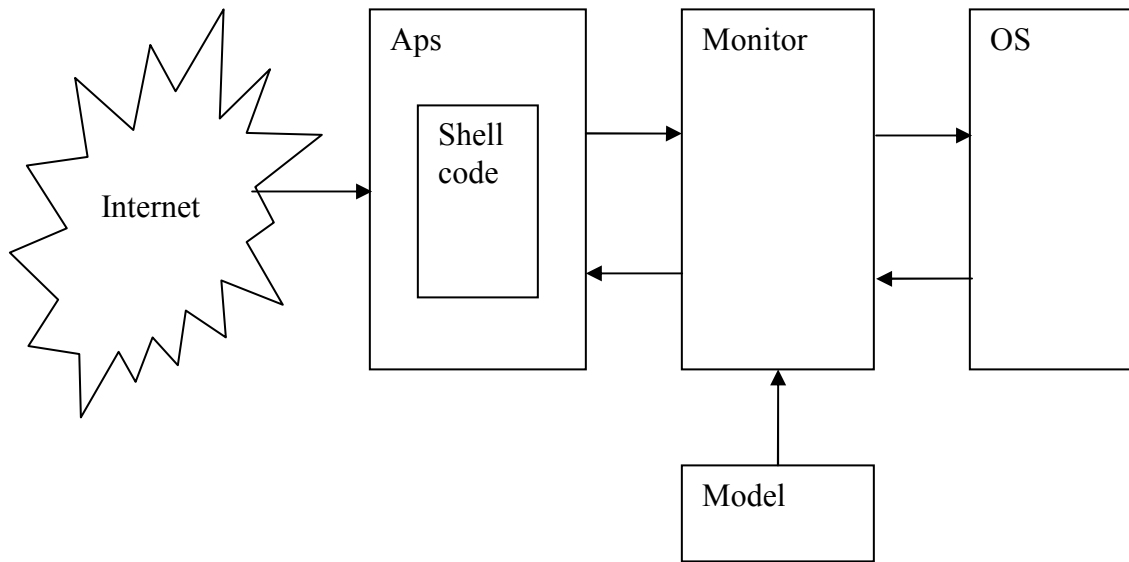


4/25/2006 Lecture Notes: DOS
Beili Wang

Last lecture we talked about how Intrusion Detection works. Today we will talk about the attacks.

Intrusion Detection



In Intrusion Detection, we call “Model” instead of “Policy”, because “Model” is derived from the application itself. It describes the correction behavior of the application, what the application allows to do, and what system calls that the application can make.

The attacker writes the shell code to attack.

Mimicry Attack:

Mimicry Attack:
Shell code agrees with the Model.

1. Simple Model: Set of system calls.
How the attacker will attack if he knows the set? The attacker writes the shell code that the monitor pass to model and the model says the code is ok. → shell code agrees the model.
Shell code only uses system calls in the set of allowed calls.
The attacker can only use the system call in the set to write the shell code. He may download Aps and just use the system calls allowed in the Model.
2. FSA (Finite State Automaton):
What about the FSA model? Shell code agrees with FSA.
The attacker has to
 1. download Aps
 2. construct model (same model as the victim's)
 3. search for sequence of system calls accepted by FSA and that violates securityTypically, this is not too hard. The attacker can calculate average of number of options for every step. It is only slightly more complicate than the Simple Model.

Static models:

- over approximation
- valid sequence accepted
- no bogus warnings
- easier mimicry attacks

Dynamic models:

- more precise
- mimicry attacks are harder
- may get false positives

Mimicry attack: any models that get shell code agrees with model.

What if attack does not make nay system call?

What kind of attacks does not make system calls:

- DOS attack (Denial – of – Service)
- changing important data structure
 - authenticated flag
 - copy security data into outgoing buffers (for example: gain private key)
 - others (mimicry attack more flexible)

Therefore, Intrusion Detection cannot prevent attack but detect attack. It raises the bar for attackers and makes them more creative.

DOS (Denial-of-Service) attack

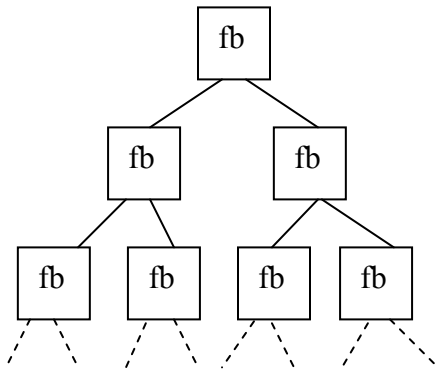
Resource Exhaustion Attacks:

- Memory
- CPU
- Network
- OS objects
- Disk

For example: fork() bomb exhaust:

- Memory
- CPU
- OS objects

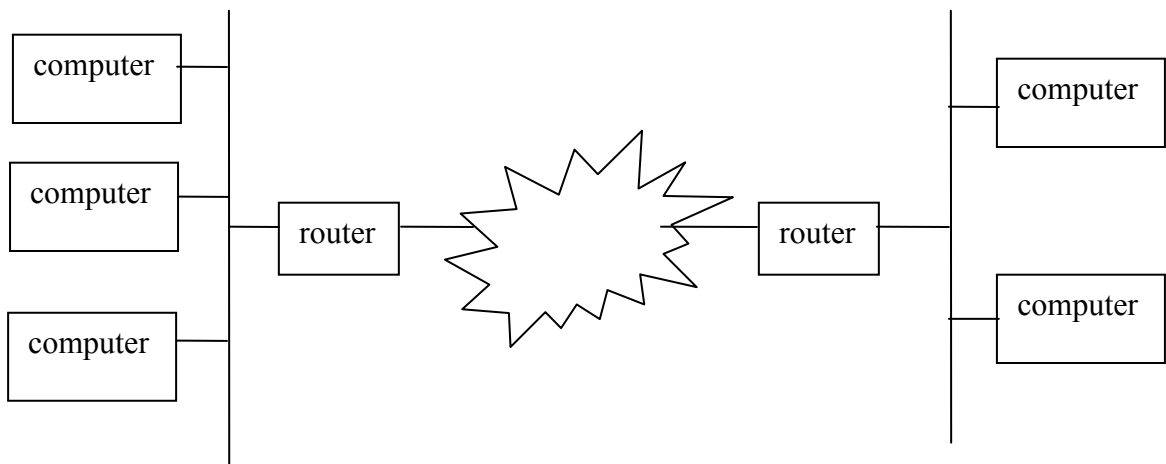
```
Code: while(1)
      fork( );
```



Another example: Randomly access to a huge array and causes thrashing.

DOS on Networks:

Ethernet



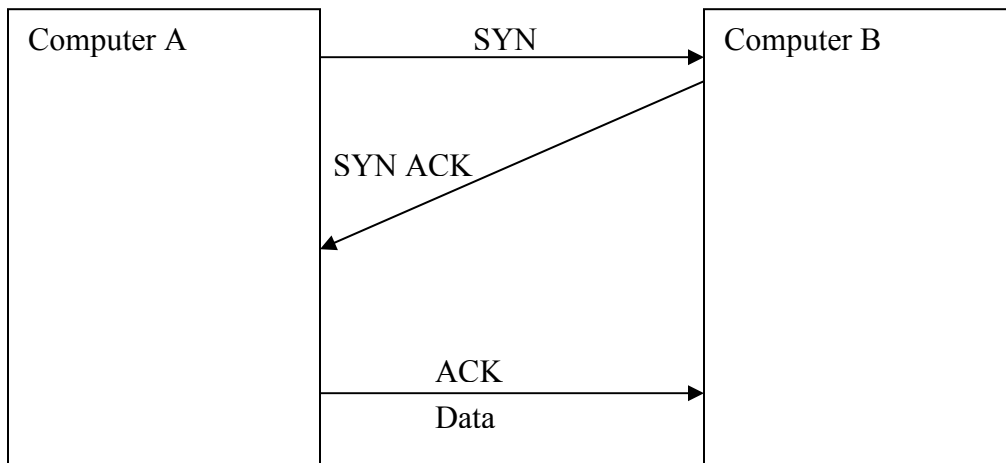
net: 237.128.7.0
(0 – network address)
(1 to 255 for machines)
broadcast: 237.128.7.255

Protocol: TCP (Transmission Control Protocol)

Use of TCP/IP:

- In order (delivery)
- Reliable (If drop, resend again)
- Consecution /flow control
- Ports (to which computer, which process on computer)

TCP: 3 ways handshakes:



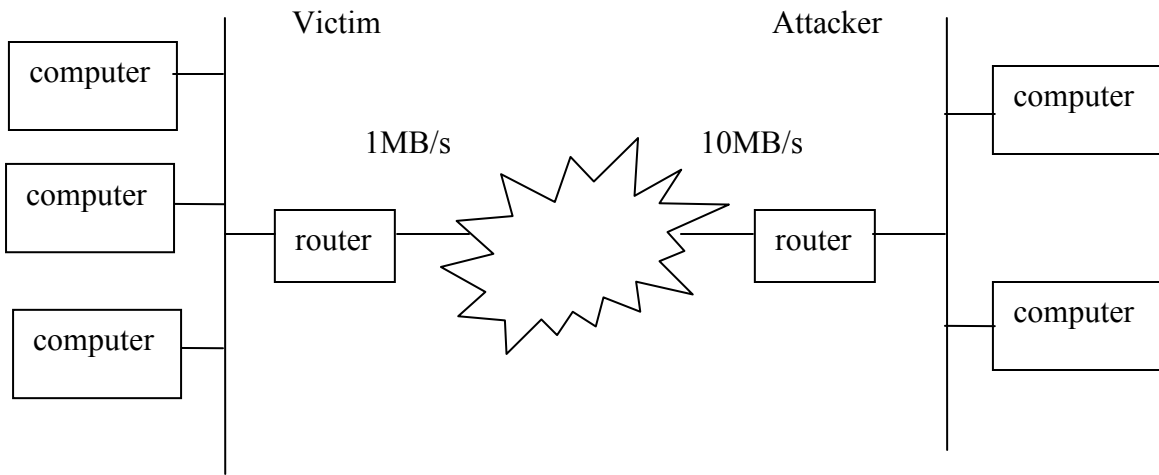
1. Computer A sends SYN packet
2. Computer B replies SYN ACK (acknowledgement) packet
3. Computer A sends ACK includes Data to Computer B

Attacks:

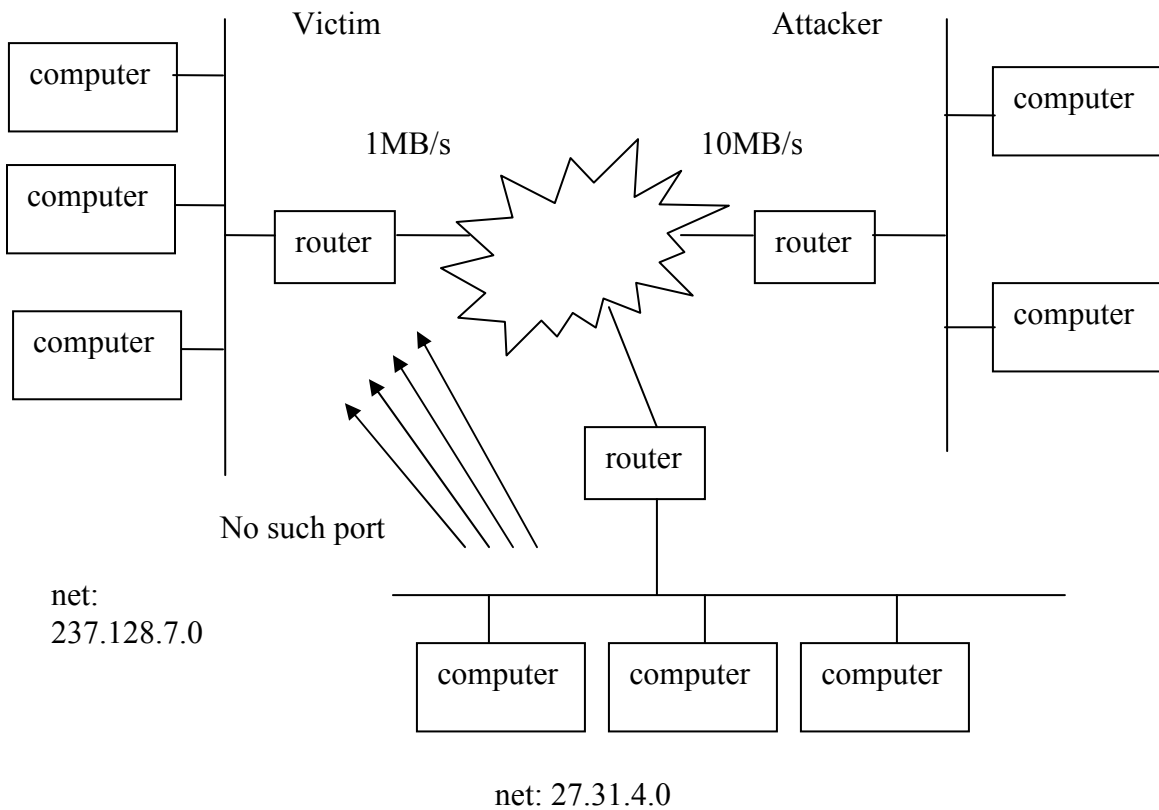
- **Small site – easy (Bandwidth exhaustion)**
- **Medium sized sites – (Smurf Attack)**
- **Huge sites – (Worm)**

Bandwidth exhaustion:

Attacker sends 10 MB/s packets flood down the connection. The victim's 1MB/s router spends all the time to deal with them. No space for the victim's packets in or out.



Medium sized site: Smurf Attack



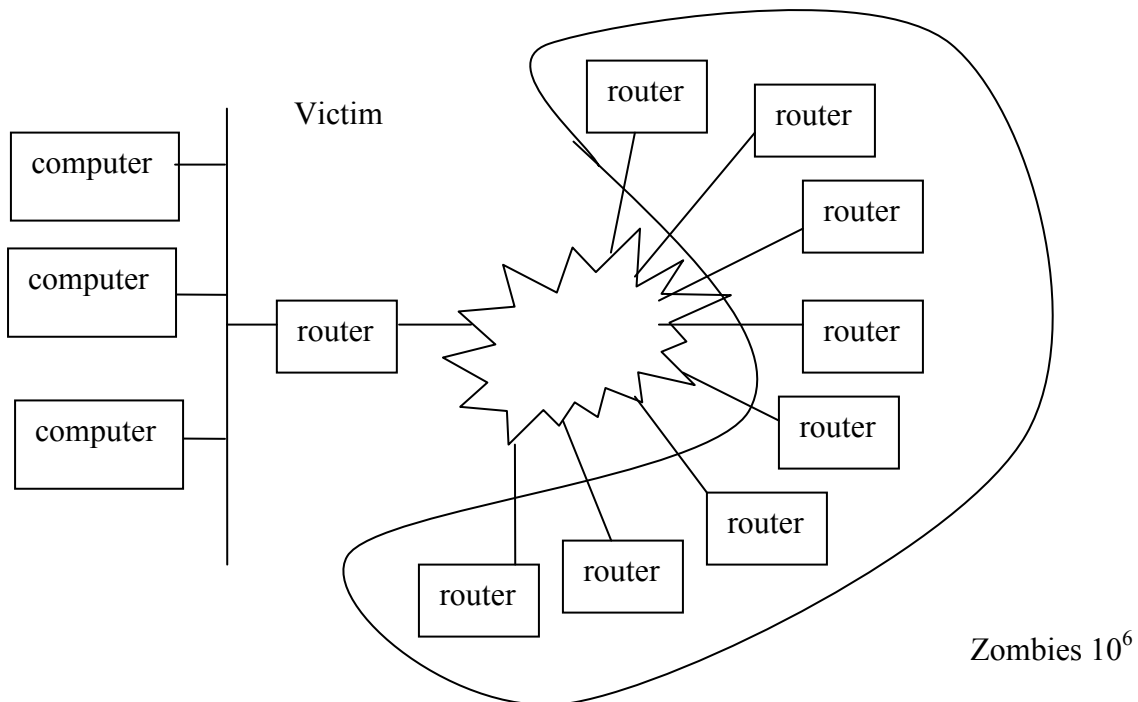
Conditions that make the attack possible on the networks:

1. Attacker can lie about the return address.
2. There is no such web server at the address.

For example, attacker sends package to 27.31.4.255 (broadcast) with port 27,000, so that everyone on the network gets the package. Since there is no process on the 27,000 port, everyone on the network definitely will return the package to the sender. The attacker lies about the “From” address and writes the victim’s address: 237.128.7.12, so everyone on the networks sends back the package to the victim. The attacker sends 1 package, but victim receives tones. 10MB/s becomes GB/s.

This attack may not be realistic today, because the router will reject if it uses firewall, net machine, etc.

Huge Sites (for example: Amazon.com) – Worm



- Finds buffer overflow.
 - Exploit code (worm).
 - Write 1 command message, all the machine send to the victim.
- 15 minutes, about 100,000 machines are under control.

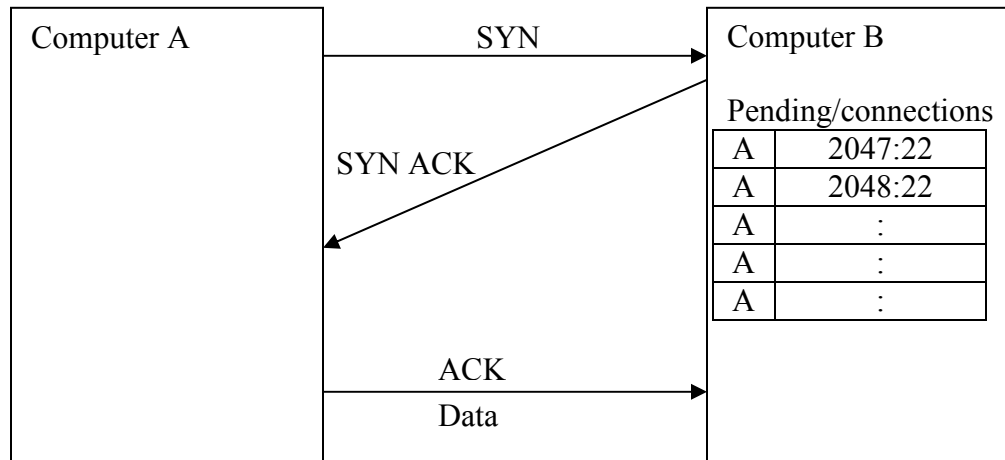
Victim side to defeat the attack:

- **Small site – firewall, drop all the packets from the attacker**
- **Medium sized sites – drop all the packets from the particular net**
- **Huge sites – cannot block all these packets from random addresses**

Solution:

- **Get a really big connection**
- **Mirror server everywhere**

3 Ways handshake TCP/IP



Idea: Computer B has a little array which stores pending connections. For example, Computer A sends SYN message, Computer B stores A with port number 2047 : 22 (22 is SSH) in the array. Then, Computer B sends SYN ACK message to Computer A and waits for ACK message from Computer A. When 3 ways handshake connection completes, Computer B erases the entry with the port in the array. If Computer A does not send ACK message, Computer B keeps the entry and will time out, then waits Computer A to send again.

Attack: If attacker sends all the SYN messages and does not complete any of them, it will take all the entry spaces in the array so that when there is Computer C wants to talk to Computer B, Computer B does not have space available for Computer C and eventually drop message from C. This attack is called SYN Flood Attack. Early Implementation of TCP/IP only has 5 entries in the array.

Solution: To defeat SYN Flood Attack, use SYN Cookies.

Idea: How to make backwards compatible? The basic idea is used for a lot of sever design to prevent OS resources get used up. SYN Cookies uses MAC (Message Authentication Code) and security key. Computer B does not store anything in the array or table, but packages up the entry, sends back to client Computer A and let the client store it. Computer B uses MAC so that the client Computer A cannot modify or change the entry. And, to complete the connection, Computer A has to send back the MAC entry with data to Computer B. By this way, Computer B does not need to use the table any more.

Per client:

- Package it up
- Send it to client
- Let client to store it
- MAC so that client cannot modify it
- Client has to send it back again

