

Integrating Web Application Security into the IT Curriculum

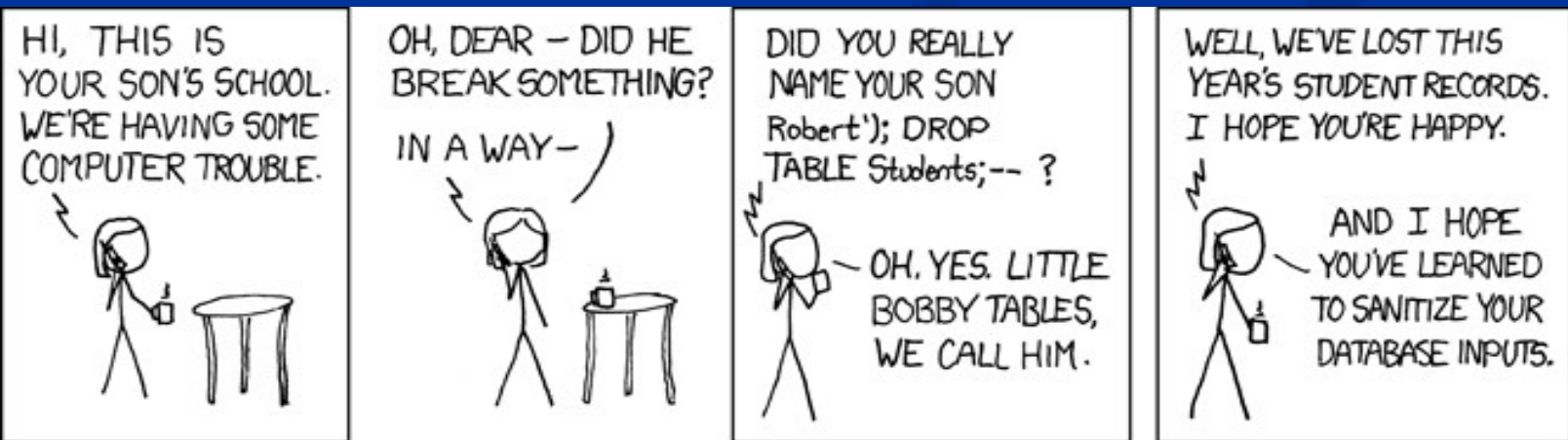
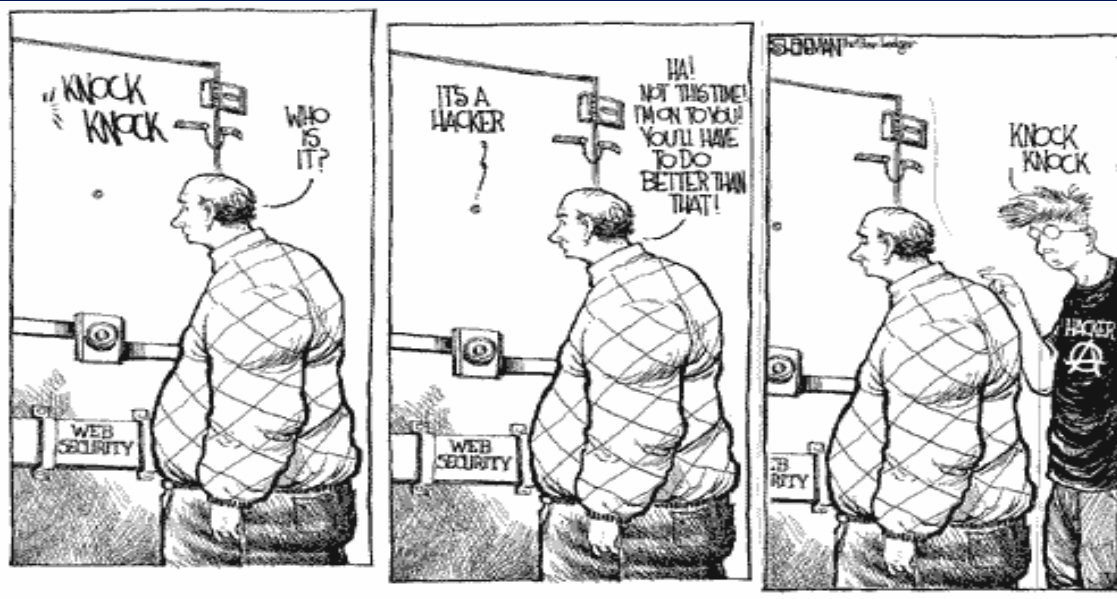
James Walden

Northern Kentucky University

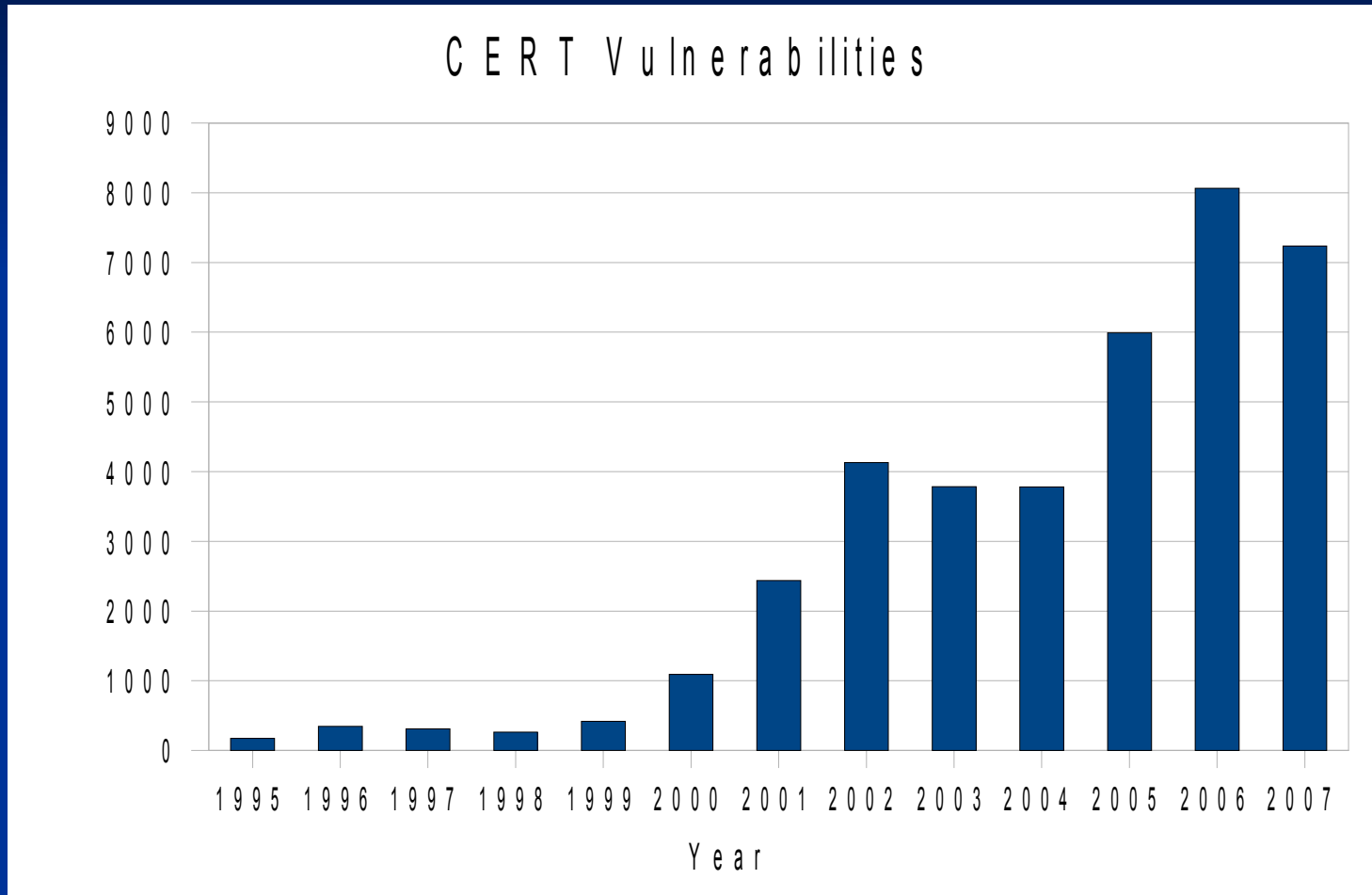
Topics

1. Why should we teach web application security?
2. What material do we need to cover?
3. How should we cover that material?
4. Where do we go from here?

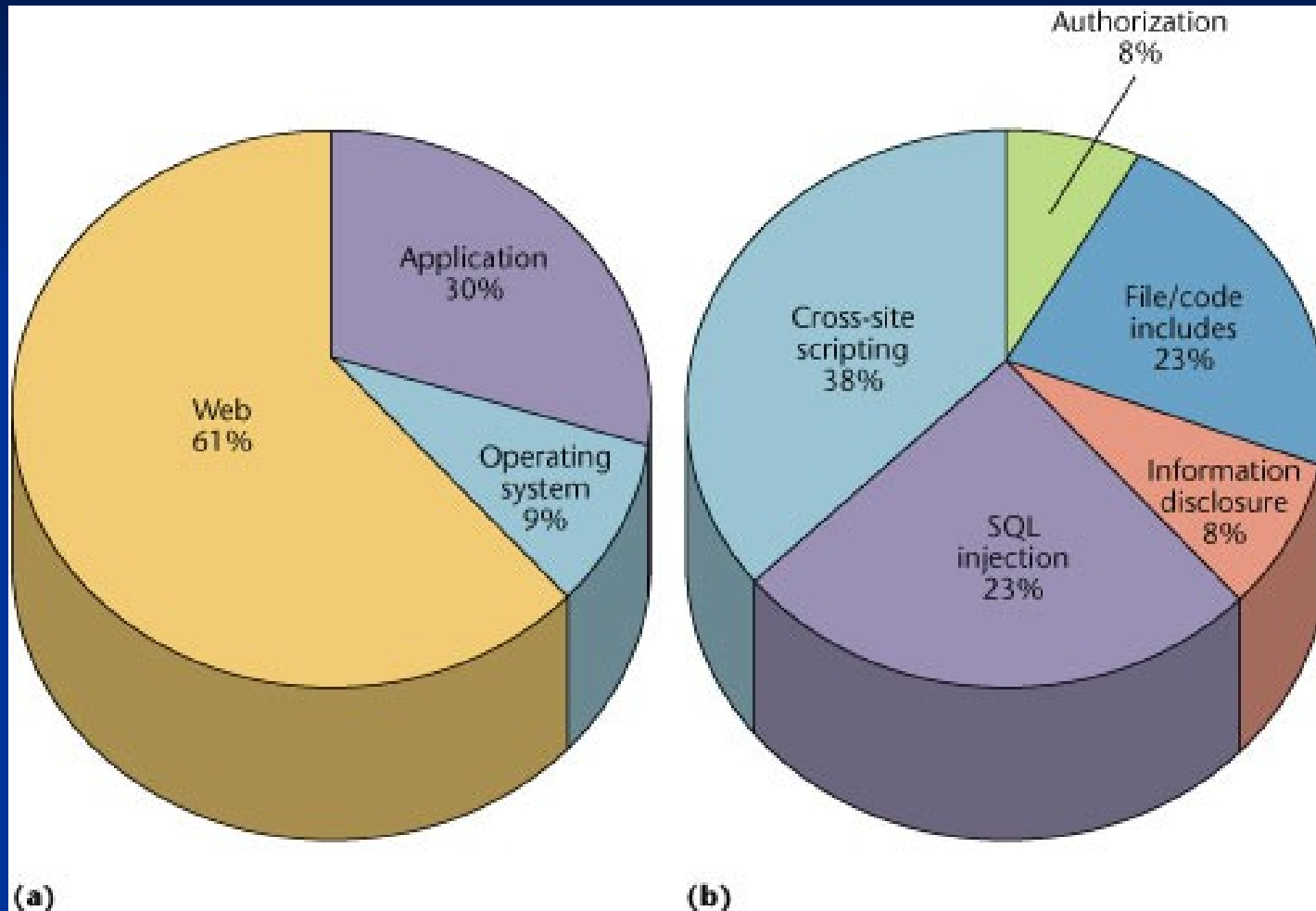
Is Web Hacking Really That Easy?



Vulnerability Growth

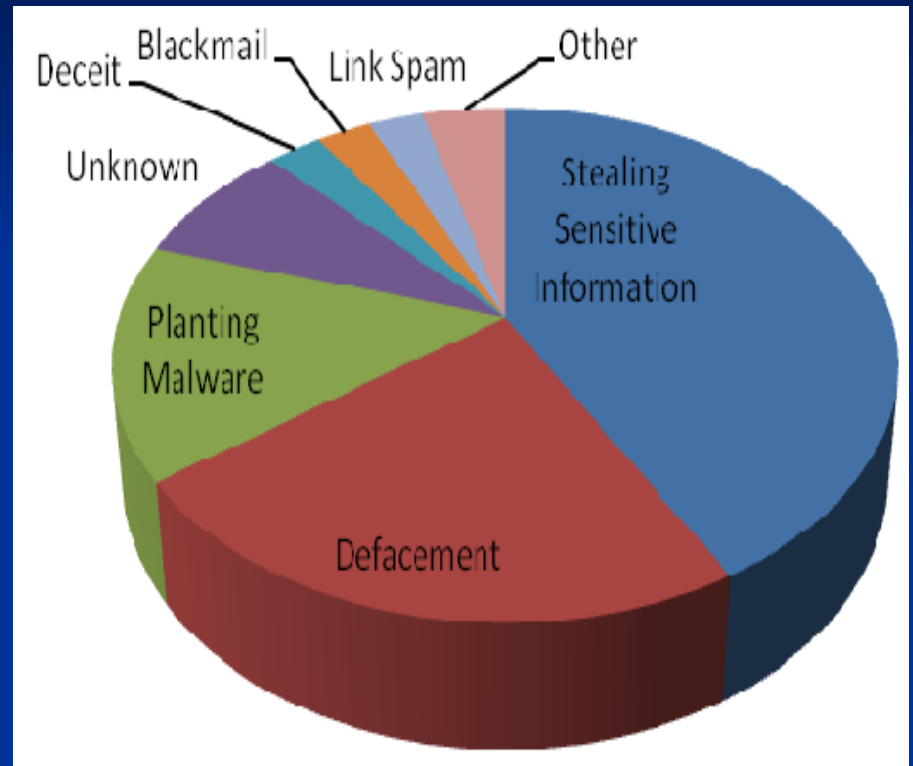


Web Vulnerabilities Dominate



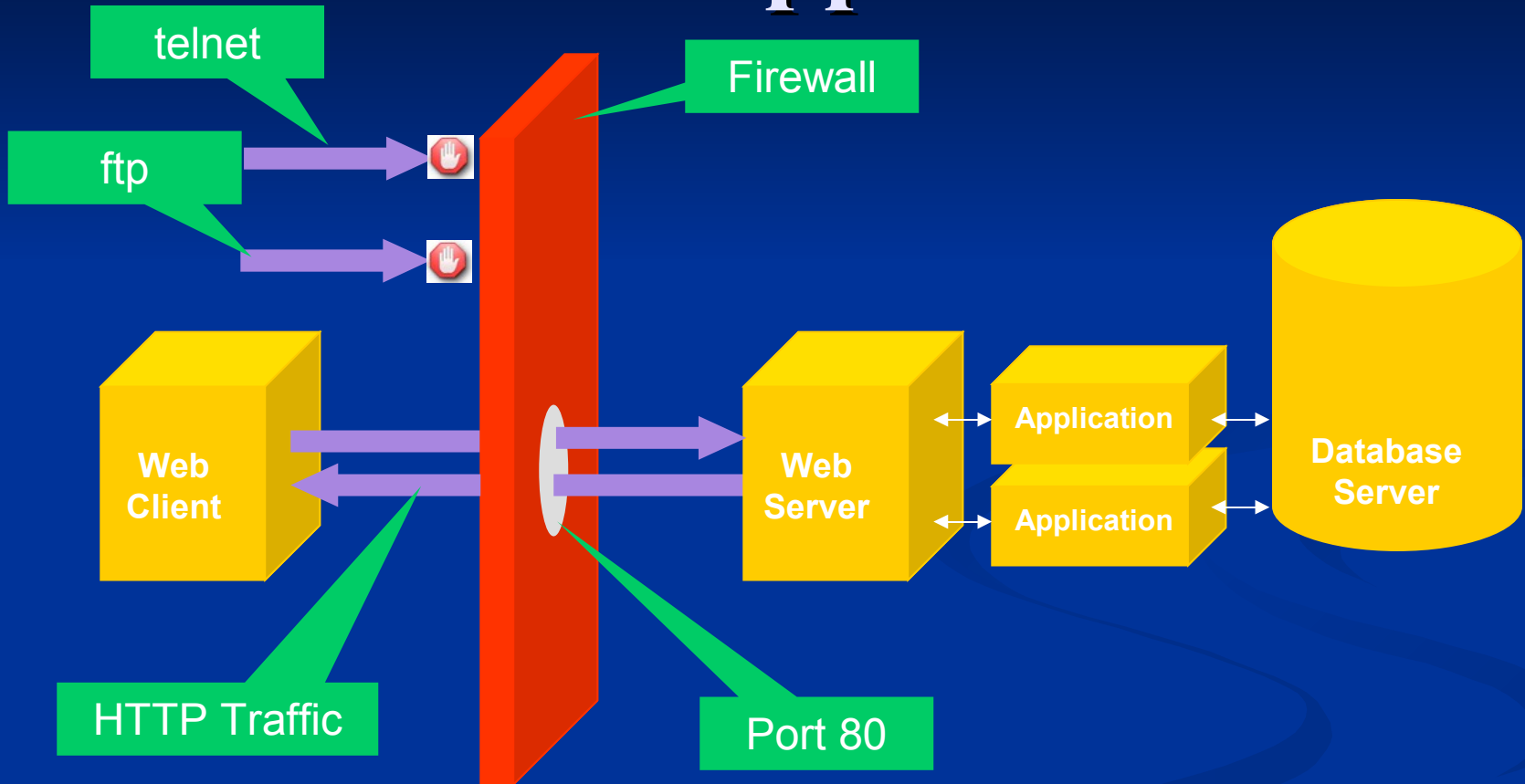
Reasons for Attacking Web Apps

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%



Firewalls Don't Protect Web

Apps



Browser Malware Bypasses Firewall

Port Scanning in JavaScript - SPI Dynamics - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://kosh.nku.edu/js-port-scanner.html

Hotlist Classes SoftEng ProgLang UNIX SoftSec Security Web Research CS NKU

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

Done

Now: Cloudy, 52° F Wed: 65° F Thu: 67° F

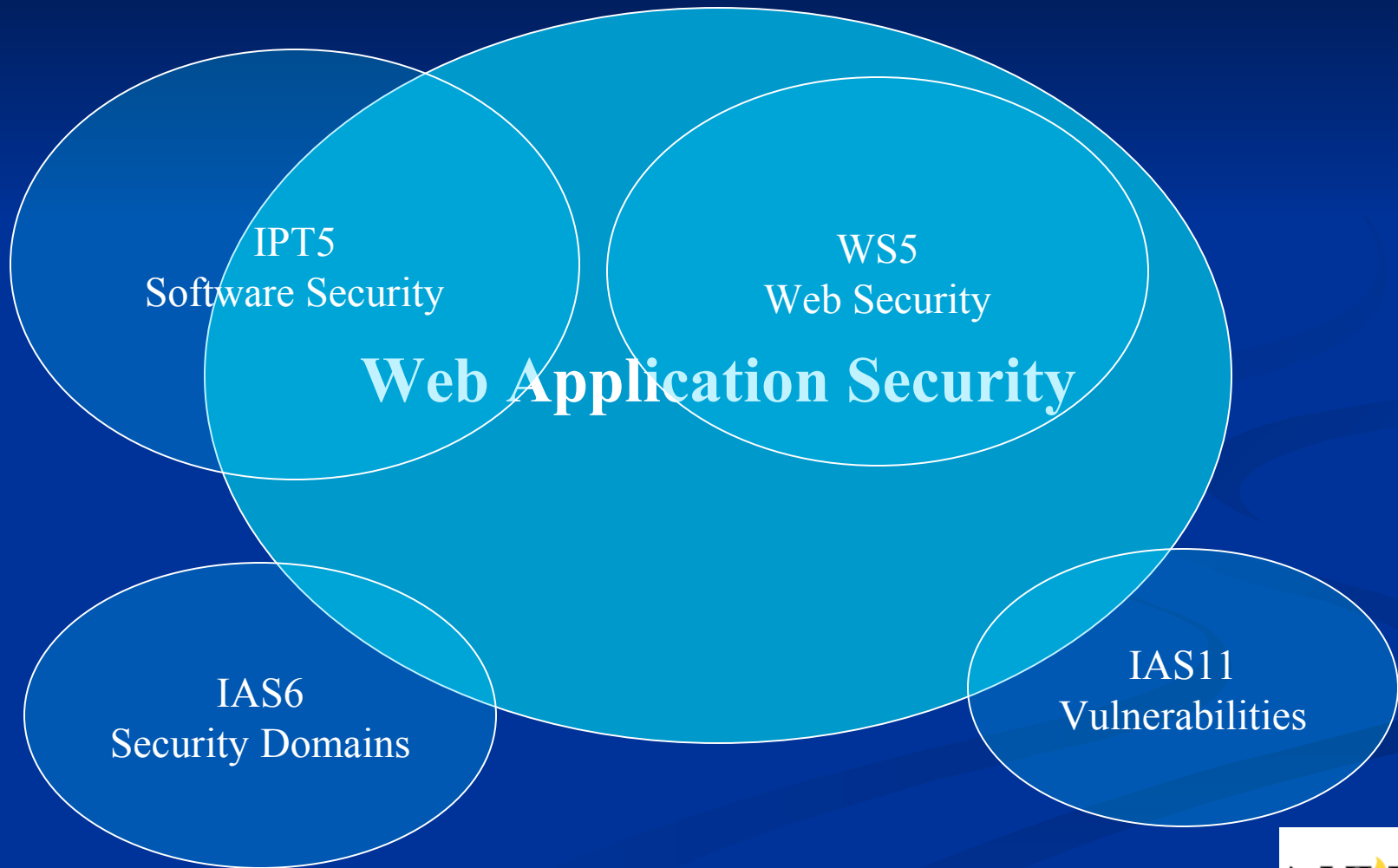
Goals

1. Identify and explain common vulnerabilities.
2. Explain security implications of client-side technologies like Javascript and ActiveX.
3. Detect security vulnerabilities in web applications using appropriate tools.
4. Design and implement web applications that do not contain common vulnerabilities.
5. Deploy and configure a web application in a secure manner.

Topic Outline

1. Web Application Input
2. Client-side Technologies
3. Input-based Attacks
4. Injection Attacks
5. Cross-site Attacks
6. Authentication
7. Secure Programming
8. Operational Security

Web App Security in IT2005



Labs

1. WebGoat exercises on specific vulnerabilities.
2. Using a testing proxy to solve more advanced WebGoat exercises.
3. Assessing an application using a web vulnerability scanner.
4. Assessing a web application using a testing proxy.
5. Reviewing the code of an application using a static analysis tool.
6. Deploying a web application firewall.
7. Participating in the international CTF competition.

WebGoat

The screenshot shows a Netscape browser window with the following elements:

- Browser Title:** Http Basics - Netscape
- Address Bar:** http://localhost/WebGoat/attack
- Navigation:** Back, Forward, Home, Reload buttons.
- Search:** Search bar with "Enter Search Terms" placeholder.
- Page Header:** "Highlight your search words on this page" and "Sponsored" text.
- WebGoat Logo:** "WebGoat *Blame it on the Goat!"
- Sidebar (Left):**
 - Admin Functions:**
 - Refresh Database
 - View Database
 - Products
 - Users
 - Admin Functions (repeated):**
 - Report Card
 - General:**
 - Http Basics
 - Thread Safety
 - Code Quality:**
 - HTML Clues
 - Unvalidated Parameters:**
 - Hidden Field Tampering
 - Unchecked Email
 - JavaScript Validation
 - Broken Access Control:**
 - Remote Admin Access
 - Path Based Access Control
 - Role Based Access Control
 - Broken Authentication and Session Management:**
 - Forgot Password
 - Weak Authentication Cookie
- Main Content Area:**
 - Hint:** Includes two speaker icons.
 - Form Elements:**
 - Show Params
 - Show Cookies
 - Show HTML
 - Show Java
 - Buttons: Show Lesson Plan, Logout - Closes Window
 - Text:** "Enter your name in the input field below and press 'go' to submit. The server will accept the request, reverse input, and display it back to the user, illustrating the basics of handling an http request." and "The user should become familiar with the features of the WebGoat by manipulating the above buttons to show source html, Java source code, http request parameters, and http request cookies."
 - Form:** "Enter your name:" followed by an input field and a "Go!" button.

Tools

Web Proxies



Web Application Firewalls



Vulnerability Scanners



Static Analysis



Web Proxies

The screenshot shows the 'Tamper Data - Ongoing requests' window. It features a toolbar with 'Start Tamper', 'Stop Tamper', and 'Clear' buttons, along with 'Options' and 'Help' links. A 'Filter' input field and a 'Show All' button are also present. The main area contains a table of ongoing requests:

Time	Method	Status	Content Type	URL	Load Flags
12:46:32.908	GET	302	text/html	http://www.google.com/search?hl=en&q=&btnG=Google+Search	LOAD_DOCUMENT_URI LO...
12:46:35.579	GET	200	text/html	http://www.google.com/webhp?hl=en&btnG=Google+Search	LOAD_DOCUMENT_URI LO...

Below the table are two panels showing request and response headers:

Request Header Name	Request Header Value
Host	www.google.com
User-Agent	Mozilla/5.0 (Windows; U; Windows NT 5.1; en...
Accept	text/xml,application/xml,application/xhtml+x...
Accept-Language	en-us,en;q=0.5
Accept-Encoding	gzip,deflate
Accept-Charset	ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive	300
Connection	keep-alive
Referer	http://www.google.com/
Cookie	SID=DQAAAAGcAAADHzma8UwXvT_5vhSNQ8...

Response Header Name	Response Header Value
Status	Found - 302
Location	http://www.google.com/webhp?hl=en&btnG...
Cache-Control	private
Content-Type	text/html
Server	GWS/2.1
Transfer-Encoding	chunked
Content-Encoding	gzip
Date	Tue, 06 Mar 2007 17:46:32 GMT

Altering Form Parameters

Tamper Popup

https://mailfe1.nku.edu/exchweb/bin/auth/owaauth.dll

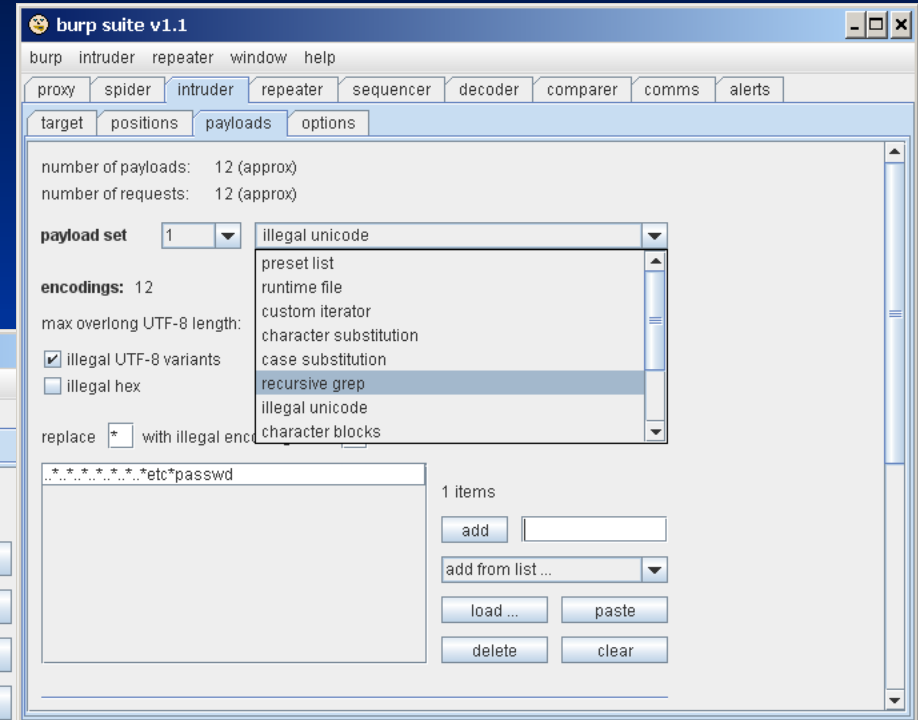
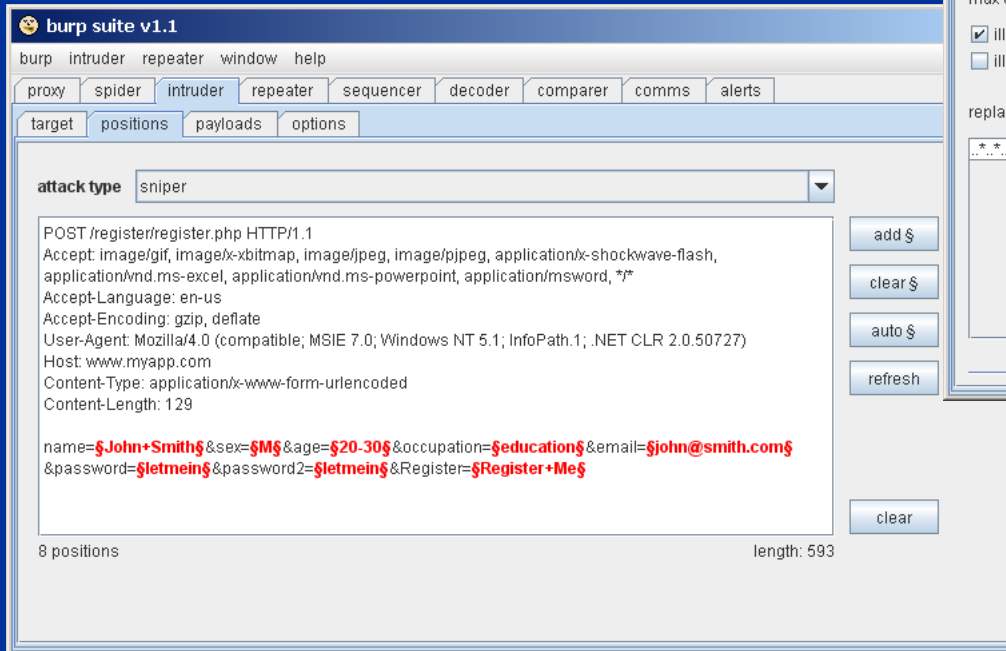
Request Header Name	Request Header ...	Post Parameter Name	Post Parameter V...
Host	mailfe1.nku.edu	destination	https%3A%2F%2F
User-Agent	Mozilla/5.0 (Windov	flags	0
Accept	text/xml,application	username	frank
Accept-Language	en-us,en;q=0.5	password	AAAAAA
Accept-Encoding	gzip,deflate	SubmitCreds	Log+On
Accept-Charset	ISO-8859-1,utf-8;c		
Keep-Alive	300		
Connection	keep-alive		
Referer	https://mailfe1.nku		
Cookie	s_pers=%20s_vsn,		

OK Cancel

Fuzz Testing

Fuzz testing consists of

- Sending unexpected input.
- Monitoring for exceptions.



Web Application Firewalls

What is a WAF?

- Web monitoring.
- Access control.
- Behind SSL endpoint.

A/K/A

- Deep packet inspection.
- Web IDS/IPS.
- Web App Proxy/Shield.

mod_security

- Open source.
- Embeds in Apache.
- Reverse proxy.

POST /action/submit.php

Headers

Host: mail.companyx.com

Referer: .{0,256}

User-Agent: .{0,256}

Postparameters

username: [0-9a-zA-Z]{4,16}

password: .{0,16}

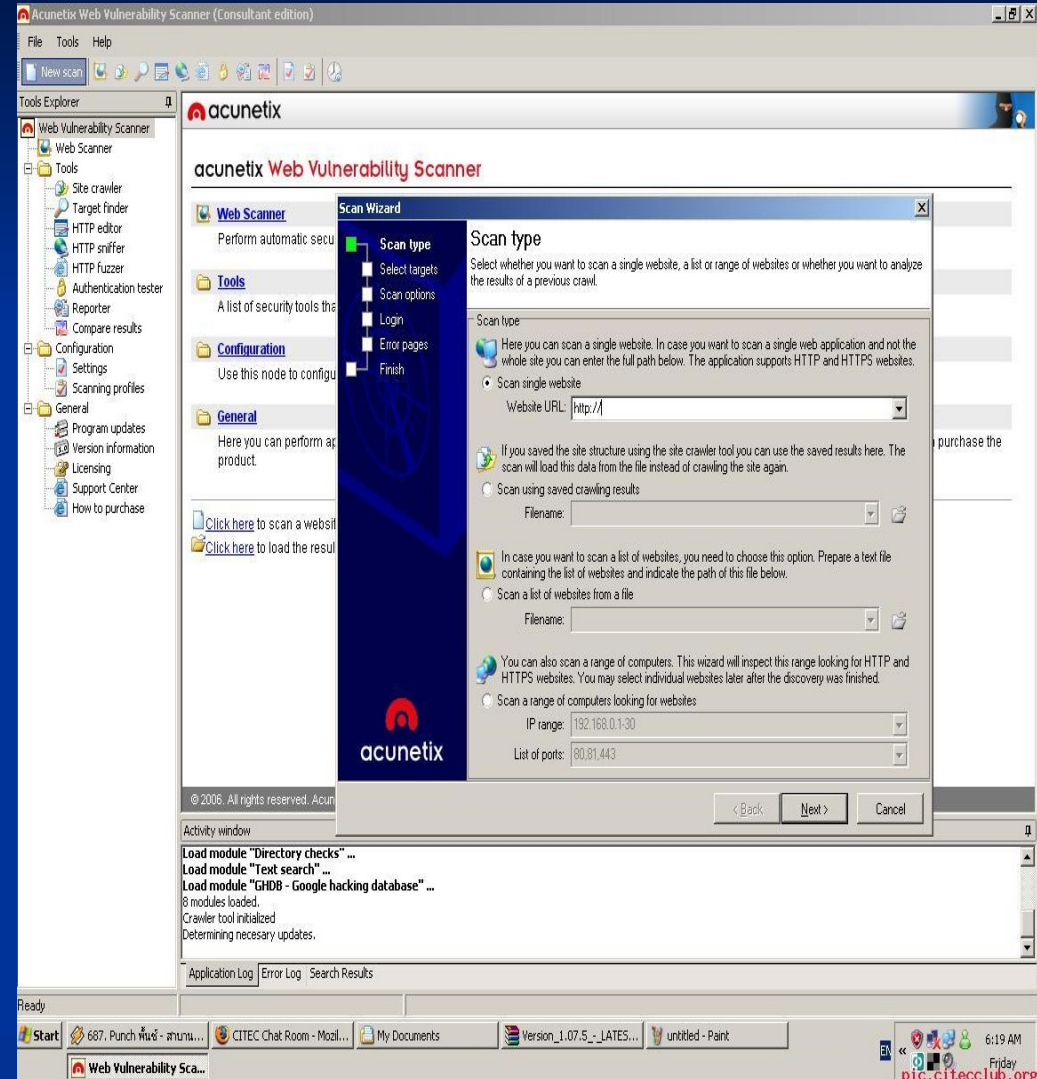
submit: login

The positive rule from Remo is translated into a whitelist ModSecurity rule. This means, that you have to define the good arguments in Remo. Requests with arguments, that do not match this positive definition, are considered bad requests (note the exclamation point in the regex below). They are dropped by ModSecurity.

REQUEST_HEADERS:Host	!^(mail.companyx.com)\$	deny
REQUEST_HEADERS:Referer	!^(.{0,256})\$	deny
REQUEST_HEADERS>User-Agent	!^(.{0,256})\$	deny
ARGS:username	!^([0-9a-zA-Z]{4,16})\$	deny
ARGS:password	!^(.{0,16})\$	deny
ARGS:submit	!^(login)\$	deny

Vulnerability Scanners

1. Spiders site.
2. Identifies inputs.
3. Sends list of malicious inputs to each input.
4. Monitors responses.

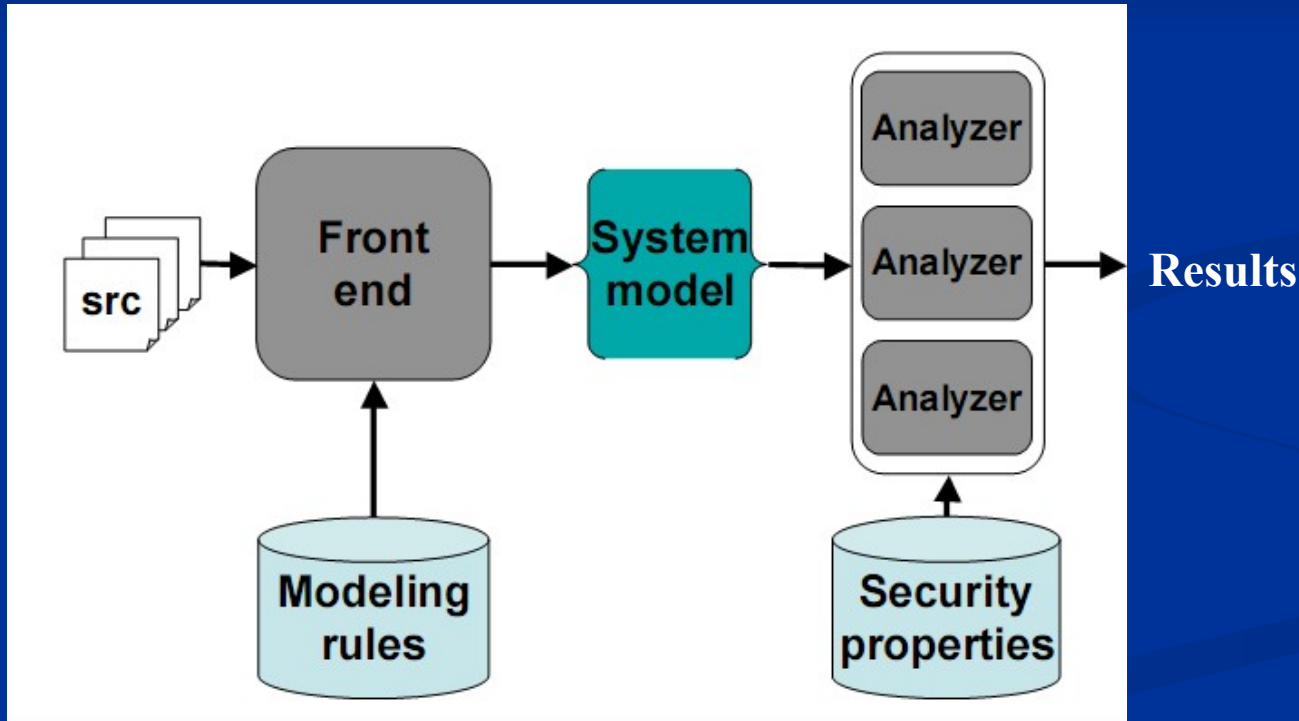


Static Analysis

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

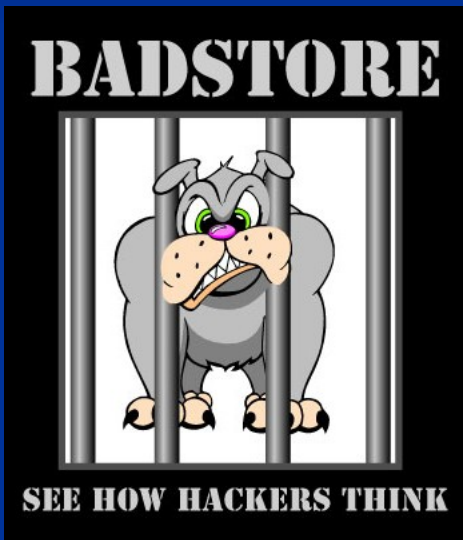
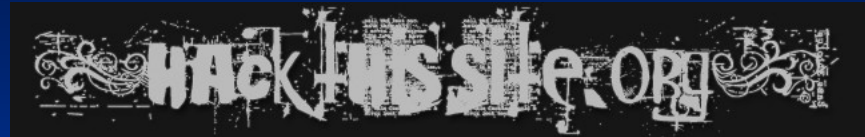
Labs

- WebGoat exercises on specific vulnerabilities.
- Using a **testing proxy** to solve more advanced WebGoat exercises.
- Assessing an application using a **web vulnerability scanner**.
- Assessing a web application using a **testing proxy**.
- Reviewing the code of an application using a **static analysis** tool.
- Deploying a **web application firewall**.
- Participating in the international CTF competition.

Approaches

1. Students evaluate and fix their own code.
 - Students learn about their own coding mistakes.
 - Scale of project limited to what students can write.
2. Students evaluate and fix your code.
 - Write a web application designed for teaching students.
3. Students evaluate and fix someone else's code.
 1. Use a web application designed for teaching.
 2. Analyze an open source web application with known vulnerabilities reported in NVD or other bug db.

Teaching Applications



Hacme Bank, Books, Casino, Travel

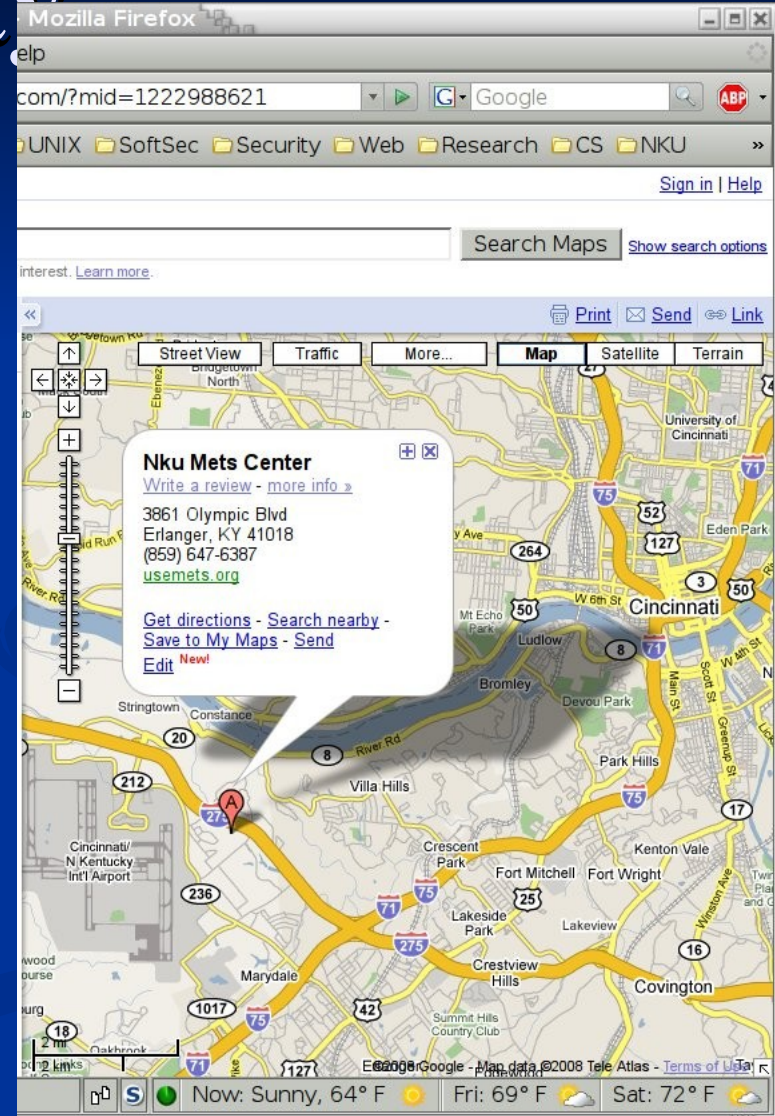
SIGITE 2008: 16-18 Oct



Future Directions: AJAX Security

Asynchronous Javascript and XML

- Expanded server side API.
- Server API calls can be issued in any order by attacker; cannot assume calls issued in order by your client.
- Larger amount of client state.
- Client/server communication using data (XML/JSON) rather than presentation (HTML.)



Future Directions: Web Sec Class

1. Web Application Input
2. Client-side Technologies
3. Service Oriented Architectures
4. AJAX
5. Input-based Attacks
6. Injection Attacks
7. Race Conditions
8. Cross-site Attacks
9. Authentication
10. Secure Programming
11. Operational Security

Conclusions

1. Defense is shifting from network to application layer. →
Firewalls, anti-virus, SSL input validation, WAF
2. Students need to learn to identify vulnerabilities.
 - Static analysis of source code.
 - Web proxies and scanners for testing.
3. Students need to learn to remediate vulnerabilities.
 1. Web application firewalls for immediate short-term fixes.
 2. Repairing source code for long term fixes.