consulthink
GET THE FUTURE, NOW

GDG DevFest
Central Italy / 16 Novembre 2013
Roma - LUISS Enlabs Stazione Termini
Supported by Google

Android Security
Key Management

Roberto Gassirà (r.gassira@mseclab.com)
Roberto Piccirillo (r.piccirillo@mseclab.com)

mobilesecuritylab

Roberto Piccirillo

- Senior Security Analyst - Mobile Security Lab
  - Vulnerability Assessment (IT, Mobile Application)
  - Hijacking Mobile Data Connection
    - BlackHat Europe 2009
    - DeepSec Vienna 2009
    - HITB Amsterdam 2010
  - Android Secure Development

CompTIA Mobile App Security+ Android Edition CERTIFIED

@robpicone

Android Security
Key Management

Roberto Gassirà

- Senior Security Analyst - Mobile Security Lab
  - Vulnerability Assessment (IT, Mobile Application)
  - Hijacking Mobile Data Connection
    - BlackHat Europe 2009
    - DeepSec Vienna 2009
    - HITB Amsterdam 2010
  - Android Secure Development
- IpTrack Developer 

CompTIA Mobile App Security+ Android Edition CERTIFIED

@robgas

Android Security
Key Management

Agenda

- Key Management e CryptoSystem
- Mobile Application: protezione dei dati
- Key Management in Android e sue evoluzioni
- Keychain e AndroidKeyStore
- Tipologie di AndroidKeyStore
- Codelab
  - Generazione chiave pubblica/privata
  - Accesso AndroidKeyStore
  - Digital signature e Encryption

Key Management

"***Key management*** *is the management of cryptographic keys in a cryptosystem.*"

CryptoSystem

- "*refers to a suite of algorithms needed to implement a particular form of encryption and decryption*"

- Tipologie di encryption:
  - Symmetric Key Algorithms
    - Identical encryption key for encryption/decryption
  - Asymmetric Key Algorithms
    - Different key for encryption/decryption

In app?

- Protezione dati riservati
  - Dati dell'applicazione
  - Dati su */sdcard*
  - Chiavi di cifratura

- Scambio sicuro di dati
  - Documento
  - Mail
  - SMS
  - Chiave di sessione

- Firma digitale
  - Documento
  - Mail

Key Management in Android

- Application Level
  - ~~File nell'applicazione (Shared Prefs, File, ...)~~
  - ~~Code Obfuscation~~
  - ~~Generazione a run-time~~

forum.xda-developers.com/showthread.php?t=2525552

[SECURITY] Multiple Android Superuser vulnerabilities

cernekee — 13th November 2013, 08:16 PM

Member - OP
Thanks Meter 126
Posts: 82
Join Date: Jun 2013

[SECURITY] Multiple Android Superuser vulnerabilities

Well, I was going to hold off until all of the vendors had new releases posted, but now that the cat is out of the bag and the evildoers have sufficient information to figure out what got fixed:

Current Superuser/SuperSU releases have security holes that allow any application to execute commands as root without the user's permission (even apps with no permissions). Please upgrade immediately to SuperSU >= v1.69 or another patched release.

This is expected to impact the vast majority of rooted devices and custom ROMs.

Details follow:

Superuser unsanitized environment vulnerability on Android <= 4.2.x

Vulnerable releases of several common Android Superuser packages may allow malicious Android without notifying the device owner:

- ChainsDD Superuser (current releases, including v3.1.3)
- CyanogenMod/ClockWorkMod/Koush Superuser (current releases, including v1.0.2.1)
- Chainfire SuperSU prior to v1.69

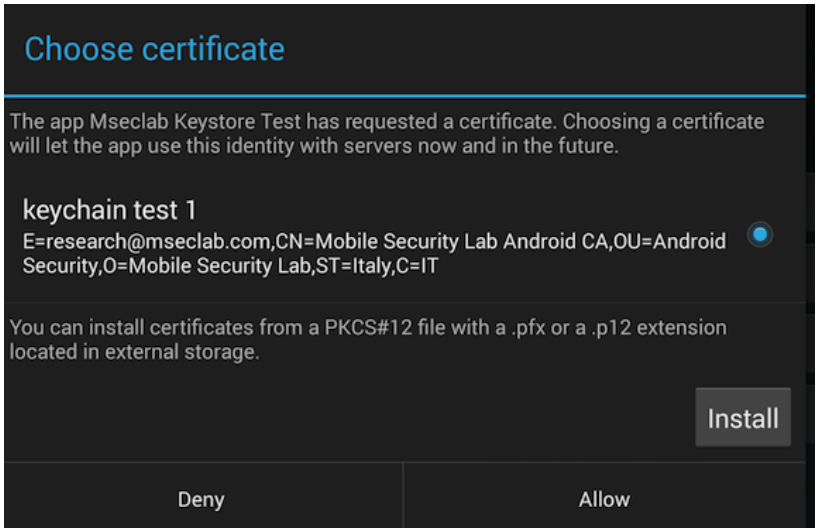
Key Management in Android

- "User" level
  - PBKDF2 (Password Based Key Derivation Function)
    - Algoritmo di generazione (**PBEWithSHA256And256BitAES-CBC-BC**)
    - Salt
    - Numero di Iterazioni

- System Level
  - KeyChain (da >= 4.0)
  - AndroidKeyStore (ufficiale da >= 4.3)

KeyChain e AndroidKeyStore

- KeyChain
  - Accessibile da qualunque applicazione



- AndroidKeyStore
  - **<u>Accessibile alla singola applicazione ed al singolo utente</u>**
  - Memorizza **solo** coppia chiave pubblica/privata RSA 2048

Key Management Evolution

API LEVEL 14

Global Level Only:

Default TrustStore
cacerts.bks

(**ROOTED device**)

Global Level:
KeyChain
(**Public API**)

App Level:
KeyStore
(**Closed API**)

API LEVEL 18

Global Level:
KeyChain
(**Public API**)

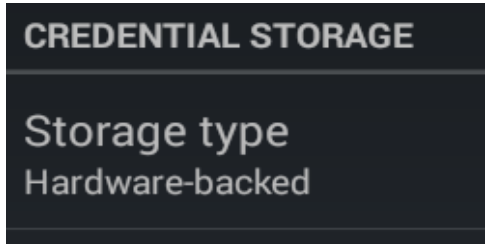
App Level and
per User Level:
AndroidKeyStore
(**Public API**)

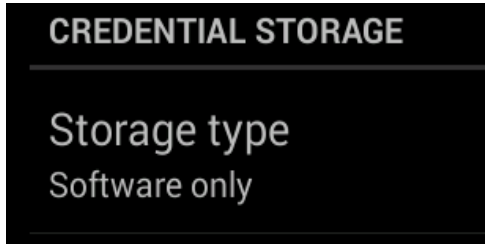
AndroidKeyStore Storage

- Due tipologie di Storage
  - Hardware-backed (Nexus 7, Nexus 4, Nexus 5 :-) con OS >= 4.3)
    - Secure Element
    - TPM
    - TrustZone

TrustZone®
System Security by ARM

  - Software only (Rimanenti dispositivi con OS >= 4.3)





Storage?

KeyChain enhancements

The KeyChain API now provides a method that allows applications to confirm that system-wide keys are bound to a **hardware root of trust** for the device. This provides a place to create or store private keys that **cannot be exported** off the

```
import android.security.KeyChain;

if (KeyChain.isBoundKeyAlgorithm("RSA"))
        // Hardware-Backed
    else
        // Software Only
```


mobile security lab

AndroidKeyStore in pratica

Android Security
Key Management

Cosa faremo

CodeLab diviso in 4 step:

1. Generazioni Chiavi
2. Accesso AndroidKeyStore
3. Firma e Verifica
4. Cifratura/Decifratura

Definizione Specifiche Certificato

```
Context cx = getActivity();
String pkg = cx.getPackageName();
```

```
Calendar notBefore = Calendar.getInstance();
Calendar notAfter = Calendar.getInstance();
notAfter.add(1, Calendar.YEAR);
```

Definizione Paramentri Temporali

```
import android.security.KeyPairGeneratorSpec.Builder;
Builder builder = new KeyPairGeneratorSpec.Builder(cx);
builder.setAlias("DEVKEY1");
String infocert = String.format("CN=%s, OU=%s", "DEVKEY1", pkg);
builder.setSubject(new X500Principal(infocert));
builder.setSerialNumber(BigInteger.ONE);
builder.setStartDate(notBefore.getTime());
builder.setEndDate(notAfter.getTime());
```

ALIAS per indicizzare il certificato

Self-Signed X.509

- Common Name (CN)
- Subject (OU)
- Serial Number

```
KeyPairGeneratorSpec spec = builder.build();
```

Generazione delle specifiche del Certificato

Generazione chiave pubblica/privata

```
KeyPairGenerator kpGenerator;
```

Engine per generazione chiave Privata/Pubblica

```
kpGenerator = KeyPairGenerator
        .getInstance("RSA", "AndroidKeyStore");
```

Istanza Engine con:

- Algoritmo RSA
- Provider: AndroidKeyStore

```
kpGenerator.initialize(spec);
```

Init Engine con le specifiche del certificato

```
KeyPair kp;
kp = kpGenerator.generateKeyPair();
```

- **<u>Generazione chiave Privata/Pubblica</u>**

<u>Dopo la generazione, le chiavi saranno memorizzate in Android Key Store e potranno essere recuperate mediante l'ALIAS</u>

Inizializzazione Android Key Store

```
keyStore = KeyStore.getInstance("AndroidKeyStore");
```

Richiesta di Android Key Store

```
keyStore.load(null);
```

Dovrebbe essere utilizzato nel caso si ha un InputStream da caricare (per esempio il nome di un KeyStore importato). Se non invocato l'applicazione andrà in CRASH

Abbiamo il riferimento *keyStore* che utilizzeremo per accedere alla coppia chiave Pubblica/Privata mediante l'identificativo ALIAS

RSA Digital Signature

- Digital Signature
  - Authentication, Non-Repudiation and Integrity
  - RSA Private key to Sign
  - RSA Public Key to Verify

```
KeyStore.Entry entry = ks.getEntry("DEVKEY1", null);
```

Accesso chiave Pubblica e Privata identificata dall'ALIAS==DEVKEY1

byte[] data = "**DevFest Rome 2013!**".getBytes();

```
Signature s = Signature.getInstance("SHA256withRSA");
```

Scelta dell'algoritmo

```
s.initSign(((KeyStore.PrivateKeyEntry) entry).getPrivateKey());
```

Chiave Privata per firmare

```
s.update(data);
byte[] signature = s.sign();
String result = null;
result = Base64.encodeToString(signature, Base64.DEFAULT);
```

Firma e codifica in Base64

Verify RSA Digital Signature

```
byte[] data = input.getBytes();
byte[] signature;
signature = Base64.decode(signatureStr, Base64.DEFAULT);
```

Decodifica Base64

```
KeyStore.Entry entry = ks.getEntry(“DEVKEY1”, null);
```

Accesso chiave Pubblica e Privata identificata dall’ALIAS==DEVKEY1

```
Signature s = Signature.getInstance("SHA256withRSA");
```

Scelta dell’algoritmo

```
s.initVerify(((KeyStore.PrivateKeyEntry) entry).getCertificate());
```

Chiave Pubblica nel certificato utilizzata per verificare

```
s.update(data);
boolean valid = s.verify(signature);
```

TRUE == Verified
FALSE== Not Verified

RSA Encryption

- Encryption
  - Confidentiality
  - RSA Public key to Encrypt
  - RSA Private key to Decrypt

```
PublicKey publicKeyEnc = ((KeyStore.PrivateKeyEntry) entry)
                                        .getCertificate().getPublicKey();
```

Accesso alla chiave pubblica per cifrare

```
String textToEncrypt = new String("DevFest Rome 2013");
byte[] textToEncryptToByte = textToEncrypt.getBytes();
Cipher encCipher = null;
byte[] encryptedText = null;
```

```
encCipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
encCipher.init(Cipher.ENCRYPT_MODE, publicKeyEnc);
```

- Scelta algoritmo
- Encryption con chiave pubblica

```
encryptedText = encCipher.doFinal(textToEncryptToByte);
```

Testo Cifrato

RSA Decryption

```
Cipher decCipher = null;
byte[] plainTextByte = null;
```

```
decCipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
```

Scelta algoritmo

```
decCipher.init(Cipher.DECRYPT_MODE,
        ((KeyStore.PrivateKeyEntry) entry).getPrivateKey());
```

Decryption con la chiave privata

```
plainTextByte = decCipher.doFinal(ecryptedText);
```

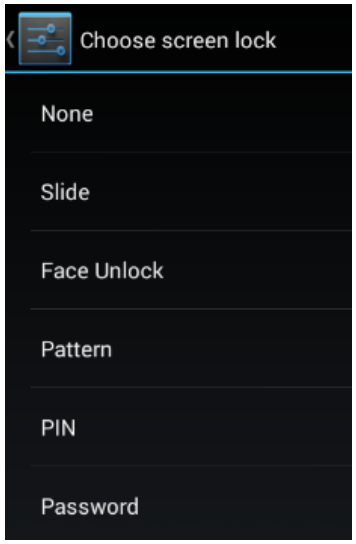
Decryption con chiave privata

```
String plainText = new String(plainTextByte);
```

Testo decifrato

Si osserva che...

- Vari tipi di screen lock



- La scelta di screen lock impatta sulla persistenza delle chiavi generate

- Cambiando il tipo di screen lock le chiavi vengono cancellate

```
No key found under alias: DEVKEY1
Exiting signData()...
Signature: null
```

Comportamento atteso?

- La documentazione riporta

The KeyPair generator will create a self-signed certificate with the subject as its X.509v3 Subject Distinguished Name and as its X.509v3 Issuer Distinguished Name along with the other parameters specified with the `KeyPairGeneratorSpec.Builder`.

The self-signed X.509 certificate may be replaced at a later time by a certificate signed by a real Certificate Authority.

- Le chiavi non dovrebbero essere cancellate quando il tipo di screen lock viene cambiato dall'utente

Cryptographic stuff on devices

- Device con Storage “Hardware-backed”

```
shell@grouper:/data $ su root
root@grouper:/data # cd /data/misc/keystore/user_0/
root@grouper:/data/misc/keystore/user_0 # ls -la
-rw------- keystore keystore       84 2013-11-08 06:15 .masterkey
-rw------- keystore keystore      804 2013-11-07 10:03 10069_USRCERT_DEVKEY1
-rw------- keystore keystore       84 2013-11-07 10:03 10069_USRPKEY_DEVKEY1
-rw------- keystore keystore      724 2013-11-07 11:01 10070_USRCERT_myKey
-rw------- keystore keystore       84 2013-11-07 11:01 10070_USRPKEY_myKey
root@grouper:/data/misc/keystore/user_0 #
```

- Device con Storage “Software-only”

```
root@generic:/ # cd /data/misc/keystore/user_0/
root@generic:/data/misc/keystore/user_0 # ls -la
-rw------- keystore keystore       84 2013-11-05 12:35 .masterkey
-rw------- keystore keystore      724 2013-11-05 13:04 10050_USRCERT_myKey
-rw------- keystore keystore     1524 2013-11-05 13:04 10050_USRPKEY_myKey
-rw------- keystore keystore      804 2013-11-08 01:16 10051_USRCERT_DEVKEY1
-rw------- keystore keystore     1524 2013-11-08 01:16 10051_USRPKEY_DEVKEY1
root@generic:/data/misc/keystore/user_0 #
```

References

- http://developer.android.com/about/versions/android-4.3.html#Security
- http://developer.android.com/reference/java/security/KeyStore.html
- http://en.wikipedia.org/wiki/Encryption
- http://en.wikipedia.org/wiki/Digital_signature
- http://nelenkov.blogspot.it/2013/08/credential-storage-enhancements-android-43.html
- http://nelenkov.blogspot.it/2012/05/storing-application-secrets-in-androids.html
- http://nelenkov.blogspot.it/2012/04/using-password-based-encryption-on.html
- http://nelenkov.blogspot.it/2011/11/ics-credential-storage-implementation.html
- http://developer.android.com/reference/android/security/KeyPairGeneratorSpec.html

Grazie
Q&A

www.mseclab.com
www.consulthink.it
research@mseclab.com

Requisiti

- Portatile
- Eclipse con ADT Plugin 22.3.0
- SDK Android 4.4 (API 19)
- Android SDK Build-tools 19