Android WebViews

Andrew Lee-Thorp

SecAppDev2019

© 2019 Synopsys, Inc.

SYNOPSYS®

About Me

Andrew Lee-Thorp

@Synopsys Software Integrity Group (from Cigital + others)

Android assessments, tool development, system design and development, code review, cutting code, threat modelling

> 10 yrs cutting code, nearly 10 yrs in security

© 2019 Synopsys, Inc.

SYNOPSYS®

Agenda

1. WebView basics
2. Javascript to Java bridge (historical)
3. file:// vector attacks
4. HTTPS
5. URL schemes
6. Cross application scripting

© 2019 Synopsys, Inc.

SYNOPSYS®

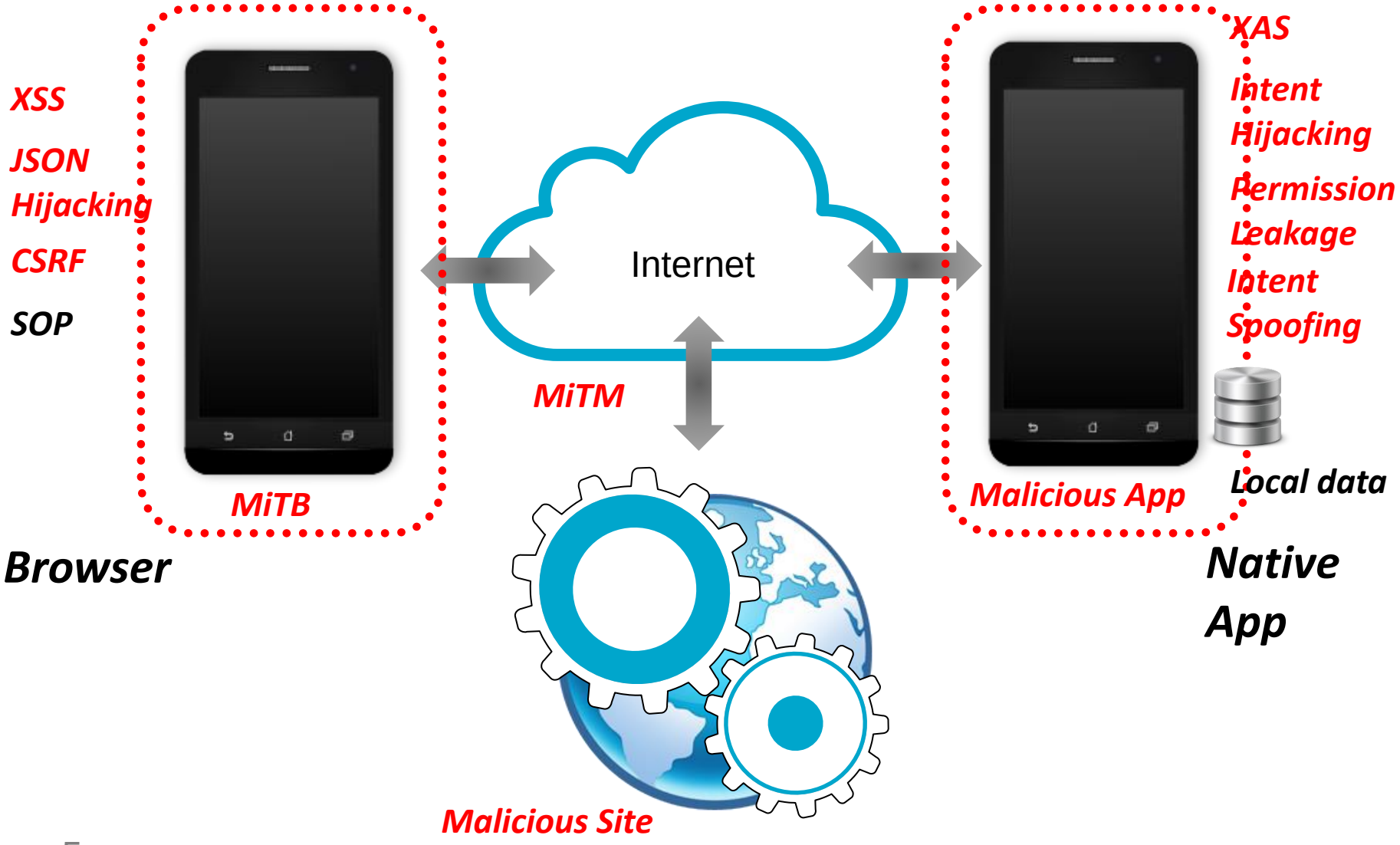
What is a WebView?

- Control to load and display webpages.
- Allows app to react to events
- Based on WebKit (pre-Kitkat 4.4)
- Based on chromium (blink rendering engine) post KK (WebKit was too buggy)
- Since Android 5.0 (Lollipop), update APK separately
- Android 8 Oreo the renderer runs in a separate, sandboxed process + seccomp filter

© 2019 Synopsys, Inc.

SYNOPSYS®

Threats Facing WebView Apps

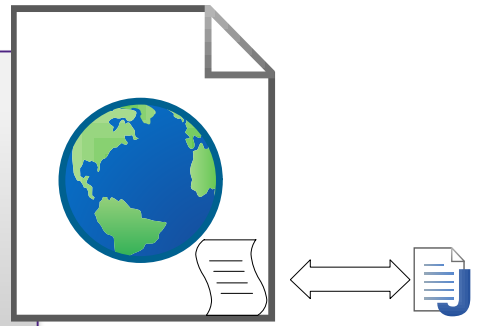


© 2019 Synopsys, Inc.

1. A bit of history

- Javascript to Java bridge
- Access Java object through Javascript in page

```
// Java code
fileUtilsObject = new FileUtils();          1
webview.addJavascriptInterface              2
      fileUtilsObject, "Futil");
```



```
// JavaScript
<script>
file = '/data/data/com.example.myapp/cache.txt';   3
FUtil.write(file, data, false);                    4
</script>
```

© 2019 Synopsys, Inc.

SYNOPSYS®

1. What about the Same Origin Policy?

Answer: SOP is supported

… but there is no notion of SOP as far as JavascriptInterface is concerned.

© 2019 Synopsys, Inc.

SYNOPSYS®

1. Abusing JavascriptInterface

- Disclosed 2012 (Neil Bergman), Luo et al. (2011)

```
<script>
function run(cmd) { return
FUtil.getClass().forName('java.lang.Runtime')                    1
      .getMethod('getRuntime',null)
      .invoke(null,null).exec(cmd);                              2
}
run(['/system/bin/sh','-c','date > /mnt/sdcard/test']);          3
</script>
```

```
FUtil.getClass().forName("android.telephony.SmsManager")
      .getMethod("getDefault", null)
      .invoke(null, null)
      .sendTextMessage("123456", null, "Body", null, null);      4
```

© 2019 Synopsys, Inc.

SYNOPSYS®

1. @JavascripInterface to the rescue?

- @JavascriptInterface annotation limits exposure

```
I/chromium(13478): [INFO:CONSOLE(1)] "Uncaught TypeError:
Object [object Object] has no method 'secret'", source:  (1)
```

- Came with plenty of caveats
- Use @JavascriptInterface **and** target Android 4.2 (targetSdkVersion = API level 17) **and** run Android 4.2+, **then safe**

© 2019 Synopsys, Inc.

SYNOPSYS®

1. The curious case of the appearing Javascript bridges

Q: Am I safe if I don’t have a JavaScript to Java bridge?

DEMO

- No! [CVE-2014-1939 – Joshua Drake, MWR]
- Why?
- A: System may silently insert other bridges (e.g. accessibilityTraversal)

© 2019 Synopsys, Inc.

1. Proactive defense

- Defense:

```
webView.removeJavascriptInterface
          (“searchBoxJavaBridge_”);
```

- and repeat for other interfaces

© 2019 Synopsys, Inc.

SYNOPSYS®

1. Small snag

- How can I remove the other bridges if I can’t name them?

```
webView.removeJavascriptInterface
          (“foobarbazBridge_”);
```

DEMO

- One answer: use a javacript bridge to find other javascript bridges**

** Note: Correct fix was to target API version 17 or above and enforce Android 4.2 or above using minSdkVersion.

© 2019 Synopsys, Inc.

SYNOPSYS®

2. SOP and file:// attack vectors

- Common pattern to bundle and load local HTML assets

```
String url = "file:///android_asset/load.html";          1
mainWebView.loadUrl(url);                                2
```

Q: What is the implication? [DEMO]

A: Scheme is file://. <script> loaded (or injected) will have access to the same origin, namely file://

- ICS and earlier: vulnerable
- JB and later: can switch on unsafe behaviour!

```
setAllowFileAccessFromFileURLs(true);
```

© 2019 Synopsys, Inc.

2. Mitigation

- Script originates from http://, https:// - CORS disallow access
- **setAllowFileAccess**: file system access. Doesn't apply to file:///android_asset and file:///android_res (default:enabled)
- **setAllowFileAccessFromFileURLs**: JavaScript in the context of a file scheme can access other file scheme URLs (enabled for ICS (15) and below, else disabled)
- **setAllowUniversalAccessFromFileURLs**: JavaScript in the context of a file scheme can access any origin (enabled for ICS (15) and below, else disabled)
- **setAllowContentAccess**: load content from a content provider (default:enabled)

© 2019 Synopsys, Inc.

SYNOPSYS®

3. HTTPS

1. Validating the trust chain
2. Hostname verification
3. Certificate pinning <-- talk about this

Goal of the attacker is to man in the middle the connection

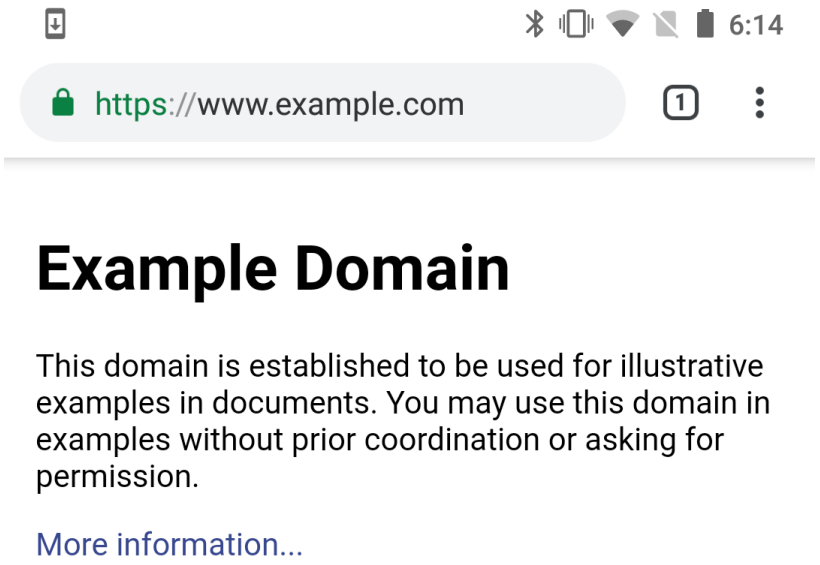
Notation: A = attacker, C = client, S = server, A(S) = attacker acting as server

© 2019 Synopsys, Inc.

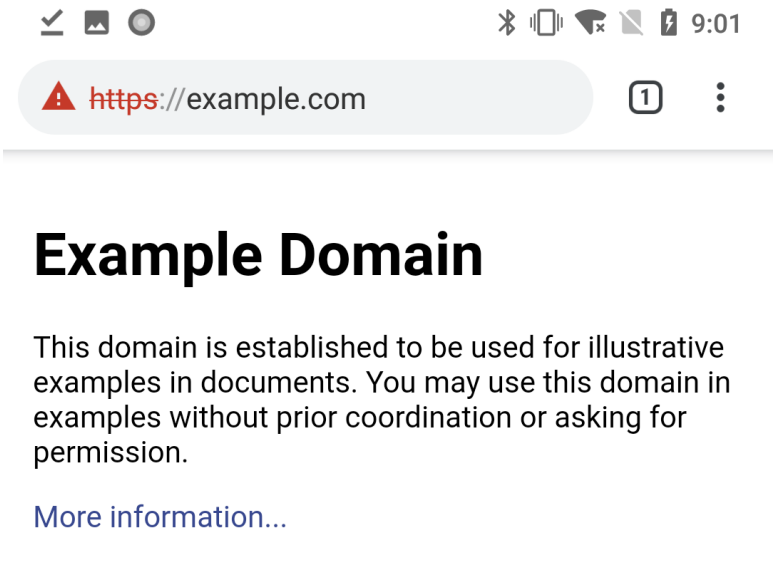
SYNOPSYS®

3. Browser user feedback

- Something tells me it is secure



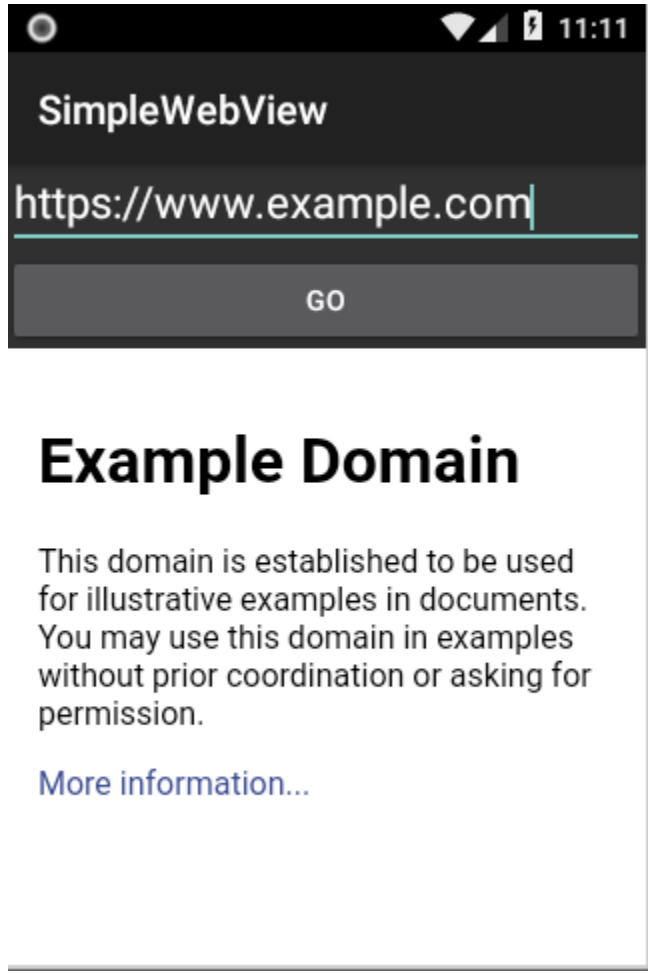
- Something tells me it is insecure



© 2019 Synopsys,

3. WebView user feedback

- Is it secure?
- Am I being phished?
  - SafeBrowsing can help



© 2019 Synopsys, Inc.

SYNOPSYS®

3. Trust the app vendor?

```
@Override
        public void onReceivedSslError(WebView view, SslErrorHandler handler,
                SslError error) {
            handler.proceed();
        }
```

Google PlayStore will complain

© 2019 Synopsys, Inc.

SYNOPSYS®

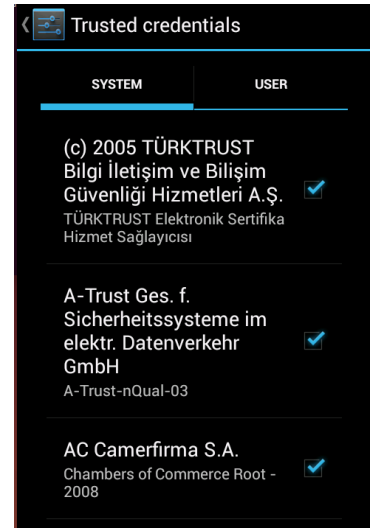
3. Trust the device vendor?

Owner: CN=thawte Primary Root CA, OU="(c) 2006 thawte, Inc. - For authorized use only", OU=Certification Services Division, O="thawte, Inc.", C=US
Owner: CN=Certplus Root CA G1, O=Certplus, C=FR
Owner: CN=Secure Certificate Services, O=Comodo CA Limited, L=Salford, ST=Greater Manchester, C=GB
Owner: CN=CA 沃通根证书, O=WoSign CA Limited, C=CN
Owner: CN=USERTrust ECC Certification Authority, O=The USERTRUST Network, L=Jersey City, ST=New Jersey, C=US
Owner: CN=AddTrust Qualified CA Root, OU=AddTrust TTP Network, O=AddTrust AB, C=SE
Owner: CN=WellsSecure Public Root Certificate Authority, OU=Wells Fargo Bank NA, O=Wells Fargo WellsSecure, C=US
Owner: CN=TÜRKTRUST Elektronik Sertifika Hizmet Sağlayıcısı H5, O=TÜRKTRUST Bilgi İletişim ve Bilişim Güvenliği Hizmetleri A.Ş., L=Ankara, C=TR
Owner: CN=GlobalSign, O=GlobalSign, OU=GlobalSign ECC Root CA - R4
Owner: CN=Starfield Services Root Certificate Authority - G2, O="Starfield Technologies, Inc.", L=Scottsdale, ST=Arizona, C=US
Owner: CN=GlobalSign, O=GlobalSign, OU=GlobalSign Root CA - R2
Owner: CN=Trusted Certificate Services, O=Comodo CA Limited, L=Salford, ST=Greater Manchester, C=GB
Owner: CN=DST Root CA X3, O=Digital Signature Trust Co.
Owner: CN=China Internet Network Information Center EV Certificates Root, O=China Internet Network Information Center, C=CN

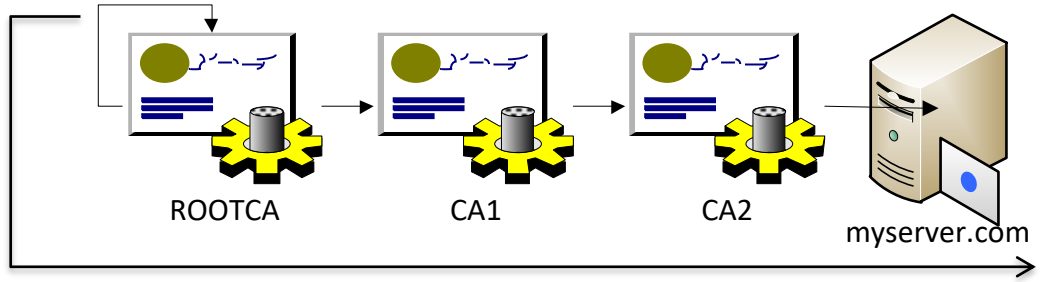
© 2019 Synopsys, Inc.

SYNOPSYS®

3. The weakness

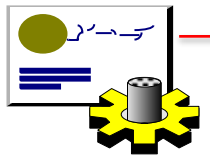


S -> C:

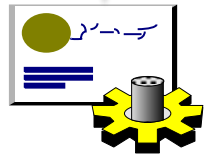


ROOTCA is in system or user trust store

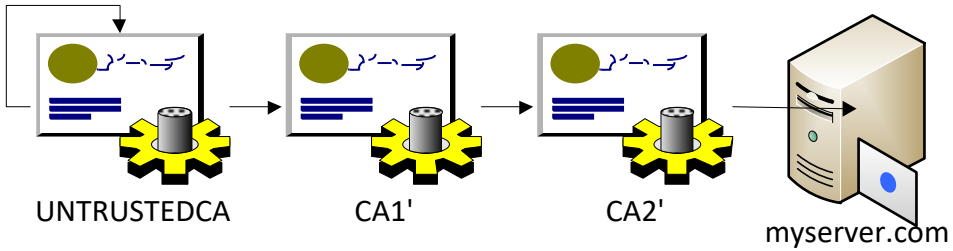
Social engineering



into /data/misc/keychain/cacerts-added



A(S) -> C:



Hacked CA
Compelled creation

© 2019 Synopsys, Inc.

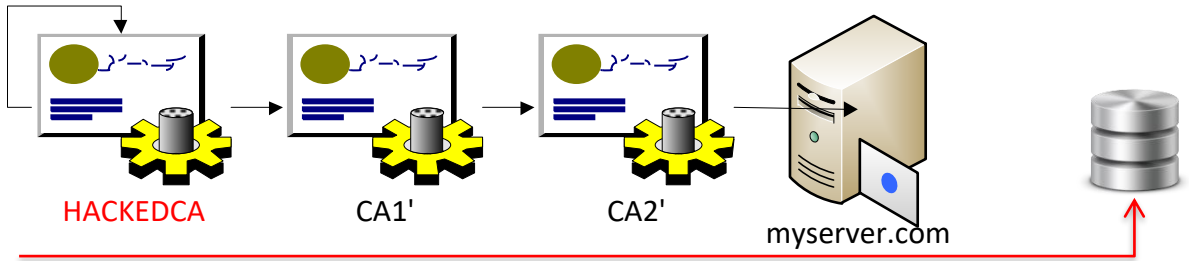
SYNOPSYS®

3. The idea

- Pin to an expected trust anchor

HACKEDCA **is** in system or user trust store

A(S) -> C:

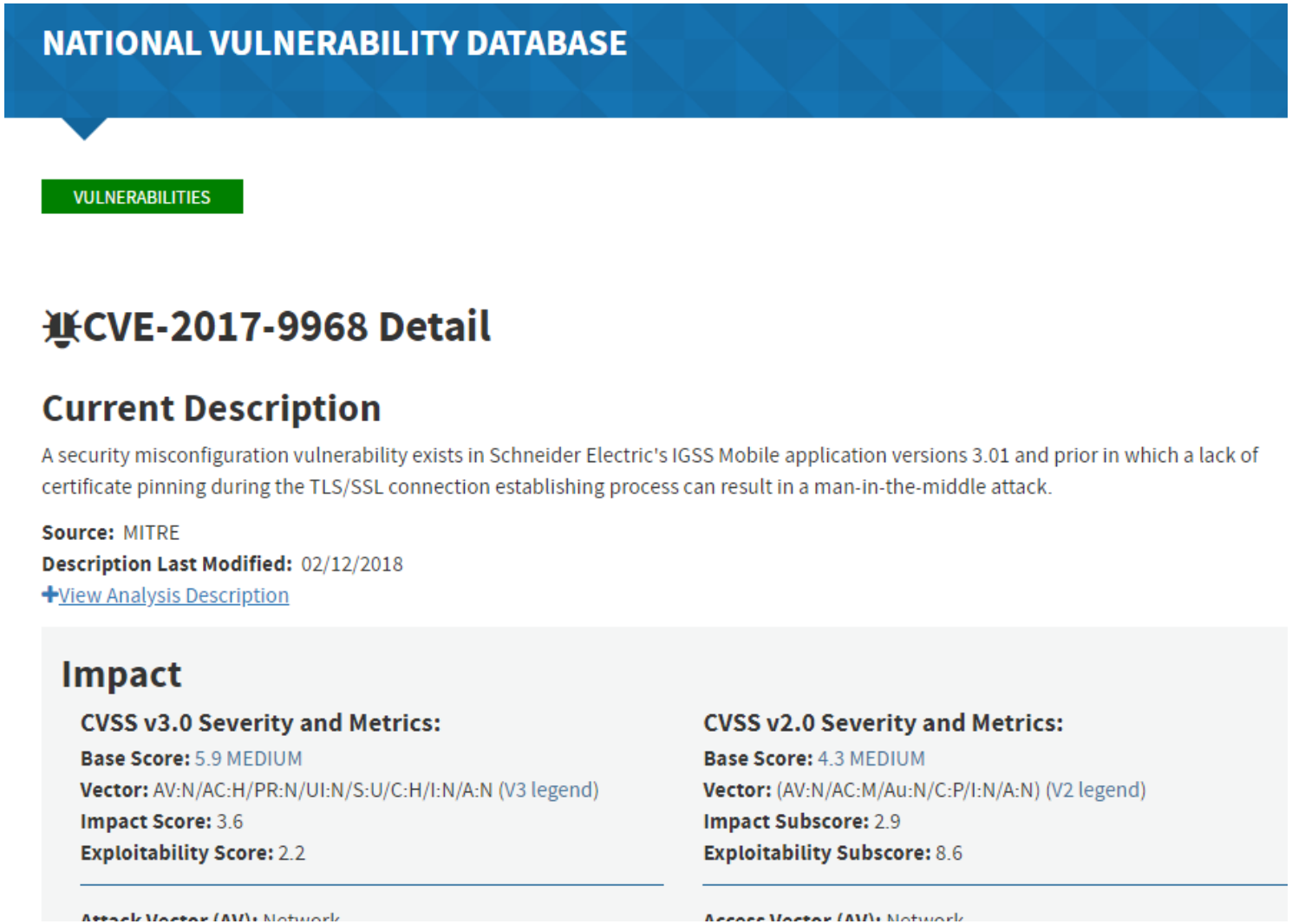


HACKEDCA is not in **app** pins

© 2019 Synopsys, Inc.

SYNOPSYS®

3. “Lack of Certificate Pinning”



NATIONAL VULNERABILITY DATABASE

VULNERABILITIES

CVE-2017-9968 Detail

Current Description

A security misconfiguration vulnerability exists in Schneider Electric's IGSS Mobile application versions 3.01 and prior in which a lack of certificate pinning during the TLS/SSL connection establishing process can result in a man-in-the-middle attack.

Source: MITRE
Description Last Modified: 02/12/2018
View Analysis Description

Impact

CVSS v3.0 Severity and Metrics:
Base Score: 5.9 MEDIUM
Vector: AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:N/A:N (V3 legend)
Impact Score: 3.6
Exploitability Score: 2.2

CVSS v2.0 Severity and Metrics:
Base Score: 4.3 MEDIUM
Vector: (AV:N/AC:M/Au:N/C:P/I:N/A:N) (V2 legend)
Impact Subscore: 2.9
Exploitability Subscore: 8.6

© 2019 Synopsys, Inc.

SYNOPSYS®

3.The problem

Security community did not give correct guidance on how to PIN

```
public void checkServerTrusted(X509Certificate[] chain,
       String authType) throws CertificateException
 {
   ...

   // Perform customary SSL/TLS checks
   try {
     TrustManagerFactory tmf =
       TrustManagerFactory.getInstance("X509");
     tmf.init((KeyStore) null);

     for (TrustManager trustManager : tmf.getTrustManagers()) {
       ((X509TrustManager) trustManager).checkServerTrusted(chain,
authType);
     }
```

```
   } catch (Exception e) {
     throw new CertificateException(e);
     }

     RSAPublicKey pubkey = (RSAPublicKey)
chain[0].getPublicKey();
     String encoded = new BigInteger(1 /* positive
pubkey.getEncoded()).toString(16);
     final boolean expected =
PUB_KEY.equalsIgnoreCase(encoded);
```

https://www.owasp.org/index.php?title=Certificate_and_Public_Key_Pinning&diff=145753&oldid=145134

© 2019 Synopsys, Inc.

SYNOPSYS®

3. Can you spot the bug?

1. Is there a valid certificate chain somewhere in chain (with a trusted anchor)?

```
for (TrustManager trustManager : tmf.getTrustManagers()) {
        ((X509TrustManager) trustManager).checkServerTrusted(chain,
authType);
      }
```

2. Is the cert at chain[0] THE PIN?

```
RSAPublicKey pubkey = (RSAPublicKey) chain[0].getPublicKey();
    String encoded = new BigInteger(1 /* positive */,
pubkey.getEncoded()).toString(16);
    final boolean expected = PUB_KEY.equalsIgnoreCase(encoded);
```

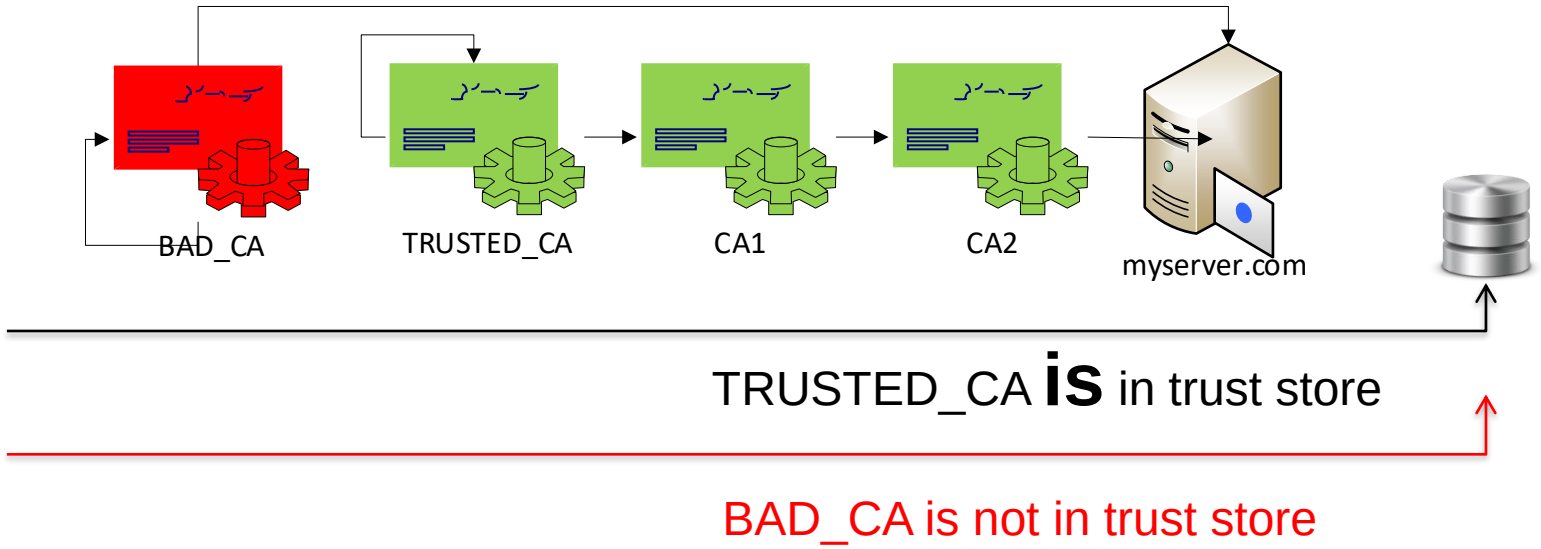
© 2019 Synopsys, Inc.

SYNOPSYS®

3. The attack

- Attacker relays valid chain

A(S) -> C:



© 2019 Synopsys, Inc.

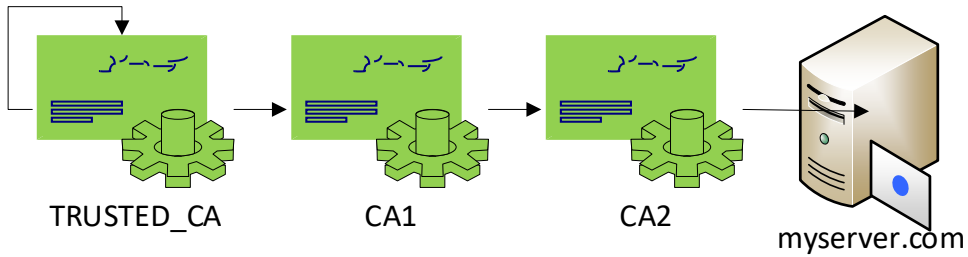
SYNOPSYS®

3. The attack

- However, verification is a two step process

A(S) -> C:

TLS handshake



Valid trust chain?

© 2019 Synopsys, Inc.

SYNOPSYS®

3. The libraries had bugs

Vulnerability in OkHttp's Certificate Pinner

We fixed a bug that could have been used to defeat certificate pinning

Square Engineering Follow

Mar 11, 2016

Written by Jesse Wilson.

Security researcher John Kozyrakis from Cigital recently discovered a vulnerability in OkHttp's *CertificatePinner*. He responsibly disclosed the issue to us via Square's open source bug bounty program at HackerOne.

© 2019 Synopsys, Inc.

SYNOPSYS®

3. Integrating into a Webview was an ugly hack

```
mainWebView.setWebViewClient(new WebViewClient()
{
        @Override public WebResourceResponse shouldInterceptRequest(WebView
view, String url)
        {
                Log.d(TAG, "shouldInterceptRequest() - enter, url -- " +
url);
                 //
                 // make the request over a "pinned" connection
                 //
                 return new WebResourceResponse(mimeType, encoding, data);

        }
```

- Serialised, only GETs, limited schemes (e.g. not ws://), no header support (until API 21, Lollipop 5.0)

© 2019 Synopsys, Inc.

SYNOPSYS®

3. Then came Nougat 7.0

- Declarative network security configuration
- By default apps won’t trust user installed certificates on Android 7.

```
02-12 16:13:14.786 13756 13800 D NetworkSecurityConfig: No Network Security
Config specified, using platform default
02-12 16:13:14.557 13756 13756 D MyActivity: doGo() - url --
https:///www.example.com

02-12 16:04:31.838 13130 13130 D MyActivity: doGo() - url --
https://www.example.com
02-12 16:04:31.954 13130 13189 D NetworkSecurityConfig: Using Network
Security Config from resource network_security_config debugBuild: true
```

DEMO

© 2019 Synopsys, Inc.

SYNOPSYS®

3. networkSecurityConfig is flexible

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

Only trusted when android:debuggable is true

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <debug-overrides>
        <trust-anchors>
            <certificates src="@raw/debug_cas"/>
        </trust-anchors>
    </debug-overrides>
</network-security-config>
```

© 2019 Synopsys, Inc.

SYNOPSYS®

3. The default config is a networkSecurityConfig

Nougat 7.0 (API 24) to Oreo 8.1 (API 27)

```
<base-config cleartextTrafficPermitted="true">
    <trust-anchors>
        <certificates src="system" />
    </trust-anchors>
</base-config>
```

Marshmallow 6.0 API 23

```
<base-config cleartextTrafficPermitted="true">
    <trust-anchors>
        <certificates src="system" />
        <certificates src="user" />
    </trust-anchors>
</base-config>
```

© 2019 Synopsys, Inc.

SYNOPSYS®

4. URL schemes

Register a `scheme://host/[path]` to be launched in response to a matching URL

```
tel:<phone-number>

mailto:someone@example.com
geo:47.6,-122.3
```

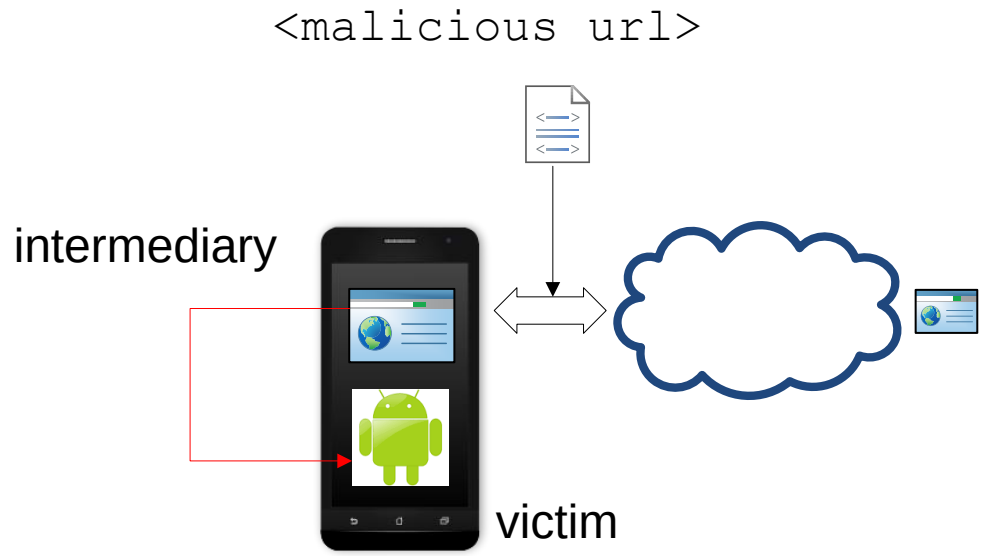
URL handlers are invoked through intents

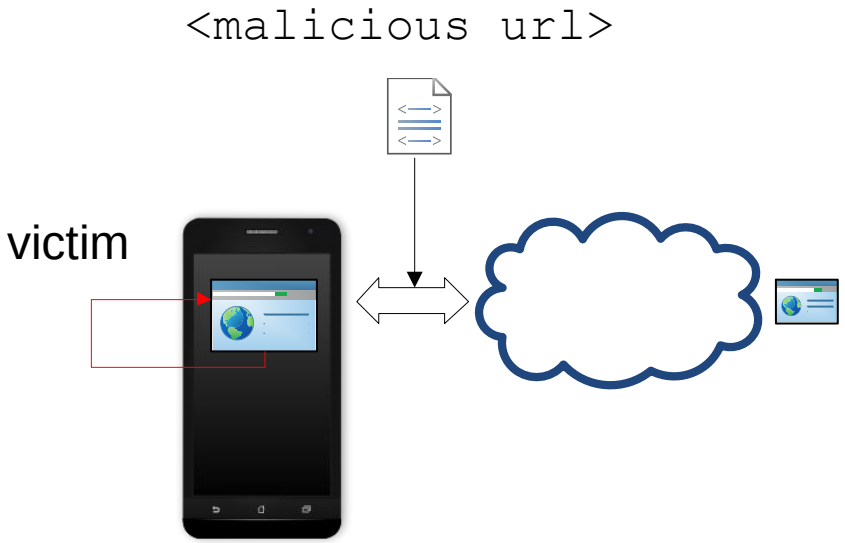
```
<iframe src="tel:555-5555">Call  me now!</iframe>

START {act=android.intent.action.VIEW dat=tel:xxx-xxxx
cmp=com.android.contacts/.activities.DialtactsActivity}
from pid 1945
```

© 2019 Synopsys, Inc.

SYNOPSYS®

4. Webview can be intermediary for intents





Remote attacker can send a intent to a **private** activity and **control** the intent **extras**

```
location.href =
    "intent:data1#Intent
    action=myaction;type=text/plain;end";
```

© 2019 Synopsys, Inc.

SYNOPSYS®

4. URL Schemes

- DEMO
- Prefer ~~android.net.Uri.parse(url)~~ java.net.URI
- Avoid using Intent.parseUri()
- but if you must then …

```
intent.addCategory("android.intent.category.BROWSABLE");
intent.setComponent(null); // Enforce implicit
intent.setComponent(XYZ.class());
                            // Or, enforce explicit
intent.setSelector (null);// Forbid selector [Takeshi Terada, 2014]
```

© 2019 Synopsys, Inc.

SYNOPSYS®

5. XAS (Cross-application scripting)

<malicious url>

Local attacker **tricks victim** into **loading a malicious URL** or content

```
myWebView.loadUrl(urlFromExtrasInIntent)
```

© 2019 Synopsys, Inc.

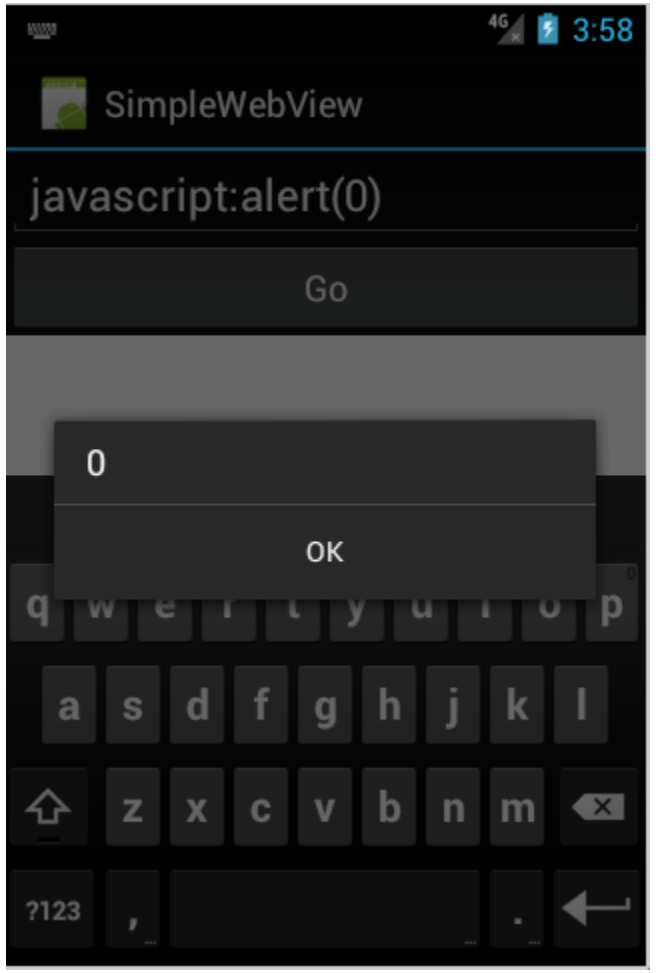
SYNOPSYS®

5. Plenty of dangerous sinks

```
postUrl()
loadUrl()
loadData()
loadDataWithBaseUrl()
startActivity()
```

```
adb shell am start -n
com.example.SimpleWebView/.MyActivity --es "url"
"http://www.google.co.uk"
```

```
adb shell am start -n
com.example.SimpleWebView/.MyActivity --es "url"
"javascript:alert\(0\)"
```



© 2019 Synopsys, Inc.

SYNOPSYS®

5. XAS due to misunderstanding URL validation

```
public boolean shouldOverrideUrlLoading(WebView view,String url){
        if(!url.startsWith("http://www.facebook.com")){
                // do something with “good” URL
                // like loading it
        }
```

Attack string: http://www.facebook.com@attacker.com

© 2019 Synopsys, Inc.

SYNOPSYS®

5. XAS due to bugs in the framework

Attack string:
https://www.fack.website.com\\@www.true.website.com/../headers/”

`android.net.Uri.parse(url).getHost()` → www.true.website.com

“sees”

`webView.loadUrl(url);` → www.fack.website.com

https://medium.com/@hanru.yeh/prefer-java-net-uris-parse-than-android-net-uri-s-one-6f24aee01a8d

© 2019 Synopsys, Inc.

SYNOPSYS®

5. Bugs in the framework

```
D/MyActivity( 1642): uri.getHost() - www.true.website.com
D/MyActivity( 1642): doGo() - url --
https://www.fack.website.com\@www.true.website.com/../headers/
D/MyActivity( 1642): shouldInterceptRequest() - enter, url --
https://www.fack.website.com/headers/
```

Does not appear to be vulnerable in Android 8

© 2019 Synopsys, Inc.

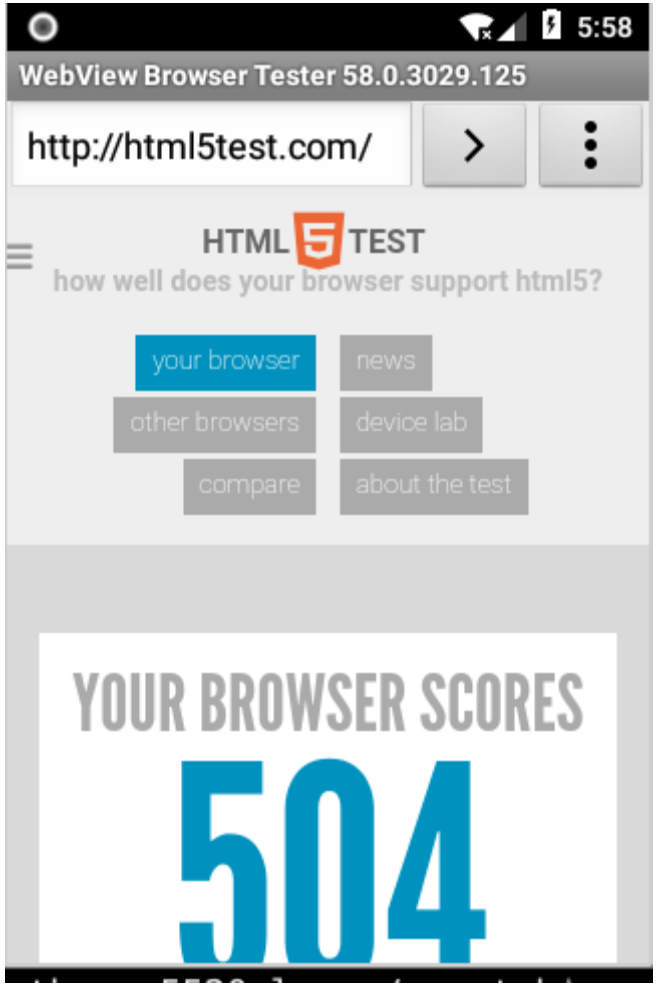
SYNOPSYS®

6. What about HTML5?

- Inherits from chrome

Gelocation
XMLHttpRequest
WebRTC 1.0
CSP 1 and 2
CORS
Subresource Integrity
Sandboxed iframe
Local Storage and Session Storage

- Watch this space!



© 2019 Synopsys, Inc.

SYNOPSYS®

© 2019 Synopsys, Inc.