EEC 693/793

Special Topics in Electrical Engineering

Secure and Dependable Computing

Lecture 9

Wenbing Zhao

Department of Electrical and Computer Engineering

Cleveland State University

wenbing@ieee.org

Outline

- Network intrusion
  - Reconnaissance: collection host and network information => find vulnerability to exploit
  - Act of intrusion: denial of service, TCP session hijacking
- Intrusion detection systems
  - Overview
  - Case study: snort
- Reference: Network Intrusion Detection, 3r Ed., By Stephen Northcutt and Judy Novak, New Riders Publishing, 2002
  - http://proquest.safaribooksonline.com/0735712654

Purpose of Network Attacks

- Reconnaissance
- Compromising systems for notoriety for "10 minutes of fame"
- Gathering corporate or sensitive company information for financial compensation
- Destructive or malicious behavior

Counter Measures

- Firewalls
- Access control lists (ACLs)
- Physical security
- Limiting network access points
- Monitoring and auditing systems
- Intrusion detection systems

Background - ICMP

- ICMP: It provides a simple means of communicating between hosts or a router and a host to alert them to some kind of problem situation
- ICMP doesn't use ports to communicate like the transport protocols do
- ICMP messages can get lost and not be delivered
- ICMP can be broadcast to many hosts
- Hosts and routers are the senders of ICMP messages.
- Hosts listen for ICMP, and most will respond unless they deliberately have been altered for silence

Background - TCPdump

- TCPdump is a UNIX tool used to gather data from the network, decipher the bits, and display the output in a semi coherent fashion
  - See http://www.tcpdump.org for more information
- TCPdump output format
  - **09:32:43:910000 nmap.edu.1173 > dns.net.21: S 62697789:62697789(0) win 512**
  - 09:32:43:9147882 - time stamp in the format of two digits for hours, two digits for minutes, two digits for seconds, and six digits for fractional parts of a second
  - nmap.edu - source host name. If there is no resolution for the IP number or the default behavior of host name resolution is not requested, the IP number appears and not the host name
  - 1173 - source port number, or port service
  - > - marker to indicate a directional flow going from source to destination
  - dns.net - destination host name
  - 21 - The destination port number (for example, 21 might be translated as FTP)
  - S - TCP flag. The *S* represents the SYN flag, which indicates a request to start a TCP connection
  - 62697789:62697789(0) - beginning TCP sequence number:ending TCP sequence number (data bytes)
  - win 512 - receiving buffer size (in bytes) of nmap.edu for this connection

Reconnaissance

- Host and network mapping:
  - To determine what hosts or services are available in a facility
- To map a class B network
  - Up to 65,536 hosts
  - About 50 TCP and UDP ports account for the probable services
  - So the target space is something in the range of 163 million => which could be scanned in less than four months at 18 packets per second

Host Scan Using UDP Echo Requests

- In the following trace, the attacker is targeting multiple network addresses

```
02:08:48.088681 slowpoke.mappem.com.3066 > 192.168.134.117.echo: udp 6
02:15:04.539055 slowpoke.mappem.com.3066 > 172.31.73.1.echo: udp 6
02:15:13.155988 slowpoke.mappem.com.3066 > 172.31.16.152.echo: udp 6
02:22:38.573703 slowpoke.mappem.com.3066 > 192.168.91.18.echo: udp 6
02:27:07.867063 slowpoke.mappem.com.3066 > 172.31.2.176.echo: udp 6
02:30:38.220795 slowpoke.mappem.com.3066 > 192.168.5.103.echo: udp 6
02:49:31.024008 slowpoke.mappem.com.3066 > 172.31.152.254.echo: udp 6
02:49:55.547694 slowpoke.mappem.com.3066 > 192.168.219.32.echo: udp 6
```

- This scan is seeing whether any host will reply on the echo port. The echo port echoes back (imagine that) any characters sent to it
- Good system administrators should not have this port listening and good network administrators should not allow in traffic to this port

Host Scan Using ICMP Echo Requests

- The “ping” utility generate ICMP echo requests
- If an ICMP echo request is sent to a broadcast address, all the hosts in the subnet might reply

```
02:21:06.700002 pinger> 172.20.64.0: icmp: echo request
02:21:06.714882 pinger> 172.20.64.64: icmp: echo request
02:21:06.715229 pinger> 172.20.64.63: icmp: echo request
02:21:06.715561 pinger> 172.20.64.127: icmp: echo request
02:21:06.716021 pinger> 172.20.64.128: icmp: echo request
02:21:06.746119 pinger> 172.20.64.191: icmp: echo request
02:21:06.746487 pinger> 172.20.64.192: icmp: echo request
02:21:06.746845 pinger> 172.20.64.255: icmp: echo request
```


Port Scan

- After our attacker has found a host, he may want to scan it to see what services are active
- In the following trace, TCP SYN segment is used to probe each port

```
09:52:25.349706 bad.guy.org.1797 > target.mynetwork.com.12: S
09:52:25.375756 bad.guy.org.1798 > target.mynetwork.com.11: S
09:52:26.573678 bad.guy.org.1800 > target.mynetwork.com.10: S
09:52:26.603163 bad.guy.org.1802 > target.mynetwork.com.9: S
09:52:28.639922 bad.guy.org.1804 > target.mynetwork.com.8: S
09:52:28.668172 bad.guy.org.1806 > target.mynetwork.com.7: S
09:52:32.749958 bad.guy.org.1808 > target.mynetwork.com.6: S
09:52:32.772739 bad.guy.org.1809 > target.mynetwork.com.5: S
09:52:32.802331 bad.guy.org.1810 > target.mynetwork.com.4: S
09:52:32.824582 bad.guy.org.1812 > target.mynetwork.com.3: S
09:52:32.850126 bad.guy.org.1814 > target.mynetwork.com.2: S
09:52:32.871856 bad.guy.org.1816 > target.mynetwork.com.1: S
```


Stealth Scanning

- Intentionally violating the TCP three-way handshake to bypass firewalls and intrusion detectors
  - Send a TCP segment with FIN flag on to a host that never had such a connection
  - Send a TCP segment with both SYN and FIN flag on
  - In both cases, a RST segment is sent by if the host exists, an ICMP message will be sent back otherwise

Inverse Mapping

- Inverse mapping techniques
  - Compile a list of networks, or hosts, that are not reachable
  - Then use the converse of that map to determine where things probably are
- Counter measure
  - Do not allow “ICMP unreachables” out of your network

Use IP Fragmentation

- Only first fragment chunk comes with protocol information
- For later fragments, the firewalls would assume that this was just another segment of traffic that had already passed their access lists
- On receiving a fragment, if one of the target hosts does not exist, the router sends back an unreachable message.
- The attacker can then compile a list of all the hosts that do not exist and, by taking the inverse of that list, has a list of the hosts that do exist

Denial of Service

- A **denial-of-service attack** (**DoS attack**) is an attack on a computer system or network that causes a loss of service to users, typically the loss of network connectivity and services by consuming the bandwidth of the victim network or overloading the computational resources of the victim system
- Techniques of DoS
  - Brute force: UDP floods, SYN floods, Smurf, Echo-Chargen
  - One-packet kills: Teardrop, Land, Ping of death

UDP Flooding

- UDP is a connectionless protocol and it does not require any connection setup procedure to transfer data
- A UDP Flooding Attack is possible when an attacker sends a UDP packet to a random port on the victim system
  - When the victim system receives a UDP packet, it will determine what application is waiting on the destination port
  - When it realizes that there is no application that is waiting on the port, it will generate an ICMP packet of destination unreachable to the forged source address
  - If enough UDP packets are delivered to ports on victim, the system will go down

SYN Flooding

- The goal of SYN flooding is to throw hundreds or thousands of packets per second at a server to exhaust either system resources or even network resources when the rate is high enough
  - SYN flooding was used against Yahoo! and other high-profile Internet sites in February 2000
- When an attacker sets up a SYN flood, he has no intention to complete the three-way handshake and establish the connection. Rather, the goal is to exceed the limits set for the number of connections waiting to be established for a given service

Smurf Attack

- The Smurf attack relies on ICMP's capability to send traffic to broadcast address => Use intermediate networks as amplification points

Echo-Chargen Attach

- Echo uses UDP port 7; if it receives a packet it echoes back the payload. If you send echo an "a," it replies with an "a."
- Chargen (character generator) uses UDP port 19. If you send Chargen any characters, it replies with a pseudo random string of characters
- An attacker spoofs a number of connections to various hosts' Chargen ports. If both services are enabled, a game of **Echo <--> Chargen ping-pong** will begin burning bandwidth and CPU cycles

Teardrop Attack

- **Teardrop:** An attacker sends two fragments that cannot be reassembled properly by manipulating the offset value of packet and cause reboot or halt of victim system due to resource exhaustion

```
10:25:48.205383 wile-e-coyote.45959 > target.net.3964: udp 28 (frag 242:36@0+)
10:25:48.205383 wile-e-coyote > target.net: (frag 242:4@24)

18:49:54.519006 10.0.0.1.59108 > 10.0.0.2.161:  GetRequest(33)
.1.3.6.1.2.1.1.5.0[len3<asnlen4294967295] (DF)
4500 004c 0000 4000 4011 269f 0a00 0001
0a00 0002 e6e4 00a1 0038 0efc 302e 0201
0004 0670 7562 6c69 63a0 2102 0206 9202
0100 0201 0030 1530 1306 082b 0601 0201
0105 0044 84ff ffff ff02 0100
```


Land Attack

- **Land:** An attacker sends a forged packet with the same source and destination IP address. The victim system will be confused and crashed or rebooted

```
12/03/97 02:19:48     192.168.1.1  80    -> 192.168.1.1      80
12/03/97 02:21:53     192.168.1.1  31337 -> 192.168.1.1      31337
```

Ping of Death Attack

- **Ping of Death:** An attacker sends an ICMP echo request packet that is much larger than the maximum IP packet size to victim
  - Generally, sending a ping packet of a size such as 65,536 bytes is illegal according to networking protocol, but a packet of such a size can be sent if it is fragmented
  - When the target computer reassembles the packet, a buffer overflow can occur, which often causes a system crash

TCP Session Hijacking

- Conventional TCP exchanges do not require any authentication or confirmation that they are the actual hosts involved in a previously established connection
- After a session has been established between two hosts, those hosts use the following to reconfirm the corresponding host:
  - **IP number**
  - **Port numbers**
  - **Sequence numbers**
  - **Acknowledgement numbers**
- If a hostile user can observe data exchanges and successfully intercept an ongoing connection with all the authentication parameters properly set, he can hijack a session

Mitnick Attack

- The Mitnick attack is one of the most famous intrusion cases to ever occur
- The attack used two techniques:
  - SYN flooding – keep one system from being able to transmit
  - TCP hijacking – while the system was in a mute state, the attacker assumed its apparent identity and hijacked the TCP connection
- Mitnick detected a trust relationship between two computers and exploited that relationship

Mitnick Attack

- Step 1: recon probes

```
14:09:32 toad.com# finger -l @target
14:10:21 toad.com# finger -l @server
14:10:50 toad.com# finger -l root@server
14:11:07 toad.com# finger -l @x-terminal
14:11:38 toad.com# showmount -e x-terminal
14:11:49 toad.com# rpcinfo -p x-terminal
14:12:05 toad.com# finger -l root@x-terminal
```

Mitnick Attack

- Examining Network Traces – find how the host establishes ISN
  - 2021824000 – 2021952000 = 128,000

```
+++
14:18:25.906002 apollo.it.luc.edu.1000 > x-terminal.shell: S
1382726990:1382726990(0) win 4096
14:18:26.094731 x-terminal.shell > apollo.it.luc.edu.1000: S
2021824000:2021824000(0) ack 1382726991 win 4096
14:18:26.172394 apollo.it.luc.edu.1000 > x-terminal.shell: R
1382726991:1382726991(0) win 0
+++

+++
14:18:26.507560 apollo.it.luc.edu.999 > x-terminal.shell: S
1382726991:1382726991(0) win 4096
14:18:26.694691 x-terminal.shell > apollo.it.luc.edu.999: S
2021952000:2021952000(0) ack 1382726992 win 4096
14:18:26.775037 apollo.it.luc.edu.999 > x-terminal.shell: R
1382726992:1382726992(0) win 0
+++
```


Mitnick Attack

- Step 3: SYN flood the login server

```
14:18:22.516699 130.92.6.97.600 > server.login: S 1382726960:1382726960(0) win 4096
14:18:22.566069 130.92.6.97.601 > server.login: S 1382726961:1382726961(0) win 4096
14:18:22.744477 130.92.6.97.602 > server.login: S 1382726962:1382726962(0) win 4096
14:18:22.830111 130.92.6.97.603 > server.login: S 1382726963:1382726963(0) win 4096
14:18:22.886128 130.92.6.97.604 > server.login: S 1382726964:1382726964(0) win 4096
14:18:22.943514 130.92.6.97.605 > server.login: S 1382726965:1382726965(0) win 4096
```


Mitnick Attack

- Step 4: TCP session hijacking
  - Initiate a connection

```
14:18:36.245045 server.login > x-terminal.shell: S 1382727010:1382727010(0) win 4096
14:18:36.755522 server.login > x-terminal.shell: . ack 2024384001 win 4096
```

  - Compromise the host (x-terminal): the trusted connection is used to execute the following UNIX command with rshell: **rsh x-terminal "echo + + >>/.rhosts"**. The result of this causes x-terminal to trust, as root, all computers and all users on these computers

```
14:18:37.265404 server.login > x-terminal.shell: P 0:2(2) ack 1 win 4096
14:18:37.775872 server.login > x-terminal.shell: P 2:7(5) ack 1 win 4096
14:18:38.287404 server.login > x-terminal.shell: P 7:32(25) ack 1 win 4096
```

  - Terminate the connection

```
14:18:41.347003 server.login > x-terminal.shell: . ack 2 win 4096
14:18:42.255978 server.login > x-terminal.shell: . ack 3 win 4096
14:18:43.165874 server.login > x-terminal.shell: F 32:32(0) ack 3 win 4096
```


Intrusion Detection Systems - IDS

- IDS Systems can be defined as the tools, methods and resources to help identify, assess, and report unauthorized or unapproved network activity
- Loosely compare IDS Systems to an alarm system
- IDSs work at the network layer, they analyze packets to find specific patterns, if so an alert is logged
- Similar to antivirus software, i.e. use known signatures to recognize traffic patterns

IDS Types

- *Host-based intrusion detection system* (HIDS):
  - Requires software that resides on the system and can scan all host resources for activity
- *Network-based intrusion detection system* (NIDS):
  - Analyzes network packets looking for attacks
  - Recieves all packets on a particular network segment via taps or port mirroring
- *Hybrids of the two:*
  - *combines a HIDS with a NIDS*

Basic Process for an IDS

- Information Flow – collects data, preprocess and classifies them
- Exploit Detection – determine if information falls outside a normal activity, is so, it is matched against a knowledge base
- If a match is found, an alert is sent

Information Flow

- Raw packet capture
  - Must save raw packets so they can be processed
- Filtering
  - Filter out certain types of packets that are not interested
  - Example: capture only TCP traffic
  - Desirable in very high speed networks
- Packet decoding
  - Packets are sent to a series of decoder routines that define the packets structure
  - packets that cannot be properly decoded are dropped

Information Flow

- Storage
  - Packet decoded are often stored in a file or into a data structure
- Fragment reassembly
  - Critical consideration: which fragments will be retained
  - Information needed: packet header
  - Retaining only the first fragment more efficient
- Stream reassembly
  - Important when data arrives in different order

Exploit Detection

- Signature matching
  - A string that is a part of what an attack host send to an intended victim that uniquely identifies a particular attack
- Rule-based matching
  - Based on combinations of possible indicators of attacks, aggregating them to see if a rule condition is fulfilled
- Profile-based matching
  - When a users action deviates to much from a normal pattern, the profiling system flags this event and passes info to output routines

Snort

- Function
  - Command-line use only
- Write your own rules
  - Set custom filters
  - Automate update of signatures
- User's Manual and Tutorial
  - http://www.snort.org

Modes of Operation

- Three general operational modes
  - Sniffer
  - Packet logger
  - NIDS (Network Intrusion Detection System)
- Run-time mode is determined by command-line switches
- Variables for writing own rules and filters available

Sniffer Mode

- Sniff and dump packets to standard output (or to the screen)
- Run-time switches
  - Verbose mode: **-v**
  - Dump packet payloads: **-d**
  - Display ARP packets: **-a**
  - Display link layer data: **-e**
- For example:
  - `snort -dvae`

How Does Snort Differ from tcpdump?

- Snort is descriptive and verbose
  - tcpdump output in hexidecimal is primitive and esoteric
- Snort determines each entry's value
  - It identifies the individual fields
- Snort computes the corresponding fields
  - It does not print out all the fields in the headers
  - No Snort output for version number or checksums

Snort Packet Logger Mode

- Tell Snort to output packets to a log file
- Command line options
  - Dump packets into <logdir>: **-l <logdir>**
- Examples
  - `snort -l /var/log`

What to do with binary logs?

- Snort binary logs are kept in "tcpdump" format
- These can be read back through Snort using the '-r' command line switch
- Example
  - `snort –dvr /var/log/snort/snort01.log`
- Readback can be used to dump, log (again), or perform detection on packets in the log file

NIDS Mode

- Load Snort with a full set of rules, configure packet analysis plug-ins and allow it to monitor your network for hostile activity
- Snort at its most complex
  - Variety of options for packet analysis and logging
  - Runs in "real-time" mode
  - Generates alerts
  - Logs offending packets

NIDS Configuration

- Specify a configuration file
  - `snort -c snort.conf`
  - Automatically puts Snort in NIDS mode
- Default configuration
  - Output directory is `/var/log/snort`
  - Alert mode is full

Snort Rules

- Simple format with flexibility
  - Define the "who" and "what" that Snort looks for
  - Inspects packet header, payload or both
  - Standard rules alone are enough to detect attacks or interesting events
  - Multi-packet events or attacks are best detected with preprocessors
- http://www.snort.org/docs/writing_rules/
  - Lots of data here, more than a few slides' worth

Snort Rule Anatomy

- Each rule has 2 parts:
  - Rule header
  - Rule options
- Specific syntax for both
- Rule header is *required*, rule options are not
- Rule may be on multiple lines if the "\" continuation character is used
  - Each rule is typically a single line

Rule Headers and Options

alert tcp !10.1.1.0/24 any > 10.1.1.0/24 any (flags: SF; ,sg: "SYN-FIN scan";)

Rule Header | Rule Option

- Headers define "who" is involved
  - Includes action, protocol, source and destination IPs, source and destination ports, *and* direction of traffic
- Options define "what" is involved
  - Tells Snort what packet attributes to inspect
  - Forms a signature for a specific attack or probe