Intense Introduction to Hacking Web Applications

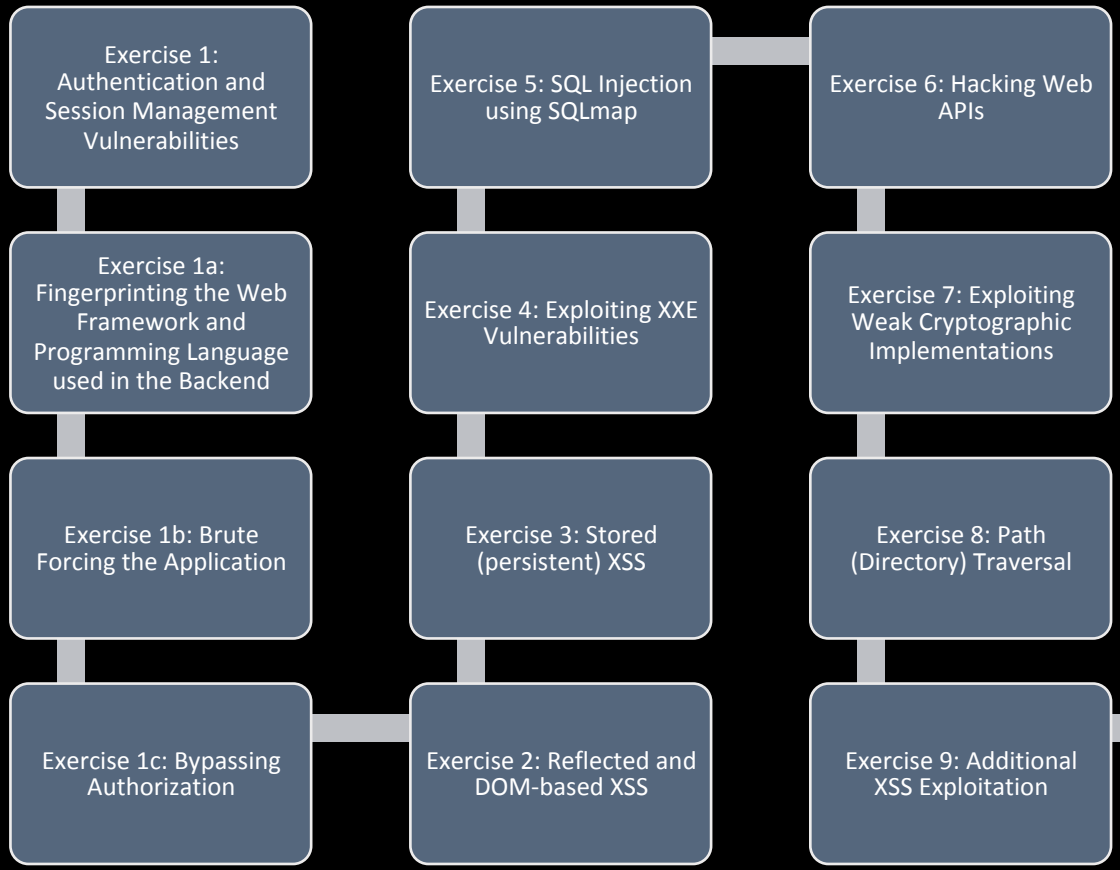
Omar Santos

@santosomar

Agenda

H4CKER.ORG

Exercises



H4CKER.ORG

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

Resources

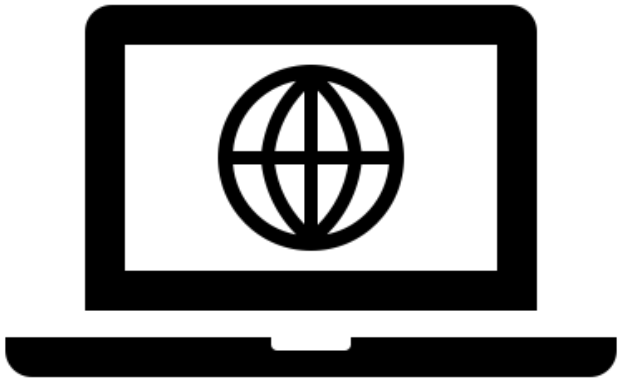
The Art of Hacking: https://theartofhacking.org
GitHub: https://theartofhacking.org/github
Additional Live Training: https://h4cker.org/training

GitHub Repo with over 6,000 Cyber Security References

http://h4cker.org/github

http://h4cker.org/webapps

This is a hands-on class!



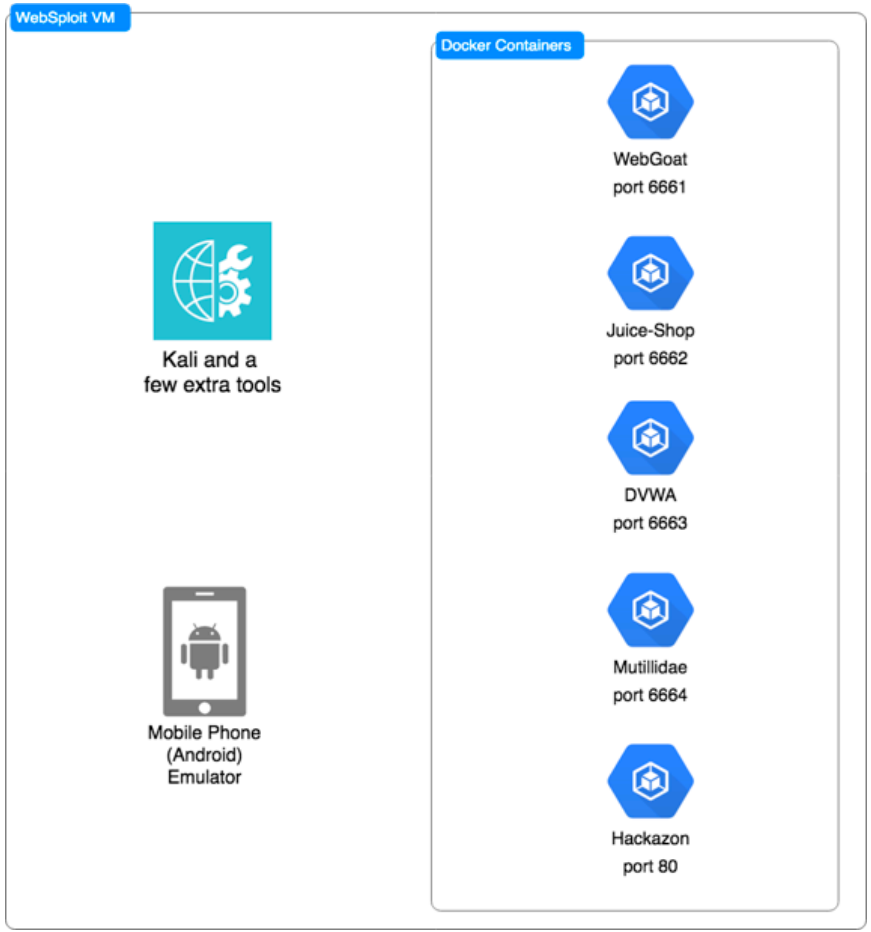
Logistics

WebSploit: https://websploit.h4cker.org

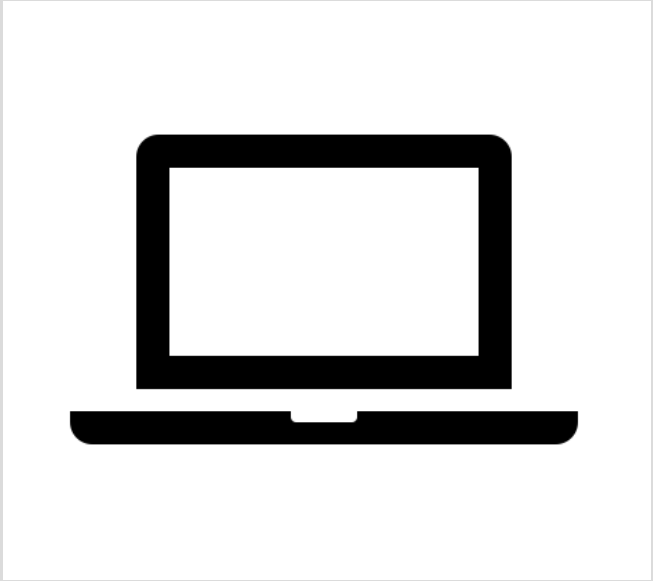
Your Lab Guide

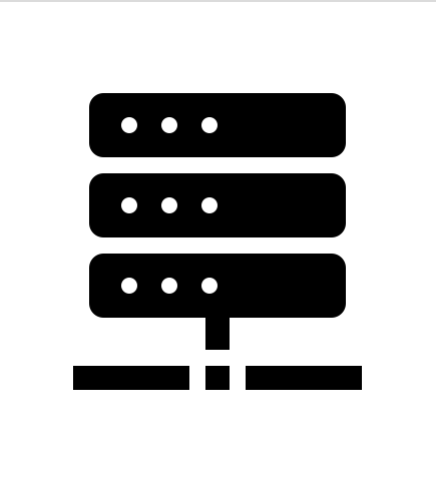
Download it under “Resources”

WebSploit Components



Building Your Own Lab





Penetration Testing

Methodologies

Become Familiar with Penetration Testing Methodologies

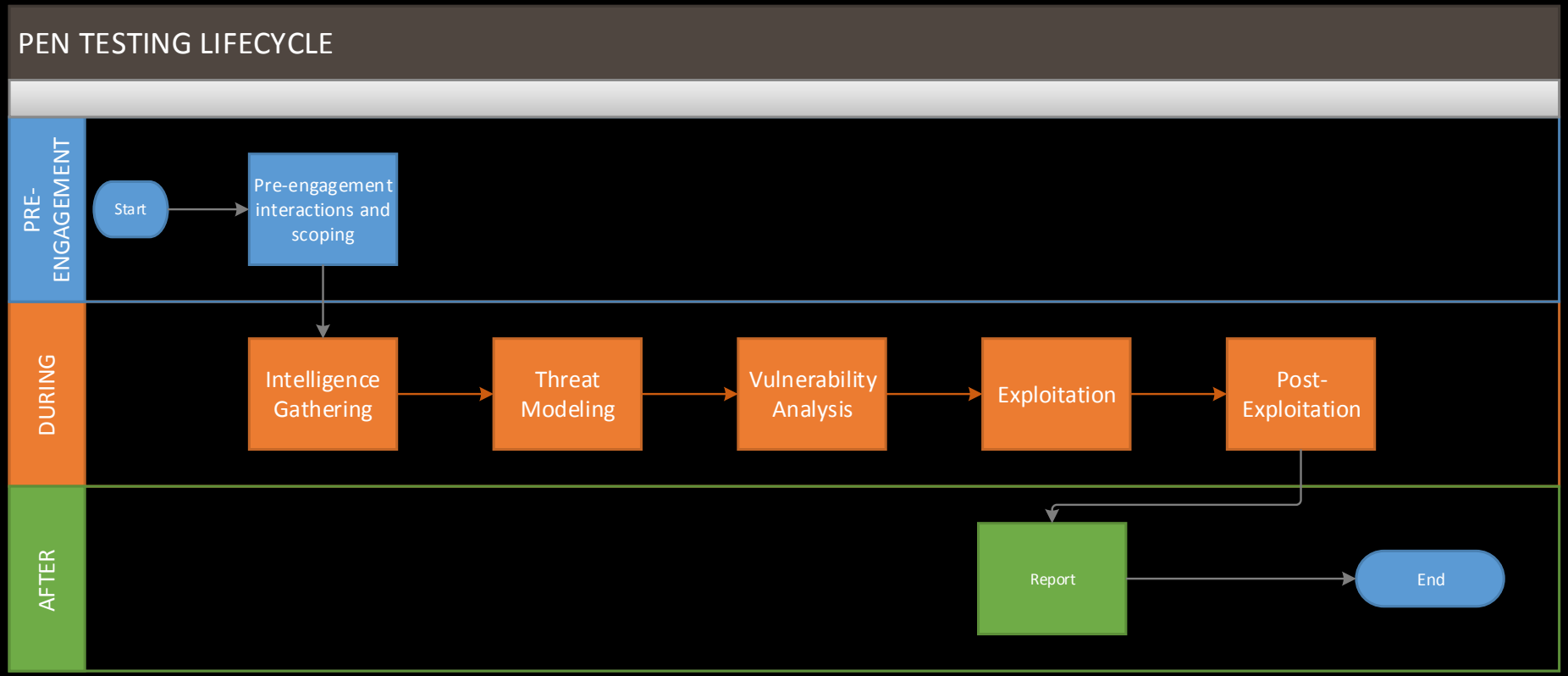
Penetration Testing Execution Standard
http://www.pentest-standard.org

OWASP Testing Guide:
https://www.owasp.org/index.php/OWASP_Testing_Project

NIST 800-115: Technical Guide to Information Security Te
Assessment:
http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublicati

Open Source Security Testing Methodol
Manual (OSSTMM):

PEN TESTING LIFECYCLE



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

OWASP Open Web Application Security Project | Testing Guide 4.0

)release(

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

The Art of Hacking

https://www.owasp.org/index.php/OWASP_Testing_Project

A Balanced Approach:

- Pen Testing
- Source code review
- Threat Modeling

(SDLC)

4.2 Information Gathering

4.2.1 Conduct Search Engine Discovery and Reconnaissance for Information Leakage (OTG-INFO-001)

4.2.2 Fingerprint Web Server (OTG-INFO-002)

4.2.3 Review Webserver Metafiles for Information Leakage (OTG-INFO-003)

4.2.4 Enumerate Applications on Webserver (OTG-INFO-004)

4.2.5 Review Webpage Comments and Metadata for Information Leakage (OTG-INFO-005)

4.2.6 Identify application entry points (OTG-INFO-006)

4.2.7 Map execution paths through application (OTG-INFO-007)

4.2.8 Fingerprint Web Application Framework (OTG-INFO-008)

4.2.9 Fingerprint Web Application (OTG-INFO-009)

4.2.10 Map Application Architecture (OTG-INFO-010)

All part of https://www.owasp.org/index.php/OWASP_Testing_Project

4.3 Configuration and Deployment Management Testing

4.3.1 Test Network/Infrastructure Configuration (OTG-CONFIG-001)

4.3.2 Test Application Platform Configuration (OTG-CONFIG-002)

4.3.3 Test File Extensions Handling for Sensitive Information (OTG-CONFIG-003)

4.3.4 Review Old, Backup and Unreferenced Files for Sensitive Information (OTG-CONFIG-004)

4.3.5 Enumerate Infrastructure and Application Admin Interfaces (OTG-CONFIG-005)

4.3.6 Test HTTP Methods (OTG-CONFIG-006)

4.3.7 Test HTTP Strict Transport Security (OTG-CONFIG-007)

4.3.8 Test RIA cross domain policy (OTG-CONFIG-008)

4.3.9 Test File Permission (OTG-CONFIG-009)

All part of https://www.owasp.org/index.php/OWASP_Testing_Project

4.4 Identity Management Testing

4.4.1 Test Role Definitions (OTG-IDENT-001)

4.4.2 Test User Registration Process (OTG-IDENT-002)

4.4.3 Test Account Provisioning Process (OTG-IDENT-003)

4.4.4 Testing for Account Enumeration and Guessable User Account (OTG-IDENT-004)

4.4.5 Testing for Weak or unenforced username policy (OTG-IDENT-005)

All part of https://www.owasp.org/index.php/OWASP_Testing_Project

4.5 Authentication Testing

4.5.1 Testing for Credentials Transported over an Encrypted Channel (OTG-AUTHN-001)

4.5.2 Testing for default credentials (OTG-AUTHN-002)

4.5.3 Testing for Weak lock out mechanism (OTG-AUTHN-003)

4.5.4 Testing for bypassing authentication schema (OTG-AUTHN-004)

4.5.5 Test remember password functionality (OTG-AUTHN-005)

4.5.6 Testing for Browser cache weakness (OTG-AUTHN-006)

4.5.7 Testing for Weak password policy (OTG-AUTHN-007)

4.5.8 Testing for Weak security question/answer (OTG-AUTHN-008)

4.5.9 Testing for weak password change or reset functionalities (OTG-AUTHN-009)

4.5.10 Testing for Weaker authentication in alternative channel (OTG-AUTHN-010)

All part of https://www.owasp.org/index.php/OWASP_Testing_Project

4.6 Authorization Testing

4.6.1 Testing Directory traversal/file include (OTG-AUTHZ-001)

4.6.2 Testing for bypassing authorization schema (OTG-AUTHZ-002)

4.6.3 Testing for Privilege Escalation (OTG-AUTHZ-003)

4.6.4 Testing for Insecure Direct Object References (OTG-AUTHZ-004)

All part of https://www.owasp.org/index.php/OWASP_Testing_Project

4.7 Session Management Testing

4.7.1 Testing for Bypassing Session Management Schema (OTG-SESS-001)

4.7.2 Testing for Cookies attributes (OTG-SESS-002)

4.7.3 Testing for Session Fixation (OTG-SESS-003)

4.7.4 Testing for Exposed Session Variables (OTG-SESS-004)

4.7.5 Testing for Cross Site Request Forgery (CSRF) (OTG-SESS-005)

4.7.6 Testing for logout functionality (OTG-SESS-006)

4.7.7 Test Session Timeout (OTG-SESS-007)

4.7.8 Testing for Session puzzling (OTG-SESS-008)

All part of https://www.owasp.org/index.php/OWASP_Testing_Project

4.8 Input Validation Testing

4.8.1 Testing for Reflected Cross Site Scripting (OTG-INPVAL-001)

4.8.2 Testing for Stored Cross Site Scripting (OTG-INPVAL-002)

4.8.3 Testing for HTTP Verb Tampering (OTG-INPVAL-003)

4.8.4 Testing for HTTP Parameter pollution (OTG-INPVAL-004)

4.8.5 Testing for SQL Injection (OTG-INPVAL-005)

4.8.5.1 Oracle Testing

4.8.5.2 MySQL Testing

4.8.5.3 SQL Server Testing

4.8.5.4 Testing PostgreSQL (from OWASP BSP)

4.8.5.5 MS Access Testing

4.8.5.6 Testing for NoSQL injection

4.8.6 Testing for LDAP Injection (OTG-INPVAL-006)

4.8.7 Testing for ORM Injection (OTG-INPVAL-007)

4.8.8 Testing for XML Injection (OTG-INPVAL-008)

4.8.9 Testing for SSI Injection (OTG-INPVAL-009)

4.8.10 Testing for XPath Injection (OTG-INPVAL-010)

4.8.11 IMAP/SMTP Injection (OTG-INPVAL-011)

4.8.12 Testing for Code Injection (OTG-INPVAL-012)

4.8.12.1 Testing for Local File Inclusion

4.8.12.2 Testing for Remote File Inclusion

4.8.13 Testing for Command Injection (OTG-INPVAL-013)

4.8.14 Testing for Buffer overflow (OTG-INPVAL-014)

4.8.14.1 Testing for Heap overflow

4.8.14.2 Testing for Stack overflow

4.8.14.3 Testing for Format string

4.8.15 Testing for incubated vulnerabilities (OTG-INPVAL-015)

4.8.16 Testing for HTTP Splitting/Smuggling (OTG-INPVAL-016)

4.8.17 Testing for HTTP Incoming Requests (OTG-INPVAL-017)

All part of https://www.owasp.org/index.php/OWASP_Testing_Project

4.9 Testing for Error Handling

4.9.1 Analysis of Error Codes (OTG-ERR-001)

4.9.2 Analysis of Stack Traces (OTG-ERR-002)

4.10 Testing for weak Cryptography

4.10.1 Testing for Weak SSL/TLS Ciphers, Insufficient Transport Layer Protection (OTG-CRYPST-001)

4.10.2 Testing for Padding Oracle (OTG-CRYPST-002)

4.10.3 Testing for Sensitive information sent via unencrypted channels (OTG-CRYPST-003)

4.10.4 Testing for Weak Encryption (OTG-CRYPST-004)

All part of https://www.owasp.org/index.php/OWASP_Testing_Project

4.11 Business Logic Testing

4.11.1 Test Business Logic Data Validation (OTG-BUSLOGIC-001)

4.11.2 Test Ability to Forge Requests (OTG-BUSLOGIC-002)

4.11.3 Test Integrity Checks (OTG-BUSLOGIC-003)

4.11.4 Test for Process Timing (OTG-BUSLOGIC-004)

4.11.5 Test Number of Times a Function Can be Used Limits (OTG-BUSLOGIC-005)

4.11.6 Testing for the Circumvention of Work Flows (OTG-BUSLOGIC-006)

4.11.7 Test Defenses Against Application Mis-use (OTG-BUSLOGIC-007)

4.11.8 Test Upload of Unexpected File Types (OTG-BUSLOGIC-008)

4.11.9 Test Upload of Malicious Files (OTG-BUSLOGIC-009)

All part of https://www.owasp.org/index.php/OWASP_Testing_Project

4.12 Client Side Testing

4.12.1 Testing for DOM based Cross Site Scripting (OTG-CLIENT-001)

4.12.2 Testing for JavaScript Execution (OTG-CLIENT-002)

4.12.3 Testing for HTML Injection (OTG-CLIENT-003)

4.12.4 Testing for Client Side URL Redirect (OTG-CLIENT-004)

4.12.5 Testing for CSS Injection (OTG-CLIENT-005)

4.12.6 Testing for Client Side Resource Manipulation (OTG-CLIENT-006)

4.12.7 Test Cross Origin Resource Sharing (OTG-CLIENT-007)

4.12.8 Testing for Cross Site Flashing (OTG-CLIENT-008)

4.12.9 Testing for Clickjacking (OTG-CLIENT-009)

4.12.10 Testing WebSockets (OTG-CLIENT-010)

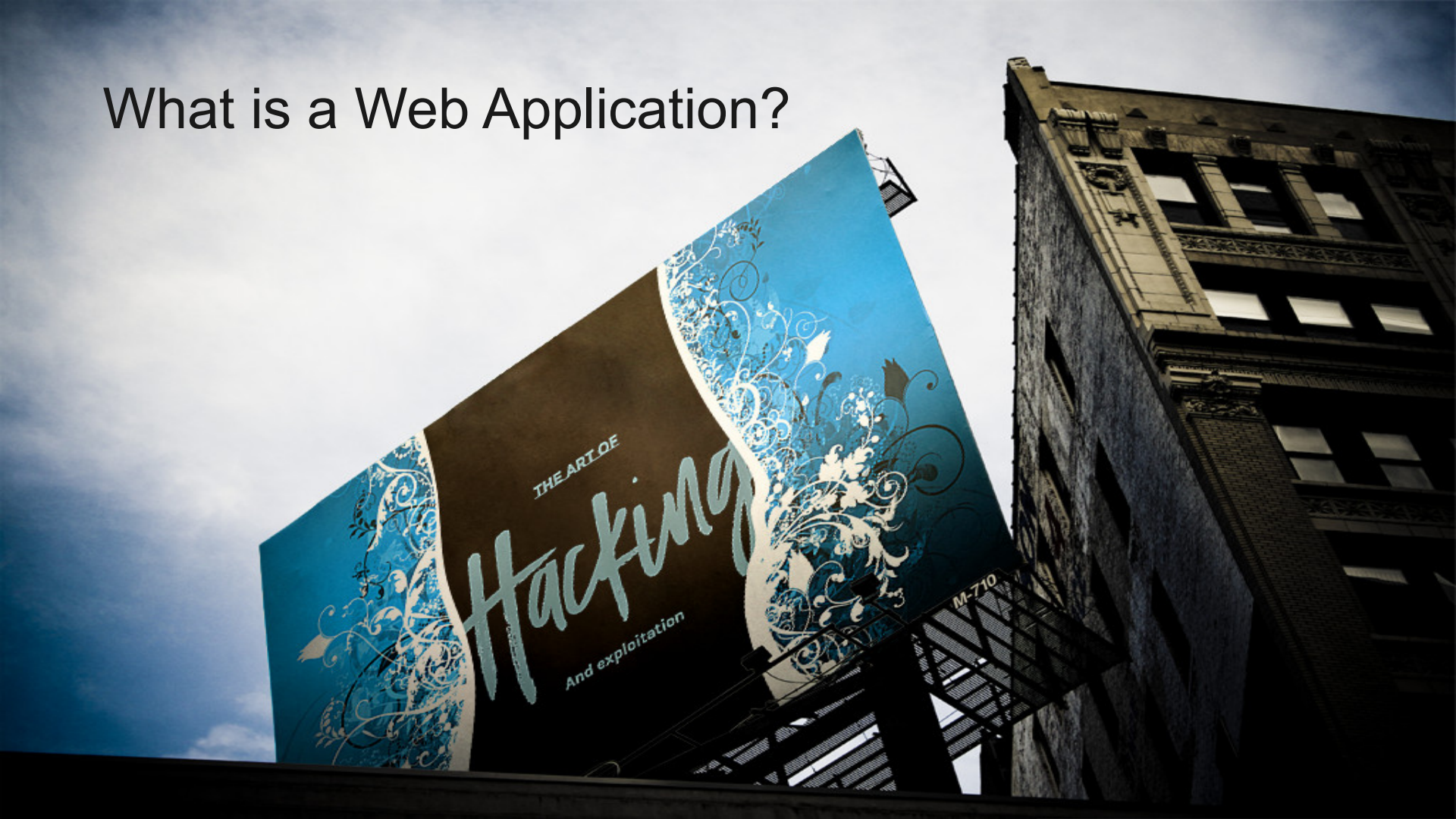
4.12.11 Test Web Messaging (OTG-CLIENT-011)

4.12.12 Test Local Storage (OTG-CLIENT-012)

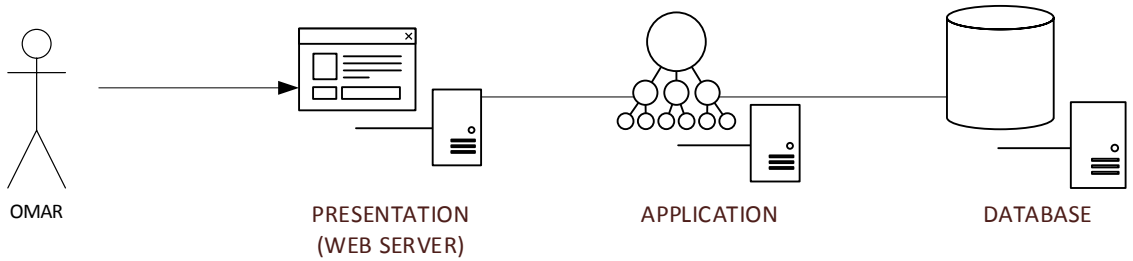
All part of https://www.owasp.org/index.php/OWASP_Testing_Project

Exploring How Web Applications Have Evolved Over Time

What is a Web Application?



THE TRADITIONAL WEB APPLICATION ARCHITECTURE




Waterfall
Agile
DevOps
Process
Datacenter
Hosted
Hybrid
Infrastructure
Monolith
N-Tier
Microservices
Architecture

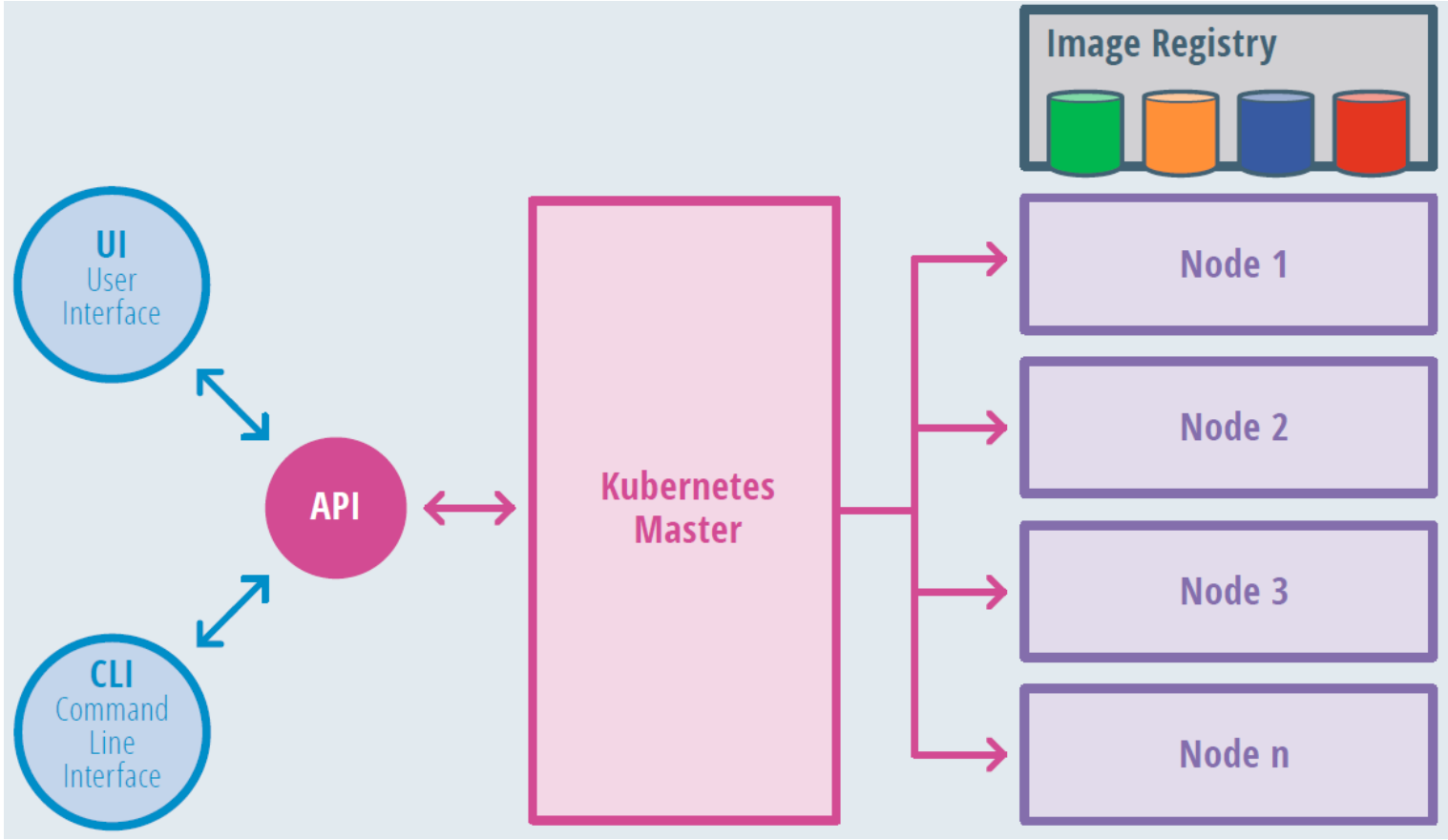
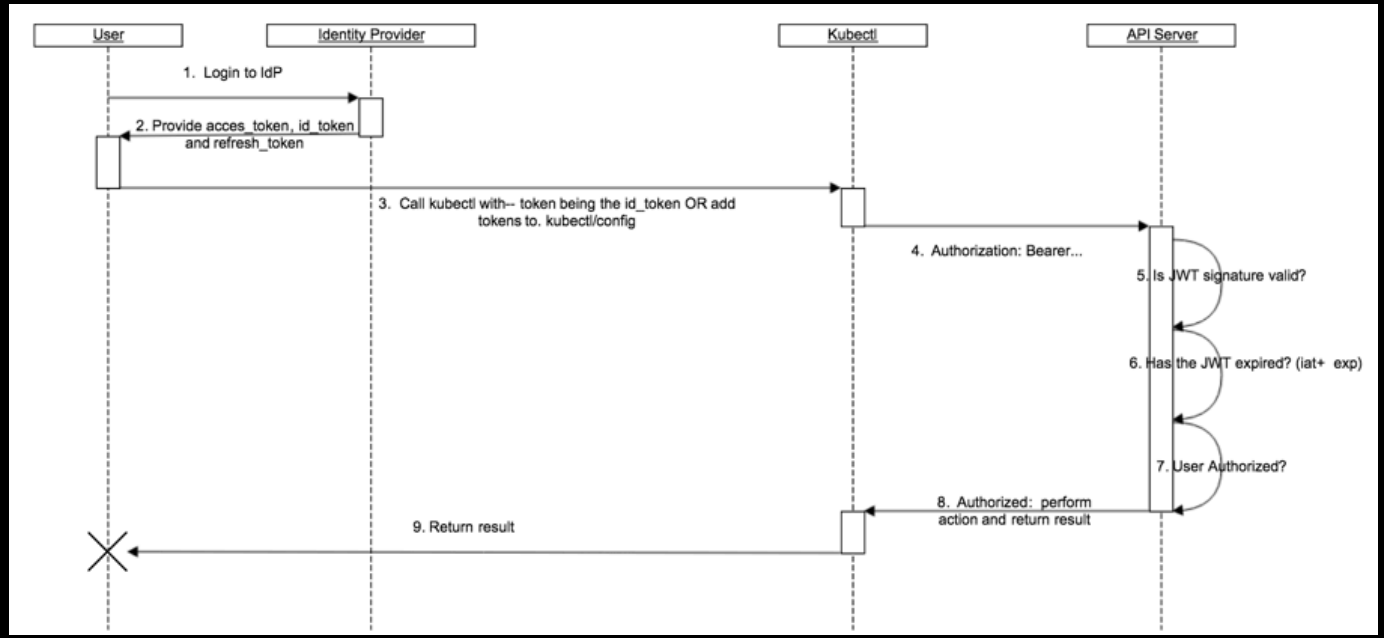


Image Registry
UI
User
Interface
API
CLI
Command
Line
Interface
Kubernetes
Master
Node 1
Node 2
Node 3
Node n



OPENID AUTHENTICATION FLOW

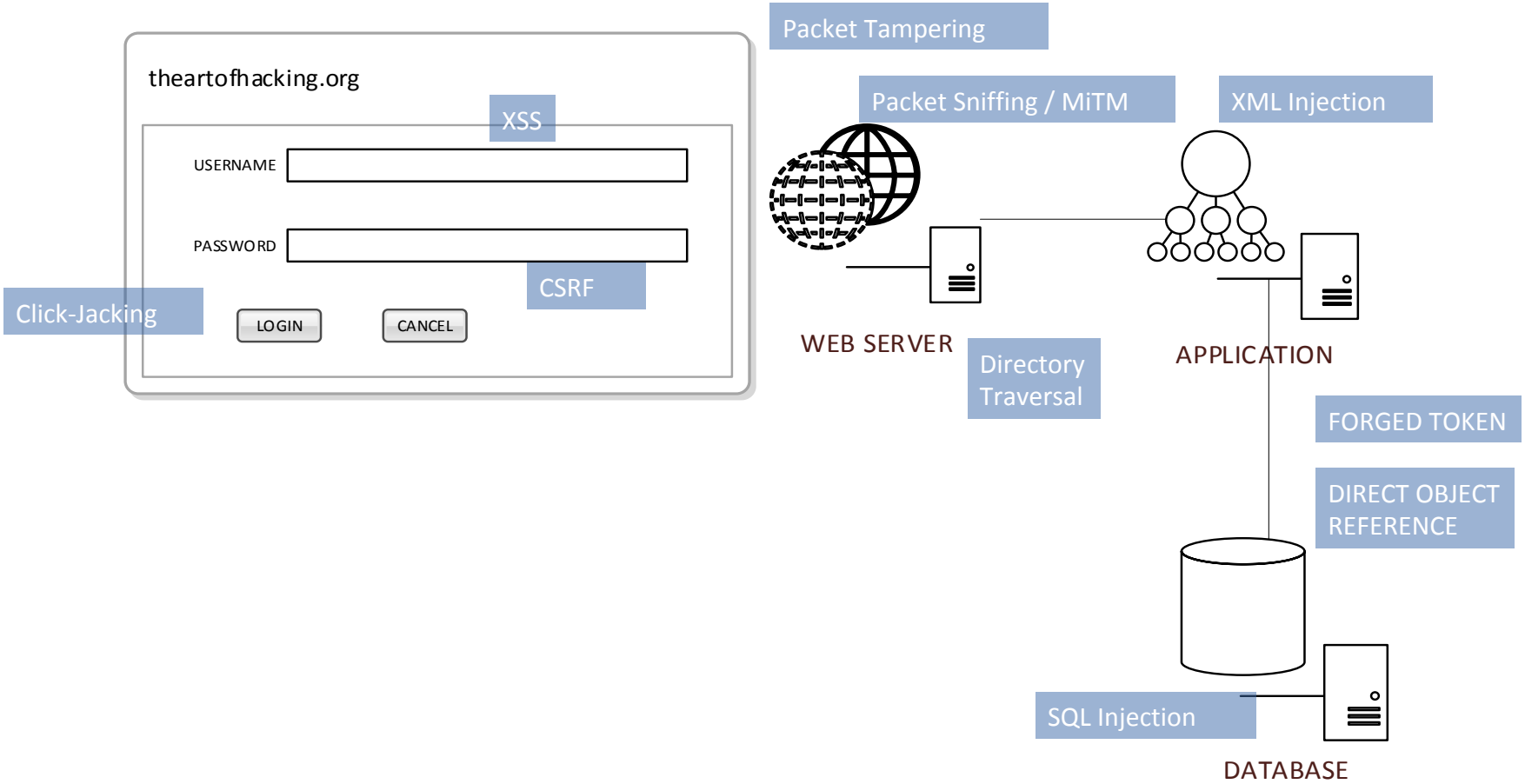
```
{
  "apiVersion": "authorization.k8s.io/v1beta1",
  "kind": "SubjectAccessReview",
  "spec": {
    "resourceAttributes": {
      "namespace": "kittensandponies",
      "verb": "get",
      "group": "unicorn.example.org",
      "resource": "pods"
    },
    "user": "jane",
    "group": [
      "group1",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

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)


theartofhacking.org
USERNAME
PASSWORD
LOGIN
CANCEL
WEB SERVER
APPLICATION
DATABASE


theartofhacking.org
XSS
USERNAME
PASSWORD
CSRF
Click-Jacking
LOGIN
CANCEL
Packet Tampering
Packet Sniffing / MiTM
XML Injection
WEB SERVER
Directory Traversal
APPLICATION
FORGED TOKEN
DIRECT OBJECT REFERENCE
SQL Injection
DATABASE

Machine-to-machine

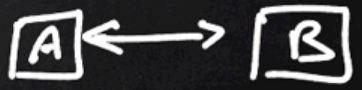
OAuth 2

OpenID Connect

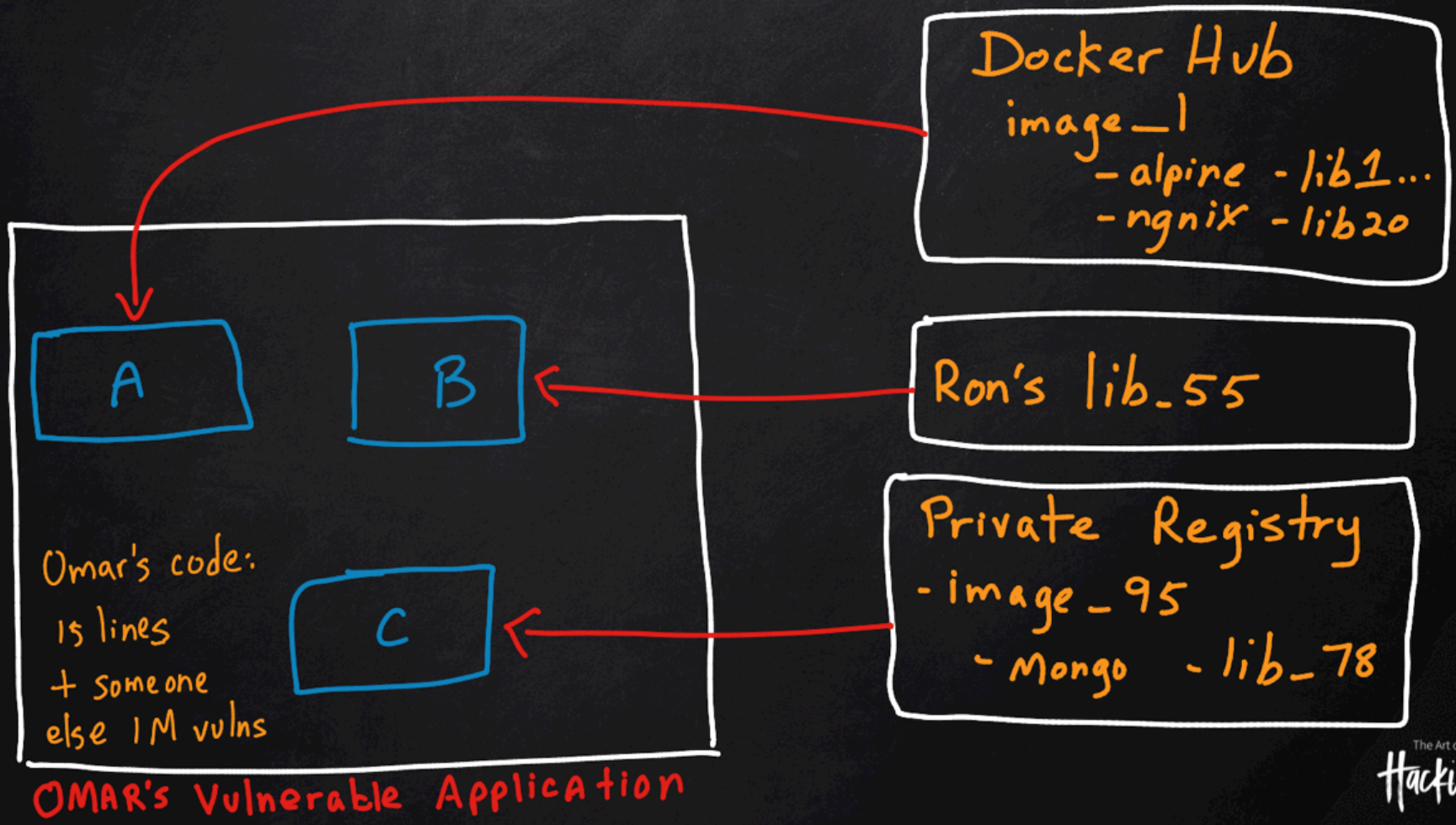
SAML

... more

MAchine -to- machine



- NO GUIs
- useful docs:
  - swagger / Open API
  - Graph QL docs
  - WSDL / WADL docs



santosomar organizing files — Latest commit 7e9db6b 30 minutes ago

..

README.md — Adding websploit — 34 minutes ago

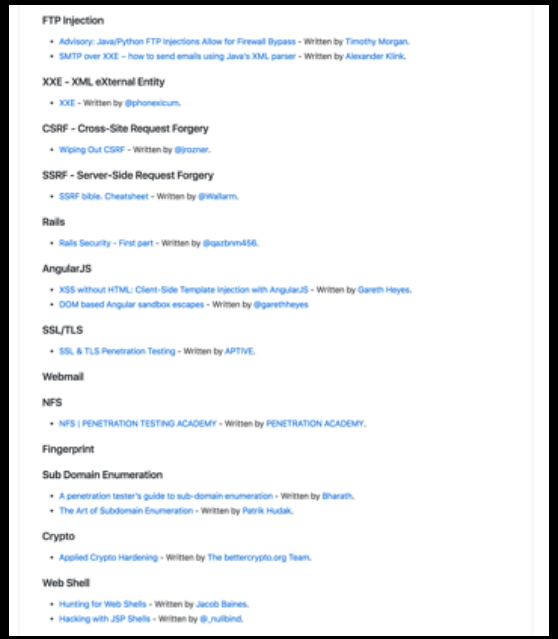
README.md

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
- 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
- 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

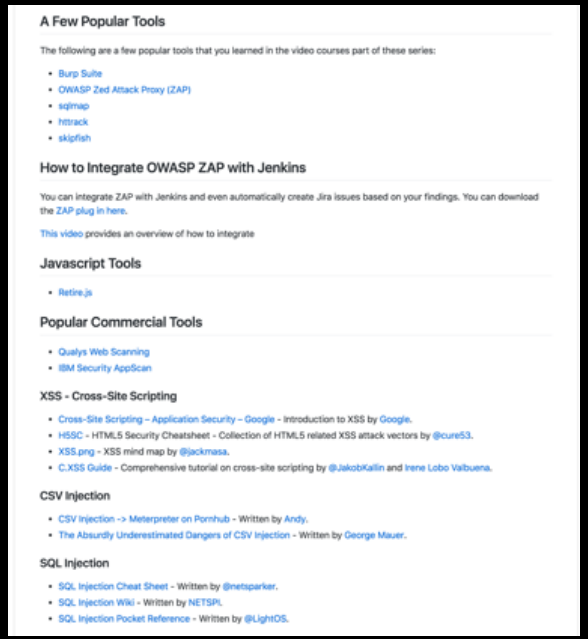
- A penetration tester's guide to sub-domain enumeration - Written by Bharath.
- The Art of Subdomain Enumeration - Written by Patrik Hudak.

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
- httrack
- 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.
- H5SC - HTML5 Security Cheatsheet - Collection of HTML5 related XSS attack vectors by @cure53.
- XSS.png - XSS mind map by @jackmasa.
- C.XSS 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

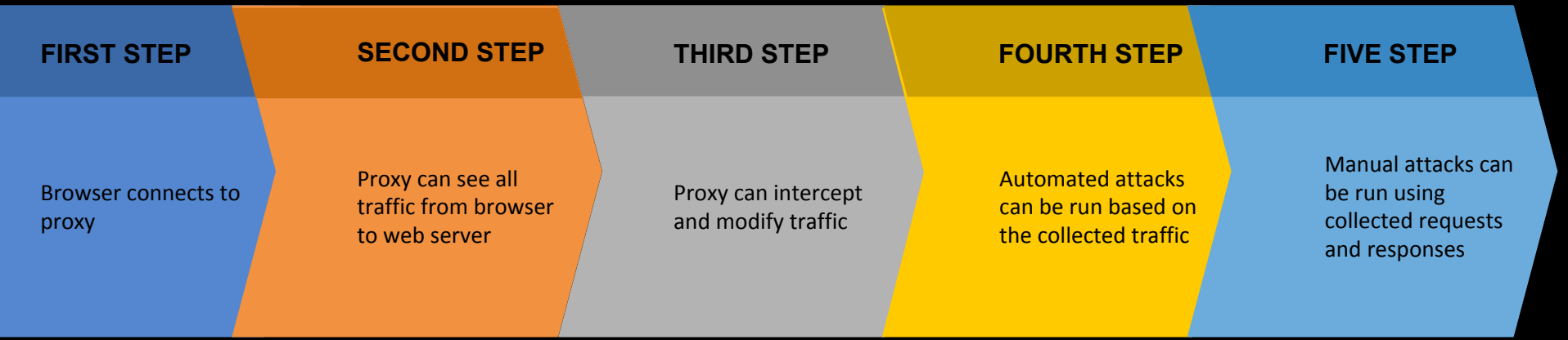
netsparker

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
predictable tokens, as previously discussed in this sect

Session sniffing: by collecting packets of unencrypted
sessions.

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

Man-in-the-browser attack: similar approach as man-i
middle attacks; however, in this case browser (or exte
plugin) is compromised and used to intercept and mar

Exercise 1: Authentication and Session Management Vulnerabilities

Exercise 1a	Exercise 1b	Exerc
Fingerprinting the Web Framework and Programming Language used in the Backend	Brute Forcing the Application	Bypassing Authoriza

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

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 ↷

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.

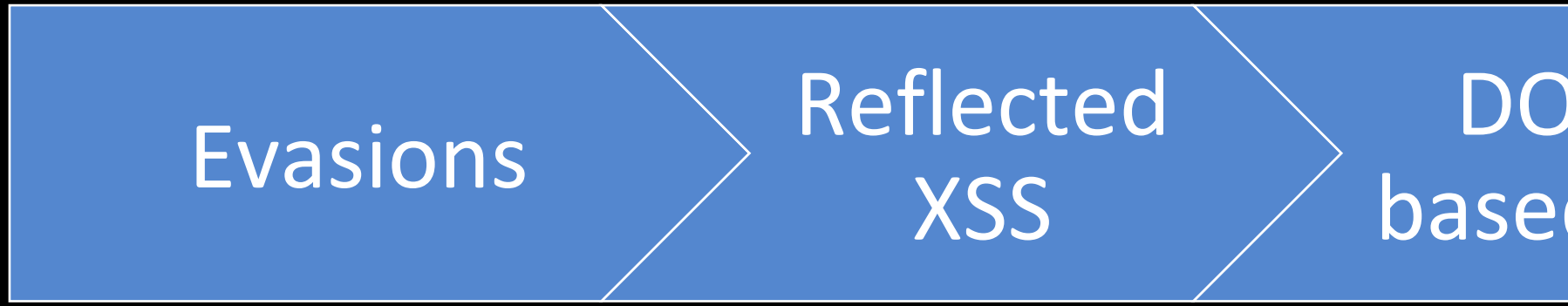
How to Test for Cross-site scripting Vulnerabilities

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)

Amazing Resources

https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)



Exercise 2: Reflected XSS

Stored (persistent) XSS

Exercise 3

XML External Entity (XXE)

Poorly configured XML parsers e
external entity references within
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.

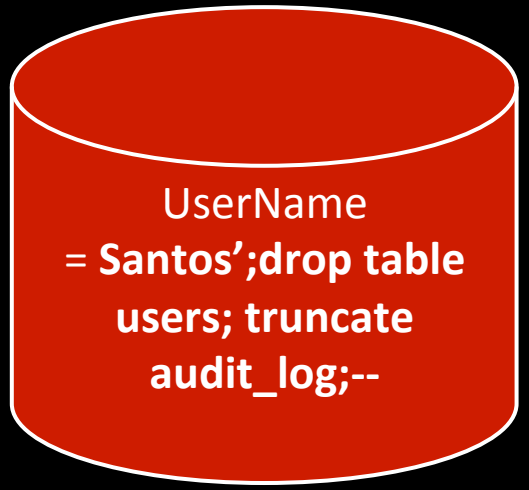
Listen to the OWASP Top Ten CSRF Podcast.

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

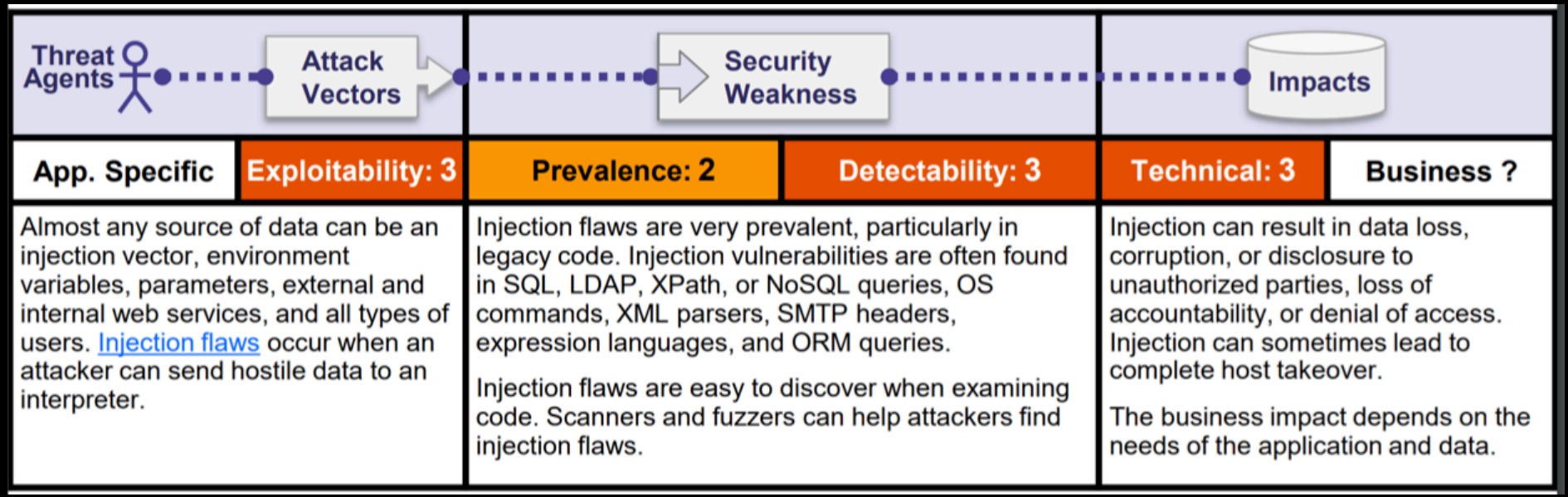
Use OWASP CSRF Guard to add CSRF protection to your Java applications. You can use CSRFProtector Project to protect your php applications or any project deployed using Apache Server. There is a .Net CSRF Guard at OWASP as well, but it's old and doesn't look complete.

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)



Introduction to Hacking Databases



Source: OWASP

https://www.owasp.org/index.php/Injection_Prevention_Cheat_Sheet
https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

Exercise 5

SQL Injection

Hacking Mobile Devices

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

M1 - Improper Platform Usage	This category covers misuse of a platform feature or failure to use platform security controls. It might include Android intents, platform permissions, misuse of TouchID, the Keychain, or some other security control that is part of the mobile operating system. There are several ways that mobile apps can experience this risk.
M2 - Insecure Data Storage	This new category is a combination of M2 + M4 from Mobile Top Ten 2014. This covers insecure data storage and unintended data leakage.
M3 - Insecure Communication	This covers poor handshaking, incorrect SSL versions, weak negotiation, cleartext communication of sensitive assets, etc.
M4 - Insecure Authentication	This category captures notions of authenticating the end user or bad session management. This can include: • Failing to identify the user at all when that should be required • Failure to maintain the user's identity when it is required • Weaknesses in session management
M5 - Insufficient Cryptography	The code applies cryptography to a sensitive information asset. However, the cryptography is insufficient in some way. Note that anything and everything related to TLS or SSL goes in M3. Also, if the app fails to use cryptography at all when it should, that probably belongs in M2. This category is for issues where cryptography was attempted, but it wasn't done correctly.

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

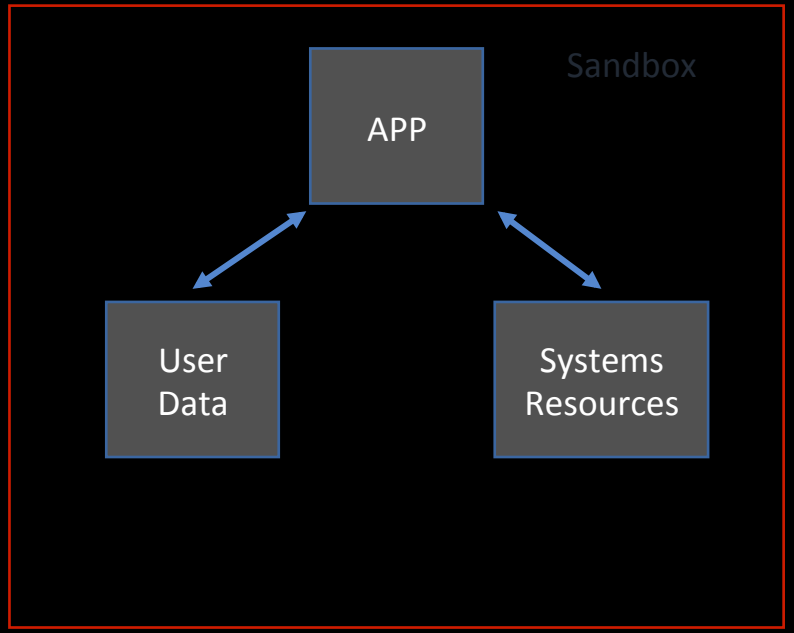
Other Client-based Attacks Still Apply

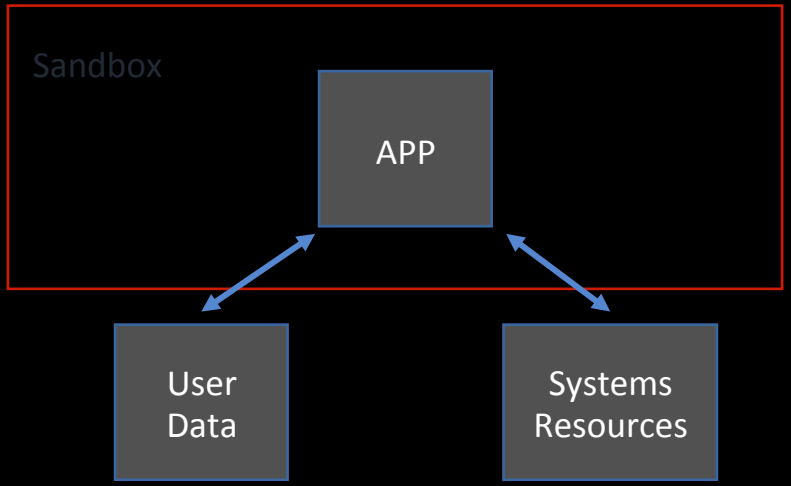
- Phishing
- Framing (iFrame-based attacks)
- Clickjacking
- Man-in-the-mobile / Man-in-the-browser
- Buffer overflows
- Data Caching
- XSS / CSRF
- Brute-force
- Weak input validation
- Command execution

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





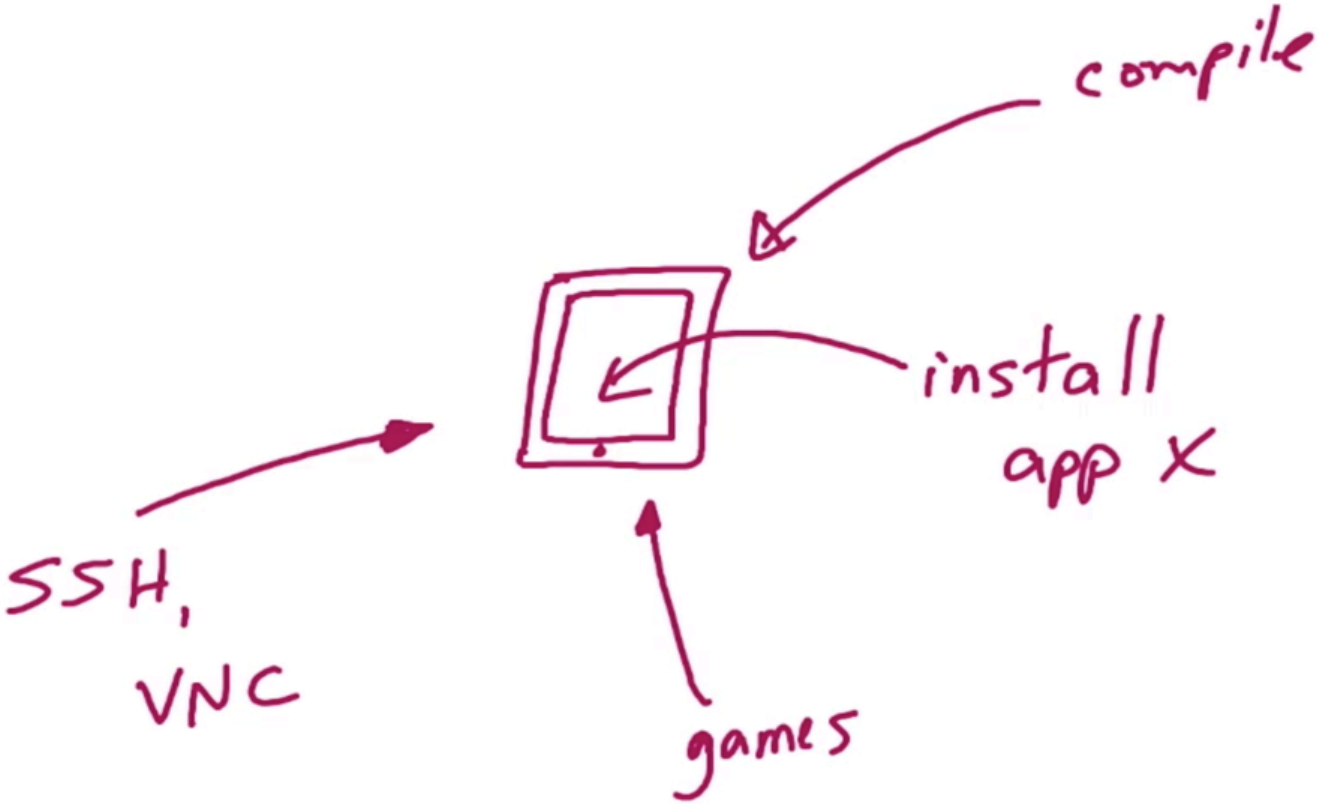
Apple iOS

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

Threats

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

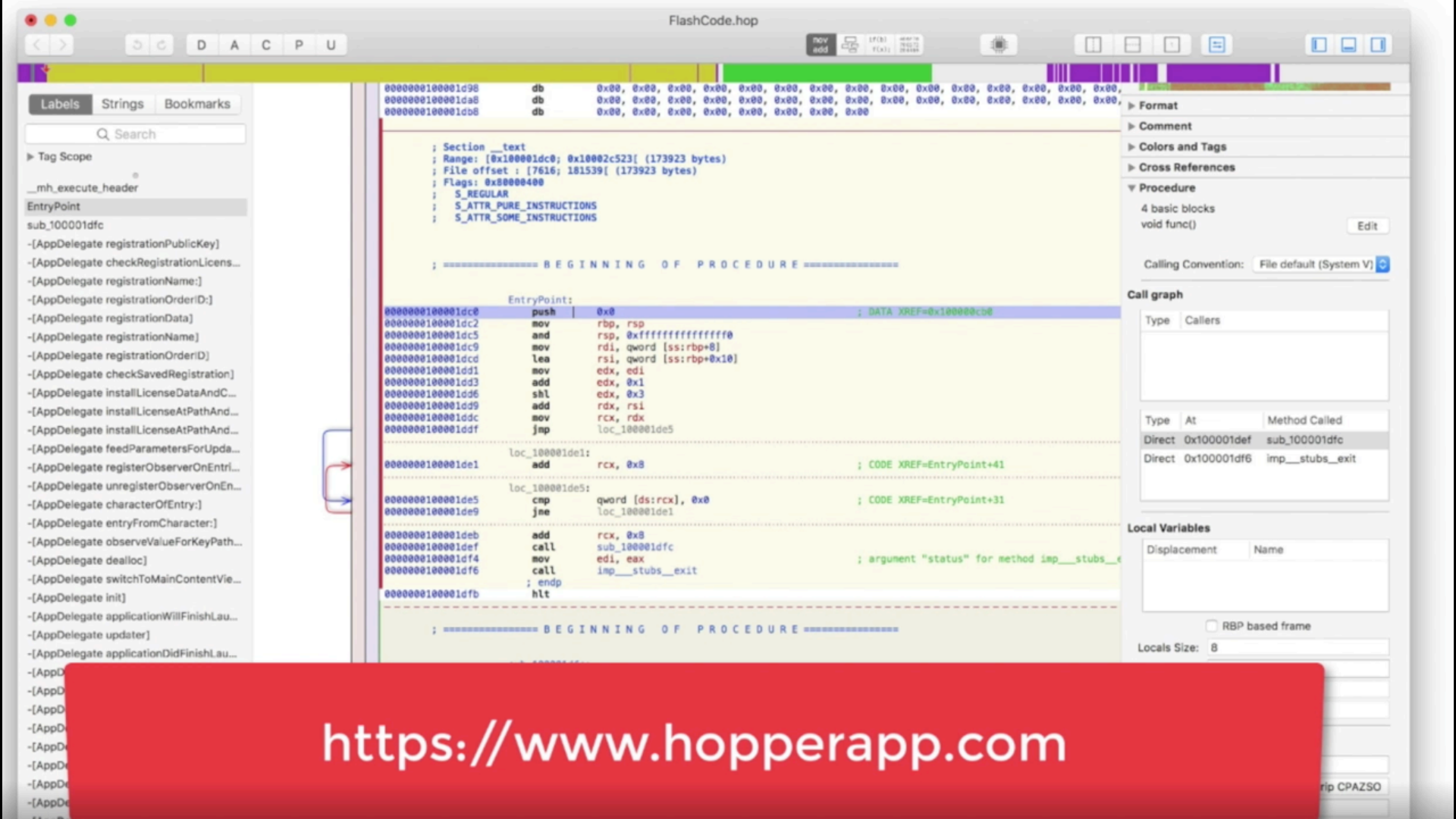
Jailbreak

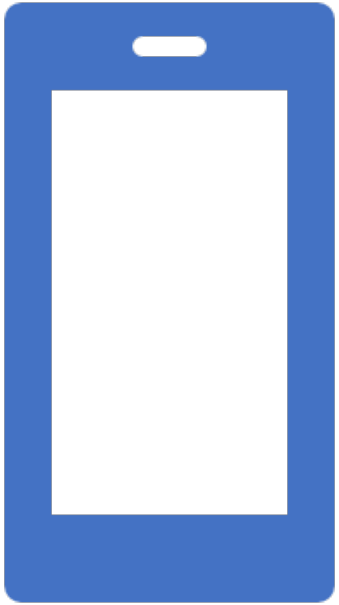


Jailbreaking is Harder, but not impossible...

iOS 11.x

LiberiOS


FlashCode.hop
D A C P U
Labels Strings Bookmarks
Search
Tag Scope
__mh_execute_header
EntryPoint
sub_100001dfc
-[AppDelegate registrationPublicKey]
-[AppDelegate checkRegistrationLicens...
-[AppDelegate registrationName:]
-[AppDelegate registrationOrderID:]
-[AppDelegate registrationData]
-[AppDelegate registrationName]
-[AppDelegate registrationOrderID]
-[AppDelegate checkSavedRegistration]
-[AppDelegate installLicenseDataAndC...
-[AppDelegate installLicenseAtPathAnd...
-[AppDelegate installLicenseAtPathAnd...
-[AppDelegate feedParametersForUpda...
-[AppDelegate registerObserverOnEntri...
-[AppDelegate unregisterObserverOnEn...
-[AppDelegate characterOfEntry:]
-[AppDelegate entryFromCharacter:]
-[AppDelegate observeValueForKeyPath...
-[AppDelegate dealloc]
-[AppDelegate switchToMainContentVie...
-[AppDelegate init]
-[AppDelegate applicationWillFinishLau...
-[AppDelegate updater]
-[AppDelegate applicationDidFinishLau...
0000000100001d98 db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0000000100001da8 db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0000000100001db8 db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
; Section __text
; Range: [0x100001dc0; 0x10002c523[(173923 bytes)
; File offset : [7616; 181539[(173923 bytes)
; Flags: 0x80000400
; S_REGULAR
; S_ATTR_PURE_INSTRUCTIONS
; S_ATTR_SOME_INSTRUCTIONS
; ========== B E G I N N I N G O F P R O C E D U R E ==========
EntryPoint:
0000000100001dc0 push 0x0 ; DATA XREF=0x100000cb0
0000000100001dc2 mov rbp, rsp
0000000100001dc5 and rsp, 0xfffffffffffffff0
0000000100001dc9 mov rdi, qword [ss:rbp+8]
0000000100001dcd lea rsi, qword [ss:rbp+0x10]
0000000100001dd1 mov edx, edi
0000000100001dd3 add edx, 0x1
0000000100001dd6 shl edx, 0x3
0000000100001dd9 add rdx, rsi
0000000100001ddc mov rcx, rdx
0000000100001ddf jmp loc_100001de5
loc_100001de1:
0000000100001de1 add rcx, 0x8 ; CODE XREF=EntryPoint+41
loc_100001de5:
0000000100001de5 cmp qword [ds:rcx], 0x0 ; CODE XREF=EntryPoint+31
0000000100001de9 jne loc_100001de1
0000000100001deb add rcx, 0x8
0000000100001def call sub_100001dfc
0000000100001df4 mov edi, eax ; argument "status" for method imp___stubs__
0000000100001df6 call imp___stubs__exit
; endp
0000000100001dfb hlt
; ========== B E G I N N I N G O F P R O C E D U R E ==========
Format
Comment
Colors and Tags
Cross References
Procedure
4 basic blocks
void func()
Edit
Calling Convention: File default (System V)
Call graph
Type Callers
Type At Method Called
Direct 0x100001def sub_100001dfc
Direct 0x100001df6 imp___stubs__exit
Local Variables
Displacement Name
RBP based frame
Locals Size: 8
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

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:

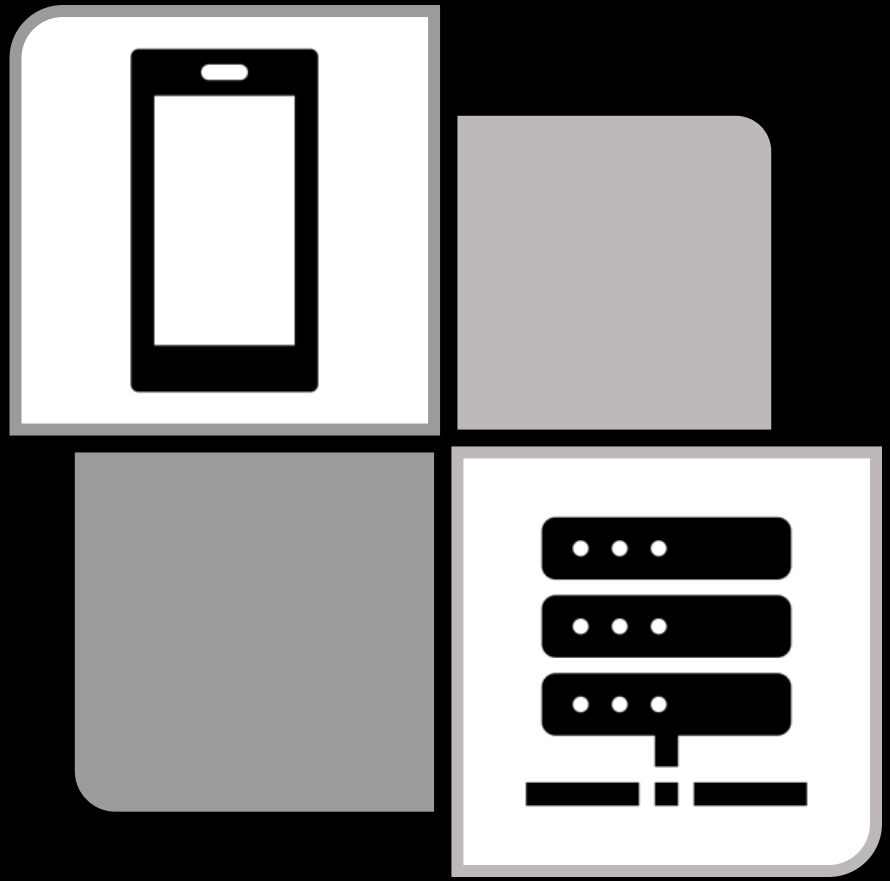
- **Google Play**: Google Play is a collection of services that allow users to discover, install, and purchase applications from their Android device or the web. Google Play makes it easy for developers to reach Android users and potential customers. Google Play also provides community review, application 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

Hacking Android
https://cehreview.com/go/hacking_android

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

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

Exercise 9

Additional XSS Exploitation

Bypassing Additional Web Application Flaws

Exercise 10

Fuzzing

Exercise 11


Thank You!