Name: Sadiq Sonalkar

Position: Cyber Security Intern

Application Name: Androgoat

INDEX

Vulnerability No.	Vulnerability Name	Vulnerability Level
1	Insecure Logging Security Issues	High
2	Sensitive Data Copied to Clipboard	Medium
3	Cryptographic Storage Issues	Critical
4	Insecure Sensitive Hardcoding Issues	Critical
5	Application-Level Denial-of-Service	Low
6	Insecure Protocols	High

Vulnerability Number: **1**

Vulnerability Name: Insecure Logging Security Issues

Vulnerability Critical Level: High

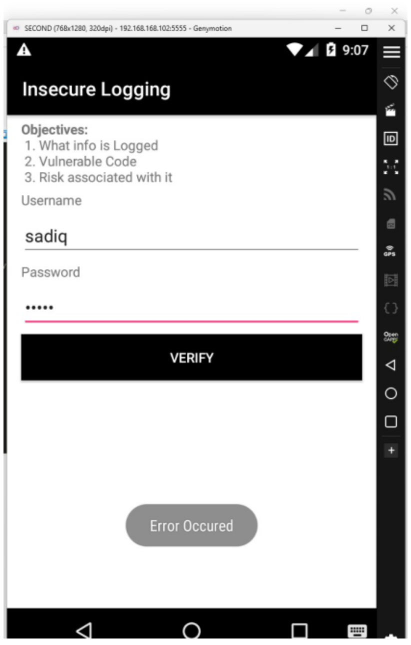
Vulnerable Application Name: AndroGoat

Vulnerability Description: Logging is a method that developers use for tracing the code and watching warnings or errors. Unfortunately, sometimes developers write sensitive data in logs. In such a situations attacker may gain access and read logs for stealing sensitive data.

Steps to reproduce:

1. Install AndroGoat in your Android.
2. Start Appie, and in appie type ‘pidcat’. Pidcat is already installed.
3. Pidcat is already started and it will store all the Log
4. Now go to your AndroGoat application and insert data into insecure logging. And pidcat will capture it.

Proof of Concept:

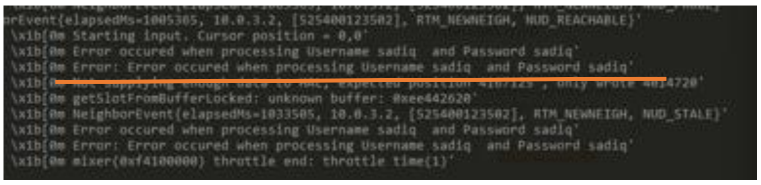


Insecure Logging

Objectives:
1. What info is Logged
2. Vulnerable Code
3. Risk associated with it

Username: sadiq

Password: •••••



```
rEvent{elapsedMs=1005305, 10.0.3.2, [525400123502], RTM_NEWNEIGH, NUD_REACHABLE}'
\x1b[0m Starting input. Cursor position = 0,0'
\x1b[0m Error occured when processing Username sadiq  and Password sadiq'
\x1b[0m Error: Error occured when processing Username sadiq  and Password sadiq'
\x1b[0m Not supplying enough data to HAL, expected position 4167125 , only wrote 4014720'
\x1b[0m getSlotFromBufferLocked: unknown buffer: 0xee442620'
\x1b[0m NeighborEvent{elapsedMs=1033505, 10.0.3.2, [525400123502], RTM_NEWNEIGH, NUD_STALE}'
\x1b[0m Error occured when processing Username sadiq  and Password sadiq'
\x1b[0m Error: Error occured when processing Username sadiq  and Password sadiq'
\x1b[0m mixer(0xf4100000) throttle end: throttle time(1)'
```

-Sadiq S.

How to Mitigate the Vulnerability:

1. Involve Security at every phase of SDLC when implement log function.
2. Implement a standard and centralized logging function.
3. Consistent monitoring and scan your log data.
4. Understand that data you are logging.

Vulnerability Number: 2

Vulnerability Name: Sensitive Data Copied to Clipboard

Vulnerability Critical Level: Medium

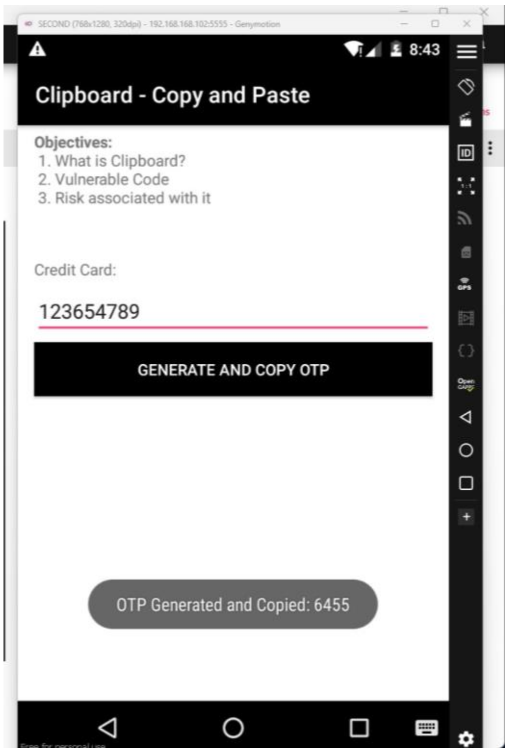
Vulnerable Application Name: AndroGoat

Vulnerability Description: Sensitive data may be stored, recoverable, or could be modified from the clipboard in clear text, regardless of whether the source of the data was initially encrypted. If it is in plaintext at the moment the user copies it, it will be in plaintext when other applications access the clipboard.

Steps to reproduce:

1. Start Appie, and in your AndroGoat application go to clipboard copy and paste.
2. In appie, type 'adb shell ps | grep owasp.sat.agoat'
3. We will get the UID, mine is u0_a52.
4. In Androgoat enter credit card no and generate OTP.
5. In appie, type 'adb shell su u0_a52 service call clipboard 2 s16 owasp.sat.agoat'.
6. Now in appie, we can see the same OTP copied to clipboard.

Proof of Concept:



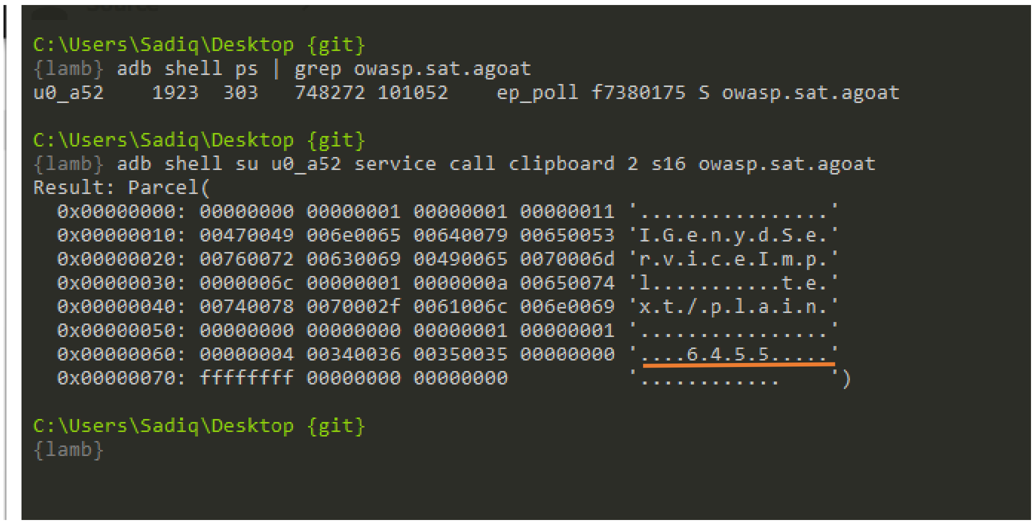
Clipboard - Copy and Paste

Objectives:
1. What is Clipboard?
2. Vulnerable Code
3. Risk associated with it

Credit Card:

123654789

OTP Generated and Copied: 6455



```
C:\Users\Sadiq\Desktop {git}
{lamb} adb shell ps | grep owasp.sat.agoat
u0_a52    1923  303   748272 101052    ep_poll f7380175 S owasp.sat.agoat

C:\Users\Sadiq\Desktop {git}
{lamb} adb shell su u0_a52 service call clipboard 2 s16 owasp.sat.agoat
Result: Parcel(
  0x00000000: 00000000 00000001 00000001 00000011 '................'
  0x00000010: 00470049 006e0065 00640079 00650053 'I.G.e.n.y.d.S.e.'
  0x00000020: 00760072 00630069 00490065 0070006d 'r.v.i.c.e.I.m.p.'
  0x00000030: 0000006c 00000001 0000000a 00650074 'l...........t.e.'
  0x00000040: 00740078 0070002f 0061006c 006e0069 'x.t./.p.l.a.i.n.'
  0x00000050: 00000000 00000000 00000001 00000001 '................'
  0x00000060: 00000004 00340036 00350035 00000000 '....6.4.5.5.....'
  0x00000070: ffffffff 00000000 00000000          '............    ')

C:\Users\Sadiq\Desktop {git}
{lamb}
```

How to Mitigate the Vulnerability:

1. Where appropriate, disable copy/paste for areas handling sensitive data.
2. Eliminating the option to copy can help avoid data exposure.
3. On Android the clipboard can be accessed by any application and so it is recommended that appropriately configured Content Providers be used to transfer complex sensitive data.

4. In addition, it can be interesting to clear the clipboard after taking the contents, to avoid other apps read them and leak what the user is doing.

Vulnerability Number: 3

Vulnerability Name: Cryptographic Storage Issues

Vulnerability Critical Level: Critical

Vulnerable Application Name: AndroGoat

Vulnerability Description: Insecure Cryptographic Storage vulnerability occurs when an application fails to encrypt sensitive data or encrypt data with poorly designed older cryptographic algorithms. This flaw can lead to sensitive information disclosure to attackers.

Steps to reproduce:

Part 1:

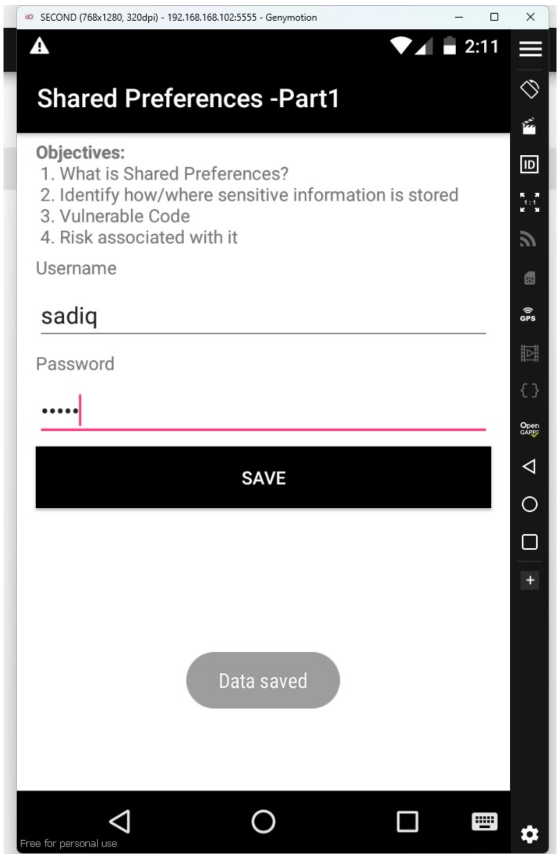
1. In appie type ‘adb shell’
2. Then type ‘cd data/data’
3. Using ‘ls’ we will get the package name
4. Now type ‘cd owasp.sat.agoat’
5. Now type ‘ls’ and then change the directory to shared preference using ‘cd shared_prefs’.
6. We can see users.xml. Open users.xml using ‘cat users.xml’.
7. Then we can see our username and password.

Part 2:

1. In appie type ‘adb shell’
2. Then type ‘cd data/data’
3. Using ‘ls’ we will get the package name
4. Now type ‘cd owasp.sat.agoat’
5. Now type ‘ls’ and then change the directory to shared preference using ‘cd shared_prefs’.
6. We can see score.xml. Open score.xml using ‘cat score.xml’.
7. Then we can see our Level as well as Score.

Proof of Concept:

Part 1:


SECOND (768x1280, 320dpi) - 192.168.168.102:5555 - Genymotion
2:11
Shared Preferences -Part1

Objectives:
1. What is Shared Preferences?
2. Identify how/where sensitive information is stored
3. Vulnerable Code
4. Risk associated with it

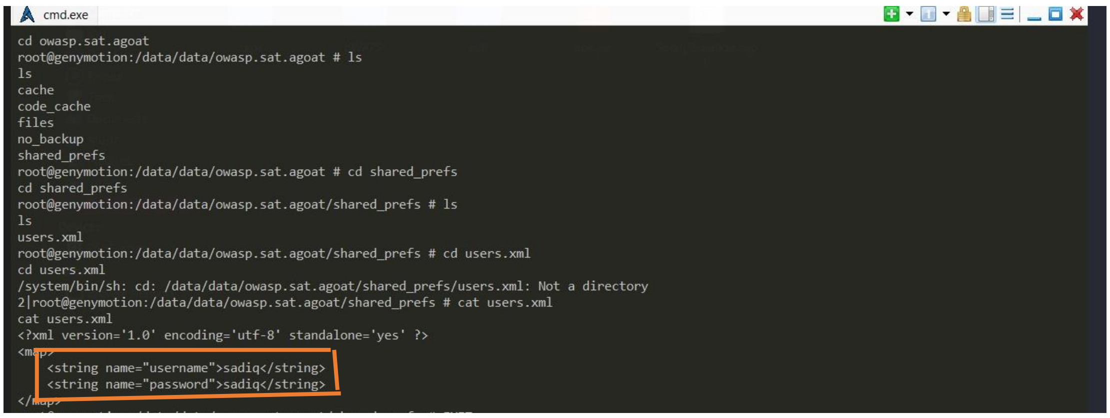
Username

sadiq

Password

•••••

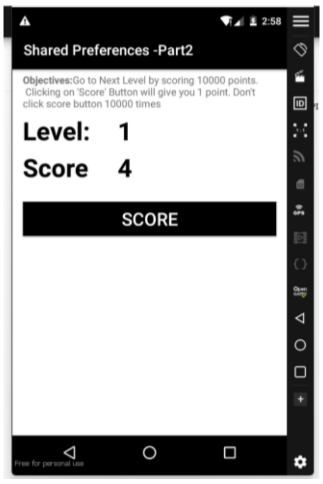
SAVE
Data saved
Free for personal use



```
cd owasp.sat.agoat
root@genymotion:/data/data/owasp.sat.agoat # ls
ls
cache
code_cache
files
no_backup
shared_prefs
root@genymotion:/data/data/owasp.sat.agoat # cd shared_prefs
cd shared_prefs
root@genymotion:/data/data/owasp.sat.agoat/shared_prefs # ls
ls
users.xml
root@genymotion:/data/data/owasp.sat.agoat/shared_prefs # cd users.xml
cd users.xml
/system/bin/sh: cd: /data/data/owasp.sat.agoat/shared_prefs/users.xml: Not a directory
2|root@genymotion:/data/data/owasp.sat.agoat/shared_prefs # cat users.xml
cat users.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="username">sadiq</string>
    <string name="password">sadiq</string>
</map>
```

-Sadiq S.

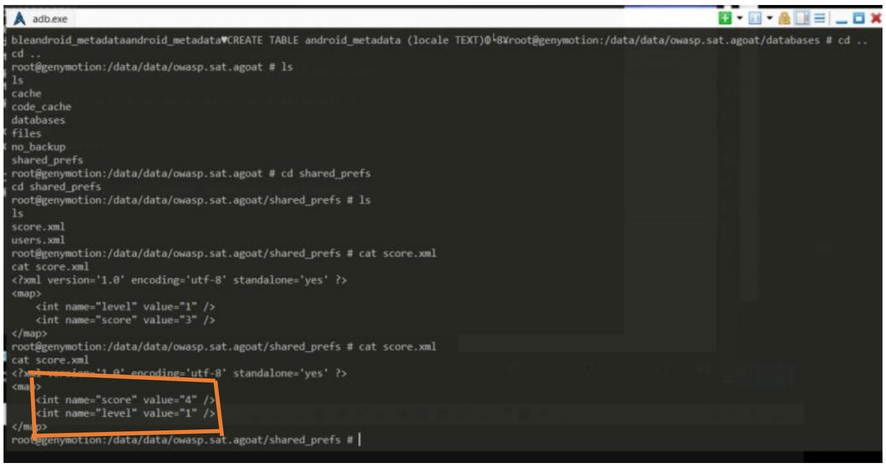
Part 2:


Shared Preferences -Part2

Objectives:Go to Next Level by scoring 10000 points. Clicking on 'Score' Button will give you 1 point. Don't click score button 10000 times

Level: 1

Score 4



```
bleandroid_metadataandroid_metadata♥CREATE TABLE android_metadata (locale TEXT)0└8¥root@genymotion:/data/data/owasp.sat.agoat/databases # cd ..
cd ..
root@genymotion:/data/data/owasp.sat.agoat # ls
ls
cache
code_cache
databases
files
no_backup
shared_prefs
root@genymotion:/data/data/owasp.sat.agoat # cd shared_prefs
cd shared_prefs
root@genymotion:/data/data/owasp.sat.agoat/shared_prefs # ls
ls
score.xml
users.xml
root@genymotion:/data/data/owasp.sat.agoat/shared_prefs # cat score.xml
cat score.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <int name="level" value="1" />
    <int name="score" value="3" />
</map>
root@genymotion:/data/data/owasp.sat.agoat/shared_prefs # cat score.xml
cat score.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <int name="score" value="4" />
    <int name="level" value="1" />
</map>
root@genymotion:/data/data/owasp.sat.agoat/shared_prefs # |
```

How to Mitigate the Vulnerability:

1. Make your application secure and try to patch common vulnerabilities.
2. Store Sensitive data only when needed.
3. Only use Widely Accepted Cryptographic Algorithms.
4. Use secure way of Password encryption to avoid risks.
5. Generate keys offline and never transmit over insecure channels.
6. Ensure Offsite Backups are encrypted.

Vulnerability Number: 4

Vulnerability Name: Insecure Sensitive Hardcoding Issues

Vulnerability Critical Level: Critical

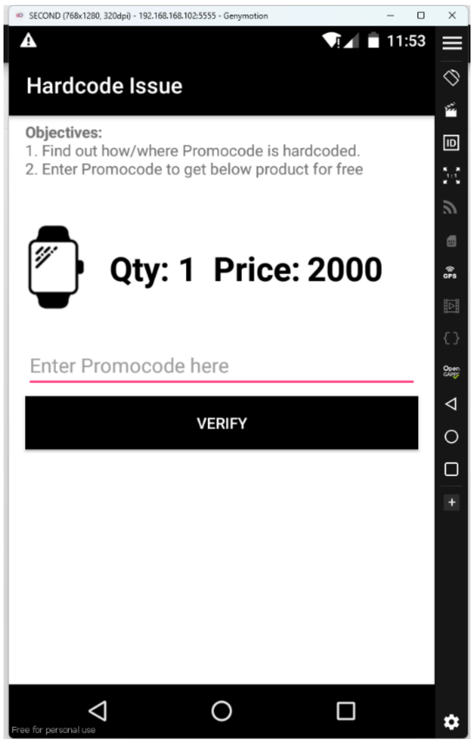
Vulnerable Application Name: AndroGoat

Vulnerability Description: Hard coding sensitive information, such as passwords, server IP addresses, and encryption keys can expose the information to attackers. Anyone who has access to the class files can decompile them and discover the sensitive information. Leaking data can also have legal consequences. Consequently, programs must not hard code sensitive information.

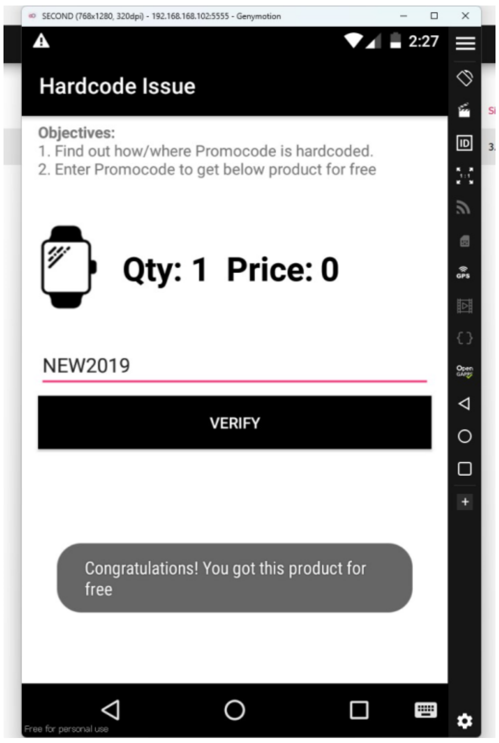
Steps to reproduce:

1. Using ApkTool we will decompile the application.
2. And we can go to 'C:\Appie\bin\adt\sdk\platform-tools\AndroGoat\res\values' and we can go in strings and get hard coded information.
3. And to get the source code we will use dex2jar and jd gui.
4. We will put our apk into dex2jar folder. And open cmd in that folder.
5. Then type: 'd2j-dex2jar.bat AndroGoat.apk'. And we will get a jar file.
6. And we will open the jar file in JD-gui. And we will get the source code.
7. There we can see our package name 'owasp.sat.agoat'. In that folder we can see HardCodeActivity.class. Inside that class file we can get the promocode as 'NEW2019'.

Proof of Concept:



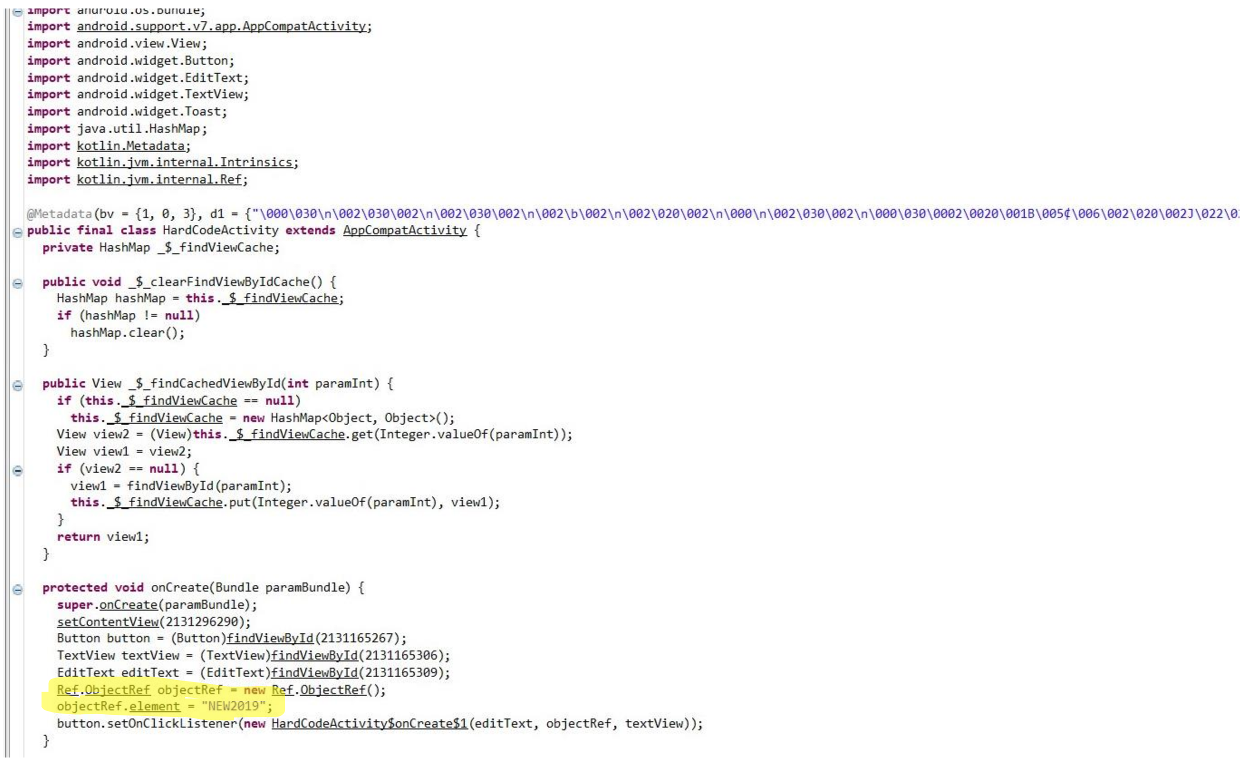
Objectives:
1. Find out how/where Promocode is hardcoded.
2. Enter Promocode to get below product for free



Objectives:
1. Find out how/where Promocode is hardcoded.
2. Enter Promocode to get below product for free

Congratulations! You got this product for free

-Sadiq S.



```
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import java.util.HashMap;
import kotlin.Metadata;
import kotlin.jvm.internal.Intrinsics;
import kotlin.jvm.internal.Ref;

@Metadata(bv = {1, 0, 3}, d1 = {"\000\030\n\002\030\002\n\002\030\002\n\002\b\002\n\002\020\002\n\000\n\002\030\002\n\000\030\0002\0020\001B\005¢\006\002\020\002J\022\0
public final class HardCodeActivity extends AppCompatActivity {
  private HashMap _$_findViewCache;

  public void _$_clearFindViewByIdCache() {
    HashMap hashMap = this._$_findViewCache;
    if (hashMap != null)
      hashMap.clear();
  }

  public View _$_findCachedViewById(int paramInt) {
    if (this._$_findViewCache == null)
      this._$_findViewCache = new HashMap<Object, Object>();
    View view2 = (View)this._$_findViewCache.get(Integer.valueOf(paramInt));
    View view1 = view2;
    if (view2 == null) {
      view1 = findViewById(paramInt);
      this._$_findViewCache.put(Integer.valueOf(paramInt), view1);
    }
    return view1;
  }

  protected void onCreate(Bundle paramBundle) {
    super.onCreate(paramBundle);
    setContentView(2131296290);
    Button button = (Button)findViewById(2131165267);
    TextView textView = (TextView)findViewById(2131165306);
    EditText editText = (EditText)findViewById(2131165309);
    Ref.ObjectRef objectRef = new Ref.ObjectRef();
    objectRef.element = "NEW2019";
    button.setOnClickListener(new HardCodeActivity$onCreate$1(editText, objectRef, textView));
  }
```

How to Mitigate the Vulnerability:

1. Developer should not hardcode sensitive information.
2. Developers should prevent from leaking data in the source code.
3. If hardcoding is needed, use cryptography to encrypt that information

Vulnerability Number: 5

Vulnerability Name: Application-Level Denial-of-Service

Vulnerability Critical Level: Low

Vulnerable Application Name: AndroGoat

Vulnerability Description: Application layer DoS attacks are designed to attack the application itself, focusing on specific vulnerabilities or issues, resulting in the application not being able to deliver content to the user.

Steps to reproduce:

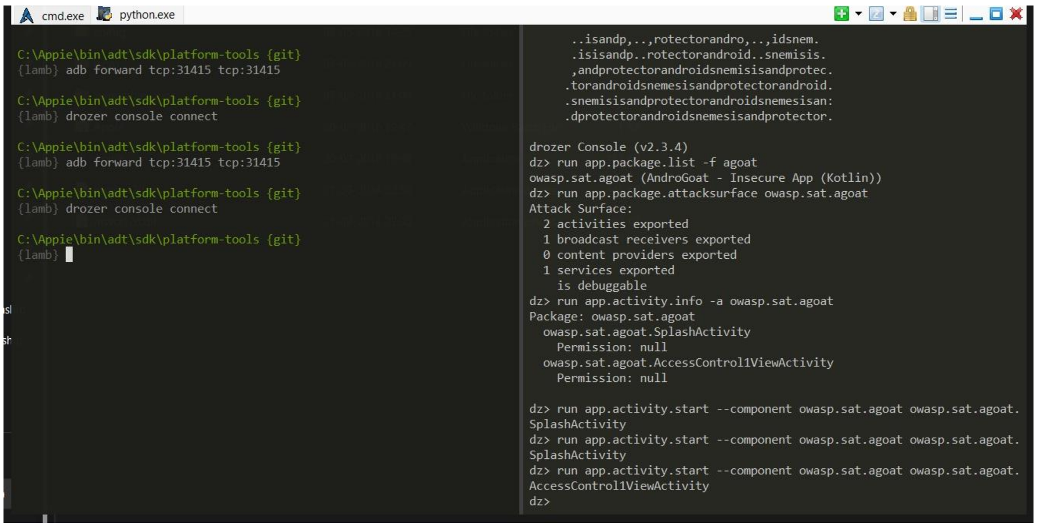
1. Install drozer agent in android.
2. In appie type ‘adb forward tcp:31415 tcp:31415’
3. Then type ‘drozer console connect’

This will start a drozer console.

4. In drozer console type ‘run app.package.list -f agoat’ – This will give us package name.
5. Type ‘run app.package.attacksurface owasp.sat.agoat’ – This will return attacksurface.
6. We can see there are 2 Activities and 1 service exported.
7. First, we will exploit application using activity. Type ‘run app.activity.info -a owasp.sat.agoat’- This will return the activities name.
8. To exploit this type ‘run.app.activity.start --component owasp.sat.agoat owasp.sat.agoat.SplashActivity’ – This will crash the application and show splash screen.
9. Type ‘run.app.activity.start --component owasp.sat.agoat owasp.sat.agoat.AccessControlViewActivity’ – This will crash the application and take us to unprotected Android Component.
10. After using activity we will use the service to exploit. Type ‘run app.service.info -a owasp.sat.agoat’ - This will return the service name.
11. To exploit this type ‘run.app.service.start --component owasp.sat.agoat owasp.sat.agoat.DownloadInvoiceService’ – This will crash the application and it will create a service first then it will download an invoice.

Proof of Concept:

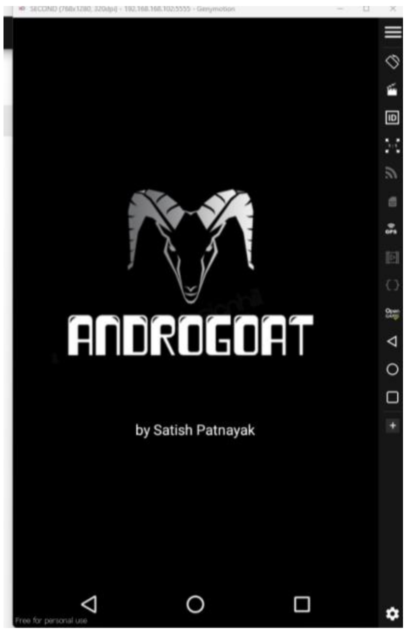
For activity:



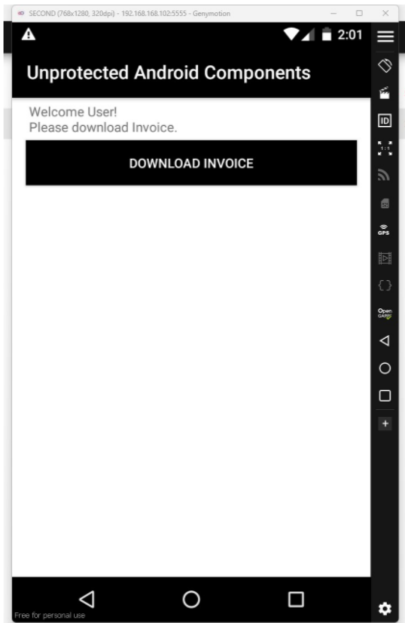
```
C:\Appie\bin\adt\sdk\platform-tools {git}
{lamb} adb forward tcp:31415 tcp:31415

C:\Appie\bin\adt\sdk\platform-tools {git}
{lamb} drozer console connect

C:\Appie\bin\adt\sdk\platform-tools {git}
{lamb} adb forward tcp:31415 tcp:31415

C:\Appie\bin\adt\sdk\platform-tools {git}
{lamb} drozer console connect

C:\Appie\bin\adt\sdk\platform-tools {git}
{lamb}
```

```
       ..isandp,..,rotectorandro,..,idsnem.
       .isisandp..rotectorandroid..snemisis.
       ,andprotectorandroidsnemisisandprotec.
      .torandroidsnemesisandprotectorandroid.
      .snemisisandprotectorandroidsnemesisan:
      .dprotectorandroidsnemesisandprotector.

drozer Console (v2.3.4)
dz> run app.package.list -f agoat
owasp.sat.agoat (AndroGoat - Insecure App (Kotlin))
dz> run app.package.attacksurface owasp.sat.agoat
Attack Surface:
  2 activities exported
  1 broadcast receivers exported
  0 content providers exported
  1 services exported
    is debuggable
dz> run app.activity.info -a owasp.sat.agoat
Package: owasp.sat.agoat
  owasp.sat.agoat.SplashActivity
    Permission: null
  owasp.sat.agoat.AccessControl1ViewActivity
    Permission: null

dz> run app.activity.start --component owasp.sat.agoat owasp.sat.agoat.
SplashActivity
dz> run app.activity.start --component owasp.sat.agoat owasp.sat.agoat.
SplashActivity
dz> run app.activity.start --component owasp.sat.agoat owasp.sat.agoat.
AccessControl1ViewActivity
dz>
```

Splash Screen:

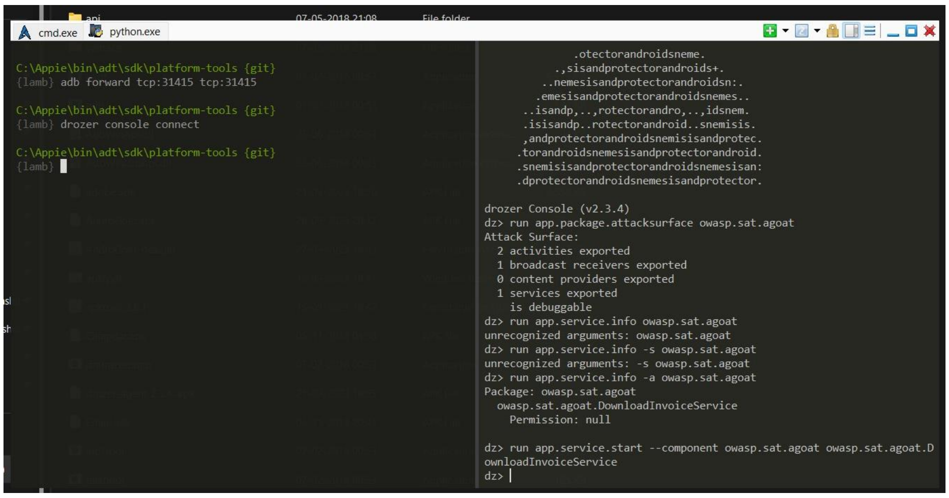


Access Control View Activity:

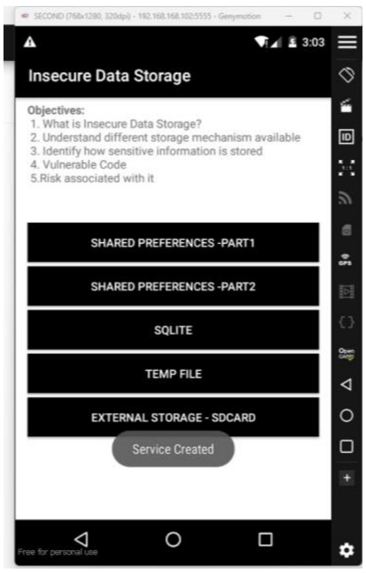

Unprotected Android Components

Welcome User!
Please download Invoice.

For service:

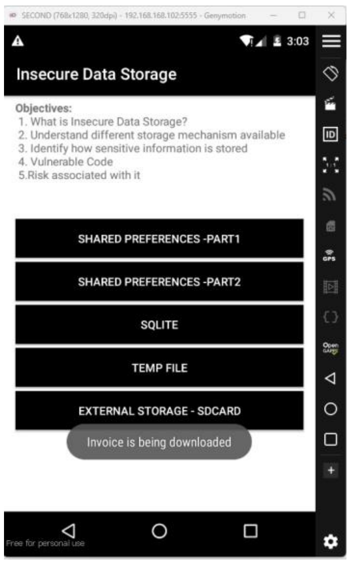

```
C:\Appie\bin\adt\sdk\platform-tools {git}
{lamb} adb forward tcp:31415 tcp:31415

C:\Appie\bin\adt\sdk\platform-tools {git}
{lamb} drozer console connect

C:\Appie\bin\adt\sdk\platform-tools {git}
{lamb}
```

```
              .otectorandroidsneme.
           .,sisandprotectorandroids+.
         ..nemesisandprotectorandroidsn:.
        .emesisandprotectorandroidsnemes..
      ..isandp,..,rotectorandro,..,idsnem.
      .isisandp..rotectorandroid..snemisis.
      ,andprotectorandroidsnemisisandprotec.
     .torandroidsnemesisandprotectorandroid.
     .snemisisandprotectorandroidsnemesisan:
     .dprotectorandroidsnemesisandprotector.

drozer Console (v2.3.4)
dz> run app.package.attacksurface owasp.sat.agoat
Attack Surface:
  2 activities exported
  1 broadcast receivers exported
  0 content providers exported
  1 services exported
    is debuggable
dz> run app.service.info owasp.sat.agoat
unrecognized arguments: owasp.sat.agoat
dz> run app.service.info -s owasp.sat.agoat
unrecognized arguments: -s owasp.sat.agoat
dz> run app.service.info -a owasp.sat.agoat
Package: owasp.sat.agoat
  owasp.sat.agoat.DownloadInvoiceService
    Permission: null

dz> run app.service.start --component owasp.sat.agoat owasp.sat.agoat.D
ownloadInvoiceService
dz>
```


Insecure Data Storage

Objectives:
1. What is Insecure Data Storage?
2. Understand different storage mechanism available
3. Identify how sensitive information is stored
4. Vulnerable Code
5.Risk associated with it

SHARED PREFERENCES -PART1
SHARED PREFERENCES -PART2
SQLITE
TEMP FILE
EXTERNAL STORAGE - SDCARD
Service Created


Insecure Data Storage

Objectives:
1. What is Insecure Data Storage?
2. Understand different storage mechanism available
3. Identify how sensitive information is stored
4. Vulnerable Code
5.Risk associated with it

SHARED PREFERENCES -PART1
SHARED PREFERENCES -PART2
SQLITE
TEMP FILE
EXTERNAL STORAGE - SDCARD
Invoice is being downloaded

-Sadiq S.

How to Mitigate the Vulnerability:

1. Use flow telemetry analysis supplemented with behavioral analysis to detect abnormalities and attacks. Focus on understanding what is normal. This will simplify the identification of abnormalities.
2. Use an IDMS to detect abnormal behavior and application layer attacks that require advanced and active mitigation; and using this approach in conjunction with BGP FlowSpec Offload when and where appropriate.

Vulnerability Number: 6

Vulnerability Name: Insecure Protocols

Vulnerability Critical Level: High

Vulnerable Application Name: AndroGoat

Vulnerability Description: A protocol, service, or port that introduces security concerns due to the lack of controls over confidentiality and/or integrity.

These security concerns include services, protocols, or ports that transmit data or authentication credentials (for example, password/passphrase) in clear-text over the Internet, or that easily allow for exploitation by default or if misconfigured.

Examples of insecure services, protocols, or ports include but are not limited to FTP, Telnet, POP3, IMAP, and SNMP v1 and v2.

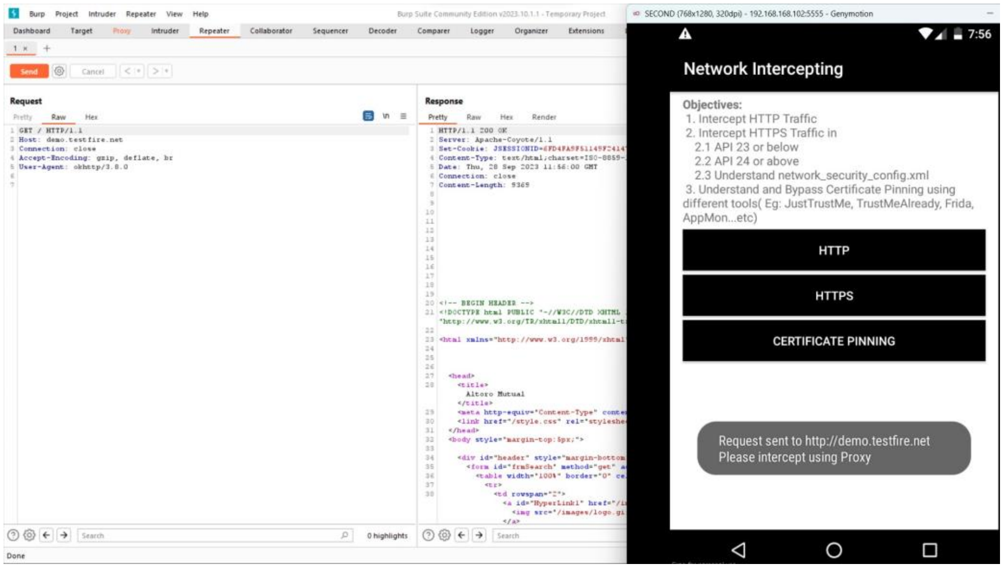
Steps to reproduce:

1. Just Connect Burpsuite proxy to our android. Once connected start the intercept. It will intercept each and every request coming from our android.
2. Now in androgoat application go to Network intercept. And click on http. It will send a http request using demo.testfire.net. And we can see burpsuite have intercept it. Also if there was any password in the request we could have seen it in clear text.

3. Now click on https. It will send a https request using owasp.org. And we can see burpsuite have intercept it. Also if there was any password in the request we could have seen it in clear text.

Proof of Concept:

HTTP request:



Burp Suite Community Edition v2023.10.1.1 - Temporary Project

Request

```
GET / HTTP/1.1
Host: demo.testfire.net
Connection: close
Accept-Encoding: gzip, deflate, br
User-Agent: okhttp/3.8.0
```

Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONID=6FD4FA9F51149F2414
Content-Type: text/html;charset=ISO-8859-
Date: Thu, 28 Sep 2023 11:56:00 GMT
Connection: close
Content-Length: 9369

<!-- BEGIN HEADER -->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-t

<html xmlns="http://www.w3.org/1999/xhtml

  <head>
    <title>
      Altoro Mutual
    </title>
    <meta http-equiv="Content-Type" conte
    <link href="/style.css" rel="styleshe
  </head>
  <body style="margin-top:5px;">

    <div id="header" style="margin-bottom
      <form id="frmSearch" method="get" a
        <table width="100%" border="0" ce
          <tr>
            <td rowspan="2">
              <a id="HyperLink1" href="/i
                <img src="/images/logo.gi
              </a>
```

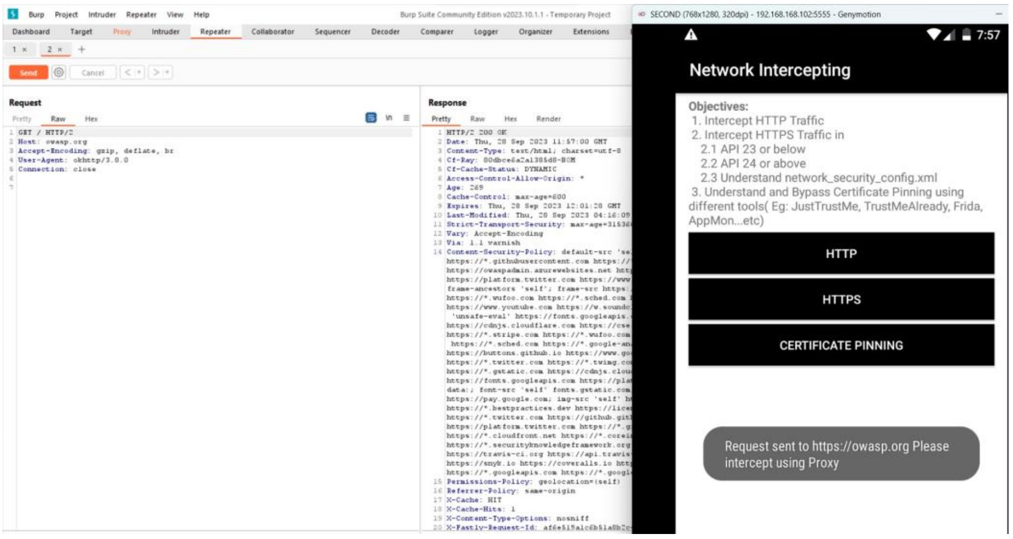
SECOND (768x1280, 320dpi) - 192.168.168.102:5555 - Genymotion

Network Intercepting

Objectives:
1. Intercept HTTP Traffic
2. Intercept HTTPS Traffic in
2.1 API 23 or below
2.2 API 24 or above
2.3 Understand network_security_config.xml
3. Understand and Bypass Certificate Pinning using different tools(Eg: JustTrustMe, TrustMeAlready, Frida, AppMon...etc)

Request sent to http://demo.testfire.net Please intercept using Proxy

HTTPS request:



Burp Suite Community Edition v2023.10.1.1 - Temporary Project

Request

```
GET / HTTP/2
Host: owasp.org
Accept-Encoding: gzip, deflate, br
User-Agent: okhttp/3.8.0
Connection: close
```

Response

```
HTTP/2 200 OK
Date: Thu, 28 Sep 2023 11:57:00 GMT
Content-Type: text/html; charset=utf-8
Cf-Ray: 80dbce6a2a1385d8-BOM
Cf-Cache-Status: DYNAMIC
Access-Control-Allow-Origin: *
Age: 289
Cache-Control: max-age=600
Expires: Thu, 28 Sep 2023 12:01:28 GMT
Last-Modified: Thu, 28 Sep 2023 04:16:09
Strict-Transport-Security: max-age=31536
Vary: Accept-Encoding
Via: 1.1 varnish
Content-Security-Policy: default-src 'se
https://*.githubusercontent.com https://
https://owaspadmin.azurewebsites.net htt
https://platform.twitter.com https://www
frame-ancestors 'self'; frame-src https:
https://*.wufoo.com https://*.sched.com
https://www.youtube.com https://w.soundc
 'unsafe-eval' https://fonts.googleapis.
https://cdnjs.cloudflare.com https://cse
https://*.stripe.com https://*.wufoo.com
 https://*.sched.com https://*.google-an
https://buttons.github.io https://www.go
https://*.twitter.com https://*.twimg.co
https://*.gstatic.com https://cdnjs.clou
https://fonts.googleapis.com https://pla
data:; font-src 'self' fonts.gstatic.com
https://pay.google.com; img-src 'self' h
https://*.bestpractices.dev https://lice
https://*.twitter.com https://github.git
https://platform.twitter.com https://*.g
https://*.cloudfront.net https://*.corei
https://*.securityknowledgeframework.org
https://travis-ci.org https://api.travis
https://snyk.io https://coveralls.io htt
https://*.googleapis.com https://*.googl
Permissions-Policy: geolocation=(self)
Referrer-Policy: same-origin
X-Cache: HIT
X-Cache-Hits: 1
X-Content-Type-Options: nosniff
X-Fastly-Request-Id: af6e519a1c6b51a8b2c
```

SECOND (768x1280, 320dpi) - 192.168.168.102:5555 - Genymotion

Network Intercepting

Objectives:
1. Intercept HTTP Traffic
2. Intercept HTTPS Traffic in
2.1 API 23 or below
2.2 API 24 or above
2.3 Understand network_security_config.xml
3. Understand and Bypass Certificate Pinning using different tools(Eg: JustTrustMe, TrustMeAlready, Frida, AppMon...etc)

Request sent to https://owasp.org Please intercept using Proxy

How to Mitigate the Vulnerability:

1. The first and most obvious is to find devices running services like Telnet and SNMPv1/2, and to replace them with a more secure option.
2. Locating devices like these is easy with a network operations system that keeps track of which ports and services are running on which devices.
3. Once you've identified the insecure devices, replace Telnet with SSH-2, and upgrade installations of SNMPv1 and SNMPv2 with SNMPv3, which is much more secure than its predecessors.