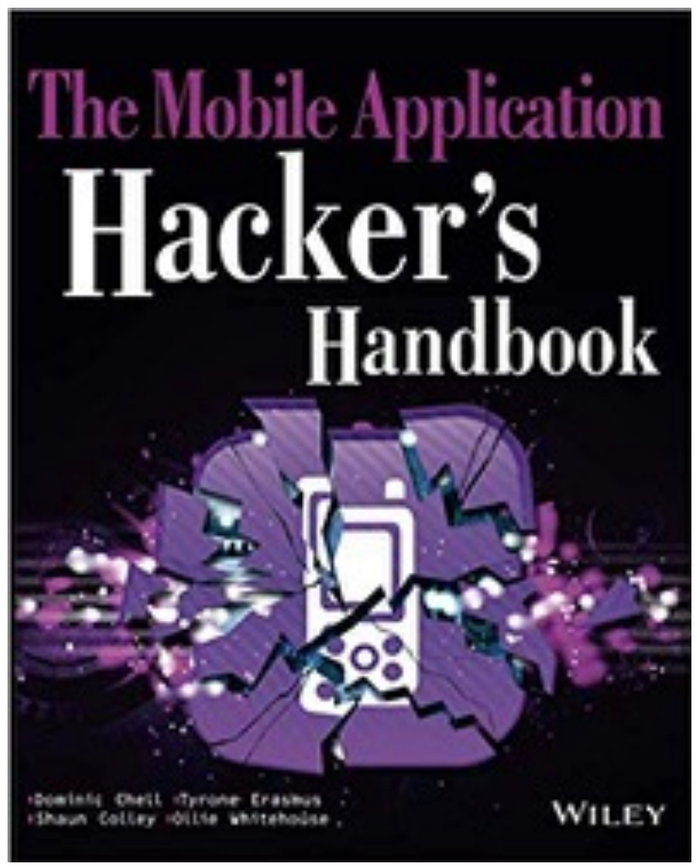
CNIT 128
Hacking Mobile Devices



8. Identifying and Exploiting Android Implementation Issues Part 2

Updated 3-24-2021

Topics

- Part 1
  - Reviewing Pre-installed Applications
  - Exploiting Devices
    - Start through "Explanation of Privilege Levels" (up to p. 402)

Topics

- Part 2
  - Exploiting Devices
    - "Practical Physical Attacks" (p. 375) through
    - "Polaris Viewer Memory Corruption" (up to p. 402)

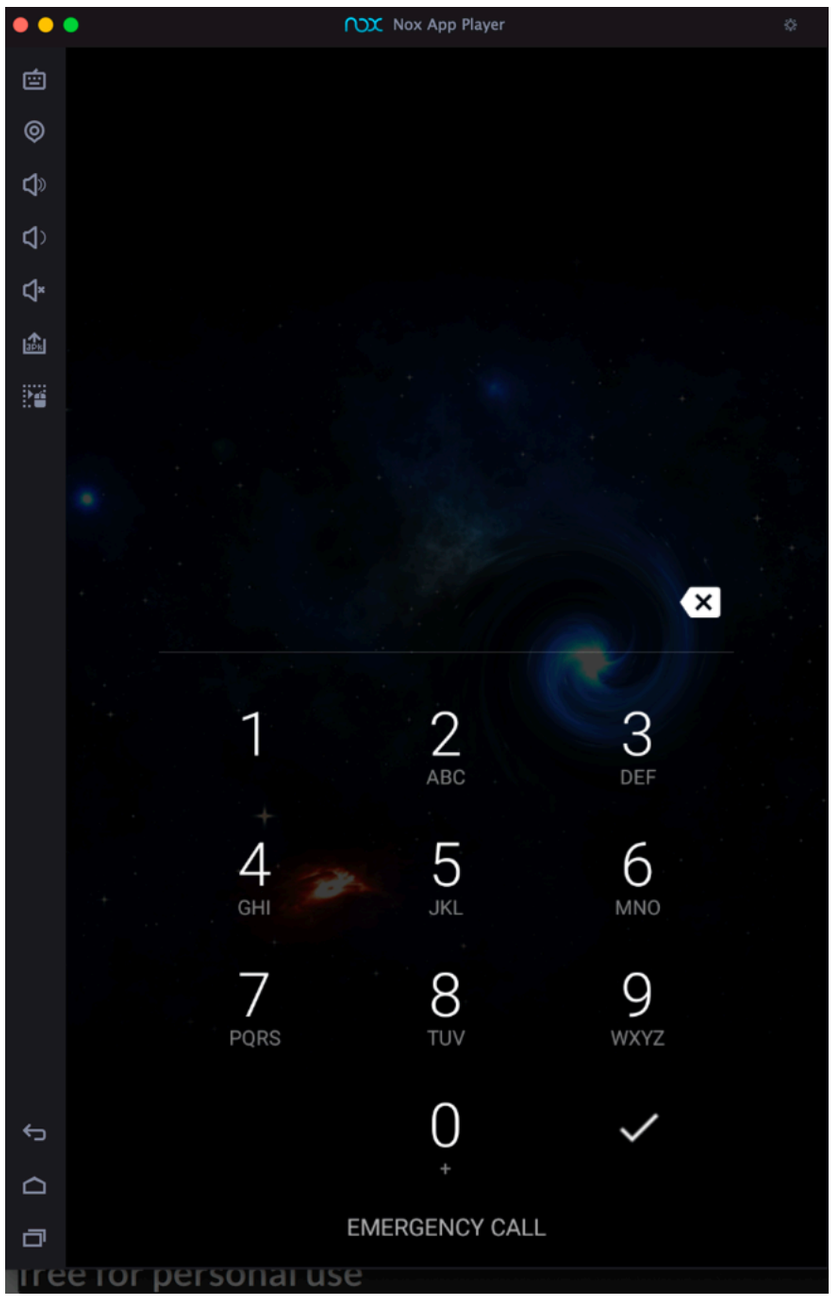
Topics

- Part 3
- Exploiting Devices
  - "Injecting Exploits for JavaScript Interfaces" (p. 402) and following
- Infiltrating User Data

Practical Physical Attacks

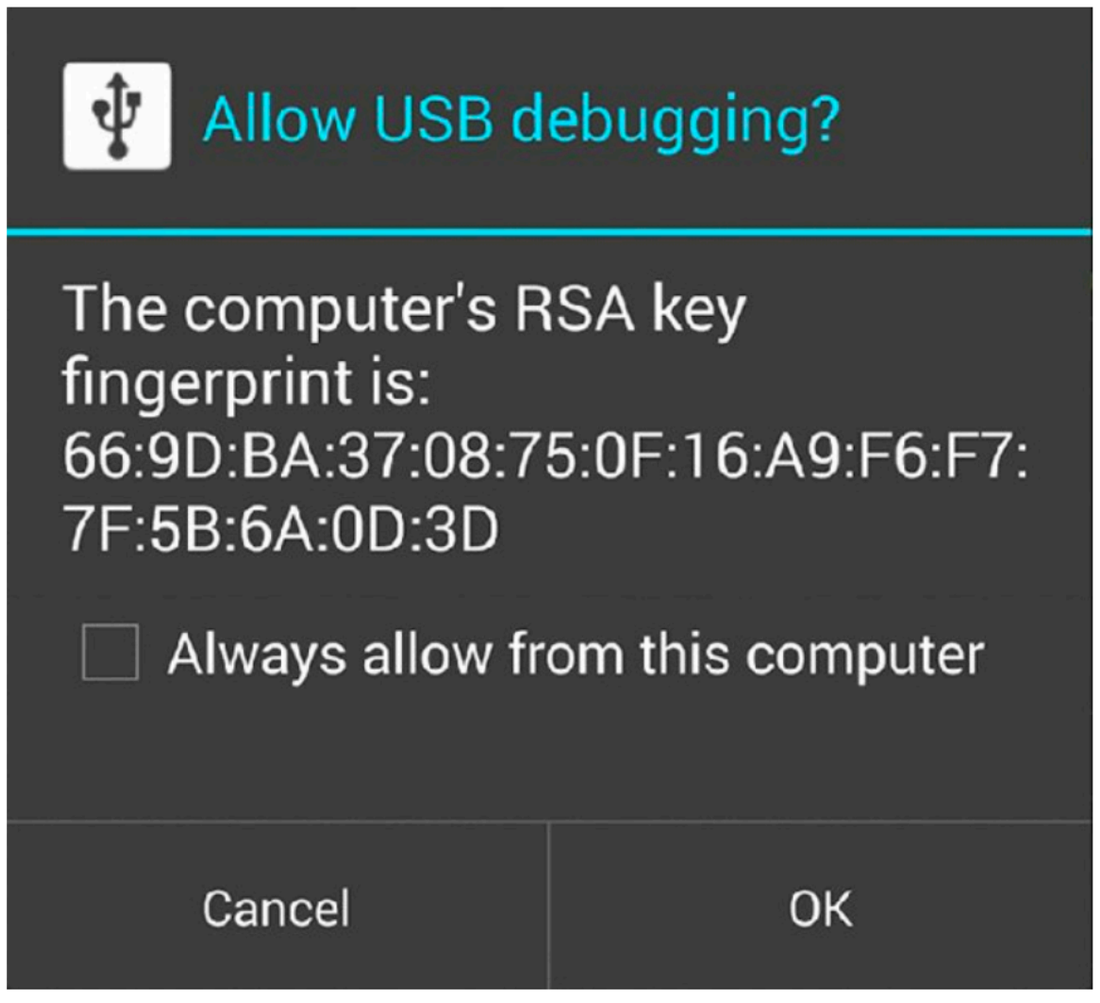
Bypassing Lock Screen

- Getting ADB Shell Access: Two Ways
  - USB Debugging
  - Unlocked Bootloaders



USB Debugging

- **adb shell**
- Usually turned off by default
- Exposes data and can be used to install new packages
- User must approve it
- Not possible if screen is locked



Bug

- In Android 4.2.2 up to 4.4.2
- Navigating to emergency dialer or lock screen camera
- Shows the USB debugging authorization prompt
- With the screen locked

Privilege Levels

- **/default.prop** file controls ADB privileges
  - By default: **ro.secure=1**
    - Adb runs as the shell user
  - If **ro.secure=0**
    - adbd runs as root

```
vbox86p:/ # cat default.prop
#
# ADDITIONAL_DEFAULT_PROPERTIES
#
ro.actionable_compatible_property.enabled=false
ro.secure=1
```

ALLOW_ADBD_ROOT

- From Android 4.3 onwards
- ADB won't run as root unless it's compiled with the **ALLOW_ADBD_ROOT** flag
- Even if **ro.secure=0**
- To get root, compile a custom version of adbd and overwrite the binary on the device

Unlocked Bootloaders

- First boot phone into Fastboot mode
  - Hold down power and volume keys while turning on the phone
    - or **adb reboot bootloader**
- Then flash or boot a custom image
- Not possible with Genymotion, Nox, or Bluestacks
  - They have no recovery partition

Unlock Bootloader

- Forces factory reset
- Wipes all user data
- To stop thieves

```
$ fastboot oem unlock

...

(bootloader) erasing userdata...

(bootloader) erasing userdata done

(bootloader) erasing cache...

(bootloader) erasing cache done

(bootloader) unlocking...

(bootloader) Bootloader is unlocked now.

OKAY [ 40.691s]

finished. total time: 40.691s
```

Unlocked Bootloader

- If user unlocked the bootloader and left it unlocked
- Boot into ClockworkMod Recovery ROM (not updated since 2014)
  - Get a root ADB shell

Bypassing Lock Screens

- Using the DISABLE_KEYGUARD Permission
  - Allows an app to remove the lock screen
- App code:

```
KeyguardManager kgm =
((KeyguardManager)getSystemService("keyguard"));

KeyGuardManager.KeyguardLock kgl =
kgm.newKeyguardLock("mahh");

kgl.disableKeyguard(); Even though the
KeyguardManager.KeyguardLock
```

Custom Drozer Agent

`drozer agent build --permission android.permission.DISABLE_KEYGUARD`

- Install agent with adb, launch it, and bypass screen lock
- Works on Nox (Android 5.1.1)
- Fails on Genymotion (Android 9.0)

```
dz> run post.perform.disablelockscreen
[*] Attempting to disableKeyguard()
[*] Done. Check device.
dz>
```

Removing Key Files

- Pattern lock screen uses data from
  - **/data/system/gesture.key**
- PIN or password lock uses data from
  - **/data/system/password.key**
- Removing these files disables lock screen entirely

Removing Key Files

- But that requires running as **system** or **root**
- Privilege escalation

```
-rw------- system   system    20 2014-11-03 15:10 gesture.key
...
-rw------- system   system    72 2014-11-03 15:10 password.key
```

Abusing Android Application Issues

- On Android 4.3 and earlier, this intent unlocks the phone from an adb shell in any context:

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

Starting: Intent { flg=0x8000
cmp=com.android.settings/.ChooseLockGeneric (has
extras) }
```

Thought your Android phone was locked? THINK AGAIN

Another day, another vulnerability

By Richard Chirgwin 10 Dec 2013 at 22:58 49 SHARE ▼

In fact, the Curesec post states, the bug – present in Android 4.0 to 4.3 but not 4.4 – exposes any locking technique: PINs, passwords, gestures or facial recognition.

“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,” Curesec writes.

A user changing the type of lock they're using should have to enter their previous lock – so if you swap from PIN to gesture, you would have to provide your PIN before Android allows the change.

The problem is that the program flow in the ChooseLockGeneric class lets an attacker bypass the confirmation:

“As a result, any rogue app can at any time remove all existing locks.”

Using Logic Flaws that Don't Require Shell Access

- Some actions are allowed when a phone is locked
  - Make emergency phone calls
  - Receive phone calls
  - Allow third-party apps to temporarily disable the lock screen
    - Or place an activity in front of it

Motorola Droid

- Phone the locked device
- Answer the call
- Press Back button
- Escape the lock screen

Viber

- Messaging and calling app
- Place a Viber call
- Answer it
- Press the back button multiple times
- Escape the lock screen

Using Legitimate Lock Screen Reset Functionality

- You can bypass a lock screen if you have the use's Google credentials
- Entering the PIN five times incorrectly
- Connects to a linked Google account
- But only for Android 4 and earlier

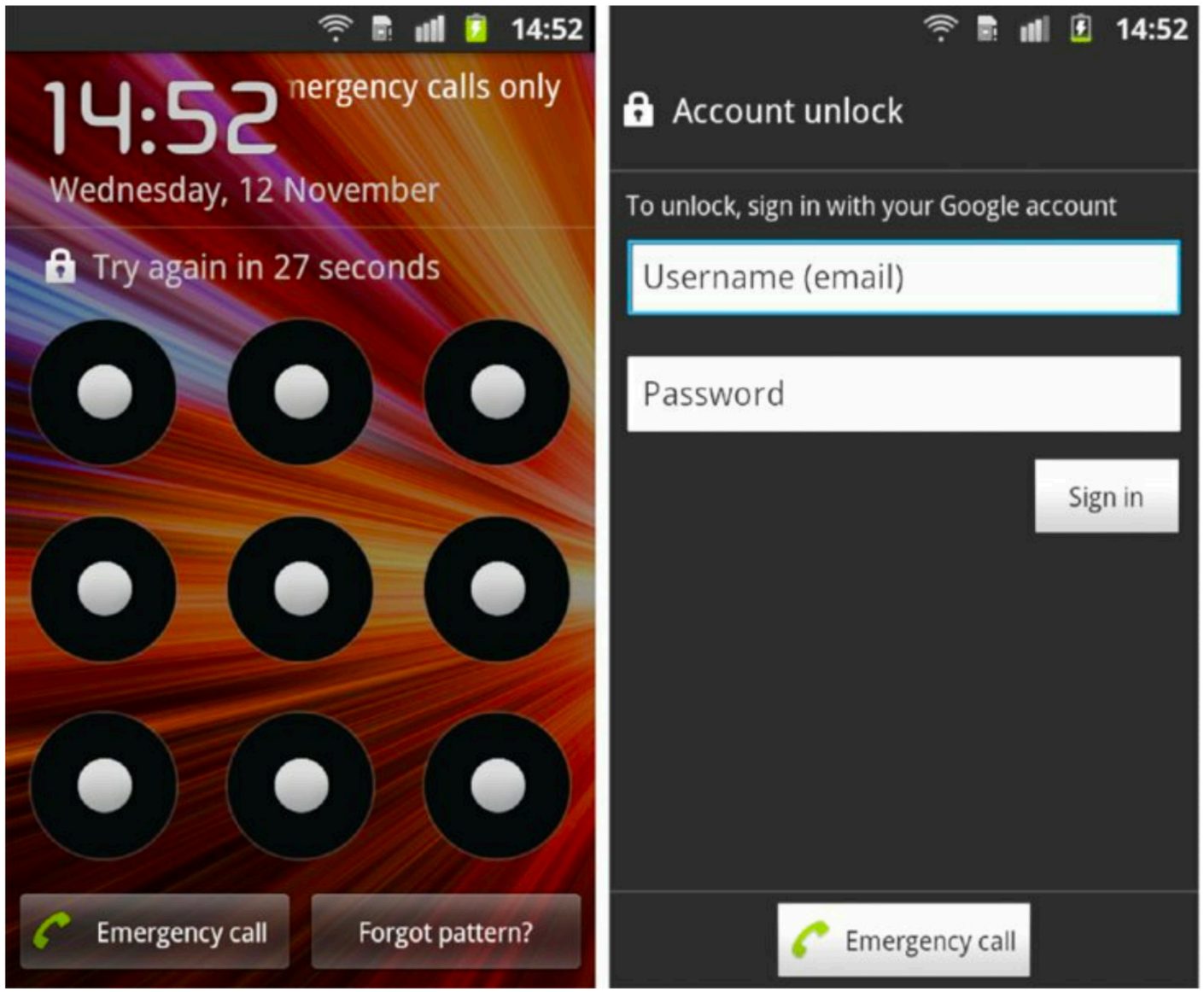


Figure 8.3 Showing the Forgot pattern? button and the resulting screen by pressing it

Android Device Manager

- Allows lock screen to be bypassed from a Google account
- Not active by default--user must enable it
- And it erases all your data (link Ch 8b)

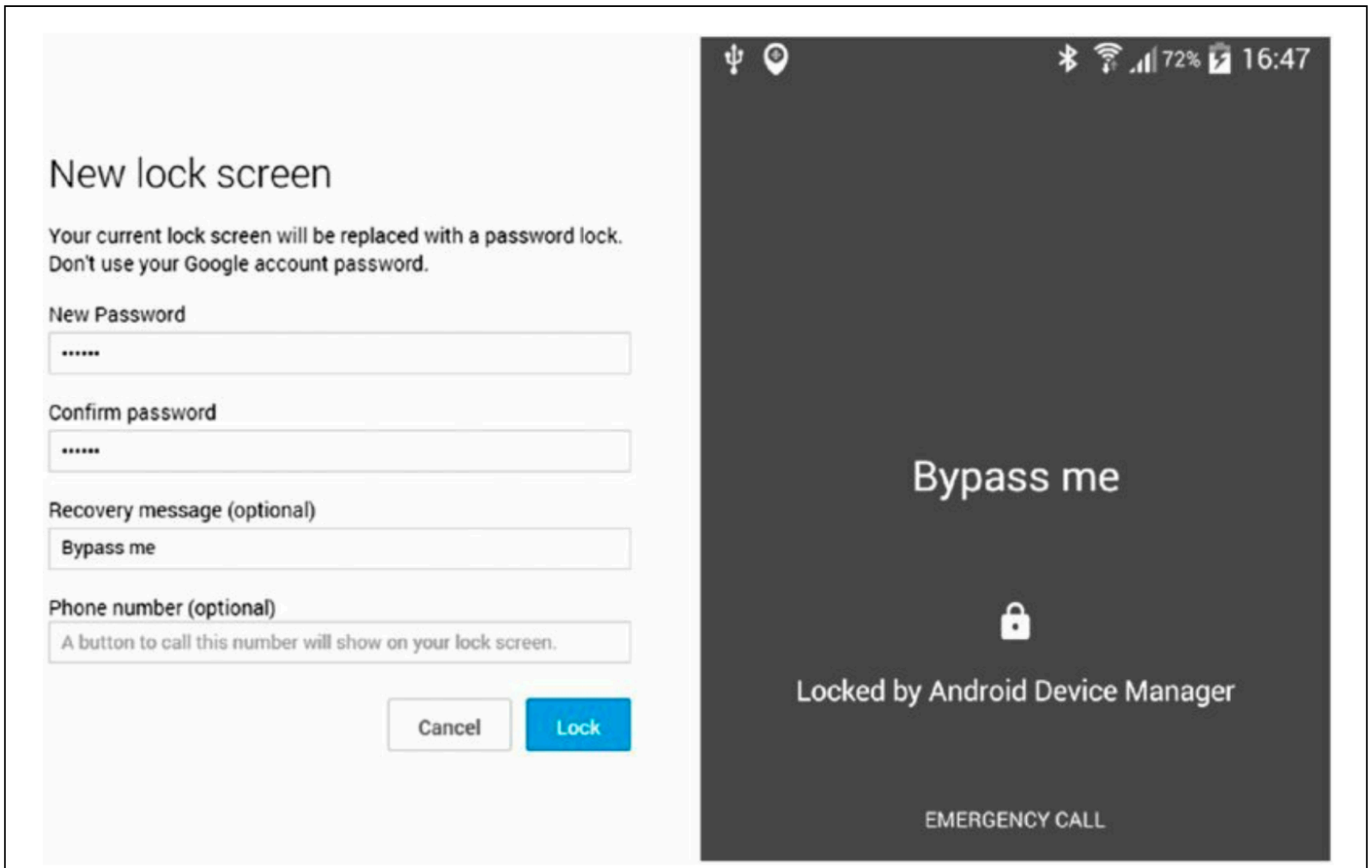


Figure 8.4 The Android Device Manager Lock functionality and the resulting screen of the locked device

Practical Remote Attacks

Remote Exploits

- Launched over the Internet
- Three modes of exploitation:
  - Loading a drozer JAR that loads a limited agent
  - Installing and starting a rogue drozer agent by abusing I**NSTALL_PACKAGES**
  - Loading a drozer JAR that is passed **Context**

Browser Memory Corruption

- The most technical method
- Reverse shells via buffer overflow, etc.
- Becoming rarer as the browser becomes more secure

Polaris Viewer Memory Corruption

- Office and PDF file viewer
- Pre-installed on some devices
- Exploited in 2012 with a crafted DOCX file
  - Using a stack-based buffer overflow
- App also had **INSTALL_PACKAGES** permission

Android Browser JavaScript Interface

- All WebViews using **JavaScriptInterface**
  - And targeting API before 17
    - Are vulnerable to code execution flaws
- Includes all stock Android browsers on Android 4.1.1 and below
- Can get **Context** and use full permissions of browser

Use Metasploit

Next we define the Webview exploit:

```
msf > use exploit/android/browser/webview_addjavascriptinterface
```

- Link Ch 8c

```
msf exploit(webview_addjavascriptinterface) > set LHOST 192.168.0.24
LHOST => 192.168.0.24
msf exploit(webview_addjavascriptinterface) > show options

Module options (exploit/android/browser/webview_addjavascriptinterface):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   Retries  true             no        Allow the browser to retry the module
   SRVHOST  0.0.0.0          yes       The local host to listen on. This must be an address on the local machin
e or 0.0.0.0
   SRVPORT  8080             yes       The local port to listen on.
   SSL      false            no        Negotiate SSL for incoming connections
   SSLCert                   no        Path to a custom SSL certificate (default is randomly generated)
   URIPATH                   no        The URI to use for this exploit (default is random)
```

Privilege Escalation

- Exynos driver exploit on some devices
- Drozer has a **exploit.mmap_abuse** module
  - Tries to get root by abusing the map device operation
  - Similar to the exynos exploit
  - Dangerous: may cause a kernel panic and reboot the device

Maintaining Access

- Must install a special **su** binary bundled with Drozer, called "**minimal su**"
- Allows every app to escalate to root
- Without alerting the user

Man-in-the-Middle Exploits

- Getting in the middie
- Host a wireless network
- ARP Poisoning
- Use Burp

Man-in-the-Middle Exploits

- Intercepting SSL requires the ability to get a root CA certificate
  - Only nation-states or very rich attackers
- But many apps ignore this defense and allow MITM attacks anyway


Kahoot!