Yersinia Improvement Project

Adding functionality to an existing network switch penetration tool to expand it to work for non-Cisco switches

Group EH4:

Paul Oates 2001642

CMP311: Professional Project Development & Delivery

BSc Ethical Hacking

2022/23

Note that Information contained in this document is for educational purposes only.

Team Member	Section	Technical Contribution
Paul Oates	Methodology	Programming Researched ARP Flooding Researched RSTP Change the Root Bridge Setup the Ubuntu development environment Setup the GNS3 environment Implemented the RSTP network topology Wrote the RSTP Change the Root Bridge attack Assisted with the ARP Flooder

Table of Contents

Figure 1 - Flooder code

1.1 CREATION OF THE NETWORK FOR THE RSTP PROTOCOL

After consulting with the end client, David McLuskie and to meet the requirements of the project it was agreed to develop another exploit, an RSTP Change the Root Bridge Attack. The RSTP Change the Root Bridge Attack involves changing the root bridge on the network to the MAC address of the malicious computer, this can enable an attacker running this exploit to packet sniff for credentials, files, and configuration data. After further consultation with the end client, he stipulated that the exploit must be written in C and callable within Yersinia, it was also essential that the Readme file must include instructions on how to run the exploit.

To design and test the exploit GNS3 was utilised. This tool allowed us to emulate the network environment. An Extreme Networks EXOS switch was chosen due to it being free to use and well documented. A rough configuration of an RSTP network already existed on Extreme Networks GitHub and this was used as the basis of our RSTP topology design. (GitHub.com, June 2017)

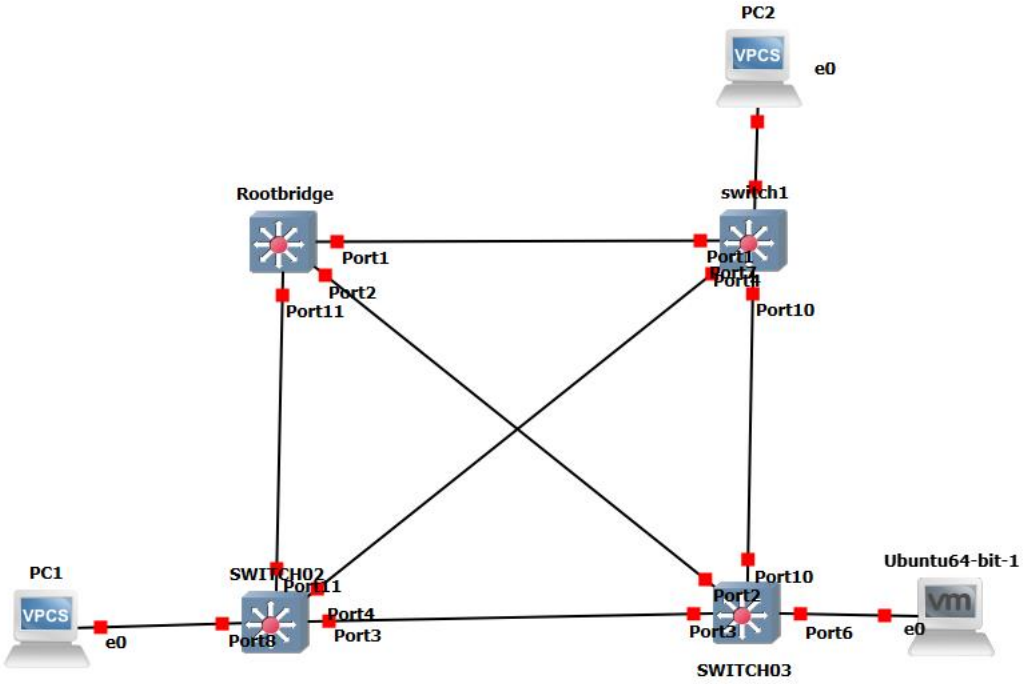

Figure 2 - RSTP Network

We began by downloading the four EXOS switch images and set up the root bridge as shown in Figure 6. We also added three PCs, two of which were virtual PCs and one exploit development VMware Ubuntu machine, this is where the exploit would be designed and initially ran. The two virtual PCs were setup on the 192.168.2.0/24 subnet and were pinged in order to ensure the network was properly set up. The exploit development machine was connected to the internet and Python, Scapy, VScode, Libnet and Pcap were installed, GCC was already installed. These tools enabled me to write the C and Python exploits.

1.2 CREATION OF THE RSTP CHANGE THE ROOT BRIDGE ATTACK

I then reviewed the ARP Flooder in both C and Python to understand how they had been implemented. I recognised that the ARP Flooder was targeting the broadcast address in order to get the packet broadcast across the network. Initially using the "Scapy" library in Python a RSTP Change the Root Bridge exploit was written and executed. The exploit attacked the multicast address and sets the source address to the root bridge.

```
frame = Dot3(src=args.mac, dst=Multicast)
llc_layer = LLC(dsap=0x042, ssap=0x042, ctrl=3)
stp_layer = STP(version=0x02, bpduflags=0x03c, rootid=0x01, rootmac=args.mac, pathcost=0x01, bridgeid=0x01, bridgemac=args.mac, portid=1)
```

Figure 3 - Packet assembly

The packet was constructed in three parts as shown in Figure 7:

- The frame consisted of the source MAC address and the destination multicast address
- The Logical Link Control (LLC) layer set up to use a TCP connection
- The STP layer which sets the root ID, bridge ID and path cost all set to 0x01. The root bridge is set to the source MAC address

Due to the packets low path cost and continual broadcast across the network, it triggers a root bridge re-election. The was relayed with a delay of two seconds, in order to not flood the network or trigger any preventions that the EXOS switch may have.

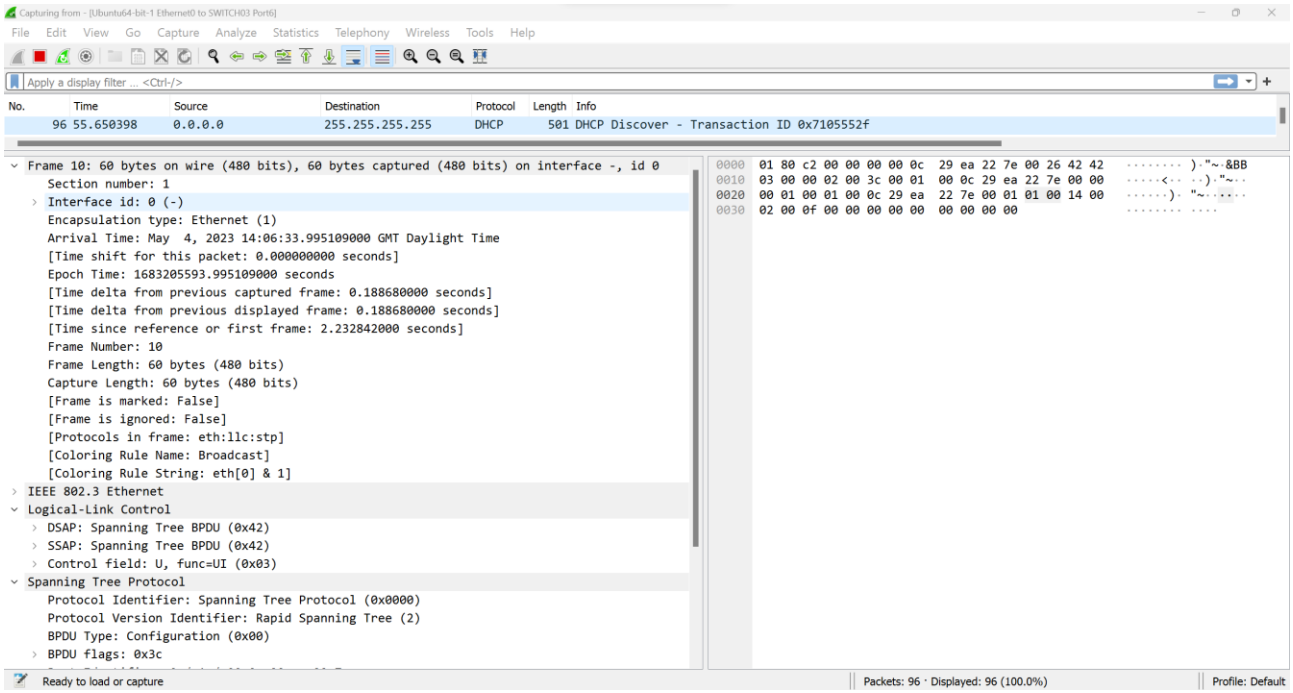

Figure 4 - Hex code

To write this exploit in C, the hex that the Python generated shown in Figure 8 was reformatted into a C char array. This enables the packet to be sent in C. This was then modified so it was possible to change the source MAC, root bridge and destination address from the command line.

```
// create a raw socket
int sockfd = socket(AF_PACKET, SOCK_RAW, htons(ETH_P_ALL));


// set  interface
struct sockaddr_ll destaddr = {0,};
destaddr.sll_family = AF_PACKET;
destaddr.sll_ifindex = if_nametoindex(interface);
destaddr.sll_halen = 6;
memcpy(destaddr.sll_addr, mac, 6);

// send  packet
for (;;) {
    ssize_t sent = sendto(sockfd, packet, sizeof(packet), 0, (struct sockaddr*)&destaddr, sizeof(destaddr));
    if (sent < 0) {
        perror("sendto");
        exit(1);
    }

    printf("Packet sent");
    //sleep for two secodns to avoid flooding
    sleep(2);


}
```

Figure 5 - RSTP Code Snippet

Figure 9 identifies the socket assembly ,including the interface and MAC address that the end user passes through to the exploit. This enables the exploit to send the packet across the network. Using the network interface the program sends the packet every two seconds in order not to trigger any protections.

The client was then consulted to demonstrate the RSTP Change the Root Bridge Attack. The client was pleased and stated that the programming requirements for the project had been met.

Next the exploit code was edited in order to embed this within Yersinia. To make the exploit callable within Yersinia a switch of "-EH4STP" was added. This was then committed to our GitHub fork of Yersinia and documented in our Readme file as shown in Appendix C – README file.

1.3 Testing for Key Performance Indicators (KPI)

Testing was carried out throughout the development of the two exploits, the ARP Flooder and RSTP Change the Root Bridge. Initially the Key Performance Indicators (KPI) below were identified:

ARP Flooder

- Time taken to flood the network:
  This was identified as a KPI as it is vital that the program can quickly flood a network. Under five seconds is a good target as any longer an Intrusion Prevention System would likely detect the malicious packets
- Time taken for recovery:
  This was identified as a KPI as the longer it takes to recover from the ARP Flooder the more effective the exploit
- Extent of network damage:
  This was identified as a KPI as the more devices, VLAN's and subnets the ARP Flooder achieves to impact the more successful the exploit has been

RSTP Attack

- Time taken to disrupt the network:
  This was identified as a KPI as the RSTP attack has to quickly assign the root bridge in order to not raise suspicion or cause the network to crash
- Time taken for recovery:
  This was identified as a KPI as the quicker the network recovers from an RSTP attack the less likely it is to be detected by an Intrusion Prevention System
- Total disruption:
  This was identified as a KPI as the more switches the exploit can change the root bridge on, whilst minimising disruption to the network, the more effective the attack

The exploits KPI were discussed among the programmers and then the wider group in order to ensure the exploits were thoroughly tested. In order for the ARP Flooder to be successful it has to be as loud and disruptive as possible while the RSTP attack has to be a quiet as possible. The environment used in development will also be used for final testing.

1.4 SUMMARY

Throughout this process there was a bi-weekly consultation with the client David McLuskie. This enabled the group to customise the exploits that David required. The iterative (evolutionary) methodology that we followed enabled us to benefit from continual feedback from the client. Currently there are two exploits in our forked version of Yersinia, a RSTP Change the Root Bridge Attack and an ARP Flooder. We successfully managed to implement the exploits within the timeframe and without any cost overruns. A Readme was also included to achieve our project delivery requirements. The exploits can be found on our forked Yersinia repository:

https://github.com/ZYXMoodester123/yersinia