CSCI 476: Computer Security
Spring 2023
**Final Lab**

Print your name: Benjamin Haedt
Print your Net ID: S68j891
Print your GID: -01637454

1. What operating system mechanism must be used by software writers to access computing resources, such as memory or disk? Please briefly describe how this mechanism work.

System Calls is the operating system mechanism that must be used by software writers to access computing resources. System calls is the interface that allows interaction between user-level applications and the OS kernel. This mechanism works by switching our input, or in other words elevating our access / privileges from user mode to kernel mode so we can work closer to the operating system.

2.What is a **Set-UID** program? Why do we need them?

A SET-UID program is a program that a user runs with the program owner's privilege.

"Set-UID allows a user to run a program with the program owner's privilege.
- User runs a program w/ temporarily elevated privileges".

What this means is that when a program is owned by say, Alice, then when the program is ran by Bob, it can be executed with the permission of the user Alice.
We need SET-UID programs so that we can allow a special program, or a program with elevated privileges be run by another user who normally wouldn't have access, or the right permissions, to execute that program and the SET-UID program allows the user to access that program to perform a specific function. So, say Alice has access to a file, we can create a SET-UID program that will allow Bob to access that file using the SET-UID program.

3.While studying Set-UID programs, we looked at two functions used to invoke and execute external programs: **system()** and the **exec()** family of functions. Why is **system()** considered unsafe, while **execve()** is considered to be safe?

System() is considered unsafe since it invokes a shell, to elaborate more, it creates a new process that runs with the privileges of the owner of the program we ran our malicious program on, where we can pass in multiple commands that are elevated when running the Set-UID program that shouldn't be allowed. Essentially, we can pass in extra commands when we execute a program with system() function in it since it creates a shell that we can run commands from. It can execute a shell that can run commands on the command line. For example, we can run a malicious program using system() to execute a Set-UID program that will give us elevated privileges and add extra commands if the bash is not the updated one.

execve() is considered to be safe since when we run a program with execve() in it, it executes the program directly and is treated as a single program, so it does not invoke a shell.

4.We studied environment variables as an attack vector for exploiting Set-UID programs. Describe *one* of these approaches that uses environment variables to exploit a Set-UID program (Hint: we looked at two methods: one used PATH and the other one used LD_PRELOAD)

Using PATH to exploit a Set-UID program, if we run a attack on a Set-UID program, if the full path is not provided, the shell process will use the PATH environment variable to search for it. This allows us to access locations that we should not have access to but since we are exploiting the unsecured bash, it will let us access areas a normal user should not have access to. Also, we could set a path variable to a malicious program that invokes a Set-UID program and runs your malicious code.

5.What is the Shellshock attack? What vulnerability is this attack exploiting?

A shellshock attack is a vulnerability in the bash shell. It allows us to execute extra commands since does not check the entire command after it thinks it ends. After the check we can inject anything, we want into the shell.

6. What is a reverse shell? Why did we need a reverse shell during the shellshock attack rather than a normal shell?

A reverse shell is a way that we can get our target machine to connect back to us, open a backdoor, or line of communication between the two computers, and then interact with the victim's computer, hopefully with root privileges. We need a reverse shell during a shellshock attack since once the original shellshock attack has been run, it will only perform the actions in the shellshock attack and not open any lines of communication.

7. What is a **buffer overflow attack**? What is the goal of a buffer overflow attack?

A buffer overflow attack is a way we can run extra code loaded onto the stack. It works because sometimes there is extra space on the stack when code creates extra space that is not needed on the stack, and it writes our malicious code onto memory beyond the stack when loaded. The goal of a buffer overflow attack is to create a reverse shell, execute other code, modify data, escalate privileges, crash a system, or cause/ help in DOS attacks.

A buffer overflow attack occurs when a program attempts to write more data to a buffer in memory than it can hold, causing the excess data to overwrite adjacent memory locations. An attacker can exploit this vulnerability by inserting malicious code into the program's memory and causing it to execute this code, potentially allowing them to gain unauthorized access, modify data, or crash the system.

A buffer overflow attack is a type of cyber-attack that exploits a vulnerability in a software program. It occurs when a program tries to write more data to a buffer, which is a temporary storage area in memory, than it can handle. This extra data overwrites parts of the memory reserved for the program and can cause the program to crash or behave unexpectedly.

In a buffer overflow attack, an attacker can take advantage of this vulnerability by overwriting parts of the memory beyond the buffer with malicious code. If successful, the attacker can execute this code and potentially gain unauthorized access to the system, modify data, escalate privileges, or cause the system to crash.

8. What is the stack? What data gets put onto the stack during execution?

The stack is where all the code to be executed is loaded into to be ran top down, hence the stack. It is a region of memory used by a program during its execution. The data that gets put onto the stack during execution is all the code that we told the computer to run is put. The stack is a last in first out data structure. It stores temporary data related to the program we are loading into it to be run as well as local variables, return address of function calls, values of local variables, and function calls made within the code.

9. What is **shellcode**? Why did we inject shellcode rather than our own malicious program that we compiled?

Shellcode is typically used to deliver a reverse shell so we can have access to the victim machine and run our own commands, if we just ran a malicious program, it would run once and be done probably, with a shellcode we could run as much as we wanted to.

10. What is address space layout randomization (**ASLR**)? Why does ASLR make buffer overflow more difficult? How did we bypass ASLR?

ASLR randomizes the locations of the addresses on the stack every time a program is executed. Making it harder to find the address that will allow us to perform a buffer overflow attack.

To bypass ASLR, we can brute force our way to find where the buffer overflow attack will occur, this involves just going from top to bottom, attempting to run our code in each address of the free space of the stack.

11. Enter your name: ', Naughty_or_Nice='Nice' where name ='Eve'#

12. What is a Cross-Site-Scripting (XSS) attack?

Cross-site-scripting is a way that we can have extra JavaScript code run when a victim attempts to access a certain webpage.

13. During the Cross-Site-Scripting attack lecture, we found a method for stealing another person's browser cookies. How did we get their cookies sent back to us (the attacker) ?

First, we need to set up a Netcat session that will listen on the specified port we want to use, then we construct our malicious XSS code and put it onto a vulnerable website. Once our malicious code is ran, it will direct the cookies from the victims browser and then send the cookies we requested from the victim machine back to our Netcat session on the port specified which should be listening.

14. What is the TCP/SYN flooding attack?

We send a bunch of TCP/SYN packets to a machine, where the host machine will send out syn ack packets back but we will never respond to them. We keep sending the TCP/SYN packet in hopes that it will overwhelm the host and either crash it or cause issues with others connecting since it has a limited number of connections it can handle at once usually.

15. In order to prevent SYN flooding attacks, a security engineer proposes the following solution:

*"We will collect the source IP addresses of all SYN packets, and after 5 retries we will assume that IP address was used for the SYN flooding attack and mark that IP address as "bogus". For all bogus IP addresses we find, we will add a rule to the firewall that will block all incoming traffic from that IP address."*
Will this solution help prevent SYN flooding attacks? Why or why not?

No, for a couple of different reasons.

1. We can spoof the source IP address.
2. We can have multiple computers with different IP addresses to send the requests. Which is the most common form of DDOS attacks that occur today.
3. If we block all traffic that attempted to connect and after 5 tries it assumes that the IP address is just trying to SYN flood, then I'm sure everybody would be blocked from everywhere.

16. In the spoofed packet for the TCP Reset and TCP Hijack attacks, why was the sequence number required for the packet?

The sequence number is required since TCP works by verifying what packet the current stream of data is on. If the sequence number is off, then it will ignore that packet and wait for a packet that has the correct sequence number. TCP works by verifying all of the data is received and in order.

17. What is the goal of a DNS cache poisoning attack?

The goal of DNS cache poisoning is to change the DNS entry in the local DNS server to redirect traffic to a custom IP address. So for example, say we want to visit google.com, but we don't have the IP address required to connect to google.com, we will query our local DNS server to see if it has an entry, if it does, then it will send back the IP address associated with the query and give it to us. If it doesn't have the IP address for google, then it will send a query to another DNS server and check to see if they have it.

18. ECB was one mode of encryption we looked at during symmetric key cryptography. Why is ECB unsecure? How do other modes of encryption (CBC, CFB, CTR) fix this?

ECB is unsecure since it always encrypts the same plaintext block to the same ciphertext block.
"For the same key, a plaintext always maps to the same ciphertext."
This makes it unsecure since we can figure out the original message by using pattern recognition attacks.
CBC, CFB, and CTR fix this by randomizing when encrypting, such as XORing the contents or generating a unique key that is then XORed with the original message being encrypted.

19. What are the three important properties for a cryptographic hash function?

Preimage Resistance, or "one-way", "Given h(x) = z, hard to find x"
Second Preimage Resistance, "Given x and h(x), hard to find y such that h(x) = h(y)
Collision resistance, "Difficult to find x and y such that hash(x) = hash(y)"

20. How can hashing be used for message integrity?

It can take a message and when put through a hashing function allow it to appear as though it is just a random bunch of data that nobody else can read unless you know the key and way that the original message was hashed.

21. From a high level, please describe why RSA / Diffie-Hellman Key Exchange is considered to be secure. (i.e Why is it difficult for a malicious third party to crack the secret key, but not for the two hosts?

Since it takes two large prime numbers and then finds the keys for each person, one that is public and one that is kept private, it is very hard to figure out what two prime numbers were originally used to figure out the keys. RSA relies on the difficulty of factoring large composite numbers, and Diffie-Hellman relies on the difficulty of computing discrete logarithms in a finite field. It would be difficult for a malicious third party to crack the secret key since it would have to find the original two prime numbers used to generate the keys, this requires a lot of computing power and is not practical with most computer systems. The two hosts could find the secret key since they both have the information that can be used to derive the private keys, where they can generate a shared secret key.

22. In RSA, both a public key and private key are used in the cryptosystem. What is the difference between them, and when is each key used?

A public key is the key that you give out to others that allows them to encrypt a message using your public key, then when they send it to you, you can use your private key to decrypt the message. A private key is just that, it is a key that only you should know about and keep safe. A private key can also be used to encrypt a message and anybody with the public key would be able to decrypt the message. Also, since a private key is only supposed to be known by its owner, then it could be used to sign a message and ensure that the message you are sending is indeed from that person. We do this by using somebodies public key to encrypt a message, and then encrypt that using our private key, so when the recipient gets the message, they will use their public key to decrypt the message, and when they do, they will see that they had to use your public key, and then they can use their private key to decrypt the message.

23. As a future computer scientist, why is it important to learn about security?

1. Helps us understand computer systems better.
2. Helps us learn how to prevent security breaches by doing the following.
   a. Keeping your systems up to date.
   b. Understanding the implications of what your code does, such as when creating a C program, make sure to not create extra space on the stack that would allow for a buffer overflow attack.
   c. Understand when there has possibly been a breach in security and how to remedy it.
   d. Understand the types of attacks out there in the world.
3. It makes you more employable, especially when applying for certain jobs such as a system administrator, cyber security specialist, or network administrator.

24. Consider the following software system.

1. What are the assets?

   Honestly, everything is an asset, the clients, the server, the SQL database. That is why it is important to understand the types of different cyber security threats and how we can protect ourselves from them.

2. What can go wrong? What are the threats? Think about all the attacks/security principles that we've talked about

   The specific threats in this model are as follows

   a. SQL injection could be possible depending on what protections and updates you have to your SQL database and what types of SQL attacks are being performed.
   b. A DOS or DDOS attack could be performed on the server, crashing it and or not allowing any other connections to the server that handles the clients connecting to it.
   c. Since it is running a bash shell that is running as a Set-UID program, it could potentially allow us to perform a Set-UID attack or a shellshock attack if we can get the server to run our code and it is an unsecure or outdated version of bash.
   d. Since RaceMessenger is written in C, if it is not programmed properly, it could allow for a buffer overflow attack.
   e. We could also spoof the IP address of a client and with TCP hijacking we could tell the server to send us the data intended for another client to be sent to us, or we could perform a TCP reset attack if we know what other IP address the server is communicating with, though, as long as the server is handling the data and strips out or changes where the data is coming/going, it would protect against it.

3. What mechanisms can we implement to prevent things from going wrong? How can we recover if things go wrong? Think about all the countermeasures/security mechanisms we've talked about

   To prevent things from going wrong, we could do the following.

   a. make sure that everything is up to date. This is the most important part, always keep your software and machines up to date with the latest patches.
   b. attempt to perform our own attacks on the system.
   c. Change settings regarding TCP connection and communication, such as limiting the number of connections to happen all at once from an IP address as well as how long we want to wait for a SYN/ACK response.
   d. Use a different operating system for our server, such as a windows server, though this can open us up to different types of attacks based on windows servers, but would protect us from shellshock attack, even though if we make sure we keep everything up to date on our server, then we should be protected.
   e. Not use Set-UID program and ensure that permissions are kept separate from users and administrators or root users.
   f. Using firewalls with up-to-date lists could potentially block suspicious IP addresses if we detect that they are trying to perform actions that shouldn't be allowed or are not following standard connection. We could also ensure that we use anti-virus and intrusion detection systems that help automate when it sees irregular traffic and data.
   g. Force the use of HTTPS instead of using HTTP for communication, or we could communicate using a different protocol.

h. Not allow users to update their own information, though this is quite silly, and instead have them send into the administrators what they want to update. This could protect us from JavaScript attacks and XSS attacks that the user could potentially enter into their own profile.
i. Hire or just set “bug bounties”, this will encourage people to try and gain unauthorized access to your systems and then report the issue to you for a reward (money).