



Level 0x0b

Fingerd, Inetd, and always learning new stuff...

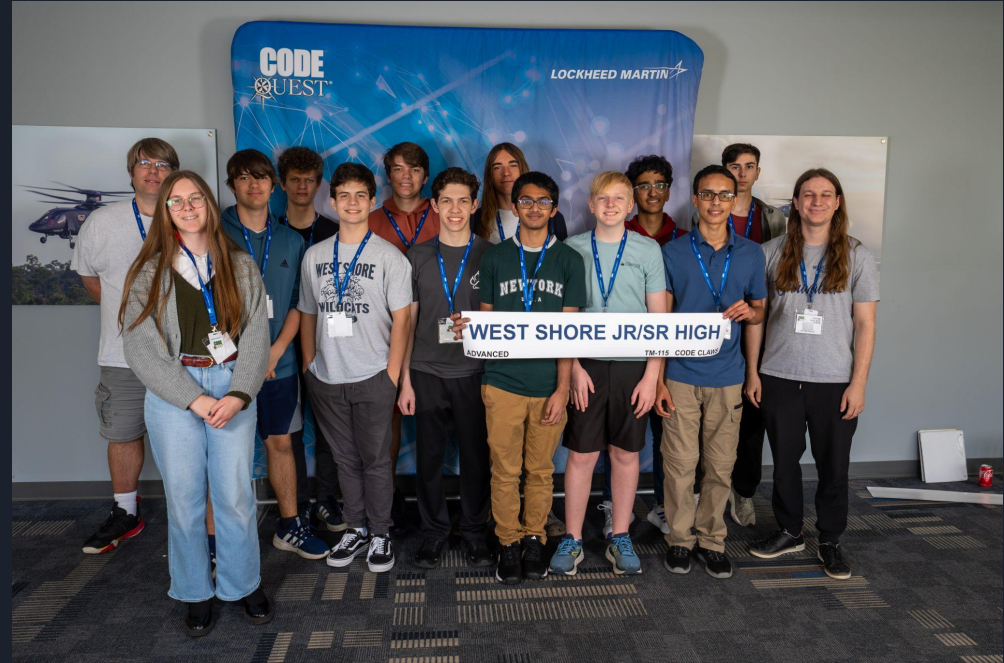


Topics

- Events
- Hacker History
- Internetz

Cyber Quest and Code Quest

- Code Quest
 - Saturday, Feb 28th
 - Registration Nov 17th -
- Spring Break
 - March 23rd - 27th
- Cyber Quest
 - Saturday, March 28th
 - Registration Jan 5th -





Code Quest

LLM Ninjas	Team 2 Last Names
Parker W Zachary W Charles L	Nathan d W Ian B P Anisha C
Wildcat New Blood	Wildcat Interns
Jasper B Kirin R Hope R	Sam S Eshan V Jay J

Code Quest

- Details
 - Saturday, February 28th, 9:00 - 1:30 ish
 - Teams are 2-3 students each, 4 teams max per school
 - 1 computer / laptop per team (one keyboard and one mouse too 😊)
 - No cell phones inside, no cameras (not even outside)
 - No buses, must provide your own transportation (they can't stay on site during the competition)
 - Breakfast and Lunch provided
- What you need to provide me
 - Team members and teams (team names)
 - Birthday (must be 11 yrs and older, middle-school officially allowed)
 - Citizenship (not related to ICE current politics, standard procedure for military contractors)
 - Return 2 Lockheed forms (Liability and Photo Release)
 - Return 1 school permission slip



Cyber Quest

- Details:
 - Saturday, March 28th, 9:00 - 1:30 ish
 - Teams are 3-5 students each, 4 teams max per school
 - Very large monitors discouraged
 - No cell phones inside, no cameras (not even outside)
 - No buses, must provide your own transportation (they can't stay on site during the competition)
 - Breakfast and Lunch provided
- What you need to provide me
 - Team members and teams (team names)
 - Birthday (must be 14 yrs and older)
 - Citizenship (not related to ICE current politics, standard procedure for military contractors)
 - Return 2 Lockheed forms (Liability and Photo Release)
 - Return 1 school permission slip



Bill Joy

- Wrote ex and vi text editors
 - Get triggered nano crew
- C shell
- Replaced AT&T TCP/IP code with the BSD TCP/IP code
 - Unix source distribution was hampered by AT&T licensing
 - Became the defacto standard for *nix network code
- Co-founded Sun Microsystems
 - Creators of that Java language some of you familiar with





Brief Networking Primer

- Addresses of computers
 - Ethernet / MAC : 01:02:03:ab:cd:ef
 - Set by manufacturer
 - Used at local level / Local-Area-Network (LAN)
 - ARP (Address Resolution Protocol) is used to convert IP addr to MAC addr
 - Internet Address (IPv4) Example: 192.168.0.101
 - 4 billion available, most have been allocated
 - Domain names (like [google.com](https://www.google.com)) can be converted to IP addr by DNS (Domain Name Service)
 - Port Number (kinda like a PO Box for your computer)
 - Ports 0 - 65535
 - Ports 0 - 1023 are the well-known ports (your OS will require higher privilege to listen on)
 - Ephemeral ports 32,000 - 60,000, assigned by OS when program doesn't need to specify a specific port (just give me one not in use already)



Internet Protocols

- UDP: User Datagram Protocol
 - Connectionless: no connection required, you just send 1 message at a time to address and port
 - Unreliable: may or may not get there, they get dropped sometimes...
- TCP: Transmission Control Protocol
 - Connection required: 3-way handshake to start using
 - Guarantees:
 - Delivery: Messages get acknowledged, may be retransmitted
 - Order: Your program gets them in the correct order, sequence numbering
 - Congestion avoidance: Will slow down when messages are failing
- No security offered, must encrypt yourself (TLS/SSL sockets)
- TCP and UDP have the same concept of port numbers, but not all protocols do
- Other protocols: ICMP, IGMP, RIP



Typical TCP Communication

- Server application:
 - Listens on a pre-determined port number (HTTP is usually port 80)
 - Waits for clients to connect
 - Client application
 - Picks an interface
 - Asks OS for ephemeral port to use
 - Connects to server
 - Connection handshake
 - Both sides can now send and receive, until one side closes connection
-
- Blocking / Non-blocking:
 - Do we wait for send operations (could be a while if congestion)
 - Do we wait when trying to read for data

Experimenting with Networking

- Netcat lets you do lots of things (**nc**)
 - Use **-l** to create a listener (lowercase L)
 - Use another netcat to connect and send data

TCP Example:

- 3 packets to start connection
- 4th packet is actual user data
- 5th packet is ACK
- Packets 6-8 to close connection

The screenshot displays two windows side-by-side. The left window is Wireshark, titled '*Loopback: lo (as superuser)', showing a packet capture on the loopback interface. The packet list shows eight packets, with packet 4 selected. The packet details pane shows the structure of packet 4: Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol (Seq: 52478, Dst Port: 12345). The packet bytes pane shows the raw data of packet 4, which is the text 'Hello port 12345'. The right window is a terminal titled 'user@ctf22: ~', showing the execution of a netcat listener on port 12345. The terminal output shows the connection being established and the received data 'Hello port 12345'.

```
*Loopback: lo (as superuser)
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
2 0.000014250 127.0.0.1 127.0.0.1 TCP 74 12345
3 0.000023360 127.0.0.1 127.0.0.1 TCP 66 52478
4 0.000296015 127.0.0.1 127.0.0.1 TCP 83 52478
5 0.000301385 127.0.0.1 127.0.0.1 TCP 66 12345
6 5.942185745 127.0.0.1 127.0.0.1 TCP 66 52478
7 5.942574032 127.0.0.1 127.0.0.1 TCP 66 12345
8 5.942589222 127.0.0.1 127.0.0.1 TCP 66 52478

Frame 4: 83 bytes on wire (664 bits), 83 bytes captured (664 bits) on interface lo, interface 0
Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 52478, Dst Port: 12345, Seq: 1, Ack: 1, Len: 17
Data (17 bytes)
0000 00 00 00 00 00 00 00 00 00 00 00 00 00 45 00 .....E..
0010 00 45 c9 1e 40 00 00 06 73 92 7f 00 00 01 7f 00 ..E..@..s....
0020 00 01 cc fe 30 39 d4 c2 82 aa bd 43 94 76 80 18 ...09...C.v..
0030 02 00 fe 39 00 00 01 01 08 0a aa f2 26 f1 aa f2 ...9...&...
0040 26 f0 48 65 6c 6c 6f 20 70 6f 72 74 20 31 32 33 &Hello port 123
0050 34 35 0a .....45..

wireshark_loDTBSJ3.pcapng Packets: 8 - Displayed: 8 (100.0%) - Dropped: 0 (0.0%) Profile: Default
```

```
user@ctf22: ~
File Edit View Search Terminal Help
user@ctf22:~$ nc -l 12345
Hello port 12345
user@ctf22:~$

user@ctf22:~$ echo "Hello port 12345" | nc 127.0.0.1 12345
AC
user@ctf22:~$

[1] 0: bash* 1: sudo "ctf22" 01:43 23-Jan-20
```

Sending UDP with netcat

- Netcat lets you do lots of things (**nc**)
 - Use **-u** to use UDP protocol instead of TCP (the default for netcat)
 - Use another netcat to connect and send data (use **-u** here too!)
- Wireshark for viewing traffic

UDP Example:

- 1 packet, so simple!

127.0.0.1 is localhost IP
address (send it to myself)

The image displays two side-by-side screenshots from a Linux terminal and Wireshark network analyzer.

The left screenshot shows a terminal window titled "user@ctf22: ~". It contains the following commands and output:

```
user@ctf22:~$ nc -l -u 12345
Hello UDP

user@ctf22:~$ echo "Hello UDP" | nc -u 127.0.0.1 12345
```

The right screenshot shows the Wireshark network analyzer interface, titled "Capturing from Loopback: lo (as superuser)". It displays a single captured packet:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	UDP	52	40988 → 12345

Below the packet list, the packet details pane shows the structure of the captured packet:

- Frame 1: 52 bytes on wire (416 bits), 52 bytes captured (416 bits) on interface lo, interface 0
- Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
- Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
- User Datagram Protocol, Src Port: 40988, Dst Port: 12345
- Data (10 bytes)

The packet bytes pane shows the raw data in hexadecimal and ASCII:

```
0000  00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00  .....E..
0010  00 26 e7 ee 40 00 40 11 54 d6 7f 00 00 01 7f 00  .&. @ @ T.....
0020  00 01 a0 1c 30 39 00 12 fe 25 48 65 6c 6c 6f 20  .01 a0 1c 30 39 00 12 fe 25 48 65 6c 6c 6f 20
0030  55 44 50 0a                                         .UHello
                                         UDP..
```



Asking Chat GPT for a fun historical bug

- Chat GPT suggests the Morris Worm bug in service fingerd
- What is fingerd?
- How can I try out this bug / run an old version of fingerd?
- Can I cause an affect or crash?

Fingerd

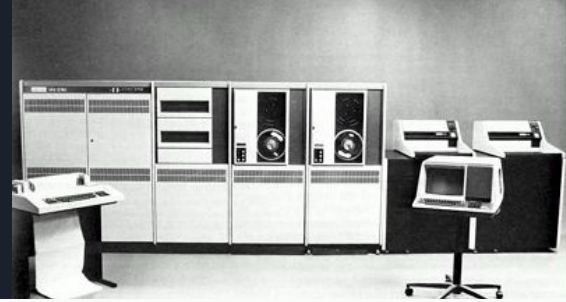
- Give me info about a user on a system
 - What is their name?
 - What is their email?
 - What are they working on (.plan file)
- Listens on TCP port 79
- From an era of computing
 - We are barely concerned about security and privacy
 - Computers are used by academia
 - No privacy needed
- *nix Daemon
 - Background process
 - Finger Daemon
 - Fingerd, sshd, httpd, etc
 - Windows service



```
user@ctf22:~$ finger root@127.0.0.1
Login name: root                      In real life: Charlie Root
Directory: /                          Shell: /bin/csh
On since Jan 21 21:45:40 on console    23 hours Idle Time
No Plan.
user@ctf22:~$ echo "root" | nc 127.0.0.1 79
Login name: root                      In real life: Charlie Root
Directory: /                          Shell: /bin/csh
On since Jan 21 21:45:40 on console    1 day 0 hours Idle Time
No Plan.
user@ctf22:~$
```

What did this fingerd app run on?

- VAX-11/780
 - First of the VAX computer family
 - DEC follow-on to the PDP-11
- Unix created on PDP-7 and PDP-11
- Runs BSD Unix 4.3
 - 1986 vintage...
- Emulated via project called SIMH
- Applications
 - ~400 applications
 - 66 files in /bin for BSD 4.3
 - 3400 files in /usr/bin on Ubuntu
- We have: f77, cc, vi, sum
- No nano, neofetch, zip, md5sum

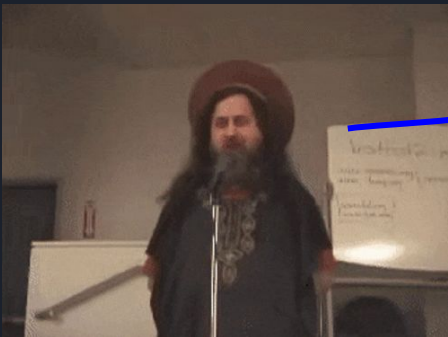


```
simh# ps aux
USER      PID %CPU %MEM    SZ   RSS TT  STAT   TIME COMMAND
root       45  0.0  0.7   55    44 co  I      0:00 /etc/syslogd
root       96  0.0  0.3   30    14 03  I      0:00 - std.9600 tty03 (getty)
root       82  0.0  0.5   59    30 ?   I      0:00 /etc/inetd
root       94  0.0  0.3   30    14 01  I      0:00 - std.9600 tty01 (getty)
root       95  0.0  0.3   30    14 02  I      0:00 - std.9600 tty02 (getty)
root       87  0.0  1.0   80    67 ?   I      0:00 /usr/lib/lpd
root       78  0.0  0.8   49    47 ?   I      0:00 /etc/rwhod
root       70  0.0  0.5   40    27 ?   I      0:00 /etc/cron
root       67  0.0  0.1    5     3 ?   S      0:00 /etc/update
root       97  0.0  0.3   30    14 04  I      0:00 - std.9600 tty04 (getty)
root       57  0.0  2.5  177   169 ?   I      0:00 /usr/lib/sendmail -bd -q30
root       93  0.0  0.3   30    14 00  I      0:00 - std.9600 tty00 (getty)
root       92  0.0  1.8  142   118 co  S      0:00 -csh (csh)
root        2  0.0  0.3 2048    0 ?   D      0:00 pagedaemon
root        1  0.0  0.5   36    28 ?   I      0:00 init
root        0  0.0  0.1    0     0 ?   D      0:00 swapper
root       767  0.0  0.5   59    32 co  I      0:00 inetd
root       790  0.0  1.3  109    83 co  R      0:00 ps aux
root      100  0.0  0.3   30    14 07  I      0:00 - std.9600 tty07 (getty)
root       99  0.0  0.3   30    14 06  I      0:00 - std.9600 tty06 (getty)
root       98  0.0  0.3   30    14 05  I      0:00 - std.9600 tty05 (getty)
simh#
```

Where is fingerd on this system

- I don't see fingerd running...
- Where is it, run find...

```
simh# find / | grep fingerd
/etc/fingerd
/usr/man/man8/fingerd.8c
/usr/src/etc/fingerd.c
simh#
```



```
main(argc, argv)
char *argv[];
{
    register char *sp;
    char line[512];
    struct sockaddr_in sin;
    int i, p[2], pid, status;
    FILE *fp;
    char *av[4];

    i = sizeof(sin);
    if (getpeername(0, &sin, &i) < 0)
        fatal(argv[0], "getpeername");

    line[0] = '\0';
    gets(line);
    sp = line;
    av[0] = "finger";
    i = 1;
    while (1) {
        while (isspace(*sp))
            sp++;
        if (!*sp)
            break;
        if (*sp == '/' && (sp[1] == 'W' || sp[1] == 'w')) {
            sp += 2;
            av[i++] = "-l";
        }
        if (*sp && !isspace(*sp)) {
            av[i++] = sp;
            while (*sp && !isspace(*sp))
                sp++;
            *sp = '\0';
        }
    }
    av[i] = 0;
    if (pipe(p) < 0)
        fatal(argv[0], "pipe");
    if ((pid = fork()) == 0) {
        close(p[0]);
        if (p[1] != 1) {
            dup2(p[1], 1);
        }
    }
}
```


BSD Unix came with source code

- The source was right there...
- Compilation was as simple as helloworld
- I suddenly understood why Richard Stallman is the way he is...

```
simh# cc /usr/src/etc/fingerd.c
simh# ls -l /etc/fingerd
-rwxr-xr-x  1 root          11264 Jun  6  1986 /etc/fingerd*
simh# ls -l a.out
-rwxrwxr-x  1 root          14336 Jan 23 00:28 a.out*
simh# strip a.out
simh# ls -l a.out
-rwxrwxr-x  1 root          11264 Jan 23 00:28 a.out*
simh#
```



Where is the network code...

```
main(argc, argv)
char *argv[];
{
    register char *sp;
    char line[512];
    struct sockaddr_in sin;
    int i, p[2], pid, status;
    FILE *fp;
    char *av[4];

    i = sizeof(sin);
    if (getpeername(0, &sin, &i) < 0)
        fatal(argv[0], "getpeername");
    line[0] = '\0';
    gets(line);
    sp = line;
    av[0] = "finger";
    i = 1;
    while (1) {
        while (isspace(*sp))
            sp++;
        if (!*sp)
            break;
        if (*sp == '/' && (sp[1] == 'W' || sp[1] == 'w')) {
            sp += 2;
            av[i++] = "-l";
        }
        if (*sp && !isspace(*sp)) {
            av[i++] = sp;
            while (*sp && !isspace(*sp))
                sp++;
            *sp = '\0';
        }
    }
    av[i] = 0;
    if (pipe(p) < 0)
        fatal(argv[0], "pipe");
    if ((pid = fork()) == 0) {
        close(p[0]);
        if (p[1] != 1) {
            dup2(p[1], 1);

```

```
simh# cat /etc/inetd.conf; echo " "; cat /tmp/header
#
# Internet server configuration database
#
ftp      stream  tcp      nowait  root    /etc/ftpd      ftpd
telnet   stream  tcp      nowait  root    /etc/telnetd   telnetd
shell    stream  tcp      nowait  root    /etc/rshd      rshd
login    stream  tcp      nowait  root    /etc/rlogind   rlogind
exec     stream  tcp      nowait  root    /etc/rexecd    rexecd
# Run as user "uucp" if you don't want uucpd's wtmp entries.
#uucp    stream  tcp      nowait  root    /etc/uucpd     uucpd
finger   stream  tcp      nowait  nobody  /etc/fingerd   fingerd
#tftp    dgram   udp      wait    nobody  /etc/tftpd     tftpd
comsat   dgram   udp      wait    root    /etc/comsat    comsat
talk     dgram   udp      wait    root    /etc/talkd     talkd
ntalk    dgram   udp      wait    root    /etc/ntalkd    ntalkd
echo     stream  tcp      nowait  root    internal
discard  stream  tcp      nowait  root    internal
chargen  stream  tcp      nowait  root    internal
daytime  stream  tcp      nowait  root    internal
time     stream  tcp      nowait  root    internal
echo     dgram   udp      wait    root    internal
discard  dgram   udp      wait    root    internal
chargen  dgram   udp      wait    root    internal
daytime  dgram   udp      wait    root    internal
time     dgram   udp      wait    root    internal

port     type      proto
simh#
```



inetd

- Internet super-server
 - Listens on all the ports listed in `inetd.conf`
 - Runs a program when connection occurs
-
- 1 daemon running instead of 10-20
 - Save CPU and memory overhead
 - Daemon programs don't need network code
 - Read from `stdin`
 - Write to `stdout`

Where is the bug then?

```
main(argc, argv)
char *argv[];
{
    register char *sp;
    char line[512];
    struct sockaddr_in sin;
    int i, p[2], pid, status;
    FILE *fp;
    char *av[4];

    i = sizeof(sin);
    if (getpeername(0, &sin, &i) < 0)
        fatal(argv[0], "getpeername");
    line[0] = '\0';
    gets(line);
    sp = line;
    av[0] = "finger";
    i = 1;
    while (1) {
        while (isspace(*sp))
            sp++;
        if (!*sp)
            break;
        if (*sp == '/' && (sp[1] == 'W' || sp[1] == 'w')) {
            sp += 2;
            av[i++] = "-l";
        }
        if (*sp && !isspace(*sp)) {
            av[i++] = sp;
            while (*sp && !isspace(*sp))
                sp++;
            *sp = '\0';
        }
    }
    av[i] = 0;
    if (pipe(p) < 0)
        fatal(argv[0], "pipe");
    if ((pid = fork()) == 0) {
        close(p[0]);
        if (p[1] != 1) {
            dup2(p[1], 1);
        }
    }
}
```

GETS(3S) UNIX Programmer's Manual GETS(3S)

NAME
gets, fgets - get a string from a stream

SYNOPSIS
#include <stdio.h>

char *gets(s)
char *s;

char *fgets(s, n, stream)
char *s;
FILE *stream;

DESCRIPTION
Gets reads a string into s from the standard input stream stdin. The string is terminated by a newline character, which is replaced in s by a null character. Gets returns its argument.

GETS(3) Linux Programmer's Manual GETS(3)

NAME
gets - get a string from standard input (DEPRECATED)

SYNOPSIS
#include <stdio.h>

char *gets(char *s);

DESCRIPTION
Never use this function.

gets() reads a line from stdin into the buffer pointed to by s until either a terminating newline or EOF, which it replaces with a null byte ('\0'). No check for buffer overrun is performed (see BUGS below).

gets() vulnerable to buffer overflow EVERY TIME IT'S USED



Exploiting the bug

- No python to script, have to write it in C
- No way to see crash, unless just run it myself...

Don't have time or skills to write the VAX shellcode to exploit this for real like Robert Morris, so we just crash it

```
simh# cat test.c
#include <stdio.h>

void main()
{
    int i;
    for( i = 0; i < 700; i += 1)
    {
        printf("a");
    }
}

simh# cc test.c
simh# echo `a.out` | fingerd
Illegal instruction (core dumped)
simh#
```

You can still use inetd today

- But don't...
- Xinetd replaced inetd
 - DDOS / rate-limiting of connections
 - More configurable
- Now replaced with
 - Systemd - Linux startup and service control system
 - Containers (docker)

```
user@ctf22:~/scratch$ head -n 22 /etc/inetd.conf
# /etc/inetd.conf: see inetd(8) for further informations.
#
# Internet superserver configuration database.
#
#
# Lines starting with "#:LABEL:" or "#<off>#" should not
# be changed unless you know what you are doing!
#
# If you want to disable an entry so it is not touched during
# package updates just comment it out with a single '#' character.
#
# Packages should modify this file by using update-inetd(8).
#
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
#
#:INTERNAL: Internal services
#discard          stream  tcp6   nowait  root    internal
#discard          dgram   udp6   wait    root    internal
#daytime          stream  tcp6   nowait  root    internal
#time             stream  tcp6   nowait  root    internal
9999              stream  tcp4   nowait  user    /home/user/scratch/a.out

user@ctf22:~/scratch$ cat stupid.c
#include <stdio.h>

int main(int argc, char** argv)
{
    char data[512];
    gets(data);

    printf("%s is an idiot!\n", data);
    return 0;
}

user@ctf22:~/scratch$ echo "Michael" | nc 127.0.0.1 9999
Michael is an idiot!
user@ctf22:~/scratch$
```



Links

- <https://github.com/simh/simh>
- <https://github.com/mwales/ye-olde-bsd>
-