Research on Mobile VAPT Tools -5

Introduction

Mobile penetration testing is a critical aspect of cybersecurity, focusing on evaluating the security of mobile applications and operating systems. As mobile devices store vast amounts of sensitive data, identifying vulnerabilities and securing them against cyber threats is essential. This report highlights key mobile penetration testing tools and the knowledge gained from research in this domain

Mobile Penetration Testing Tools

Mobile penetration testing involves a variety of tools designed to assess security vulnerabilities in mobile applications and operating systems. Below are some widely used tools:

1. ADB toolkit
2. Jadx
3. APKtool
4. Objection
5. Frida
6. Drozer
7. Dex2jar
8. MOBsf
9. Apk leaks
10. APK-mitm
11. Apkhuntn etc.

Now we will discus about top 5 Mobile VAPT Tools.

1. MobSF (Mobile Security Framework)

MobSF is an open-source tool for static and dynamic analysis of Android and iOS applications. It helps in identifying security vulnerabilities, malware, and privacy risks within mobile applications.

Key Features:

- Static and dynamic analysis
- API key leakage detection
- Code obfuscation analysis
- Web traffic monitoring

2. Frida

Frida is a dynamic instrumentation toolkit used for analyzing and modifying applications at runtime. It allows security researchers to bypass security controls and inspect applications for vulnerabilities.

Key Features:

- Injects scripts into running applications
- Reverse engineering capabilities
- Supports Android and iOS platforms
- Hooking and API monitoring

3. Burp Suite

Burp Suite is a widely used web security testing tool that can also be applied to mobile app penetration testing. It is mainly used for intercepting and analyzing network traffic between mobile applications and back-end servers.

Key Features:

- Proxy-based interception
- Automated vulnerability scanning
- WebSocket and API testing
- SSL/TLS traffic analysis

4. Drozer

Drozer is a powerful security testing tool that helps assess Android application security. It allows testers to interact with installed applications and identify vulnerabilities.

Key Features:

- Exploit Android application vulnerabilities
- Identify misconfigurations in Android apps
- Automated testing and scripting
- Built-in vulnerability assessment modules

5. APKTool

APKTool is a widely used tool for reverse engineering Android applications. It enables security professionals to decompile and analyze the code for vulnerabilities and security flaws.

Key Features:

- Decompiling and rebuilding APK files
- Modification of application resources
- Code debugging and analysis
- Understanding app structure

6. Zed Attack Proxy (ZAP)

ZAP, developed by OWASP, is an open-source security testing tool that helps identify vulnerabilities in web applications, including mobile web applications.

Key Features:

- Automated scanner for web vulnerabilities
- Passive and active scanning

- Proxy-based interception and traffic analysis
- Custom scripting support

7. ADBToolkit

ADBToolkit is a collection of scripts and tools designed to enhance the capabilities of the Android Debug Bridge (ADB) for penetration testing and security assessments. It simplifies common ADB commands and automates various security-related tasks.

Key Features:

- Automated extraction of application data
- Forensic analysis of Android devices
- APK installation and uninstallation management
- Log analysis and debugging
- Security configuration checks

Knowledge Gained

Through research in mobile penetration testing, several key insights were obtained:

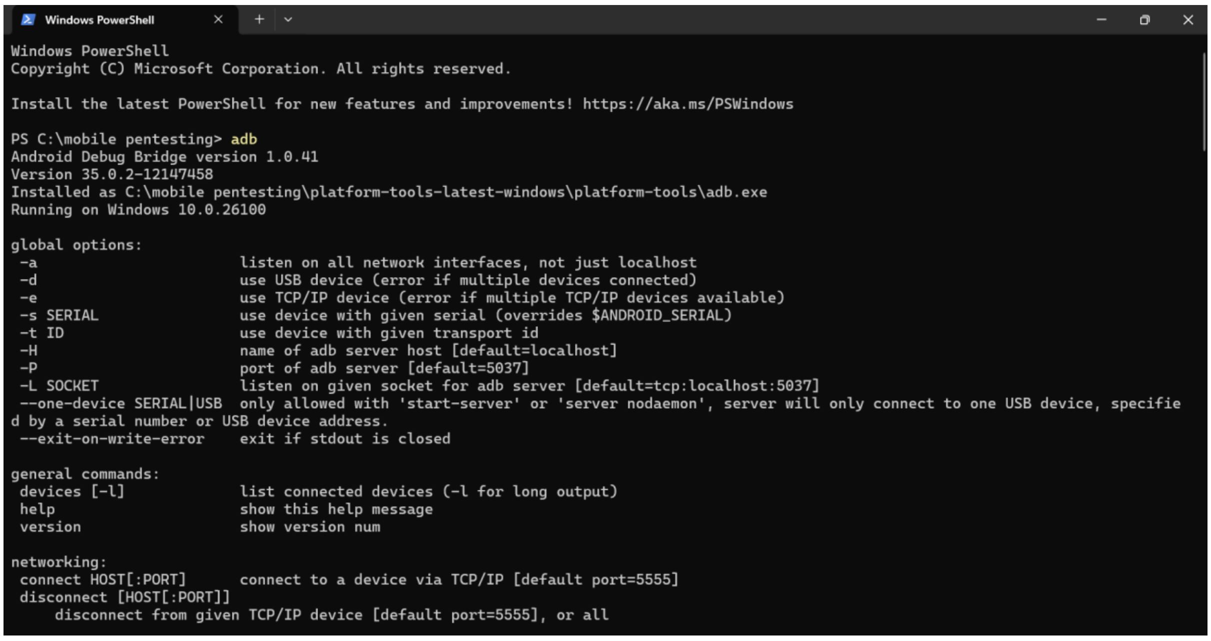
1. **Understanding Security Risks** – Mobile applications often contain vulnerabilities such as insecure data storage, weak authentication mechanisms, and improper cryptographic implementations.
2. **Importance of Dynamic and Static Analysis** – Both static and dynamic analysis play crucial roles in identifying security weaknesses in mobile applications.
3. **Role of Reverse Engineering** – Decompiling and analyzing application code helps security researchers detect hidden vulnerabilities and malicious code.
4. **Network Security Concerns** – Mobile applications frequently communicate with remote servers, making them susceptible to man-in-the-middle attacks and data interception.
5. **Compliance and Best Practices** – Following security standards such as OWASP Mobile Security Testing Guide (MSTG) enhances the security posture of mobile applications.

Conclusion

Mobile penetration testing is an essential practice for securing mobile applications against evolving cyber threats. By leveraging tools like MobSF, Frida, Burp Suite, and others, security professionals can identify and mitigate vulnerabilities effectively. Understanding the functionality and application of these tools is crucial for ensuring robust mobile security.

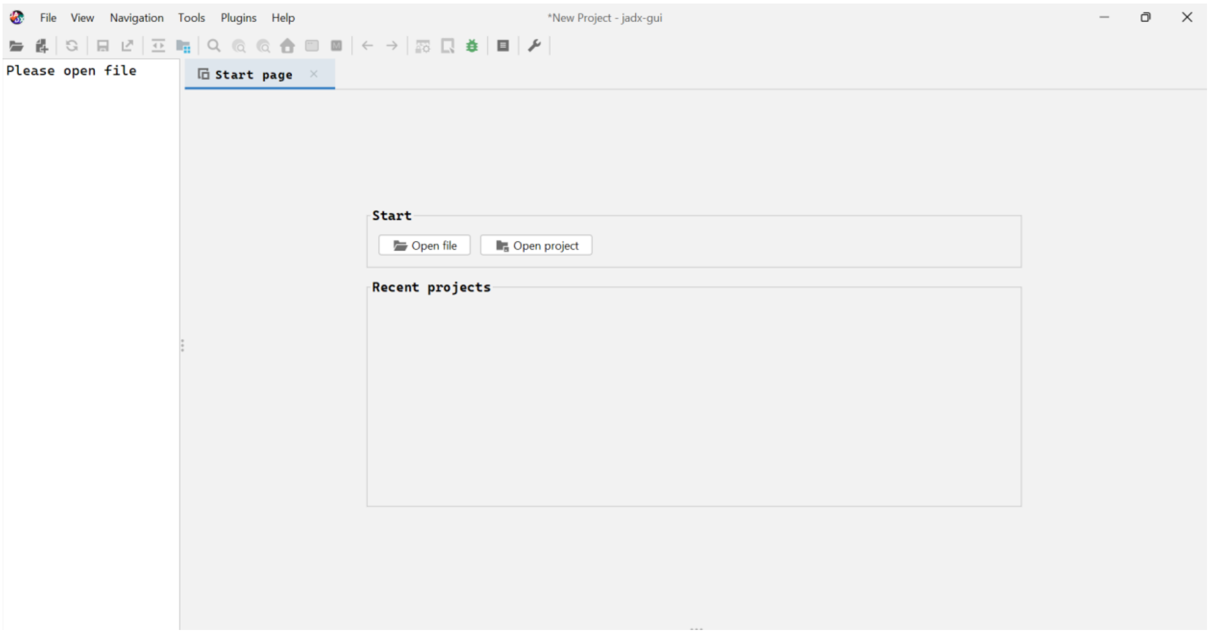
Screenshots of installed tools:

1. ADB toolkit

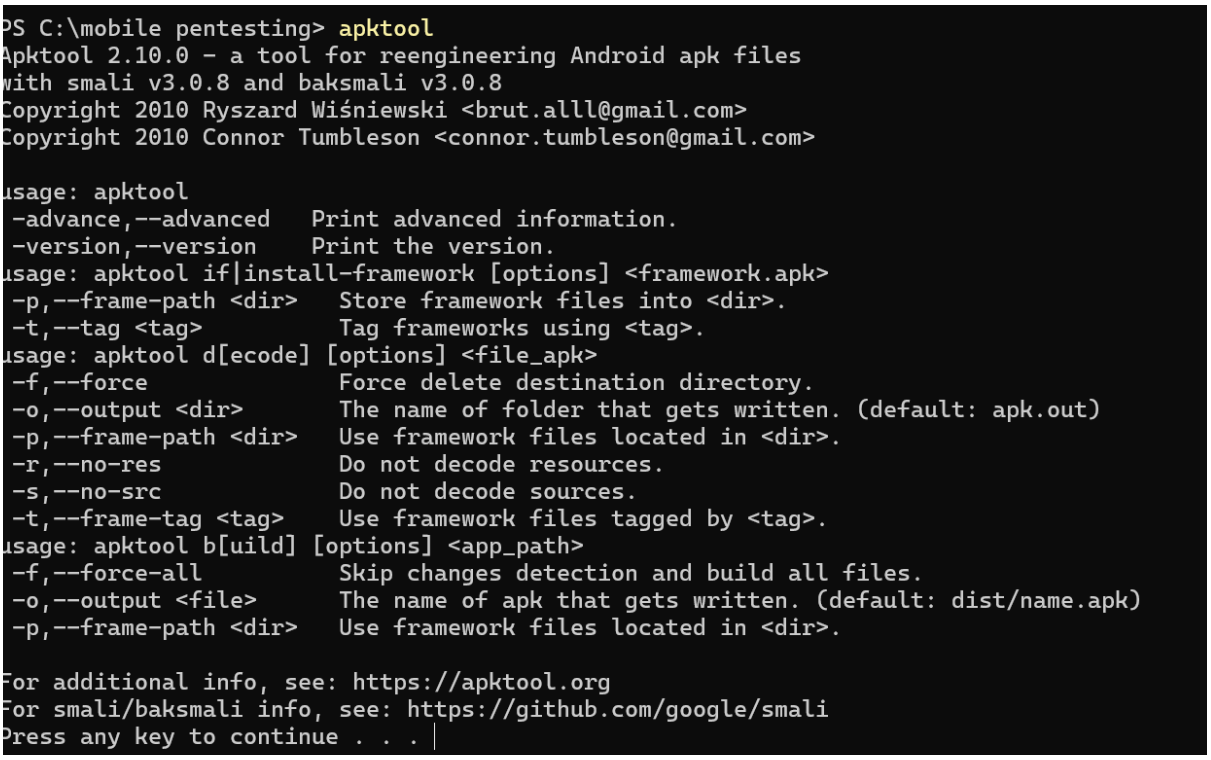


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\mobile pentesting> adb
Android Debug Bridge version 1.0.41
Version 35.0.2-12147458
Installed as C:\mobile pentesting\platform-tools-latest-windows\platform-tools\adb.exe
Running on Windows 10.0.26100

global options:
 -a                       listen on all network interfaces, not just localhost
 -d                       use USB device (error if multiple devices connected)
 -e                       use TCP/IP device (error if multiple TCP/IP devices available)
 -s SERIAL                use device with given serial (overrides $ANDROID_SERIAL)
 -t ID                    use device with given transport id
 -H                       name of adb server host [default=localhost]
 -P                       port of adb server [default=5037]
 -L SOCKET                listen on given socket for adb server [default=tcp:localhost:5037]
 --one-device SERIAL|USB  only allowed with 'start-server' or 'server nodaemon', server will only connect to one USB device, specifie
d by a serial number or USB device address.
 --exit-on-write-error    exit if stdout is closed

general commands:
 devices [-l]             list connected devices (-l for long output)
 help                     show this help message
 version                  show version num

networking:
 connect HOST[:PORT]      connect to a device via TCP/IP [default port=5555]
 disconnect [HOST[:PORT]]
     disconnect from given TCP/IP device [default port=5555], or all
```

2. Jadx

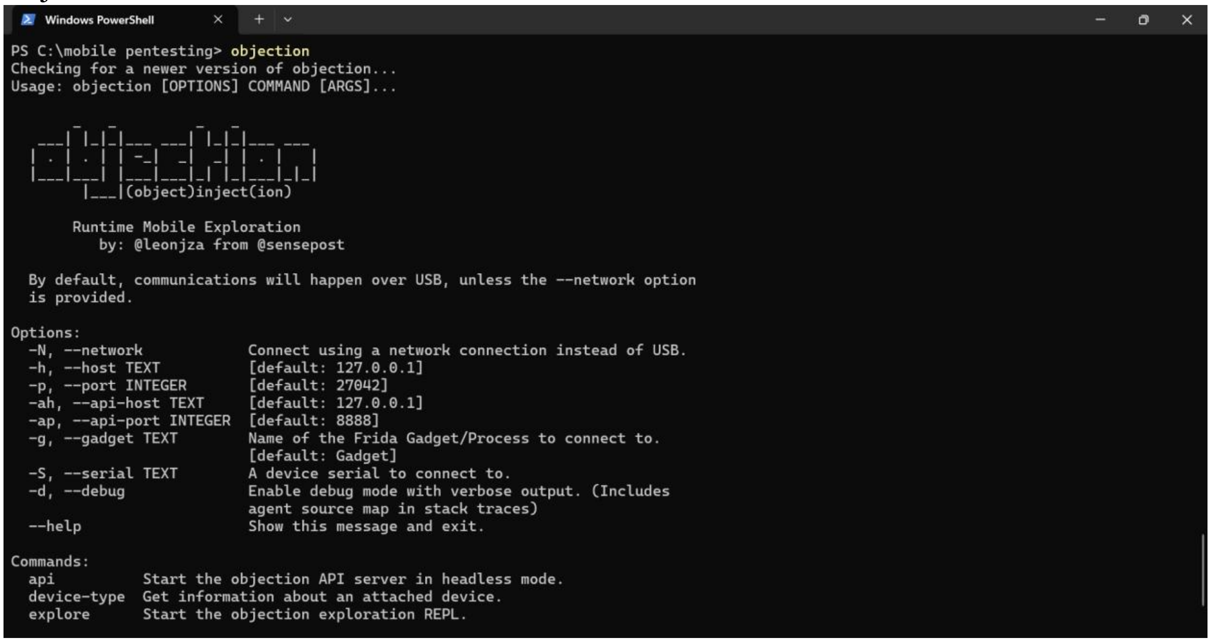


3. APK tool



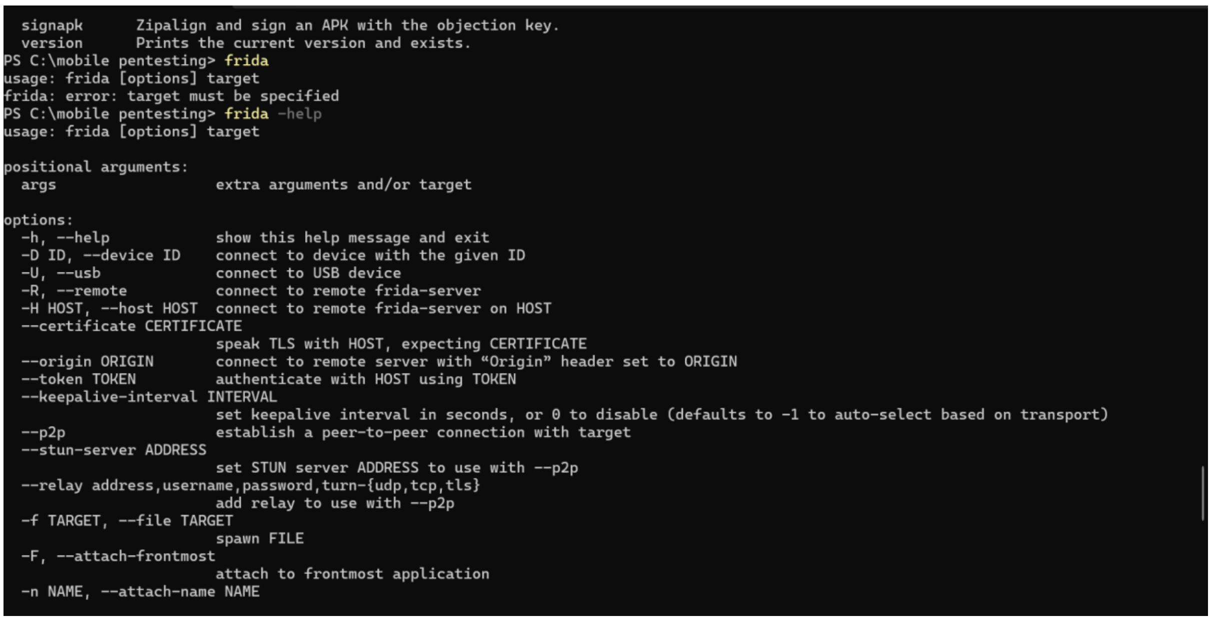
```
PS C:\mobile pentesting> apktool
Apktool 2.10.0 - a tool for reengineering Android apk files
with smali v3.0.8 and baksmali v3.0.8
Copyright 2010 Ryszard Wiśniewski <brut.alll@gmail.com>
Copyright 2010 Connor Tumbleson <connor.tumbleson@gmail.com>

usage: apktool
 -advance,--advanced   Print advanced information.
 -version,--version    Print the version.
usage: apktool if|install-framework [options] <framework.apk>
 -p,--frame-path <dir>   Store framework files into <dir>.
 -t,--tag <tag>          Tag frameworks using <tag>.
usage: apktool d[ecode] [options] <file_apk>
 -f,--force              Force delete destination directory.
 -o,--output <dir>       The name of folder that gets written. (default: apk.out)
 -p,--frame-path <dir>   Use framework files located in <dir>.
 -r,--no-res             Do not decode resources.
 -s,--no-src             Do not decode sources.
 -t,--frame-tag <tag>    Use framework files tagged by <tag>.
usage: apktool b[uild] [options] <app_path>
 -f,--force-all          Skip changes detection and build all files.
 -o,--output <file>      The name of apk that gets written. (default: dist/name.apk)
 -p,--frame-path <dir>   Use framework files located in <dir>.

For additional info, see: https://apktool.org
For smali/baksmali info, see: https://github.com/google/smali
Press any key to continue . . .
```

4. Objection



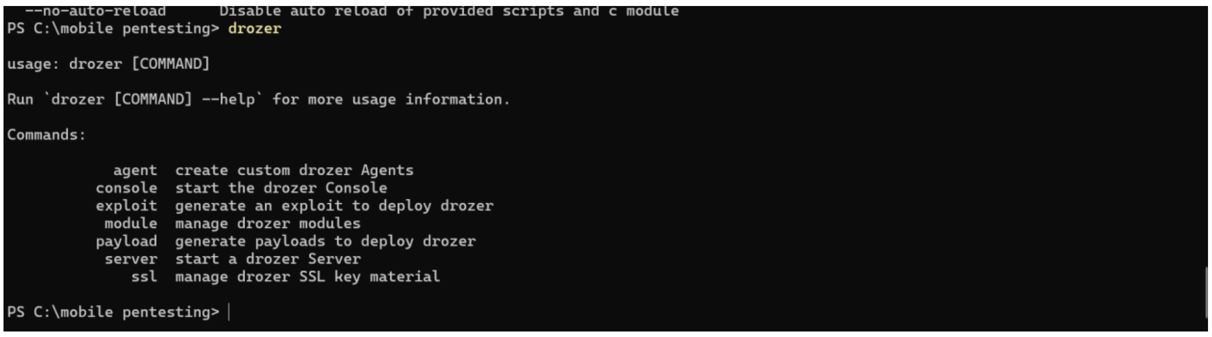
```
PS C:\mobile pentesting> objection
Checking for a newer version of objection...
Usage: objection [OPTIONS] COMMAND [ARGS]...

       _   _         _   _
   ___| |_|_|___ ___| |_|_|___ ___
  | . | . | | -_|  _|  _| | . |   |
  |___|___| |___|___|_| |_|___|_|_|
        |___|(object)inject(ion)

       Runtime Mobile Exploration
          by: @leonjza from @sensepost

  By default, communications will happen over USB, unless the --network option
  is provided.

Options:
  -N, --network            Connect using a network connection instead of USB.
  -h, --host TEXT          [default: 127.0.0.1]
  -p, --port INTEGER       [default: 27042]
  -ah, --api-host TEXT     [default: 127.0.0.1]
  -ap, --api-port INTEGER  [default: 8888]
  -g, --gadget TEXT        Name of the Frida Gadget/Process to connect to.
                           [default: Gadget]
  -S, --serial TEXT        A device serial to connect to.
  -d, --debug              Enable debug mode with verbose output. (Includes
                           agent source map in stack traces)
  --help                   Show this message and exit.

Commands:
  api          Start the objection API server in headless mode.
  device-type  Get information about an attached device.
  explore      Start the objection exploration REPL.
```

5. Frida



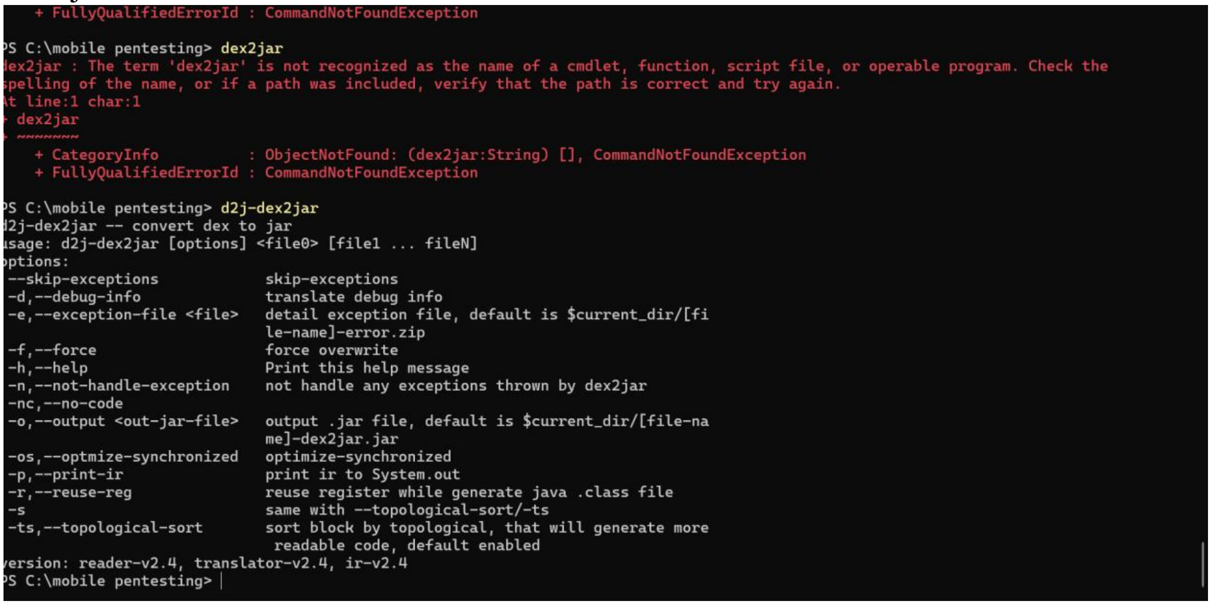
```
  signapk      Zipalign and sign an APK with the objection key.
  version      Prints the current version and exists.
PS C:\mobile pentesting> frida
usage: frida [options] target
frida: error: target must be specified
PS C:\mobile pentesting> frida -help
usage: frida [options] target

positional arguments:
  args                  extra arguments and/or target

options:
  -h, --help            show this help message and exit
  -D ID, --device ID    connect to device with the given ID
  -U, --usb             connect to USB device
  -R, --remote          connect to remote frida-server
  -H HOST, --host HOST  connect to remote frida-server on HOST
  --certificate CERTIFICATE
                        speak TLS with HOST, expecting CERTIFICATE
  --origin ORIGIN       connect to remote server with “Origin” header set to ORIGIN
  --token TOKEN         authenticate with HOST using TOKEN
  --keepalive-interval INTERVAL
                        set keepalive interval in seconds, or 0 to disable (defaults to -1 to auto-select based on transport)
  --p2p                 establish a peer-to-peer connection with target
  --stun-server ADDRESS
                        set STUN server ADDRESS to use with --p2p
  --relay address,username,password,turn-{udp,tcp,tls}
                        add relay to use with --p2p
  -f TARGET, --file TARGET
                        spawn FILE
  -F, --attach-frontmost
                        attach to frontmost application
  -n NAME, --attach-name NAME
```

6. Drozer



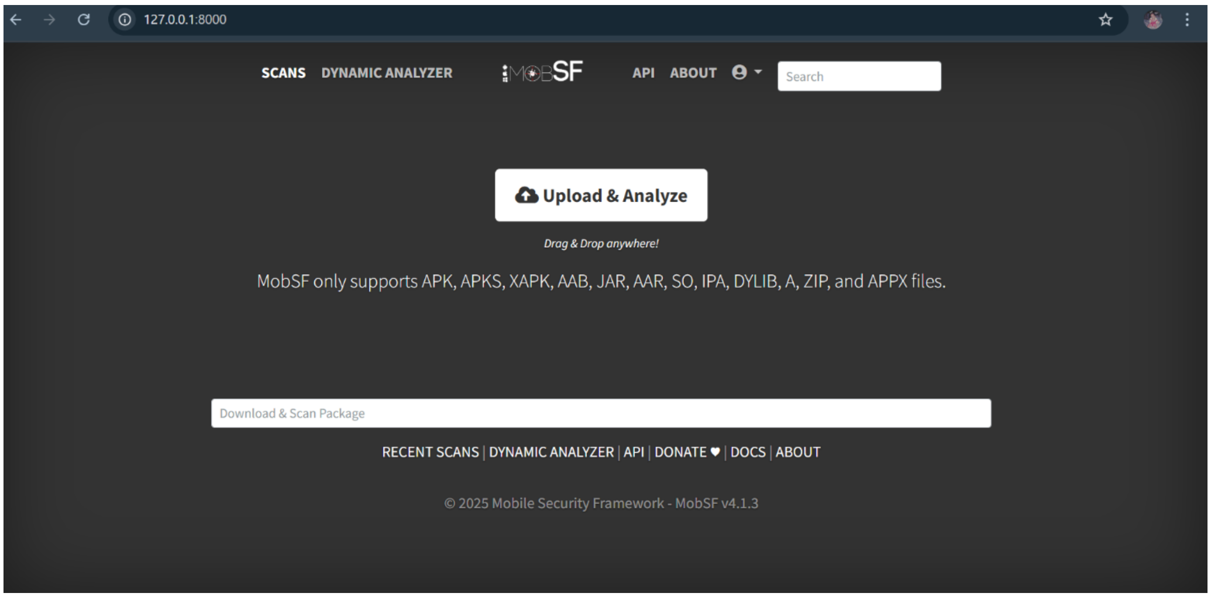
```
  --no-auto-reload      Disable auto reload of provided scripts and c module
PS C:\mobile pentesting> drozer

usage: drozer [COMMAND]

Run `drozer [COMMAND] --help` for more usage information.

Commands:

            agent  create custom drozer Agents
          console  start the drozer Console
          exploit  generate an exploit to deploy drozer
           module  manage drozer modules
          payload  generate payloads to deploy drozer
           server  start a drozer Server
              ssl  manage drozer SSL key material

PS C:\mobile pentesting> |
```

7. Dex2jar



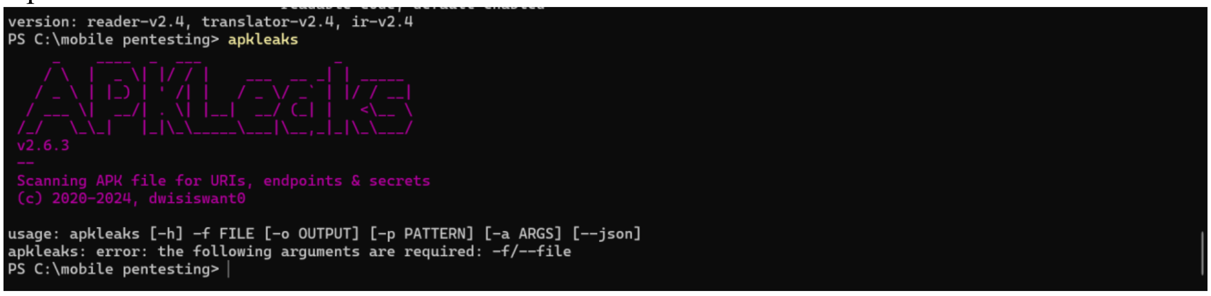
```
    + FullyQualifiedErrorId : CommandNotFoundException

PS C:\mobile pentesting> dex2jar
lex2jar : The term 'dex2jar' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the
pelling of the name, or if a path was included, verify that the path is correct and try again.
t line:1 char:1
 dex2jar
 ~~~~~~~
    + CategoryInfo          : ObjectNotFound: (dex2jar:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException

S C:\mobile pentesting> d2j-dex2jar
12j-dex2jar -- convert dex to jar
sage: d2j-dex2jar [options] <file0> [file1 ... fileN]
ptions:
 --skip-exceptions              skip-exceptions
 -d,--debug-info                translate debug info
 -e,--exception-file <file>     detail exception file, default is $current_dir/[fi
                                le-name]-error.zip
 -f,--force                     force overwrite
 -h,--help                      Print this help message
 -n,--not-handle-exception      not handle any exceptions thrown by dex2jar
 -nc,--no-code
 -o,--output <out-jar-file>     output .jar file, default is $current_dir/[file-na
                                me]-dex2jar.jar
 -os,--optmize-synchronized     optimize-synchronized
 -p,--print-ir                  print ir to System.out
 -r,--reuse-reg                 reuse register while generate java .class file
 -s                             same with --topological-sort/-ts
 -ts,--topological-sort         sort block by topological, that will generate more
                                 readable code, default enabled
ersion: reader-v2.4, translator-v2.4, ir-v2.4
S C:\mobile pentesting> |
```

8. MOBSF



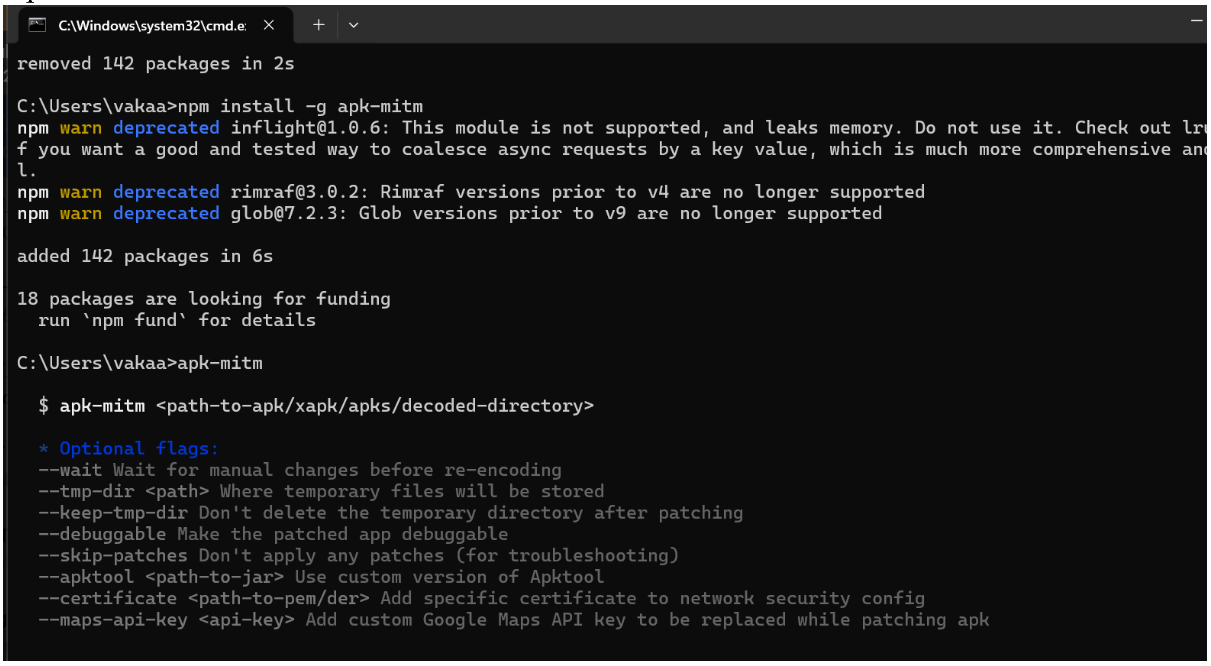
```
127.0.0.1:8000

SCANS   DYNAMIC ANALYZER   MobSF   API   ABOUT   Search

Upload & Analyze
Drag & Drop anywhere!
MobSF only supports APK, APKS, XAPK, AAB, JAR, AAR, SO, IPA, DYLIB, A, ZIP, and APPX files.

Download & Scan Package

RECENT SCANS | DYNAMIC ANALYZER | API | DONATE ♥ | DOCS | ABOUT

© 2025 Mobile Security Framework - MobSF v4.1.3
```

9. Apk leaks



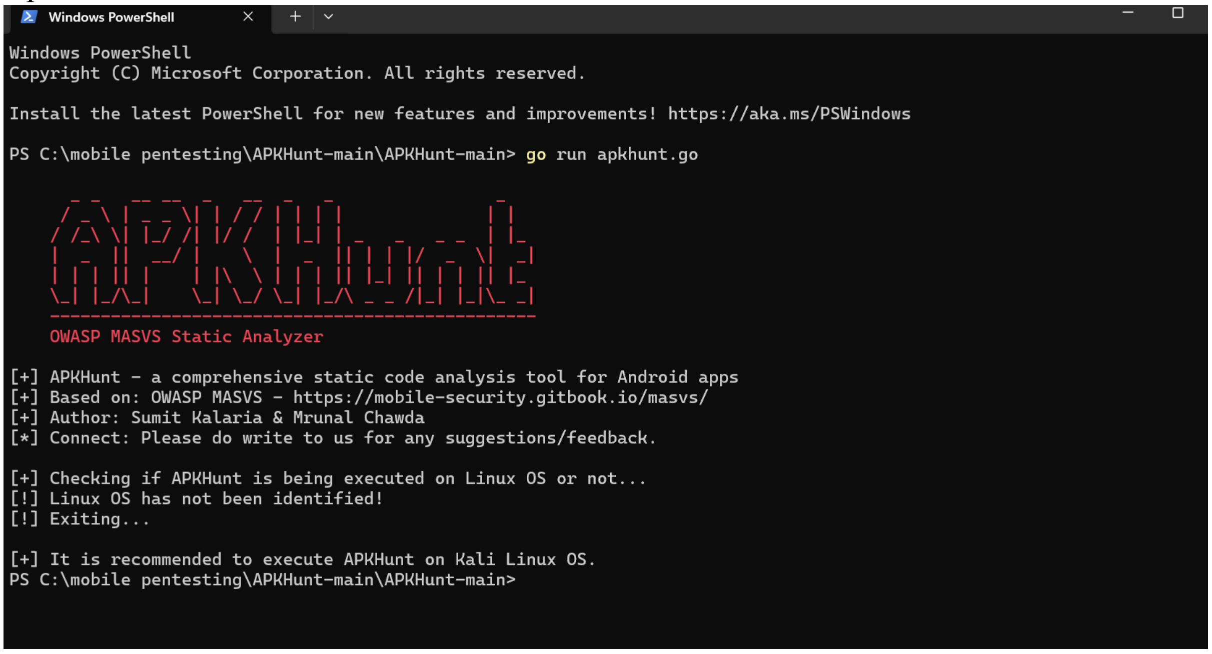
```
version: reader-v2.4, translator-v2.4, ir-v2.4
PS C:\mobile pentesting> apkleaks
     _    ____  _  ___               _
    / \  |  _ \| |/ / |    ___  __ _| | _____
   / _ \ | |_) | ' /| |   / _ \/ _` | |/ / __|
  / ___ \|  __/| . \| |__|  __/ (_| |   <\__ \
 /_/   \_\_|   |_|\_\_____\___|\__,_|_|\_\___/
 v2.6.3
 --
 Scanning APK file for URIs, endpoints & secrets
 (c) 2020-2024, dwisiswant0

usage: apkleaks [-h] -f FILE [-o OUTPUT] [-p PATTERN] [-a ARGS] [--json]
apkleaks: error: the following arguments are required: -f/--file
PS C:\mobile pentesting>
```

10. Apk-mitm



```
C:\Windows\system32\cmd.e

removed 142 packages in 2s

C:\Users\vakaa>npm install -g apk-mitm
npm warn deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check out lru
f you want a good and tested way to coalesce async requests by a key value, which is much more comprehensive and
l.
npm warn deprecated rimraf@3.0.2: Rimraf versions prior to v4 are no longer supported
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported

added 142 packages in 6s

18 packages are looking for funding
  run `npm fund` for details

C:\Users\vakaa>apk-mitm

  $ apk-mitm <path-to-apk/xapk/apks/decoded-directory>

  * Optional flags:
  --wait Wait for manual changes before re-encoding
  --tmp-dir <path> Where temporary files will be stored
  --keep-tmp-dir Don't delete the temporary directory after patching
  --debuggable Make the patched app debuggable
  --skip-patches Don't apply any patches (for troubleshooting)
  --apktool <path-to-jar> Use custom version of Apktool
  --certificate <path-to-pem/der> Add specific certificate to network security config
  --maps-api-key <api-key> Add custom Google Maps API key to be replaced while patching apk
```

11. Apk hunt



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\mobile pentesting\APKHunt-main\APKHunt-main> go run apkhunt.go

      ___  ______ _   __ _   _             _
     / _ \ | ___ \ | / /| | | |           | |
    / /_\ \| |_/ / |/ / | |_| |_   _ _ __ | |_
    |  _  ||  __/|    \ |  _  | | | | '_ \| __|
    | | | || |   | |\  \| | | | |_| | | | | |_
    \_| |_/\_|   \_| \_/\_| |_/\__,_|_| |_|\__|
    ------------------------------------------------
    OWASP MASVS Static Analyzer

[+] APKHunt - a comprehensive static code analysis tool for Android apps
[+] Based on: OWASP MASVS - https://mobile-security.gitbook.io/masvs/
[+] Author: Sumit Kalaria & Mrunal Chawda
[*] Connect: Please do write to us for any suggestions/feedback.

[+] Checking if APKHunt is being executed on Linux OS or not...
[!] Linux OS has not been identified!
[!] Exiting...

[+] It is recommended to execute APKHunt on Kali Linux OS.
PS C:\mobile pentesting\APKHunt-main\APKHunt-main>
```

References:

1. https://apktool.org/
2. https://github.com/pxb1988/dex2jar
3. https://github.com/skylot/jadx/releases
4. https://github.com/frida/frida
5. https://github.com/sensepost/objection
6. https://medium.com/@elanustechno/top-mobile-application-penetration-testing-tools-for-android-and-ios-ee599f0a00c4