Caveat

- These slides have more information than you need for 859 course
- I left it in so that people who want more information can find it
- For quick reading
  - Read up to slide 4 – intrusion phases
  - Read first slides on each of the intrusion phases
  - Spend more time on buffer overflow description
  - Read all slides on worms

Disclaimer

- Some techniques and tools mentioned in this class could be:
  - Illegal to use
  - Dangerous for others – they can crash machines and clog the network
  - Dangerous for you – downloading the attack code you provide attacker with info about your machine
  - Don’t use any such tools in real networks – especially not on EECIS network
  - You can only use them in a controlled environment

Intrusions

- Why do people break into computers?
- What type of people usually breaks into computers?
- I thought that this was a security course. Why are we learning about attacks?

Intrusion Scenario

1. Reconnaissance
2. Scanning
3. Gaining access at OS, application or network level
4. Maintaining access
5. Covering tracks

Phase 1: Reconnaissance

- Get a lot of information about intended target:
  - Learn how its network is organized
  - Learn any specifics about OS and applications running

Low Tech Reconnaissance

- Social engineering
  - Instruct the employees not to divulge sensitive information on the phone
- Physical break-in
  - Insist on using badges for access, everyone must have a badge, lock sensitive equipment
  - How about wireless access?
- Dumpster diving
  - Shred important documents
Web Reconnaissance

- Search organization’s web site
- Make sure not to post anything sensitive
- Search information on Usenet postings
- Instruct your employees what info should not be posted
- Find out what is posted about you
- Use Google to find all documents mentioning this company to find out partner companies
- Find out what is posted about you

Whois databases

- When an organization acquires domain name it provides information to a registrar
- Looking at public registrar files one can find out:
  - Registered domain names
  - Domain name servers
  - Contact person names, phone numbers, E-mail addresses

Domain Name System

- What does DNS do?
- How does DNS work?
- Types of information an attacker can gather:
  - Range of addresses used
  - Address of a mail server
  - Address of a web server
  - OS information
  - Comments

Interrogating DNS – Zone Transfer

```
$ nslookup
Default server: evil.attacker.com
Address: 10.11.12.13 server 1.2.3.4
Default server: dns.victimsite.com
Address: 1.2.3.4
set type=any
ls -d victimsite.com
system1 1DINA 1.2.2.1
1DININFO "Solaris 2.6 Mailserver"
1DNMX 10 mail1
web 1DINA 1.2.11.27
1DININFO "NT4www"
```

Protecting DNS

- Provide only necessary information – no OS info and no comments
- Restrict zone transfers – allow only a few necessary hosts
- Use split-DNS

Split-DNS

Show different view to external and internal users
Reconnaissance Tools

- Tools that integrate ping, whois, ARIN, DNS interrogation and many more services:
  - Applications
    - http://www.samspade.org/saw
  - Web based portals
    - http://nettool.false.net
    - http://www.samspade.org
    - http://members.tripod.com/mixersecurity/evil.html
    - http://www.network-tools.com

Phase 2: Scanning

- Detecting information useful for break-in
- Live machines
- Network topology
- Firewall configuration
- Applications and OS types
- Vulnerabilities

Network Mapping

- Finding live hosts
- Ping sweep
- TCP SYN
- Map network topology
- Traceroute
  - Sends out ICMP or UDP packets with increasing TTL
  - Gets back ICMP_Time_Extended message from intermediate routers

Traceroute Example

```
traceroute to copland.udel.edu (128.175.13.92) 30 hops max, 30 byte packets
1. dwarn (131.179.192.11) 0.276 ms 0.288 ms 0.288 ms
2. 131.179.197.3 (131.179.197.3) 0.413 ms 0.413 ms 0.413 ms
3. Border CI UCLA EDU (131.179.32.1) 0.795 ms 0.808 ms 0.808 ms
4. compsci--csb--b--ncu.edu (158.252.8.65) 0.842 ms 0.858 ms 0.858 ms
5. mathsci--csb--b--ncu.edu (169.252.8.69) 0.815 ms 0.815 ms 0.815 ms
6. core-border backbone udel.edu (192.30.3.19) 0.739 ms 0.743 ms 0.743 ms
7. tun--c--sds--s--e--cmu.edu (137.164.24.135) 1.609 ms 2.036 ms 2.571 ms
8. 4-c-fox--d--2--ns-0-1--cmu.edu (137.164.22.42) 1.919 ms 1.903 ms 2.338 ms
9. bps-lab-ib--ds-2--g--2--cmu.edu (137.164.22.21) 1.974 ms 2.577 ms 2.803 ms
10. abilene-la--bps-lab-ger1-abilene.net (137.164.25.3) 16.122 ms 1.835 ms 10.820 ms
11. frs-ing-ib--abilene udel.edu (198.32.3.22) 33.863 ms 33.734 ms 34.023 ms
12. ating-ing-ah--abilene udel.edu (198.32.8.34) 46.941 ms 46.941 ms 46.941 ms
13. washing-at--abilene udel.edu (198.32.6.66) 73.733 ms 73.886 ms 73.937 ms
14. csp-ps--d--ncu udel.edu (128.175.13.92) 77.276 ms 77.861 ms 77.723 ms
15. csp-ps--d--ncu udel.edu (128.175.13.92) 77.276 ms 77.861 ms 77.723 ms
16. copland.udel.edu (128.175.13.92) 77.276 ms 77.861 ms 77.723 ms
```

Network Mapping Tools

- Cheops
  - Linux application
  - http://www.marko.net/cheops
  - Automatically performs ping sweep and network mapping and displays results in GUI

Defenses Against Network Mapping

- Filter out outgoing ICMP traffic
- Maybe allow for your ISP only
- Use NAT

```
Request 1.2.3.4
Reply 1.2.3.4
NAT
Request 192.168.13.75
Reply 192.168.13.75
Address 8.9.10.11
Address 1.2.3.4
```

Internal hosts with 192.168.0.0/16 addresses
Port Scanning

- Finding applications that listen on ports
- Send various packets:
  - Establish and tear down TCP connection
  - Half-open and tear down TCP connection
  - Send invalid TCP packets: FIN, Null, Xmas scan
  - Send TCP ACK packets – find firewall holes
  - Obscure the source – FTP bounce scans
  - UDP scans
  - Find RPC applications

Port Scanning Tools

- Nmap
  - Unix and Windows NT application and GUI
  - Various scan types
  - Adjustable timing

Defenses Against Port Scanning

- Close all unused ports
- Remove all unnecessary services
- Filter out all unnecessary traffic
- Find openings before the attackers do
- Use smart filtering

Firewalk: Determining Firewall Rules

- Find out firewall rules for new connections
- We don’t care about target machine, just about packet types that can get through the firewall
- Find out distance to firewall using traceroute
- Ping arbitrary destination setting TTL=distance+1
- If you receive ICMP_Time_Exceeded packet went through

Defenses Against Firewalking

- Filter out outgoing ICMP traffic
- Use firewall proxies
Vulnerability Scanning

- The attacker knows OS and applications installed on live hosts
- He can now find for each combination
  - Vulnerability exploits
  - Common configuration errors
  - Default configuration
- Vulnerability scanning tool uses a database of known vulnerabilities to formulate packets and send them to hosts
- Vulnerability scanning is also used for sysadmin

Vulnerability Scanning Tools

- SARA
  - http://www.arc.com/sara
- SAINT
  - http://www.wwdsi.com/saint
- VLAD
  - http://razor.bindview.com/tools
- Nessus
  - http://www.nessus.org

Defenses Against Vulnerability Scanning

- Close your ports and keep systems patched
- Find your vulnerabilities before the attackers do

Phase 3: Gaining Access

- Exploit vulnerabilities
  - Exploits for a specific vulnerability can be downloaded from hacker sites
  - Skilled hackers write new exploits

Stack-based Overflow Attacks

- Stack stores important data on procedure call

<table>
<thead>
<tr>
<th>TOS</th>
<th>Saved frame ptr</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Return address</td>
</tr>
<tr>
<td></td>
<td>Local variables for called procedure</td>
</tr>
<tr>
<td></td>
<td>Function call arguments</td>
</tr>
<tr>
<td></td>
<td>Memory address increases</td>
</tr>
</tbody>
</table>

Consider a function

```c
void sample_function(char* s) {
    strcpy(buffer, string);
    return;
}
```

And a main program

```c
void main() {
    int i;
    char buffer[200];
    for(i=0; i<200; i++) buffer[i] = A;
    sample_function(buffer);
    return;
}
```
Stack-based Overflow Attacks

- Large input will be stored on the stack, overwriting information
- TOS
  - Saved frame ptr
  - Return address
  - Function call arguments
  - Memory address increases
- Overwritten by A’s

Stack-based Overflow Attacks

- IDS could look for sequence of NOPs to spot buffer overflows
- Attacker uses polymorphism: he transforms the code so that NOP is changed into some other command that does the same thing, e.g. MV R1, R1
- Attacker XORs important commands with a key
- Attacker places XOR command and the key just before the encrypted attack code, for decryption
- XOR command is also obscured

Stack-based Overflow Attacks

- What type of commands does the attacker execute?
  - Commands that help him gain access to the machine
  - Writes a string into inetd.conf file to start shell application listening on a port, then uses Netcat to make raw interactive connection to the port
  - Starts TFTP to transfer Netcat onto the victim, then accesses it
  - Starts XTerm

Defenses Against Stack-based Overflow

- How does an attacker discover stack-based overflow?
  - Looks at the source code
  - Runs application on his machine, tries to supply long inputs and looks at system registers
  - Read more at
    - http://packetstormsecurity.nl/docs/hack/smashstack.txt

- For system administrators:
  - Apply patches, keep systems up-to-date
  - Disable execution from the stack
  - Monitor writes on the stack
  - Store return address somewhere else
  - Monitor outgoing traffic
- For software designers
  - Apply checks for buffer overflows
  - Use safe functions
Password Attacks

- Attacker attempts to log in with some known username, and to guess a password
- Trying dictionary words
- Trying combinations of dictionary words
- Performing brute-force search
- Attacker steals encrypted or hashed password file and tries to decrypt it

Defenses Against Password Attacks

- Make strong passwords
- Think of a phrase, take first letters, mix big caps and special characters
- Use password filtering software
- Use strong encryption/hash techniques

Web Application Attacks

- Account harvesting
  - Gather usernames by observing error messages, then try to guess passwords
  - Defense: use same error messages for everything
- Hijack a session ID
  - Observe session ID and how it changes between sessions
  - Change your session ID to another one
  - Defense: digitally sign or hash session ID, make them long enough and apply timestamps

Web Application Attacks

- SQL Piggybacking
  - Malformed input into Web form may trigger informative message from an SQL server
    - Error in SQL syntax near 111111111 at line 1
    - SELECT * FROM account WHERE (userid='10001'
    - and number='111111111' 
    - and number='111111111' or userid='10002'
    - Attack then adds SQL commands into input
    -Defense: filter user input

Sniffing

- Looking at raw packet information on the wire
- Some media is more prone to sniffing – Ethernet
- Some network topologies are more prone to sniffing – hub vs. switch

Sniffing on a Hub

- Ethernet is a broadcast media – every machine connected to it can hear all the information
  - Passive sniffing
    - For X
    - A
    - R
    - Y
    - X

CIS 859 – Advanced Topics in Network Security – Spring 2004 – Class 3 – 2/17/04
Sniffing on a Hub
- Attacker can get anything that is not encrypted and is sent to LAN
- Defense: encrypt all sensitive traffic
- Tcpendump
  - http://www.tcpdump.org
- Windump
  - http://netgroup-serv-polito.it/windump
- Snort
  - http://www.snort.org
- Ethereal
  - http://www.ethereal.com

Sniffing on a Switch
- Switch is connected by a separate physical line to every machine and it chooses only one line to send the message

Sniffing on a Switch – Take 1
- Attacker sends a lot of ARP messages for fake addresses to R
- Some switches send on all interfaces when their table overloads

Sniffing on a Switch – Take 2
- Address Resolution Protocol (ARP) maps IP addresses with MAC addresses

Active Sniffing Tools
- Dsniff
  - http://www.monkey.org/~dugsong/dsniff
- Also parses application packets for a lot of applications
- Sniffs and spoofs DNS
Spoofing DNS
- Attacker sniffs DNS requests, replies with his own address faster than real server
- When real reply arrives client ignores
- This can be coupled with man-in-the-middle attack on HTTPS and SSH

Sniffing Defenses
- Use end-to-end encryption
- Use switches
  - Statically configure MAC and IP bindings with ports
  - Don’t accept suspicious certificates

IP Address Spoofing
- Attacker cannot see reply packets

Guessing a Sequence Number
- It used to be ISN=f(Time), still is in Windows

Guessing a Sequence Number
- On Linux ISN=f(time)+rand
Guessing a Sequence Number
- On BSD ISN=rand

Spoofing with Source Routing
- Attacker uses *loose source routing* option to specify himself as a hop
- Spoof's Alice’s address, sends packets to Bob
- Bob sends replies back on the same route

Spoofing Defenses
- Ingress and egress filtering
- Prohibit source routing option
- Don’t use trust models with IP addresses
- Randomize sequence numbers

Netcat Tool
- Similar to Linux *cat* command
- http://netcat.sourceforge.net/
- Server: Initiates connection to any port on remote machine
- Client: Listens on any port
- To transfer file
  - On source machine: `nc -l -p 1234 < file.txt`
  - On remote machine: `nc 123.32.34.54 1234 > file.txt`
  or
  - On source machine: `nc 44.22.123.212 1234 < file.txt`
  - On remote machine: `nc -l -p 1234 > file.txt`

Phase 4: Maintaining Access
- Attacker establishes a listening application on a port (backdoor) so he can log on any time with or without a password
- Attackers frequently close security holes they find
- Netcat as a backdoor
  - `nc -l -p 12345 -e /bin/sh`
Trojans
➢ Application that claims to do one thing (and looks like it) but it also does something malicious
➢ Users download Trojans from Internet (thinking they are downloading a free game) or get them as greeting cards in E-mail, or as ActiveX controls when they visit a Web site
➢ Trojans can scramble your machine
➢ They can also open a back door on your system
➢ They will also report successful infection to the attacker

Back Orifice
➢ Trojan application that can
➢ Log keystrokes
➢ Steal passwords
➢ Create dialog boxes
➢ Mess with files, processes or system (registry)
➢ Redirect packets
➢ Set up backdoors
➢ Take over screen and keyboard
➢ http://www.bo2k.com/

Trojan Defenses
➢ Antivirus software
➢ Don’t download suspicious software
➢ Check MD5 sum on trusted software you download
➢ Disable automatic execution of attachments

Rootkits
➢ Attackers will modify all key system applications that could reveal his presence
➢ List processes
➢ List files
➢ Show open ports
➢ Show system utilization
➢ He will also substitute modification date with the one in the past

Defenses Against Rootkits
➢ Don’t let attackers gain root access
➢ Use integrity checking of files:
  ➢ Carry a floppy with md5sum, check hashes of system files against hashes advertised on vendor site or hashes you stored before
➢ Use Tripwire
  ➢ Free integrity checker that saves md5 sums of all important files in a secure database (read only CD), then verifies them periodically
➢ http://www.tripwire.org/
Kernel Rootkits
- Replace system calls
- Intercept calls to open one application with calls to open another, of attacker’s choosing
- Now even checksums don’t help as attacker did not modify any system applications
- You won’t even see attacker’s files in file listing
- You won’t see some processes or open ports
- Usually installed as kernel modules
- Defenses: detect some fingerprints, disable kernel modules and pray

Phase 5: Covering Tracks
- Rootkits
- Alter logs
- Create hard-to-spot files
- Use covert channels

Altering Logs
- For binary logs:
  - Stop logging services
  - Load files into memory, change them
  - Restart logging service
  - Or use special tool
- For text logs simply change file through scripts
- Change login and event logs, command history file, last login data

Defenses Against Altering Logs
- Use separate log servers
- Machines will send their log messages to these servers
- Encrypt log files
- Make log files append only
- Save logs on write-once media

Creating Hard-to-Spot Files
- Names could look like system file names, but slightly changed
- Start with .
- Start with . and add spaces
- Make files hidden
- Defenses: intrusion detection systems and caution

Covert Channels
- Transfer data across the network in unsuspicious way
- Wrapping it up in ICMP packets
- Or in HTTP
  - Server on infected machine goes to master “Web server” periodically
  - If master has typed some commands, server executes them and pushes the result
  - It appears as if machine is engaged in Web surfing
- Or in SMTP
- Or in TCP (SYN and ACK fields) and IP headers (ID field)
Defenses Against Covert Channels

- Detect malformed packets for certain protocols
- Use port scan, detect unusual services

Firewalls

- Control what comes in and out of a network
  - How do they know what should go in and out?
  - How are rules specified?
  - What are big challenges?
- Flavors of firewalls
  - Traditional packet filters
  - Stateful packet filters
  - Proxy-based firewalls

Traditional Packet Filters

- Examine each packet based on simple rules and forward or drop it
  - Source IP, destination IP
  - Source/destination ports
  - Protocol
  - TCP flags
  - Direction
  - Interface
- Rules are implemented in top-down manner, first rule that matches is applied

Stateful Packet Filters

- Remember the session context
  - Packets are let through or denied based on the session state
  - Each connection remembered in a state table for a given time
  - Afterwards, inactive records are deleted

Proxy-Based Firewalls

- Analyze the application information, determine which packets to forward and which to drop
  - It could authenticate users before passing packets to internal network
  - It could cache frequently accessed information

What is a Worm?

- A program that:
  - Scans network for vulnerable machines
  - Breaks into machines by exploiting the found vulnerability
  - Installs some piece of malicious code
  - Moves on
  - Unlike viruses, worms don’t need any user action to spread – they spread silently and on their own
  - Unlike viruses, worms don’t attach themselves onto other programs – they exist as a separate code in memory
  - Sometimes you may not even know your machine has been infected by a worm
**What is a Worm?**

- A program that:
  - Scans network for vulnerable machines
  - Breaks into machines by exploiting the found vulnerability
  - Installs some piece of malicious code – backdoor, DDoS tool
  - Moves on
- Unlike viruses, worms don’t need any user action to spread – they spread silently and on their own
- Unlike viruses, worms don’t attach themselves onto other programs – they exist as a separate code in memory
- Sometimes you may not even know your machine has been infected by a worm

---

**Why Are Worms Dangerous?**

- They spread extremely fast
- They are silent
- Once they are out, they cannot be recalled
- They usually install malicious code
- They clog the network

---

**First Worm Ever – Morris Worm**

- Robert Morris, a PhD student at Cornell was interested in network security
- He created the first worm with a goal to have a program *live* on the Internet in November 1988
- Worm was supposed only to spread, fairly slowly
- It was supposed to take just a little bit of resources so not to draw attention to itself
- But things went wrong …
- Worm was supposed to avoid duplicate copies by asking a computer whether it is infected
  - To avoid false “yes” answers, it was programmed to duplicate itself every 7th time it received “yes” answer
  - This turned out to be too much

---

**First Worm Ever – Morris Worm**

- It exploited four vulnerabilities to break in
  - A bug in sendmail
  - A bug in finger daemon
  - A trusted hosts feature (/etc/hosts)
  - Password guessing
- Worm was replicating at a much faster rate than anticipated
- At that time Internet was small and homogeneous (SUN and VAX workstations running BSD UNIX)
- It infected around 6,000 computers, one tenth of then-Internet, in a day

---

**First Worm Ever – Morris Worm**

- People quickly devised patches and distributed them (Internet was small then)
- A week later all systems were patched and worm code was removed from most of them
- No lasting damage was caused
- Robert Morris paid 10,000$ fine, was placed on probation and did some community work
- Worm exposed not only vulnerabilities in UNIX but moreover in Internet organization
- Users didn’t know who to contact and report infection or where to look for patches

---

**First Worm Ever – Morris Worm**

- In response to Morris Worm DARPA formed CERT (Computer Emergency Response Team) in November 1988
  - Users report incidents and get help in handling them from CERT
  - CERT publishes security advisory notes informing users of new vulnerabilities that need to be patched and how to patch them
  - CERT facilitates security discussions and advocates better system management practices
**Code Red v2**

- Spread on July 19, 2001
- Exploited the same vulnerability in Microsoft Internet Information Server that allows attacker to get full access to the machine (turned on by default)
- Two variants – both probed random machines, one with static seed for RNG, another with random seed for RNG
- Infected more than 359,000 computers in less than 14 hours
- It doubled in size every 37 minutes
- At the peak of infection more than 2000 hosts were infected each minute

- Infected more than 359,000 computers in less than 14 hours
- It doubled in size every 37 minutes
- At the peak of infection more than 2000 hosts were infected each minute

- It had hardcoded IP address so White House was able to thwart the attack by simply changing the IP address-to-name mapping
- Estimated damage ~2.6 billion

**Sapphire/Slammer Worm**

- Spread on January 25, 2003
- The fastest computer worm in history
  - It doubled in size every 8.5 seconds.
  - It infected more than 90% of vulnerable hosts within 10 minutes
  - It infected 75,000 hosts overall
- Exploited buffer overflow vulnerability in Microsoft SQL server, discovered 6 months earlier

- No malicious payload
- The aggressive spread had severe consequences
  - It created DoS effect
  - It disrupted backbone operation
  - Airline flights were canceled
  - Some ATM machines failed

**Estimated damage ~2.6 billion**

**Sapphire/Slammer Worm**

- Spread on January 25, 2003
- The fastest computer worm in history
  - It doubled in size every 8.5 seconds.
  - It infected more than 90% of vulnerable hosts within 10 minutes
  - It infected 75,000 hosts overall
- Exploited buffer overflow vulnerability in Microsoft SQL server, discovered 6 months earlier

- No malicious payload
- The aggressive spread had severe consequences
  - It created DoS effect
  - It disrupted backbone operation
  - Airline flights were canceled
  - Some ATM machines failed

**Estimated damage ~2.6 billion**
Why Was Slammer So Fast?

- Both Slammer and Code Red 2 use random scanning
- Code Red uses multiple threads that invoke TCP connection establishment through 3-way handshake – must wait for the other party to reply or for TCP timeout to expire
- Slammer packs its code in single UDP packet – speed is limited by how many UDP packets can a machine send
- Could we do the same trick with Code Red?
- Slammer authors tried to use linear congruent generators to generate random addresses for scanning, but programmed it wrong

Sapphire/Slammer Worm

- 43% of infected machines were in US
- 59% of infected machines were home computers
- Response was fast – after an hour sites started filtering packets for SQL server port

BGP Impact of Slammer Worm

Scanning Strategies

- All so far discovered worms use random scanning
- This works well only if machines have very good RNGs with different seeds
- Getting large initial population represents a problem
  - Then the infection rate skyrockets
  - The infection eventually reaches saturation since all machines are probing same addresses

Random Scanning

Scanning Strategies

- Worm can get large initial population with hitlist scanning
- Assemble a list of potentially vulnerable machines prior to releasing the worm – a hitlist
  - E.g., through a slow scan
- When the scan finds a vulnerable machine, hitlist is divided in half and one half is communicated to this machine upon infection
  - This guarantees very fast spread – under one minute!
Hitlist Scanning

Effect of hitlist size

Scanning Strategies
- Worm can get prevent die out in the end with permutation scanning
- All machines share a common pseudorandom permutation of IP address space
- Machines that are infected continue scanning just after their point in the permutation
  - If they encounter already infected machine they will continue from a random point
- **Partitioned permutation** is the combination of permutation and hitlist scanning
  - In the beginning permutation space is halved, later scanning is simple permutation scan

Permutation Scanning

3 infection modes

Scanning Strategies
- Worm can get behind the firewall, or notice the die-out and then switch to subnet scanning
- Goes sequentially through subnet address space, trying every address

Worst Case Warhol Worm

- Hypothetical worm
- Uses vulnerabilities in Microsoft IIS (to spread to many places) and Microsoft Exchange (to spread beyond firewalls)
- Use hitlist scanning, subnet and permutation scanning, could spread in 15 minutes
- Malicious payload is activated upon installation but guaranteed not to slow down worm spread
  - E.g., overwrite random pieces of non-system files
  - Then DDoS-es some targets

Infection Strategies

- Several ways to download malicious code
  - From a central server
  - From the machine that performed infection
  - Send it along with the exploit in a single packet
Worm Defense

- Three factors define worm spread:
  - Size of vulnerable population
    - Prevention – patch vulnerabilities, increase heterogeneity
  - Rate of infection (scanning and propagation strategy)
    - Deploy firewalls
    - Distribute worm signatures
  - Length of infectious period
    - Patch vulnerabilities after the outbreak