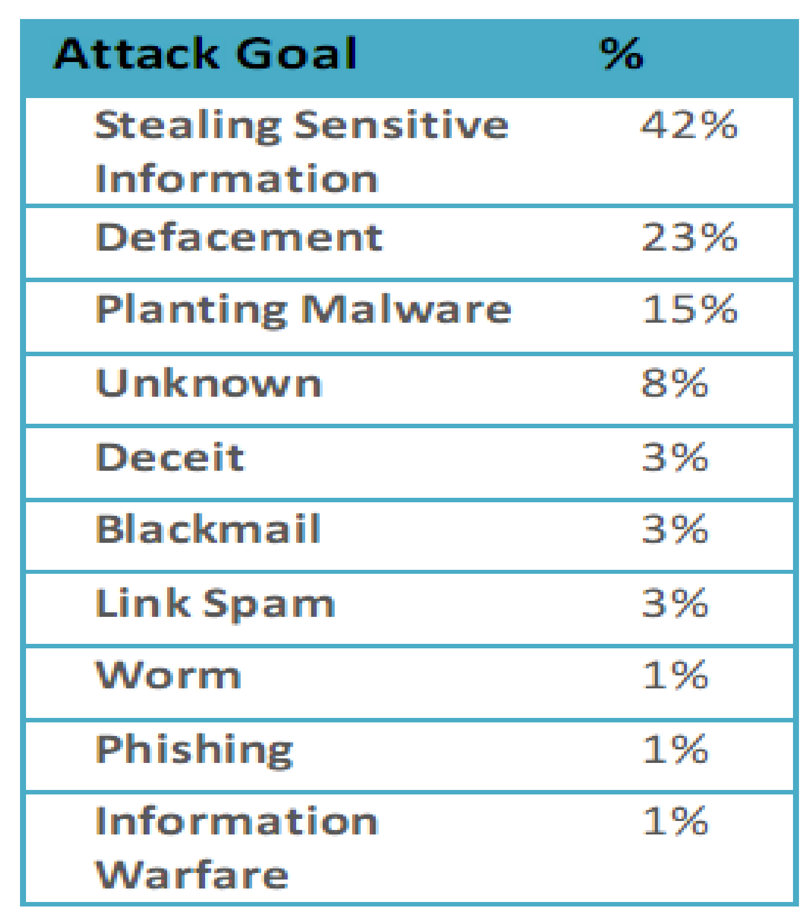
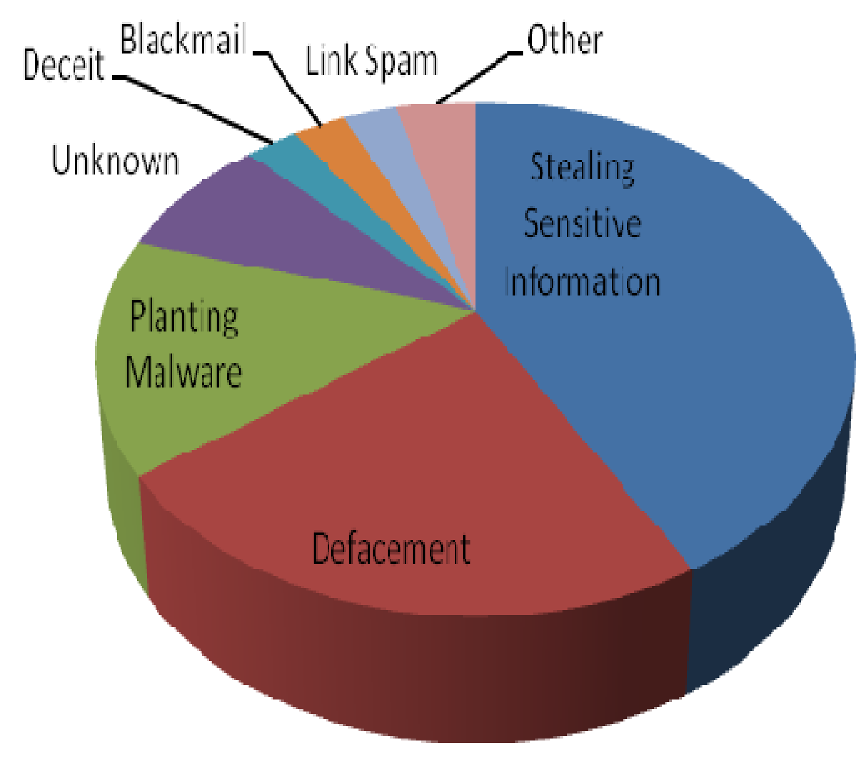
Application Security through a Hacker's Eyes

James Walden
Northern Kentucky University
waldenj@nku.edu

Why Do Hackers Target Web Apps?



Attack Goal	%
Stealing Sensitive Information	42%
Defacement	23%
Planting Malware	15%
Unknown	8%
Deceit	3%
Blackmail	3%
Link Spam	3%
Worm	1%
Phishing	1%
Information Warfare	1%

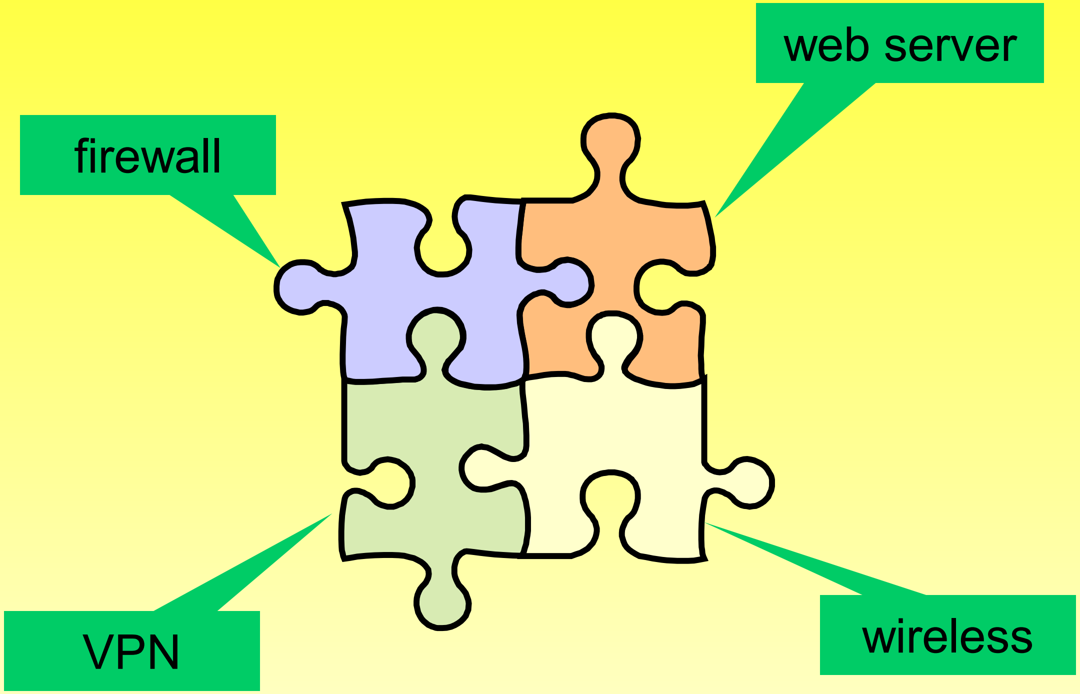


Attack Surface

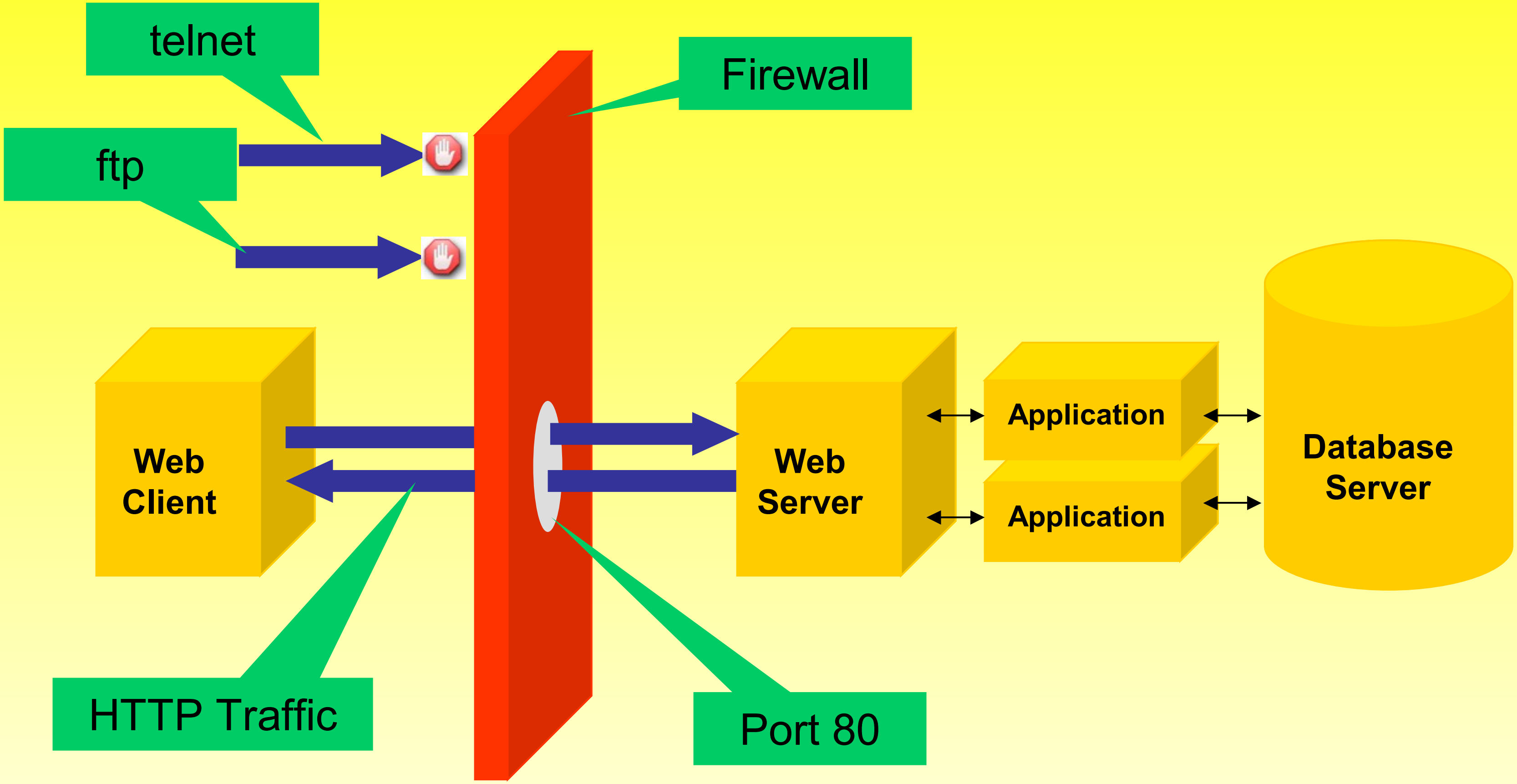
A system's ***attack surface*** consists of all of the ways an adversary can enter the system.

Merchant's Bank Building

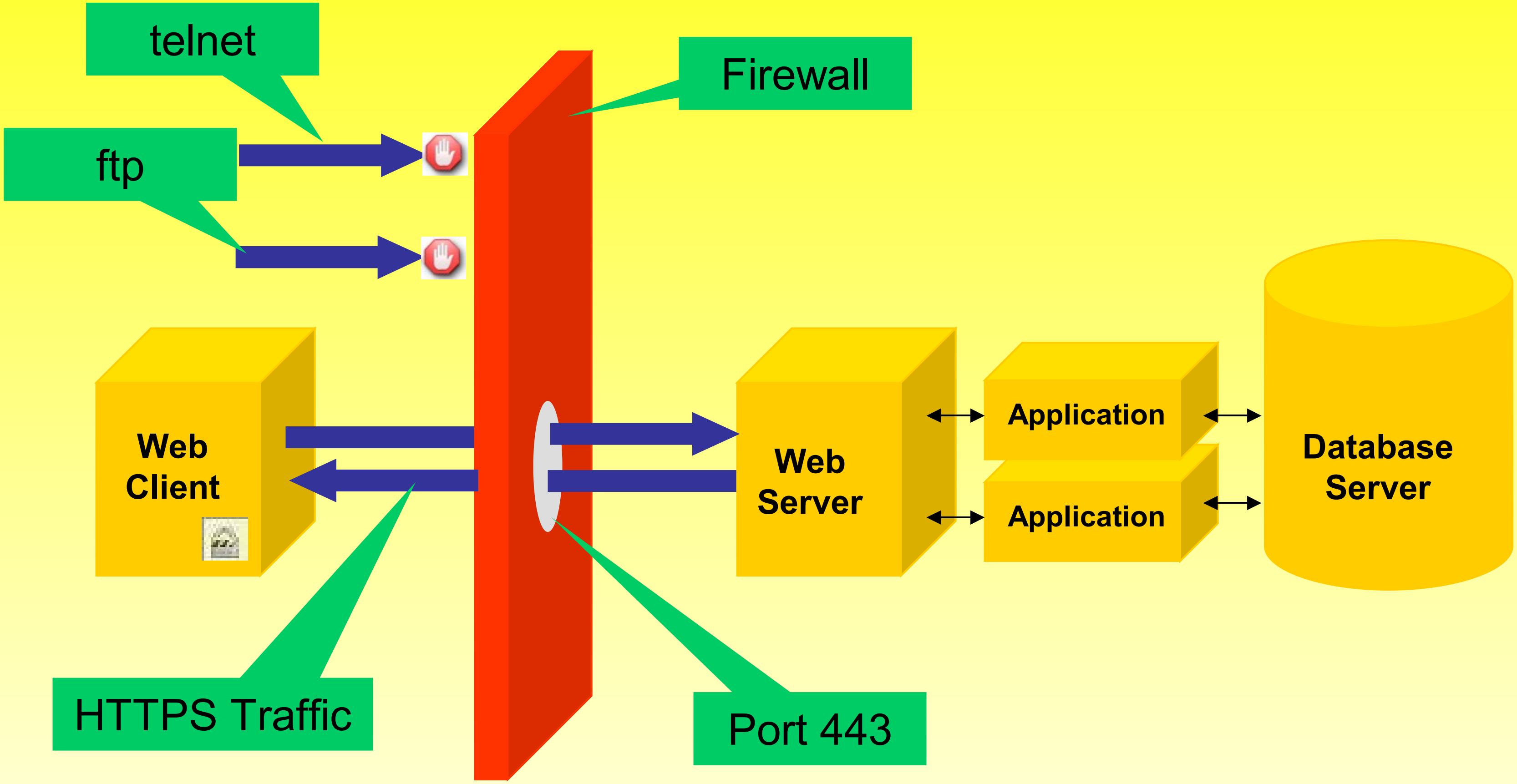
Defender's View of Attack Surface



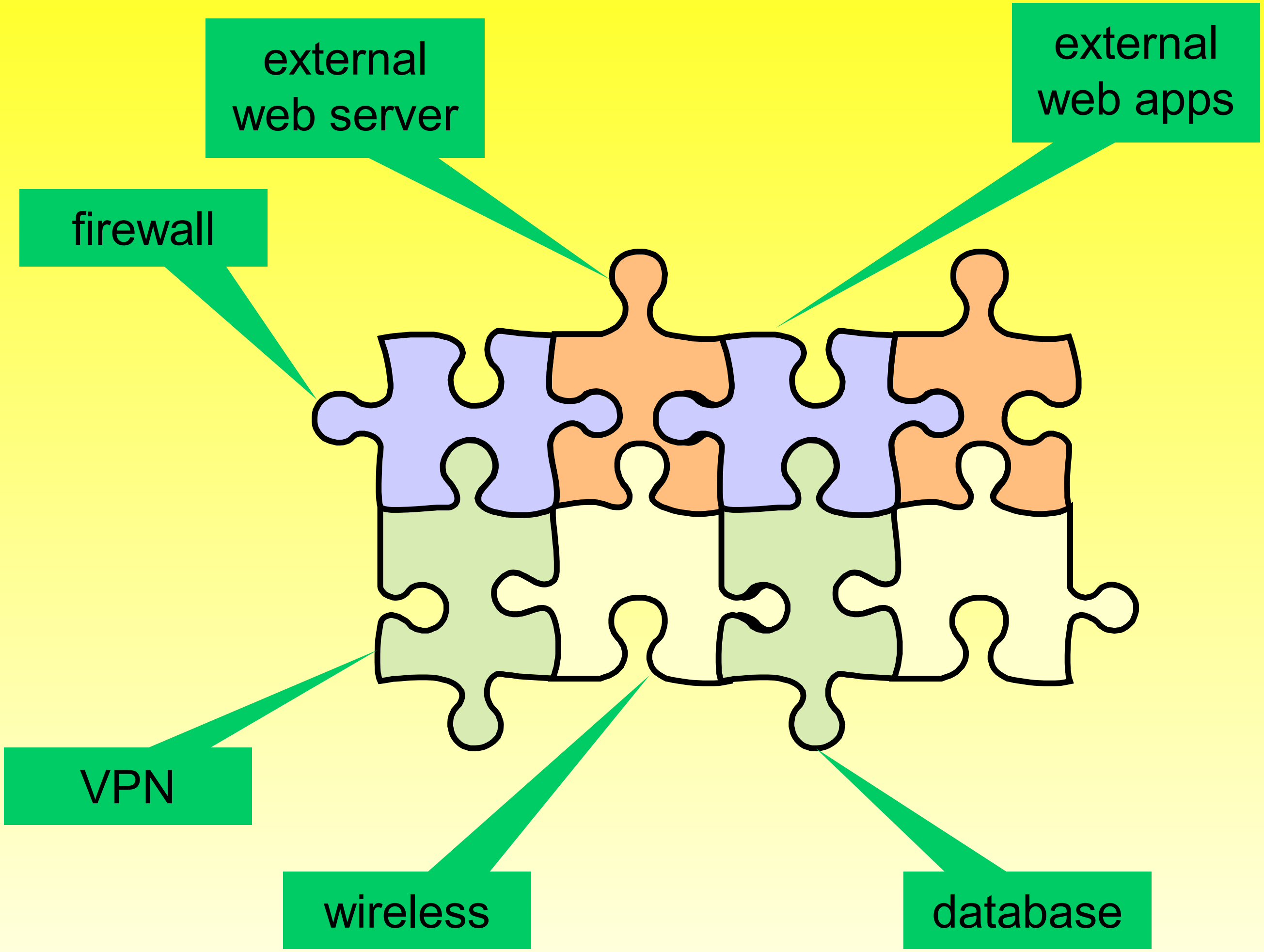
Firewalls don’t protect Web Apps



SSL won’t stop injection attacks, XSS



Revised View of Attack Surface



external web server

external web apps

firewall

VPN

wireless

database

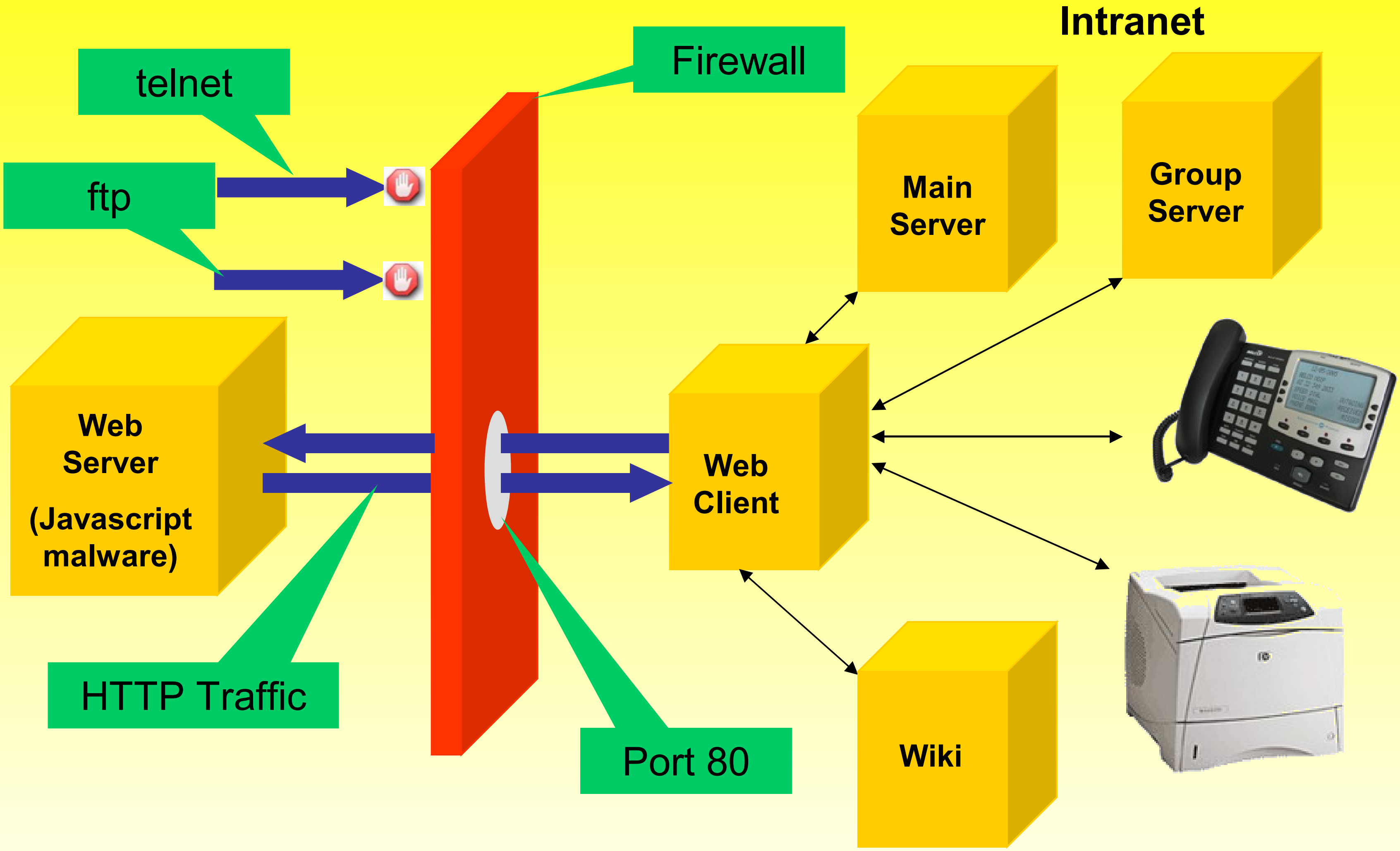
Intranet Security Assumptions

Since the firewall protects you

- Patches don’t have to be up to date.
- Passwords don’t have to be strong.
- There’s no need to be careful when you code.
- There’s no need to audit your source code.
- There’s no need to run penetration tests.

But do your users have web browsers?

Javascript Malware controls Clients



Intranet

Firewall

telnet

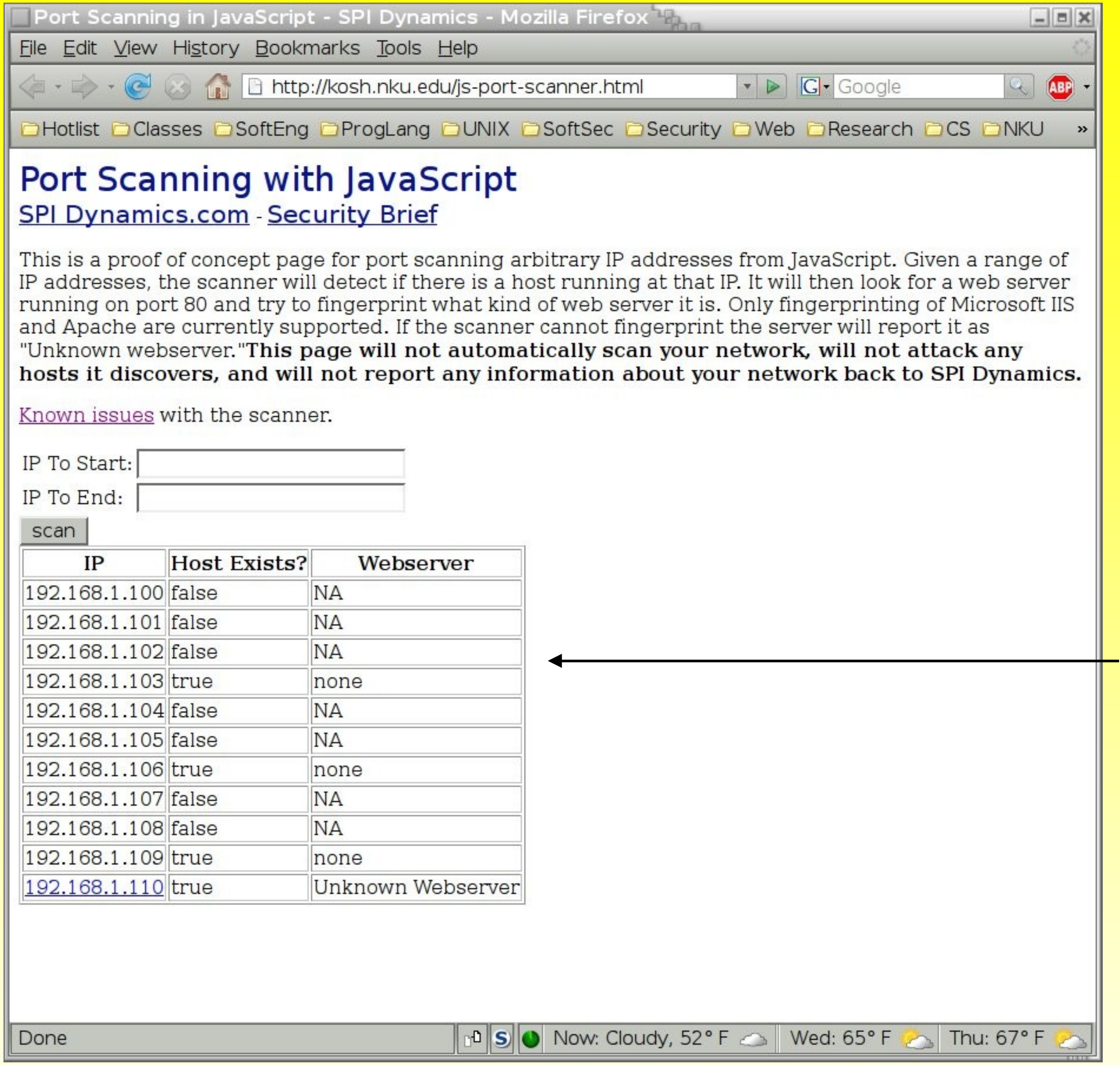
ftp

Web Server (Javascript malware)

HTTP Traffic

Port 80

Web Client


Port Scanning in JavaScript - SPI Dynamics - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://kosh.nku.edu/js-port-scanner.html
Google
Hotlist Classes SoftEng ProgLang UNIX SoftSec Security Web Research CS NKU
Done
Now: Cloudy, 52° F
Wed: 65° F
Thu: 67° F

Port Scanning with JavaScript

SPI Dynamics.com - Security Brief

This is a proof of concept page for port scanning arbitrary IP addresses from JavaScript. Given a range of IP addresses, the scanner will detect if there is a host running at that IP. It will then look for a web server running on port 80 and try to fingerprint what kind of web server it is. Only fingerprinting of Microsoft IIS and Apache are currently supported. If the scanner cannot fingerprint the server will report it as "Unknown webserver."**This page will not automatically scan your network, will not attack any hosts it discovers, and will not report any information about your network back to SPI Dynamics.**

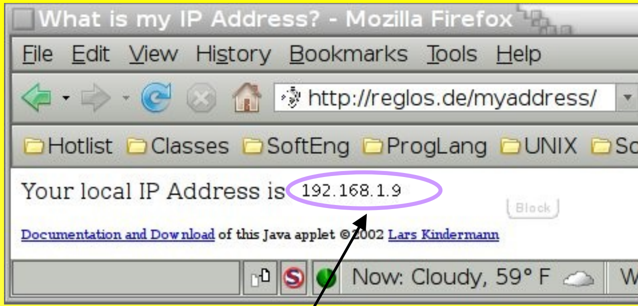
Known issues with the scanner.

IP To Start:

IP To End:

scan

IP	Host Exists?	Webserver
192.168.1.100	false	NA
192.168.1.101	false	NA
192.168.1.102	false	NA
192.168.1.103	true	none
192.168.1.104	false	NA
192.168.1.105	false	NA
192.168.1.106	true	none
192.168.1.107	false	NA
192.168.1.108	false	NA
192.168.1.109	true	none
192.168.1.110	true	Unknown Webserver


What is my IP Address? - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://reglos.de/myaddress/
Hotlist Classes SoftEng ProgLang UNIX So
Block
Now: Cloudy, 59° F

Your local IP Address is 192.168.1.9

Documentation and Download of this Java applet ©2002 Lars Kindermann

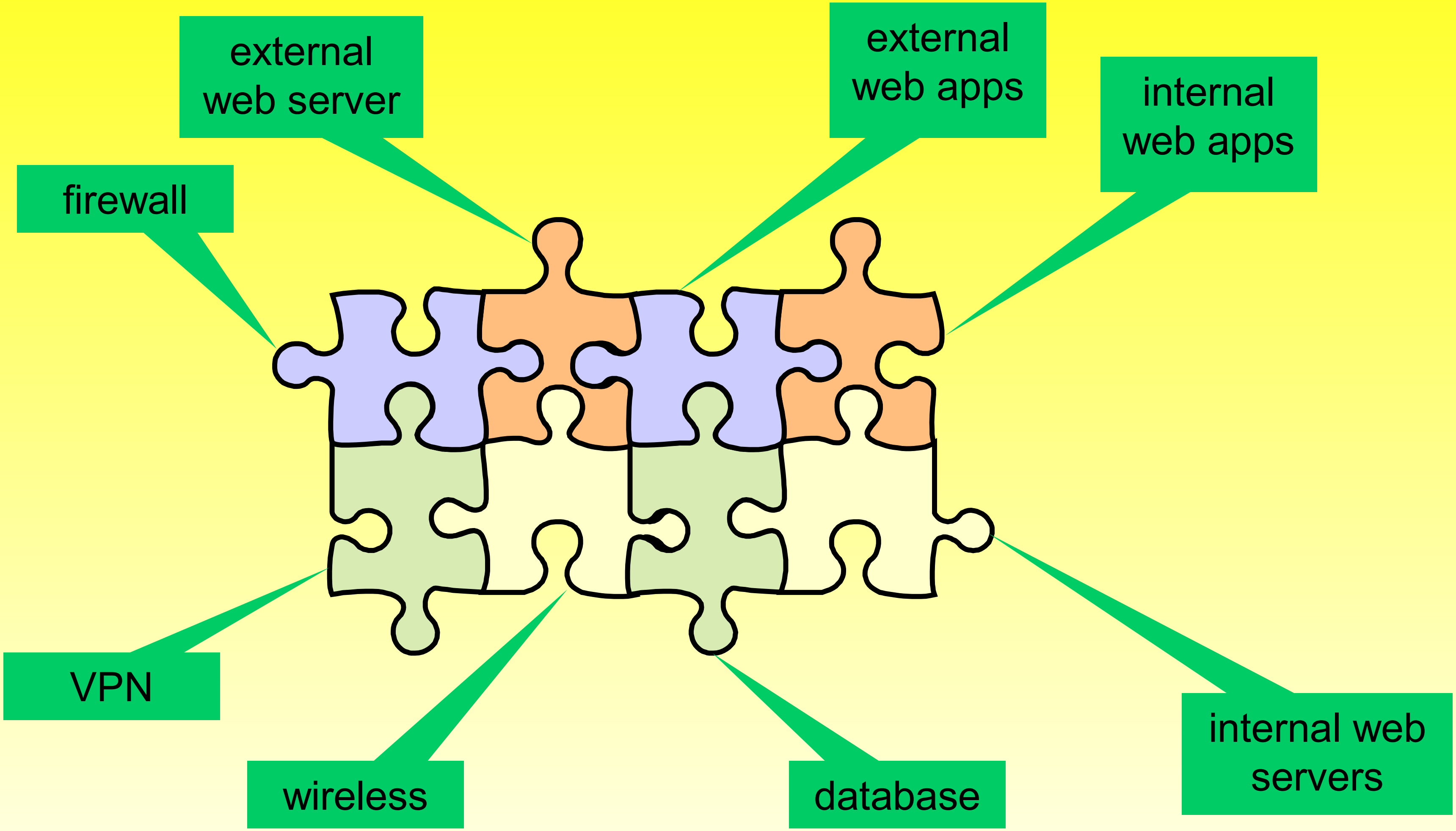
Java can see your real IP address behind NAT router.

Javascript can scan your intranet behind NAT router.

Sources of Javascript Malware

1. Evil web site owner inserts in page.
2. Attacker inserts malware into defaced page.
3. Attacker inserts malware into a public comment or forum post (stored XSS.)
4. Attacker creates link that causes web site to echo malware to user (reflected XSS.)

Re-revised View of Attack Surface



external web server

external web apps

internal web apps

firewall

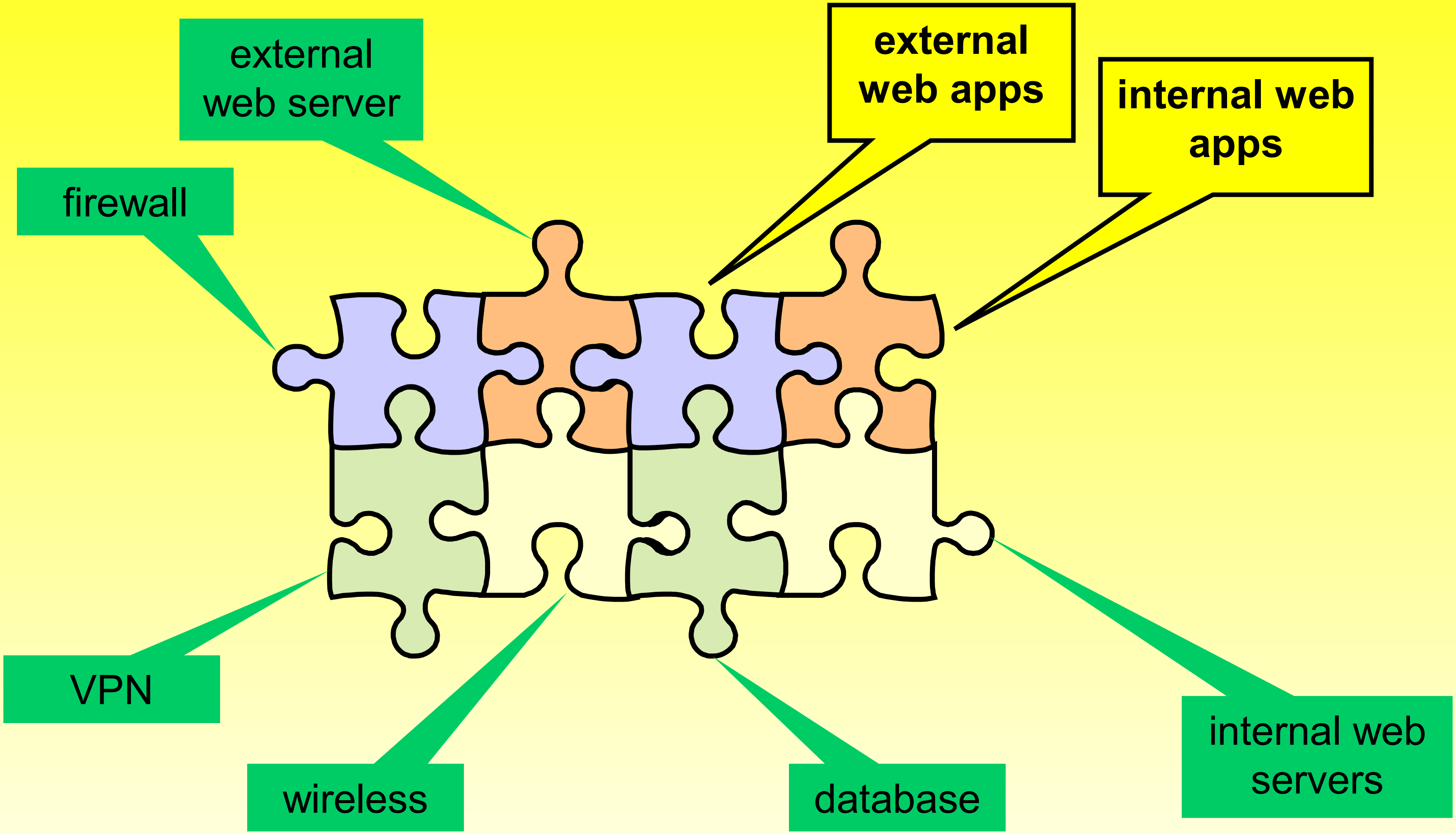
VPN

wireless

database

internal web servers

Web Applications



Web Application Vulnerabilities

Input-based Security Problems

- Injection Flaws
- Insecure Remote File Inclusion
- Unvalidated Input

Authentication and Authorization

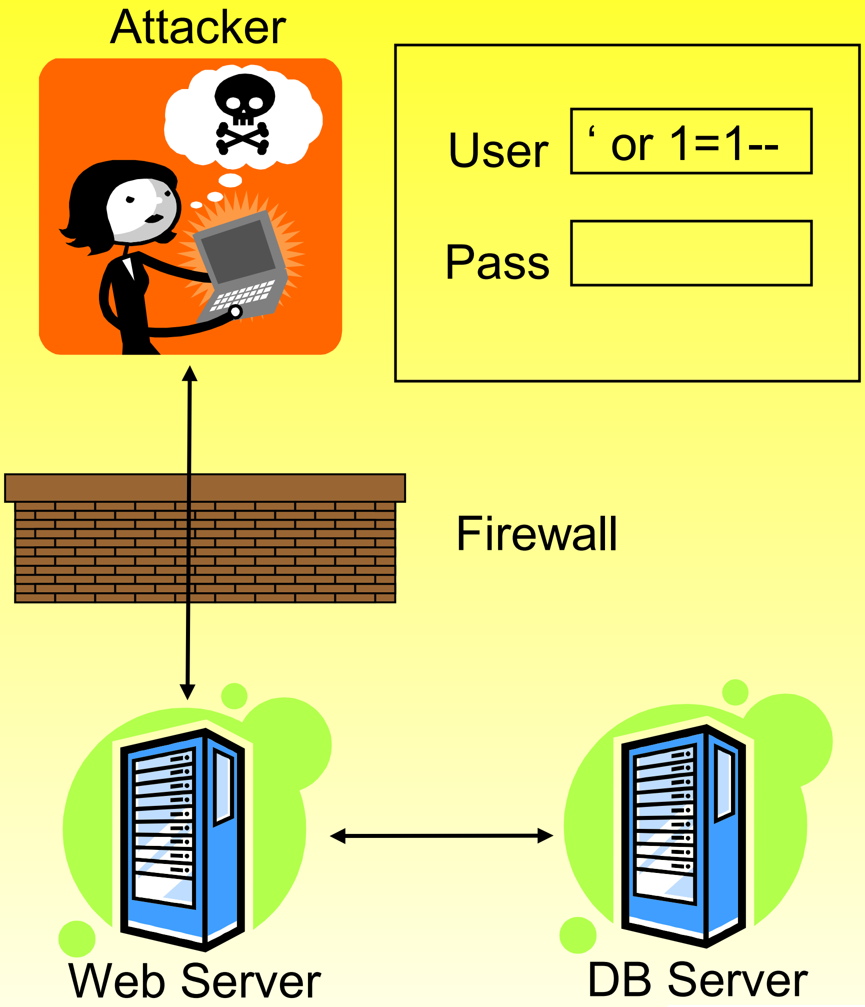
- Authentication
- Access Control
- Cross-Site Attacks

Other Bugs

- Error Handling and Information Leakage
- Insecure Storage
- Insecure Communications

SQL Injection

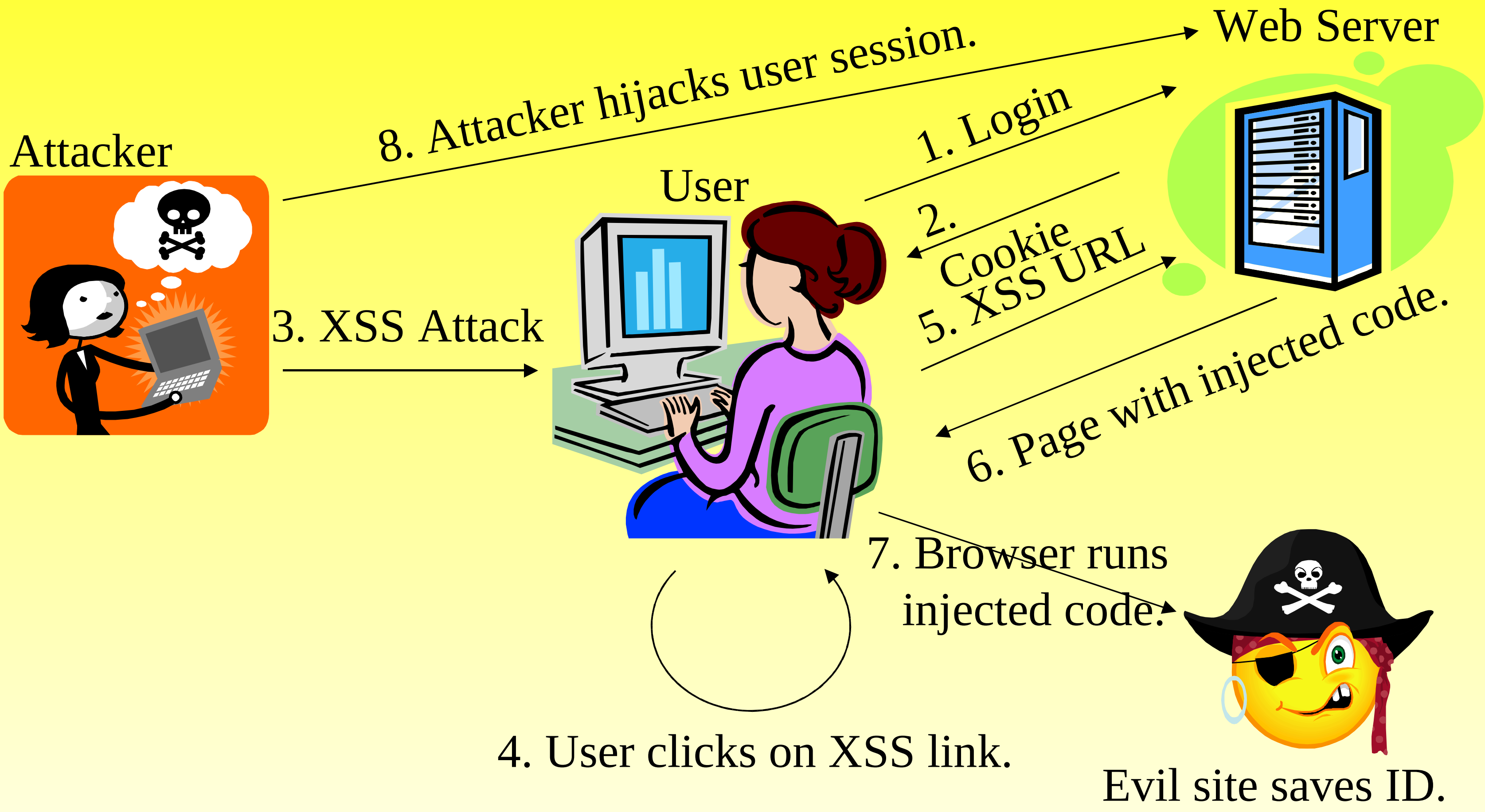
1. App sends form to user.
2. Attacker submits form with SQL exploit data.
3. Application builds string with exploit data.
4. Application sends SQL query to DB.
5. DB executes query, including exploit, sends data back to application.
6. Application returns data to user.



User ‘ or 1=1--

Pass

Cross-Site Scripting



Attacker

User

Web Server

1. Login

2. Cookie

3. XSS Attack

4. User clicks on XSS link.

5. XSS URL

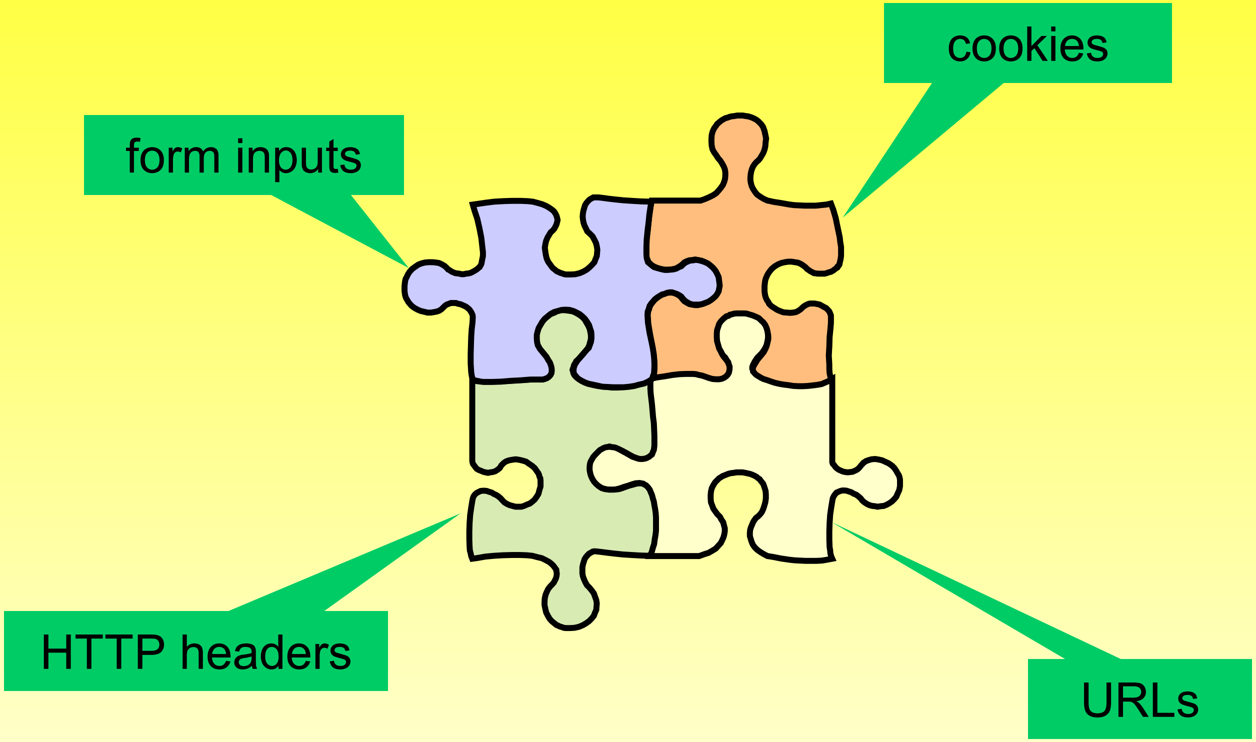
6. Page with injected code.

7. Browser runs injected code.

Evil site saves ID.

8. Attacker hijacks user session.

Web Application Attack Surface



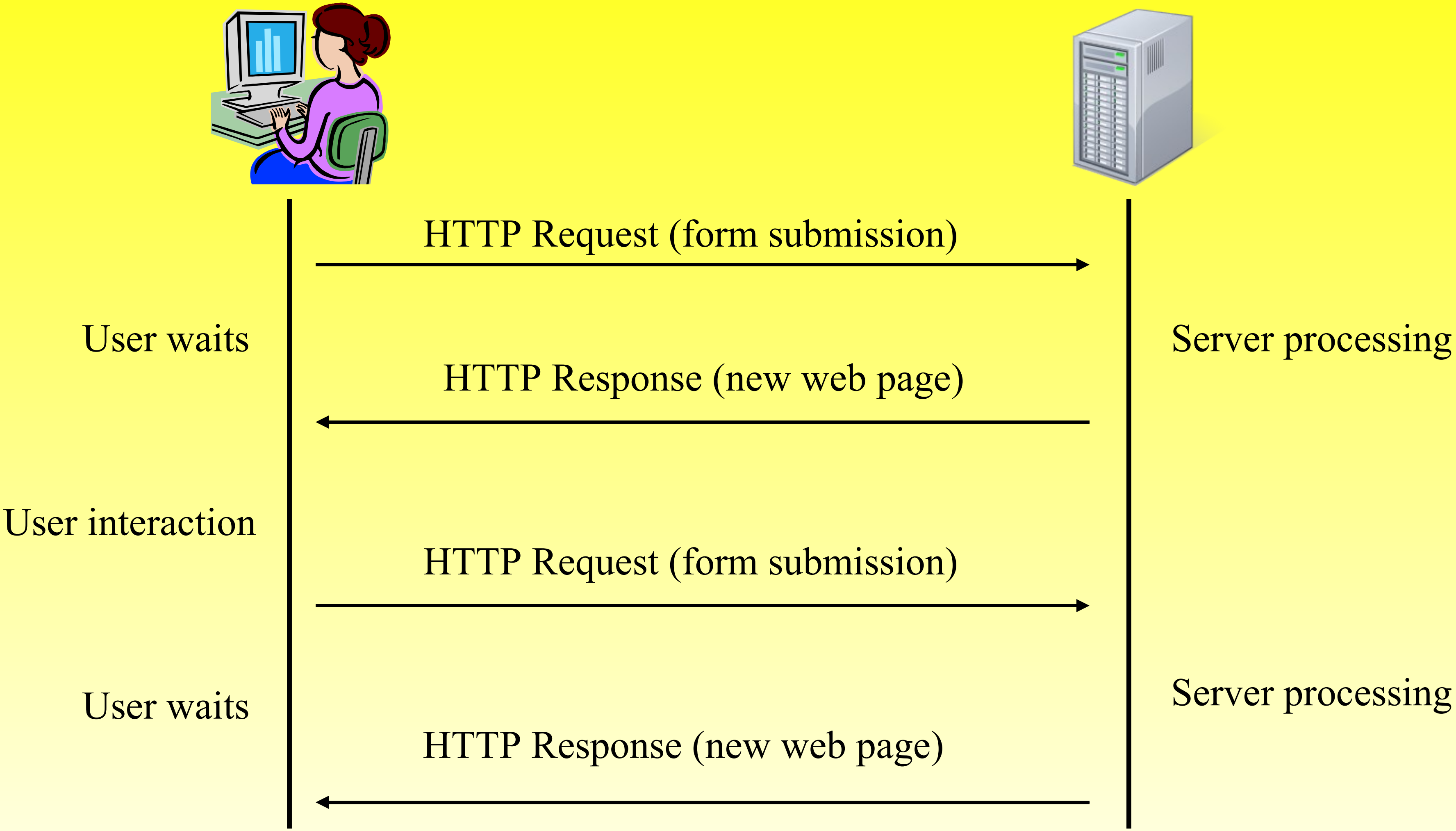
cookies

form inputs

HTTP headers

URLs

Traditional Web Applications



HTTP Request (form submission)

User waits

Server processing

HTTP Response (new web page)

User interaction

HTTP Request (form submission)

User waits

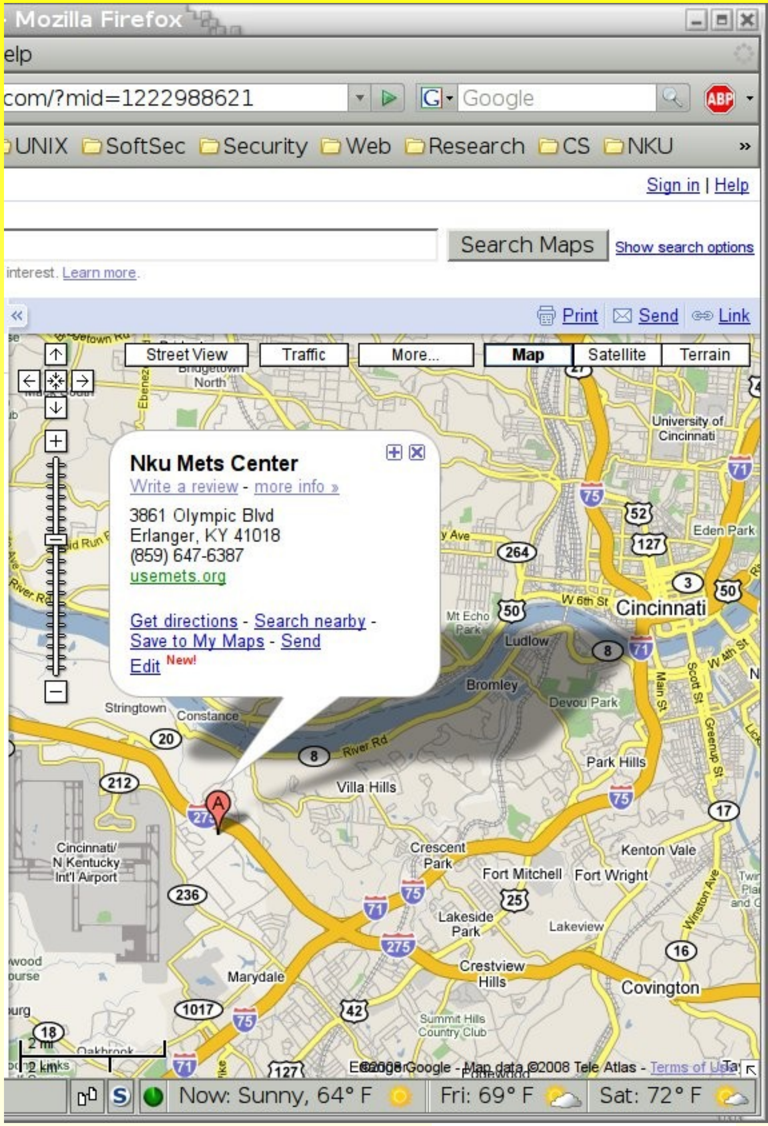
Server processing

HTTP Response (new web page)

AJAX

Asynchronous Javascript and XML

- User interacts with client-side Javascript.
- Javascript makes asynchronous requests to server for data.
- Continues to allow user to interact with application.
- Updates when receives encoded data from server.



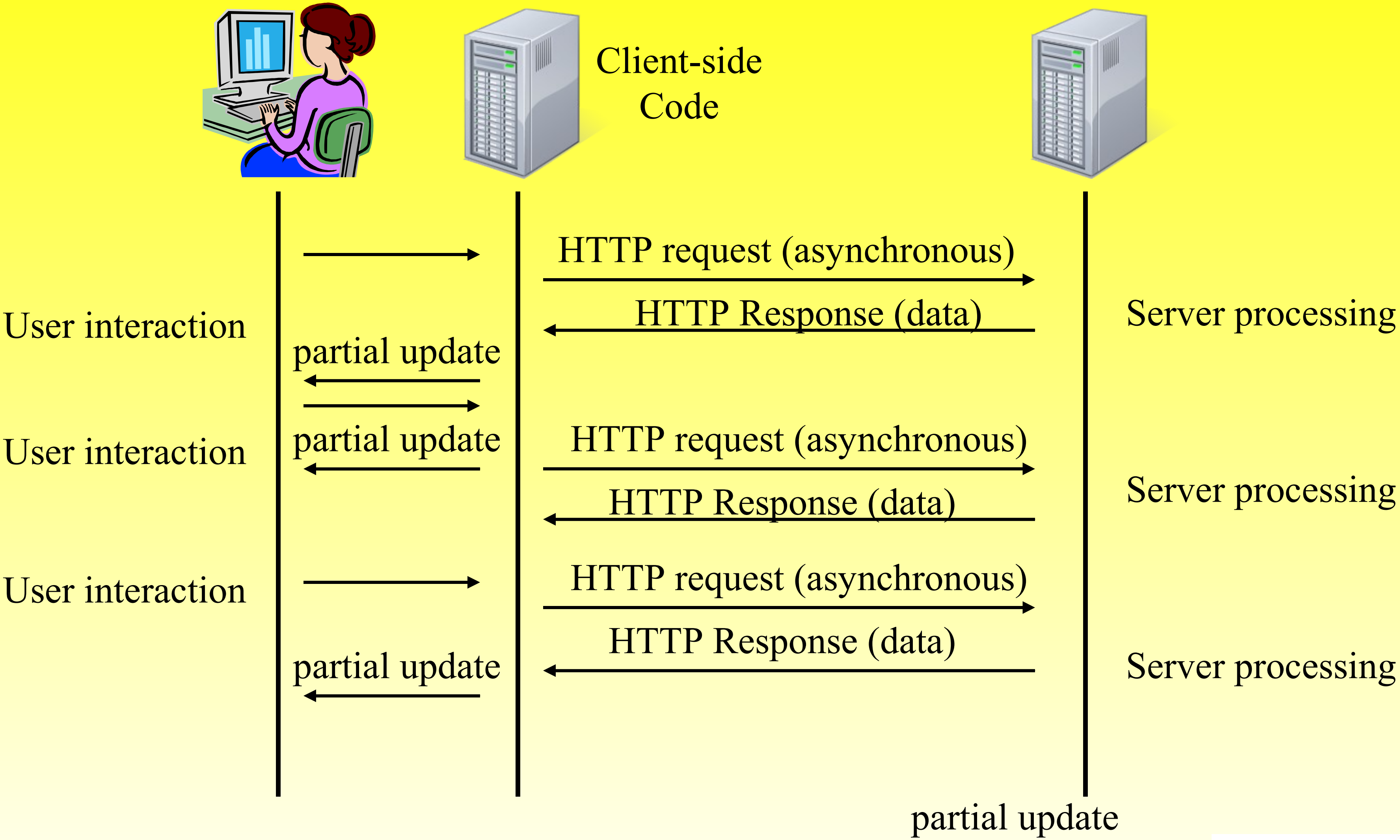
Nku Mets Center

Write a review - more info »

3861 Olympic Blvd
Erlanger, KY 41018
(859) 647-6387
usemets.org

Get directions - Search nearby -
Save to My Maps - Send
Edit New!

AJAX Applications



Client-side Code

User interaction

partial update

User interaction

partial update

User interaction

partial update

HTTP request (asynchronous)

HTTP Response (data)

HTTP request (asynchronous)

HTTP Response (data)

HTTP request (asynchronous)

HTTP Response (data)

Server processing

Server processing

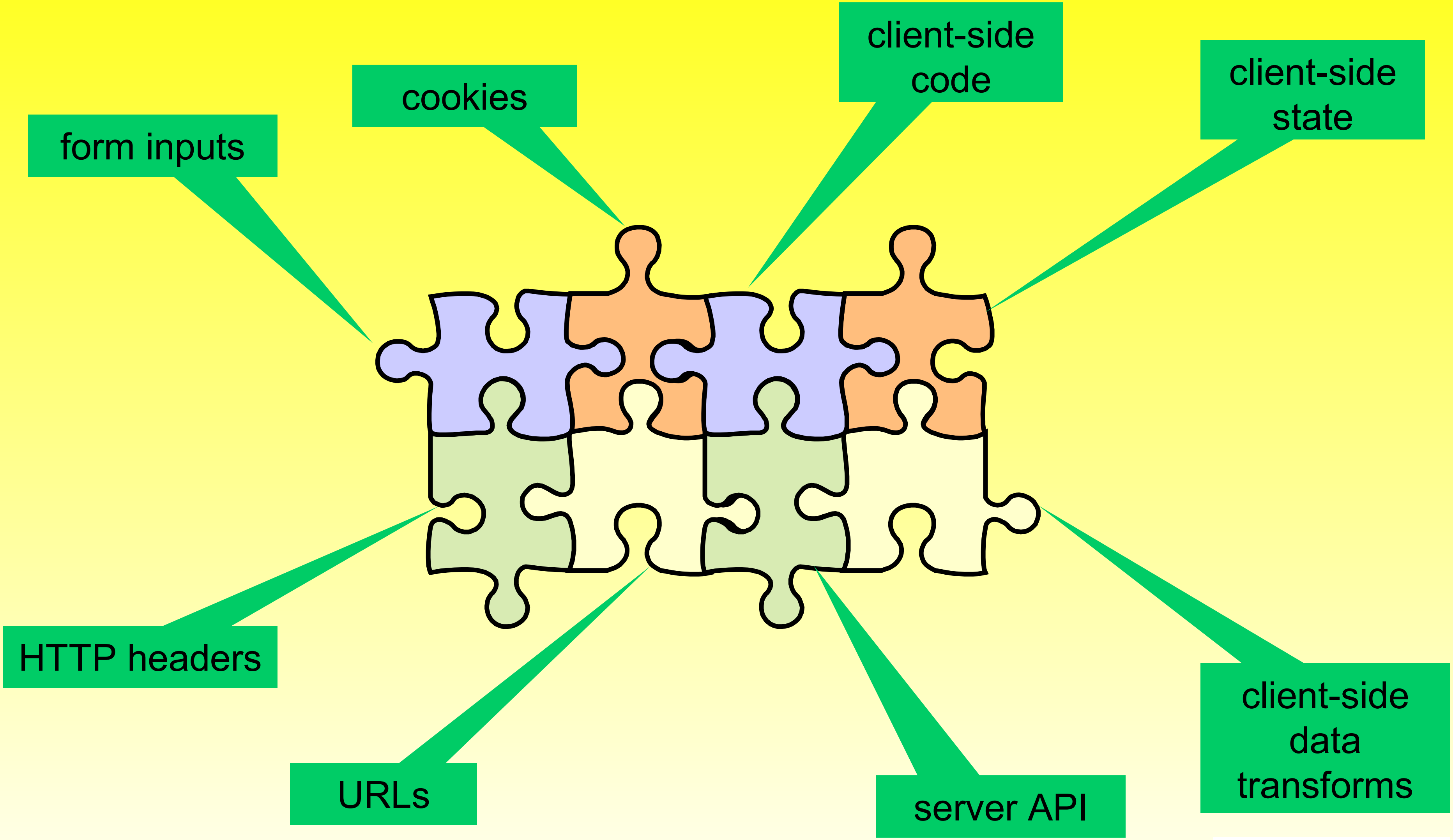
Server processing

partial update

Example Client-side Code

```
var auth = checkPassword(user, pass);
if (auth == false) {
   alert(‘Authentication failed.’);
   return;
}
var itemPrice = getPrice(itemID);
debitAccount(user, itemPrice);
downloadItem(itemID);
```

AJAX Application Attack Surface



form inputs

cookies

client-side code

client-side state

HTTP headers

URLs

server API

client-side data transforms

Drilling Down: Mapping the Application

1. Visible Content
    - Spider the site.
    - Browse site while using intercepting proxy.
2. Hidden Content
    1. Unlinked sections of site.
    2. Backup copies of live files.
    3. Configuration and include files.
    4. Source code.
    5. Log files.

Entry Points

For each resource found, identify inputs:

- Additional path parameters
- Query string
- POST parameters
- Cookies
- HTTP headers

Application Feature Vulnerability Map

Database interaction → SQL injection.

Displays user-supplied data → Cross-site scripting.

Error messages → Information leakage.

File upload/download → Path traversal.

Login → Authentication, session management, access control flaws.

Code Auditing

Why?

- Find vulnerabilities faster than testing.
- Find different vulnerabilities than testing.

What?

- Identify modules of high business risk.
- Use static analysis to find common vulnerabilities.
- Manually review code + static analysis results.

Who?

- Developers, security team, outside auditors.

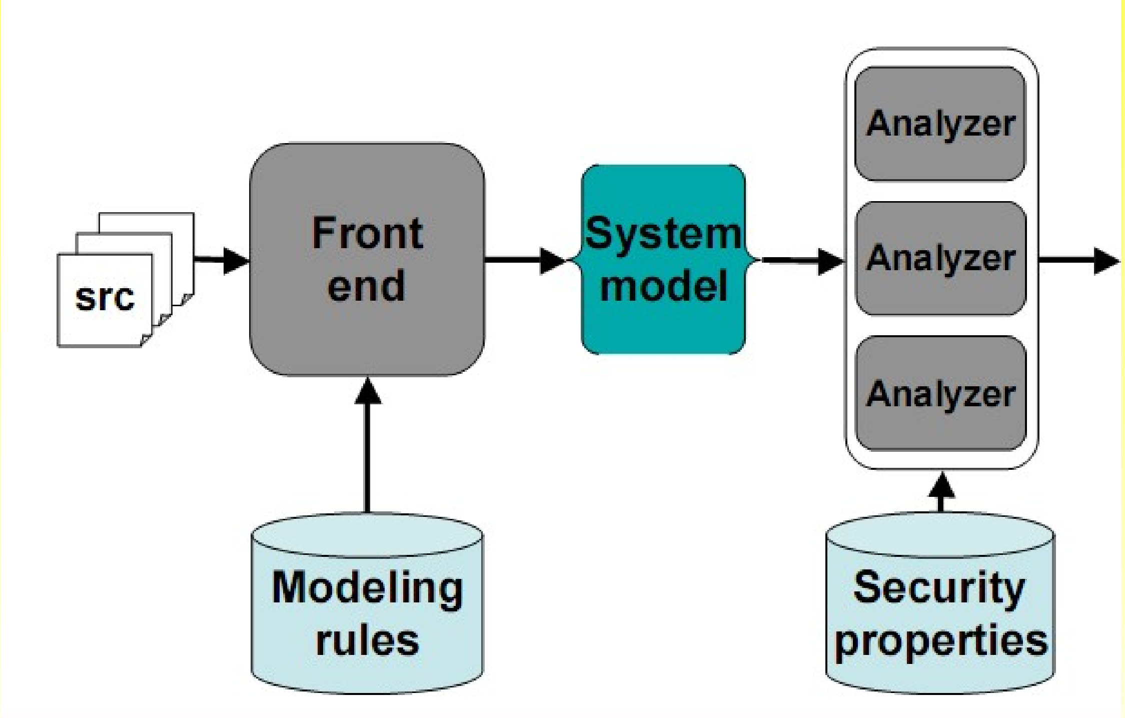
When?

- On a regular basis, at least once before release.

Static Analysis

Automated assistance for code auditing

- Speed: review code faster than humans can
- Accuracy: hundreds of secure coding rules



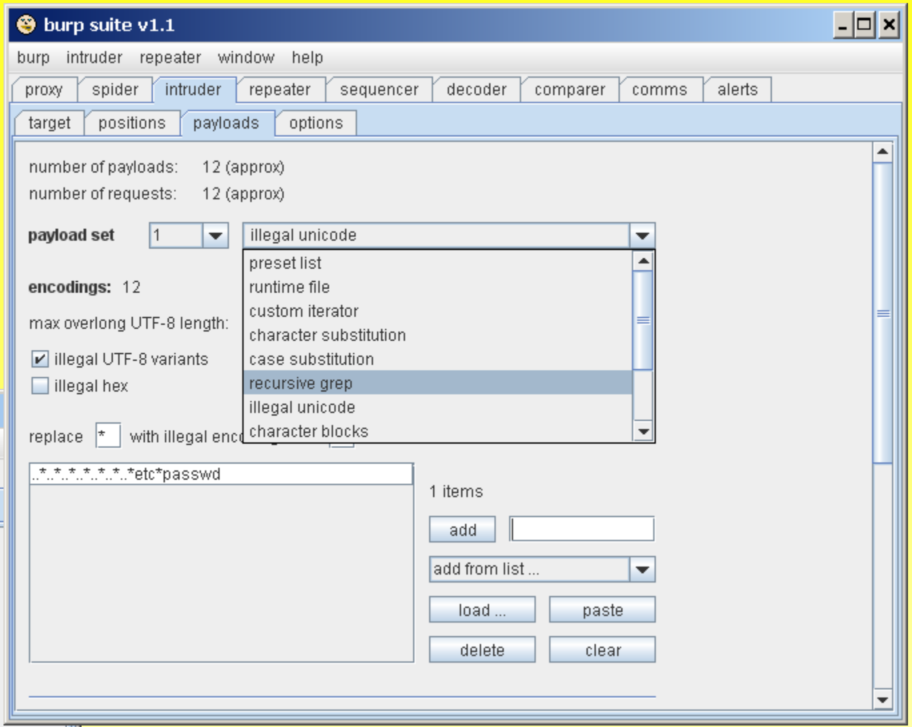
Tools

- Coverity
- FindBugs
- Fortify
- Klocwork
- Ounce Labs

Fuzz Testing

Fuzz testing consists of

- Sending unexpected input.
- Monitoring for exceptions.



number of payloads: 12 (approx)
number of requests: 12 (approx)

payload set 1 illegal unicode

encodings: 12

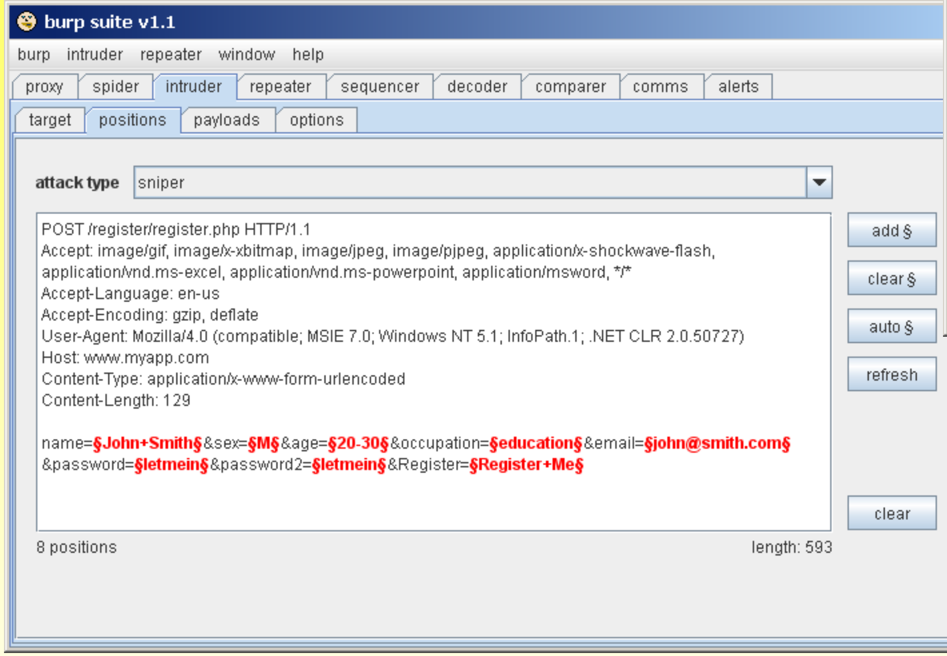
max overlong UTF-8 length:

☑ illegal UTF-8 variants
☐ illegal hex

replace * with illegal enc

```
..*..*..*..*..*..*..*etc*passwd
```

1 items


burp suite v1.1
burp intruder repeater window help
proxy spider intruder repeater sequencer decoder comparer comms alerts
target positions payloads options
add §
clear §
auto §
refresh
clear

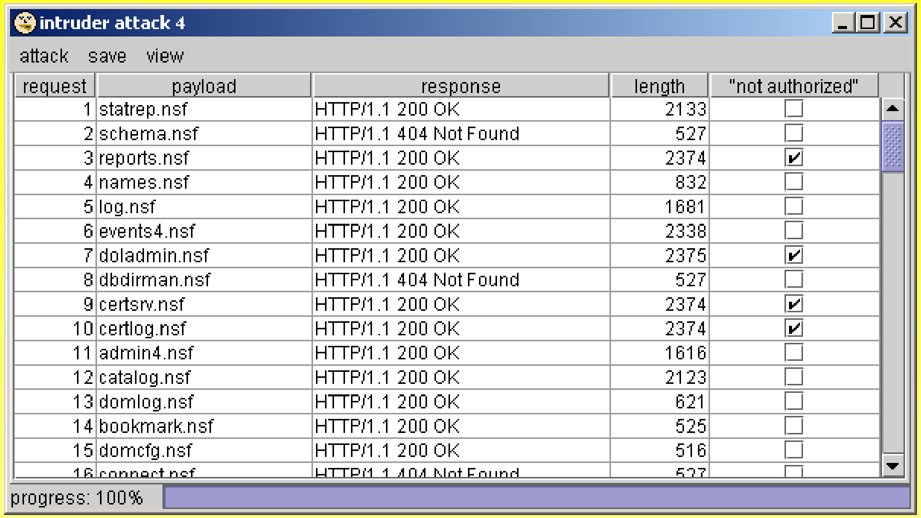
attack type sniper

```
POST /register/register.php HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash,
application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; InfoPath.1; .NET CLR 2.0.50727)
Host: www.myapp.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 129

name=§John+Smith§&sex=§M§&age=§20-30§&occupation=§education§&email=§john@smith.com§
&password=§letmein§&password2=§letmein§&Register=§Register+Me§
```

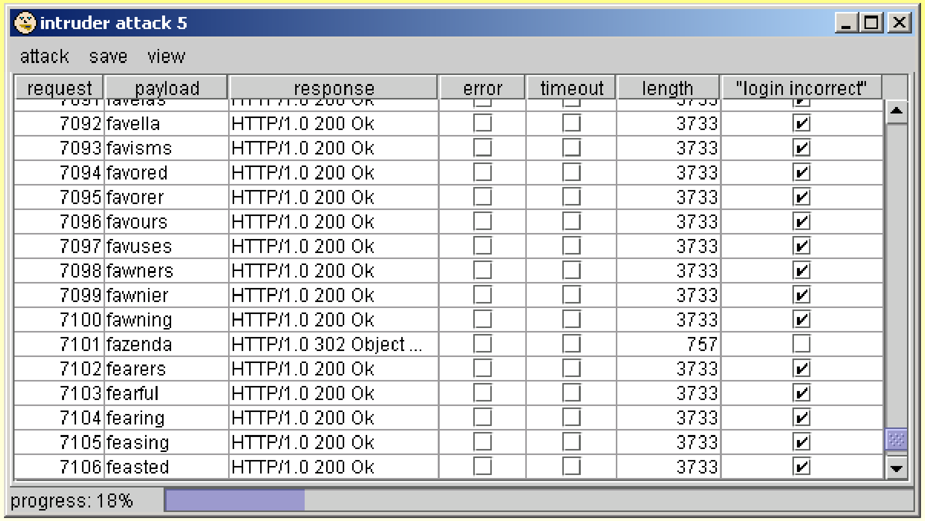
8 positions length: 593

Monitoring for Exceptions



request	payload	response	length	"not authorized"
1	statrep.nsf	HTTP/1.1 200 OK	2133	□
2	schema.nsf	HTTP/1.1 404 Not Found	527	□
3	reports.nsf	HTTP/1.1 200 OK	2374	☑
4	names.nsf	HTTP/1.1 200 OK	832	□
5	log.nsf	HTTP/1.1 200 OK	1681	□
6	events4.nsf	HTTP/1.1 200 OK	2338	□
7	doladmin.nsf	HTTP/1.1 200 OK	2375	☑
8	dbdirman.nsf	HTTP/1.1 404 Not Found	527	□
9	certsrv.nsf	HTTP/1.1 200 OK	2374	☑
10	certlog.nsf	HTTP/1.1 200 OK	2374	☑
11	admin4.nsf	HTTP/1.1 200 OK	1616	□
12	catalog.nsf	HTTP/1.1 200 OK	2123	□
13	domlog.nsf	HTTP/1.1 200 OK	621	□
14	bookmark.nsf	HTTP/1.1 200 OK	525	□
15	domcfg.nsf	HTTP/1.1 200 OK	516	□
16	connect.nsf	HTTP/1.1 404 Not Found	527	□

progress: 100%



request	payload	response	error	timeout	length	"login incorrect"
7091	favelas	HTTP/1.0 200 Ok	□	□	3733	☑
7092	favella	HTTP/1.0 200 Ok	□	□	3733	☑
7093	favisms	HTTP/1.0 200 Ok	□	□	3733	☑
7094	favored	HTTP/1.0 200 Ok	□	□	3733	☑
7095	favorer	HTTP/1.0 200 Ok	□	□	3733	☑
7096	favours	HTTP/1.0 200 Ok	□	□	3733	☑
7097	favuses	HTTP/1.0 200 Ok	□	□	3733	☑
7098	fawners	HTTP/1.0 200 Ok	□	□	3733	☑
7099	fawnier	HTTP/1.0 200 Ok	□	□	3733	☑
7100	fawning	HTTP/1.0 200 Ok	□	□	3733	☑
7101	fazenda	HTTP/1.0 302 Object ...	□	□	757	□
7102	fearers	HTTP/1.0 200 Ok	□	□	3733	☑
7103	fearful	HTTP/1.0 200 Ok	□	□	3733	☑
7104	fearing	HTTP/1.0 200 Ok	□	□	3733	☑
7105	feasing	HTTP/1.0 200 Ok	□	□	3733	☑
7106	feasted	HTTP/1.0 200 Ok	□	□	3733	☑

progress: 18%

Application mapping

- Response code
- Response size
- Presence of string “not authorized”

Password guessing

- Response code
- Response size
- Presence of string “login incorrect”

Prevention Guidelines

1. Use a standard, secure authentication scheme.
2. Check access control on every transaction.
3. Avoid using interpreters where possible.
4. Don’t leak sensitive information in error pages.
5. Encrypt sensitive data in transit and on disk.
6. Encode user data in output.
7. Don’t trust any data from the client.
8. Validate all input.

Input Validation

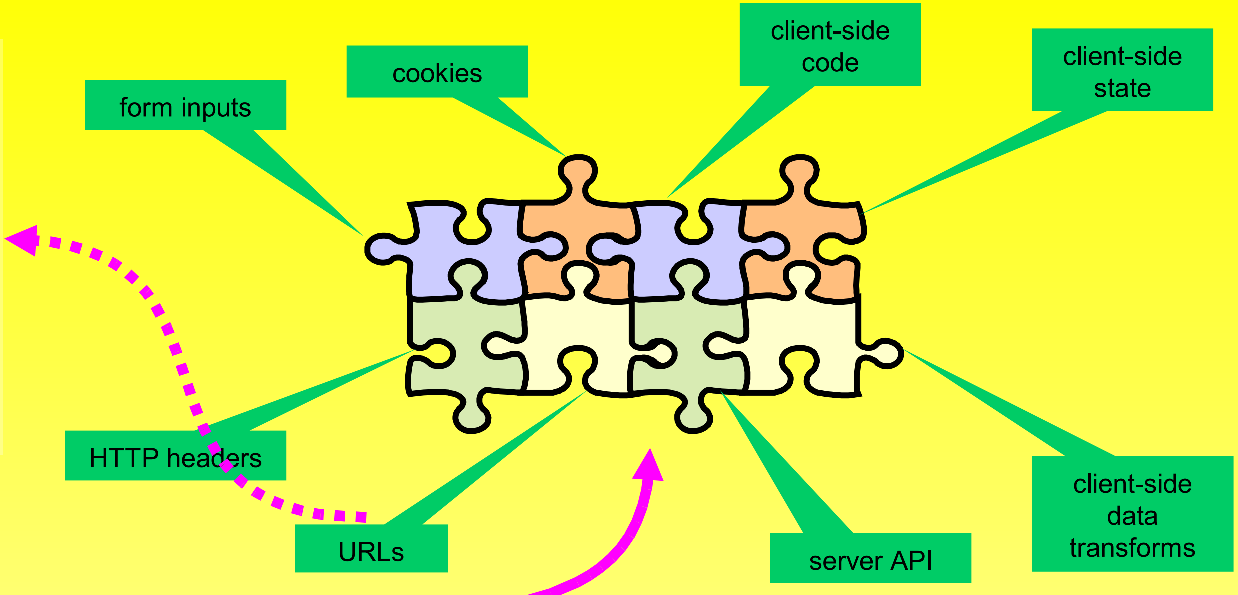
Blacklist: reject known bad input

- Reject input matching list of bad strings/patterns.
- Accept all other input.
- Vulnerable to encoding attacks.

Whitelist: accept known good input

- Accept input matching list of good strings/patterns.
- Reject all other input.
- Highly effective, but not always feasible.

1. Visible Content
   - Linked URLs.
   - Authenticated URLs.
2. Hidden Content
   1. Unlinked sections of site.
   2. Backup copies of live files.
   3. Configuration/include files.
   4. Source code.
   5. Log files.



form inputs

cookies

client-side code

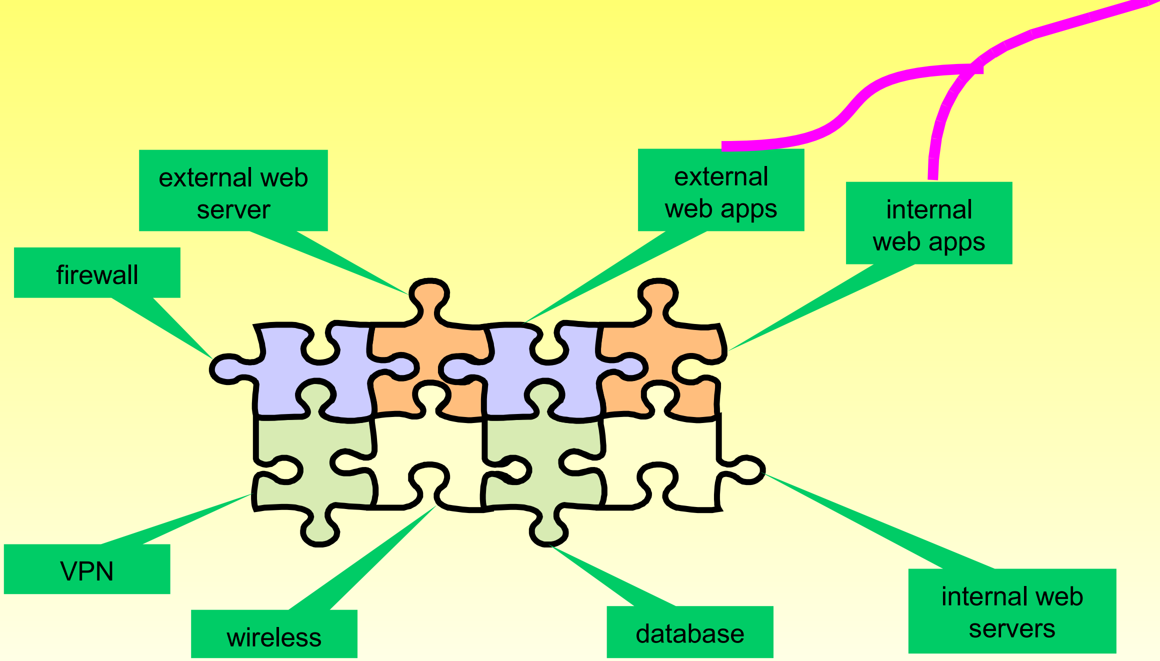
client-side state

HTTP headers

URLs

server API

client-side data transforms



external web server

firewall

external web apps

internal web apps

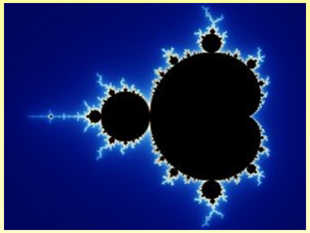
VPN

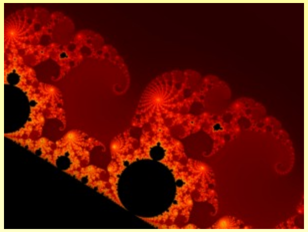
wireless

database

internal web servers

A site’s attack surface





is nearly fractal.