

Exploring Android Security: Penetration Testing, Vulnerabilities, and Exploitation Techniques

Rahul Meena(230832)
Chaitanya Bramhapurikar (230305)

Indian Institute of Technology Kanpur

April 16, 2025

Course: Undergraduate Project(CS496)

Professor: Prof. Subhajit Roy

Android Platform Overview and it's Security Features



Android Platform Overview and it's Security Features

- Linux Based, it uses a modified version of Linux kernel.
- Process Seperation, sandboxing: each process has a unique user id, storage and memory.
- Permission System - runtime and install-time permissions
- App Signing - All apps must be cryptographically signed
- Verified Boot - Cryptographic verification of OS integrity during startup
- Hardware-backed Keystore - Separate Hardware for cryptographic operations
- TLS by Default - Internet Communication only through TLS protocol
- ASLR, KASLR, PIE and DEP : standard protections against buffer-overflow exploits.
- Secure IPC Mechanisms - Framework for controlled inter-process/app communication

Testing Part 1: Android Apps and their attack surfaces

- Android uses sandboxing to isolate apps using separate UIDs and data directories, preventing unauthorized access by default.
- But apps often need to communicate internally and with outside. For this android provides secure ways and interfaces for IPC. This opens many attack surfaces.
- Weak security measure, bad code and other such factors opens these surfaces to exploits and attacks.
- **Identifying these attack surfaces is the first step in testing**
- To spread awareness about these potential risks, the Open Web Application Security Project(OWASP), a nonprofit foundation, releases it's widely recognized list of the 10 most critical mobile application security risks.

Owasp Top 10 Mobile security risks 2024

- Improper Credential Usage
- Inadequate Supply Chain Security
- Insecure Authentication/Authorization
- Insufficient Input/Output Validation
- Insecure Communication
- Inadequate Privacy Controls
- Insufficient Binary Protections
- Security Misconfiguration
- Insecure Data Storage
- Insufficient Cryptography

Common Android App Attack Surfaces

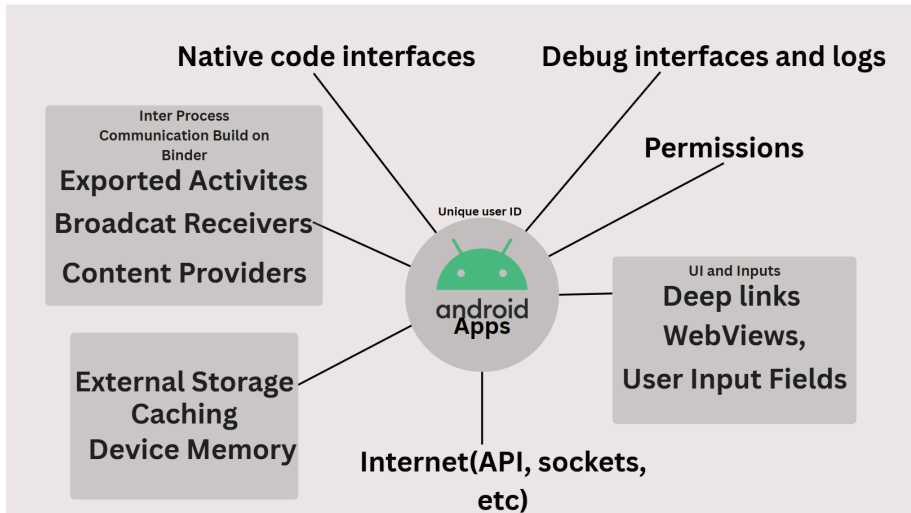


Figure: common android attack surfaces

Common Android Platform/OS Attack Surfaces

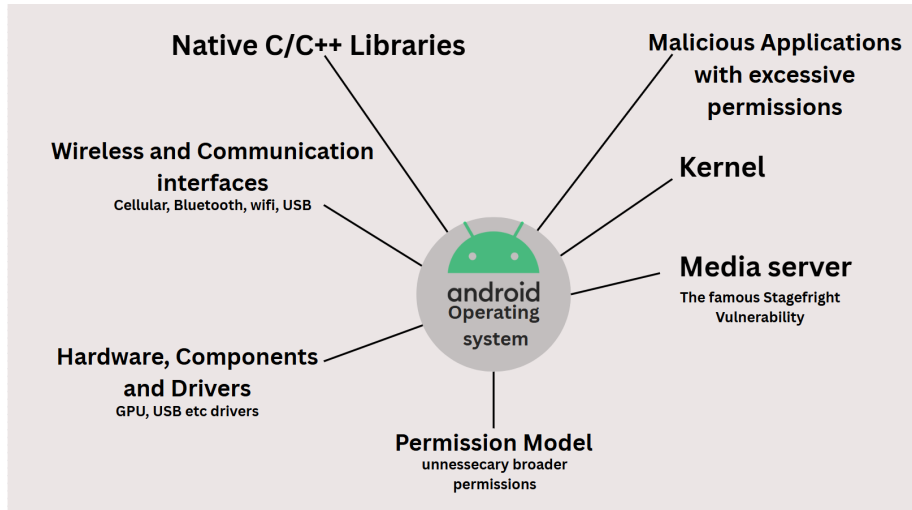


Figure: common android os attack surfaces

How to reveal more attack surfaces?

Techniques like Fuzzing : Feeding varied and often random data inputs like wifi packets, corrupted media files into an application's input interfaces, observer crashes and logs. Crashes often suggest a presence of potentially bad code or a condition the code did not take account of.

Testing Part 2: Tools and Techniques, Static Analysis

- Review the source code or binary without executing it.
- Ensure appropriate implementation of security controls
- Automatic scans and tools can catch basic insecure practices like Hardcoded secrets, insecure cryptography, exported components, WebView vulnerabilities, Outdated Libraries with known CVEs
- the human tester can explore the code base with specific usage contexts in mind
- when the code is technically secure but logically flawed
- Tools : MobSF, APKTool, Jadx, AndroBugs, Radare2(for native binary analysis)

Static Analysis : Java/Kotlin Code Decompilation using jadx (Un-obfuscated code)

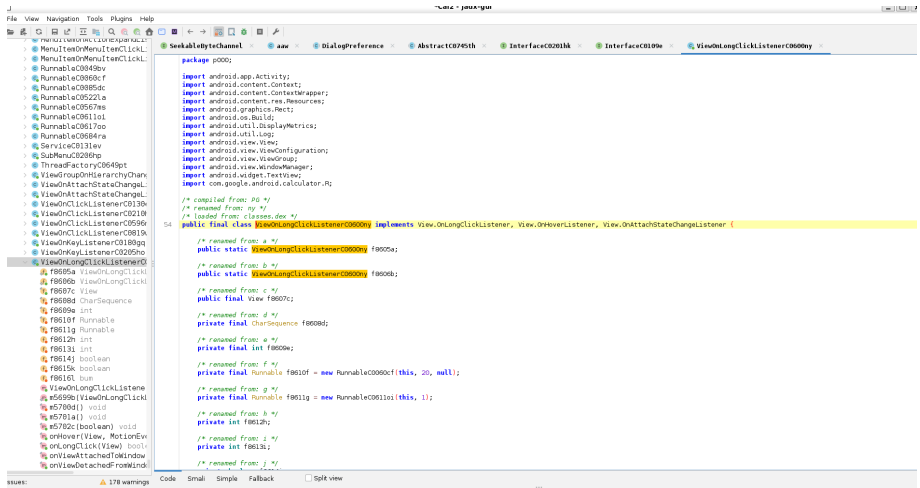


Figure: Using Jadx to decompile .dex files into java

Static Analysis: Native C/C++ Code Disassembly using radare2

Android provides Java Native Interface (JNI), which defines a way for Java code to interact with native code written in C/C++. As on other Linux-based operating systems, native code is packaged (compiled) into ELF dynamic libraries (*.so), which the Android app loads at runtime. Mishandling of these interfaces introduces traditional C/C++ vulnerabilities like Memory Corruption.

```
→ FRIDA r2 demo
WARN: Relocs has not been applied. Please use `-e bin.relocs.apply=true` or `-e bin.cache=true` next time
[0x00001080]> aaa
INFO: Analyze all flags starting with sym. and entry0 (aa)
INFO: Analyze imports (af@@@i)
INFO: Analyze entrypoint (af@ entry0)
INFO: Analyze symbols (af@@@s)
INFO: Analyze all functions arguments/locals (afva@@@F)
INFO: Analyze function calls (aac)
INFO: Analyze len bytes of instructions for references (aar)
INFO: Finding and parsing C++ vtables (avrr)
INFO: Analyzing methods (af @@ method.*)
INFO: Recovering local variables (afva@@@F)
INFO: Type matching analysis for all functions (aaft)
INFO: Propagate noreturn information (aanr)
INFO: Use -AA or aaaa to perform additional experimental analysis
[0x00001080]> afl
0x00001030  1      6 sym.imp.puts
0x00001030  1      6 sym.imp.puts
```

Testing Part 3: Tools and Techniques, Dynamic Analysis

- A typical android Application can have thousands of classes and methods. Looking through all of them through can be hard. Dynamic Analysis helps us narrow our search.
- Running the application in a test environment, observe behaviors under a variety of conditions. This can help identify, logic flaws, and unintended interactions, how an app interacts with inputs, systems, and networks while it's running.
- Trace function calls, change their behaviour and inputs at runtime. Can be used to bypass security checks like root checks, SSL Pinning, Login and Premium checks.
- Also helpful when dealing with obfuscated code, where it is hard to understand the logic of the code.

Dynamic Analysis: Objection

Demo : SSL Unpinning. SSL Pinning ensures an app only trusts a pre-defined server certificate (or its public key/hash) embedded within the app. Instead of relying solely on the device's trust store (system-level certificates), the app compares the server's certificate to the pinned value during handshakes. This features prevents anyone from intercepting the communication. A big problem while testing network realated vulnerabilities. Objection(a tool built on frida) can bypass this.

```
(new) PS D:\SEMESTER_4\TERMINAL\new\Scripts> objection -g "tech.httptoolkit.pinning_demo" explore
Using USB device 'Android Emulator 5554'
Agent injected and responds ok!
```

```

[...](object)inject(ion) v1.11.8

```

Runtime Mobile Exploration
by: @leoniza from @sensepost

```

[tab] for command suggestions
tech.httpoolkit.pinning_demo on (google: 15) [usb] # android sslpinning disable
(agent) Custom TrustManager ready, overriding SSLContext.init()
(agent) Found okhttp3.CertificatePinner, overriding CertificatePinner.check()
(agent) Found okhttp3.CertificatePinner, overriding CertificatePinner.check$okhttp()
(agent) Found com.android.org.conscrypt.TrustManagerImpl, overriding TrustManagerImpl.verifyChain()
(agent) Found com.android.org.conscrypt.TrustManagerImpl, overriding TrustManagerImpl.checkTrustedRecursive()
(agent) Registering io.591211.Type: android.ssl.pinning.disable
tech.httpoolkit.pinning_demo on (google: 15) [usb] # (agent) [591211] Called SSLContext.init(), overriding TrustManager with empty one.
(agent) [591211] Called SSLContext.init(), overriding TrustManager with empty one.
(agent) [591211] Called (Android 7+) TrustManagerImpl.checkTrustedRecursive(), not throwing an exception.
(agent) [591211] Called SSLContext.init(), overriding TrustManager with empty one.
tech.httpoolkit.pinning_demo on (google: 15) [usb] #

```

Figure: Objection with frida gadget on non rooted device

Dynamic Analysis: Tracing function calls in Instagram

The screenshot displays a web browser window at `localhost:64506`. The left sidebar shows a file explorer with a tree view of Android system classes, including `android.widget.Spinner`, `android.service.quicksettings.IQTileService`, `android.preference.DialogPreference`, `android.service.quicksettings.TileService$2`, `android.text.style.AccessibilityURLSpan`, `android.preference.PreferenceScreen`, `com.android.internal.view.menu.ActionMenuItemView`, `android.service.quicksettings.TileService`, `android.widget.AutoCompleteTextView$PassThroughClickListener`, `android.view.textclassifier.TextLinks$TextLinkSpan`, `com.android.internal.telephony.uicc.UiccSlot$1`, `com.android.internal.telephony.SMSDispatcher$ConfirmDialogListener`, `com.tct.calculator.activity.CalculatorActivity`, and `com.tct.calculator.view.CalculatorScienceKeyboard`. The `CalculatorScienceKeyboard.onClick` method is selected.

The main content area shows a Frida script being deployed. The script is a JavaScript function that intercepts the `onClick` event of the `CalculatorScienceKeyboard` button. It logs the button's details and the original value, then updates the button text to the negated value. The script is as follows:

```
10 // },
11 // },
12 // },
13 onEnter(log, args, state) {
14   Java.perform(() => {
15     const Button = Java.use('android.widget.Button');
16     const view = Java.cast(args[0], Button);
17     const originalValue = view.getText().toString();
18
19     // Print original value with button details
20     log('[+] Clicked button: ${view} | Value: ${originalValue}');
21
22     // Check if it's a numeric button (customize this regex as ne
23     if (/^d+$/, test(originalValue)) {
24       // Negate the value
25       const negatedValue = originalValue.startsWith('-') ?
26         originalValue.substring(1) :
27         `-${originalValue}`;
28
29       // Update the button text on UI thread
30       Java.scheduleOnMainThread(() => {
31         view.setText(negatedValue);
32       });
33     }
34   });
35 }
```

The bottom of the browser window shows a list of events. The events are as follows:

```
Events Disassembly Memory
/* TID 0x19ba */
[+] onClick triggered for button: android.widget.Button[7fa7f61 VFED..C...P.... 270,0-540,233 #7f090065 app:id/digit_5_aid=1073741847]
+9.1s [+] onClick triggered for button: android.widget.Button[cac29e5 VFED..C...P.... 270,0-540,233 #7f090068 app:id/digit_8_aid=1073741843]
+1.3s [+] onClick triggered for button: android.widget.Button[cac29e5 VFED..C...P.... 270,0-540,233 #7f090068 app:id/digit_8_aid=1073741843]
```

The background of the browser window shows the Instagram login page. The page has a header with the Instagram logo and the text "English (US)". Below the header is a form with the following fields:

- Enter your mobile number, username or email (input type: text)
- Enter your password (input type: password)
- Log in (button)
- Forgot password? (link)
- Log in with Facebook (link)
- OR
- Create new account (link)

Figure: Frida Tracing Web UI

Attack Vectors and Exploitation

- Exploits are the specific actions taken to take advantage of vulnerabilities identified through those vectors.
- Vulnerabilities are harmless unless someone tries to exploit them, attackers use many pathways to gain access to the system to exploit these vulnerabilities, these pathways are called attack vectors.
- Vulnerabilities in itself are harmless unless an attacker tries to exploit them. Common Attack vectors are
 - Malicious Applications
 - Network-based Attacks(MITM, open wifis)
 - Phishing Attacks(fake emails, sms, login pages)
 - Native Code Vulnerabilities by Malformed input to native methods or media files.

Stagefright: A Short Case Study

- Arguably the most severe known Vulnerability discovered in android till date.
- Found by Zimperium zLabs (2015) in Android's libstagefright (media engine)
- Buffer overflow in MP3/MP4 parsing let to Remote Code Execution in Android 5/5.1.
- The media processor would process these files automatically without user interaction
- Zero-click attack possible—no user interaction needed, exploited just by sending a carefully crafted MMS or MP4 file.
- Lessons:
 - Memory safety is critical(C/C++ risks)
 - Automatic parsing and Weak input validations = high risk.
 - Avoid Unnecessary Permissions(the media enginer was running as a very elevated process)

Communication Between Computer and Network

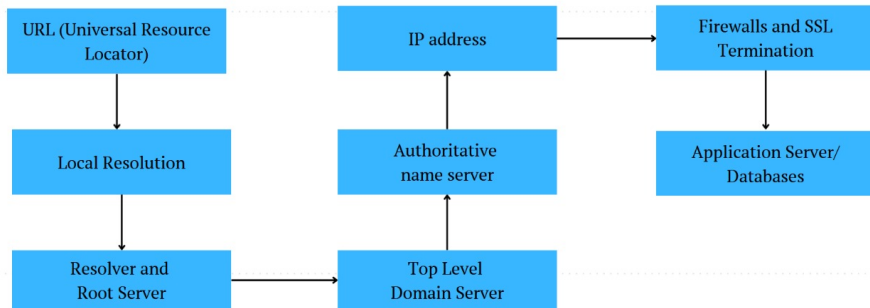


Figure: network flowchart

Examples of common vulnerabilities in network and tools to exploit them

- Weak Authentication

- john the ripper
- hydra
- aircrack-ng
- metasploit framework
- Burpsuite

- Misconfigured Firewalls

- wafw00f
- nmap
- metasploit framework

- Open Ports and Services

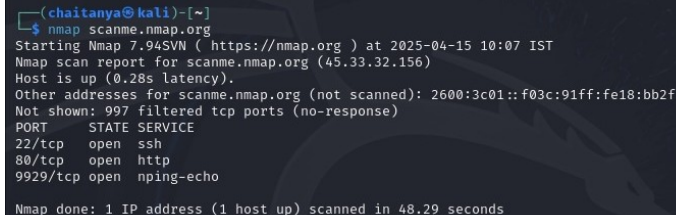
- nmap
- metasploit framework
- masscan
- nikto

Examples of common vulnerabilities in network and tools to exploit them

- Weak encryption
 - hashcat
 - john the ripper
 - wireshark
- Phishing and Social Engineering
 - Zphisher
 - metasploit

Nmap is short for Network Mapper. It is an open-source Linux command-line tool that is used to scan IP addresses and ports in a network.

- Basic Scan:

A terminal window with a dark background and light-colored text. The prompt is '(chaitanya@kali)-[~]'. The command 'nmap scanme.nmap.org' has been executed. The output shows the Nmap version (7.94SVN), the scan time (2025-04-15 10:07 IST), the target IP (45.33.32.156), and that the host is up with a latency of 0.28s. It also lists other unscanned addresses and filtered ports. A table of open ports is shown: 22/tcp (ssh), 80/tcp (http), and 9929/tcp (nping-echo). The scan completed in 48.29 seconds.

```
(chaitanya@kali)-[~]  
$ nmap scanme.nmap.org  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-15 10:07 IST  
Nmap scan report for scanme.nmap.org (45.33.32.156)  
Host is up (0.28s latency).  
Other addresses for scanme.nmap.org (not scanned): 2600:3c01::f03c:91ff:fe18:bb2f  
Not shown: 997 filtered tcp ports (no-response)  
PORT      STATE SERVICE  
22/tcp    open  ssh  
80/tcp    open  http  
9929/tcp  open  nping-echo  
  
Nmap done: 1 IP address (1 host up) scanned in 48.29 seconds
```

Figure: nmap basic scan

• Stealth Scan:

- Stealth scanning is performed by sending an SYN packet and analyzing the response. If SYN/ACK is received, it means the port is open, and you can open a TCP connection.
- However, a stealth scan never completes the 3-way handshake, which makes it hard for the target to determine the scanning system.

```
(chaitanya@kali)-[~]  
$ sudo nmap -sS scanme.nmap.org  
[sudo] password for chaitanya:  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-15 10:17 IST  
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0  
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0  
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0  
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0  
Nmap scan report for scanme.nmap.org (45.33.32.156)  
Host is up (3.4s latency).  
Other addresses for scanme.nmap.org (not scanned): 2600:3c01::f03c:91ff:fe18:bb2f  
Not shown: 993 closed tcp ports (reset)  
PORT      STATE      SERVICE  
22/tcp    open       ssh  
80/tcp    open       http  
445/tcp    filtered   microsoft-ds  
514/tcp    filtered   shell  
8008/tcp   open       http  
9929/tcp   open       nping-echo  
31337/tcp  open       Elite  
  
Nmap done: 1 IP address (1 host up) scanned in 814.22 seconds
```

Figure: nmap stealth scan

• Version Scan:

```
(chaitanya@kali)-[~]  
$ nmap -sV scanme.nmap.org  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-15 10:23 IST  
Nmap scan report for scanme.nmap.org (45.33.32.156)  
Host is up (0.29s latency).  
Other addresses for scanme.nmap.org (not scanned): 2600:3c01::f03c:91ff:fe18:bb2f  
Not shown: 998 filtered tcp ports (no-response)  
PORT      STATE SERVICE VERSION  
22/tcp    open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)  
80/tcp    open  http      Apache httpd 2.4.7 ((Ubuntu))  
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel  
  
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .  
Nmap done: 1 IP address (1 host up) scanned in 56.46 seconds
```

Figure: nmap version scan

- Aggressive Scan:
 - Nmap has an aggressive mode that enables OS detection, version detection, script scanning, and traceroute.
 - Aggressive scans provide far better information than regular scans. However, an aggressive scan also sends out more probes, and it is more likely to be detected during security audits.

```
(chaitanya@kali)-[~]
$ nmap -A scanme.nmap.org
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-15 10:35 IST
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.28s latency).
Other addresses for scanme.nmap.org (not scanned): 2600:3c01::f03c:91ff:fe18:bb2f
Not shown: 997 filtered tcp ports (no-response)
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|   1024 ac:00:a0:1a:82:ff:cc:55:99:dc:67:2b:34:97:6b:75 (DSA)
|   2048 20:3d:2d:44:62:2a:b0:5a:9d:b5:b3:05:14:c2:a6:b2 (RSA)
|   256 96:02:bb:5e:57:54:1c:4e:45:2f:56:4c:4a:24:b2:57 (ECDSA)
|_  256 33:fa:91:0f:e0:e1:7b:1f:6d:05:a2:b0:f1:54:41:56 (ED25519)
80/tcp    open  http         Apache httpd 2.4.7 ((Ubuntu))
|_ http-title: Go ahead and ScanMe!
|_ http-server-header: Apache/2.4.7 (Ubuntu)
|_ http-favicon: Nmap Project
9929/tcp  open  nping-echo   Nping echo
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 66.82 seconds
```

Figure: nmap aggressive scan

SpiderFoot is an open-source, cross-platform OSINT tool designed to automate the process of gathering intelligence about domains, IP addresses, email addresses, and other digital assets.

- spiderfoot web-ui:

The screenshot shows the SpiderFoot web interface for creating a new scan. The top navigation bar includes 'New Scan', 'Scans', and 'Settings' buttons, along with a 'Dark Mode' toggle and an 'About' link. The main heading is 'New Scan'. Below this, there are two input fields: 'Scan Name' (placeholder: 'The name of this scan.') and 'Scan Target' (placeholder: 'The target of your scan.'). To the right of these fields is a grey box containing a tip: 'Your scan target may be one of the following. SpiderFoot will automatically detect the target type based on the format of your input.' This box lists several target types with examples: Domain Name (e.g. example.com), IPv4 Address (e.g. 1.2.3.4), IPv6 Address (e.g. 2606:4700:4700:1111), Hostname/Sub-domain (e.g. abc.example.com), Subnet (e.g. 1.2.3.0/24), Bitcoin Address (e.g. 11ksYJSP1Q3yFCjRQ2vzBL1wjuuK0e7R), E-mail address (e.g. bob@example.com), Phone Number (e.g. +12345678901 (E.164 format)), Human Name (e.g. "John Smith" (must be in quotes)), Username (e.g. "jsmith2000" (must be in quotes)), and Network ASN (e.g. 1234). Below the input fields, there are three tabs: 'By Use Case', 'By Required Data', and 'By Module'. The 'By Use Case' tab is active, showing four radio button options: 'All' (selected), 'Footprint', 'Investigate', and 'Passive'. Each option has a description of what it does. At the bottom left, there is a blue button labeled 'Run Scan Now'.

spiderfoot New Scan Scans Settings Dark Mode About

New Scan

Scan Name
The name of this scan.

Scan Target
The target of your scan.

ⓘ Your scan target may be one of the following. SpiderFoot will automatically detect the target type based on the format of your input.

- Domain Name** e.g. example.com
- IPv4 Address** e.g. 1.2.3.4
- IPv6 Address** e.g. 2606:4700:4700:1111
- Hostname/Sub-domain** e.g. abc.example.com
- Subnet** e.g. 1.2.3.0/24
- Bitcoin Address** e.g. 11ksYJSP1Q3yFCjRQ2vzBL1wjuuK0e7R
- E-mail address** e.g. bob@example.com
- Phone Number** e.g. +12345678901 (E.164 format)
- Human Name** e.g. "John Smith" (must be in quotes)
- Username** e.g. "jsmith2000" (must be in quotes)
- Network ASN** e.g. 1234

By Use Case **By Required Data** **By Module**

☒ **All** **Get anything and everything about the target.**
All SpiderFoot modules will be enabled (slow) but every possible piece of information about the target will be obtained and analysed.

☐ **Footprint** **Understand what information this target exposes to the Internet.**
Gain an understanding about the target's network perimeter, associated identities and other information that is obtained through a lot of web crawling and search engine use.

☐ **Investigate** **Best for when you suspect the target to be malicious but need more information.**
Some basic footprinting will be performed in addition to querying of blacklists and other sources that may have information about your target's maliciousness.

☐ **Passive** **When you don't want the target to even suspect they are being investigated.**
As much information will be gathered without touching the target or their affiliates, therefore only modules that do not touch the target will be enabled.

Run Scan Now

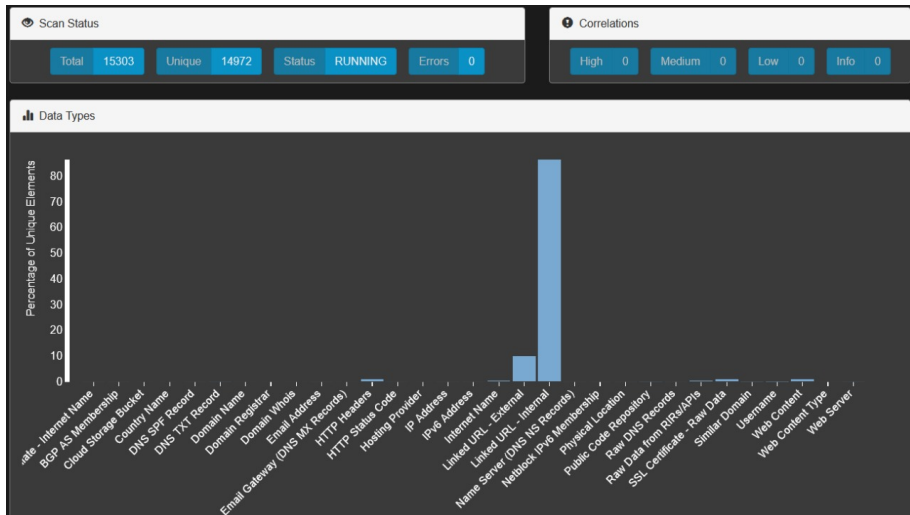


Figure: spiderfoot sample use

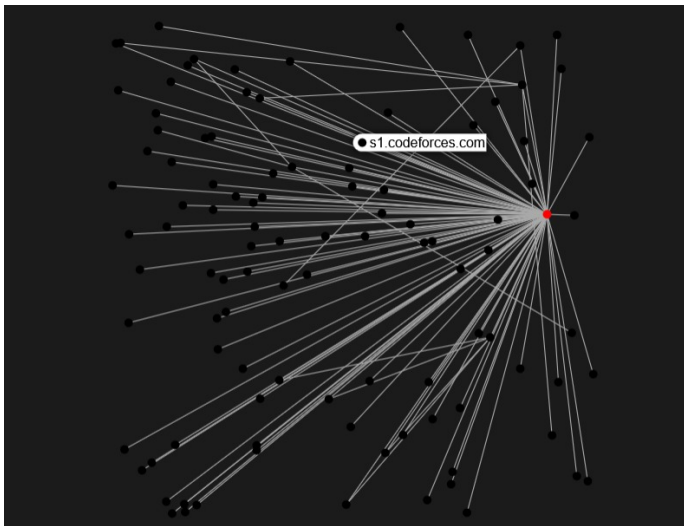


Figure: codeforces network graph

- spiderfoot cli:

```
sf> help
```


Command	Description
help [command]	This help output.
debug	Enable/Disable debug output.
clear	Clear the screen.
history	Enable/Disable/List command history.
spool	Enable/Disable spooling output.
shell	Execute a shell command.
exit	Exit the SpiderFoot CLI (won't impact running scans).
ping	Test connectivity to the SpiderFoot server.
modules	List available modules.
types	List available data types.
correlationrules	List available correlation rules.
set	Set variables and configuration settings.
scans	List all scans that have been run or are running.
start	Start a new scan.
stop	Stop a scan.
delete	Delete a scan.
scaninfo	Scan information.
data	Show data from a scan's results.
correlations	Show correlation results from a scan.
summary	Scan result summary.
find	Search for data within scan results.
query	Run SQL against the SpiderFoot SQLite database.
logs	View/watch logs from a scan.

Figure: spiderfoot cli

This tool identifies and fingerprints Web Application Firewall (WAF) products using the following logic:

- Sends a normal HTTP request and analyses the response; this identifies a number of WAF solutions.
- If that is not successful, it sends a number of (potentially malicious) HTTP requests and uses simple logic to deduce which WAF it is.
- If that is also not successful, it analyses the responses previously returned and uses another simple algorithm to guess if a WAF or security solution is actively responding to the attacks.

```
(chaitanya@kali)-[~]  
$ wafw00f codeforces.com
```



404 Hack Not Found

405 Not Allowed

403 Forbidden

502 Bad Gateway

500 Internal Error

~ WAFW00F : v2.2.0 ~

The Web Application Firewall Fingerprinting Toolkit

[*] Checking https://codeforces.com

[+] The site <https://codeforces.com> is behind Cloudflare (Cloudflare Inc.) WAF.

[~] Number of requests: 2

Figure: wafw00f sample use

- Fierce is a semi-lightweight scanner that helps locate non-contiguous IP space and hostnames against specified domains
- It's really meant as a pre-cursor to nmap, unicornscan, nessus, nikto, etc, since all of those require that you already know what IP space you are looking for.
- It is meant specifically to locate likely targets both inside and outside a corporate network.

```
(chaitanya@kali)-[~]  
$ fierce --domain codeforces.com  
NS: jean.ns.cloudflare.com. fred.ns.cloudflare.com.  
SOA: fred.ns.cloudflare.com. (108.162.193.113)  
Zone: failure  
Wildcard: failure  
Found: cdn.codeforces.com. (195.123.220.85)  
Nearby:  
{'195.123.220.80': 'itech4web.com.',  
  '195.123.220.85': 'vds-955881.hosted-by-itldc.com.',  
  '195.123.220.86': 'store.milos.in.ua.',  
  '195.123.220.88': 'clubstrike.email.',  
  '195.123.220.89': 'hosted-by-itldc.com.'}  
Found: dev.codeforces.com. (92.100.38.102)  
Found: mail.codeforces.com. (95.163.252.67)  
Nearby:  
{'95.163.252.66': 'ada.codeforces.com.',  
  '95.163.252.67': 'mx1.codeforces.com.',  
  '95.163.252.68': 'cormen.codeforces.com.'}  
Found: media.codeforces.com. (172.67.68.254)  
Found: mirror.codeforces.com. (213.248.110.126)  
Found: printer.codeforces.com. (77.234.204.2)
```

Figure: fierce sample use

Referenes and Acknowledgements

- <https://www.kali.org>
- <https://medium.com>
- <https://www.itjones.com/blogs>
- <https://github.com/OWASP/owasp-mastg>
- Stagefright: Scary Code in the Heart of Android

Thank You!!!