

# CS162 Operating Systems and Systems Programming Lecture 22

## Security (II)

April 16, 2012  
Anthony D. Joseph and Ion Stoica  
<http://inst.eecs.berkeley.edu/~cs162>

## Recap: Security Requirements in Distributed Systems


- Authentication
  - Ensures that a user is who is claiming to be
- Data integrity
  - Ensure that data is not changed from source to destination or after being written on a storage device
- Confidentiality
  - Ensures that data is read only by authorized users
- Non-repudiation
  - Sender/client can't later claim didn't send/write data
  - Receiver/server can't claim didn't receive/write data

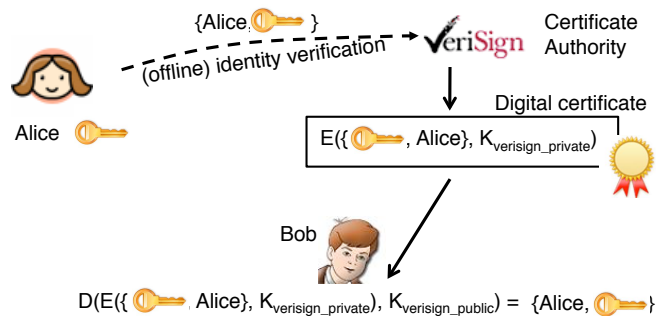
4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.2

## Recap: Digital Certificates

- How do you know  is Alice's public key?
- Main idea: trusted authority signing binding between Alice and its private key



4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.3

## Some HTTPS Vulnerabilities

- Break into any of the 600+ Certificate Authorities that your browser will trust
  - This attack has occurred 14 times total (as of 10/27/2011)
  - *4 times since June!*
  - <https://www.eff.org/deeplinks/2011/10/how-secure-https-today>
- A government could order a Certificate Authority to produce a malicious certificate for any domain
  - CAs are located in 52+ countries
  - There is circumstantial evidence that has happened
- Alternative: <http://convergence.io/>
  - User chooses one or more "notaries", servers that anyone can run and anyone can trust
  - CS 161 guest lecture: <http://www.youtube.com/watch?v=Z7WI2FW2TcA>

4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.4

## Passwords: How easy to guess?

- Three common ways of compromising passwords
- Password Guessing:
  - Often people use obvious information like birthday, favorite color, girlfriend's name, etc...
  - Trivia question 1: what is the most popular password?
  - Trivia question 2: what is the next most popular password?
  - Answer: (from 32 million stolen passwords– Rockyou 2010) <http://www.nytimes.com/2010/01/21/technology/21password.html>
- Dictionary Attack (against stolen encrypted list):
  - Work way through dictionary and compare encrypted version of dictionary words with entries in `/etc/passwd`
  - <http://www.skullsecurity.org/wiki/index.php/Passwords>
- Dumpster Diving:
  - Find pieces of paper with passwords written on them
  - (Also used to get social-security numbers, etc.)

4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.5

## Passwords: How easy to guess? (cont'd)

- Paradox:
  - Short passwords are easy to crack
  - Long ones, people write down!
- Technology means we have to use longer passwords
  - UNIX initially required lowercase, 5-letter passwords: total of  $26^5=10$ million passwords
    - » In 1975, 10ms to check a password→1 day to crack
    - » In 2005, .01μs to check a password→0.1 seconds to crack
  - Takes less time to check for all words in the dictionary!

4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.6

## Passwords: Making harder to crack

- Can't make it impossible to crack, but can make it harder
- Technique 1: Extend everyone's password with a unique number ("Salt" – stored in password file)
  - Early UNIX used 12-bit "salt" →dictionary attacks 4096x harder
  - Without salt, could pre-compute all the words in the dictionary hashed with UNIX algorithm (modern salts are 48-128 bits)
- Technique 2: Require more complex passwords
  - Make people use at least 8-character passwords with upper-case, lower-case, and numbers
    - »  $70^8=6 \times 10^{14}=6$ million seconds=69 days@0.01μs/check (no salt)
  - Unfortunately, people still pick common patterns
    - » e.g. Capitalize first letter of common word, add one digit
- Technique 3: Delay checking of passwords
  - If attacker doesn't have access to `/etc/passwd`, delay every remote login attempt by 1 second
  - Makes it infeasible for rapid-fire dictionary attack

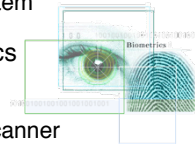
4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.7

## Passwords: Making harder to crack (cont'd)

- Technique 4: Assign very long passwords/passphrases
  - Can have more entropy (more randomness→harder to crack)
  - Embed password in a smart card (or ATM card)
    - » Requires physical theft to steal password
    - » Can require PIN from user before authenticates self
  - Better: have smartcard generate pseudorandom number
    - » Client and server share initial seed
    - » Each second/login attempt advances random number
- Technique 5: "Zero-Knowledge Proof"
  - Require a series of challenge-response questions
    - » Distribute secret algorithm to user
    - » Server presents number; user computes something from number; returns answer to server; server never asks same "question" twice
  - Often performed by smartcard plugged into system
- Technique 6: Replace password with Biometrics
  - Use of one or more intrinsic physical or behavioral traits to identify someone
  - Examples: fingerprint/palm reader, iris/retinal scanner



4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.8

## This Lecture

- Host Compromise
  - Attacker gains control of a host
- Denial-of-Service
  - Attacker prevents legitimate users from gaining service
- Attack can be both
  - E.g., host compromise that provides resources for denial-of-service

4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.9

## Host Compromise

- One of earliest major Internet security incidents
  - Internet Worm (1988): compromised almost every BSD-derived machine on Internet
- Today: estimated that a single worm could compromise 10M hosts in < 5 min using a zero-day exploit
- Attacker gains control of a host
  - Reads data
  - Compromises another host
  - Launches denial-of-service attack on another host
  - Erases data

4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.10

## Stepping Stone Compromise

- RSA SecurID compromise (March 2011)
  - 2-factor authentication
  - Code changes every few secs
  - Data on codes stolen using an *Advanced Persistent Threat* attack
- 760 companies attacked using stolen SecurID info
  - 20% of Fortune 100
  - Charles Schwab & Co., Cisco Systems, eBay, European Space Agency, Facebook, Freddie Mac, Google, General Services Administration, IBM, Intel Corp., IRS, MIT, Motorola, Northrop Grumman, *Verisign*, VMWare, Wachovia, Wells Fargo, ...
  - <http://krebsonsecurity.com/2011/10/who-else-was-hit-by-the-rsa-attackers/>



4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.11

## Definitions

- Worm
  - Replicates itself usually using buffer overflow attack
- Virus
  - Program that attaches itself to another (usually trusted) program or document
- Trojan horse
  - Program that allows a hacker a back door to compromised machine
- Botnet (Zombies)
  - A collection of programs running autonomously and controlled remotely
  - Can be used to spread out worms, mounting DDoS attacks
- Advanced Persistent Threat
  - Highly sophisticated, long duration attack typically launched using one or more zero-day exploits

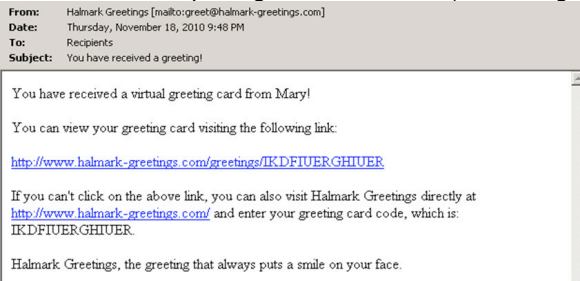
4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.12

## Trojan Example

- Nov/Dec e-mail message sent containing holiday message and a link or attachment
- Goal: trick user into opening link/attachment (social engineering)



- Adds keystroke logger or turns into zombie
- How? Typically by using a buffer overflow exploit

4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.13

## Buffer Overflow

- Part of the request sent by the attacker **too large** to fit into buffer program uses to hold it
- Spills over into memory beyond the buffer
- Allows **remote** attacker to inject executable code

```
void get_cookie(char *packet) {
    . . . (200 bytes of local vars) . . .
    munch(packet);
    . . .
}

void munch(char *packet) {
    int n;
    char cookie[512];
    . . .
    code here computes offset of cookie in
    packet, stores it in n
    strcpy(cookie, &packet[n]);
    . . .
}
```

4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.14

## Example: Normal Execution

```
void get_cookie(char *packet) {
    . . . (200 bytes of local vars) . . . X + 200
    munch(packet);
    . . .
}

void munch(char *packet) {
    int n;
    char cookie[512];
    . . .
    code here computes offset of cookie in
    packet, stores it in n
    strcpy(cookie, &packet[n]);
    . . .
}
```

4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.15

## Example: Normal Execution

```
void get_cookie(char *packet) {
    . . . (200 bytes of local vars) . . . X + 200
    munch(packet);
    . . .
}

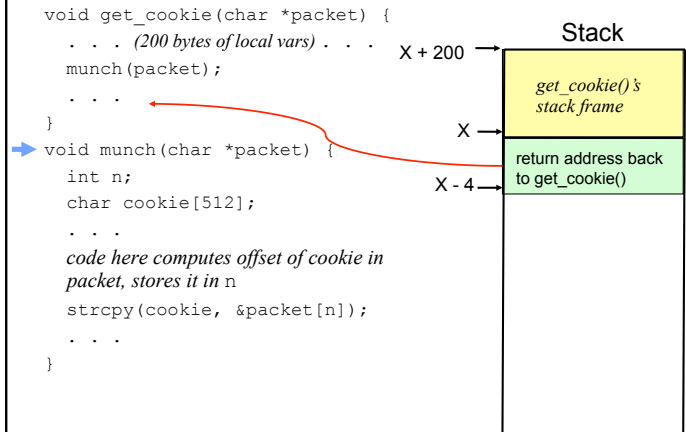
void munch(char *packet) {
    int n;
    char cookie[512];
    . . .
    code here computes offset of cookie in
    packet, stores it in n
    strcpy(cookie, &packet[n]);
    . . .
}
```

4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.16

### Example: Normal Execution

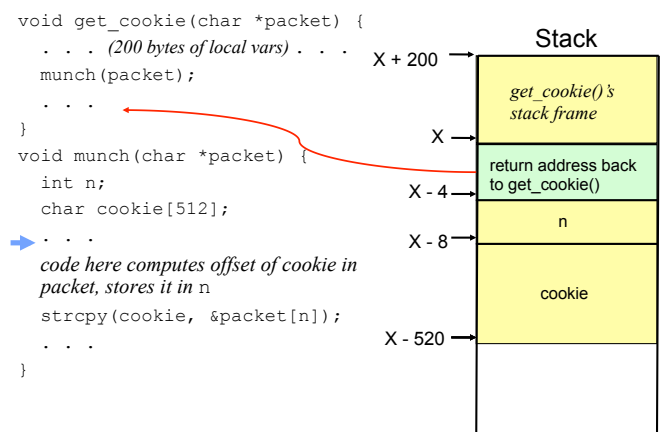


4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.17

### Example: Normal Execution

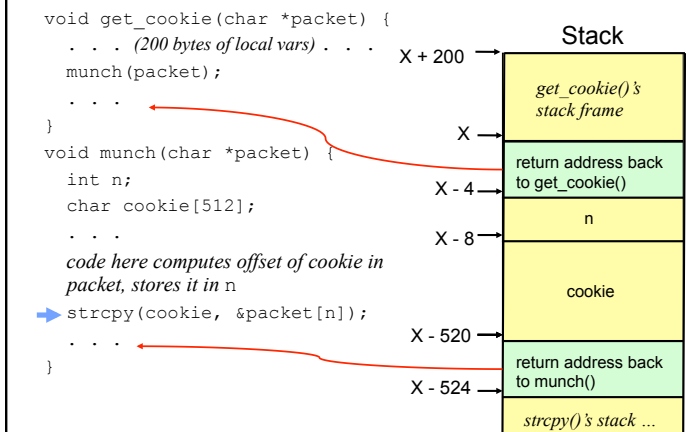


4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.18

### Example: Normal Execution

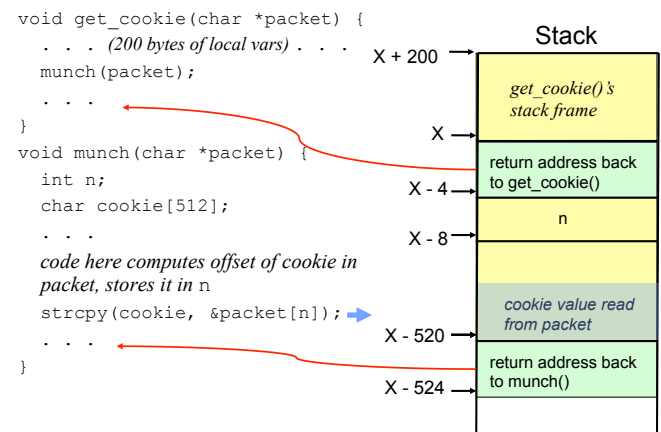


4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.19

### Example: Normal Execution

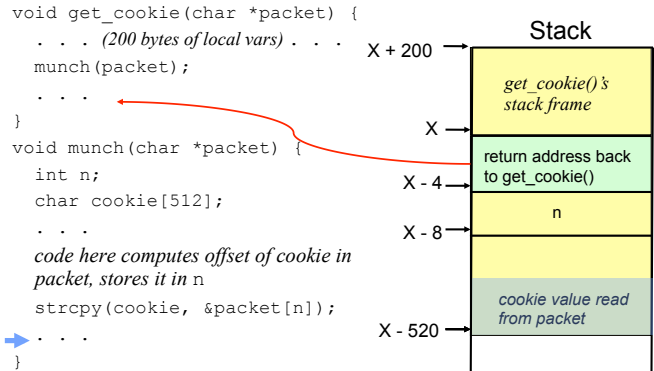


4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.20

### Example: Normal Execution

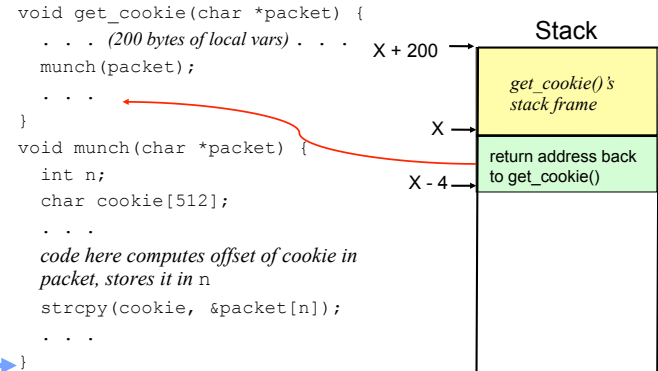


4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.21

### Example: Normal Execution

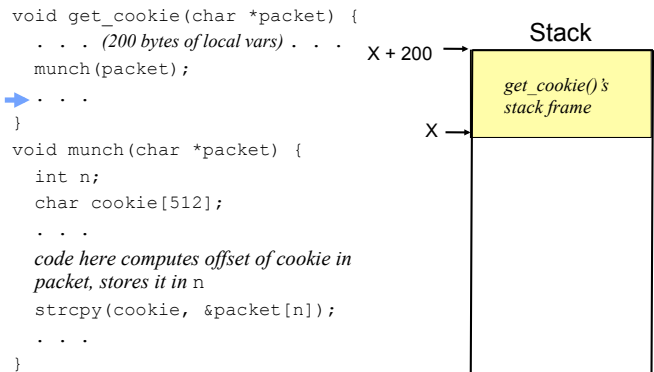


4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.22

### Example: Normal Execution

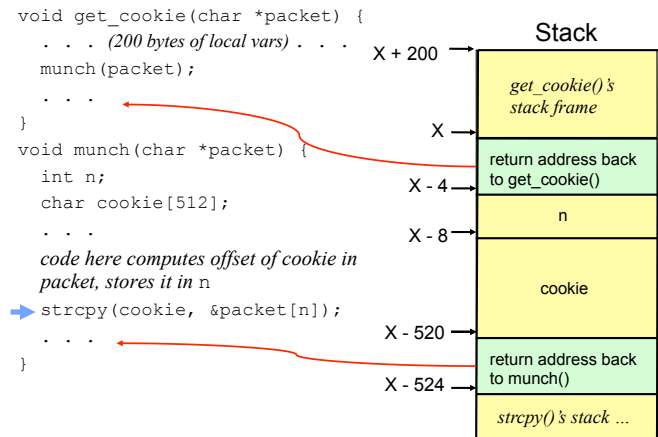


4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.23

### Example: Buffer Overflow

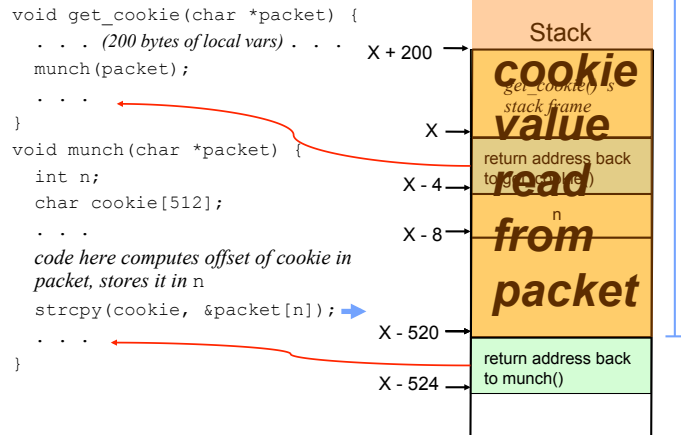


4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

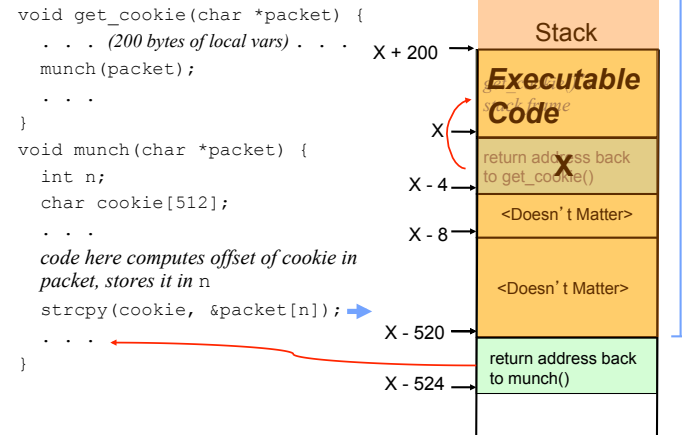
22.24

### Example: Buffer Overflow



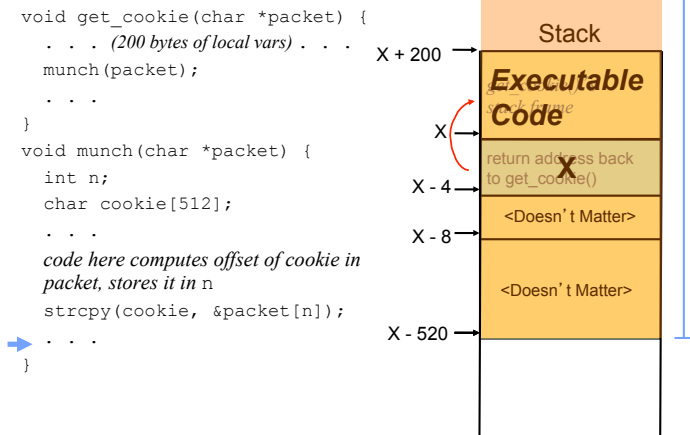
4/16/2012 Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012 22.25

### Example: Buffer Overflow



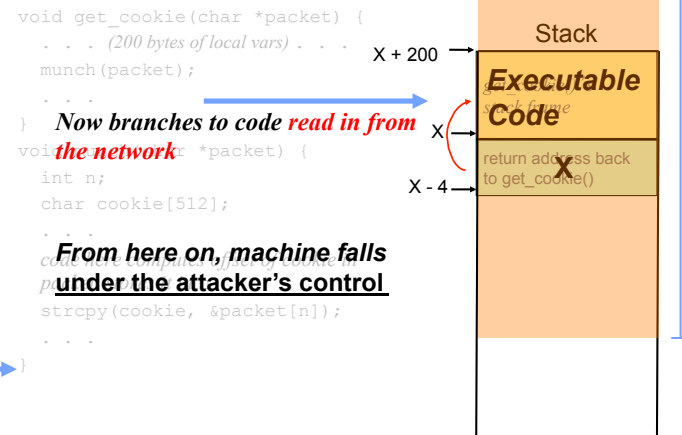
4/16/2012 Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012 22.26

### Example: Buffer Overflow



4/16/2012 Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012 22.27

### Example: Buffer Overflow



4/16/2012 Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012 22.28

## Automated Compromise: Worms

- When attacker compromises a host, they can instruct it to do **whatever they want**
- Instructing it to find more vulnerable hosts to repeat the process creates a worm: a program that **self-replicates** across a network
  - Often spread by picking 32-bit Internet addresses at random to probe ...
  - ... but this isn't fundamental
- As the worm repeatedly replicates, it grows *exponentially fast* because each copy of the worm works in parallel to find more victims

4/16/2012

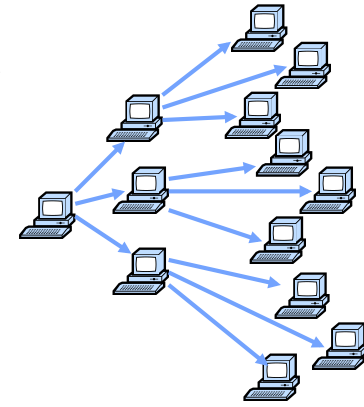
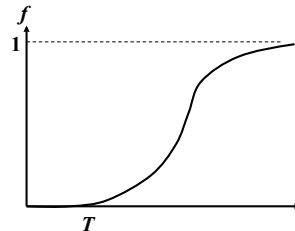
Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.29

## Worm Spreading

$$f = (e^{K(t-T)} - 1) / (1 + e^{K(t-T)})$$

- $f$  – fraction of hosts infected
- $K$  – rate at which one host can compromise others
- $T$  – start time of the attack



4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.30

## Worm Examples

- Morris worm (1988)
- Code Red (2001)
  - 369K hosts in 10 hours
- MS Slammer (January 2003)
- Theoretical worms
  - Zero-day exploit, efficient infection and propagation
  - 1M hosts in 1.3 sec
  - \$50B+ damage

4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.31

## Morris Worm (1988)

- Infect multiple types of machines (Sun 3 and VAX)
  - Was supposed to be benign: estimate size of Internet
- Used multiple security holes including
  - Buffer overflow in `fingerd`
  - Debugging routines in `sendmail`
  - Password cracking
- Intend to be benign but it had a bug
  - Fixed chance the worm wouldn't quit when reinfesting a machine → number of worm on a host built up rendering the machine unusable

4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.32



## Code Red Worm (2001)

- Attempts to connect to TCP port 80 (i.e., HTTP port) on a randomly chosen host
- If successful, the attacking host sends a crafted HTTP GET request to the victim, attempting to exploit a buffer overflow
- Worm “bug”: all copies of the worm use the same random generator and seed to scan new hosts
  - DoS attack on those hosts
  - Slow to infect new hosts
- 2<sup>nd</sup> generation of Code Red fixed the bug!
  - It spread much faster

4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.33

## MS SQL Slammer (January 2003)

- Uses UDP port 1434 to exploit a buffer overflow in MS SQL server
  - 376-bytes plus UDP and IP headers: one packet
- Effect
  - Generate massive amounts of network packets
  - Brought down as many as 5 of the 13 internet root name servers
- Others
  - The worm only spreads as an in-memory process: it never writes itself to the hard drive
    - » Solution: close UDP port on firewall and reboot

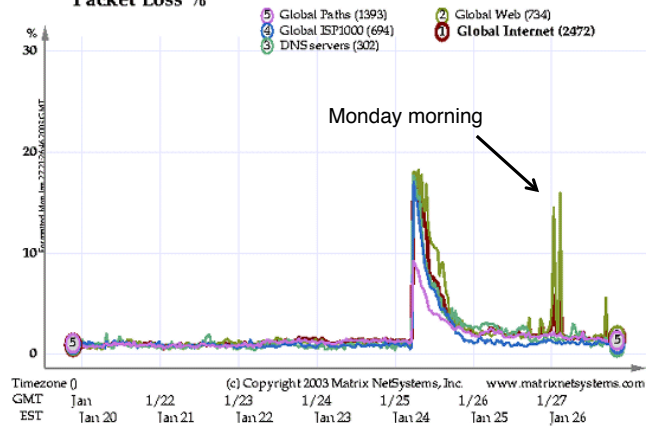
4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.34

## MS SQL Slammer (January 2003)

Packet Loss %



(From <http://www.f-secure.com/v-descs/mssqlm.shtml>)

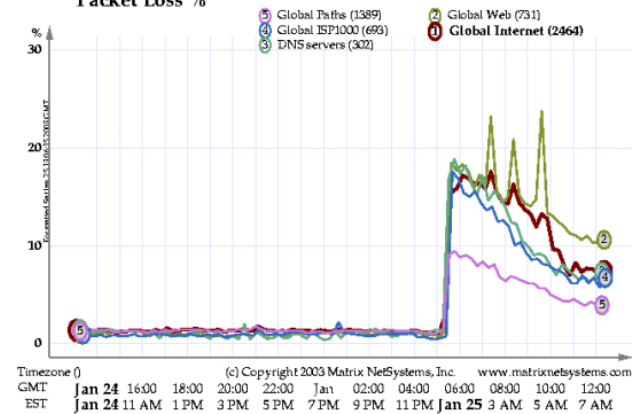
4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.35

## MS SQL Slammer (January 2003)

Packet Loss %



(From <http://www.f-secure.com/v-descs/mssqlm.shtml>)

4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.36

## Hall of Shame

- Software that have had many stack overflow bugs:
  - BIND (most popular DNS server)
  - RPC (Remote Procedure Call, used for NFS)
    - » NFS (Network File System), widely used at UCB
  - Sendmail (most popular UNIX mail delivery software)
  - IIS (Windows web server)
  - SNMP (Simple Network Management Protocol, used to manage routers and other network devices)

4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.37

## Administrivia

- Project 2 grades posted
- Project 4 design due Monday 4/23 at 11:59pm
- Break question: What techniques could we use to avoid buffer overflows?

### 5 minute break

4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.38

<p>UNCOMMON (NON-GIBBERISH) BASE WORD</p> <p>ORDER UNKNOWN</p> <p>Tr0ub4dor&amp;3</p> <p>CAPS? COMMON SUBSTITUTIONS NUMERAL PUNCTUATION</p> <p>(YOU CAN ADD A FEW MORE BITS TO ACCOUNT FOR THE FACT THAT THIS IS ONLY ONE OF A FEW COMMON FEATURES)</p>	<p>~28 BITS OF ENTROPY</p> <p><math>2^{28} = 3 \text{ DAYS AT } 1000 \text{ GUESSES/SEC}</math></p> <p>(PLAUSIBLE ATTACK ON A WEAK REMOTE WEB SERVICE: YES, CRAWLING AT STORAGE HEADS IS FASTER, BUT IT'S NOT WHAT THE AVERAGE USER SHOULD WORRY ABOUT.)</p> <p>DIFFICULTY TO GUESS: <b>EASY</b></p>	<p>WAS IT TROMBONE? NO, TROUBADOR, AND ONE OF THE 0s WAS A ZERO? AND THERE WAS SOME SYMBOL...</p> <p>DIFFICULTY TO REMEMBER: <b>HARD</b></p>
<p>correct horse battery staple</p> <p>FOUR RANDOM COMMON WORDS</p>	<p>~44 BITS OF ENTROPY</p> <p><math>2^{44} = 550 \text{ YEARS AT } 1000 \text{ GUESSES/SEC}</math></p> <p>DIFFICULTY TO GUESS: <b>HARD</b></p>	<p>THAT'S A BATTERY STAPLE. CORRECT?</p> <p>DIFFICULTY TO REMEMBER: <b>YOU'VE ALREADY MEMORIZED IT</b></p>

THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS. <https://xkcd.com/936/>

## Potential Solutions

- Don't write buggy software
  - Program defensively – validate all user-provided inputs
  - Use code checkers (slow, incomplete coverage)
- Use Type-safe Languages (Java, Perl, Python, ...)
- Eliminate unrestricted memory access of C/C++
- Use HW support for no-execute regions (stack, heap)
- Leverage OS architecture features
  - Address space randomization
  - Compartmentalize programs
    - » E.g., DNS server doesn't need total system access
- Add network firewalls

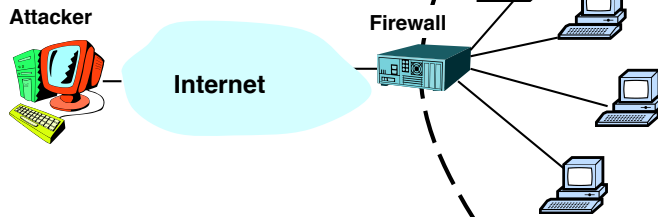
4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.40

## Firewall

- Security device whose goal is to prevent computers from outside to gain control to inside machines
- Hardware or software



4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

22.41

## Firewall (cont'd)

- Restrict traffic between Internet and devices (machines) behind it based on
  - Source address and port number
  - Payload
  - Stateful analysis of data
- Examples of rules
  - Block any external packets not for port 80
  - Block any email with an attachment
  - Block any external packets with an internal IP address
    - » Ingress filtering

4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

22.42

## Firewalls: Properties

- Easier to deploy firewall than secure all internal hosts
- Doesn't prevent user exploitation/social networking attacks
- Tradeoff between availability of services (firewall passes more ports on more machines) and security
  - If firewall is too restrictive, users will find way around it, thus compromising security
  - E.g., tunnel all services using port 80

4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

22.43

## Denial of Service

- Huge problem in current Internet
  - Major sites attacked: Yahoo!, Amazon, eBay, CNN, Microsoft
  - 12,000 attacks on 2,000 organizations in 3 weeks
  - Some more than 600,000 packets/second
    - » More than 192Mb/s
  - Almost all attacks launched from compromised hosts
- General Form
  - Prevent legitimate users from gaining service by overloading or crashing a server
  - E.g., SYN attack

4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

22.44

## Affect on Victim

- Buggy implementations allow unfinished connections to eat all memory, leading to crash
- Better implementations limit the number of unfinished connections
  - Once limit reached, new SYNs are dropped
- Affect on victim's users
  - Users can't access the targeted service on the victim because the unfinished connection queue is full → DoS

4/16/2012

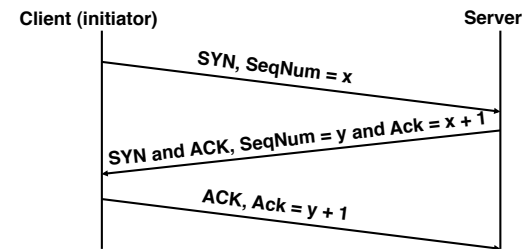
Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.45

## SYN Attack

### (Recap: 3-Way Handshaking)

- Goal: agree on a set of parameters: the start sequence number for each side
  - Starting sequence numbers are random.



4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.46

## SYN Attack

- Attacker: send at max rate TCP SYN with random spoofed source address to victim
  - Spoofing: use a different source IP address than own
  - Random spoofing allows one host to pretend to be many
- Victim receives many SYN packets
  - Send SYN+ACK back to spoofed IP addresses
  - Holds some memory until 3-way handshake completes
    - » Usually never, so victim times out after long period (e.g., 3 minutes)

4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.47

## Solution: SYN Cookies

- Server: send SYN-ACK with sequence number  $y$ , where
  - $y = H(\text{client\_IP\_addr}, \text{client\_port})$
  - $H()$ : one-way hash function
- Client: send ACK containing  $y+1$
- Server:
  - verify if  $y = H(\text{client\_IP\_addr}, \text{client\_port})$
  - If verification passes, allocate memory
- Note: server doesn't allocate any memory if the client's address is spoofed

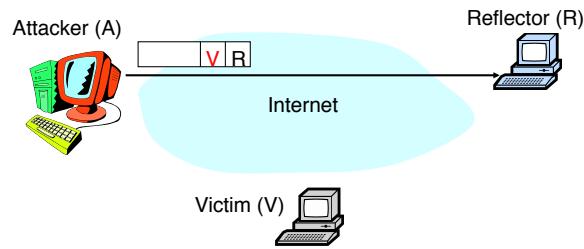
4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.48

## Other Denial-of-Service Attacks

- Reflection
  - Cause one non-compromised host to attack another
  - E.g., host A sends DNS request or TCP SYN with source V to server R. R sends reply to V



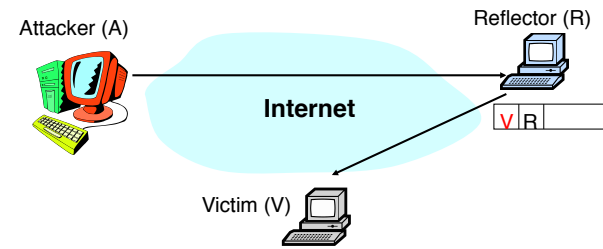
4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.49

## Other Denial-of-Service Attacks

- Reflection
  - Cause one non-compromised host to attack another
  - E.g., host A sends DNS request or TCP SYN with source V to server R. R sends reply to V



4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.50

## Other Denial-of-Service Attacks

- DNS
  - Ping flooding attack on DNS root servers (October 2002)
  - 9 out of 13 root servers brought down
  - Relatively small impact (why?)

4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.51

## Identifying and Stop Attacking Machines

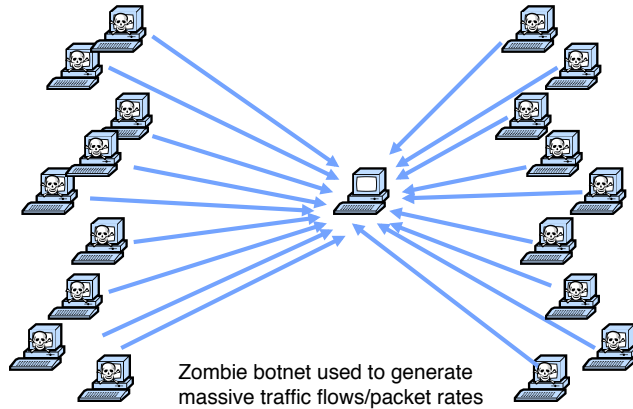
- Develop techniques for defeating spoofed source addresses
- Egress filtering
  - A domain's border router drop outgoing packets which do not have a valid source address for that domain
  - If universal, could abolish spoofing
- IP Traceback
  - Routers probabilistically tag packets with an identifier
  - Destination can infer path to true source after receiving enough packets

4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.52

## Distributed Denial-of-Service Attacks



4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.53

## Summary

- Security is one of the biggest problem today
- Host Compromise
  - Poorly written software
  - Partial solutions: better OS security architecture, type-safe languages, firewalls
- Denial-of-Service
  - No easy solution: DoS can happen at many levels
  - DDoS attacks can be very difficult to defeat

4/16/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring2012

22.54