

# Biztonságos alkalmazások Androidon

Balogh Tamás

[balogh.tamas@autsoft.hu](mailto:balogh.tamas@autsoft.hu)



# Tematika

- Android biztonsági alapfogalmak
- Alkalmazások visszafejtése és ez ellen való védekezés: **Obfuszkáció**
- Alkalmazások visszafejtése és ez ellen való védekezés: **Analízis védelem**
- Biztonságos hálózati kommunikáció

# Biztonsági teszt Androidon

# Biztonsági teszt (Penetration test)

- A biztonsági teszt/behatolási teszt alatt a rendszer azon **vizsgálatát** értjük, amikor biztonsági **sebezhetőségeket** keresünk, olyan gyakorlati megoldásokkal, melyek során **megtámadjuk** a rendszert olyan **eszközök** használatával, melyek egy **potenciális támadónak** is rendelkezésére állnak.
- Számos vállalat bíz meg **független harmadik felet**, hogy ezt a tesztet elvégezze.
  - > Potenciális sebezhetőséget keresnek
  - > Ennek a végterméke egy **biztonsági riport**, melynek tételesen felsorolják a megtalált sérülékenységeket, és **ajánlásokat** adnak ezek kivédésére.
- Ezt mi magunk, fejlesztők is megcsinálhatjuk
  - > A független harmadik fél előtt, és nem helyett

# Mobil környezetben

- **Debuggability** – lehet-e debuggert csatolni az alkalmazáshoz
- **Napló szintek** – rendszer naplók vizsgálata, érzékeny információk után kutatva.
- **Obfuscation** – az alkalmazás szét bontása, forráskód visszafejtése céljából
- **Kliens-szerver kommunikáció** – API vizsgálat és a kommunikációs csatorna vizsgálata
- **Érzékeny adatok tárolása/titkosítása** – tokenek, személyes ad
- **Alkalmazás aláírások...**
- ...

# OWASP Mobile Top 10

# OWASP Mobile Top 10 - 2016

- Open Web Application Security Project
- 2016-ban frissítve
- A leggyakoribb mobil biztonsági sérülékenységek
- Nem csak Android (általánosan mobil)
- OWASP Security Testing Guide
  - > <https://github.com/OWASP/owasp-mstg>



# OWASP Mobile Top 10 - 2016

- M1 - Improper Platform Usage
- M2 - Insecure Data Storage
- M3 - Insecure Communication
- M4 - Insecure Authentication
- M5 - Insufficient Cryptography
- M6 - Insecure Authorization
- M7 - Poor Code Quality
- M8 - Code Tampering
- M9 - Reverse Engineering
- M10 - Extraneous Functionality

# BYTECODE

# Java

- Managelt nyelv
  - > Forráskód
  - > Bytecode (platform független)
  - > Gépi kód (platform függő)
- JVM
  - > Java Virtual Machine
  - > Bytekódot futtatja
  - > A legtöbb platformra létezik implementáció
  - > Számos implementáció létezik, van amelyhez saját/eltérő bytekód tartozik (pl. J2SE bytecode nem futtatható közvetlenül egy Android eszközön)

# Java

- A bytekód a közös pont
  - > Erre fordul a forrás
  - > Ezt értelmezi a virtuális gép
- Polyglot
  - > Nem csak Java nyelvet lehet Java bytekódra fordítani
    - Pl. Scala, Groovy, Kotlin, Clojure...
  - > Ugyan arra a bytekódre fordul, többségében interoperábilisan

# Java Virtual Machine

- Számos platformra elérhető implementáció
  - > J2SE implementációk (Windows, MacOS, Linux-ök, BSD, Solaris)
  - > J2ME implementációk (pl. Nokia S60)
  - > J2EE szerver implementációk (Glassfish, JBoss, ...)
  - > Android based on (Apache Harmony) / OpenJDK
- Feladatai
  - > Classloading – Osztályok betöltése
  - > Sandboxing – Az alkalmazás szigetek
  - > Bytecode futtatás – értelmezés
  - > Garbage collection – Memória kezelés

.....

# Java

- A forráskódból fordítás során Bytecode lesz
  - > J2SE esetén **.class** fileok minden osztályhoz
  - > Android esetén egy (vagy több...) **.dex** file
- A bytecode megfelelő eszközökkel könnyedén visszafejthető (algoritmusok, stringek, kulcsok megszerezhetők)
- Ezt valahogy célszerű kivédeni -> Obfuscaciós technikák

# Obfuscáció

# Obfuszkáció

- Az obfuszkációs technikák lényege,
  - > hogy a bytekódot úgy módosítják,
  - > hogy a tényleges működést ne befolyásolják,
  - > de az esetleges visszafejtett kód olvashatósága jelentősen romlik.
- Legegyszerűbb módszer
  - > Osztályok, függvények, változók átnevezése véletlenszerű karakterláncokra

# Obfuszkáció

- További megoldások
  - > Bizonyos hívások átmozgatása JNI-n keresztül natív kódba
    - Java Native Interface – C/C++ kódrészletek hívása
  - > Bytekód titkosítás, majd futtatás előtt dekódolás
    - Inicializációs overhead
    - A kulcsot mellé kell tennie
  - > Kódrészletek elfedése reflection segítségével
    - Direkt fv hívás helyett futásidejű reflection.
    - Futás idejű kódrészletek összerakása.
    - Jelentős overhead

# Obfuszálás

- Ezt sokkal nehezebb megérteni ...

```
public static float fxsa(float z, float fxdr,
                         float zdafff, float opii) {
    double tzeskj = 6371000;
    double dsw = Math.toRadians(zdafff - z);
    double distLng = Math.toRadians(opii - fxdr);
    double kk1kj = Math.sin(dsw / 2) * Math.sin(dsw / 2)
                  + Math.cos(Math.toRadians(z))
                  * Math.cos(Math.toRadians(zdafff))
                  * Math.sin(tzeskj / 2) * Math.sin(dsw / 2);
    double bbnef = 2 * Math.atan2(Math.sqrt(kk1kj),
                                  Math.sqrt(1 - kk1kj));
    float lkjkl = (float) (tzeskj * bbnef);
    return lkjkl;
}
```

# Obfuscáció

- ... mint ezt.

```
public static float calcDistanceOnEarth(float lat1, float lng1,
                                         float lat2, float lng2) {
    double earthRadiusInMeters = 6371000;
    double dLat = Math.toRadians(lat2 - lat1);
    double dLng = Math.toRadians(lng2 - lng1);
    double a = Math.sin(dLat / 2) * Math.sin(dLat / 2)
               + Math.cos(Math.toRadians(lat1))
               * Math.cos(Math.toRadians(lat2))
               * Math.sin(dLng / 2) * Math.sin(dLng / 2);
    double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
    float dist = (float) (earthRadiusInMeters * c);
    return dist;
}
```

# Obfuszkáció

- Elérhető megoldások
  - > *Jshrink*
  - > *yGuard*
  - > *JavaGuard*
  - > *KlassMaster*
  - > **ProGuard (free)**
  - > *DexGuard (paid)*
  - > *Dexprotector (paid)*
  - > ...

# Obfuscation

- Egyéb technikák
  - > Arithmetic obfuscation
    - Numerikus értékek obfuscációja

```
double tzeskj = 6371000;
```
    - Az érték alapján a működést könnyebb megérteni

```
double zzzgj = 1 >> (778 - 755);  
double tzeskj = zzzgj + 231321 + 1945375;
```

# Obfuscáció

- Egyéb technikák
    - > Control flow obfuscation
      - Vezérlési struktúra átalakítása, feltételek átrendezése
- ```
if (age > 18) {
    return 1000;
}
else {
    return 500;
}

for (int zzjf = 2; zzjf < 1; Math.pow(55, 33));{
    while ((age + 113) > 95) {
        return 775 + 225;
    }
    return 411 + 189;
}
```

# Obfuszkáció

- További megoldások
  - > Bizonyos hívások átmozgatása JNI-n keresztül natív kódba
    - Java Native Interface – C/C++ kódrészletek hívása
  - > Bytekód titkosítás, majd futtatás előtt dekódolás
    - Inicializációs overhead
    - A kulcsot mellé kell tennie
  - > Kódrészletek elfedése reflection segítségével
    - Direkt fv hívás helyett futásidejű reflection.
    - Futás idejű kódrészletek összerakása.
    - Jelentős overhead

# Proguard

# ProGuard

- Egyik legelterjedtebb megoldás Java környezetben
- Android default elérhető
- Ingyenes
  - > Fizetős változata a DexGuard

# ProGuard - Használat

- Példa használat Androidon
  - > Fordítórendszer konfigurációja (Androidon alapból elérhető)
  - > Proguard default configurációs file
    - A legtöbb technológiához elérhető
  - > Alkalmazás specifikus configurációs file
    - A fejlesztő feladata megírni
    - Az alkalmazás alapos ismerete szükséges hozzá

# ProGuard - Használat

- Alkalmazás specifikus konfigurációs file
  - > Adott osztályok megtartása  
`-keep class hu.aut.demoapp.model** { *; }`
  - > Adott osztály olyan tagváltozójának megtartása amely annotálva van @Serialize-al  
`-keepclasseswithmembernames class * {  
@hu.aut.demoapp.Serialize.* <fields>; }`

# ProGuard - Használat

- Alkalmazás specifikus konfigurációs file
  - > Adott osztály olyan függvényének megtartása amely annotálva van @Framework-el
    - keepclasseswithmembernames class \* {  
@hu.aut.demoapp.Framework.\* <methods>; }
  - > Annotációk megtartása
    - keepattributes \*Annotation\*
  - > Exceptionök megtartása
    - keepattributes Exceptions

# ProGuard - Használat

- Alkalmazás specifikus konfigurációs file
  - > Ha egy file hiányzik (lehetséges obfuscáció hiba), jelez a ProGuard. Ez kikapcsolható.  
-dontwarn org.junit.\*\*

# Optimalizáció

- A ProGuard, hasonlóan több más obfuscaciós megoldáshoz támogatja az adott bytecode optimalizációját is.
  - > Nem használt osztályok, metódusok kivágása
  - > Writeonly mezők kivágása ...
- Bizonyos műveletekre definiálható, hogy nincs mellékhatása, így lehet kódrészleteket kivágni.  
-assumenosideeffects class hu.aut.demoapp.DebugHelper{ \*; }

# Optimalizáció

- Pl. Android log hívások kivágása release kódból
- ```
-assumenosideeffects class android.util.Log {  
    public static boolean isLoggable(java.lang.String, int);  
    public static int v(...);  
    public static int i(...);  
    public static int w(...);  
    public static int d(...);  
    public static int e(...);  
    public static int wtf(...);  
}
```

# Kivételek

# Kivételek

- Az előbb bemutatott megoldásokkal lehet definiálni hogy az alkalmazás mely részeit ne obfuscálja a rendszer.
- Miért van erre szükség? Miért nem lehet az egész forrást obfuscálni
  - > Pl. A kezdő osztály megvan adva egy konfigurációs fileban
  - > Ebben az esetekben a definíciós file az eredeti néven hivatkozik a kezdő osztályra, de ha azt átnevezi az obfuscátor, már nem fogja a rendszer megtalálni a futtatás során.

# Kivételek

- Gyakori példák
  - > Pl. A kezdő osztály megvan adva egy konfigurációs fileban
  - > Pl. Az Android komponensen definiálva vannak a manifestben
  - > Az egyes kontrollerek definiálva vannak a szerver definíciós filejában
  - > Az alkalmazás bárhol reflectiont használ
  - > Reflection alapú sorosító megoldások (pl. GSON, Jackson)
  - > Kódgeneráláson alapuló megoldások (pl ORM)
  - > JNI hívások
  - > Könyvtárak, API-k esetén a belépési pontok
  - > ...

# Kivételek

- Ezekre a fejlesztőnek kell figyelnie
  - > A megfelelő kivétel szabályokkal úgy kell hangolnia az obfuscációt, hogy a lehető legtöbb kódrészletet obfuscálja anélkül hogy az alkalmazás működését befolyásolná.
  - > A legtöbb külső megoldás, és könyvtár tartalmazza ezeket a szabályokat, de a saját kódunkra nekünk kell ezeket beállítanunk.

# Demo

- Obfuscate demo alkalmazás
- Vizsgálat obfuszkáció nélkül
- Vizsgálat obfuszkációval

# Analízis védelem

# Analízis védelem

- Ami az APK-ban van, az elérhető a támadó számára
- Nincs 100%-os védelem
- De meg lehet (és meg kell) nehezíteni a támadó munkáját
- Statikus vizsgálat ellen - Obfuscation
- Futás idejű vizsgálat és módosítása ellen
  - > Babrálás vizsgálat (Tamper detection)
  - > Emulator vizsgálat
  - > Root vizsgálat
  - > Telepítő vizsgálat

# Tamper detection – Babrálás vizsgálat

- Az alkalmazás bináris módosítása
- A támadó másik aláíró kulcsot kell hogy használjon
- Az aláírás elérhető az alkalmazásba
- Az aláírás lenyomatát beleégerjük az alkalmazásba (obfuszkálni érdemes), majd ezzel összevetjük.

```
fun isTampered(context: Context): Boolean {  
    val appSignature = "bRp+MQPxbGqdIomzJafonWGwHJY="  
    val signatures = context.packageManager.getPackageInfo(  
        context.packageName, GET_SIGNATURES).signatures  
  
    signatures.forEach {  
        val sha1Digest = MessageDigest.getInstance("SHA1")  
        sha1Digest.update(it.toByteArray())  
        val sha1Bytes = sha1Digest.digest()  
        val sha1Base64 = Base64.encodeToString(sha1Bytes, Base64.NO_WRAP)  
        if (sha1Base64 != appSignature) return true  
    }  
    return false  
}
```

# Emulátor vizsgálat

- A támadó gyakran használ emulátort, a felhasználók viszont nem
- Az emulált eszközöknek számos belső paramétere eltér
- Rendszer beállítások, nevek kiolvasása

```
private fun getSystemProperty(context: Context, key: String): String {  
    val cl = context.classLoader  
    val systemProperties = cl.loadClass("android.os.SystemProperties")  
    val paramTypes = arrayOfNulls<Class<*>>(1)  
    paramTypes[0] = String::class.java  
    val get = systemProperties.getMethod("get", *paramTypes)  
    val params = arrayOfNulls<Any>(1)  
    params[0] = String(key.toByteArray())  
    return get.invoke(systemProperties, *params) as String  
}
```

# Emulátor védelem

- Nem 100%-os
- Ha nincs obfuszkálva, akkor könnyen kikerülhető
- Testreszabott emulátorral átverhető

```
val goldfish = getSystemProperty(context, "ro.hardware").contains("goldfish")
val emu = getSystemProperty(context, "ro.kernel.qemu").isNotEmpty()
val sdk = getSystemProperty(context, "ro.product.model") == "sdk"
return emu || goldfish || sdk
```

# Root vizsgálat

- A támadók számos esetben használnak Rootolt eszközöket
- Hasonló az emulator detekcióhoz
  - > Bizonyos partíciók, fileok, binárisok megléte
  - > Bizonyos rootoláshoz használt processzek megléte
- Root Cloaking – Root jogok elrejtése
- A Root detektáló eszközök együtt fejlődnek a Root Cloaking megoldásokkal
  - > Általában a Root Cloaking nyer ( vagy ezek megfelelő kombinációja)
- Rootbeer - Open-Source könyvtár
  - > <https://github.com/scottyab/rootbeer>
  - > Apache 2.0 Licence

```
val rootBeer=RootBeer(context)
if(rootBeer.isRooted()) {
    //...
}
```



# Root vizsgálat

- Sok a false-pozitív
  - > pl. Busybox bináris meglétének ellenőrzése
  - > Bizonyos gyártók, pl. OnePlus, Moto rajta hagyják
  - > `rootBeer.isRootedWithoutBusyBoxCheck()`



# Telepítő védelem

- Egy Android alkalmazás többféle képen is települhet
  - > Google Play Store-ból
  - > Külső alkalmazás boltok
  - > APK-ból telefonról telepítve
  - > APK-ból ADB-n keresztül
- minden alkalmazás lekérdezheti, melyik csomag telepítette.
- A támadók gyakran APK-ból telepítik az alkalmazást
- A legtöbb alkalmazást csak Play Store-on keresztül terjesztik hivatalosan
  - > Vizsgáljuk meg, hogy a Play Store telepítette-e

# Telepítő védelem

- Bizonyos esetekben problémát okozhat
  - > APK alapú belső tesztelés
  - > Fabric.io Beta teszt
  - > Külső alkalmazás boltok
- Ezekre külön konfigurációt érdemes készíteni

```
const val PLAY_STORE_APP_ID = "com.android.vending"

private fun verifyInstaller(context: Context): Boolean {
    val installer = context.packageManager.
        getInstallerPackageName(context.packageName)
    return installer != null && installer.startsWith(PLAY_STORE_APP_ID)
}
```

# SafetyNet Attestation API

- Hivatalos Google API
- Az alkalmazás biztonsági és kompatibilitási vizsgálatát végzi
- Google API Console – API kulcs regisztráció
- Google Play Services-t használja
- A válasz tartalmazza
  - > **ctsProfileMatch** – Az eszköz megfelel-e az Android Compatiblty Test Suite (CTS)-nek
  - > **basicIntegrity** – Általános rendszer integritás

# SafetyNet Attestation API

Device Status	Value of "ctsProfileMatch"	Value of "basicIntegrity"
Certified, genuine device that passes CTS	true	true
Certified device with unlocked bootloader	false	true
Genuine but uncertified device, such as when the manufacturer doesn't apply for certification	false	true
Device with custom ROM (not rooted)	false	true
Emulator	false	false
No device (protocol emulator script)	false	false
Signs of system integrity compromise, such as rooting	false	false
Signs of other active attacks, such as API hooking	false	false

# A rendszer paramétereinek konfigurációja

# Követelmények

- Konfigurációk meghatározása és bevezetése
  - > Általában környezetek szerint bontjuk fel (éles, teszt, fejlesztői...)
- A konfigurációk közötti váltás egyszerű kell legyen
- A konfigurációhoz tartozó feladatok automatikusan lefutnak
- A konfigurációhoz tartozó követelmények teljesülnek
- A követelmények teljesülését automatizált tesztek garantálják

# Konfigurációs paraméterek

- Üzleti logika, funkcionalitás
- Debuggolhatóság
- Loggolási szint
- Obfuscáció
- Kliens-szerver kommunikáció
- Érzékeny adatok titkosítása
- Aláírás

# Konfiguráció példa

	DEBUG	RELEASE
Kód	Mock	Production
Debuggolható	Igen	Nem
Loggolási szint	Verbose	X
Obfuszkáció	Nem	Igen
Kommunikáció	Titkosítatlan - HTTP	Titkosított - HTTPS
Érzékeny adatok tárolása	Titkosítatlan	Titkosított
Álaírás	Debug kulccsal	Release kulccsal

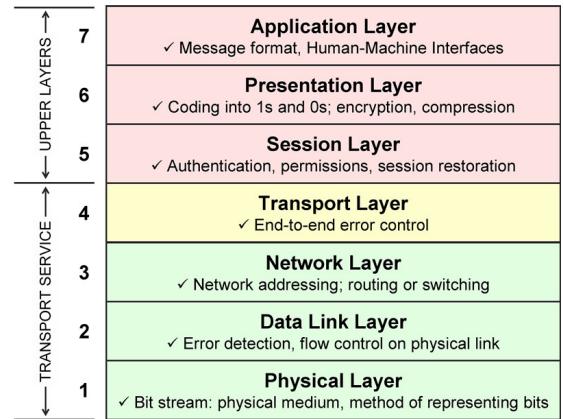
# Demo

- Konfiguráció kezelés

# Hálózat kezelés

# HTTPS

- HTTP + Transport Layer Security
- TLS: Transport réteg
- HTTP: Application réteg
- Mi a helyzet a GET paraméterekkel?
  - Titkosított, de a szerver naplózza!



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

# Transport Layer Security

- Legfrissebb verzió: TLS 1.3 (2018 aug.)
- Android - Töredézett
- 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>

# TLS 1.2 Android 5 alatt

```
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!!
    }
}
```

# Használata 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()
```

# Ciphersuit-ok

TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384

- TLS - TLS/SSL
- ECDHE – Kulcs csere protokoll
- RSA – Azonosítási protokoll
- AES\_256\_GCM – Blokk rejtjelező/Kulcs méret/Blokk mód
- SHA384 – Üzenet authentikációra használt hash függvény

# Ciphersuit-ok

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_GCM_SHA256	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+

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

# Ciphersuit-ok

```
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)  
}  
  
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)  
}  
  
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)  
}
```

# Ciphersuit-ok

- A szervernek is támogatnia kell a felsorolt TLS verzió és ciphersuit valamelyikét
- OpenSSL-el leellenőrizhető

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

# Ciphersuit-ok

```
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_256_CBC_SHA384 (secp256r1) - A
|     TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (secp256r1) - A
|     TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (rsa 2048) - A
|     TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (rsa 2048) - A
|     TLS_RSA_WITH_AES_128_CBC_SHA256 (rsa 2048) - A
|     TLS_RSA_WITH_AES_256_CBC_SHA256 (rsa 2048) - 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_SHA (rsa 2048) - A
|     TLS_RSA_WITH_AES_256_CBC_SHA (rsa 2048) - A
|   compressors:
|     NULL
|   cipher preference: server
| least strength: A
```

# HTTPS - Azonosítás

- Azonosítás aszimmetrikus kriptografiát használva
  - > Véd a támadásokkal szemben

## (1) CA Certified tanúsítvány

- > CA felel érte
- > CA tanúsítja a tanúsítvány láncon keresztül

## (2) Self-signed tanúsítvány

- > Szerver adminisztrátor készíti
- > Alapértelmezetten kivételt okoz

# HTTPS – Certificate pinning

- ~Tanúsítvány letűzés
- Biztosan a megfelelő féllel kommunikálunk?
- Az alkalmazásba égetjük az elvárt tanúsítványt, és csak azzal állunk szóba!
- Szerver adminisztrátortól kérhetjük el
- Vagy az alábbi parancssal le is tölthetjük

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

- A PEM formátumot még át kell konvertálni BKS-V1-re (ajánlott: Portecle alkalmazás)

# HTTPS – 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
}
```

# HTTPS – 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  
}
```

# Köszönöm a figyelmet!



*balogh.tamas@autsoft.hu*