Name: Onkar vikas mhaskar Email: onkarvmhaskar@gmail.com Contact No: 9623918184

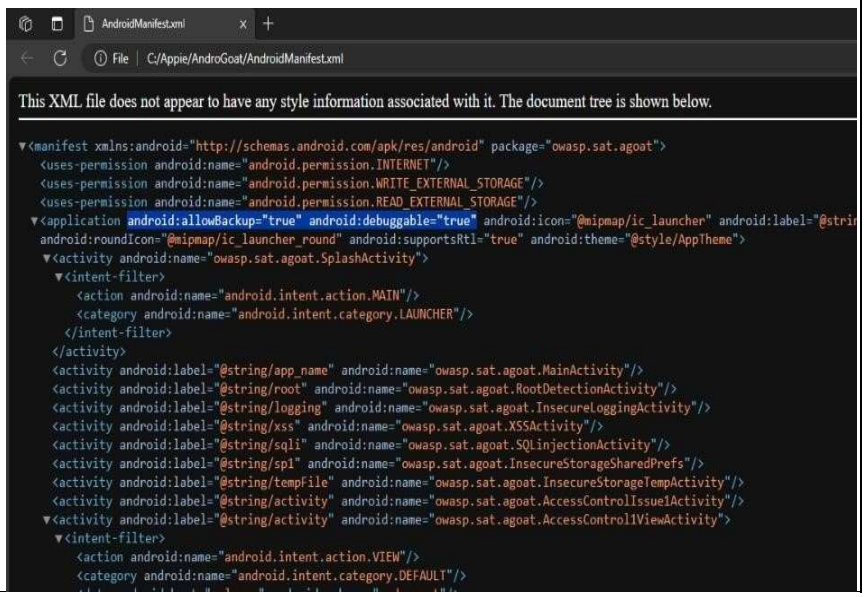
Overview:

This report outlies the project for finding vulnerabilities in androgoat android application and I have done this by watching walkthroughs, youtube and recordings of sessions.

Objective: To find vulnerabilities of given application.

INDEX

Vulnerability Details

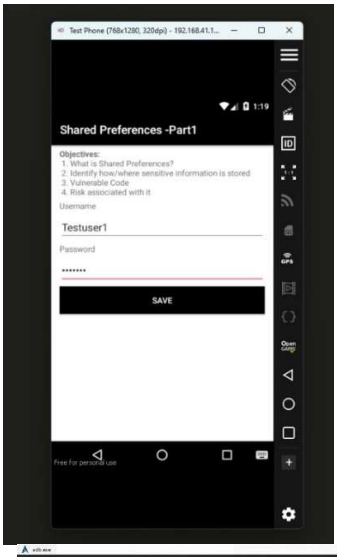
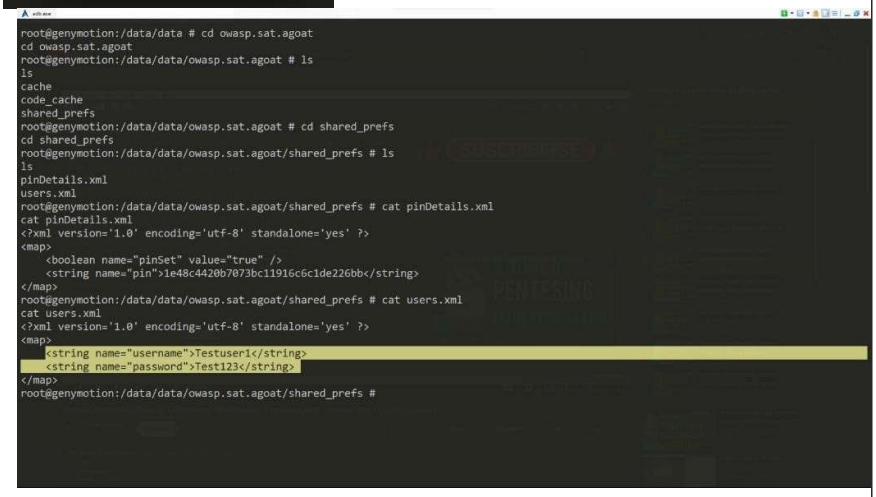
Location of Vulnerability	AndroGoat Android Application
Name of Vulnerability	Flags: allowBackup and debuggable
Proof of Concept and Steps of verification of vulnerability with Screen Shots	Steps: 1. open appie Type this - Apktool d AndroGate.apk 2. Now go to its folder and check AndroidManifest.xml The android:allowBackup attribute defines whether application data can be backed up and restored by a user who has enabled usb debugging. If backup flag is set to true, it allows an attacker to take the backup of the application data via adb even if the device is not rooted. The android:debuggable attribute defines whether the application can be debugged or not. If an application is marked as debuggable then an attacker can access the application data by assuming the privileges of that application and can even run arbitrary code under that application 

	permission. In the case of non-debuggable application, attacker would first need to root the device to extract any data.
Solution	References: https://source.android.com/docs/security/overview/appsecurity#the-android-permission-model-accessing-protected-apis http://developer.android.com/guide/topics/manifest/application-element.html#allowbackup

Location of Vulnerability	AndroGoat Android Application
Name of Vulnerability	Unprotected Android Components

Proof of Concept and Steps of verification of vulnerability with Screen Shots	Steps: 1. Open appie 2. Type adb shell 3. Change your directory to /data/data/ 4. Type am start -n owasp.sat.agoat/.AccessControl1ViewActivity 
Solution	References: https://source.android.com/docs/security/overview/appsecurity#the-android-permission-model-accessing-protected-apis https://blog.c22.cc/advisories/cve-2013-5112-evernote-androidinsecure-storage-of-pin-data-bypass-of-pin-protection/

Location of Vulnerability	AndroGoat Android Application
Name of Vulnerability	Insecure Data Storage – Shared Preferences Part 1

Proof of Concept and Steps of verification of vulnerability with Screen Shots	Steps: 1. Open appie and Type adb shell 2. Type cd /data/data/owasp.sat.agoat/shared_prefs 3. cat users.xml  
Solution	Secure Storage: Encrypt credentials. Strong Authentication: Implement MFA. Regular Audits: Assess and fix vulnerabilities. Data Encryption: Encrypt data in transit and at rest. Compliance Check: Ensure regulatory compliance. User Alert: Notify users and advise password changes.

Location of Vulnerability	AndroGoat Android Application
Name of Vulnerability	Insecure Data Storage – Shared Preferences Part 2

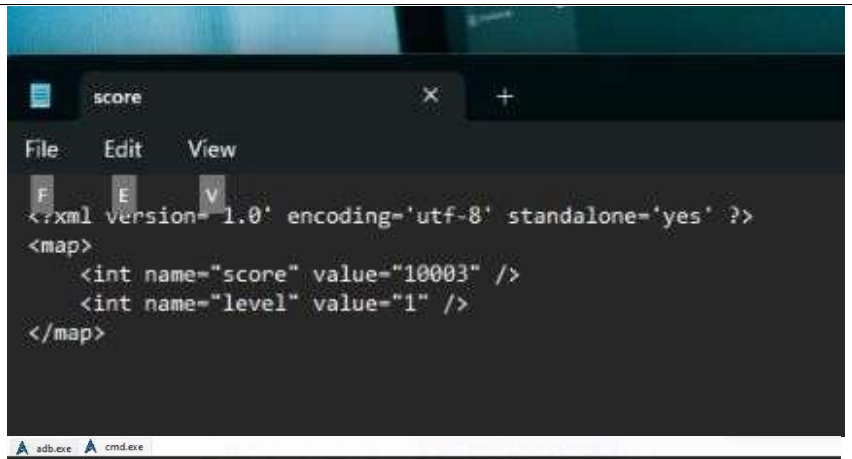
Proof of Concept and Steps of verification of vulnerability with Screen Shots	Steps: 1. 2. Open appie 3. Type adb shell 4. Type cd /data/data/owasp.sat.agoat/shared_prefs Type the ls command and there will be new file present there named score.xml. 5. Now edit the score.xml file either by exporting it or by editing it in the location and values will be reflected on the activity page.

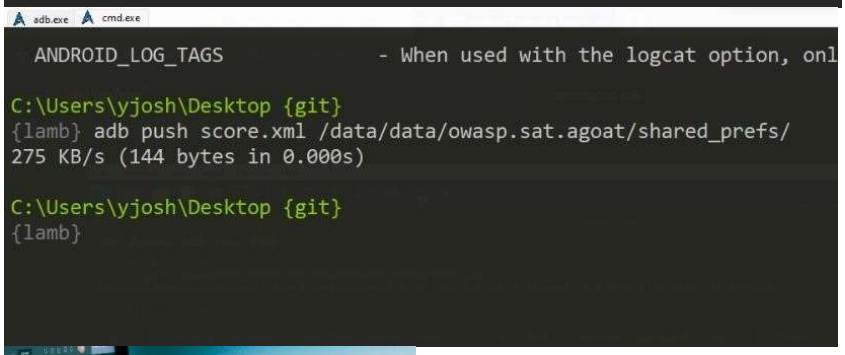
adb.exe

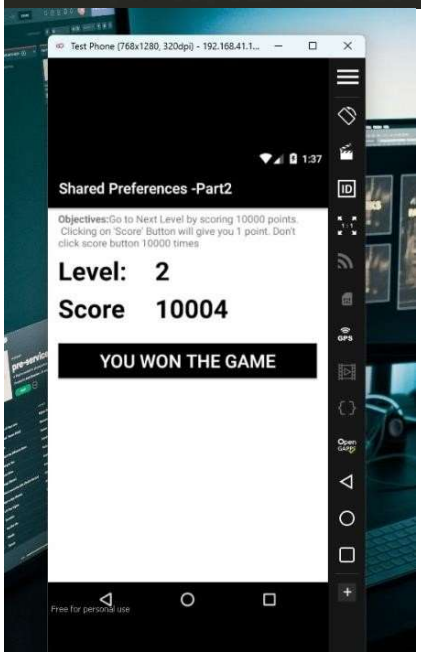
```
root@genymotion:/data/data # cd owasp.sat.agoat
cd owasp.sat.agoat
root@genymotion:/data/data/owasp.sat.agoat # ls
ls
cache
code_cache
shared_prefs
root@genymotion:/data/data/owasp.sat.agoat # cd shared_prefs
cd shared_prefs
root@genymotion:/data/data/owasp.sat.agoat/shared_prefs # ls
ls
pinDetails.xml
score.xml
users.xml
root@genymotion:/data/data/owasp.sat.agoat/shared_prefs #
```

adb.exe cmd.exe

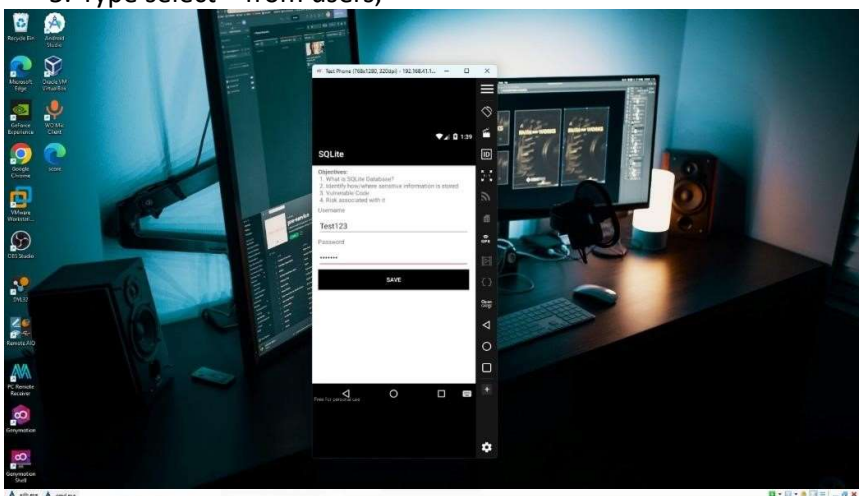
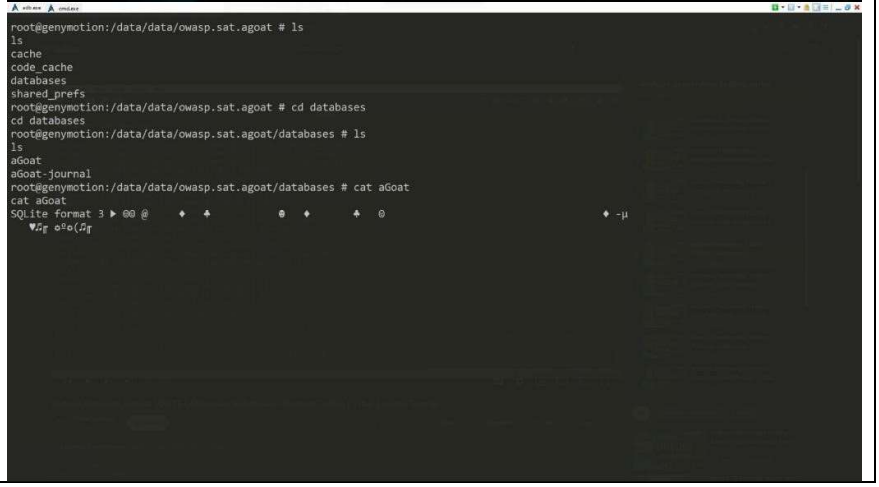
```
C:\Users\yjosh\Desktop {git}
{lamb} adb pull /data/data/owasp.sat.goat/shared_prefs/score.xml
remote object '/data/data/owasp.sat.goat/shared_prefs/score.xml' does not exist

C:\Users\yjosh\Desktop {git}
{lamb} adb pull /data/data/owasp.sat.goat/shared_prefs/score.xml
remote object '/data/data/owasp.sat.goat/shared_prefs/score.xml' does not exist

C:\Users\yjosh\Desktop {git}
{lamb} adb pull /data/data/owasp.sat.agoat/shared_prefs/score.xml
134 KB/s (140 bytes in 0.001s)

C:\Users\yjosh\Desktop {git}
{lamb} ls
Android Studio.lnk*  WO Mic Client.lnk*  desktop.ini  score.xml

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

<table>
<tr>
<td></td>
<td>




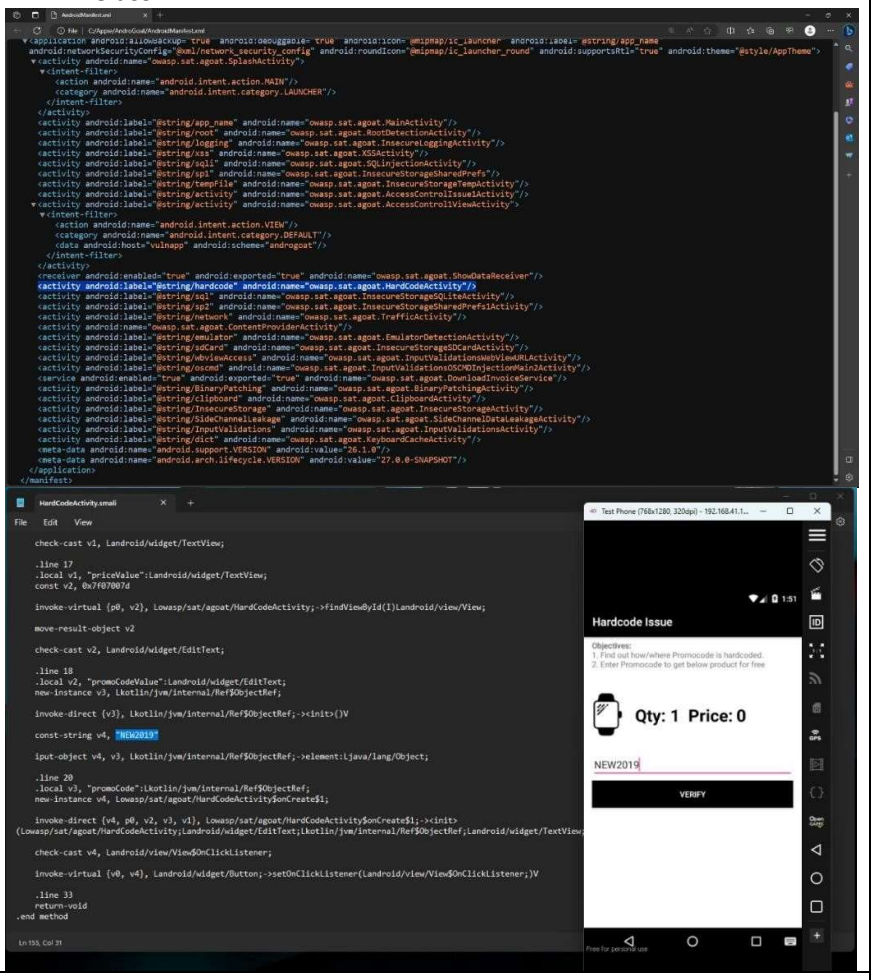


</td>
</tr>
<tr>
<td>Solution</td>
<td>Secure Data Storage: Implement data encryption or hashing to protect game data.
Input Validation: Verify data legitimacy through server-side validation.
Authentication and Authorization: Restrict data modification to authorized users.</td>
</tr>
</table>

Location of Vulnerability	AndroGoat Android Application
Name of Vulnerability	Insecure Data Storage - SQLite
Proof of Concept and Steps of verification of vulnerability with Screen Shots	Steps: 1. 2. Open appie and Type adb shell 3. Now go to cd /data/data/owasp.sat.agoat/databases 4. Type sqlite3 aGoat Type .tables 5. Type select * from users;   root@genymotion:/data/data/owasp.sat.agoat # ls ls cache code_cache databases shared_prefs root@genymotion:/data/data/owasp.sat.agoat # cd databases cd databases root@genymotion:/data/data/owasp.sat.agoat/databases # ls ls aGoat aGoat-journal root@genymotion:/data/data/owasp.sat.agoat/databases # cat aGoat cat aGoat SQLite format 3 ▶ @@ @ ♦ ♣ ☻ ♦ ♣ ☺ ♦ -µ ♥♫╦ ☼¤☼(♫╦

	adb.exe root@genymotion:/data/data/owasp.sat.agoat # ls ls cache code_cache databases shared_prefs root@genymotion:/data/data/owasp.sat.agoat # cd databases cd databases root@genymotion:/data/data/owasp.sat.agoat/databases # ls ls aGoat aGoat-journal root@genymotion:/data/data/owasp.sat.agoat/databases # sqlite3 aGoat sqlite3 aGoat SQLite version 3.8.10.2 2015-05-20 18:17:19 Enter ".help" for usage hints. sqlite> .tables .tables android_metadata users sqlite> select * from users select * from users ...> ; ; 1|Test123|Test123 sqlite>
Solution	Use SQLCipher or similar libraries to add encryption capabilities to SQLite or encrypt the sensitive data using cryptographically secure algorithms before storing it in the database.

Location of Vulnerability	AndroGoat Android Application

Name of Vulnerability	HardCode Issue
Proof of Concept and Steps of verification of vulnerability with Screen Shots	Steps: 1. open appie Type this - Apktool d AndroGate.apk 2. Now go smali\owasp\sat\agoat and check HardCodeActivity Class 
Solution	To mitigate this issue, consider using the KeyChain API when you want system-wide credentials, or the Android Keystore provider to let an individual app store its own credentials that only the app itself can access. Refrences: https://developer.android.com/training/articles/keystore https://developer.android.com/reference/java/security/KeyStore https://developer.android.com/reference/android/security/KeyChain https://cwe.mitre.org/data/definitions/321.html

Conclusion: I have learned vapt from this project and pentesting too and special thanks to cyberhost to give this opportunity and project as well.