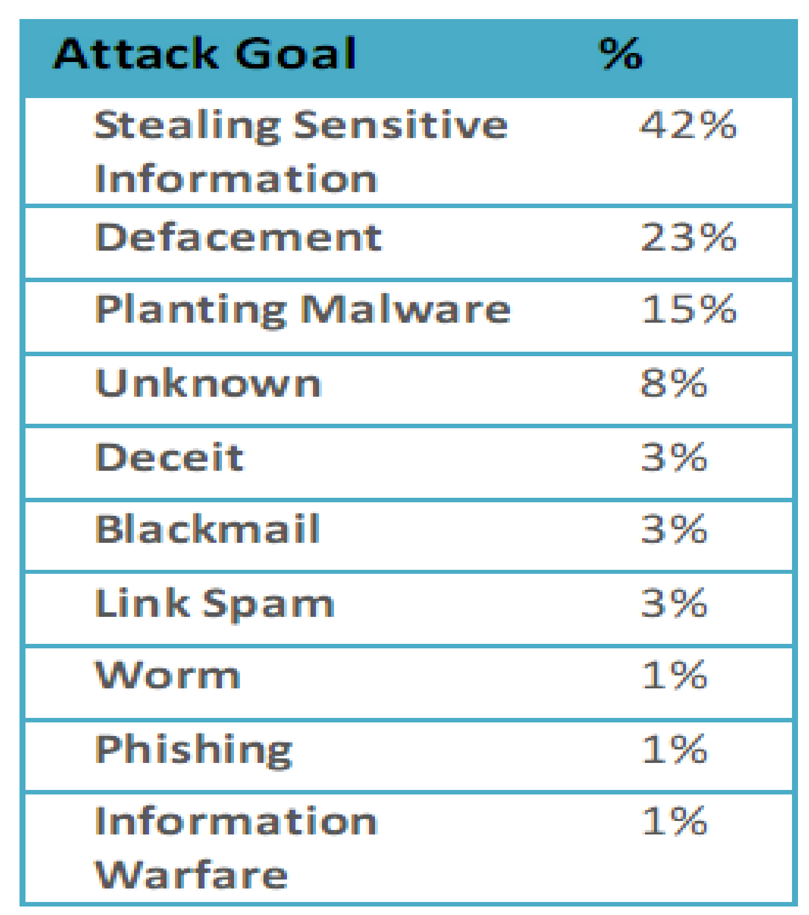
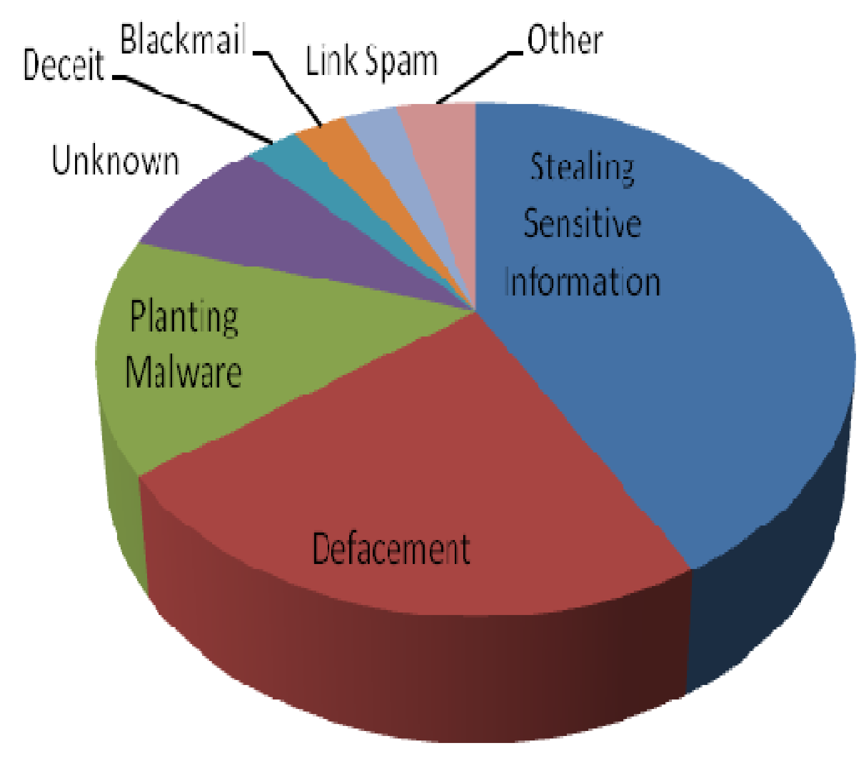
Attack Surface of Web Applications

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%

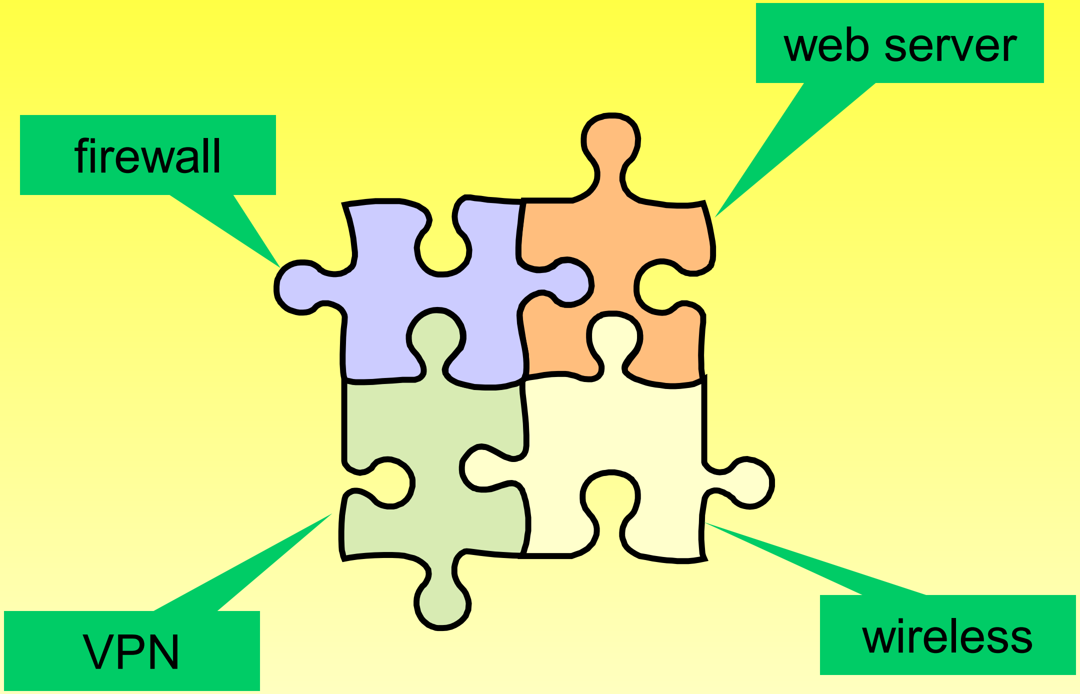


Attack Surface

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



web server

firewall

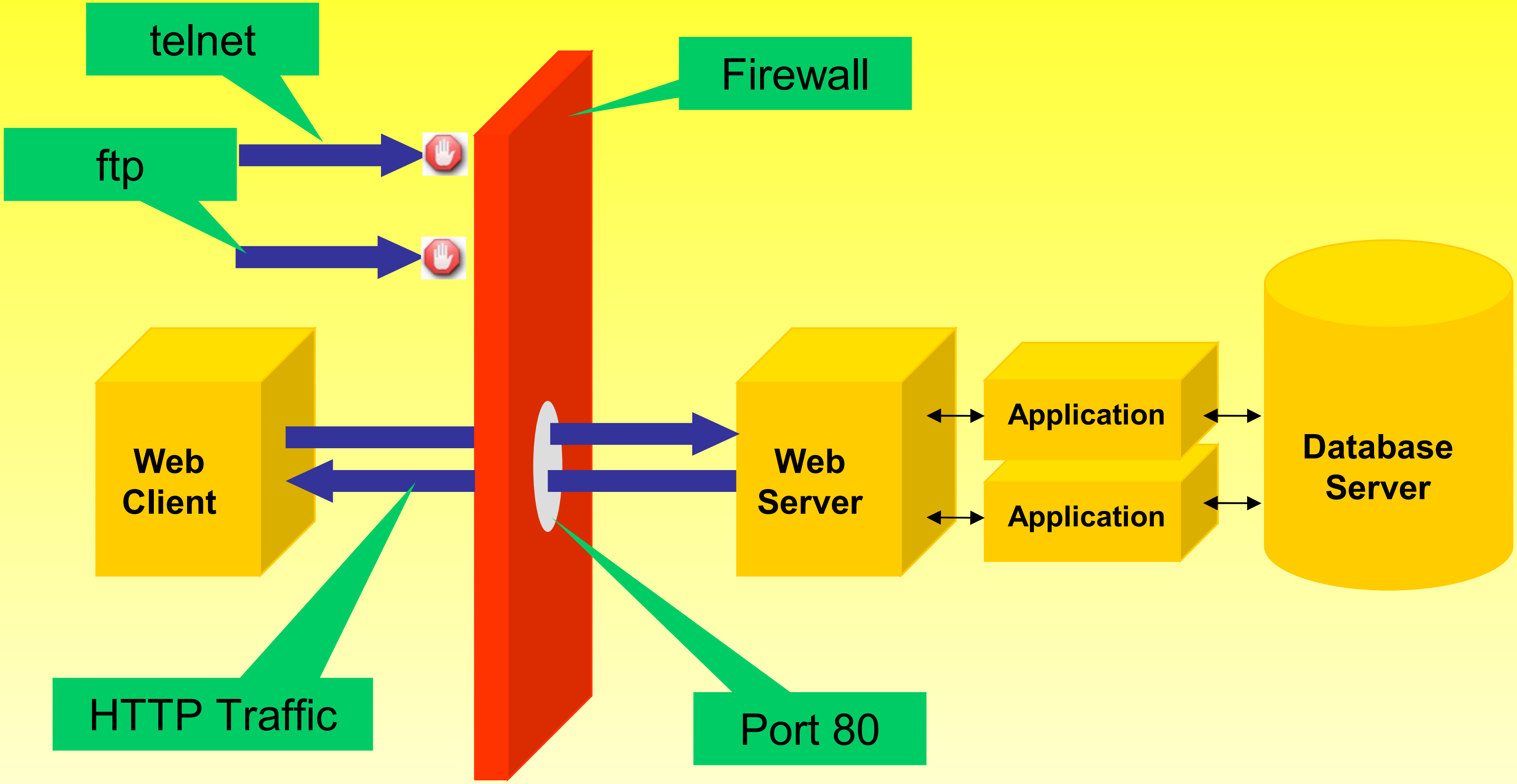
VPN

wireless

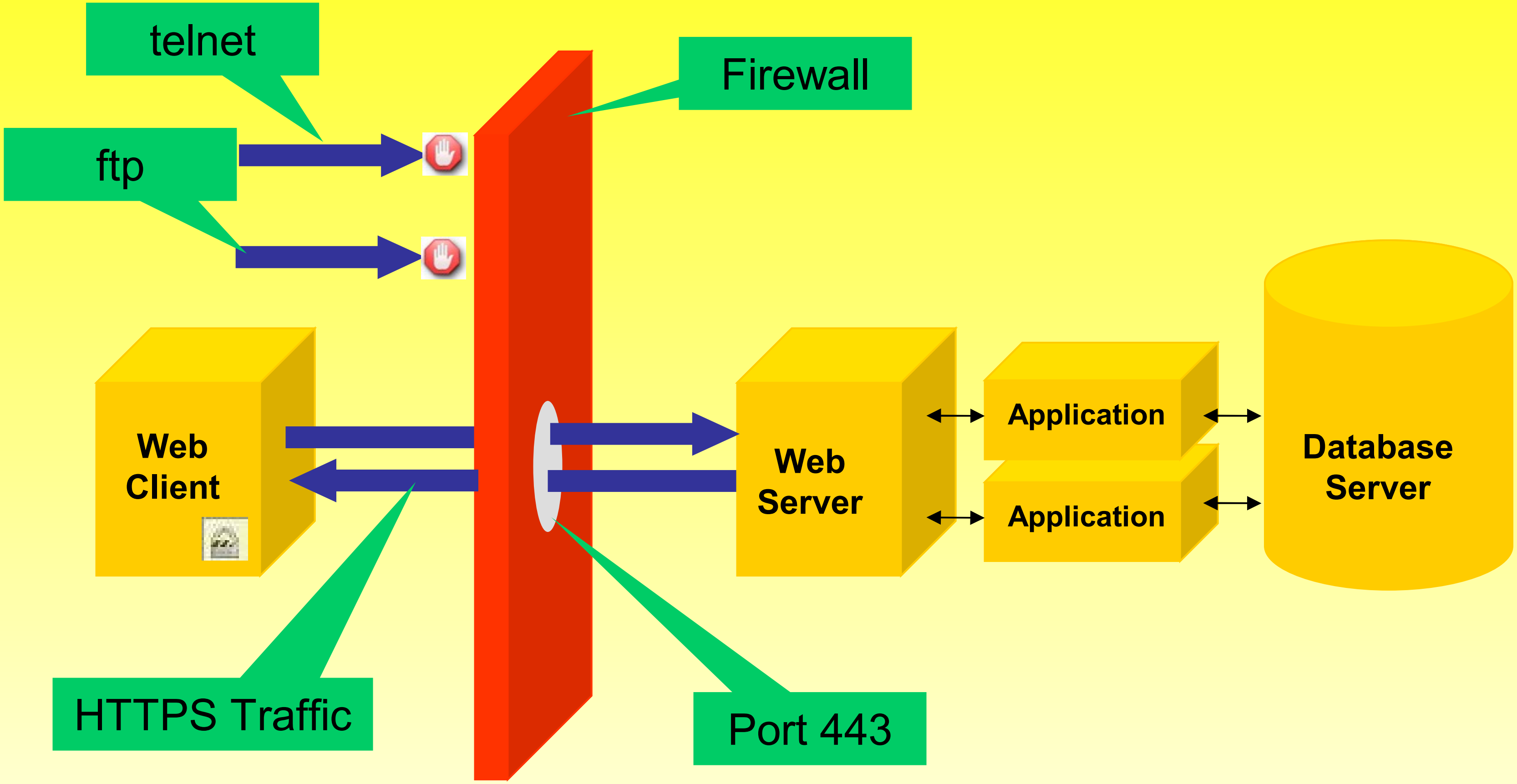
History of Web Security

Year	Technology	Security
1993	CGI	Firewalls, SSL
1995	PHP, Javascript	Firewalls, SSL
1997	ASP, JSP	Firewalls, SSL
2000	REST, SOA	Firewalls, SSL
2006	AJAX	Firewalls, SSL

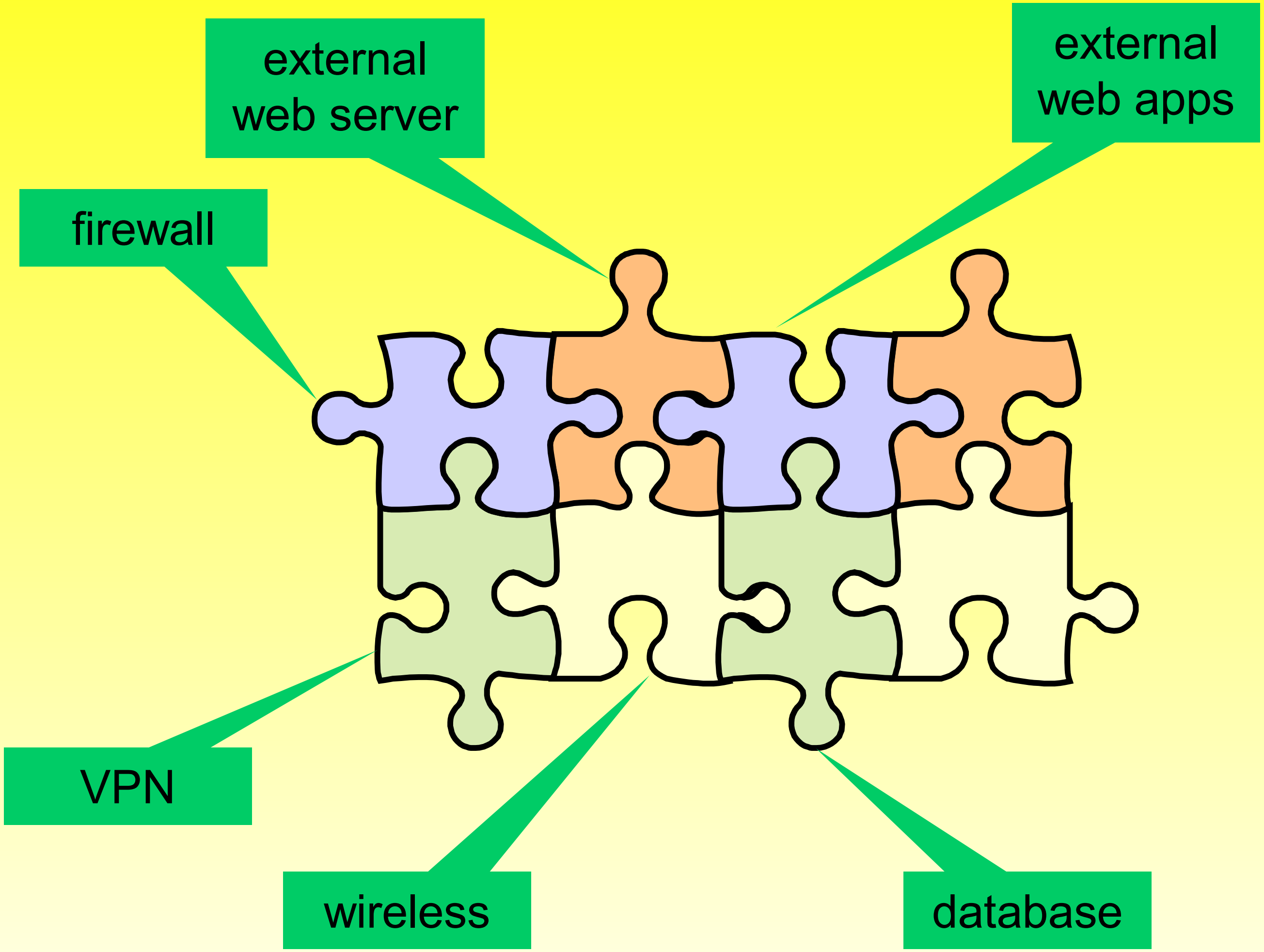
Firewalls don’t protect Web Apps



SSL won’t stop injection attacks, XSS



Revised View of Attack Surface



external web server

external web apps

firewall

VPN

wireless

database

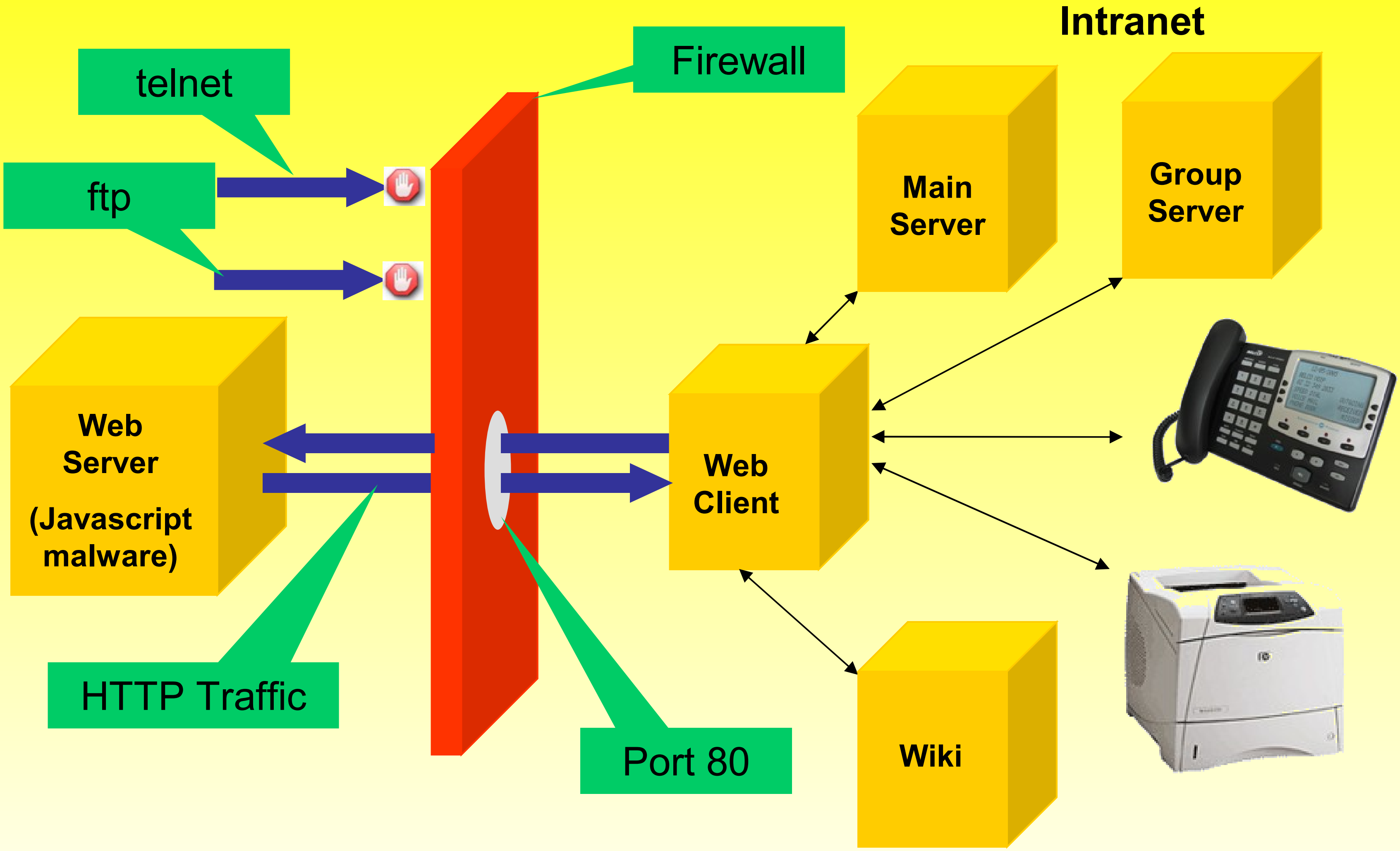
Intranet Security Assumptions

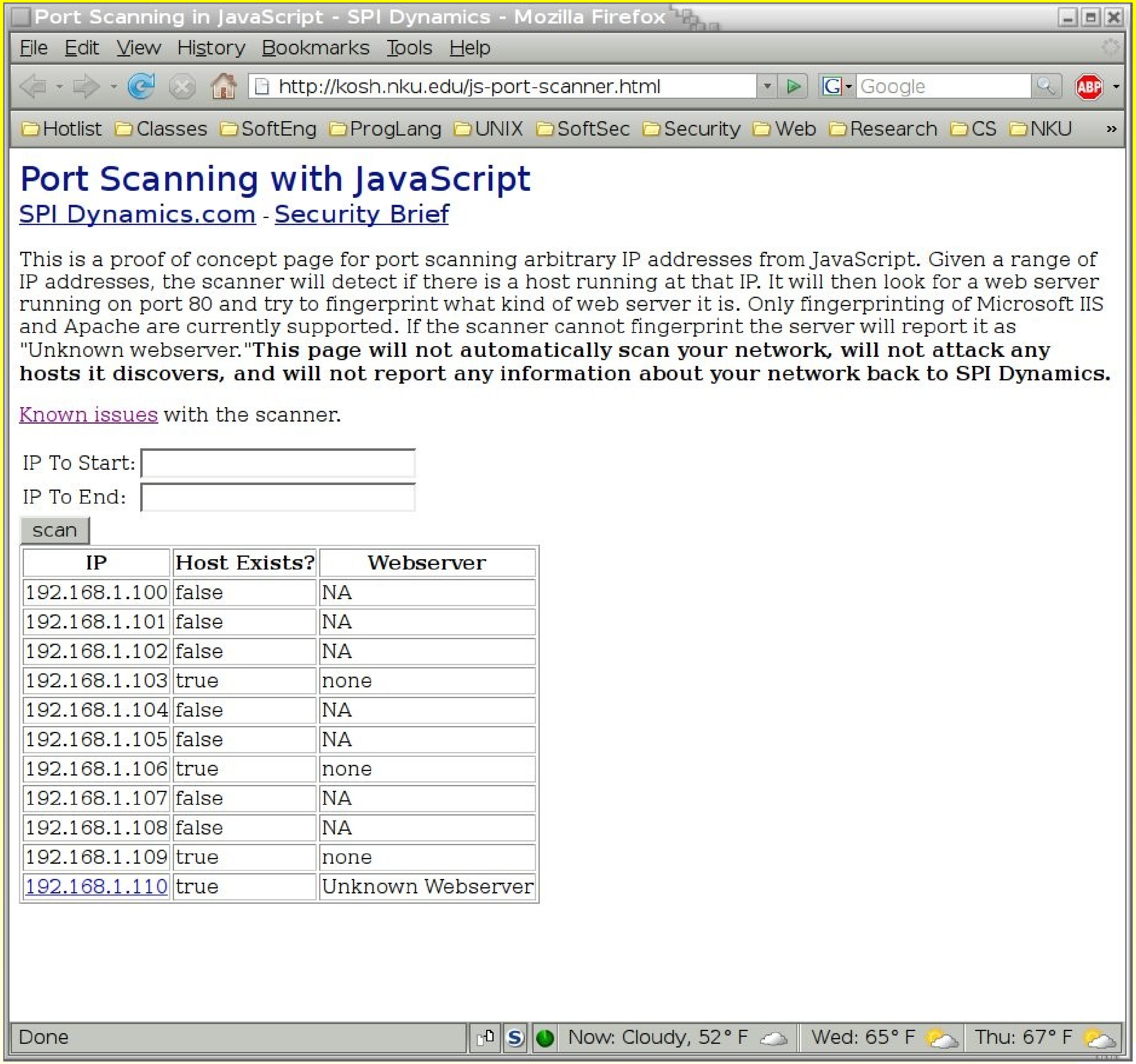
Since the firewall protects you

- 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




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:

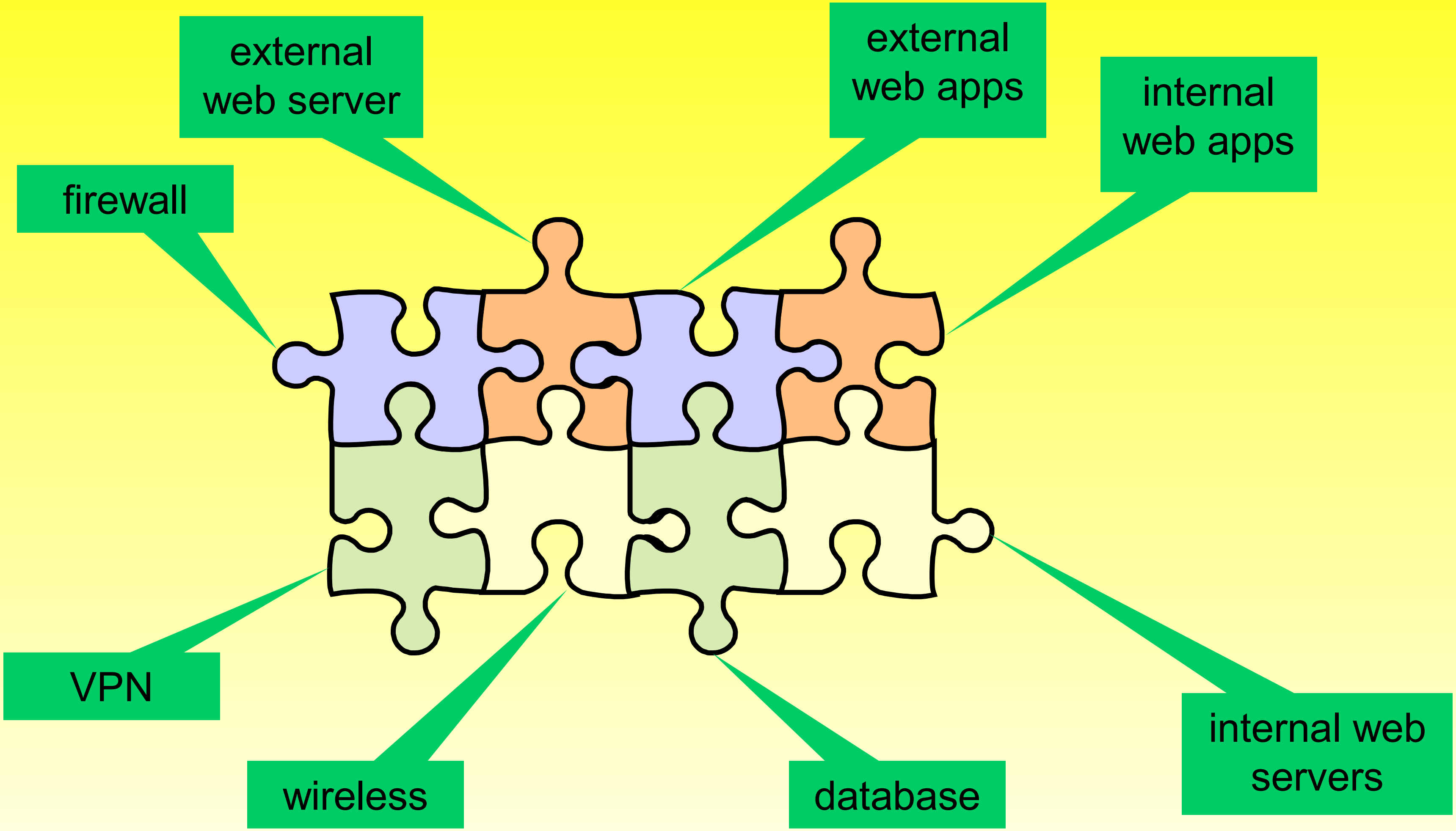
scan

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

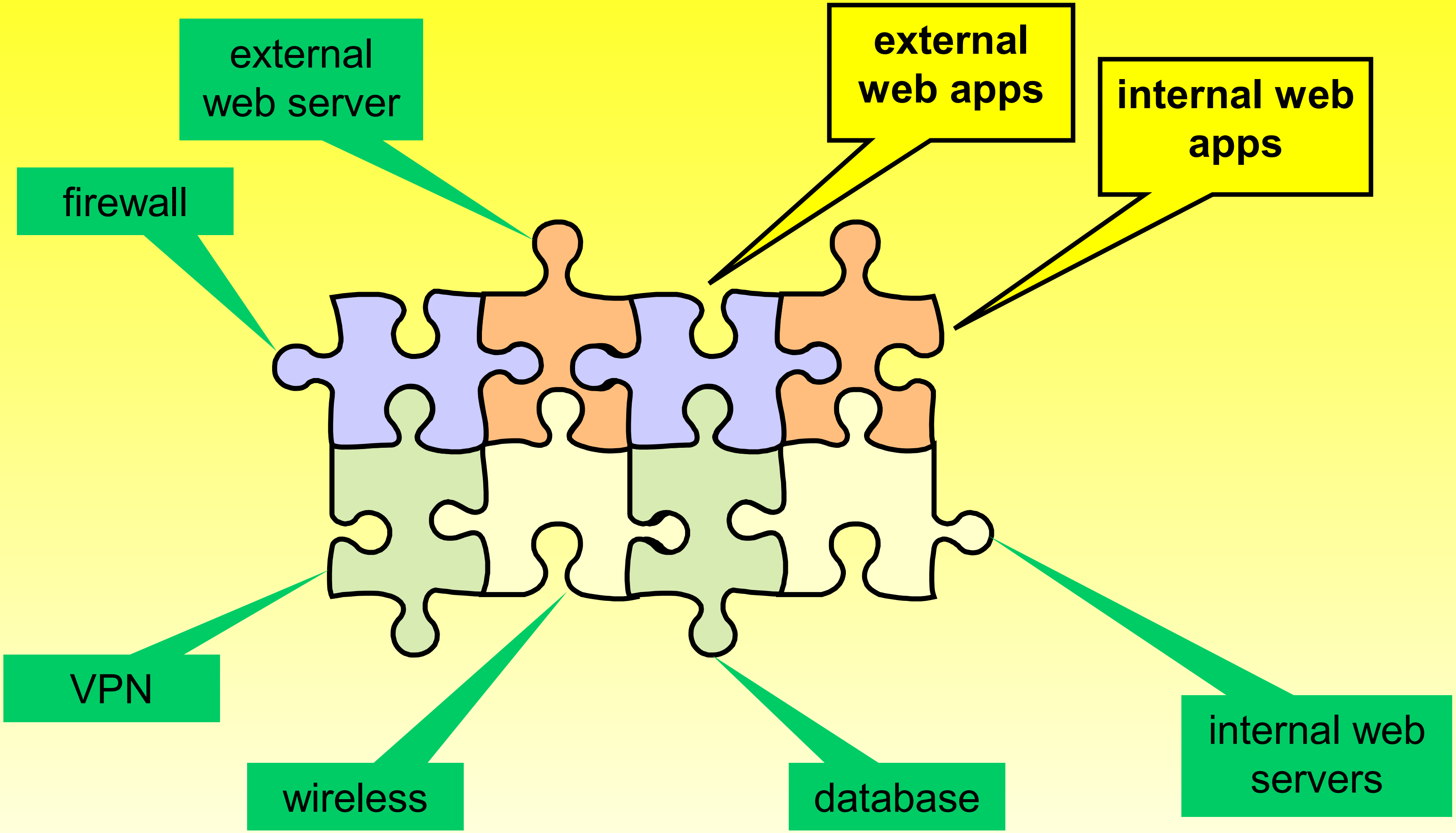
Sources of Javascript Malware

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



Web Applications



external web server

firewall

external web apps

internal web apps

VPN

wireless

database

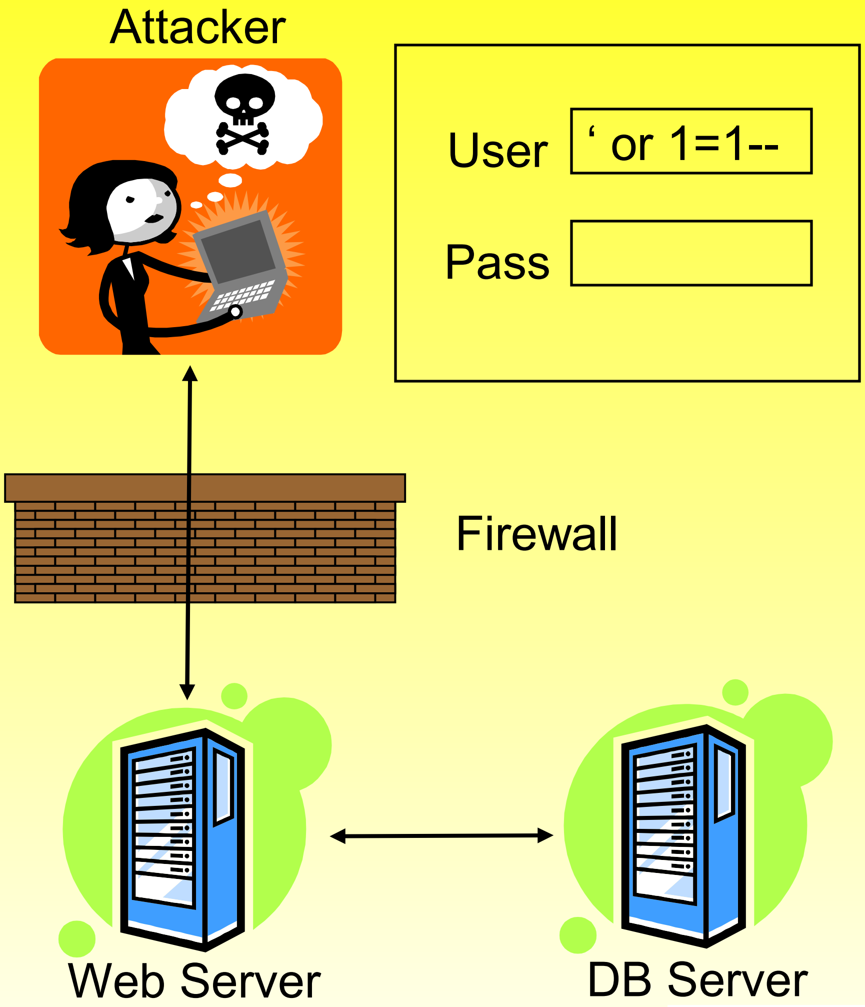
internal web servers

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

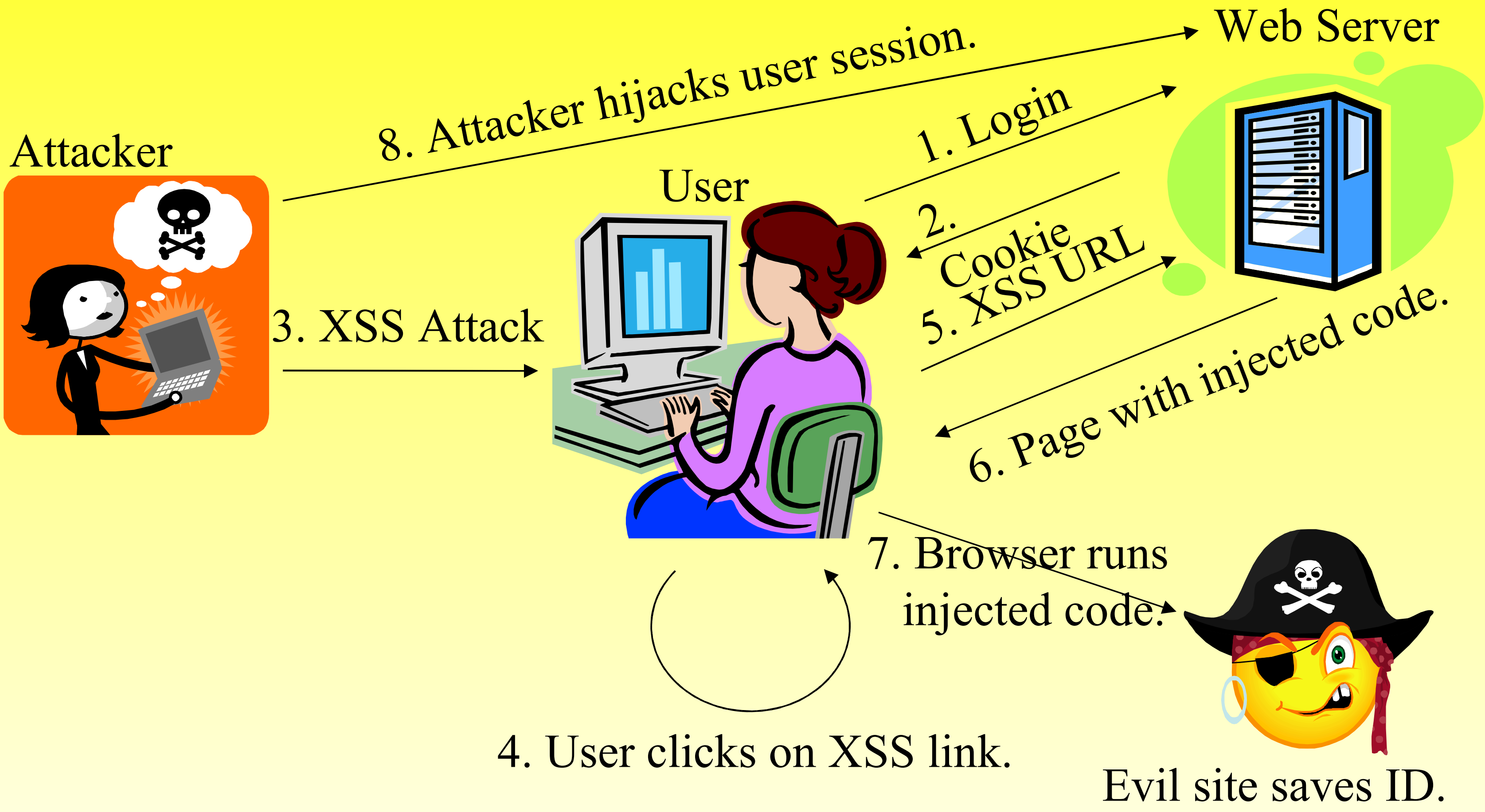
1. App sends form to user.
2. Attacker submits form with SQL exploit data.
3. Application builds string with exploit data.
4. Application sends SQL query to DB.
5. DB executes query, including exploit, sends data back to application.
6. Application returns data to user.



User ‘ or 1=1--

Pass

Cross-Site Scripting



Attacker

8. Attacker hijacks user session.

Web Server

1. Login

User

2. Cookie

5. XSS URL

3. XSS Attack

6. Page with injected code.

7. Browser runs injected code.

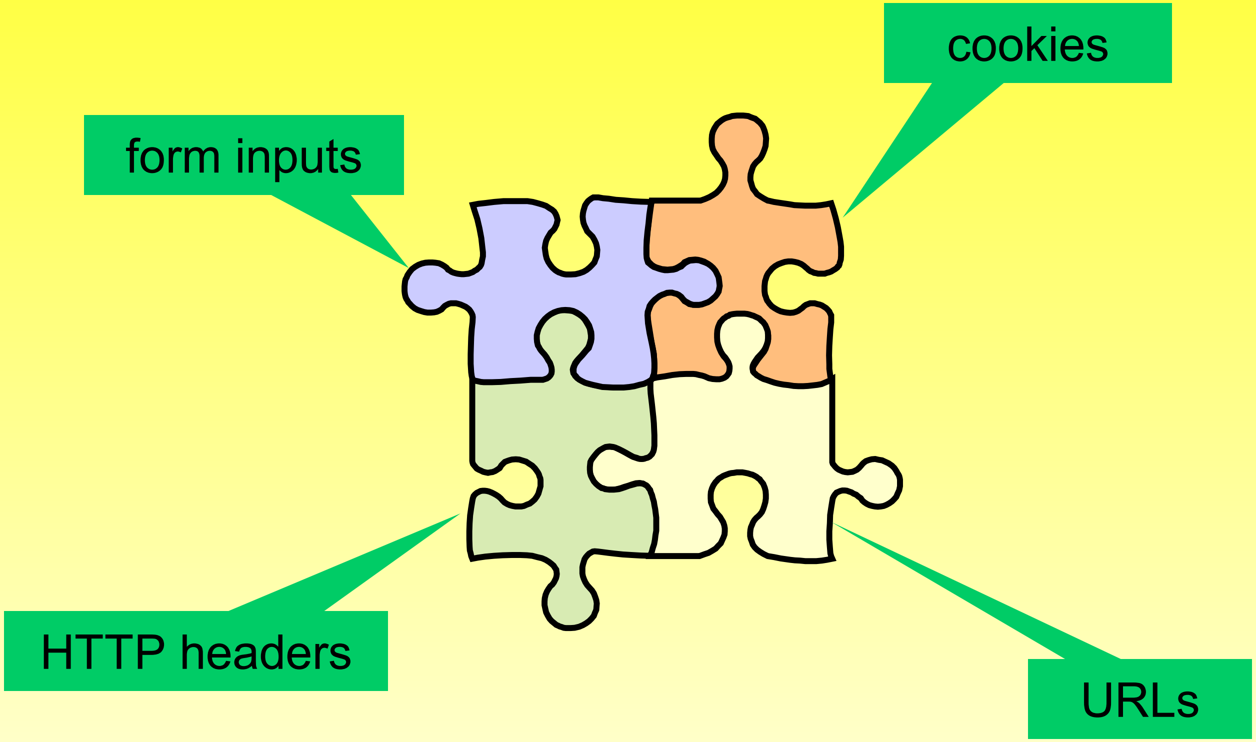
4. User clicks on XSS link.

Evil site saves ID.

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.

Web Application Attack Surface



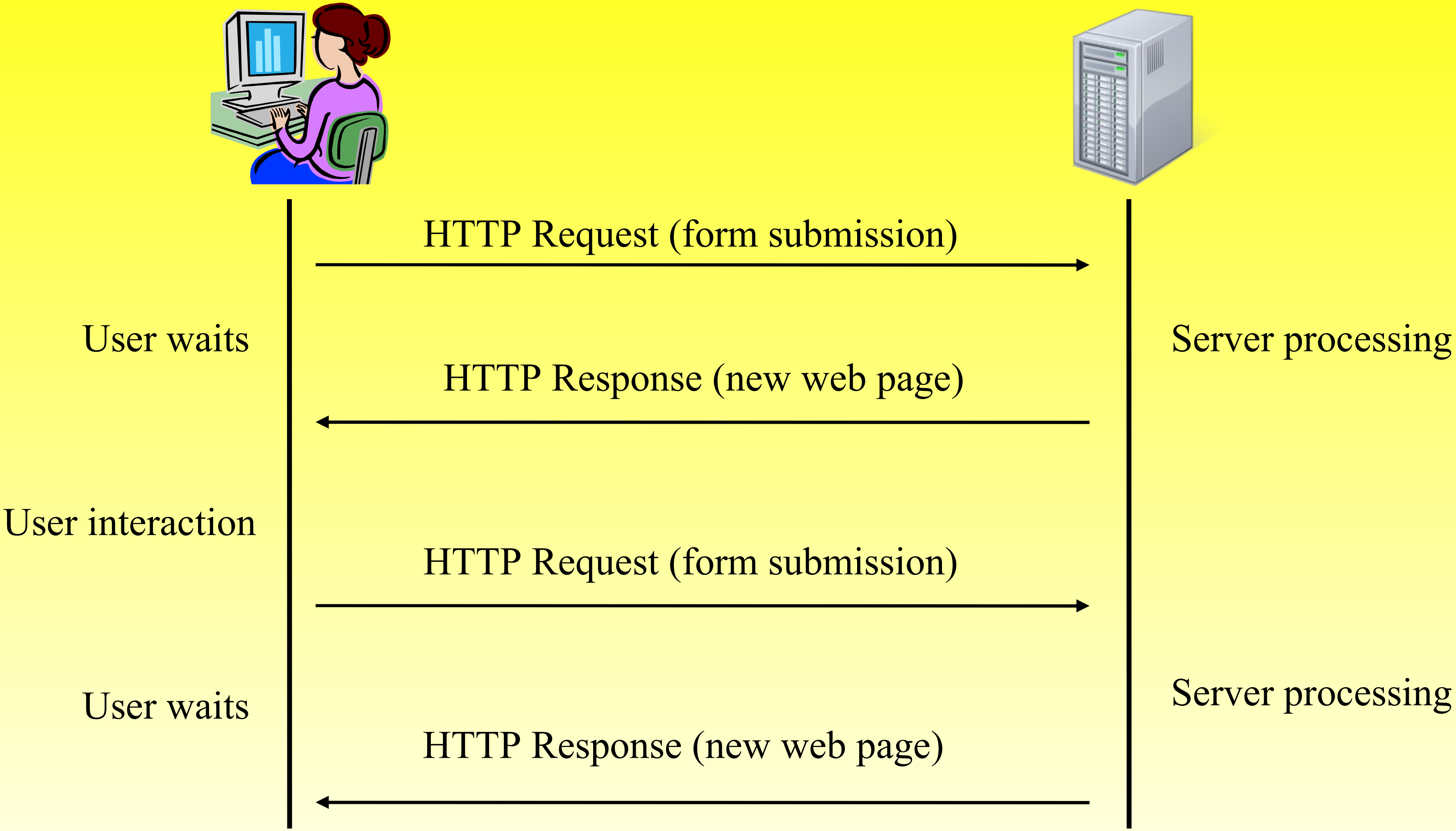
cookies

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

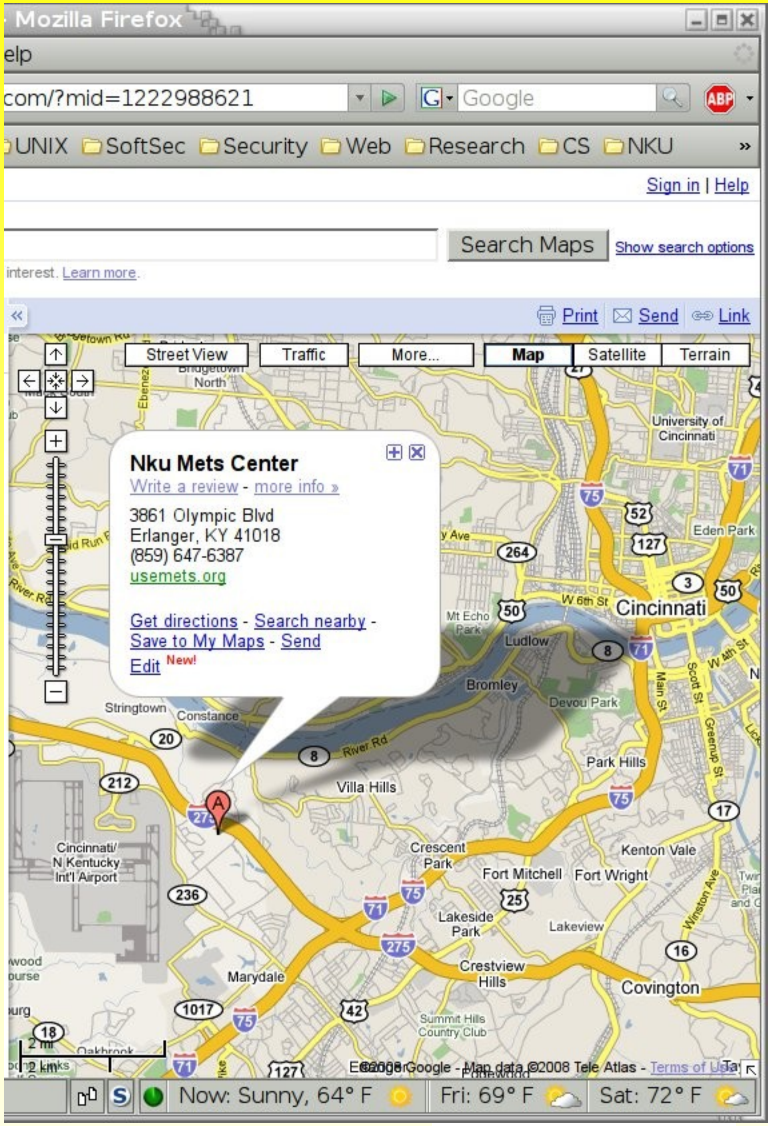
Server processing

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.



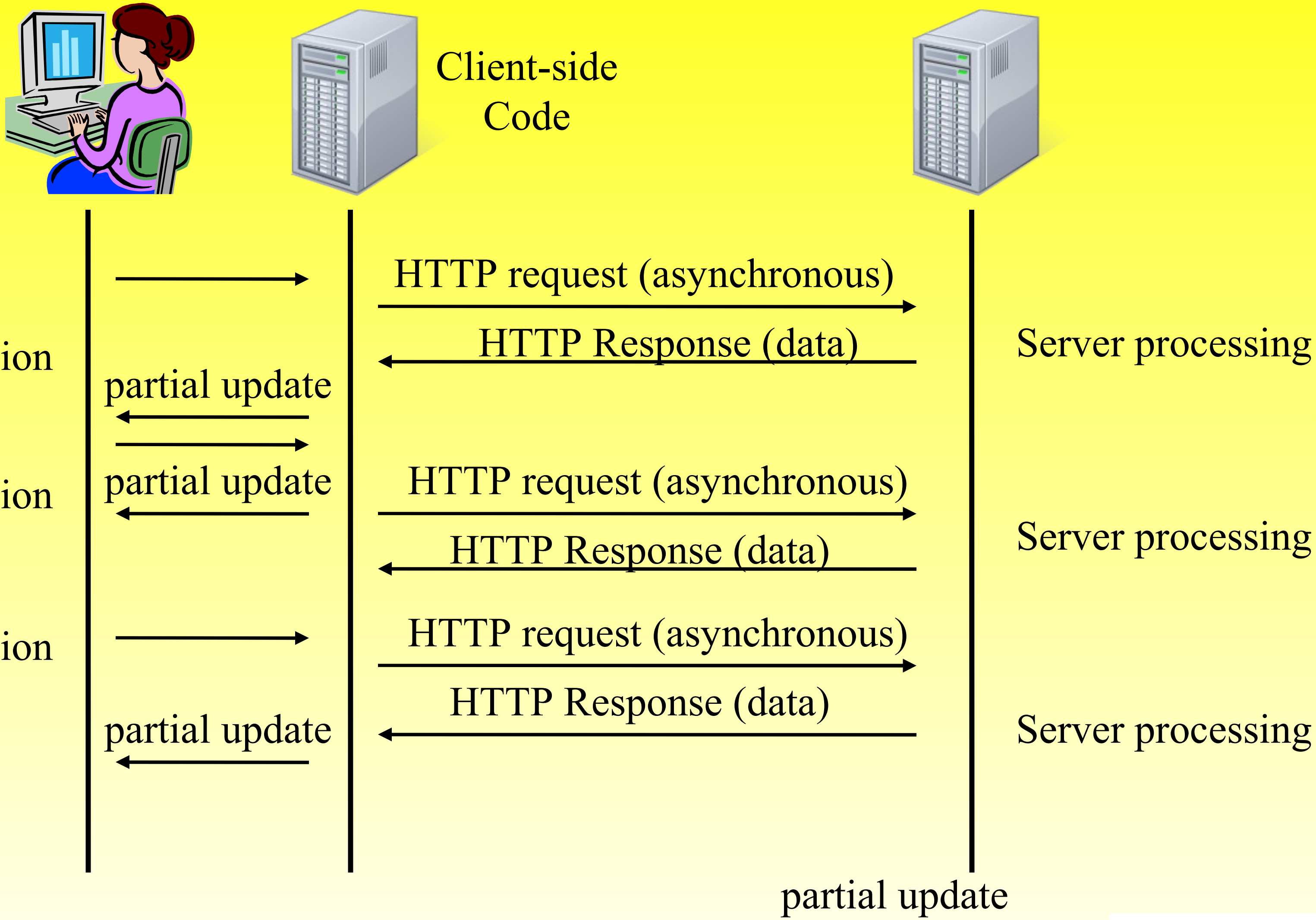
Nku Mets Center

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

Architecture Differences

Traditional

- Application on server.
- Entire form sent to server.
  - User fills in input items.
  - Clicks on submit.
- Server returns new page.
  - Presentation + Data.

AJAX

- App on client and server.
- JavaScript receives user input, issues function calls to server when needed.
  - Get map tile.
  - Save location data.
- Server returns individual data items.
- JavaScript incorporates data items into existing page.

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);
```

JSON

```
var json = getItem()
// json = “[ ‘Toshiba’, 499, ‘LCD TV’]”

var item = eval(json)
// item[0] = ‘Toshiba’
// item[1] = 499
// item[2] = ‘LCD TV’
```

JSON Injection

```
Evil input: ‘];alert(‘XSS’);//

var json = getItem()
// json = “[ ‘Toshiba’, 499, ‘’];alert(‘XSS’);//”

var item = eval(json)
// Alert box with ‘XSS’ appears.
// Use json2.js validation library to prevent.
```

Client-Side State

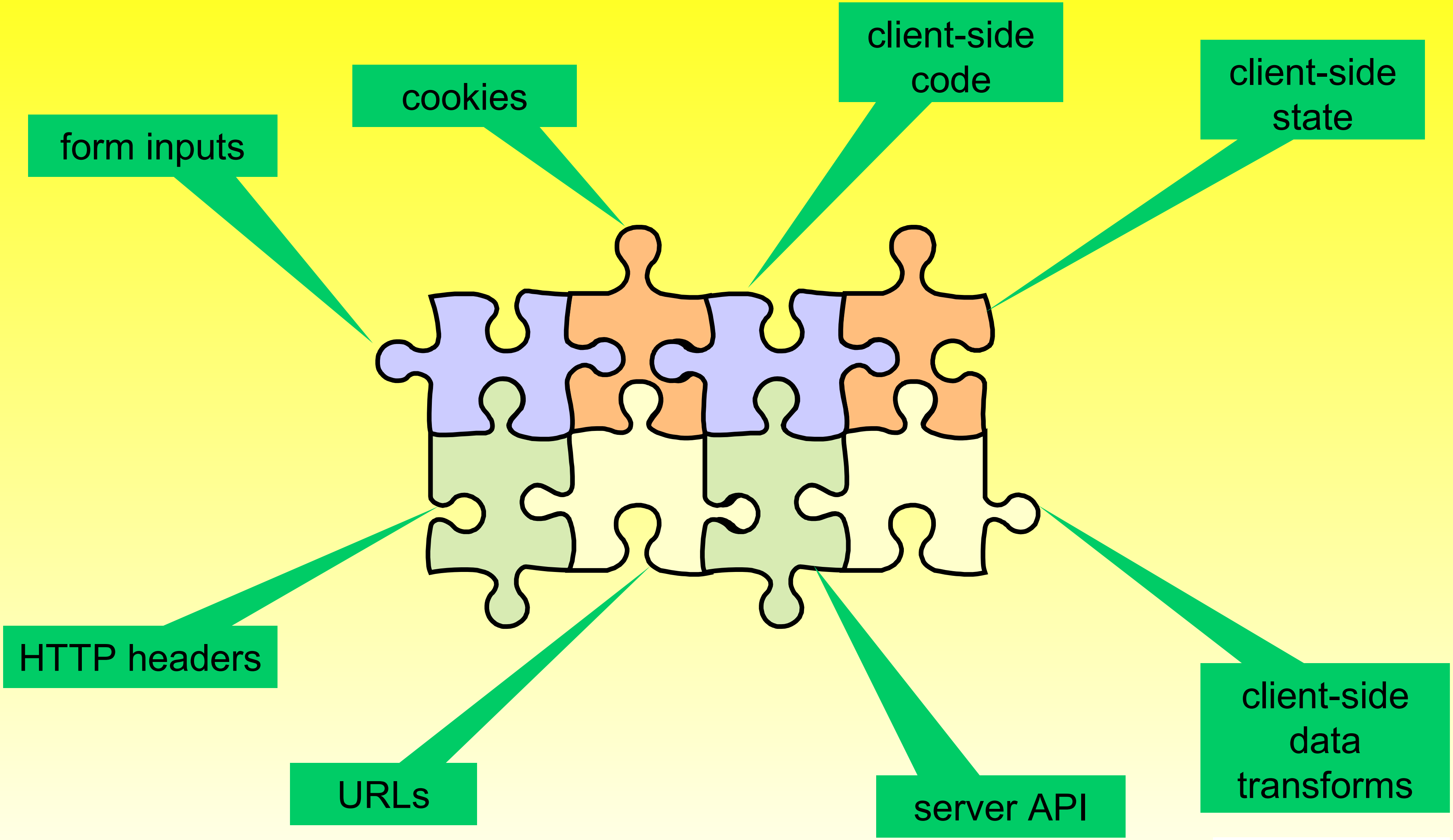
Storage Technologies

- Cookies
- DOM Storage (HTML5)
- Flash LSOs
- UserData (IE)

Client-Side Storage Issues

- User can always modify client-side data.
- Cross-Domain Attacks (between subdomains).
- Cross-directory Attacks.
- Cross-port Attacks.

AJAX Application Attack Surface



form inputs

cookies

client-side code

client-side state

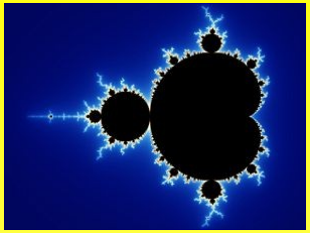
HTTP headers

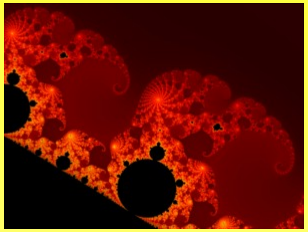
URLs

server API

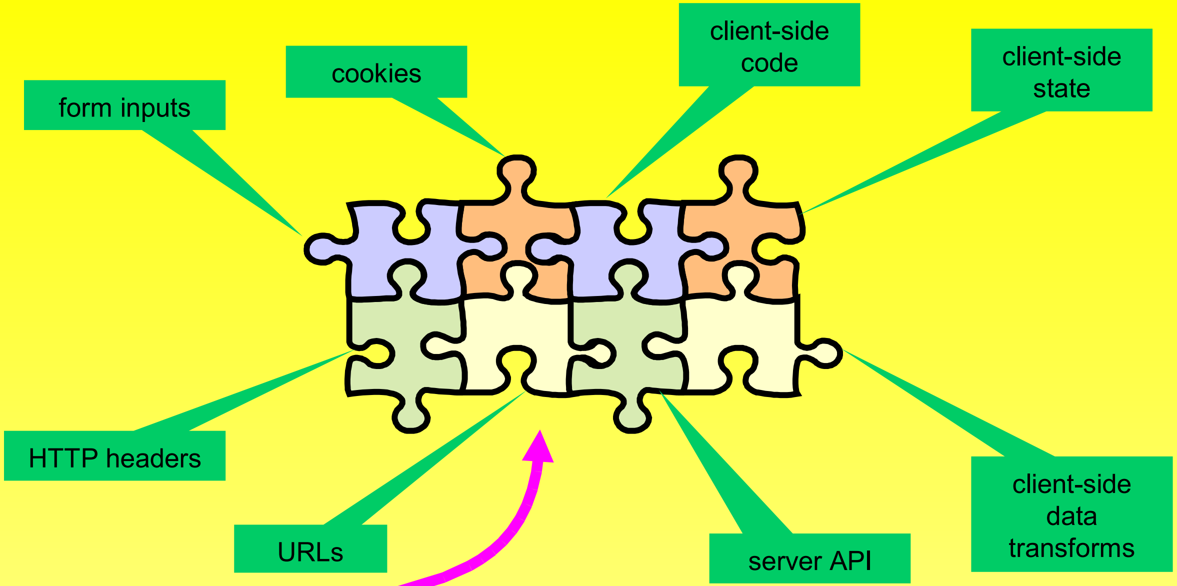
client-side data transforms

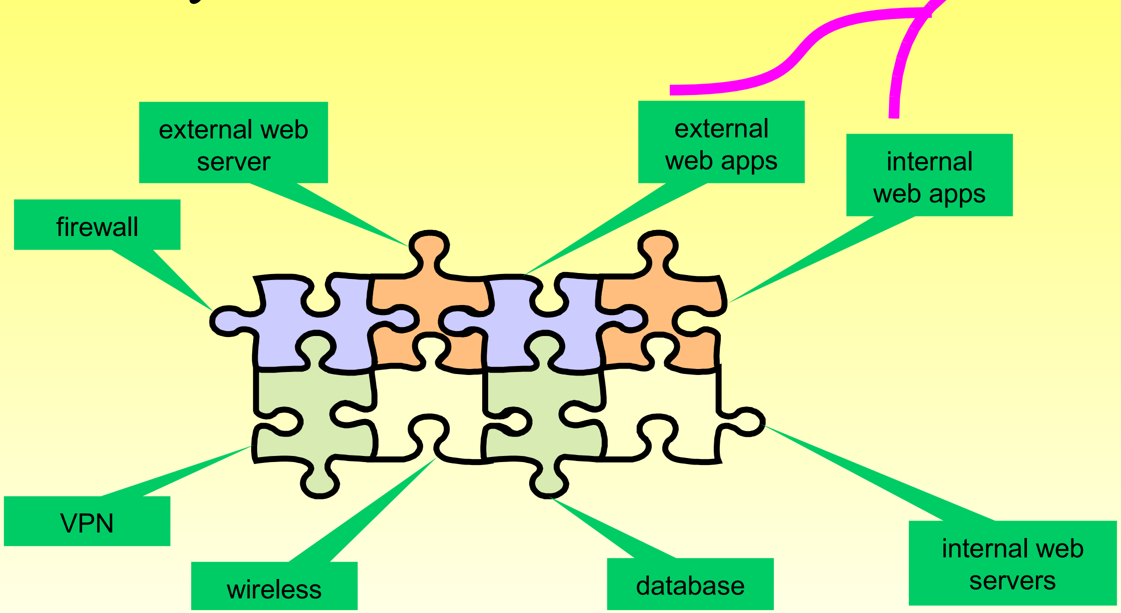
A site’s attack surface





is nearly fractal.


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