Bai7 GmbH

Sven Schleier B.Eng.

August 2025

The Mobile Playbook:
A guide to iOS & Android App Security - Day 1

Android Training

Preparation

Github

For the training today you need to use a free Github account to follow the labs. If you don’t have one, please create one now.

https://github.com/

Sven Sch

© 2025 by Bai7 GmbH | All rights reserved

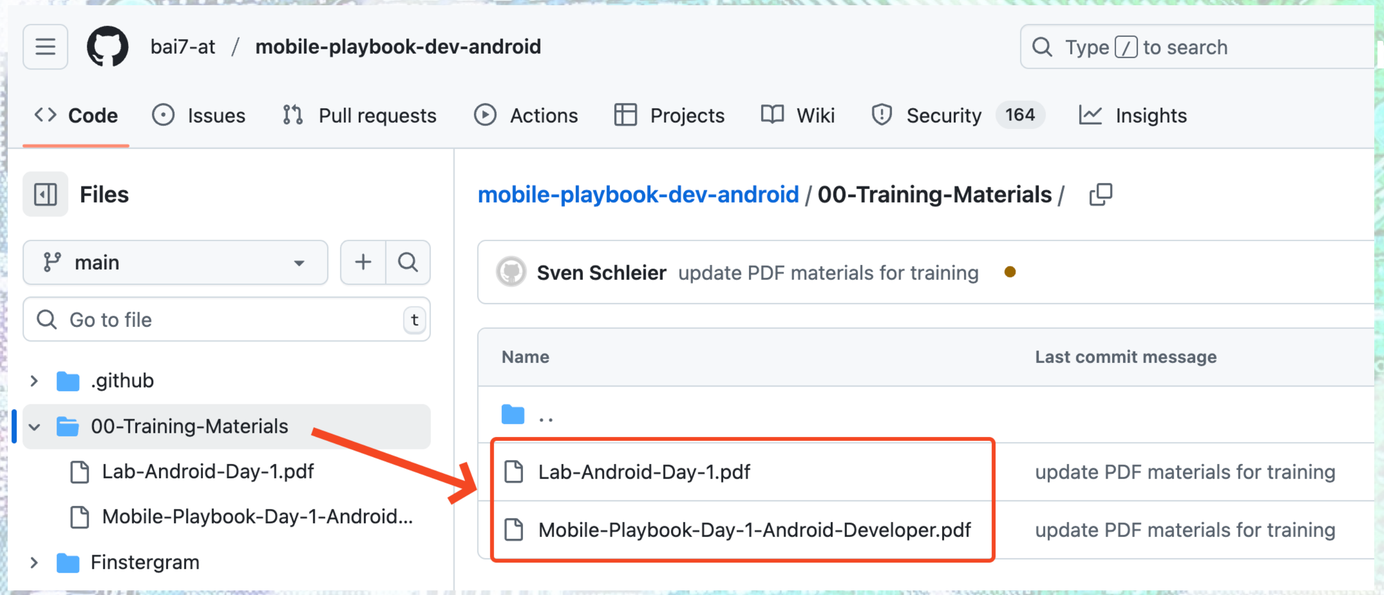
Github

Clone the repo bai7-at/mobile-playbook-dev-android to your machine:

http://bit.ly/4mp9MtA

In the folder 00-Training-Materials there are 2 PDFs:

“Lab-Android-Day-1.pdf” => These are the lab instructions you will go through today.
“Mobile-Playbook-Day-1-Android-Developer.pdf” => This is the slide deck for today.



© 2025 by Bai7 GmbH | All rights reserved

Android Training

Introduction

/usr/bin/whoami

My name is Sven and I will be your instructor!

- Doing consulting for 15+ years in various roles in Cyber Security (focus on AppSec, Pentesting and Cloud Security)
- Worked and lived in Germany, China, Singapore and now Austria.
- Principal Security Consultant at Crayon, leading the cyber security consulting team
- Since 2016 one of the project leaders for the OWASP Mobile Application Security (MAS) Project

Instructor

Sven Schleier

Contact:
- Linkedin
- sven@bai7.at

© 2025 by Bai7 GmbH | All rights reserved

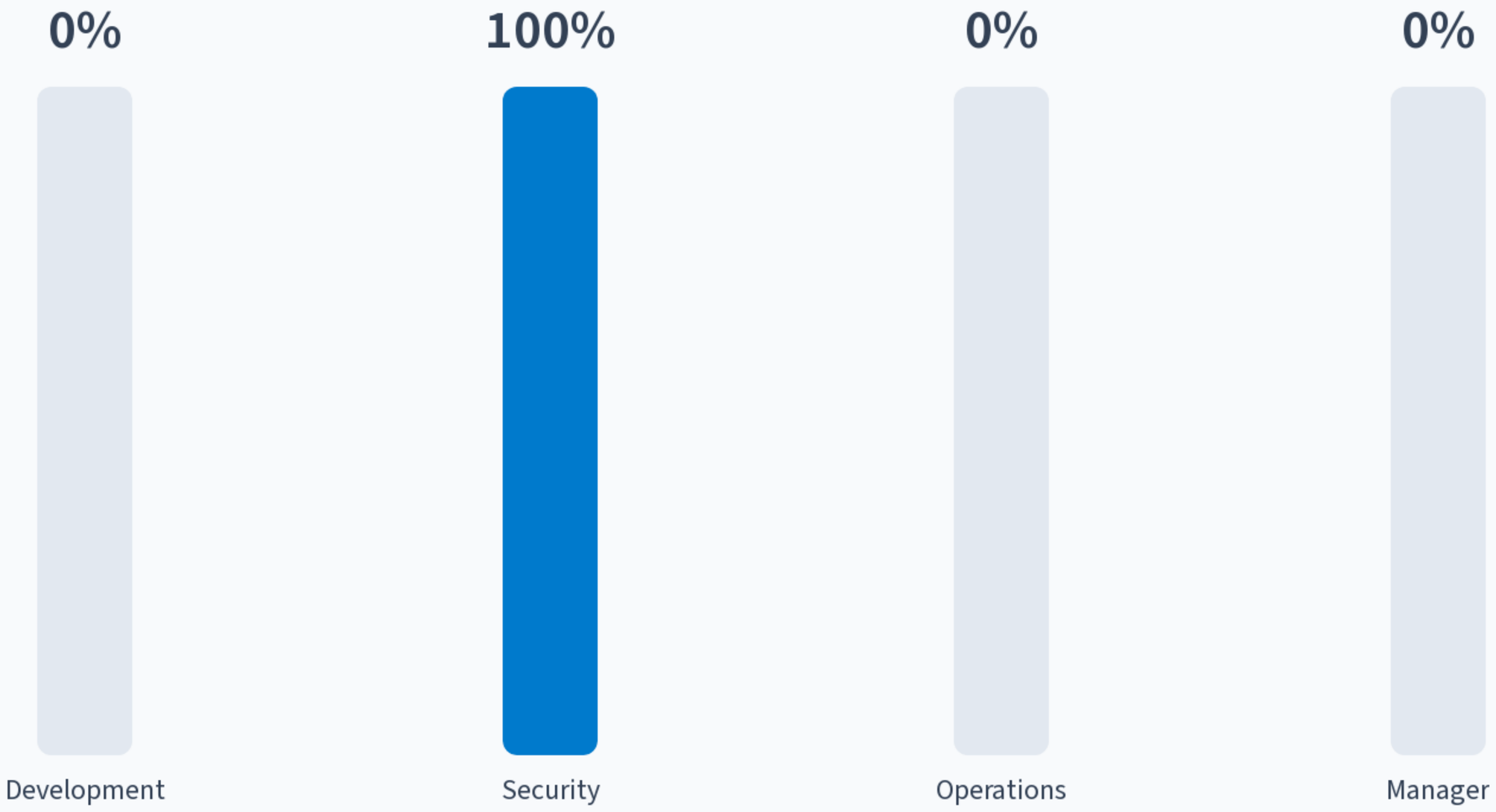
/usr/bin/whoareyou

Let me ask you some questions!

- Grab your phone and scan the QR code or click on the link below!
- https://pollev.com/svens131

© 2025 by Bai7 GmbH | All rights reserved

What do you do?



Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

What is your experience in Mobile App Security?

I am pretty new to the whole Cyber Security industry

0%

I've got experience in Web Application Security

0%

I've got already experience in Mobile App Security

0%

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Are you familiar with the OWASP MAS Project (MASVS and MASTG)?

Nope, never heard of it.

0%

Yes, but never really used it.

0%

I am using during my work.

0%

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Introduction

© 2025 by Bai7 GmbH | All rights reserved

Agenda - Android Modules Day 1

9:00 – 10:30

- Introduction of the OWASP MASTG / MASVS
- Scanning for secrets in a Github repository
- Scanning a compiled Android app (APK)

10:30 – 11:00 - Coffee break

11:00 - 12:15

- Perform Software Composition Analysis (SCA)
- Static scanning of an Android app (developed in Kotlin)
- Testing for sensitive data in an Android App

12:15 – 13:30 - Lunch

13:30 – 15:00

- Using Corellium to set up a rooted Android device
- Android Reverse Engineering and Frida 101
- Bypassing root detection in a real world app and Android Reverse Engineering Controls
- SSL Pinning

15:00 – 15:30 break

15:30 – 17:00

- Frida Flow and Anti-Frida
- CTF

© 2025 by Bai7 GmbH | All rights reserved

Rules / Disclaimer

There are no stupid questions!

You are highly encouraged:

- To ask during the training
- Reach out to me during the breaks
- Or contact me via Linkedin/OWASP Slack/ Mail/$foo

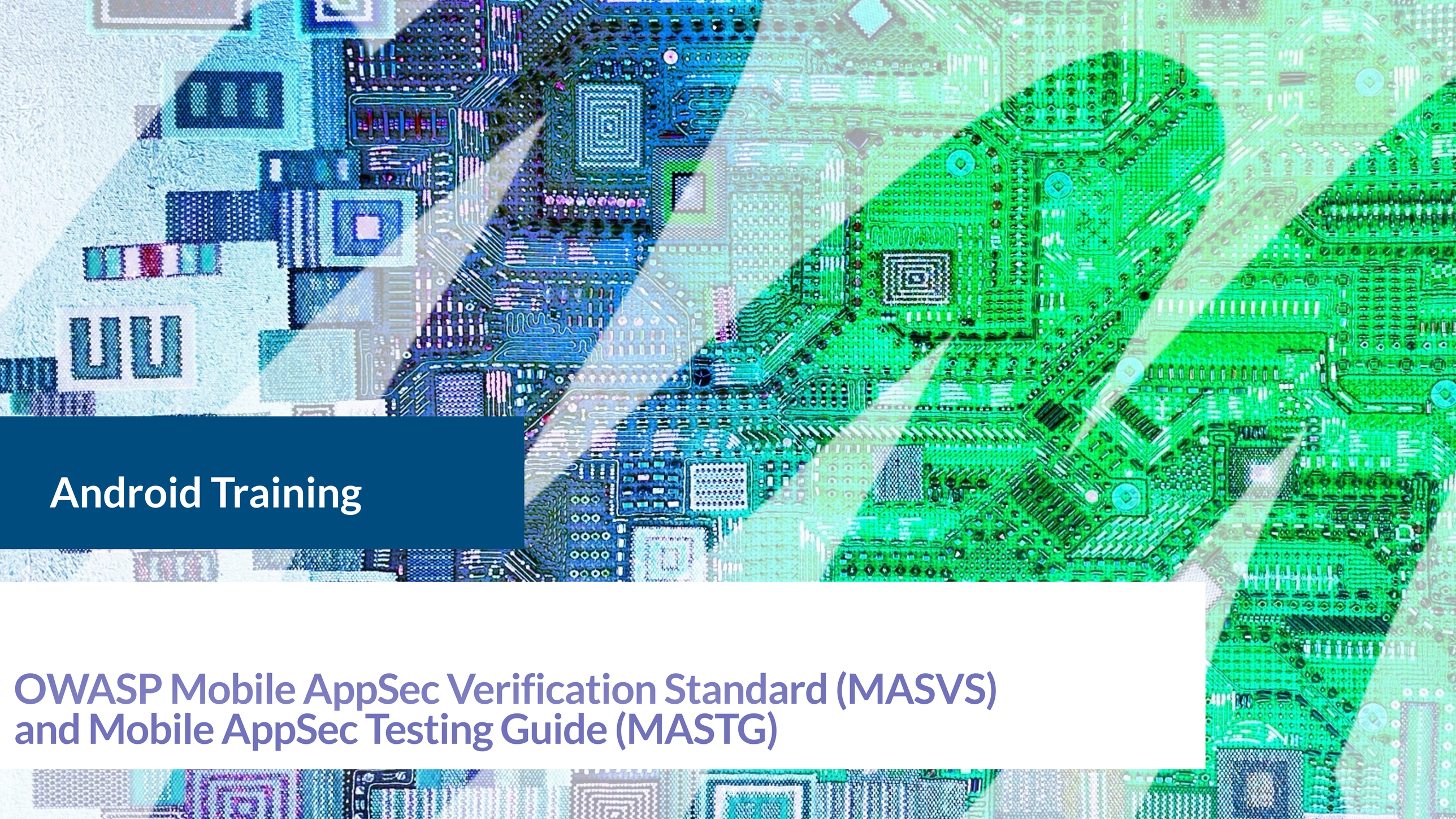
The intention of this course is to be interactive, and this depends on your participation!

The content for today is based on the OWASP Mobile App Sec Testing Guide and my experience during mobile pentests.

Feel free to use it after the training, but please don't share the training content publicly!

All views and opinions in this training slides and that are mentioned during the training are my own and not the view of our employer.

© 2025 by Bai7 GmbH | All rights reserved



Android Training

OWASP Mobile AppSec Verification Standard (MASVS) and Mobile AppSec Testing Guide (MASTG)

Open Worldwide Application Security Project (OWASP)

What is OWASP?

© 2025 by Bai7 GmbH | All rights reserved

Open Worldwide Application Security Project (OWASP)

What is OWASP?

- Focused on the improvement of software and advocates for application security
- Worldwide non-profit organization
- AppSec Conferences
- Chapters and meetups worldwide
- All materials, documentation and tools are open source

Flagship projects:

- Dependency Check
- AppSec Verification Standard (ASVS)
- Cheat Sheet Series
- Security Knowledge Framework (SKF)

© 2025 by Bai7 GmbH | All rights reserved

Introduction - OWASP Mobile Application Security (MAS) Project

Our mission:
"Define the industry standard for mobile application security."

https://mas.owasp.org

© 2025 by Bai7 GmbH | All rights reserved

Introduction - OWASP Mobile Application Security (MAS) Project

The OWASP MASVS is the industry 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.

The OWASP MASTG is a comprehensive manual for mobile app security testing (penetration testing) and reverse engineering.
It describes technical processes for verifying the controls listed in the OWASP MASVS.

© 2025 by Bai7 GmbH | All rights reserved

What do you think are the most critical areas of a mobile app that should be secured?

Nobody has responded yet.

Hang tight! Responses are coming in.

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Introduction - MASVS - Control Groups

https://mas.owasp.org/MASVS/

The standard is divided into various groups of controls, labeled MASVS-XXXXX, that represent the most critical areas of the mobile attack surface:

- MASVS-STORAGE: Storage
- MASVS-PLATFORM: Platform Interaction
- MASVS-CRYPTO: Cryptography
- MASVS-CODE: Code Quality
- MASVS-AUTH: Authentication and Authorisation
- MASVS-RESILIENCE: Resilience against Reverse Engineering and Tampering
- MASVS-NETWORK: Network Communication
- MASVS-PRIVAY: Privacy

© 2025 by Bai7 GmbH | All rights reserved

Introduction - MASVS - Control Groups

https://mas.owasp.org/MASVS/

Example of controls in a control group

MASVS-STORAGE:
Storage

MASVS-STORAGE-1
The app securely stores sensitive data.

MASVS-STORAGE-2
The app prevents leakage of sensitive data.

MASVS-PRIVAY:
Privacy

MASVS-PRIVACY-3
The app is transparent about data collection and usage.

MASVS-PRIVACY-4
The app offers user control over their data.

© 2025 by Bai7 GmbH | All rights reserved

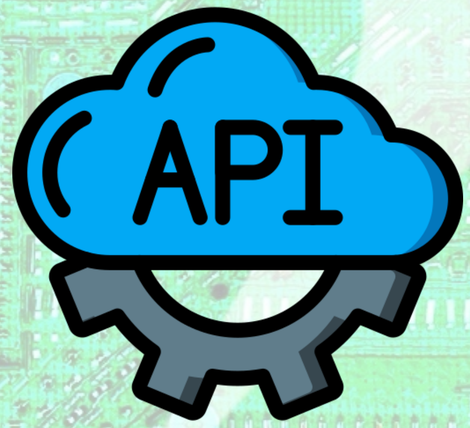
Your app stores an auth token in local storage. What MASVS area(s) does that touch?

0

MASVS-STORAGE	0%
MASVS-CRYPTO	0%
MASVS-AUTH	0%
MASVS-NETWORK	0%
MASVS-PLATFORM	0%
MASVS-CODE	0%
MASVS-RESILIENCE	0%
MASVS-PRIVACY	0%

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Introduction - MASVS and ASVS



The MASVS is covering the Mobile App only, for anything related to the APIs the app is communicating with the OWASP ASVS project should be used.

The Application Security Verification Standard (ASVS) defines security requirements for web applications and services and is made up of a total of around 350 requirements which are divided into 17 chapters.

For example one chapter defines the usage of tokens:

V9 Self-contained Tokens

https://github.com/OWASP/ASVS

© 2025 by Bai7 GmbH | All rights reserved

Introduction - MASVS - Testing Profiles

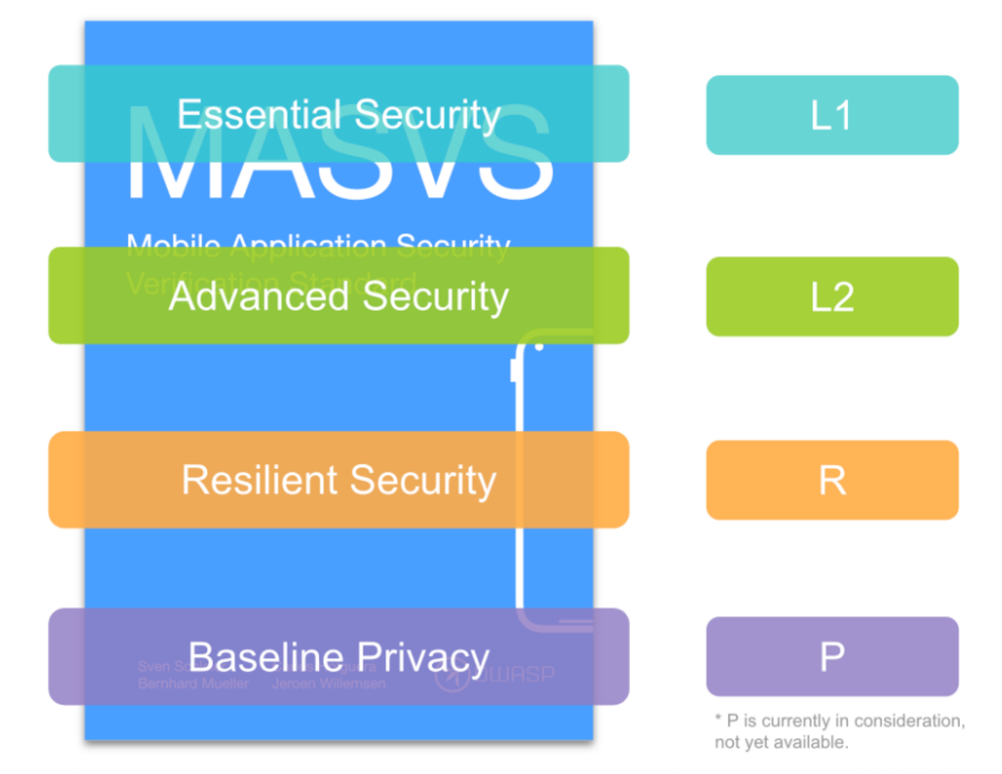
Testing Profiles in the MASVS

MAS-L1 - Essential Security

MAS-L1 provides a baseline for the most fundamental security requirements and best practices that every mobile app should meet to protect against common threats.

MAS-L2 - Advanced Security

MAS-L2 extends MAS-L1 introducing additional security measures and best practices for mobile apps to address advanced threats requiring more rigorous threat modeling and testing strategies.



https://mas.owasp.org/news/2023/07/28/mas-testing-profiles-and-mastg-atomic-tests/

© 2025 by Bai7 GmbH | All rights reserved

Introduction - MASVS - Control Groups

https://mas.owasp.org/MASVS/

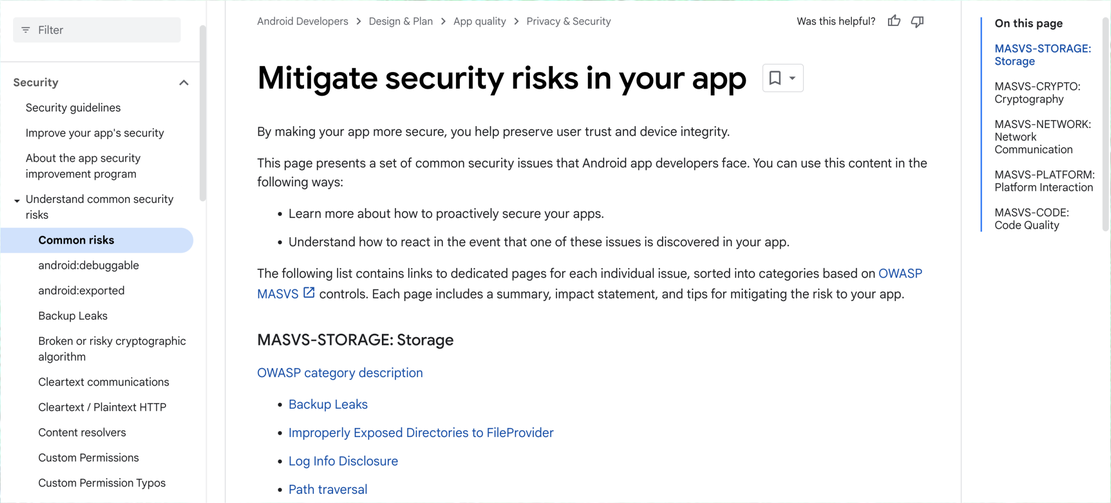
- The control groups and controls are a baseline that need to be tailored to your needs.
- Don't blindly follow the controls!
  - Requirements might be missing (e.g. regulations in your country/industry)
  - Requirements might not be applicable (or you may want to accept the risk)
- Usage ensures consistency of mobile app security when developing / testing an app
- Can be part of your threat model to select the requirements that address your gaps to create your own test profile.

See also: The Case for Standards in Mobile App Security

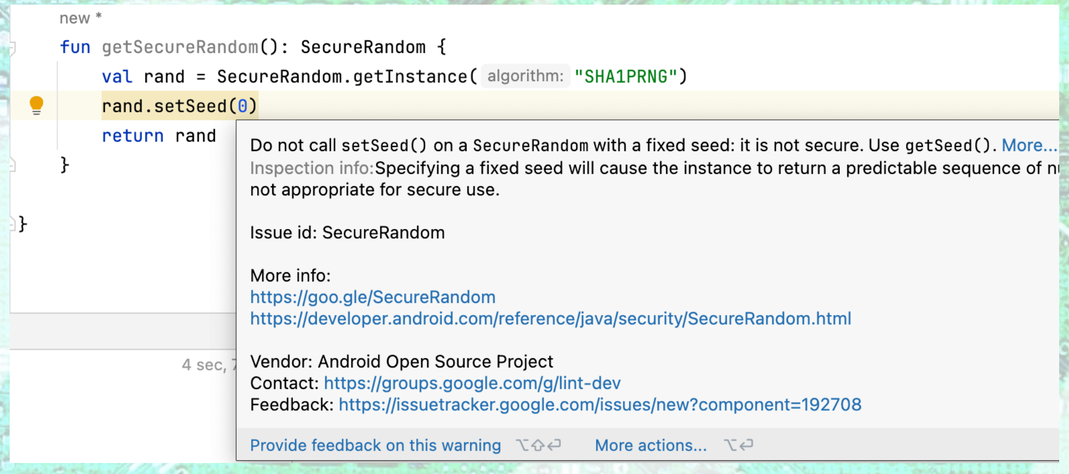
© 2025 by Bai7 GmbH | All rights reserved

Introduction - MASVS and Android Application Security Knowledge Base (AAKB)

OWASP MASVS is recognised by Android and linked into the developer documentation, as part of the Android Application Security Knowledge Base (AAKB) Project.



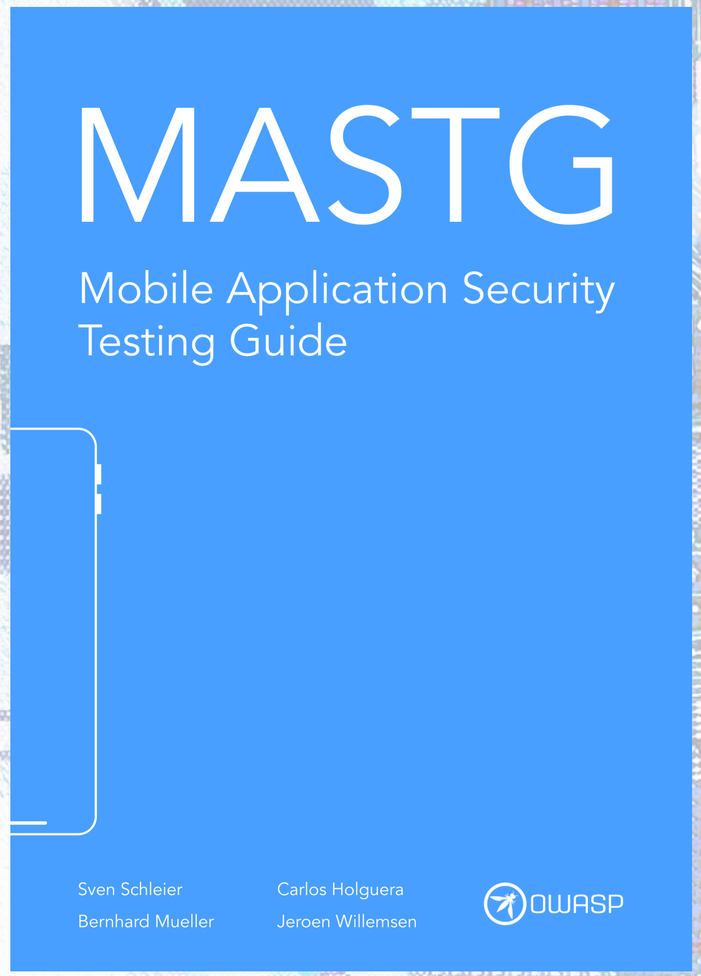
Existing security lint checks within Android Studio Giraffe+ have had their descriptions updated to include a link to the relevant AAKB article.
An open-source Android Security lint check has also been released.



https://android-developers.googleblog.com/2024/07/making-security-easy-vuln-remediation.html

© 2025 by Bai7 GmbH | All rights reserved

Introduction - MASTG



The OWASP MASTG is a comprehensive manual for mobile app security testing (penetration testing) and reverse engineering. It describes technical processes for verifying the controls listed in the OWASP MASVS.

Structure of MASTG:

- Tests
- Techniques
- Tools
- Apps

https://mas.owasp.org/MASTG

© 2025 by Bai7 GmbH | All rights reserved

Introduction - Terminology

Penetration Test (“Pentest”)

A penetration test involves simulating attacks on the app and server-side APIs to identify security vulnerabilities and assess the robustness of its defenses.

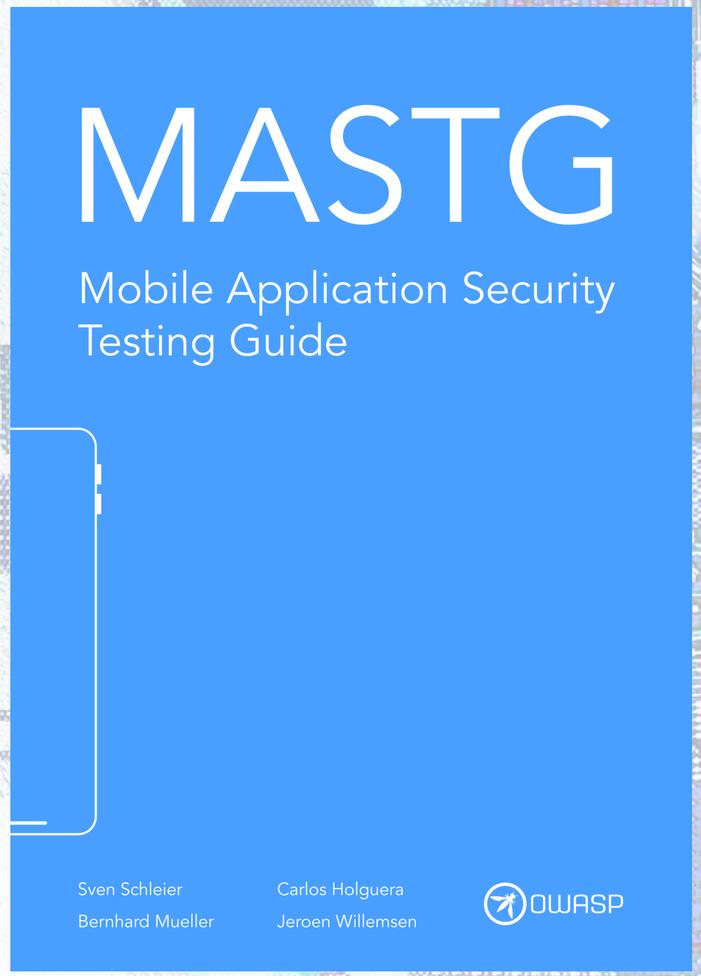
Reverse engineering

Involves decompiling of the app binary (APK) and analyzing the app's code and architecture to understand its functionality, uncover potential security flaws, bypass resilience controls (like root detection) or replicate its features.

© 2025 by Bai7 GmbH | All rights reserved

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Introduction - MASTG



https://mas.owasp.org/MASTG

The OWASP MASTG is a comprehensive manual for mobile app security testing (penetration testing) and reverse engineering. It describes technical processes for verifying the controls listed in MASVS.

f MASTG:

- Techniques
- Apps

© 2025 by Bai7 GmbH | All rights reserved

Introduction of MASVS and MASTG - Enjoy our talks!

OWASP Mobile Application Security

Search

OWASP/mastg
v1.7.0 12.4k 2.5k

OWASP Mobile Application Security | MASWE (Beta) | MASTG | MASVS | MAS Checklist | MAS Crackmes | News | Talks | Contribute | Donate | Connect with Us

Talks

Showing 1 to 42 of 42 entries

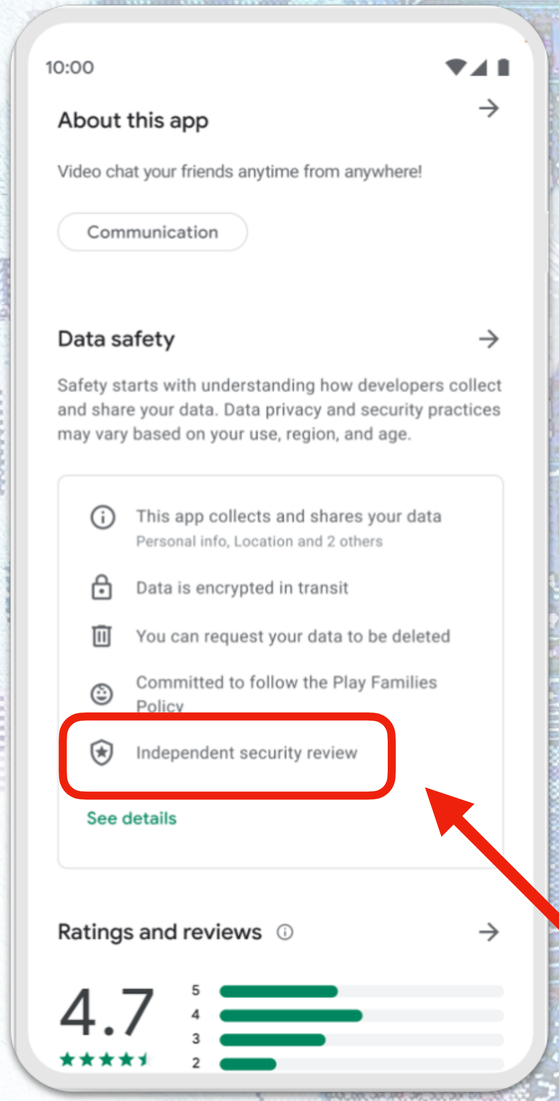
Search:

Date	Event	Title	Video	Slides
May 29, 2025	OWASP AppSec EU 2024	Hack-Proof Your Code: Exploring Mobile App Weaknesses with OWASP MAS	Not yet available	Slides
November 9, 2024	r2con Barcelona	Hack-Proof Your Mobile Apps	Video	Slides
September 26, 2024	OWASP AppSec US 2024 - San Francisco	OWASP Mobile Application Security	Video	Slides
August 22, 2024	OWASP Atlanta Chapter	Hack-Proof Your Code: Exploring Mobile App Weaknesses with OWASP MAS	Video	Slides
June 2024	OWASP AppSec EU 2024	OWASP MAS: A look into the new Mobile App Weaknesses & Tests	Video	Slides
June 2024	NSConnect	OWASP Mobile Power up: A look into the New MAS Risks & Tests	Video	Slides
October 2023	OWASP AppSec US 2023	Refactoring Mobile App Security	Video	Slides

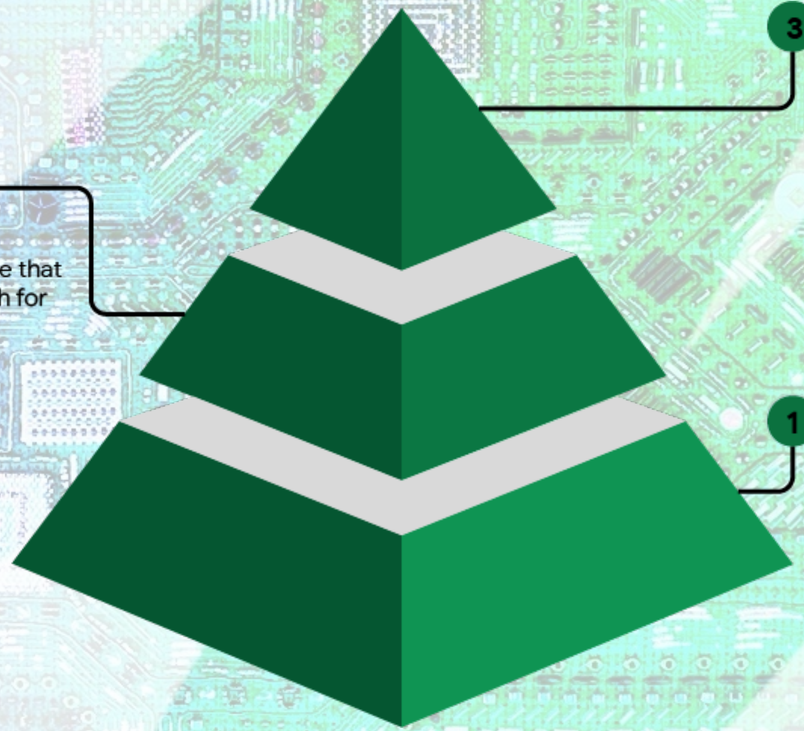
https://mas.owasp.org/talks/

© 2025 by Bai7 GmbH | All rights reserved

Introduction - Adoption in the Industry



Google Play + ADA



3 Authorized labs

Google provides a list of accredited labs for developers

OWASP MASTG 2

Corresponding testing guidance that provides a single source of truth for how to objectively test for compliance with MASVS

1 OWASP MASVS

Industry recognized set of security criteria for mobile applications

https://appdefensealliance.dev

© 2025 by Bai7 GmbH | All rights reserved

Android Training

Secret Scanning of an Android project

Android - Secret Scanning

Be honest:

Have you ever committed a password to your repo?

© 2025 by Bai7 GmbH | All rights reserved

Android - Secret Scanning

If your project communicates with an external service, you might use a token or private key for authentication. Tokens and private keys are examples of secrets that a service provider can issue. If you check a secret into a repository, anyone who has read access to the repository can use the secret to access the external service with your privileges. We recommend that you store secrets in a dedicated, secure location outside of the repository for your project.

https://docs.github.com/en/code-security/secret-scanning/about-secret-scanning

© 2025 by Bai7 GmbH | All rights reserved

Android - Secret Scanning

Scanning for secrets is a common attack vector against repositories (including scanning the git commit history!):

- Public repos are being scanned regularly
- Private repos are being scanned after a breach by attackers to gain access to more systems

Home > News > Security > Toyota discloses data leak after access key exposed on GitHub

Toyota discloses data leak after access key exposed on GitHub

By Bill Toulas

October 10, 2022 01:50 PM

https://www.bleepingcomputer.com/news/security/toyota-discloses-data-leak-after-access-key-exposed-on-github/

© 2025 by Bai7 GmbH | All rights reserved

Android - Fork and setup a Github Repository

Open the file Lab-Android-Day-1.pdf

Section:
Lab - Fork and setup Repo

Note: Some PDF readers do not copy properly and miss whitespaces etc.

© 2025 by Bai7 GmbH | All rights reserved

Android - Secret Scanning

What do you consider as secrets?

© 2025 by Bai7 GmbH | All rights reserved

What do you consider as secrets?

Nobody has responded yet.

Hang tight! Responses are coming in.

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Android - Secret Scanning

Where and how can secrets be exposed during development (what can go wrong)?

© 2025 by Bai7 GmbH | All rights reserved

Where and how can secrets be exposed during the SDLC (what can go wrong)?

Nobody has responded yet.

Hang tight! Responses are coming in.

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Android - Secret Scanning - Gitleaks

Many secret scanning tools are available, we will focus today on gitleaks.

It's open source, actively maintained, fast, has customizable detection rules and offers various output formats.

Secret scanning should happen throughout the SDLC, e.g.:

- Pre-commit hook
- During build in the pipeline

README MIT license

Gitleaks

Test passing | docker pulls 20M | gitleaks playground | protected by gitleaks | reference | go report A+ | license MIT

Join our Discord! 81 online

Gitleaks is a tool for **detecting** secrets like passwords, API keys, and tokens in git repos, files, and whatever else you wanna throw at it via `stdin`. If you wanna learn more about how the detection engine works check out this blog: Regex is (almost) all you need.

```
→  ~/code(master) gitleaks git -v

    ○
    │╲
    │ ○
    ○ ░
    ░    gitleaks


Finding:     "export BUNDLE_ENTERPRISE__CONTRIBSYS__COM=cafebabe:deadbeef",
Secret:      cafebabe:deadbeef
RuleID:      sidekiq-secret
Entropy:     2.609850
File:        cmd/generate/config/rules/sidekiq.go
```

© 2025 by Bai7 GmbH | All rights reserved

Android - Secret Scanning - Challenges

What are the challenges when doing secret scanning?

© 2025 by Bai7 GmbH | All rights reserved

Android - Secret Scanning - Terminology

False Positive

- ***Definition***: The scanner reports a vulnerability that isn't real.
- ✅ Code is safe and functions correctly — but the tool misunderstood something.
- ***Example***: gitleaks flags a Firebase test API key that is only used in development — not a real secret.
- Fix:
  - Tune the rule, allow it in .gitleaks.toml or
  - use#gitleaks:allow in your code.

False Negative

- ***Definition***: The scanner fails to report a real vulnerability.
- ⚠️ Code contains a real secret or bug — but the tool misses it.
- ***Example***: Old AWS keys hidden deep in commit history go unnoticed because the tool only scans the latest commit.
- Fix:
  - Use tools that scan full history (like gitleaks)

- ⏰ False Positives waste developer time.
- ❌ False Negatives leave the door open to real attacks.
- Secret scanning tools will usually flag out some false positives
- Needs tweaking and customisation to your environment

© 2025 by Bai7 GmbH | All rights reserved

Android - Secret Scanning - Gitleaks

Gitleaks - Pre-commit hook

https://github.com/gitleaks/gitleaks?tab=readme-ov-file#pre-commit

Pre-Commit

1. Install pre-commit from https://pre-commit.com/#install
2. Create a `.pre-commit-config.yaml` file at the root of your repository with the following content:

```
repos:
  - repo: https://github.com/gitleaks/gitleaks
    rev: v8.24.2
    hooks:
      - id: gitleaks
```

   for a native execution of gitleaks or use the `gitleaks-docker` pre-commit ID for executing gitleaks using the official Docker images

3. Auto-update the config to the latest repos' versions by executing `pre-commit autoupdate`
4. Install with `pre-commit install`
5. Now you're all set!

```
→ git commit -m "this commit contains a secret"
Detect hardcoded secrets.....................................................Failed
```

Note: to disable the gitleaks pre-commit hook you can prepend `SKIP=gitleaks` to the commit command and it will skip running gitleaks

```
→ SKIP=gitleaks git commit -m "skip gitleaks check"
Detect hardcoded secrets.....................................................Skipped
```

© 2025 by Bai7 GmbH | All rights reserved

Android - Secret Scanning - Gitleaks

Gitleaks - Pre-commit hook

- Scan as early as possible to avoid even committing a secret to the repo.
- You need to have go installed (for macOS: “$ brew install go”)

```
~/Dr/C/Tr/i/iOS-Conf-Training main ↓1 +2 ❯ git commit -m "test"
[INFO] Installing environment for https://github.com/zricethezav/gitleaks.
[INFO] Once installed this environment will be reused.
[INFO] This may take a few minutes...
Detect hardcoded secrets.....................................................Passed
[main 66597ed] test
 2 files changed, 7 insertions(+)
 create mode 100644 .pre-commit-config.yaml
```

© 2025 by Bai7 GmbH | All rights reserved

Android - Secret Scanning - Gitleaks

Fail commit

.pre-commit-config.yaml — iOS-Conf-Training

EXPLORER

OPEN EDITORS 1 UNSAVED

IOS-CONF-TRAINING

- .github / workflows
  - dependency-checker.yml
  - github-actions-demo.yml
  - gitleaks.yml
  - MobSF.yml
  - mobsfscan.yml
- Find_my_Data.xcodeproj
- Find_my_Data.xcworkspace
- iOS-Shack-v2
- .gitignore
- .mobsf
- .pre-commit-config.yaml
- Find_my_Data.ipa
- gitleaks.json
- gitleaks.sarif
- gitleaks.toml
- LICENSE
- Podfile
- Podfile.lock
- README.md

OUTLINE

TIMELINE

.pre-commit-config.yaml | github-actions-demo.yml | dependency-checker.yml | .gitignore

.pre-commit-config.yaml > ...

.pre-commit-config.yml (pre-commit-config.json)

```yaml
repos:
  - repo: https://github.com/zricethezav/gitleaks
    rev: v8.2.0
    hooks:
      - id: gitleaks
```

TERMINAL | PROBLEMS 114 | DEBUG CONSOLE

```
~/Dr/C/Tr/i/iOS-Conf-Training main ↓1↑1 +2 !2 ❯
```

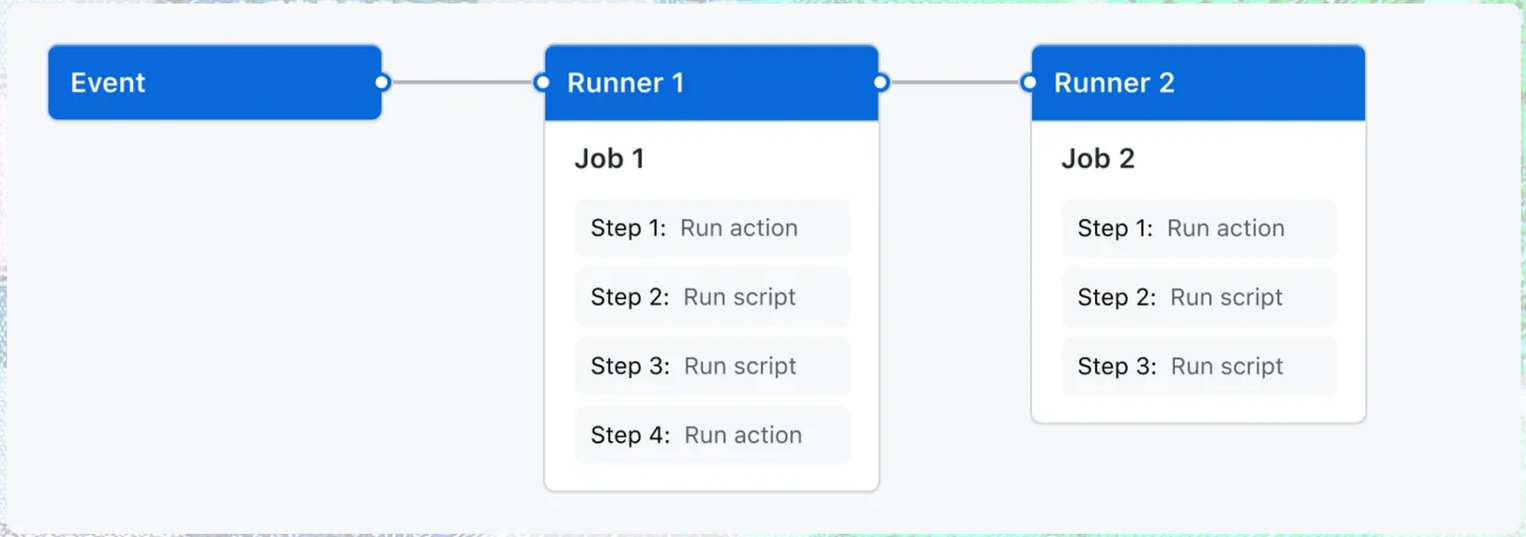
main*+ 1↓ 1↑ 0 0 114 AppScan found 0 security issues. AppScan marker On Ln 6, Col 1 Spaces: 2 UTF-8 LF YAML 1 Spell

© 2025 by Bai7 GmbH | All rights reserved

Android - Github Actions

GitHub Actions is a continuous integration and continuous delivery (CI/CD) platform that allows you to automate your build, test, and deployment pipeline.

You can create workflows that build and test every pull request to your repository, or deploy merged pull requests to production.



https://docs.github.com/en/actions/learn-github-actions/

© 2025 by Bai7 GmbH | All rights reserved

Android - Github Actions

I have decompiled the New York Times Android App and we are assessing the decompiled code for secrets.

The NYT has their own bug bounty program and vulnerability disclosure program.

The New York Times Responsible Disclosure Powered by Synack > RoE

Scope and Rules of Engagement

Rules of Engagement

- No Denial of Service testing
- No Physical or Social Engineering
- No testing of Third-party Services
- No uploading of any vulnerability or client-related content to third-party utilities (e.g. Github, DropBox, YouTube)
- All attack payload data must use professional language
- If able to gain access to a system, accounts, users, or user data, stop at point of recognition and report. Do not dive deeper to determine how much more is accessible.
- Vulnerabilities submitted through this process are considered confidential, and should not be discussed or disclosed until approved by all parties involved.

https://nytimes.responsibledisclosure.com/hc/en-us

© 2025 by Bai7 GmbH | All rights reserved

Android - Secret Scanning - Gitleaks

Open the file Lab-Android-Day-1.pdf

Section:
Lab – Secret Scanning

Note: Some PDF readers do not copy properly and miss whitespaces etc.

© 2025 by Bai7 GmbH | All rights reserved

Android - SARIF

What is SARIF?

- Static Analysis Results Interchange Format (SARIF)
- Is an output file format and defined as JSON schema
- Is trying to harmonize security tool output and is becoming more popular in various security scanning tools.
- Supported by Github Security / Code Scanning alerts


```
207        "results": [
319          {
320            "message": {
321              "text": "Insecure Implementation of SSL. Trusting all the certificates or
                 accepting self signed certificates is a critical Security Hole. This
                 application is vulnerable to MITM attacks."
322            },
323            "level": "error",
324            "locations": [
325              {
326                "physicalLocation": {
327                  "region": {
328                    "snippet": {
329                      "text": "        final SSLContext context = SSLContext.getInstance
                         (\"TLS\");\n        context.init(null, tmf.getTrustManagers(),
                         null);"
330                    },
331                    "endColumn": 58,
332                    "endLine": 95,
333                    "startColumn": 9,
334                    "startLine": 94
335                  },
336                  "artifactLocation": {
337                    "uri": "app/src/main/java/sg/vp/owasp_mobile/OMTG_Android/
                       OMTG_NETW_004_SSL_Pinning_Certificate.java"
338                  }
339                }
340              },
```


https://github.com/microsoft/sarif-tutorials

© 2025 by Bai7 GmbH | All rights reserved

Android - Secret Scanning

What to do if sensitive information was identified or committed?

Don’t panic, most likely the project was living with committed secrets in the repo already for some time and you just finally identified them.

Warning: Once you have pushed a commit to GitHub, you should consider any sensitive data in the commit compromised. If you committed a password, change it! If you committed a key, generate a new one. Removing the compromised data doesn't resolve its initial exposure, especially in existing clones or forks of your repository. Consider these limitations in your decision to rewrite your repository's history.

Don’t revoke the secret immediately! First check who is using this secret and notify them that its compromised and that the secret is about to be revoked. This is to avoid disruption to any users and services.

© 2025 by Bai7 GmbH | All rights reserved

Android - Secret Scanning

What to do if sensitive information was identified or committed?

- Generate the new secret according to best practices
- Use a secure mechanism to store the secret and make it available in the CI/CD pipeline. In our case:
  - Github Secrets - For Github actions:
    - https://docs.github.com/en/actions/deployment/targeting-different-environments/using-environments-for-deployment#environment-secrets
    - https://docs.github.com/en/actions/security-guides/encrypted-secrets
- Hashicorp Vault, AWS Secrets Manager, $cloudprovider
- Revoke the key/reset the password and make the leaked secret unusable.

© 2025 by Bai7 GmbH | All rights reserved

Android - Secret Scanning

Rewrite Git history?

- Optional: Rewrite the Git history and remove the secret
- You can re-write the GitHub commit history with tools like
  - BFG (https://rtyley.github.io/bfg-repo-cleaner/) or using
  - “git filter-repo”
- Can be a tedious task and it is unlikely that the secret get’s completely redacted (will still be in forks etc.)

Fully removing the data from GitHub

After using `git-filter-repo` to remove the sensitive data and pushing your changes to GitHub, you must take a few more steps to fully remove the data from GitHub.

1. Contact us through the GitHub Support portal, and provide the following information:
   - The owner and repository name in question (e.g. YOUR-USERNAME/YOUR-REPOSITORY).
   - The number of affected pull requests, found in the previous step. This is used by Support to verify you understand how much will be affected.
   - The First Changed Commit(s) reported by `git-filter-repo` (Look for `NOTE: First Changed Commit(s)` in its output.)
   - If `NOTE: There were LFS Objects Orphaned by this rewrite` appears in the git-filter-repo output (right after the First Changed Commit), then mention you had LFS Objects Orphaned and upload the named file to the ticket as well.

   If you have successfully cleaned up all references other than PRs, and no forks have references to the sensitive data, Support will then:

   - Dereference or delete any affected PRs on GitHub.
   - Run a garbage collection on the server to expunge the sensitive data from storage.
   - Remove cached views.
   - If LFS Objects are involved, delete and/or purge the orphaned LFS objects.

   > **Important**
   >
   > GitHub Support won't remove non-sensitive data, and will only assist in the removal of sensitive data in cases where we determine that the risk can't be mitigated by rotating affected credentials.

2. Collaborators must rebase, *not* merge, any branches they created off of your old (tainted) repository history. One merge commit could reintroduce some or all of the tainted history that you just went to the trouble of purging. They may need to take additional steps as well; see Make sure other copies are cleaned up: clones of colleagues in the `git-filter-repo` manual.

© 2025 by Bai7 GmbH | All rights reserved

Android - Secret Scanning - Challenges

How can you avoid to commit secrets?

© 2025 by Bai7 GmbH | All rights reserved

Android - Secret Scanning

Avoiding accidental commits in the future

Preventing contributors from making accidental commits can help you prevent sensitive information from being exposed. For more information see Best practices for preventing data leaks in your organization.

There are a few things you can do to avoid committing or pushing things that should not be shared:

- If the sensitive data is likely to be found in a file that should not be tracked by git, add that filename to `.gitignore` (and make sure to commit and push that change to `.gitignore` so other developers are protected).
- Avoid hardcoding secrets in code. Use environment variables, or secret management services like Azure Key Vault, AWS Secrets Manager, or HashiCorp Vault to manage and inject secrets at runtime.
- Create a pre-commit hook to check for sensitive data before it is committed or pushed anywhere, or use a well-known tool in a pre-commit hook like git-secrets or gitleaks. (Make sure to ask each collaborator to set up the pre-commit hook you have chosen.)
- Use a visual program like GitHub Desktop or gitk to commit changes. Visual programs generally make it easier to see exactly which files will be added, deleted, and modified with each commit.
- Avoid the catch-all commands `git add .` and `git commit -a` on the command line—use `git add filename` and `git rm filename` to individually stage files, instead.
- Use `git add --interactive` to individually review and stage changes within each file.
- Use `git diff --cached` to review the changes that you have staged for commit. This is the exact diff that `git commit` will produce as long as you don't use the `-a` flag.
- Enable push protection for your repository to detect and prevent pushes which contain hardcoded secrets from being committed to your codebase. For more information, see About push protection.

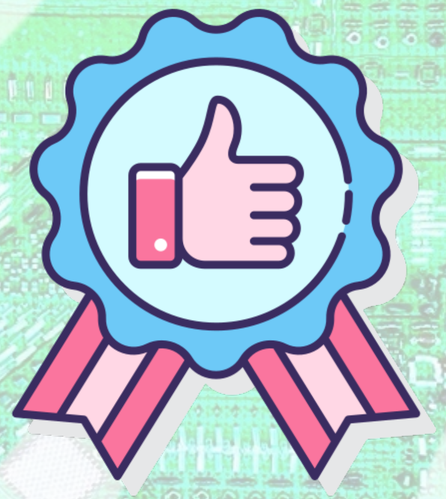
Avoiding accidental commits in the future

Best Practices for Preventing Data Leaks

© 2025 by Bai7 GmbH | All rights reserved

Android - Secret Scanning - Summary

- Use tools like git-secrets, truffleHog, or gitleaks to scan your codebase for potential exposure of secrets, both before committing and periodically as part of your CI/CD pipeline.
- Secret Scanners need maintenance and adjustment and will produce false positives, as they are based on Regular Expressions.
- Avoid hard-coding secrets directly in your source code. This includes API keys, passwords, and any other sensitive information. Once committed to a repository, it's easy for these secrets to be inadvertently exposed.
- Use .gitignore for Sensitive Files



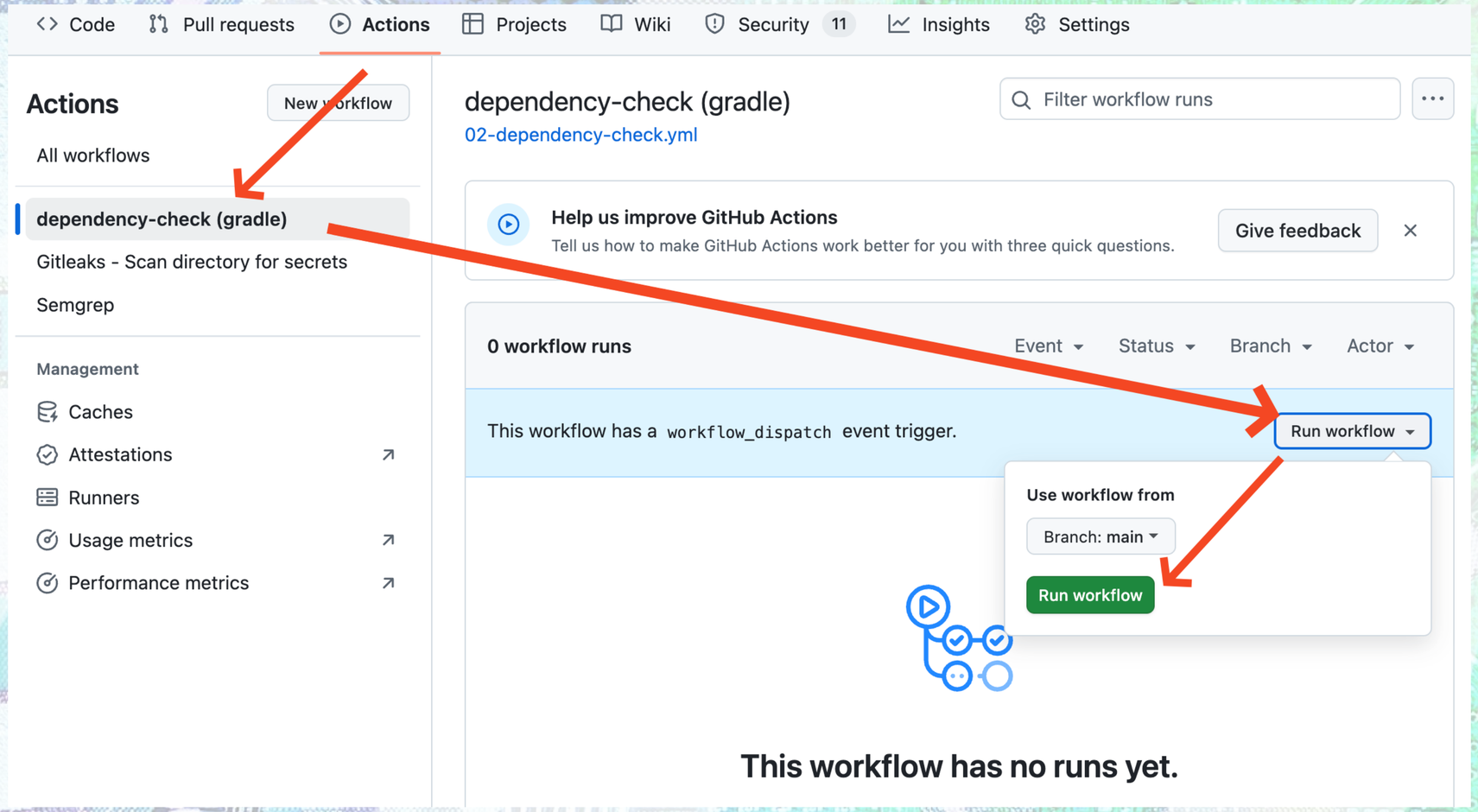
© 2025 by Bai7 GmbH | All rights reserved

Android Training

Scanning for secrets in an APK

Android - Software Composition Analysis (SCA)

As preparation for the upcoming lab, please trigger the "**dependency-check**" workflow already now in your forked repo. This first run will take around 10-15 minutes.

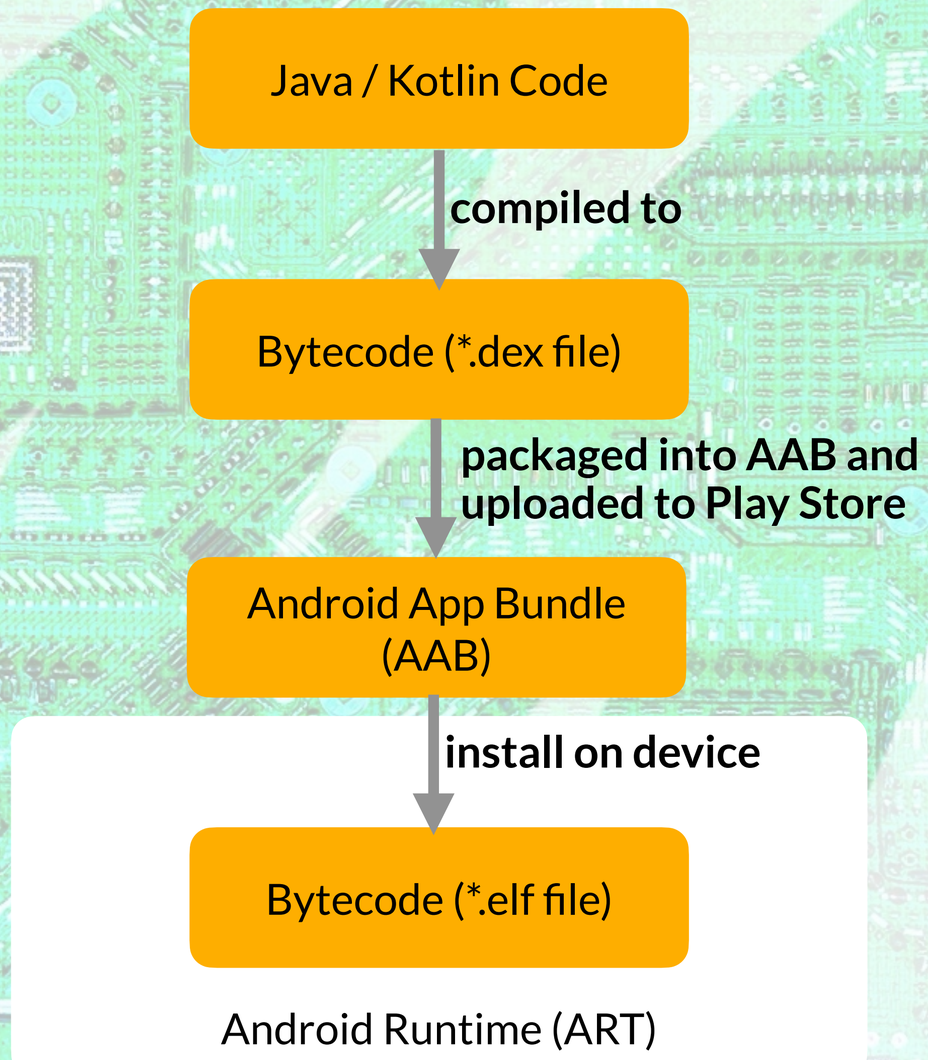


© 2025 by Bai7 GmbH | All rights reserved

Android App Creation - Acronym Jungle

Java/Kotlin source code is compiled to DEX bytecode and packaged into an APK. During installation the APK is downloaded and the Android Runtime (ART) performs the translation of the application's DEX bytecode into native instructions (ELF) through AOT, to optimise the code for the underlying hardware architecture.

- DEX means “Dalvik Executable file”
- APK means “Android Package” and is the Android App (basically a zip)
- AOT means “Ahead-of-Time” compilation, which compiles the entire application into the processor's architecture native machine code upon their installation. This will generate a so called OAT file as an intermediate step. The OAT file will be embedded into a ELF binary.
- ELF means “Executable and Linkable Format”
- ART means “Android RunTime” and executes the ELF binary

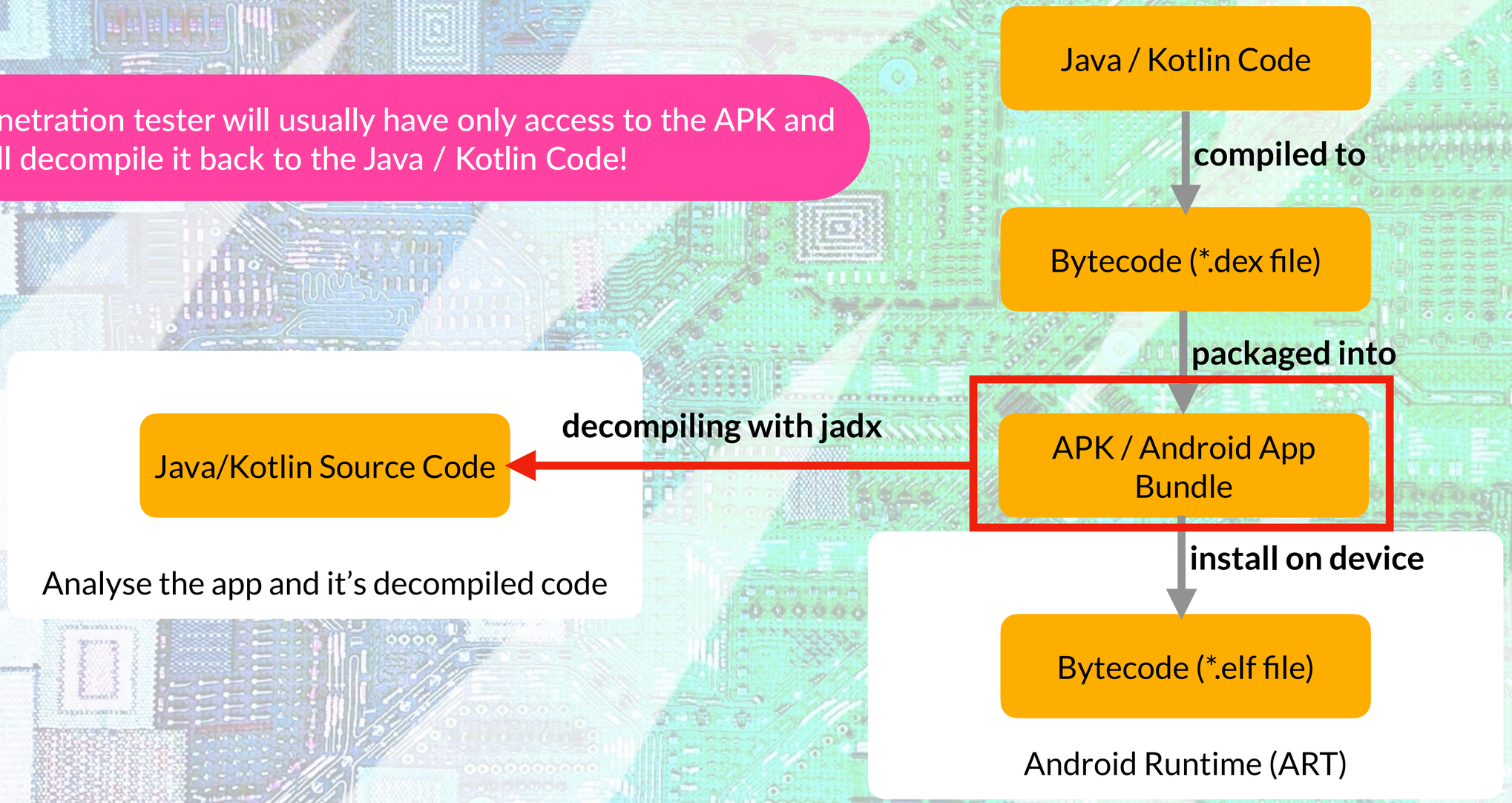


https://source.android.com/docs/core/runtime

© 2025 by Bai7 GmbH | All rights reserved

Android - Decompiling

A researcher / penetration tester will usually have only access to the APK and will decompile it back to the Java / Kotlin Code!



© 2025 by Bai7 GmbH | All rights reserved

Android - Decompiling with Jadx

JADX

build passing | contributors 130 | downloads 3.2M
downloads@latest 106k | release v1.5.2 | maven-central v1.5.2
Java 11+ | license apache

jadx - Dex to Java decompiler

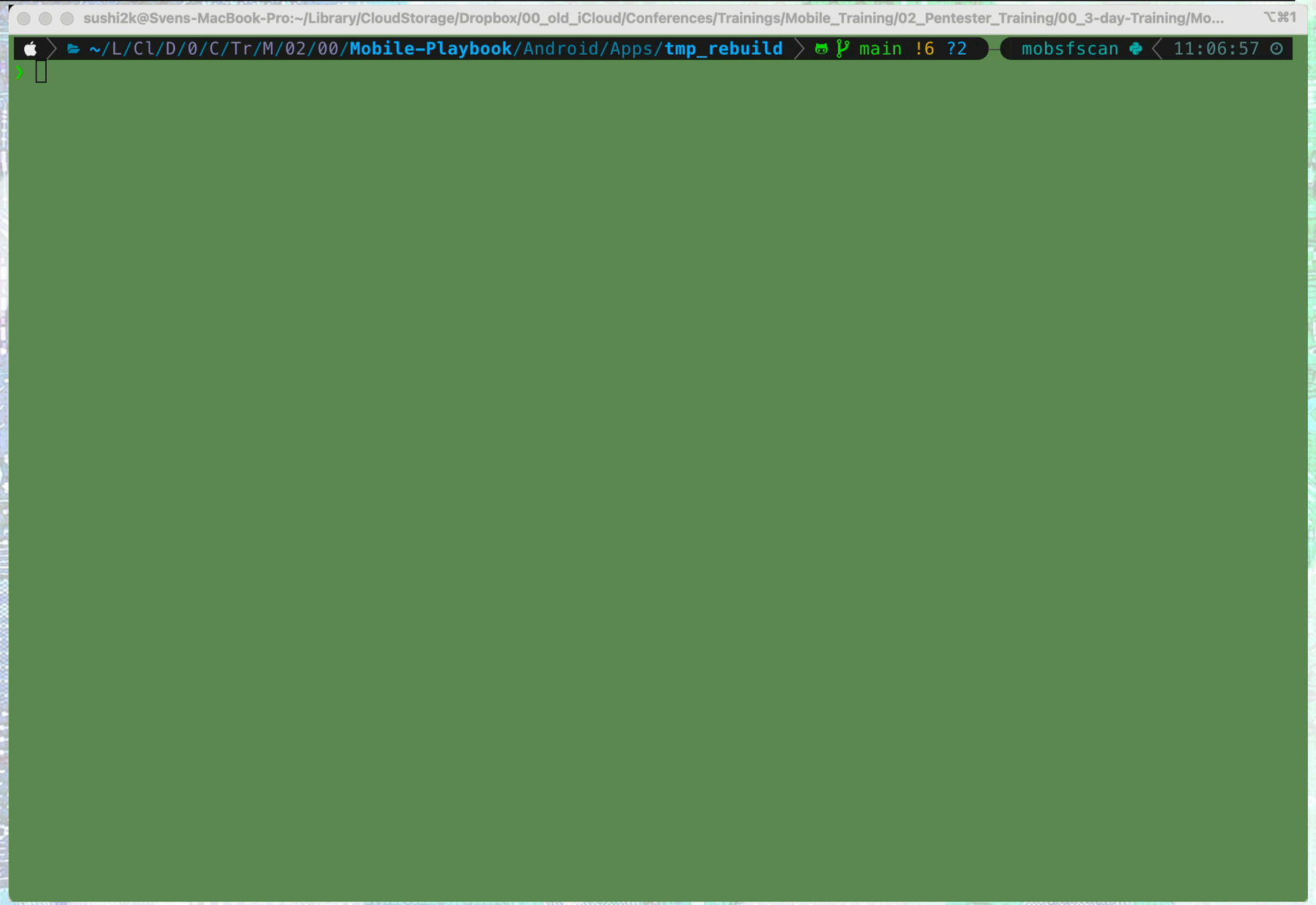
Command line and GUI tools for producing Java source code from Android Dex and Apk files

> ⚠ **Warning**
>
> Please note that in most cases **jadx** can't decompile all 100% of the code, so errors will occur.
> Check Troubleshooting guide for workarounds.

Main features:

- decompile Dalvik bytecode to Java code from APK, dex, aar, aab and zip files
- decode `AndroidManifest.xml` and other resources from `resources.arsc`
- deobfuscator included

https://github.com/skylot/jadx



© 2025 by Bai7 GmbH | All rights reserved

Scanning for secrets in an APK - apkleaks

Secret scanning is a critical security task during (mobile app) development, aimed at identifying and eliminating the risk of exposing sensitive data within the app's codebase and binaries.

Apkleaks is an example of a secret scanning tool, that allows you to scan an APK file for:

- URIs,
- endpoints and
- secrets.

```
~ ❯ apkleaks -h
v2.6.2
--
Scanning APK file for URIs, endpoints & secrets
(c) 2020-2024, dwisiswant0

usage: apkleaks [-h] -f FILE [-o OUTPUT] [-p PATTERN] [-a ARGS] [--json]

options:
  -h, --help            show this help message and exit
  -f FILE, --file FILE  APK file to scanning
  -o OUTPUT, --output OUTPUT
                        Write to file results (random if not set)
  -p PATTERN, --pattern PATTERN
                        Path to custom patterns JSON
  -a ARGS, --args ARGS  Disassembler arguments (e.g. --threads-count 5 --deobf)
  --json                Save as JSON format
```

https://github.com/dwisiswant0/apkleaks

© 2025 by Bai7 GmbH | All rights reserved

Scanning for secrets in an APK - apkleaks

The apkleaks scan is already done for you!
But you can try on your own if you can install jadx and apkleaks locally.

1. You can install apkleaks by following the install guide:

https://github.com/dwisiswant0/apkleaks?tab=readme-ov-file#installation

2. You can install jadx by following the install guide:

https://github.com/skylot/jadx?tab=readme-ov-file#download

3. Afterwards, you can start by opening a terminal and executing the following commands:

```
$ cd ~/mobile-playbook-dev-android
$ apkleaks -f com.nytimes.android.rebuild.apk -o nyt-apkleaks.txt
```

The scan will take a few minutes!

```
┌──(mastg㉿mobile-playbook)-[~/MASTG-Hands-on/Android/Apps]
└─$ apkleaks -o nyt-apkleaks.txt  -f com.nytimes.android.apk
v2.6.3
--
Scanning APK file for URIs, endpoints & secrets
(c) 2020-2024, dwisiswant0

** Decompiling APK...
INFO  - loading ...
INFO  - processing ...
ERROR - finished with errors, count: 49

** Scanning against 'com.nytimes.android'
```

© 2025 by Bai7 GmbH | All rights reserved

Android - Lab Time

Open the file Lab-Android-Day-1.pdf

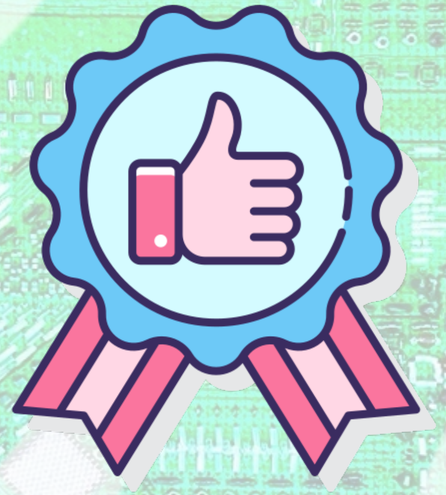
Section:
Lab – Scanning an APK for secrets

Note: Some PDF readers do not copy properly and miss whitespaces etc.

© 2025 by Bai7 GmbH | All rights reserved

Scanning for secrets in an APK - Summary

- Secret Scanning is an essential task when analysing mobile apps.
- False Positives are common in secret scanning, as the detection is usually based on Regular Expressions (RegEx).
- Validate the secrets to identify the risk, not every exposed secret is a security issue!
- Find tools that allow you to add your own rules for scanning, like in APKLeaks.
- Scan not only your git repo and source code for secrets, but also the APK.
- Besides APKLeaks, there are other useful open-source tools for secret scanning:
  - Gitleaks
  - Trufflehog



© 2025 by Bai7 GmbH | All rights reserved

Android Training

Software Composition Analysis (SCA) of Android Apps

Android - Software Composition Analysis (SCA)

- Majority of software consists of (open source) libraries; mobile apps are no exception.
- Libraries ensure a consistent approach and avoid “reinventing the wheel”.
- For security critical operations, like encryption, it is encouraged to use libraries.
- Also libraries can have security vulnerabilities that need to be detected!

- SCA allows you to:
  - Identity open source and 3rd party libraries in your code base / binary
  - Scan these libraries for known security vulnerabilities
  - Inspect the license information of open source libraries to highlight the limitations and obligations when using it in your software.

© 2025 by Bai7 GmbH | All rights reserved

Android - Software Composition Analysis (SCA)

How does SCA work? It inspects:

- Package Managers
- Manifest files
- Source Code
- Binary files and more

This produces a Bill of Material (BOM), which includes the name and version of the library which is compared against various open databases that hold information of known vulnerabilities.

This allows SCA tools to identify known vulnerabilities in the libraries being used.

© 2025 by Bai7 GmbH | All rights reserved

Android - Software Composition Analysis (SCA)

Terminology:

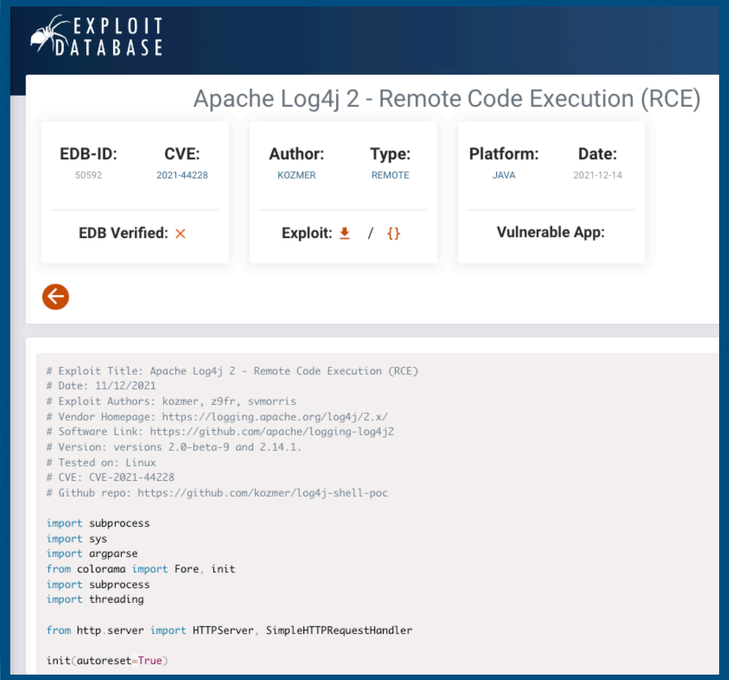
Vulnerability

- The identified weakness
- Example of vulnerabilities: SQL-Injection, Cross-Site Scripting (XSS), etc.

Exploit:

- Abusing the vulnerability
- Example:
  - Exploiting SQL Injection to dump the entire database
  - Exploiting XSS vulnerability to steal user credentials

Example of a public exploit:



https://www.exploit-db.com/exploits/50592

© 2025 by Bai7 GmbH | All rights reserved

Android - Software Composition Analysis (SCA)

OWASP Dependency Check is an open source tool to scan libraries (dependencies) in software projects to detect publicly disclosed vulnerabilities in them.

OWASP dependency-check-cli is a command line tool and can be installed on all main operating systems.

The tool will generate a report listing the dependency and the associated Common Vulnerability and Exposure (CVE) and CVSS entries.

© 2025 by Bai7 GmbH | All rights reserved

Android - Software Composition Analysis (SCA)

Common Vulnerability and Exposure (CVE) is a list of publicly disclosed computer security flaws. For example CVE-2021-45046 for log4j:

CVE-2021-45046

Summary	Thread Context Lookup is vulnerable to remote code execution in certain configurations
CVSS 3.x Score & Vector	9.0 CRITICAL (CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:H/I:H/A:H)
Components affected	`log4j-core`
Versions affected	`[2.0-beta9, 2.3.1) ∪ [2.4, 2.12.3) ∪ [2.13.0, 2.17.0)`
Versions fixed	`2.3.1` (for Java 6), `2.12.3` (for Java 7), and `2.17.0` (for Java 8 and later)

Common Vulnerability Scoring System (CVSS) is an open framework for communicating the characteristics and severity of software vulnerabilities.

CVSS v4.0 Ratings

Severity	Severity Score Range
None*	0.0
Low	0.1-3.9
Medium	4.0-6.9
High	7.0-8.9
Critical	9.0-10.0

© 2025 by Bai7 GmbH | All rights reserved

Android - Software Composition Analysis (SCA)

Build automation tools like Gradle or Maven are loading libraries for Android during build time.

Gradle:

```
implementation(“com.squareup.okhttp3:okhttp:4.9.1")
```

Maven:

```
<dependency>
  <groupId>com.squareup.okhttp3</groupId>
  <artifactId>okhttp</artifactId>
  <version>4.9.1</version>
</dependency>
```

© 2025 by Bai7 GmbH | All rights reserved

Android - Software Composition Analysis (SCA) in Android Studio

deeplink | main | Medium Phone API 35 | app

Android

- app
  - manifests
  - kotlin+java
    - at.bai7.deeplink
      - ui.theme
      - MainActivity.kt
      - TokenBridge
    - at.bai7.deeplink (androidTest)
    - at.bai7.deeplink (test)
  - res
- Gradle Scripts
  - build.gradle.kts (Project: deeplink)
  - build.gradle.kts (Module :app)
  - proguard-rules.pro (ProGuard Rules for ":ap
  - gradle.properties (Project Properties)
  - gradle-wrapper.properties (Gradle Version)
  - libs.versions.toml (Version Catalog "libs")

build.gradle.kts (:app) | gradle-wrapper.properties | build.gradle.kts (deeplink) | MainActivity.kt

You can use the Project Structure dialog to view and edit your project configuration Open (⌘;) Hide notification

```
8   dependencyCheck {
13          assemblyEnabled = false  // disables .NET assembly analyzer
14          nodeAuditEnabled = false // disables node analyzer
15      }
16
17
18      nvd {
19          apiKey = "e2e35caf-3da5-403b-b4f4-4d3973d8b225"
20          delay = 16000
21      }
22
23      //    suppressionFile = "suppression.xml"
24
25  }
```

Terminal | Local

~/L/Cl/D/0/C/Trainings/Mobile_Training/10_Source_Code/tasky-android/tasky main !3 15:26:28

tasky > app > build.gradle.kts > dependencyCheck 17:1 LF UTF-8 4 spaces

© 2025 by Bai7 GmbH | All rights reserved

Android - Software Composition Analysis (SCA)

Dependency-check report and results:

DEPENDENCY-CHECK

Dependency-Check is an open source tool performing a best effort analysis of 3rd party dependencies; false positives and false negatives may exist in the analysis performed by the tool. Use of the tool and the reporting provided constitutes acceptance use in an AS IS condition, and there are NO warranties, implied or otherwise, with regard to the analysis or its use. Any use of the tool and the reporting provided is at the user's risk. In no event shall the copyright holder or OWASP be held liable for any damages whatsoever arising out of or in connection with the use of this tool, the analysis performed, or the resulting report.

How to read the report | Suppressing false positives | Getting Help: github issues

Sponsor

Project: project ':app'

deeplink:app:unspecified

Scan Information (show all):

- *dependency-check version*: 12.1.1
- *Report Generated On*: Sun, 3 Aug 2025 15:27:29 -0700
- *Dependencies Scanned*: 325 (257 unique)
- *Vulnerable Dependencies*: 37
- *Vulnerabilities Found*: 116
- *Vulnerabilities Suppressed*: 0
- ...

Summary

Display: Showing Vulnerable Dependencies (click to show all)

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count
android-device-provider-local-0.0.9-alpha02.jar (shaded: com.google.guava:guava:28.1-jre)	cpe:2.3:a:google:guava:28.1:*:*:*:*:*:*:*	pkg:maven/com.google.guava/guava@28.1-jre	HIGH	2
commons-io-2.13.0.jar	cpe:2.3:a:apache:commons_io:2.13.0:*:*:*:*:*:*:*	pkg:maven/commons-io/commons-io@2.13.0	MEDIUM	1
core-0.0.9-alpha02.jar (shaded: io.netty:netty-buffer:4.1.72.Final)	cpe:2.3:a:netty:netty:4.1.72:*:*:*:*:*:*:*	pkg:maven/io.netty/netty-buffer@4.1.72.Final	HIGH*	5
core-0.0.9-alpha02.jar (shaded: io.netty:netty-codec-http2:4.1.72.Final)	cpe:2.3:a:netty:netty:4.1.72:*:*:*:*:*:*:*	pkg:maven/io.netty/netty-codec-http2@4.1.72.Final	HIGH*	5
core-0.0.9-alpha02.jar (shaded: io.netty:netty-codec-http:4.1.72.Final)	cpe:2.3:a:netty:netty:4.1.72:*:*:*:*:*:*:*	pkg:maven/io.netty/netty-codec-http@4.1.72.Final	HIGH*	6

core-0.0.9-alpha02.jar (shaded: io.netty:netty-codec-http:4.1.72.Final)

File Path: /Users/sushi2k/.gradle/caches/modules-2/files-2.1/com.google.testing.platform/core/0.0.9-alpha02/8d72767495b76dfdfe1a4f02bab67cb035da816d/core-0.0.9-alpha02.jar/META-INF/maven/io.netty/netty-codec-http/pom.xml
MD5: e15408c965bab2841440316319ec55cf
SHA1: 84ad0a0bbd43a62df88b74749b8763da5124e0a4
SHA256:a8cd15659a5720013918843fc8bf29f38a81b83e34816f82e397a4db3c864f53
Referenced In Project/Scope: app:_internal-unified-test-platform-core

Evidence

Identifiers

- pkg:maven/io.netty/netty-codec-http@4.1.72.Final (*Confidence*:High)
- cpe:2.3:a:netty:netty:4.1.72:*:*:*:*:*:*:* (*Confidence*:Highest) suppress

Published Vulnerabilities

CVE-2022-41881 suppress

Netty project is an event-driven asynchronous network application framework. In versions prior to 4.1.86.Final, a StackOverflowError can be raised when parsing a malformed crafted message due to an infinite recursion. This issue is patched in version 4.1.86.Final. There is no workaround, except using a custom HaProxyMessageDecoder.

CWE-674 Uncontrolled Recursion

CVSSv3:

- Base Score: HIGH (7.5)
- Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H/E:3.9/RC:R/MAV:A

© 2025 by Bai7 GmbH | All rights reserved

Android - Software Composition Analysis (SCA)

Why are there so many findings?

© 2025 by Bai7 GmbH | All rights reserved

Android - Software Composition Analysis (SCA)

OWASP Dependency Check is in this scan building the app and is scanning the ./gradle/cache folder and NOT the AAB/APK that is being generated afterwards.

This means all dependencies downloaded by Gradle will be scanned, regardless if they are part of the Android app or not.

That's why you need to maintain a suppression.xml file to focus on the things that could be a vulnerability in the app.

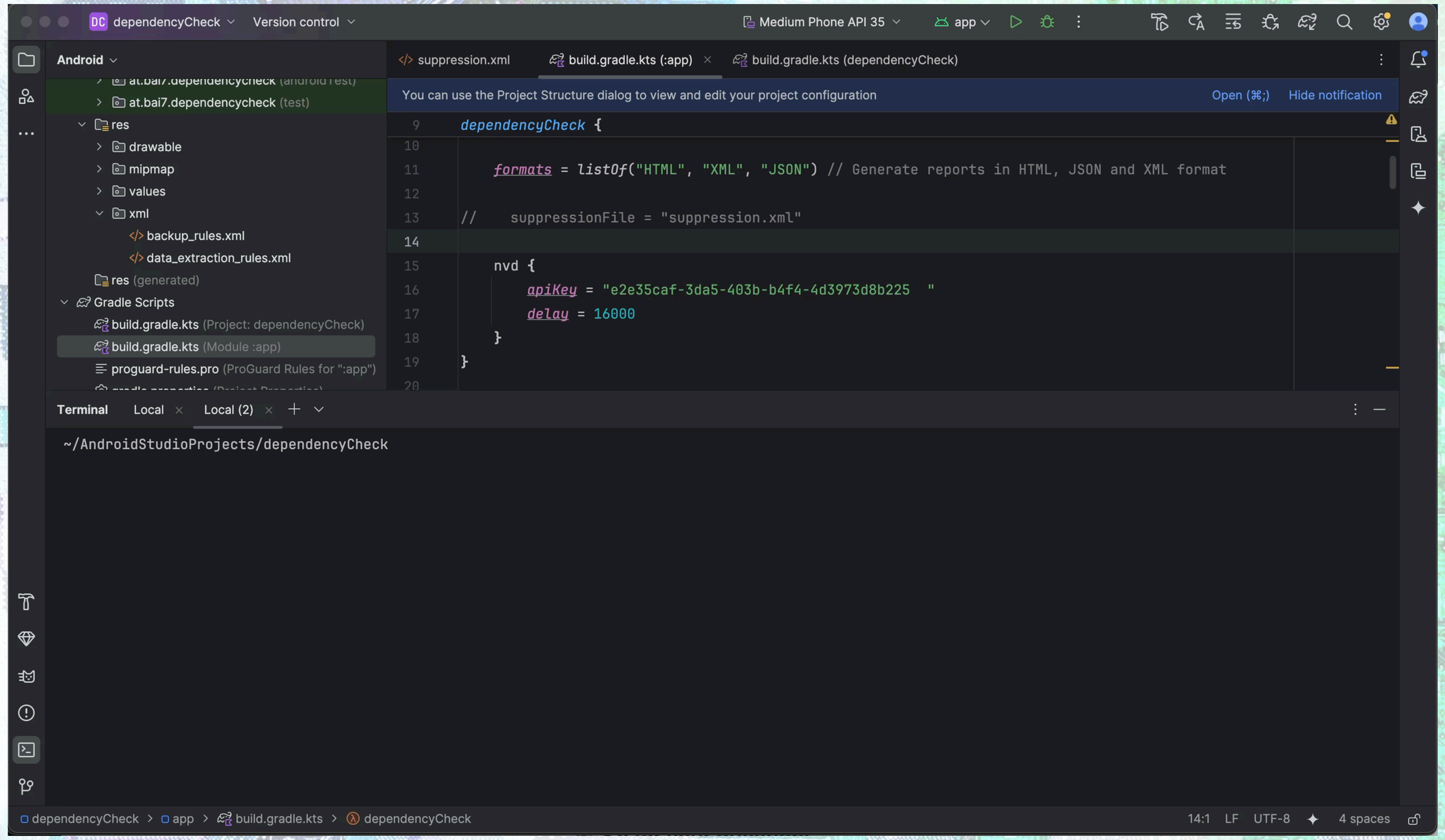
© 2025 by Bai7 GmbH | All rights reserved

Android - Software Composition Analysis (SCA)

How to remove the false positives?

© 2025 by Bai7 GmbH | All rights reserved

Android - Software Composition Analysis (SCA) in Android Studio



© 2025 by Bai7 GmbH | All rights reserved

Android - Lab Time

Open the file Lab-Android.pdf

Section:
Lab – Software Composition Analysis (SCA)

Note: Some PDF readers do not copy properly and miss whitespaces etc.

© 2025 by Bai7 GmbH | All rights reserved

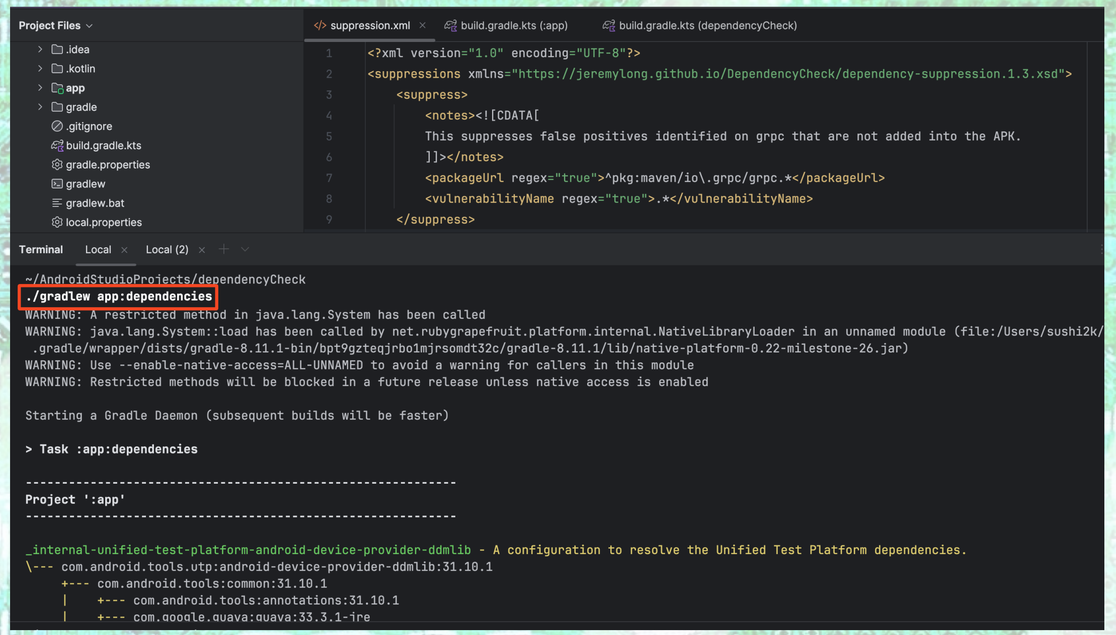
Android - Software Composition Analysis (SCA)

Executing the following gradle task will display a tree of all dependencies used by your app module.

```
./gradlew app:dependencies
```

“releaseRuntimeClasspath” in Gradle for Android is the set of runtime dependencies that your Android app will use when you build it in release mode.

```
releaseRuntimeClasspath - Runtime classpath of '/release'.
+--- org.jetbrains.kotlin:kotlin-stdlib:2.0.21
|    +--- org.jetbrains:annotations:13.0 -> 23.0.0
|    +--- org.jetbrains.kotlin:kotlin-stdlib-common:2.0.21 (c)
|    +--- org.jetbrains.kotlin:kotlin-stdlib-jdk8:1.8.0 -> 1.8.20 (c)
|    \--- org.jetbrains.kotlin:kotlin-stdlib-jdk7:1.8.0 -> 1.8.20 (c)
...
\--- com.squareup.okhttp3:okhttp:4.9.1
     +--- com.squareup.okio:okio:2.8.0
     |    +--- org.jetbrains.kotlin:kotlin-stdlib:1.4.0 -> 2.0.21 (*)
     |    \--- org.jetbrains.kotlin:kotlin-stdlib-common:1.4.0 -> 2.0.21 (*)
     \--- org.jetbrains.kotlin:kotlin-stdlib:1.4.10 -> 2.0.21 (*)
```



https://developer.android.com/build/gradle-dependency-resolution

© 2025 by Bai7 GmbH | All rights reserved

Android - Software Composition Analysis (SCA)

Local Scan:

- The first scan will take a while as the tool is downloading all public known vulnerabilities (CVE) from National Vulnerability Database (NVD) and is building up its local database.
- Subsequent scans will only download the diff of new CVEs and will be faster.

Build Scan:

There are several plugins available to execute the scan during build time, for example Gradle, Jenkins or Maven.

© 2025 by Bai7 GmbH | All rights reserved

Android - Software Composition Analysis (SCA)

How to fix identified vulnerabilities?

Several options:

- Upgrade the library to the “nearest” version that is free of vulnerabilities, e.g. vulnerable version is 2.4.5 and fixed versions are 2.4.6 and 3.0.1. Major version version upgrades might have many changes in the usage of the API and might need refactoring of the code, so upgrading to 2.4.6 might be the easiest way to upgrade.
- Prioritize fixing of vulnerabilities with the highest impact to your apps, as not all vulnerabilities can be fixed on the same time, by
  - Checking if there are public exploits; can be done through EPSS https://www.first.org/epss/
  - Is the vulnerable function in the library called in your code (some enterprise tools support this)

© 2025 by Bai7 GmbH | All rights reserved

Android - Software Composition Analysis (SCA)

How to fix identified vulnerabilities?

How do you know which library is free of vulnerabilities? => Manual research or enterprise tools.

Commercial products (sorted alphabetically) can detect outdated libraries and offer an overview of library versions that are free of known vulnerabilities:

- Blackduck | Github Dependabot | Snyk | Sonatype Nexus IQ | $vendor

In case there is no fix available for the library:

- Remove the library (if not needed)
- Change to another library
- Accept the risk and security team will track it in their risk register

© 2025 by Bai7 GmbH | All rights reserved

Android - Software Composition Analysis (SCA) - Summary

- SCA (Software Composition Analysis) => security scanning of 3rd party libraries
  - On OS level (Debian or Red Hat packages)
  - On application level (package managers such as Maven, Node.JS or PyPI).
- Very effective way to identify vulnerabilities
- Vulnerable libraries should be fixed or removed

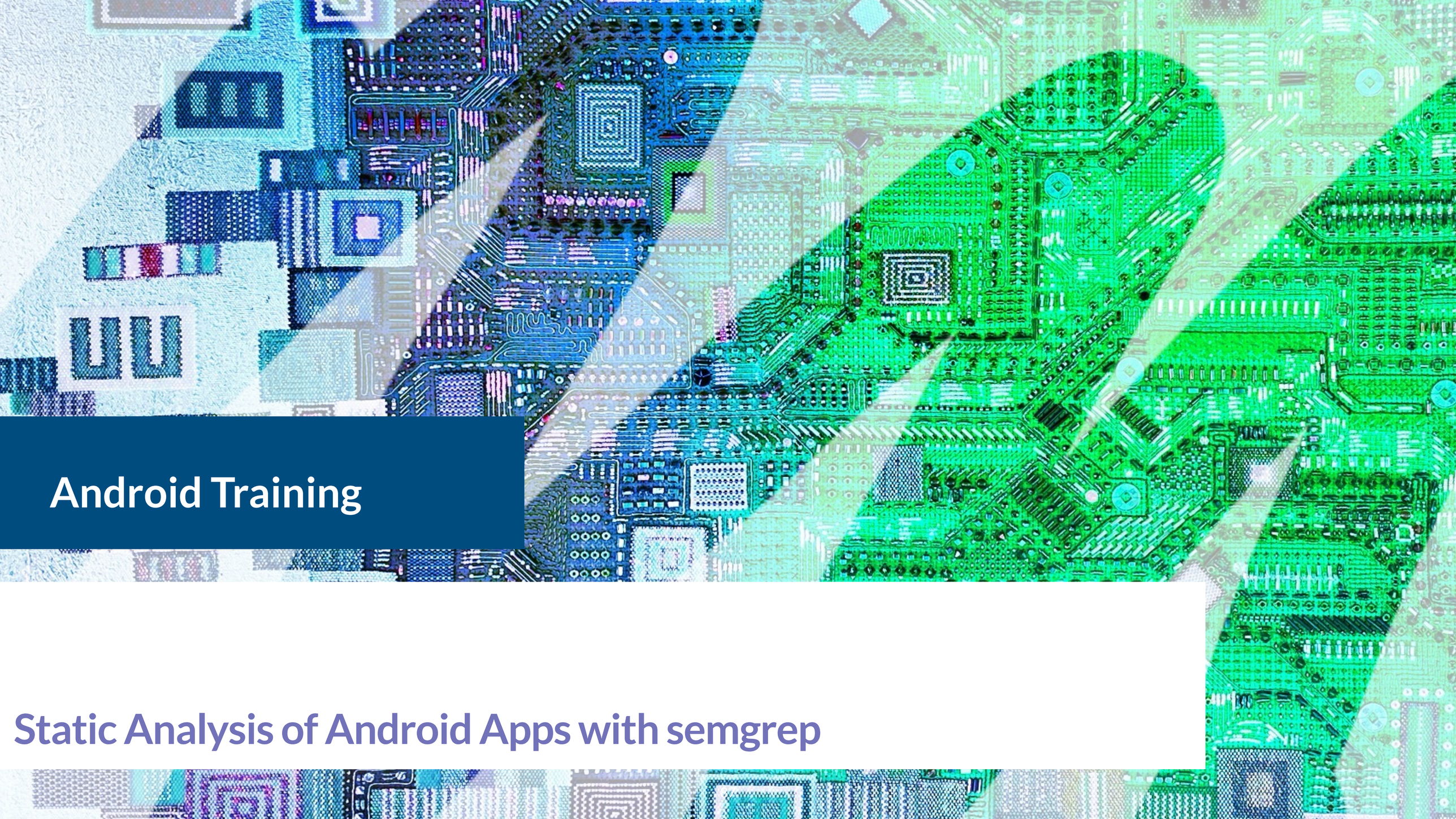
© 2025 by Bai7 GmbH | All rights reserved

Android - Software Composition Analysis (SCA) - Summary

Evaluate tool support for SCA:

- Identify which package managers you are using (for server-side and app-side)
- All SAST tools also cover SCA, find a tool that covers both and that can give you full coverage of your tech stack
- Find a tool that can show you if the vulnerable library is called and therefore exploitable to prioritise the right libraries that should be fixed.
- Integrate it into your build pipeline and scan frequently (e.g., daily)

© 2025 by Bai7 GmbH | All rights reserved



Android Training

Static Analysis of Android Apps with semgrep

Android - Static Application Security Testing (SAST) - Terminology

Linting	Static Testing	SCA	Dynamic Testing
Continuous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs or code smells.	Static Application Security Testing (SAST) involves examining an application's components without executing them, by analyzing the source code only.	Software Composition Analysis (SCA) is scanning the open-source libraries used in your software for known security vulnerabilities.	Dynamic Application Security Testing (DAST) involves examining the app during runtime (Ideally done during QA). This can be done automatically by using a tool or manually. through a penetration test.
SonarQube	semgrep, mobsfscan, MobSF	dependency-checker	MobSF, Corellium, NowSecure

© 2025 by Bai7 GmbH | All rights reserved

Android - Static Application Security Testing (SAST)

Static Application Security Testing (SAST) involves examining an application's components without executing them, by analyzing the source code only.

Strengths

- Scales well – can be run on various software, and can be run repeatedly (on PR level or continuous integration).
- Identifies certain well-known mobile vulnerabilities, such as misconfigurations (network, cryptography).
- Easy to trace - SAST tools highlight the problematic code, by filename, function and line number.

Weaknesses

- Difficult to automate for many types of security vulnerabilities, like AuthN or AuthZ
- High numbers of false positives
- Difficult to ‘prove’ that an identified security issue is an actual vulnerability.
- Manual customizing and ongoing maintenance

© 2025 by Bai7 GmbH | All rights reserved

Android - Static Application Security Testing (SAST) - semgrep

What is semgrep?

- **Static Code Analysis Tool:** Semgrep is a lightweight static analysis tool that scans code to identify security vulnerabilities, bugs, and patterns in multiple programming languages.
- **Customizable Rules:** It allows users to write and customize rules in a simple syntax, making it flexible for catching specific code patterns or potential issues in large codebases.
- **Open Source**: Semgrep is open-source, runs locally without dependencies, and is designed to be fast, making it suitable for integration into CI/CD pipelines for continuous security checks.

Semgrep

https://github.com/semgrep/semgrep

© 2025 by Bai7 GmbH | All rights reserved

Android - Static Analysis

This Github project is a compilation of Semgrep rules derived from the OWASP Mobile Application Security Testing Guide (MASTG) specifically for Android applications.

https://github.com/mindedsecurity/semgrep-rules-android-security/blob/main/rules/platform/mstg-platform-2.2.yaml

mindedsecurity / semgrep-rules-android-security

Type / to search

Code | Issues | Pull requests 1 | Actions | Projects | Security | Insights

Files

main

Go to file t

- mstg-code-8.4.java
- mstg-code-8.4.yaml
- mstg-code-8.5.java
- mstg-code-8.5.yaml
- crypto
- network
- platform
  - .gitkeep
  - mstg-platform-1.xml
  - mstg-platform-1.yaml
  - mstg-platform-2.1.java
  - mstg-platform-2.1.yaml
  - mstg-platform-2.2.java
  - mstg-platform-2.2.yaml
  - mstg-platform-2.3.java
  - mstg-platform-2.3.yaml
  - mstg-platform-2.4.xml
  - mstg-platform-2.4.yaml

semgrep-rules-android-security / rules / platform / mstg-platform-2.2.yaml

gand3lf mstg-platform-2 10eeff0 · 4 months ago History

Code | Blame | 43 lines (43 loc) · 1.57 KB | Raw

```yaml
rules:
  - id: MSTG-PLATFORM-2.2
    severity: WARNING
    languages:
      - java
    metadata:
      authors:
        - Riccardo Cardelli @gand3lf (IMQ Minded Security)
      owasp-mobile: M7
      category: security
      area: platform
      verification-level:
        - L1
        - L2
      references:
        - https://github.com/OWASP/owasp-mastg/blob/v1.5.0/Document/0x05h-Testing-Platform-Interaction.md#testing-for-i
    message: Identified the potential presence of SQL Injection in the ContentProvider class "$CLASS".
    options:
      symbolic_propagation: true
    mode: taint
    pattern-sources:
        - patterns:
            - pattern-inside: public class $CLASS extends ContentProvider{...}
            - patterns:
                - pattern-inside: $METHOD(..., $A, ...){...}
                - metavariable-regex:
                    metavariable: $METHOD
                    regex: (query|insert|update|delete|getType|getTypeAnonymous)
                - focus-metavariable: $A
    pattern-sinks:
      - pattern: $DB.delete(...)
      - pattern: $DB.update(...)
      - pattern: $DB.updateWithOnConflict(...)
      - pattern: $DB.execSQL(...)
      - pattern: $DB.rawQuery(...)
      - pattern: $DB.query(...)
      - pattern: $DB.queryWithFactory(...)
      - pattern: $DB.rawQueryWithFactory(...)
      - patterns:
        - pattern: <... "$X" ...>
        - metavariable-regex:
            metavariable: $X
            regex: .*(?i)((WHERE )|(AND )|(OR )|(SELECT )|(INSERT INTO )|(CREATE TABLE )|(DROP TABLE )|(ALTER TABLE )|(
```

© 2025 by Bai7 GmbH | All rights reserved

Android - Static Analysis

In the following lab I decompiled two apps (Finstergram and Kotlin-Shack) into their source code by using jadx. This is what attackers can do, to do a SAST scan against your APK.

The scan with semgrep is targeted against the Finstergram apps source code only, not against other packages

The scan with semgrep is targeted against the Kotlin-Shack apps source code only, not against other packages.

.github > workflows > 03-semgrep.yml

```yaml
26  jobs:
27    semgrep:
40      steps:
43        # Run the "semgrep ci" command on the command line of the docker image.
44        - run: >
45            semgrep -c semgrep-rules-android-security/rules
46            Finstergram/sources/com/netlight/sec/finstergram
47            --sarif-output=finstergram.sarif --sarif
48
49
50        - run: >
51            semgrep -c semgrep-rules-android-security/rules
52            Kotlin-Shack/sources/org/owasp/mstgkotlin
53            --sarif-output=Kotlin-Shack.sarif --sarif
```

© 2025 by Bai7 GmbH | All rights reserved

Android - Lab Time

Open the file Lab-Android-Day-1.pdf

Section:
Lab - Static Scanning with semgrep

Note: Some PDF readers do not copy properly and miss whitespaces etc.

© 2025 by Bai7 GmbH | All rights reserved

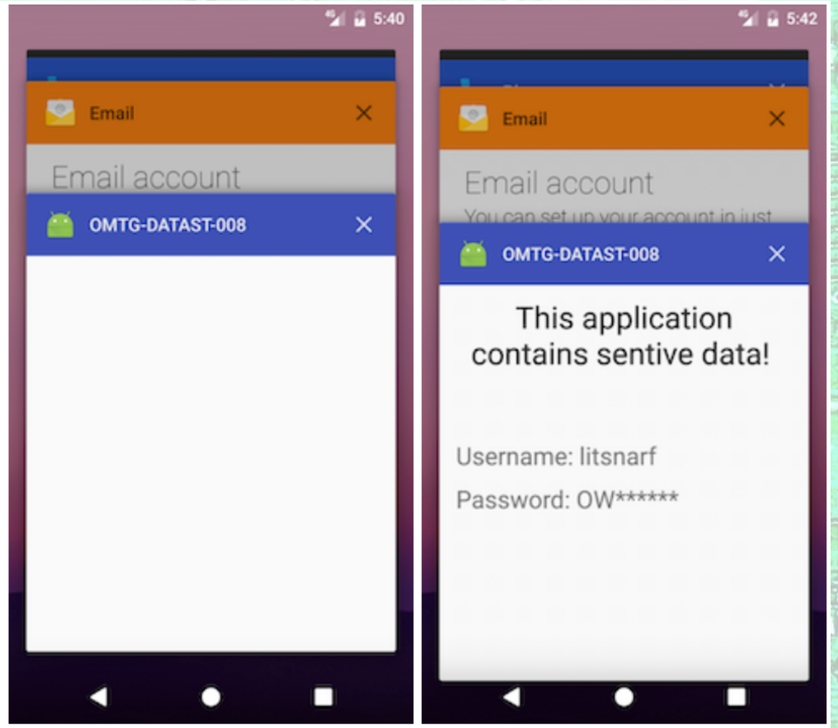
Android - Static Analysis of Android App – Auto-Generated Screenshots

A screenshot of the current activity is taken when an Android app goes into background and displayed for aesthetic purposes when the app returns to the foreground. However, this may leak sensitive information.

Best Practice (for sensitive apps like mobile banking):

- Use "FLAG_SECURE" to use a so called "secure surface" that will be shown (white screen, company logo etc.)

Note: Blocking screenshots might cause troubles for users when debugging, if no screenshot can be taken!



https://developer.android.com/about/versions/14/features/screenshot-detection

© 2025 by Bai7 GmbH | All rights reserved

Android - Static Analysis of Android App – Tapjacking

"The tapjacking attack works by cleverly arranging a non-tappable foreground layer to fool the user into tapping on an object in the background layer"

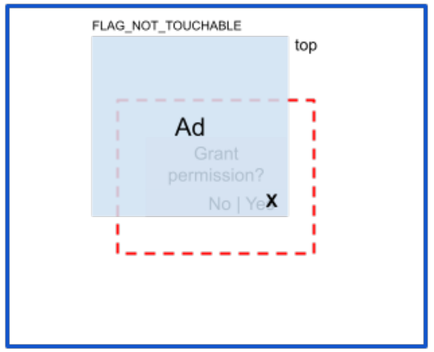
- Full and partial occlusion tapjacking attacks are often used by malware
- Full occlusion tapjacking attacks can be blocked in your code by setting:

```
View.setFilterTouchesWhenObscured(true)
```

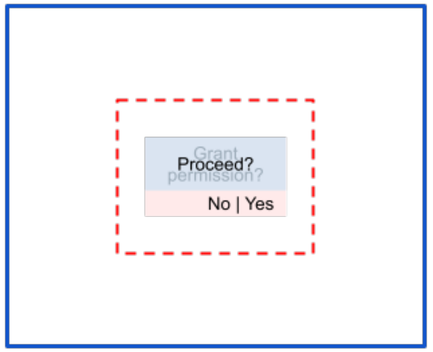
- Exploitability depends on Android version, recent versions >= Android 12 have secure defaults against Tapjacking, but are not a silver bullet.

Detailed overview of Tapjacking attacks by Jeroen Beckers.

In full occlusion, the attacker overlays the touch area to hijack the touch event:



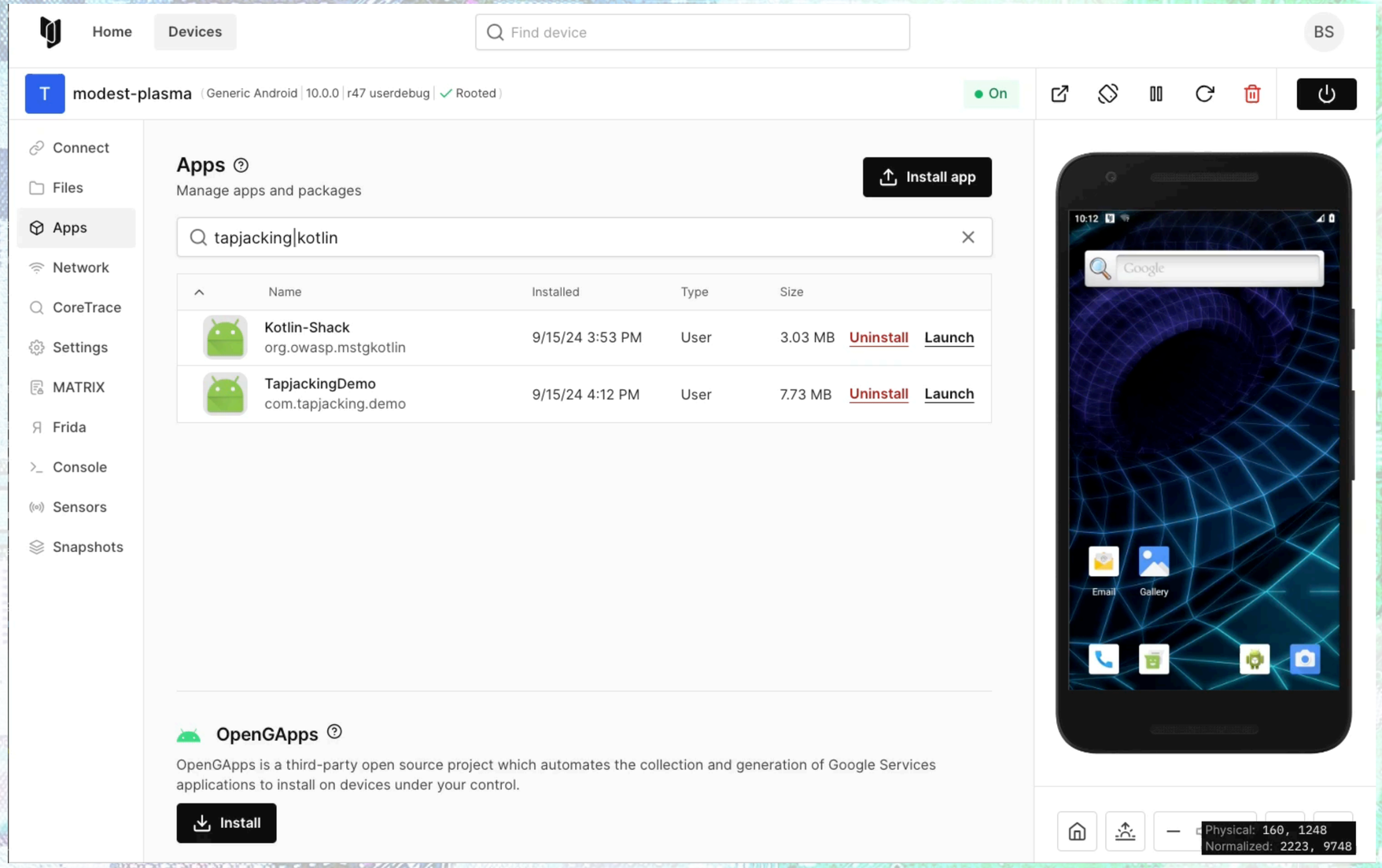
In partial occlusion attacks, the touch area remains unobscured:



https://developer.android.com/privacy-and-security/risks/tapjacking

© 2025 by Bai7 GmbH | All rights reserved

Android - Static Analysis of Android App – Tapjacking



https://github.com/carlospolop/Tapjacking-ExportedActivity

© 2025 by Bai7 GmbH | All rights reserved

Android - Static Analysis of Android App – Insecure Random Number Generator

Not every usage of “weak” cryptography paradigms like a weak PRNG (Pseudo-Random-Number-Generator) is a security issue. It depends on the context on where it is used. If it’s not a security context it shouldn't be raised as a finding.

In case it is a security context, the following should be used:

- Use `java.security.SecureRandom` when there are *security implications*
- Use `java.util.Random` for any other cases
- **Never** use `Math.random`!

https://developer.android.com/privacy-and-security/risks/weak-prng

© 2025 by Bai7 GmbH | All rights reserved

Android - Static Analysis of Android App – SQL Injection

SQL Injection available, as rawQuery is used, see here. Instead Data access objects (DAO) should be used by relying on Room, instead of using the SQLIte API's directly.

Using ? as a replaceable parameter in selection clauses and a separate array of selection arguments binds the user input directly to the query rather than interpreting it as part of a SQL statement.

Kotlin Java

```kotlin
// Constructs a selection clause with a replaceable parameter.
val selectionClause = "var = ?"

// Sets up an array of arguments.
val selectionArgs: Array<String> = arrayOf("")

// Adds values to the selection arguments array.
selectionArgs[0] = userInput
```

https://developer.android.com/privacy-and-security/risks/sql-injection

© 2025 by Bai7 GmbH | All rights reserved

Android - Static Analysis of Android App – Hardcoded Encryption Keys

Line 58 contains a hardcoded key.

```
57      public final String encryption() {
58          byte[] keyBytes = {108, 97, 107, 100, 115, 108, 106, 107, 97, 108, 107, 106, 108, 107, 108, 115};
59          Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding");
60          SecretKeySpec secretKey = new SecretKeySpec(keyBytes, "AES");
61          cipher.init(1, secretKey);
62          byte[] bytes = "my secret here".getBytes(Charsets.UTF_8);
63          Intrinsics.checkNotNullExpressionValue(bytes, "this as java.lang.String).getBytes(charset)");
64          SecretKeySpec badSecretKeySpec = new SecretKeySpec(bytes, "AES");
65          return "SUCCESS!!\n\nThe keys were generated and used successfully with the following details:\n\nHardcode
66      }
```

To mitigate this issue, consider using the KeyChain API when you want system-wide credentials, or the Android Keystore provider to let an individual app store its own credentials that only the app itself can access.

https://developer.android.com/privacy-and-security/risks/hardcoded-cryptographic-secrets

© 2025 by Bai7 GmbH | All rights reserved

Android - Static Analysis of Android App – WebViews

Best Practices for WebViews:

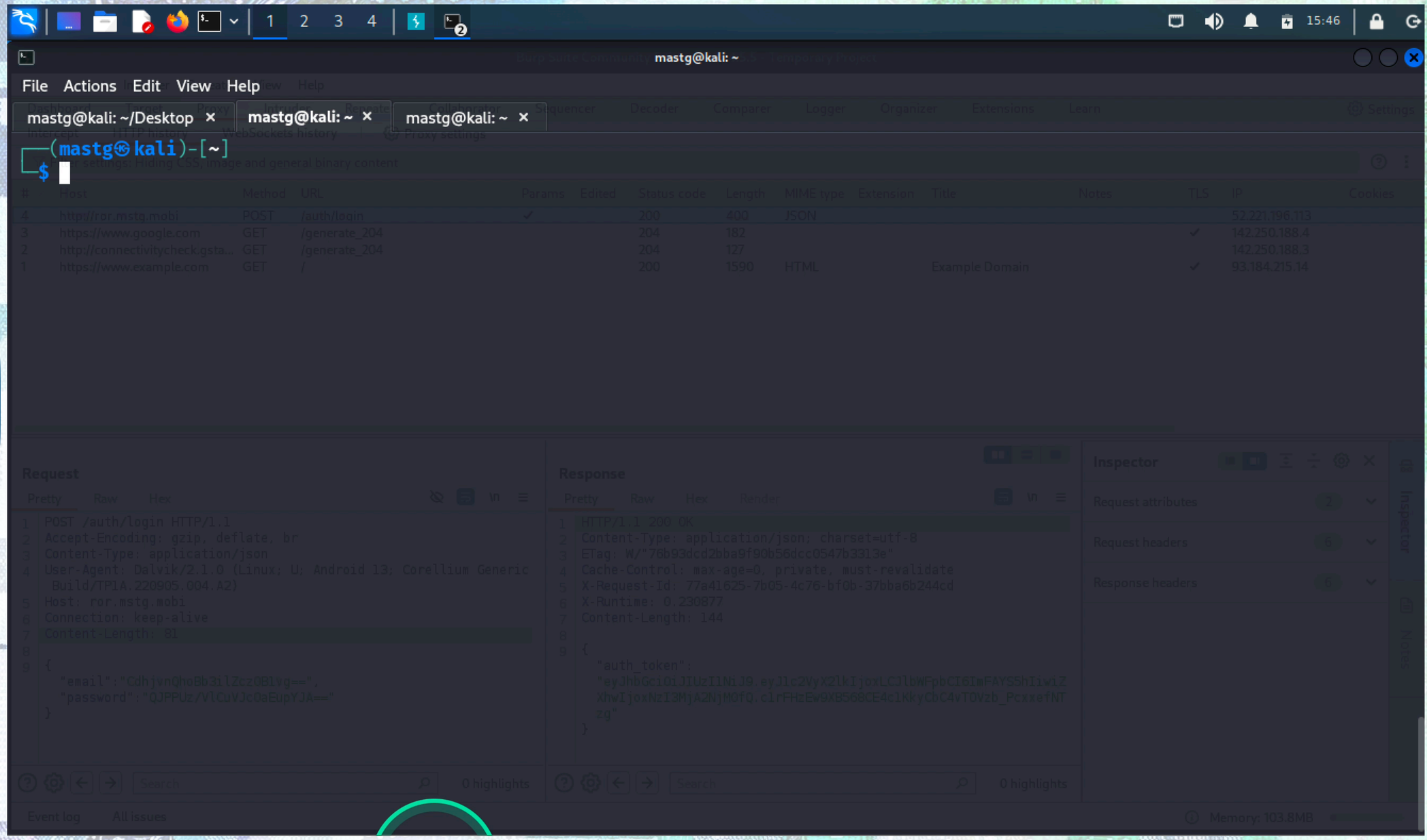
- It is recommended exposing ***addJavaScriptInterface()*** only to JavaScript that is contained within your application APK.
- If your application doesn't directly use JavaScript within a WebView, don't call setJavaScriptEnabled.
- Don't load code in WebView from external storage or untrusted endpoints, because that code might be modified to include malicious behavior.

https://developer.android.com/privacy-and-security/security-tips#webview

© 2025 by Bai7 GmbH | All rights reserved

Android - Static Analysis - SARIF

Local scan by using semgrep, SARIF, VS Code and the Sarif plugin.



© 2025 by Bai7 GmbH | All rights reserved

Android - Static Analysis of Android App – Summary for Pentesters

Decompile the app with jadx and scan the decompiled code with:

- Semgrep and the Android rules from Minded Security, see here for details, or
- mobsfscan

Be aware of common risks in Android that may not be covered by SAST tools, but that you should account for in a penetration test.

Mitigate security risks in your app

On this page

MASVS-STORAGE: Storage
MASVS-CRYPTO: Cryptography
MASVS-NETWORK: Network Communication
MASVS-PLATFORM: Platform Interaction
MASVS-CODE: Code Quality

By making your app more secure, you help preserve user trust and device integrity.

This page presents a set of common security issues that Android app developers face. You can use this content in the following ways:

- Learn more about how to proactively secure your apps.
- Understand how to react in the event that one of these issues is discovered in your app.

The following list contains links to dedicated pages for each individual issue, sorted into categories based on OWASP MASVS controls. Each page includes a summary, impact statement, and tips for mitigating the risk to your app.

© 2025 by Bai7 GmbH | All rights reserved

Android - Static Analysis of Android App – Summary for developers

Tips you can give to a developer, as there are several tools available that can execute SAST scans, but not all are supporting Android Apps. Selection criteria for a SAST tool:

- How does your mobile app landscape look like and which programming languages are you using?
- Which dependencies do you rely on? The SAST tool should also cover scanning of your dependencies (SCA).
- Which tests in the context of Android are being executed (many times the test cases are quite limited)?
- How are false positives managed and can you suppress them?
- Customization, can checks be added, modified or disabled? The tool need to be tweaked to your environment and needs!
- Open-Source or commercial tool (aka do I have budget)?
- If you only cover a small amount of programming languages SAST is a viable option!

First step for SAST: Start small with a subset of the available vulnerability checks and enable more checks after others have been remediated and are under control!

© 2025 by Bai7 GmbH | All rights reserved

Android - Static Analysis of Android App – Summary for developers

If you have a zoo of mobile apps (Flutter, React, Kotlin, ObjC/Swift, $foo)?

- You might have gaps from day 1, as with a SAST Solution you usually cannot cover all mobile dev frameworks and languages. You might want to look for a dynamic testing solution such as NowSecure that covers dynamic and static scanning of the mobile app binary.

Be aware of common risks in Android that may/may not be covered by SAST tools.

Mitigate security risks in your app

On this page

MASVS-STORAGE: Storage
MASVS-CRYPTO: Cryptography
MASVS-NETWORK: Network Communication
MASVS-PLATFORM: Platform Interaction
MASVS-CODE: Code Quality

By making your app more secure, you help preserve user trust and device integrity.

This page presents a set of common security issues that Android app developers face. You can use this content in the following ways:

- Learn more about how to proactively secure your apps.
- Understand how to react in the event that one of these issues is discovered in your app.

The following list contains links to dedicated pages for each individual issue, sorted into categories based on OWASP MASVS controls. Each page includes a summary, impact statement, and tips for mitigating the risk to your app.

© 2025 by Bai7 GmbH | All rights reserved

Android Training

Analysing Local Storage

Android - Analysing Local Storage

Conventional wisdom suggests that as little sensitive data as possible should be stored on permanent local storage. In most practical scenarios, however, sensitive user data must be stored, e.g.:

- Session Information (Cookies etc.)
- Access Tokens (JWT)
- Encryption keys
- User data for offline usage… etc.

© 2025 by Bai7 GmbH | All rights reserved

Android - Analysing Local Storage

Storing Data is essential for mobile apps and data can be stored persistently in several ways. The following list of storage techniques are widely used on the Android platform:

- ***Preferences***: Store private, primitive data in key-value pairs.
- ***Databases***: Store structured data in a private database using the Room persistence library.
- **Files**: Store data in customised file types (e.g. PDF).

https://developer.android.com/training/data-storage

© 2025 by Bai7 GmbH | All rights reserved

Android - Analysing Local Storage

Android uses a file system that's similar to disk-based file systems on other platforms. The system provides several options for you to save your app data:

App Specific Storage

App Sandbox
- Internal Storage
  - Dir: /data/data/<package-name>
  - Access methods:
    - getFilesDir()
    - getCacheDir()

External Storage (SDCard)
- Scoped Storage (since Android 11)

Shared / External Storage (SDcard)

Access granted by user
(e.g. by using file/photo picker)

App Sandbox explicitly shared directories and files

Access requires permissions
(e.g. READ_EXTERNAL_STORAGE)

- External Storage
  - Dir: /sdcard
  - Access methods:
    - getExternalFilesDir()
    - getExternalCacheDir()
    - getExternalStorage*()

Scoped Storage (before Android 11)

MediaStore

https://developer.android.com/training/data-storage

© 2025 by Bai7 GmbH | All rights reserved

Android Security - Sandbox

Sandbox by default!

- Apps are running in a self-contained environment through Mandatory Access Control (MAC)
- SELinux is used on Android to enforce MAC through labels and access control policies

```
cloud_x86:/ # cd /data/data/
cloud_x86:/data/data # ls -lah
total 402K
drwxrwx--x 101 system    system    4.0K 2021-04-01 23:43 .
drwxrwx--x  38 system    system    4.0K 2020-01-28 00:18 ..
drwx------   4 system    system    4.0K 2021-04-01 23:03 android
drwx------   4 u0_a9     u0_a9     4.0K 2020-01-28 00:01 android.ext.services
drwx------   4 u0_a45    u0_a45    4.0K 2020-01-28 00:01 android.ext.shared
drwxr-x--x   7 u0_a31    u0_a31    4.0K 2020-02-09 12:19 com.amaze.filemanager
```

All internal data is in the/data/data/<package-name> directory.

© 2025 by Bai7 GmbH | All rights reserved

Android - Analysing Local Storage

How can data be exposed or leaked?

© 2025 by Bai7 GmbH | All rights reserved

Android - Analysing Local Storage

How can data be exposed or leaked?

Data Breach and Unauthorized Access

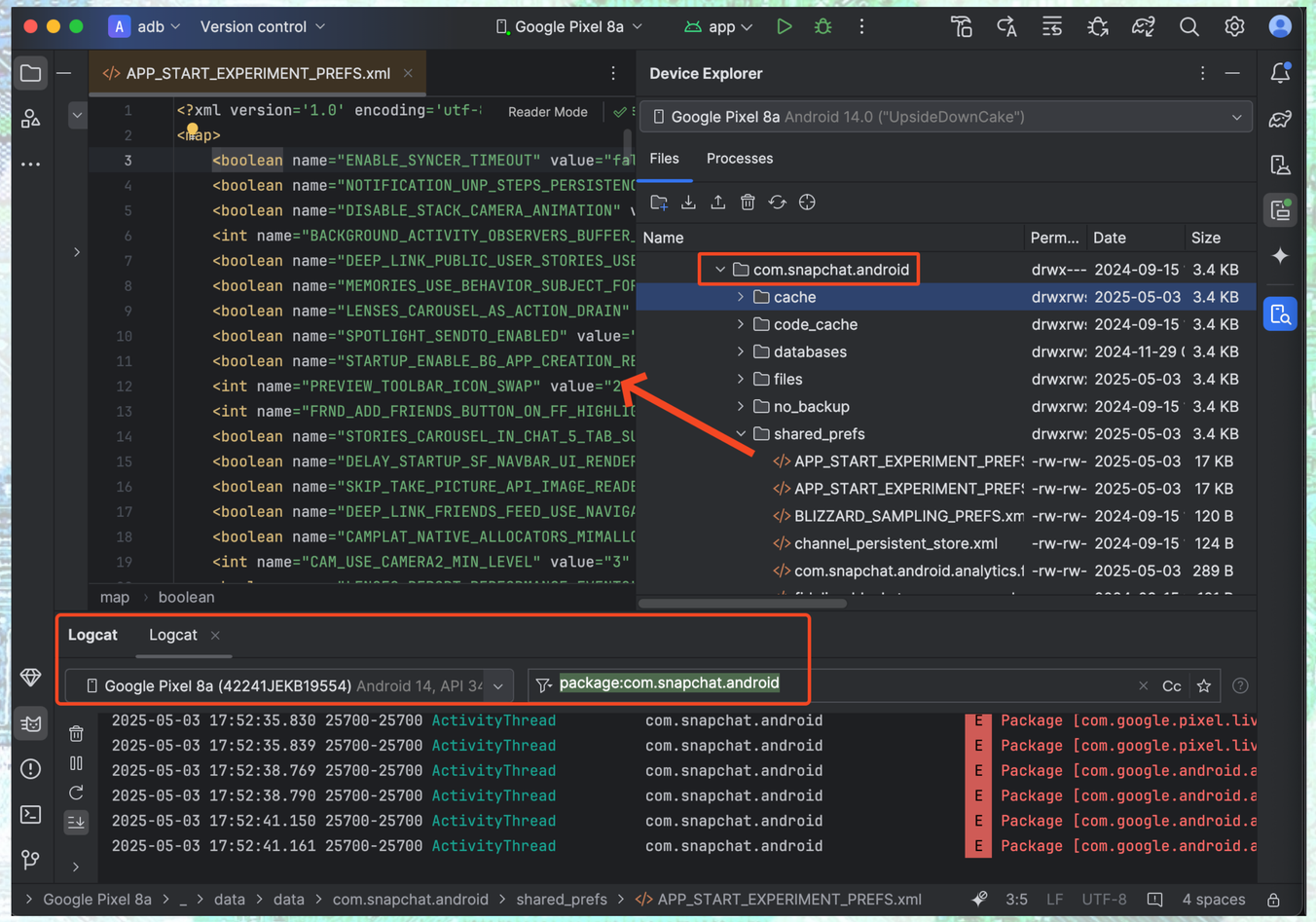
- Malicious Apps
  - Other apps with elevated permissions or exploiting vulnerabilities may be able to access unencrypted data stored in the sandbox.
- Physical Access
  - If someone gains physical access to the device, they could potentially access the app's data if it's not encrypted, even if the device is locked.
- Backup
  - Users might back up their apps and data, and if these backups are not encrypted, they can be extracted and analyzed on a different device or computer. (Caveat: newer Android versions encrypt the backup end2end)

© 2025 by Bai7 GmbH | All rights reserved

Android - Analysing Local Storage - Android Studio

With Android Studio you can do several things for analyzing Local Storage:

- Use your rooted Android Device (or AVD) in Android Studio through “Running Devices”. This allows you to mirror the device and you can control the device with your mouse on your laptop.
- Use the “Device Explorer” to navigate through the file system of your device.
- Additionally you can use ADB logcat to analyze the logs created when doing dynamic analysis.



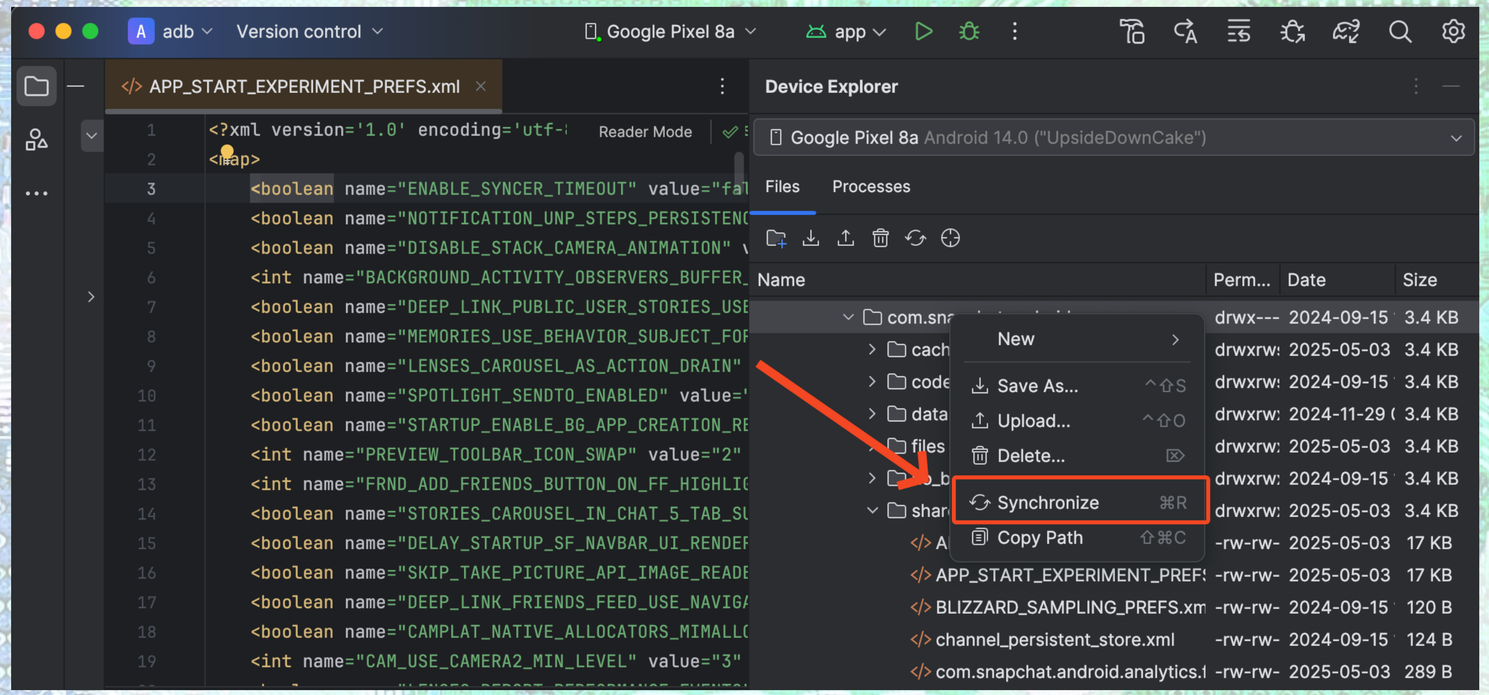
https://developer.android.com/studio/run/device
https://developer.android.com/studio/debug/device-file-explorer

© 2025 by Bai7 GmbH | All rights reserved

Android - Analysing Local Storage - Android Studio

Caveat:

There is no built-in auto-refresh in the “Device Explorer”. You always need to manually make a right click and select “Synchronize” or use the shortcut, before you analyze the files to have the latest version.



© 2025 by Bai7 GmbH | All rights reserved

Android - Analysing Local Storage - Android Studio

adb Version control Google Pixel 8a app

Device Explorer

Google Pixel 8a Android 14.0 ("UpsideDownCake")

Files Processes

Name	Per...	Date	Size
/	drwxr	2009-01-	4 KB

Search Everywhere Double ⇧

Go to File ⇧⌘O

Recent Files ⌘E

Navigation Bar ⌘↑

Drop files here to open them

Logcat Logcat

Google Pixel 8a (42241JEKB19554) Android 14, API 34

package:com.snapchat.android

```
2025-05-03 17:43:48.337 16849-16849 ActivityThread          com.snapchat.android  E Package [com.google.android.y
2025-05-03 17:43:48.357 16849-16849 ActivityThread          com.snapchat.android  E Package [com.google.android.y
2025-05-03 17:43:50.413 16849-16849 ActivityThread          com.snapchat.android  E Package [com.google.android.c
2025-05-03 17:43:50.418 16849-16849 ActivityThread          com.snapchat.android  E Package [com.google.android.c
2025-05-03 17:43:50.621 16849-16849 VRI[LoginS...pActivity] com.snapchat.android  D visibilityChanged oldVisibili
2025-05-03 17:43:50.621 16849-16849 VRI[LoginS...pActivity] com.snapchat.android  D visibilityChanged oldVisibili
```

adb > app > src > main > java > at > bai7 > adb > MainActivity.kt > storeCredentials

yout:

ntLayout in your activity_main.xml file.
he ConstraintLayout programmatically.
the button to position it correctly.
plex solution, but it is more flexible. **Key**

s a Button programmatically.
': Sets the button's text.
.stener { ... }: Sets the code to execute
ed.
con): Adds the button to the layout.
LinearLayout or ConstraintLayout to

have the correct imports. **Security Reminder**
Preferences is *not* secure for real user
otedSharedPreferences for that.
purposes: This example is only for training
ot be used in a real application. Let me know if
stions!

/Users/cs/adina/teaching/it472/spring2021/
ctivity-data-sharing/
r.com/KotlinBook/KotlinNotesForProfessionals.

chan/n/na0624b52e420

© 2025 by Bai7 GmbH | All rights reserved

Android - Analysing Local Storage

Should all sensitive data always be encrypted?

© 2025 by Bai7 GmbH | All rights reserved

Android - Analysing Local Storage

Should all sensitive data always be encrypted?

Tests

Android

MASVS-STORAGE

MASTG-TEST-0001: Testing Local Storage for Sensitive Data

MASTG-TEST-0003: Testing Logs for Sensitive Data

- This includes `SharedPreferences`, databases, Internal Storage, External Storage, etc.

NOTE: For MASVS L1 compliance, it is sufficient to store data unencrypted in the application's internal storage directory (sandbox). For L2 compliance, additional encryption is required using cryptographic keys securely managed in the Android KeyStore. This includes using envelope encryption (DEK+KEK) or equivalent methods, or using the Android Security Library's `EncryptedFile` ↗/ `EncryptedSharedPreferences` ↗.

© 2025 by Bai7 GmbH | All rights reserved

Android - Analysing Local Storage

What is “L1/L2 compliance” aka Testing Profiles?

MAS-L1 - Essential Security

MAS-L1 provides a baseline for the most fundamental security requirements and best practices that every mobile app should meet to protect against common threats.

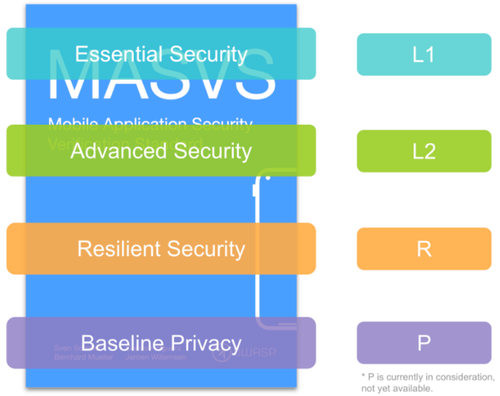
MAS-L2 - Advanced Security

MAS-L2 extends MAS-L1 introducing additional security measures and best practices for mobile apps to address advanced threats requiring more rigorous threat modeling and testing strategies.

MAS Testing Profiles

As part of the MASVS refactoring, we've replaced the three traditional verification levels (L1, L2, and R) with security testing profiles in the MASTG. These new profiles are designed to enhance our ability to capture various security nuances associated with mobile apps, allowing us to evaluate different situations for the same MASVS control. For instance, in MASVS-STORAGE-1, it's acceptable to store data unencrypted in app internal storage for MAS-L1, but MAS-L2 requires data encryption.

The new MAS Testing Profiles include revamped versions of the traditional levels and one new addition:



https://mas.owasp.org/news/2023/07/28/mas-testing-profiles-and-mastg-atomic-tests/

© 2025 by Bai7 GmbH | All rights reserved

Android - Analysing Local Storage

The Android KeyStore supports a secure storage of cryptographic keys. As of Android 4.3, it provides public APIs for storing and using app-private keys.

An app can create a new private/public key pair for encrypting application secrets, and it can decrypt the secrets with the private key.

You can protect keys stored in the Android KeyStore with user authentication by using the user's lock screen credentials (pattern, PIN, password, or fingerprint).

Most modern devices offer a hardware-backed KeyStore implementation. Keys are generated and used in a Trusted Execution Environment (TEE) and the operating system can't access them directly. This means that the encryption keys used in your app can't be easily retrieved, even from a rooted device as they are encrypted with a key from the TEE.

```
generic_x86_64:/data/misc/keystore/user_0 # ls -lah
total 10K
drwx------ 2 keystore keystore 4.0K 2020-12-03 13:40 .
drwx------ 3 keystore keystore 4.0K 2020-12-03 11:35 ..
-rw------- 1 keystore keystore  113 2020-12-03 11:35 .10069_chr_USRSKEY__androidx_security_master
_key_
-rw------- 1 keystore keystore  213 2020-12-01 17:17 .10070_chr_USRPKEY_Dummy
-rw------- 1 keystore keystore  234 2020-12-03 11:35 10069_USRSKEY__androidx_security_master_key_
generic_x86_64:/data/misc/keystore/user_0 # cat 10069_USRSKEY__androidx_security_master_key_
►► ►►►OB6~►G►►►q9►►►
                    ►►    L0 ►   ►`►►►&v►►►8►►0I1  ►►`►`►►►&v►►►8►►0I►~►lg►generic_x86_64:/data/misc/ke
```

© 2025 by Bai7 GmbH | All rights reserved

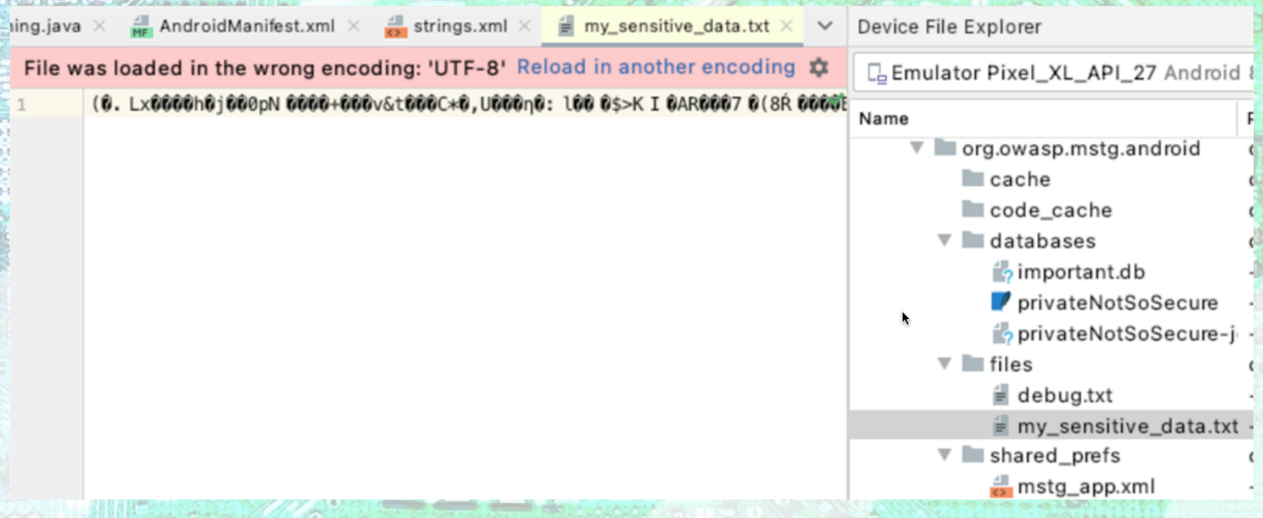
Android - Analysing Local Storage

The Security library provides an implementation of the security best practices related to reading and writing data at rest, as well as key creation and verification.

Security library offers 2 classes:

- EncryptedFile
- EncryptedSharedPreferences

```java
MasterKey masterKey = new MasterKey.Builder(context)
    .setKeyScheme(MasterKey.KeyScheme.AES256_GCM)
    .build();

File file = new File(context.getFilesDir(), "my_sensitive_data.txt");
EncryptedFile encryptedFile = EncryptedFile.Builder(
    context,
    file,
    masterKey,
    EncryptedFile.FileEncryptionScheme.AES256_GCM_HKDF_4KB
).build();

// write to the encrypted file
FileOutputStream encryptedOutputStream = encryptedFile.openFileOutput();

// read the encrypted file
FileInputStream encryptedInputStream = encryptedFile.openFileInput();
```



https://developer.android.com/reference/androidx/security/crypto/package-summary

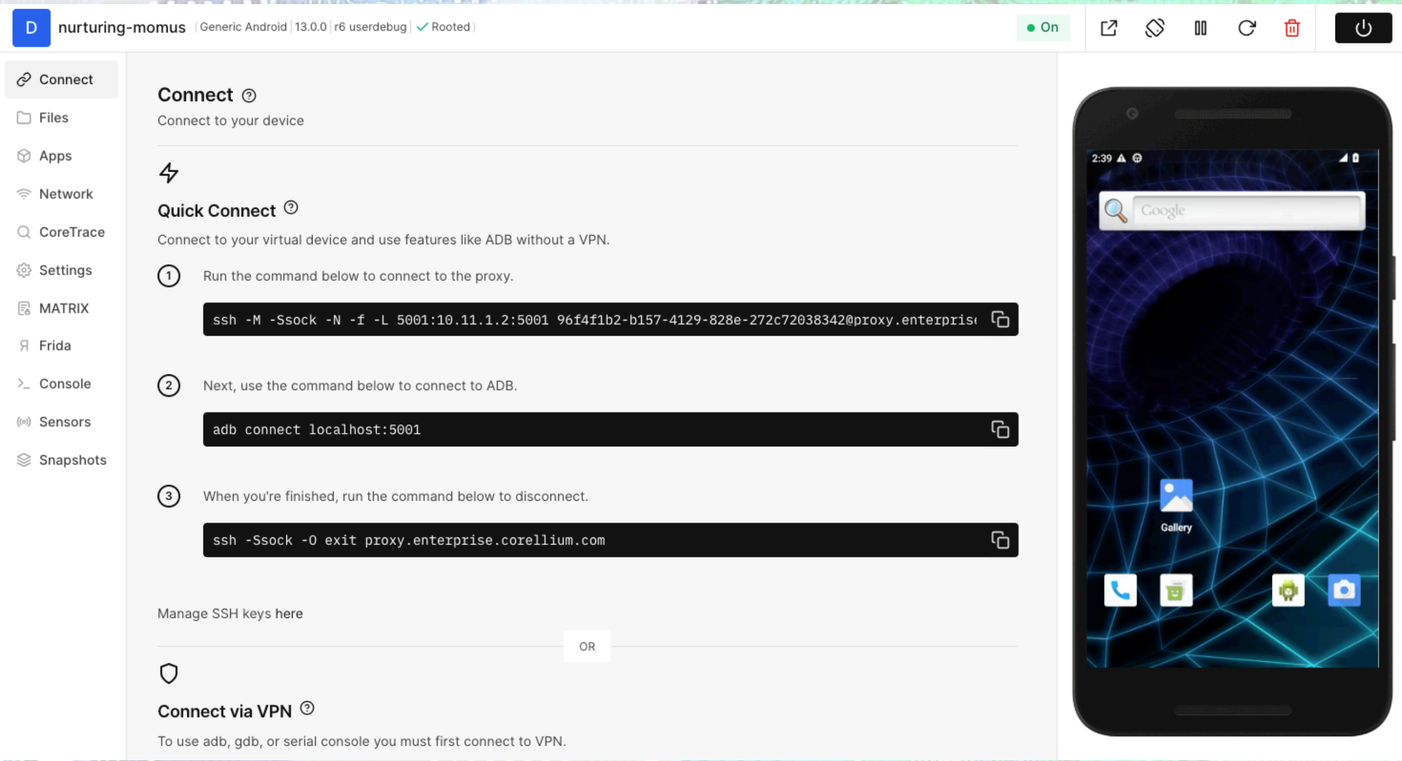
© 2025 by Bai7 GmbH | All rights reserved

Android Training

Corellium Setup

Corellium

- Corellium is SaaS and allows to emulate iOS in the cloud and also Android
- Please follow the next slides to start your Android device



© 2025 by Bai7 GmbH | All rights reserved

Corellium - Setup

Everybody is getting one rooted Android device, please stick to it!

Password: Cyber_2025

Username	Student (first name)
01@s7ven.info	Deji
02@s7ven.info	Kostis
03@s7ven.info	Dmytro
04@s7ven.info	Raz
05@s7ven.info	Jose
06@s7ven.info	Anil
07@s7ven.info	Paras
08@s7ven.info	Muhammadjon
09@s7ven.info	Anna
10@s7ven.info	Kamil

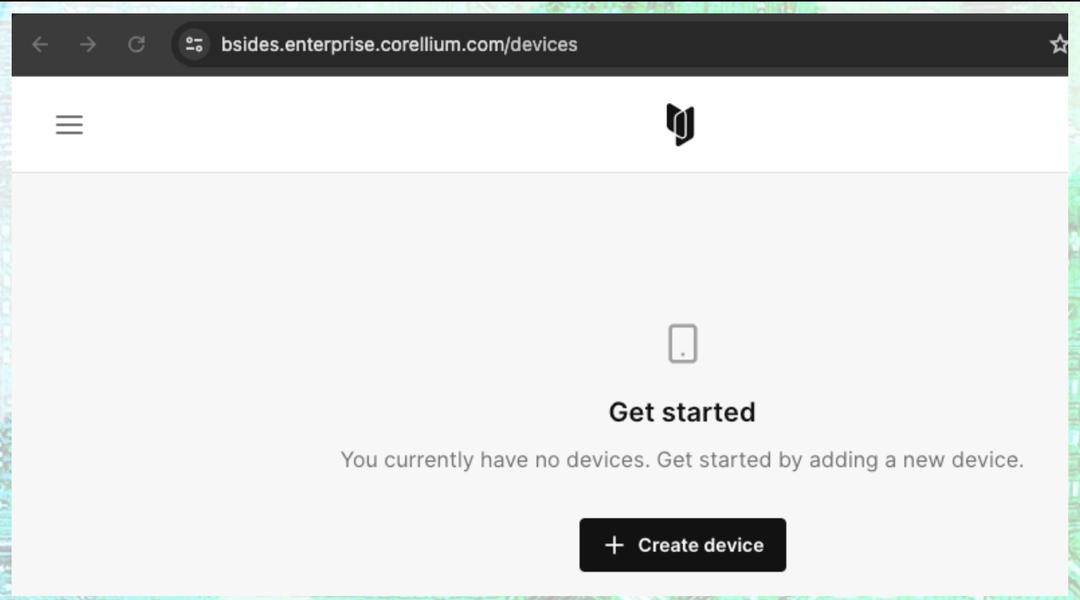
Username	Student (first name)
11@s7ven.info	Alfonso
12@s7ven.info	Mier
13@s7ven.info	Richard
14@s7ven.info	Jon
15@s7ven.info	Daniil
16@s7ven.info	Tom
17@s7ven.info	Kimmy
18@s7ven.info	Aristos
19@s7ven.info	Kenny

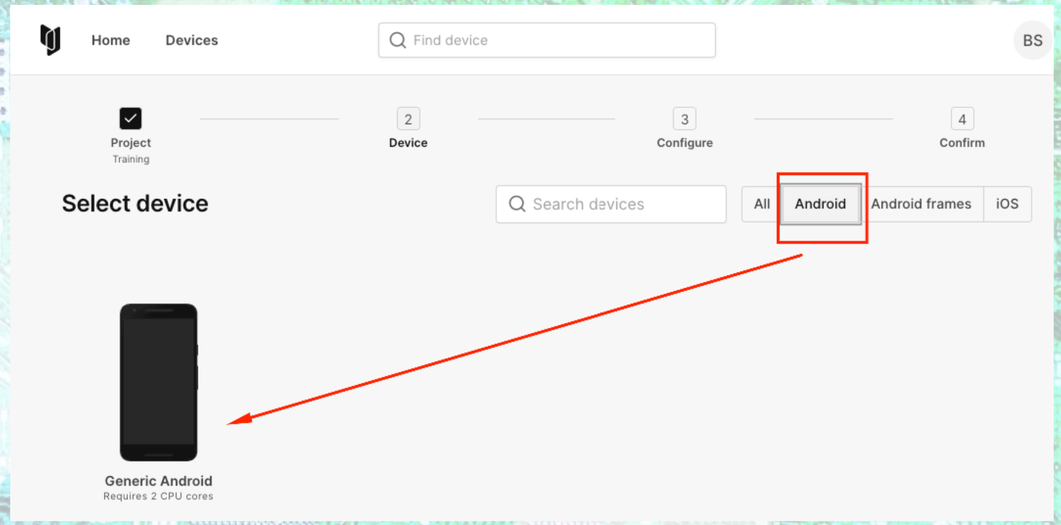
© 2025 by Bai7 GmbH | All rights reserved

Corellium - Setup

Disable your VPN first !

- Go to https://bsides.enterprise.corellium.com
- Login with your credentials
- Click on “Create Device”
- Select your project (there should only be one)
- Filter for Android and select “Generic Android”

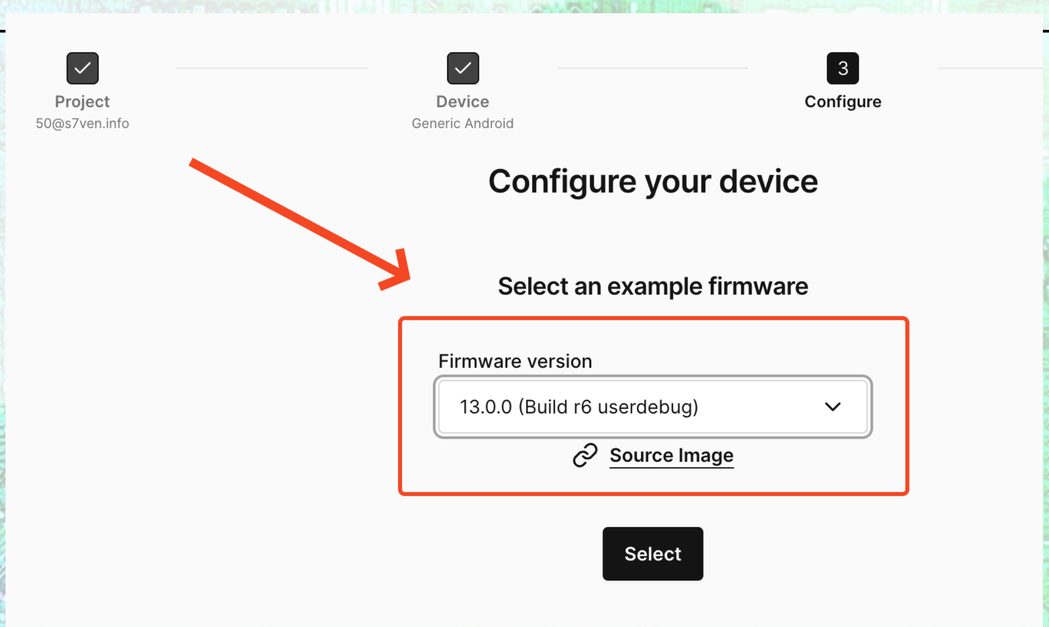


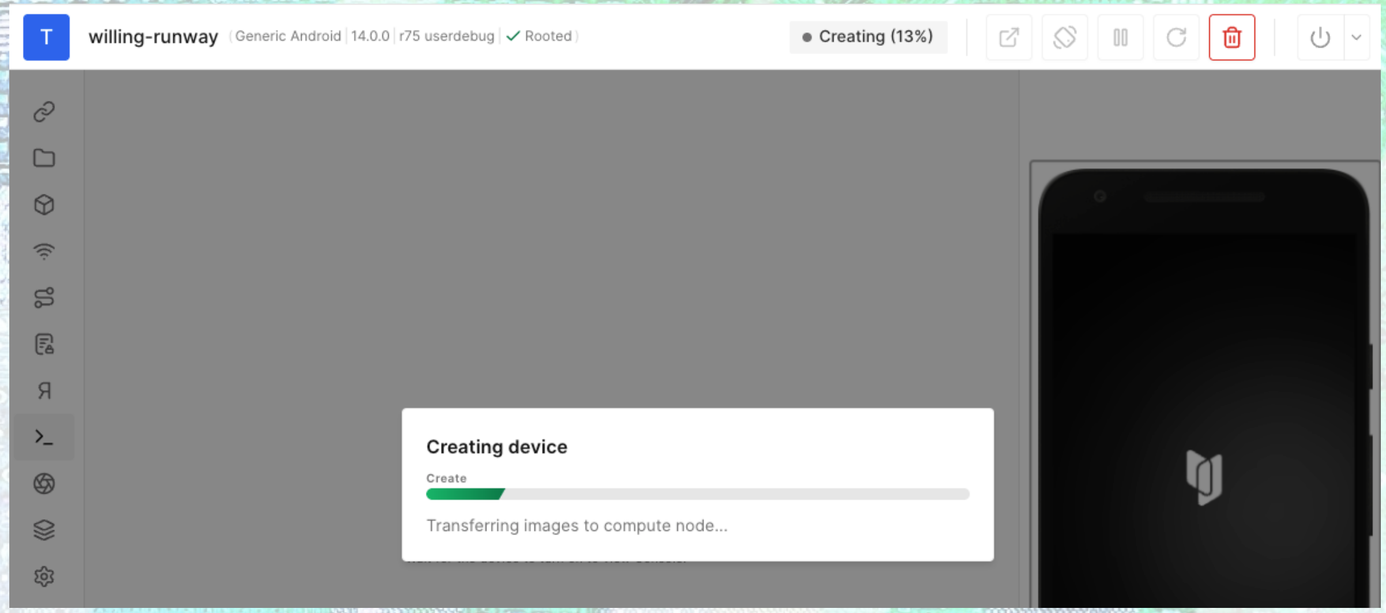


© 2025 by Bai7 GmbH | All rights reserved

Corellium - Setup

- Select Android 13.0.0 (Build r6 userdebug)
- Click on "Create device" with the default settings
- Wait for a few minutes till the device has booted





© 2025 by Bai7 GmbH | All rights reserved

Corellium Introduction

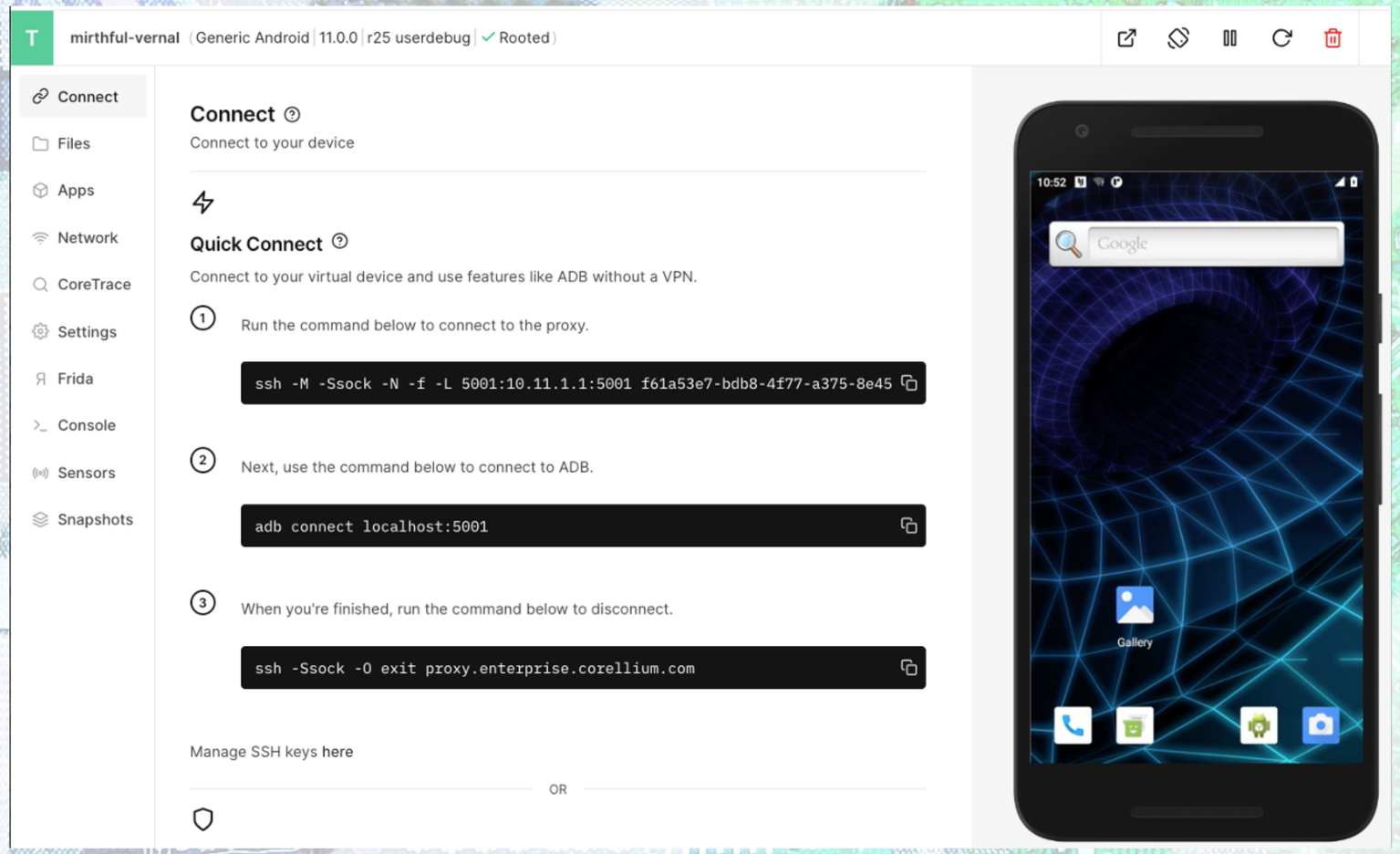
Few tips:

- You need to “swipe up” by clicking on the bottom of the screen with your mouse, hold it and go up to access the installed apps.
- The “Home” button and “Back” button is below the phone with several other buttons.

© 2025 by Bai7 GmbH | All rights reserved

Corellium Introduction

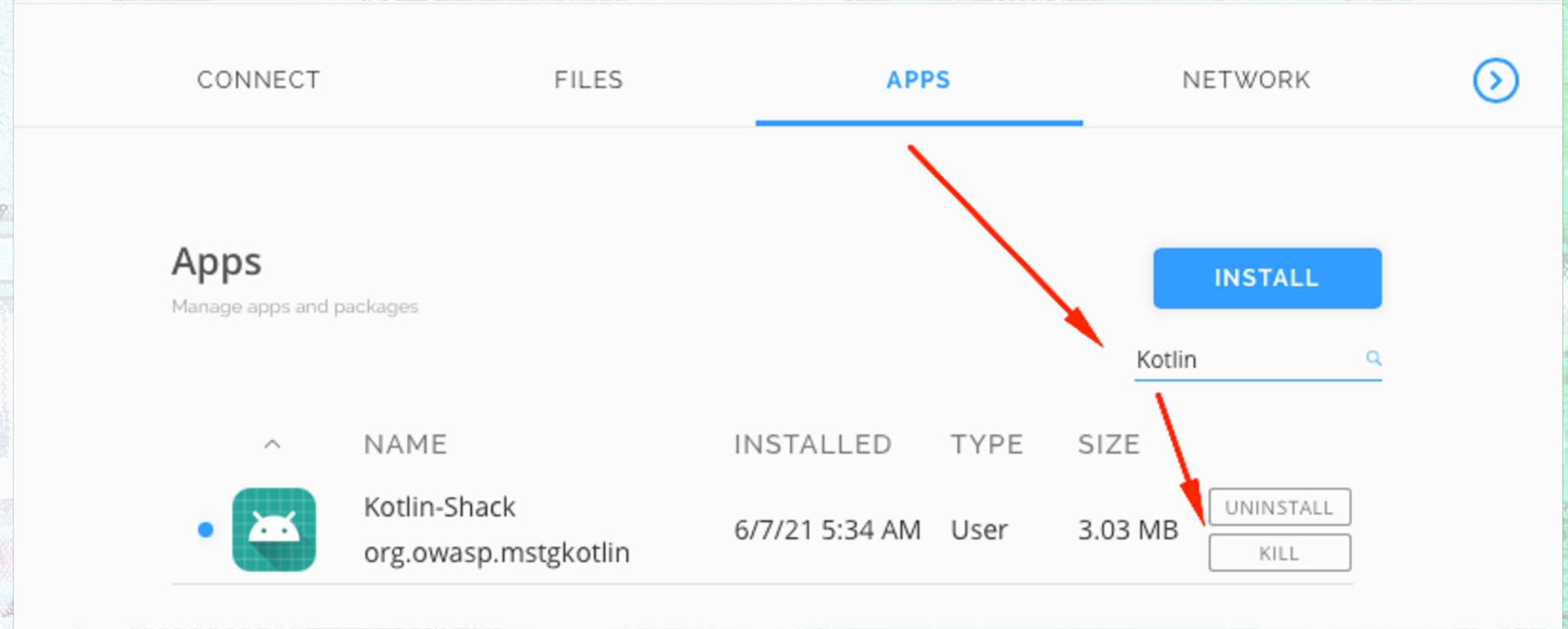
Get familiar with Corellium



© 2025 by Bai7 GmbH | All rights reserved

Corellium Introduction

How to close/kill an app in Corellium:

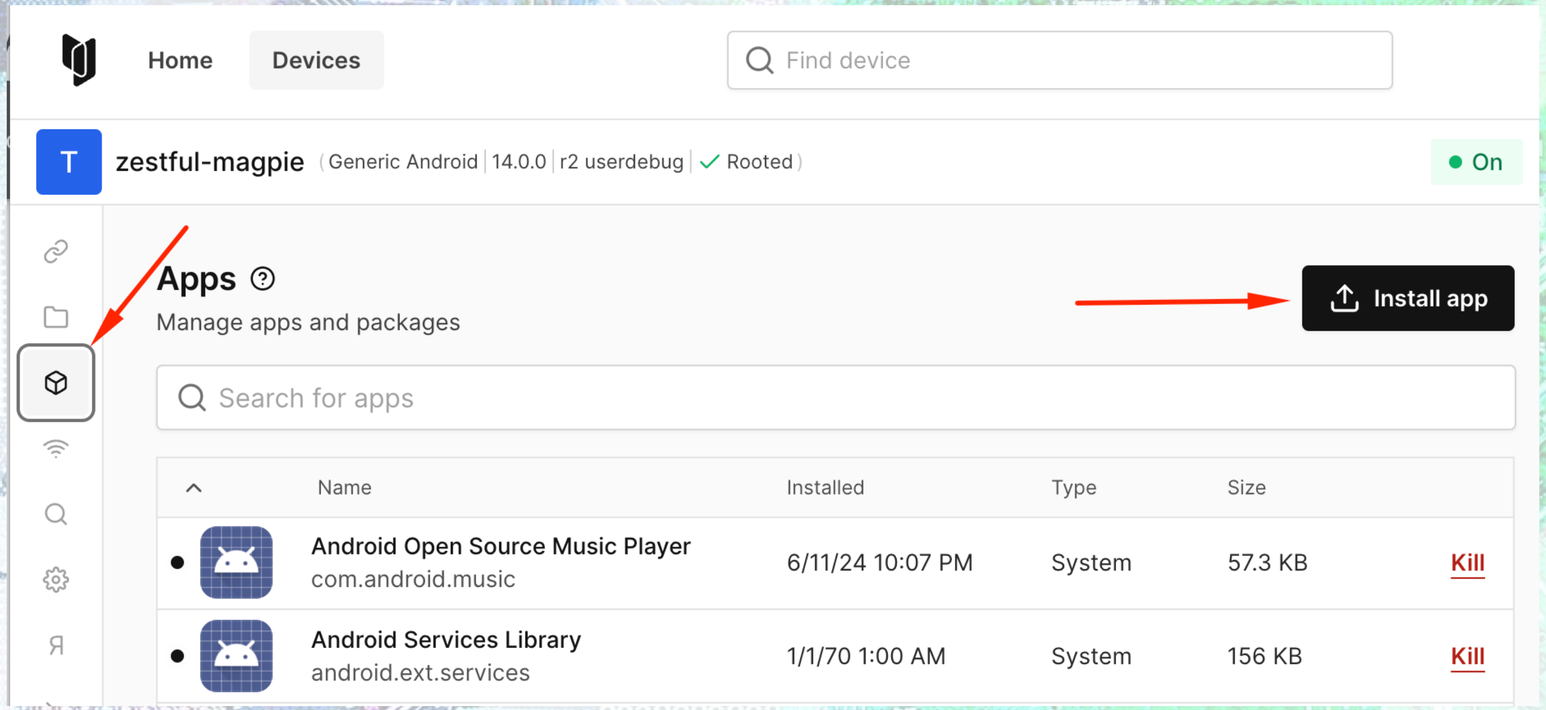


© 2025 by Bai7 GmbH | All rights reserved

Android Training

Android Reverse Engineering

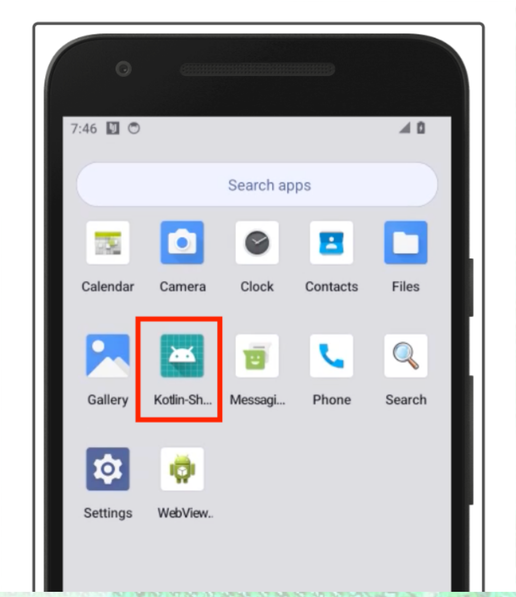
Install the app “Kotlin-Shack.apk” that is in the “apps” folder in the repo you cloned earlier today.

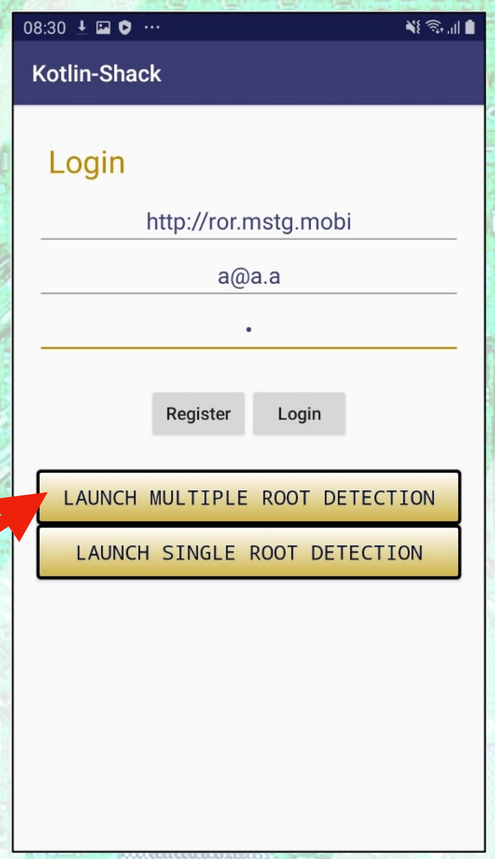


© 2025 by Bai7 GmbH | All rights reserved

Android Reverse Engineering

- Start the App
- Click on the first button “Launch Multiple Root Detection”
- The Kotlin Shack app has root detection…





© 2025 by Bai7 GmbH | All rights reserved

How do you think Root Detection works? What is the app checking?

© 2025 by Bai7 GmbH | All rights reserved

What is Root Detection?

The goal of root detection is to make running the app on a rooted device more difficult, which in turn blocks some of the tools and techniques reverse engineers like to use.

Common Root Detection Methods:

- File existence checks
- Checking running processes (not possible on newer Android versions)
- Checking installed app packages
- Checking for writable partitions and system directories
- Checking for custom Android builds

MASTG - Android Anti-Reversing Defenses - Root Detection and Common Root Detection Methods

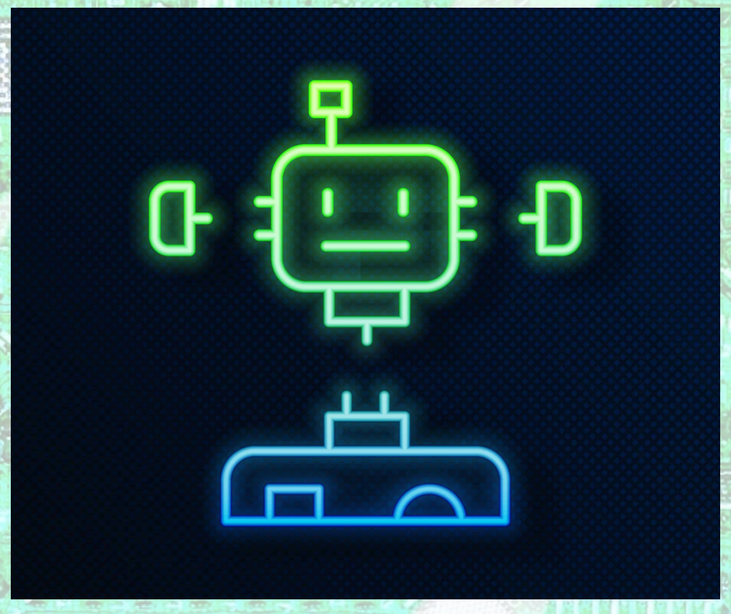
© 2025 by Bai7 GmbH | All rights reserved

Android Reverse Engineering

In the context of Android application penetration testing, reverse engineering skills can be used to bypass client-side security controls such as root detection, anti debugging etc.

Topics we will be covering:

- Decompiling to Smali and Patching the App
- Magisk (modifying the Android system and rooting it)
- Dynamic instrumentation (Frida)



© 2025 by Bai7 GmbH | All rights reserved

Android Reverse Engineering - Smali and Repacking

What is Smali?

Smali is an assembly language representing the bytecode executed by the Android Runtime (ART). It is used to decompile and recompile Android application packages (APKs), allowing for analysis and modification of the app's code.

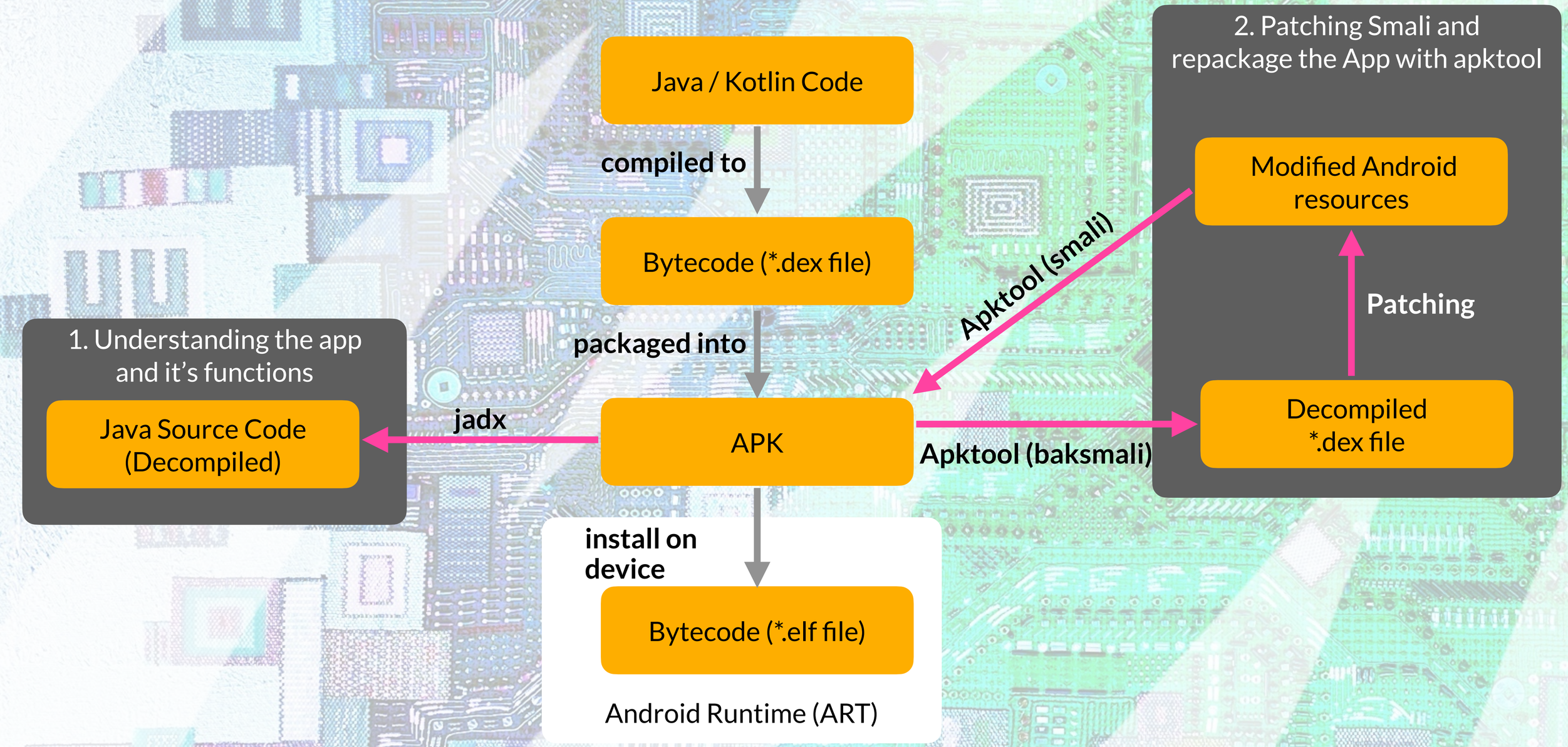
However, it's not easy to read or modify binary code, so there are tools that convert dex to assembly and vice versa like Apktool.

```
.method private final checkRootMethod1()Z
    .locals 6

    .line 139
    sget-object v0, Landroid/os/Build;->TAGS:Ljava/lang/String;

    .line 140
    .local v0, "buildTags":Ljava/lang/String;
    const/4 v1, 0x0

    if-eqz v0, :cond_0

    move-object v2, v0

    check-cast v2, Ljava/lang/CharSequence;

    const-string v3, "test-keys"

    check-cast v3, Ljava/lang/CharSequence;

    const/4 v4, 0x2

    const/4 v5, 0x0

    invoke-static {v2, v3, v1, v4, v5}, Lkotlin/text/StringsKt;
    ->contains$default(Ljava/lang/CharSequence;Ljava/lang/CharSequence;
    ZILjava/lang/Object;)Z

    move-result v2

    if-eqz v2, :cond_0

    const/4 v1, 0x1

    nop

    :cond_0
    return v1
.end method
```

MASTG-TECH-0016: Disassembling Code to Smali

© 2025 by Bai7 GmbH | All rights reserved

Android Reverse Engineering - Smali and Repacking



© 2025 by Bai7 GmbH | All rights reserved

Android Reverse Engineering - Smali and Repacking

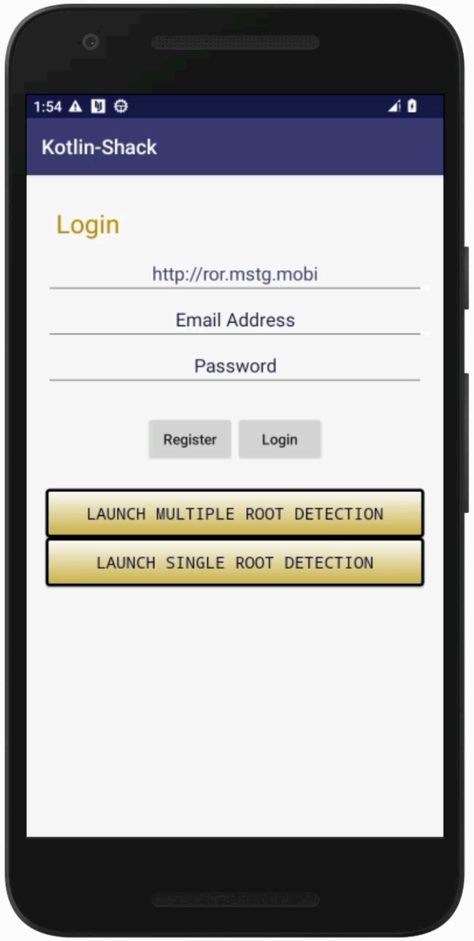
```
~/L/Mobile /com~apple~C/Conf/Tr/M/02/00_3/MASTG-Hands-on/Android/Apps                15:55:30
❯
```

bsides.enterprise.corellium.com/devices/6...

Clone of Clone of delighted-hero

Generic Android | 13.0.0 | r6 userdebug | Rooted

On



© 2025 by Bai7 GmbH | All rights reserved

Android Reverse Engineering

Attackers will try create new apps or modify popular apps through reverse engineering, add malicious payloads, and then upload the modified app to App Stores (“Repackaging Attacks”) or lure users into side-loading (installing without the Play Store) the modified apps.

Repackaging can also be used when testing Android Apps. To repackage an application, after decompiling you need to:

1. Decompile the app using Apktool
2. Modify the app in Smali
3. Repackage the app using Apktool
4. Resign the app using jarsigner

https://www.techradar.com/pro/security/malware-riddled-android-apps-spotted-on-google-play-store-heres-what-to-avoid

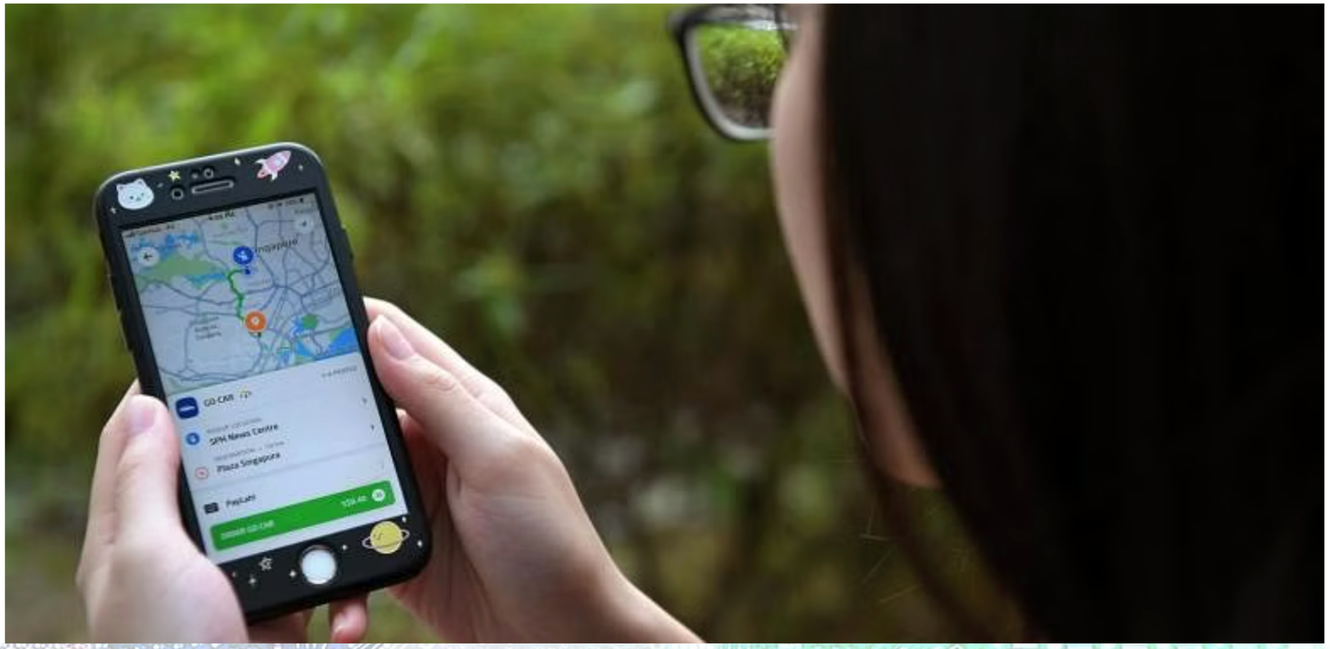
© 2025 by Bai7 GmbH | All rights reserved

Android Reverse Engineering - Smali and Repacking

SINGAPORE

LOG

120 Gojek drivers suspended for using modified app that helps them bypass verification



https://www.straitstimes.com/singapore/gojek-to-suspend-120-drivers-for-fake-app-use

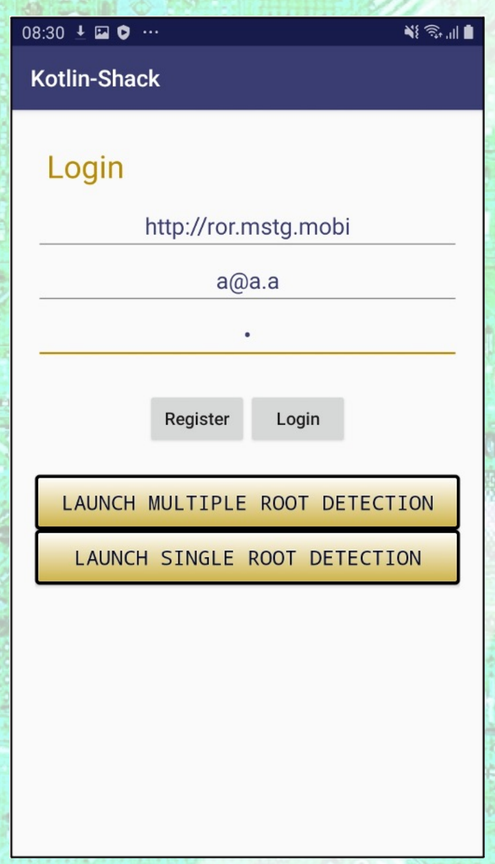
© 2025 by Bai7 GmbH | All rights reserved

Android Reverse Engineering - Frida

When patching the smali code, we noticed that there are several other root detection mechanisms and we didn't bypass "Multiple Root Detection".

We could now also patch this function in Smali, but this time we use:

FRIDA



© 2025 by Bai7 GmbH | All rights reserved

Android Reverse Engineering - Frida

- ***Dynamic Instrumentation Toolkit:*** Frida is a dynamic instrumentation toolkit designed to inject JavaScript into processes for runtime manipulation and analysis.
- ***Versatile Debugging and Testing:*** It allows developers, security researchers, and reverse engineers to perform tasks such as API tracing, function hooking, and code injection, making it ideal for debugging, testing, and vulnerability analysis.
- ***Scripting with JavaScript:*** Users can write JavaScript scripts to interact with the target application in real-time, enabling powerful and flexible manipulation of the app's behaviour and data.
- ***Cross-Platform Support:*** Frida supports a wide range of platforms and architectures, including Android, iOS, Windows, macOS, and Linux, making it a versatile tool for working with different types of applications and systems.
- ***Open Source and Extensible:*** As an open-source project, Frida has an active community contributing to its development and extending its capabilities through various plugins and scripts.

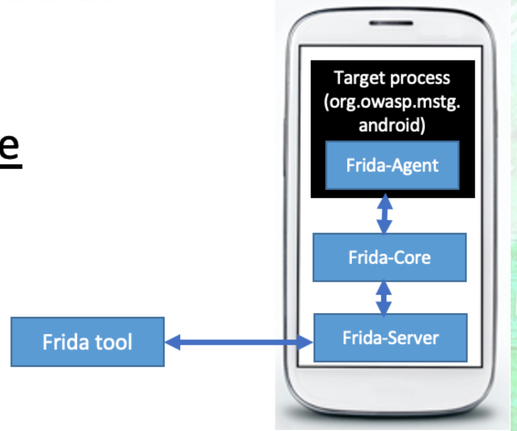
https://frida.re/

© 2025 by Bai7 GmbH | All rights reserved

Android Reverse Engineering - Frida

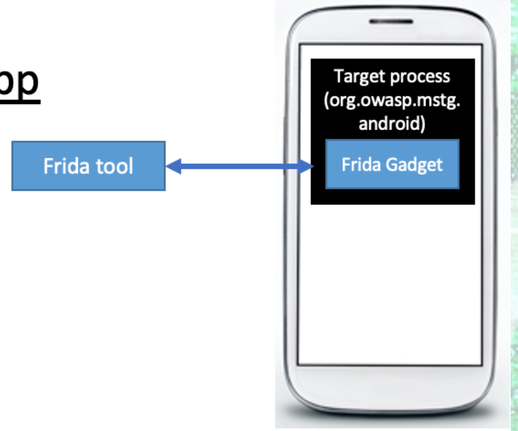
Injected into a process by running the Frida server on the device

- Possible on rooted / jailbroken devices
- Frida handles the injection



Embedded as shared library (frida-gadget.so) into the mobile app

- Non-rooted / non-jailbroken devices
- Repackaging and resigning required



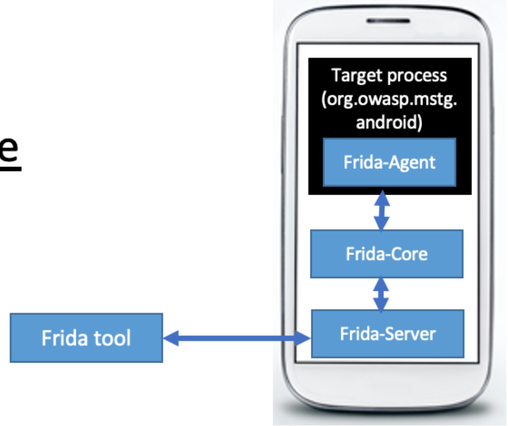
Frida injects a complete JavaScript runtime (QuickJS) into the process, along with a powerful API that you can sent commands to or execute scripts.

© 2025 by Bai7 GmbH | All rights reserved

Android Reverse Engineering - Frida

Injected into a process by running the Frida server on the device

- Possible on rooted / jailbroken devices
- Frida handles the injection



Embedded as shared library (frida-gadget.so) into the mobile app

- Non-rooted / non-jailbroken devices
- Repackaging and resigning required

We focus on the Frida Server scenario today!

© 2025 by Bai7 GmbH | All rights reserved

Android Reverse Engineering - Frida - Term Definition

Hook

- ***Definition***: Hooking is the act of intercepting function calls or method invocations in a target application. This allows you to alter the flow of execution, modify arguments, observe behaviour, or change return values.
- ***Usage***: In line 11 the script is hooking into the method “targetMethod”

Instrument

- ***Definition:*** Instrumentation in Frida refers to the process of adding custom code to an application to gather information about its execution. This allows you to monitor, trace, and analyze the behavior of the application in real-time.
- ***Usage:*** From line 13 to 29 we are adding additional custom code.

```
1   Java.perform(function() {
2       // Find the class you want to hook
3       var targetClass = "com.example.targetapp.TargetClass";
4       // Specify the method you want to hook
5       var targetMethod = "targetMethod";
6
7       // Get a reference to the target class
8       var TargetClass = Java.use(targetClass);
9
10      // Hook the target method with specified arguments (e.g., int and String)
11      TargetClass[targetMethod].overload('int', 'java.lang.String').implementation =
        function(arg1, arg2) {
12          // Instrumentation: Log and inspect when the method is called
13          console.log("targetMethod called");
14
15          // Instrumentation: Log and inspect the arguments passed to the method
16          console.log("Argument 1: " + arg1);
17          console.log("Argument 2: " + arg2);
18
19          // Call the original method with the original arguments
20          var result = this[targetMethod](arg1, arg2);
21
22          // Instrumentation: Log and inspect the return value of the method
23          console.log("Return value: " + result);
24
25          // Instrumentation / Manipulate the return value
26          result = 42;
27
28          // Return the manipulated result
29          return result;
30      };
31  });
```

© 2025 by Bai7 GmbH | All rights reserved

Android Reverse Engineering - Frida - Term Definition

Inspect

- **Definition**: Inspecting in Frida involves examining the internal state of an application, including its memory, function calls, and variable values, without necessarily altering its behaviour.
- **Usage:** In line 16 and 17 we inspect the arguments.

Manipulate

- ***Definition:*** Manipulating in Frida means changing the behavior or data of an application during its execution. This can involve altering function parameters, return values, or modifying the application's state in memory.
- ***Usage:*** In line 26 we manipulate the variable “result” and then return the manipulate value back whenever the method “targetMethod” is called.

```
1  Java.perform(function() {
2      // Find the class you want to hook
3      var targetClass = "com.example.targetapp.TargetClass";
4      // Specify the method you want to hook
5      var targetMethod = "targetMethod";
6
7      // Get a reference to the target class
8      var TargetClass = Java.use(targetClass);
9
10     // Hook the target method with specified arguments (e.g., int and String)
11     TargetClass[targetMethod].overload('int', 'java.lang.String').implementation =
       function(arg1, arg2) {
12         // Instrumentation: Log and inspect when the method is called
13         console.log("targetMethod called");
14
15         // Instrumentation: Log and inspect the arguments passed to the method
16         console.log("Argument 1: " + arg1);
17         console.log("Argument 2: " + arg2);
18
19         // Call the original method with the original arguments
20         var result = this[targetMethod](arg1, arg2);
21
22         // Instrumentation: Log and inspect the return value of the method
23         console.log("Return value: " + result);
24
25         // Instrumentation / Manipulate the return value
26         result = 42;
27
28         // Return the manipulated result
29         return result;
30     };
31 });
```

© 2025 by Bai7 GmbH | All rights reserved

Android Reverse Engineering - Frida Basics

Spawn an App

- ***Definition***: Spawning involves starting a new instance of an application directly under Frida's control. This method is useful when you want to instrument the application from the very beginning of its execution.
- Use case for “early Instrumentation”: Ideal for scenarios where you need to hook or monitor code that runs at the startup of the application.

```
$ frida -U -f [package_name] --no-pause
```

- -U specifies the USB device.
- -f [package_name] indicates spawning a new process for the given package.
- --no-pause prevents the application from pausing immediately after being spawned.

Attach to an App

- ***Definition:*** This is useful when you want to inspect or modify an app that is currently running.
- ***Usage:***
  - ***Runtime Analysis:*** Ideal for analyzing applications at specific states or during specific activities.
  - ***Dynamic Interaction:*** Allows you to interact with the application without needing to restart it, preserving its current state and context.

```
$ frida -U -n [process_name]
```

- -U specifies the USB device.
- -n [process_name] indicates attaching to a process with the given name.

© 2025 by Bai7 GmbH | All rights reserved

Android Reverse Engineering - Frida Basics

Frida CLI

- ***Definition***: The Frida Command Line Interface (CLI) provides an interactive Read-Eval-Print Loop (REPL) environment where users can write and execute JavaScript commands in real-time against a target process.
- Allows for immediate execution and feedback, making it useful for quick tests, debugging, and learning how the target application behaves.

```
$ frida -U -n [process_name]
```

- -U specifies the USB device.
- -n [process_name] indicates attaching to a process with the given name.
- --no-pause prevents the application from pausing immediately after being spawned.

Frida Scripts

- ***Definition:*** Frida scripts can be written in JS and can be injected directly into the process (App).
- ***Usage:***
  - ***Automated Tasks:*** Suitable for automated and repetitive tasks that require complex logic and multiple steps.
  - ***Reusability:*** Scripts can be reused, shared, and version-controlled, making them useful for collaborative work and consistent execution.

```
$ frida -U -n [process_name] -l [script.js]
```

- -U specifies the USB device.
- -n [process_name] indicates attaching to a process with the given name.
- -l [script.js] loads the specified script.

© 2025 by Bai7 GmbH | All rights reserved

Android Reverse Engineering - Frida Basics

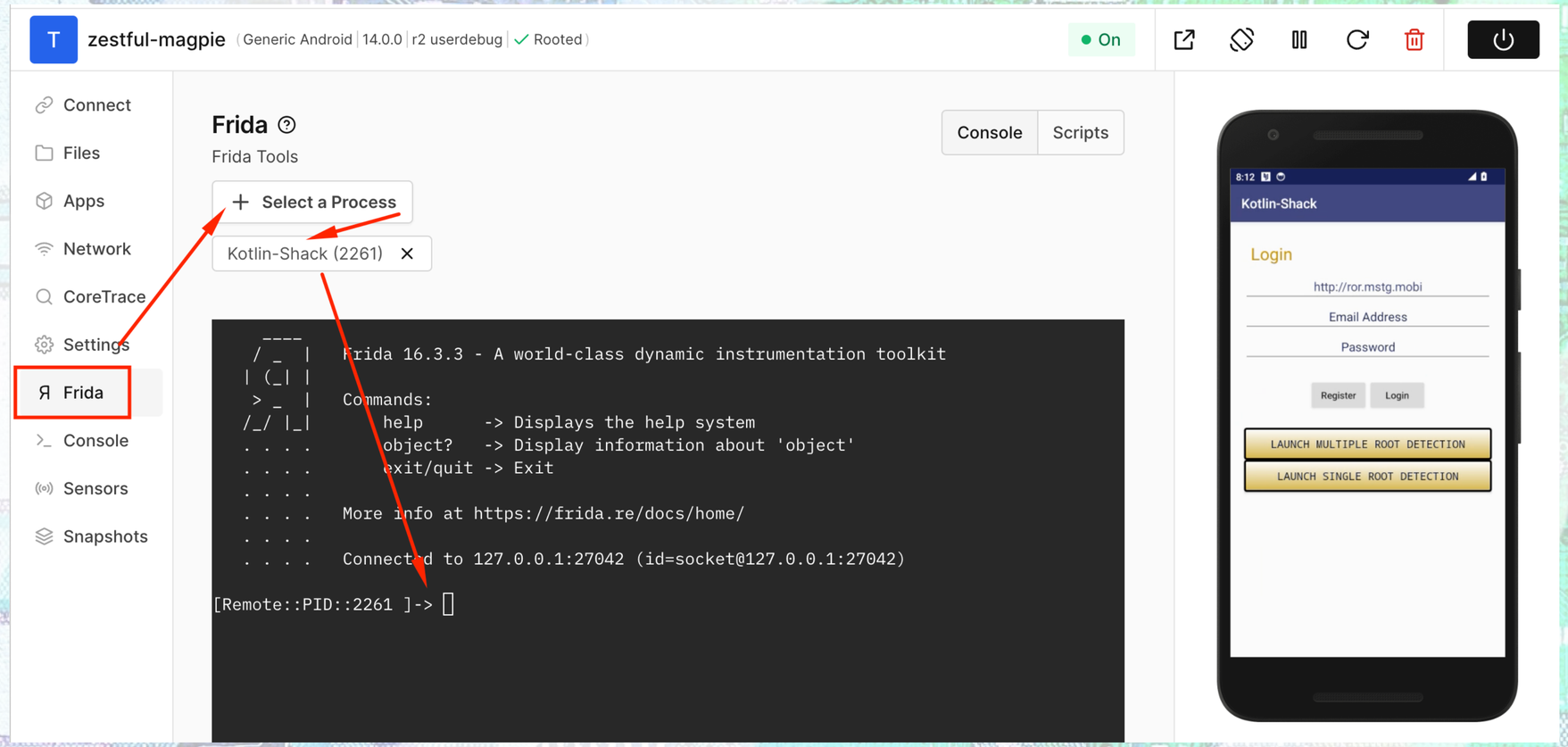
```
┌──(kali㉿kali)-[~/MSTG-Handson/Frida-Scripts]
└─$ frida -l bypass-multi-root.js -U org.owasp.mstgkotlin
     ____
    / _  |   Frida 14.2.14 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://frida.re/docs/home/
Attaching ...
[*] Script loaded
[Android Emulator 5554::org.owasp.mstgkotlin]-> Root detection script triggered
Root detection script triggered
Root detection script triggered
```

© 2025 by Bai7 GmbH | All rights reserved

Android Reverse Engineering - Frida Basics

Frida is already integrated into Corellium, therefore there is no need for you to use the command line tools.

We can directly hook into the running processes and use the GUI in Corellium and can get directly a Frida prompt.



© 2025 by Bai7 GmbH | All rights reserved

Open the file Lab-Android.pdf

Section:
Lab – Frida 101

Note: Some PDF readers do not copy properly and miss whitespaces etc.

© 2025 by Bai7 GmbH | All rights reserved

Android Reverse Engineering - Frida

Overview of 2 different methods we learned to bypass client-side reverse engineering controls, such as Root Detection:

	Frida	Smali Patching
Deployment	Adb + Connected PC	Adb + Connected PC
Invasiveness	Modify Target Process	Change Compiled App
Granularity	Method	Instruction
Updates	Instantaneous	Recompilation

© 2025 by Bai7 GmbH | All rights reserved

Android Reverse Engineering - Frida

Frida can help you to:

- Intercept network traffic, crypto calls, file access etc.
- Pentest and fuzz apps
- Overcome protection mechanisms
- Analyse unknown apps (Malware)
- Solve crackmes/CTFs
- Build your own analysis scripts, frameworks and tools

Official Frida documentation: https://frida.re/docs/home/

FRIDA

© 2025 by Bai7 GmbH | All rights reserved

Android Reverse Engineering - Frida Resources

Read code and go through existing Frida scripts (don't reinvent the wheel, leverage on what is out there 🤓):

- https://codeshare.frida.re
- https://github.com/dweinstein/awesome-frida
- https://github.com/iddoeldor/frida-snippets

These scripts are great to get an understanding of Frida and to use it as a starting point if you have a specific use case you want to implement.

Tools based on Frida:

- medusa
- objection

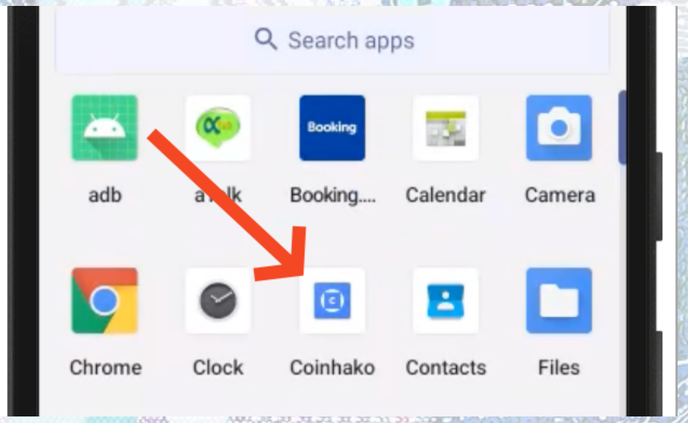
© 2025 by Bai7 GmbH | All rights reserved

Android Training

Bypass root detection in a real world mobile app

Attacking a real world mobile app - Coinhako

The app “Coinhako” is an Android app and is a crypto exchange platform.



This app is part of a bug bounty program from HackerOne. This means we are legally allowed to research it, as long as we follow the program rules!

https://hackerone.com/coinhako

Program Rules

Please provide detailed reports with reproducible steps. If the report is not detailed enough to reproduce the issue, the issue will not be eligible for a reward.

- Submit one vulnerability per report, unless you need to chain vulnerabilities to provide impact.
- When duplicates occur, we only award the first report that was received (provided that it can be fully reproduced).
- Multiple vulnerabilities caused by one underlying issue will be awarded one bounty.
- Social engineering (e.g. phishing, vishing, smishing) is prohibited.
- Make a good faith effort to avoid privacy violations, destruction of data, and interruption or degradation of our service. Only interact with accounts you own or with explicit permission of the account holder. Make a good faith effort to avoid privacy violations, destruction of data, interruption or degradation of our businesses, including Denial of Services attacks.
- Not exploit the vulnerability in any way, including through making it public or by obtaining a profit (other than a reward under this Program).
- Do not attempt non-technical attacks such as social engineering, phishing, or physical attacks against our employees, users, or infrastructure.
- Do not use scanners or automated tools to find vulnerabilities. They are noisy and we may ban your IP address.
- By submitting a bug, you agree to be bound by the rules.

© 2025 by Bai7 GmbH | All rights reserved

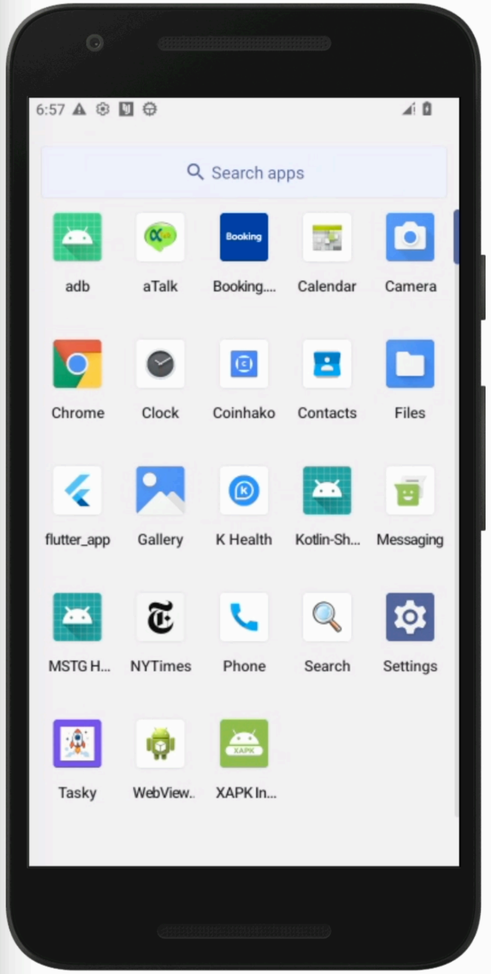
Bypass root detection in a real world mobile app

mastg@mobile-playbook: ~/Mobile-Playbook/Android/Apps

File Actions Edit View Help

mastg@...t_tool | mastg@...esktop | mastg@mobile-playb.../iOS/Frida-Scripts | mastg@mobile-pla...ook/Android/Apps

```
┌──(mastg㉿mobile-playbook)-[~/Mobile-Playbook/Android/Apps]
└─$
```



© 2025 by Bai7 GmbH | All rights reserved

Bypass root detection in a real world mobile app

Root detection is implemented through the Jumio mobile SDK. By only hooking into this method, the root detection can be bypassed.

Files

master

Go to file t

- docs
  - images
  - changelog.md
  - integration_faq.md
  - integration_glossary.md
  - integration_guide.md
  - known_issues.md
  - maintenance_policy.md
  - transition_guide.md
- licenses
- sample
- .gitignore
- README.md

mobile-sdk-android / docs / integration_guide.md ↑ Top

Preview | Code | Blame | 1149 lines (845 loc) · 74.1 KB | Raw

SDK Version Check

Use `JumioSDK.sdkVersion` to check which SDK version is being used.

Root Detection

For security reasons, applications implementing the SDK should not run on rooted devices. Use either the below method or a self-devised check to prevent usage of SDK scanning functionality on rooted devices.

```
JumioSDK.isRooted(context: Context)
```

⚠️ **Note:** Please be aware that the JumioSDK root check uses various mechanisms for detection, but doesn't guarantee to detect 100% of all rooted devices.

Device Supported Check

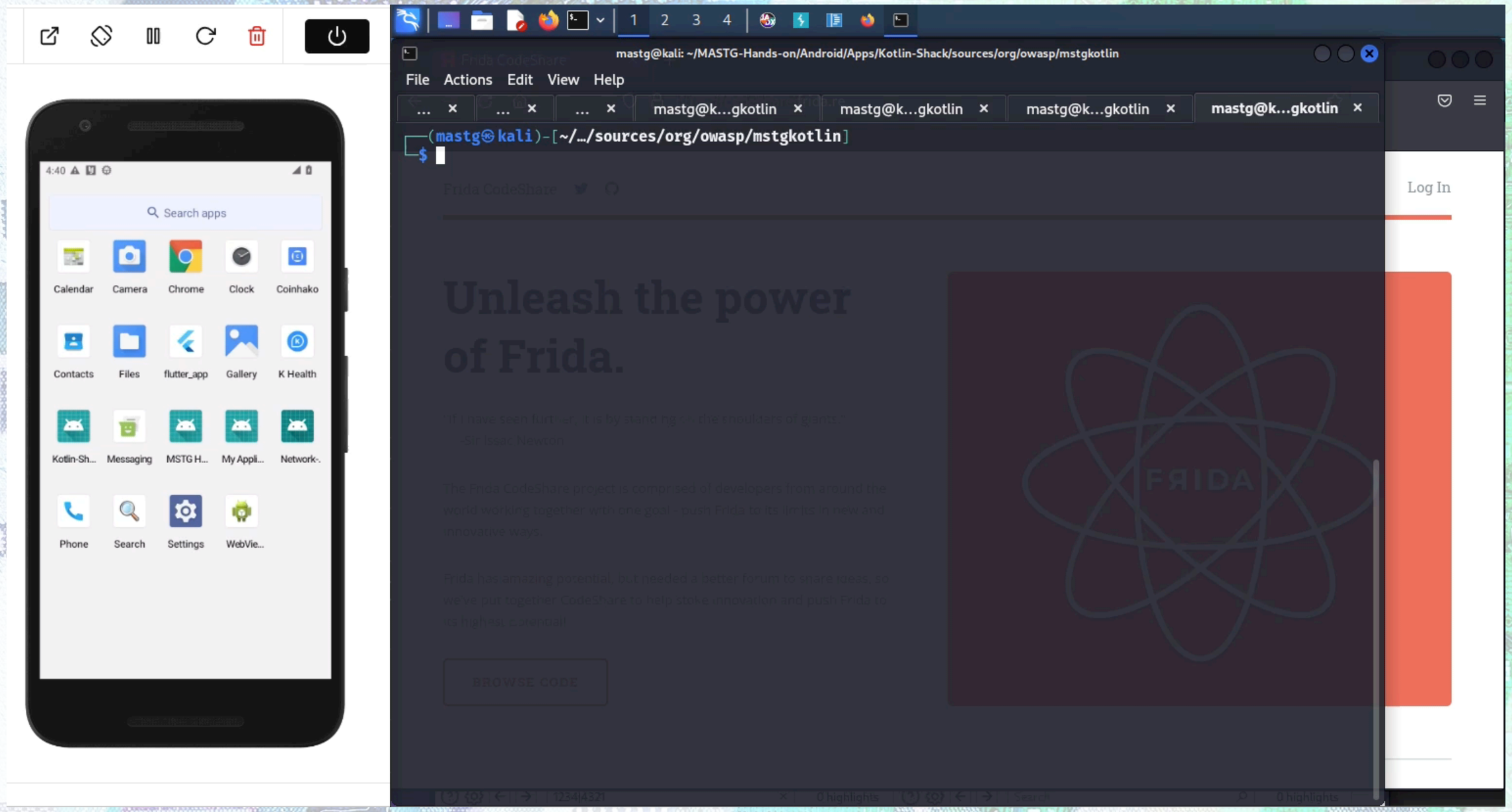
Use the method below to check if the current device platform is supported by the SDK.

```
JumioSDK.isSupportedPlatform(context: Context)
```

https://github.com/Jumio/mobile-sdk-android/blob/master/docs/integration_guide.md#root-detection

© 2025 by Bai7 GmbH | All rights reserved

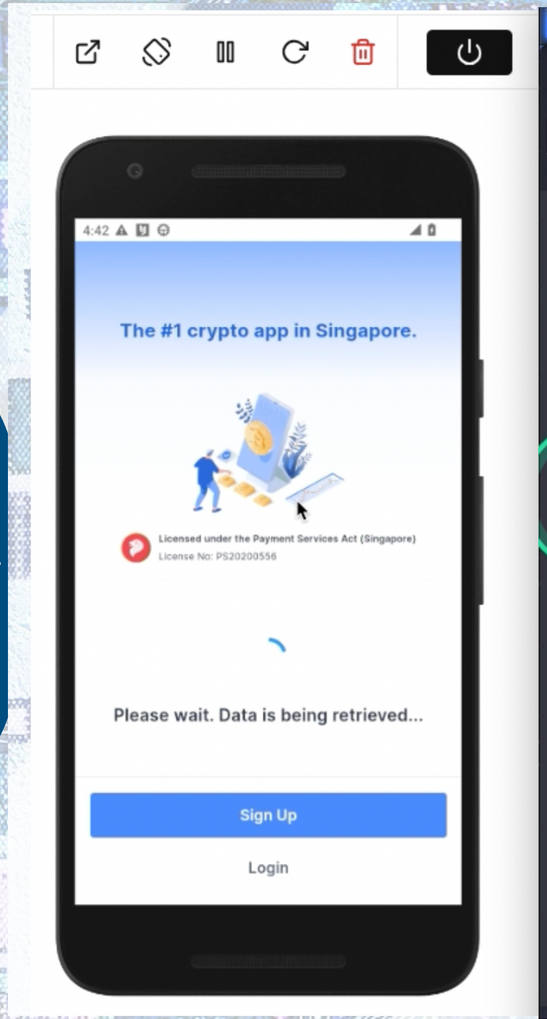
Bypass root detection in a real world mobile app

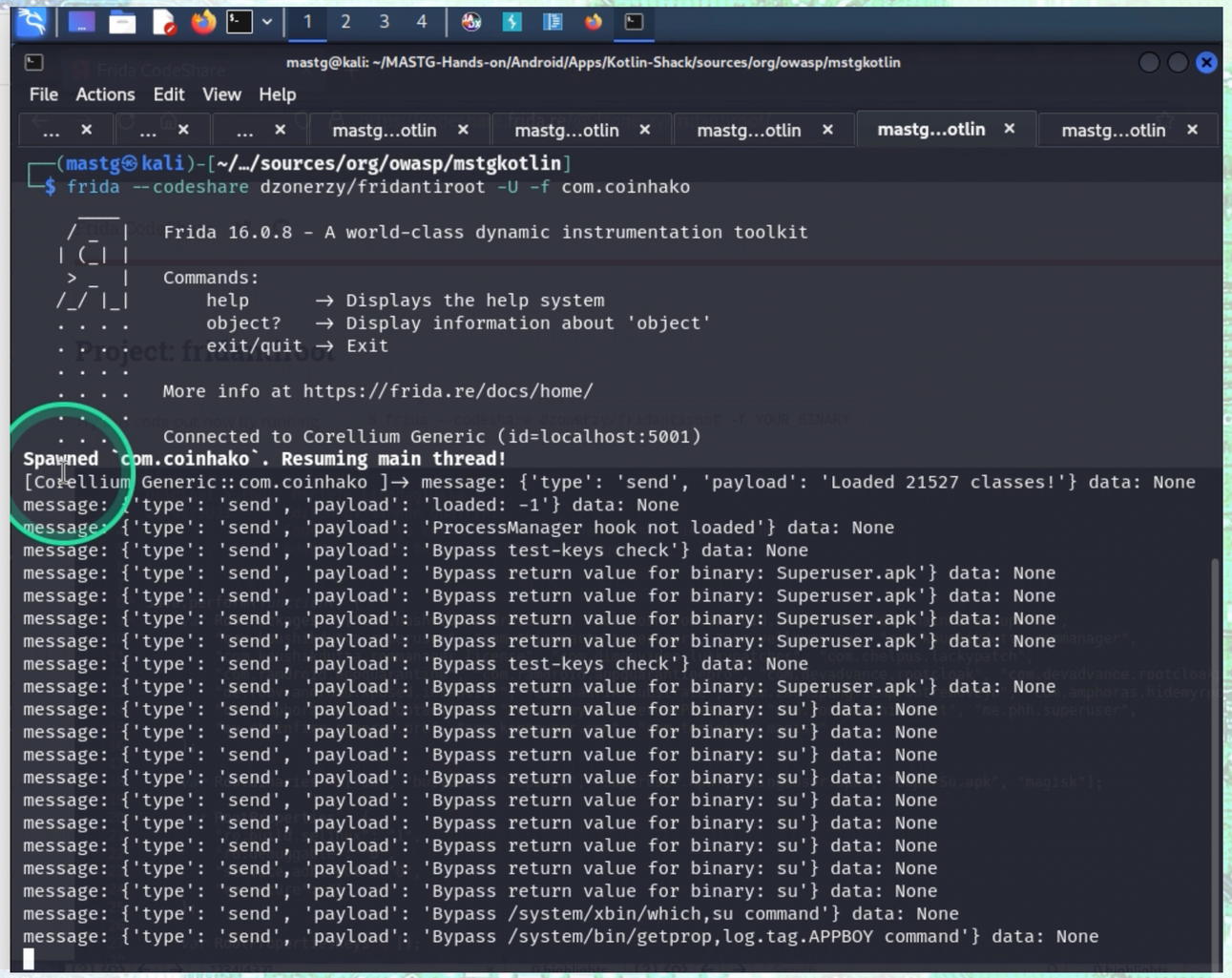


© 2025 by Bai7 GmbH | All rights reserved

Bypass root detection in a real world mobile app

Root detection is implemented through the Jumio mobile SDK. Also generic scripts to bypass root detection will work.





https://codeshare.frida.re/@dzonerzy/fridantiroot/

© 2025 by Bai7 GmbH | All rights reserved

Android Training

Anti Reverse Engineering Controls

Anti Reverse Engineering Controls

How can developers make Reverse Engineering attacks harder?

© 2025 by Bai7 GmbH | All rights reserved

Anti Reverse Engineering Controls

How can developers make Reverse Engineering attacks harder?

- Root detection ***combined*** with
- Detection of Dynamic Instrumentation (Frida)
- Anti Tampering
- Code Obfuscation
- and many more…

Client-Side Security Controls are always a cat and mouse game!

© 2025 by Bai7 GmbH | All rights reserved

Anti Reverse Engineering Controls

Overview of Android reverse engineering controls and it's test cases in MASVS-RESILIENCE:

https://mas.owasp.org/MASTG/tests/android/MASVS-RESILIENCE/MASTG-TEST-0038/

MASVS-RESILIENCE

- MASTG-TEST-0038: Making Sure that the App is Properly Signed
- MASTG-TEST-0039: Testing whether the App is Debuggable
- MASTG-TEST-0040: Testing for Debugging Symbols
- MASTG-TEST-0041: Testing for Debugging Code and Verbose Error Logging
- MASTG-TEST-0045: Testing Root Detection
- MASTG-TEST-0046: Testing Anti-Debugging Detection
- MASTG-TEST-0047: Testing File Integrity Checks
- MASTG-TEST-0048: Testing Reverse Engineering Tools Detection
- MASTG-TEST-0049: Testing Emulator Detection
- MASTG-TEST-0050: Testing Runtime Integrity Checks
- MASTG-TEST-0051: Testing Obfuscation

© 2025 by Bai7 GmbH | All rights reserved

Anti Reverse Engineering Controls

Overview of Android reverse engineering controls and it's test cases in MASVS-RESILIENCE:

https://mas.owasp.org/MASTG/tests/android/MASVS-RESILIENCE/MASTG-TEST-0038/

MASVS-RESILIENCE

- MASTG-TEST-0038: Making Sure that the App is Properly Signed
- MASTG-TEST-0039: Testing whether the App is Debuggable
- MASTG-TEST-0040: Testing for Debugging Symbols
- MASTG-TEST-0041: Testing for Debugging Code and Verbose Error Logging
- MASTG-TEST-0045: Testing Root Detection
- MASTG-TEST-0046: Testing Anti-Debugging Detection
- MASTG-TEST-0047: Testing File Integrity Checks
- MASTG-TEST-0048: Testing Reverse Engineering Tools Detection
- MASTG-TEST-0049: Testing Emulator Detection
- MASTG-TEST-0050: Testing Runtime Integrity Checks
- MASTG-TEST-0051: Testing Obfuscation

Would you implement all of these controls to be more resilient?

© 2025 by Bai7 GmbH | All rights reserved

Anti Reverse Engineering Controls

Never blindly implement them, as they add complexity into the app!

You should make a Threat Model of the app and identify if additional protection is needed and which controls might make sense to address the threat(s).

MASVS-RESILIENCE

- MASTG-TEST-0038: Making Sure that the App is Properly Signed
- MASTG-TEST-0039: Testing whether the App is Debuggable
- MASTG-TEST-0040: Testing for Debugging Symbols
- MASTG-TEST-0041: Testing for Debugging Code and Verbose Error Logging
- MASTG-TEST-0045: Testing Root Detection
- MASTG-TEST-0046: Testing Anti-Debugging Detection
- MASTG-TEST-0047: Testing File Integrity Checks
- MASTG-TEST-0048: Testing Reverse Engineering Tools Detection
- MASTG-TEST-0049: Testing Emulator Detection
- MASTG-TEST-0050: Testing Runtime Integrity Checks
- MASTG-TEST-0051: Testing Obfuscation

© 2025 by Bai7 GmbH | All rights reserved

Anti Reverse Engineering Controls

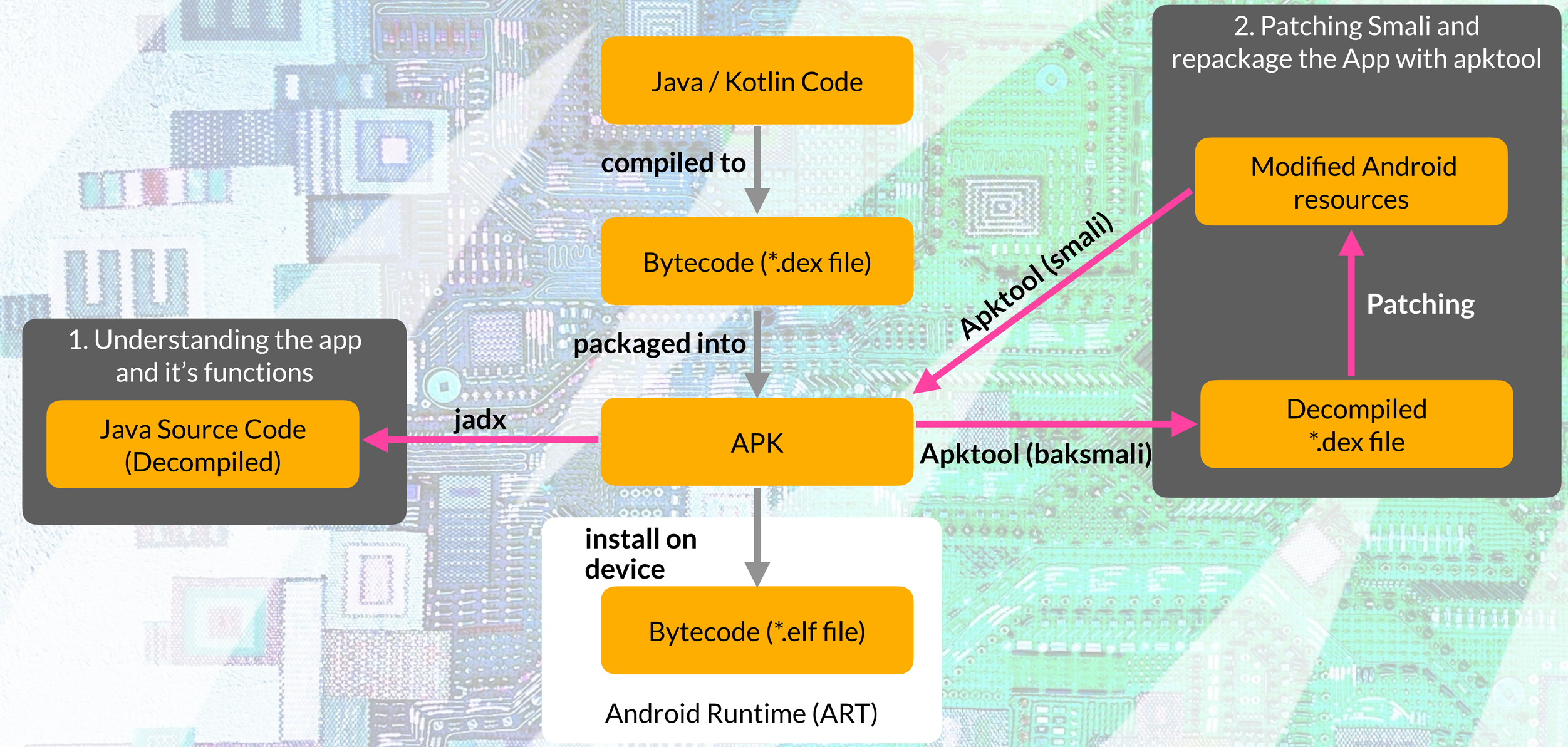
Example threat:

An attacker is able to easily decompile an APK to retrieve the original source code.

Mitigation:
The developer wants to protect their source code and wants to make the reverse engineering process harder through code obfuscation.

© 2025 by Bai7 GmbH | All rights reserved

Android Reverse Engineering - Smali and Repacking



© 2025 by Bai7 GmbH | All rights reserved

Anti Reverse Engineering Controls

```
/tmp/Android ❯                                                    ✘ INT ♣ base
```

By Sven Schleier

© 2025 by Bai7 GmbH | All rights reserved

Anti Reverse Engineering Controls

Code obfuscation is an umbrella term, what does it mean? Encrypting strings, "minifyEnabled = true", Apply complex control flow obfuscation ?

R8 is a tool that comes with Android Studio to shrink, optimise and obfuscate the code. An alternatives is ProGuard.

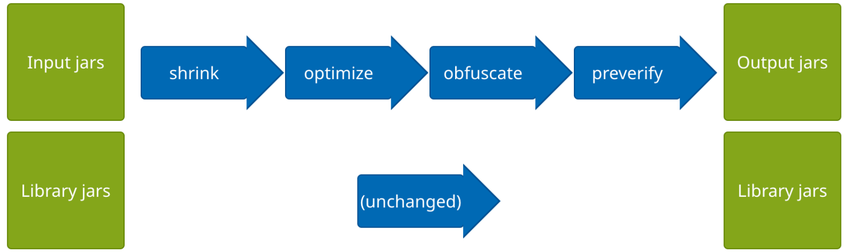
For example, what *can* be obfuscated:

- Function names and Class names (public and private)
- Variable Names

For example, what *cannot* be obfuscated :

- Activity names referenced in AndroidManifest.xml
- SDKs or libraries being used

How it works



ProGuard first reads the **input jars** (or aars, wars, ears, zips, apks, or directories). It then subsequently shrinks, optimizes, obfuscates, and preverifies them. You can optionally let ProGuard perform multiple optimization passes. ProGuard writes the processed results to one or more **output jars** (or aars, wars, ears, zips, apks, or directories). The input may contain resource files, whose names and contents can optionally be updated to reflect the obfuscated class names.

ProGuard requires the **library jars** (or aars, wars, ears, zips, apks, or directories) of the input jars to be specified. These are essentially the libraries that you would need for compiling the code. ProGuard uses them to reconstruct the class dependencies that are necessary for proper processing. The library jars themselves always remain unchanged. You should still put them in the class path of your final application.

https://www.guardsquare.com/manual/home

© 2025 by Bai7 GmbH | All rights reserved

Anti Reverse Engineering Controls

Example of using code shrinking, obfuscation and optimization for an Android App, which is available in Android Studio for free:

Kotlin build.gradle.kts | Groovy build.gradle

```
android {
    buildTypes {
        getByName("release") {
            // Enables code shrinking, obfuscation, and optimization for only
            // your project's release build type. Make sure to use a build
            // variant with `isDebuggable=false`.
            isMinifyEnabled = true

            // Enables resource shrinking, which is performed by the
            // Android Gradle plugin.
            isShrinkResources = true

            proguardFiles(
                // Includes the default ProGuard rules files that are packaged with
                // the Android Gradle plugin. To learn more, go to the section about
                // R8 configuration files.
                getDefaultProguardFile("proguard-android-optimize.txt"),

                // Includes a local, custom Proguard rules file
                "proguard-rules.pro"
            )
        }
    }
    ...
}
```

https://developer.android.com/build/shrink-code

© 2025 by Bai7 GmbH | All rights reserved

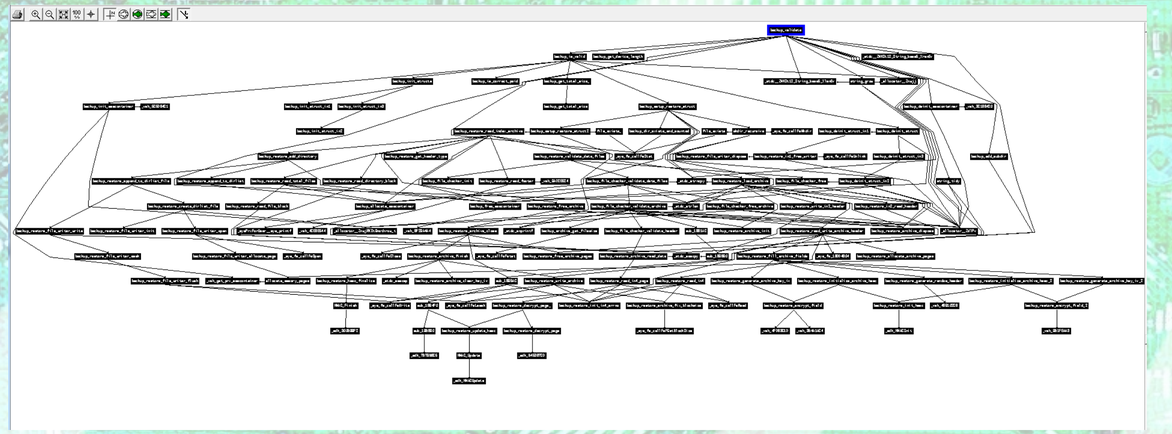
Anti Reverse Engineering Controls

© 2025 by Bai7 GmbH | All rights reserved

Anti Reverse Engineering Controls

Downside of code obfuscation:

- You shouldn't apply it when using:
  - Java Native Interface (JNI)
  - Reflection
  - For such instances exceptions need to be defined for the obfuscation tool to exclude the classes/methods.
- Obfuscation will not be applied to 3rd party libraries.
- Debugging is becoming more complex and also stack traces will be harder to understand. See "Retracing Stacktraces" on how to handle debugging with an obfuscated app.



© 2025 by Bai7 GmbH | All rights reserved

Anti Reverse Engineering Controls

Every Resilience Control comes with a price in terms of complexity and overhead for the development team and it will always only raise the bar for a reverse engineer.

It is important to understand what you want to protect against and how much money and effort you are willing to spend to slow down a reverse engineer.

Eventually the Reverse Engineer will always win, but you need to find the sweet spot to make it complex enough that attackers loose interest but that the development overhead is still manageable.

MASVS-RESILIENCE

- MASTG-TEST-0038: Making Sure that the App is Properly Signed
- MASTG-TEST-0039: Testing whether the App is Debuggable
- MASTG-TEST-0040: Testing for Debugging Symbols
- MASTG-TEST-0041: Testing for Debugging Code and Verbose Error Logging
- MASTG-TEST-0045: Testing Root Detection
- MASTG-TEST-0046: Testing Anti-Debugging Detection
- MASTG-TEST-0047: Testing File Integrity Checks
- MASTG-TEST-0048: Testing Reverse Engineering Tools Detection
- MASTG-TEST-0049: Testing Emulator Detection
- MASTG-TEST-0050: Testing Runtime Integrity Checks
- MASTG-TEST-0051: Testing Obfuscation

© 2025 by Bai7 GmbH | All rights reserved

Anti Reverse Engineering Controls

Provide 2 builds of the app to the penetration tester!

- Resiliceny controls, such as anti-debugging, obfuscation and root detection, may dramatically slow down a penetration test.
- The goal of a pentest is not to overcome these protection mechanisms but to test the business logic and functionality of the app.
- To efficiently use the testers time, you want to share two different app build variants:
  - Release build
  - Debug build (deactivated security controls)
- Testing two different builds is the most efficient way to cover all test cases for the mobile app and the APIs it's talking to.

© 2025 by Bai7 GmbH | All rights reserved

Android Training

(SSL | Certificate) Pinning

Android - Certificate Pinning

Real world attack vectors for an Adversary-in-the-middle attack (AITM)

- Attacks against Certificate Authorities (CA's)
- "Planting" a CA when installing a mobile app
- Nation state attacks against its citizens through the Internet Service Provider (ISP)
- Network traffic inspection through a TLS Proxy

© 2024 by Bai7 GmbH | All rights reserved

Protecting our Users in Kazakhstan

Wayne Thayer | **August 21, 2019**

Russian translation: Если вы хотите ознакомиться с этим текстом на русском языке, нажмите здесь.

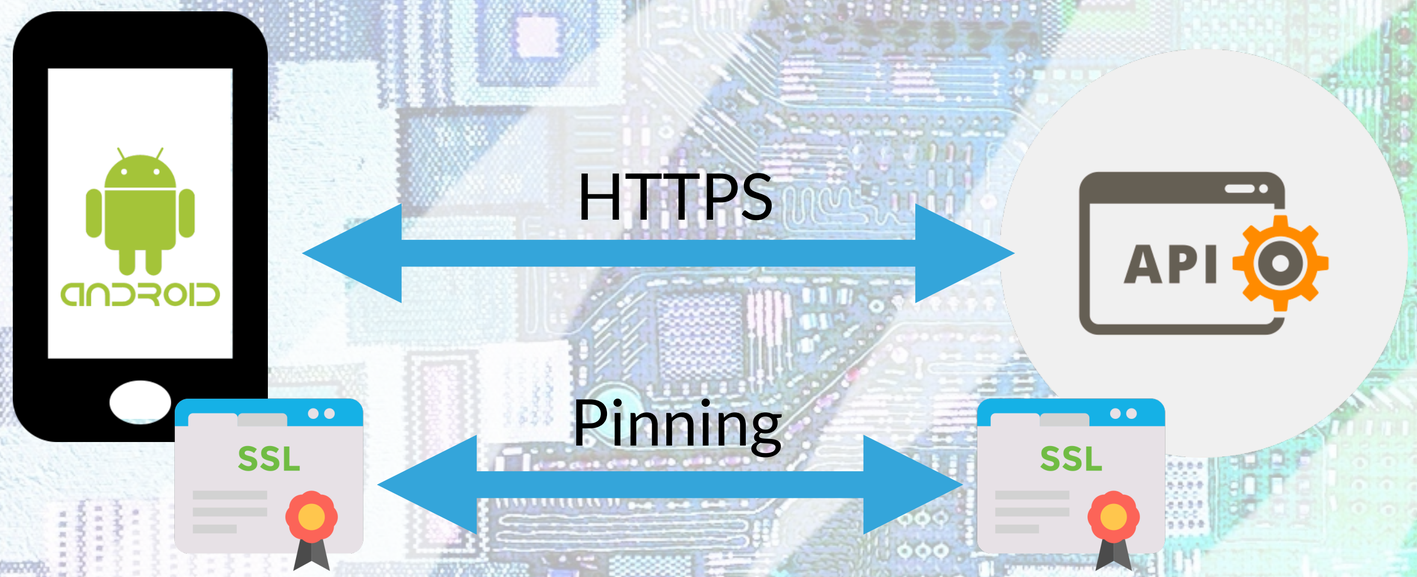
Kazakh translation: Бұл постыны қазақ тілінде мына жерден оқыңыз.

In July, a Firefox user informed Mozilla of a security issue impacting Firefox users in Kazakhstan: They stated that Internet Service Providers (ISPs) in Kazakhstan had begun telling their customers that they must install a government-issued root certificate on their devices. What the ISPs didn't tell their customers was that the certificate was being used to intercept network communications. Other users and researchers confirmed these claims, and listed 3 dozen popular social media and communications sites that were affected.

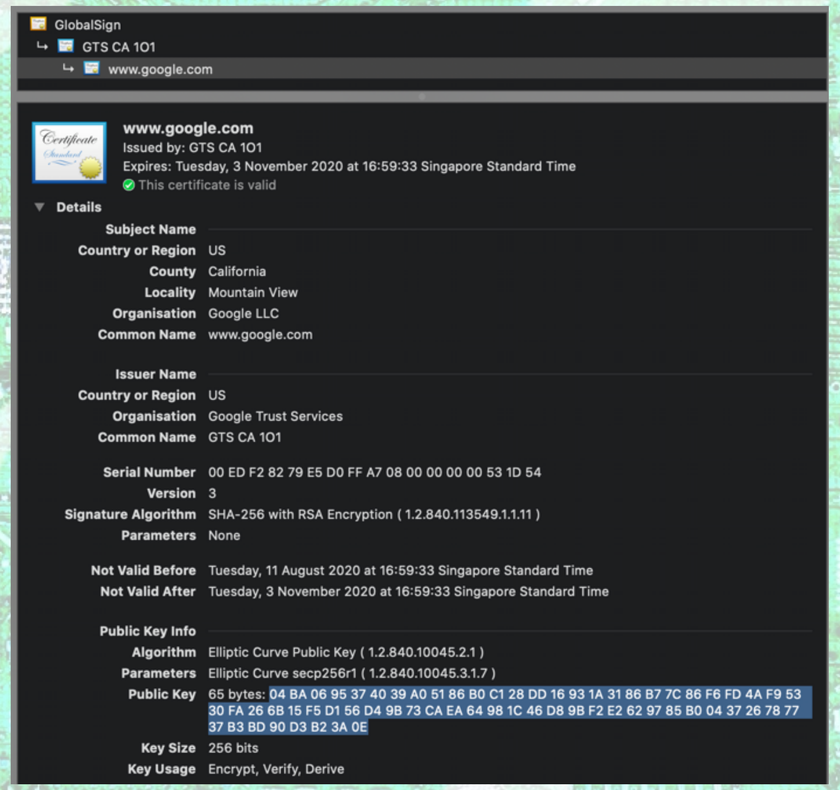
https://blog.mozilla.org/security/2019/08/21/protecting-our-users-in-kazakhstan/

© 2024 by Bai7 GmbH | All rights reserved

Android - Certificate Pinning



X.509 v3 Digital Certificate



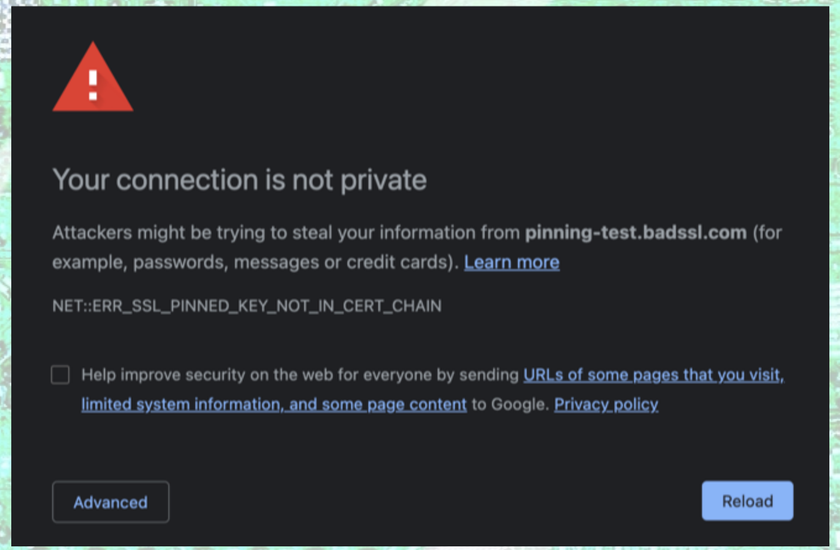
- SSL Pinning is an additional protection mechanism after the TLS handshake and it's checks were done successfully (expiry date, hostname etc.)
- Copy of server certificate or public key available in app (pinned in the app)
- For HTTPS requests the server certificate is verified with the pinned certificate or public key.

© 2024 by Bai7 GmbH | All rights reserved

Android - Certificate Pinning

What are the dangers of SSL Pinning aka should I pin?

- Is the organization mature enough…? Certificate replacement need to be communicated to development team pro-actively!
- PKI with certificate lifecycle management need to be established
- Private key need to be protected!
- Fail hard / Fail soft?



© 2024 by Bai7 GmbH | All rights reserved

Android - Certificate Pinning

Who should do Certificate Pinning?

© 2024 by Bai7 GmbH | All rights reserved

Android - Certificate Pinning

Who should do Certificate Pinning?

The decision for (not) using SSL Pinning should be the result of a Threat Model. Meaning Pentesters shouldn't blindly flag out missing SSL Pinning as a finding in their test reports.

© 2024 by Bai7 GmbH | All rights reserved

Android - Certificate Pinning

What can we do to disable Certificate Pinning

- Repackage the App and patch the SSL Pinning function
- Check out the MASTG-TECH-0012 in the MASTG
- Frida

© 2024 by Bai7 GmbH | All rights reserved

Open the file Lab-Android.pdf

Section:
Lab – Bypassing SSL Pinning

Note: Some PDF readers do not copy properly and miss whitespaces etc.

© 2024 by Bai7 GmbH | All rights reserved

Android - Certificate Pinning

Seems SSL Pinning can be bypassed quite easily…

Anything we can suggest to developers to make it harder for an attacker to bypass?

- SSL Pinning is a client-side control and with enough time and skills it will always be possible to bypass for a reverse engineer!
- Additional security controls might make it harder (Root detection, detection of dynamic instrumentation or manipulation of the APK)

© 2024 by Bai7 GmbH | All rights reserved

Android - Certificate Pinning

Pinning Certificates in Network Security Configuration

- ***Definition***: Enables certificate pinning, which restricts an app to trust only specific certificates or public keys, mitigating the risk of man-in-the-middle attacks even if a trusted CA is compromised.
- Use case:
  - Enhances security by ensuring that the app only trusts specific certificates.
  - Connecting to a host with a custom CA, such as a CA that is self-signed or is issued internally within a company.

```xml
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config>
        <domain includeSubdomains="true">example.com</domain>
        <pin-set expiration="2018-01-01">
            <pin digest="SHA-256">7HIpactkIAq2Y49orFOOQKurWxmmSFZhBCoQYcRhJ3Y=</pin>
            <!-- backup pin -->
            <pin digest="SHA-256">fwza0LRMXouZHRC8Ei+4PyuldPDcf3UKgO/04cDM1oE=</pin>
        </pin-set>
    </domain-config>
</network-security-config>
```

© 2024 by Bai7 GmbH | All rights reserved

Android - Certificate Pinning

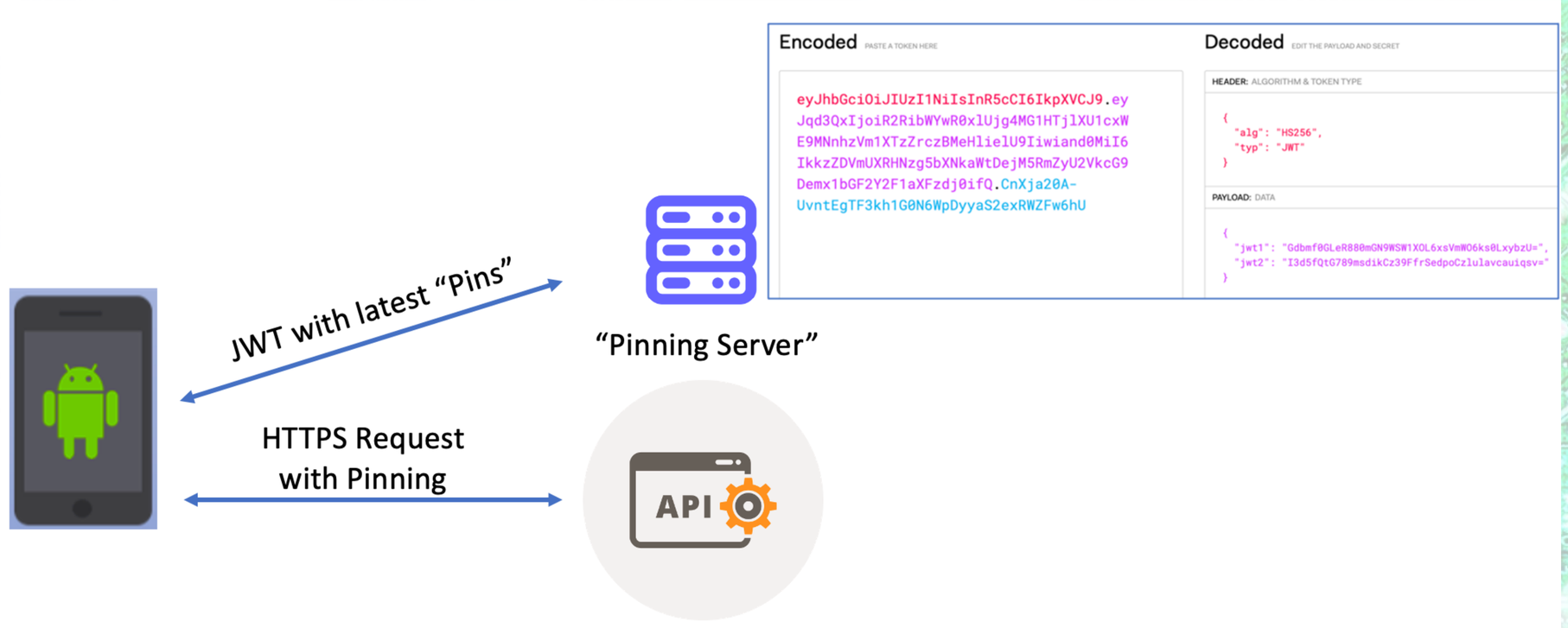
The certificate / Pins might change over time. How do I handle this?

- *Preloading:* Storing the pins hardcoded as part of the app is by far the easiest to implement. It can also be one of the more complex to maintain as a change in server certificates can mean you need to force your clients to upgrade your app.
- ***Trust on first use:*** The app can on first execution determine the public key by making a call to the server and storing this for all future executions. This can help when your certificates rotate frequently however leads the app open to key tainting on initialisation and pin expiration. If you don't know your endpoints ahead of time though this can be a good option however you still need to carefully manage pin failures.
- ***Over the air:*** For maximum flexibility creating a pin server that returns the current set of valid pins is a great option. You still need to pin the pin server but its then incredibly easy to manage the pins stored and update this when disaster strikes.

© 2024 by Bai7 GmbH | All rights reserved

Android - Certificate Pinning

Over-the-air example:



© 2024 by Bai7 GmbH | All rights reserved

Android - Certificate Pinning

Challenges that certificate pinning was addressing are less relevant today:

- Shorter certificate lifetimes
- Regular rotation of intermediate certificates
- Increased use of certificate transparency (available in Android 16+ via Network Security Configuration)

Don't pin unless you're sure you need to — and if you do, use key pinning with a backup key, not full cert pinning.

Restricting your app to specific certificates

Caution: Certificate pinning, the practice of restricting the certificates that are considered valid for your app to those you have previously authorized, is not recommended for Android apps. Future server configuration changes, such as changing to another CA, render apps with pinned certificates unable to connect to the server without receiving a client software update.

If you want to restrict your app to accept only certificates that you specify, it's critical to include multiple backup pins, including at least one key that's fully in your control, and a sufficiently short expiration period to prevent compatibility issues. The Network Security Config provides pinning with these capabilities.

https://developer.android.com/privacy-and-security/security-ssl

https://developer.apple.com/news/?id=g9ejcf8y

© 2025 by Bai7 GmbH | All rights reserved

Android - Certificate Pinning - Summary

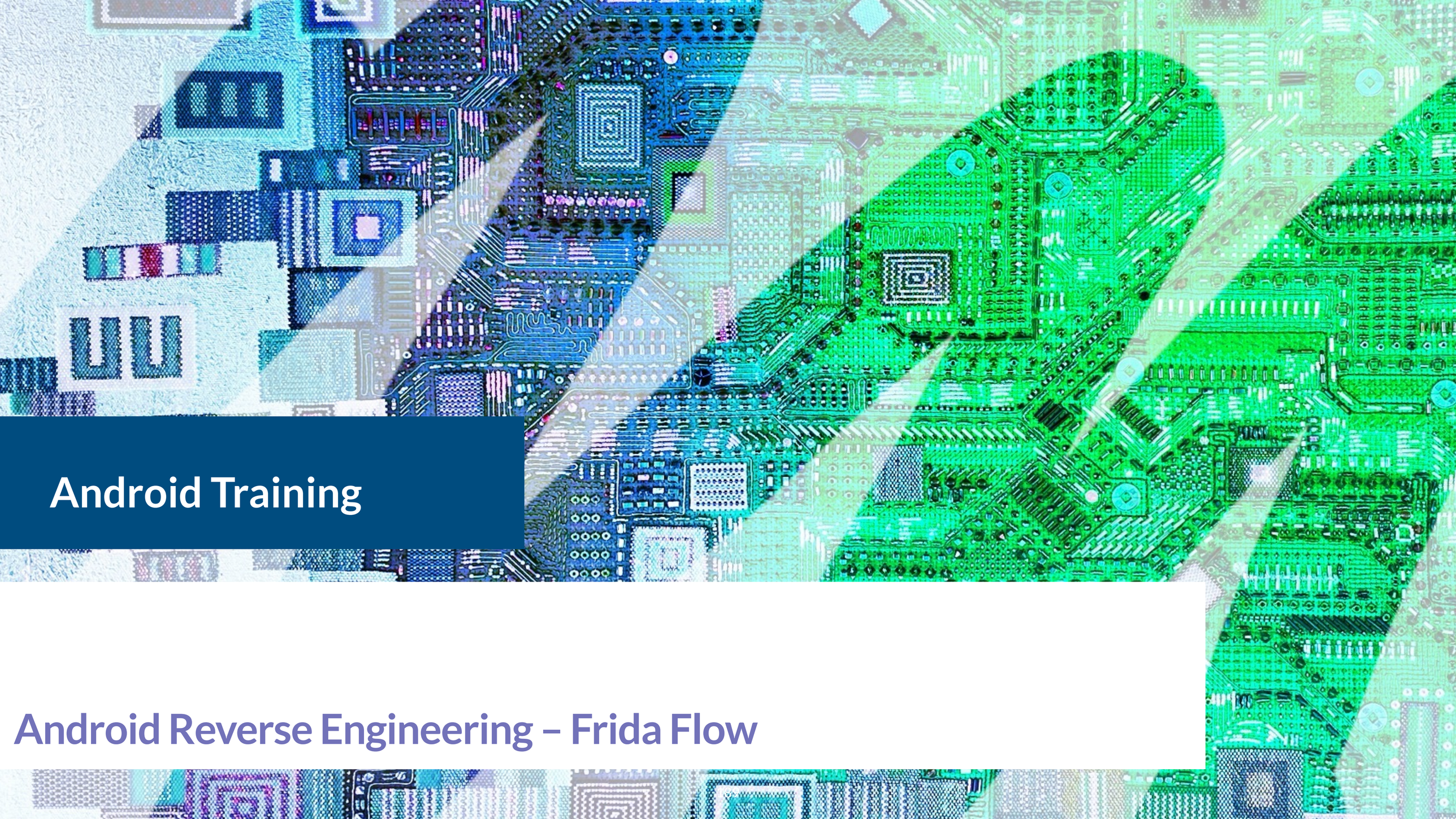
Certificate Pinning – Summary:

- TLS is based on trust and there are several ways to attack it.
- Certificate Pinning is an additional protection mechanism after the TLS handshake, but should only be implemented if absolutely necessary.
- Always use a library for implementing Certificate Pinning!
- BEFORE implementing Certificate Pinning make sure that a process is in place to renew the certificate in the mobile app. As otherwise you are risking a Denial of Service against your user base.

Certificate Pinning is a **proactive** control to prevent Man-in-the-Middle (MitM) attacks, where the app only trusts specific certificates or public keys. While highly effective, pinning requires ongoing maintenance as certificates change or expire.

Certificate Transparency ***(CT)*** was originally designed as a **reactive** control, allowing detection and auditing of misissued or rogue certificates after they are logged. However, modern implementations such as Chrome's enforcement on Android turn CT into a **proactive** control — by blocking HTTPS connections during the TLS handshake if a certificate is not found in the public CT logs.

© 2025 by Bai7 GmbH | All rights reserved



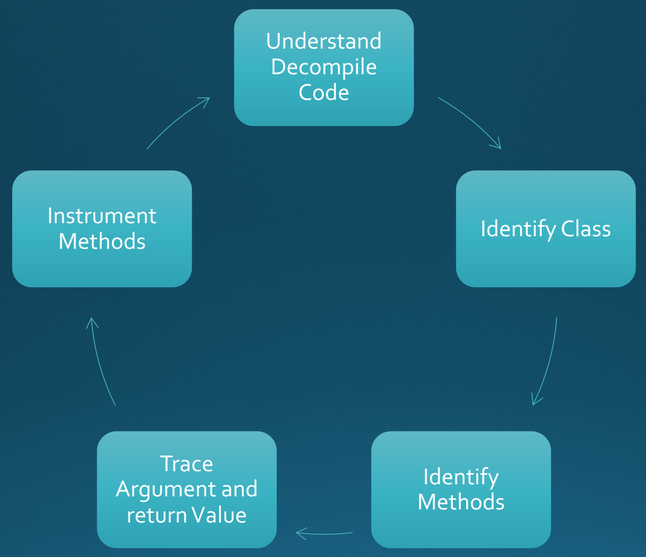
Android Training

Android Reverse Engineering – Frida Flow

Android Reverse Engineering - Frida Flow

- Understand decompiled code
- Enumerate and Identify Class
- Enumerate and Identify Method
- Trace argument and return value
- Instrument methods
- Repeat

The Ultimate Instrumentation Methodology



© 2025 by Bai7 GmbH | All rights reserved

Android Reverse Engineering - Frida Flow

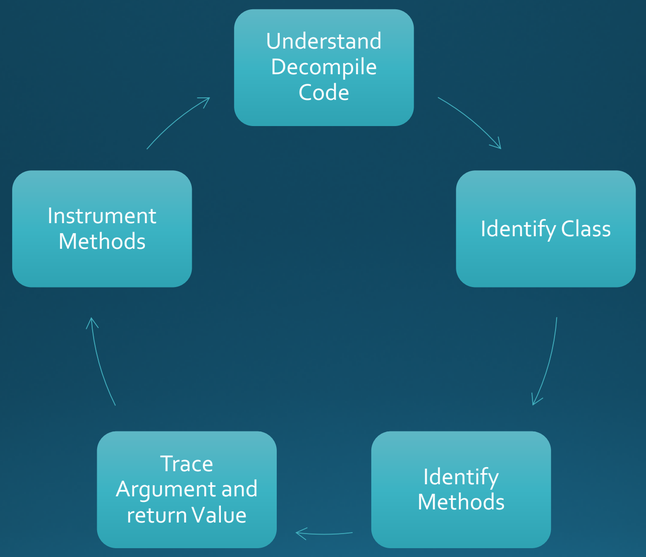
Understand decompiled code

- Not mandatory but aid to instrumentation process
- Tools such Jadx can help to decompile
- Essential in black box environment

Enumerate and Identify Class / Methods

- Usage of jadx-gui (see demo)

The Ultimate Instrumentation Methodology



© 2025 by Bai7 GmbH | All rights reserved

Android Reverse Engineering - Frida Flow

kali@kali: ~/MSTG-Handson/Android/Apps

File Actions Edit View Help

kali@kali: ~ | kali@kali: ~ | kali@kali: ~/MSTG-Handson/Android/Apps

```
┌──(kali㉿kali)-[~/MSTG-Handson/Android/Apps]
└─$ _
```

© 2025 by Bai7 GmbH | All rights reserved

Android Reverse Engineering - Frida Flow

Trace arguments and return values and Instrument methods

- Frida Java API
- Overload function and modify return value

```
setTimeout(function(){
    Java.perform(function (){
        console.log("[*] Script loaded")

        var MainActivity = Java.use("<Package-and-Class-Name>")

        MainActivity.<FUNCTION>.overload().implementation = function() {
            console.log("Script triggered")

        }

    });

});
```

© 2025 by Bai7 GmbH | All rights reserved

Android Training

Android Reverse Engineering – Anti-Frida

Android Reverse Engineering - Anti Frida

- Everyone loves Frida (not really ☹️)
- More apps are including mechanisms to detect when Frida is running.
- It is commonly implemented by:
  - Checking if Frida is in memory (frida-agent)
  - Checking if Frida server is running
  - Checking if signature is tampered

Check the MASTG for "Detection of Reverse Engineering Tools" and also "Detect Frida"

© 2025 by Bai7 GmbH | All rights reserved

Open the file Lab-Android.pdf

Section:
Lab – Anti Frida

Note: Some PDF readers do not copy properly and miss whitespaces etc.

© 2025 by Bai7 GmbH | All rights reserved

Android Training

CTF TIME!

Android Training

Wrap up Day 1

Android - Summary of Day 1

- **OWAPS MAS project:**
  - Use the OWASP MASVS to be aware of the most critical areas of the mobile attack surface.
  - Use the OWASP MASTG to get recipes on how to do mobile app security testing.
  - The OWASP MASVS is recognised by Android and linked into the developer documentation, as part of the Android Application Security Knowledge Base (AAKB) Project.
- **Secret Scanning:**
  - Use tools like git-secrets, truffleHog, or gitleaks to scan your codebase for potential exposure of secrets, both before committing and periodically as part of your CI/CD pipeline.
  - Secret Scanners need maintenance and adjustment and will produce false positives, as they are based on Regular Expressions.

© 2025 by Bai7 GmbH | All rights reserved

Android - Summary of Day 1

- **SAST**:
  - ***First step for SAST:*** Start small with a subset of the available vulnerability checks and enable more checks after others have been remediated and are under control!
  - In case you have a variety of different programming languages to cover, consider DAST scanning instead of SAST!

- **Software Composition Analysis (SCA):**
  - Scanning of 3rd party libraries for security vulnerabilities
  - Vulnerable libraries need to be updated or removed
  - SBOM is an essential part Application Security and need to be continuously created and monitored

- **Sensitive Data in mobile Apps:**
  - Only store data in the internal/external storage that is needed for the mobile app.
  - Only encrypt sensitive data in the App's sandbox if there is a need to. If so, use the Android KeyStore, or the Security Crypto Library.

© 2025 by Bai7 GmbH | All rights reserved

Android - Summary of Day 1

- **Reverse Engineering:**
  - To efficiently use the testers time, you want to share two different app build variants:
    - Release build
    - Debug build (deactivated security controls)
  - Reverse Engineers can decompile an (X)APK back to it's source code.
  - Frida is a powerful reverse engineering tool to attack mobile apps and bypass it's security controls and assess the security level of mobile apps.

© 2025 by Bai7 GmbH | All rights reserved

And we're done… Thanks for your time!

Any more questions?

Please drop them in the chat or shout out.

Otherwise contact me:
sven@bai7.at

Linkedin

Thank you for your time today and I hope you can use your new skills soon and spread your new knowledge!

See you tomorrow.

© 2025 by Bai7 GmbH | All rights reserved