Semgrep Rules for Android Application Security

Riccardo Cardelli

OWASP Italy Day 2023

Politecnico of Milan - 11th September 2023

OWASP®

POLITECNICO
MILANO 1863

Riccardo Cardelli

Senior Software Security Consultant & Trainer @ IMQ Minded Security

Engineer in Computer Science | Penetration Tester | Trainer | Project Leader | CTF Player | Bug Bounty Hunter | ex Cyberchallenger

but most of all

Pianist | Guitarist | Reader | Gameboard Enthusiast | Maker | Traveler | Very Bad Skier

- https://github.com/gand3lf
- https://twitter.com/gand3lf
- https://www.linkedin.com/in/riccardo-cardelli/

Static Application Security Testing (SAST)

https://owasp.org/www-community/Source_Code_Analysis_Tools

- What is a SAST tool?

A Static Application Security Testing (**SAST**) tool is a software application that analyzes source code to find potential security vulnerabilities.

- How to choose the correct SAST tool?

- Is it free?
- Is it customizable?
- What is the effort to implement specific rules?
- Does it upload customer source code on cloud?
- Is it fast?
- Is it reliable?
- What kinds of features does it provide?
- Does it require that the code is buildable?

SAST (open source)	Commits
CodeQL	~ 58000
SonarQube	~ 35000
Semgrep	~ 6000
Joern	~ 2600
. . .	. . .

Semgrep

- What is?

Semgrep is a Static Application Security Testing (**SAST**) tool.

- What does it do?

“*Semgrep performs **intra-file analysis**, that is, **it analyzes one file at a time** and in isolation.*”

- How?

Semgrep allows you to **define patterns** of code to **detect** misconfigurations or security issues.

- Is it secure?

Semgrep analyzes code locally on your computer or in your build environment: **code is never uploaded.**

- What does it need to work?

The target source (eventually partial) and the Semgrep ruleset.

Supported languages:

C#
Go
Java
JavaScript
JSON
JSX
PHP
Python
Ruby
Scala
Terraform
TSX
TypeScript

Experimental:

Bash
C
C++
Clojure
Dart
Dockerfile
Elixir
HTML
Jsonnet
Lisp
Lua
OCaml
R
Scheme
Solidity
Swift
YAML
XML
Generic (ERB, Jinja, etc.)

Semgrep rules - examples

```
rules:
  - id: hardcoded-credentials
    languages:
      - java
    message: Hardcoded credentials
detected.
    severity: WARNING
    pattern: password = "..."
```

```
  public class A {
    String password = "MyS3cr3tPwd$";
    private void method(int a){
      String password = "An0th3rPwd!";
    }
  }
```

```
rules:
  - id: sqli
    languages:
      - java
    message: Potential SQLi detected.
    severity: WARNING
    options:
      symbolic_propagation: true
    pattern: |
      $X = $R.getParameter("...");
      ...
      $DB.executeQuery(<... $X ...>);
```

```
String param = request.getParameter("name");
String query = "select * from users where name='"+ param + "'";
conn = DriverManager.getConnection("...");
stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery(query);
...
```

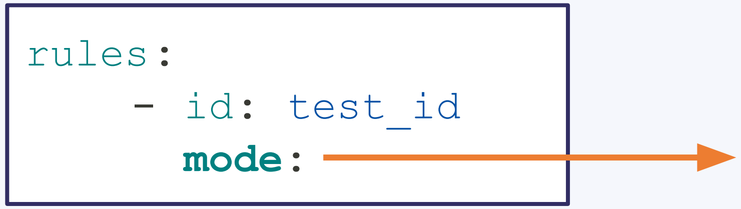
Semgrep rules

https://semgrep.dev/docs/writing-rules/pattern-syntax/

A Semgrep pattern is characterized by a simple syntax. Some of the main elements you can find inside a Semgrep rules are:

	match	example
. . .	a sequence of zero or more items such as arguments, statements, parameters, fields, characters	*pattern* try{ . . . }catch(. . .){ . . . }
“. . .“	any single hardcoded string	*pattern* password = “. . .”
$A	variables, functions, arguments, classes, object methods, imports, exceptions, and more	*pattern* $R.getParameter(. . .)
<. . . e . . .>	an expression that could be deeply nested within another expression	*pattern* <. . . 99 . . .>

Semgrep - Modes



```
rules:
    - id: test_id
      mode:
```

- **default**:
  Just the standard Semgrep rule

- **taint**:
  It enables the data-flow analysis feature allowing to specify sources and sinks

- **join**:
  It allows to use multiple rules on more than one file and to join the results

- **extract**:
  It allows to run existing rules on subsections of files where the rule language is different from the language of the file

Semgrep Rules for Android Application Security

https://github.com/mindedsecurity/semgrep-rules-android-security

The project was started in March 2023 by the **IMQ Minded Security** team with the purpose to contribute to the ethical hacking and mobile development communities.

https://blog.mindedsecurity.com/2023/06/a-cool-new-project-semgrep-rules-for.html

Proposal

The main purpose of this project is to provide a collection of Semgrep rules to cover the static tests described by the OWASP Mobile Application Security Testing Guide (MASTG) for Android applications.
The project will be released publicly on the IMQ Minded Security official Github page: https://github.com/mindedsecurity.

Advantages:

- useful
- highly parallelizable
- low effort
- technically resalable
- dynamic organization
- DevOps friendly

Supervisor

Stefano Di Paola (@WisecWisec)

Special thanks

Andrea Agnello
Federico Dotta (@apps3c)
Giovanni Fazi (@giovifazi)
Maurizio Siddu (@akabe1)
Michele Tumolo (@zer0s0urce)
Christian Cotignola (@b4dsheep)
Giacomo Zorzin (@gellge)
Martino Lessio (@mlessio)
Michele Di Bonaventura (@cyberaz0r)
Riccardo Granata

OWASP Mobile Application Security

https://mas.owasp.org/

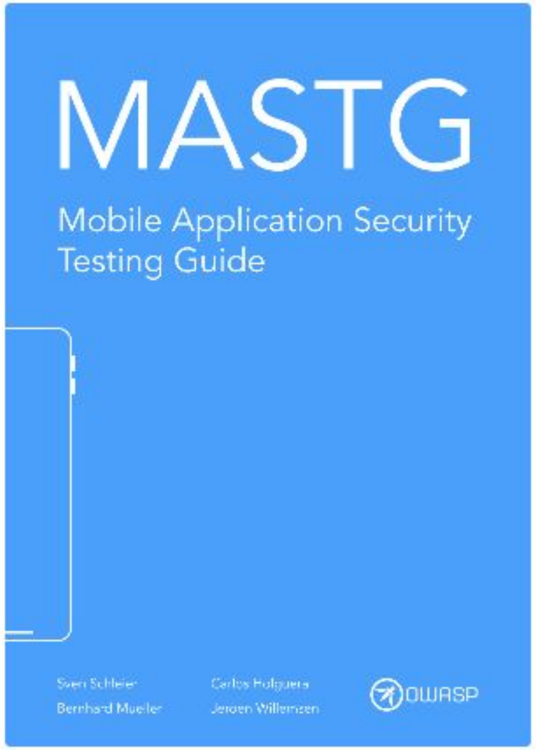
MASVS

Mobile Application Security Verification Standard

Sven Schleier
Bernhard Mueller
Jeroen Beckers

Carlos Holguera
Jeroen Willemsen

OWASP



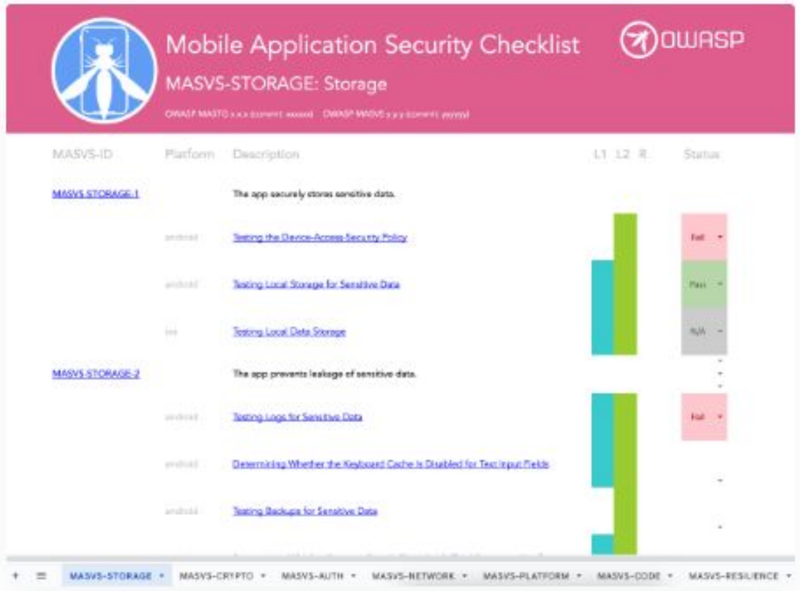
MASTG

Mobile Application Security Testing Guide

Sven Schleier
Bernhard Mueller

Carlos Holguera
Jeroen Willemsen

OWASP



Mobile Application Security Checklist

MASVS-STORAGE: Storage

OWASP MASTG x.x.x (commit: xxxxxx) OWASP MASVS y.y.y (commit: yyyyyy)

MASVS-ID	Platform	Description	L1	L2	R	Status
MASVS-STORAGE-1		The app securely stores sensitive data.				
	android	Testing the Device-Access-Security Policy				Fail
	android	Testing Local Storage for Sensitive Data				Pass
	ios	Testing Local Data Storage				N/A
MASVS-STORAGE-2		The app prevents leakage of sensitive data.				
	android	Testing Logs for Sensitive Data				Fail
	android	Determining Whether the Keyboard Cache Is Disabled for Text Input Fields				
	android	Testing Backups for Sensitive Data				

+ ☰ MASVS-STORAGE ▾ MASVS-CRYPTO ▾ MASVS-AUTH ▾ MASVS-NETWORK ▾ MASVS-PLATFORM ▾ MASVS-CODE ▾ MASVS-RESILIENCE ▾

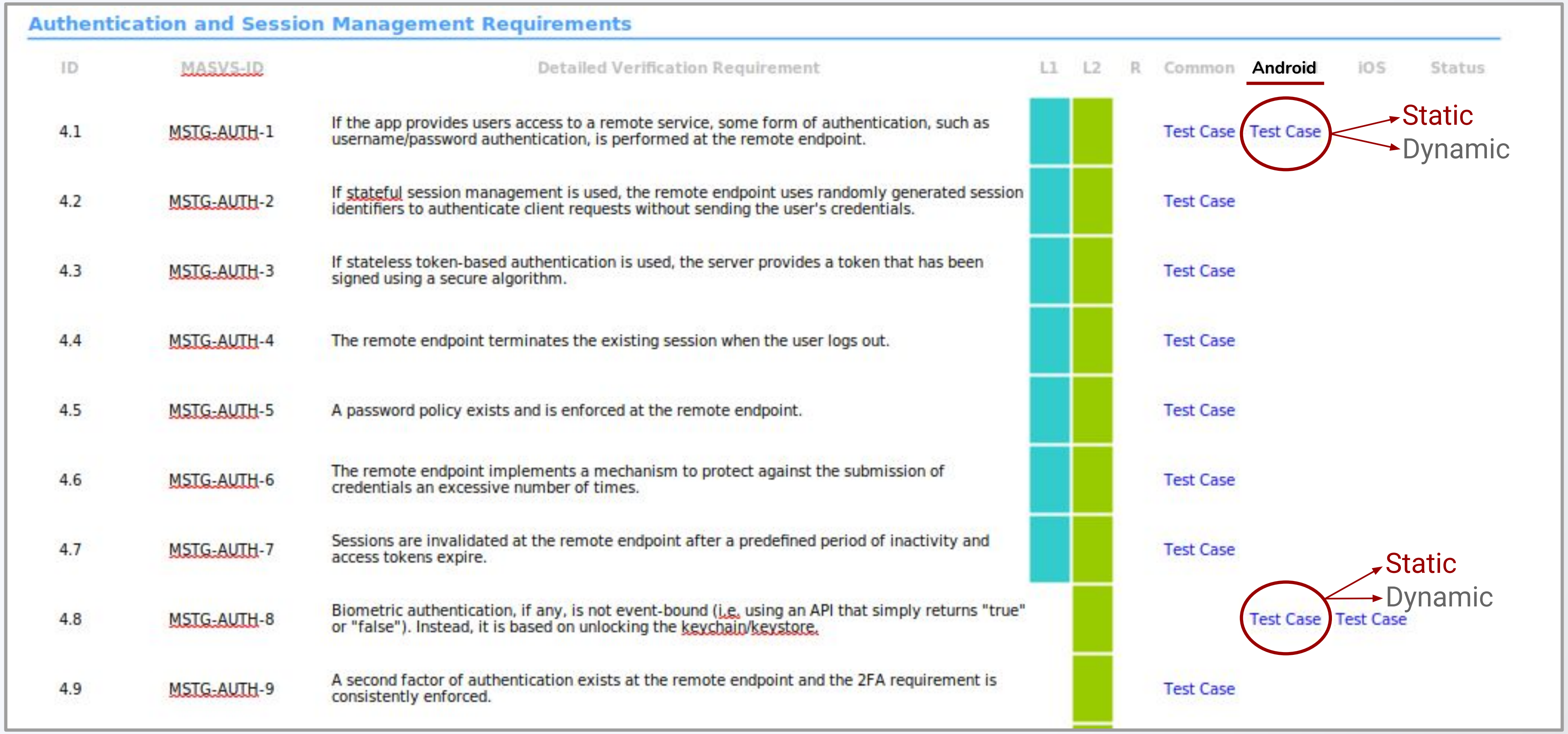
MASVS v2.0.0
MASVS v1.5.0

MASTG v1.6.0
MASTG v1.5.0

Checklist

OWASP MAS - Checklist

https://github.com/OWASP/owasp-mastg/releases/download/v1.5.0/Mobile_App_Security_Checklist_en.xlsx



Authentication and Session Management Requirements

ID	MASVS-ID	Detailed Verification Requirement	L1	L2	R	Common	Android	iOS	Status
4.1	MSTG-AUTH-1	If the app provides users access to a remote service, some form of authentication, such as username/password authentication, is performed at the remote endpoint.	🟦	🟩		Test Case	Test Case		
4.2	MSTG-AUTH-2	If stateful session management is used, the remote endpoint uses randomly generated session identifiers to authenticate client requests without sending the user's credentials.	🟦	🟩		Test Case			
4.3	MSTG-AUTH-3	If stateless token-based authentication is used, the server provides a token that has been signed using a secure algorithm.	🟦	🟩		Test Case			
4.4	MSTG-AUTH-4	The remote endpoint terminates the existing session when the user logs out.	🟦	🟩		Test Case			
4.5	MSTG-AUTH-5	A password policy exists and is enforced at the remote endpoint.	🟦	🟩		Test Case			
4.6	MSTG-AUTH-6	The remote endpoint implements a mechanism to protect against the submission of credentials an excessive number of times.	🟦	🟩		Test Case			
4.7	MSTG-AUTH-7	Sessions are invalidated at the remote endpoint after a predefined period of inactivity and access tokens expire.	🟦	🟩		Test Case			
4.8	MSTG-AUTH-8	Biometric authentication, if any, is not event-bound (i.e. using an API that simply returns "true" or "false"). Instead, it is based on unlocking the keychain/keystore.		🟩			Test Case	Test Case	
4.9	MSTG-AUTH-9	A second factor of authentication exists at the remote endpoint and the 2FA requirement is consistently enforced.		🟩		Test Case			

Checklist rows

84

Android test cases

51

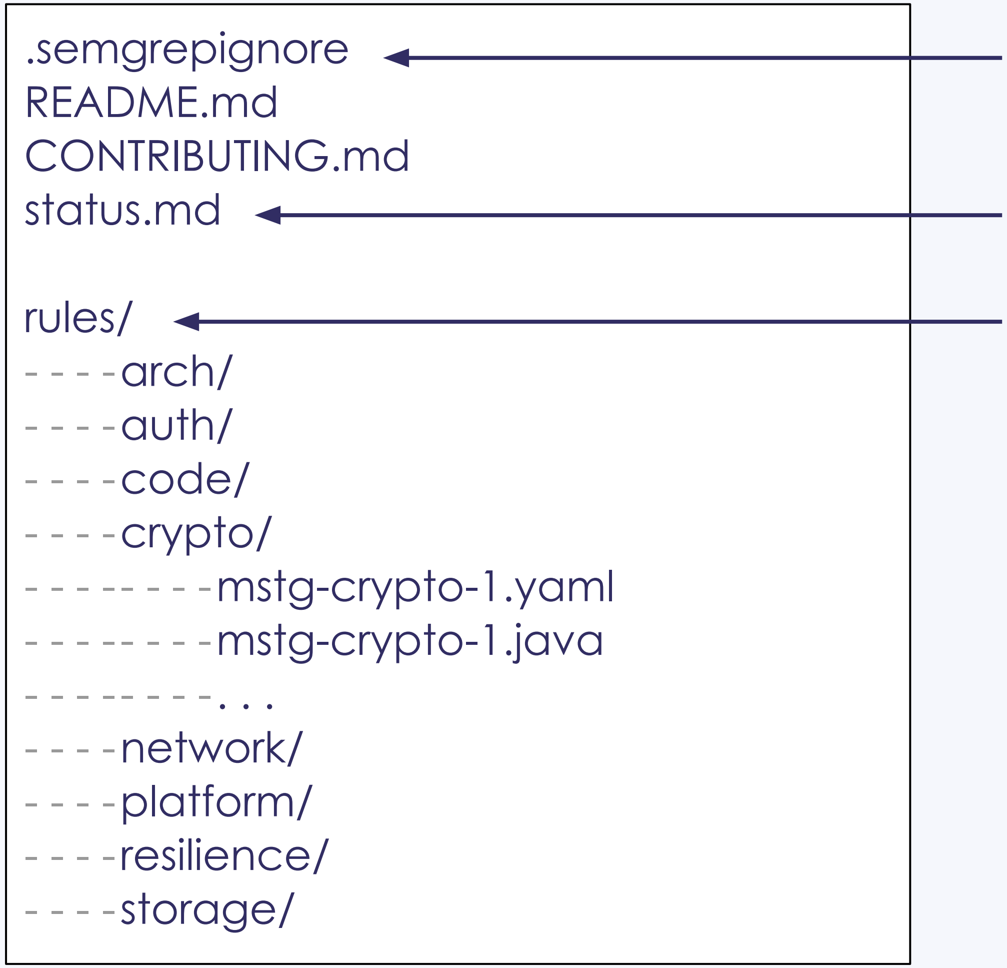
Implemented rules

42

New checklist rows

~141

Directory Structure



```
.semgrepignore
README.md
CONTRIBUTING.md
status.md

rules/
----arch/
----auth/
----code/
----crypto/
--------mstg-crypto-1.yaml
--------mstg-crypto-1.java
--------...
----network/
----platform/
----resilience/
----storage/
```

This file specify some **directory**/files that are **excluded** from the scan.

Each rule has been **classified** in order to keep track of the reliability of the project.

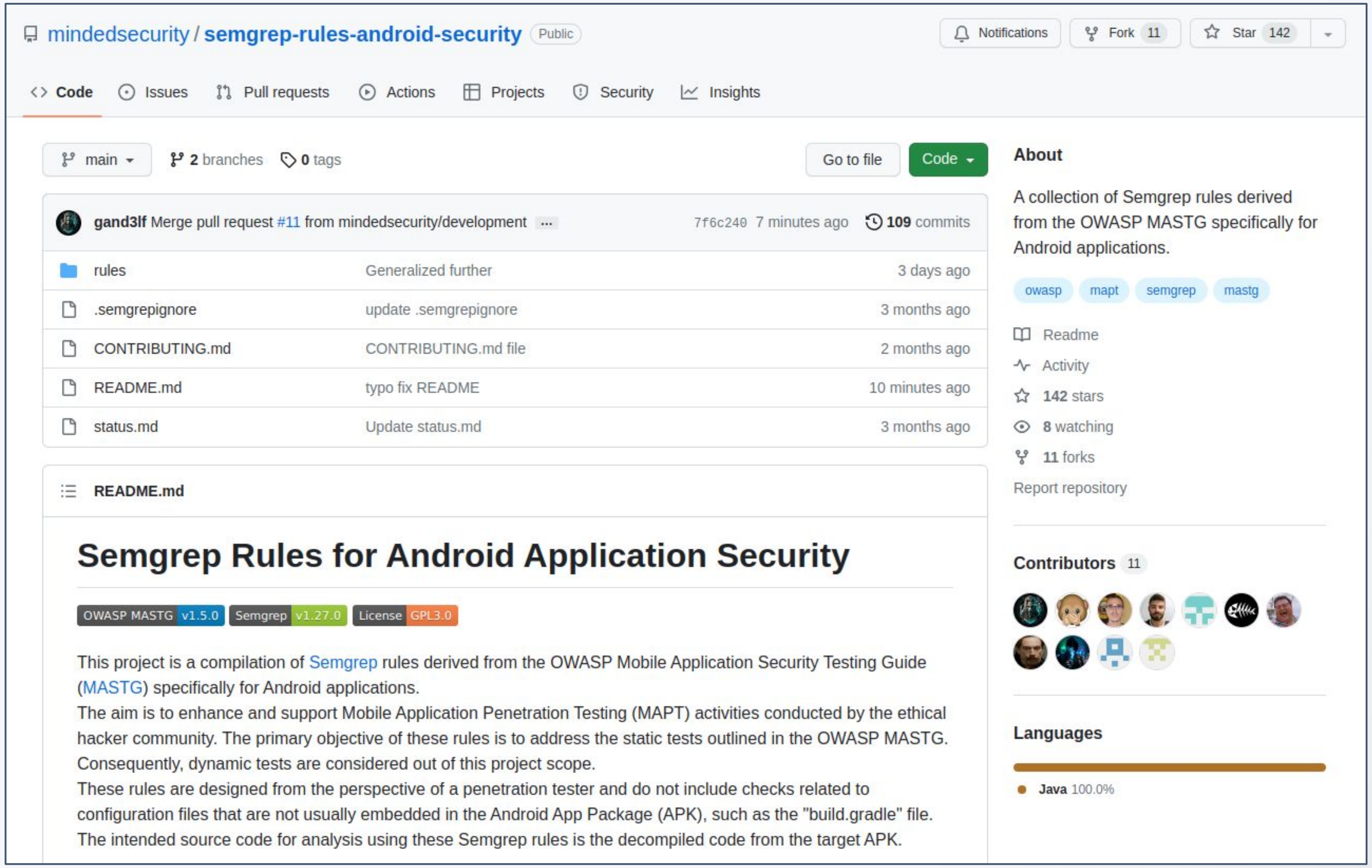
This is the main folder of the project. It contains both the **Semgrep rules** in YAML format and the related **test files**.

In the case where a test needs more than one rule to be implemented, the adopted nomenclature is the following:

```
mstg-platform-4.1.yaml
mstg-platform-4.1.java
mstg-platform-4.2.yaml
mstg-platform-4.2.java
mstg-platform-4.3.yaml
mstg-platform-4.3.xml
```

Semgrep Rules for Android Application Security

https://github.com/mindedsecurity/semgrep-rules-android-security



gand3lf Merge pull request #11 from mindedsecurity/development ... 7f6c240 7 minutes ago 109 commits

rules	Generalized further	3 days ago
.semgrepignore	update .semgrepignore	3 months ago
CONTRIBUTING.md	CONTRIBUTING.md file	2 months ago
README.md	typo fix README	10 minutes ago
status.md	Update status.md	3 months ago

README.md

Semgrep Rules for Android Application Security

OWASP MASTG v1.5.0 Semgrep v1.27.0 License GPL3.0

This project is a compilation of Semgrep rules derived from the OWASP Mobile Application Security Testing Guide (MASTG) specifically for Android applications.
The aim is to enhance and support Mobile Application Penetration Testing (MAPT) activities conducted by the ethical hacker community. The primary objective of these rules is to address the static tests outlined in the OWASP MASTG. Consequently, dynamic tests are considered out of this project scope.
These rules are designed from the perspective of a penetration tester and do not include checks related to configuration files that are not usually embedded in the Android App Package (APK), such as the "build.gradle" file.
The intended source code for analysis using these Semgrep rules is the decompiled code from the target APK.

About

A collection of Semgrep rules derived from the OWASP MASTG specifically for Android applications.

owasp mapt semgrep mastg

142 stars
8 watching
11 forks

Contributors 11

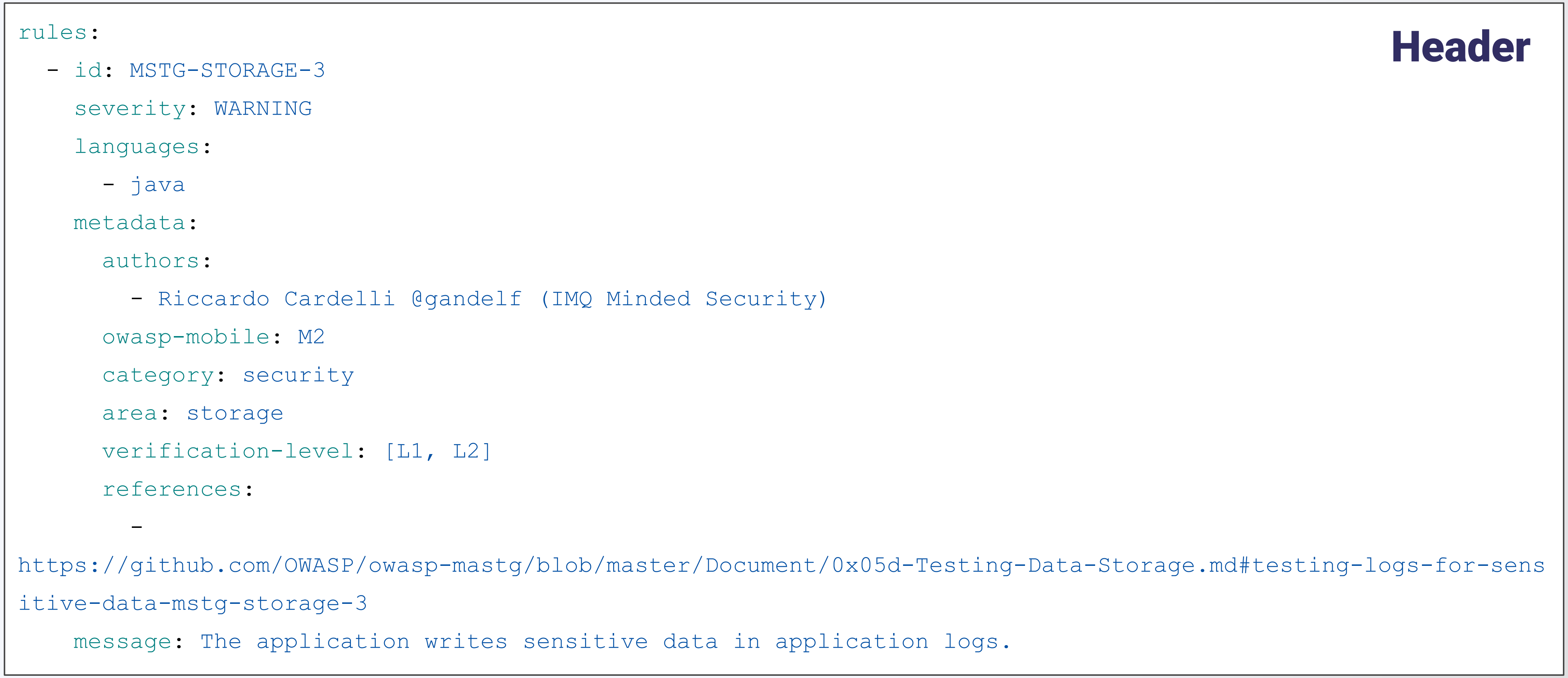
Languages

Java 100.0%

Testing Logs for Sensitive Data

(MSTG-STORAGE-3) L1(L2)

This test case focuses on identifying any **sensitive application data within** both system and application **logs**.

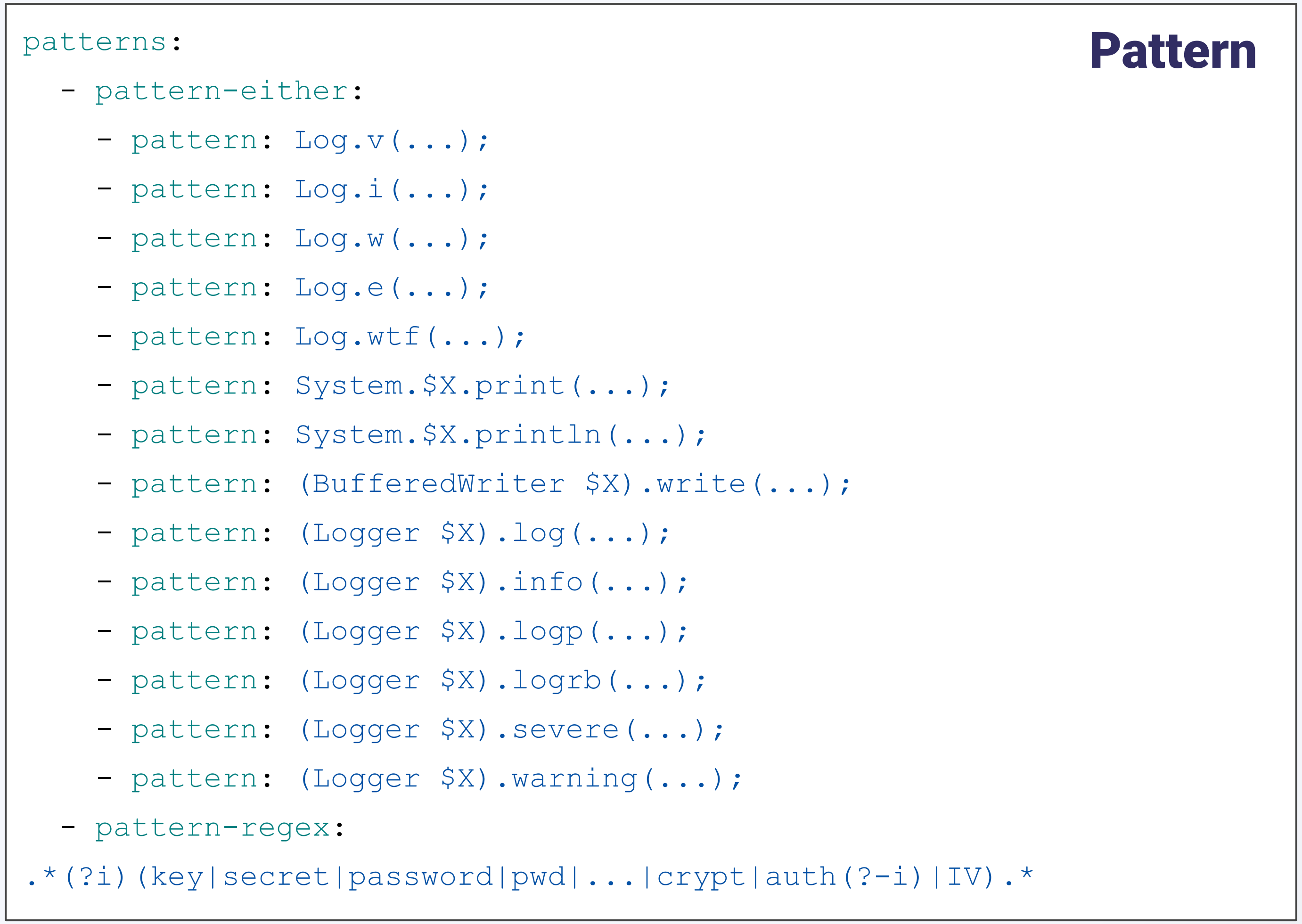


Header

```
rules:
  - id: MSTG-STORAGE-3
    severity: WARNING
    languages:
      - java
    metadata:
      authors:
        - Riccardo Cardelli @gandelf (IMQ Minded Security)
      owasp-mobile: M2
      category: security
      area: storage
      verification-level: [L1, L2]
      references:
        -
https://github.com/OWASP/owasp-mastg/blob/master/Document/0x05d-Testing-Data-Storage.md#testing-logs-for-sensitive-data-mstg-storage-3
    message: The application writes sensitive data in application logs.
```

Testing Logs for Sensitive Data

(MSTG-STORAGE-3) L1(L2)



Pattern

```
patterns:
  - pattern-either:
    - pattern: Log.v(...);
    - pattern: Log.i(...);
    - pattern: Log.w(...);
    - pattern: Log.e(...);
    - pattern: Log.wtf(...);
    - pattern: System.$X.print(...);
    - pattern: System.$X.println(...);
    - pattern: (BufferedWriter $X).write(...);
    - pattern: (Logger $X).log(...);
    - pattern: (Logger $X).info(...);
    - pattern: (Logger $X).logp(...);
    - pattern: (Logger $X).logrb(...);
    - pattern: (Logger $X).severe(...);
    - pattern: (Logger $X).warning(...);
  - pattern-regex:
.*(?i)(key|secret|password|pwd|...|crypt|auth(?-i)|IV).*
```

This test case focuses on identifying any **sensitive application data within** both system and application **logs**.

Testing Custom Certificate Stores and Certificate Pinning

(MSTG-NETWORK-4) L2

“This test verifies if the app properly implements identity pinning (certificate or public key pinning).”

The test consists of several checks both related to **XML** and **Java** documents:

- Validate pin expiration date (*network_security_config.xml*).
- Verify the presence of backup pins (*network_security_config.xml*).
- Verify custom <trust-anchors> tags, for example to avoid user certificates (*network_security_config.xml*).
- Identify dangerous snippet possibly causing from test source code.
- Verify the use of strong hashing functions.
- Verify that the SSL connection sets correctly the SSLSocketFactory.
- . . .

mstg-network-4.1.yaml

mstg-network-4.1.xml

mstg-network-4.2.yaml

mstg-network-4.2.java

Testing Custom Certificate Stores and Certificate Pinning

(MSTG-NETWORK-4) L2

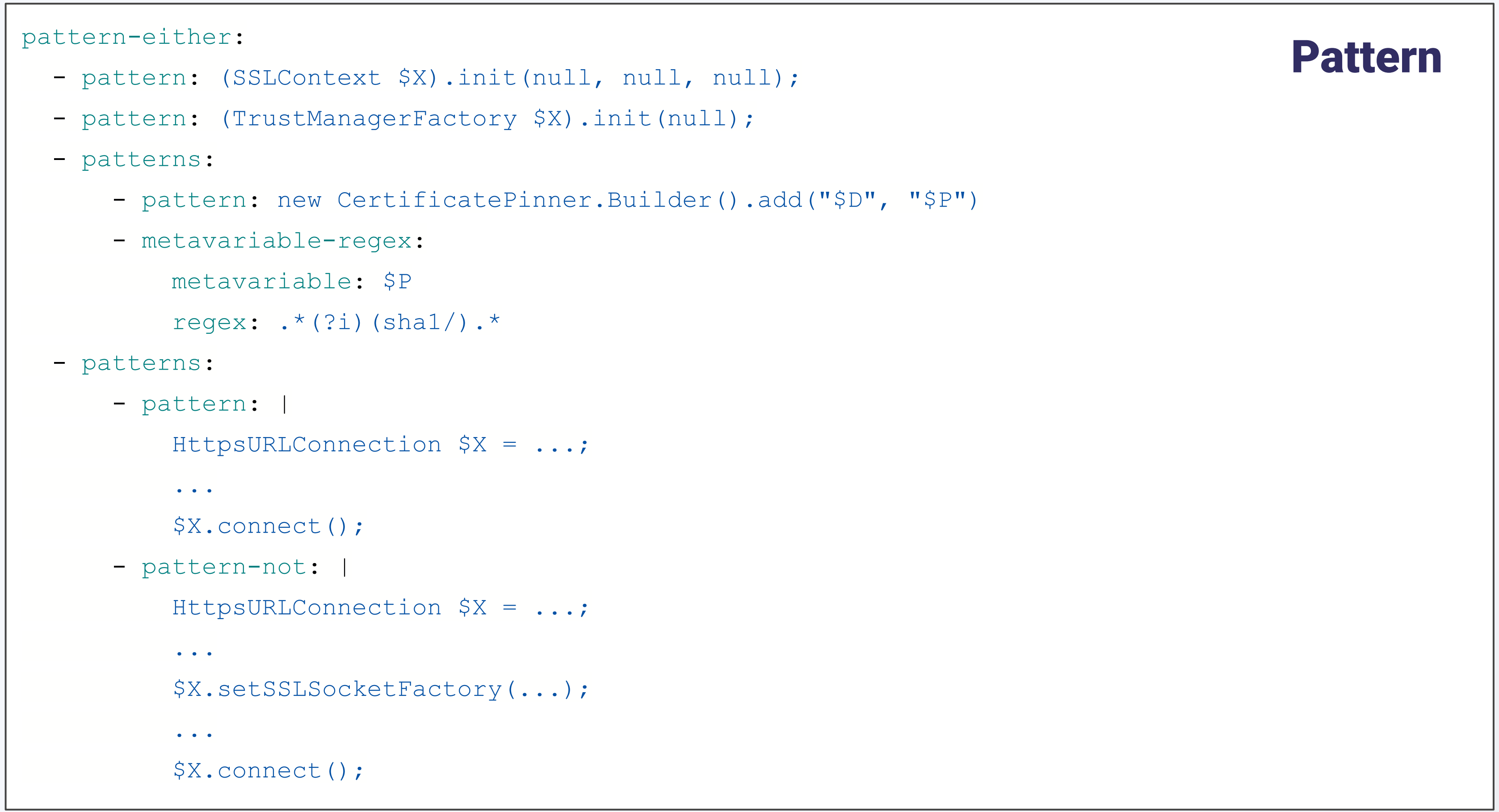
Pattern

```
pattern-either:
  # 1) Pin expiration not present
  - patterns:
      - pattern: <pin-set ... />
      - pattern-not: <pin-set expiration="..." />
  # 2) Pin expired
  - patterns:
      - pattern: <pin-set expiration="$X" />
      - metavariable-comparison:
          comparison: strptime($X) < today()
  # 3) Backup pin not present
  - patterns:
      - pattern: <pin-set ... />
      - pattern-not: <pin-set><pin/><pin/></pin-set>
  # 4) Trust anchors contains user certificates
  - patterns:
      - pattern: <trust-anchors>...<certificates src="user" />...</trust-anchors>
```

- Validate pin expiration date (*network_security_config.xml*).
- Verify the presence of backup pins (*network_security_config.xml*).
- Verify custom <trust-anchors> tags, for example to avoid user certificates (*network_security_config.xml*).

Testing Custom Certificate Stores and Certificate Pinning

(MSTG-NETWORK-4) L2

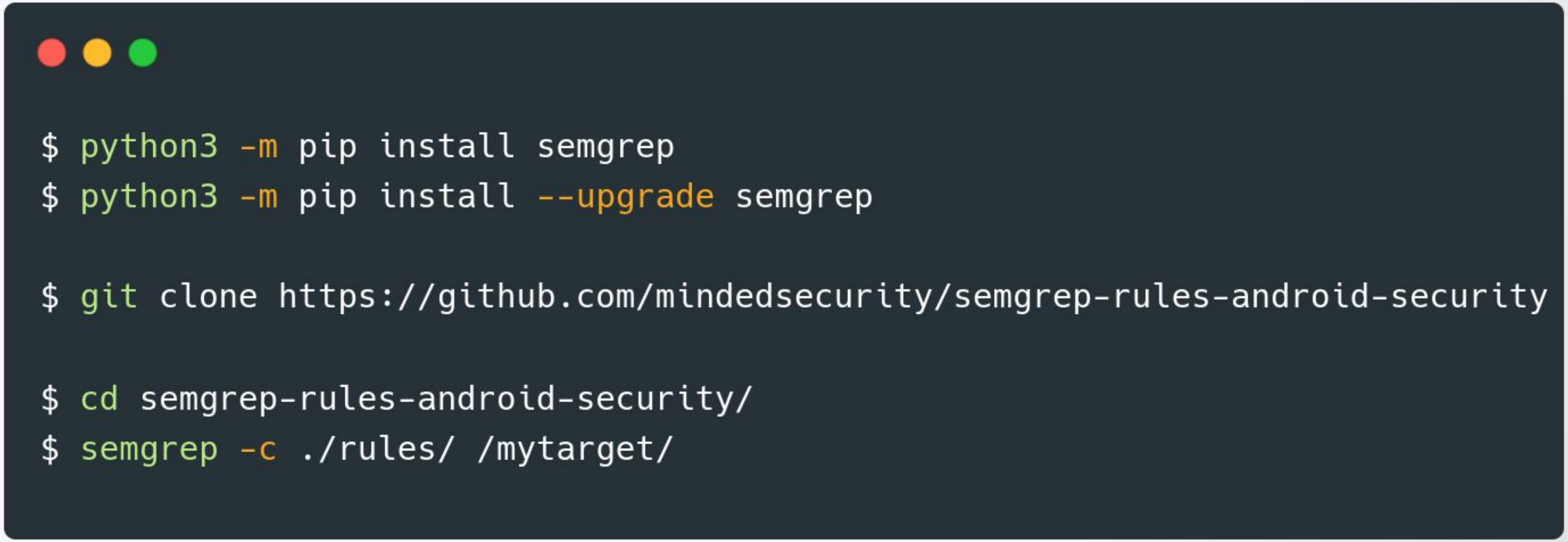


Pattern

```
pattern-either:
  - pattern: (SSLContext $X).init(null, null, null);
  - pattern: (TrustManagerFactory $X).init(null);
  - patterns:
      - pattern: new CertificatePinner.Builder().add("$D", "$P")
      - metavariable-regex:
          metavariable: $P
          regex: .*(?i)(sha1/).*
  - patterns:
      - pattern: |
          HttpsURLConnection $X = ...;
          ...
          $X.connect();
      - pattern-not: |
          HttpsURLConnection $X = ...;
          ...
          $X.setSSLSocketFactory(...);
          ...
          $X.connect();
```

- Identify dangerous snippet possibly causing from test source code.
- Verify the use of strong hashing functions.
- Verify that the SSL connection sets correctly the SSLSocketFactory.

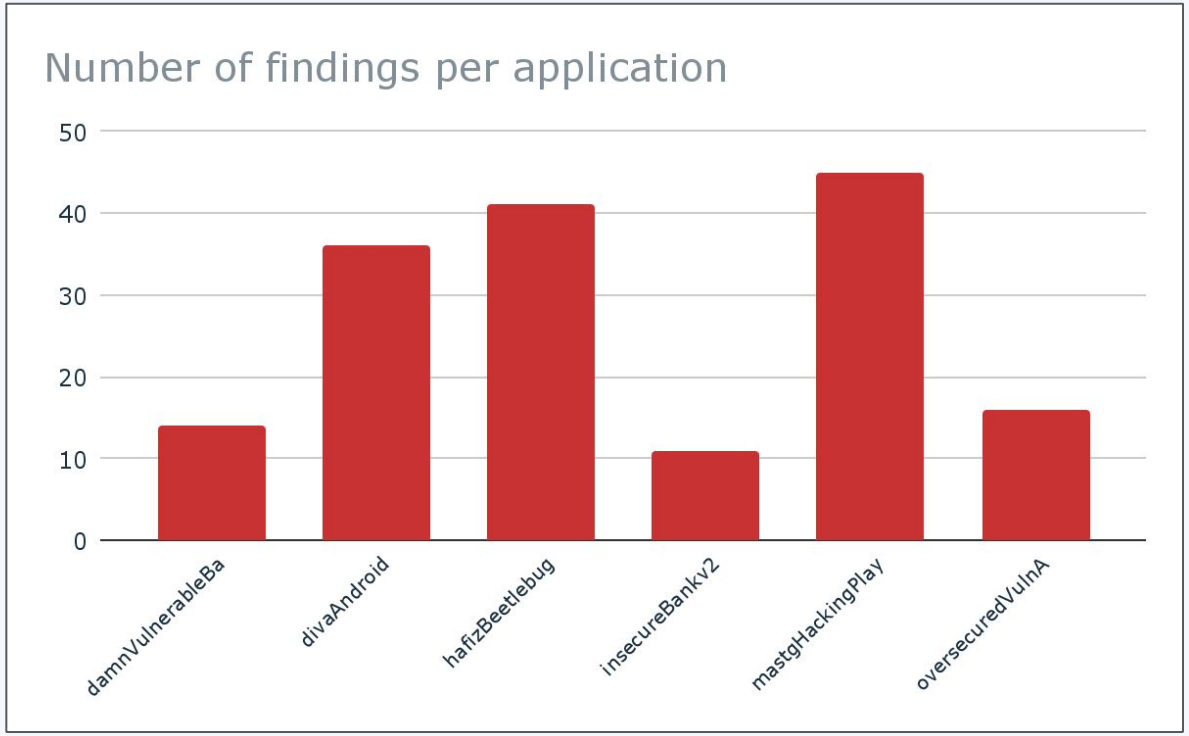
Tutorial



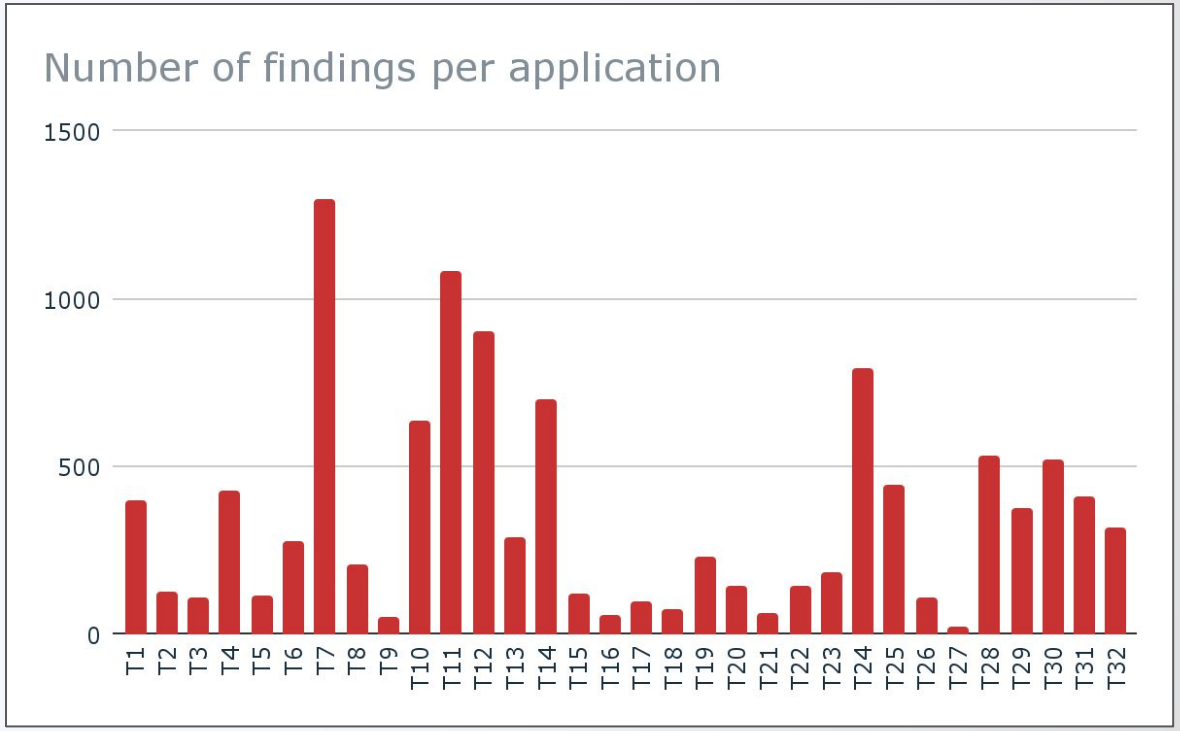
```
$ python3 -m pip install semgrep
$ python3 -m pip install --upgrade semgrep

$ git clone https://github.com/mindedsecurity/semgrep-rules-android-security

$ cd semgrep-rules-android-security/
$ semgrep -c ./rules/ /mytarget/
```

Testing the Rules

Damn vulnerable Android applications



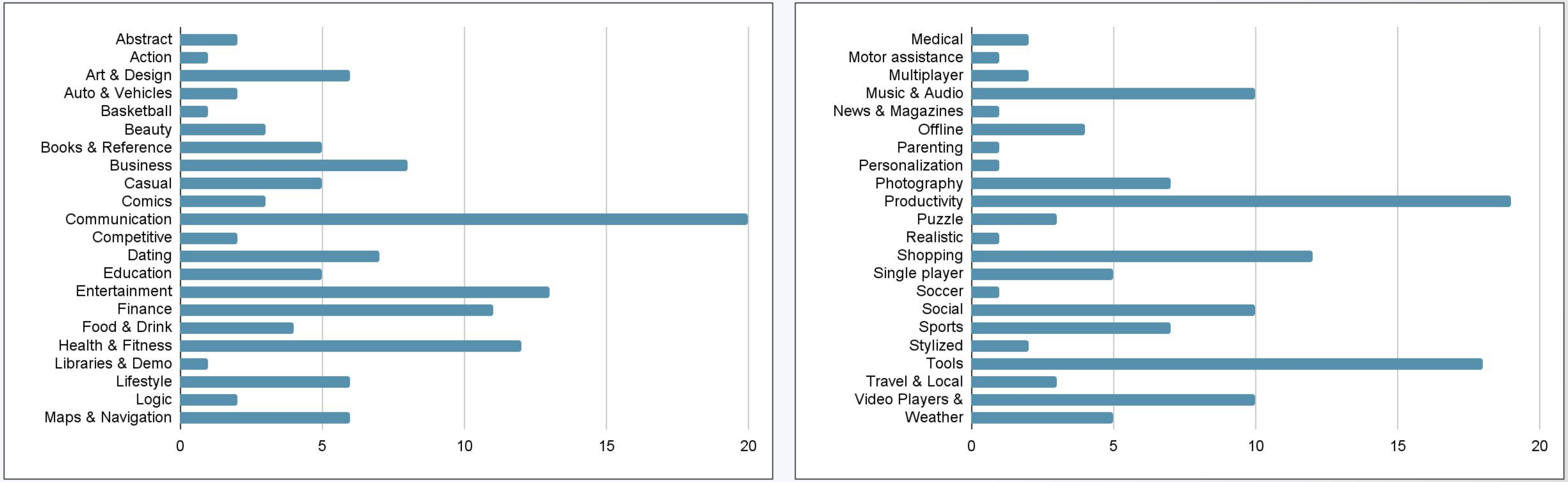
Android application from bug bounty programs



* The findings presented can include False Positives (FP).

280 Android Application

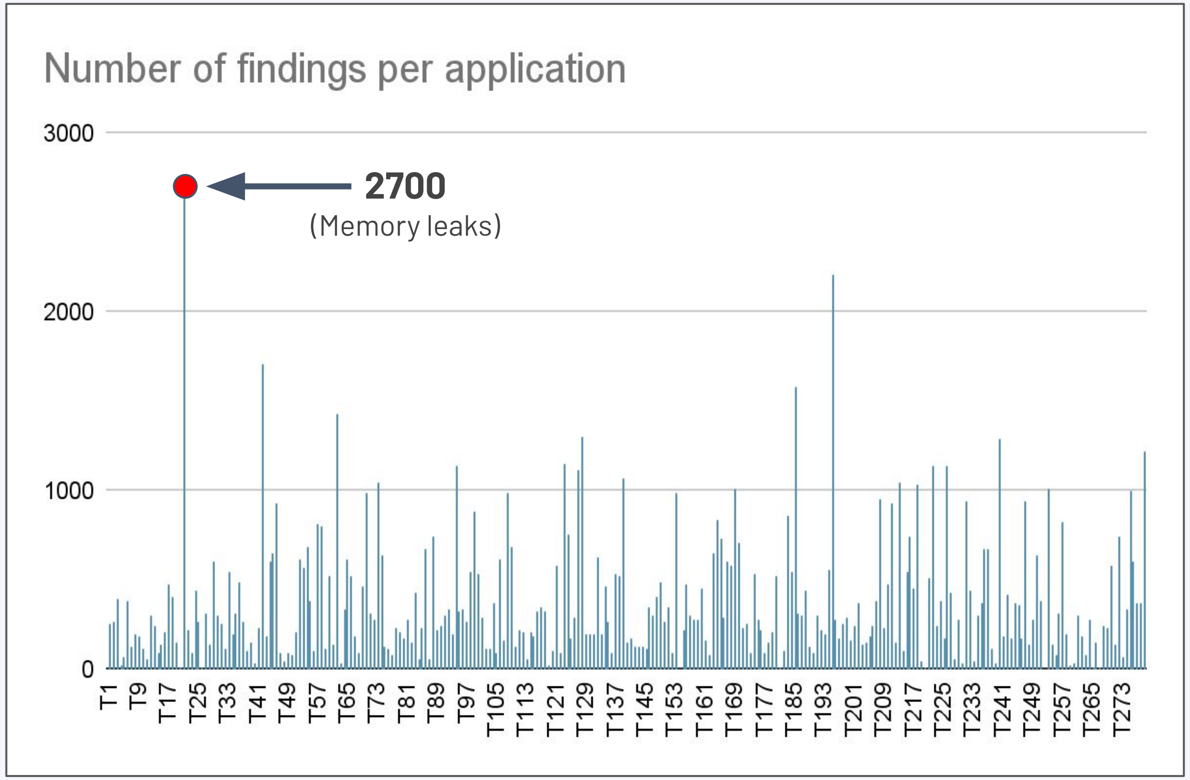
Number of application per category

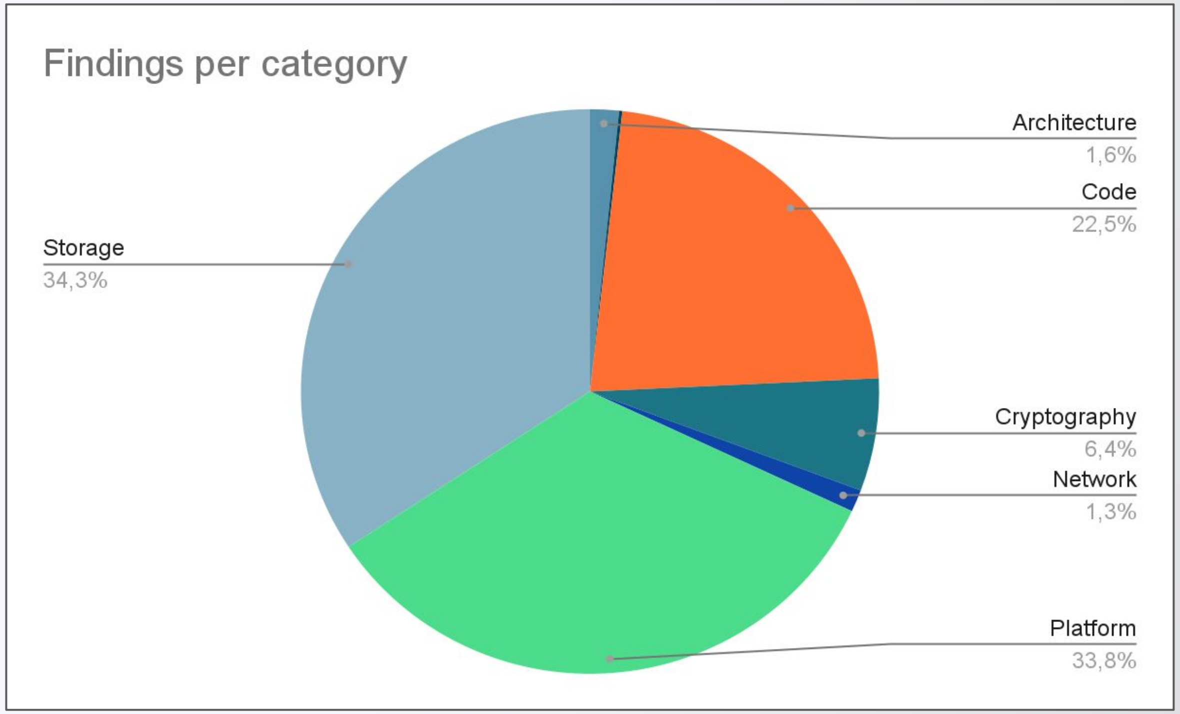


Total number of download: **111˙885˙468˙639**

Stress Testing the Rules

280 Android applications from Google Play Store





Total findings: **107173** **(*)**

Average findings per application: **382**

* The findings presented can include False Positives (FP).

What about the other tests?

The following list reports some examples of implemented tests:

- Check that the application forces **updates** in the main activity
- Check for improper **biometric authentication**
- Check for **debuggable** application
- Check for **unregistered** Broadcast **Receivers**
- Check for **custom cryptographic** primitives
- Check for **deprecated cryptographic** algorithms
- Check for **insecure random** number generators
- Check for improper **SSL pinning** implementation
- Check for unnecessary **permissions**
- Check for improper manifest **permissions** (service, receivers, **provider**)
- Check for **hardcoded credentials**
- Check for **cacheable sensitive input** text
- . . .

Question time.

OWASP®
ITALY

Thanks for your attention!

Thank you to our sponsors

Contrast
SECURITY
Qualys.
SecureFlag
IMQ
group

OWASP 2023

ITALY DAY