





Hálózatbiztonság Androidon



Tamas Balogh
Tech lead @ AutSoft



Hálózatbiztonság Androidon

Key Reinstallation AttaCK

2017 őszi sérülékenység

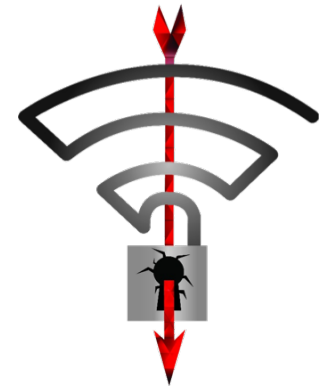
Biztonsági rés a **WPA2** (Wi-Fi Protected Access) protokollban

Nem csak Androidon - más platform is

Minden Android eszköz, amin a wifi be van kapcsolva

Kulcskezelési sérülékenység a handshake során

Sokkal könnyebben kikövetkeztetni a titkostáshoz használt kulcsot





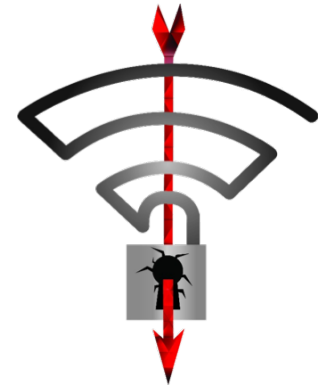
Hálózatbiztonság Androidon

Key Reinstallation AttaCK

Az Android 6.0 és felett egy speciális implementációs hiba
Kikényszeríthető a teljes nulla kulcs

Sérülékenységek

- Titkosítás teljes visszafejtése
- Üzenetek újra játszása
- TCP session lopás
- HTTP tartalom beszúrás



Android Security Bulletin - 2017 November 6.

Hálózatbiztonság Androidon

Mit tehetünk?

Az Alkalmazás fejlesztője nem tudja ezeket kijavítani

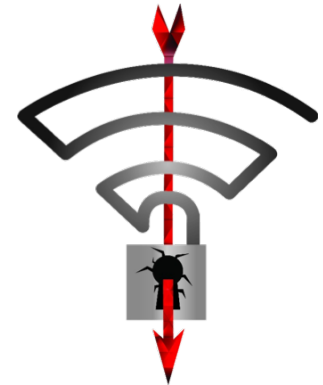
Főként kernel szintű modulban vannak hibák

A Google kiadja a javításokat...

... na de az eszköz gyártók milyen gyorsak?

További biztonsági rétegek hozzáadása (pl. HTTPS)

Fontos tisztába lenni a platform sérülékenységeivel







HTTPS

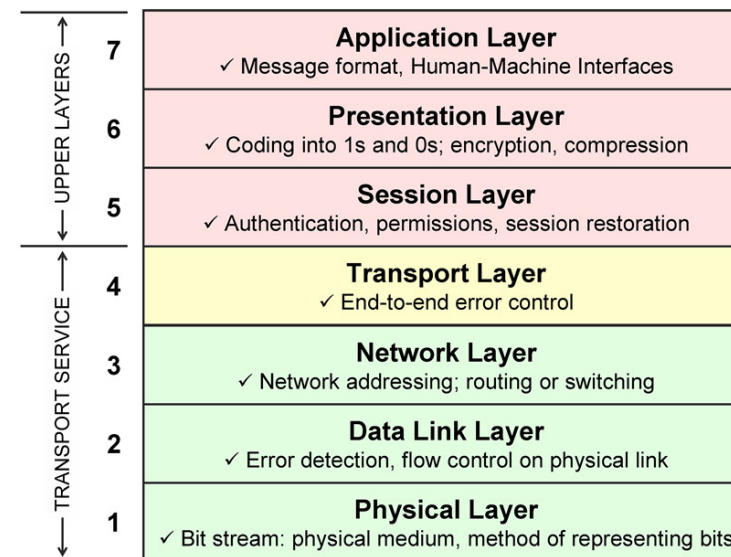
HTTP + Transport Layer Security

TLS: Transport réteg

HTTP: Application réteg

Mi a helyzet a GET paraméterekkel?

- Titkosított, de a szerver logolja!



<http://nhprice.com/what-is-ios-model-the-overall-explanation-of-ios-7-layers.html>



Transport Layer security

Legutóbbi verzió: TLS 1.2

Androidon nagyon fragmentált

Supported vs Enabled

Protocols

Client socket:

Protocol	Supported (API Levels)	Enabled by default (API Levels)
SSLv3	1–25	1–22
TLSv1	1+	1+
TLSv1.1	16+	20+
TLSv1.2	16+	20+

<https://developer.android.com/reference/javax/net/ssl/SSLSocket.html>



A TLS 1.2 kikényszerítése

```
class ForceTlsSocketFactory
@Throws(KeyManagementException::class, NoSuchAlgorithmException::class)
constructor(val internalSSLSocketFactory: SSLSocketFactory) : SSLSocketFactory() {

    @Throws(IOException::class)
    override fun createSocket(): Socket {
        return enableTLSOnSocket(internalSSLSocketFactory.createSocket())
    }

    //...

    private fun enableTLSOnSocket(socket: Socket?): Socket {
        if (socket != null && socket is SSLSocket) {
            socket.enabledProtocols = arrayOf("TLSv1.2")
        }
        return socket!!
    }
}
```



Példa OkHttp 3-al

```
val builder = ConnectionSpec.Builder(ConnectionSpec.MODERN_TLS)

// For below API 16 use TLS 1.0

if (FORCE_TLS_1_2) {
    if (Build.VERSION.SDK_INT >= 16) {
        builder.tlsVersions(TlsVersion.TLS_1_2)
    }
}

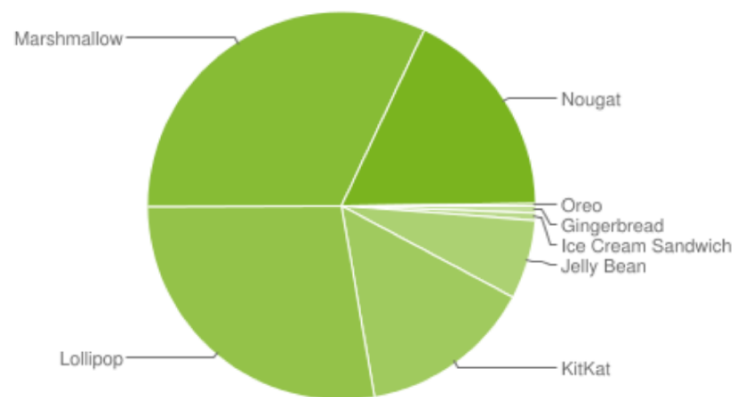
if (Build.VERSION.SDK_INT >= 20) {
    builder.tlsVersions(TlsVersion.TLS_1_2)
}

val client = OkHttpClient.Builder()
    .connectionSpecs(listOf(builder.build()))
    .build()
```



Android verziók

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.6%
4.1.x	Jelly Bean	16	2.3%
4.2.x		17	3.3%
4.3		18	1.0%
4.4	KitKat	19	14.5%
5.0	Lollipop	21	6.7%
5.1		22	21.0%
6.0	Marshmallow	23	32.0%
7.0	Nougat	24	15.8%
7.1		25	2.0%
8.0	Oreo	26	0.2%





TLS - Cipher Suitok

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

TLS - Definiálja, hogy TLS/SSL

ECDHE – Kulcs-csere protokoll

RSA – Résztvevő felek azonosítási protokollja

AES_256_GCM – Blokk rejtjelező/Kulcs méret/Blokk mód

SHA384 – Üzenet integritás ellenőrző protokoll



TLS - Cipher Suitok

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	20+	20+
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	20+	
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	20+	20+
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	20+	20+
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	20+	
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	20+	20+
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256	24+	24+
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	20+	20+
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	20+	
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	20+	20+
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	20+	20+
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	20+	
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	20+	20+
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	24+	24+
TLS_RSA_WITH_AES_128_CBC_SHA	9+	9+
TLS_RSA_WITH_AES_128_CBC_SHA256	20+	
TLS_RSA_WITH_AES_128_GCM_SHA256	20+	20+
TLS_RSA_WITH_AES_256_CBC_SHA	9+	20+
TLS_RSA_WITH_AES_256_CBC_SHA256	20+	
TLS_RSA_WITH_AES_256_GCM_SHA384	20+	20+



Cipher Suitok – Példa OkHttp-vel

```
//Not secure enough, but compatible – OkHttp Docs: The following ciphers are on the HTTP/2's bad cipher suites list
if (Build.VERSION.SDK_INT >= API_LEVEL_11) {
    builder.cipherSuites(
        CipherSuite.TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,
        CipherSuite.TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,
        CipherSuite.TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA,
        CipherSuite.TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA)
}

//Secure – Chrome 51 supports
if (Build.VERSION.SDK_INT >= API_LEVEL_20) {
    builder.cipherSuites(
        CipherSuite.TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
        CipherSuite.TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,
        CipherSuite.TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,
        CipherSuite.TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384)
}

//Secure – Chrome 51 supports
if (Build.VERSION.SDK_INT >= API_LEVEL_24) {
    builder.cipherSuites(
        CipherSuite.TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
        CipherSuite.TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,
        CipherSuite.TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,
        CipherSuite.TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,
        CipherSuite.TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256,
        CipherSuite.TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256)
}
```



TLS - Cipher Suitok

A sikeres kommunikációhoz a szervernek is támogatnia kell a megfelelő TLS verziót és CipherSuitot, a kliens által definiált halmazzal metszete kell legyen.

Több találat esetén a szerver sorrend dönt!

Tesztelhető NMAP-al:

```
nmap -sV --script ssl-enum-ciphers -p 443 hostname
```

TLS - Cipher Suitok

```
1. bata@Baloghs-MacBook-Pro: ~/Desktop (zsh)
| ssl-enum-ciphers:
|   TLSv1.0:
|     ciphers:
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (secp256r1) - A
|       TLS_RSA_WITH_AES_128_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_AES_256_CBC_SHA (rsa 2048) - A
|     compressors:
|       NULL
|     cipher preference: server
|   TLSv1.1:
|     ciphers:
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (secp256r1) - A
|       TLS_RSA_WITH_AES_128_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_AES_256_CBC_SHA (rsa 2048) - A
|     compressors:
|       NULL
|     cipher preference: server
|   TLSv1.2:
|     ciphers:
|       TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (secp256r1) - A
|       TLS_RSA_WITH_AES_128_GCM_SHA256 (rsa 2048) - A
|       TLS_RSA_WITH_AES_256_GCM_SHA384 (rsa 2048) - A
|       TLS_RSA_WITH_AES_128_CBC_SHA256 (rsa 2048) - A
|       TLS_RSA_WITH_AES_128_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_AES_256_CBC_SHA256 (rsa 2048) - A
|       TLS_RSA_WITH_AES_256_CBC_SHA (rsa 2048) - A
|     compressors:
|       NULL
|     cipher preference: server
|_ least strength: A
```



TLS - Azonosítás

Aszimmetrikus kriptográfia

- Védelem a **Man-in-the-middle** támadásokkal szemben

Tanúsítvány

– Gyakorlatilag a szerver publikus kulcsa és meta-adatai aláírva

(1) Tanúsító hatóság által aláírt tanúsítvány

- CA felel érte
- CA-nél regisztrálva van, és tanúsítja

(2) Self-signed certificate

- Szerver adminisztrátor készíti, másnál nincs meg, saját magát tanúsítja
- Alapértelmezetten kivételt eredményez..



TLS – Certificate pinning

Azért használjuk hogy biztosan a megfelelő féllel kommunikáljuk

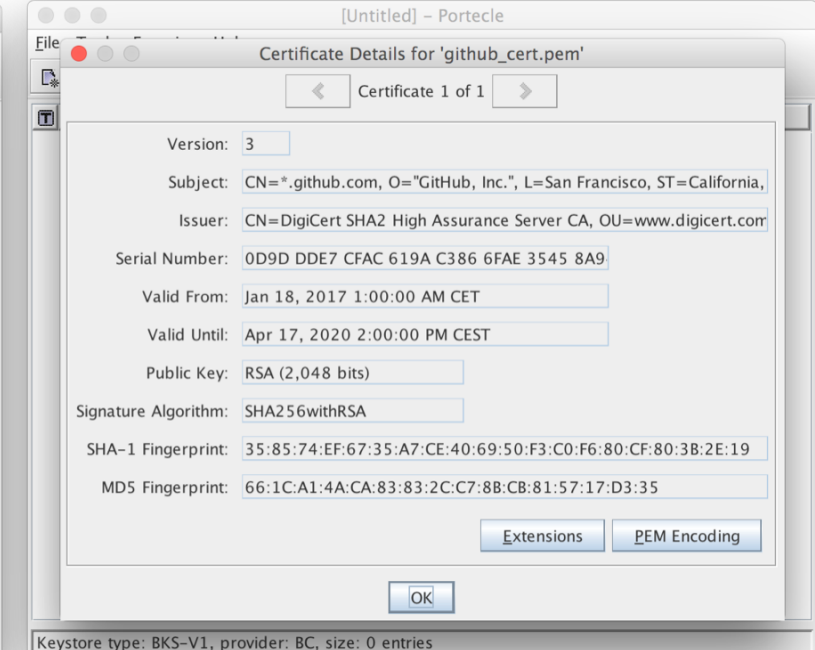
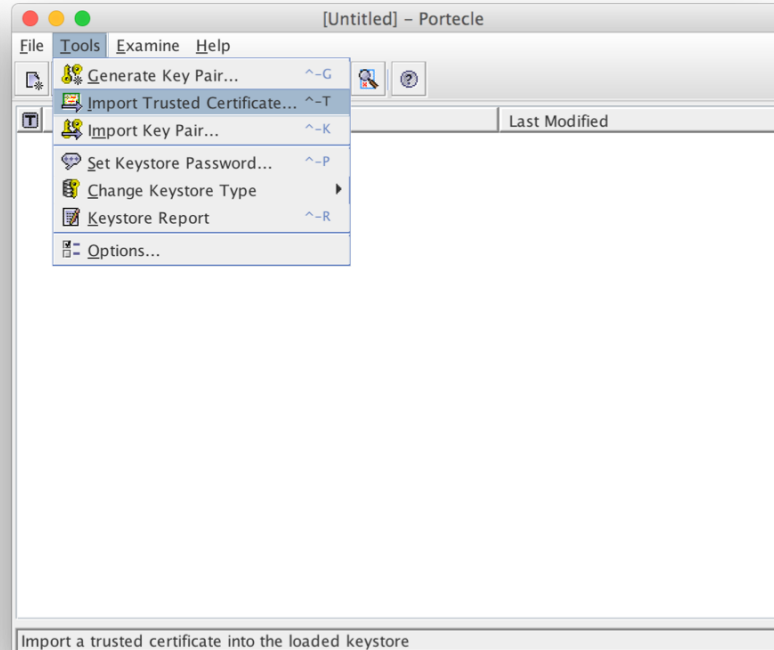
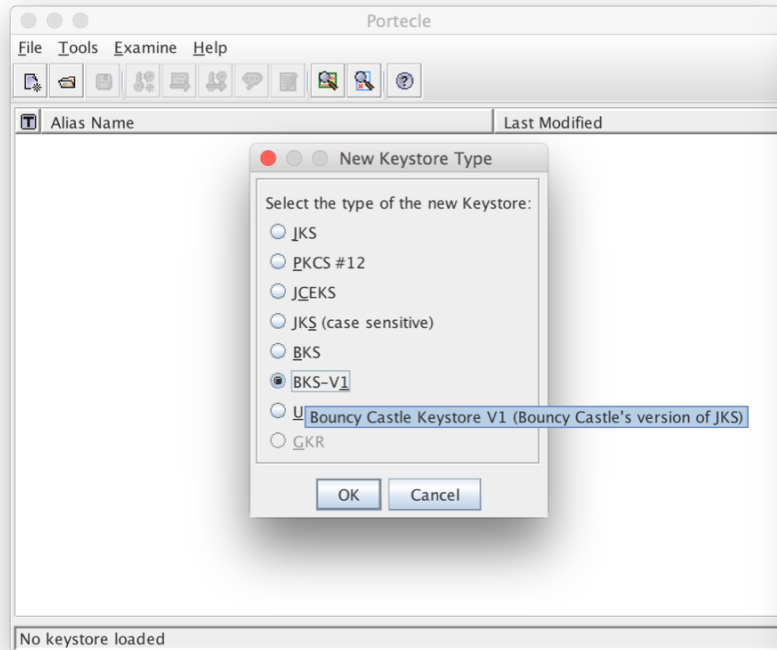
A szerver tanúsítványt (vagy annak lenyomatát) az alkalmazásba tesszük
Kommunikáció során ezt használjuk a validálásra.

Hogyan szerezzük meg ezt a tanúsítványt?

```
echo | openssl s_client -connect hostname:443 2>&1 | sed -ne '/-BEGIN  
CERTIFICATE-/,/-END CERTIFICATE-/p' > hostname.pem
```

Az Android csak **BKS-V1** tanúsítványokat fogadja el a régebbi verziókban,
használjuk a **Portecle** alkalmazást, hogy **PEM** ből **BKS-V1** be alakítsunk.

TLS – Certificate pinning





TLS – Certificate pinning

```
private fun getTrustManagers(context: Context, keyStoreFile: Int, keystorePassword:
String): Array<TrustManager> {
    val trusted = KeyStore.getInstance("BKS")
    val rawResource = context.resources.openRawResource(keyStoreFile)
    trusted.load(rawResource, keystorePassword.toCharArray())

    val trustManagerFactory =
TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm())
    trustManagerFactory.init(trusted)

    return trustManagerFactory.trustManagers
}
```



TLS – Certificate pinning

```
@Throws(KeyStoreException::class, CertificateException::class,  
NoSuchAlgorithmException::class, IOException::class, KeyManagementException::class)  
fun getPinnedCertSocketFactory(context: Context, keyStoreFile: Int, keystorePassword:  
String, forceTls12: Boolean): SSLSocketFactory {  
    val trustManagers = getTrustManagers(context, keyStoreFile, keystorePassword)  
    val sslContext = SSLContext.getInstance("TLS")  
    sslContext.init(null, trustManagers, null)  
  
    var socketFactory = sslContext.socketFactory  
  
    if (forceTls12 && Build.VERSION.SDK_INT in 16..19) {  
        socketFactory = ForceTlsSocketFactory(sslContext.socketFactory)  
    }  
    return socketFactory  
}
```



Referenciák

<http://nhprice.com/what-is-ios-model-the-overall-explanation-of-ios-7-layers.html>

<https://developer.android.com/reference/javax/net/ssl/SSLSocket.html>

<https://developer.android.com/about/dashboards/index.html>

<http://square.github.io/okhttp/>



github.com/BaloghTamas



balogh.tamas@autsoft.hu



Kérdések?

Köszönöm a figyelmet!



github.com/BaloghTamas



balogh.tamas@autsoft.hu