

# Hacker

**H4CKER.ORG**

## Intense Introduction to Hacking Web Applications

Omar Santos

 @santosomar

# About // Omar Ωr Santos



Omar Santos is an active member of the security community, where he leads several industry-wide initiatives and standard bodies. His active role helps businesses, academic institutions, state and local law enforcement agencies, and other participants that are dedicated to increasing the security of the critical infrastructure.

Omar is the author of over 20 books and video courses; numerous white papers, articles, and security configuration guidelines and best practices. Omar is a Principal Engineer of Cisco's Product Security Incident Response Team (PSIRT) where he mentors and lead engineers and incident managers during the investigation and resolution of security vulnerabilities.

Omar is often presenting at many cybersecurity conferences and he is the co-lead of the DEF CON Red Team Village ([redteamvillage.io](http://redteamvillage.io)). He is also the chair of the OASIS Common Security Vulnerability Framework (CSAF) Technical Committee and the co-chair of the Forum of Incident Response and Security Teams (FIRST) PSIRT Open Source Security Working Group.

Omar has been quoted by numerous media outlets, such as TheRegister, Wired, ZDNet, ThreatPost, CyberScoop, TechCrunch, Fortune Magazine, Ars Technica, and more.

Omar's PGP Key: 0x8e19a9d13af27edc



<https://h4cker.org>



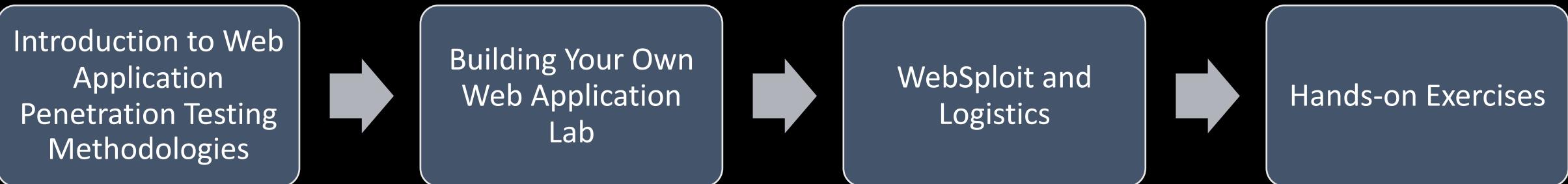
@santosomar



<https://h4cker.org/discord>

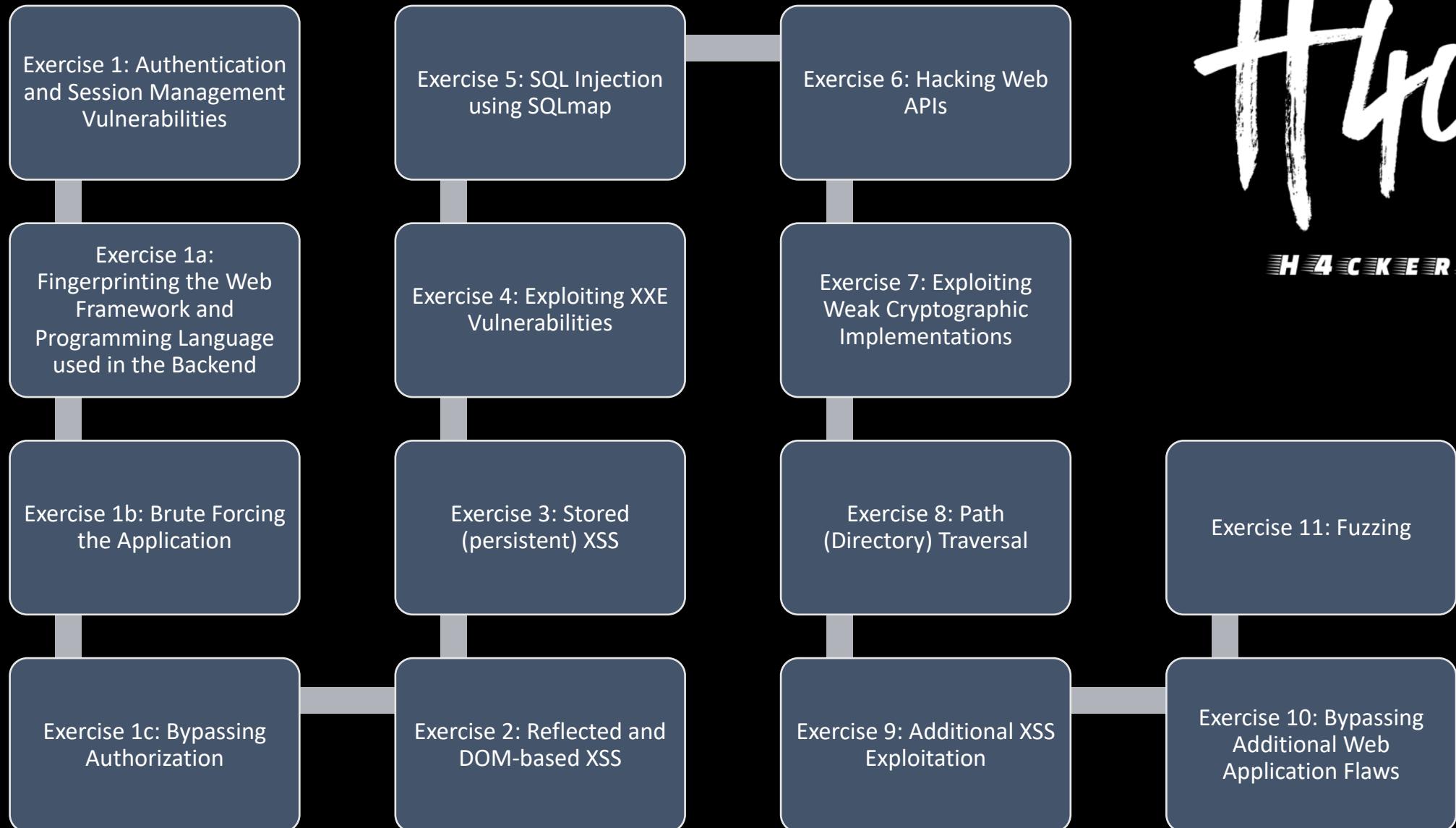


[/in/santosomar](https://in/santosomar)



# Agenda

# Exercises



# 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.



# Poll: How long have you been performing penetration testing (ethical hacking)?

- Just started
- 6 months to a year
- 1-2 years
- More than 3 years

POLL QUESTION

# Poll: Why are you interested in this class?

- Just curious about pen testing and hacking web apps
- Preparing for a certification
- My job is pen testing / ethical hacking

POLL QUESTION



Ethical Hacking  
PEN TESTING

Omar's  
HACKING BOOK

Hacking Modern<sup>®</sup>  
WEB APPS  
Resources

---

The Art of Hacking: <https://theartofhacking.org>

GitHub: <https://theartofhacking.org/github>

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



GitHub Repo with  
over 8,000 Cyber  
Security References

<http://h4cker.org/github>

livelessons™

# Hacking Web Applications (The Art of Hacking Series)

Security Penetration Testing for Today's  
DevOps and Cloud Environments

Omar Santos

video

<http://h4cker.org/webapps>



This is a hands-on class!

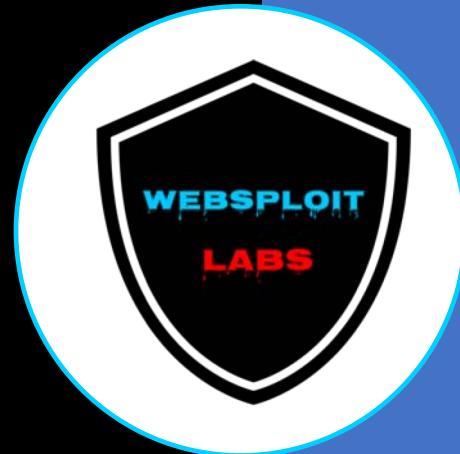
# Logistics

Setup WebSploit from  
<https://websploit.org>

This is the best option for most of you (to avoid incompatibility problems)

1. Download Kali from [kali.org](https://kali.org) (they have .OVA files for VMWare, Virtual Box, and Hyper-V).
2. Install Kali on a VM (or download their .OVA file).
3. From the new Kali VM, run the following script:

```
curl -sSL https://websploit.org/install.sh | sudo bash
```





Hacking  
Your  
Guide

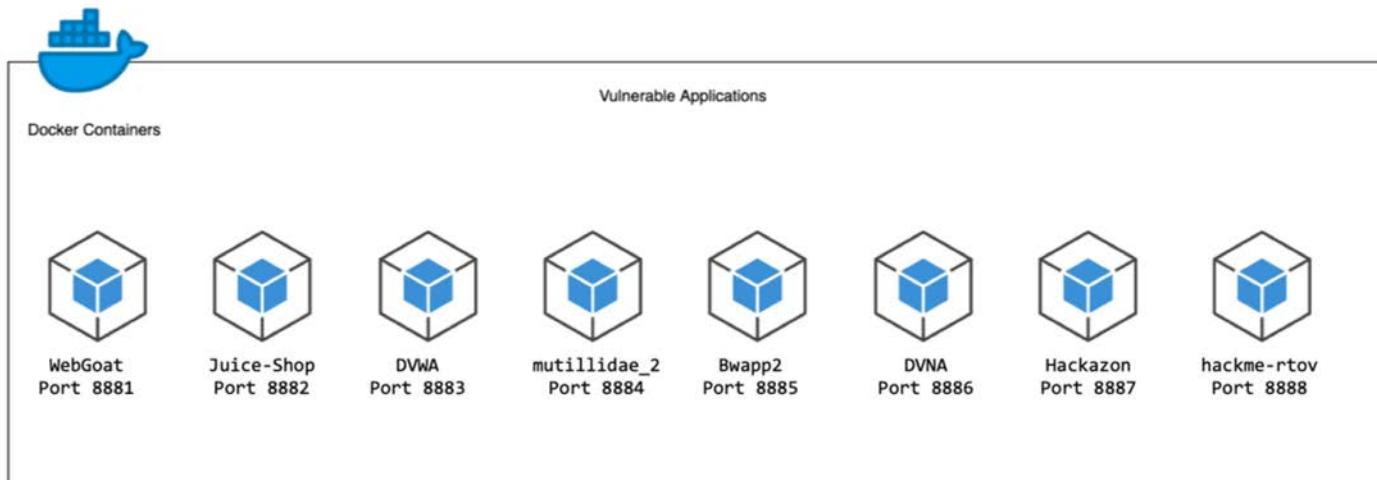
Your Lab Guide

Download it under “Resources”

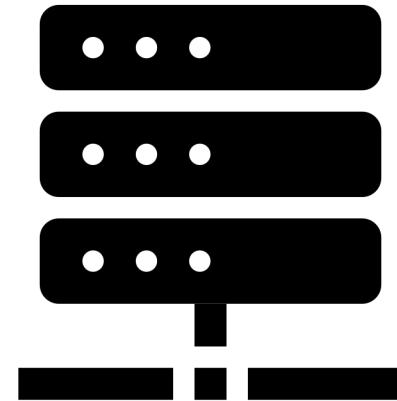
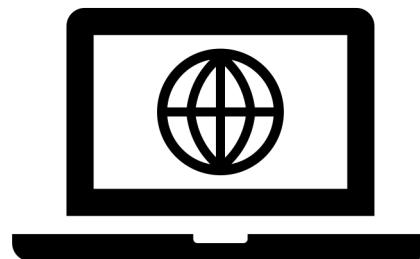
# WebSploit Components

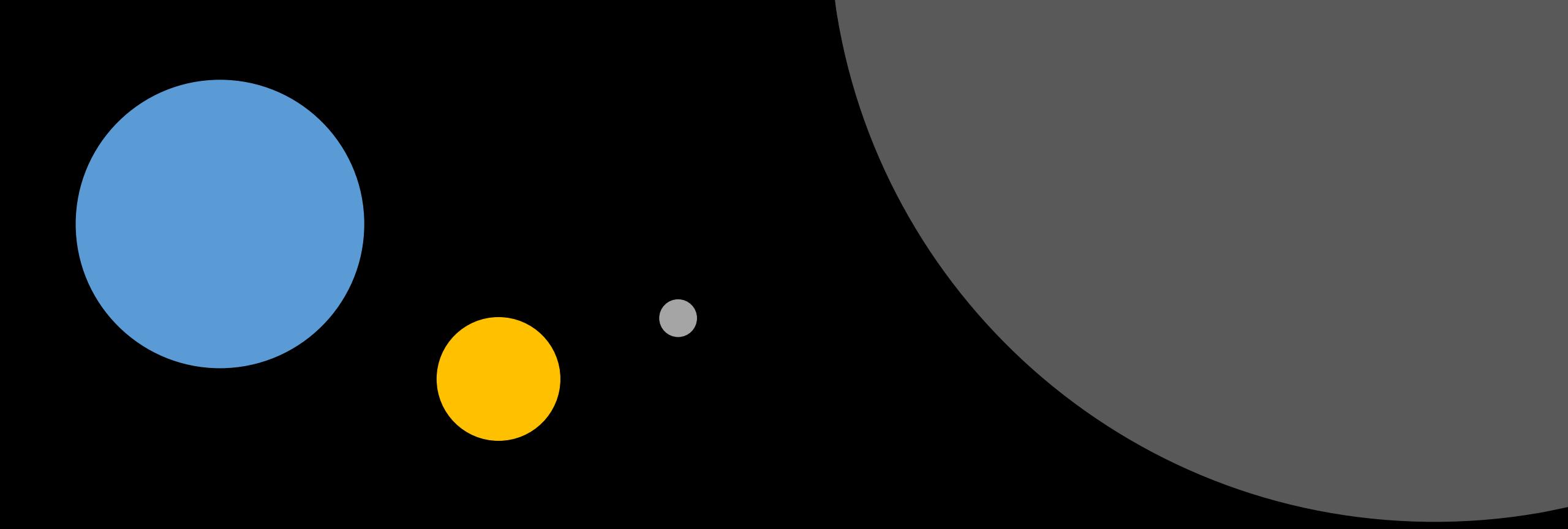


PLUS A FEW EXTRA TOOLS



# Building Your Own Lab





# Penetration Testing

Methodologies

# Become Familiar with Penetration Testing Methodologies

---

Penetration Testing Execution Standard:  
<http://www.pentest-standard.org>

---

OWASP Testing Guide:  
[https://www.owasp.org/index.php/OWASP\\_Testing\\_Project](https://www.owasp.org/index.php/OWASP_Testing_Project)

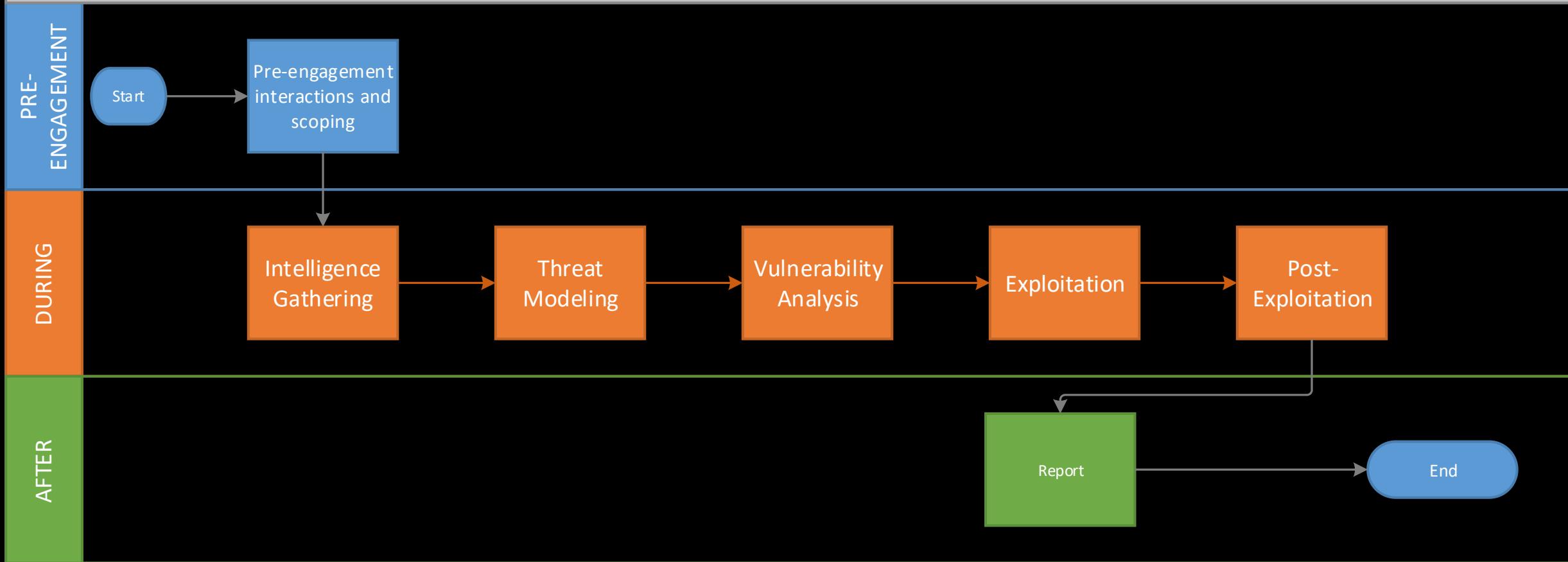
---

NIST 800-115: Technical Guide to Information Security Testing and Assessment:  
<http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-115.pdf>

---

Open Source Security Testing Methodology Manual (OSSTMM):  
<http://www.isecom.org/research>

## PEN TESTING LIFECYCLE



Aligned with: <http://www.pentest-standard.org>



Testing Guide

# )release(



Project Leaders: Matteo Meucci and Andrew Muller

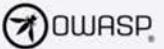
Creative Commons (CC) Attribution Share-Alike  
Free version at <http://www.owasp.org>

[https://www.owasp.org/index.php/OWASP\\_Testing\\_Project](https://www.owasp.org/index.php/OWASP_Testing_Project)

← One of the  
most comprehensive  
testing guides  
and methodologies  
for web application  
pen testing



The Art of  
Hacking



## OWASP Web Security Testing Guide

[Main](#) [Release Versions](#) [FAQ](#)[owasp](#) [flagship](#) [Stars on GitHub](#) 1.5k [Follow @owasp\\_wstg](#) 672

The Web Security Testing Guide (WSTG) Project produces the premier cybersecurity testing resource for web application developers and security professionals.

The WSTG is a comprehensive guide to testing the security of web applications and web services. Created by the collaborative efforts of cybersecurity professionals and dedicated volunteers, the WSTG provides a framework of best practices used by penetration testers and organizations all over the world.

### Contributions

Any contributions to the guide itself should be made via the [guide's project repo](#).

### Stable

View the always-current stable version at [stable](#).

### Latest

We are currently developing release version 5.0.

You can [read the latest development documents in our official GitHub repository](#) or view the bleeding-edge content at [latest](#).

### Versioned Releases

Only [v4.1](#) is currently available as a web-hosted release. Previous releases are available as PDFs on the [Release Versions](#) tab.

### How To Reference WSTG Scenarios

Each scenario has an identifier in the format WSTG-<category>-<number>, where: 'category' is a 4 character upper case string that identifies the type of test or weakness, and 'number' is a zero-padded numeric value from 01 to 99. For example: WSTG-INFO-02 is the second Information Gathering test.

The identifiers may change between versions therefore it is preferable that other documents, reports, or tools use the format: WSTG-<version>-<category>-<number>, where: 'version' is the version tag with punctuation removed. For example: WSTG-v41-INFO-02 would be understood to mean specifically the second Information Gathering test from version 4.1.

[Watch](#) 117 [Star](#) 1,518

The OWASP® Foundation works to improve the security of software through its community-led open source software projects, hundreds of chapters worldwide, tens of thousands of members, and by hosting local and global conferences.

### Project Classification

- Flagship Project
- Documentation
- Breaker
- Builder

### Project Links

[Latest](#)  
[GitHub Repository](#)

### Getting Involved

[Code of Conduct](#)  
[Contribution Guide](#)

### Social

[Join OWASP Slack](#)  
Channel: #testing-guide  
[Twitter: @owasp\\_wstg](#)

### License

CC BY-SA 4.0

### Leaders

[Elie Saad](#)  
[Rick Mitchell](#)  
[Matteo Meucci](#)

# A Balanced Approach :

- Pen Testing
- Source code review
- Threat Modeling

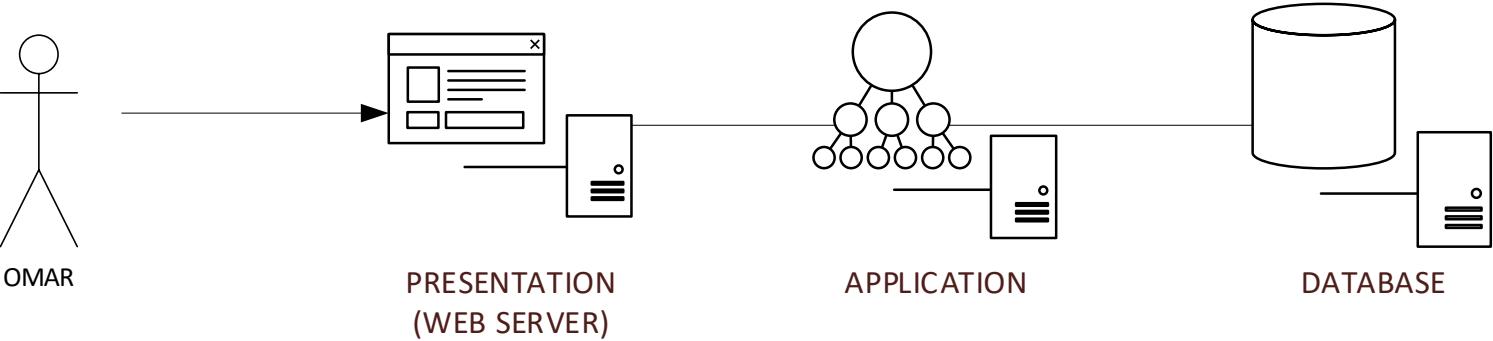
(SDLC)

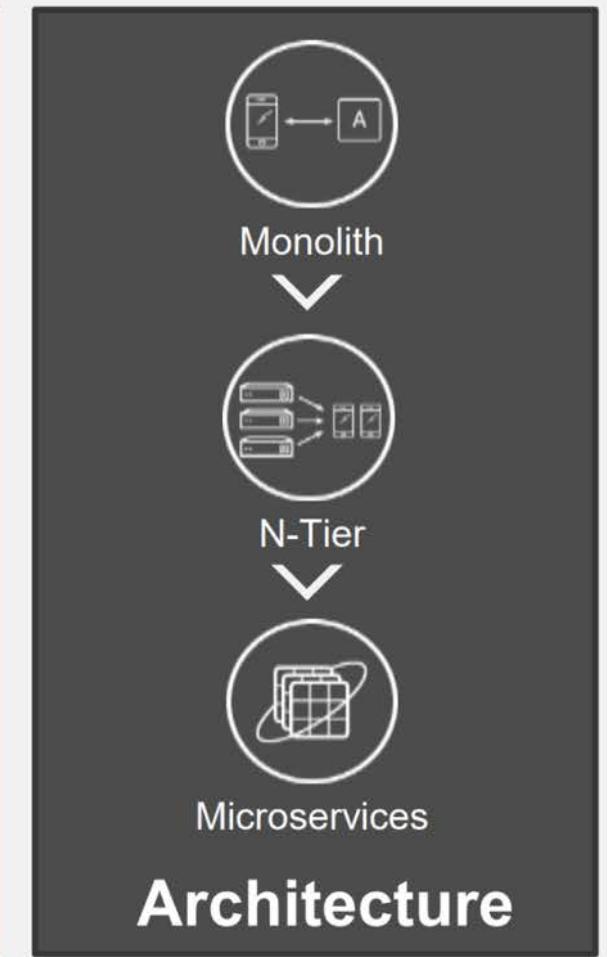
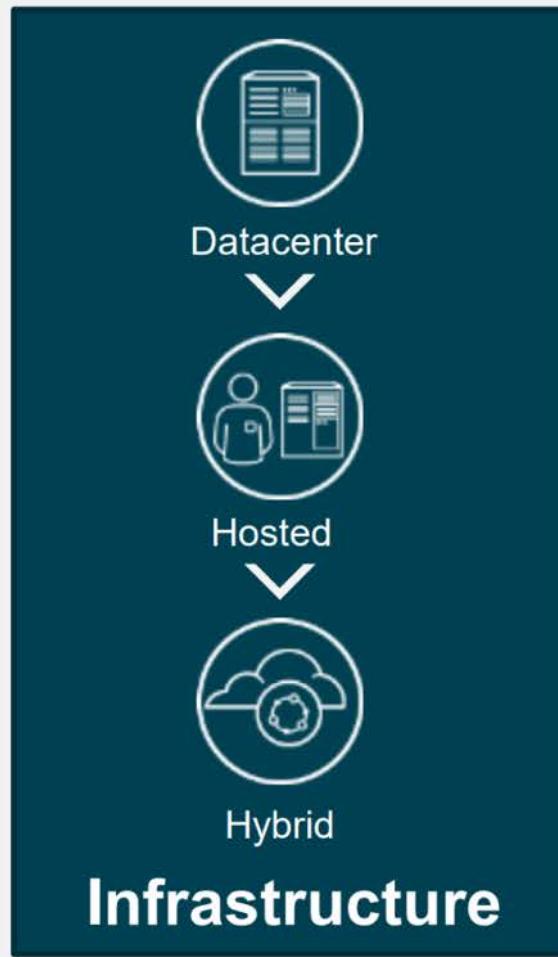
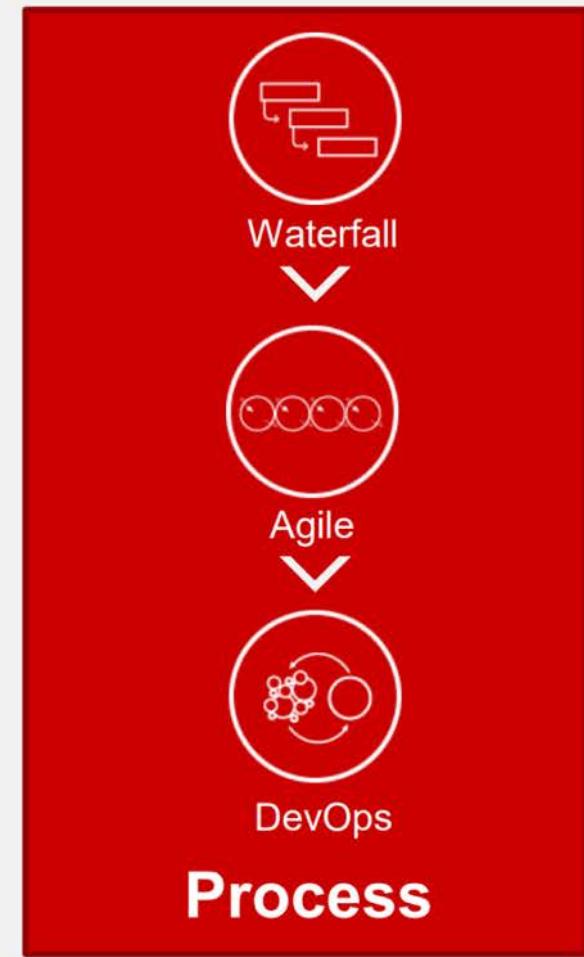
# Exploring How Web Applications Have Evolved Over Time

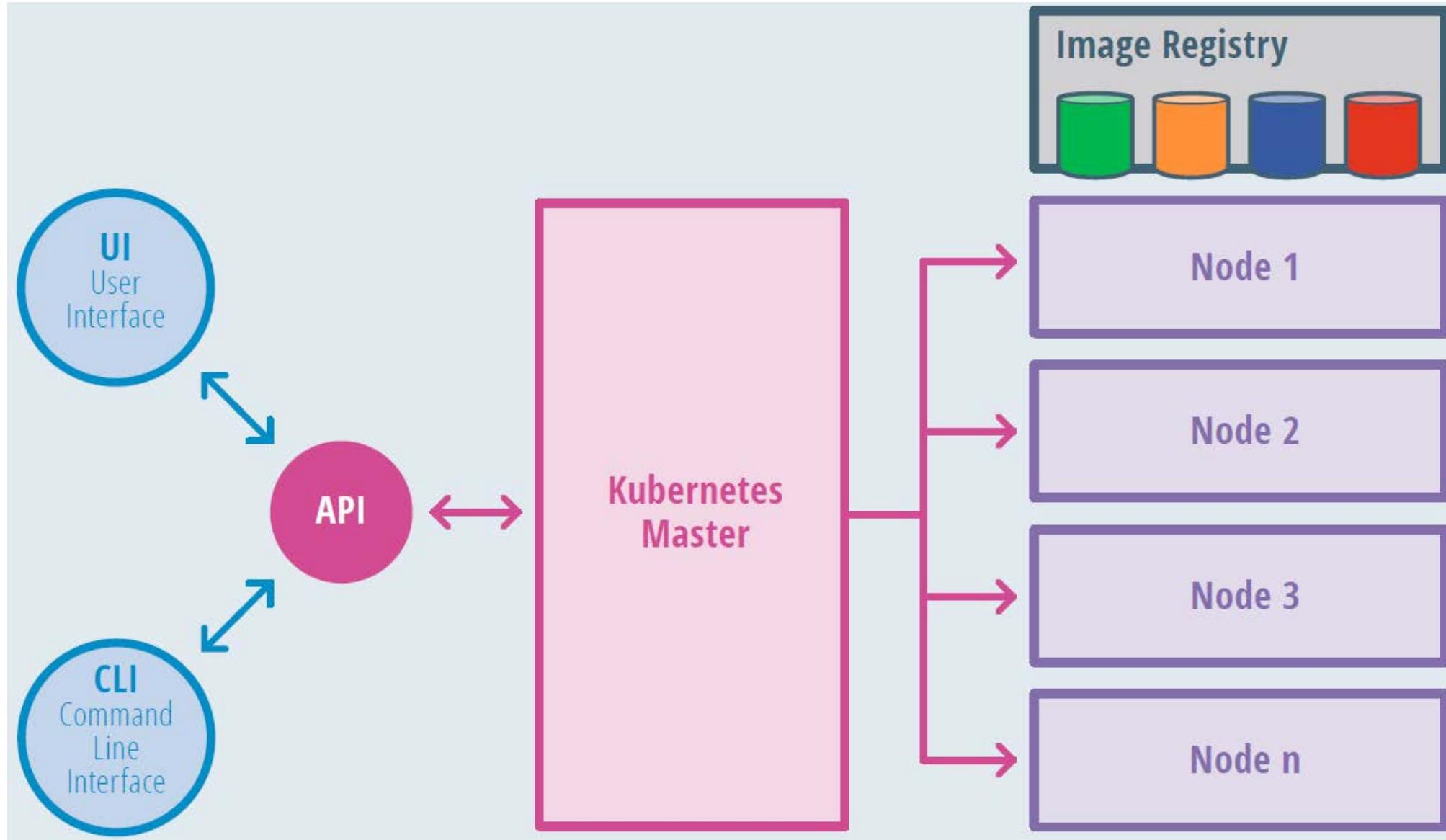
# What is a Web Application?

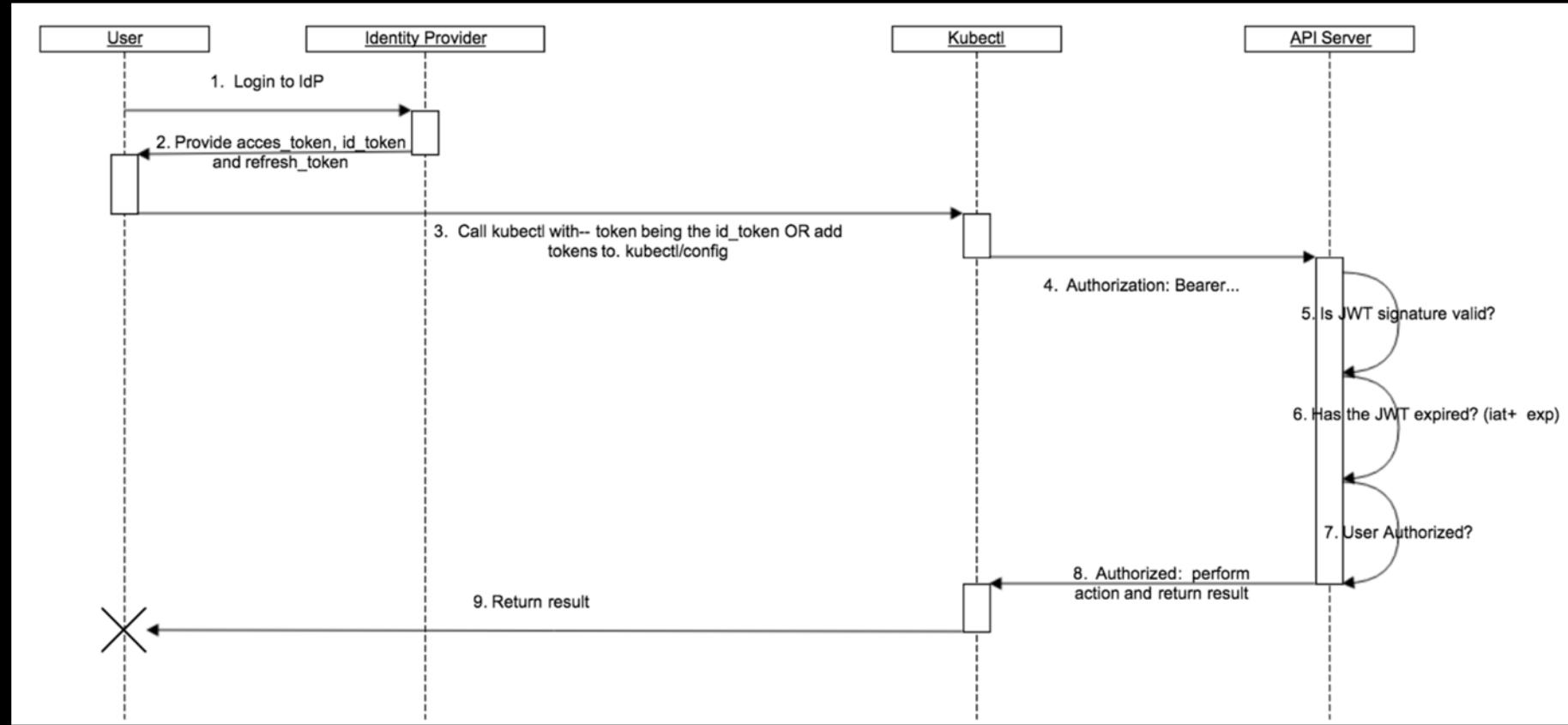


# THE TRADITIONAL WEB APPLICATION ARCHITECTURE









# OPENID AUTHENTICATION FLOW

```
{  
  "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"  
  }  
}
```

# WEBHOOK MODE EXAMPLE REQUEST

# Hacker

H 4 C K E R . O R G

## Understand

Understand how the application works

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

## Gather

Gather needed information

- credentials, tokens, API configurations

## Break

Break the application to manageable chunks

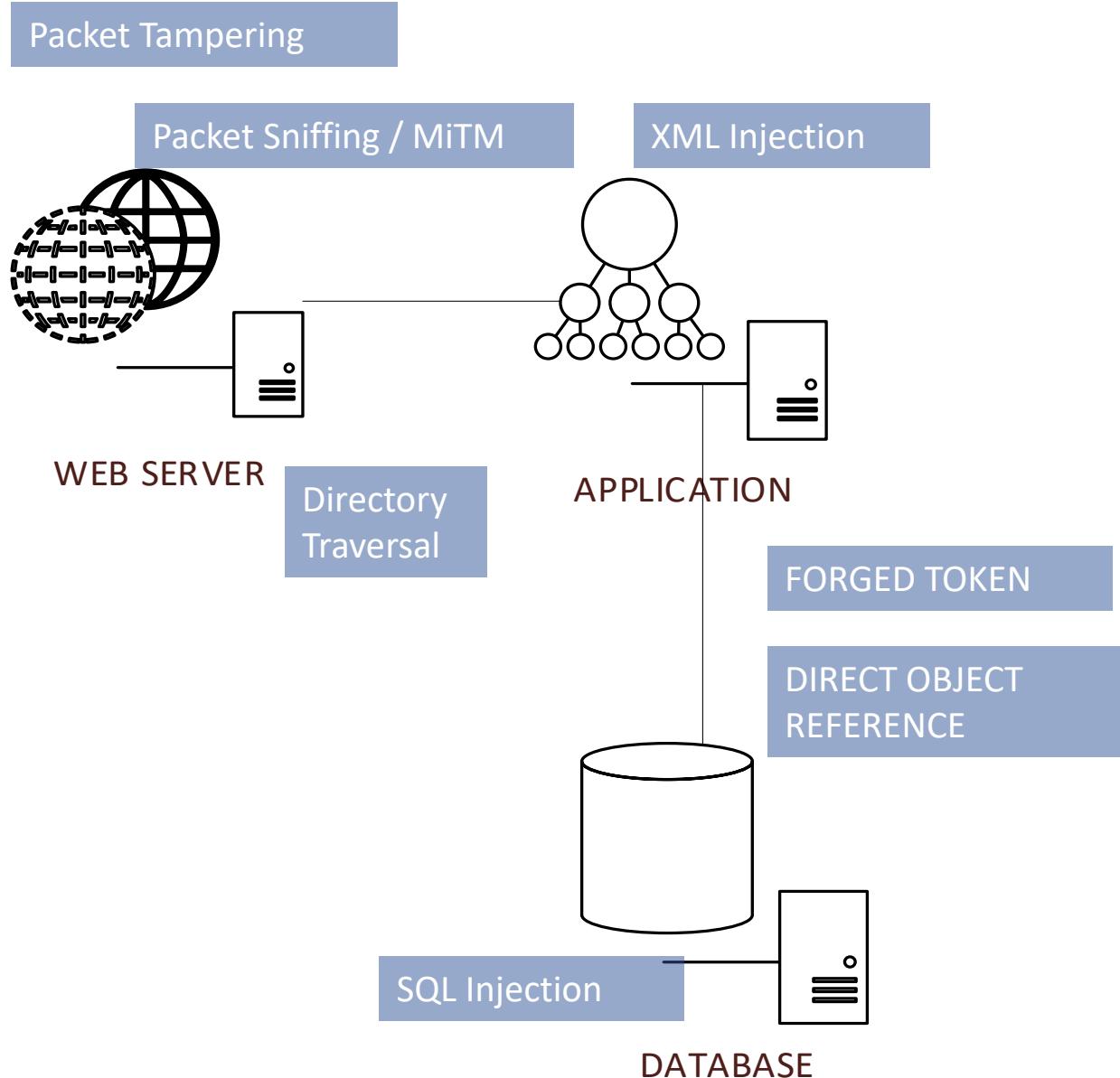
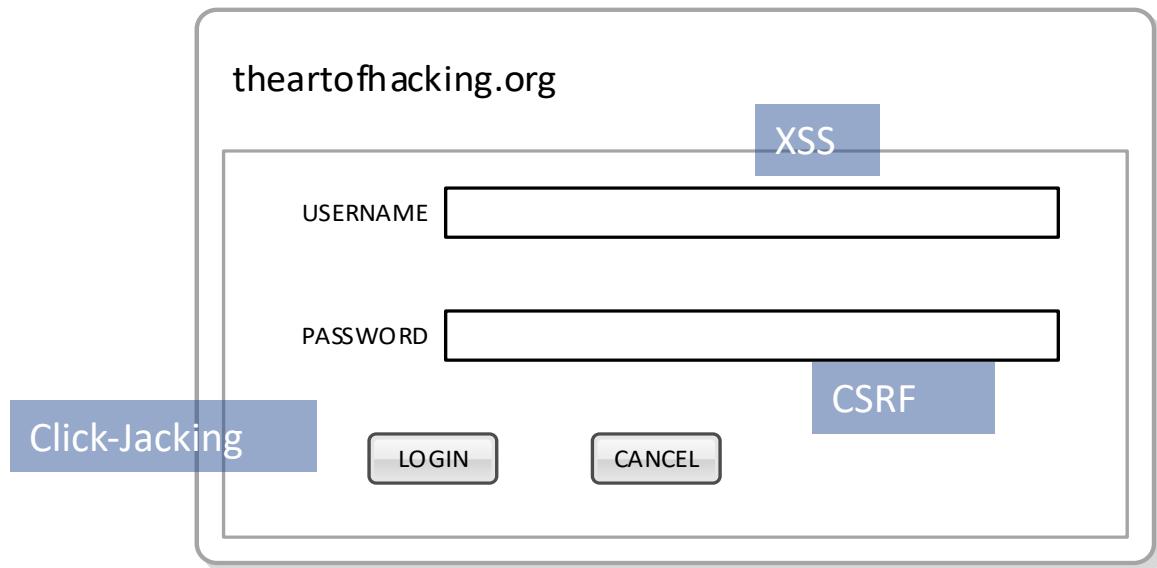
- Large app scans can take hours
- Functional breakout (admin, reporting)

## Allot

Allot time to reset the application & purge backend database

- Scans inject bad data
- Can use VM snapshots





# Machine-to-machine



OAuth 2

OpenID Connect

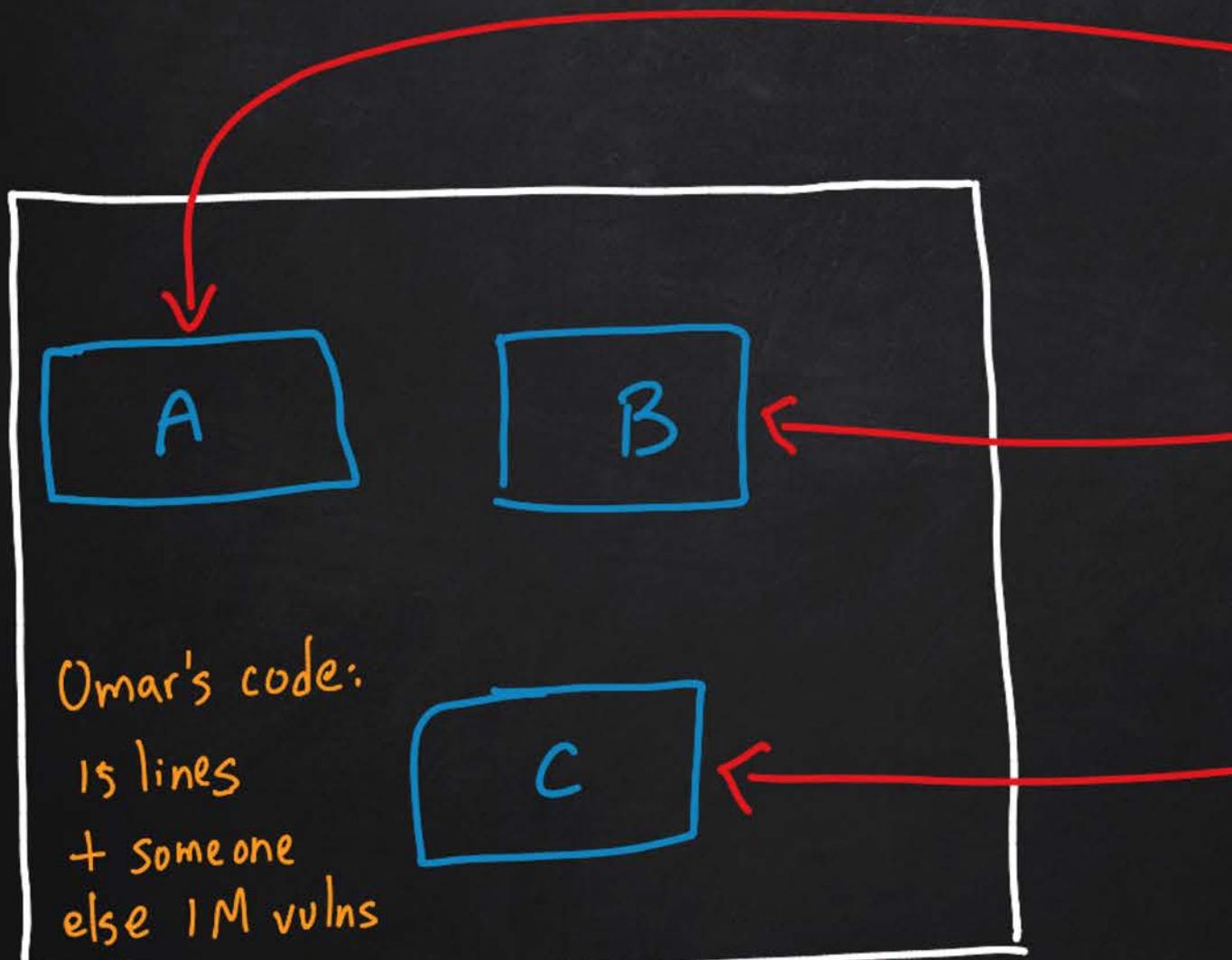
SAML

... more

## Machine -to- machine



- NO GUIs
- useful docs:
  - Swagger / Open API
  - GraphQL docs
  - WSDL / WADL docs



Omar's code:  
15 lines  
+ someone  
else 1M vulns

OMAR's Vulnerable Application

Docker Hub

image\_1

- alpine - lib1...
- nginx - lib20

Ron's lib\_55

Private Registry

- image - 95

- Mongo - lib - 78

..

[README.md](#)

Adding websploit

34 minutes ago

[README.md](#)

## Vulnerable Apps, Servers, and Websites

The following is a collection of vulnerable servers (VMs) or websites that you can use to practice your skills (sorted alphabetically).

- bWAPP : <https://sourceforge.net/projects/bwapp/files/bWAPP>
- Damn Vulnerable ARM Router (DVAR): <http://blog.exploitlab.net/2018/01/dvar-damn-vulnerable-arm-router.html>
- Damn Vulnerable iOS Application (DVIA): <http://damnvulnerableiosapp.com>
- Damn Vulnerable Linux (DVL):  
[https://osdn.net/projects/sfnet\\_virtualhacking/downloads/os/dvl/DVL\\_1.5\\_Infectious\\_Disease.iso](https://osdn.net/projects/sfnet_virtualhacking/downloads/os/dvl/DVL_1.5_Infectious_Disease.iso)
- Damn Vulnerable Web App (DVWA): <https://github.com/ethicalhack3r/DVWA>
- DOMXSS: <http://www.domxss.com/domxss/>
- Game of Hacks: <http://www.gameofhacks.com>
- Gruyere: <https://google-gruyere.appspot.com>
- Hack This Site: <https://www.hackthissite.org>
- Hack This: <https://www.hackthis.co.uk>
- Hackazon : <https://github.com/rapid7/hackazon>
- HellBound Hackers: <https://www.hellboundhackers.org>
- Metasploitable2 : <https://community.rapid7.com/docs/DOC-1875>
- Metasploitable3 : <https://blog.rapid7.com/2016/11/15/test-your-might-with-the-shiny-new-metasploitable3/>
- Over The Wire Wargames: <http://overthewire.org/wargames>
- OWASP Juice Shop : [https://www.owasp.org/index.php/OWASP\\_Juice\\_Shop\\_Project](https://www.owasp.org/index.php/OWASP_Juice_Shop_Project)
- OWASP Mutillidae II: <https://sourceforge.net/projects/mutillidae>
- Peruggia: <https://sourceforge.net/projects/peruggia>
- RootMe: <https://www.root-me.org>
- Samurai Web Testing Framework: <http://www.samurai-wtf.org/>
- Try2Hack: <http://www.try2hack.nl>
- Vicnum: <http://vicnum.ciphertechs.com>
- VulnHub:<https://www.vulnhub.com>
- Web Security Dojo: <https://www.mavensecurity.com/resources/web-security-dojo>
- WebSploit (maintained by Omar Santos): <https://websploit.h4cker.org>
- WebGoat: <https://github.com/WebGoat/WebGoat>

Dozens of vulnerable applications, VMs,  
and websites that you can use to  
practice your skills.



- FTP Injection**
- Advisory: Java/Python FTP Injections Allow for Firewall Bypass - Written by [Timothy Morgan](#).
  - SMTP over XXE - how to send emails using Java's XML parser - Written by [Alexander Klink](#).
- XXE - XML eXternal Entity**
- XXE - Written by [@phonexicum](#).
- CSRF - Cross-Site Request Forgery**
- Wiping Out CSRF - Written by [@jrozner](#).
- SSRF - Server-Side Request Forgery**
- SSRF bible, Cheatsheet - Written by [@Wallarm](#).
- Rails**
- Rails Security - First part - Written by [@qazbnm456](#).
- AngularJS**
- XSS without HTML: Client-Side Template Injection with AngularJS - Written by [Gareth Heyes](#).
  - DOM based Angular sandbox escapes - Written by [@garethheyes](#)
- SSL/TLS**
- SSL & TLS Penetration Testing - Written by [APTIVE](#).
- Webmail**
- NFS**
- NFS | PENETRATION TESTING ACADEMY - Written by [PENETRATION ACADEMY](#).
- Fingerprint**
- Sub Domain Enumeration**
- A penetration tester's guide to sub-domain enumeration - Written by [Bharath](#).
  - The Art of Subdomain Enumeration - Written by [Patrik Hudak](#).
- Crypto**
- Applied Crypto Hardening - Written by [The bettercrypto.org Team](#).
- Web Shell**
- Hunting for Web Shells - Written by [Jacob Baines](#).
  - Hacking with JSP Shells - Written by [@.nullbind](#).

## A Few Popular Tools

The following are a few popular tools that you learned in the video courses part of these series:

- Burp Suite
- OWASP Zed Attack Proxy (ZAP)
- sqlmap
- htttrack
- skipfish

## How to Integrate OWASP ZAP with Jenkins

You can integrate ZAP with Jenkins and even automatically create Jira issues based on your findings. You can download the ZAP plug in [here](#).

[This video](#) provides an overview of how to integrate

## Javascript Tools

- Retire.js

## Popular Commercial Tools

- Qualys Web Scanning
- IBM Security AppScan

## XSS - Cross-Site Scripting

- Cross-Site Scripting – Application Security – Google - Introduction to XSS by Google.
- HSSC - HTML5 Security Cheatsheet - Collection of HTML5 related XSS attack vectors by [@cure53](#).
- XSS.png - XSS mind map by [@ackmasa](#).
- CXSS Guide - Comprehensive tutorial on cross-site scripting by [@JakobKallin](#) and [Irene Lobo Valbuena](#).

## CSV Injection

- CSV Injection -> Meterpreter on Pornhub - Written by [Andy](#).
- The Absurdly Underestimated Dangers of CSV Injection - Written by [George Mauer](#).

## SQL Injection

- SQL Injection Cheat Sheet - Written by [@netsparker](#).
- SQL Injection Wiki - Written by [NETSPI](#).
- SQL Injection Pocket Reference - Written by [@LightOS](#).

<https://theartofhacking.org/github>

# Vulnerability Discovery



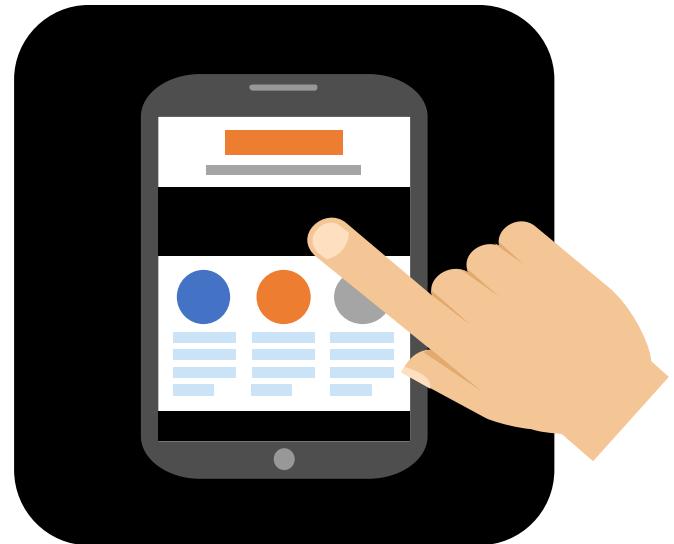
## Manual Testing

Try different input and see how the application reacts



## Automated scanning

Use a web scanning application to probe for vulns



## Hybrid

Use an automated scanner as well as manual testing

# Automated Scanning



## netsparker

### AppScan

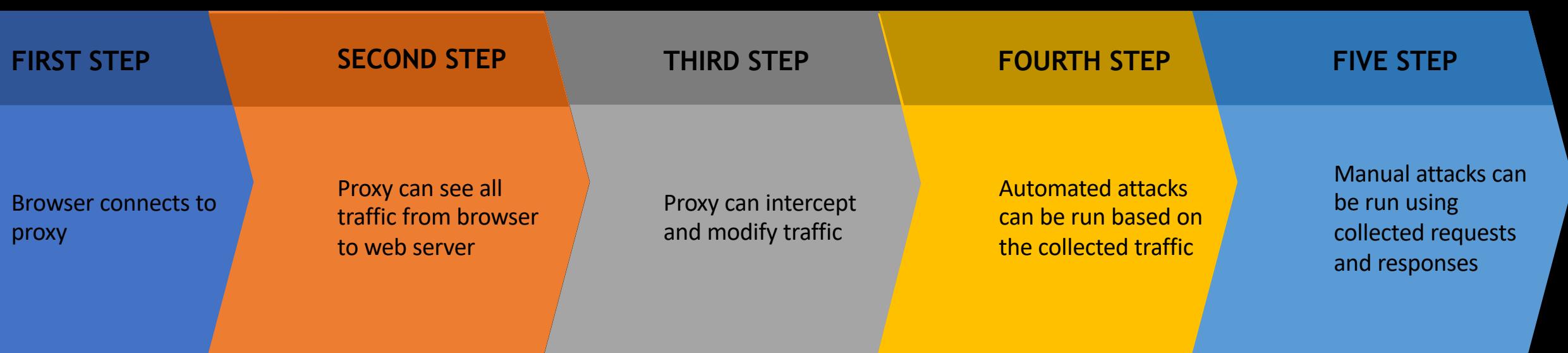
IBM Security



**HP WebInspect™**

BURPSUITE  
PROFESSIONAL

# How an Interception Proxy Works





# Authentication and Session Management Vulnerabilities

# Auth & Session Management

- credential brute force and stealing
- session hijacking
- Redirects
- Default creds & weak creds
- Kerberos
- OAuth2, SAML, OpenID weak implementations

# Session IDs

- Predictables
- weak PRNGs
- cookies
  - session (non-persistent)
  - local (persistent)
    - "Max-Age" or "Expires"

There are several ways that an attacker can perform a session hijack and where a session token could be compromised:

---

Predicting session tokens: this is why it is important to use non-predictable tokens, as previously discussed in this section.

---

Session sniffing: by collecting packets of unencrypted web sessions.

---

Man-in-the-middle attacks: where the attacker sits in the path between the client and the web server.

---

Man-in-the-browser attack: similar approach as man-in-the-middle attacks; however, in this case browser (or extension or plugin) is compromised and used to intercept and manipulate web sessions between the user and the web server.

# Exercise 1: Authentication and Session Management Vulnerabilities

## Exercise 1a

Fingerprinting the Web Framework and Programming Language used in the Backend

## Exercise 1b

Brute Forcing the Application

## Exercise 1c

Bypassing Authorization

# Cross-site Scripting (XSS) & Cross-site Request Forgery (CSRF)

## Types of XSS:

- Reflected (not persistent)
- Stored (persistent)
- DOM-based

XSS Exploitation could lead to installation / exec of malicious code , account compromise , session cookie hijacking , site redirection & more

2



You typically find  
XSS vulnerabilities  
in...

- 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 displays user-supplied data

Example of XSS test in web browser's address bar: ↴

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

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

XSS test in web form

Amazing  
Resources

## How to Avoid Cross-site scripting Vulnerabilities

See the [XSS \(Cross Site Scripting\) Prevention Cheat Sheet](#)

See the [DOM based XSS Prevention Cheat Sheet](#)

See the [OWASP Development Guide article on Phishing.](#)

See the [OWASP Development Guide article on Data Validation.](#)

## How to Review Code for Cross-site scripting Vulnerabilities

See the [OWASP Code Review Guide article on Reviewing Code for Cross-site scripting Vulnerabilities.](#)

## How to Test for Cross-site scripting Vulnerabilities

See the latest [OWASP Testing Guide article on how to test for the various kinds of XSS vulnerabilities.](#)

- [Testing\\_for\\_Reflected\\_Cross\\_site\\_scripting\\_\(OWASP-DV-001\)](#)
- [Testing\\_for\\_Stored\\_Cross\\_site\\_scripting\\_\(OWASP-DV-002\)](#)
- [Testing\\_for\\_DOM-based\\_Cross\\_site\\_scripting\\_\(OWASP-DV-003\)](#)



Evasions

Reflected  
XSS

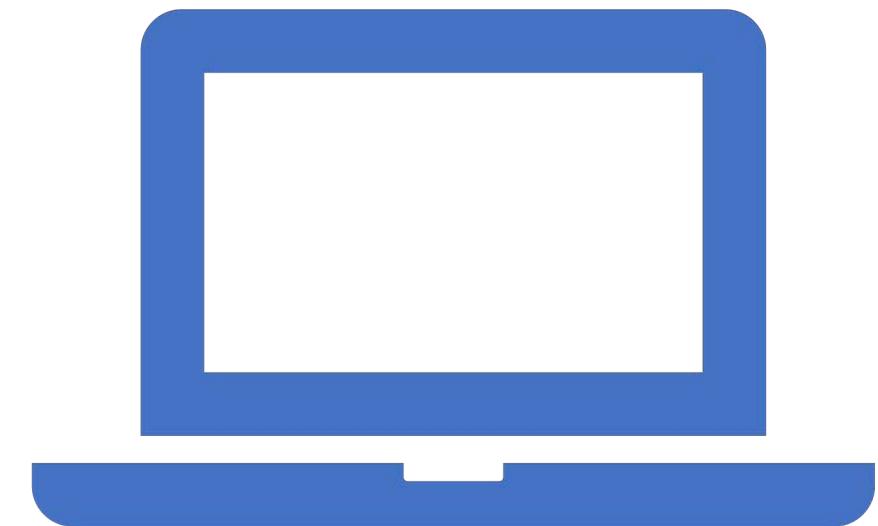
DOM-  
based XSS



Exercise 2: Reflected XSS

Stored (persistent) XSS

## Exercise 3



# XML External Entity (XXE)

Poorly configured XML parsers evaluate external entity references within XML docs.

External entities can be used to:

- \* disclose internal files using the file URI handler
- \* internal file shares
- \* internal port scanning
- \* remote code execution
- \* denial of service attacks.

Exploiting XXE Vulnerabilities  
Exercise 4



# Cross-site request forgery

- Cross-site request forgery (CSRF or XSRF) attacks occur when unauthorized commands are transmitted from a user that is trusted by the application.
- CSRF is different from XSS because it exploits the trust that an application has in a user's browser.



## **How to Review Code for CSRF Vulnerabilities**

See the [OWASP Code Review Guide](#) article on how to [review code for CSRF vulnerabilities](#).

## **How to Test for CSRF Vulnerabilities**

See the [OWASP Testing Guide](#) article on how to [test for CSRF vulnerabilities](#).

## **How to Prevent CSRF Vulnerabilities**

See the [CSRF Prevention Cheat Sheet](#) for prevention measures.

Listen to the [OWASP Top Ten CSRF Podcast](#).

Most frameworks have built-in CSRF support such as [Joomla](#), [Spring](#), [Struts](#), [Ruby on Rails](#), [.NET](#) and others.

Use [OWASP CSRF Guard](#) to add CSRF protection to your Java applications. You can use [CSRFProtector Project](#) to protect your php applications or any project deployed using Apache Server. There is a [.Net CSRF Guard](#) at OWASP as well, but it's old and doesn't look complete.

John Melton also has an [excellent blog post](#) describing how to use the native anti-CSRF functionality of the [OWASP ESAPI](#).

[https://www.owasp.org/index.php/Cross-Site Request Forgery \(CSRF\)](https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF))



# Introduction to Hacking Databases

UserName  
= Santos';drop table  
users; truncate  
audit\_log;--

```

graph LR
    TA[Threat Agents] --> AV[Attack Vectors]
    AV --> SW[Security Weakness]
    SW --> I[Impacts]
  
```

Threat Agents	Attack Vectors	Security Weakness	Impacts		
App. Specific	Exploitability: 3	Prevalence: 2	Detectability: 3	Technical: 3	Business ?
Almost any source of data can be an injection vector, environment variables, parameters, external and internal web services, and all types of users. <a href="#">Injection flaws</a> occur when an attacker can send hostile data to an interpreter.		Injection flaws are very prevalent, particularly in legacy code. Injection vulnerabilities are often found in SQL, LDAP, XPath, or NoSQL queries, OS commands, XML parsers, SMTP headers, expression languages, and ORM queries.	Injection flaws are easy to discover when examining code. Scanners and fuzzers can help attackers find injection flaws.	Injection can result in data loss, corruption, or disclosure to unauthorized parties, loss of accountability, or denial of access. Injection can sometimes lead to complete host takeover.	The business impact depends on the needs of the application and data.

Source: OWASP

[https://www.owasp.org/index.php/Injection Prevention Cheat Sheet](https://www.owasp.org/index.php/Injection_Prevention_Cheat_Sheet)

[https://www.owasp.org/index.php/SQL Injection Prevention Cheat Sheet](https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet)



## Exercise 5

### SQL Injection

# Hacking Mobile Devices

Hacker

**HACKER.ORG**



What you  
need to  
know...

Mobile platforms  
attack vectors

Android threats  
and attacks

iOS threats and  
attacks

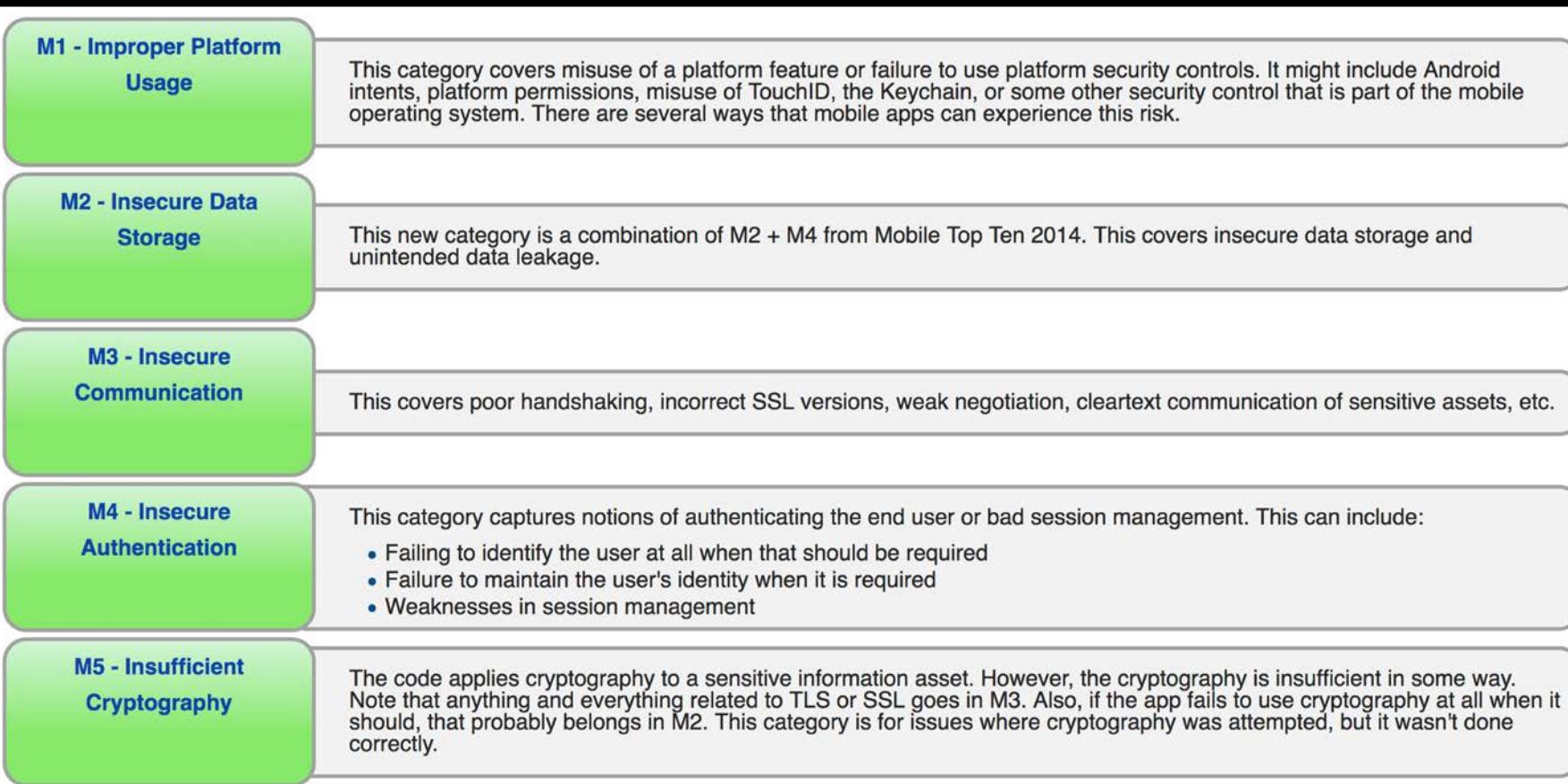
Mobile spyware

Mobile Device  
Management  
(MDM)

Mobile security  
guidelines and  
security tools

Overview of  
mobile penetration  
testing

# OWASP Mobile Security Top 10



[https://www.owasp.org/index.php/Mobile\\_Top\\_10\\_2016-Top\\_10](https://www.owasp.org/index.php/Mobile_Top_10_2016-Top_10)

# OWASP Mobile Security Top 10

M6 - Insecure Authorization	This is a category to capture any failures in authorization (e.g., authorization decisions in the client side, forced browsing, etc.). It is distinct from authentication issues (e.g., device enrolment, user identification, etc.). If the app does not authenticate users at all in a situation where it should (e.g., granting anonymous access to some resource or service when authenticated and authorized access is required), then that is an authentication failure not an authorization failure.
M7 - Client Code Quality	This was the "Security Decisions Via Untrusted Inputs", one of our lesser-used categories. This would be the catch-all for code-level implementation problems in the mobile client. That's distinct from server-side coding mistakes. This would capture things like buffer overflows, format string vulnerabilities, and various other code-level mistakes where the solution is to rewrite some code that's running on the mobile device.
M8 - Code Tampering	This category covers binary patching, local resource modification, method hooking, method swizzling, and dynamic memory modification. Once the application is delivered to the mobile device, the code and data resources are resident there. An attacker can either directly modify the code, change the contents of memory dynamically, change or replace the system APIs that the application uses, or modify the application's data and resources. This can provide the attacker a direct method of subverting the intended use of the software for personal or monetary gain.
M9 - Reverse Engineering	This category includes analysis of the final core binary to determine its source code, libraries, algorithms, and other assets. Software such as IDA Pro, Hopper, otool, and other binary inspection tools give the attacker insight into the inner workings of the application. This may be used to exploit other nascent vulnerabilities in the application, as well as revealing information about back end servers, cryptographic constants and ciphers, and intellectual property.
M10 - Extraneous Functionality	Often, developers include hidden backdoor functionality or other internal development security controls that are not intended to be released into a production environment. For example, a developer may accidentally include a password as a comment in a hybrid app. Another example includes disabling of 2-factor authentication during testing.

[https://www.owasp.org/index.php/Mobile\\_Top\\_10\\_2016-Top\\_10](https://www.owasp.org/index.php/Mobile_Top_10_2016-Top_10)

Other  
Client-based  
Attacks Still  
Apply

---

Phishing

---

Framing (iFrame-based attacks)

---

Clickjacking

---

Man-in-the-mobile / Man-in-the-browser

---

Buffer overflows

---

Data Caching

---

XSS / CSRF

---

Brute-force

---

Weak input validation

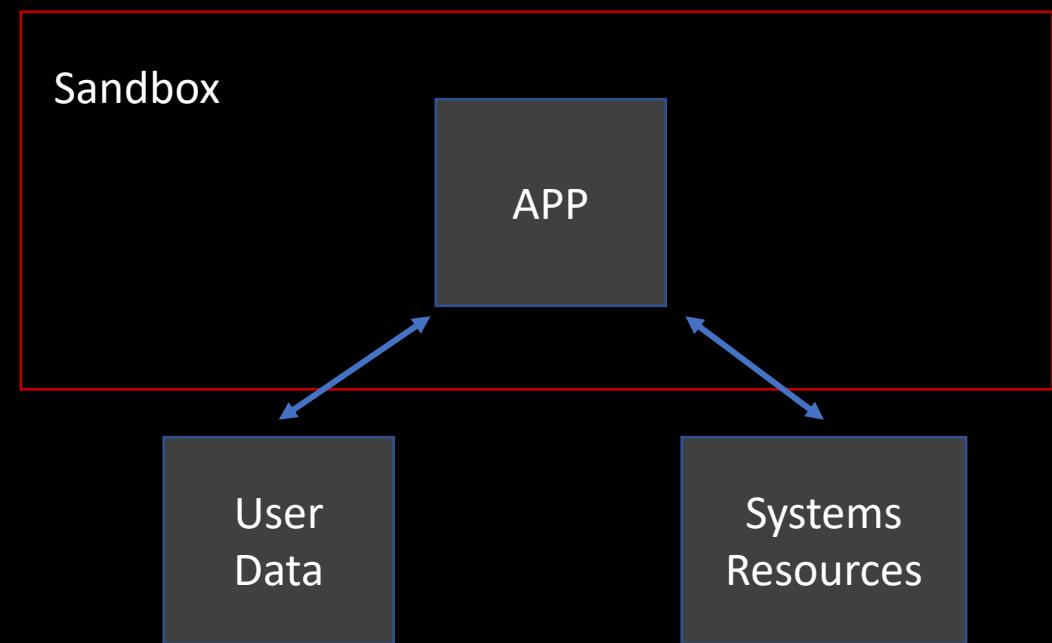
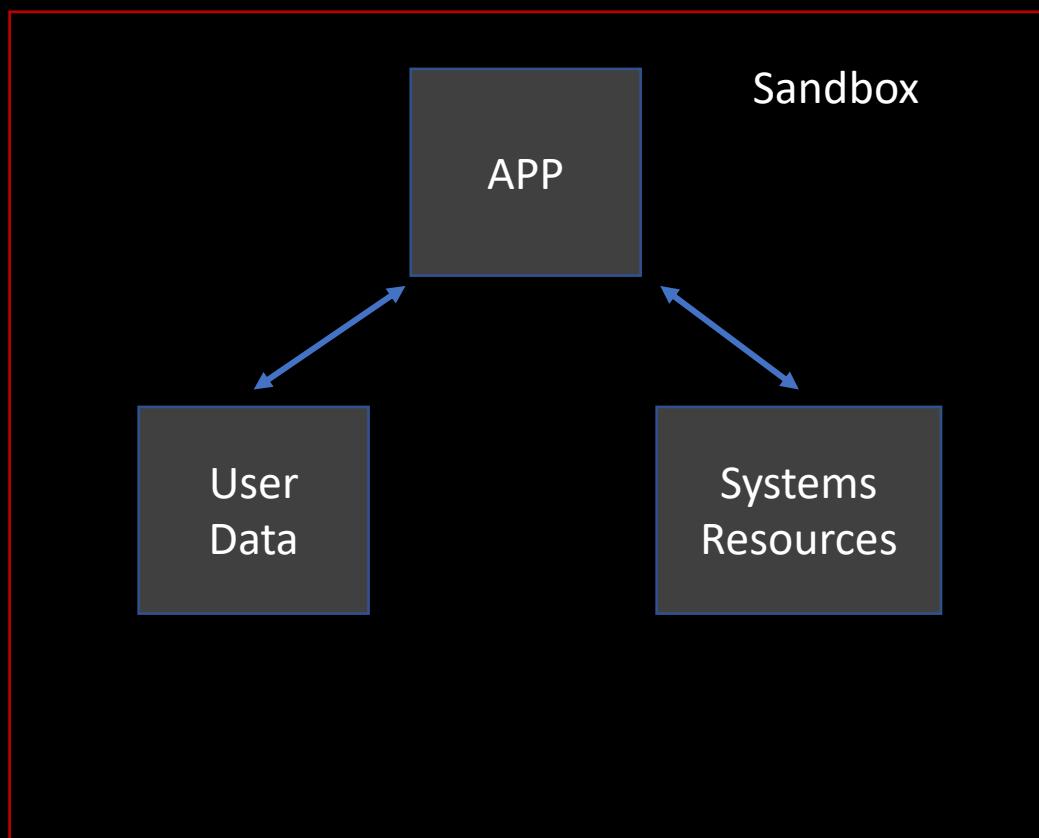
---

Command execution

# Mobile Device Specific

- Baseband attacks (GSM/3GPP)
- SMS Phishing (SMiShing)
- Jailbroken / Rooted Devices
- Mobile Malware
- Insecure App Stores
- Sensitive Data Storage
- No Encryption / Weak Encryption
- Improper SSL/TLS Validation
- Configuration Manipulations via in secure Apps
- Dynamic Runtime Injection
- Unintended Permissions
- Privilege Escalation

# App Sandboxing Issues



## Apple iOS

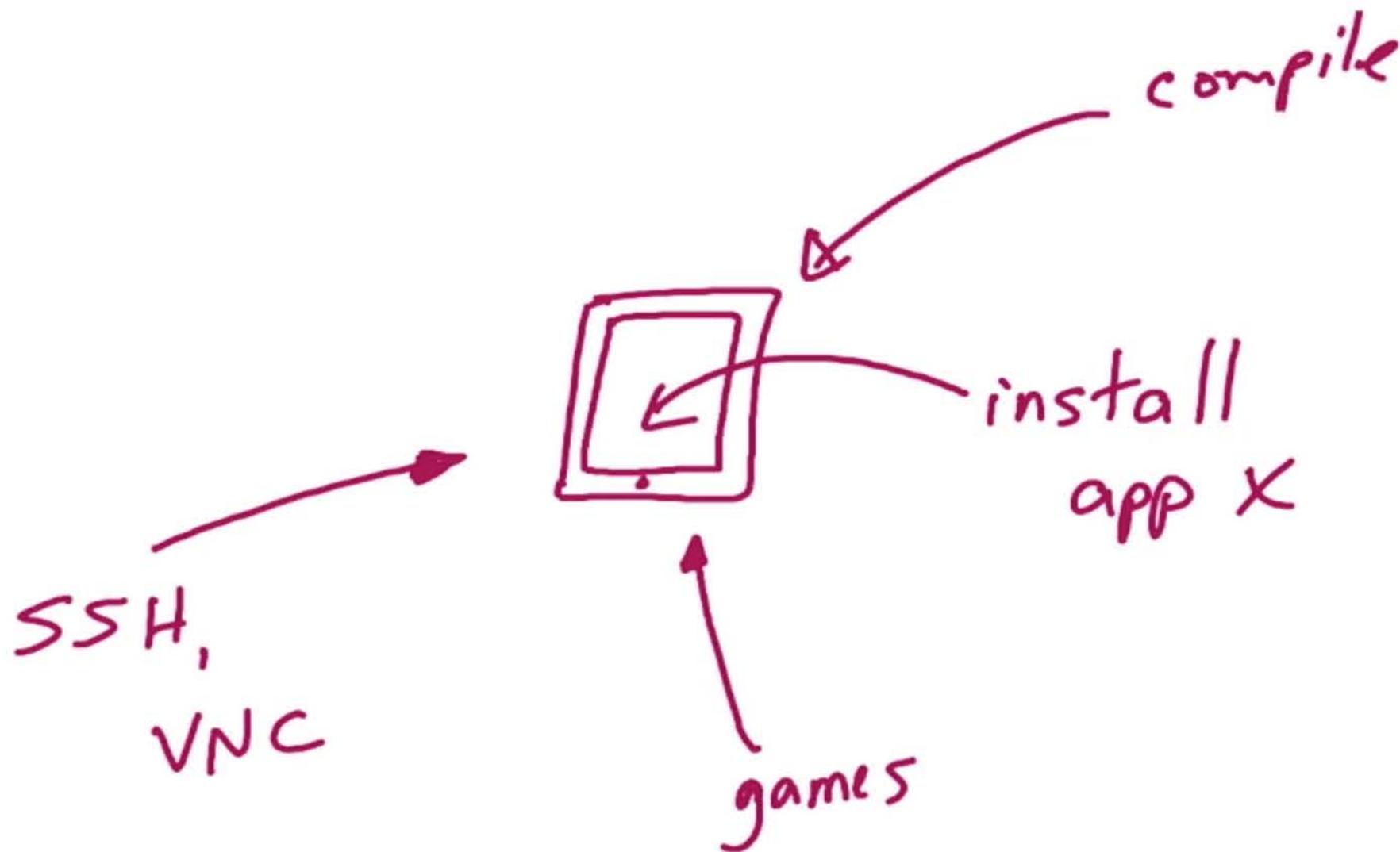
- Address Space Layout Randomization
- Code Signing
- App Sandboxing
- Other security features...

# Threats



- Authentication and authorization vulnerabilities
- Session management
- Input validation
- Error handling
- Weak Crypto

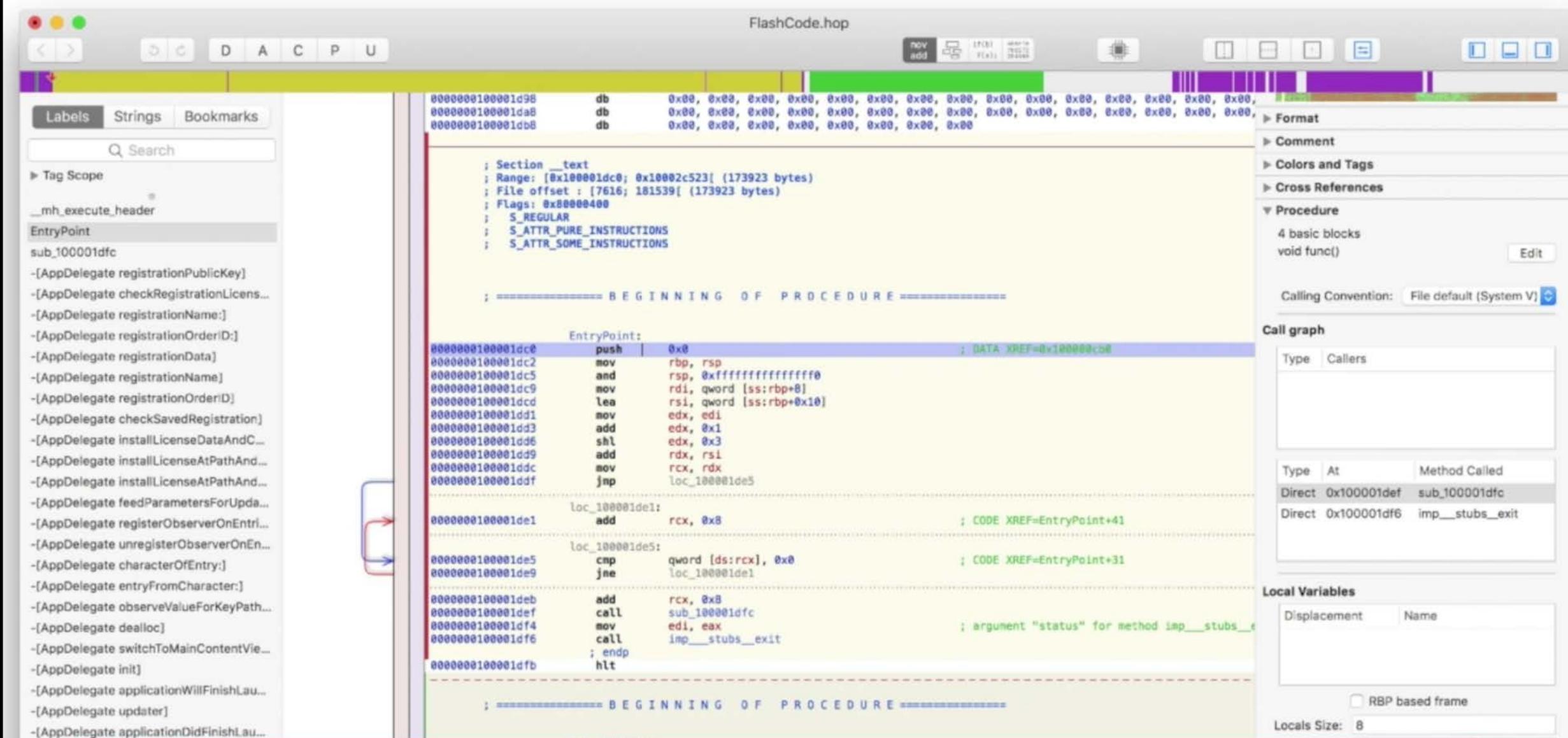
# Jailbreak



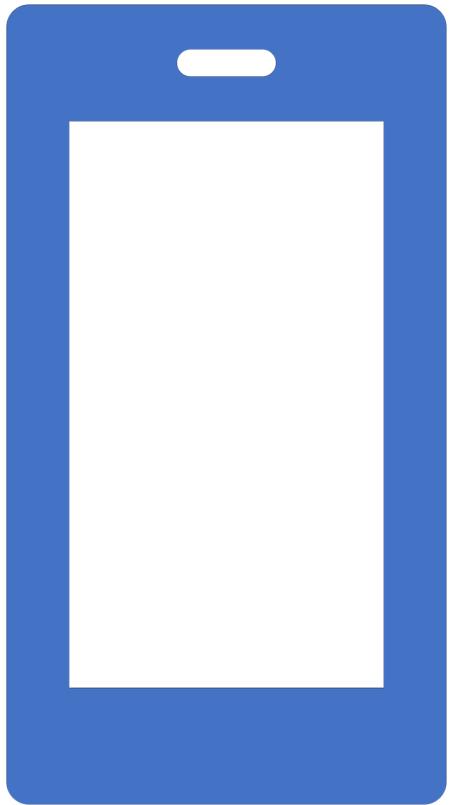
Jailbreaking is  
Harder, but not  
impossible...

iOS 11.x

LiberiOS



<https://www.hopperapp.com>



# IOS Testing Guide

<https://github.com/OWASP/owasp-mstg#ios-testing-guide>



# ANDROID SECURITY

<https://source.android.com/security/>



## Android Framework

### APPLICATIONS

ALARM • BROWSER • CALCULATOR • CALENDAR •  
CAMERA • CLOCK • CONTACTS • DIALER • EMAIL •  
HOME • IM • MEDIA PLAYER • PHOTO ALBUM •  
SMS/MMS • VOICE DIAL

### ANDROID FRAMEWORK

CONTENT PROVIDERS • MANAGERS (ACTIVITY,  
LOCATION, PACKAGE, NOTIFICATION, RESOURCE,  
TELEPHONY, WINDOW) • VIEW SYSTEM

### NATIVE LIBRARIES

AUDIO MANAGER • FREETYPE • LIBC •  
MEDIA FRAMEWORK • OPENGL/ES • SQLITE  
• SSL • SURFACE MANAGER • WEBKIT

### ANDROID RUNTIME

CORE LIBRARIES •  
DALVIK VM

### HAL

AUDIO • BLUETOOTH • CAMERA • DRM •  
EXTERNAL STORAGE • GRAPHICS • INPUT •  
MEDIA • SENSORS • TV

### LINUX KERNEL

DRIVERS (AUDIO, BINDER (IPC), BLUETOOTH,  
CAMERA, DISPLAY, KEYPAD, SHARED MEMORY,  
USB, WIFI) • POWER MANAGEMENT

[Overview](#)[Kernel Security](#)[App Security](#)[Implementing Security](#)[Updates and Resources](#)[Reports](#)[Enhancements](#)[Acknowledgements](#)

## Google security services

Google provides a set of cloud-based services that are available to compatible Android devices with [Google Mobile Services](#). While these services are not part of the Android Open Source Project, they are included on many Android devices. For more information on some of these services, see Android Security's [2017 Year in Review](#).

The primary Google security services are:

- **Google Play:** Google Play is a collection of services that allow users to discover, install, and purchase applications from their Android device or the web. Google Play makes it easy for developers to reach Android users and potential customers. Google Play also provides community review, application [license verification](#), application security scanning, and other security services.
- **Android updates:** The Android update service delivers new capabilities and security updates to selected Android devices, including updates through the web or over the air (OTA).
- **Application services:** Frameworks that allow Android applications to use cloud capabilities such as ([backing up](#)) application data and settings and cloud-to-device messaging ([C2DM](#)) for push messaging.
- **Verify Apps:** Warn or automatically block the installation of harmful applications, and continually scan applications on the device, warning about or removing [harmful apps](#).
- **SafetyNet:** A privacy preserving intrusion detection system to assist Google tracking and mitigating known security threats in addition to identifying new security threats.
- **SafetyNet Attestation:** Third-party API to determine whether the device is CTS compatible. [Attestation](#) can also assist identify the Android app communicating with the app server.
- **Android Device Manager:** A [web app](#) and [Android app](#) to locate lost or stolen device.

# Additional Resources for Mobile Device Security



---

Mobile Device Security:

[https://cehreview.com/go/mobile\\_device\\_security](https://cehreview.com/go/mobile_device_security)

---

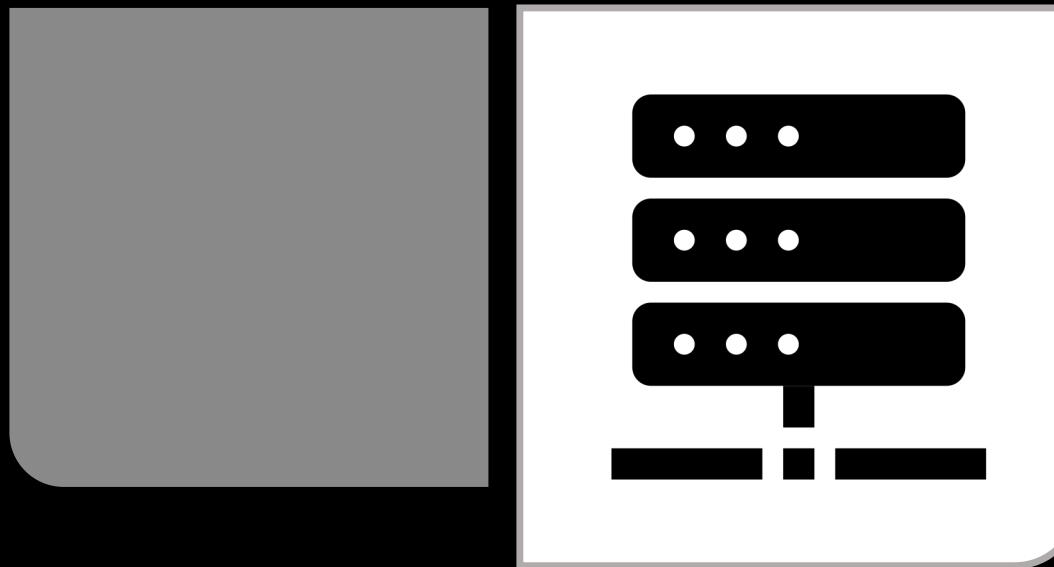
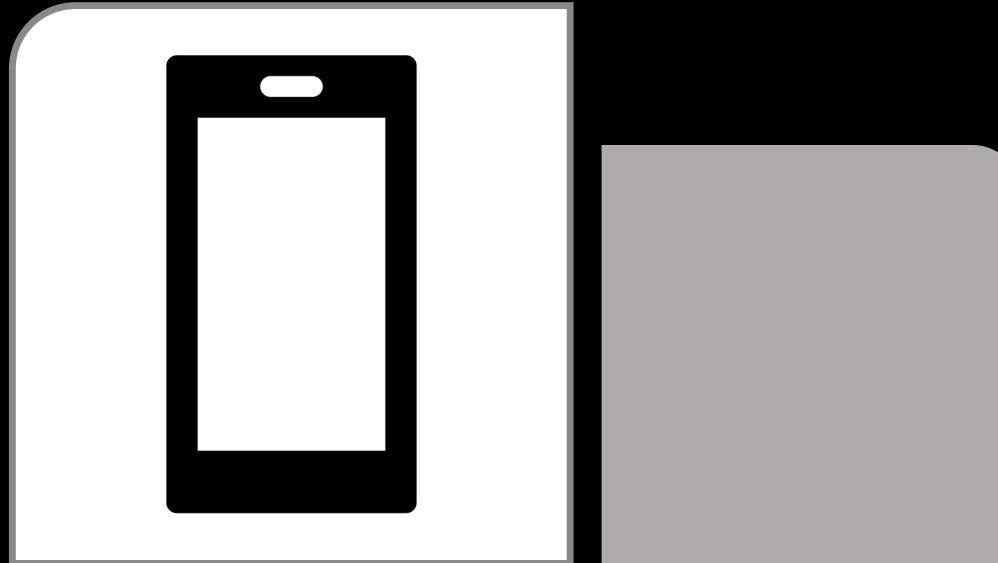
Hacking Android

[https://cehreview.com/go/hacking\\_android](https://cehreview.com/go/hacking_android)

---

Hacking iOS

[https://cehreview.com/go/hacking\\_ios](https://cehreview.com/go/hacking_ios)



## Your Setup

---

- WebSploit Container: Hackazon
- Mobile Emulator
- Detailed Instructions in Your Guide

# Exercise 6

Hacking Web Application Programming Interfaces (APIs)



# Weak Crypto Implementations

# Insecure Cryptographic Storage

Insecure Cryptographic Storage occurs when sensitive data is not stored securely. Insecure Cryptographic Storage isn't a single vulnerability, but a collection of vulnerabilities.

- Make sure you are encrypting the correct data
- Make sure you have proper key storage and management
- Make sure that you are not using known bad algorithms
- Make sure you are not implementing your own cryptography, which may or may not be secure

Exercise 7

## Exploiting Weak Cryptographic Implementations

The logo features a large, stylized, black, brush-painted font word "H4cker" on the left. To the right of the word is a smaller, bold, black sans-serif font word ".ORG". The background of the logo is a white circle with a dark blue, textured border.

H4cker

**H 4 C K E R . O R G**

# Path Traversal

`../../../../etc/shadow`

# File Inclusion and Directory Traversal

Used to access files and directories that are stored outside the web root folder.

You can manipulate variables that reference files with “dot-dot-slash (../)” or by using absolute file paths.

You may be able access arbitrary files and directories stored on file system including application source code or configuration and critical system files, like the /etc/passwd

Path (Directory) Traversal

# Exercise 8

The logo features a large, stylized, black, brush-painted font word "H4cker" where the "4" has a vertical bar through it. Below it is a smaller, bold, black sans-serif font word "H4CKER.ORG". The background of the logo is a white circle with a dark blue, textured border.

H4cker

**H4CKER.ORG**



# Exercise 9

Additional XSS Exploitation

Bypassing Additional Web Application Flaws

# Exercise 10

The logo for H4cker.org features the word "H4cker" in a large, black, cursive font. Below it, the word "H4CKER.ORG" is written in a smaller, bold, black, sans-serif font.

H4cker

H4CKER.ORG

Fuzzing

# Exercise 11

Hacker

**H 4 C K E R . O R G**



Thank You!