

# CS162 Operating Systems and Systems Programming Lecture 22

## Security (II)

November 19, 2012

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

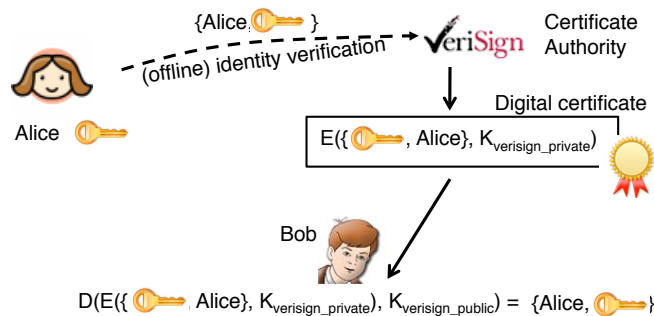
11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

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



11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.3

## Authentication: Passwords

- Shared secret between two parties
- Since only user knows password, someone types correct password  $\Rightarrow$  must be user typing it
- Very common technique
- System must keep copy of secret to check against passwords
  - What if malicious user gains access to list of passwords?
    - » Need to obscure information somehow
  - Mechanism: utilize a transformation that is difficult to reverse without the right key (e.g. encryption)



11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.4

## Passwords: Secrecy



- Example: UNIX `/etc/passwd` file
  - `passwd`→one way hash
  - System stores only encrypted version, so OK even if someone reads the file!
  - When you type in your password, system compares encrypted version

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.5

## Passwords: How easy to guess?

- Three common ways of compromising passwords
- Password Guessing:
  - Often obvious passwords 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.)

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.6

## 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  $26^5=10\text{million}$  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!

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.7

## 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 uses 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)

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.8

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

- Technique 2: Require more complex passwords
  - Make people use at least 8-character passwords with upper-case, lower-case, and numbers
    - »  $70^8 = 6 \times 10^{14} = 6 \text{ million seconds} = 69 \text{ days} @ 0.01 \mu\text{s/check}$
  - 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

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.9

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

- Technique 4: Assign very long passwords/passphrases
  - Can have more entropy (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



11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.10

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

- 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
- Technique 6: Replace password with Biometrics
  - Use of one or more intrinsic physical or behavioral traits to identify someone
  - Examples: fingerprint reader, palm reader, retinal scan



11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.11

## Rest of 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

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.12

## 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

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.13

## 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

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.14

## 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)

**From:** Halmark Greetings [mailto:greet@halmark-greetings.com]  
**Date:** Thursday, November 18, 2010 9:48 PM  
**To:** Recipients  
**Subject:** You have received a greeting!

You have received a virtual greeting card from Mary!

You can view your greeting card visiting the following link:

<http://www.halmark-greetings.com/greetings/IKDFIUERGHIUER>

If you can't click on the above link, you can also visit Halmark Greetings directly at <http://www.halmark-greetings.com/> and enter your greeting card code, which is: IKDFIUERGHIUER.

Halmark Greetings, the greeting that always puts a smile on your face.

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

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.15

## 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

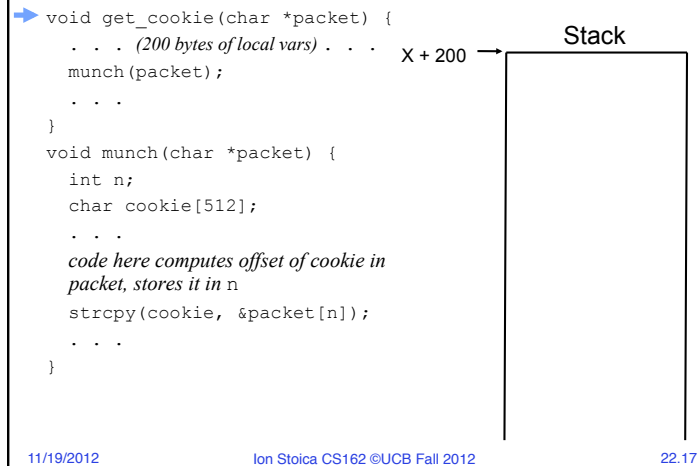
```
void get_cookie(char *packet) {  
    . . . (200 bytes of local vars) . . .  
    munch(packet);  
    . . .  
}  
  
void munch(char *packet) {  
    int n;  
    char cookie[512];  
    . . .  
    code here computes offset of cookie in  
    packet, stores it in n  
    strcpy(cookie, &packet[n]);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

11/19/2012

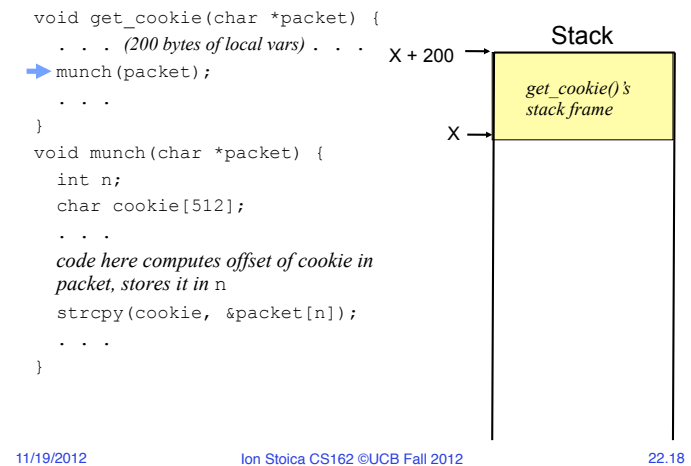
Ion Stoica CS162 ©UCB Fall 2012

22.16

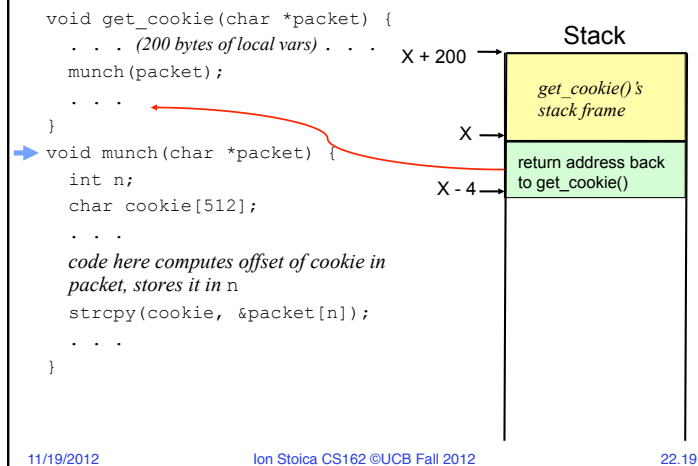
### Example: Normal Execution



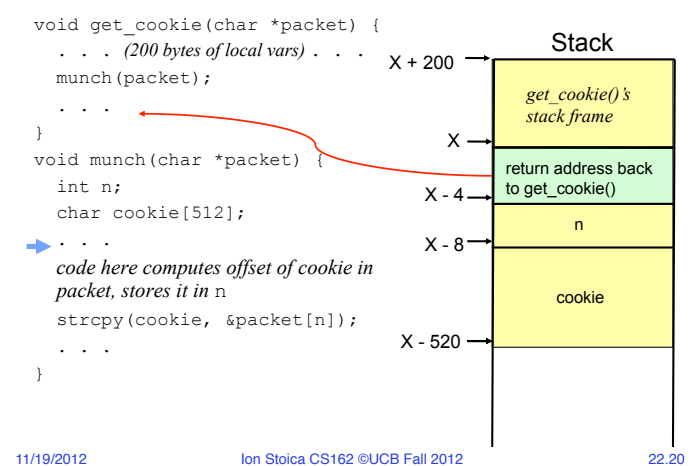
### Example: Normal Execution



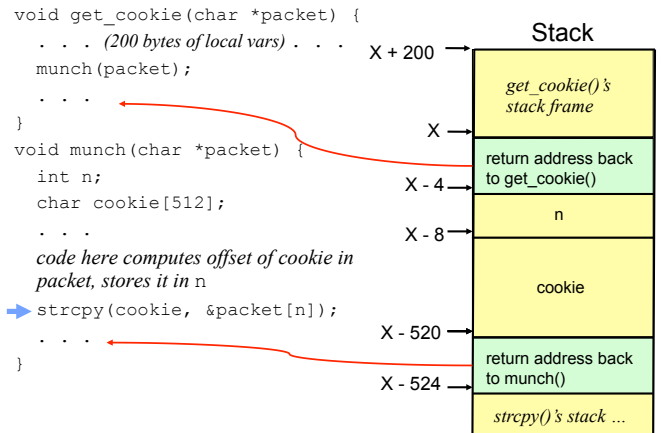
### Example: Normal Execution



### Example: Normal Execution



### Example: Normal Execution

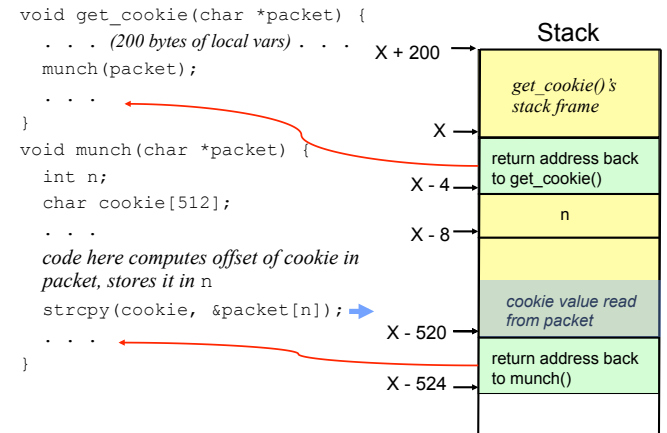


11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.21

### Example: Normal Execution

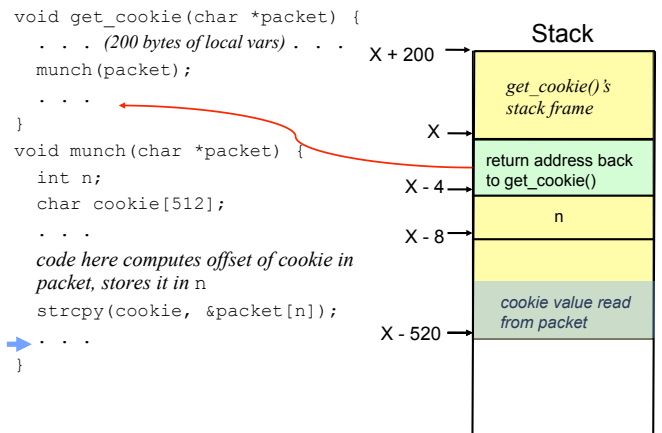


11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.22

### Example: Normal Execution

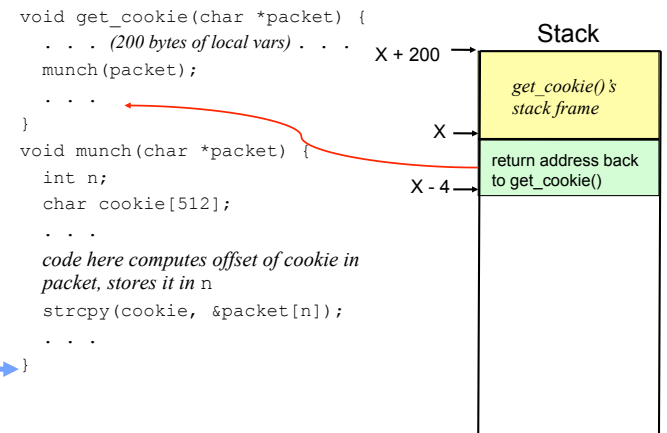


11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.23

### Example: Normal Execution

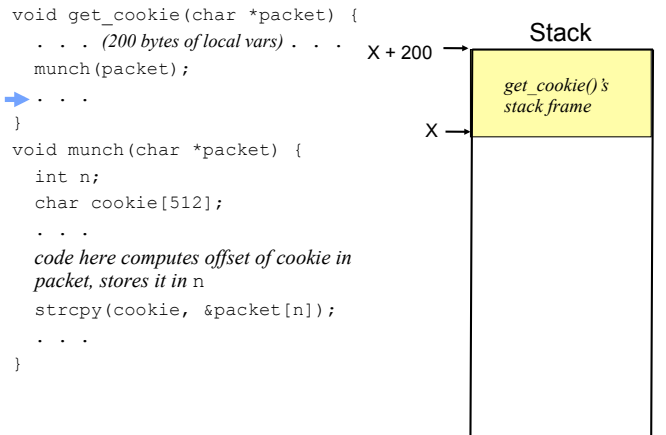


11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.24

### Example: Normal Execution

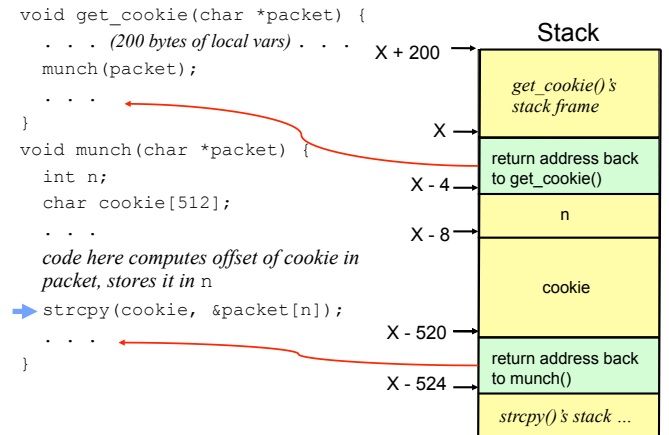


11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.25

### Example: Buffer Overflow

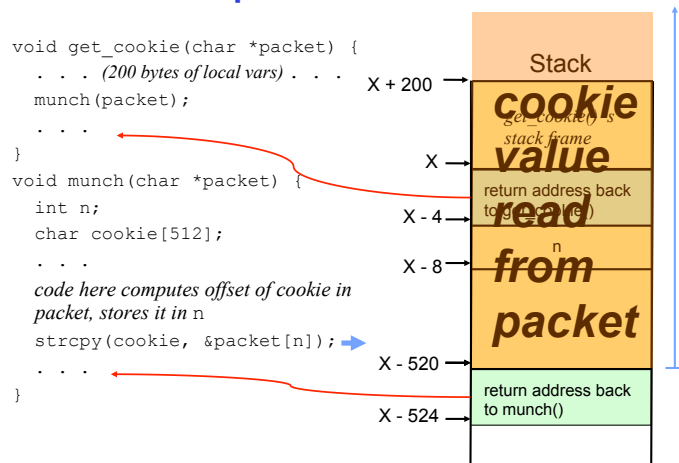


11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.26

### Example: Buffer Overflow

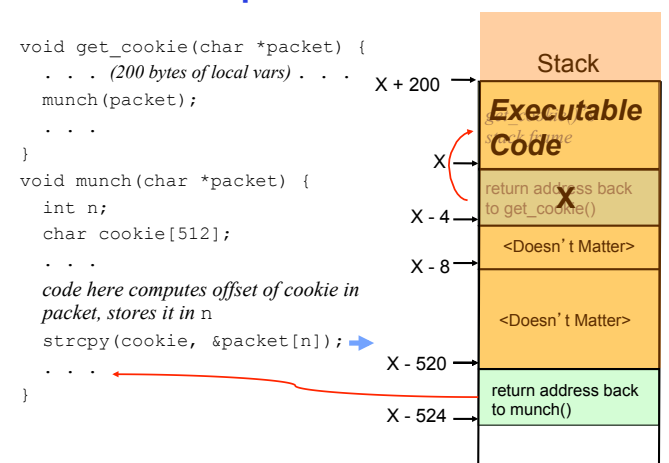


11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.27

### Example: Buffer Overflow



11/19/2012

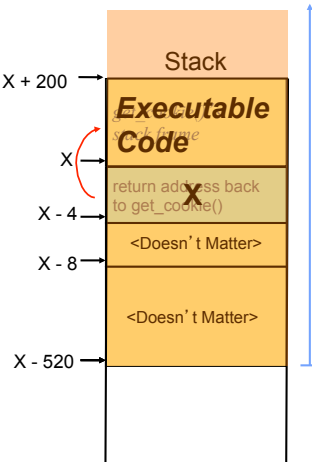
Ion Stoica CS162 ©UCB Fall 2012

22.28

## Example: Buffer Overflow

```
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]);
    . . .
}
```



11/19/2012

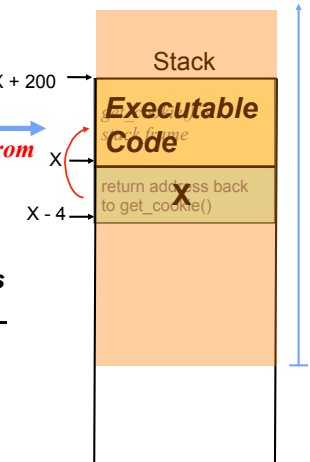
Ion Stoica CS162 ©UCB Fall 2012

22.29

## Example: Buffer Overflow

```
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]);
    . . .
}
```



Now branches to code read in from the network

From here on, machine falls under the attacker's control

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.30

## 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

11/19/2012

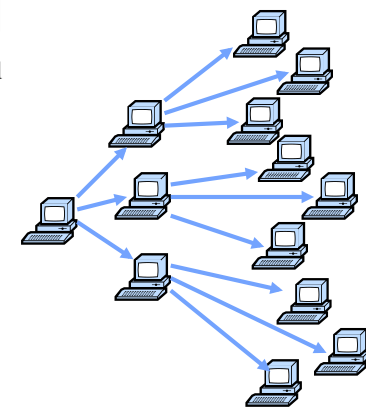
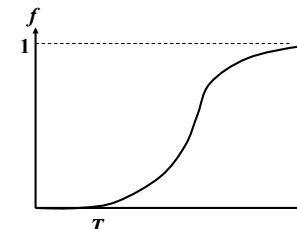
Ion Stoica CS162 ©UCB Fall 2012

22.31

## 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



11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.32

## 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

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.33

## 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 reinfecting a machine → number of worm on a host built up rendering the machine unusable

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.34

## 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

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.35

## MS SQL Slammer (January 2003)

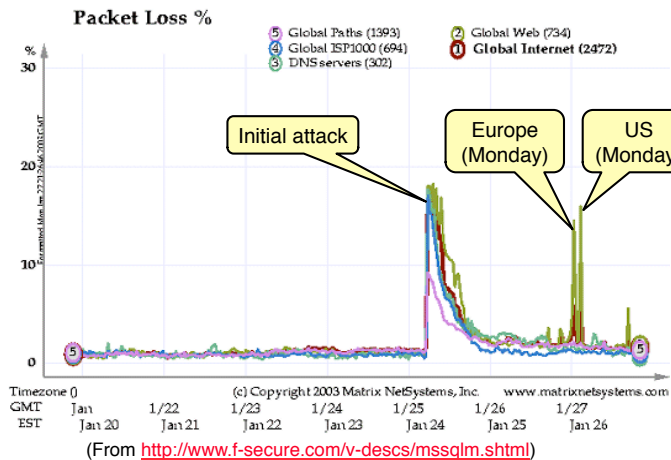
- Uses UDP port 1434 to exploit a buffer overflow in MS SQL server
  - 376-bytes plus UDP and IP headers: one packet
- Effect
  - Generate massive amounts of network packets
  - Brought down as many as 5 of the 13 internet root name servers
- Others
  - The worm only spreads as an in-memory process: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

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.36

## MS SQL Slammer (January 2003)



11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.37

## 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)

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.38

## 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
  - Compartmentalize programs
    - » E.g., DNS server doesn't need total system access
- Add network firewalls

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.39

## Announcements

- Project 4: deadlines pushed by one day
  - Initial design due on **Tuesday, Nov 27**
  - Code due on **Thursday, Dec 6**
  - Final design and evaluations due on **Friday, Dec 7**
- Review for final exam: **Wednesday, Dec 5, 6-9pm**
- Next Monday I'll be out:
  - Lecture will be given by Ali Ghodsi (Researcher at Berkeley and Professor at KTH, Sweden)

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.40

## 5min Break

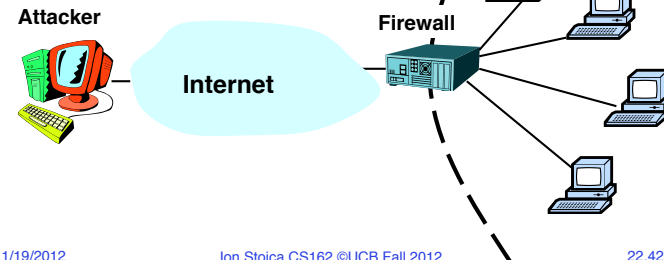
11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.41

## Firewall

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



11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.42

## 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 (i.e., HTTP port)
  - Block any email with an attachment
  - Block any external packets with an internal IP address
    - » Ingress filtering

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.43

## 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

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.44

## 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
  - 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

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.45

## 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

11/19/2012

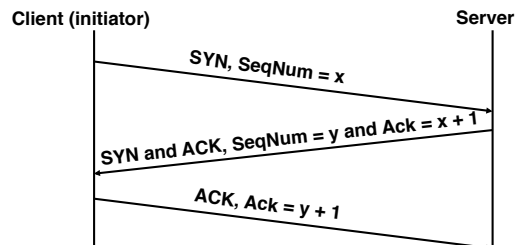
Ion Stoica CS162 ©UCB Fall 2012

22.46

## 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.



11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.47

## 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)

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.48

## 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

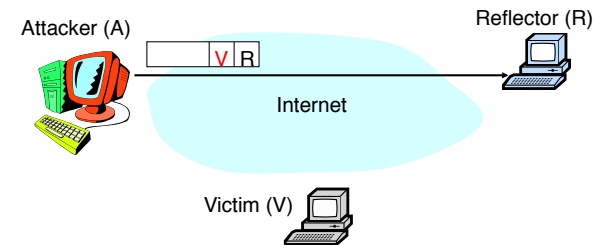
11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

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



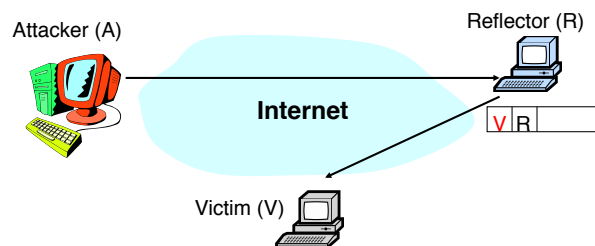
11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.50

## 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



11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.51

## Identifying and Stop Attacking Machines

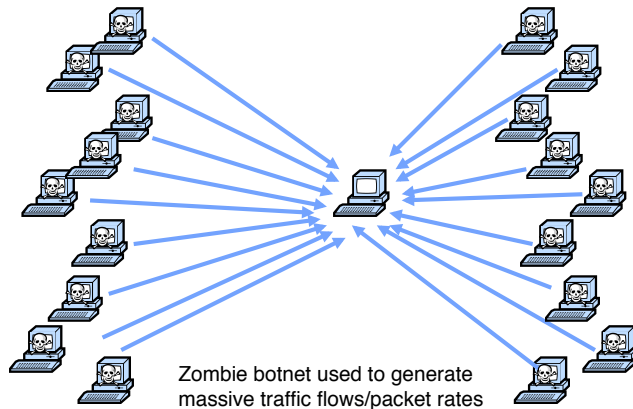
- Develop techniques for defeating spoofed source addresses
- Egress filtering
  - A domain's border router drop outgoing packets which do not have a valid source address for that domain
  - If universal, could abolish spoofing
- IP Traceback
  - Routers probabilistically tag packets with an identifier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

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.52

## Distributed Denial-of-Service Attacks



11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

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

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.54

## Additional Notes on Public Key Cryptography (Not required for Final Exam)

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.55

## Generating Public and Private Keys

- Choose two large prime numbers  $p$  and  $q$  ( $\sim 256$  bit long) and multiply them:  $n = p * q$
- Choose **encryption** key  $e$  such that  $e$  and  $(p-1)*(q-1)$  are relatively prime
- Compute **decryption** key  $d$  as
 
$$d = e^{-1} \bmod ((p-1)*(q-1))$$
 (equivalent to  $d * e = 1 \bmod ((p-1)*(q-1))$ )
- **Public** key consist of pair  $(n, e)$
- **Private** key consists of pair  $(d, n)$

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.56

## RSA Encryption and Decryption

- Encryption of message block  $m$ :
  - $c = m^e \bmod n$
- Decryption of ciphertext  $c$ :
  - $m = c^d \bmod n$

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.57

## Example (1/2)

- Choose  $p = 7$  and  $q = 11 \rightarrow n = p \cdot q = 77$
- Compute encryption key  $e$ :  $(p-1) \cdot (q-1) = 6 \cdot 10 = 60 \rightarrow$  chose  $e = 13$  (13 and 60 are relatively prime numbers)
- Compute decryption key  $d$  such that  $13 \cdot d = 1 \bmod 60 \rightarrow d = 37$  ( $37 \cdot 13 = 481$ )

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.58

## Example (2/2)

- $n = 77$ ;  $e = 13$ ;  $d = 37$
- Send message block  $m = 7$
- Encryption:  $c = m^e \bmod n = 7^{13} \bmod 77 = 35$
- Decryption:  $m = c^d \bmod n = 35^{37} \bmod 77 = 7$

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

22.59

## Properties

- Confidentiality
- A receiver  $A$  computes  $n, e, d$ , and sends out  $(n, e)$ 
  - Everyone who wants to send a message to  $A$  uses  $(n, e)$  to encrypt it
- How difficult is to recover  $d$ ? (Someone that can do this can decrypt any message sent to  $A$ !)
- Recall that
$$d = e^{-1} \bmod ((p-1) \cdot (q-1))$$
- So to find  $d$ , you need to find primes factors  $p$  and  $q$ 
  - This is provable hard

11/19/2012

Ion Stoica CS162 ©UCB Fall 2012

60

22.60