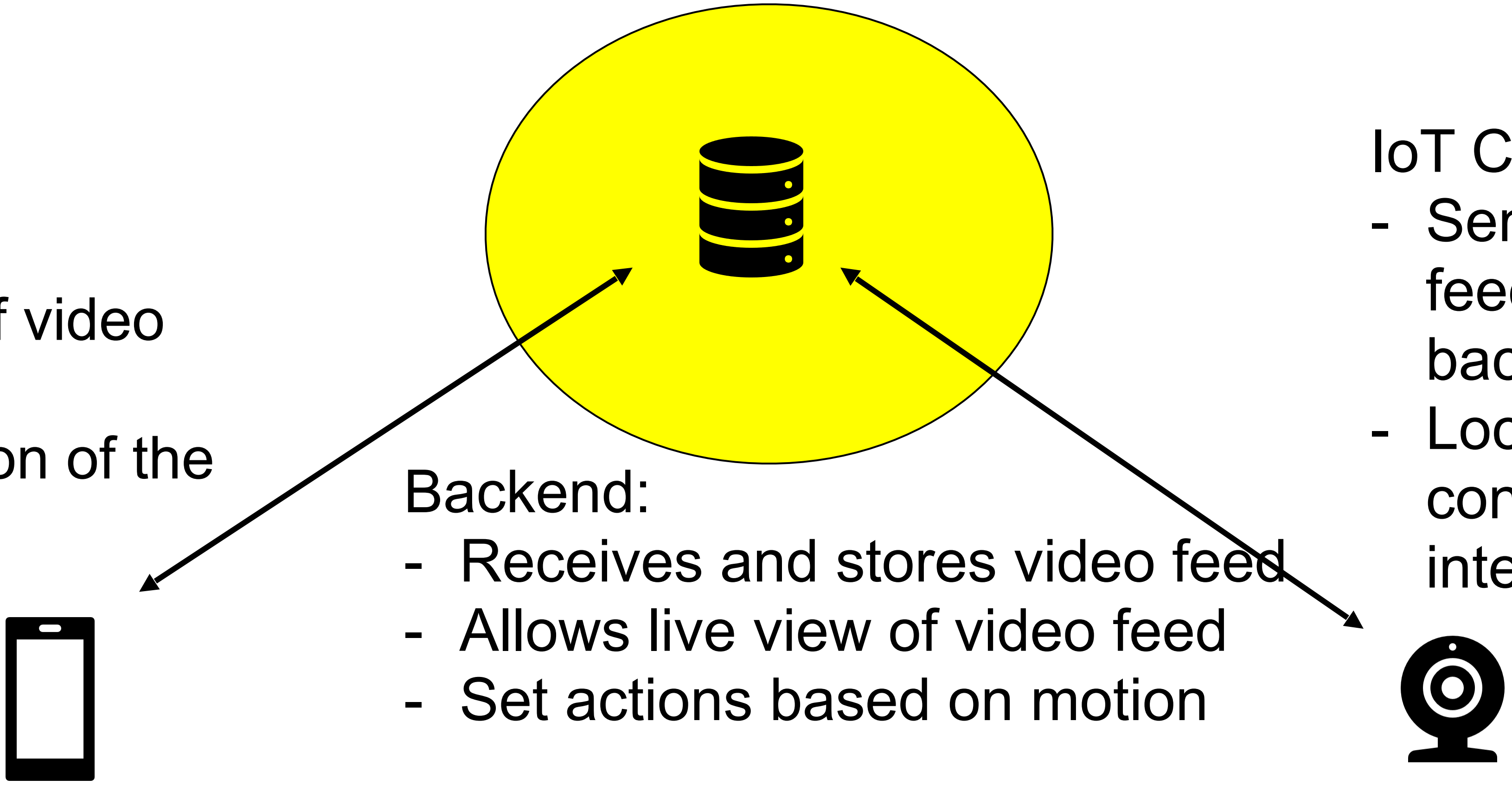
[w] www.yagoba.com, [m] info@yagoba.com
Beethovenstraße 20, 8010 Graz, Austria

Introduction to Penetration Testing

Web and Mobile Apps

Tomislav Nad

WHEN YOU NEED TO BE SURE



IoT Camera:
- Sends video feed to backend
- Local configuration interface

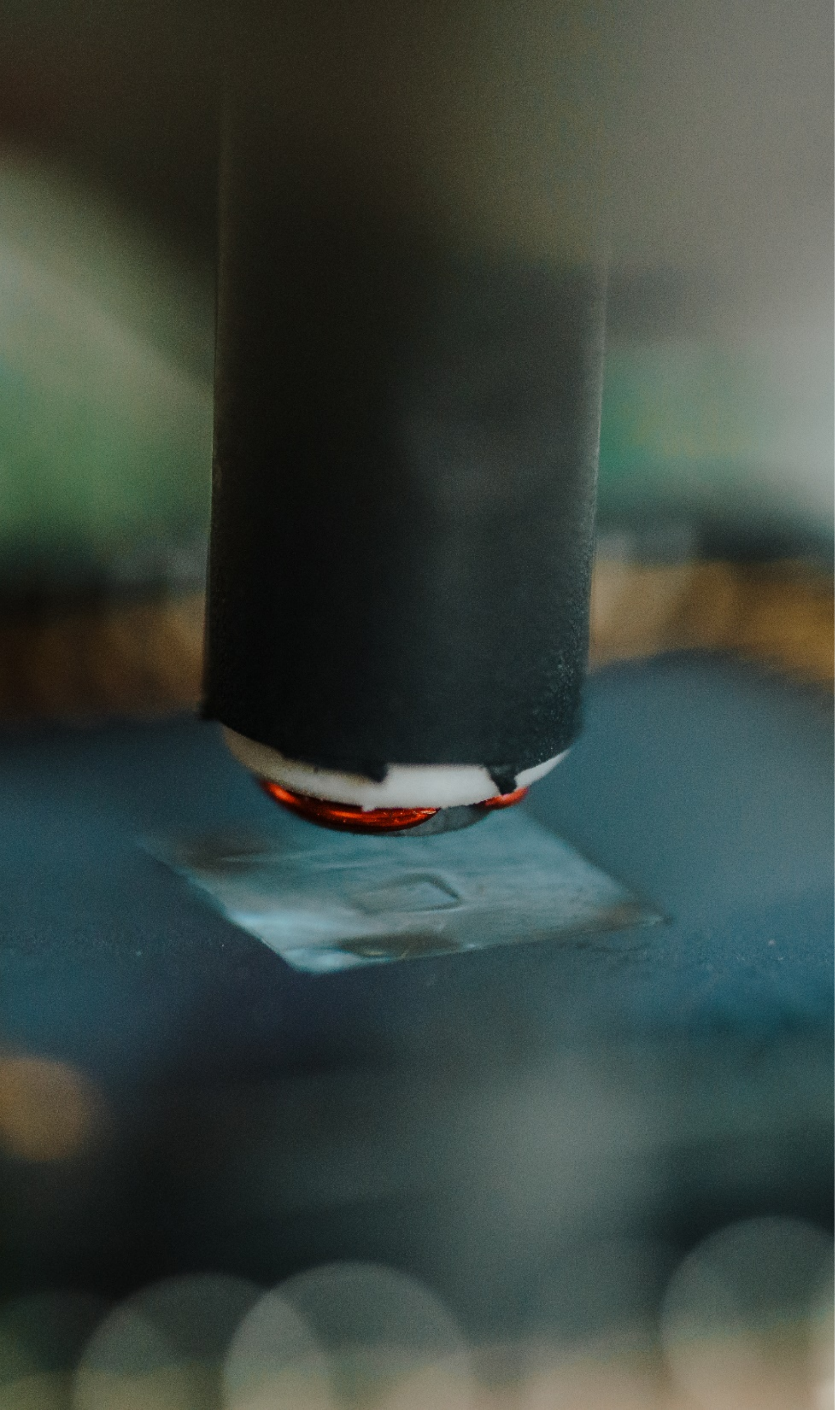
Mobile App:
- Live view of video feed
- Configuration of the backend

Backend:
- Receives and stores video feed
- Allows live view of video feed
- Set actions based on motion

THE STAGES OF A PENTEST

- Contract
  - Scoping
  - Non-Disclosure Agreement (NDA)
  - Permission to Attack (PTA)
- Discovery
  - Enumeration of life hosts
  - Enumeration of services
  - Initial ‘feel’
- Attack
  - Use discovered information to attack the target
- Reporting
  - Provide a report with enough details to reproduce your results

- You can find enormous amounts of small and big scripts to help you
  - This makes it hard to find the good ones
- Best to start with a pentesting Linux distro
  - Kali Linux (https://www.kali.org/)
  - Parrot OS (https://www.parrotsec.org/)
- Run them in a VM
  - Keeps your OS clutter free
  - Allows you to take snapshots
  - For certain tests, i.e. Wi-Fi you need to pass hardware through
- Look at the tools, that are already included
- Add your own as you go! ☺



The Contract

REMINDER

- During a penetration test you attack IT infrastructure like one of the bad guys!
- The only thing that differs is that you have permission to
- … so make sure to get this part right!

DIFFERENT TESTING APPROACHES

- Whitebox
  - „ Full knowledge“ or „open book“ test
  - Access to source code, documents, credentials, …
- Blackbox
  - „Zero-knowledge“ test
  - Simulate an „outsider“ attack
- Greybox
  - Everything between White- and Blackbox
  - Often only credentials
  - Most common kind of testing

THE CONTRACT

- Specify exactly what you will be doing
  - i.e. “Infrastructure scan using automated tools”
  - “Check for OWASP Top 10”
- Also specify what you won’t be doing
  - i.e. “Webapp tests during an infrastructure test”
- Specify the exact version of the target
  - Ask for a version freeze on the target you’ll be testing against
- Point out that a pentest does not reveal all vulnerabilities
  - Time constraints
  - For black box: not full visibility into the target

SCOPE CREEP

- Results from doing “that little bit extra” repeatedly
- I.e. during the engagement, the client asks “While you are at it, could you also do _____”
- DON’T do it!
- You may delay your initial project and miss things
- You may get into legal trouble
  - Remember, your activity is only covered by the contract
- Kindly reply to your client that you’ll gladly do it as part of another engagement

PERMISSION TO ATTACK

- Your ‘get out of jail free’-card
- Contains
  - The parties involved
  - The target (i.e. IP range, domain, web app, …)
  - Your source IP (get a static one!) for internet-based tests
  - The duration (make sure to extend this in case you need more time!)
- Due diligence
  - Check that the target belongs to the other party (i.e. ‘whois’ lookup for IP)
  - Check authority of other party

NON-DISCLOSURE AGREEMENT - NDA

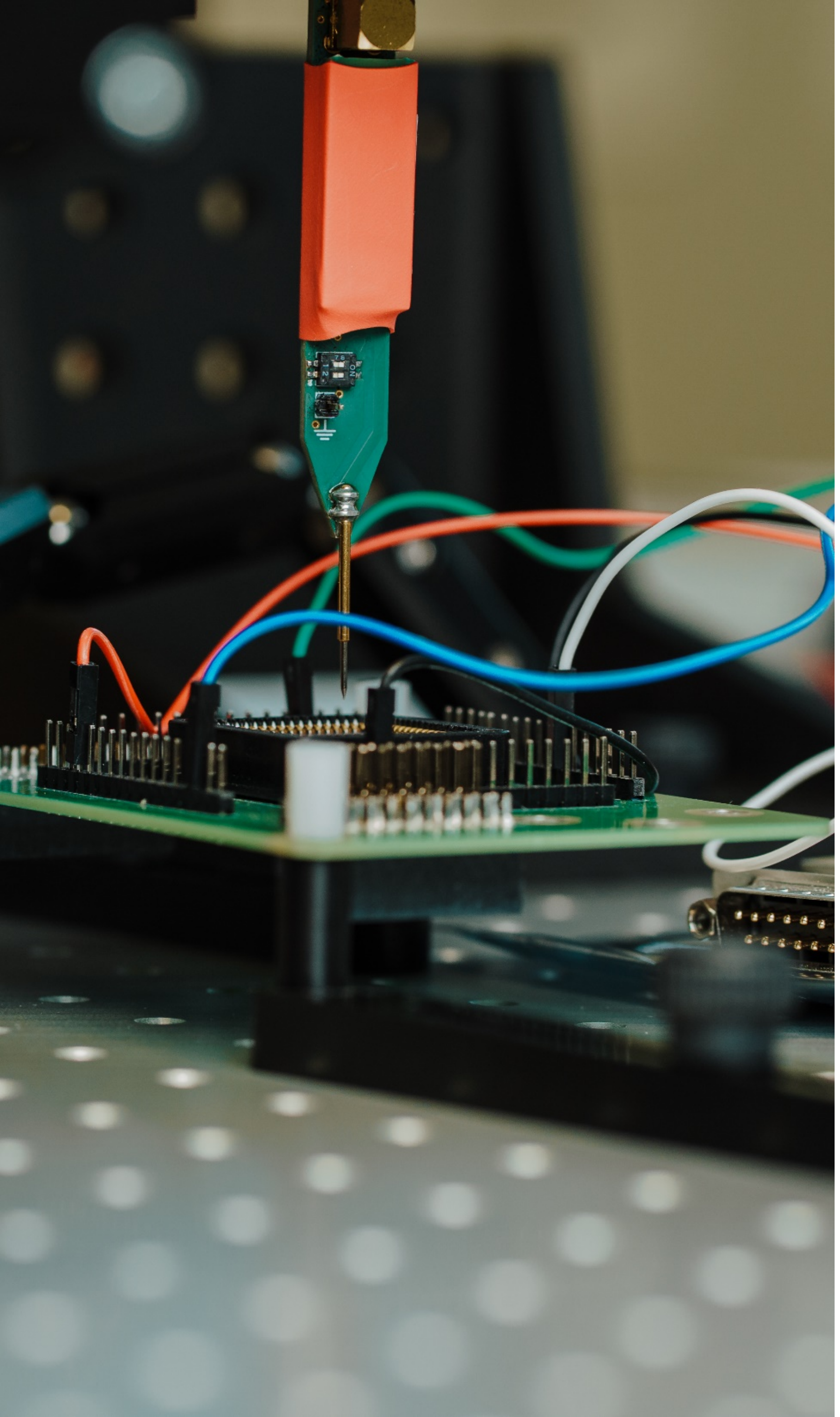
- Often asked for by clients
- Shall provide additional contractual security for company secrets
- Nothing to worry about, it’s standard

A WORD ON REDTEAMING

- The ‘cool’ way to do pentests these days
- Shall provide the most realistic results
- Combination of
  - Infrastructure test
  - Webapp test
  - Phishing
  - Social Engineering (also on-site)
- Usually the goal is to get a realistic understanding of the security of the company as a whole
- Done over an extended period (months)
- Black-box

THE “GET OUT OF JAIL FREE”-CARD

- Super important for red teaming engagements on-site
- A piece of paper, signed (ideally) by the highest person in the company
- States that you were hired
- (Always show a fake one first as an additional test if caught)



Discovery

WHERE TO START?

- If an IP range was given
  - Check for live hosts
  - Don’t rely on a ping sweep!
  - Do a full range TCP port scan on life hosts
- For a single web application
  - Start your attack proxy (i.e. Burp Suite)
    - Disable blocking traffic interception
  - Surf on the web application
  - Interact with it like a normal user
  - Get a ‘feel’ for what it’s like
  - Watch for hints of technologies being used

OPEN SOURCE INTELLIGENCE (OSINT)

- Gather ‘metadata’ about your target
- i.e. Wappalyzer Browser Extension to learn about technologies used
  - https://www.wappalyzer.com/download/
- Waybackmachine to analyze previous versions of the web app
  - May reveal formerly exposed websites
  - https://archive.org/web/
- Use shodan for generally interesting info about a target
  - Play around with it, but don’t get yourself into trouble
  - i.e. allows you to find unintentionally exposed surveillance cams
  - https://www.shodan.io/
- WHOIS info
  - May reveal interesting mail addresses that could be used for logins

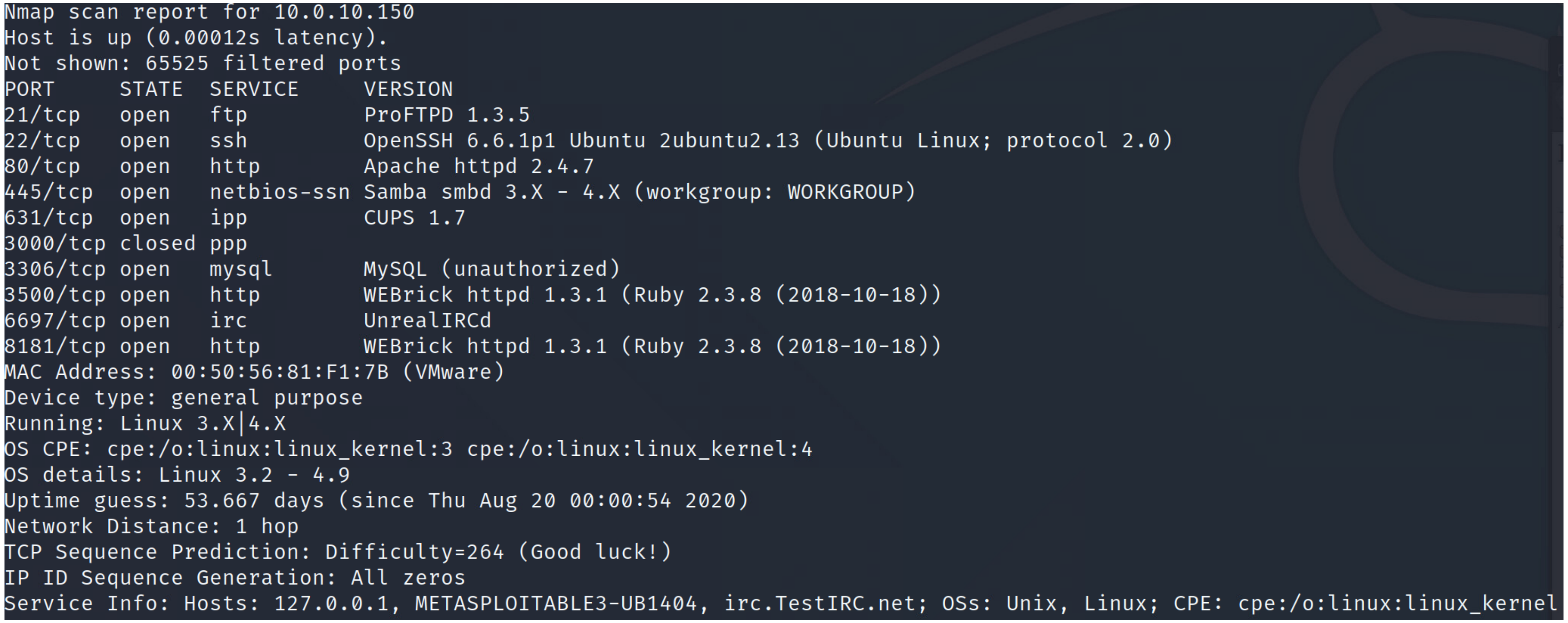
PORT SCANNING

- Most common tool: nmap
  - Command line interface
  - Zenmap for GUI frontend
- Turn on service discovery and version detection during full range port scan
- Gives you a list of potential targets per host
- Go for the ‘low ports’ (below 10 000) first
  - High ports have IANA assignments but oftentimes something homebrewed is running on them

NMAP SAMPLE COMMAND

- sudo nmap -p 1-65535 -sV -sS -T4 –O -oA testbed -v 10.0.10.1-255
  - -p .. Ports
  - -sV .. Service & Version detection
  - -sS .. SYN (Stealth) Scan
  - -T .. Timing Template, 4 is relatively aggressive
  - -O .. OS Detection
  - -oA .. Output to all available formats
  - -v .. Verbose output, so you see what‘s going on despite -oA

NMAP SAMPLE OUTPUT

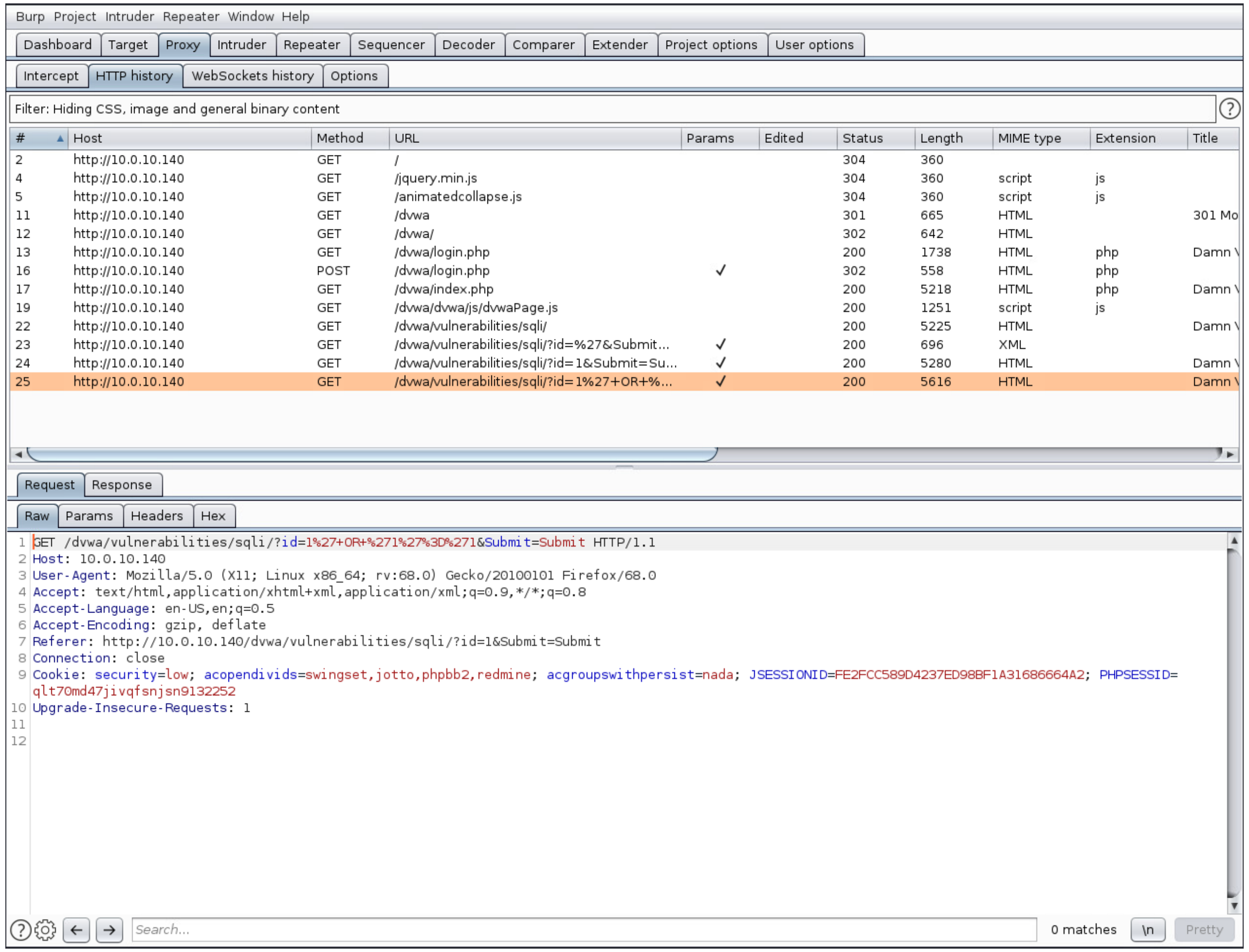


```
Nmap scan report for 10.0.10.150
Host is up (0.00012s latency).
Not shown: 65525 filtered ports
PORT      STATE  SERVICE     VERSION
21/tcp    open   ftp         ProFTPD 1.3.5
22/tcp    open   ssh         OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
80/tcp    open   http        Apache httpd 2.4.7
445/tcp   open   netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
631/tcp   open   ipp         CUPS 1.7
3000/tcp closed ppp
3306/tcp open   mysql       MySQL (unauthorized)
3500/tcp open   http        WEBrick httpd 1.3.1 (Ruby 2.3.8 (2018-10-18))
6697/tcp open   irc         UnrealIRCd
8181/tcp open   http        WEBrick httpd 1.3.1 (Ruby 2.3.8 (2018-10-18))
MAC Address: 00:50:56:81:F1:7B (VMware)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Uptime guess: 53.667 days (since Thu Aug 20 00:00:54 2020)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=264 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: Hosts: 127.0.0.1, METASPLOITABLE3-UB1404, irc.TestIRC.net; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

WEBSITE DISCOVERY

- Always have an attack proxy running
  - Burp Suite, OWASP ZAP, …
- Commercial ones point out flaws by analyzing traffic
- Analyzing requests and responses may give hints
  - Basic misconfigurations like missing cookie flags
  - Reverse-Proxy setups
  - Technologies (i.e. ASP.NET)

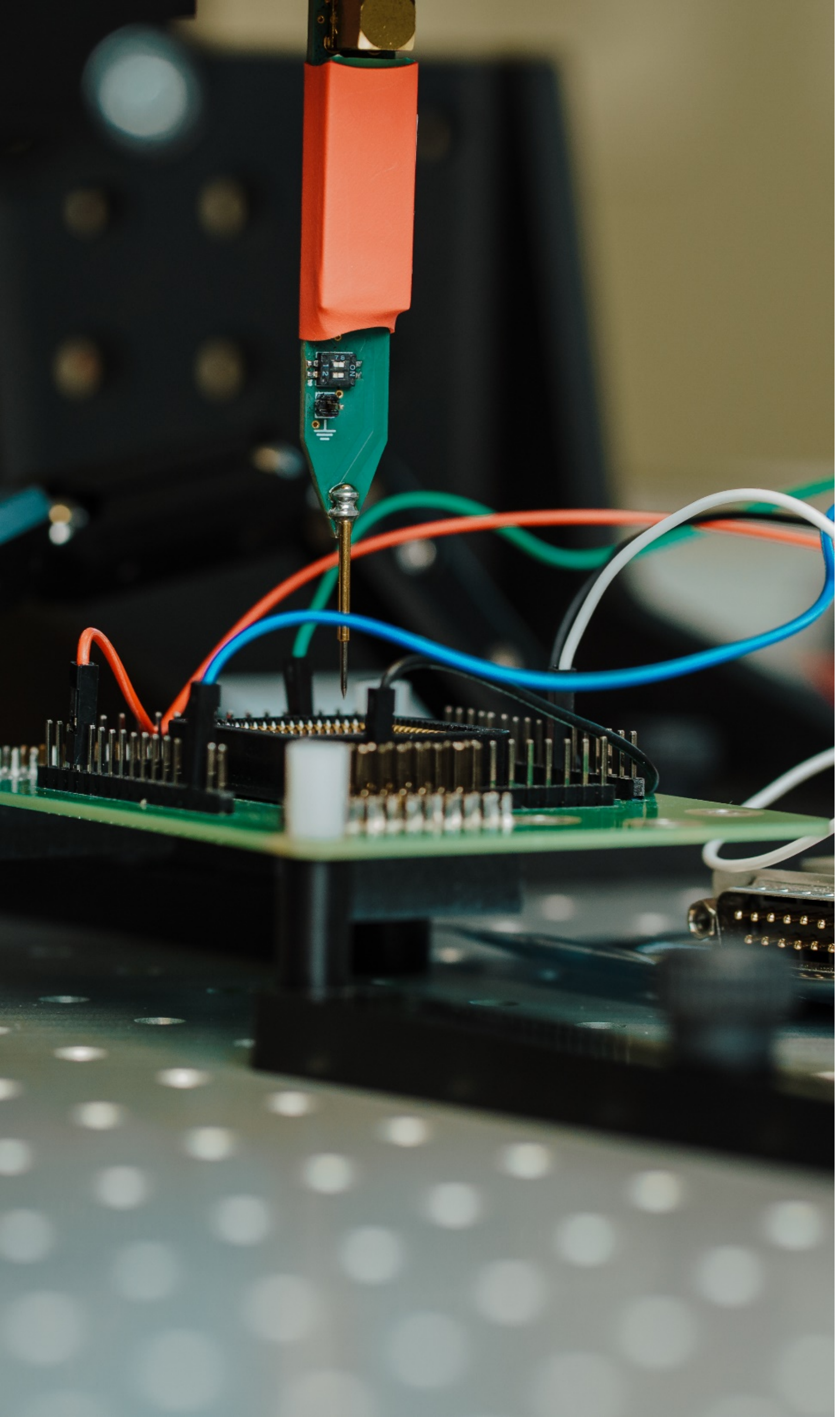
BURP SUITE SAMPLE


Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Intercept HTTP history WebSockets history Options

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title
2	http://10.0.10.140	GET	/			304	360			
4	http://10.0.10.140	GET	/jquery.min.js			304	360	script	js	
5	http://10.0.10.140	GET	/animatedcollapse.js			304	360	script	js	
11	http://10.0.10.140	GET	/dvwa			301	665	HTML		301 Mo
12	http://10.0.10.140	GET	/dvwa/			302	642	HTML		
13	http://10.0.10.140	GET	/dvwa/login.php			200	1738	HTML	php	Damn \
16	http://10.0.10.140	POST	/dvwa/login.php	✓		302	558	HTML	php	
17	http://10.0.10.140	GET	/dvwa/index.php			200	5218	HTML	php	Damn \
19	http://10.0.10.140	GET	/dvwa/dvwa/js/dvwaPage.js			200	1251	script	js	
22	http://10.0.10.140	GET	/dvwa/vulnerabilities/sqli/			200	5225	HTML		Damn \
23	http://10.0.10.140	GET	/dvwa/vulnerabilities/sqli/?id=%27&Submit...	✓		200	696	XML		
24	http://10.0.10.140	GET	/dvwa/vulnerabilities/sqli/?id=1&Submit=Su...	✓		200	5280	HTML		Damn \
25	http://10.0.10.140	GET	/dvwa/vulnerabilities/sqli/?id=1%27+OR+%...	✓		200	5616	HTML		Damn \

```
GET /dvwa/vulnerabilities/sqli/?id=1%27+OR+%271%27%3D%271&Submit=Submit HTTP/1.1
Host: 10.0.10.140
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.0.10.140/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit
Connection: close
Cookie: security=low; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada; JSESSIONID=FE2FCC589D4237ED98BF1A31686664A2; PHPSESSID=
qlt70md47jivqfsnjsn9132252
Upgrade-Insecure-Requests: 1
```

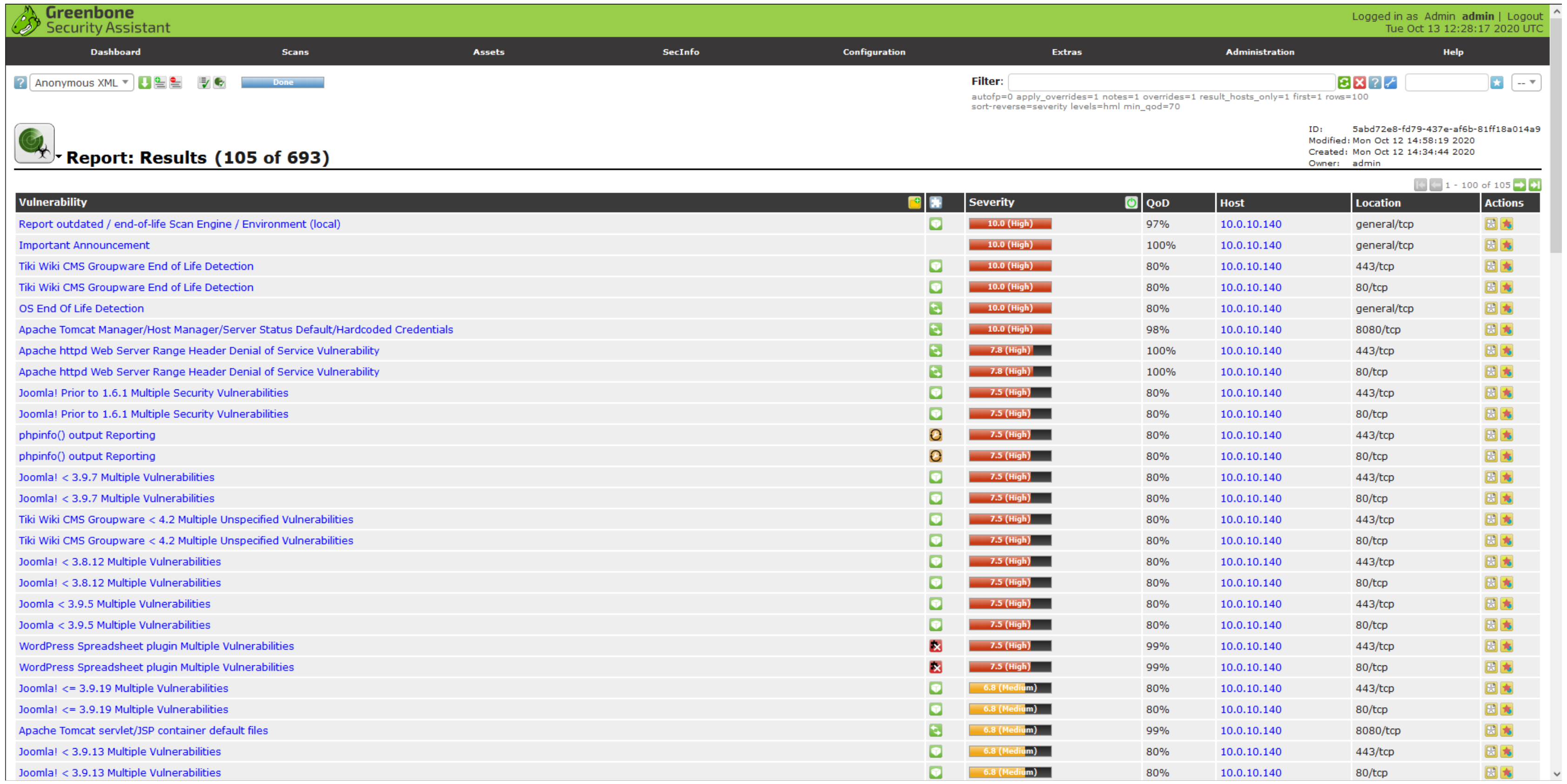


Attack Phase

AUTOMATED SCANNERS

- Great for getting the big picture of a large IP range
  - Do the discovery and port scan for you
- Most well known:
  - Nessus (commercial)
  - OpenVAS (FLOSS, fork from Nessus)
- Regularly updated with most recent exploits
- Can be run on a schedule to monitor security development
- Not suitable for web apps

OPENVAS SAMPLE



Greenbone Security Assistant

Logged in as Admin **admin** | Logout
Tue Oct 13 12:28:17 2020 UTC

Dashboard | Scans | Assets | SecInfo | Configuration | Extras | Administration | Help

Anonymous XML | Done

Filter:
autofp=0 apply_overrides=1 notes=1 overrides=1 result_hosts_only=1 first=1 rows=100
sort-reverse=severity levels=hml min_qod=70

Report: Results (105 of 693)

```
ID:       5abd72e8-fd79-437e-af6b-81ff18a014a9
Modified: Mon Oct 12 14:58:19 2020
Created:  Mon Oct 12 14:34:44 2020
Owner:    admin
```

1 - 100 of 105

Vulnerability		Severity	QoD	Host	Location	Actions
Report outdated / end-of-life Scan Engine / Environment (local)		10.0 (High)	97%	10.0.10.140	general/tcp	
Important Announcement		10.0 (High)	100%	10.0.10.140	general/tcp	
Tiki Wiki CMS Groupware End of Life Detection		10.0 (High)	80%	10.0.10.140	443/tcp	
Tiki Wiki CMS Groupware End of Life Detection		10.0 (High)	80%	10.0.10.140	80/tcp	
OS End Of Life Detection		10.0 (High)	80%	10.0.10.140	general/tcp	
Apache Tomcat Manager/Host Manager/Server Status Default/Hardcoded Credentials		10.0 (High)	98%	10.0.10.140	8080/tcp	
Apache httpd Web Server Range Header Denial of Service Vulnerability		7.8 (High)	100%	10.0.10.140	443/tcp	
Apache httpd Web Server Range Header Denial of Service Vulnerability		7.8 (High)	100%	10.0.10.140	80/tcp	
Joomla! Prior to 1.6.1 Multiple Security Vulnerabilities		7.5 (High)	80%	10.0.10.140	443/tcp	
Joomla! Prior to 1.6.1 Multiple Security Vulnerabilities		7.5 (High)	80%	10.0.10.140	80/tcp	
phpinfo() output Reporting		7.5 (High)	80%	10.0.10.140	443/tcp	
phpinfo() output Reporting		7.5 (High)	80%	10.0.10.140	80/tcp	
Joomla! < 3.9.7 Multiple Vulnerabilities		7.5 (High)	80%	10.0.10.140	443/tcp	
Joomla! < 3.9.7 Multiple Vulnerabilities		7.5 (High)	80%	10.0.10.140	80/tcp	
Tiki Wiki CMS Groupware < 4.2 Multiple Unspecified Vulnerabilities		7.5 (High)	80%	10.0.10.140	443/tcp	
Tiki Wiki CMS Groupware < 4.2 Multiple Unspecified Vulnerabilities		7.5 (High)	80%	10.0.10.140	80/tcp	
Joomla! < 3.8.12 Multiple Vulnerabilities		7.5 (High)	80%	10.0.10.140	443/tcp	
Joomla! < 3.8.12 Multiple Vulnerabilities		7.5 (High)	80%	10.0.10.140	80/tcp	
Joomla < 3.9.5 Multiple Vulnerabilities		7.5 (High)	80%	10.0.10.140	443/tcp	
Joomla < 3.9.5 Multiple Vulnerabilities		7.5 (High)	80%	10.0.10.140	80/tcp	
WordPress Spreadsheet plugin Multiple Vulnerabilities		7.5 (High)	99%	10.0.10.140	443/tcp	
WordPress Spreadsheet plugin Multiple Vulnerabilities		7.5 (High)	99%	10.0.10.140	80/tcp	
Joomla! <= 3.9.19 Multiple Vulnerabilities		6.8 (Medium)	80%	10.0.10.140	443/tcp	
Joomla! <= 3.9.19 Multiple Vulnerabilities		6.8 (Medium)	80%	10.0.10.140	80/tcp	
Apache Tomcat servlet/JSP container default files		6.8 (Medium)	99%	10.0.10.140	8080/tcp	
Joomla! < 3.9.13 Multiple Vulnerabilities		6.8 (Medium)	80%	10.0.10.140	443/tcp	
Joomla! < 3.9.13 Multiple Vulnerabilities		6.8 (Medium)	80%	10.0.10.140	80/tcp	

OWASP

- Open Web Application Security Project
- Provide Top 10 lists of most common vulnerabilities
  - Web Apps
  - IoT
- … and helpful cheatsheets/guides
  - https://cheatsheetseries.owasp.org/
- Creators of ZAP attack proxy

SAMPLES

- Samples taken from
  - DVWA
  - WebGoat
- Great entry level learning platforms for web hacking
- Get them and many more from OWASP Broken Web Applications
  - https://owasp.org/www-project-broken-web-applications/

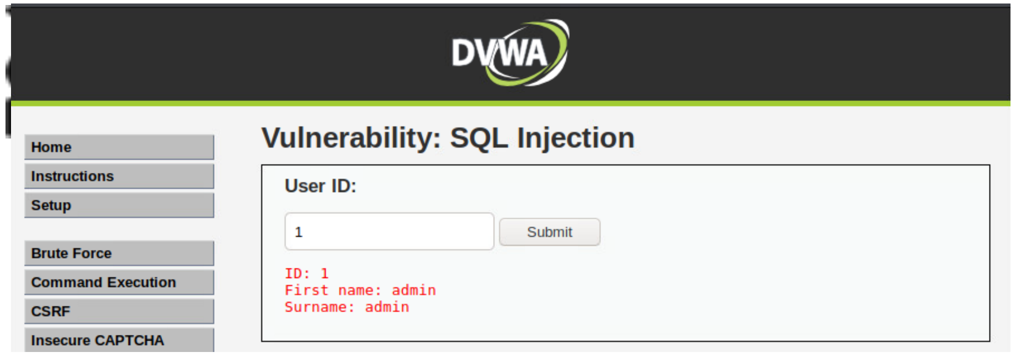
OWASP (WEB) TOP 10

1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities
5. Broken Access Control
6. Security Misconfiguration
7. Cross-Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components with Known Vulnerabilities
10. Insufficient Logging and Monitoring

1. INJECTION

- Place code where it is not expected
- Any data input might be affected
- Refers to : SQL, NoSQL, Command, LDAP, …
- SQL example:
  - 1’ or ‘1’=‘1
- Command example:
  - i.e. in a ping command: 1.2.3.4; ls –l
- Prevention: Input sanitization

SQLI SAMPLE

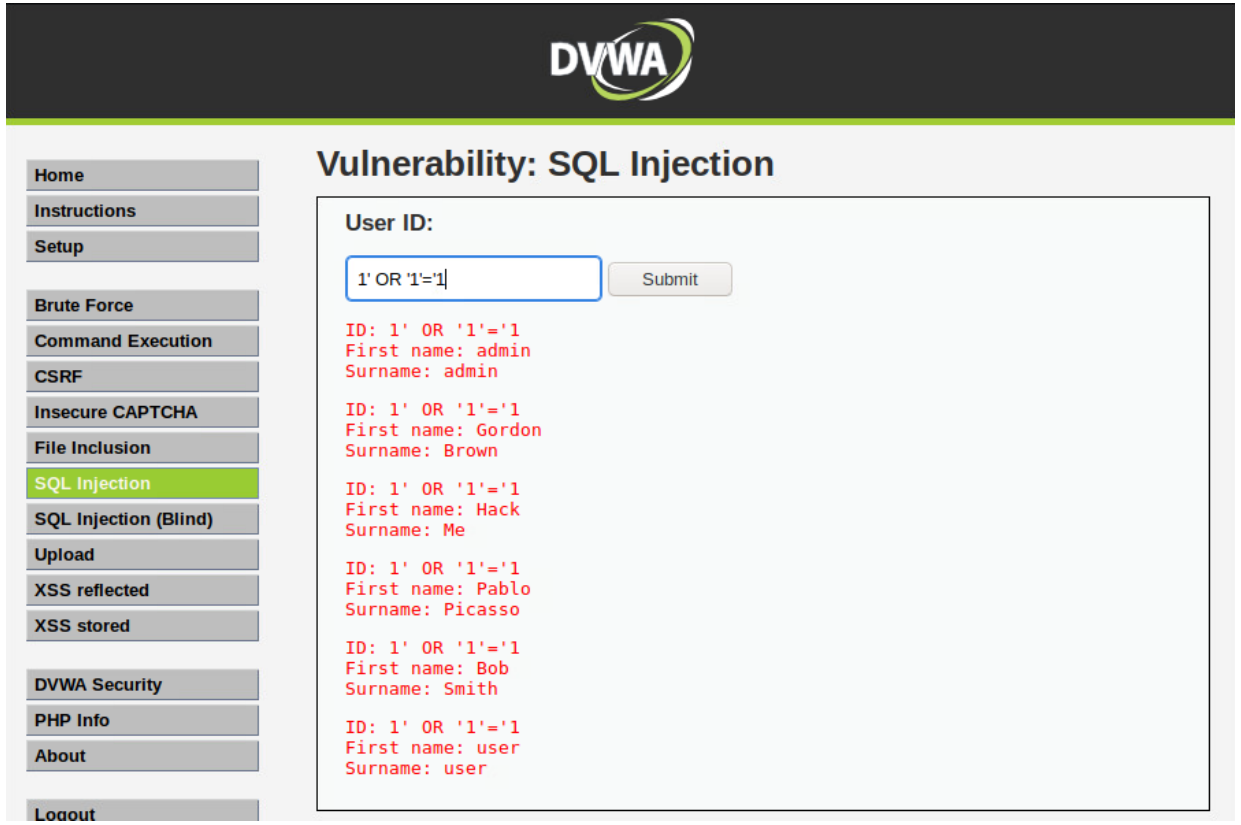


Vulnerability: SQL Injection

User ID:

1

```
ID: 1
First name: admin
Surname: admin
```



Vulnerability: SQL Injection

User ID:

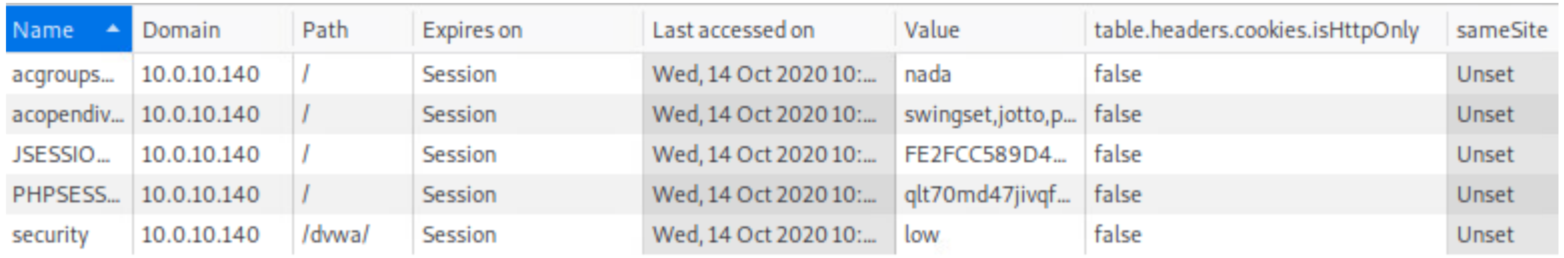
1' OR '1'='1

```
ID: 1' OR '1'='1
First name: admin
Surname: admin

ID: 1' OR '1'='1
First name: Gordon
Surname: Brown

ID: 1' OR '1'='1
First name: Hack
Surname: Me

ID: 1' OR '1'='1
First name: Pablo
Surname: Picasso

ID: 1' OR '1'='1
First name: Bob
Surname: Smith

ID: 1' OR '1'='1
First name: user
Surname: user
```

2. BROKEN AUTHENTICATION

- Umbrella term for mistakes that facilitate getting into the web app
  - Allowing brute-force attacks on password fields
  - Allowing weak passwords/usernames (admin/admin)
  - “Knowledge-based answers” for password recovery
  - Bad session ID generation
  - Improper session ID invalidation
- Various attacks:
  - Brute-force tools
  - Observation (i.e. does the session ID change?)
  - Cookie re-using
- Prevention: strongly depends
  - If frameworks are used, make sure to use built-in features

COOKIES DONE WRONG



Name	Domain	Path	Expires on	Last accessed on	Value	table.headers.cookies.isHttpOnly	sameSite
acgroups...	10.0.10.140	/	Session	Wed, 14 Oct 2020 10:...	nada	false	Unset
acopendiv...	10.0.10.140	/	Session	Wed, 14 Oct 2020 10:...	swingset,jotto,p...	false	Unset
JSESSIO...	10.0.10.140	/	Session	Wed, 14 Oct 2020 10:...	FE2FCC589D4...	false	Unset
PHPSESS...	10.0.10.140	/	Session	Wed, 14 Oct 2020 10:...	qlt70md47jivqf...	false	Unset
security	10.0.10.140	/dvwa/	Session	Wed, 14 Oct 2020 10:...	low	false	Unset

- Security value stored in cookie
- HttpOnly flag set to false
  - Enforces cookie to be sent via HTTPS
- sameSite
  - Enforces cookie to only be used on one Domain
  - Helps defend against cookie stealing via XSS

3. SENSITIVE DATA EXPOSURE

- Multiple attacks stemming from lack of data encryption
  - Plain text data transmission (i.e. FTP)
  - Weak cryptography
  - No encryption at rest
- Various attacks
  - Man-in-the-Middle to grab data in transit
  - Offline cracking of weak encryption
  - Obtaining access data through another vulnerability
- Prevention: get your crypto together
  - Identify what must be encrypted (i.e. passwords) and find an appropriate solution (i.e. PBKDF2)
  - Make sure that all data in transit is encrypted, always! (No excuses, we've got "Let's encrypt!")
  - Only use secure protocol implementations (SSH, FTPS, IMAPS, …)

4. XML EXTERNAL ENTITY (XXE)

- Attack against XML processors
- Applies whenever XML can be sent to the web application
- Example:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<foo>&xxe;</foo>
```

  - Extracts the contents of /etc/passwd
- Prevention: Disable External Entities in XML processor
  - Make sure it is up to date

5. BROKEN ACCESS CONTROL

- Attacks allowing bypass of access restrictions
  - May affect APIs and Web Apps
- Attacks
  - Modification of URL or application state
  - Elevation of privilege, i.e. acting as a logged-in user when not being logged-in
  - Metadata manipulation like replaying JSON Web Tokens (JWT)
  - Force browsing to authenticated pages
- Generally requires some background knowledge of web app
- Prevention: ‘Deny by default’ mindset
  - Disable directory listing
  - Invalidation of JWT tokens
  - Implement access control once and re-use it throughout the web application

6. SECURITY MISCONFIGURATION

- A somewhat oversized umbrella
  - Lack of security hardening
  - Unnecessary features enabled
  - Default accounts (and passwords)
  - Bad error handling
  - Disabled security features
  - …
- Attacks: highly dependent :/
- Defense: same :\

7. CROSS SITE SCRIPTING (XSS)

- Insertion of scripting code that gets returned to other users
- 3 Types:
  - Reflected: temporarily generated by crafting i.e. a malicious link
  - Stored: Code that is saved in the web application and echoed back to many
  - DOM-based: Dynamic inclusion of attacker controllable data
    - Extremely rare in the wild
    - Few scanners detect it automatically (Burp Suite Pro does)
- Most basic example: <script>alert(1)</script>
  - Many cheat sheets on the net with more advanced vectors
- Defense: Output sanitization
  - You can start with the input but more importantly whatever the web app displays must be sanitized

8. INSECURE DESERIALIZATION

- Applies whenever objects (i.e. in the object oriented programming sense) are transferred between programs
- Example: Use of a ‘super’ cookie to capture program state:

```
a:4:{i:0;i:132;i:1;s:7:"Mallory";i:2;s:4:"user";
i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";}
```

  could be changed to

```
a:4:{i:0;i:1;i:1;s:5:"Alice";i:2;s:5:"admin";
i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";}
```

  for admin privileges

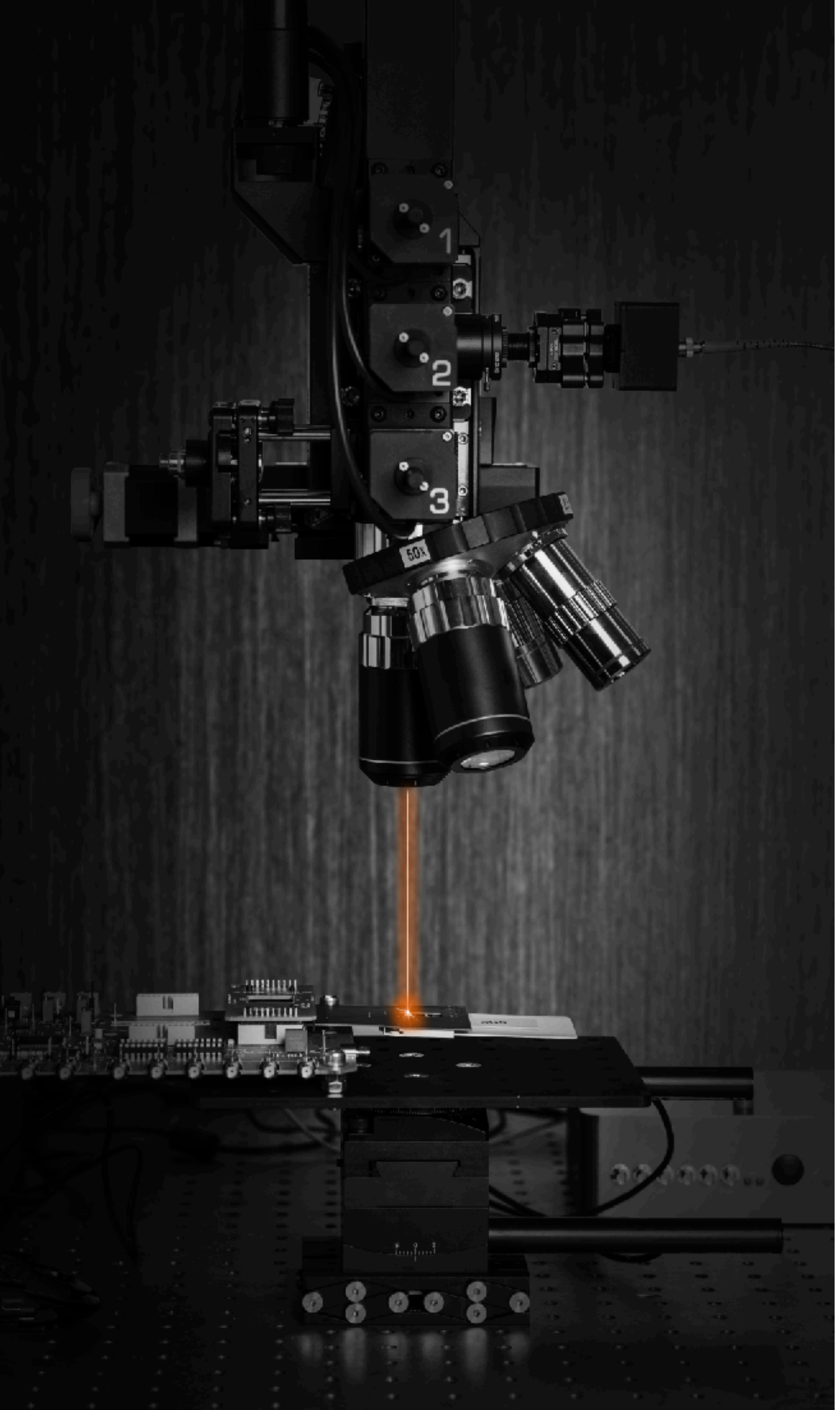
- Hard to find in general
- Defense: avoid, only allow primitive data types, use integrity checks

9. USING COMPONENTS WITH KNOWN VULNERABILITIES

- Many reasons:
  - lack of security consciousness
  - Lack of knowledge which versions of software are used
  - staying on an old version due to changes in a new version
  - Impossibility to upgrade in an IoT device
- Attacks:
  - Infrastructure scanners (i.e. OpenVAS)
  - Banner grabbing
  - Just looking at displayed version numbers in web apps
- Defense:
  - Proper development process
  - Applying patches regularly

10. INSUFFICIENT LOGGING & MONITORING

- Two main causes
  - Not enough monitoring: important events are not recorded/alerted
  - Too much monitoring: important events are drowned out by unimportant ones
- Rarely tested directly in a pentest
  - Attack tools are usually configured for time efficiency, not stealth
  - Exceptions in defenses made for pentesters
- Defense
  - Centralized logging
  - Understand logging capabilities of infrastructure
  - Log a bit too much
  - … but make sure to apply filters



Mobile App Testing

WHY TEST MOBILE APPS?

- Smartphones store sensitive data and perform sensitive actions
  - Communication (Messengers, Mail, …)
  - Financial transactions (Online banking)
  - Passwords, keys (Password managers)
  - Health information
  - …
- Smartphones have many sensors and interfaces (sensitive data / attack surface)
  - Camera
  - GPS
  - WiFi
  - …

DIFFERENT TESTING METHODS

- Static Analysis
  - Non-runtime environment (no execution of the program)
  - Examining application binaries (apk, ipa)
  - Examining source code (if available)
  - Reverse engineering
- Dynamic Analysis
  - Runtime environment (program is executed)
  - Inspection and manipulation during runtime
  - Inspection of network traffic
  - Inspection of results after execution

TOOLS

- Mobexler
  - Dedicated VM image for mobile application penetration testing
  - A lot of useful tools preinstalled (Android & iOS Zone)
- MobSF
  - Automated mobile application penetration testing framework
  - Static & Dynamic Analysis (Android & iOS)
  - API
- Frida
  - Dynamic instrumentation framework (Android & iOS)
- Other Tools
  - Similar tools for other penetration testing tasks (E.g., tools from Kali)

ROOT / JAILBREAK

- Get access to device on root level
- Useful for getting a better understanding of the app
- Access to filesystem
- Additional tools and modules
- Android
  - Magisk (All versions, unlock bootloader, e.g., by TWRP)
  - Others
- iOS
  - checkra1n (A7-A11-equipped iPhones and iPads)
  - unc0ver (iOS from 11 to 14)

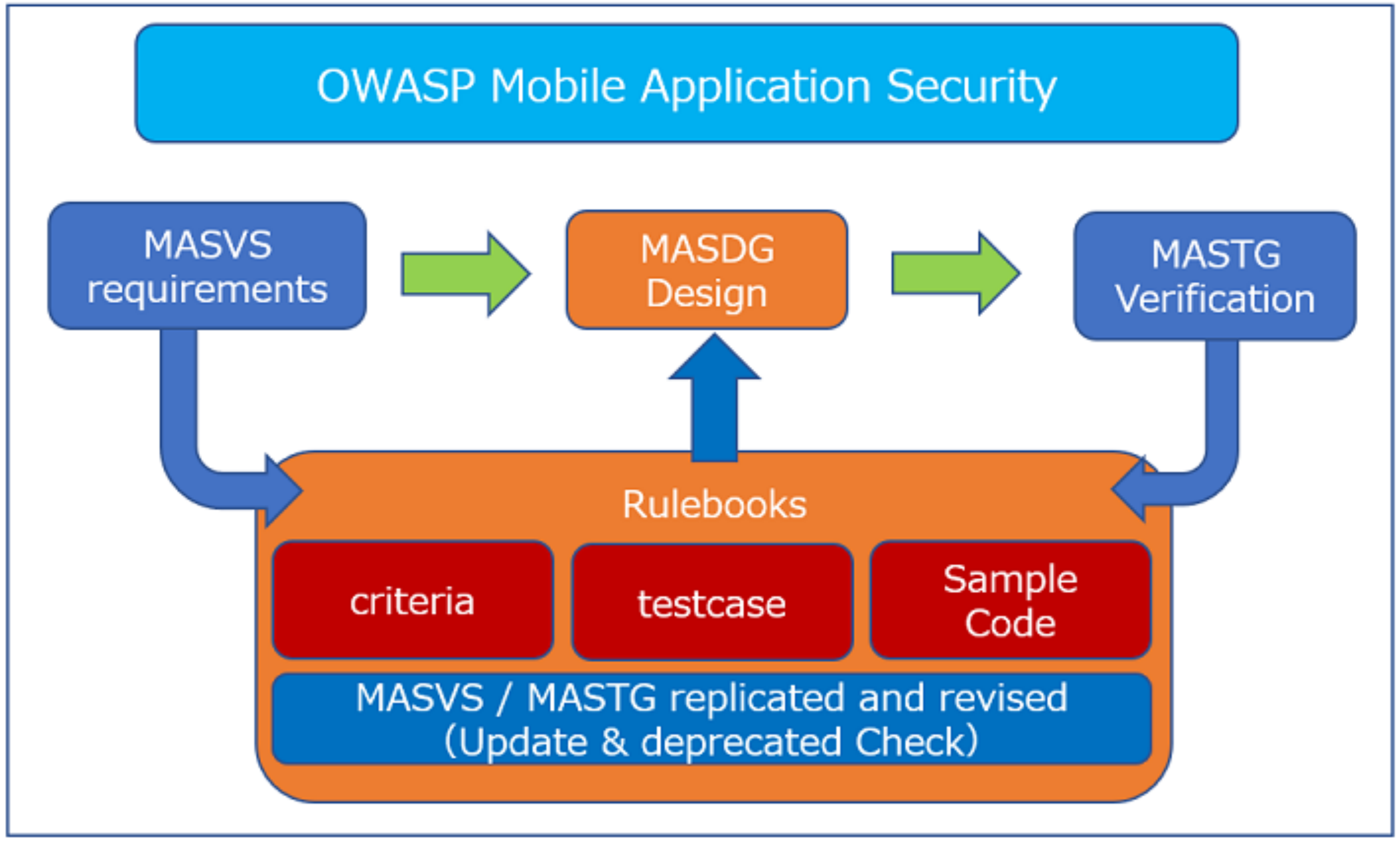
OWASP Mobile Application Security

OWASP MOBILE APPLICATION SECURITY

https://mas.owasp.org/

- OWASP MAS Checklist: https://mas.owasp.org/checklists/
  - The OWASP Mobile Application Security Checklist contains links to the MASTG test cases for each MASVS control
- MASVS: https://mas.owasp.org/MASVS/
  - The OWASP Mobile Application Security Verification Standard (MASVS) is a standard for mobile app security. It can be used by mobile software architects and developers seeking to develop secure mobile applications, as well as security testers to ensure completeness and consistency of test results.
- MASTG: https://mas.owasp.org/MASTG/
  - The OWASP Mobile Application Security Testing Guide (MASTG) is a comprehensive manual for testing the security of mobile apps. It describes processes and techniques for verifying the requirements listed in the Mobile Application Security Verification Standard (MASVS), and provides a baseline for complete and consistent security tests.

OWASP MASVS & MASTG (2)


OWASP Mobile Application Security
MASVS requirements
MASDG Design
MASTG Verification
Rulebooks
criteria
testcase
Sample Code
MASVS / MASTG replicated and revised (Update & deprecated Check)

https://owasp.org/www-project-mobile-application-security-design-guide/

CHAPTERS OF MASVS

MASVS-STORAGE: Storage

- Mobile apps handle sensitive data like personally identifiable information (PII), cryptographic material, secrets, and API keys.
- Sensitive data is stored in private (app's internal storage) or public locations (folders accessible by users or other apps).
- Unintentional exposure of sensitive data can occur in public locations due to misuse of APIs or system capabilities like backups or logs.
- The goal is to help developers protect any sensitive data stored by the app, intentionally or unintentionally.
- Emphasis on safeguarding against unintentional leaks due to improper API or system capabilities usage.

CHAPTERS OF MASVS

MASVS-CRYPTO: Cryptography

- Apply cryptographic mechanisms correctly
- No hardcoded keys
- Proven / well-known cryptographic mechanisms / primitives
- Follow best practices
- Different keys for different purposes

CHAPTERS OF MASVS

MASVS-AUTH: Authentication and Authorization

- Login to remote services
- User authorization and authentication must be in place
- Remote session termination
- Password policy
- Two-factor authentication if required

MASVS-NETWORK: Network Communication

- Protect confidentiality and integrity (TLS)
- Follow best practices
- X.509 certificate verification

CHAPTERS OF MASVS

MASVS-PLATFORM: Platform Interaction

- Mobile app security is influenced by interactions with the mobile platform, involving data or functionality exposure through IPC mechanisms and WebViews.
- These IPC mechanisms and WebViews can be exploited, risking the app's security.
- Sensitive data like passwords and credit card details in the app's UI can be unintentionally leaked through auto-screenshots or visible to others.
- Controls are needed for secure interaction with the mobile platform, including secure IPC use, WebView configurations, and safe data display in the UI.
- Implementing these controls helps protect sensitive user information and prevent unauthorized access by attackers.

CHAPTERS OF MASVS

MASVS-CODE: Code Quality

- Security coding practices are followed (compiler features)
  - Stack protection
- App is signed and private key is properly protected
- Release build → debugging symbols removed
- No dead code
- Proper error and exception handling
- Proper memory allocation in unmanaged code

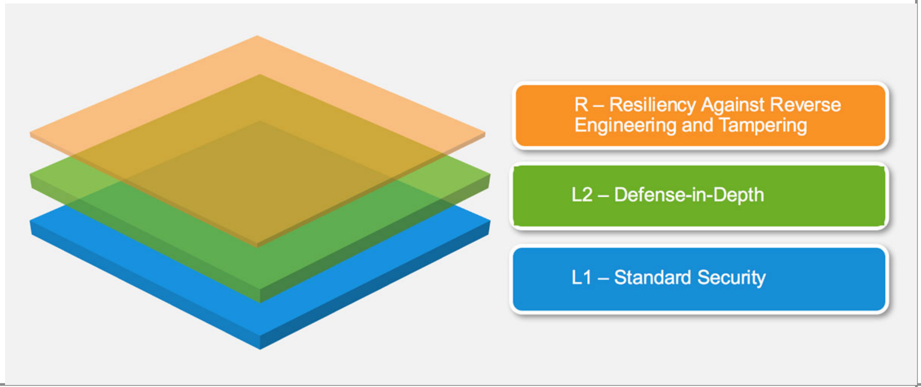
CHAPTERS OF MASVS

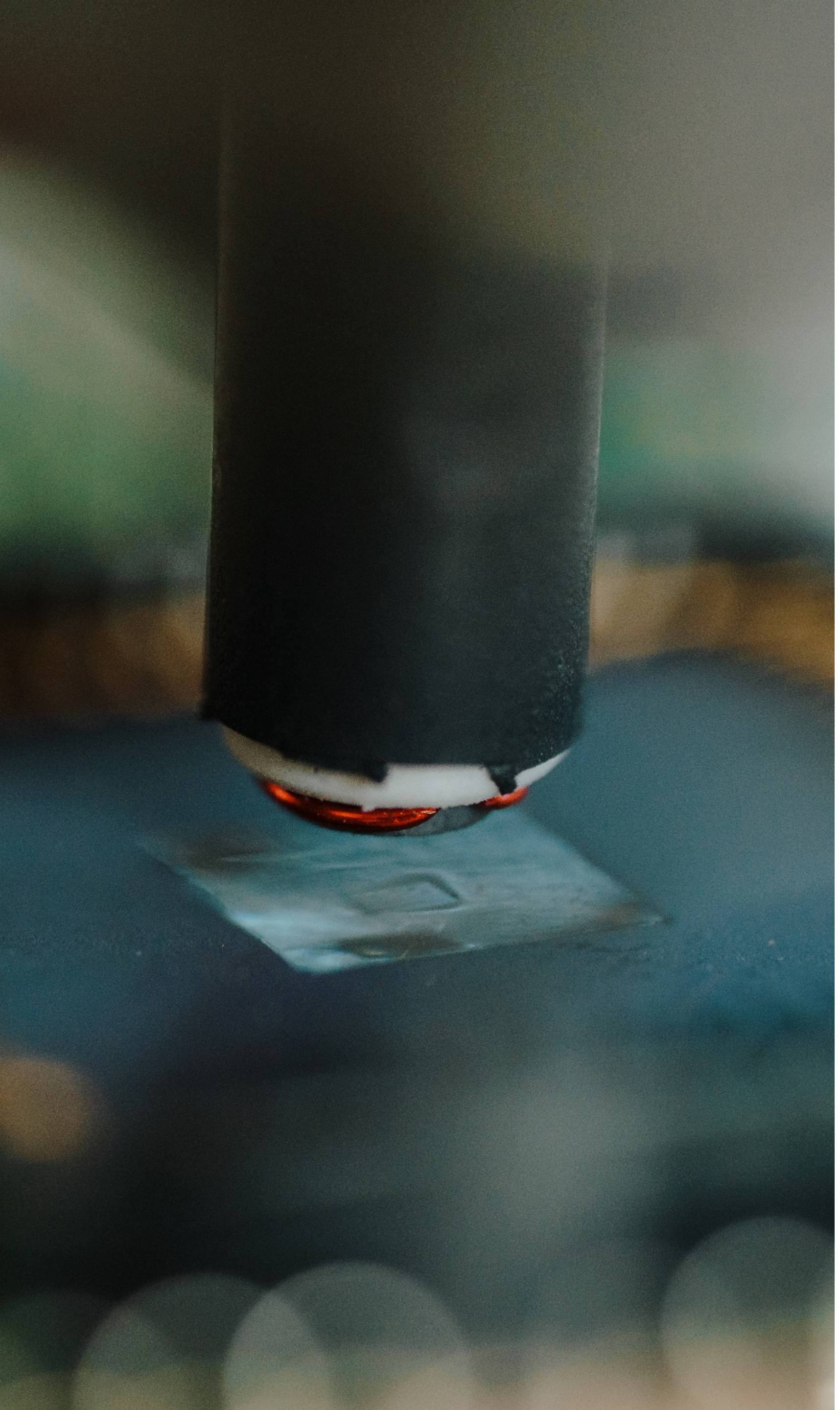
MASVS-RESILIENCE: Resilience Against Reverse Engineering and Tampering

- Add multiple layer of security
- App responds to presence of rooted or jailbroken device
- App detects tampering of files in its own sandbox
- App detects execution in emulator
- App is obfuscated

VERIFICATION LEVELS

- L1: Standard Security
  - Handles basic requirements
  - Appropriate for all mobile applications
- L2: Defense-in-Depth
  - Advanced security controls
  - Appropriate for apps that handle highly sensitive data
- R: Resiliency Against Reverse Engineering and Tampering
  - Client-side attacks
  - Appropriate for apps that handle highly sensitive data
  - Protecting intellectual property and prevent tampering

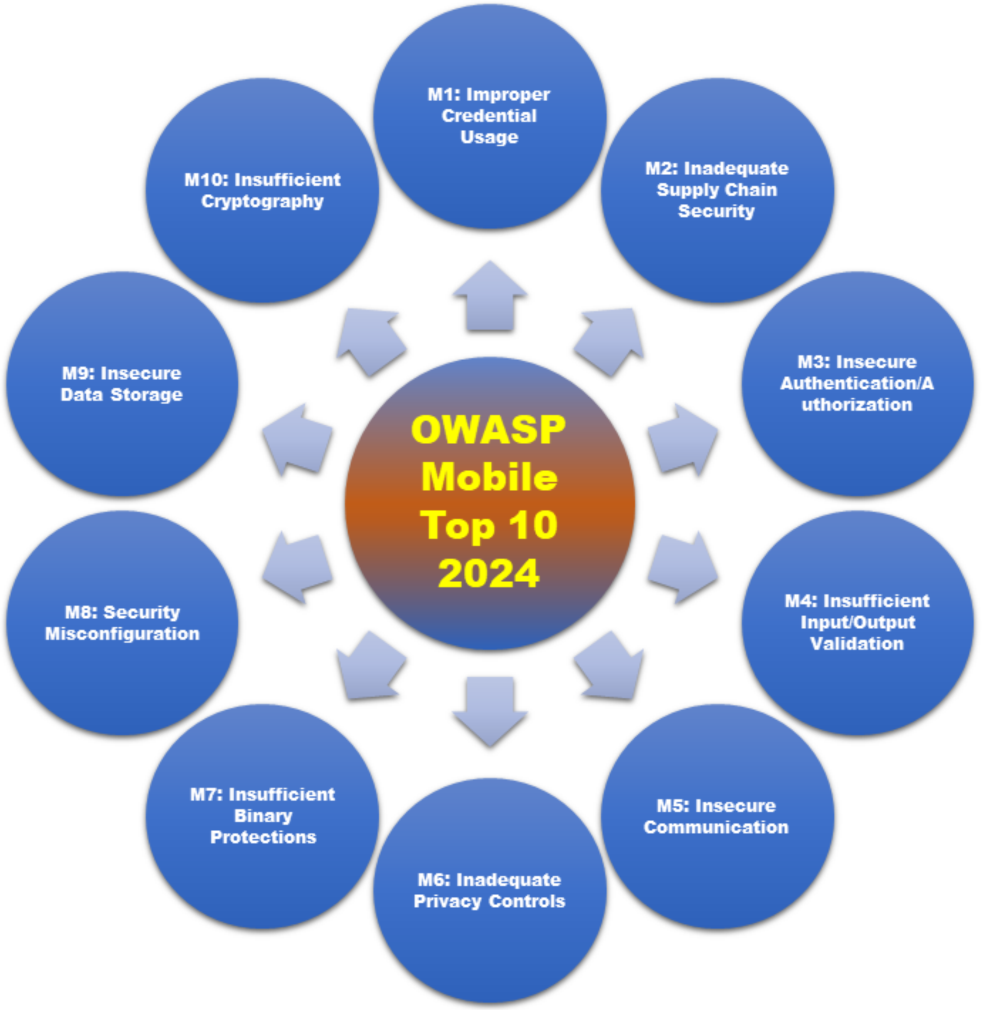




OWASP Mobile Top 10

OWASP MOBILE TOP 10 (2023)

https://owasp.org/www-project-mobile-top-10/


OWASP Mobile Top 10 2024
M1: Improper Credential Usage
M2: Inadequate Supply Chain Security
M3: Insecure Authentication/Authorization
M4: Insufficient Input/Output Validation
M5: Insecure Communication
M6: Inadequate Privacy Controls
M7: Insufficient Binary Protections
M8: Security Misconfiguration
M9: Insecure Data Storage
M10: Insufficient Cryptography

M1: IMPROPER CREDENTIAL USAGE

- Hardcoded Credentials - If the mobile app contains hardcoded credentials within the app’s source code or any configuration files, this is a clear indicator of vulnerability.
- Insecure Credential Transmission - If credentials are transmitted without encryption or through insecure channels, this could indicate a vulnerability.
- Insecure Credential Storage - If the mobile app stores user credentials on the device in an insecure manner, this could represent a vulnerability.
- Weak User Authentication - If user authentication relies on weak protocols or allows for easy bypassing, this could be a sign of vulnerability.

M2: INADEQUATE SUPPLY CHAIN SECURITY

- **Lack of Security in Third-Party Components:** Third-party components, such as libraries or frameworks, can contain vulnerabilities that can be exploited by attackers. If the mobile application developer does not vet the third-party components properly or keep them updated, the application can be vulnerable to attacks.
- **Malicious Insider Threats:** Malicious insiders, such as a rogue developer or a supplier, can introduce vulnerabilities into the mobile application intentionally. This can occur if the developer does not implement adequate security controls and monitoring of the supply chain process.
- **Inadequate Testing and Validation:** If the mobile application developer does not test the application thoroughly, it can be vulnerable to attacks. The developer may also fail to validate the security of the supply chain process, leading to vulnerabilities in the application.
- **Lack of Security Awareness:** If the mobile application developer does not have adequate security awareness, they may not implement the necessary security controls to prevent supply chain attacks.

M3: INSECURE AUTHENTICATION/AUTHORIZATION

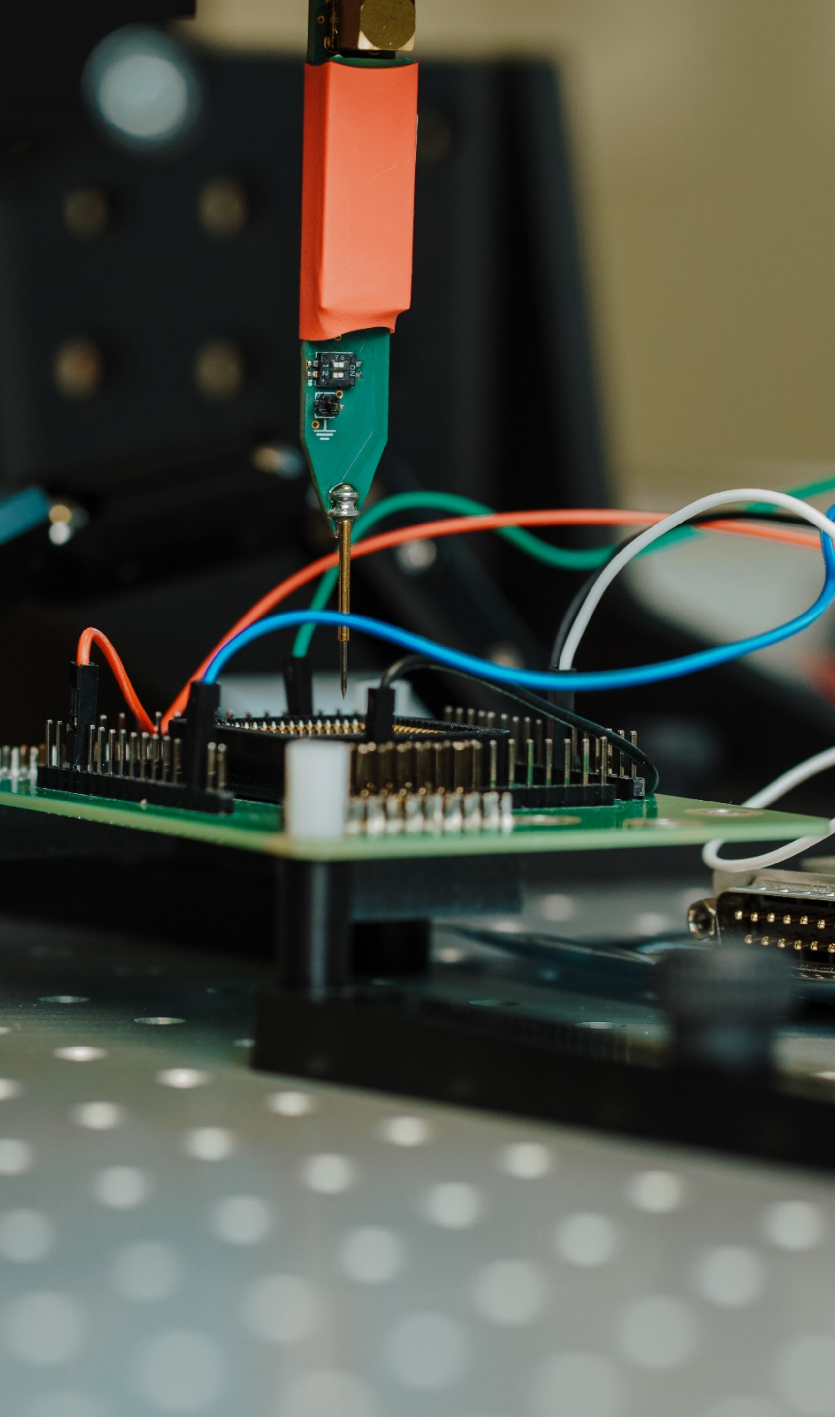
- Presence of Insecure Direct Object Reference (IDOR) vulnerabilities
- Hidden Endpoints
- User Role or Permission
- Anonymous Backend API Execution
- Local Storage of Passwords or Shared Secrets
- Weak Password Policy
- Usage of Features like FaceID and TouchID

M4: INSUFFICIENT INPUT/OUTPUT VALIDATION

- **Lack of Input Validation:** Failure to properly validate user input can expose the application to injection attacks like SQL injection, command injection, or XSS.
- **Inadequate Output Sanitization:** Insufficient sanitization of output data can result in XSS vulnerabilities, allowing attackers to inject and execute malicious scripts.
- **Context-Specific Validation Neglect:** Neglecting to consider specific validation requirements based on data context can create vulnerabilities, such as path traversal attacks or unauthorized access to files.
- **Insufficient Data Integrity Checks:** Not performing proper data integrity checks can lead to data corruption or unauthorized modification, compromising reliability and security.
- **Poor Secure Coding Practices:** Neglecting secure coding practices, such as using parameterized queries or escaping/encoding data, contributes to input/output validation vulnerabilities.

M5: INSECURE COMMUNICATION

- No usage of TLS or wrong configuration
- Traffic and sensitive information can be sniffed (privacy violations) and altered (MITM attacks)



Reporting

THE REPORT

- Exact version/configuration that was tested
- Provides a detailed explanation of the tests performed
- Clearly states the results
  - Results should be repeatable and preproducable
- Points out the limitations of the test
  - “Absence of proof is not the proof of absence”
  - You can be held liable for what is written in the report!
- (Optional) rates findings
- (Optional) provides pointers on how to fix findings

ANATOMY OF A REPORT

- Executive Summary
- Introduction
- Results Summary
- Testing Approach
- Detailed Results
- Recommendations

EXECUTIVE SUMMARY

- Provide the big picture briefly
- Only two to three paragraphs
- State
  - What was tested
  - When
  - Most important results

INTRODUCTION

- Basically a longer version of the executive summary
- State
  - What was tested
  - When
  - How (briefly, i.e. type of test)
  - Results (overview, still in greater detail than before)
- Provide an outlook of the rest of the report

RESULTS SUMMARY

- List all the tests you have done
- Whether you found something or not
- Provide some charts
  - If severity rating was used, a pie chart looks nice
- Make it easy for technical guys to find the results they should look at

TESTING APPROACH

- Provide a methodology of your tests
- You don’t need to reveal how but what
- Refer to standards you used
- Or mention what parts of the OWASP Top 10 you examined for web apps
- This helps cover your butt in case your results are questioned
  - i.e. a vulnerability is found but it was not in scope for your test

DETAILED RESULTS

- List all test cases or findings
- Provide a pass/fail or severity rating
- Give a technical explanation of what the test is about
  - i.e. describe what XSS means and its effects
- If you found nothing
  - Provide a list of everywhere you looked
- If you found something
  - Give a step-by-step guide on how to reproduce the error
- (Optional) provide a rating
  - Only do so, if client asks for it
  - Again, liability

RATINGS AND POINTERS FOR FIXING ISSUES

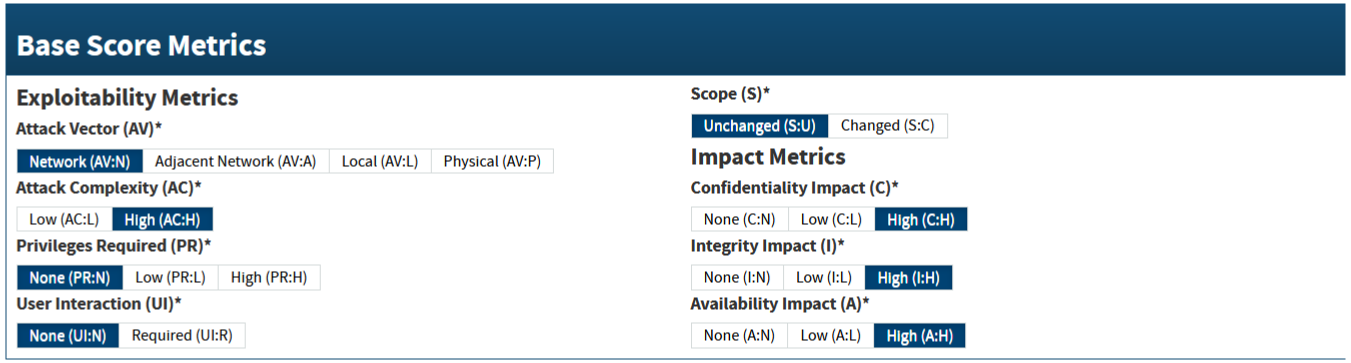
- Ratings
  - Various systems:
    - Traffic Light
    - Severity x Probability
    - CVSSv3 (more on next slide)
  - Best to agree with client
- Pointers
  - Either very general
  - Or very specific to test target
    - Goes beyond the scope of a pentest
    - Requires good software development knowledge on your end

CVSSV3

- Common Vulnerability Scoring System
- Agreed upon standard for rating IT security vulnerabilities
  - Still offers debatable results depending on your type of tests
- Handy ‘calculator’ available online
  - https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator
  - Play around with it!

CVSSV3 EXAMPLE - SQLI

- Assume you found an SQL injection exposed on the internet



Base Score Metrics

Exploitability Metrics

Attack Vector (AV)*
[✓ Network (AV:N)] Adjacent Network (AV:A) Local (AV:L) Physical (AV:P)

Attack Complexity (AC)*
Low (AC:L) [✓ High (AC:H)]

Privileges Required (PR)*
[✓ None (PR:N)] Low (PR:L) High (PR:H)

User Interaction (UI)*
[✓ None (UI:N)] Required (UI:R)

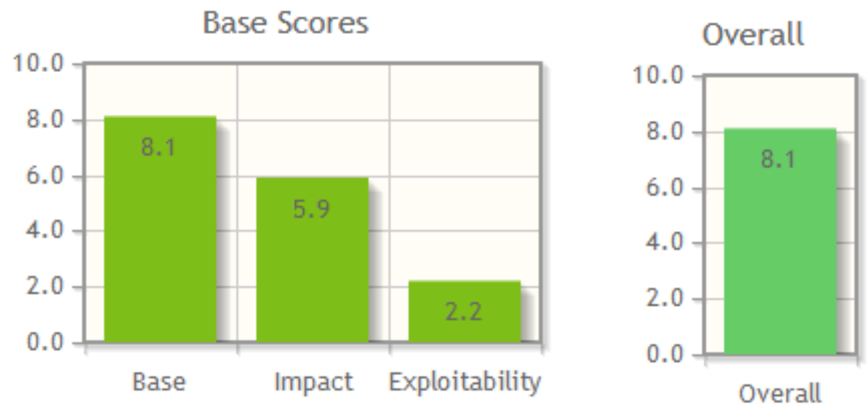
Scope (S)*
[✓ Unchanged (S:U)] Changed (S:C)

Impact Metrics

Confidentiality Impact (C)*
None (C:N) Low (C:L) [✓ High (C:H)]

Integrity Impact (I)*
None (I:N) Low (I:L) [✓ High (I:H)]

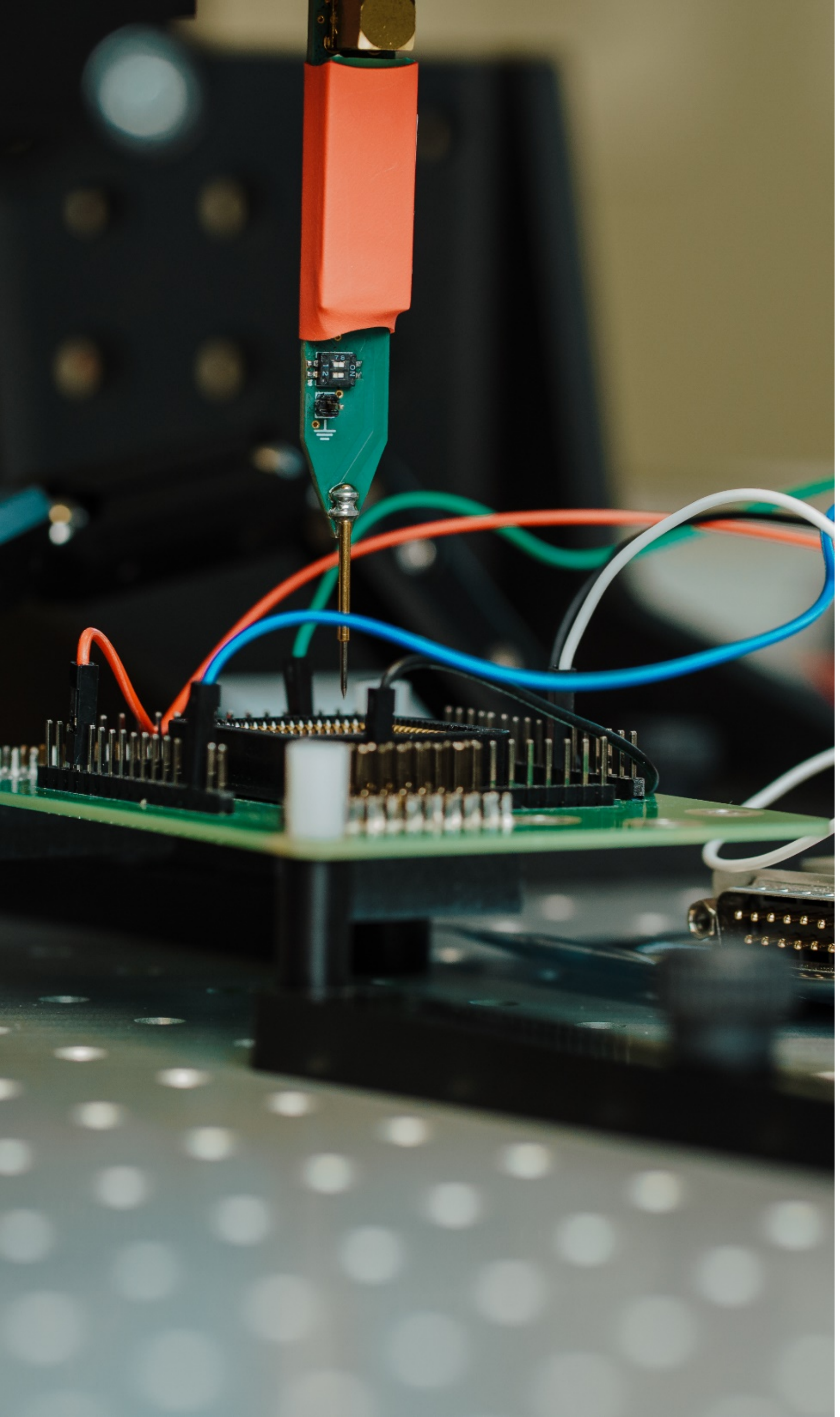
Availability Impact (A)*
None (A:N) Low (A:L) [✓ High (A:H)]



Sounds about right

RECOMMENDATIONS

- Only provide them if they are an agreed upon part of the contract
  - Liability
- Indicate which vulnerabilities should be closed first and why
  - First everything that has a big impact
  - Then the ‘low hanging fruits’ with the biggest impact first



Wrap up

SUMMARY

- Kinds of pentest
  - Black, Grey, White
- Stages of the pentest
  - Contract
  - Discovery
  - Attack
  - Reporting
- Types of targets
  - Web Apps
  - Infrastructure
  - Mobile Apps
- Reporting

[w] www.yagoba.com, [m] info@yagoba.com
Beethovenstraße 20, 8010 Graz, Austria

Security testing

Introduction to Penetration Testing

Tomislav Nad

WHEN YOU NEED TO BE SURE

REFERENCES

- https://portswigger.net/daily-swig/vulnerability-in-facebook-android-app-nets-10k-bug-bounty
- https://nakedsecurity.sophos.com/2018/11/16/hacking-misafes-smartwatches-for-kids-is-childs-play/
- https://medium.com/@c0nk3r/bypassing-apples-touch-id-but-from-certain-apps-e7c6fbe3d848
- https://www.idc.com/promo/smartphone-market-share/os
- https://arxiv.org/pdf/1901.10062.pdf
- https://owasp.org/
- https://mobile-security.gitbook.io/mobile-security-testing-guide
- https://mobile-security.gitbook.io/masvs
- https://www.nowsecure.com/blog/2018/07/11/a-decade-in-how-safe-are-your-ios-and-android-apps/
- https://owasp.org/www-project-mobile-top-10/
- https://source.android.com/security
- https://support.apple.com/de-at/guide/security/welcome/web (pdf at bottom of page)
- https://github.com/tanprathan/sievePWN