



Introducción al análisis estático de código

¿Quien soy?

- Jorge Louzao
- Ingeniero de infraestructuras IT en una empresa que cotiza en el NASDAQ
- Certified Ethical Hacker
- DevSecOps, la primera línea de batalla
- Conferenciante en C1b3rWall y C1b3rwall Academy
- Mentor en la National CyberLeague de la Guardia Civil
- Paranoico a tiempo parcial



Antecedentes

- Security Development Lifecycle define una serie de prácticas para mejorar la seguridad y el cumplimiento de requisitos de las aplicaciones.
- Esto es una práctica cada vez más común en grandes empresas de software que debería aplicarse en empresas de cualquier tamaño, adaptando la metodología los recursos de los que se disponga.
- En las charlas que he dado en los últimos tiempo sobre SDL he descubierto que los alumnos saben de que se trata porque sus profesores se lo han mostrado pero no son conscientes de la importancia real que esto tiene.



Antecedentes

- Ante la escalada de ciberataques para comprometer organizaciones, usuarios y datos SDL debería ser de obligada aplicación en
- Aplicaciones implementadas en un entorno empresarial
- Aplicaciones que procesan información de identificación personal (PII) u otro tipo de información confidencial
- Aplicaciones que se comunican frecuentemente a través de Internet u otras redes



SDL

Existen varios modelos.

- MS Security Development Lifecycle (MS SDL): Uno de los primeros modelos, es el que usan internamente.
- NIST 800-64: Provee al Software Development Lifecycle de conceptos de seguridad. Este estándar ha sido desarrollado por el National Institute of Standards and Technology para las agencias federales de EE.UU.
- OWASP SAMM (Software Assurance Maturity Model): Fácil de implementar e inicialmente basado en MS SDL



SDL

- ¿Como debe ser el software?
- Seguro por diseño: el software debe ser diseñado e implementado para protegerse a si mismo y a la información que procesa.
- Seguro por defecto: la seguridad 100% no existe. El arquitecto y el diseñador del software deben asumir que el software va a tener fallos de seguridad y tomarán las medidas necesarias para minimizar un ataque, por ejemplo ejecutando el software sin más permisos de los necesarios, restringiendo el acceso a servicios solo a los usuarios necesarios.
- Seguro en el despliegue: Utilidades e instrucciones deben acompañar para ayudar al administrador de los servidores a realizar el despliegue con precisión y seguridad.



SDL

No siempre va a ser necesario el uso de una metodología SDL



SDL

- No vamos a ver las áreas en las que se divide la metodología SDL, nos vamos a quedar con algunos puntos que van a suponer la primera línea de defensa:
- Codificación segura
 - Saturaciones de búfer (para aplicaciones que usen C y C++)
 - Errores aritméticos de enteros (para aplicaciones que usen C y C++)
 - XSS (para código administrado y aplicaciones web)
 - Inyección de código SQL (para código administrado y aplicaciones web)
 - Criptografía débil



SDL

- Fase de Implementación:
 - Definición de herramientas aprobadas y comprobaciones de seguridad asociadas, opciones de compilado, versiones, etc.
 - Eliminación de funciones y API no seguras, especialmente en código heredado.
 - Y finalmente el análisis estático del código fuente.



SDL

- Fase de Comprobación
 - Análisis dinámico
 - Fuzzing
 - Revisión de la superficie de ataques
 - Control especial sobre el Código heredado



Análisis Estático de Código

- El análisis estático de código busca
 - Asegurar que se aplican las directivas de codificación segura.
 - Que no se usan funciones inseguras en el código y ofrecer una alternativa segura.
 - Detección de vulnerabilidades.
 - Examinar funciones críticas como, por ejemplo, las criptográficas.
- El análisis estático de código no substituye la revisión manual del software en busca de fallos de programación y vulnerabilidades.



Análisis Estático de Código

- En cuanto a los puntos débiles
 - Exceso de falsos negativos y la dificultad de configurar los analizadores para ocultarlos.
 - Algunos mensajes de error son un poco incomprensibles.
 - Como encajar el analizador en el flujo de trabajo.
- Con tiempo y entrenamiento esto se supera y obtenemos un beneficio de encontrar errores en una fase temprana.



Análisis Estático de Código

- Aunque esta fase está pensada para formar parte del proceso de integración continua disponemos de herramientas para ayudarnos en la fase de desarrollo.
- Existen decenas de herramientas de pago y open source, en esta charla pretendo centrarme en las Open Source, aunque algunas hayan sido desarrolladas por empresas que hasta fechas más recientes no han destacado precisamente por su implicación en el movimiento del Software Libre



Análisis Estático de Código

- Para obtener información en tiempo real durante el Desarrollo de la aplicación tenemos herramientas como DevSkim, un plugin para Visual Studio y su hermano libre Visual Studio Code, también soporta otros editores como Sublime Text
- <https://github.com/Microsoft/DevSkim>
- <https://marketplace.visualstudio.com/items?itemName=MS-CST-E.vscode-devskim>
- <https://github.com/microsoft/DevSkim-Action>



DevSkim

2022-c1b3rwall.cpp 1 ×

Users > louzao > Documents > C1b3rwall 2022 > 2022-c1b3rwall.cpp > main()

```
1  #include <stdio.h>
2
3  int main()
4  {
5      char str [50];
6      gets(str)
7      return 0
```

DevSkim: Change to fgets

DevSkim: Change to gets_s (Recommended for VC++)

DevSkim: Suppress DS181021 for 30 days

DevSkim: Suppress DS181021 permanently

```
char str [50];
fgets(str, <size of str>, stdin)
```



DevSkim en GitHub

louzaonet / c1b3rwall2022 Public

Pin

Unwatch 1

Fork 0

Star 0

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security 1

Insights

Settings

Code scanning alerts / #2

Banned C function detected (gets)

Dismiss alert

Create issue

Open in main 1 minute ago

c1b3rwall.cpp:6

```
3 int main()
4 {
5     char str [50];
6     gets(str)
```

Banned C function detected (gets)

devskim

```
7 return 0
8 }
9
```

Severity

Error

Affected branches

main



Tool	Rule ID
devskim	DS181021

Visit <https://github.com/Microsoft/DevSkim/blob/main/guidance/DS181021.md> for guidance on this issue.



DevSkim en GitHub

- <https://github.com/louzaonet/c1b3rwall2022/blob/main/.github/workflows/devskim.yml>

```
name: DevSkim

on:
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]
  schedule:
    - cron: '42 13 * * 4'

jobs:
  lint:
    name: DevSkim
    runs-on: ubuntu-20.04
    permissions:
      actions: read
      contents: read
      security-events: write
    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Run DevSkim scanner
        uses: microsoft/DevSkim-Action@v1

      - name: Upload DevSkim scan results to GitHub Security tab
        uses: github/codeql-action/upload-sarif@v2
        with:
          sarif_file: devskim-results.sarif
```



DevSkim

- DevSkim funciona mediante expresiones regulares y puede ampliarse para ser usado sobre cualquier lenguaje de programación.
- Soporta C/C++, Java, C#, JavaScript y PHP entre otros lenguajes de programación.
- <https://github.com/Microsoft/DevSkim/wiki/Writing-Rules>



Bandit

- Para Python Bandit es una buena herramienta.
- Permite el uso de perfiles para buscar solo determinados tipos de vulnerabilidades
- No tienen versión para Action en GitHub, en su lugar podemos usar DevSkim, CodeQL... hay 24 analizadores de código con soporte Python disponibles.
- <https://github.com/PyCQA/bandit>



Bandit

```
root@SkullCanyon:/home# bandit -n 3 -lll /usr/share/sqlmap/sqlmap.py
[main] INFO     profile include tests: None
[main] INFO     profile exclude tests: None
[main] INFO     cli include tests: None
[main] INFO     cli exclude tests: None
[main] INFO     running on Python 2.7.17
Run started:2020-04-12 14:02:08.458109

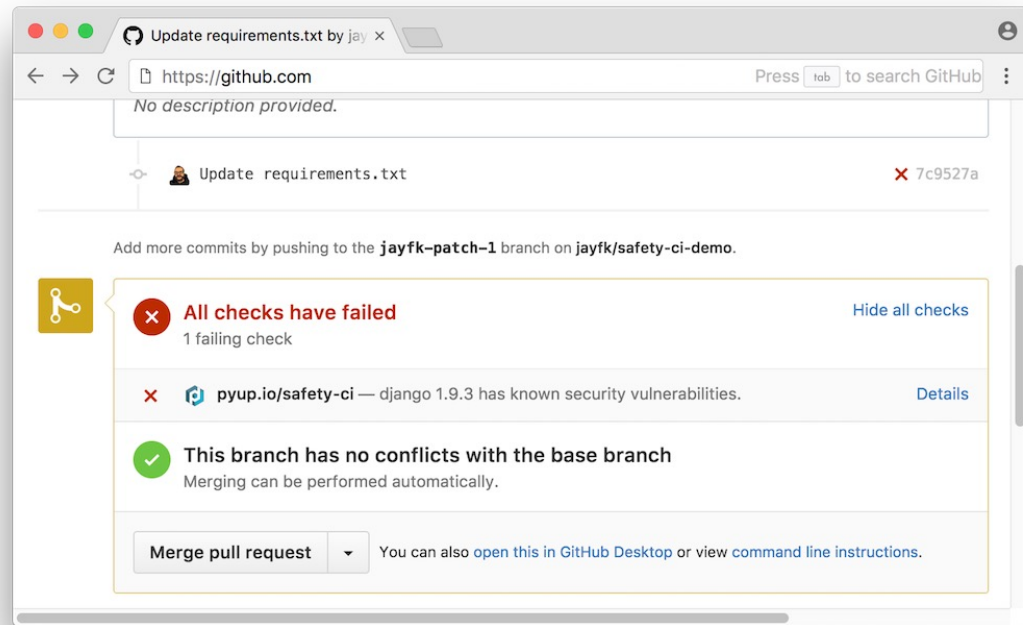
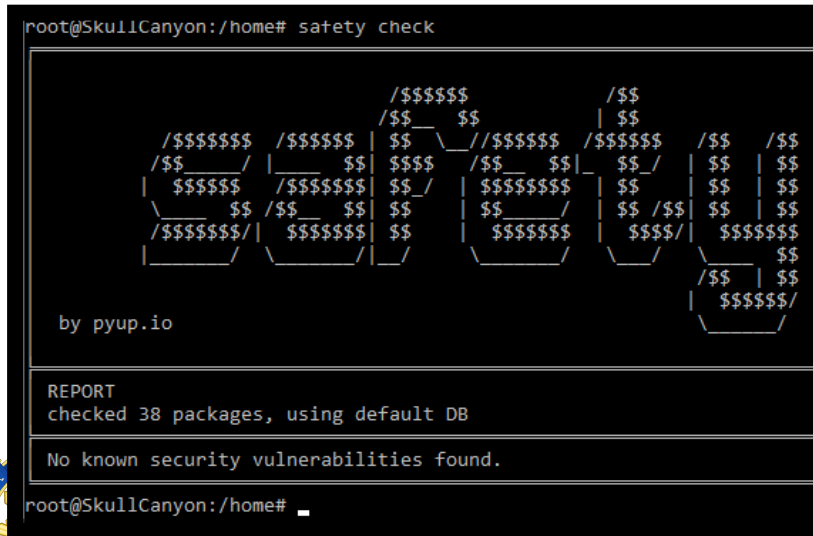
Test results:
    No issues identified.

Code scanned:
    Total lines of code: 430
    Total lines skipped (#nosec): 0

Run metrics:
    Total issues (by severity):
        Undefined: 0
        Low: 0
        Medium: 0
        High: 0
    Total issues (by confidence):
        Undefined: 0
        Low: 0
        Medium: 0
        High: 0
Files skipped (0):
root@SkullCanyon:/home# _
```



-



Javascript

- Para JavaScript tenemos una serie de reglas para NodeSecurity que se integran con Github vía pull requests
- <https://github.com/nodesecurity/eslint-plugin-security>



Javascript

- Como complemento para JavaScript está la herramienta Retire que incluso tiene una versión en plugin para navegadores bastante útil.
- <https://github.com/RetireJS/retire.js>



Retire JS

← → ↻ 🔒 https://c1b3rwall.policia.es

Retire.js ☒ Enabled ☐ Show unknown

jquery	1.11.2	Found in https://c1b3rwall.policia.es/js/jquery.min.js - Vulnerability info: Medium 2432 3rd party CORS request may execute CVE-2015-9251 [1] [2] [3] [4] Medium CVE-2015-9251 11974 parseHTML() executes scripts in event handlers [1] [2] [3] Medium CVE-2019-11358 jQuery before 3.4.0, as used in Drupal, Backdrop CMS, and other products, mishandles jQuery.extend(true, {}, ...) because of Object.prototype pollution [1] [2] [3] Medium CVE-2020-11022 Regex in its jQuery.htmlPrefilter sometimes may introduce XSS [1] Medium CVE-2020-11023 Regex in its jQuery.htmlPrefilter sometimes may introduce XSS [1]
bootstrap	5.0.2	Found in https://c1b3rwall.policia.es/js/bootstrap.min.js

Save

Proyecto C1b3rWall
Proyecto de Capacitación Digital y Ciberseguridad



C++

- Para C/C++ CPPCheck.
- Está disponible para Windows, Linux y Mac a través de brew
- <https://cppcheck.sourceforge.io/>
- Se integra con GitHub a través de:
 - <https://www.codacy.com/>
 - <https://softacheck.com/>



CPP Check

- CPPCheck detecta:
 - Dead pointers
 - Division by zero
 - Integer overflows
 - Invalid bit shift operands
 - Invalid conversions
 - Invalid usage of STL
 - Memory management
 - Null pointer dereferences
 - Out of bounds checking
 - Uninitialized variables
 - Writing const data



FlawFinder

- Otra herramienta para C/C++ es Flawfinder, que además se puede integrar en Github Actions.
- <https://github.com/david-a-wheeler/flawfinder>



FlawFinder

```
root@SkullCanyon:~/nmap# flawfinder ./
Flawfinder version 2.0.11, (C) 2001-2019 David A. Wheeler.
Number of rules (primarily dangerous function names) in C/C++ ruleset: 223
Warning: Skipping directory with initial dot ./git
Examining ./FPEngine.cc
Examining ./FPEngine.h
Examining ./FPModel.cc
Examining ./FPModel.h
Examining ./FingerPrintResults.cc
Examining ./FingerPrintResults.h
Examining ./MACLookup.cc
Examining ./MACLookup.h
Examining ./NewTargets.cc
Examining ./NewTargets.h
Examining ./NmapOps.cc
Examining ./NmapOps.h
Examining ./NmapOutputTable.cc
Examining ./NmapOutputTable.h

ANALYSIS SUMMARY:

Hits = 2801
Lines analyzed = 364368 in approximately 10.31 seconds (35341 lines/second)
Physical Source Lines of Code (SLOC) = 224387
Hits@level = [0] 1033 [1] 680 [2] 1822 [3] 64 [4] 228 [5] 7
Hits@level+ = [0+] 3834 [1+] 2801 [2+] 2121 [3+] 299 [4+] 235 [5+] 7
Hits/KSLOC@level+ = [0+] 17.0866 [1+] 12.4829 [2+] 9.45242 [3+] 1.33252 [4+] 1.0473 [5+] 0.0311961
Dot directories skipped = 1 (--followdotdir overrides)
Minimum risk level = 1
Not every hit is necessarily a security vulnerability.
There may be other security vulnerabilities; review your code!
See 'Secure Programming HOWTO'
(https://dwheeler.com/secure-programs) for more information.
root@SkullCanyon:~/nmap#
```



FlawFinder

FINAL RESULTS:

```
./libpcap/lbl/os-sunos4.h:129: [5] (race) readlink:
This accepts filename arguments; if an attacker can move those files or
change the link content, a race condition results. Also, it does not
terminate with ASCII NUL. (CWE-362, CWE-20). Reconsider approach.
./libpcap/pcap-linux.c:607: [5] (race) readlink:
This accepts filename arguments; if an attacker can move those files or
change the link content, a race condition results. Also, it does not
terminate with ASCII NUL. (CWE-362, CWE-20). Reconsider approach.
./libz/contrib/untgz/untgz.c:32: [5] (race) chmod:
This accepts filename arguments; if an attacker can move those files, a
race condition results. (CWE-362). Use fchmod( ) instead.
./libz/contrib/untgz/untgz.c:277: [5] (race) chmod:
This accepts filename arguments; if an attacker can move those files, a
race condition results. (CWE-362). Use fchmod( ) instead.
./nbase/nbase_misc.c:829: [5] (race) readlink:
This accepts filename arguments; if an attacker can move those files or
change the link content, a race condition results. Also, it does not
terminate with ASCII NUL. (CWE-362, CWE-20). Reconsider approach.
./nping/ProbeMode.cc:1752: [5] (buffer) strncpy:
Easily used incorrectly (e.g., incorrectly computing the correct maximum
size to add) [MS-banned] (CWE-120). Consider strcat_s, strlcat, snprintf,
or automatically resizing strings. Risk is high; the length parameter
appears to be a constant, instead of computing the number of characters
left.
./nping/ProbeMode.cc:1780: [5] (buffer) strncpy:
Easily used incorrectly (e.g., incorrectly computing the correct maximum
size to add) [MS-banned] (CWE-120). Consider strcat_s, strlcat, snprintf,
or automatically resizing strings. Risk is high; the length parameter
appears to be a constant, instead of computing the number of characters
left.
```

ANALYSIS SUMMARY:

```
Hits = 7
Lines analyzed = 364368 in approximately 6.12 seconds (59580 lines/second)
Physical Source Lines of Code (SLOC) = 224387
Hits@level = [0] 1033 [1] 680 [2] 1822 [3] 64 [4] 228 [5] 7
Hits@level+ = [0+] 3834 [1+] 2801 [2+] 2121 [3+] 299 [4+] 235 [5+] 7
Hits/KSLOC@level+ = [0+] 17.0866 [1+] 12.4829 [2+] 9.45242 [3+] 1.33252 [4+] 1.0473 [5+] 0.0311961
Dot directories skipped = 1 (--followdotdir overrides)
Minimum risk level = 5
Not every hit is necessarily a security vulnerability.
There may be other security vulnerabilities; review your code!
See 'Secure Programming HOWTO'
(https://dwheeler.com/secure-programs) for more information.
```



Java

- Para Java tenemos SonarSource Community y Find Security Bugs
- <https://www.sonarsource.com/plans-and-pricing/community/>
- <https://find-sec-bugs.github.io/>



Java

- SonarSource Community soporta más lenguajes de programación, como PHP, C#, Go, Flex, Ruby, JavaScript, VB.NET entre otros.
- Para CI funciona con Jenkins, Bamboo y Azure DevOps.
- Se integra con IntelliJ, Visual Studio Code, Eclipse y Visual Studio.
- Find Security Bugs está especializado en auditar aplicaciones web desarrolladas en Java.
- Se integra con Jenkins y SonarQube, además de los IDE Eclipse, IntelliJ, NetBeans y Android Studio
- Integración como tarea en Ant y Maven
- Cubre el OWASP Top 10 y CWE de Mitre

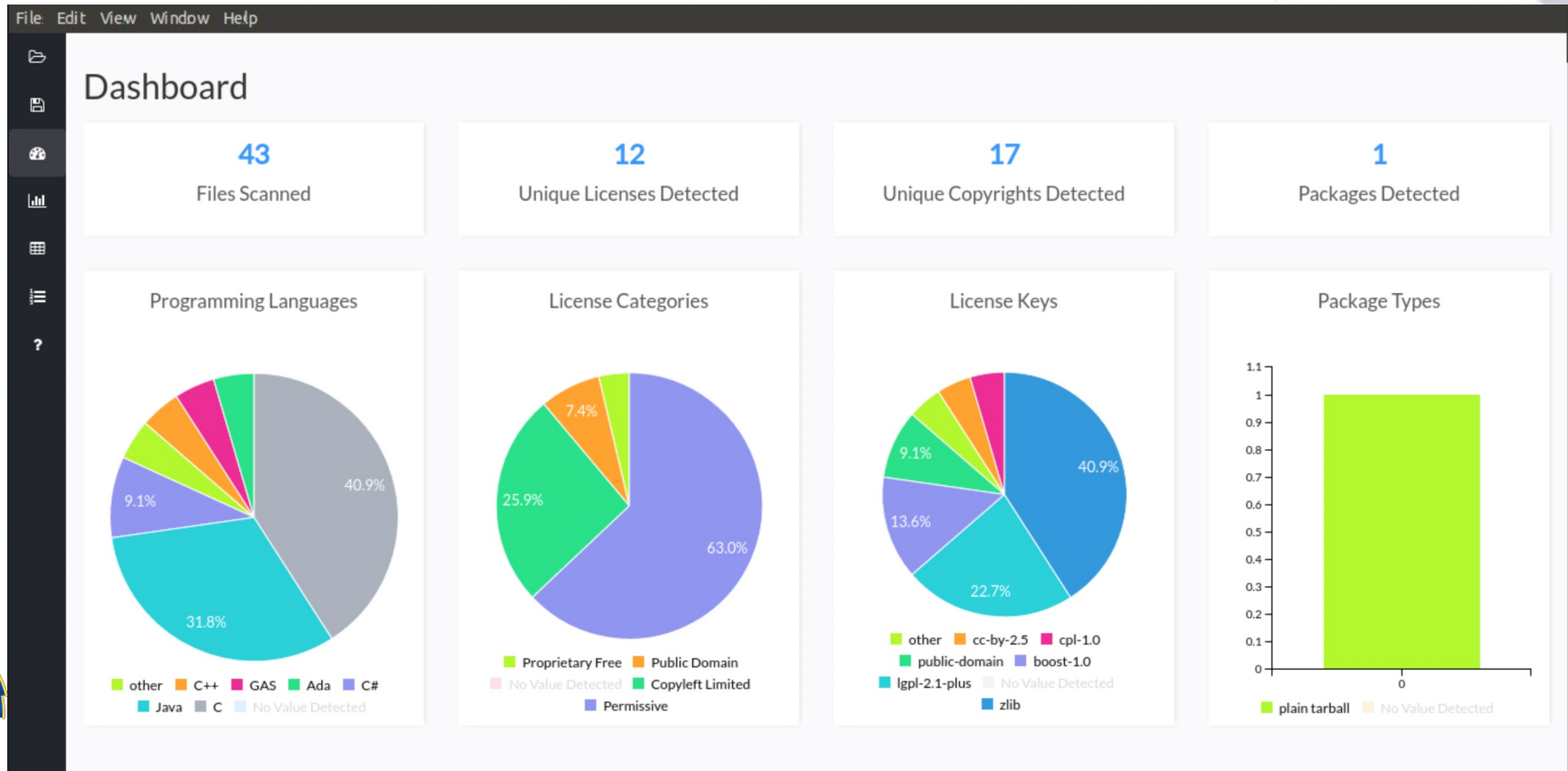


Otros Analizadores de Código

- Otro tipo de analizador, el que busca copyrights y licencias de las librerías que empleamos en nuestro software.
- <https://github.com/nexB/scancode-toolkit/>



Scancode Toolkit



Otros Analizadores de Código

- TFSec es un analizador especializado en plantillas de Terraform y una herramienta muy útil antes de un despliegue
- Encuentra información sensible
- Violaciones de las recomendaciones de mejores prácticas de AWS, Azure y Google Cloud
- Disponible en GitHub Actions <https://github.com/marketplace/actions/run-tfsec-pr-commenter>

<https://github.com/liamg/tfsec>



Otros Analizadores de Código

```
tfsec ./example
```

4 potential problems detected:

Problem 1

[AWS006] Resource 'aws_security_group_rule.my-rule' defines a fully open ingress security group rule.
/Users/liamg/example/main.tf:4

```
1 |  
2 | resource "aws_security_group_rule" "my-rule" {  
3 |     type      = "ingress"  
4 |     cidr_blocks = ["0.0.0.0/0"]  
5 | }  
6 |  
7 | resource "aws_alb_listener" "my-alb-listener"{
```

Problem 2

[AWS004] Resource 'aws_alb_listener.my-alb-listener' uses plain HTTP instead of HTTPS.
/Users/liamg/example/main.tf:9

```
6 |  
7 | resource "aws_alb_listener" "my-alb-listener"{  
8 |     port      = "80"  
9 |     protocol = "HTTP"  
10 | }  
11 |  
12 | resource "aws_db_security_group" "my-group" {
```

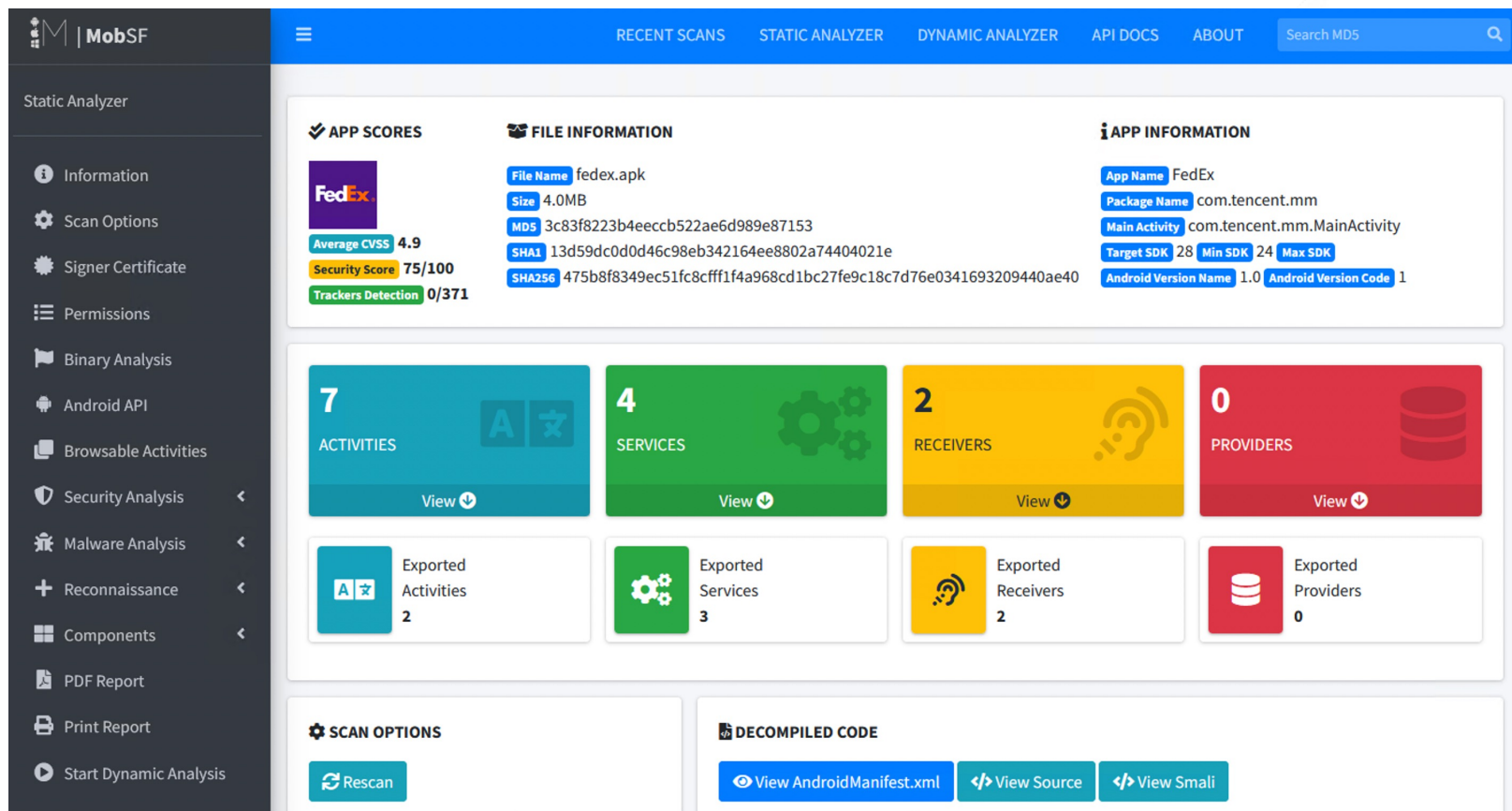


Otros Analizadores de Código

- Para aplicaciones móviles tenemos MoBSF, que hace un completo análisis estático y con un poco más de configuración puede llegar a realizar análisis dinámico.
- Está disponible en GitHub Actions
- <https://github.com/MobSF/Mobile-Security-Framework-MobSF>
- <https://github.com/marketplace/actions/github-action-for-mobsf>



Otros Analizadores de Código



The screenshot displays the MobSF Static Analyzer interface. The left sidebar contains navigation options: Information, Scan Options, Signer Certificate, Permissions, Binary Analysis, Android API, Browsable Activities, Security Analysis, Malware Analysis, Reconnaissance, Components, PDF Report, Print Report, and Start Dynamic Analysis. The main content area is divided into several sections:

- APP SCORES:** Shows the FedEx app logo, Average CVSS 4.9, Security Score 75/100, and Trackers Detection 0/371.
- FILE INFORMATION:** Lists File Name (fedex.apk), Size (4.0MB), MD5 (3c83f8223b4eccc522ae6d989e87153), SHA1 (13d59dc0d0d46c98eb342164ee8802a74404021e), and SHA256 (475b8f8349ec51fc8cff1f4a968cd1bc27fe9c18c7d76e0341693209440ae40).
- APP INFORMATION:** Lists App Name (FedEx), Package Name (com.tencent.mm), Main Activity (com.tencent.mm.MainActivity), Target SDK (28), Min SDK (24), Max SDK, Android Version Name (1.0), and Android Version Code (1).
- ACTIVITIES:** Shows 7 activities with a 'View' button and 'Exported Activities' (2).
- SERVICES:** Shows 4 services with a 'View' button and 'Exported Services' (3).
- RECEIVERS:** Shows 2 receivers with a 'View' button and 'Exported Receivers' (2).
- PROVIDERS:** Shows 0 providers with a 'View' button and 'Exported Providers' (0).
- SCAN OPTIONS:** Includes a 'Rescan' button.
- DECOMPILED CODE:** Includes buttons for 'View AndroidManifest.xml', 'View Source', and 'View Smali'.



Análisis Estático de Código

FIN
iii MUCHAS GRACIAS!!!

Twitter @louzaonet
Email: jorge@louzao.net

