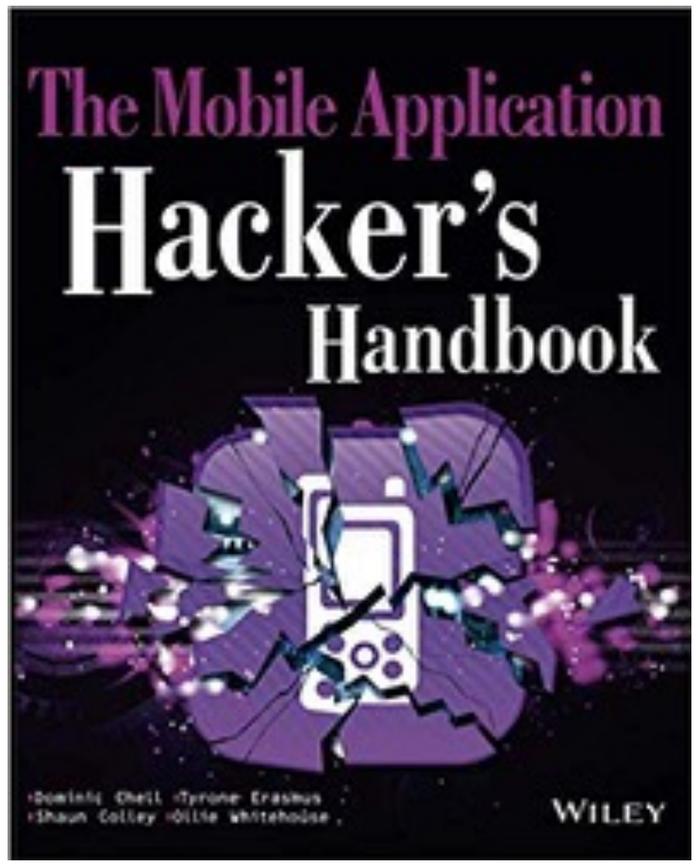
CNIT 128
Hacking Mobile Devices



7. Attacking Android Applications Part 3

Updated 3-10-21

Topics

- Part 1
  - Exposing Security Model Quirks
  - Attacking Application Components
    (to p. 271)
- Part 2
  - Attacking Application Components (finishes)

Topics

- Part 3
- Accessing Storage and Logging
- Misusing Insecure Communications
- Exploiting Other Vectors
- Additional Testing Techniques

Accessing Storage and Logging

File and Folder Permissions

```
vbox86p:/data/system # ls -l packages.list
-rw-r----- 1 system package_info 18148 2019-03-04 17:49 packages.list
```

- 10 characters in permissions section
  - -rw-r-----
    - First char: **-** for file, **d** for directories
    - Next 3 chars: **rwx** for owner (user)
    - Next 3 chars: **rwx** for group
    - Next 3 chars: **rwx** for others
      - ***World readable, writable, executable***

Numerical Permissions

- 4 = Read
- 2 = Write
- 1 = Execute

- In octal (base 8)
- **chmod 777** assigns **`rwxrwxrwx`**
- **chmod 644** assigns **`rw-r--r--`**
- **chmod 755** assigns **`rwxr-xr-x`**

Umask

- Determines default permissions for newly created files and folders
- For Android 4.0 and up umask is 0077
  - So applications have **`rwx------`**
  - Owner-only

Traversal Checking

- Android, like other Linux systems, enforces ***traversal checking***
- A file is only accessible to other apps
  - If all the parent folders also allow access

Droidwall Privilege Escalation

- The Droidwall app used the **iptables** firewall
- And made a **droidwall.sh** script with **777** permissions, running as root
- A malicious app could add commands to that file

File Encryption

- Encrypting files protects them from attackers
- But where do you put the key?
  - Often in the source code, where attackers can easily find it
- **SQLCipher** is often used this way

```
SQLiteDatabase database = SQLiteDatabase.openOrCreateDatabase(

databaseFile, "test123", null);
```

SQLite Cracker

- Tries all words in the app

```
$ for pass in 'strings classes.dex'; do echo -n "[*] '$pass' ...";
C='sqlcipher encrypted.db "PRAGMA key='$pass';select * from
sqlite_master;"'; echo $C; done
```

SD Card Storage

- Android can use built-in SD cards and external ones
- SD cards are typically formatted with FAT32
  - No permissions possible
- Android versions before 4.4 required **android.permission.WRITE_EXTERNAL_STORAGE** permission to write to SD card
  - But no permissions to read from it

SD Card Storage

- Common locations for external SD card
  - `/sdcard/external_sd`
  - `/sdcard/ext_sd`
- Android 4.4 and later enforce **android.permission.READ_EXTERNAL_STORAGE** permission

WhatsApp Database Storage

- WhatsApp stored its database on the SD card
- Any app granted **android.permission.READ_EXTERNAL_STORAGE** permission could read it
- It was AES encrypted, but with a static key
- The "WhatsApp Xtract" tool could open it

Logging

- Useful for developers
- Applications with **READ_LOGS** permission can read the logs
- Android 4.1 and later changed **READ_LOGS** to **signature | system | development**
  - No third-party app can obtain this permission through the normal installation process

Enabling Permission from ADB

- You can enable the permission from adb

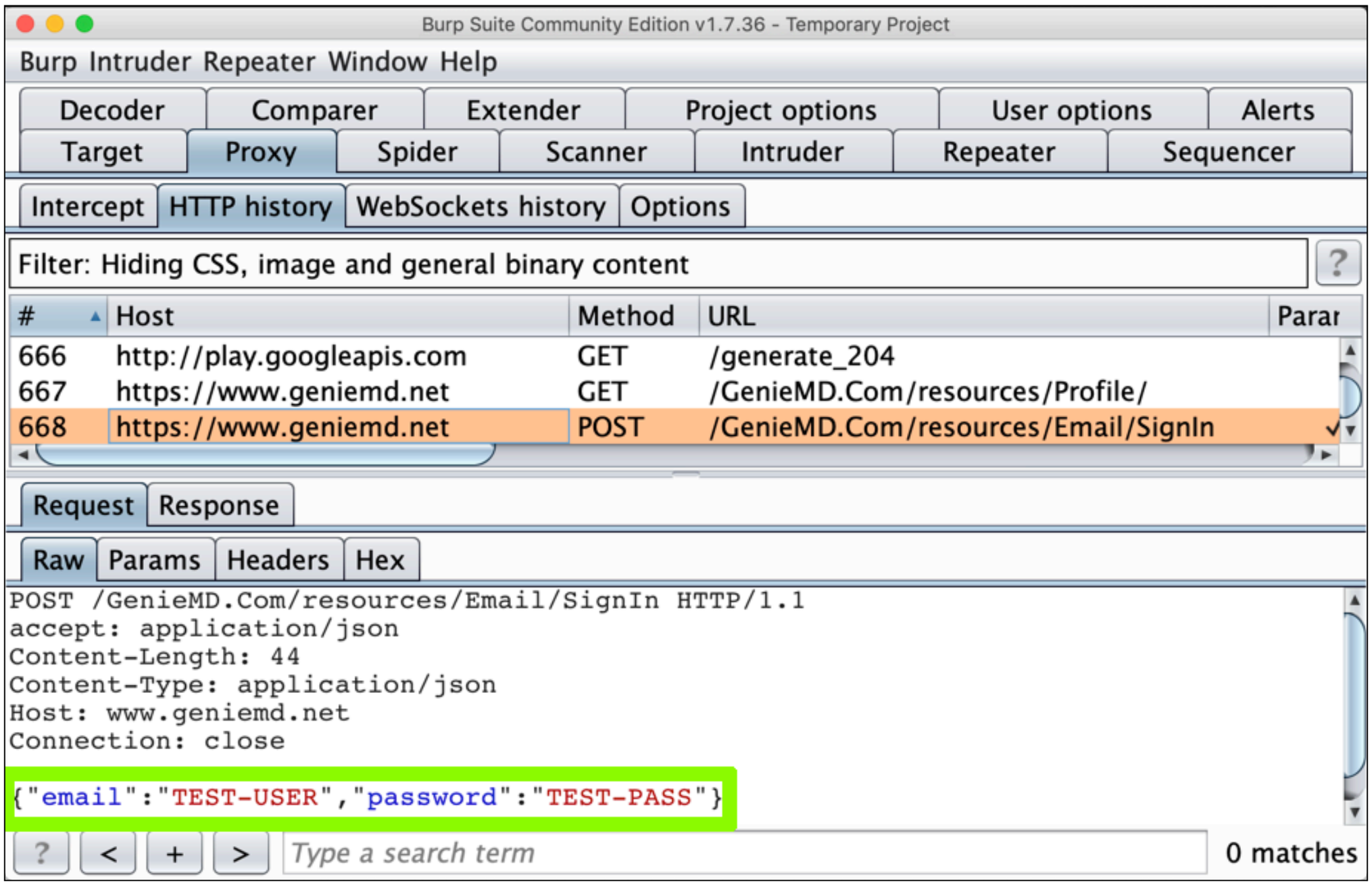
```
root@generic:/ # pm grant com.logging.app android.permission.READ_LOGS
```

Misusing Insecure Communications

Insecure Communications

- HTTP (unencrypted)
- HTTPS without validating TLS certificate
- We've done it in the projects with Burp

Insecure Communications



Certificate Pinning

- App checks the TLS certificate with custom code
  - It won't accept the 'Trusted Credentials" in Android
- There are several ways to overcome this

Four Ways to Bypass Android SSL Verification and Certificate Pinning

Cody Wass
January 9th, 2018

- Adding a custom CA to the trusted certificate store
- Overwriting a packaged CA cert with a custom CA cert
- Using Frida to hook and bypass SSL certificate checks
- Reversing custom certificate code

- Link Ch 7q

SSL Validation Flaws

- Developers often disable validation in the app
  - For testing and to prevent annoying error messages
  - By adding a **HostNameVerifier** method that always returns **true**
  - Or a **TrustManager** method that does nothing

WebViews

- **WebView** allows web pages to be rendered within an app
  - Prior to Android 4.4 it used WebKit
  - Later versions use Chromium
- **WebView** runs in the app's context
  - Allows the developer's hooks, often disabling TLS

WebSettings Class

- Controls configuration of WebView

Table 7.2 Configuration options available in the WebSettings class that pertain to security

METHOD	DEFAULT VALUE	IMPLICATION OF BEING ENABLED
`setAllowContent Access`	`true`	`WebView` **has access to content providers on the system.**
`setAllowFileAccess`	`true`	**Allows a** `WebView` **to load content from the filesystem using file:// scheme.**
`setAllowFileAccessFromFileURLs`	`true (<= API 15) false (>= API 16)`	**Allows the HTML file that was loaded using file:// scheme to access other files on the filesystem.**

`setAllowUniversalAccessFromFileURLs`	`true (<= API 15)false (>= API 16)`	**Allows the HTML file that was loaded using file:// to access content from any origin (including other files).**
`setJavaScriptEnabled`	`false`	**Allows the `WebView` to execute JavaScript.**
`setPluginState (deprecated in API 18)`	`PluginState.OFF`	**Allows the loading of plug-ins (for example, Flash) inside the `WebView`. This could in some cases even be used to load a malicious plug-in (see Google Bug #13678484 aka “Fake ID**

`setSavePassword (deprecated in API 18)`	`true`	**The `WebView` will save passwords entered.**

Exploiting WebView

- If it loads content over HTTP, add code to the page with a MITM attack
- Send a malicious intent with an extra containing a URI like
  - **file:///data/data/com.malicious/app/exploit.html**

JavaScript Interfaces

- Connects JavaScript within WebView to Java

```
/* Java code */

class JavaScriptObj

{

    @JavascriptInterface

    public String hello()

    {

        return "I am from Java code";

    }

}

webView.addJavascriptInterface(new JavaScriptObj(), "jsvar");

String content = "<html><script>alert(jsvar.hello());</script></html>";

webView.loadData(content, "text/html", null);
```

CVE-2012-6636—ADDJAVASCRIPTINTERFACE ARBITRARY CODE EXECUTION

- Execute any OS command
  - On Android before 4.1
- And any app with targetSdkVersion < 17
- On any devioce

```
window.jsvar.getClass().forName('java.lang.Runtime').getMethod(

'getRuntime',null).invoke(null,null).exec(cmd);
```

Drozer Module

```
dz> run scanner.misc.checkjavascriptbridge -a com.vulnerable.js

Package: com.vulnerable.js
  - vulnerable to WebView.addJavascriptInterface + targetSdkVersion=15
```

Other Communication Mechanisms

- Clipboard
- Local Sockets

Clipboard

- Data can be read and written by any app
  - No permission required
- Password managers often use the clipboard to move password into an app
  - Any app can easily steal it
- Drozer can read the clipboard

```
dz> run post.capture.clipboard

[*] Clipboard value: password123
```

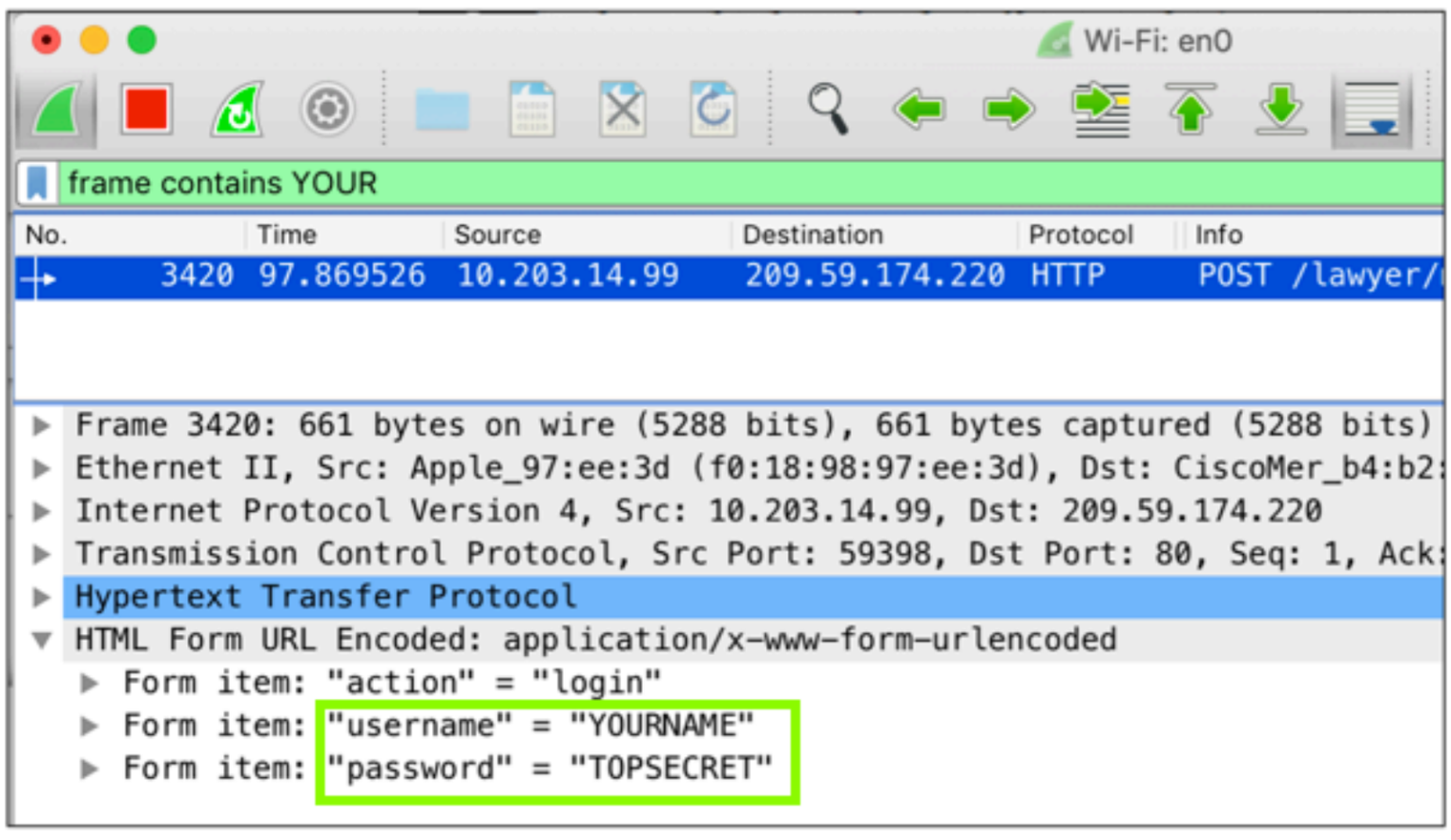
Local Sockets

- Apps can use sockets to communicate with other apps
  - TCP, UDP, or Unix
- Any app on the same phone can connect to it

```
$ adb shell netstat -antp

Proto Recv-Q Send-Q Local Address       Foreign Address     State

...

 tcp        0      0 127.0.0.1:5555     0.0.0.0:*           LISTEN

...
```

TCP/UDP Traffic

- You can inspect network traffic sent to the Internet
  - With Wireshark, Burp, or tcpdump



Exploiting Other Vectors

Abusing Native Code

- Compilied C/C++ Code
- Analyze with IDA
  - Covered in CNIT 126
  - Vulnerabilties like buffer overflows
- Develop exploits with **GDB** debugger
  - Covered in CNIT 127
  - Available for Android

Exploiting Misconfigured Package Attributes

- AndroidManifest.xml may allow
  - **Application Backups**
    - **true** by default
    - Make backup with **adb backup**
    - Exposes all an app's data
- **Debuggable Flag**
  - **false** by default

Debuggable Flag

- Exposes application data
- Allows code execution in context of application
- Can be exploited with physical access to the USB port (if USB debugging is enabled)
- Uses a Unix socket **@jdwp-control**
- But it's not enabled by default

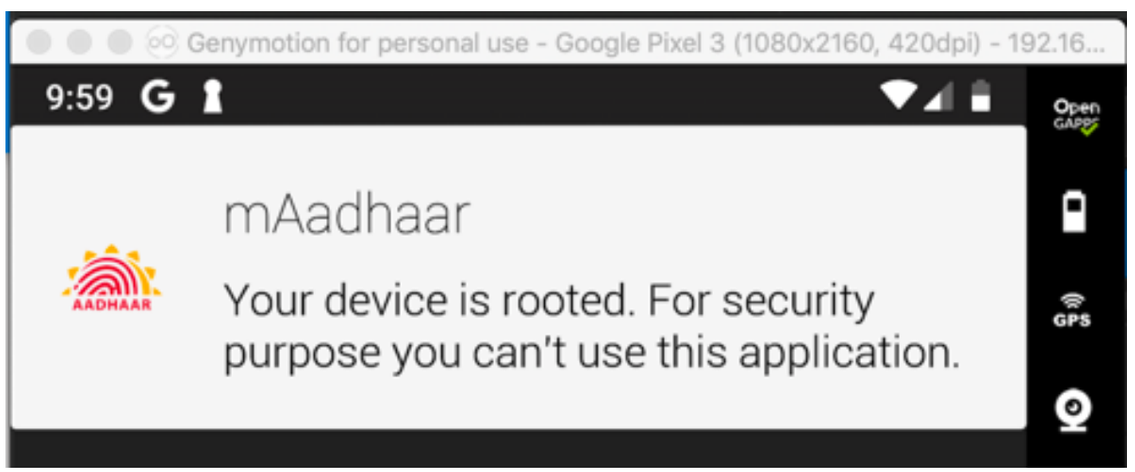
Data Exposed by Debugger

```
$ adb shell run-as com.mwr.example.sieve sqlite3 databases/database.db

.dump

PRAGMA foreign_keys=OFF;

BEGIN TRANSACTION;

CREATE TABLE android_metadata (locale TEXT);

INSERT INTO "android_metadata" VALUES('en_US');

CREATE TABLE Passwords (_id INTEGER PRIMARY KEY,service TEXT,username

TEXT,password BLOB,email );

INSERT INTO Passwords VALUES(1,'Gmail','tyrone',X'CC0EFA591F665110CD344C

384D48A2755291B8A2C46A683987CE13','tyrone@gmail.com');

INSERT INTO Passwords VALUES(2,'Internet Banking','tyrone123',X'5492FBCE

841D11EC9E610076FC302B94DBF71B59BE7E95821248374C5529514B62',
```

Additional Testing Techniques

Layered Security

- Certificate pinning
  - App checks TLS certificate
  - Won't accept a proxy like Burp
- Root detection
  - App refuses to run on a rooted device



Patching Apps

- Modify code with apktool to remove security tests

```
...2018-09-26/smali/in/gov/uidai/mAadhaarPlus/ui/activity/SplashScreenActivity.smali Modified

   invoke-static {p0}, Lin/gov/uidai/mAadhaarPlus/b/f;->b(Landroid/content/Context;)Z

   move-result v0

   if-eqz v0, :cond_0

   goto :goto_0

   :cond_0
   const/4 p1, 0x1

   :goto_0
   if-eqz p1, :cond_3

#   new-instance p1, Lcom/scottyab/rootbeer/b;

#   invoke-direct {p1, p0}, Lcom/scottyab/rootbeer/b;-><init>(Landroid/content/Context;)V

#   invoke-virtual {p1}, Lcom/scottyab/rootbeer/b;->a()Z

#   move-result p1

#   if-eqz p1, :cond_1

#   const p1, 0x7f0f0061

#   invoke-virtual {p0, p1}, Lin/gov/uidai/mAadhaarPlus/ui/activity/SplashScreenActivity;->$

#   move-result-object p1

#   goto :goto_2

   :cond_1
   invoke-static {p0}, Lin/gov/uidai/mAadhaarPlus/j/i;->c(Landroid/content/Context;)Z
```

Manipulating the Runtime

- Insert low-level hooks into the Android system
- So the API calls the apps make are changed
- Analogous to placing the app into a virtual machine
- Allows you to change the operation of an app without modifying the app itself

Tools

- Cydia Substrate
- Xposed Framework
- Frida


Kahoot!