WHITE PAPER

Trusting Certificates in Android Nougat and Above

Make mobile application penetration testing work for you

WHITE PAPER

Trusting Certificates in Android Nougat and Above

Make mobile application penetration testing work for you

As mobile device use increases dramatically worldwide, hackers are targeting mobile applications with their exploits. This has emerged as a big challenge for developers working to create secure mobile applications.

Mobile platform developers have responded to the challenge by implementing their own security best practices. Google, for example, has changed some key security features in the Android platform, notably how Android handles trusted certificate authorities.[1]

Bottom line: Android applications that target applications programming interface (API) Level 24 and above will not trust user- or administrator-added certificate authorities (CA) for secure connections. This change makes it more difficult to conduct mobile application penetration testing.

About the Author

Deepak Choudhary
Choudhary is a senior security consultant at McAfee and has been working in the security industry since 2009. He specializes in application, mobile, and network penetration testing. His responsibilities span web services, PCI-DSS, and most services under the McAfee® information security umbrella. Along with applying his advanced technical skills, he is instrumental in training new team members to foster smooth team transitions. Choudhary earned a bachelor's degree in electronics and communication and currently holds the Certified Ethical Hacker (CEH) certification.

Connect With Us

What Has Been Uncovered?

How was this done before Google changed these security features?

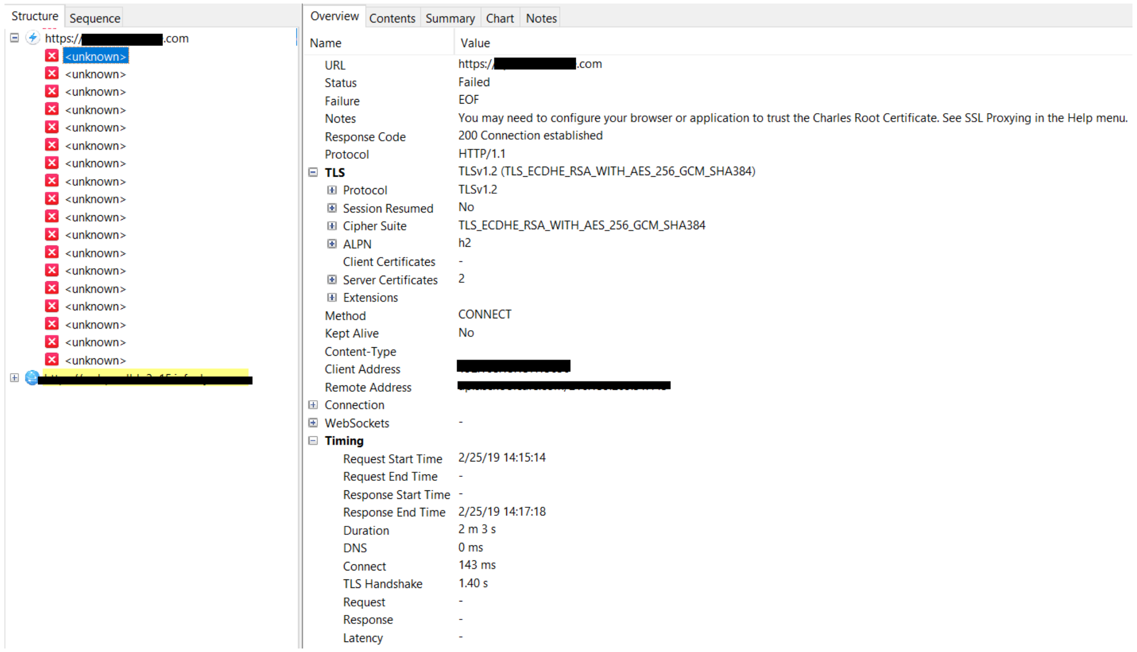
1. The tester provided a test application (APK) for penetration testing.
2. The application was installed on an emulator (Genymotion/Android studio) or on devices.
3. The tester installed a proxy certificate on a device or on an emulator for man-in-the-middle (MiTM) attacks, which intercepted the request/response. The rationale behind this was to trust the proxy certificate and push it to the Android certificate store. (But this does not take into consideration the SSL pinning case.)
4. Testers would set the proxy setting and then would see the traffic in proxy.

Unfortunately, this approach presented a problem when evaluating Android Nougat applications that target API Level 24 and above.

Here's an example: After completing all prerequisites for intercepting the Android traffic, there was no successful connection request/response from the proxy server.

To ensure this was correct, the code was also verified to see if there was any certificate pinning in the picture. (Fortunately, there was none.) The application was then able to leverage MiTM on most of the other applications installed on devices.

However, the testing application still had some issues making the server connection through the proxy server, as shown in Figure 1.



Structure | Sequence

```
⊟ https://[redacted].com
    <unknown>
    <unknown>
    <unknown>
    <unknown>
    <unknown>
    <unknown>
    <unknown>
    <unknown>
    <unknown>
    <unknown>
    <unknown>
    <unknown>
    <unknown>
    <unknown>
    <unknown>
    <unknown>
    <unknown>
    <unknown>
⊞ [redacted]
```

Overview | Contents | Summary | Chart | Notes

Name	Value
URL	https://[redacted].com
Status	Failed
Failure	EOF
Notes	You may need to configure your browser or application to trust the Charles Root Certificate. See SSL Proxying in the Help menu.
Response Code	200 Connection established
Protocol	HTTP/1.1
TLS	TLSv1.2 (TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384)
Protocol	TLSv1.2
Session Resumed	No
Cipher Suite	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
ALPN	h2
Client Certificates	-
Server Certificates	2
Extensions	
Method	CONNECT
Kept Alive	No
Content-Type	
Client Address	[redacted]
Remote Address	[redacted]
Connection	
WebSockets	-
Timing	
Request Start Time	2/25/19 14:15:14
Request End Time	-
Response Start Time	-
Response End Time	2/25/19 14:17:18
Duration	2 m 3 s
DNS	0 ms
Connect	143 ms
TLS Handshake	1.40 s
Request	-
Response	-
Latency	-

Figure 1. Depicts connection failed status while doing a MiTM attack test on Android traffic.

This gave us reason to dig more deeply. Fortunately, we studied changes to Google's Android applications that target API Level 24 and above. It was uncovered that they no longer are trusted-user or administrator-added certificate authorities (CA) for secure connections. More configuration changes are now required to make proxying work successfully.

This raises two questions: "What's next?" and "How do you proceed?"

Make Mobile Application Penetration Testing Work for You

The following processes explain how to conduct mobile application penetration, or pen, testing. The first one explains in detail what happens inside the system. This will help you understand what happens "behind the curtain."

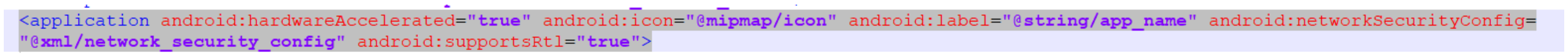
If you choose the second method, the procedure is easier since it includes automation.

Solution 1: How to Prepare for Man-in-the-Middle Android Traffic

The following process entails a detailed, eight-step procedure that explains how to conduct mobile application penetration testing without automation.

1. Install the proxy (Burp, Charles, or others) certificate on the emulator or the device. For more help, refer to these McAfee resources:
   - https://securingtomorrow.mcafee.com/business/testing-android-application-security-part-1/
   - https://securingtomorrow.mcafee.com/business/testing-android-application-security-part-2/
2. Using the APK tool, decompile the APK bundle.
3. Add the following attribute inside the application part of Manifest.XML: **android:networkSecurityConfig="@xml/network_security_config"**

Here's an example of the change in Manifest.XML:

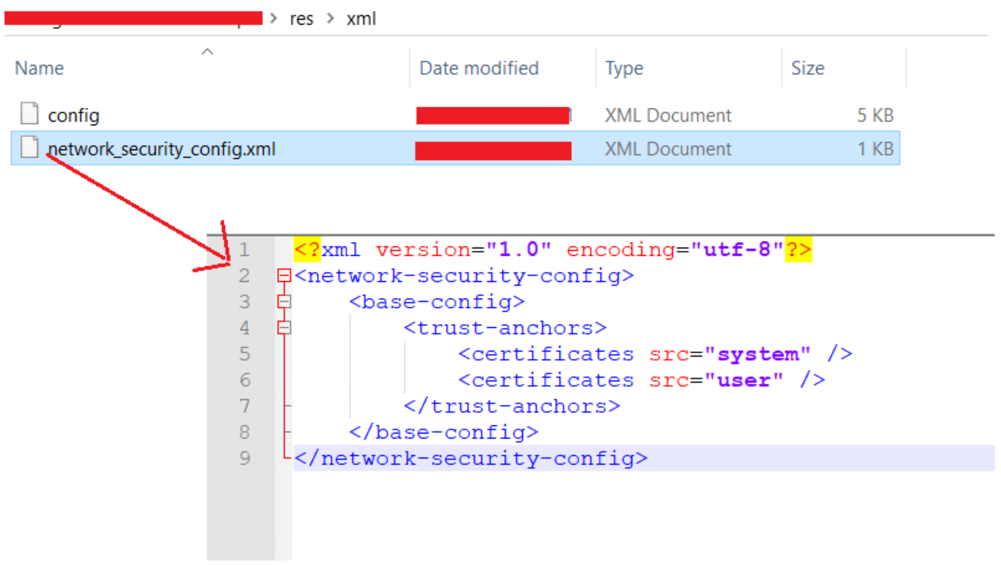


```
<application android:hardwareAccelerated="true" android:icon="@mipmap/icon" android:label="@string/app_name" android:networkSecurityConfig=
"@xml/network_security_config" android:supportsRtl="true">
```

Figure 2. Shows an attribute added to Manifest.XML file.

4. Add a new "**network_security_config.xml**" file under \res\xml with the following content:

   ```
   <?xml version="1.0" encoding="utf-8"?>

   <network-security-config>

       <base-config>

           <trust-anchors>

               <certificates src="system" />

               <certificates src="user" />

           </trust-anchors>

       </base-config>

   </network-security-config>
   ```



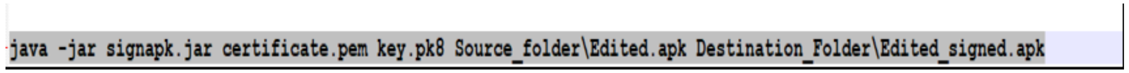
[redacted] > res > xml

Name	Date modified	Type	Size
config	[redacted]	XML Document	5 KB
network_security_config.xml	[redacted]	XML Document	1 KB

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <base-config>
        <trust-anchors>
            <certificates src="system" />
            <certificates src="user" />
        </trust-anchors>
    </base-config>
</network-security-config>
```

Figure 3. Depicts network_security_config.xml file added under \res\xml folder.

5. Rebuild the edited package again using the APK tool.
6. Since the APK package was modified, you will need to sign the application.
7. Download open source "Signapk.jar" and use the following command:



```
java -jar signapk.jar certificate.pem key.pk8 Source_folder\Edited.apk Destination_Folder\Edited_signed.apk
```

8. Install this new version of application -> Configure proxy. You are now ready to use MiTM test on Android traffic.

Solution 2: The Automated Solution

This second process includes automation and makes the process easier.

To follow this specific method click here.[2] You will be able to leverage a quick script that makes steps 3 and 4 even easier.

After you add an exception, the script will then generate *"Appname_new.apk"*. Again, sign it as shown in step 7 above, and install this version of the application.

You can now execute the request and response actions.

Conclusion

The challenge in reviewing Android web application traffic in the latest versions is now resolved for penetration testers working with mobile applications.

If you would like a security review of a mobile application used at your organization, McAfee® Mobile Application Assessment (McAfee MAA) Service can help.

Learn More

To learn more, visit McAfee® Consulting Services, McAfee MAA Service, or contact your sales account manager or partner to learn more.

1. Changes to Trusted Certificate Authorities in Android Nougat
2. Add Security Exception to APK

About McAfee

McAfee is the device-to-cloud cybersecurity company. Inspired by the power of working together, McAfee creates business and consumer solutions that make our world a safer place. By building solutions that work with other companies' products, McAfee helps businesses orchestrate cyber environments that are truly integrated, where protection, detection, and correction of threats happen simultaneously and collaboratively. By protecting consumers across all their devices, McAfee secures their digital lifestyle at home and away. By working with other security players, McAfee is leading the effort to unite against cybercriminals for the benefit of all.

www.mcafee.com.

McAfee™

2821 Mission College Blvd.
Santa Clara, CA 95054
888.847.8766
www.mcafee.com