CS329E: Elements of Security

Availability

Dr. Bill Young
Department of Computer Sciences
University of Texas at Austin

Last updated: April 24, 2015 at 08:39

Aspects of Computer Security

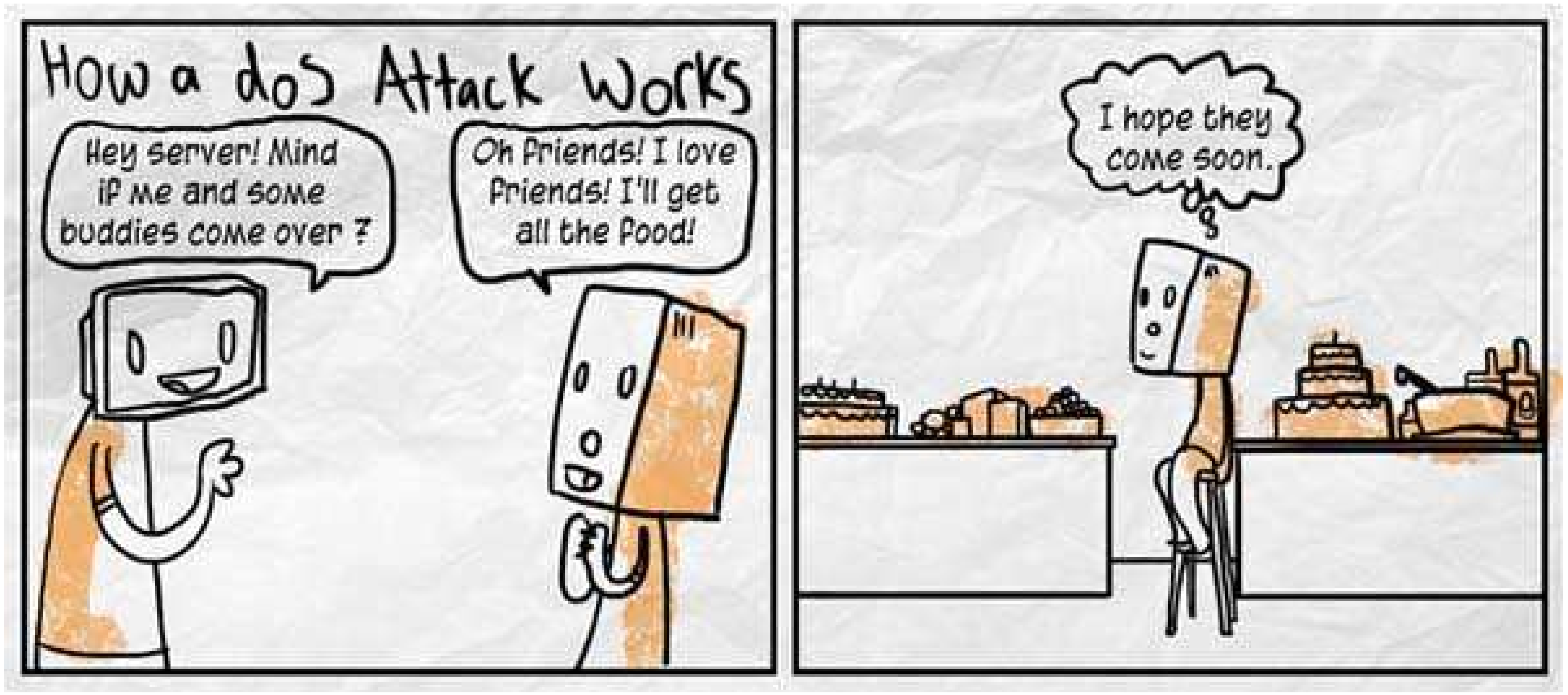
Recall that historically computer security has been defined to encompass:

Confidentiality:	(also called secrecy/privacy) who can *read* information;
Integrity:	who can *write* or modify information;
Availability:	are resources available when needed.

In this section we very briefly touch on Availability.

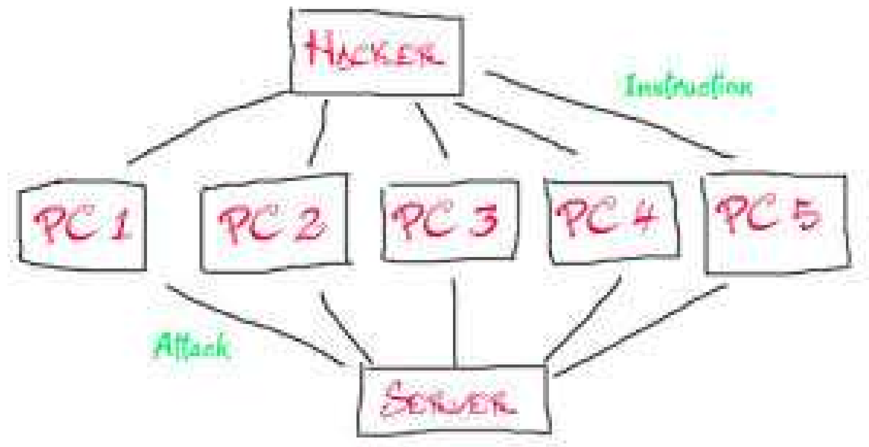
Availability Attacks

Attacks on availability are typically referred to as *denial of service* or DoS attacks. An attacker somehow prevents a user from accessing or utilizing available system resources, or disables the functioning of the user process.


How a doS Attack Works
Hey server! Mind if me and some buddies come over ?
Oh friends! I love friends! I'll get all the food!
I hope they come soon.

Availability Attacks

A particular class of DoS attacks are labeled *Distributed Denial of Service* or DDoS attacks. These typically involve an attacker co-opting the services of many other machines to participate in the attack.

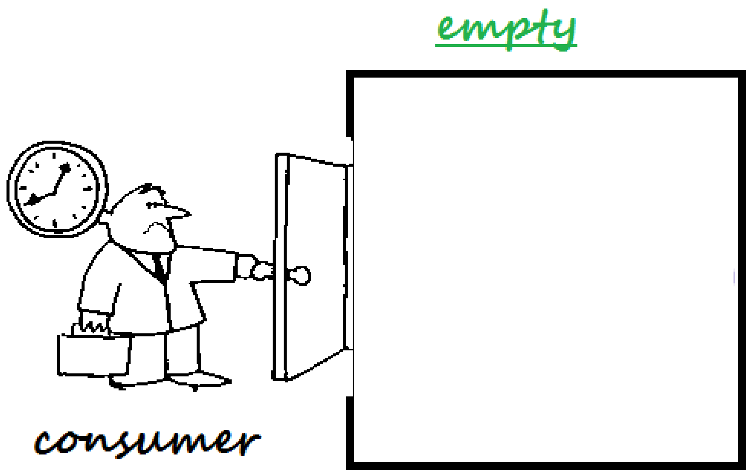


Many worms and virus attacks are DoS attacks. E.g., in Feb. 2000 a DDoS attack on Amazon, eBay, Yahoo, and others caused tens of millions of dollars in damage. Buy.com reported traffic of 800Mbit/sec, eight times the site's capacity.

Gresty's Framework

David Gresty at Liverpool John Moore's University decomposes DoS attacks into two groups:

1. **the consumer problem**: (also known as the "man-in-the-middle" attack) the attacker gets logically between the client and service and somehow disrupts the communication.
2. **the producer problem**: the attacker produces, offers or requests so many services that the server is overwhelmed.



Local vs Remote Attacks

Local DoS attacks include physical attacks:

1. cutting the power;
2. unplugging connections;
3. welding a door shut;
4. stealing the hardware.

These can typically be countered with physical security.

Remote attacks mean those that happen over a network or remote access link. These typically require technological solutions to counter.

Typical Scenarios

In a typical connection, the user sends a message asking for some service. There are two common DoS scenarios:

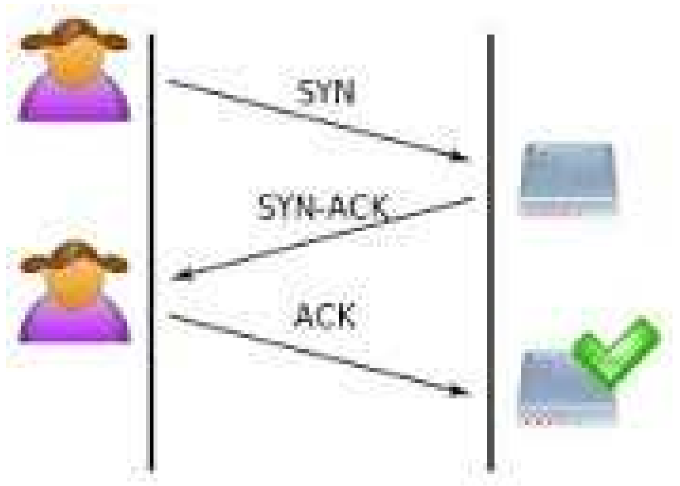
- The volume of requests may overwhelm the server.
- The transaction may involve some handshake. The attacker does not respond and the server ties up space and resources waiting for a response.

A classic example of the second is so-called *syn flooding*.

SYN Flooding

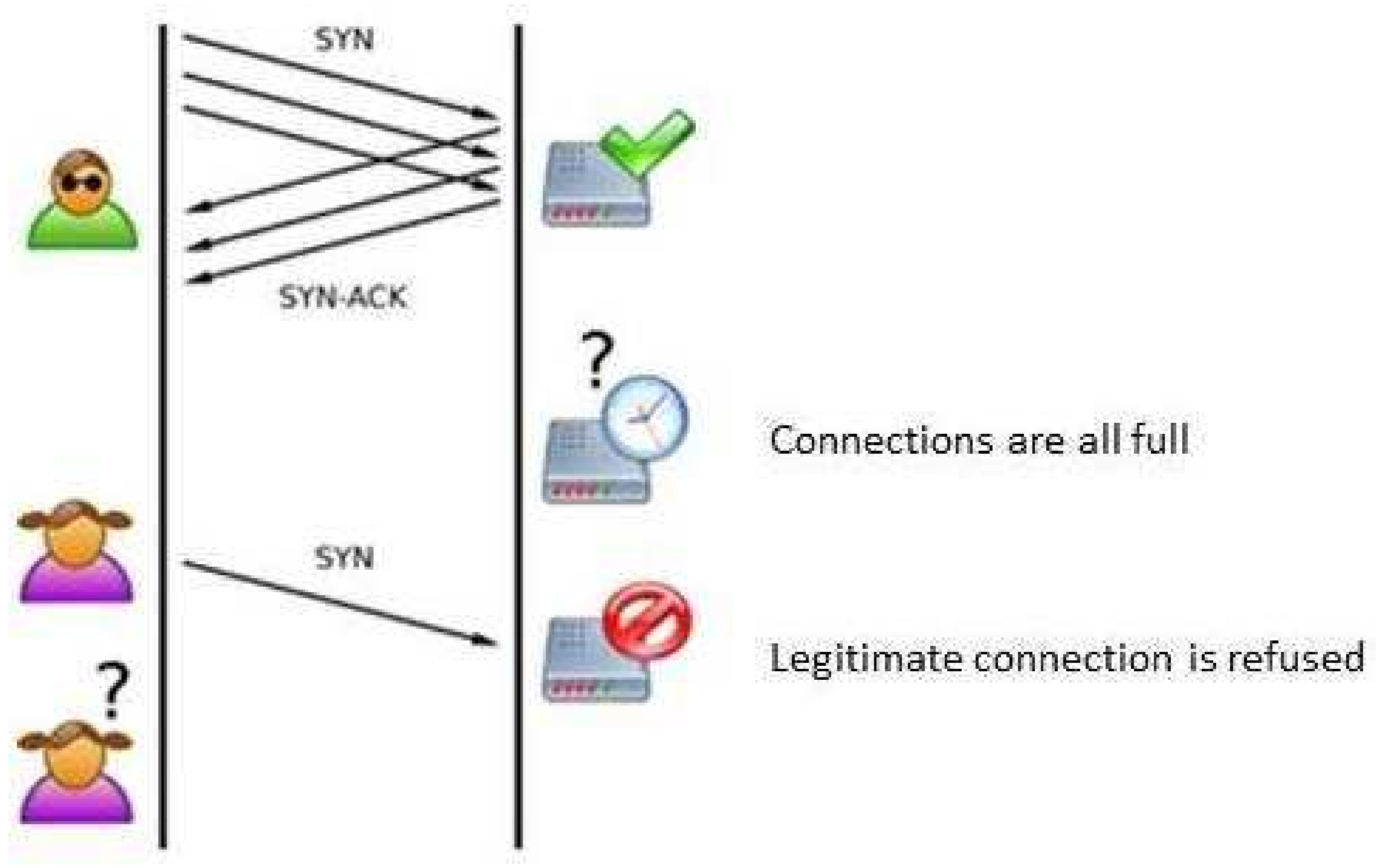
There is a standard protocol by which a client establishes a TCP connection with a server.

When the server receives the SYN packet (the Synchronize flag is set), it allocates space in an internal table (usually 600 bytes) and sends the SYN/ACK packet back to the caller. The connection remains "half-open" until the ACK packet is received by the server or the connection times out.



SYN Flooding (cont.)

A *SYN Flooding* attack happens when an attacker forges the return address on a number of SYN packets. The server allocates space in its queue for half-open connections and sends the SYN/ACK packets.



SYN Flooding (cont.)

Because the return address has been faked, the receiver may be unavailable or unable to ACK. The server's queue is quickly filled by records corresponding to half-open sessions. Legitimate accesses are denied until the connections time-out.

Is the SYN flooding problem inherent in the way TCP connections are established? How might you close the vulnerability?

SYN Flooding (cont.)

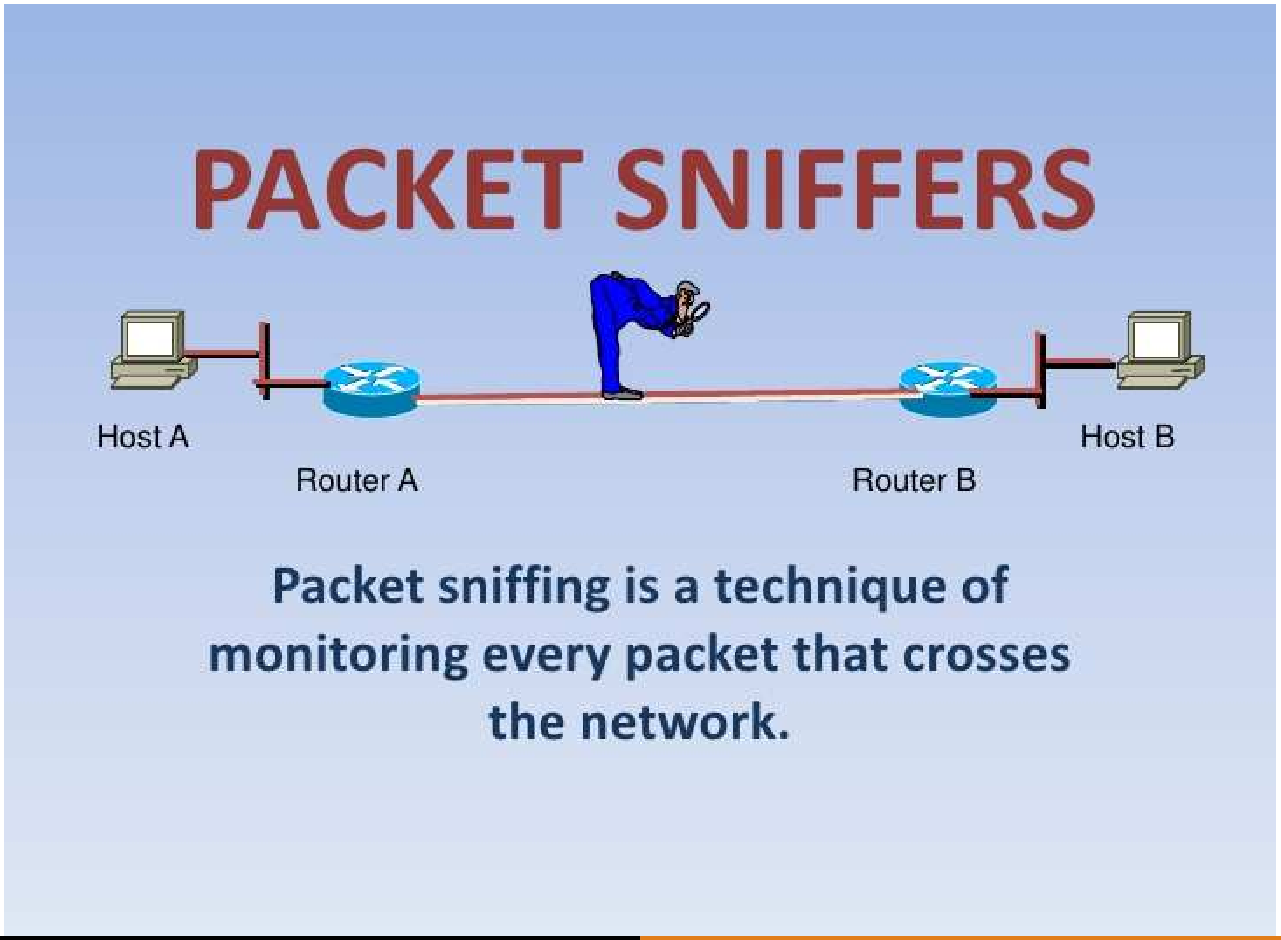
How might you close the vulnerability?

1. *Increase the queue size:* typically only 8 half-open connections are allowed; could consume considerable resources.
2. *Shorten the time-out period:* might disallow connections by slower clients.
3. *Filter suspicious packets:* if the return address does not match the apparent source, discard the packet. May be hard to determine.
4. *Change the algorithm:* instead of storing the record in the queue, send the information encrypted along with the SYN/ACK. A legitimate client will send it back with the ACK.

The last was what was actually done and defeated the syn flood attack.

Blocking Flooding Attacks

A filter or *packet sniffer* can detect patterns of identifiers in the request stream and block messages in that pattern. *Ingress filtering* means sniffing incoming packets and discarding those with source IP addresses outside a given range (e.g., those known to be reachable via that interface).



Packet sniffing is a technique of monitoring every packet that crosses the network.

Blocking Flooding Attacks

It is a very hard problem to be able to discriminate patterns of attack from patterns of standard usage.

An overly aggressive filter also gives a type of denial of service by discarding too many legitimate requests.

Protection from DoS Attacks

A good firewall can help by filtering out illegal requests. However, a typical DoS flooding attack may comprise only legal requests.

An *intrusion detection system* (IDS) can analyze traffic patterns and react to anomalous patterns. However, often there is nothing apparently wrong but the volume of requests.

An IDS reacts after the attack has begun.

Protection from DoS Attacks

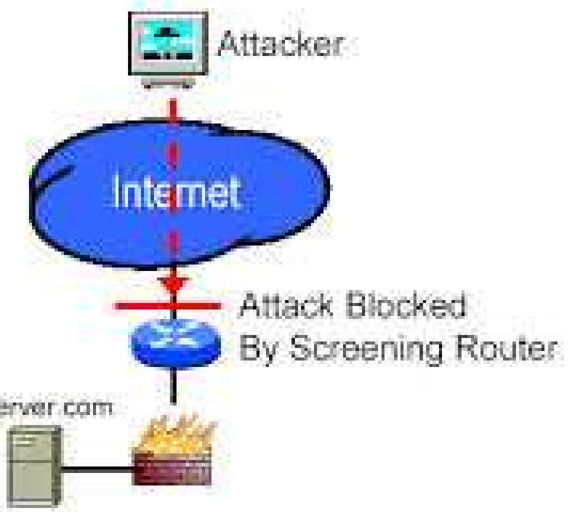


Figure 1-2: Screening Routers to prevent DOS attack

An *intrusion prevention system* (IPS) attempts to prevent intrusions by more aggressively blocking attempted attacks. This assumes that the attacking traffic can be identified.

IDS/IPS are useful for confidentiality and integrity attacks, not just DoS attacks.

Potential DDoS Solutions

A DDos attack comes when an attacker takes over a number of nodes in a network and uses them as bots to launch a coordinated producer attack. How might you counter them?

Potential DDoS Solutions

A DDos attack comes when an attacker takes over a number of nodes in a network and uses them as bots to launch a coordinated producer attack. How might you counter them?

1. *over-provisioning the network*—have too many servers to be overwhelmed (expensive and unworkable);
2. *filtering attack packets*—somehow distinguish the attack packets from regular packets (may not be possible);
3. *slow down processing*—disadvantages all requestors, but perhaps disproportionately disadvantages attackers;
4. *"Speak-up" solution* (Mike Walfish)—request *additional* traffic from all requestors.

Walfish's solution assumes that the attacker's bots are already maxed out. So this solution raises the proportion of valid to invalid requests.

Intrusion Detection Errors

An intrusion detection system is:

accurate:	if it detects all genuine attacks;
precise:	if it never reports legitimate behavior as an attack.

It is easy to make an IDS that is either accurate or precise! Why?
It's hard to do both simultaneously.

Intrusion Detection Errors

There are two types of errors when considering any intrusion detection system.

False negatives: a genuine attack is not detected.

False positives: harmless behavior is mis-classified as an attack.

...GREAT NEWS! YOU'RE PREGNANT!!!

Which do think is a bigger problem? Why?

Evaluating IDS Performance

For any intrusion detection system there are two major indicators of performance:

The detection rate: the number of intrusions detected divided by the total number of intrusions present.

The false positive rate: the number of normal instances flagged as intrusions divided by the total number of normal instances.

Which of these seem most useful? Which of these can you typically compute?

Intrusion Detection Errors

A undetected attack can lead to severe problems. But frequent false alarms can lead to the system being disabled or ignored. A perfect IDS would be *both accurate and precise.*

- Statistically, attacks are fairly rare events.
- Most intrusion detection systems suffer from the *base-rate fallacy*.

Base-Rate Fallacy

Suppose that only 1% of traffic are actually attacks and the detection accuracy of your IDS is 90%. What does that even mean?

Base-Rate Fallacy

Suppose that only 1% of traffic are actually attacks and the detection accuracy of your IDS is 90%. What does that even mean?

- the IDS classifies an attack as an attack with probability 90%
- the IDS classifies a valid connection as attack with probability 10%

What is the probability that a connection flagged as an attack is not really an attack, i.e., a false positive?

Base-Rate Fallacy Example

Suppose that only 1% of traffic are actually attacks and the detection accuracy of your IDS is 90%.

What is the probability that a connection flagged as an attack is not really an attack, i.e., a false positive?

Base-Rate Fallacy Example

Suppose that only 1% of traffic are actually attacks and the detection accuracy of your IDS is 90%.

What is the probability that a connection flagged as an attack is not really an attack, i.e., a false positive?

There is approximately 92% chance that a raised alarm is false.

The Base-Rate Fallacy

Another example: suppose this were a pregnancy test that is 90% accurate. Is that a good diagnostic test or not? Suppose only 1% of the population were actually pregnant. Then 92% of the positive results would be wrong.

Now suppose you apply the test to 100 *men*. You could expect 10 positives, probably all false!

Bottom line: if you have an IDS in place, it must be very accurate or you'll soon turn it off because *almost all* of your alarms will be false alarms.

DoS Case Study: CodeRed

On June 18, 2001 eEye publicized a buffer-overflow vulnerability in Microsoft's IIS webservers. It allows system-level execution of code and arises due to inadequate bounds checking on some input buffers.

On July 12, 2001, the CodeRed virus began attacking machines running unpatched versions of Microsoft's IIS webserver.

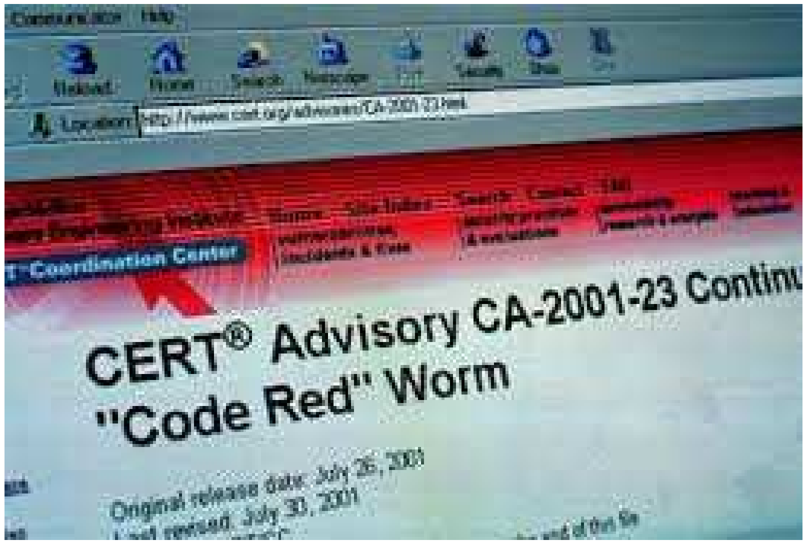
CodeRed (continued)

On July 13, 2001, investigators from eEye Digital Security worked overnight to analyze the worm. The called it "CodeRed" because of the CodeRed Mountain Dew that they used to fuel their efforts and because of the "Hacked by Chinese" message.

Code Red (version I) does the following:

- If date is between 1st and 19th of the month, generate a random list of IP addresses and attempt to infect those machines.
- On 20th to 28th of the month, launch a DoS flooding attack on `www1.whitehouse.gov`.
- The worm also defaces some webpages with the words "Hacked by Chinese."

CodeRed Version 1


CERT® Advisory CA-2001-23 Continu
"Code Red" Worm
Original release date: July 26, 2001
Last revised: July 30, 2001

The worm uses a *static seed* in its random number generator and thus generates identical lists of IP addresses on each infected machine.

Each infected machine probed the same list of machines, so the worm spread slowly.

The IP address for `www1.whitehouse.gov` was changed so the DoS attack failed.

CodeRed (continued)

Because of flaws in the design, especially the static seed, Code Red (version 1) did very little damage.

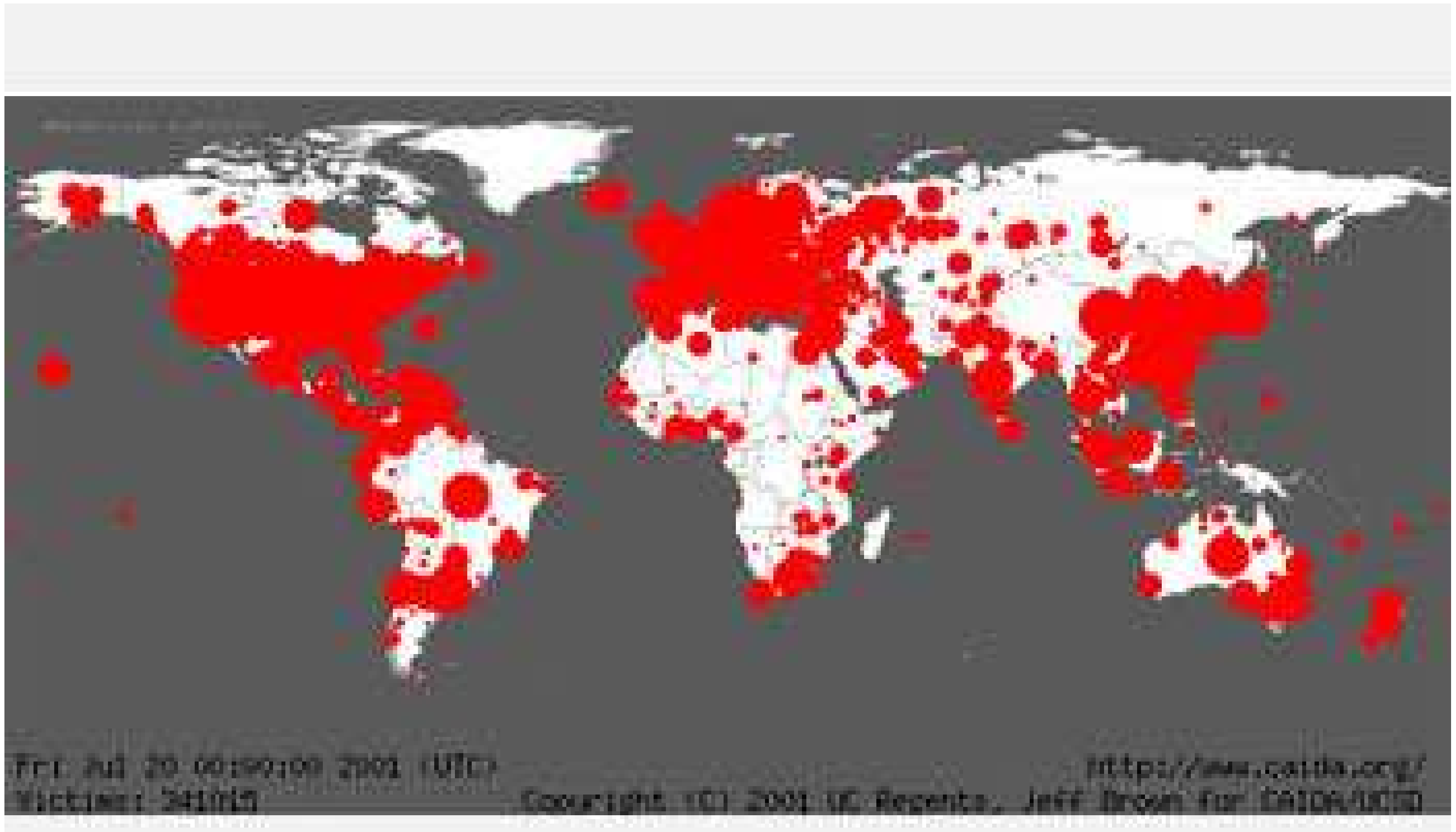
CodeRed version 1 worm is memory resident. An infected machine can be disinfected by simply rebooting it.

Once-rebooted, the machine is still vulnerable to repeat infection, and that is likely since each newly infected machine probes the same list of IP addresses in the same order.

CodeRed (version 2)

On July 19, 2001 a variant began to circulate that was identical but uses a *random seed* in the random number generator.

This had a major impact: more than 359,000 machines were infected with CodeRed version 2 in just fourteen hours.


Fri Jul 20 00:00:00 2001 (UTC)
Victims: 341015
http://www.caida.org/
Copyright (C) 2001 UC Regents. Jeff Brown for CAIDA/UCSD

CodeRed (version 2)

Had a much greater impact on global infrastructure due to the sheer volume of hosts infected and probes sent to infect new hosts.

Also wreaked havoc on some additional devices with web interfaces: routers, switches, DSL modems, and printers. They either crashed or rebooted when an infected machine attempted to send them a copy of the worm.

CodeRedII

On August 4, 2001, an entirely new worm, CodeRedII began to exploit the buffer-overflow vulnerability in Microsoft's IIS webservers.

The source code contained the string "CodeRedII" which became the name of the new worm.

Works as follows:

- When a worm infects a new host, it first determines if the system has already been infected.
- If not, the worm initiates its propagation mechanism, sets up a "backdoor" into the infected machine, becomes dormant for a day, and then reboots the machine.
- Begins a process of propogating itself (see below).

CodeRedII Propogation

Launches 300 or 600 threads in propogation attempt.

CodeRedII generates a random IP address and then applies a mask to produce the IP address to probe. The length of the mask determines the similarity between the IP address of the infected machine and the probed machine.

1/8th of the time, CodeRedII probes a completely random IP address.

1/2 of the time, CodeRedII probes a machine in the same /8 (so if the infected machine had the IP address 10.9.8.7, the IP address probed would start with 10.).

3/8ths of the time, it probes a machine on the same /16 (so the IP address probed would start with 10.9.).

CodeRedII Propogation

CodeRedII avoids probing IP addresses in 224.0.0.0/8 (multicast) and 127.0.0.0/8 (loopback).

The bias towards the local /16 and /8 networks means that an infected machine may be more likely to probe a susceptible machine, since machines on a single network are more likely to be running the same software as machines on unrelated IP addresses.

Danger of CodeRedII

Unlike CodeRed, CodeRedII neither defaces web pages on infected machines nor launches a Denial-of-Service attack.

Also unlike CodeRed, CodeRedII is not memory resident, so rebooting an infected machine does not eliminate CodeRedII.

Installs a mechanism for remote, root-level access to the infected machine. This backdoor allows any code to be executed, so the machines could be used as zombies for future attacks.

Rates of Response

Studies of CodeRed show that the rate at which vulnerable machines were patched varied widely from country to country. The attack began on July 19. On August 14 the following statistics were estimated:

Country	Patched	Unpatched
United Kingdom	66%	34%
United States	60%	40%
Canada	58%	42%
Germany	56%	44%
Netherlands	46%	54%
Japan	39%	61%
Australia	37%	63%
Korea	20%	80%
Taiwan	15%	85%
China	13%	87%

A large number of machines remained vulnerable to the same or similar attack.

Lessons of CodeRed

1. DoS attacks have serious financial and social consequences ($2.6 Billion according to one estimate).
2. A known vulnerability may be exploited very quickly.
3. Attackers adapt quickly.
4. Unpatched machines remain vulnerable.
5. An infected machine becomes a potential weapon.
6. Others?

Install Patches

[A report from Verizon Business] *covering 500 forensic investigations, involving 230 million compromised customer records, found that nine out of 10 breaches attributed to hacking attacks took advantage of a vulnerability for which a fix was available at least six months prior to the attack.*