Lecture 26: Security

Prof. Alan Mislove  (amislove@ccs.neu.edu)

Slides used with permissions from Edward W. Knightly,
T. S. Eugene Ng, Ion Stoica, Hui Zhang
Basic Security Requirements
Basic Security Requirements

• Authentication
  – Ensures that the sender and the receiver are who they are claiming to be
Basic Security Requirements

• Authentication
  – Ensures that the sender and the receiver are who they are claiming to be

• Data integrity
  – Ensure that data is not changed from source to destination
Basic Security Requirements

• Authentication
  – Ensures that the sender and the receiver are who they are claiming to be

• Data integrity
  – Ensure that data is not changed from source to destination

• Confidentiality
  – Ensures that data is read only by authorized users
Basic Security Requirements

• Authentication
  – Ensures that the sender and the receiver are who they are claiming to be

• Data integrity
  – Ensure that data is not changed from source to destination

• Confidentiality
  – Ensures that data is read only by authorized users

• This is not a crypto course, so we will just skim the surface of the crypto algorithms to give you a rough idea
Cryptographic Algorithms

• Security foundation: cryptographic algorithms
  – Secret key cryptography, e.g. Data Encryption Standard (DES)
  – Public key cryptography, e.g. RSA algorithm
  – Message digest, e.g. MD5
Symmetric Key

- Both the sender and the receiver use the same secret keys
Public-Key Cryptography: RSA (Rivest, Shamir, and Adleman)

- Sender uses a **public** key
  - Advertised to everyone
- Receiver uses a **private** key

**plaintext**

![Diagram](image.png)
Message Digest (MD) 5

- Can provide data integrity
  - Used to verify the authenticity of a message
- Idea: compute a hash value on the message and send it along with the message
- Receiver can apply the same hash function on the message and see whether the result coincides with the received hash

- Very hard to forge a message that produces the same hash value
  - i.e. Message -> hash is easy
  - Hash -> Message is hard
  - Compare to other error detection methods (CRC, parity, etc)
MD 5 (cont’d)

• Basic property: digest operation very hard to invert
  – Send the digest via a different channel
    • used it in FTP mirrors, user download MD5 digest of file separately from the file, hope no one can forge the MD5 digest before you even download the intended file
  – In practice someone cannot alter the message without modifying the digest
Importance of Network Security

• Internet currently used for important services
  – Financial transactions, medical records

• Could be used in the future for critical services
  – 911, surgical operations, energy system control, transportation system control

• Networks more open than ever before
  – Global, ubiquitous Internet, wireless

• Malicious Users
  – Selfish users: want more network resources than you
  – Malicious users: would hurt you even if it doesn’t get them more network resources
Network Security Problems

• 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
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
• Attacker gains control of a host
  – Reads data
  – Erases data
  – Compromises another host
  – Launches denial-of-service attack on another host
Definitions

• Worm
  – Replicates itself
  – Usually relies on stack overflow attack

• Virus
  – Program that attaches itself to another (usually trusted) program

• Trojan horse
  – Program that gives a hacker a back door
  – Usually relies on user exploitation
Host Compromise: Stack Based Buffer Overflow

• Typical code has many bugs because those bugs are not triggered by common input
• Network code is vulnerable because it accepts input from the network
• Network code that runs with high privileges (i.e., as root) is especially dangerous
  – E.g., web server
Example

What is wrong here?

```c
#define MAXNAMELEN 64
int offset = OFFSET_USERNAME;
char username[MAXNAMELEN];
int name_len;

name_len = ntohl(*(int *)packet);
memcpy(&username, packet[offset], name_len);
```

![Packet diagram](https://via.placeholder.com/150)
# Example

```c
void foo(packet) {
    #define MAXNAMELEN 64
    int offset = OFFSET_USERNAME;
    char username[MAXNAMELEN];
    int name_len;

    name_len = ntohl(*(int*)packet);
    memcpy(&username, packet[offset], name_len);
    ...
}
```
void foo(packet) {
    #define MAXNAMELEN 64
    int offset = OFFSET_USERNAME;
    char username[MAXNAMELEN];
    int name_len;

    name_len = ntohl(*(int *) packet);
    memcpy(&username,
           packet[offset],name_len);
    ...
}

Effect of Stack Based Buffer Overflow

• Write into part of the stack or heap
  – Write arbitrary code to part of memory
  – Cause program execution to jump to arbitrary code

• Worm
  – Probes host for vulnerable software
  – Sends bogus input
  – Attacker can do anything that the privileges of the buggy program allows
    • Launches copy of itself on compromised host
  – Spread at exponential rate
  – 10M hosts in < 5 minutes
Worm Spreading

\[ f = \frac{e^{K(t-T)} - 1}{1 + e^{K(t-T)}} \]

- \( f \) – fraction of hosts infected
- \( K \) – rate at which one host can compromise others
- \( T \) – start time of the attack
Worm Spreading

\[ f = \frac{e^{K(t-T)} - 1}{1 + e^{K(t-T)}} \]

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

\[ f \]

\[ 1 \]

\[ T \]

\[ t \]
Worm Spreading

\[ f = \frac{e^{K(t-T)} - 1}{1 + e^{K(t-T)}} \]

- \( f \) – fraction of hosts infected
- \( K \) – rate at which one host can compromise others
- \( T \) – start time of the attack
Worm Examples

- Morris worm (1988)
- Code Red (2001)
- MS Slammer (January 2003)
- MS Blaster (August 2003)
MS SQL Slammer (January 2003)

• Uses UDP port 1434 to exploit a buffer overflow in MS SQL server

• 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
MS SQL Slammer (January 2003)

Packet Loss %

(From http://www.f-secure.com/v-descs/mssqlm.shtml)
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)
  - Sendmail (most popular UNIX mail delivery software)
  - IIS (Windows web server)
  - SNMP (Simple Network Management Protocol, used to manage routers and other network devices)
Potential Solutions

• Don’t write buggy software
  – It’s not like people try to write buggy software
• Type-safe Languages
  – Unrestricted memory access of C/C++ contributes to problem
  – Use Java, Perl, or Python instead
• OS architecture
  – Compartmentalize programs better, so one compromise doesn’t compromise the entire system
  – E.g., DNS server doesn’t need total system access
• Firewalls
Firewall

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

• Easier to deploy firewall than secure all internal hosts
• Doesn’t prevent user exploitation
• 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., have all services use port 80

• Can’t prevent problem from spreading from within
Address Blacklisting and Content Filtering Solutions against Code Red Worm

• Result: content filtering is more effective.

Number of susceptible host decreases

20 min (a) Address Blacklisting

Worms unchecked

2 hr (b) Content Filtering
Host Compromise: User Exploitation

• Some security architectures rely on the user to decide if a potentially dangerous action should be taken, e.g.,
  – Run code downloaded from the Internet
    • “Do you accept content from Microsoft?”
  – Run code attached to email
    • “subject: You’ve got to see this!”
  – Allow a macro in a data file to be run
    • “Here is the latest version of the document.”
User Exploitation

• Users are not good at making this decision
  – Which of the following is the real name Microsoft uses when you download code from them?
    • Microsoft
    • Microsoft, Inc.
    • Microsoft Corporation

• Typical email attack
  – Attacker sends email to some initial victims
  – Reading the email / running its attachment / viewing its attachment opens the hole
  – Worm/trojan/virus mails itself to everyone in address book
Solutions

• OS architecture
• Don’t ask the users questions which they don’t know how to answer anyway
• Separate code and data
  – Viewing data should not launch attack
• Be very careful about installing new software
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 that 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
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
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.
**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.

![Diagram showing SYN attack process]

- Client (initiator)
- Server

SYN, SeqNum = x
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.

Diagram:
- Client (initiator)
- Server
- SYN, SeqNum = x
- SYN and ACK, SeqNum = y and Ack = x + 1
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.
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)
Solution: SYN Cookies

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

- Very small but aggressive mammal that ferociously attacks and kills much larger animals with a venomous bite

http://thezenafile.files.wordpress.com/2009/04/shrew1.jpg
TCP Congestion Control

- Slow-start phase
- Double the sending rate each round-trip time
- Reach high throughput quickly
TCP Congestion Control

- Additive Increase
- Multiplicative Decrease
- Fairness among flows

Sending Rate vs. Time

packet loss
TCP Congestion Control

- Exponential backoff
- System stability
- Vulnerability to high-rate attacks

Diagram showing sending rate over time with packet loss events.
TCP: a Dual Time-Scale Perspective

- Two time-scales **fundamentally** required
  - RTT time-scales (~10-100 ms)
    - AIMD control
  - RTO time-scales ($RTO = SRTT + 4 \times RTTVAR$)
    - Avoid congestion collapse

- Lower-bounding the RTO parameter:
  - [AllPax99]: minRTO = 1 sec
    - to avoid spurious retransmissions
  - [RFC2988] recommends minRTO = 1 sec

Discrepancy between RTO and RTT time-scales is a **key source of vulnerability** to low rate attacks
The Shrew Attack

TCP Sending Rate vs Time

DoS Rate vs Time
The Shrew Attack

- A short burst (~RTT) sufficient to create outage
- Outage – event of correlated packet losses that forces TCP to enter RTO mechanism

TCP Sending Rate

short burst (~RTT)

dos Rate

random initial phase

Time

Victim

Attacker

Alan Mislove  amislove at ccs.neu.edu
Northeastern University
The Shrew Attack

- The outage synchronizes all TCP flows
  - All flows react simultaneously and identically
  - backoff for minRTO
The Shrew Attack

• Once the TCP flows try to recover – hit them again

• Exploit protocol determinism
The Shrew Attack

• And keep repeating…

• RTT-time-scale outages interspaced on minRTO periods can deny service to TCP traffic
Shrew Principles

• A single RTT-length outage forces all TCP flows to simultaneously enter timeout
  – All flows respond identically and backoff for the minRTO period

• Shrews exploit protocol determinism, and repeat the outage after each minRTO period

• Periodic outages synchronize TCP flows and deny their service

• Outages occur relatively slowly (RTO-scale) and can be induced with low average rate
Shrews are Hard to Detect

- $l/T << 1$
- Low-rate flow is hard to detect
  - Most counter-DOS mechanisms tuned for high-rate attacks
  - Detecting Shrews may have unacceptably many false alarms (due to legitimate bursty flows)
The Shrew in Action

- How much is TCP throughput degraded?
- **DoS** stream:
  - $R=C=1.5\text{Mb/s}$;
  - $I=70\text{ms (\~TCP RTT)}$
The Shrew in Action

- How much is TCP throughput degraded?
- **DoS** stream:
  - $R=C=1.5\text{Mb/s}$;
  - $I=70\text{ms}$ (~TCP RTT)

![Graph showing TCP and DoS throughput over time.](image)
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
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
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 $V$
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
Dealing with Attacks

• Distinguish attack from flash crowd
• Prevent damage
  – Distinguish attack traffic from legitimate traffic
  – Rate limit attack traffic
• Stop attack
  – Identify attacking machines
  – Shutdown attacking machines
  – Usually done manually, requires cooperation of ISPs, other users
• Identify attacker
  – Very difficult, except
  – Usually brags/gloats about attack on IRC
  – Also done manually, requires cooperation of ISPs, other users
Incomplete Solutions

- Fair queueing, rate limiting (e.g., token bucket)
- Prevent a user from sending at 10Mb/s and hurting a user sending at 1Mb/s
- Does not prevent 10 users from sending at 1Mb/s and hurting a user sending a 1Mb/s
Identify and Stop Attacking Machines

• Defeat spoofed source addresses
• Does not stop or slow attack
• Ingress 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
Summary

• Network security is possibly the Internet’s biggest problem
  – Preventing Internet from expanding into critical applications
• Host Compromise
  – Poorly written software
  – Solutions: better OS security architecture, type-safe languages, firewalls
• Denial-of-Service
  – No easy solution: DoS can happen at many levels