SOFTWARE SECURITY – COSE451

ASSIGNMENT #2:
TARGETING TOY EXAMPLE

아이샤 - 2022320071

OVERVIEW

Vulnerabilities are security weakness or flaw in a system, network, software, or any other entity that could be exploited by attackers to gain unauthorized access to the system. These vulnerabilities can compromise the CIA (Confientiality, Integrity and Availability) of the system. There are many types of vulnerabilities exist such as buffer overflows, SQL injection and command injection. In this project, I will focus more on command injection vulnerability.

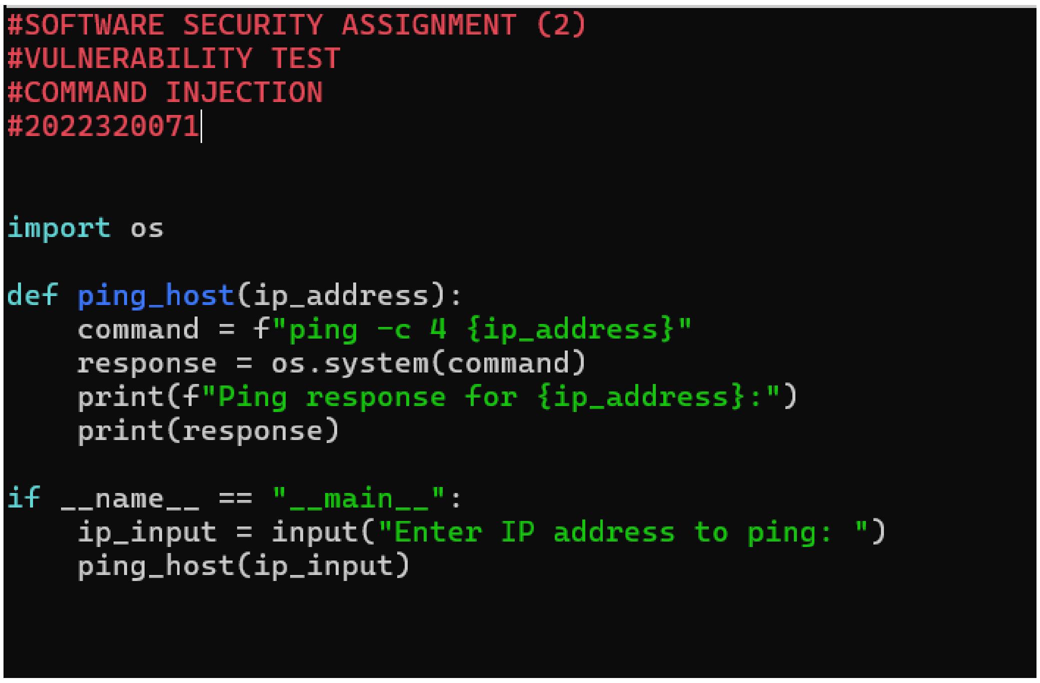
VULNERABILITY: COMMAND INJECTION

Command injection is an attack in which can result to the execution of arbitrary commands on the host operating system via a vulnerable application. A successful attack can cause significant damage because the injection can execute the server's own console commands.

This type of attacks will exploit the failure of application to validate data provided by user before incorporating it into the system command. The attacker can manipulate the command execution to run malicious command just by crafting malicious input that contains special characters or command sequences. Unauthorized access to system resources, service disruption and data exfiltration are some of the consequences of successful command injection. To reduce this risk, developers must employ strict input validation to secure the coding.

VULNERABLE SCRIPT

The code below shows how command injection vulnerability can be exploited.



```python
#SOFTWARE SECURITY ASSIGNMENT (2)
#VULNERABILITY TEST
#COMMAND INJECTION
#2022320071


import os

def ping_host(ip_address):
    command = f"ping -c 4 {ip_address}"
    response = os.system(command)
    print(f"Ping response for {ip_address}:")
    print(response)

if __name__ == "__main__":
    ip_input = input("Enter IP address to ping: ")
    ping_host(ip_input)
```

In the situation where we want to check the reachability of specific device on our network, we can do so with the 'ping' command. After running the script, we need to type the IP address of the specific device we want to check and the script will form a command to ping the address four times. The command will be executed on the computer and output the result.

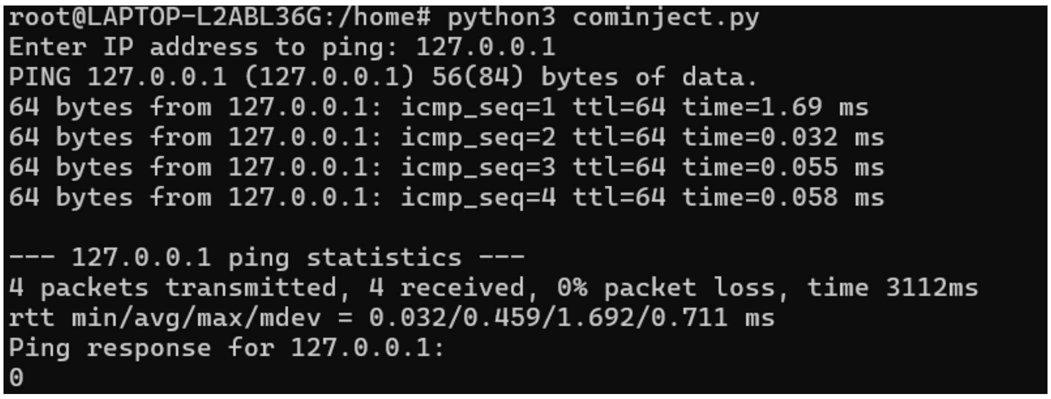
The code consists of 3 main features.

1) OS Module: The OS Module is first imported in the beginning of the code. This module is imported to allow interaction with the operating system including the running system commands.
2) 'ping_host' function: one parameter called 'ip_address' is took in this function. The parameter is then used in the 'ping -c 4 {ip_address}' command string. This line will send 4 ping requests to the specified IP address. Next, 'os.system(command)' function will execute the constructed command in the system's shell. The variable 'response' will store the result of the command execution. Lastly the function prints will be executed.
3) Main block: in the main block, the line 'if __name__ == "__main__": 'ensures that the code inside will not run if it is imported as a module in another script. It must be executed directly to run properly. Inside the block, the script receives input of IP address from user. The input will then be stored in the variable 'ip_input'. Lastly the 'ping_host' function is called.

EXECUTION: Normal input

'127.0.0.1'

Output



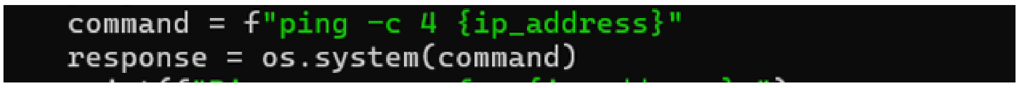
```
root@LAPTOP-L2ABL36G:/home# python3 cominject.py
Enter IP address to ping: 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=1.69 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.032 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.055 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.058 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3112ms
rtt min/avg/max/mdev = 0.032/0.459/1.692/0.711 ms
Ping response for 127.0.0.1:
0
```

The image above shows the example output with normal input. It successfully pings the '127.0.0.1' address. 4 packets were transmitted   and 4 packets were received meaning that no packet loss, It displayed the round-trip times for the packets and execute the exit status '0' confirming that the command is executed successfully.

LOCATION

Although the above python script is functioning correctly but there is a potential command injection vulnerability.

The potential vulnerability is located in the 'ping_host' function, in the line shown below.



```
    command = f"ping -c 4 {ip_address}"
    response = os.system(command)
```

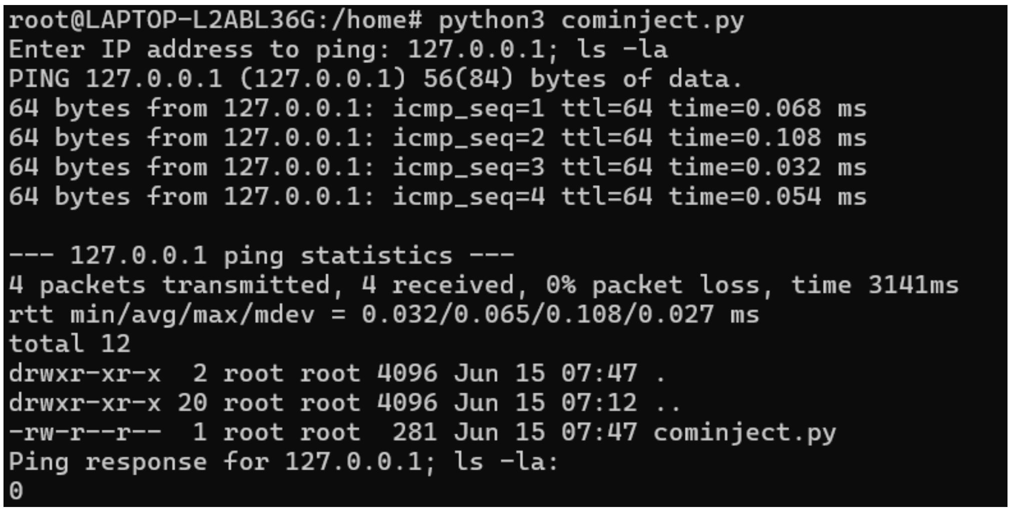
The lines receive user input into the shell command without any proper validation or sanitization. The attacker can freely inject any additional commands using shell metacharacters.

TRIGGERING VULNERABILITY

EXECUTION: malicious input

127.0.0.1; ls -la

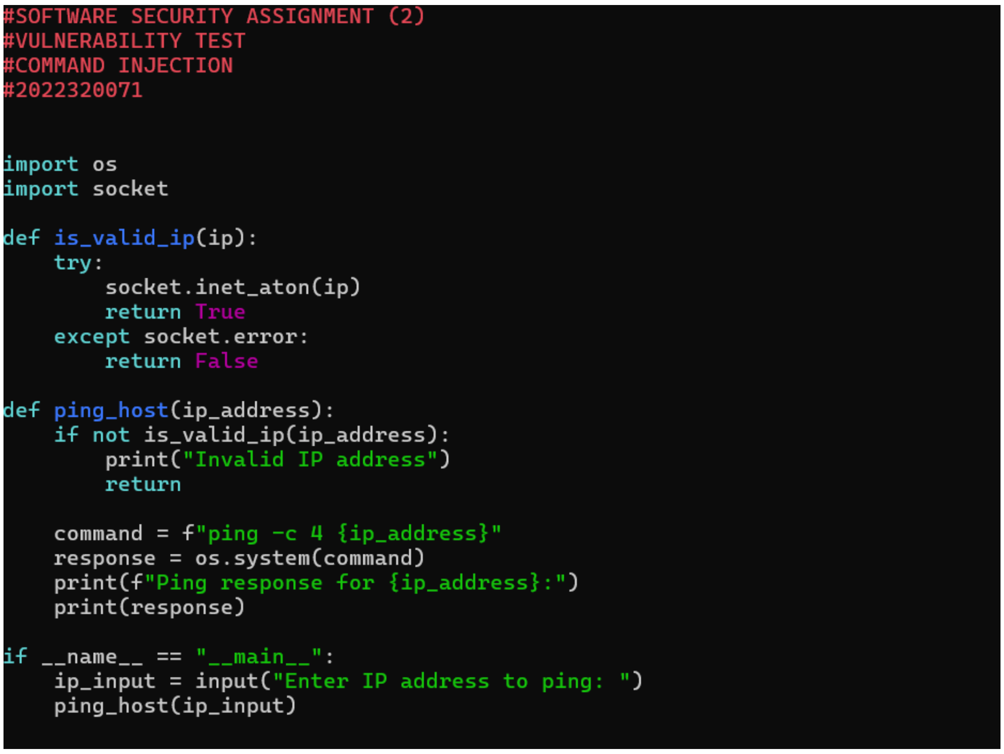
Output



```
root@LAPTOP-L2ABL36G:/home# python3 cominject.py
Enter IP address to ping: 127.0.0.1; ls -la
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.068 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.108 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.032 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.054 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3141ms
rtt min/avg/max/mdev = 0.032/0.065/0.108/0.027 ms
total 12
drwxr-xr-x  2 root root 4096 Jun 15 07:47 .
drwxr-xr-x 20 root root 4096 Jun 15 07:12 ..
-rw-r--r--  1 root root  281 Jun 15 07:47 cominject.py
Ping response for 127.0.0.1; ls -la:
0
```

The image above shows the result of command injection attack. The input ‘127.0.0.1; ls -la’ is not sanitized or validated. That is why both ‘ping’ and ‘ls -la’ command are executed by shell. The command ‘ls -la’ executed and as a result, the files in current directory are exposed! This can lead to unauthorized access, data exfiltration, system compromise, or denial of service.

PATCHING VULNERABILITY



```
#SOFTWARE SECURITY ASSIGNMENT (2)
#VULNERABILITY TEST
#COMMAND INJECTION
#2022320071


import os
import socket

def is_valid_ip(ip):
    try:
        socket.inet_aton(ip)
        return True
    except socket.error:
        return False

def ping_host(ip_address):
    if not is_valid_ip(ip_address):
        print("Invalid IP address")
        return

    command = f"ping -c 4 {ip_address}"
    response = os.system(command)
    print(f"Ping response for {ip_address}:")
    print(response)

if __name__ == "__main__":
    ip_input = input("Enter IP address to ping: ")
    ping_host(ip_input)
```

This is the revised version of the original python script. It includes security measures that can prevent command injection vulnerabilities. Here are some additional features added into the script.

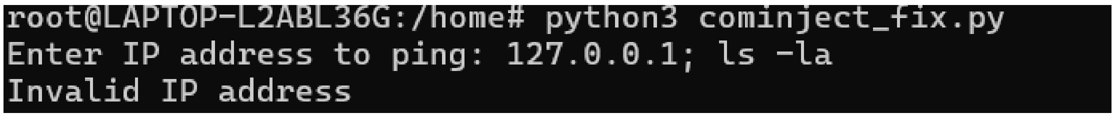
1. ‘socket’ Module : ‘socket’ module is added for IP address validation.
2. ‘is_valid_ip(ip)’ function: this is where we can prevent the command injection vulnerability. It validate the input from user. ‘socket.inet_aton(ip)’ return True if the IP address is valid and ‘socket.error’ will return error if the invalid IP address entered by user.
3. ‘ping_host(ip_address)’ : this function works the same with the previous script but it first calls the ‘is_valid_ip’ function to check wether the validaty the IP address. If error returned the ‘Invalid IP address’ print statement will be executed. Otherwise, the ping command will be executed.

RESULT

EXECUTION: Malicious input

127.0.0.1; ls -la

Output



```
root@LAPTOP-L2ABL36G:/home# python3 cominject_fix.py
Enter IP address to ping: 127.0.0.1; ls -la
Invalid IP address
```

Command injection solved!

CONCLUSION

the command injection vulnerability is identified in the python script and patched successfully. It is now including proper input validation and uses safe methods for command execution and automatically improve the security. In order to maintain the security and integrity regular code reviews and security testing are needed.

LIMITATIONS AND POSSIBLE SOLUTION

1) IP address validation
   - The function in the python script only checks if the format of the IP address. It does not check if the address is within a valid range on the network and whether it is reachable or not.
   - Solution: insert function that can verify if the IP address is exists in the range and reachable
2) No concurrency
   - The python script cannot support multiple IP addresses at once.
   - Solution: implement threading or asynchronous programming
3) Security of user input
   - More additional security prevention can be implemented.
   - Solution: implement additional security such as IP whitelisting or blacklisting