Added - Exploits

- Monitoring and Sniffing
- Sniffing Tools
- Buffer Overflow
- Denial of Service (DoS)

Monitoring/Sniffing

- LAN: broadcast-based transmission
- NIC
  - it own MAC address
  - Broadcast MAC address
- NIC in promiscuous mode :
  - can receive all frames
- Listening to all messages received by the card as an administrator is called monitoring
- Listening to all messages received by the card in violation of your user agreement is sniffing
- Recording instances of a daemon's execution is called logging

Sniffing Tools

- Most tools are based on TCPdump and the libpcap (packet capture library) extensions uses promiscuous mode
- Other tools are
  - libnet library
  - snort and dsniff
  - for windows there are many, windump and winpcap is a direct port of TCPdump and libpcap

Sniffing Attacks

- password-based remote access (telnet) is vulnerable to sniffing
- One defense is to use an ssh instead of telnet or rsh
- With secure shell (ssh), the two computers negotiate a symmetric key for use during the session
  - Key exchange is achieved using public key cryptosystem
  - But there is no trusted/certificate authority, so the initial exchange of keys is subject to a man-in-the-middle attack
- dsniff can be used to look for the initial exchange

Using TCPdump

- List interfaces on a unix machine:

  ifconfig -a

```
eth0    Link encap:Ethernet  HWaddr 00:C0:4F:A1:46:0E
          inet addr:128.119.245.66  Bcast:128.119.247.255  Mask:255.255.248.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:344734848 errors:0 dropped:0 overruns:33473 frame:0
          TX packets:251651044 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:19 Base address:0xdc00

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:3924  Metric:1
          RX packets:3756831 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3756831 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
```

TCPdump

tcpdump -i eth0 tcp

```
08:23:10.367677 gslacks.cs.umass.edu.1072 > horton.cse.ucsc.edu.22:
   P 18977 31:1897751(20) ack 2681372961 win 8168 <nop,nop,timestamp
   19745 653063287> (DF) [tos 0x5]
08:23:10.532319 horton.cse.ucsc.edu.22 > gslacks.cs.umass.edu.1072:
   P 1:45(44) ack 20 win 24616 <nop,nop,timestamp 653083760 19745>
   (DF)
08:23:10.638393 gslacks.cs.umass.edu.1072 > horton.cse.ucsc.edu.22: .
   ack 4 5 win 8124 <nop,nop,timestamp 19748 653083760> (DF) [tos
   0x5]
```

Other calls

tcpdump -i eth0 src biff.cs.umass.edu and tcp
tcpdump -i eth0 not dest blinky.cs.umass.edu and arp

Snort

- Snort
  - a lightweight network intrusion detection system, capable of performing real-time traffic analysis and packet logging on IP networks
- Three primary uses
  - Packet sniffer
  - Packet logger
  - Intrusion Detection System
- http://www.snort.org

Snort

- Features
  - Full tcpdump compatibility – Snort can create tcpdump data files
  - Full support for bpf (Berkeley packet filter) rules
  - Simple rules language

Snort: Sample packet capture output

```
Initializing Network Interface...
Decoding Ethernet on interface hme0
-*> Snort! <*-Version 1.6By Martin Roesch (roesch@clark.net, www.clark.net/~roesch)
11/09-10:42:23.760220 128.119.175.17:41745 -> 128.119.166.5:21
TCP TTL:255 TOS:0x0 ID:61543  DF
**S***** Seq: 0x36A3E59   Ack: 0x0   Win: 0x2238 TCP Options => MSS: 1460
11/09-10:42:23.761102 128.119.166.5:21 -> 128.119.175.17:41745
TCP TTL:59 TOS:0x0 ID:3606  DF
**S***A* Seq: 0xD9DB1A5   Ack: 0x36A3E5A   Win: 0x832CTCP Options => MSS: 1460 00 00
11/09-10:42:23.761133 128.119.175.17:41745 -> 128.119.166.5:21
TCP TTL:255 TOS:0x0 ID:61544  DF
******A* Seq: 0x36A3E5A   Ack: 0xD9DB1A6   Win: 0x2238
11/09-10:42:23.829616 128.119.166.5:21 -> 128.119.175.17:41745
TCP TTL:59 TOS:0x10 ID:3620  DF
*****PA* Seq: 0xD9DB1A6   Ack: 0x36A3E5A   Win: 0x832C
        32 32 30 20 65 6D 69 6C 79 2E 6F 69 74 2E 75 6D    220 emily.oit.um
        61 73 73 2E 65 64 75 20 46 54 50 20 73 65 72 76    ass.edu FTP serv
        65 72 20 28 56 65 72 73 69 6F 6E 20 77 75 2D 32    er (Version wu-2
        2E 34 2E 32 2D 61 63 61 64 65 6D 5B 42 45 54 41    .4.2-academ[BETA
        2D 31 38 5D 28 31 29 20 54 68 75 20 46 65 62 20    -18](1) Thu Feb
        31 31 20 30 38 3A 31 34 3A 32 38 20 45 53 54 20    11 08:14:28 EST
        31 39 39 39 29 20 72 65 61 64 79 2E 0D 0A          1999) ready...
```

Homework #2

- Open source NIDSs
  - open source NIDS 시스템 조사
  - Snort tool 기능 조사

  - Submission file: HW2-학번-이름 (한글 또는 워드 파일)

Buffer Overflows

- Program buffer overflows are the most common form of security vulnerability
  - http://www.cert.org
- buffer overflow
  - Getting a SUID root program that can run an arbitrary command means you can login as root without knowing root's password
  - Get and run the root-privileged code, and insert a exploit code in the program address space
  - Get the program to jump to that code

SUID programs are dangerous

- On a computer logged on as a root:
  - cp /bin/sh /tmp/evil
  - chmod 04755 /tmp/evil

- SUID program: a program that set-uid bit is set
  - if the owner of the program is root, then that program is executed with root privileges
  - (e.g.) passwd program

Processes in memory

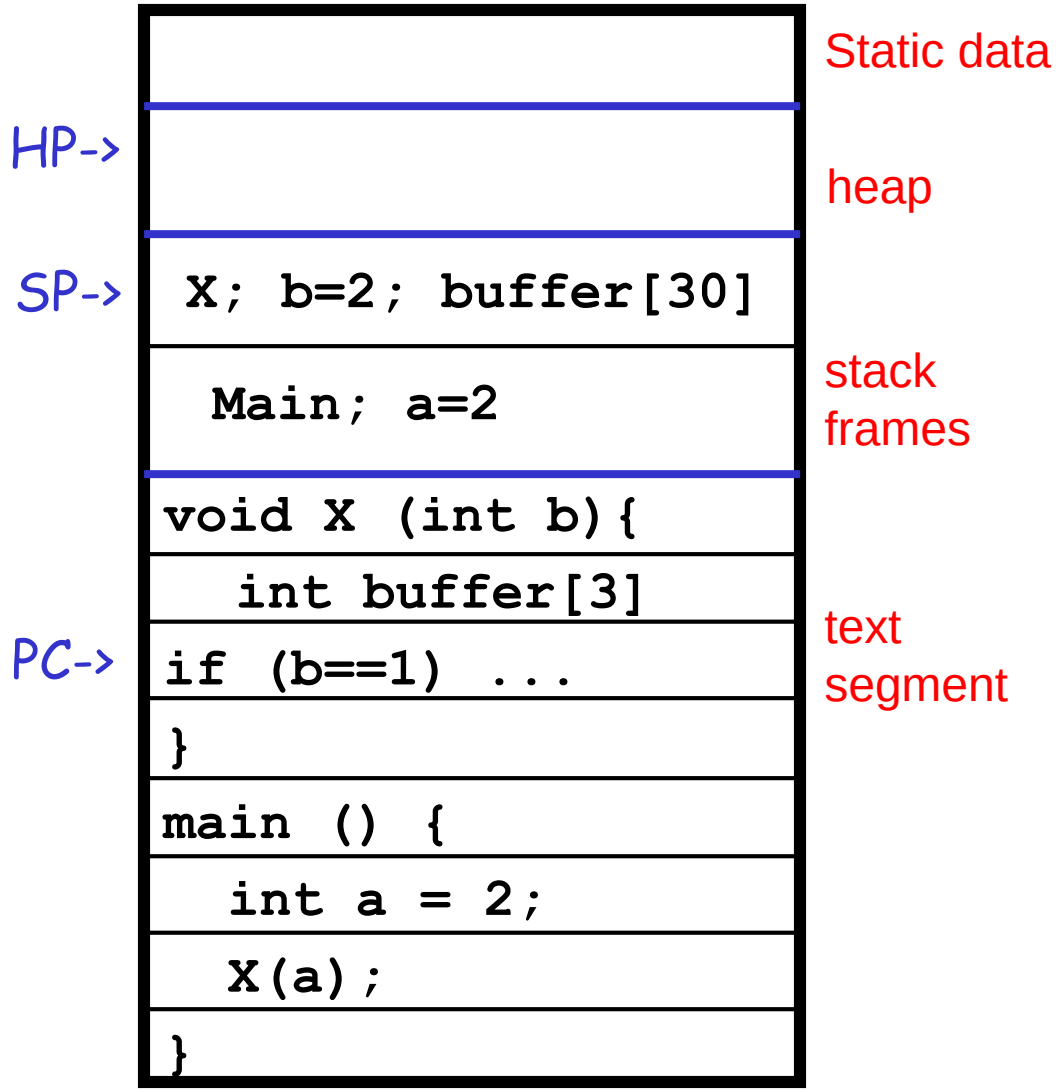
- Process state in memory consists of several items:
  - the code for running the program
  - the static data for the running program
  - space for dynamic data (the heap) and the heap pointer (hp)
  - the program counter (PC), indicating the next instruction to execute
  - an execution stack with the program's function call chain (the stack)
  - values of CPU registers
  - a set of OS resources in use; e.g., open files
  - process execution state (ready, running, waiting, etc)

Processes in Memory

- We need consider only four regions in memory:
  - static data: pre-allocation memory (int array[9];)
  - text: instructions and read-only data
  - heap: re-sizeable portion containing data malloc()'d and free()'d by the user
  - Stack: a push and pop data structure
    - Used to allocate local variables used in functions, pass variables, return values from function calls, and return address

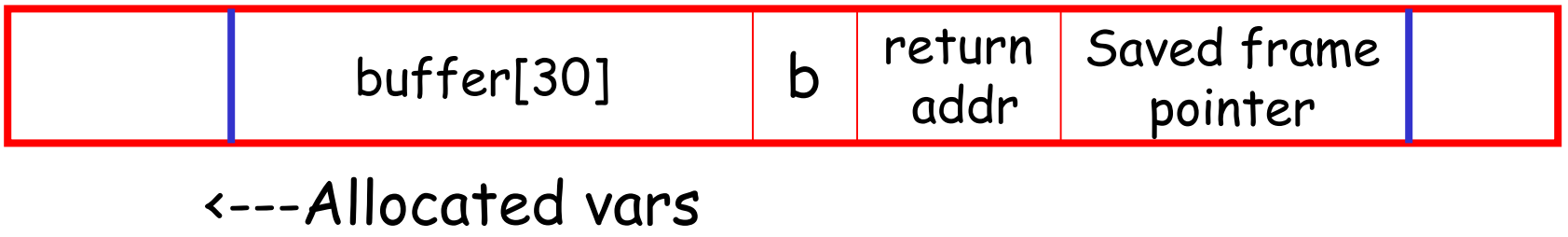
Process State

```
void X (int b) {
   int buffer[30];

   if (b==1) ...
}


main () {
   int a = 2;
   X(a);
}
```



Calling a function

- The stack consists of a logical stack of ***frames***
- Frames consist of parameters given to a function, local variables, return address, and previous frame pointer, etc.
- So in the previous example, (each frame in) the stack looks like this:



Segmentation fault

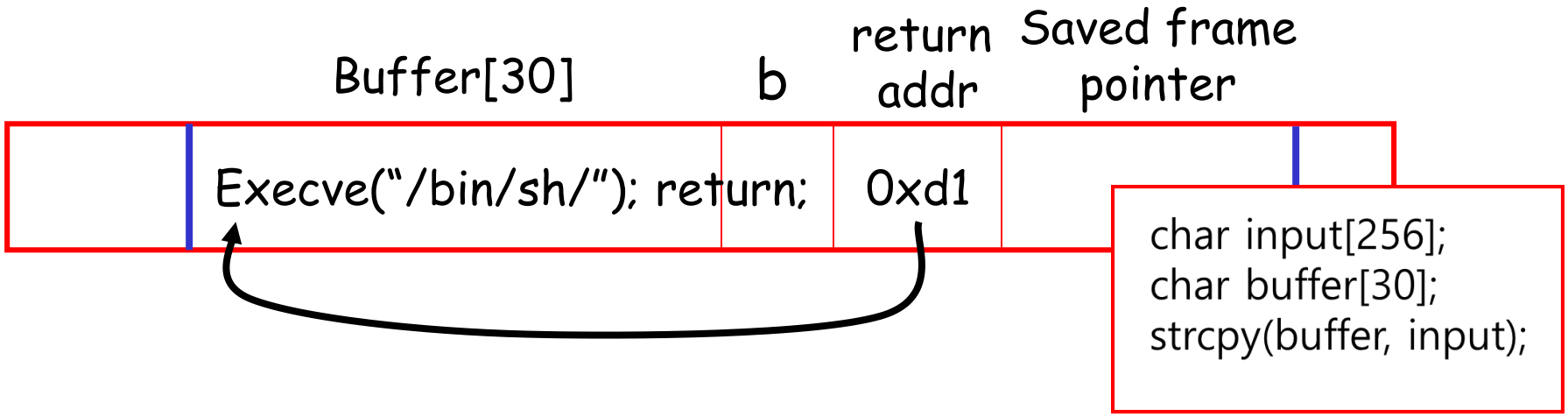
- Segmentation fault:
  - In memory, if you read data into a buffer, you might write over other variables necessary for program execution
  - Normally this results in a segmentation fault

```
char input[256];
char buffer[30];
strcpy(buffer, input);
```

Segmentation fault

- When you read in too many characters into a buffer, you can modify the rest of the stack, altering the flow of the program
- Normally, writing over array bounds causes a segmentation fault as you'll actually overwrite into other variables in the program
- If you are careful about what you overwrite, then you can alter what the program does next

Smashing the Stack



- If buffer[] gets its input from the command line, and the input is longer than the allocated memory, the program will write into the return address
- If you do it perfectly, you can write into the RA the memory location of your input
- When your function completes, it will execute next the first command in your input

Buffer overflow over the net: Morris Worm

- Fingerd takes input about whom to finger without checking input size
- Morris wrote the following code after the buffer overflow to create the Morris worm:

```
pushl $68732f '/sh₩0'
pushl $6e69622f '/bin'
movl sp,r10
pushl $0
pushl $0
pushl r10
pushl $3
movl sp,ap
chmk $3b
```

upon return to main()

execve("/bin/sh",0,0);

was executed, opening a shell on the remote machine

Defenses

- Use input functions carefully
  - Don't use strcpy(), strcat(), sprintf(), gets()
  - Use instead strncpy(3), strncat(3), snprintf(3), and fgets(3)
- Other problematic constructs:

  fscanf(3), scanf(3), vsprintf(3), realpath(3), getopt(3), getpass(3), streadd(3), strecpy(3), and strtrns(3)
- Reference: "Smashing the Stack" by Aleph One

Homework

- "Secure-Programs-HOWTO"
  - read and make summary the article "Secure-Programs-HOWTO"
  - Submission file: HW1-학번-이름 (한글 또는 워드 파일)

Denial Of Service

- The goal of a denial of service attack is to deny legitimate users access to a particular resource
- Types of attacks
  - Resource exhaustion (consume all bandwidth, CPU, or disk space)
  - Programming faults: uses faults in programs, OS, or embedded logic chips
  - Routing and DNS attacks: invalid routing table updates, DNS cache poisoning
- There are three general categories of attacks
  - Against users, against hosts, or against networks

Local DOS against hosts

- fork() bomb
- intentionally generate errors to fill logs, consuming disk space, crashing
- Countermeasures
  - partition disks
  - disk quotas
  - set process limits
  - monitor system activity/CPU/Disk Usage
  - Physical Security

Network Based Denial of Service Attacks

- UDP bombing
- tcp SYN flooding
- ping of death
- smurf attack

- Most involve either resource exhaustion or corruption of the operating system runtime environment

UDP bombing

- Uses two UDP services: echo (echoes back any char received) and chargen (generates chars)
- These services can be used to launch a DOS by connecting chargen (port 19) to echo ports (port 7) on the same or another machine and generating large amounts of network traffic
- Countermeasures:
  - Disable echo, chargen and all other unused services whenever possible: /etc/inetd.conf on Unix, and "no udp small-services" on Cisco IOS
  - Filter UDP traffic at the firewall level; only allow legitimate traffic such as UDP port 53 (DNS)

Windows UDP attacks

- NewTear, Newtear2, Bonk, and Boink are tools that exploit the weakness in the MS Windows 95/NT TCP/IP stack implementation
- The attacker sends the victim a pair of malformed IP fragments which get re-assembled into an invalid UDP datagram. Upon receiving the invalid datagram, the victim host “blue-screens” and freezes or reboots (The pathologic offset attack)
- Countermeasure: Apply vendor patches

TCP SYN Flooding

- To establish a legitimate TCP connection:
  - 3-way handshaking: SYN – SYN/ACK - ACK exchange
- TCP SYN flooding: TCP "half-open" attack
  - The attacker initiates a TCP connection to the server with a SYN
  - The server replies with a SYN-ACK
  - The client then doesn't send back an ACK
  - (If the client spoofed the initial source address, it will never receive the SYN-ACK)

TCP SYN Flooding: Results

- half-open connections queue on the victim server will eventually fill
  - the system will be unable to accept any new incoming connections until the buffer is emptied out
- Countermeasures
  - Increase the TCP connection queue (backlog)
  - Decrease connection establishment timeout
  - Use IDS
  - Install Ingress/Egress router filters to prevent some IP spoofing

Ping of Death

- The TCP/IP specification allows for a maximum packet size of 65,536 octets
- The ping of death attack sends oversized ICMP datagrams (encapsulated in IP packets) to the victim
- Some systems, upon receiving the oversized packet, will crash, freeze, or reboot, resulting in denial of service
- Countermeasures: most systems are now immune, but apply vendor patches if needed

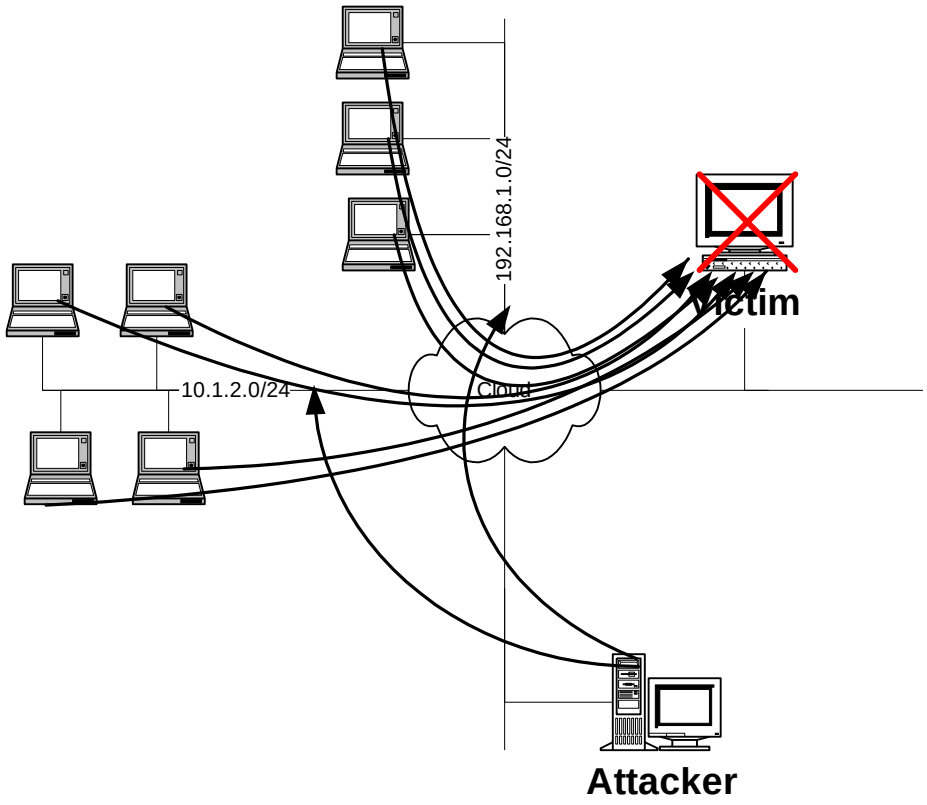
Smurf Attack

- A host sends an ICMP echo request (ping) to a network or broadcast address
  - Network address: the host portion is 0
  - Directed broadcast: the host portion having all **1**s
- Every host on the network receives the ICMP echo request and sends back an ICMP echo response inundating the initiator with network traffic

Smurf Attack

- There are 3 players in the smurf attack
  - attacker, amplification (intermediary) network, victim
- In most scenarios the attacker spoofs the IP source address as the IP of the intended victim to the intermediary network broadcast address
- Every host on the intermediary network replies, flooding the victim and the intermediary network with network traffic
- The intermediary network amplifies the traffic

Smurf Example



1. Attacker sends ICMP (echo request) packet with spoofed source IP

   Attacker→10.1.2.255

2. Victim is flooded with ICMP echo responses

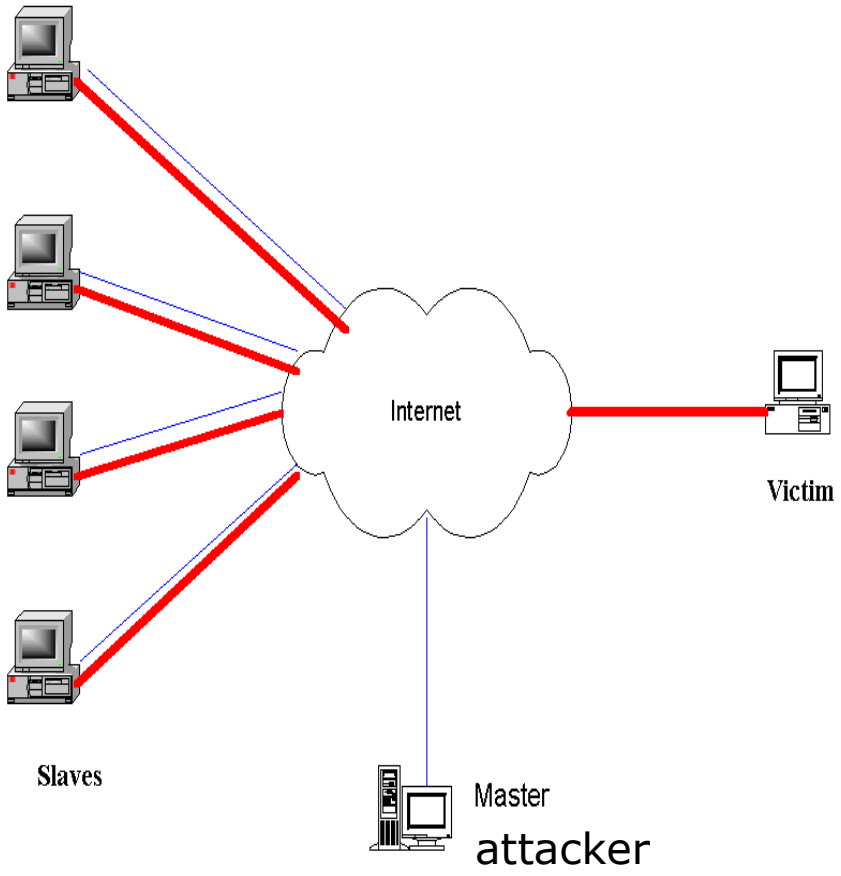
3. Victim hangs?

Smurf: Countermeasures

- Configure routers to deny IP broadcast traffic onto your network from other networks
- Configure hosts (via kernel variable) to NOT reply to a packet sent to a broadcast address
- Configure Ingress/Egress filters on routers to counteract IP address spoofing

Distributed Denial of Service Attacks (DDoS)

- Attacker runs multiple DoS attacks in parallel by installing remote agents on many machines on Internet
- Attacker logs into Master and signals slaves to launch an attack on a specific target address (victim)
- Slaves then respond by initiating TCP, UDP, ICMP or Smurf attack on victim



Distributed Denial of Service Attacks (DDoS)

- trin00 (WinTrinoo)
- Tribe Flood Network (TFN) (TFN2k)
- Shaft
- stacheldraht
- mStream