**CprE 530**

Lecture 12

**Topics**

• IPv6

– Overview

– Packet Format

– ICMP V6

**Reasons for IPv6**

• IPv4 uses 32 bits for addresses

• Real time/streaming traffic (voice, audio)

• Security issues with IPv4

**IPv6 – Larger Address Space**

• Header format – separates state information

from dynamic routing info to simplify router

actions

• New Options

• Quality of Service

• Added Security

**IPv6 Address Space**

• 128 bits (16 bytes)

• 4 hex digits: xx:xx:xx:xx:xx:xx:xx:xx

• Can abbreviate by removing leading

zeros

– :0F: :F:

– xx:0:0:0:AD64:0:0:xx xx::AD64:0:0:xx

• CIDR Rules also supported (/ nbr of

bits)

**IPv6 Address Types**

• Address types:

– Unicast: A B

– Anycast: same first part; subnet broadcast

– Multicast

• IPv6 Address Format:

Type Prefix Address

**IPv6 Address Format**

• Common Type Prefixes

– 010 = Provider based Unicast

– 100 = Geographic Unicast

– 1111 1110 10 = Link Local

– 1111 1110 11 = Site Local

– 1111 1111 = Multicast

– 0000 010 = IPX

– 0000 001 = NSAP

**Provider Based Unicast**

• A = 8 bits = 010 + Registry

• B = variable (16 bits recommended) = Provider

• C = 24 bits = Subscriber

• D = variable (32 bits recommended) = Subnet

• E = variable (48 bits recommended) = Node

– If Ethernet, Ethernet MAC address recommended

**Reserved Addresses**

• Starts with: 0000 0000

• Unspecified Address= ::

• Loopback= ::1

• IPv4 Address:

– 0000 0000 | 88 0’s | 32 bit IPv4 Address

**Local Address**

• Starts with: 1111 1110

• Link Local:

– 10 | 70 0’s | 48 bit node address |

• Site Local:

– 11 | 38 0’s | 32 bit subnet | 48 bit node |

**Multicast**

• Starts with: 1111 1111

• 4 bits = flag

• 4 bits = scope (node local, link local,

site local, organization, global

• 112 bits = Group ID

**IPv6 Header**

**IPv6 Header**

**Packet Format**

• 40 byte base header; N byte Extension

Headers

– 4 bits = 6 (IP version)

– 4 bits = Priority

– 24 bits = Flow label

– 16 bits = Length

– 8 bits = Next Header

– 8 bits = Hop Limit

– 128 bits = Source Address

– 128 bits = Destination Header

**Next Header Codes**

• 2 = ICMP

• 6 = TCP

• 17 = UDP

• 43 = Source Routing

• 44 = Fragmentation

• 50 = Encrypted

• 51 = Authentication

**Priority (Part 0-7)**

• Congestion Controlled

– 0 = None

– 1 = background (news)

– 2 = unattended (email)

– 3 = reserved

– 4 = Attended bulk (HTTP/FTP)

– 5 = Reserved

– 6 = Interactive

– 7 = Control traffic (routing)

**Priority (8-15)**

• Noncongention Controlled

– 8 = Most redundancy

– :

– 15 = Least redundancy

**Flow Label**

• Flow Label + Source Address is unique

• Router can cache “Flow Label + Source

Address” to speed up routing

• TCP routing can take up to 70% of the

processing with IPv4

**Items not in IPv6 Headers**

• ID/Offset = only needed if handling

fragmentation/reassembly (not needed

by routers)

• No checksum = minimal value

**Extension Headers**

• Can be chained

• If Next Header = 59, last header

**Extension Header Types**

• 1 = Hop by Hop Option

• 2 = Jumbo Payload (if payload > 65535 bytes)

– up to 232-1

• 3 = Source Routing

• 4 = Fragmentation (use Path MTU Discovery)

• 5 = Authentication (Authenticates sender)

• 6 = Encrypted

**ICMPv6**

• ICMPv6: Internet Control Messaging

Protocol

• Many of the TCP/IP protocols (ARP,

etc) are covered by ICMPv6 so are no

longer needed

**Error Reporting Packet Format**

• A = Type

A B C

D

E

• B = Code

• C = Checksum

• D = Other Information

• E = Data

**Error Reporting Types**

• 1 = Destination Unreachable

• 2 = Packet too big

• 3 = Time exceeded

• 4 = Parameter problem

• 137 = Redirection

**Type 1 Codes**

• 0 = No path

• 1 = Communications is prohibited

• 2 = Source routing is impossible

• 3 = Destination address is unreachable

• 4 = Port

**Type 2 & 3 Codes**

• Type 2 Codes

– 0 = MTU exceeded

• Type 3 Codes

– 0 = Hop Count

– 1 = Fragment timeout

**Type 4 & 5 Codes**

• Type 4 Codes

– 0 = Header

– 1 = Extension Header

• Type 5 Codes

– 0 = Router finds better path

**CprE 530**

Lecture 13

**General countermeasures**

• Since IP is so ingrained in the Internet it is

hard to provide security. There are a few

general countermeasures.

– IP Filtering

– Network Address Translation (NAT)

– Virtual Private Network (VPN)

– Encrypted IPV4 & IPV6 (IPSec)

**IP Filtering**

• Routers can be configured to filter out

packets based on:

– IP Address (black listing)

• Hard to keep list current

• Hard to get off the list (DOS)

– Port numbers

• Rogue protocols use multiple ports

– Protocol types (TCP, UDP, ICMP)

• Course grain filtering

**Network Address Translation**

**Network Address Translation**

• Used to extend the address space

– Internal address ranges

• 10/8 10.0.0.0

• 172.16/12 172.16.0.0 (16 class B networks)

• 192.168/16 192.168.0.0 (class B network)

• Static NAT

• Dynamic NAT

**NAT**

• Not really designed as a security device

• Does not provide security and is often

coupled with a firewall

**Static NAT**

• One to one mapping of external

addresses to internal addresses

• Used when a small number of machines

need Internet access.

• NAT looks like a router to the inside

machines and the destination to outside

machines

**Static NAT**

Public Port Private Port

129.186.5.100 80 192.168.20.30 80

129.186.5.150 25 192.168.20.50 80

**Dynamic NAT**

• More machines on the inside than IP

addresses on the outside.

• Used for outgoing access

• Can use tunnels for servers or combine

with static NAT

• Inside can have same address range as a

valid outside network (overlapping)

B

192.168.0.0 NAT Internet

Web

129.186.5.3

192.168.0.254 207.5.2.7

192.168.0.20

192.168.0.10

1

2

3

4

**Dynamic NAT (Port mapping)**

Figure 6.31 Sample Private Network

TCP/IP

Port = 80

App

Port 8357

Public IP

Ports

Private IP

Ports

SRC DST SRC DST

129.186.5.3 NAT 80 192.168.0.10 8357 80

Computer A

NAT Mapping Table

Packet

IP Ports

SRC DST SRC DST

1 192.168.0.10 129.186.5.3 8357 80

2 207.5.2.7 129.186.5.3 NAT 80

3 129.186.5.3 207.5.2.7 80 NAT

4 129.186.5.3 192.168.0.10 80 8357

Public servers

• Servers need a public address

– Two networks

– Tunneling

**Public & Private Networks**

**Tunneling through a NAT**

**Tunneling through a NAT**

**Pass-by NAT**

C

Internet

NAT

D B

Public IP

Address

Router

A

Public IP

Address

192.168.0.254

192.168.0.10

Public IP

Address

Public IP

Address

192.168.0.20

C

Internet

A

B

D

Public IP

Address

NAT Router

Public IP

192.168.0.254 Address

192.168.0.10

Public IP

Address

Public IP

Address

192.168.0.20

PhysicalConfiguration

LogicalConfiguration

**Virtual Private Network**

• Used to created encrypted tunnels

between devices

• Uses many different protocols

– SSH

– IPSEC

– Proprietary

C

**Network to network VPN**

VPN only when talking to target network

Other traffic goes directly to destination

VPN Internet VPN Remote

Network

Main

Network

Public IP

Address Public IP

Address

Encrypted

B Authenticated A

Remote Network

**Network to network VPN**

Always uses VPN

All traffic is routed through target network

**Client to client VPN**

C

**Client to Network**

Always uses VPN

All traffic is routed through target network

Internet Client

VPN VPN Main

Network

Public IP

Address

Public IP

Address

Encrypted

B Authenticated

Remote Access

**Client to network**

Always uses VPN

All traffic is routed through target network

**IPSEC**

• Two Purposes

– Authentication: sender & receiver (prevents

IP spoofing)

– Encryption: data privacy

• IPSEC is not end-to-end

**IPSEC**

• AH = Authentication Header Not used much)

• ESP = Encapsulating Security Payload

• Not Specified in IPSEC Policy

– Encryption Algorithms

– Key Management

– Domain of Interpretation

**IPSEC Services**

AH ESP BOTH IPSEC Service

X X X Access Control

X Connectionless Integrity

X Data Origin Authentication

X X X Reject of Replay

X Confidentiality

X Limited Traffic Flow Confidentiality

**AH**

Size Field

8 bits Next

8 bits Length of Header

16 bits Reserved

32 bits Security Parameters

32 bits Sequence Number

Variable Authentication Data

Authentication Data: MD5 (1-way Hash)

AH Use: End-to-End or End-to-Intermediate Node

**IPv4 Use of AH in IPSEC**

IP Hdr TCP Hdr Data

Transport Mode IPv6 Packet

Original IPv4 Packet

IP Hdr AH TCP Hdr Data

New IP Hdr AH IP Hdr TCP Hdr Data

Tunnel Mode IPv6 Packet

<-------Original Packet----------->

**ESP**

• Encapsulating Security Payload

– Security Parameters: help identify the

encryption algorithm (eg: DES, blowfish)

– Sequence number: an ever increasing

number used for replay

– Authentication data: a hash of everything,

proves non-alteration

– Data, Padding, Length, and Next fields are

all encrypted

**Encapsulating Security Payload**

Security Parameters

Sequence number

Data

Authentication Data

Length Next

Padding

32 bits

Encrypted Authenticated

**Encapsulating Security Payload**

• There are two ways encryption can be

handled:

– Transport Level (end-to-end)

– Tunnel mode (also referred to as VPN)

• Packet format for IPv4:

IP Hdr ESP Hdr TCP Data ESP Ftr ESPAuth

authenticated

encrypted

**Encapsulating Security Payload**

• Packet format for IPv6:

New IP Hdr ESP Hdr Orig IP Pkt ESP Ftr ESPAuth

authenticated

• Tunneling mode:

encrypted

Clear text **I** Clear text

IP SEC

**CprE 530**

Lecture 14

**Topics**

• TCP Layer

– Responsible for reliable end-to-end

transfer of application data.

**TCP Services**

**Multiplexing:**

• A process within a host using TCP service is

identified with a **port**. A port, when

concatenated with an internet address, forms

a **Socket**, which is unique throughout the

internet. Service provided by TCP is provided

by means of a logical connection between a

pair of sockets.

Application

1

Application

2

Application

3

Application

N

Source

Port 1

Source

Port 2

Source

Port 3

Source

Port N

Other

**Multiplexing service**

Figure 7.1 TCP Multiplexing

TCP Layer

IP Layer

Protocol type = 17 (TCP)

Transport

Protocols

**TCP port numbers**

5 RJE 68 Bootstrap Protocol Client

7 echo 69 Trivial FTP

9 Discard 75 any private dialout service

11 Active Users 77 any Private RJE service

13 daytime 79 FINGER

15 Who is up 101 NIC host name server

17 Quote of the day 102 ISO-TSAP

19 Character Generator 103 X.400

20 FTP (default data) 104 X.400-SND

21 FTP (control) 105 CSnet Name server

23 TELNET 109 Post Office Protocol Ver 2

25 SMTP 113 Authentication Service

37 Time 115 Simple FTP

42 Host name service 119 NNTP

53 Domain name server 123 NTP

67 BOOTP 161 SNMP agent

162 SNMP management station

**TCP Connection Management**

Consists of three services:

• Connection Establishment: Allow two TCP users to

setup a logical connection between their respective

sockets. A connection may be setup if:

• No connection between the two sockets currently

exists. From a given socket, it is possible to

simultaneously maintain more than one connection,

but only one connection to any specific remote socket

at a time is permitted.

– Internal TCP resources are sufficient.

– Both users have agreed to the connection.

**TCP Connection Management**

• Connection Maintenance service provides for

the exchange of data between the two

sockets and supports the data transport

(described in the next slide).

• Connection Termination may be either abrupt

or graceful. With abrupt termination, data in

transit may be lost. A graceful termination

prevents either side from shutting down until

all data have been received.

**TCP Data Transport**

• Full Duplex: Both users may transmit at once.

• Timely: The user may request timely delivery of data by

associating a timeout with data submitted for transmission. If

TCP detects a timeout the connection is abruptly terminated.

• Ordered: TCP is stream oriented. TCP guaranteed that the

stream of data presented by one user to TCP will be delivered in

the same order to the destination user.

• Labeled: TCP establishes a connection only if the security

designation provided by both users match.

• Flow Control: Used to prevent internal TCP congestion

• Error Control: TCP uses a simple checksum.

**TCP**

• **Stream Orientation -** When two application

processes transfer large volumes of data, we can

think of the as a stream of bits divided into 8-bit

bytes The stream service on the destination passes

the same sequence of octets to the receiver that the

sender passed to the source machine. Data are not

treated as packets but as a stream of data that is

passed to the transport entity. The transport entity

will divide the data into packets for transmission to

the destination. The destination transport entity will

pass the data to the user as a stream.

**TCP**

**Stream**

**TCP Special Capabilities**

TCP supports two special capabilities associated with

the transfer of data

• Data Stream Push: Used to force the delivery of all

data waiting to be sent.

• Urgent Data Signaling: Provides a means of

informing the destination TCP user that urgent data

is in the incoming data stream.

**TCP Error Reporting**

• TCP will report service failure stemming

from catastrophic conditions

**TCP Services**

• Unspecified Passive open

• Fully Specified Passive Open

• Active Open

• Active Open with data

• Send

• Deliver

• Allocate

• Close

• Abort

• Terminate

• Error

**TCP Protocol**

Connection Establishment:

• TCP uses a three handshake for

connection establishment. We will see

TCP defines only one packet format that

contains flags to indicate what type of

packet it is. The connection packets

have the SYN flag set.

**TCP 3-way Handshake**

**TCP Protocol**

Data Transfer:

• Sequence numbers are used for data

transfer. The sequence numbers

represent the number of bytes not the

number of packets. Flow control is

handled by using a credit allocation

scheme as describe earlier.

**TCP Data Transfer**

**TCP Connection Termination**

Connection Termination:

• The connection is terminated by

sending a packet with the FIN flag set.

This packet contains the number of the

last packet sent.

**TCP Connection termination**

Source Port Destination Port

Sequence Number

Acknowledgement Number

Hdr-Len Reserved Flags Window Size

Checksum Urgent Pointer

Options

Flags

**TCP Header Format**

Figure 7.6 TCP Header Format

URG ACK PSH RST SYN FIN

Flag Function

URG Packet contains urgent data

ACK Acknowledgment number is valid

PSH Data should be pushed to the application

RST Reset Packet

SYN Synchronize packet

FIN Finish packet

**CprE 530**

Lecture 15

**Topics**

• TCP vulnerabilities

• UDP

• UDP vulnerabilities

• DNS

**Header Based**

• There have been several attacks using

invalid flag combinations.

• Most have been fixed, however this is

now used to help determine the type of

operating system

– Probing attacks

• Invalid header responses

• Initial values

– sequence numbers

– Window size

**Protocol Based**

• Syn flood

• Reset Packets

• Session Hijacking

**SYN Flood**

Attacker A2

Attacker A3

Attacker A1

Victim

Internet

Attacker A4

Attacker A5

Valid User

**SYN Flood**

**Reset**

**Shutdown**

**Session Hijacking**

Victim Attacker Server

Internet

Network where the attacker

can see the traffic between the

Victim and the Server

Router

**Session Hijacking**

Victim Attacker Server

SYN

SYN + ACK

ACK

DATA

RST DATA

DATA

DATA + ACK

DATA + ACK

DATA + ACK

**Passive Network Filter**

User Filter Server

Internet

Network where the filter

can see the traffic between the

user and the server

Router

**Passive Network Filter**

**Mitigation**

• Encryption can fix Session hijacking

• Reset is harder

• Syn flood is hard

**Authentication Based**

• No authentication in TCP

• Ports might be considered an

authentication of the application

**Traffic Based**

• Flooding (using all of the TCP resources)

• QOS

• Sniffing

**User Datagram Protocol**

• Designed to allow connectionless

protocols

• Typical applications will send one

packet and wait for a single response.

Source Port Destination Port

UDP Total Length Checksum

**UDP Attacks**

• Header & Protocol: None since there is no

protocol and very simple header

• Authentication: same as TCP

• Traffic: typically not a problem. Sniffing is a

potential problem, but most UDP protocols

don’t try to hide data. Flooding is hard with

UDP.

• Mitigation: Most organizations block all

UDP except port 53 (DNS)

**Domain Name Service**

• Designed to give organizations a way of

controlling their name space

• Distributed control over computer name

to IP address mapping

• DNS normally uses UDP and port 53

– If the answer is bigger than 512 bytes, can

use TCP

Domain Names

• Tree Structure - max 128 levels, root = level 0

• Domain name: www.iastate.edu

– Each name between the dots is called a **label**

– Label <= 63 characters

• Fully qualified domain name: www.iastate.edu.

– Adds “.” at the end

• Partially qualified domain name

– Supported by the client

– The leftmost part of a domain name

– E.g., www. Gets filled in to www.iastate.edu by the client

**DNS Name Space**

**Server Types**

• Server Types

– Root Server

– Primary Server

– Secondary Server

• Can only push data from Primary to

Secondary (not Secondary to Primary)

**DNS Queries**

• DNS Queries

– Name to Address

– Address to Name

• Resolver: Client code that queries DNS

using two lookup methods:

– Recursive

– Iterative

**Reverse Query**

• IP to Name

• 129.186.5.100 – what is its name

• Query is made to:

– 100.5.186.129.in-addr.arpa.

• This way it can be parsed just like a

name

– 129 then 186 then 5 then 100

**Reverse Lookups**

• IP to Name

conversion

• Not all IP

Root Server

arpa

in-addr

addresses will

resolve to a name

Figure 7.13 DNS Reverse Name Hierarchy

129

186

5

103

103.5.186.129.in-addr.arpa..

**DNS System**

resolver

resolver

DNS

Server

Application

Next

Level

Server

IP

Figure 7.14 DNS System

cache

cache

config

cache

Application

Address

of DNS

Server

IP

Address

of DNS

Device with resolver only Server

Device with DNS server

**Recursive Query Method**

8

7

3

4

10

9

6

1

2

5

**Iterative Query Method**

Root Server

First Level

Server

First Level

Server

DNS

Figure 7.16 DNS Iterative Mode

DNS

2

4

6

8

10

What is the

IP address of

vulcan.dougj.net

dougj.net

IP address of

vulcan.dougj.net

Access web server:

vulcan.dougj.net

5

7

9

3

1

Ask this

server

**Responses**

• If the answer comes back from any

DNS server that has the answer cached

it is called unauthoritative

• To handle the stale cache issue there is

a time to live for each response.

**DNS Uses Two Messages**

• Query := two fields

– header | question

• Response := five fields

– header | question | answer | authoritative |

additional

**DNS Packet Format**

**DNS Packet Format**

**DNS Message Header**

• Header = 12 bytes

– Id = 2 bytes

– Flags = 2 bytes (see next slide)

– # of questions = 2 bytes

– # of answers = 2 bytes (0 in query)

– # of authoritative answers = 2 bytes (0 in

query)

– # of additional answers = 2 bytes (0 in

query)

**Flags Field**

• 1 bit – Q/R 0=query, 1= response

• 4 bits – opcode

– 0 = standard

– 1 = inverse

– 2 = server status request

• 1 bit AA – 1 = Authoritative answer

• 1 bit TC – 1 = answer > 512 bytes

• 1 bit RA – 1 = recursion available

• 3 bits of zero

• 4 bits – response code ( see next slide)

**Response codes**

• 0 No Error

• 1 format error

• 2 problem at name server

• 3 domain reference problem

• 4 query type not supported

• 5 administratively prohibited

**DNS Question section**

• Variable length – Query name

• 16 bits – query type

• 16 bits – query class

**DNS Query Name**

• 6vulcan2ee7iastate3edu0

• Numbers are the count fields, they are

in binary

• The count fields are only 6 bits to tell

the difference between a count value

and a offset pointer used for

compression

**DNS Types**

• 1- A – Address

• 2 – NS – Name server

• 5 – CNAME – Alias

• 6 – SOA – Start of Authority

• 11 – WKS – Well known services

• 12 – PTR – IP to name conversion

• 13 – HINFO – Host info

• 15 – MX – Mail exchange

• 28 – AAAA – IPV6 address

• 252 – AXFR – Request a zones transfer

• 255 – ANY – Request all records

**DNS Resource Record**

• Domain name – Variable length (pointer

to the name in the query section

• Domain type (16 bits) same as query

• Domain class (16 bits) same as query

• Time to Live (32 bits) number of

seconds, 0 = don’t cache

• Resource data length (16 bits)

• Resource data (variable length)

**Resource data**

• Number (4 bytes – V4)

• Domain name (variable length)

• Offset pointer (upper two bits of first

byte = 11

• Char string – 1 byte length followed by

characters

**Compression**

• 11 [address of the beginning byte]

• 12 is the first byte of the question

section

**Header & Protocol attacks**

• Header

– Not many attacks, bad headers are

rejected.

– Can be used to leak data through a firewall

• Protocol

– Simple protocol

– Can use the DNS port number to

communicate through a firewall

**Authentication**

• Bad DNS Entries

– Break in DNS server

– Rouge DNS server

– DNS cache poisoning

– Bogus DNS replies

• Scope of Damage

Root Server Root Server

**DNS attack damage scope**

ISP.net

abc.com

xyz.com

Figure 7.18 DNS Attack Damage Scope

First Level

Server

DNS

Zone 1

Zone 2

Zone 3

User 1

Traffic

• DNS server flooding can cause delayed

to dropped responses. DNS client will

try 4 times so they often will get an

answer

• Sniffing is not a problem

**DNS**

• DNSSEC is a new protocol and server

that offers authenticated DNS with

certificates.

– Not widely adopted

• DNS is a major weak point in the

Internet. Taking down the DNS system

can take down the entire Internet.

Application Application Application

TLS TLS

**Transport Layer Security**

Figure 7.19 TLS Stack

TCP

IP

NET

**TLS Protocol**

**CprE 530**

Lecture 16

**Application Layer Security**

• TCP stream Service

• Socket Layer

• Common Attack Methods

**TCP Stream**

**Service**

Application Application

User A User B

Hello John:

Do you have the report finished?

We need to get it to headquarters

today

write("<start data>\r\n");

write("Hello John:\r\n\r\nDo you have the

report finished?\r\nWe need to get it to

headquarters\r\ntoday");

read() = "<start data>\r\n"

read() = "Hello John:\r\n\Do you have t"

read() = "he report finished?\r"

read() = "\nWe need to get it to head"

Hello John:

Do you have the report finished?

We need to get it to headquarters

today

<start data>

Hello John:

Do you have the report finished?

We need to get it to headquarters

today

<end data>

Internet

TCP

IP

Network

TCP

IP

Network

Hello John:\r\n\r\n Do you

have the report finished?\r

\nWe need to get it to head

quarters\n\rtoday

write("<end data>\r\n");

<start data>\r\n

<end data>\r\n

read() = "quarters\n\rtoday\r\n<end "

read() = "data>\r\n"

**Sockets**

• Application to TCP interface

• Socket protocol

**Application / TCP Interface**

Client Server

Port Port

TCP TCP

IP, Port IP, Port

Handle = OS\_Connection() Handle = OS\_Connection()

**Socket**

**Protocol**

Client

Application

Server

Application

Client OS

TCP Layer

Server OS

TCP Layer

Open OS connection

return OS handle

Bind to listen port

OK

Wait for

client

Wait

Connection request

Open OS connection

return OS handle

Connec tion Connect to IP, Port

Request to

IP and Port

Bind to a port

Accept

Conversion

Connection

Connection accepted

Finished Close Connection

Close

Close OK Client Done

**Socket Code – Server Side**

nsaddr.sin\_family = AF\_INET;

nsaddr.sin\_addr.s\_addr = INADDR\_ANY; // Accept connection from all

/\* nsaddr.sin\_addr.s\_addr = inet\_addr("129.186.5.101"); \*/

nsaddr.sin\_port = htons(2000);

// Open stream port.

if ((vs = socket(AF\_INET, SOCK\_STREAM, 0)) < 0) {

printf("socket(SOCK\_DGRAM): %d\n",errno);

exit(1);

}

// bind stream to port 2000 from any address

if (bind(vs, (struct sockaddr \*)&nsaddr, sizeof(nsaddr)) < 0) {

printf("bind(vs, %s[%d]) errno = %d\n "

,inet\_ntoa(nsaddr.sin\_addr), ntohs(nsaddr.sin\_port),errno);

perror("bind error");

exit(1);

}

fprintf(stderr,"SERVER: bind(vs, %s[%d]):\n ",

inet\_ntoa(nsaddr.sin\_addr), ntohs(nsaddr.sin\_port));

**Socket Code – Server Side**

printf("SERVER: listen waiting\n");

// allow 5 pending connection requests to this port

if ((listen(vs,5)) < 0 ) {

perror("listen");

exit(1);

}

printf("SERVER: waiting buf size = %d\n",sizeof(buf));

from\_len = sizeof(from\_addr);

// wait for incoming connection

if ((ns = accept(vs, &from\_addr, &from\_len)) < 0) perror("accept");

**Socket Code – Client Side**

// this calls the DNS system

h\_name = gethostbyname("vulcan.ee.iastate.edu");

/\* s\_name = getservbyname("phone", "udp");\*/

/\* sin.sin\_port = s\_name->s\_port; \*/

sin.sin\_family = AF\_INET;

sin.sin\_port = htons(2000); // port to connect to

sin.sin\_addr.s\_addr = \*(u\_long \*)h\_name->h\_addr;

printf("port = %d %s\n",ntohs(sin.sin\_port),

inet\_ntoa(sin.sin\_addr));

// open socket

sockFD = socket(AF\_INET, SOCK\_STREAM, 0);

// open connection to server

if (connect(sockFD, &sin, sizeof(sin)) < 0) {

perror("connect request");

(void) close(sockFD);

exit(1);

}

**Socket Code – data xfer**

strcpy(buf,"from client");

// client sends first

if (send(sockFD, buf, strlen(buf),0) != strlen(buf)) {

perror("send request");

(void) close(sockFD);

exit(1);

}

// Client waits for answer

cp = answer;

if ((n = recv(sockFD, cp, 100, 0)) < 0){

perror("SendRequest");

(void) close(sockFD);

}

cp[n] = 0;

printf("===<%s>===\n",cp);

(void) close(sockFD);

**Socket Code – data xfer**

printf("SERVER: accepted call\n");

// print where the connection came from

fprintf(stderr,"SERVER: from\_addr(ns, %s[%d]):\n ",

inet\_ntoa(from\_addr.sin\_addr), ntohs(from\_addr.sin\_port));

// get the data from the client

blen = recv(ns,buf,sizeof(buf), 0);

buf[blen] = 0;

printf("SERVER: --<%s>--\n",buf);

strcpy(buf,"hello");

printf("SERVER: sending\n");

// send response to client

if (send(ns, buf, strlen(buf), 0) != strlen(buf)) {

perror("Sendto");

}

// shutdown connection, leaves socket open

shutdown(ns,2);

**Application Layer Vulnerabilities**

• Same four categories

• Applications do have some attacks in

common

• Attacks are often limited to the application

• Can allow access to the computer

(privileged applications are a common

target)

**Header-Based**

• Common attack against applications

• Most applications have a freeform

header which means the header must

be parsed

• Buffer overflow is a common form

**Buffer overflow**

Received data Partially Valid header Filler Attack Code

Variables Input Buffer Variables Return

value

ApplicationMemory

**Protocol-Based**

• Application specific

• Often part of an authentication attack

**Authentication-Based**

• The most common type of attack

• Two types

• Direct Attack

– Using the applications authentication

mechanism to gain access (password

guessing)

• Indirect Attack

– Using one of the other attack categories to

circumvent authentication (primary focus)

**Traffic-Based**

• DOS

• Sniffing

**CprE 530**

Lecture 17

**Topics**

• Email

– SMTP

– POP

– IMAP

– MIME

**Email**

Simple Mail Transfer Protocol:

First we will look at Electronic Mail systems in general and then we will

look at SMTP. A basic electronic mail system performs four

functions:

Creation: A user creates and edits a message, generally using a

rudimentary editing capability. Most systems also allow the user to

create a message using the system editor or a word processor, and

then incorporate the resulting file as the body of the message.

**Email**

Sending: The user designates the recipient (or recipients)

of the message, and the facility stores the message in

the appropriate mailbox(es)

Reception: The intended recipient may invoke the

electronic mail facility to access and read the delivered

mail

Storage: Both sender and recipient may chose to save the

message in a file for permanent storage

**Email System**

UA UA

User

User

Storage

User

Authentication

MIME

MTA MTA

SMTP

Post Office Protocol

OS dependent

Storage

OS dependent

Storage

SMTP

OS dependent Storage

transfer

User

Authentication

Mail server Mail server

Internet

server

**Email Message Format**

UA

UA

MIME Header

Hello

John

Hello

John

MIME Header

MTA 2 Header

MTA 1 Header

Hello

John

Reduced Header

MTA 1 MTA 2

SMTP

Internet

Hello

John

Hello

John

MIME Header MIME Header

MTA 1 Header

MTA 2 Header

MTA 1 Header

Hello

John

**Email**

The SMTP protocol is the standard protocol for transferring mail

between hosts. The protocol was defined in RFC 821 and later

formalized as MIL-STD-1781.

SMTP is not concerned with the format or content of the messages

themselves, with two minor exceptions.

SMTP requires a 7 bit ASCII character set.

SMTP adds logging information to message that indicates the path the

message took.

**Email**

The SMTP protocol attempts to provide reliable operation, but does not

guarantee to recover from hosts that lose files. No end-to-end

acknowledgment is returned to a message's originator when a

message is successfully delivered, and errors are not guaranteed to

be returned either. However, the mail system is sufficiently reliable

that this is not an issue.

In most cases mail goes directly from the mail originator's machine to

the destination machine. However, mail will occasionally go through

intermediate systems.

The SMTP protocol is made up of a set of simple commands.

**Email**

SMTP has 14 commands.

Command syntax is a set of 4 letter commands with parameters

Not all commands need to be implemented

The commands are:

CMD Syntax Action

HELO <domain> Used by the sending system to

identify itself

(HELOeeclass.ee.iastate.edu)

**Email**

CMD Syntax Action

MAIL FROM: <path> Identifies who the message is from.

(MAIL FROM doug@iastate,edu)

error messages have a NULL from field

to prevent answers.

RCPT TO: <forward path> Identifies who the message

should be mailed to. There is

separate RCPT for each

recipient.

**Email**

CMD Syntax Action

DATA Indicates that the next transmission

contains the message text. Terminated

with a line containing <CR LF>.<CR LF>

RSET Terminate current transaction

SEND FROM: <path> Used instead of MAIL if message should

be displayed on user's terminal.

**Email**

CMD Syntax Action

SOML FROM: <path> (Send or Mail) Used instead of MAIL if

message should be mailed or displayed

on user's terminal.

SAML FROM: <path> (Send And Mail) Used instead of MAIL if

message should be mailed and displayed

on user's terminal.

VRFY <string> Returns to the sender the full name of

the user specified in the parameter

EXPN <string> Returns to the sender a list of mailboxes

corresponding to the alias provided

**Email**

CMD Syntax Action

NOOP Performs no actions: returns a "250 OK"

for debugging

QUIT Sent after completion of transfer, prior

to closing TCP connection.

TURN Reverses the role of SMTP sender and

receiver.

A reply code is returned for each command sent. The next slide

shows the reply code format.

**Email**

The reply codes are designed to make implementation of SMTP easier.

Each digit of the three digit code has a unique purpose.

First digit specifies whether the response was good, bad, or or

incomplete.

The second digit specifies what type of error occurred.

The third digit details specific failures.

The values for the codes are given on the next slide.

**Email**

1XX Positive Preliminary Reply - The command has been

accepted, but the receiver requires more information. (not used by

SMTP, used by other protocols)

2XX Positive Completion Reply - The requested action has been

successfully completed. A new request may be initiated.

3XX Positive Intermediate Reply - The command has been

accepted, but action is being held, pending receipt of further

information. The SMTP sender should send another command

specifying this information.

**Email**

4XX Transient Negative Completion Reply - The

command was not accepted, however, the error

condition is temporary

5XX Permanent Negative Completion Reply - The

command was not accepted.

**Email**

X0X Syntax Error or unimplemented commands

X1X Information: reply to requests for information

X2X Connections - reply to the request for connection

X3X Unspecified

X4X Unspecified

X5X Mail System - indicates the status of the receiver during, for

example, a transfer.

The next slide has come common reply codes.

**Email**

211 System status or system help reply

214 helpmessage

220 service ready

221 Service closing transmission channel

250 Requested mail action okay, completed

251 User not local; will forward to <forward path>

354 Start mail input

421 Service not available; closing channel

450 Mail box busy

451 requested action terminated; local error in processing

452 Requested action not taken; insufficient system storage

**Email**

500 Syntax Error, command unrecognized

501 Syntax Error in parameters or

arguments

502 Command not implemented

550 mailbox not found

551 user not local; please try <forward

path>

554 transaction failed

**SMTP**

SMTP

Client

SMTP

Server

Open TCP Connection to port 25

220 mail.foo.bar SMTP Ver; Date and time

HELO machine.iseage.org

250 mail.foo.bar Hello machine.iseage.org [IP]

MAIL FROM: dougj@iseage.org

250 dougj@iseage.org... Sender ok

RCPT TO: mary

550 mary... User unknown

Hello John

RCPT TO: john

250 john... Recipient ok

DATA

354 Enter mail, end with "." on a line by itself

250 ID Message accepted for delivery

QUIT

"."

**Header based**

• Not common

• Some buffer overflow issues in old

implementations

**Protocol Based**

• Not common in command/response

protocols

• Out of order commands are reported back

as errors

• Multiple open connections could limit

access to the email server.

**Authentication Based**

• Most common attack

• No authentication in SMTP

• Sender tells MTA the name of the sender

• Spam and phishing attacks

• Sometimes we want to spoof the senders

address (email relay)

**Email Address Propagation** Client - UA

h1.hacker.net

MTA 1

mail.domain1.net

MTA 2

mail.domain2.net

HELO h1.hacker.net

250

250

250

MAIL FROM: john@iseage.org

RCPT TO: joe@issl.org

HELO mail.domain1.net

250

250

MAIL FROM: john@iseage.org

RCPT TO: joe@issl.org

MTA 3

mail.domain3.net

250

UA Header

Mail Message

MTA1 Header

UA Header

MailMessage

MTA 1 Header

MTA 2 Header

UA Header

Mail Message

MTA 1 Header

MTA 2 Header

MTA 3 Header

HELO mail.domain2.net

250

250

250

MAIL FROM: john@iseage.org

RCPT TO: joe@issl.org

UA Header

Mail Message

**Email Relay**

UA

Host 1

MTA 1

iseage.org

SMTP

Internet

UA

Host 2

mary

john

MAIL FROM: mary@iseage.org

RCTP TO: john@issl.org

MAIL FROM: john@iseage.org

RCTP TO: john@issl.org

SMTP

UA

Host 3

Mail Message

jill

MAIL FROM: jill@iseage.org

RCTP TO: john@issl.org

To: john@issl.org

From: john@iseage.org

Mail Message

To: john@issl.org

From jill@iseage.org

Mail Message

To: john@issl.org

From:mary@iseage.org

**Traffic Based**

• Flooding of the email server

– Too many messages

– Messages are too large

– Sending email to B from A with C as the

return address could cause an attack on C

• Sniffing

**General Countermeasures**

• STARTTLS cause SMTP to use transport

layer security (encrypted traffic)

• AUTH provides a method for users to

authenticate with the MTA.

• Typically used for remote access to MTA

for relaying

• Being discussed as a method to reduce

spam

**CprE 530**

Lecture 18

**Topics**

• Email

– POP & IMAP

• Protocol

• Vulnerabilities and countermeasures

– MIME

• Vulnerabilities and countermeasures

**Local User Agent**

UA

OS dependent

User

Authentication

MTA

OS dependent

Storage

transfer

Mail server

Local User agent

**Remote User Agent**

UA

User

Storage

MTA

Post Office Protocol

OS dependent

Storage

SMTP

User

Authentication

Mail server

Remote User Agent

**Remote access to local UA**

UA

User

Authentication

User's

Host

WEB or other

protocol

MTA

OS dependent

Storage

OS dependent

transfer

Mail server

Remote Access to Local User agent

User

Storage

**POP**

Post Office Protocol

Used to transfer mail between the mail

server and a PC

Provides user Authentication

**POP3 protocol**

• POP3 client “logs in” to a POP3 server (TCP

port 110)

• Login name and password in clear text

• User can configure how often mail is checked

– this means the login and password can be sent many

times a day

– easy to capture since when there is no mail there are

only a few packets exchanged.

**POP3 block diagram**

POP3

inbound Server

UA

SMTP

outbound Server

**POP3 Commands**

• USER name Login name

• PASS string User password

• STAT returns number of messages

• LIST [msg] returns the size of msg or all messages if

[msg] is not supplied

• RETR msg send client the full message [msg]

• DELE msg Delete message from server

• NOOP No operation

• RSET Reset deletion indicators

**POP3 Commands**

• Quit Quit the session

• APOP name digest Optional authentication

• TOP msg n return first n lines of message

• UIDL returns a unique ID string for the

requested message, does not change

during session. Message ID can used

to request message.

**POP3 Responses**

• Two response codes

– -ERR message

– +OK message

**POP3**

**Protocol**

POP3

Client

POP3

Server

Open TCP Connection to port 110

+OK POP (version X.X) at mail.foo.bar

USER john

+OK Password required for john

PASS mypassword

+OKjohn has 4 messages

RETR 1

+OK 486 octets

Message 1 terminated with a <cr><lf>.<cr><lf>

+OK Message 1 has been deleted

+OK Pop server at mail.foo.bar signing off

QUIT

DELE 1

**IMAP**

• Supports message retrieval

• Support message filing

• POP, does not work well in a multiple

client configuration since mail is deleted

after it is read.

• IMAP can keep messages on the server

and an be used by multiple clients.

**IMAP Mail Boxes**

IMAPd

User

Authentication

UA

IMAP User

Mailboxes

MTA

OS dependent

Storage

Mail server

Remote Access to Local User agent

Local

User

Mailboxes

SMTP

User's Local

Computer

**IMAP Commands**

• CAPABILITY List server capabilities

• NOOP No operation

• LOGOUT

• AUTHENTICATE type

• LOGIN name passwd

• SELECT mailbox

• EXAMINE mailbox read only version of select

• CREATE mailbox

• DETELE mailbox

**IMAP COMMANDS**

• RENAME current-name new-name rename mailbox

• SUBCRIBE mailbox add mailbox to servers list of

active mailboxes

• UNSUBSCRIBE mailbox

• LIST ref mailbox provide a list of mailboxes

• LSUB provide a list based on subscribe

• APPEND mailbox mess Append the message to the mailbox

• CHECK Flush mailboxes to disk

• CLOSE Close mailbox, all messages

marked as deleted are removed

**IMAP Commands**

• EXPUNGE Remove messages marked as deleted

• SEARCH criteria Search the mailbox for messages that match

• FETCH message-setget message

• PARTIAL message len get partial message

• STORE

• COPY message-set Mailbox copy a message to another mailbox

• UID gets unique ID for messages

**Header & Protocol based**

• Very few header or protocol based attacks

**Authentication Based**

• User authentication over the network

• Password guessing using POP or IMAP

• Every attempt can be logged

• Restrict POP and IMAP authentication to

know IP addresses

• Use web client for remote access

**Traffic Based**

• Flooding is not much of an issue

• Sniffing is an issue

– There are encrypted versions of both IMAP

and POP, but they are not widely used.

**MIME**

• Multipurpose Internet Mail Extensions

• Email message format

– Embedded pictures

– Embedded code

– Attachments

**Encode and Decode**

Encode

Program

Decode

Program

encode Header encodeHeader

MTA 2 Header

MTA 1 Header

Binary File

MTA 1 MTA 2

SMTP

Internet

ASCII ASCII

UA

ASCII File

UA

Save email to

a file

Copy ASCII

Data into

Email

SMTP Headers **MIME Structure**

MIME Version

MIME Headers

Email Object

MIME Headers

Email Object

MIME Headers

Email Object

MIME Headers

Email Object

**MIME Headers**

**MIME Header Function**

MIME-Version: Indicates a MIME message. The current version is 1.1

Content-Type: Indicates the type of content contained in the message

Content-Transfer-Encoding: Indicates how the content is encoded

Content-Id: Optional Identifier used for multiple messages

Content-Description: Optional description of the object that can be displayed by

the user agent

Content-Disposition: Optional description of the method to use to display the

object in receiving the user agent

**Content-Type**

Type Subtype Description

Plain Unformatted text

Text Html Text in HTML format

Multipart

Mixed Multiple ordered objects

Parallel Multiple object, not ordered

Digest Multiple ordered RFC822 objects

Alternative Alternate methods of representing the same object

Message

RFC822 Encapsulated message

Partial Part of a larger message

External-Body Object is a reference to an external message

Image

JPEG JPEG Image

GIF GIF Image

Video MPEG MPEG movie

Audio Basic Audio object

Application

Postscript Adobe Postscript object

Octet-stream 8 bit binary object

**Multipart MIME**

• Next three slides show a multipart MIME

message

**Content-Description**

**Content-Disposition**

• Content-Description: <description>

– Lets user “tell” the User Agent what type of file is

attached

– Allows malicious code to look like something

else

• Content-Disposition: (Inline, Attachments)

– Allows inline documents which will be displayed

by the user agent

– Allows malicious code be open automatically

**Header based**

• Headers can be used to hide actual

content type

• HTML documents with hyperlinks where

the text is different than the link

• Countermeasures:

– User education

**Protocol Based**

• Different that normal protocols (no

message exchange)

• Attachments can be malicious (viruses,

worms, Trojan horses.

• Some can be auto opened (inline)

• Countermeasures:

– Disable UA functions

– Scanners, filters

– Education

**Authentication Based**

• MIME does not support authentication

• Can support email monitoring

– “Web Bugs”

• 1x1 pixel picture stored on a web site

• When email is read the file is downloaded

• Web server will log access to the file and

information about the machine that accessed it.

• Countermeasures:

– Disable User Agent function to auto display

pictures

**Traffic Based**

• Enables flooding of the email server

– Large messages

• Sniffing

**CprE 530**

Lecture 19

**General Email Countermeasures**

• Encryption & authentication

• Email filtering

• Content Filtering

• Email Forensics

**Encryption & Authentication**

POP/

IMAP

WEB

User level

SMTP

MTA MTA

SMTP

SMTP

POP/IMAP

Server

Web

Server

ABC

Hash

Function

Hash

ABC

Signed

Compress

& Encrypt

One time

Session Key

Key

Generator

RSA

Recipient's

Public Key

Recipient's ID

**PGP**

**Encryption**

Message

E(Key)

To

ASCII

ASCII

RSA

Sender's

Private

Key

ABC Hash

Function

Hash

ABC

Signed

Compress

One time

Session Key

RSA

Sender's

Public

Key

Hash

Compare

Sender ID

**PGP**

**Decryption**

& Encrypt

RSA

Message

E(Key)

From

ASCII

ASCII

ID

Recipient's

Private

Key

**Email Filtering**

• Check email

– Based on email addresses

– Based on domain address

– Based on malicious payload

• Either Block, pass, or modify the email

**Email Filtering**

Internet MTA

SMTP

MTA

SMTP Unmodified

Log

Email Filter

Mail server

Add

Header

**Spam Filter**

• Uses learning to decide what content is

spam.

• System is “trained” to know is spam

• Spam filter will mark the message as

spam.

• Some User agents support spam detection

and will move spam email into a spam

folder

**Bypassing a Spam Filter**

• Keyword loading

• Misspelled keywords

• Picture only

• Picture with background words

**Filtering list**

• Blacklist

– A list of bad users & domains

– Spammers just change domains

• Whitelist

– A list of good users and domains

– Very restrictive

**Greylist**

• Reject all email with a temp reject

• Maintain a whitelist that is not subject to

filtering

• Add machines to the grey list when they

resend the email

**Greylist**

Sender Email Filtering

MTA

220 mail.foo.bar SMTP Ver; Date and time

HELO machine.iseage.org

250 mail.foo.bar Hello machine.iseage.org [IP]

Spam Bot

MAIL FROM: dougj@iseage.org

250 dougj@iseage.org... Sender ok

RCPT TO: mary

451 Please try again later

IP address +

Sender +

Recipient

Spam Bot

moves on

A realMTA

will try later White list &

Attempt list

**Greylist**

In white

List

IP address +

Sender +

Recipient

Pass Email

New

Yes

No

Yes Create

Record

Tmp Fail

Email

Yes

Too

early

Tmp Fail

No

No

Pass Email

**Bypassing a grey list**

• Use real MTA to send email

**Content filter**

• Examine the payload for:

– Viruses

– Worms

– Trojan horses

• Often based on a signature

• Requires constant update of signatures

**Outbound content filtering**

• Used to keep private information from

leaving

– SS Numbers

– Account Numbers

– Medical records

• Will either log, stop, or encrypt violating

emails

**Bypassing a content filter**

• Encryption

– There are encrypted viruses

• Compression

**Email Forensics**

A B C D

HTTP web SMTP MTA SMTP MTA SMTP

MTA

10.70.63.2

9/5/2006

8:34:31 PDT

10.65.237.1

9/5/2006

8:34:31 PDT

nf-out-0910.email.mta

9/5/2006

8:34:32 PDT

vulcan.ece.mail.spam

9/5/2006

10:36:59 CDT

From: 192.168.182.188

**Email Forensics**

C

D

Received: from nf-out-0910.email.mta (nf-out-0910.email.mta

[192.168.182.188])

by vulcan.ece.mail.spam (8.12.8/8.9.3) with ESMTP id

k85FaxBT1486661

for <john@ee.mail.spam>; Tue, 5 Sep 2006 10:36:59 -0500 (CDT)

Received: by nf-out-0910.email.mta with SMTP id p77so1381355nfc

for <john@ee.mail.spam>; Tue, 05 Sep 2006 08:34:32 -0700 (PDT)

DomainKey-Signature: a=rsa-sha1; q=dns; c=nofws;

s=beta; d=spammer.fake;

h=received:message -id:date:from:to:subject:mime -

A

B

version:content-type;

b=BD9tHbNaozYZj9gNQqXmkrnHNA3N8+3W4NApcFJkKsKyX8DdOTS7Dp1VNunGx66SLcU5r

YiDxCnY6SuVCktWq73DDH7MYEfWgaOtYdl/hILBIRVNcbLxGtyCoIT7I8use4F4RgCzZWc3

Oc6fjqNzgGLe5s3RFQ9eVPhS+HxW+DA=

Received: by 10.65.237.1 with SMTP id o1mr4809264qbr;

Tue, 05 Sep 2006 08: 34:31 -0700 (PDT)

Received: by 10.70.63.2 with HTTP; Tue, 5 Sep 2006 08:34:31 -0700 (PDT)

Message-ID:

<ab156e9f0609050834v528b5b2eld9204458fe6409a1@mail.spammer.fake>

Date: Tue, 5 Sep 2006 10:34:31 -0500

From: "Harry Mudd" <Harry6502@spammer.fake>

To: john@ee.mail.spam

Subject: mail trace 2

MIME-Version: 1.0

**Email Forensics**

MTA MTA MTA

A B C D

SMTP SMTP SMTP

MTA MTA

E F

MTA

SMTP SMTP SMTP

vulcan.ece.mail.spam

9/5/2006

10:45:06 CDT

from172.21.4.7

magellan.sender.mta

9/5/2006

10:42:40 CDT

vulcan.ece.mail.spam

9/5/2006

10:45:28 CDT

From:192.168.182.188

despam-3.mail.spam

9/5/2006

10:42:55 CDT

from:192.168.16.211

devirus-2.mail.spam

9/5/2006

10:38:34 CDT

from:172.16.7.5

pop-5.mail.spam

9/5/2006

10:42:55 CDT

from:172.16.7.10

babylon4.ece.mail.spam

**Email**

**Forensics**

B

C

D

E

F

Received: from pop-5.mail.spam (pop -5.mail.spam [172.16.7.12])

by vulcan.ece.mail.spam (8.12.8/8.9.3) with ESMTP id

k85FjSBT1508024

for <john@EE.MAIL.SPAM>; Tue, 5 Sep 2006 10:45:28 -0500 (CDT)

Received: from devirus -2.mail.spam (devirus -2.mail.spam [172.16.7.10])

by pop-5.mail.spam (8.12.11.20060614/8.12.11) with SMTP id

k85Fgt28016542

for <john@mail.spam>; Tue, 5 Sep 2006 10:42:55 -0500

Received: from (despam -3.mail.spam [172.16.7.5]) by devirus-2.mail.spam

with smtp

id 0df9\_ae8af2c2\_3cca\_11db\_969a\_ 001372537fef;

Tue, 05 Sep 2006 10:38:34 +0000

Received: from magellan.sender.mta (magellan.sender.mta

[192.168.16.211])

by despam-3.mail.spam (8.12.11.20060614/8.12.4) with ESMTP id

k85FgttT020053

for <john@mail.spam>; Tue, 5 Sep 2006 10:42:55 -0500

Received: from vulcan.ece.mail.spam (vulcan.ece.mail.spam [ 172.20.5.6])

by magellan.sender.mta (8.13.6/8.13.6) with ESMTP id

k85Fgemo030599

for <dwj@sender.mta>; Tue, 5 Sep 2006 10:42:40 -0500 (CDT)

(envelope-from john@mail.spam)

Received: from [172.21.4.7] (babylon4.ece.mail.spam [ 172.21.4.7])

by vulcan.ece.mail.spam (8.12.8/8.9.3) with ESMTP id

A

Spam

Filters

k85Fj6BT1501144

for <dwj@sender.mta>; Tue, 5 Sep 2006 10:45:06 -0500 (CDT)

Message-ID: <44FD9AEC.4040103@mail.spam>

Date: Tue, 05 Sep 2006 10:42:36 -0500

From: Harry Mudd <Harry@mail.spam>

Organization: ISU Information Assurance Center

User-Agent: Mozilla Thunderbird 1.0.7 (Windows/20050923)

X-Accept-Language: en-us, en

MIME-Version: 1.0

To: Dave Johnson <dwj@sender.mta>

Subject: test 4

Content-Type: text/plain; c harset=ISO-8859-1; format=flowed

Content-Transfer-Encoding: 7bit

X-Filter-MailScanner-Information: Please contact the ISP for more

information

X-Filter-MailScanner: Found to be clean

X-Filter-MailScanner-SpamCheck: not spam, SpamAssassin (score= -2.6,

required 6, autolearn=not spam, BAYES\_00 -2.60, SPF\_PASS -0.00)

X-Filter-MailScanner-From: john@mail.spam

X-PMX-Version: 5.2.0.264296, Antispam -Engine: 2.4.0.264935, Antispam -

Data: 2006.9.5.82442

X-Perlmx-Spam: Gauge=IIIIIII, Probability=7%, Report='\_\_C230066\_ P5 0,

\_\_CP\_URI\_IN\_BODY 0, \_\_CT 0, \_\_CTE 0, \_\_CT\_TEXT\_PLAIN 0, \_\_HAS\_MSGID 0,

\_\_MIME\_TEXT\_ONLY 0, \_\_MIME\_VERSION 0, \_\_SANE\_MSGID 0, \_\_USER\_AGENT 0'

**Email Forensics**

A B C

MTA SMTP MTA SMTP

MTA SMTP

ns09.egujarat.net

9/9/2006

22:29:41

Logged

into

MTA

despam-2.mail.spam

9/9/2006

15:18:28 CDT

from: 202.149.46.162

ns09.egujarat.net

9/9/2006

22:29:41

from 127.0.0.1

**Email**

**Forensics**

A

B

C

D

(Removed local headers)

Received: from ns09.egujarat.net (202 -149-46-162.static.exatt.net

[202.149.46.162] (may be forged))

by despam-2.iastate.edu (8.12.11.20060614/8.12.4) with ESMTP id

k89KIRCr017274

for <dougj@iastate.edu>; Sat, 9 Sep 2006 15:18:28 -0500

Received: from ns09.egujarat.net (localhost.localdomain [127.0.0.1])

by ns09.egujarat.net (8.13.5/8.13.5) with ESMTP id

k89H5sYI007263

for <dougj@iastate.edu>; Sat, 9 Sep 2006 22:37:19 +0530

Received: (from administrator@localhost)

by ns09.egujarat.net (8.13.5/8.13.5/Submit) id k89Gxf4q006335;

Sat, 9 Sep 2006 22:29:41 +0530

Date: Sat, 9 Sep 2006 22:29:41 +0530

Message-Id: <200609091659.k89Gxf4q006335@ns09.egujarat.net>

To: dougj@iastate.edu

Subject: Password change required!

From: "eBay Inc." <admi n@eBay.com>

Content-Type: text/html

X-egujarat-MailScanner-Information: Please contact the ISP for more

information

X-egujarat-MailScanner: Found to be clean

X-MailScanner-From: administrator@ns09.egujarat.net

Spam

Filter 2

Spam

Filter 1

X-PMX-Version: 5.2.0.264296, Antispam -Engine: 2.4.0.264935, Antispam -

Data: 2006.9.9.124943

X-Perlmx-Spam: Gauge=XXXXXXXXXIIIIIIIII, Probability=99%,

<p><img src="**http://pics.ebaystatic.com/aw/pics/navbar/eBayLogoTM.gif** "

width="150" height="70"></p>

<BR>

Dear sir, <BR>

<BR>

We recently have determined that different computers

have logged onto your eBay account, and multiple

password failures were present before the logons. We strongly advice

CHANGE YOUR PASSWORD. <BR>

<BR>

If this is not completed by <STRONG>September 15,

2006</STRONG>, we will be forced to suspend your

account indefinitely, as it may have been used for fraudulent purposes.

Thank you for your cooperation. <BR>

<BR>

<A

href="http://linux.net zero.idv.tw/~ming/.change/index.php?MfcISAPIComma

nd=ChangeFPP"

target=\_blank>Click here to Change Your Password</A></TD>

Logo

Phishing

Site

**CprE 530**

Lecture 20

**Topics**

• WWW

• HTTP: Hyper Text Transfer Protocol

– Requests

– Responses

– Headers

**World Wide Web**

World Wide Web

Link to another

URL

(Document

Location)

Document

URL

**World Wide Web**

**Web Client/Server**

**HTTP**

• Hypertext Transfer Protocol

• Simple command/response protocol

• ASCII based commands

• Typically a new connection for each

command/response exchange

• Server runs on port 80 default

**HTTP Request & Response**

**HTTP Requests**

• Three parts:

– Request line

– Headers

– Blank line

– Body (optional)

• Request line looks like this:

Request type <sp> URL <sp> HTTP version

Example: GET http://www.ibm.com HTTP/1.1

More on request types later

**URL**

• Uniform Resource Locator

• A URL follows this format:

method://host:port/path

• The host can be a machine name or IP

address

• The port must be specified if the server

is running on a port other than 80.

• The path is the directory where data is

stored

**Request Types**

• GET

• HEAD

• POST

• PUT

• PATCH

Many of these types can pose

security problems, since they

involve modifying or deleting

data.

• COPY

• MOVE

• DELETE

• LINK

• UNLINK

• OPTION

Most servers only implement

the first three types: GET,

HEAD, POST

**Request Types**

Type Action

GET Retrieve a document specified by the URL.

HEAD Retrieve the headers from the document specified by the URL.

(Response does not contain the body.)

POST Provide data to the server.

PUT Provide new or replacement document specified by the URL.

(Disabled)

PATCH Provide differences to document specified by the URL in order to

change the document. (Disabled)

COPY Copy the document specified by the URL to the file specified in the

header. (Disabled)

MOVE Move the document specified by the URL to the file specified in the

header. (Disabled)

DELETE Delete the document specified by the URL. (Disabled)

LINK Create a link to the document specified in the URL. The name of the

link is specified in the header. (Disabled)

UNLINK Remove the link specified in the URL. (Disabled)

OPTION Ask the server what options are available.

**Response Message**

• Four parts:

– Status line

– Headers

– Blank line

– Body

• The status line looks like this:

HTTP version <sp> status code <sp> status phrase

Examples: HTTP/1.1 404 File not found

HTTP/1.1 200 OK

**Response Status Codes**

• Status codes follow a similar format to FTP

and SMTP status codes

• 3 digit ASCII

– 1xx informational

– 2xx success

– 3xx redirection

– 4xx client error

– 5xx server error

**Example Response Codes**

Code Phrase Meaning

100 Continue First part of the request has been received. The client can

continue.

200 OK Successful request

204 No Content The body contains no content

302 Moved permanently The document specified by the URL is no longer on the server.

304 Moved temporarily The document specified by the URL has temporarily moved.

400 Bad request The request contained a syntax error.

401 Unauthorized The authentication failed for the requested document.

403 Forbidden The service requested is not allowed.

404 Not found The document requested is not found.

405 Method not allowed The method requested in the URL is not allowed.

500 Internal server error The server failed.

501 Not implemented The requested action can not be preformed by the server.

503 Service unavailable The request cannot be accomplished right now, try again later.

**HTTP Headers**

**HTTP Headers**

• Headers have three parts:

– General header

– Request or response header, depending on whether the

header precedes a request or a response

– Entity header

• The general header contains the following fields:

Header Function

Cache-control Used to specify information about the client side cache.

Connection Indicates whether the connection should be closed.

Date Provides the current date.

MIME-version Indicated the MIME version being used.

Connection Use to determine connection type.

Keep-Alive Used to manage keep-alive connection.

**HTTP Headers**

• The Request header may contain the following fields

(all are optional):

Header Function

Accept Indicates which data formats the browser can accept.

Accept-charset Indicates the character set(s) the browser can accept.

Accept-encoding Indicates what encoding methods the browser can process.

Accept-language Indicates what language the browser can accept.

From Provides the e-mail of the user on the browser.

Host Provides the host and ephemeral port of the browser.

Referrer Provides the URL of the linked document.

User-agent Provides information about the browser software.

**HTTP Headers**

• The response header may contain the following

fields

Header Function

Accept-range Indicates the server accepts the range requested by the

browser.

Retry-after Indicates the date when the server will be available.

Server Provides the server application name and version.

**HTTP Headers**

• The entity header may contain the following

fields:

Header Function

Allow Provides a list of methods allowed for the URL.

Content-encoding Indicates the encoding method for the document.

Content-language Indicates the language of the document.

Content-length Indicates the length of the document.

Content-location Real name of the document requested.

Content-type Indicates the media type of the document.

Etag Provides a tag for the document.

Last-modified The date the document was last modified.

**HTTP Summary**

• Request:

Request line

General Header

Request Header

• Response:

Status line

General header

Request header

Entity header

Blank line

Optional Body

• Note: the entity header does

not always appear in the

request

Entity header

Blank line

Body

**HTTP Protocol Exchange**

Request Line

General Header

Request Header

**HTTP Request**

No Entity Header

Blank Line

No Body

Status Line

General Header

Response Header

**HTTP**

**Response**

Blank Line

Entity Header

HTML document (1456 bytes long)

Request Line

General Header

**HTTP Request**

Request Header

No Entity Header

Blank Line

No Body

Status Line

General Header

Response Header

**HTTP Response**

Blank Line

Entity Header

GIF image (2326 bytes long)

Request Line

General Header

Request Header

**HTTP Request**

No Entity Header

Blank Line

No Body

Status Line

General Header

**HTTP Response**

Response Header

Blank Line

Entity Header

HTML Document

**Header Based**

• Buffer overflow problems

• Server can pass HTTP requests to

back-end servers and applications so

header problems are not just with the

WEB server

• Some header-based attacks facilitate

authentication-based attacks

• Accessing hidden pages

**Protocol Based**

• Not many protocol based attacks since

it is a command/response protocol

**Authentication Based**

• This is the most common method of attack

in the WEB.

• The web server uses HTTP to request

user credentials.

• Authentication can also be directly with the

server side application (to be discussed

later)

• Authentication is used to access pages

within a directory on the server

**WEB Authentication**

• Server challenge:

– WWW-Authenticate: Basic realm=“Text String"

• Client Challenge:

– user-ID and password, separated by a single colon

(":") character, within a base64 encoded string.

For example:

– Authorization: Basic

QWxhZGRpbjpvcGVuIHNlc2FtZQ==

**HTTP Authentication**

**HTTP Authentication**

**Web Authentication**

• Can be sniffed (traffic based attack)

• Can be guessed

• Countermeasures:

– Encrypted sessions

– Good passwords

**Traffic Based**

• Very common attacks

– Flooding

– Web Hugging

• HTTP is clear text.

– HTTP does not support encrypted sessions.

– Encrypted sessions are supported using

transport layer encryption

**HTTPS**

• Uses the Secure Socket Layer SSL

• Port 443

• Uses public key certificates

**HTTPS**

**HTTPS Certificates**

**Certificate chain of trust**

**CprE 530**

Lecture 21

**Topics**

• HTML Protocol

• HTML Security

• Server side security

• Client Side security

**HTML**

• Hypertext Markup Language

• Two parts

– Head: contains information for the browser

– Body: contains information to display on the screen

• Contains markup codes which tell the browser

how to display the page

• Each markup code is called an element or a tag

• Tags can be nested:

<tag1>

<tag2>

</tag2>

</tag1>

**HTML**

**HTML Tags**

• Basic HTML tags

<HTML> - tells browser where page starts

<HEAD> - start of head section

<TITLE> - text to be displayed in title bar

<BODY> - start of body section

<H1> - largest header size

<P> - paragraph

<BR> - break (new line)

<UL> - unordered list

<LI> - list item

<a href=“abc.com”>link</a> - hyperlink to abc.com

<img src =“red.gif”> - display the image red.gif

<APPLET> CODE=XXX </APPLET> - java applet

**HTML Example**

• Here is a simple HTML page

<HTML>

<HEAD><TITLE>simple page</TITLE>

</HEAD>

<BODY>

<H1>Simple Example</H1>

<p>

This is a simple but complete HTML page.

<p>

<a href=http://www.iastate.edu>Iowa State University</a>

</BODY>

</HTML>

**HTML Example**

**Header based**

• HTML documents with hyperlinks where

the text is different than the link

• Pictures can come from anywhere

• Links to rouge code.

• Countermeasures:

– User education

**Protocol Based**

• Different that normal protocols (no

message exchange)

• Client side downloads can be malicious

(viruses, worms, Trojan horses)

• Countermeasures:

– Scanners, filters

– Education

**Authentication Based**

• HTML does not directly support

authentication

• HTML can be used to direct you to the

wrong site, and since there is no host to

user authentication. The site may not be

the true site.

• Countermeasures:

– User education

**Traffic Based**

• Sniffing

**Server Side Security**

• HTML documents can cause

applications to be run.

• Common method is via a CGI script

• HTML documents can also front end

other applications like databases

through a CGI script

**CGI**

• Common Gateway Interface

• Allows a server to run programs and scripts

• CGI is the method for passing data back and forth

between the server and the program or script

• Variables can be passed to the program or script

either through a form or after the ‘?’ in the URL

• Examples:

http://HOST/cgi-bin/program.pl?name=bob;state=ia

***or***

<FORM METHOD=POST ACTION=/cgi-bin/program.pl>

**CGI**

**CGI**

• CGI can access additional information

through environment variables

• Environment variables are passed from the

server to the program or script

• Environment variables include:

Query\_string HTTP\_referrer

Remote\_addr HTTP\_user\_agent

Remote\_host Path\_info

Remote\_user Server\_port

Server\_name

**Header Based**

• Buffer overflow problems on CGI scripts

• Server can pass HTTP requests to

back-end servers and applications so

header problems are not just with the

WEB server

• Some header-based attacks facilitate

authentication-based attacks or allow

direct access to the web server

**Protocol Based**

• Not many protocol based attacks since

it is not a protocol.

**Authentication Based**

• Provide access to application

authentication methods.

**Traffic Based**

• No additional attacks due to CGI scripts

**Client Side Security**

**Client Side Security**

• Cookies are placed on the client

• Executable programs can be

downloaded automatically by the

browser.

– Java Scripts

– Active X

• They can send information back to the

server.

**Cookies**

• A file on the users computer in which the website can

store data

– Why cookies?

• HTTP is stateless protocol, websites like to keep state information

on your information and habits

• First implementation of cookies allowed any site to read

another website’s cookie.

• Now only the site the storied the cookie can look at it

• Example of Amazon cookie

• Netscape has one cookie file whereas explorer has a file

for each cookie

• Passwords can be in clear text

**Clear Gifs**

• One pixel gif

• Hyperlink to another site

• This allows people to track documents

**Client side Executables**

• Plugins: Applications that are part of the

browser to help read different file types

• Scripts: Programs run by the browser

often to provide inactive graphics or forms

• Downloads: Programs that are

downloaded using the browser

**Header/Protocol Based**

• Not many attacks in these categories

since there is not really a separate

header or protocol.

**Authentication Based**

• No authentication of applications leads to

malicious code

• Client side executables provide a method for

attackers to interject code

– Trojan horses

– Spyware

– Key loggers

• Can be coupled with email attacks (using

phishing to direct a user to a web side which

downloads code

**Authentication based**

• Mitigation:

– Client side protection

– User awareness

**Traffic Based**

• Not very common since, however some

malicious programs may generate large

amounts of network traffic.

**General Countermeasures**

• Encryption and authentication

• URL Filtering

• Content filtering

**Encrypted Transactions**

• SSL

– Secure Socket Layer

– Broader application then HTTP

– Another layer to the mix, creates a secure layer

between HTTP and TCP

– Uses port 443

– Browser is shipped with certificates for support of

this service

– Communicates through an encrypted channel

**URL Filtering**

• Client side

• Proxy based

• Network based

**Client Side URL Filter**

**Proxy Based URL Filter**

**Network**

**Based**

**URL Filter**

**Connection**

**Blocking**

**Content Filters**

• Proxy based

• Network based

**Proxy Based Content Filter**

**CprE 530**

Lecture 22

**Topics**

• Remote Access

• Telnet

– Network Virtual Terminal (NVT)

– Commands

– Option Negotiation

• Rlogin

• X-Windows

**Telnet**

TELNET: a Virtual Terminal Protocol that provides

interactive access to remote computers

application The protocol defines:

Terminal

driver

OS

Terminal

• Format of data

• How control signals are passed and how

to distinguish them from data

• Data transfer mode

(half/full duplex, sync/async)

• How out-of-band signals are passed

• How data delivery is controlled

**Telnet**

NVT – Network Virtual Terminal

telnet

client

Psuedo

application

local char set local char set

terminal

terminal

device

telnet server

Port 23

TCP/IP TCP/IP

NVT char set

Local charsets of different OS’s may not be compatible. When sending over the

network, the local charset is translated to the common NVT charset by the telnet

client. The telnet server then translates the NVT charset to the local charset

OS OS

**Telnet**

**Telnet**

The virtual terminal consists of a display and a printer

• Display

– Characters are 7 bit ASCII

– Operates in scroll mode with unlimited line length, unlimited

lines per page

– Must be able to generate control signals:

Are You There Interrupt Process

Abort Output Erase Character

Erase line Break

• Printer

– Has unspecified line width and page length

– Can print the 95 ASCII graphic characters

– Can respond to the control codes:

NUL Line Feed Carriage return

**Telnet**

**Telnet Commands**

Definition Abbr code

End of subnegotiation SE 240

No Operation NOP 241

Data Mark: A stream sync character DM 242

Break BRK 243

Interrupt Process IOP 244

Abort Output AO 245

Are You There AYT 246

Erase Character EC 247

Go Ahead: turn line around for half duplex GA 249

Begin subnegotiation SB 250

WILL 251

WONT 252

DO 253

DON’T 254

Interpret as CMD IAC 255

**Telnet Commands**

How to mix user data and commands:

user data:

0 7 bit ASCII

command:

There is a special command to transfer 8 byte data

1 7 bits

**Telnet Options**

• Options can be negotiated by telnet

processes

• New options can be accommodated since

they are not part of the standard

• Three categories

1. Enhance, change, and refine NVT

characteristics

(e.g. line width)

2. Change transfer protocol

(e.g. suppress GO AHEAD)

3. Information to be passed to the host

(e.g. status, terminal type)

**Telnet Options**

This is just a subset of the options defined in many different RFC’s:

ID Name RFC Category

0 Binary transmission 856 2

1 echo 857 1

5 status 859 3

8 output line width 1

9 Output page size 1

10 Output <cr> disposition 652 1

24 terminal type 930 3

25 End of record 885 3

**Telnet Negotiation**

Option negotiation rules:

• May reject a request to enable an option

• Must accept a request to disable an option

• Options are not enabled until negotiation is

complete

• Never negotiate an option that is already true

**Telnet Negotiation**

Option negotiation commands:

• WILL Sender wants to enable the option

• WONT Sender does not want to enable the option

• DO Sender would like the other side to enable the

option

• DON’T Sender would not like the other side to enable the

option

Example 1: Side A wants to enable ECHO (857), side B agrees

A B

IAC WILL 857

IAC DO 857

**Telnet Negotiation**

• Example 2: A would like B to enable ECHO, B

agrees

A IAC DO 857 B

IAC WILL 857

• Example 3: A would like B to enable ECHO, but B

does not agree

A IAC DO 857 B

IACWONT 857

**Telnet Negotiation**

• Example 4: A would like to disable echo, B MUST

agree

A IACWONT 857 B

IAC DONT 857

• Example 5: A would like B to disable echo, B must

agree

A B

IAC DONT 857

IACWONT 857

**Telnet Negotiation**

Suboptions

SE 240 suboption end

SB 241 suboption begin

Example: A wants to set the terminal type (2Y) to vt100

A B

IAC WILL 2Y

IAC DO 2Y

IAC SB 2Y vt100 IAC SE

Direction Data Comments

CS

0xff 0xfd 0x01

0xff 0xfd 0x22

0xff 0xfb 0x05

IAC, Do Echo (request client echoes)

IAC, Do linemode (request client sends a line at a time)

IAC, Will Status (server wishes to send status info)

C S

0xff 0xfb 0x01

0xff 0xfc 0x22

0xff 0xfe 0x05

IAC, Will Echo (client will echo characters)

IAC, Won’t linemode (Client will not do linemode)

IAC, Don’t Status (client does not want server to send status information)

C S 0xff 0xfe 0x01

0xff 0xfb 0x01

IAC, Don’t Echo (tell client not to echo)

IAC, Will Echo (tell client server will echo)

C S 0xff 0xfc 0x01

0xff 0xfd 0x01

IAC, Won’t Echo (tell server client will not echo)

IAC, Do Echo (tell server it is OK to echo)

C S \r\n Login: Send authentication application prompt

C S j First char of user name

C S j Echo of the character

Repeat until enter key is pressed

C S \r\n Send carriage return + linefeed

C S \r\n Echo carriage return + linefeed

C S Password: Send authentication application prompt

C S p First char of password (server will not echo)

Repeat until enter key is pressed

C S \r\n Send carriage return + linefeed

C S \r\n Echo carriage return + linefeed

C S User is now connected and server application will send message.

**Rlogin**

• Remote login (rlogin)

• Similar to telnet, but much simpler

• Designed for unix to unix communication

• Possible for hosts to login without a password

• Uses port 513

• Sequence:

– Client sends: \0

local login name

\0

server login name

\0

terminal type

\0

– Server sends: \0

**Rlogin**

**rlogin server trust**

**rlogin**

**trust** Client host Client side user Server side user Result

A John John Trusted

Mary Not Trusted

Alice Trusted

Mary John Not Trusted

Mary Not Trusted

Alice Not Trusted

Joe John Not Trusted

Mary Not Trusted

Alice Not Trusted

Alice John Not Trusted

Mary Not Trusted

Alice Trusted

B Any User Any User Trusted

**rlogin**

**trust** Client host Client side user Server side user Result

C John John Not Trusted

Mary Not Trusted

Alice Not Trusted

Mary John Not Trusted

Mary Not Trusted

Alice Not Trusted

Joe John Not Trusted

Mary Not Trusted

Alice Not Trusted

Alice John Not Trusted

Mary Not Trusted

Alice Not Trusted

**Rlogin commands**

• Commands are distinguished by 0xFF

– Remote flow control 0x10

– Local flow control 0x20

– Window size 0x80

(asks client for current window size)

• Escape character: ~ ^d

• Everything is sent in clear text

**rlogin**

**rlogin**

Direction Data Comments

CS

john 0x00

john 0x00

xterm\34800 0x00

Client side username

Server side username

Terminal type and speed

If authentication is required (user is untrusted)

C S Password: Prompt for password

C S p First char of password (server will not echo)

Repeat until enter key is pressed

C S \r Send carriage return

C S \r\n Echo carriage return + linefeed

If authentication worked or user was trusted

C S

Data from server User is now connected and server will display the UNIX shell prompt.

**X windows**

• The user sits on the server side of X

windows

– Usually telnet into client and start X

window client

– X windows then starts and the client

authenticates to the X windows server

– X windows sends information in clear text

**X-windows**

**Communication**

• In order for two programs to

communicate in Unix, a pipe is created

between the two processes

– Pipe works like it sounds, put data in on

one side comes out the other

– Pipe created in the tmp directory

• Port 6000

**Local X-Windows**

**Server Side**

• X windows offers up your computer to the outside world to

manipulate

• Pc also has public domain X windows programs

• Xhost determines who can connect to your server

– Xhost + would allow all to connect to one’s X windows

• X windows is designed to allow applications control over the display

• Client side

– How does client know which server to connect to

– Variable called display

• :0.0 display means local display

• The second number is the monitor

• If remote machine:0.0 which is set on the client

– Tells X windows to point to server

**Server Side cont…**

• Authentication?

– Xhost command, indicates who can connect to one’s server, which is IP

address based authentication

Server

Connections

– Xhost + allow all connections

– Xhost - allows nobody

• Command set is designed to allow total control over input and

display

– Through X windows, hackers could

• Capture screen

• Capture keystrokes

• Create, destroy windows

• Enter key strokes into windows

**Local Side**

• Pipe

– /tmp/.X11 …

– Tmp directory is shared and is world read

writable

– Can do denial of service by deleting the

pipe in the tmp directory

• No new clients can connect

• Current clients stay connected

**Header Based**

• For Telnet and rlogin there is not much

of a header.

• X-Windows there is possible buffer

overflow attacks.

**Protocol Based**

• Telnet and rlogin have a simple protocol

and there is not any attacks, other than

telnet can be used to connect to any

service (not really a flaw)

• X-Windows has some issues with the

protocol since the protocol gives the

application control over the remote

computer.

**Authentication Based**

• Telnet offers access to the remote

machine and to the login prompt

• Rlogin does not need password unless

setup correctly. Uses IP address for

authenticator

• X-Windows

– server can allow any machine to control it

based on the IP address

– Client uses machine authentication to allow

a user to run the application

**Authentication Stepping stone**

**Traffic Based**

• All three are clear text (sniffing)

– Usernames & Password

– Commands and text

**CprE 530**

Lecture 23

**Topics**

• FTP

• General Countermeasures

**FTP**

**Command Action**

**Authentication**

USER username Send the username to the server

PASS password Send the user password to the server

QUIT Finish session

**File Management**

CWD directory\_name Change directory on the server

CDUP Change to the parent directory on the server

DELE filename Delete the file from the server

LIST directory\_name List the files on the server

MKD directory\_name Make a new directory on the server

PWD Print the current directory on the server

RMD directory\_name Delete a directory from the server

RNFR old\_file\_name Name of file on the server to be renamed

RNTO new\_file\_name Name of file on the server to rename the file to

**Data Format**

TYPE (A, I) Set data transfer type, A=ASCII, I=Image

**Data port**

PORT 6 digit identifier Client sends the port number for the server to connect to for the data transfer

PASV Server send the port number for the client to connect to for the data transfer

**File Transfer**

RETR filename(s) Transfer the file(s) from the server to the client using the data connection

STOR filename(s) Transfer the file(s) from the client to the server using the data connection

**Miscellaneous**

HELP Server will return information

**Response codes**

**Code Response Status**

1XX Positive Preliminary Reply – Indicates the server

will respond with another response code

before the client can continue.

2XX Positive Completion Reply – Indicates the

command was successful and a new

**Code Response type**

X0X Syntax Error or unimplemented commands

X1X Information – reply to a request for information

command can be issued.

3XX Positive Intermediate Reply – Indicates the

command was successful, but the action is

held up pending receipt of another

command from the client.

4XX Transient Negative Completion Reply – Indicates

the command was not accepted, however

the error is temporary.

5XX Permanent Negative Completion Reply –

Indicates the command was not accepted.

X2X Connections – Reply to a request for connection

X3X Authentication – Reply to authentication

commands

X4X Unspecified

X5X File System – Reply to file system based requests

**Common Response Codes**

Code Responses

150 Data connection will open

200 Command acknowledgement

220 Service ready

225 Data connection open

226 Closing data connection

230 User logged in

331 User needs password

425 Cannot open data connection

500 Syntax error

530 User login failure

FTP

Client FTP

Server

Open TCP connection to server port 21

**ftp spock.dougj.net**

220 spock.dougj.net FTP server (Version 6.00LS) ready \r\n

Connected to spock.dougj.net

**FTP Protocol Exchange**

USER cpre530

net.

220 spock.dougj.net FTP server (Version 6.00LS) ready.

User (spock.dougj.net:(none)): **cpre530**

331 Password required for cpre530

331 Password required for cpre530.

Password: **password**

PASS password

230 User cpre530 logged in. \r\n

**FTP Protocol Exchange**

**FTP Protocol Exchange**

**Anonymous FTP**

• $ **ftp spock.dougj.net**

• Connected to spock.dougj.net.

• 220 spock.dougj.net FTP server ready.

• User (spock.dougj.net:(none)): **anonymous**

• 331 Guest login ok, type your name as

password.

• Password:

• 230 Guest login ok, access restrictions apply.

• ftp>

**Anonymous FTP Server**

**TFTP**

Name

(opcode)

Parameters Function

RRQ (1) Filename (var), 0x00

Mode (var), 0x00

Read request, mode is either netascii or octet

WRQ (2) Filename (var), 0x00

Mode (var), 0x00

Write request, mode is either netascii or octet

DATA (3) Block Number (2 bytes) Block number starts at 1, all blocks except

Data (0-512 bytes)

the last block must be 512 bytes long. A

block that is less than 512 bytes is used to

indicate last block and the file transfer is

done

ACK (4) Block Number (2 bytes) Used to acknowledge the data block

ERROR (5) Error number (2 bytes)

Error data (var), 0x00

Used to indicate an error, the error data is text

data.

**RCP**

• Based on rlogin

• If user is trusted copy will take place

• If user is not trusted copy will not take

place.

**Header & Protocol Based**

• FTP has problems with buffer overflows

• Not many protocol attacks

– One is an FTP redirect attack

– Done by telneting to an FTP server that

has exploit code.

– Use ftp to transfer the code to another

server

**Redirect**

• $ **telnet klingon.iseage.org 21**

• 220 klingon.iseage.org FTP server ready.

• **user anonymous**

• 331 Guest login ok, type your name as

password.

• **pass doug**

• 230 Guest login ok, access restrictions apply.

File m1:

HELO cia.gov

MAIL FROM: badperson@cia.gov

RCPT TO: user

DATA

(any mail message)

.

• **port 192,168,1,40,0,25**

• 200 PORT command successful.

• **retr m1**

• 150 Opening ASCII mode data connection for

'm1' (84 bytes).

• 226 Transfer complete.

• **Quit**

**Authentication-Based**

• FTP Prompts for username and password

• Anonymous FTP with writable directories

• User based FTP server

**Traffic-Based**

• Clear Text

• FTP can be flooded, massive uploads or

downloads

**General Countermeasures**

• Encrypted Channels

• Encrypted copy & FTP

**Encrypted**

**Channels**

Application

Encryption

Application

Encryption

TCP/IP TCP/IP

Key Exchange

Application

Protocol

Application(s)

Encryption

Application(s)

Encryption

TCP/IP TCP/IP

Key Exchange

Application

Protocol

**Encrypted**

**protocols**

Client Server

Open TCP Connection

Version negotiation

Capability negotiation

Key negotiation

Encrypted data exchange

**Peer-to-Peer Topics**

• We will look at examples of peer-to-peer

protocols

– Napster

– KaZaA

– Gnutella

• Anonymous services

– Routing

– Surfing

• Privacy on the Internet

• Proxy servers

**Peer to peer types**

• Decentralized Supernode

Supernode

Supernode

Supernode Supernode

**Peer to Peer types**

• Central Index Server

Central Index Server

File Lists & Queries

File Transfers

**Napster**

• Napster is a controversial application that

facilitates the sharing of music files

• User’s can search for songs and download

songs from another user’s harddrive

• All clients connect to a central server

server

client file transfer client

**Napster**

• Napster has a simple packet format:

• The length and type fields are each 2 bytes

• Types:

Length Type Data

2 Login 203 Get

3 Login Ack 204 Get Ack

100 Notify 218 Download

200 Search request 219 Download complete

201 Search reply 220 Upload

221 Upload complete

**Napster**

• Sequence:

– Log in to server

– Notify the server of files you are sharing

– Search for a file to download

– Download the file

• The above sequence is illustrated on

the next slide.

• For now, assume the user is not behind

a firewall

Client 1 Server

Login

Client 2

ACK

Notify

Search

Results

Get File

ACK

TCP Connect

ACK

File Name

File Size

Download Upload

File Transfer

TCP Close

Done Done

Length Type Data

**Napster**

• When client 1 is behind a firewall, the

download is slightly different

• Client 1 tells the server the port to use

• The server then tells client 2 which port

to use

• Client 2 sends the file to the specified

port

Client 1 Server

Login

Client 2

ACK

Notify

Search

Results

Get File

Port Number

TCP Connect

Filename & port number

ACK

Send File Name & Size

ACK

Download Upload

File Transfer

TCP Close

Done Done

**Napster Issues**

• As shown in the preceding illustrations, the

server is heavily involved in facilitating the

transfer of files

• The server also keeps track of what is being

transferred where

• This may have played a part in the case against

Napster

• However, how can you verify that the filename

accurately reflects the song transferred?

KaZaA

• Central Index server based (called

super nodes)

• Uses Fasttrack protocol between server

and client

– Proprietary protocol

• All files have hash values

• Protocol between clients is HTTP 1.1

Supernode

1

6

3

2

Supernode

Index

**User 2**

Supernode

Supernode

**User 1** Index

**KaZaA**

(1) Upload User's Index

(2) Search Supernode Index

(3) Search Results

(4) Request for File

(5) Send File

(6) File added to Supernode Index

(7) Queries

5

4

Shared Folder/

User Index

Shared Folder/

User Index

**User 1**

**User 4 User 3**

**Decentralized Peer-to-Peer**

• Limewire, Bearshare, Gnutella

• Peer-to-peer arrangement

• No central server

• Each client connects to 4 other clients,

called servents

• Other clients connect to you

• Allows you to share and download any

file type, not just music

**Gnutella Protocol**

• When you search for a file, you ask the

servents nearest you, who ask the servents

nearest them, and the search propagates in a

daisy chain effect

• Logging in to the gnutella network generates

a lot of traffic, as other people’s searches are

constantly propagating through you

• You can see what other people are searching

for through you

• Gnutella clients are available for every

platform. Some examples: BearShare,

LimeWire

**Gnutella Routing**

g (4)

Ping (4)

7 Pongs

ping Ping (4)

Ping (4)

Ping (4)

Ping (4)

Ping (4)

Ping (4)

Pong

Pong

Pong

Pong

Pong

10 Pongs

**Gnutella Ping and Pong**

• The data section of the “pong” packet

contains:

– Port number of responding machine

– IP address

– Number of files shared (4 bytes)

– Total kilobytes shared (4 bytes)

• “Ping” packets contain no data

• Each client periodically pings all

connections nearest them

**Gnutella Queries**

• The “query” packet contains:

– Minimum speed in kb/s (2 bytes)

– Search string (length varies)

• The “query-hit” packet contains:

– Number of hits (1 byte)

– Port (2 bytes)

– IP address (4 bytes)

– Speed (2 bytes)

– Result set (length varies)

• Index (4 bytes), Filesize (4 bytes), Name (length varies)

– Servent name, used for push (generally the IP

address)

**Gnutella Packet Format**

ID Payload TTL Hop Length Data

Payload

00 ping

01 Pong

80 Query

81 Query Hit

Gnutella Packet

Min Speed String

Hits Port IP Speed Results IP

Query packet

Query-Hit packet

Port IP Number of files shared Number of bytes shared

Pong Packet

**Gnutella Push**

• A “push” is used when the user is

behind a firewall

• The “push” packet contains:

– Servent ID

– File index

– IP address

– Port

**Header / Protocol Based**

• Applications and protocol could be

subject to these attacks.

**Authentication Based**

• Cannot trust source of files

• Anything can be shared

• Users that share can be traced

**Traffic Based**

• Can generate large amounts of traffic

• Super nodes can draw more traffic

• Sniffing is possible, but does not matter

**Countermeasures**

• Port Blocking

• Content Blocking

* **CprE 530**
* Lecture 24
* **Topics**
* Anonymous services
  + Routing
  + Surfing
* Privacy on the Internet
* Proxy servers
* General countermeasures
* **Email Tracking**
* www.readnotify.com
* Uses web bug tracking
* Keeps a log and emails you when the recipient opens the email.
* Looks like the email came from the sender, you send the email to:
  + [user@domain.readnotify.com](mailto:user@domain.readnotify.com)
* **Anonymous Email Services**
* Login to a web site and send email from the site.
* Gmail, etc.
* Special sites for anonymous email
  + www.anonymousspeech.com
* **Privacy surfing the Internet**
* Web servers can collect demographics about you
* [www.privacy.net](http://www.privacy.net/) will show you all the things a webserver knows about you
* Examples:
  + Your browser type and Operating System
  + CPU type
  + whether JavaScript is enabled
  + Date/Time on your computer
  + Your IP address
  + Which plugins you have installed
* **Privacy on the Internet**
* Once you login and give your email address, you are no longer anonymous
* Some web sites share your email address with other sites
* This can lead to you receiving spam from sites to which you’ve never disclosed your email
* Some sites store cookies on your harddrive. Amazon.com does this to recommend books based on your previous purchases.
* One way to surf privately: connect through a proxy
* **Proxy Servers**
* A proxy is basically someone who makes requests on your behalf
* They were originally designed to cache information to prevent redundancy
* Suppose you (M) want to view a web page from server W. Here’s how it would look without a proxy:

SIP = M S Port = ephemeral

DIP = W D Port = 80

URL=http://w.com/path

* **Proxy Servers**
* Here’s how it would look if you used a proxy server. Two different packets are needed: packet A is generated by yourself, and packet B is generated by the proxy server

Packet A: Packet B:

SIP=M SPort=? SIP=P SPort=?

DIP=P DPort= DIP=W Dport=80

URL=http://w.com/path URL=http://w.com/path

* **Proxy Servers**
* There are two reasons to be anonymous
  + Don’t want webservers to know who we are
  + Don’t want big brother (ie: your boss) to know what sites we are visiting
* A proxy can provide some amount of anonymity
* Examples of existing proxy servers used to provide anonymity:
  + anonymizer.com, safeweb.com, kaxy.com, the-cloak.com
* However, if your company does not wish you to be using these proxies, they can block access to them through their firewall.
* **Secure Proxy Server**
* **Proxy Servers**
* However, TOR has a fix that prevents a company from blocking access to their site.
* It involves a a system called onion routing
* See diagram next slide
* **TOR**
* **TOR**
* Starting host builds the connection one node at a time.
* The encryption keys are between each node and the starting point, so each node is unable to read the data
* Once the end node is reached the starting node has a key with each node.
* Destination host only sees the last node
* **Security Issues**
* Bypass company security policies
* Hard to stop
* **General Countermeasures**
* Encrypted remote access
  + Application-based
  + Tunnel-based
  + SSH
  + Remote desktop
  + Secure File transfer
* **Application-Based Encryption**
* **Tunnel-Based Encryption**
* **Encrypted Remote access protocols**
* **SSH**
* SSH
  + Secure shell
  + Designed to replace rlogin, rsh, rcp
  + Provides
    - Authentication at the machine level, doesn’t care about user authentication
    - Secure communication through encryption
  + **SSH Details**
* Strong Authentication
* Public domain software
* Some versions support compression of data
* Privacy
  + Key negotiation with symmetric key
  + Key exchange based on no trust of network
  + Multiple keys to deal with replay attacks
* Can provide secure X11 sessions
* Encrypt any traffic with SSH
* Same parameters as rlogin
* If other side doesn’t support SSH drops to rlogin

* **Details cont…**
* Need server and client software
* Sshd server demon software
* Ssh is the client software
* Ssh keygen
  + Generates host key
* Ssh agent
  + Uses public and private key technique to get process started
* **SSH Protocol**
* Client sends query
* Server sends two public keys which is a 1024 bit client key and a server key which is a 768 bit key
* Server key recomputed every hour
* Client generates 256 bit random number which is the symmetric key, which is encrypted using the server and the host keys
* Server responds with ok which is encrypted with session key
* All traffic is now encrypted with session key
* Problems
  + Man in the middle attack
  + Putty is a man in the middle attack program
* **SSH**
* **SSH Man in the Middle Attack**
* **Remote Desktop**
* Uses tunnel-based encryption
  + Via RDP or TLS (newer versions)
* Key exchange is similar to SSH
* Three levels
  + High (128 bit)
  + Medium (56 or 40 bit)
  + Low (56 or 40) only client to server data
* Subject to password guessing and man in the middle attacks
* **Secure File Transfer**
* SFTP – uses SSH
* FTPS – uses SSL/TLS
* HTTPS – uses SSL/TLS
* **CprE 530**
* Lecture 25
* **Topics**
* • Network-Based Mitigation
* – Network Firewalls
* – Intrusion Detection and Prevention
* – Data Loss Prevention
* **Network Firewalls**
* • Designed to “look” at each packet on
* the network and decide if the packet
* should be allowed to pass through the
* firewall or not.
* • Uses set of rules to decide if the packet
* should be blocked
* • Rules are typically based on the packet
* headers (IP & TCP)
* • Public domain versions are available
* **Firewall**
* **Firewall Rules**
* • Stateless
* – Each packet is independent
* – Very fast and simple to implement
* – Only simple rules
* – Example: block all UDP but port 53
* • Stateful
* – Deals with packet streams
* – Slower and requires more resources
* – Can implement complex rules
* – Example: Block all port 53 unless there is a
* pending request.
* • GUI on the device
* • Network based
* – Typically password protected
* – Only allows access to control interface
* **User control**
* from inside network
* – Can use a separate control network
* **Firewall types**
* • Transparent
* • Router-based
* • NAT-based
* • Application
* **Transparent Firewall**
* **Transparent Firewall**
* • Two network interfaces
* • “sniff” traffic
* • Does not have an IP layer for the packet flow
* • No need to change network configuration
* • Can be implemented as a single port firewall
* • Typically simple rule set (mostly stateless)
* User
* Interface
* Logs Rules
* Control
* Network
* or Console
* Interface
* Auth
* **Filtering Router**
* IP
* NET
* IP
* NET
* Outside
* Network
* Inside
* Network
* Rule Engine and Router
* **Filtering Router**
* • Acts like a normal router
* • Both stateless and stateful (with simple
* rules)
* • Often some stateless firewall
* functionality included in most routers
* • Implemented as part of a NAT
* • Firewall rules can restrict traffic even
* more than a normal NAT
* **NAT-Based**
* **Application Firewall**
* **Application Firewall**
* • Uses application gateways to allow a
* user to gain access through the network.
* • Application gateways look like an
* application and typically require userbased
* authentication to gain access
* through the firewall.
* • Also typically supports NAT functionality
* for applications without a gateway
* **Firewall Deployment**
* **Firewall Deployment**
* • DMZ
* – Used to support public servers
* **Intrusion Detection/**
* **Prevention**
* • IDS
* –Watches the network traffic looking for
* traffic patterns that could be an attack
* • IPS
* – Same as an IDS, but will also block traffic
* based on rules
* User
* Interface
* Logs Rules
* Control
* Network
* or Console
* Interface
* Auth
* **Intrusion Detection**
* NET
* Rule Engine
* Stream Assembly
* TCP
* IP
* Intrusion Detection
* **Intrusion Prevention**
* **IDS/IPS rules**
* • Correctly identify the attack
* • False Positives
* – Identifying an attack that is not there
* • False Negative
* – Missing an attack
* • Balance between the false positive and
* false negative rate is difficult
* • Large log files are hard to deal with
* **IDS/IPS Deployment**
* **Data Loss Prevention**
* • Stop data from leaving an organization
* • Like an IDS/IPS except it looks at the
* payload
* • Two data types
* – Structured: data that can be matched to a
* list like credit card numbers
* – Unstructured: data like letters or memos
* **DLP**
* • Structured data
* – Pattern matching
* • Unstructured
* – Fingerprinting
* – Lexical analysis
* **DLP**
* • What to do when you find data leaving
* – Block
* – Log
* – Redirect or quarantine
* **Single Port DLP**
* **Dual port DLP**