Hacking Android Apps

By Thomas Colligan

ANDROID

Attack Vectors

- Attacking the Android Device / OS
- Attacking the mobile application
- Attacking the mobile application’s backend/web api

Hack Demo Android Application

https://github.com/tcolligan/HackExample

Attacking the Android Device/OS

- What is adb?
- What is a rooted android device?
- shell@t0ltevzw:/ $ vs root@vbox86p:/ #

Into Terminal!

HackExample — bash — 80×24

```
androidhacktest
Thomass-MBP:Github ThomasColligan$ cd HackExample/
Thomass-MBP:HackExample ThomasColligan$ ls
HackExample             README.md
LICENSE                 hack-keystore.jks
Thomass-MBP:HackExample ThomasColligan$ clear




Thomass-MBP:HackExample ThomasColligan$ pwd
/Users/ThomasColligan/Desktop/Github/HackExample
Thomass-MBP:HackExample ThomasColligan$ ls -a
.                       .gitignore              LICENSE
..                      .idea                   README.md
.git                    HackExample             hack-keystore.jks
Thomass-MBP:HackExample ThomasColligan$ cat LICENSE
                    GNU GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.
```

Attacking the Mobile Application

- Retrieve the app's apk file
  - Download from website here
  - There is an app for that
  - Pull off of rooted android device with adb
- Decompile the apk
  - There is a website for that
  - Use open source command line tools, apktool, dex2jar, JAD java descompiler
  - Android code is compiled: .java -> .class -> .dex
- Dexguard and Proguard?

Into the Source Code!

```
public class LoginActivity extends AppCompatActivity
{

    private EditText passwordEditText;
    private EditText usernameEditText;

    public LoginActivity()
    {
    }

    public static Intent getIntent(Context context)
    {
        return new Intent(context, com/tcolligan/hackexample/activities/LoginActivity);
    }

    private boolean inputIsValid()
    {
        String s = usernameEditText.getText().toString();
        String s1 = passwordEditText.getText().toString();
        return !TextUtils.isEmpty(s) && !TextUtils.isEmpty(s1);
    }

    private void login(String s, String s1)
    {
        s1 = Utils.md5(s1);
        HashMap hashmap = NetworkingHelper.createPostDict();
        hashmap.put("username", s);
        hashmap.put("password", s1);
        (new PostRequestAsyncTask(hashmap, new com.tcolligan.hackexample.networking.PostRequestA

            final LoginActivity this$0;

            public void onInvalidResponse()
            {
                Toast.makeText(getApplicationContext(), 0x7f06001a, 0).show();
            }

            public void onResponseReceived(Response response)
            {
                Context context = getApplicationContext();
                if (!response.isError())
                {
                    User user = new User(response.getData());
                    UserManager.getInstance().saveNewUser(context, user);
                    startActivity(BuyStuffActivity.getIntent(LoginActivity.this));
                    finish();
                }
```

Summary

1. Applications running on rooted devices are much easier to hack into. Extremely difficult to secure applications on rooted devices.

2. Make sure you save all of your application data in the private application sandbox.

3. Be careful with how you structure your DevOptions screens, can be automatically launched on rooted devices.

4. Easy to download an apk and de-compile it, use proguard to obfuscate the code

5. Avoid putting api tokens in your source code, have them on your server behind a login process if possible, base64 can slow hackers down

Summary

6. Always use HTTPS to avoid mitma, be careful with code that accepts self signed SSL certs

7. Do your password hashing on the server, do not use md5 or sha1, sha256 or BCrypt, the slower the hashing algorithm is the better

8. Make sure your api endpoints are tamperproof and sanitize their inputs properly, prevent SQL Injection

9. Never set the price of items in the app's source code, client should be dumb