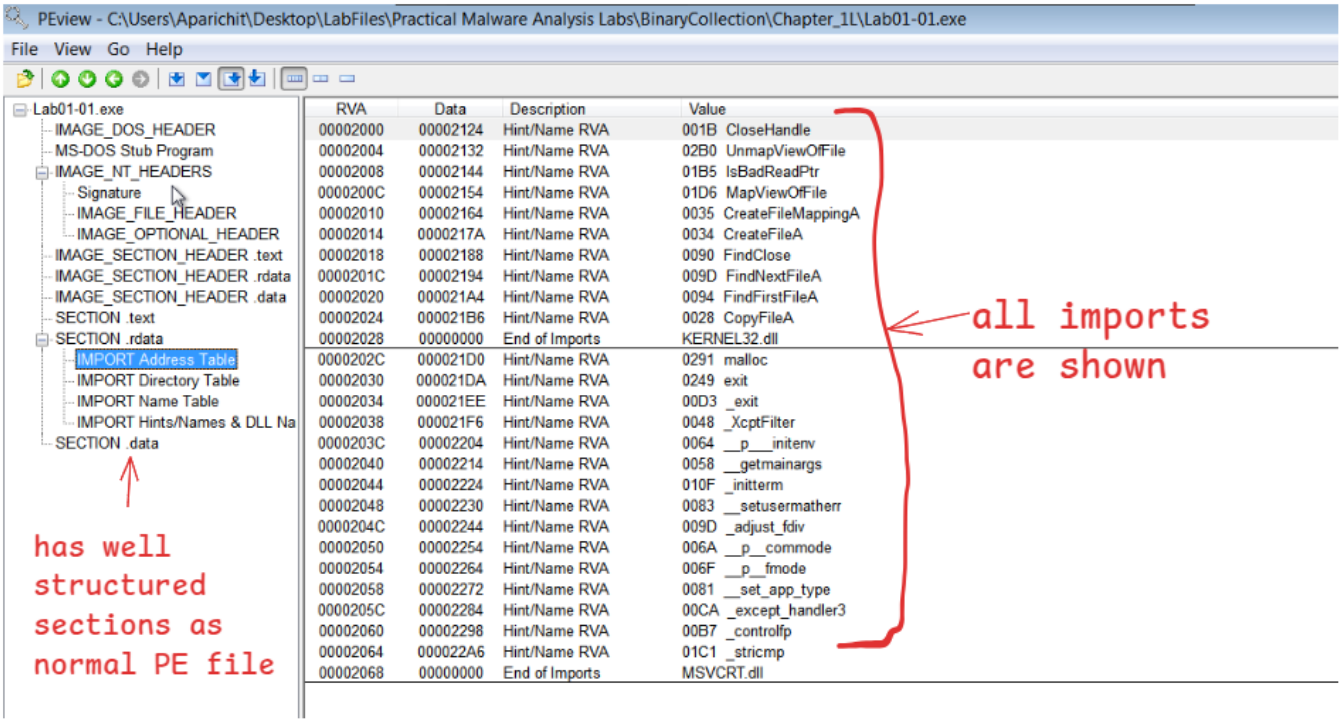


> Both of them were compiled on 19 December, 2010

---

3. Is there any indicators that the files are packed of obfuscated?

Answer

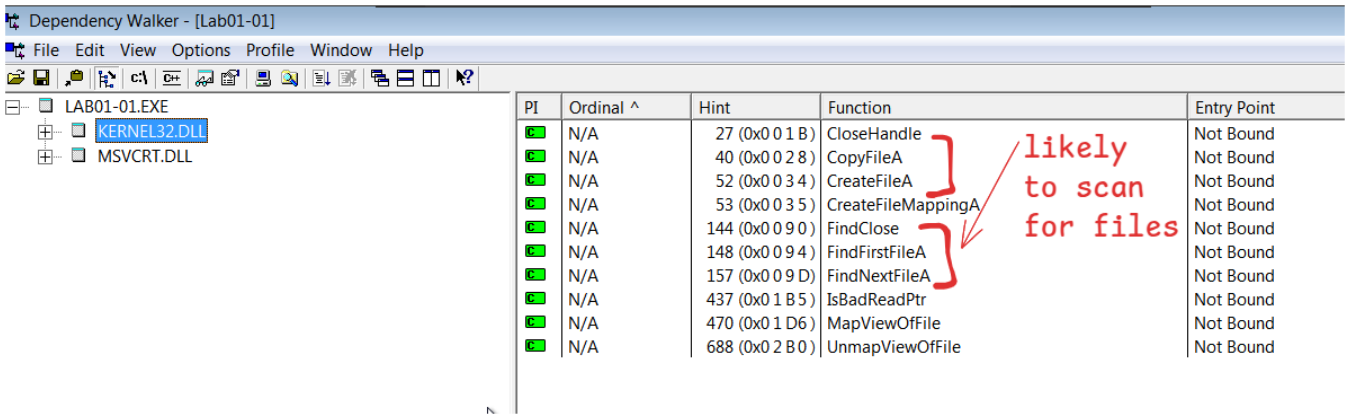


> The file has well structured sections as well as standard imports. So, there is no sign of pack of obfuscation

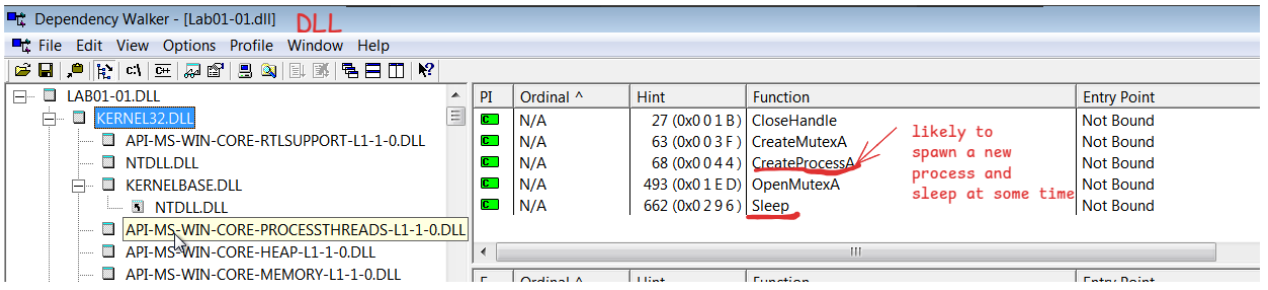
---

4. Do any imports hint at what this malware does? If so, which imports are they?

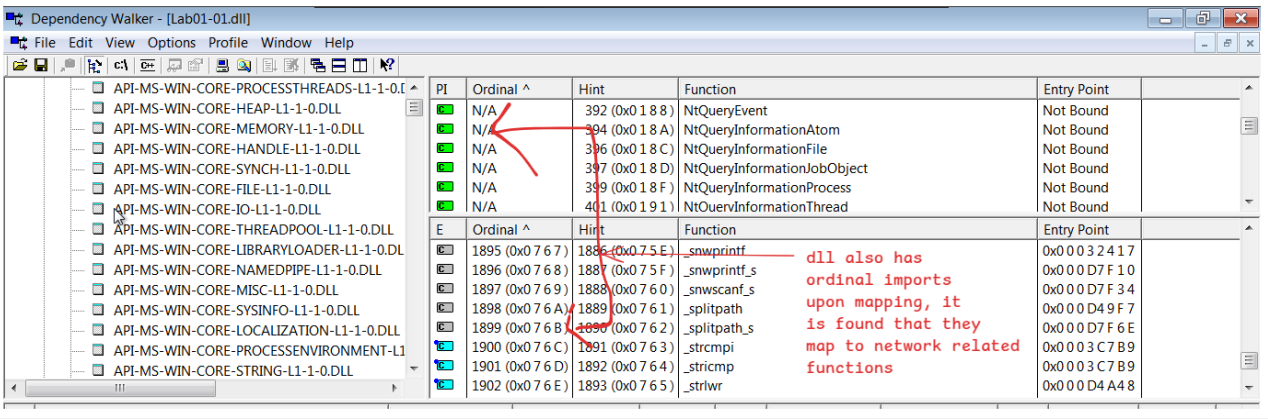
Answer



> The imports in kernel32 contains file operations realted functions. So, it must be looking scanning for files in the filesystem.



> The dll is likely to spawn a new process and sleep at some time

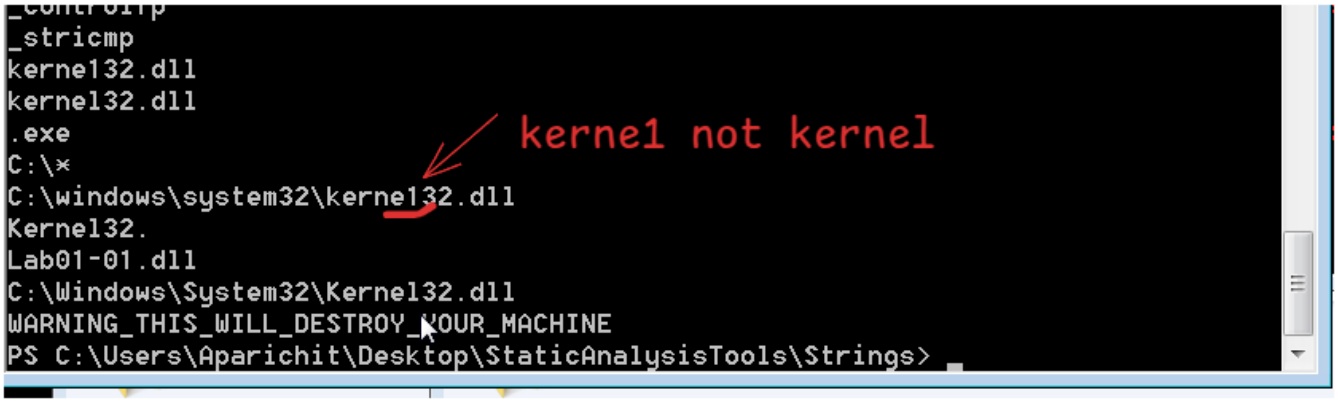


> The dll also ordinal imports, upon mapping they reveal functions from scoket programming showing that the malware is likely to make network connection

---

1. Are there any other files or host-based indicators that you could look for on infected systems?

Answer

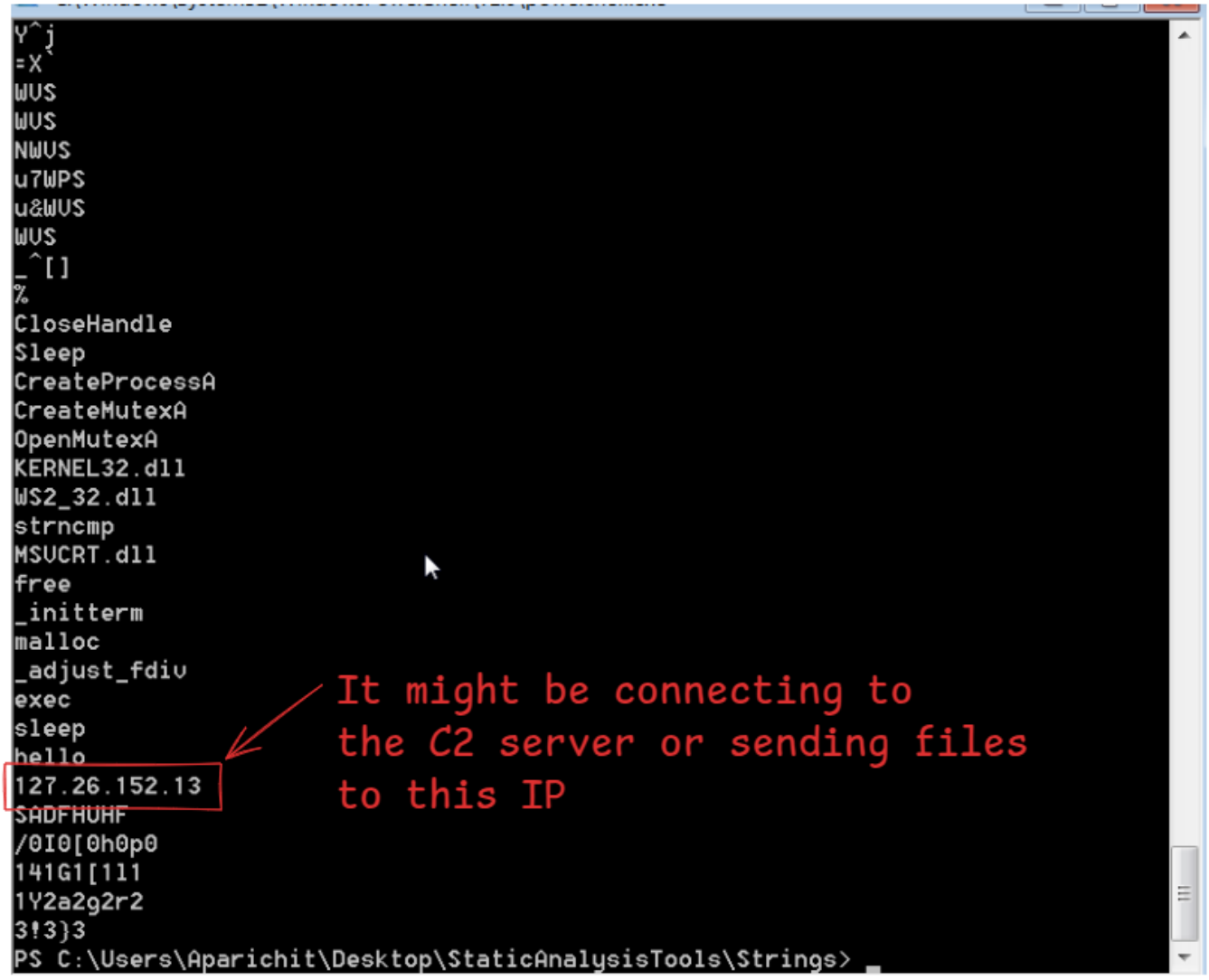


> because of this it is likely malicious and we are able to use this to search for infected systems.

---

1. What network-based indicators could be used to find this malware on infected machines?

Answer



> It must be connecting to the Command and Control Server

---

7. What would you guess the purpose of these files

Answer

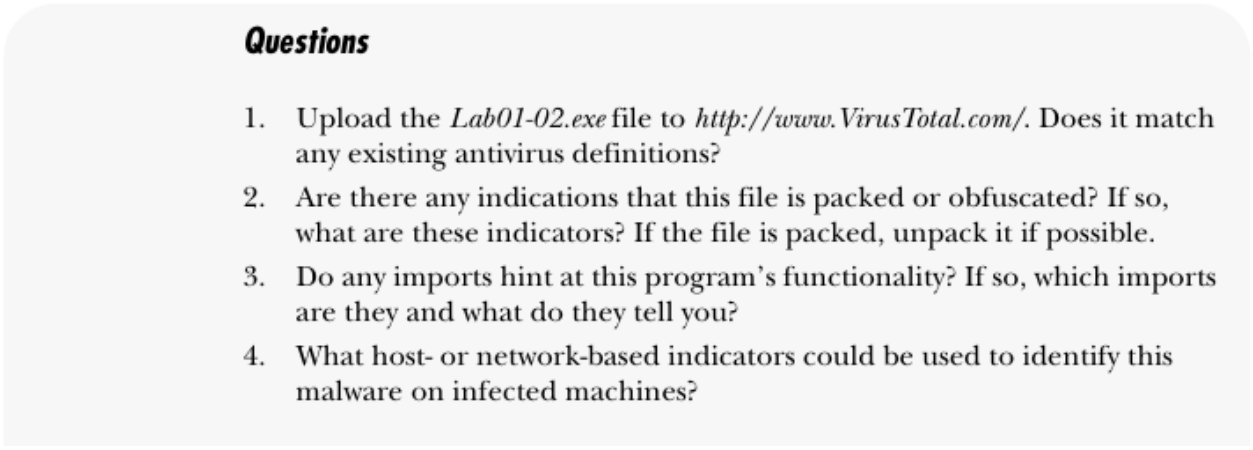
Based on everything we've enumerated above, we would guess that the executable is used to run the DLL which acts as a backdoor or remote access trojan (RAT). Based on the imports it's possible the executable searches to see if C:\windows\system32\kerne132.dll exists, and if it doesn't it may attempt to copy the malicious DLL to C:\windows\system32\kerne132.dll which is used for persistence. Upon executing the DLL, it likely contacts a C2 server at 127.26.152.13.

Lab - 1-2

Lab 1-2

Analyze the file *Lab01-02.exe*.

Basic Static Techniques 27



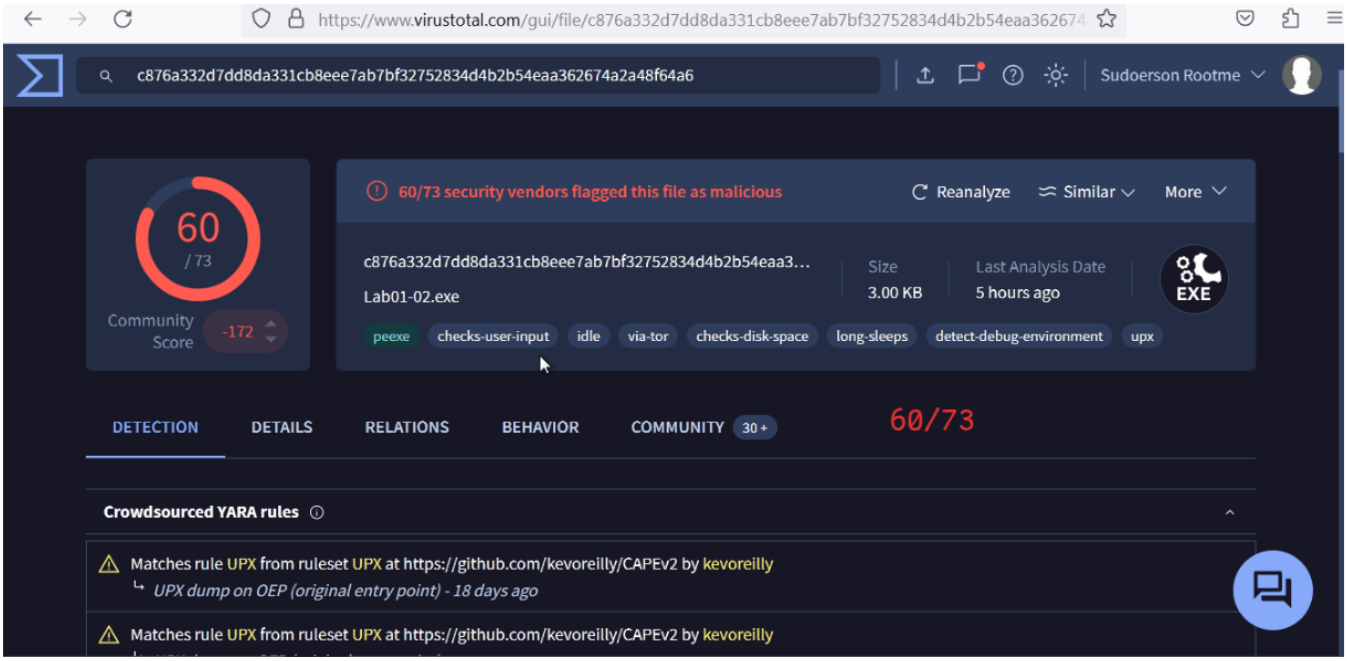
Questions

1. Upload the *Lab01-02.exe* file to *http://www.VirusTotal.com/*. Does it match any existing antivirus definitions?
2. Are there any indications that this file is packed or obfuscated? If so, what are these indicators? If the file is packed, unpack it if possible.
3. Do any imports hint at this program's functionality? If so, which imports are they and what do they tell you?
4. What host- or network-based indicators could be used to identify this malware on infected machines?

Questions

1. Upload the Lab01-02.exe file to http://www.VirusTotal.com/. Does it match any existing antivirus definitions?

Answer

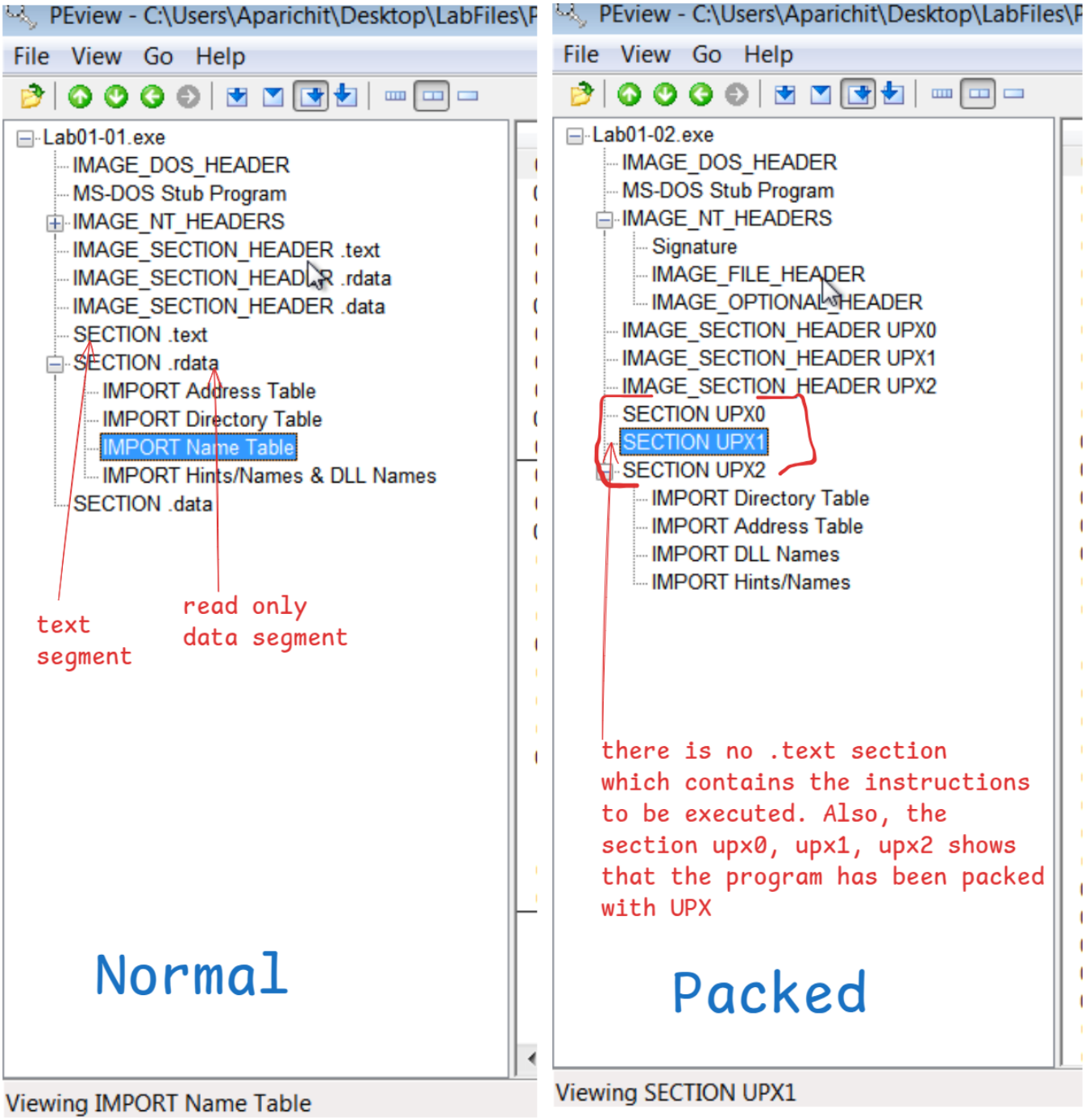


> Yes, 60/73 vendors detects this file as malware

---

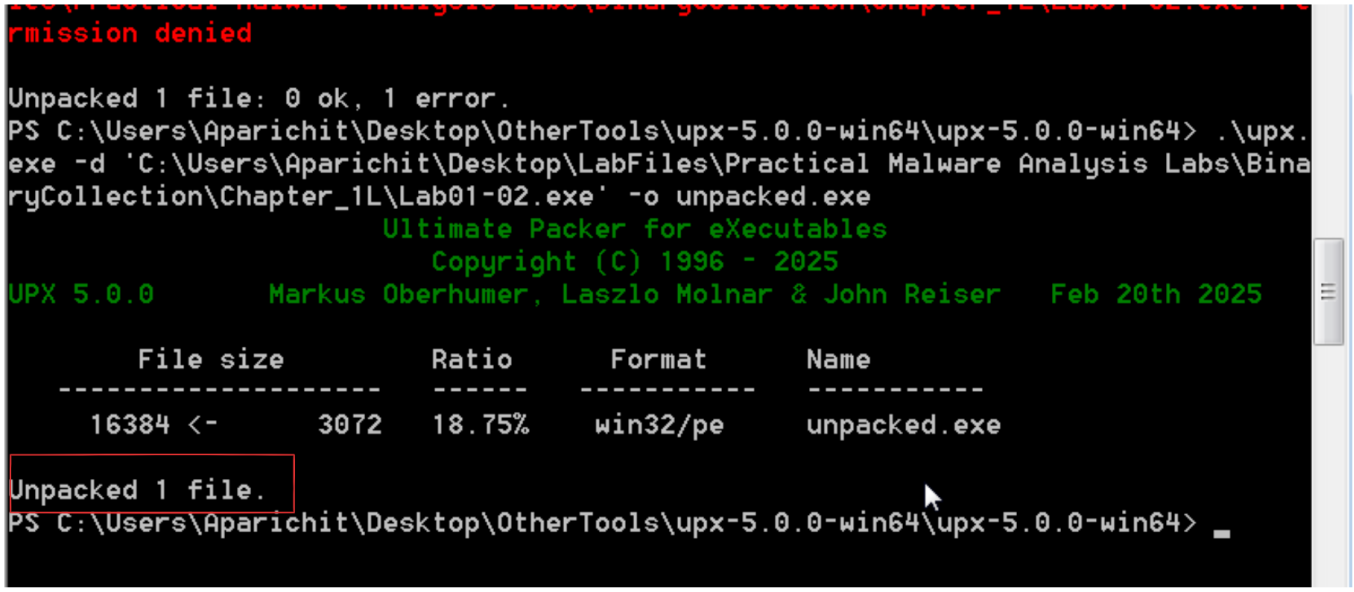
2. Are there any indications that this file is packed or obfuscated? If so, what are these indicators? If the file is packed, unpack it if possible.

Answer



> The file is not having standard sections like .text, .rdata, etc. Also sections like upx0, upx1, upx2, shows that the executable has been packed with UPX

Unpacking it using upx tool

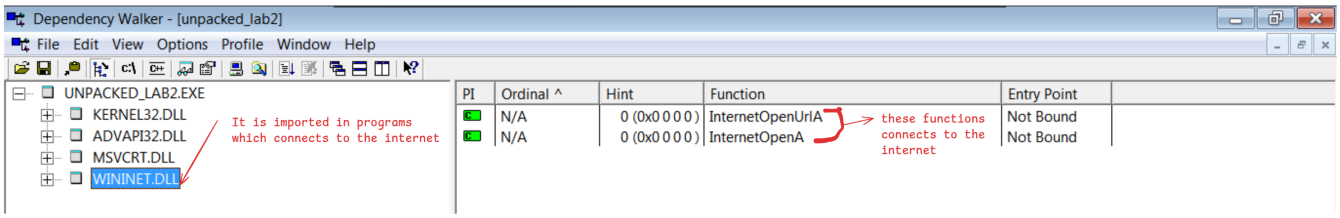


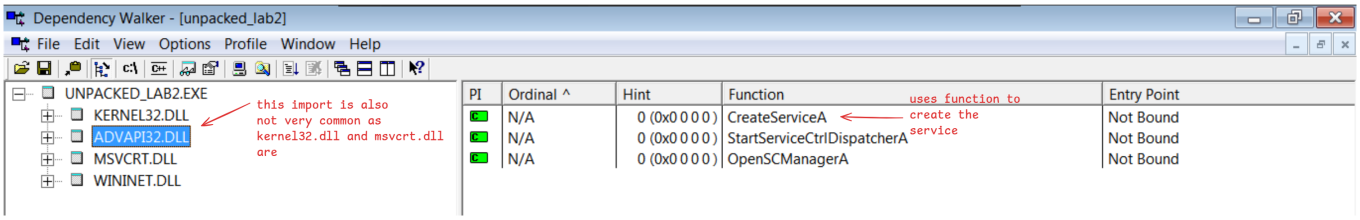

```
upx -d /path/to/packedfile -o unpackedfile.exe
```

---

3. Do any imports hint at this program's functionality? If so, which imports are they and what do they tell you?

Answer



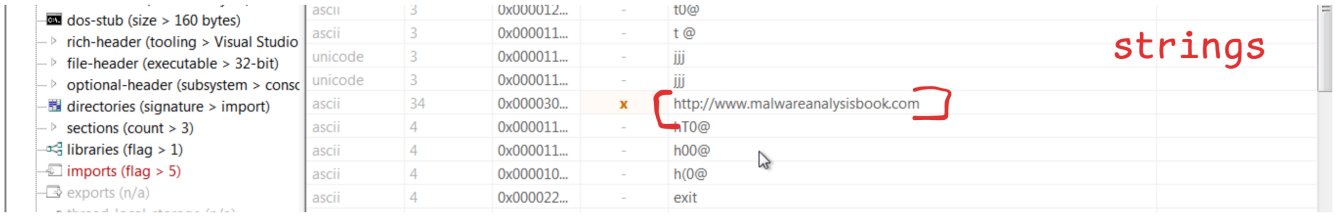


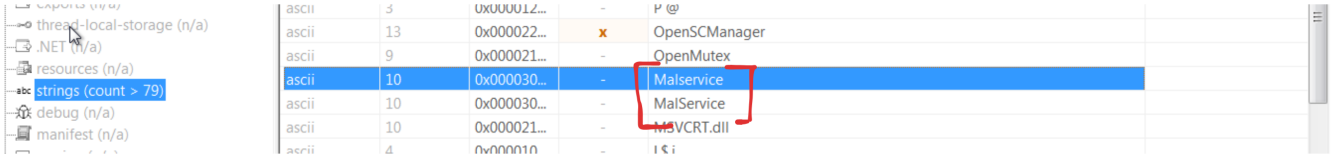
> InternetOpenA, InternetOpenUrlA connects to the internet and CreateServiceA creates a service It could be creating a process which will then be connecting to the internet.

---

4. What host- or network-based indicators could be used to identify this malware on infected machines?

Answer





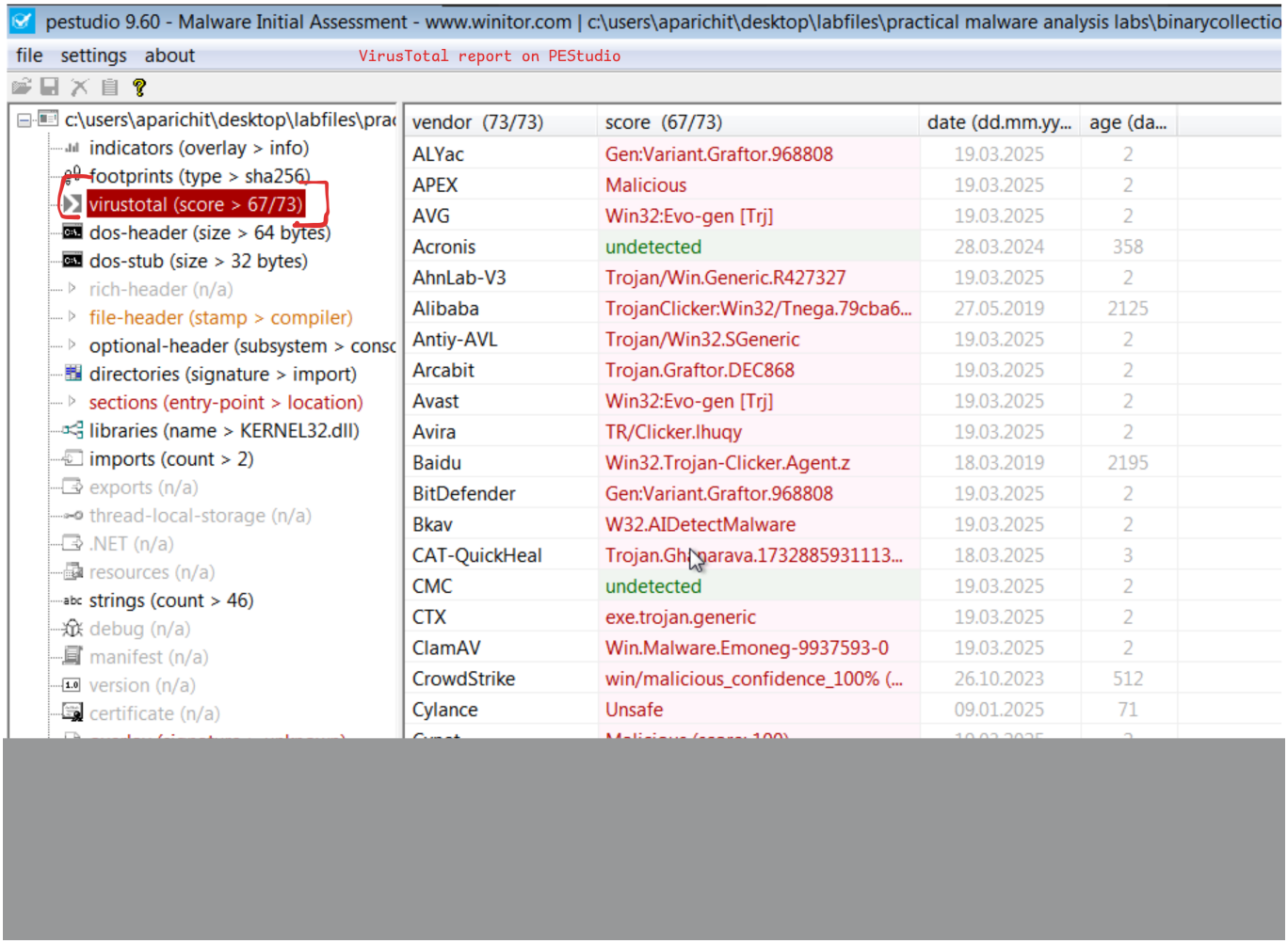
> Looking at the strings of this file shows 2 interesting elements, 'malservice' and 'http://www.malwareanalysisbook.com'. Based on this we can assume that searching hosts for the scheduled service called 'malservice' and looking at any hosts connectiong to 'http://www.malwareanalysisbook.com' would serve as reliable host and network indicators.

Lab 1-3

Questions

1. Upload the Lab01-03.exe file to http://www.VirusTotal.com/. Does it match any existing antivirus definitions?

Answer

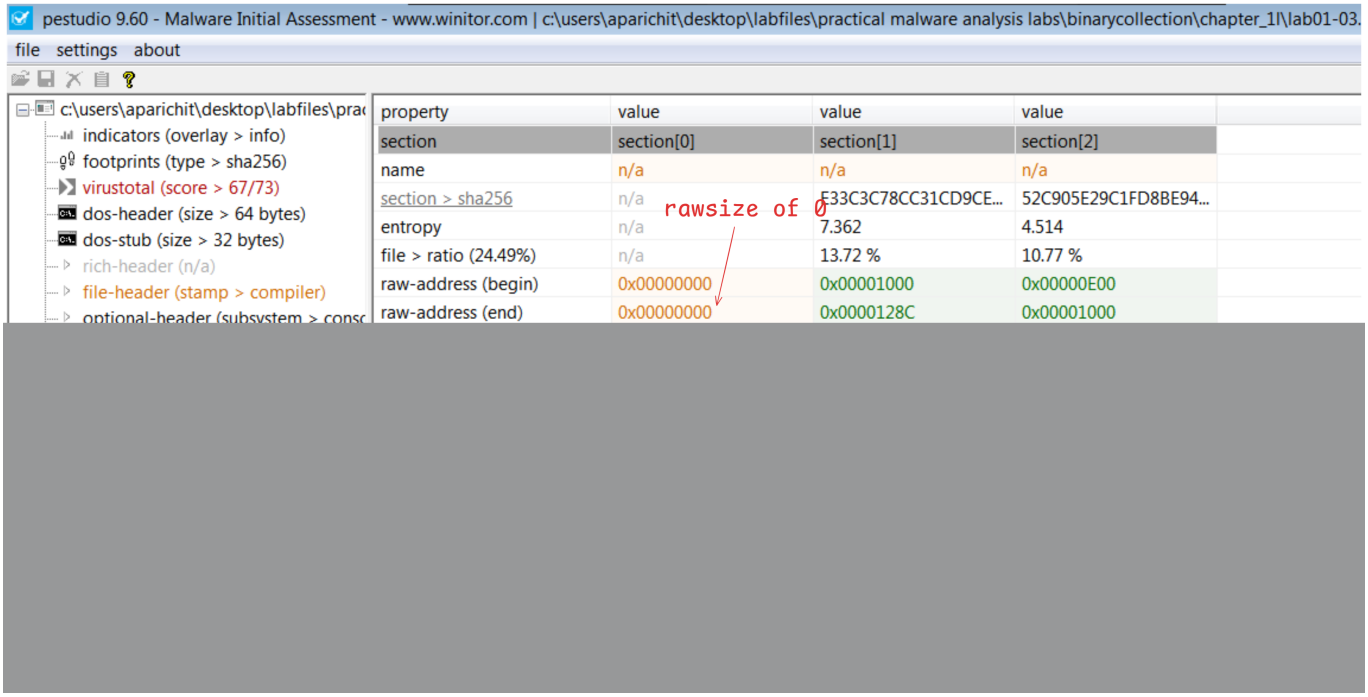


> Yes, 67/73 vendors flagged the .exe as malware

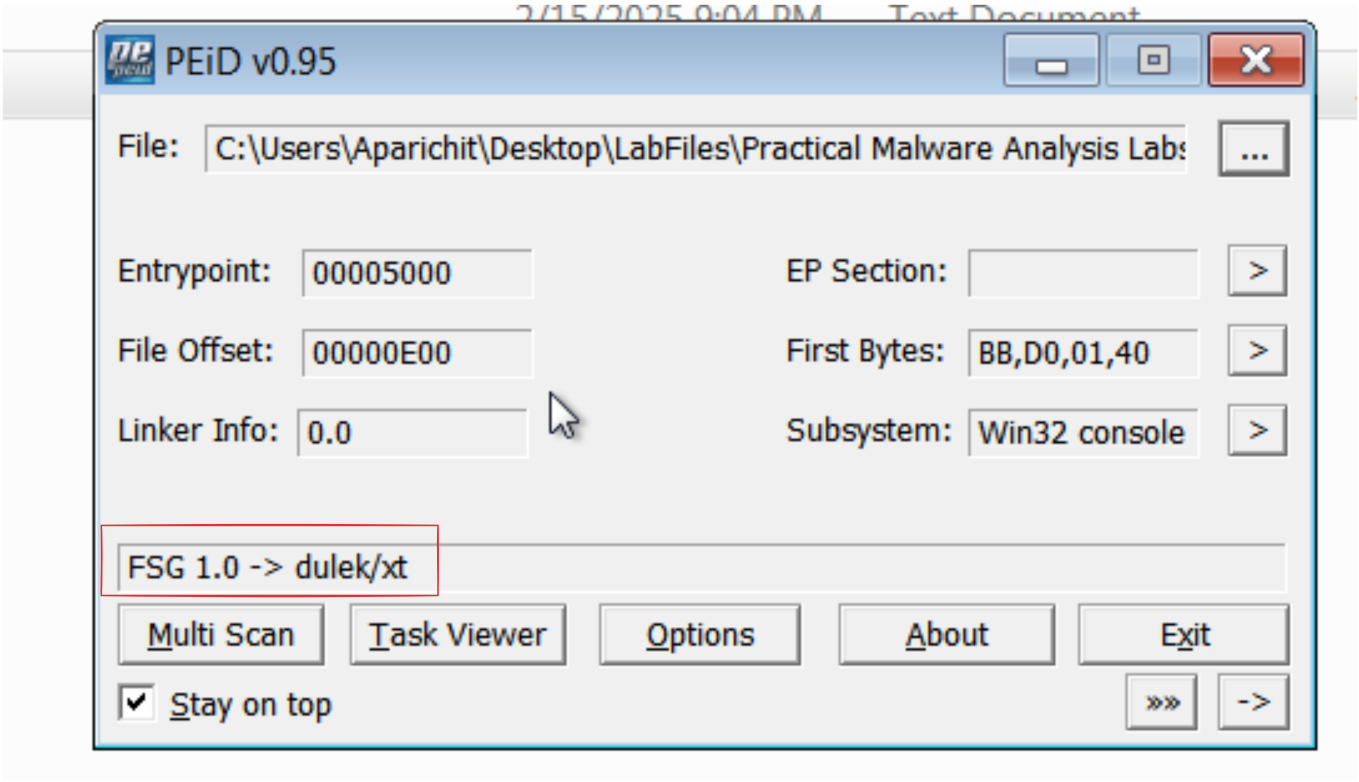
---

2. Are there any indications that this file is packed or obfuscated? If so, what are these indicators? If the file is packed, unpack it if possible.

Answer



> PEStudio shows that the file has virtual size of 0x3000 but the raw size of 0x000. This is an indiciation of that the file might have been packed of obfuscated.



> We can confirm it by using PEID tool which shows that the file is packed with FSG -> dulex/xt.

---

3. Do any imports hint at this program's functionality? If so, which imports are they and what do they tell you?

Answer

> can be answered deobfuscation

4. What host- or network-based indicators could be used to identify this malware on infected machines?

Answer

> can be answered deobfuscation

Lab 1-4

Questions

Lab 1-4

Analyze the file *Lab01-04.exe*.

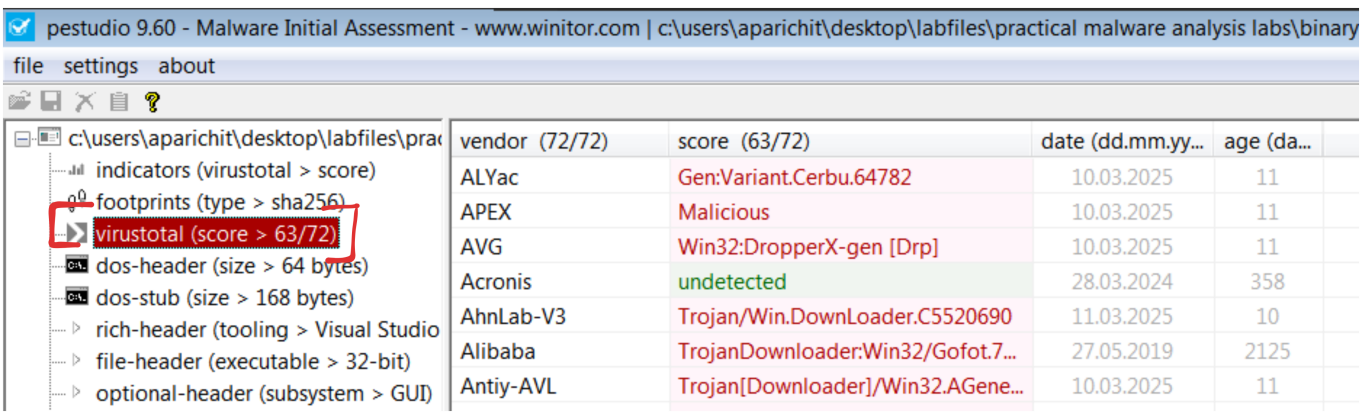
Questions

1. Upload the *Lab01-04.exe* file to *http://www.VirusTotal.com/*. Does it match any existing antivirus definitions?
2. Are there any indications that this file is packed or obfuscated? If so, what are these indicators? If the file is packed, unpack it if possible.
3. When was this program compiled?
4. Do any imports hint at this program's functionality? If so, which imports are they and what do they tell you?
5. What host- or network-based indicators could be used to identify this malware on infected machines?
6. This file has one resource in the resource section. Use Resource Hacker to examine that resource, and then use it to extract the resource. What can you learn from the resource?

28 Chapter 1

1. Upload the files to http://www.VirusTotal.com/ and view the reports. Does either file match any existing antivirus signatures?

Answer

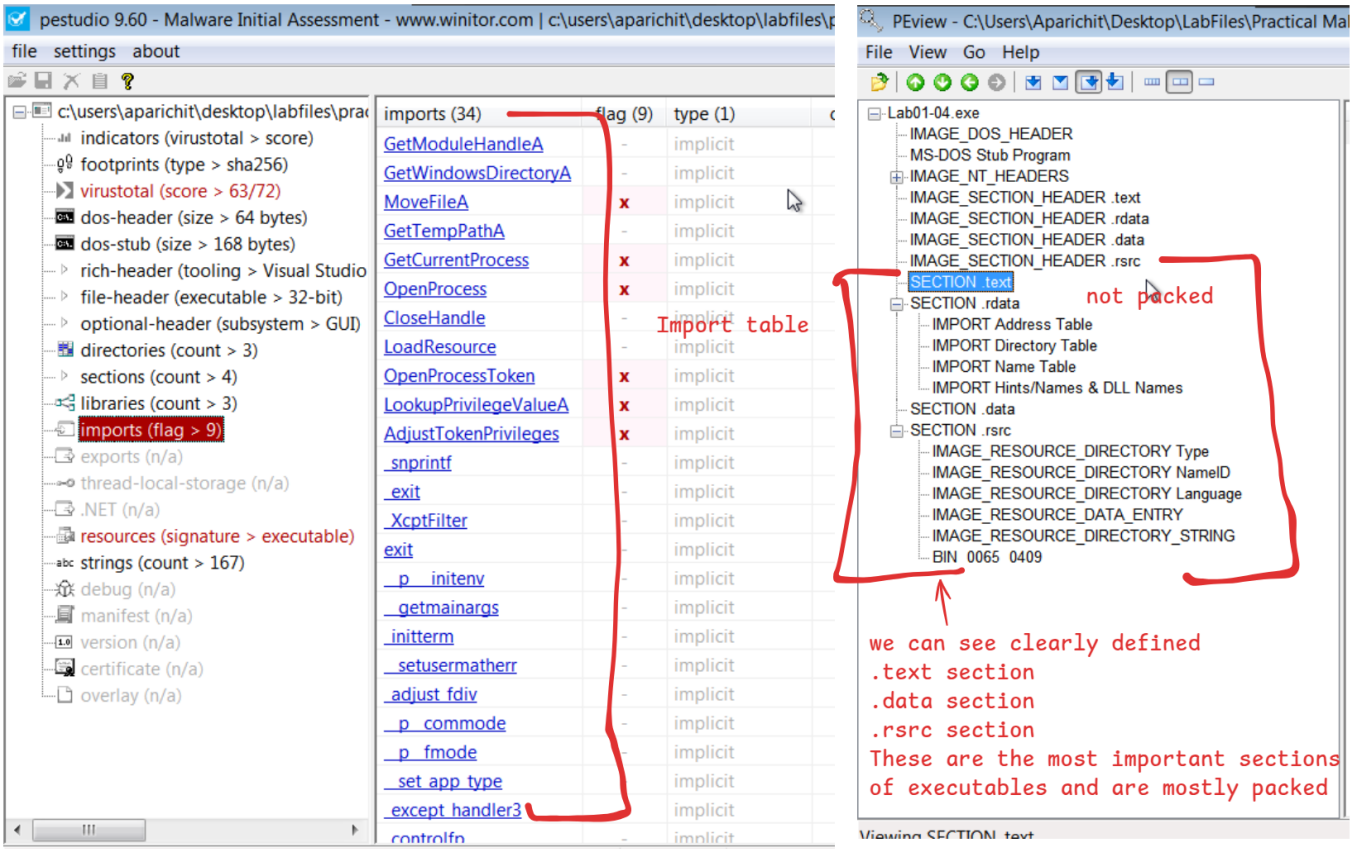


> Yes, 63/72 vendors flagged .exe as malware

---

2. Are there any indications that this file is packed or obfuscated? If so, what are these indicators? If the file is packed, unpack it if possible.

Answer

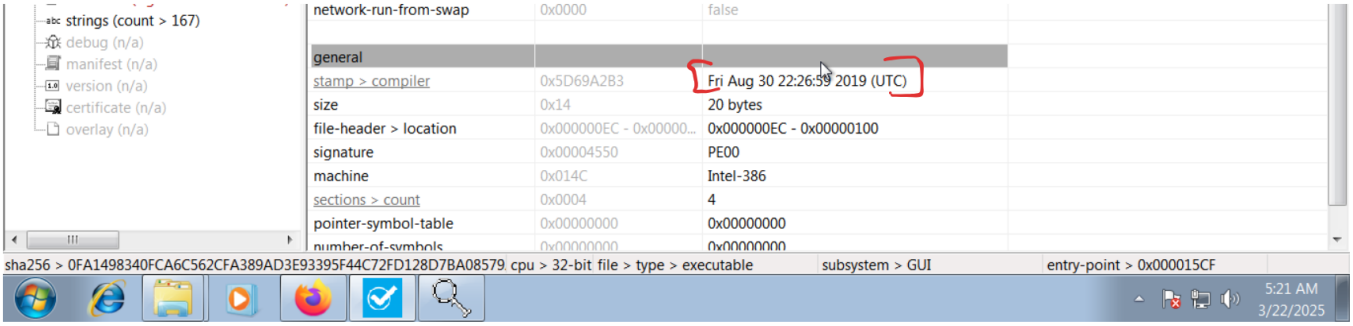


> The file has cleary defined sections and headers. Also it has proper import table. This shows that the program is not packed or obfuscated.

---

3. When was this program compiled?

Answer

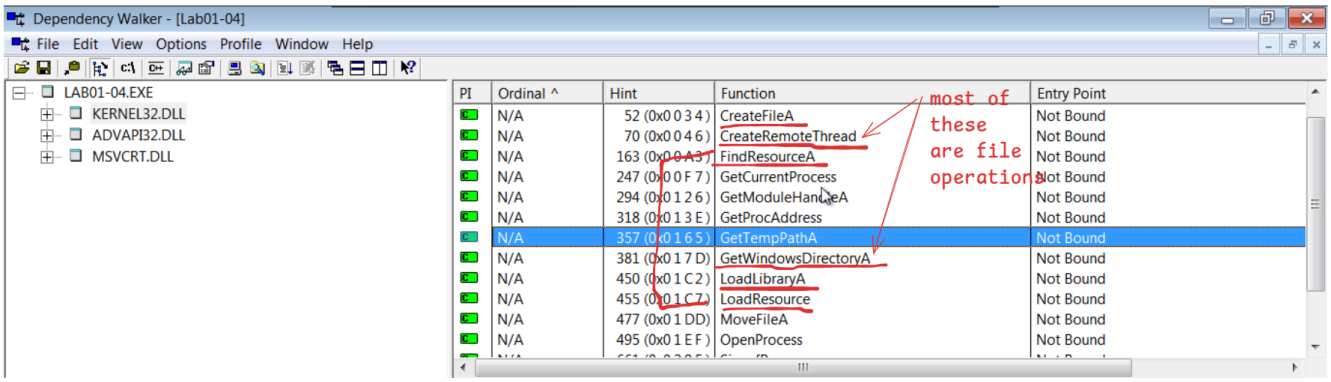


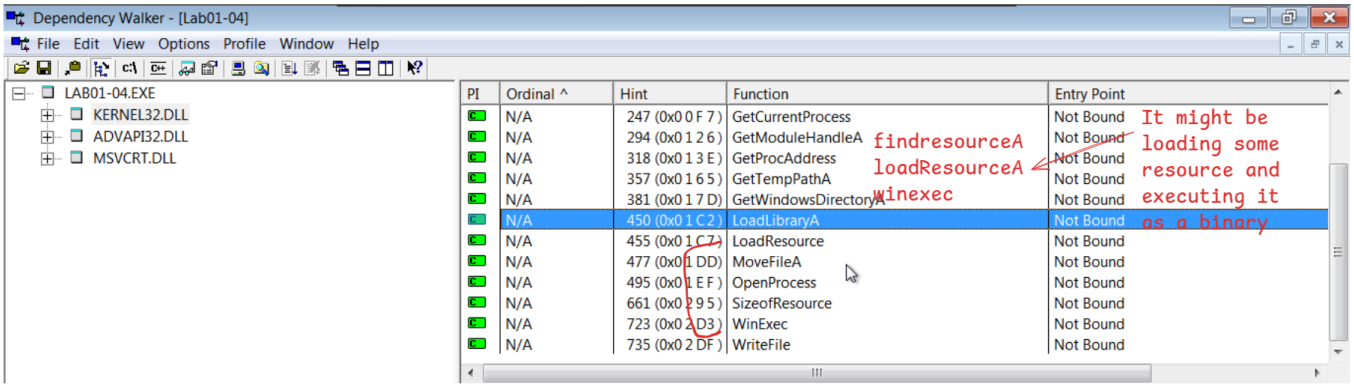
> The compiled time as shown is: Aug 2019. But this might been faked too.

---

4. Do any imports hint about program functionality

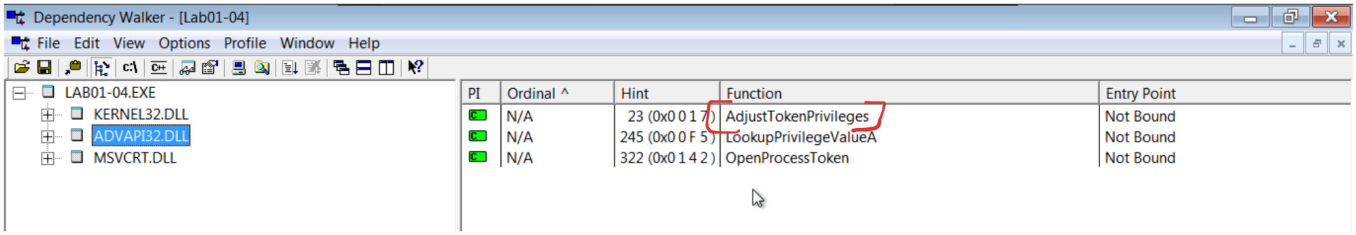
Answer






```
This is suspicious because, we had seen exe file in the resource section in PEStudio
```

> The files has imports for file operations but the imports like FindResourceA, LoadResourceA, winexec make it suspicious. Further imporing .exe from ResourceHacker makes it more suspicious.



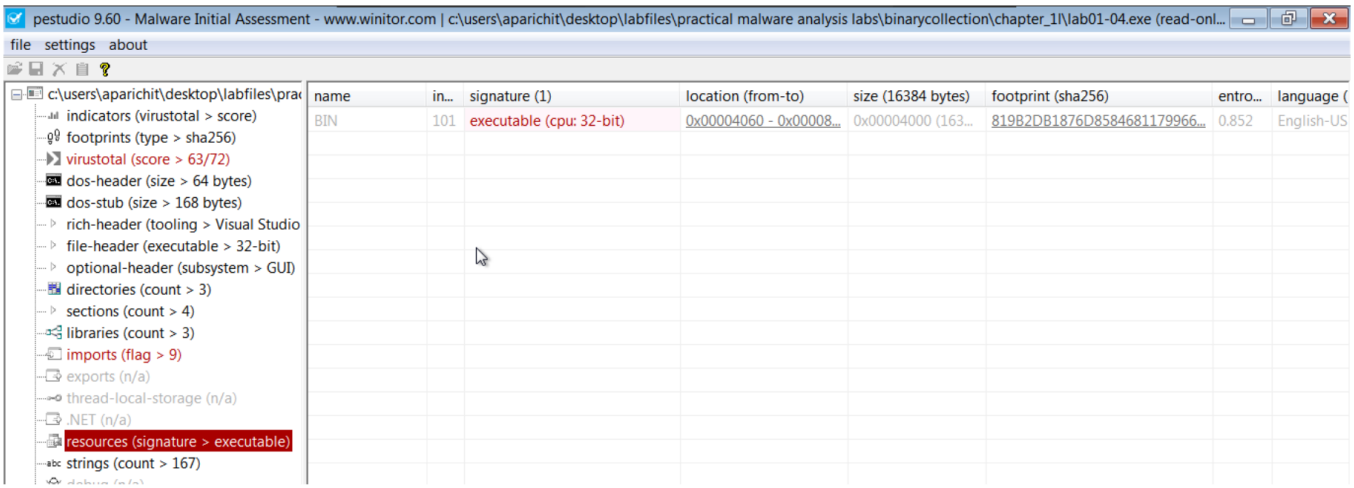
> The imports from Advapi32 indicate that this is attempting to modify or change the token assigned to the execution of this process, presumably to elevate privileges or give extended access rights.

---

5. What host- or network-based indicators could be used to identify this malware on infected machines?

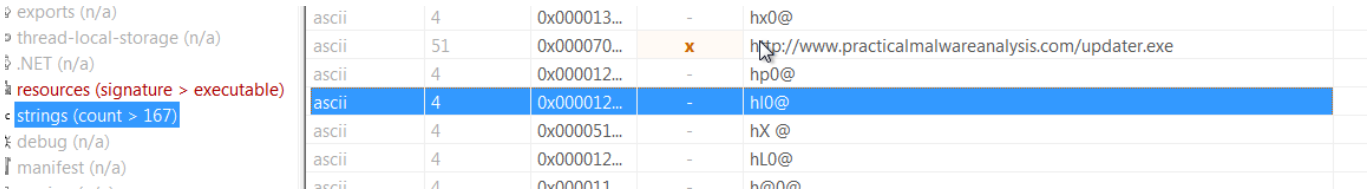
Answer

> Unexpectedly, the malware doesn't seem to have made any imports for network functions.



> But this resource is suspicous.

> If we open the resource in resource hacker, we can find the executable. which was attached as a resource.

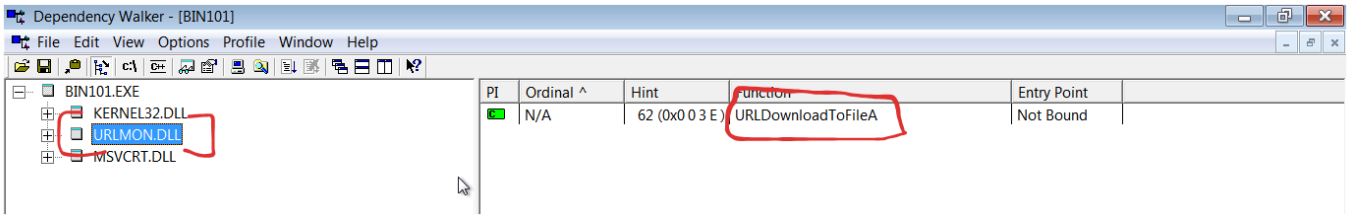


> Also, the string tools also provides some network or host based indicuator

---

6. This file has one resource in the resource section. Use Resource Hacker to examine that resource, and then use it to extract the resource. What can you learn from the resource?

Answer



> The main executable had no imports for networking functions. All networking operations was being done by the attached resource. Also, the url in the strings suggests that the attached resource was downloading another malware for further exploitation.

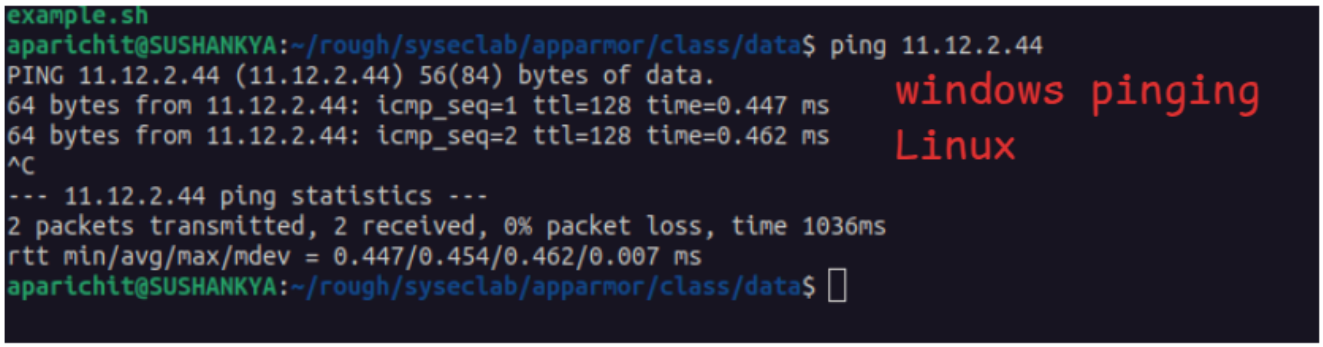
KFSensor Lab - Honeypot : Lab 10

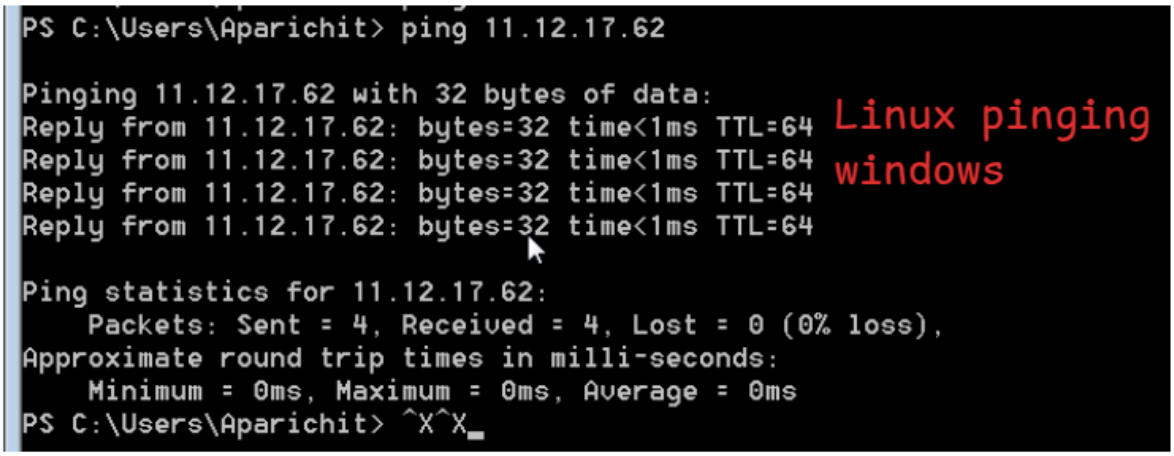
KF Sensor

Initial Setup Info

```
Linux Host machine IP: 1i.12.17.62
Windows VM IP : 11.12.2.44
Connected via bridged interface
```

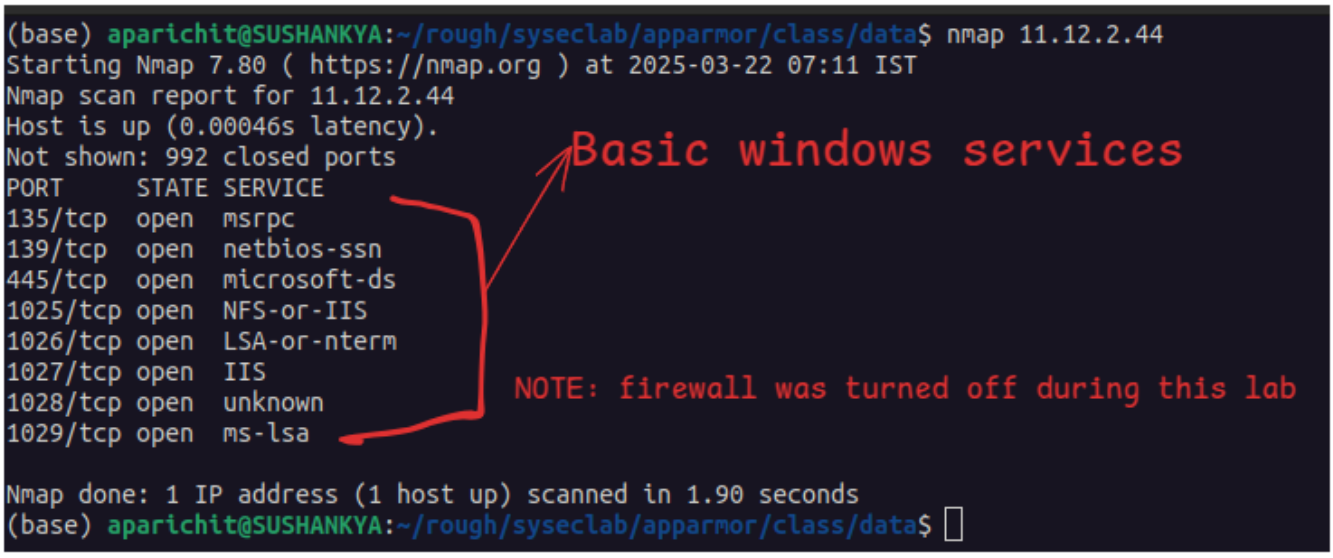
Pinging each other





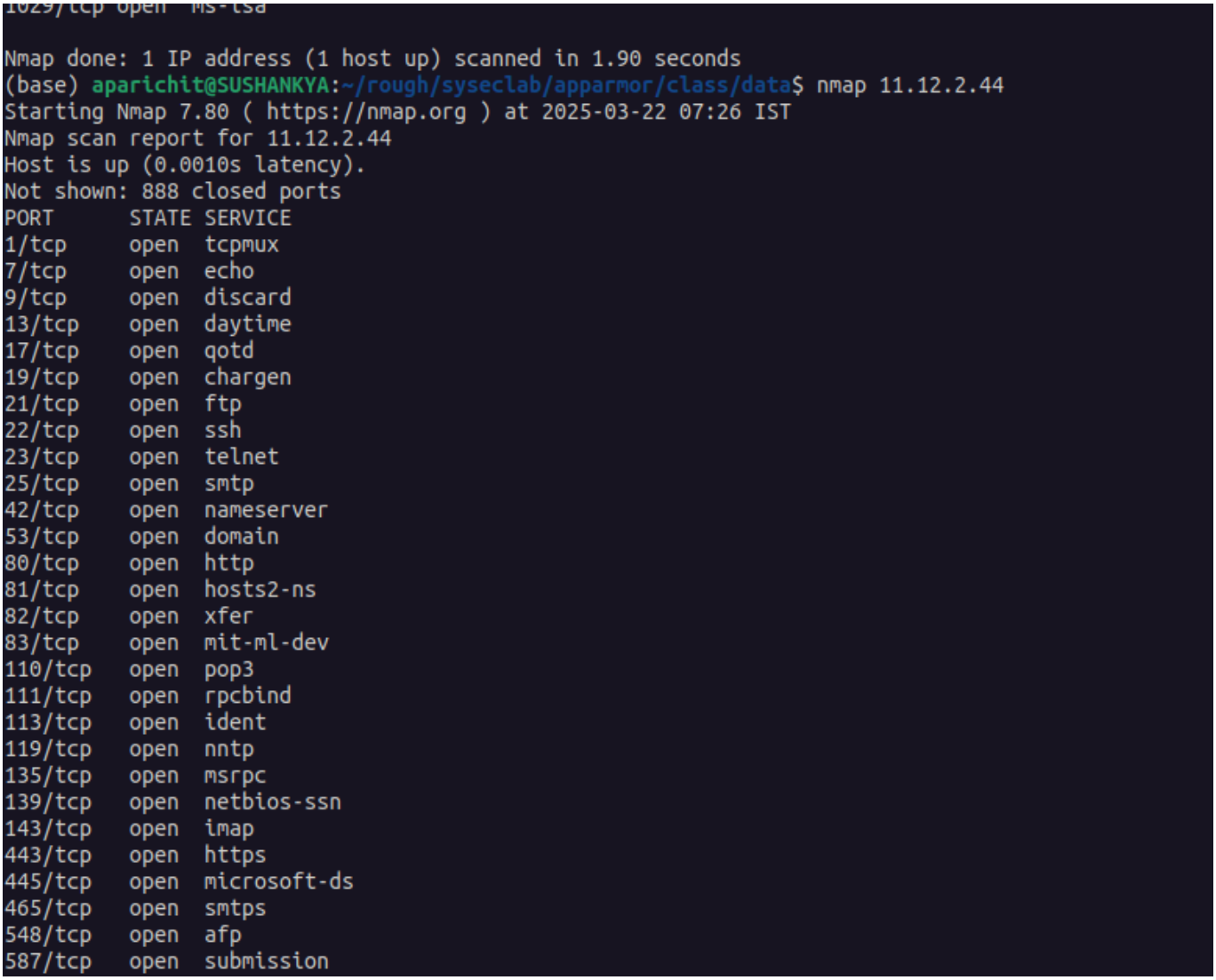
> Both the devices are connected to each other

Initial port status of windows



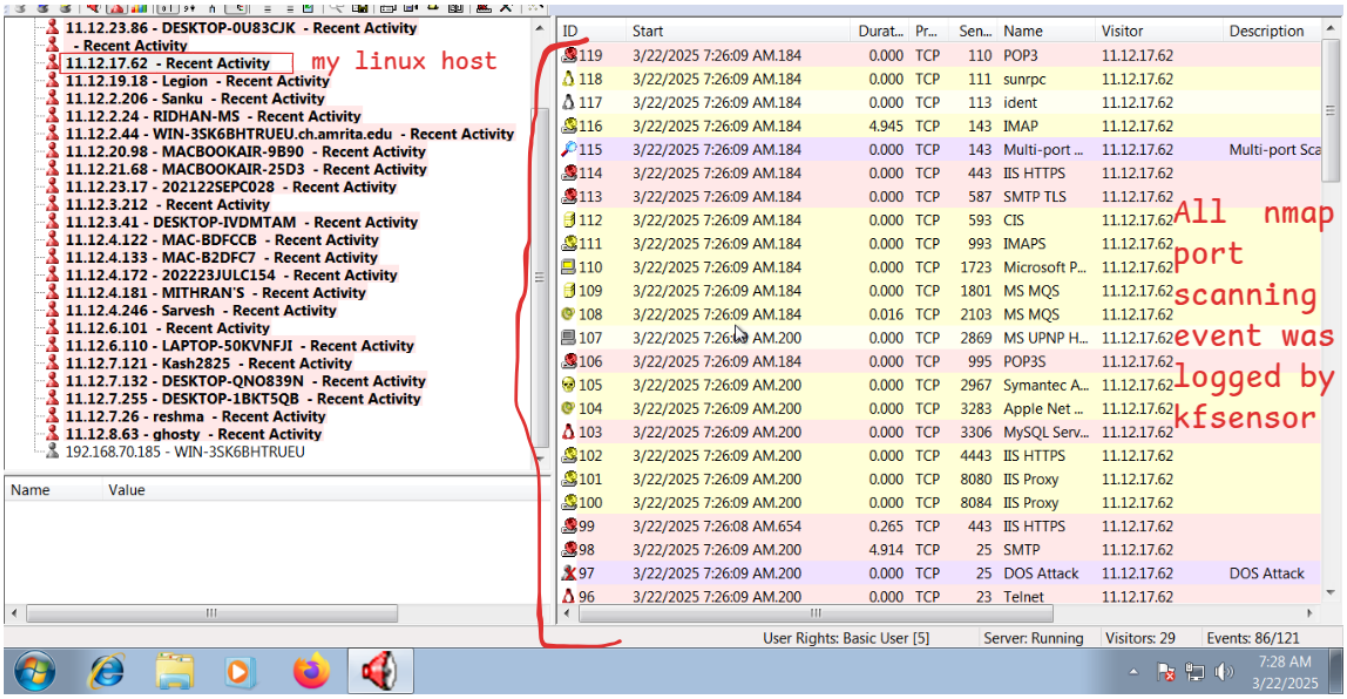
> Basic windows services are open as the windows sytem firewall was turned off

After Starting KF-Sensor



> AFter starting KFSensor we can see a lots of services open. This is exactly what a honey pot does.

Logged nmap scan in Kfsensor



Capturing dos attack using hping3

```
sudo hping3 -S -p 80 -c 100 11.12.2.44
```