Secure your mobile applications

Find vulnerabilities with IBM Security AppScan Standard

Daniel J Anderson  
Carlos Hoyos  
Nader Nassar  

April 16, 2013

With the explosive growth in the mobile ecosystem, mobile application security is a huge concern. New mobile application designs require new ways of testing to ensure data safety. In this article, explore different aspects of mobile application security. With hands-on examples, learn to use IBM® Security AppScan® Standard with mobile user agents and with emulators and actual devices for Android and iOS.

Introduction

In the rapidly growing mobile ecosystem, mobile application security is a hot topic with IT security staff, researchers, and leaders. Mobile application development incorporates new design principles that require developers to pay attention to new methods for ensuring data safety. Testing your mobile applications and associated infrastructure can be an effective way to increase the security of your mobile solutions.

Learn more about the IBM Security AppScan family for mobile and web applications.

This article discusses mobile application security and provides hands-on techniques for configuring the dynamic testing of mobile application servers using IBM Security AppScan Standard Edition. Learn to set up AppScan to scan mobile applications with three different models:

- Scanning mobile web applications by setting up a mobile user agent
- Using an emulator for both iOS and Android
- Configuring an actual mobile device for both Android and iOS

Figure 1 shows the three different configurations.
Figure 1. Methods to scan and test mobile applications

Types of mobile applications
As Omri Weisman described in his recent webcast (see Related topics), mobile applications can be divided into several categories. Your mobile application’s category determines which testing techniques and corresponding tools can be used most effectively.

Mobile native applications
Mobile native applications are targeted to run on the device and are written in a native language, such as Java for Android applications or Objective C for Apple iOS applications. Applications in this category are best tested with static analysis tools. IBM Security AppScan Source Edition version 8.6 and higher supports the static analysis of Android applications. However, many native applications also use standard web and web service communications interfaces that can often be tested using IBM Security AppScan Standard Edition.

Mobile web applications
Mobile web applications are just as the name implies. Instead of developing a native application for each of the mobile platforms, many organizations will add support for mobile devices to their existing websites. Or, they’ll develop specific websites to support mobile applications. This support often includes streamlined navigation, reduced bandwidth requirements, optimization for smaller screen sizes, different technology (for example, HTML5 versus Flash), and so on. Thus, the "application" deployed to the mobile device is often little more than a shortcut or bookmark. Applications in this category are well suited for both static testing using AppScan Source Edition and dynamic testing using AppScan Standard Edition.
Mobile hybrid applications

Hybrid mobile applications often consist of a web-based (HTML, CSS, JavaScript) user interface (WebView/UIWebView) surrounded by a layer of native application. Applications in this category are also well suited for both static testing using AppScan Source Edition (for Android) for the native layers and dynamic testing using AppScan Standard Edition for the web-based layers of the application.

Mobile applications and web service

For each type of mobile application, there's a common model that involves splitting processing between the mobile portion that's deployed to the mobile device and a central processing, or data storage, portion that's deployed to a server. This model often uses a web service interface that can be tested directly using AppScan. AppScan supports the testing of a wide variety of web services, including those that are SOAP-based and REST-based.

The methods described in this article should test the interfaces of the web service used by your application. However, for further security, you must still ensure that the web service is comprehensively tested, including all implemented functions not currently used by the mobile application. AppScan only tests the interfaces it is made aware of during the exploration phase.

Communication with the mobile web or application server is often through standard protocols and data formats that can be intercepted, such as HTTP, HTTPS, SOAP, REST, JSON, and XML. The mobile application server can be tested using IBM Security AppScan Standard Edition. The following sections describe methods to configure AppScan to capture these interactions—an essential step in testing mobile web application servers. You'll learn about the three methods (in Figure 1) to test mobile application servers with AppScan: configuring the user agent, using the emulators, and using the actual mobile devices.

Configuring the user agent

The easiest way to test mobile application servers is to configure AppScan to send User-Agent strings that mimic mobile devices. AppScan has added native support for a large variety of devices; it has the ability to configure any custom User-Agent strings that may be necessary.

You can set the User-Agent string by selecting Explore Options from the Scan Configuration window, as shown in Figure 2. Then, test the server portion of the mobile application as you normally would with URLs, manual recordings, web service interaction, and so on.
Figure 2. Set up user agent

This method is a good match for mobile web applications, and it should be familiar if you have experience using AppScan to test web applications. Making simple configuration changes should be sufficient for testing most applications.

Configuring an emulator

Another technique for testing mobile applications is to run the application in the software emulator and proxy its traffic through AppScan. This section provides details for Android and iOS.

Android emulator configuration

You can configure an Android emulator by using the emulator that is included with the Android SDK. When the application is running in the emulator, the network traffic can be directed through the AppScan Proxy Port available on the localhost address (127.0.0.1) at the port specified in the AppScan > Tools > Options window, as shown in Figure 3. By default, this port is picked dynamically. If you're using this feature actively, consider setting it to a static value by unchecking the “Let AppScan choose the port automatically” checkbox and setting your preferred port in the "AppScan proxy port" field.
Figure 3. Set up proxy

Configure the Android emulator to use this proxy port on the localhost address in the emulator command line using the following command.

```
emulator -avd myAVD -http-proxy 127.0.0.1:{port from appscan config}
```

If you're launching the emulator from Eclipse, you can configure this setting from **Window > Preferences > Android > Launch > Default Emulator Options**. Note that the proxy configurations in the device and emulators may not be used by some applications. In such cases, other approaches, including development support, may be required.

**Android OS level configuration**

In some versions of the Android OS, you may be able to configure a proxy from within the emulator instead of using the command line. If you do so, be aware that in the Android OS you will need to use a different proxy address: 10.0.2.2. This is a special address that corresponds to the loopback interface on your local PC (127.0.0.1). It is needed because 127.0.0.1 is used by the emulated Android OS as its own internal loopback port.

In Android 4.1, to adjust the proxy setting:

1. Go to the Settings menu.
2. Select **More** under Wireless and networks.
3. Select **Mobile network settings**.
4. Select **Access Point Names**.
5. Select the network listed.
6. Modify the Proxy and Port fields, as shown in Figure 4.
7. Ensure that the username is blank and a password is not set.

**Figure 4. Android OS level configuration**

![Android OS level configuration](image)

**iOS simulator**

The iOS simulator does not have dedicated proxy settings, but it's easy to set a proxy for all network traffic using a Mac computer. In **Preferences > System preferences**, select **Networking** and choose the interface that you are using (wired or wi-fi). Under the Advanced setting, you'll see a proxy tab, as shown in Figure 5. Set the IP and port to that of the machine where you are running AppScan. The AppScan proxy only binds to local traffic, so you'll need to make configuration changes to forward incoming traffic to a local adapter (see **Using an iOS mobile device**).
Configuring a mobile device

If you want to explore from your actual mobile device, some additional configuration is required. As mentioned, the AppScan proxy binds to the localhost loopback address only (127.0.0.1). You need to be able to make this proxy port available on an available interface.

There are many utilities to help you configure a mobile device. The example in this article uses rinetd (see Related topics), an Open Source (GPL 2+) utility that is easy to use, widely available, and free. After you've downloaded the rinetd utility to the PC where AppScan is installed, rinetd installation is as simple as extracting the zip file and placing the executable in the location of your choice.

After installation, you'll need to create a very simple configuration file. The format and features for the file are fully documented on the rinetd site. The example in this article requires a single line:

```
(IP to listen on) (port to listen on) (IP to forward to) (port to forward to)
192.168.1.114 28080 127.0.0.1 18080
```

This configuration line instructs rinetd to listen to the IP address 192.168.1.114 on port 28080 and forward all traffic to 127.0.0.1 on port 18080 (presumably where AppScan is configured to listen). You can place this line into a text file called rinetd.conf, or anything else of your choice, then run rinetd from the command line with the following statement.

```
rintetd -c:\PATH\TO\CONFIG\rinetd.conf
```
After rinetd is running, you'll be able to proxy a device's traffic through the available IP address and port to the AppScan proxy port on the local loopback address.

**Using an Android mobile device**

After configuring rinetd, you'll need to configure your Android device to proxy using the rinetd IP and port (192.168.1.114 and 28080 in the example). Select **Settings > Wi-Fi** and modify the current network, as shown in Figure 6.

**Figure 6. Set up Android proxy**

<table>
<thead>
<tr>
<th>Status</th>
<th>Connected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal strength</td>
<td>Excellent</td>
</tr>
<tr>
<td>Link speed</td>
<td>54Mbps</td>
</tr>
<tr>
<td>Security</td>
<td>WPA/WPA2 PSK</td>
</tr>
<tr>
<td>IP address</td>
<td>192.168.1.107</td>
</tr>
<tr>
<td>Password</td>
<td>(unchanged)</td>
</tr>
</tbody>
</table>

- **Show advanced options**
- **Proxy settings**
  - **Manual**
  - The HTTP proxy is used by the browser but may not be used by the other apps.
- **Proxy hostname**: 192.168.1.114
- **Proxy port**: 28080
- **Bypass proxy for**: example.com, mycomp.test.com, localhost
- **IP settings**: DHCP

**Using an iOS mobile device**

Similarly, on an iOS device you'll need to set the HTTP proxy from the **Settings > Wi-Fi** configuration screen. Set the Server and Port to those configured in rinetd (192.168.1.114 and 28080 in the example).
Recording

After you've set up your proxy configuration using any of the methods above, the next step is to configure a scan and begin a manual exploration.

In the emulator or device, run the application that you want to test. As you explore the application features, the traffic generated is routed through the AppScan proxy for recording. When you're done manually exploring with the emulator or device, close the browser and the traffic is loaded into the configuration. When all recording is complete, you're ready to begin testing. You can reconfigure the mobile devices and should shut down rinetd (if used).

Testing

Testing for security is very similar to the testing of typical web applications and web services. The test policy that you select will most likely be geared to testing the mobile web application, possibly with application only, vital few, or developer's essentials. It's a good idea to also test the associated infrastructure periodically. Organizations sometimes treat mobile application servers differently in terms of policy, procedures, and standards, even though they are often, in essence, web application servers and web servers.
You might need to do some additional configuration to support the communications methods for your application. Web service methods, such as SOAP or REST, may often be used. AppScan includes tests for testing SOAP methods. It can also be configured to test RESTful services. Check out Ory Segal's article on this topic (see Related topics) on the Application Security Insider blog.

## Transport layer encryption

Mobile applications can operate in varied and high-risk network environments, so transport layer encryption (SSL/HTTPS) is often critical. Properly implemented transport layer encryption can present challenges when testing with the emulator or simulators or from the actual devices. The AppScan proxy must function as a "man in the middle" in order to capture the encrypted traffic.

Challenges with transport layer encryption can usually be overcome by adjusting configuration settings in the application or device or simulator, thereby establishing trust with the AppScan certificate. However, in certain cases development support may be required, such as for non-configurable pinned certificates. Detailed configuration settings for mobile devices or the emulators are beyond the scope of this article.

## Conclusion

If your organization isn't currently supporting mobile applications, it likely soon will be. In this article, you learned about mobile application security with some hands-on techniques to dynamically test the security of mobile application servers using IBM Security AppScan Standard Edition.
Related topics

- **IBM Security AppScan Standard**: Learn about this software that can detect, analyze, and remediate web application vulnerabilities.
- **"The New Frontier - Securing Mobile Apps (WEBCAST)"**: Omri Weisman, Manager of Application Security Research Group, IBM Security, talks about how IBM Security AppScan tools can help with application security.
- **Steps for scanning Web Services in AppScan Standard**: Read this Technote for information about properly testing a web service.
- **"Testing RESTful Services with AppScan Standard"**: RESTful services can be a nightmare for automated web application scanners. Learn how to use IBM's AppScan Standard to train the scanner to cope with RESTful services using manual or automatic configuration.
- **rinetd**: Download rinetd(8), which redirects TCP connections from one IP address and port to another.

© Copyright IBM Corporation 2013
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)