

CS162 Operating Systems and Systems Programming Lecture 22

Security (II)

April 22, 2013
Anthony D. Joseph
<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

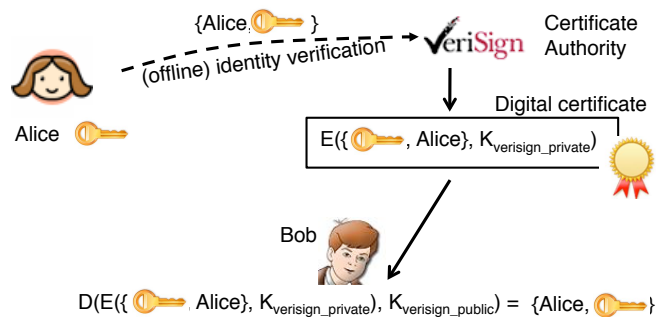
4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

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



4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.3

Goals for Today

- 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

4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.4

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

4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.5

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

4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.6

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

4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.7

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

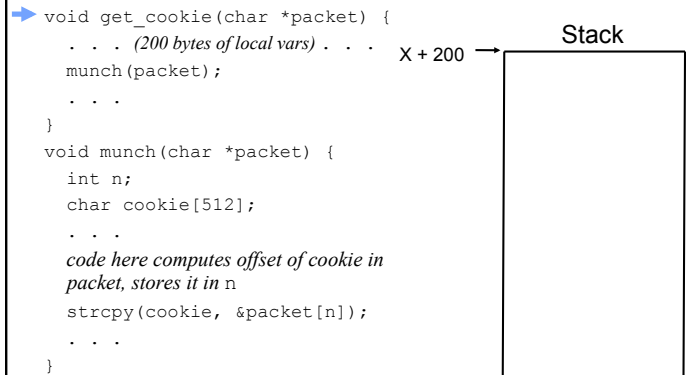
```
void get_cookie(char *packet) {  
    . . . (200 bytes of local vars) . . .  
    munch(packet);  
    . . .  
}  
  
void munch(char *packet) {  
    int n;  
    char cookie[512];  
    . . .  
    code here computes offset of cookie in  
    packet, stores it in n  
    strcpy(cookie, &packet[n]);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

4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.8

Example: Normal Execution

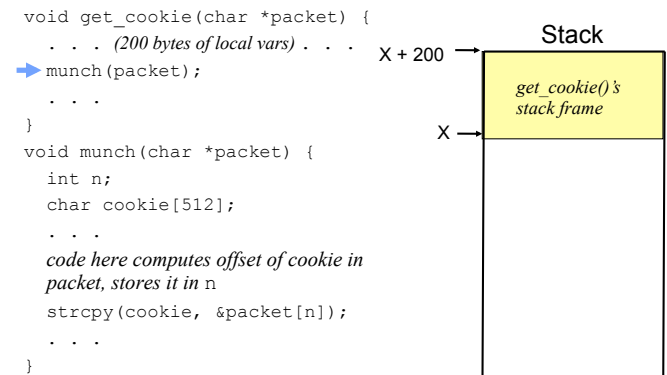


4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.9

Example: Normal Execution

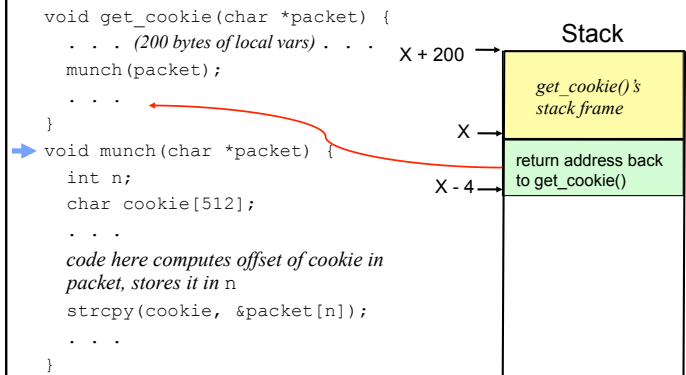


4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.10

Example: Normal Execution

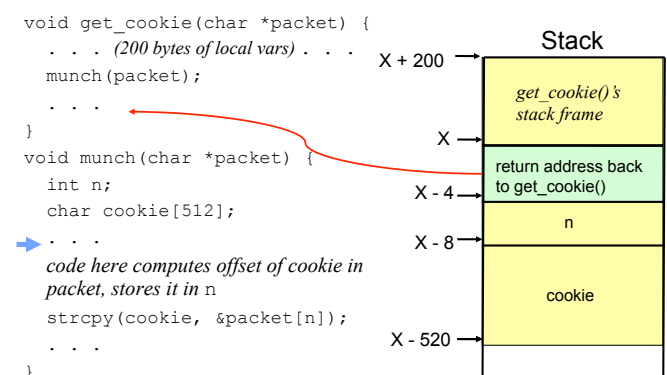


4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.11

Example: Normal Execution

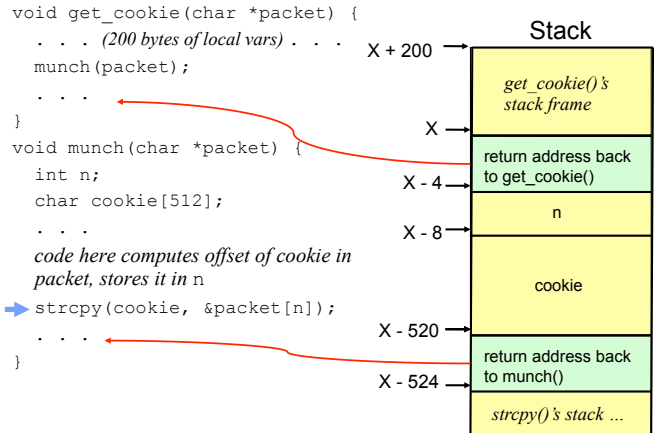


4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.12

Example: Normal Execution

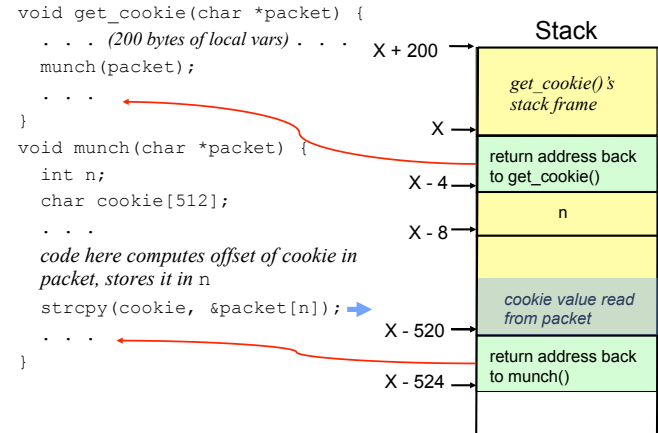


4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.13

Example: Normal Execution

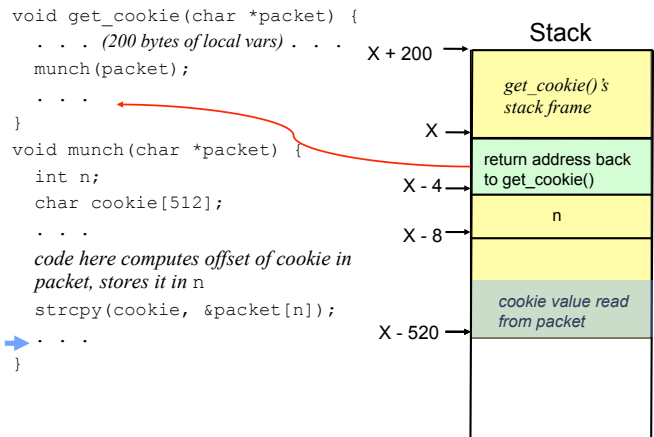


4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.14

Example: Normal Execution

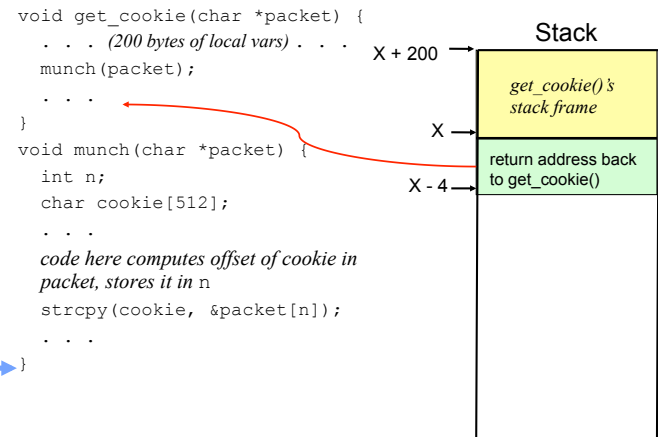


4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.15

Example: Normal Execution

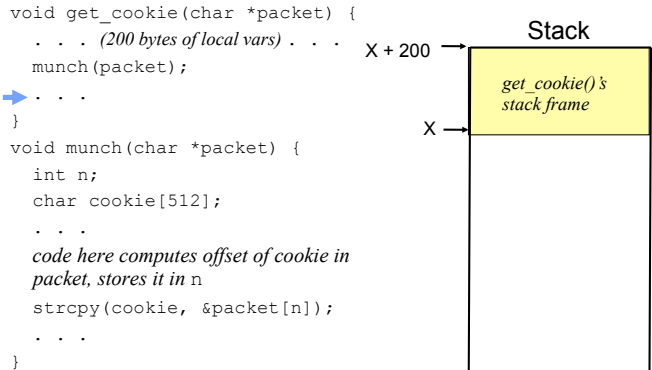


4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.16

Example: Normal Execution

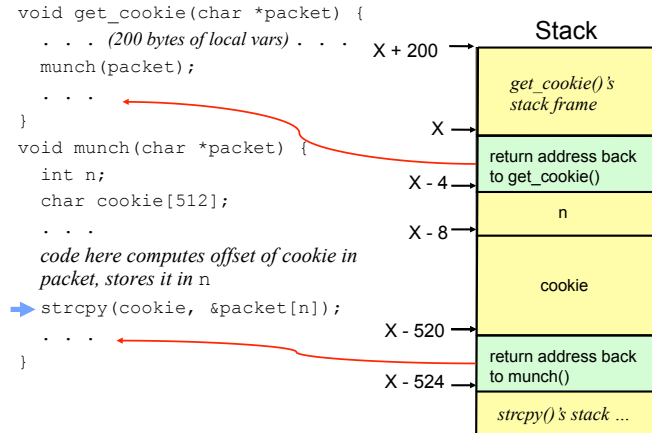


4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.17

Example: Buffer Overflow

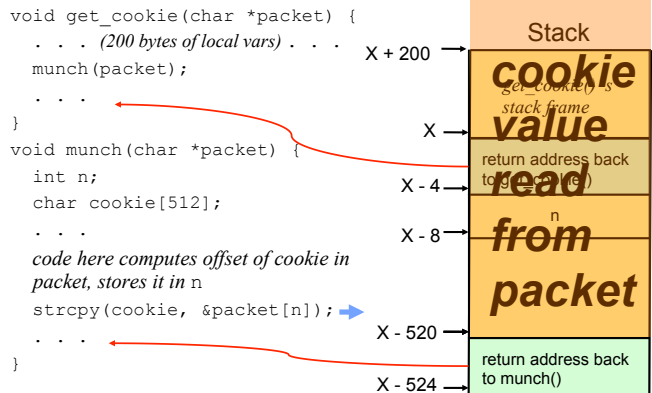


4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.18

Example: Buffer Overflow

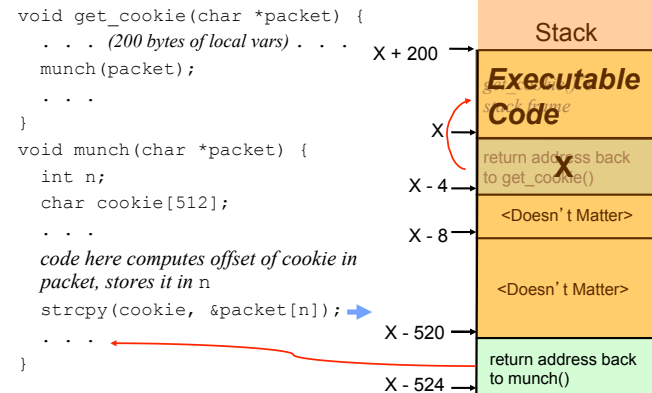


4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.19

Example: Buffer Overflow



4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

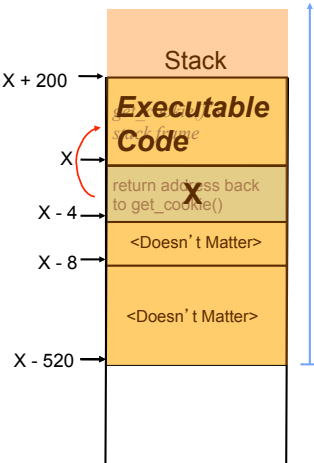
22.20

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]);
    . . .
}

```



4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

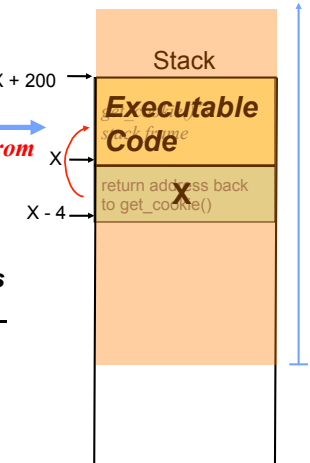
22.21

Example: Buffer Overflow

```

void get_cookie(char *packet) {
    . . . (200 bytes of local vars) . . .
    munch(packet);
    . . .
}
void the network *packet) {
    int n;
    char cookie[512];
    . . .
    From here on, machine falls
under the attacker's control
    strcpy(cookie, &packet[n]);
    . . .
}

```



4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.22

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

4/22/2013

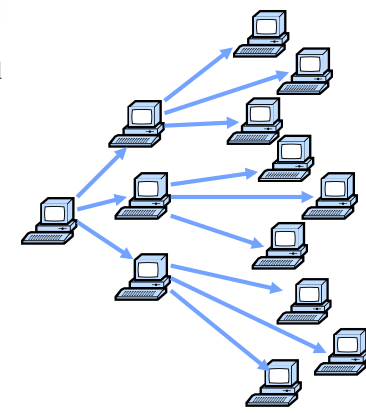
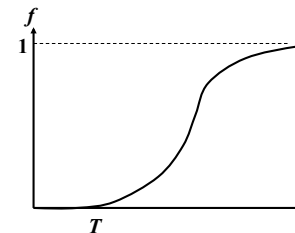
Anthony D. Joseph CS162 ©UCB Spring 2013

22.23

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



4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.24

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

4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.25

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

4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.26

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
- 2nd generation of Code Red fixed the bug!
 - It spread much faster

4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.27

MS SQL Slammer (January 2003)

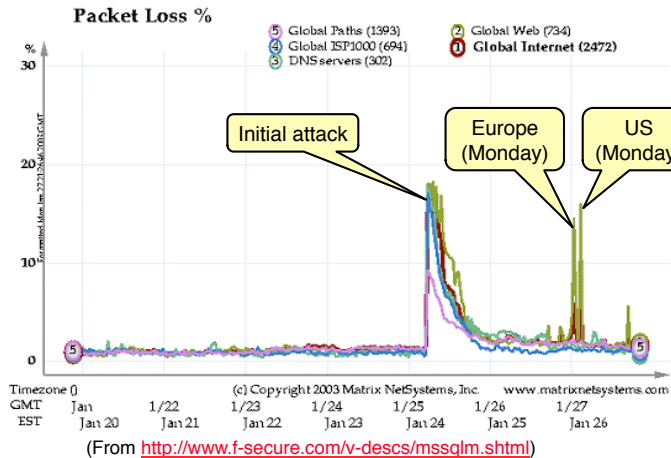
- Uses UDP port 1434 to exploit a buffer overflow in MS SQL server
 - 376-bytes plus UDP and IP headers: one packet
- Effect
 - Generate massive amounts of network packets
 - Brought down as many as 5 of the 13 internet root name servers
- Others
 - The worm only spreads as an in-memory process: it never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

4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.28

MS SQL Slammer (January 2003)



4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.29

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)

4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.30

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

4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.31

Quiz 22.1: Security

- Q1: True _ False _ A digital certificate provides a binding between a host's identity and their public key
- Q2: True _ False _ A server must store a user's password in plaintext form so it can be checked against a submitted password
- Q3: True _ False _ Two-factor authentication uses a second authentication password
- Q4: True _ False _ Worms require human intervention to propagate
- Q5: True _ False _ Using a type-safe language eliminates the risk of buffer overflows

4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.32

Administrivia

- Next lecture changed to RPC and distributed file systems
- Final Exam – **Friday 5/17, 8-11am in 1 Pimentel**
 - Review: Monday 5/6, 2-5pm in 100 Lewis Hall
 - All material from the course
 - » With slightly more focus on second half, but you are still responsible for all the material
 - Two sheets of notes, both sides
 - Dumb calculator allowed
- Should be working on Project 4
 - Last one!

4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.33

5min Break

4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.34

Quiz 22.1: Security

- Q1: True False A digital certificate provides a binding between a host's identity and their public key
- Q2: True False A server must store a user's password in plaintext form so it can be checked against a submitted password
- Q3: True False Two-factor authentication uses a second authentication password
- Q4: True False Worms require human intervention to propagate
- Q5: True False Using a type-safe language eliminates the risk of buffer overflows

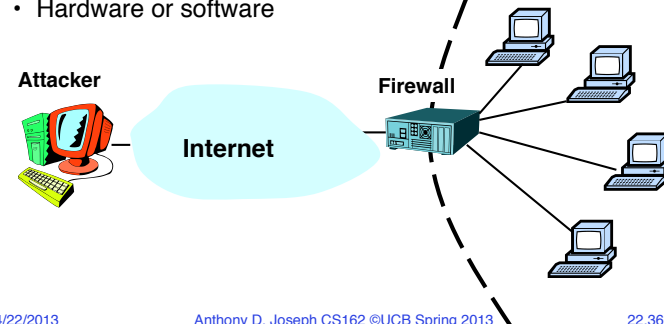
4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.35

Firewall

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



4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.36

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

4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.37

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

4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.38

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 or 300 Gb/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

4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.39

Effect 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
- Effect on victim's users
 - Users can't access the targeted service on the victim because the unfinished connection queue is full → DoS

4/22/2013

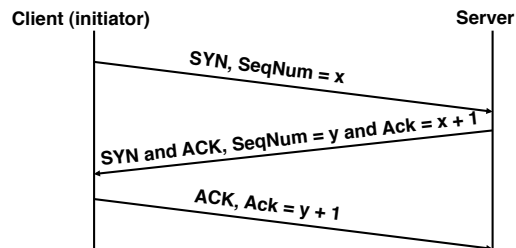
Anthony D. Joseph CS162 ©UCB Spring 2013

22.40

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.



4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.41

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)

4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.42

Solution: SYN Cookies

- Server: send SYN-ACK with sequence number y, where
 - $y = \text{HMAC}(\text{client_IP_addr}, \text{client_port}, \text{server_key})$
 - HMAC(): Hash Message Authentication Code
- Client: send ACK containing y+1
- Server:
 - verify if $y = \text{HMAC}(\text{client_IP_addr}, \text{client_port}, \text{server_key})$
 - If verification passes, allocate memory
- Note: server doesn't allocate any memory if the client's address is spoofed

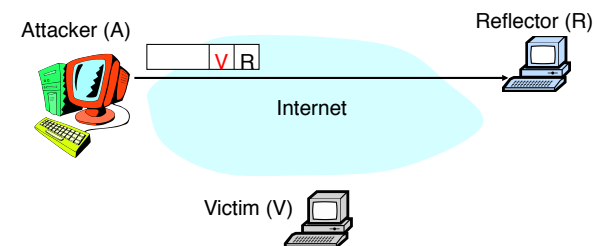
4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.43

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



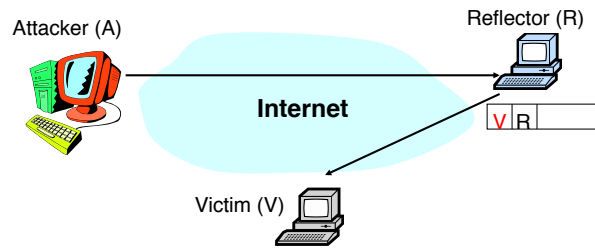
4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.44

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



4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.45

Identifying and Stop Attacking Machines

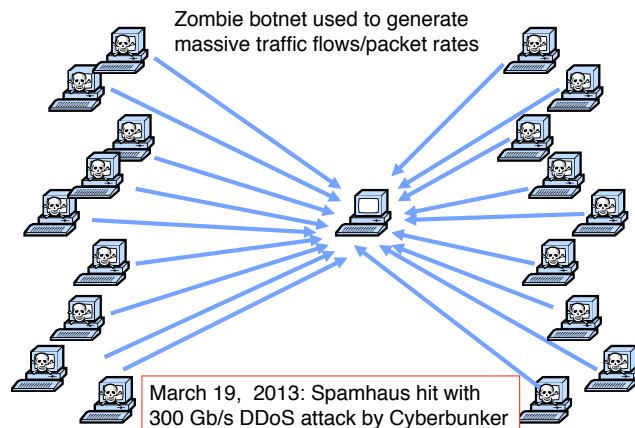
- Develop techniques for defeating spoofed source addresses
- Egress filtering
 - A domain's border router drop outgoing packets which do not have a valid source address for that domain
 - If universal, could abolish spoofing
- IP Traceback
 - Routers probabilistically tag packets with an identifier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

4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.46

Distributed Denial-of-Service Attacks



4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.47

Stepping Stone Compromise

- Today's most sophisticated attacks
 - Multi-step/compromise attack
- RSA SecurID token
 - 2-factor authentication device
 - Code changes every few seconds
 - Data on codes stolen in March 2011



- 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/>

4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.48

Advanced Persistent Threats

4/22/2013

<http://blogs.rsa.com/rivner/anatomy-of-an-attack/>

22.49

Advanced Persistent Threats

1

Phishing and Zero day attack

A handful of users are targeted by two phishing attacks; one user opens Zero day payload (CVE-02011-0609)



4

<http://blogs.rsa.com/rivner/anatomy-of-an-attack/>

Advanced Persistent Threats

1

Phishing and Zero day attack

A handful of users are targeted by two phishing attacks; one user opens Zero day payload (CVE-02011-0609)



4

<http://blogs.rsa.com/rivner/anatomy-of-an-attack/>

2

Back door

The user machine is accessed remotely by Poison Ivy tool



Advanced Persistent Threats

1

Phishing and Zero day attack

A handful of users are targeted by two phishing attacks; one user opens Zero day payload (CVE-02011-0609)



4

<http://blogs.rsa.com/rivner/anatomy-of-an-attack/>

2

Back door

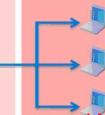
The user machine is accessed remotely by Poison Ivy tool

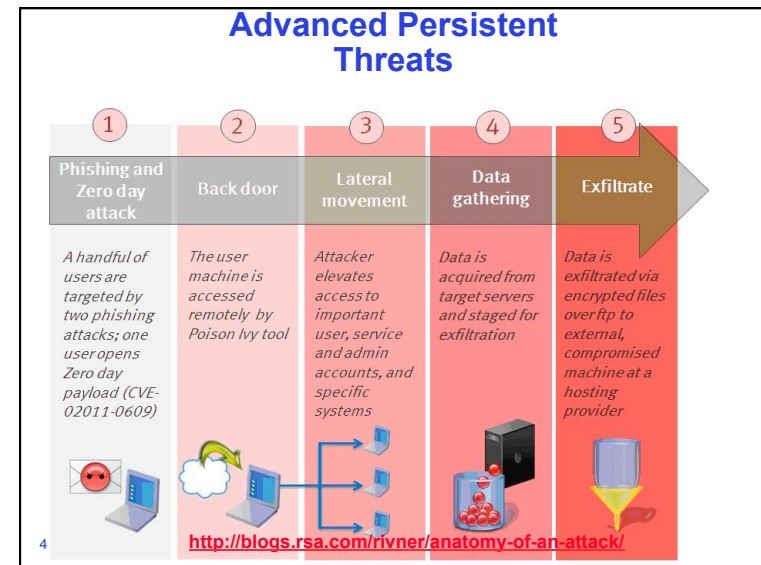
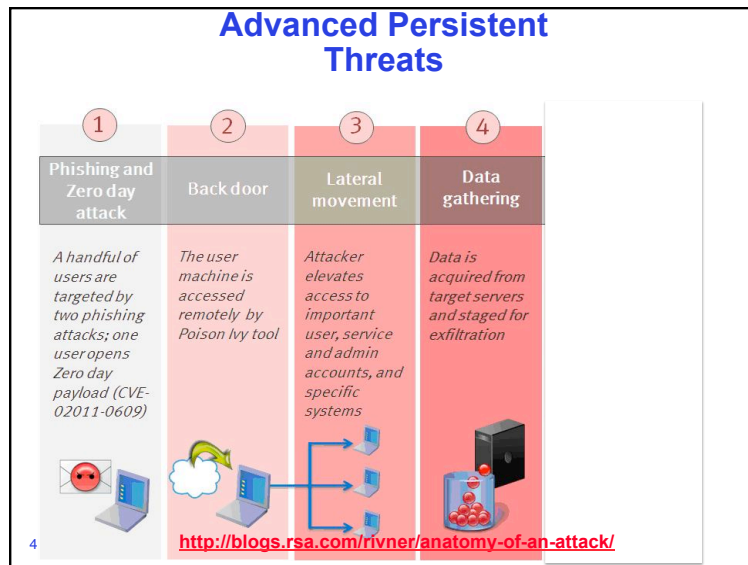


3

Lateral movement

Attacker elevates access to important user, service and admin accounts, and specific systems





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

4/22/2013 Anthony D. Joseph CS162 ©UCB Spring 2013 22.55

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

4/22/2013 Anthony D. Joseph CS162 ©UCB Spring 2013 22.56

Generating Public and Private Keys

- Choose two large prime numbers p and q (>1500 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
 $d = e^{-1} \text{ mod } ((p-1)*(q-1))$
(equivalent to $d * e = 1 \text{ mod } ((p-1)*(q-1))$)
- **Public** key consist of pair (n, e)
- **Private** key consists of pair (d, n)

4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.57

RSA Encryption and Decryption

- Encryption of message block m :
– $c = m^e \text{ mod } n$
- Decryption of ciphertext c :
– $m = c^d \text{ mod } n$

4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.58

Example (1/2)

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

4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.59

Example (2/2)

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

4/22/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

22.60

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
$$d = e^{-1} \text{ mod } ((p-1)*(q-1))$$
- So to find d , you need to find primes factors p and q
 - This is provable hard