Zachary Golla

Security HW 3

1. (15%) Classify each of the following system as an example of a mandatory, discretionary, or originator controlled policy. In each system, state who is the creator, who is the owner of the object, what is the system, who is the admin of the system, and who decides the permission for your selected type of access control.

(a) In a Linux system, a file's permission is set by the owner of the file.

Policy: Discretionary

Creator: User

Owner: User

System: Linux

Admin: User

Who decides permissions: User

(b) In a software repository, a software package can be distributed only with author's consent.

Policy: Originator

Creator: Author

Owner: Author

System: System that has the repository

Admin: Owner of the repository

Who decides permissions: Author

(c) In a classified NSA database, only generals with top secret clearance can search in the database.

Control: Mandatory

Creator: N/A

Owner: N/A

System: System that holds the database

Admin: Person in charge of the database

Permission Decider: Group of persons at the NSA who decide who gets permission

2. (15%) In each of the following situations, specify what type of access (read, write, both, or neither) is allowed with Bell-LaPadula model. If the subject cannot read the object, then give one object classification that the subject can read. If the subject cannot write to the object, then give one object classification that the subject can write.

(a) Paul, cleared for (Top secret, {A,C}), wants to access a document classified (secret, {B,C}).

Paul cannot read or write to the object.

Paul can read a (secret{A,C}) object and can write a (Top Secret{A,C}) object

(b) Anna, cleared for (Confidential, {C}), wants to access a document classified (Confidential, {B}).

Anna cannot read or write to the object.

Anna could read a (Unclassfied, {C} object and could write a (Top Secret, {A,C}) object

(c) Jesse, cleared for (Secret, {C}), wants to access a document classified (Confidential, {C}).

Jesse cannot write to the object, but Jesse can read from it.

Jesse could write to a (Secret, {C}) object.

(d) Sammi, cleared for (Top secret, {A,C}), wants to access a document classified (Confidential, {A}).

Sammi cannot write the object but Sammi can read the object.

Sammi could write to a (Top secret, {A,C}) object.

(e) Robin has no clearance, but wants to access a document classified (Confidential, {B}).

Paul can write to the object but cannot read from the object.

Paul cannot read anything that has a clearance level.

3. (10%) A system implements both Bell-LaPadula and Biba models to enforce confidentiality and integrity simultaneously. Show the access control matrix achieved in the system.

TS S CS UC

TS rw -- -- --

S -- rw -- --

C -- -- rw --

UC -- -- -- rw

4. (20%) An integer overflow is the condition that occurs when the result of an arithmetic operation, such as multiplication or addition, exceeds the maximum size of the integer type used to store it. If the integer in question is incremented past the maximum possible value, it may wrap to become a very small, or negative number, therefore providing a very incorrect value. It is often a critical security flaw in software.

For example, in a 16-bit integer system, 19458\*37=64586. The result is supposed to be 719946, but has more than 16 bits. Therefore, the result is wrapped, i.e. only the least significant 16 bits of the result remain.

Consider in a 32-bit unsigned integer system,

(1) Let Z = 0xFEDCBA98. What is the result of Z + 0x10000000 ? Note that the result must be a 32-bit integer, because this is a 32-bit integer system.

0x10edcba98

(2) Let Y = 0xFEDCBA98. What is the result of Y \* 0x10 ? Note that the result must be a 32-bit integer, because this is a 32-bit integer system.

0xedcba980

(3) Find "X" that satisfies the following equations.

X > 1337

X \* 7 + 4 = 1337

Therefore, X\*7 = 1333

Max decimal of a 32 bit system is: 2,147,483,647

We need X87 that is equal to 2,147,483,647 + 1333 = 2,147,484,980

So, x = 306,783,568.572

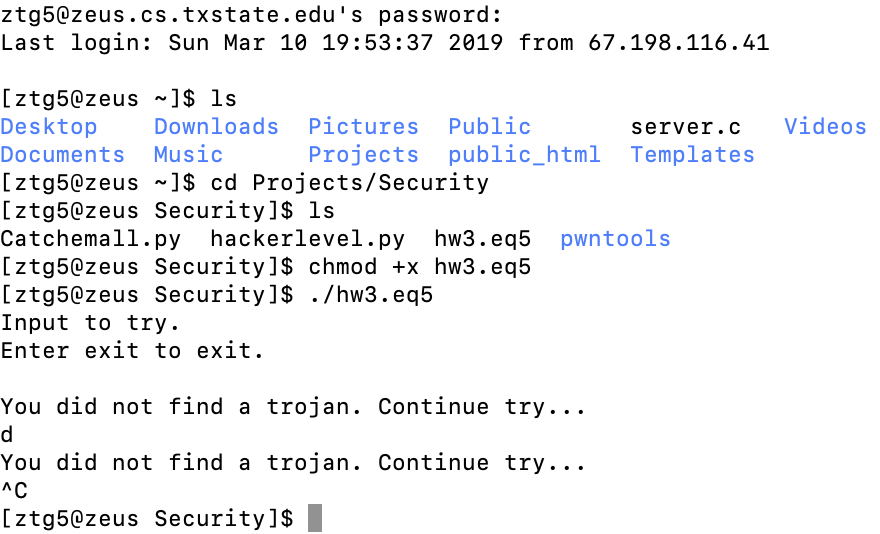
5. (20%) With a reverse engineering tool, we can identify a Trojan that is normally mixed with the other good code in an executable.

Recommend to use NSA'a Ghidra, download and install from https://ghidra-sre.org/

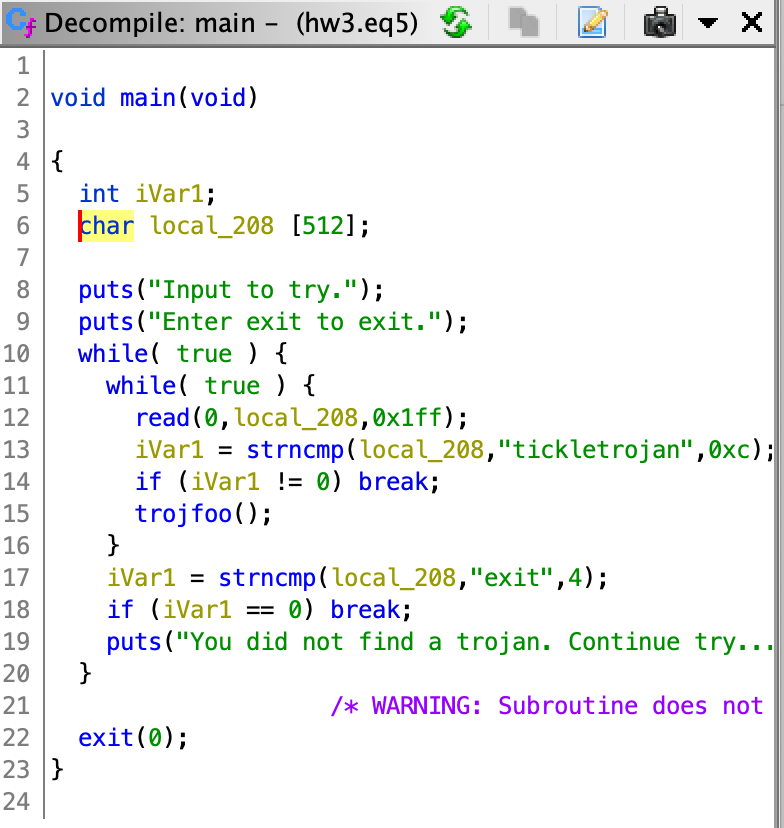
Alternatively, you can download and install Hopper (latest) from https://www.hopperapp.com/ for Linux and MAC, or Snowman (standalone, latest) from https://derevenets.com/ for Windows.

(1) Copy "hw3.eq5" to a Linux computer, for example, zeus.cs.txstate.edu (tested in this server). Make it executable (chmod +x hw3.eq5) and then run it. Show the screenshot of running "hw3.eq5".

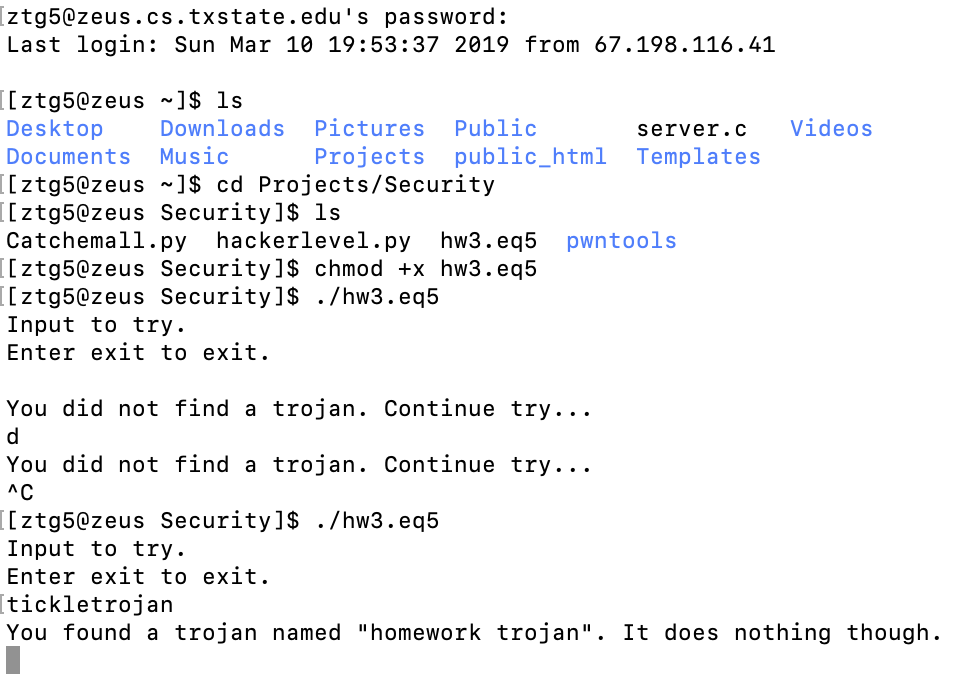
Note, after you copy "hw3.eq5" to zeus, you need to "chmod +x hw3.eq5" and then run "./hw3.eq5" to execute it.



(2) Show the screenshot of decompiling the main() function in your reverse engineering tool (Hopper or Snowman).



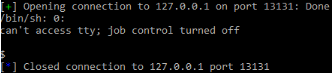
(3) Identify the secret input to trigger the Trojan in "hw3.eq5" and show the screenshot of the output of the Trojan.

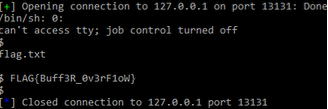


6. (10%) Run the CTF virtual box, read the partial solution of "Super Smash Bros".

(1) Show the screenshot of executing "ls" command after hacking into the server.

(2) Solve the problem and show the screenshot of getting the flag.





7. (10%) Run the CTF virtual box, read the partial solution of "Droid". Solve the problem and show the screenshot of getting the flag.

