FFI-rapport 2012/00275

Data leakage from Android smartphones

Lasse Øverlier

Norwegian Defence Research Establishment (FFI)

6 June 2012

FFI-rapport 2012/00275

1126

P: ISBN 987-82-464-2115-5
E: ISBN 987-82-464-2116-2

Keywords

Mobiltelefon

Datainnsamling

Dataoverføring

Approved by

Kjell Olav Nystuen	Project manager
Anders Eggen	Director

English summary

We are in this report giving an overview of what information a smartphone is leaking through normal usage. Our experiment intercepted all traffic, both plain text and encrypted traffic. From this data we examined which identifiers a smartphone used when we instructed it to not give away any location information at all. Most critical parameters sent from the phone include IMEI, WLAN-name, MAC-addresses, phone network operator and country, phone name and version, serial numbers, operating system version, kernel version and debugging information with system identificators. We also found a recommendation to install a phone operating system update from an unsecure site, on an unsecure connection, with the option to force the update to take place.

Sammendrag

Vi viser i denne rapporten hvordan en smarttelefon lekker identifikatorer gjennom vanlig bruk. Dette gjøres gjennom å fange opp og analysere all trafikk til og fra en smarttelefon, inkludert kryptert trafikk. Eksperimentene viser at telefonen sender fra seg identifikatorer som IMEI, navn på trådløst nettverk, adresser til trådløs basestasjon, telefonoperatør og land, telefonnavn og -modell, versjoner av hardware og programvare, serienummer, loggfiler og debuggingsinformasjon. Vi fant også et tilfelle av å bli oppfordret til å installere en usikret oppdatering på en usikker forbindelse med mulighet for en angriper til å tvinge en valgt oppdatering til å skje på smarttelefonen.

Contents

1 Introduction and background

Using communication devices plugged into the Internet, we enable these devices to reach everywhere and communicate with devices, and computer services, at any location. This is perhaps why we have adopted these devices and integrated them into our way of life so quickly - they keep us updated, let us publish information, communicate with friends, etc. And we think we know how having such a device is a risk to us. If an attacker steals and/or compromises the security of our device we are aware of the fact that the attacker might be able to gain information about ourselves that we might not want to have published. This could be information like pictures and videos, SMS, complete lists of our social media contacts, lists of phone numbers and phone contacts, our email correspondence, private social media messages, personal/private information in local documents, calendar information, etc. This is a risk many smart phone users are comfortable with, as they trust their own (physical) protection of the device. Simple protections like pin- and password codes, pattern screen locks and device control are usually considered to be good enough to achieve a comfortable level of trust among most users.

As smart phone users we also expect that the information we put into the device will remain there unless we instruct the phone to give it to someone else. And we also trust the security mechanisms mentioned above to enforce this.

Unfortunately this is not always the case. By using smart phones we also put our trust into the hands of the vendors. After all if you cannot trust the vendors of the hardware and the software running on your devices, whom can you trust? This is similar to the use of computers where you ultimately have to trust the production of every part inside, including the microchips, the main board assembler, the BIOS vendor, the operating system, the applications installed, and others. Some countries have even started to act on the lack of confirmation of backkdoor-free hardware components[15, 27, 9].

In this report we have looked beyond the physical protection and into the trust we put into the operating system vendor and the smart phone applications on top of the smartphone operating system. E.g. a Samsung phone running Android OS with Samsung's TouchWiz interface and preinstalled applications from Samsung and others. We have looked into what kind of information does the "clean" phone share with whom, during startup and normal configuration. The smartphones were during first power-on instructed to not release any location information, neither from GPS nor from wireless networks, and we have completed this for smartphones from multiple vendors and multiple versions of the Android operating system.

2 Tools and related work

Just listening to communication traffic in today's networks is easy when you have physical access to the communication channels, and there are plenty of tools for achieving this. A lot of the information sent and received on your local network is unencrypted, meaning that access to

the communication channel is access to the actual data being transferred. More and more applications and services do involve sensitive and/or private data and there is a significant increase in awareness of this, and towards using encryption to protect the data in communication.

But encryption involves trust, and as shown below this trust model can be abused by a malicious adversary. We will here describe some methods of how this trust can be manipulated, and how access to the decrypted content can be achieved. This is explained in the introduction to the experiment in the next section. In this section we will explain the previous experiments and attacks performed leading up to our experiment. We will start with how Man-in-the-middle attacks works and look at how to abuse the trust model for web traffic, then we will provide some information about prior experiments and other methods used by others.

2.1 HTTPS Man-in-the-middle attacks

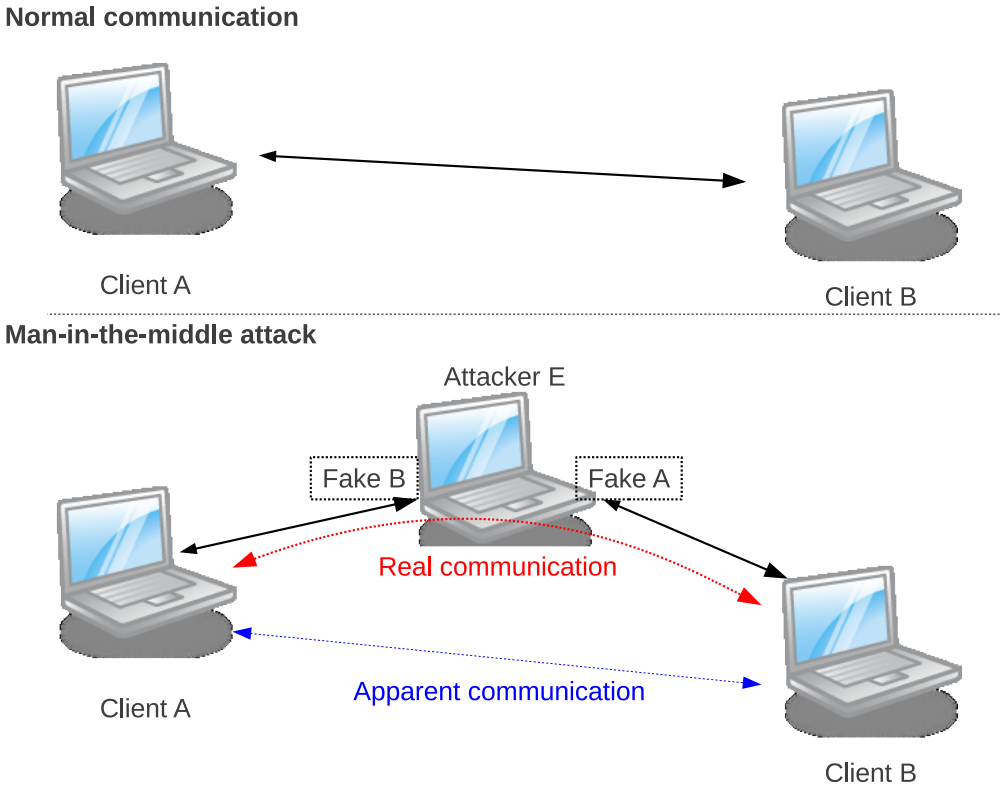

Figure 2.1 Man-in-the-middle attack

A man-in-the-middle attack works as shown in Figure 2.1. One client, A[1], is communicating with a service or another client, B, on the Internet over a communication channel as shown in the top half of the figure.

If you have control over the communication channel, no information from the communication between A and B is lost to others. But if anyone has access to the communication channel,

[1] We have used the letters from common information security litterature of Alice (A) and Bob (B) communicating and Eve (E) trying to attack the communication.

i.e. an Internet service provider or an attacker, there can be data leakage. Therefore we enforce extra security mechanisms, like encryption, on the communication channel or content while being transferred.

If the communication channel is encrypted in a secure way, all an attacker will ever observe is encrypted information. It may still be a man-in-the-middle, but the effect is that it will not see the content, only the amount of data and the communication patterns. It achieves nothing more than a network sniffer listening to the direct traffic.

The most common protocol described above, HTTPS, is achieving secure web traffic by using the unencrypted HTTP[11] protocol over an TLS-encrypted communication channel[8]. The HTTPS protocol is using certificates to verify the authenticity of the web service.

In normal secure web traffic we achieve the security between the two computers by having the web server (here B) send out a certificate signed by a trusted third party. This certificate authenticate B to the client A, as long as the client also has trust in the same third party. But in here lies the problem. There is some predefined and preconfigured trust involved in setting up encrypted channels.

If we can make the client accept another trusted third party that the attacker controls (in addition to the ones already registered), the client will accept all certificates issued by the attacker.

Having this new trust in the attacker the client is extremely vulnerable. It is now easy to set up a web site e.g. looking like the clients real on-line bank and fake every action done. It will also be easy to get login credentials from all web services that the client uses by faking their login screens, recoding the submitted data, and just issuing a message like “Internal problems. Try again later.”.

But for our purposes we will not change the content[2] of the communication - only have access to the decrypted data. Therefore we put another computer between the two communicating computers, the attacker E. This can be performed by doing a DNS-poisoning, or simply being in the communication chain like we are in this experiment (c.f. Figure 3.3). This way we make the client use two “hops” to get to the real server as shown in Figure 2.1. The first hop goes to the attacker’s computer, E, which serves the client with a certificate signed by the attacker, and that the client trusts. This enables the attacker to have A believe that it is talking to B (marked with “Fake B” in the figure) and be an endpoint of the communication channel. Simultaneously the attacker connects to the real B service and just forwards all information sent from A. The response from B is also forwarded to A and thereby E has access to the decrypted data from the SSL-encrypted information sent through it.

It is important that E gives the impression of being B when talking to A, meaning that all certificates must work as if the client was talking to the service directly. The inserted trusted third

[2] We are aware that changing the certificate is a modification of traffic, but not of the traffic we are interested in logging and analyzing.

party certificate will make the client accept all fake certificates from the attacker generates. We will describe how to insert a trusted third party certificate in Section 2.8.

There are two levels of vulnerabilities that are present at this stage already, but that we will not look further into.

The first is the possibility to observe traffic content. Just having access to unencrypted information in HTTP-headers reveals a lot of information about the communicating parties - both for the client browser and for the server.

And second. Just being a part of the communication makes the attacker able to inject, modify or delete content from the communication, and is a highly potent way of attacking a system. Just being able to redirect to other sites, inject script code to steal information and submit automatically, inject script code to dig out indentifiers, inject attack code targeted specific versions of the communicating software, etc.

2.2 sslsniff

In order to communicate with clients and servers using the TLS encryption protocol, we are in need of some libraries or tools assisting us. The most used open source library for setting up TLS connection is OpenSSL[25], a C-library, but this means building a server version and a client version ourselves. There are already tools built for this purpose and many are also open source and free to use. One suitable tool available for making such attacks possible in a easy-to-use way is *sslsniff*[19] by Moxie Marlinspike. The program *sslsniff* is built exactly for the purpose and scenarios described above. In addition it will allow for an attacker to intercept and modify the encrypted communication channel. It is a (almost[3]) fully functional working man-in-the-middle tool for TLS/SSL-encrypted traffic and makes sure in certain scenarios that neither the client nor the server should be aware that there is a man-in-the-middle attack going on, and that they will not discover that someone is listening in on the encrypted communication. This tool was built to be a man-in-the-middle interceptor and observe and possibly change and inject traffic to/from a client-server connection targeting Microsoft's SSL-chain vulnerability[17].
It has also been extended to compromise vulnerabilities like the null-prefix attacks[22] and ocsp-attacks[18].

The tool works exactly as shown in Figure 2.1 where E is running the *sslsniff* software. The communication towards the server, B is completed using the server's real certificate as the attacker has full control of the content we are sending and receiving here. The communication with the client is based on the fact that *sslsniff* on-the-fly constructs a false certificate for itself, E, signed by an authority trusted by the client, A. As long as E communicates to B on A's behalf, it is unlikely that neither A nor B will ever detect the man-in-the-middle computer.

Due to *sslsniff*'s flexibility to hand out new false (but still trusted) certificates on-the-fly to a

[3]The version we started to use has had quite a few bugs. Some we patched ourselves, some have been in later updates of the *sslsniff* software. We used version 0.06 of *sslsniff*.

client in a client-server connection, it has become a very common tool for intercepting SSL-traffic and enabling the listener to analyze the decrypted information.

There have been some countermeasures towards the use of *sslsniff*. One of these countermeasures comes from looking at the certificate dates and see if they have been recently generated (dated). This could from an attacker be easily countered by reusing certificate parameters, like the date of signing, from the real service. Another more efficient countermeasure will be that clients, especially apps which can be updated often, can contain hardcoded information about the valid certificates of its most used services. This breaks the flexibility built into the SSL/TLS trust/certificate scheme, but it is very likely something that some vendors will try even harder to protect its communication content.

2.3 Galaxy tab experiment

First real data leakage experiment from a researcher was given in January 2011 on a blog called "Big Brother Mobile Blog"[2].

This experiment intercepted the traffic to and from a Galaxy Tab 7 (first generation) running Android version 2.2. The setup here was a *squid proxy*[12] with some scripts giving out fake SSL certificates to the client, similar to the *sslsniff* program.

The results in this experiment are taken when the user has allowed the current or latest registered location to be transmitted. This means that the user is allowing the use of his/her location to be sent as a parameter to the service for use in the evaluation of the action wanted. E.g. find the best restaurants closest to where I currently am located.

The analysis shows many interesting elements in the intercepted data found in the communication, including IMEI, device type, kernel information, etc. Many of these are described in Section 3 as they are a part of our traffic analysis as well.

A deeper analysis of the Google Maps application and its use of the available location information is also performed in this experiment.

2.4 Catching auth tokens

At the University of Ulm in Germany there were published warnings about sniffing traffic from Android devices. The announcement called "Catching auth tokens"[14] in May 2011 got high press coverage but was really a specification of protocols that still used cookies and site parameters over unencrypted channels. The focus was on gathering such tokens on common WiFi areas and getting access as logged in users to the vulnerable services.

The proposed solution to the vulnerability is to use HTTPS for all protocol communication, which has been the patch for most of the applications vulnerable to data leakage. This is OK for the WiFi-attack, but will not stand against the SSL-certificate attacks which works in a similar way as described in the *sslsniff* section.

2.5 Cross-application scripting vulnerability

CVE-2011-2357[23] describes a method to execute java scripts in the context of another domain ("as if it were located on another web page"). A good example of this attack is given in a brief paper/advisory by Hay and Amit [13]. The attack can be summarized in two ways. First method is to fill up the number of "TAB"s in a browser before loading the script and the browser will mistake the script to be in the previous domain(site). The other method is to load the script so quickly after the previous page that the browser is unable to set a new front window/tab and therefore makes the browser application believe it is still in the same security domain.

This vulnerability has been patched in Android version 2.3.5 and 3.2.

2.6 Android versioning

One major problem with Android is the rapid launch of new versions, without enforcing all phones to use the newest version[16]. This is up to the phone manufacturer when (and even if) they want to implement the newest version.

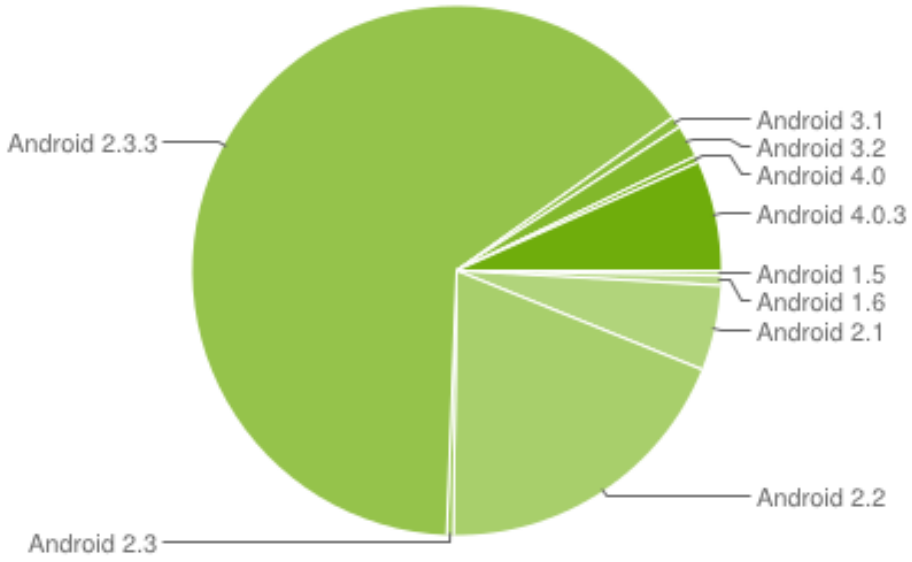

Figure 2.2 Android version[7]. "Data collected during a 14-day period ending on June 1, 2012".

As shown in Figure 2.2 there is a significant number of old versions still running on a huge number of Android devices.

2.7 Rooting phones

In this experiment's startup phase, middle to late 2010, the rooting of a smartphone running Android was an extensive procedure. For each phone the procedure involved both finding a

vulnerability for this specific smartphone, exploiting the vulnerability and gaining root access. Then using this root access to install some tools and/or modify the system.

During 2011 most of the exploits found in Android were made into public tools so the process got easier and easier. The newest tools, like SuperOneClick[4], is installed on a computer with Linux, Windows or MacOS, and gains root access on almost all phones with "one click". The application package also includes the Superuser[3] package which include precompiled shell tool components. These shell tools are very useful for navigating and accessing the system, and they are normally removed from the stripped Linux system which Android installations really are.

Some of the other known tools that are available for rooting phones are Superboot[21] demonstrating an early rooting process shown in Appendix D, GingerBreak[6], Z4Root[26] and unrEVOked 3[24].

The procedure we had to go through manually before tools like SuperOneClick were available is also described in Appendix D as a log from one of these rooting installations.

2.8 Installing trusted certificates

Once root was achieved on the device we used the access to install some additional trusted certificates. There are many ways to get to the data inside the encrypted communication channel, but we found this to be the method that changed the device the least and was a relatively minor change of one file.

There have been significant changes in the methods for installing a certificate onto an Android smartphone since 2010. The main methods include a change of the file "/system/etc/security/cacerts.bks" to include our own top level certificate with signature priveleges, but the difficult part has been to replace the original with this new version. We will first describe how to add a certificate to this file, then describe the methods involved for installing the new certificate file.

Adding a certificate to "cacerts.bks"

To modify a certificate we use the software called *keytool*[4], which will make modifications on a certificate storage file, a `.bks`-file, from command line. For a "cacerts.bks" certificate file protected with the common password "changeit" (as were the case with most early versions of Android), the command will be like this:

```
$ keytool -keystore cacerts.bks -storetype BKS -storepass changeit\
          -provider org.bouncycastle.jce.BouncyCastleProvider     \
          -alias MyCA -trustedverts -importcert -file MyCA.crt
```

[4] http://docs.oracle.com/javase/1.5.0/docs/tooldocs/solaris/keytool.html

But the BouncyCastle library used in conversion of formats must be downloaded and installed in the Java runtime hierarchy.

After this modification and adding of the new certificate, the new “cacerts.bks” can be installed back on the smartphone as described below.

Installing the new “cacerts.bks”

There are two main ways of installing the new “cacerts.bks”. Having root access and changing the file system directly, and installing a new ROM with a new/updated version of the Android operating system.

The first one is the most simple and easy to perform. All you need is root access to the phone, which can be achieved in multiple ways as described in Section 2.7. Then you can replace the “cacerts.bks” file with our new version. For many phones this is all it takes and we are able to reload the new cacert.bks when the smartphone starts up, but not for all phones. The problem with some smartphones is that that the “/system” file system is rewritten upon startup from a file system image, and the fact that these phones cannot reload the cert file into the operating system after first having started up. It needs to be loaded upon boot.

When this is the case we need to change the file system image, which is written to “/system” upon every startup. We have to change the “/system” file system image, one of the images in the Android ROM-file installation, and flash the phone with the new (modified) version of the operating system. Fortunately these images are readable and can be unpacked, modified and then repacked again. But we need to have a tool for flashing the smartphone that does not require valid signatures from the smartphone manufacturer as we are unable to fake these. One of the best tools for flashing new ROM-files is the ClockWorkMod’s *ROM Manager*[10], a backup/recovery/flash tool that is available for almost all Android smartphones with root access today. More and more phones are now allowing the use of recovery tools and flasher tools by “unlocking” the bootloader. By forcing the user to accept to void the warranty the smartphone manufacturer will let you perform this unlock. The warranty void is simply the smartphone manufacturers trying to guard themselves from having to repair bricked phones.

The use of ClockWorkMod’s *ROM Manager* is extremely simple. It is now in Android Market - Google Play and when you have root access on your phone it will enable both installation of a new recovery mode (for some phones) with the ability to flash new images, and it will allow a simple boot into this recovery mode without knowing which buttons to press (and for how long) upon restart. For more information about installing new images check online articles at MoDaCo[20] and XDA-developers[28] for specific phone types.

3 Experiment

A smart phone will try to communicate with Internet related services using either a wireless access point or a 2G (GPRS/EDGE) or 3G (HSDPA) connection determined by the users preferences. In our experiments the data-connection, both 2G and 3G, for the mobile clients were switched off before the PIN-code (for the SIM card) was entered. This was not possible for some of the phones, demanding that we entered the PIN before we got to any menu. Because of this we had the operator turn off the support for data traffic completely at their side. This blocking of 2G/3G data traffic was confirmed to be working. This way we made sure that no IP-traffic was using GPRS/EDGE or any other communication channel via the mobile network. The only way of communicating was through a wireless LAN, voice or SMS[5].

The experiment was conducted using a man-in-the-middle (MitM) attack, as shown in Figure 2.1, where the client is connecting to our wireless access point as the default gateway towards the Internet. This computer with a built in wireless card will then have access to all network connections made by the smart phone, both encrypted and unencrypted.

The information gathering was straight forward. For all plain text connections we could, at the MitM computer, log and analyze the information sent and received by the client. This worked well for many protocols and is also demonstrated and analyzed in related papers [2].

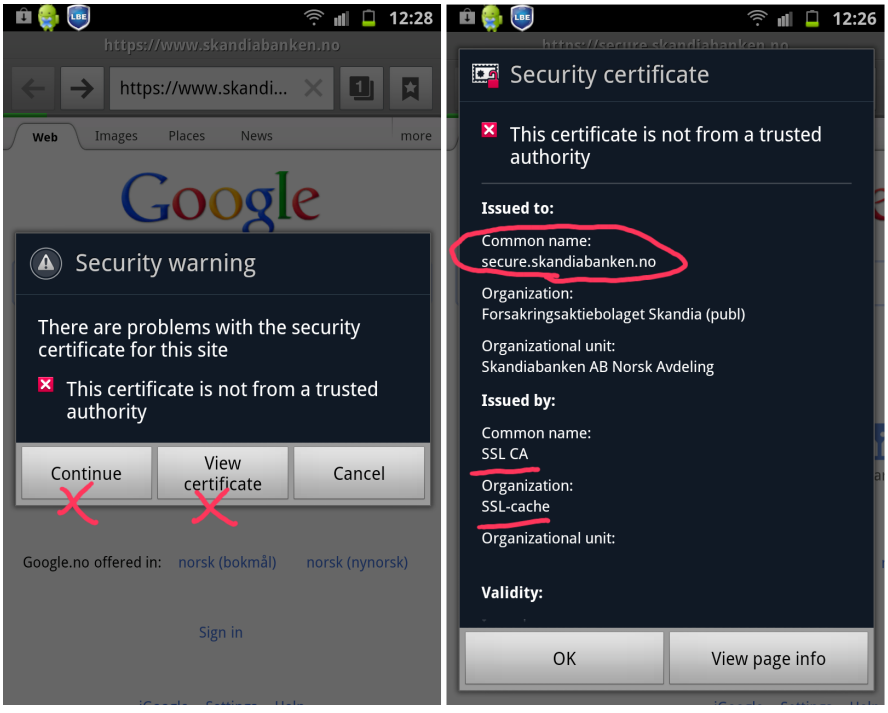

Figure 3.1 Warning from browser about certificate

When trying to intercept SSL traffic we get warnings from the local software, in the same

[5]It may be argued that both voice and SMSs can be sent and received without the users knowledge, but this is out of scope for this article and not likely for uncompromised phones.

way as browsing the network. This can look like in Figure 3.1, where we get the possibility to examine the certificate sent from the server. The fake certificate is shown on the right side of Figure 3.1. We have produced the correct "Common name", but the certificate issuer is not trusted, therefore we have the warning.

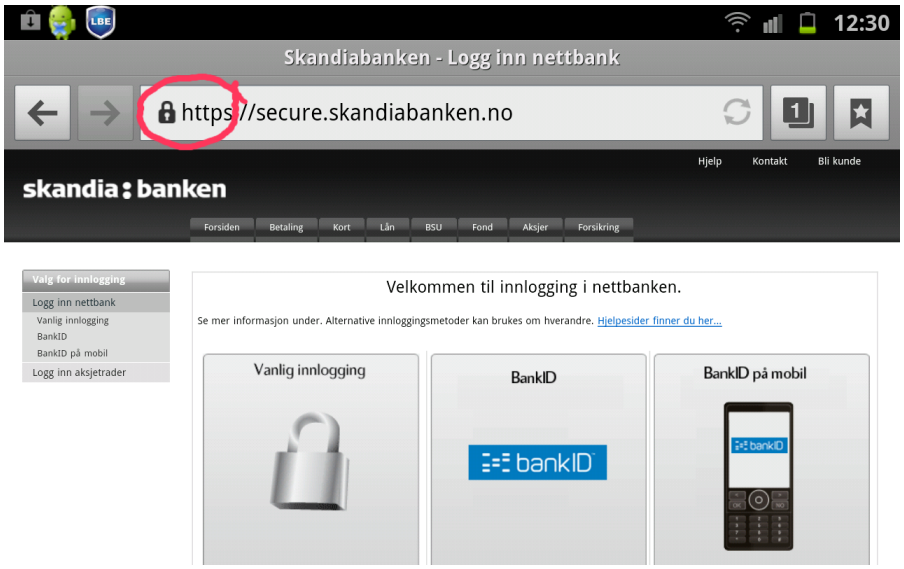

Figure 3.2 Login without warning when adding trusted certificate issuer locally

But if we are able to get the Android smartphone to trust our certificate issuer (called "SSL CA/SSL-cache") then we end up logging in to the actual site without any warnings with a (seemingly) secure connection. This is shown in Figure 3.2.

Trying to intercept and decipher SSL traffic was a bit more complicated, but the experimental setup from Figure 3.3 allowed us to get access to the encrypted communication. But we needed to decrypt the data, and this was solved by making the client accept us as a trusted certificate signer. On Android this is done by adding our certificate to the file "/system/etc/security/cacerts.bks" as shown in Section 2.8. By having the man-in-the-middle computer faking to be the end server and issuing (now trusted and self-signed) certificates, we had full access to the content of the information exchange over SSL. This had previously been done with Squid and *sslsniff*[19], but these were not functioning well in all our tests. Especially *sslsniff* had significant problems with binary POSTs in the HTTPS communication channel. We had to debug and change the source code of the *sslsniff* software to avoid these crashes, get hold of the actual content and complete a patch allowing us to remove some of these bugs found. The *sslsniff* software has later been updated to a newer versions, but this was after most of our tests were completed. There are still bugs present in *sslsniff*. These are described in a bachelor thesis[1] from Gjøvik University College as a followup of this work .

All phones we used in the experiments had a clean Android installation before the start of each log experiment. Some were factory clean and never turned on in the initial tests, and some were

erased and then flashed with a new ROM and then tested.

3.1 Experiment setup

The experiment setup as shown in Figure 3.3 requires a bit of configuration. On the client we have to install our self-signed trusted certificate as explained in Section 2.8. On the man-in-the-middle computer we need to setup both the access point, the routing of traffic through the computer, the certificate server program and the logging of traffic.

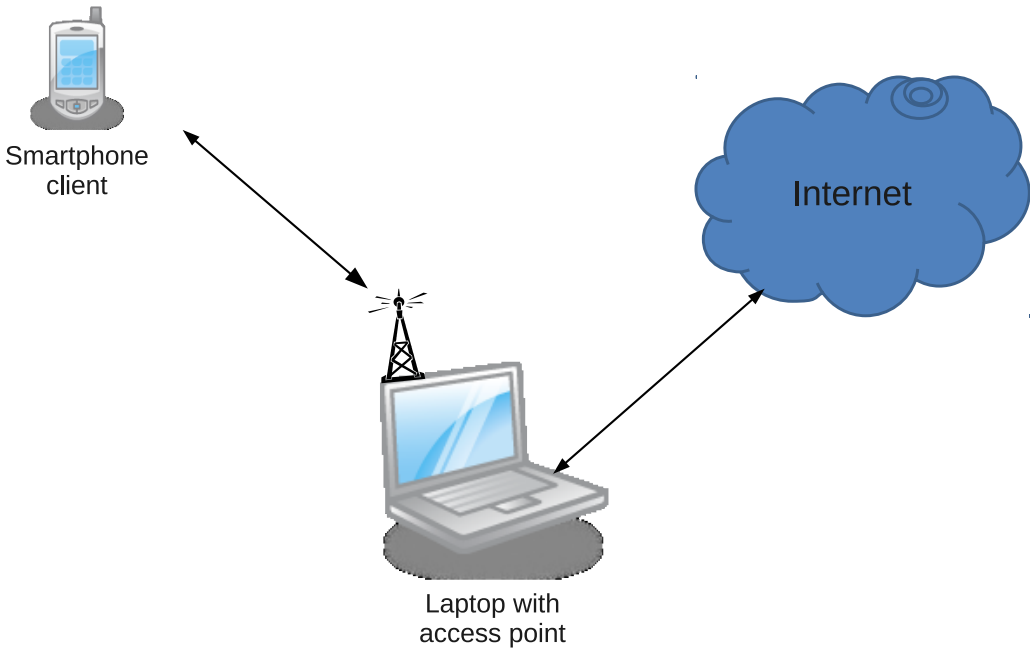

Figure 3.3 Experiment setup

There were two methods used for setting up an access point. In the beginning of the experiment we used a normal wireless access point connected to an extra ethernet port on the man-in-the-middle (MiTM) computer. This required almost no extra setup on the MiTM-computer except for an IP-address, routing and dhcp as shown below. In order to not have the access point "pollute"[6] the data traffic we were logging, we used a network card as a built-in access point on the MiTM-computer.

For setting up the built in access point we used a plain RLT8188-based network card. This requires a Linux version later than 3.0 for the drivers to be built in, but for older kernels we can download the drivers from Realtek and compile them ourselves.

[6]We found no indication for this to be taking place, but wanted to have more control over the traffic we were logging.

After locating the wireless card, wlan2, we need to use it as an access point. For this we use *hostapd* which is an open source software and easy to configure once you have a supported access point.

First we have to configure the interface with an ip address using:

```
# /sbin/ifconfig wlan2 172.16.16.1 netmask 255.255.255.0\
    broadcast 172.16.16.255 up
```

and then setup DHCP by using dnsmasq:

```
# /usr/sbin/dnsmasq -u dnsmasq -x /var/run/dnsmasq.pid\
    --listen-address=172.16.16.1 --bind-interfaces\
    --dhcp-range=interface:wlan2,172.16.16.16,172.16.16.31,\
255.255.255.0,172.16.16.255,7200\
    --dhcp-option=option:router,0.0.0.0
```

If we want the smartphone to work with all traffic we must also setup a default routing for all traffic through the MiTM-computer:

```
# /sbin/iptables -P FORWARD ACCEPT
# /sbin/iptables -F FORWARD
# /sbin/iptables -t nat -F PREROUTING
# /sbin/iptables -t nat -A PREROUTING -i wlan2 -p tcp\
   --destination-port 443 -j REDIRECT --to-ports 3129
```

The last line forces all HTTPS traffic to local port 3129 which is where *sslsniff* is listening.

Then it is time to start the access point. We have below an example *hostapd* configuration file, *hostapd.conf*.

```
interface=wlan2
driver=nl80211
ssid=LABnet
hw_mode=g
channel=5
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=2
wpa=2
wpa_passphrase=topsecret
wpa_key_mgmt=WPA-PSK
```

```
wpa_pairwise=TKIP
rsn_pairwise=CCMP
logger_stdout=-1
logger_stdout_level=1
```

This would set up an access point on interface wlan2, gave the wireless network the name "LABnet", using channel 5 on a 802.11g network, do not broadcast the name of the network, and require WPA encryption using the pre-shared password "topsecret".

The startup command for hostapd would then become:

```
/usr/sbin/hostapd -d -B ./hostapd.conf
```

where "-d" is to receive some extra debug messages and "-B" is for running the process as a daemon in the background.

The result from running *hostapd* is that we have a new interface defined, mon.wlan2, which can be used for monitoring the traffic (not using the HTTPS port) using tcpdump:

```
# /usr/sbin/tcpdump -i mon.wlan2 -n -s 0 -w logfile.pcap &
```

When this is done we are ready to setup the *sslsniff* program by using the following command:

```
# /usr/bin/sslsniff -a -s 3129 -w logfile.sslsniff -c key+cert.pem
```

where "-a" is for running in authority mode and acting as a certificate authority (CA), "-s 3129" tells the program to listen to and intercept traffic to this port, "-w logfile.sslsniff" for specifying where to write thee decrypted data, and "-c cakey+cacert.pem"for using the CA key and self-signed certificate in this file.

For intercepting the HTTPS traffic to port 443 we have to redirect the traffic to the port *sslsniff* is listening to on the MiTM-computer using:

```
# /sbin/iptables -t nat -A PREROUTING -p tcp\
   --destination-port 8443 -j REDIRECT --to-ports 4433
```

Then we should be ready to have the clients connect and have access to all traffic logs in "logfile.pcap" and the decrypted traffic in "logfile.sslsniff".

One problem with having sslsniff make the decryption is that it does not write in pcap-format, but decrypts the content and writes everything to a text file, including binary data. This makes the logfile from *sslsniff* quite unreadable at times, and even harder to try to parse if there is

binary data. This has been the case for newer smartphones with newer versions of Android, which is the main reason for not testing new phones.

Another problem with *sslsniff* is that it has a bug that shuffles the order of the packets, has some buffer length bugs, and therefore requires some work before being used on newer smartphones. A proposed patch from follow-up work performed in a Bachelor thesis at HiG[1] will make *sslsniff* write only the TLS/SSL keys to a file and leave the logging of traffic data to tcpdump/wireshark[7]. Wireshark[8] is then able to read these keys and can therefore be used to decrypt and parse the content of the encrypted traffic.

3.2 Google "checkin"

All instances of Android tested makes a connection to Google upon first contact with Internet. This is a HTTPS POST to the server *android.clients.google.com* with the path of **/checkin** sending and receiving huge amounts of data. We have extracted and analyzed some of the most interesting data in this section. But we provide the complete HTTP POST for the HTC Desire smartphone running a Cyanogenmod[5] version of the Android 2.1 operating system in Appendix A.

This is the first thing done once an Android phone gets a network connection. A later paper by Bae et al. [1] reports the interval of the checkin updates to be every nine minutes, but we did not observe that frequent updates during our tests.

We start by looking at the parameter of the provider of SIM-cards. This element is sent as one of the first parameters in Google's "checkin".

Direct smartphone and network identifiers

```
"sim_operator":"24202"
```

The mobile subscriptions we used in these tests were from multiple operators - Telenor, NetCom and Ventelo. There were no noticeable difference between the information sent through either of these operators, except for the reporting back of which national carrier that is in use. For Telenor and Ventelo this was "24201" and for NetCom it was "24202" meaning "242" for Norway and "02" is for using the NetCom network, "24201" and "24203" would have been Telenor and Network Norway's networks.

For operational puposes there is really no need for Google to have this information connected to the user's identity. But as with all other information, this is useful for statistics and commercial reasons.

```
"build":
```

[7] Tcpdump and Wireshark are the two most used tools for dumping traffic data on Linux and Windows systems.
[8] http://www.wireshark.org/

```
   {"timestamp":1269508994,
    "product":"bravo",
    "id":"htc_wwe/htc_bravo/bravo/bravo:2.1-update1/ERE27/155070:\
user/release-keys",
    "revision":"129",
    "radio":"32.42.00.32U_5.09.00.08",
    "carrier":"htc_wwe",
    "bootloader":"0.75.0000",
    "serialno":"HT03NPL04777"
   },
```

Here we find more complete information about the phone ID. First when the kernel was built and version information, product name “bravo” is the name of the HTC Desire, the radio software version, bootloader version, and the ultimate connection to the phones identity - the serial number of the phone. This phone will by this parameter always be tracked at Google for whatever that may be sent to Google during “checkin” and other connections to Google. The implications of this is addressed in the conclusions chapter.

```
  "roaming":"notmobile-notroaming",
```

Parameters indicating that the user has turned off mobile network usage (GPRS/EDGE/HSDPA/...) and that it does not allow for network connections while roaming[9].

The “checkin” also submitted logs from the operating system during these recurring calls to checkin. If the phone had been in use and was updated with a new version of the operating system, (a lot of) the update logs were automatically sent through “checkin”. We will look at some of this information next.

Transfer of startup log

```
  "event":
   [{"value":"","tag":"SYSTEM_BOOT","time_msec":1300198611137},
    {"value":"Linux version 2.6.29-82821fb5 (htc-kernel@and18-2) \
(gcc version 4.4.0 (GCC) ) #1 PREEMPT Mon Apr 26 22:40:49 CST 2010
...
[   12.569366] Bluetooth: HCILL protocol initialized
[   12.570617] mmc0: Qualcomm MSM SDCC at 0x00000000a0300000 irq \
24,0 dma 8
[   12.570739] mmc0: 4 bit data mode enabled
...
```

[9] Roaming: when using another mobile operator’s network, e.g. when traveling abroad.

```
[   12.646820] adb_open
[   12.665130] crc16: version magic '2.6.29-97da29ed preempt \
mod_unload ARMv7 ' should be '2.6.34.4-cyanogenmod preempt \
mod_unload ARMv7 '
...
[   12.848876] mmc1: host does not support reading read-only \
switch. assuming write-enable.
[   12.849121] mmc1: new high speed SDHC card at address e980
[   12.849761] mmcblk0: mmc1:e980 SU04G 3.69 GiB
[   12.850097]  mmcblk0:
[   12.852294] block 89 is bad
[   12.855865]  p1 p2 p3
[   12.873138] block 135 is bad
[   12.880676] block 162 is bad
[   12.915954] block 289 is bad
[   12.918823] block 299 is bad
...
[   13.102691] batt:  55%, 3815 mV, -460 mA (-347 avg), 32.5 C, \
755 mAh
...
[   30.816284] Restarting system.
[   30.816497]
[   30.816619] Restarting Linux version 2.6.34.4-cyanogenmod \
(android@giulio-laptop) (gcc version 4.4.0 (GCC) ) #49 PREEMPT\
 Wed Aug 18 01:33:56 CEST 2010
[   30.816650]

No errors detected
",
```

Here we can see that the phone posts an extensive dump from the Android kernel boot log. Most of these parameters are just informative about the system, but much is of the systems configuration and happenings in the last period before the latest restart. Here there are many identifying parameters and there is really no need to send this to Google. Bad blocks on memory cards are most likely unique for most memory cards, and there are other identificators like kernel versions, kernel build identity, smartphone hardware, etc.

Transfer of kernel messages

```
  "tag":"SYSTEM_LAST_KMSG","time_msec":1300198611341},
  {"value":"I:Set boot command \"\"
```

```
-- Wiping data...
Formatting DATA:...
mtd: not erasing bad block at 0x00060000
mtd: not erasing bad block at 0x00900000
...
mtd: not erasing bad block at 0x077c0000
Formatting CACHE:...
mtd: not erasing bad block at 0x00340000
...
mtd: not erasing bad block at 0x02540000
Formatting SDEXT:...
Formatting SDCARD:/.android_secure...
Data wipe complete.
I:Set boot command \"\"
",
```

The messages sent/logged from the Android operating system kernel just before the last reboot are sent to Google. As shown above, this includes the results from the “clean” wipe of the smartphone just prior to the initial startup for logging these requests. All this information is submitted to “checkin” as a part of all last kernel messages. In this example the bad blocks of the wipe will identify the phone and/or memory card as these parameters most likely are unique for the memory areas in every phone.

System recovery log

```
  "tag":"SYSTEM_RECOVERY_LOG","time_msec":1300198611421},
```

And the system recovery log will be submitted, but we have no example values in the results from our “reinstalled and unused” smartphones as we did not perform any system recoveries.

Full disclosure of phone identity

```
 "digest":"",
 "logging_id":-7439885170310449089,
 "mac_addr":"002376D5FF84",
 "imei":"357841030259226",
 "locale":"en_NO"
}
```

Then at the end of the HTTP POST there is set aside space for a digest, probably a future integrity check of the message, an ID, and the complete giveaway of the phones identity. The

MAC-address of the WiFi interface, and the IMEI of the cell phone. These are unique and will 100% connect every "checkin" from the phone to the other collected data from the phone.

3.2.1 Reply from server - setting values in the smartphone

Based on this HTTP POST information, the reply to the "checkin" message contains lots of parameter settings as shown below. We have filtered out values with minor interest like software configuration details, but kept some of the lines in order to give the reader an idea of what these values look like, are and how they are used. The complete "checkin" reply from the same HTC-phone used above is given in A.2.

```
1300198703 DEBUG sslsniff : Read from Server \
(15:18:23.487762-*.google.com) :

{"setting":
 [{"name":"android_id", "value":"230620085XXXXXXXXX"},
  {"name":"c2dm_auth_token","value":"1"},
  {"name":"checkin_interval","value":"164613"},
```

The value "android_id" is a phone ID most likely for internal use and reference. This ID is probably linked to the IMEI and/or MAC-address and then the link to the phone's configuration settings for the different Android services. Most interesting here may be the "checkin_interval" of 164613 which most likely is in seconds meaning approximately every two days[10].

```
  {"name":"gmail_host","value":"gmail.com"},
...
  {"name":"gtalk_active_heartbeat_ping_interval_ms",\
"value":"1680000"},
  {"name":"gtalk_auth_sasl","value":"false"},
  {"name":"gtalk_binary_protocol","value":"true"},
  {"name":"gtalk_hostname","value":"mtalk.google.com"},
  {"name":"gtalk_idle_timeout_ms","value":"300000"},
  {"name":"gtalk_max_server_heartbeat_time","value":"900000"},
  {"name":"gtalk_nosync_heartbeat_ping_interval_ms",\
"value":"1680000"},
  {"name":"gtalk_picasa_album_url",\
"value":"http://picasaweb.google.com/data/feed/api/user/%s/album/%s"},
...
  {"name":"gtalk_secure_port","value":"5228"},
  {"name":"gtalk_ssl","value":"true"},
```

[10] Later work[1] has found the "checkin"-interval to be every nine minutes, but this has not been tested by us

```
  {"name":"url:contacts_sync_http_proxy",
"value":"http://www.google.com/m8/feeds rewrite\
 http://android.clients.google.com/proxy/contacts"},
  {"name":"url:contacts_sync_https_proxy",
"value":"https://www.google.com/m8/feeds rewrite\
 https://android.clients.google.com/proxy/contacts"},
  {"name":"url:gmail_sync_http_proxy",
"value":"http://mail.google.com rewrite\
 http://android.clients.google.com/proxy/gmail"},
  {"name":"url:gmail_sync_https_proxy",
"value":"https://mail.google.com rewrite\
 https://android.clients.google.com/proxy/gmail"},
...
  {"name":"url:google_location_server",
"value":"http://www.google.com/loc/m/api rewrite\
 https://www.google.com/loc/m/api"},
```


GMail and GTalk configuration settings are received here. Most interesting it could be if we change these values sent back to the smartphone. Most interesting would be to change hostnames, like in “gmail_host”, “gtalk_hostname” and “gtalk_ssl”.’

There is a long list of configuration settings for rewriting URL shortnames. Changing values and having syncronization data sent to attacker controlled servers *could* compromize security and retreive the data. The effect of changing these values will be looked at in future work.


```
  {"name":"vending_require_sim_for_purchase","value":"true"},


  {"name":"youtube_use_proxy","value":"true"}
 ],
 "stats_ok":true,
 "digest":"uSQsVWb3n5DThSfkPJCFtg==",
 "time_msec":1300198703799,
 "intent":[{"action":"android.server.checkin.FOTA_CANCEL"}]}
```


And the reply from the server ends with message digest for the server message content integrity, a time stamp and a setting for not receiving FOTA-updates (Firmware Over The Air).

3.3 Some phone specific results from running the experiments

Although not all phones were equally easy to test and intercept traffic, we were able to make other findings that were of interest.

3.3.1 Sony-Ericsson X8 phone

The SonyEricsson X8 was running Android version 1.6. Testing this older version of Android revealed that a “factory install” does not remove the installed wireless networks. Meaning that you have to install a clean ROM to make a real erase of the device.

Most phones were easy to trigger the “Flight mode” or in another way turn off wireless network before entering the SIM card’s PIN code, but Sony-Ericsson X8 would not allow us to do this. We were forced to enter the PIN before getting to any menu, so we had to call the operator and have them disable 2G and 3G traffic for our subscription.

Rooting the X8 is straight forward for those devices running versions 1.6 and 2.1 of Android. Many scripts existed already back in early 2011 and these are described in Section 2.7.

Due to problems installing the new ROM and the trusted certificate on this phone we were unable to intercept TLS/SSL traffic.

3.3.2 HTC phones

The mobile devices from HTC that were tested include HTC Desire, HTC Legend, and HTC Desire HD. This is an extract of the most relevant findings from the HTC phones.

Upon startup the HTC Desire with 2.1r1 also set up a connection back to HTC as soon as it locates an Internet connection for a similar “checkin”. But in contrast from the Google “checkin”, this is done over plain text HTTP, and may be manipulated by all with access to the information without manipulating and/or installing certificates on the phone.

```
POST /android/checkin HTTP/1.1
Content-type: org/x-json
Content-Length: 421
Host: andchin.htc.com
Connection: Keep-Alive
User-Agent: Android-Checkin/2.0

{"checkin":
 {"build":
  {"product":"bravo",
   "id":"htc_wwe/htc_bravo/bravo/bravo:2.1-update1/ERE27/155070:\
user/release-keys",
   "revision":"129",
   "firmware_version":"1.15.405.4 CL155070 release-keys",
   "radio":"32.42.00.32U_5.09.00.08",
   "carrier":"htc_wwe",
   "bootloader":"0.75.0000",
```

```
   "build_type":"user",
   "changelist":"155070",
   "serialno":"HT03NPL04777"
  }
 },
 "model_number":"HTC Desire",
 "digest":"",
 "imei":"357841030259226",
 "locale":"en_NO"
}
```

This is quite smaller than the Google “checkin”, but as we see in the listing above, most of the identifying parameters are still present including the versioning information, which is used to recommend updates if they exist. The complete reply is given in Appendix B, but below we show a few of the more significant details of the HTC reply.

3.3.3 HTC system update

If your phone is recommended for an update to a more recent version of Android, or simply some patches to the current running system, HTC will inform about this in the response. First there is a “digest”, which most likely is a fingerprint of the reply. How this works is still beeing researched, but there seems to be no security mechanisms around this.

The update recommendation is with HTC’s “checkin” therefore sent over an unsecured HTTP channel, and anyone with access to the communication will be able to change or modify the response without the client beeing able to detect this. Below we show the response for an update as shown in Appendix B.

```
...
 "digest":"fc121695254dba9224698fa3f1acdfd",
 "intent":[{
  ...,
   "name":"promptVersion",
   "value":"System upgrade1.21.405.2 (27.07 MB)\
 We would recommend using a free Wi-Fi\
 hotspot or an unlimited data plan to apply this update. If not,\
 standard data connection charges may apply. Any questions?\
 Contact us via http://www.htc.com/www/CA_Hotline.aspx"},
  {"name":"promptMessage",
   "Value":"System Upgrade"},
  {"name":"promptSize",
   "value":"27.07 MB"},
```

```
  {"name":"promptFeature",
   "value":"Update:HTC application improvement"},
  ...
}
```

And given the response in plain text in an integrity insecure protocol, which can be manipulated by eveyone, we see that the download link **also** is an HTTP request. Which means that you can give the phone any link, and it will download it. As mentioned before there is a digest in here to protect the information being transferred[11], but this digest may of course also be changed since we here have an unencrypted HTTP connection without integrity at the protocol level.

The signature of the ROM file is the only security given, but this does not mean that someone can trick or force the user to install an earlier version, the same version, or potentially a modified new version in three steps. First, edit message to force the install of an old vulnerable version, then exploit a known vulnerability in this version and last install your own modified version of the operating system without a trusted signature from the phone vendor.

In addition to the above mentioned vulnerabilities the HTC update recommends in the message that the user should use an open wireless access point for the download:

> We would recommend using a free Wi-Fi hotspot or an unlimited data plan to apply this update. If not, standard data connection charges may apply. Any questions? Contact us via `http://www.htc.com/www/CA_Hotline.aspx`

This will enable an local eavesdropper, e.g. on a public wireless network both to change the download link and provide any update the attacker would prefer. The only security mechanism left for the protection of the update is now the digital signature of the update, but this can be defeated as explained above.

In the system and the system updates inside the ROM, the file *system/build.prop* includes the configuration of the HTC checkin command through the setting

```
ro.htc.checkin.url = http://andchin.htc.com/android/checkin
```

which may easily be replaced to make the “HTC checkin” send its information to other locations and receive the attacker’s configuration settings.

In addition the HTC logs contain information about the update link and the update process as shown here:

[11] And if this bf is a *keyed* digest - similar to a message authentication code - it is only a matter of time until the key will be found, as it has to be stored on the device as well.

```
  {"name":"force_update",
   "value":"false"}],
  "action":"android.server.checkin.FOTA_UPDATE",
  "data_uri":"http://fotadl.htc.com/OTA_Bravo_HTC_WWE_\
1.21.405.2-1.15.405.4_release_P4ohim1mtp6zggcyon.zip"}],
 "setting":[
  {"name":"upload_crash",
   "value":"http://63.241.57.68/crashblock"},
  {"name":"checkin_interval",
   "value":"604800"},
```


The “force_update=false” replaced with “true” will likely make the phone install any (signed) update it finds at the given URL. This has not been tested and verified and is only a hypothesis.

The “action” parameter describes the android app being used to perform an update.

The “data_uri” is the address of (i.e. link to) the update image describing where it can be downloaded. The zip file can be extracted and analyzed by anyone, and these types of update files have been a starting point for all “modified unoffical android versions” released by the Android community. The software packages are available and can be copied, replaced or removed before creating new images and new “update zip-files”.

The “upload_crash” is the address to a service just receiving information. All connection attempts to this URL gives no feedback and we have not been able to see any use of this address, which most likely is used if there is a problem with the installation of the new image. (Not tested.)

The “checkin_interval” has most likely the same functionality as in the “Google checkin” - the number of seconds until the next time the smartphone is supposed to check in again.

3.3.4 Samsung phones

Samsung Galaxy Tab

The Samsung Galaxy Tab, first version, used a default install of Android version 2.2. The following data was fetched using a clean factory startup with wifi only, gmail entered, no location allowed and not allowing background sync.

Immediately after getting WLAN connection the device connects back to Samsung with information about the device, network, language, country, and getting some setup information about currency format and other minor configuration options back. First, a DNS lookup of *hub-odc.samsungapps.com* and then an HTTP-POST to this IP 213.71.30.154 (an IP with no reverse DNS answer) and the device gets information containing configuration parameters returned from the web service as shown below.

The way to identify the network and country is interesting as this is done by sending the MCC- and MNC-fields of the mobile device setup, where the country code of Norway is 242, and Telenor's network is MNC 01. The query to *hub-odc.samsungapps.com* follows. The information sent to the server from the smartphone is shown below. The first paragraph is the HTTP POST query headers and the second paragraph contains the values from the smartphone settings. The traffic dump is taken from Wireshark and may contain special characters and unusual line shifts as they are displayed as presented in Wireshark.

```
POST /ods.as HTTP/1.1
Content-Length: 360
Content-Type: text/plain; charset=ISO-8859-1
Host: hub-odc.samsungapps.com
Connection: Keep-Alive

<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>\
<SamsungProtocol version="3.0" lang="EN" deviceModel="GT-P1000"\
 networkType="0" mcc="242" mnc="01" csc="OXX" openApiVersion="8">\
<request id="2300" name="countrySearchEx" numParam="2"\
 transactionId="0"><param name="latestCountryCode">242</param>\
<param name="whoAmI">una</param></request></SamsungProtocol>
```

As we see in the parameters sent, the MCC and MNC are here, the device type description "GT-P1000", installation language "EN" and a very strange parameter "latestCountryCode" which has not been tested, but we speculate that it will change only upon swithcing the mobile operator to operators in other countries.

```
HTTP/1.1 200 OK
Date: Wed, 01 Dec 2010 19:00:54 GMT
Server: Apache
Cache-Control: no-cache="Set-Cookie"
Set-Cookie: JSESSIONID=LjyJM2bGvynp1GQwgPdK4pnbVqQnn1wGGGX29vs\
GhWLg89vVGKp1!405452529!-264178473; path=/
Set-Cookie: WMONID=RnTLfcoA1py; expires=Thursday, 01-Dec-2011\
 19:00:54 GMT; path=/
Content-Language: en-US
X-Powered-By: Servlet/2.5 JSP/2.1
Vary: Accept-Encoding,User-Agent
Keep-Alive: timeout=30, max=10000
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/xml; charset=UTF-8
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SamsungProtocol deviceModel="GT-P1000" networkType="0" lang="EN"
 version="3.0">
<response transactionId="0" totalCount="1" endNum="1" startNum="1"
 returnCode="0" name="countrySearchEx" id="2300">
<errorInfo>
<errorString errorCode="0"/>
</errorInfo>
<list numValue="8">
<value name="countryURL">http://mkt-odc.samsungapps.com/ods.as\
</value>
<value name="countryCode">PPC</value>
<value name="freeStoreClsf">0</value>
<value name="description"></value>
<value name="currencyUnitPrecedes">1</value>
<value name="currencyUnitHasPenny">1</value>
<value name="offset">0</value>
<value name="freetabClsf">0</value>
</list>
</response>
</SamsungProtocol>
0
```

Galaxy SII

With newer phones, like the Samsung Galaxy SII, the *sslsniff* program started to have problems with forwarding the packets in a correct manner as described in the experiment setup. Therefore newer phones will be tested in later work after these problems have been fixed.

3.3.5 Google Nexus One

The Google Nexus One phone is designed to be a developers phone and easy to install new versions of Android onto, even self-modified and pre-rooted versions. The experiment shown below was using an Android 2.2.1 kernel put together by MoDaCo[20] (more specific Android 2.2.1, kernel r24, pre-rooted).

The first log is after the completely clean ROM install, without any wifi, 3G, no location give-away accepted, no background sync, no google mail account, e.g. all information restricted as good as Android allows you to do. This log reveals that the first connections after WiFi is setup goes to *www.amazon.com* to configure the music client, and consists of device information and software information as shown in Appendix C.1.

Here we can see that it sends all information about the device versions, software versions, and receives a configuration setup of the media player. The returned file configures media player to use https, turn on security features, use Amazon for everything - from search, play and to purchases of music through their servers.

It may be a coincident that the music players lookup happens first as it did take some time to setup the wireless network. Anyway, after this Android market (not confirmed yet, but market is best guess for now) seems to lookup *android.clients.google.com*, being returned to *android.l.google.com* and 74.125.77.100 to continue and set up an HTTPS connection there sending 10kb of data to Google and getting 12kb back.

The request contains a download of multiple pages with information marked as news, reporting that the agent is GMM/3.0 - Google Maps for Mobile 3.0

After 5 minutes there is another automatic connection to xtra2.gpsonextra.net with request for what seems to be configuration settings for the GPS software on the smartphone. Many Android phones are getting these update, so the accumulated requests here may be an issue.

```
GET /xtra.bin HTTP/1.1
Accept: */*, application/vnd.wap.mms-message,\
 application/vnd.wap.sic
x-wap-profile: http://www.openmobilealliance.org/tech\
/profiles/UAPROF/ccppschema-20021212#
Host: xtra2.gpsonextra.net
Connection: Keep-Alive
User-Agent: Android

HTTP/1.1 200 OK
Date: Tue, 07 Dec 2010 09:30:45 GMT
Server: Apache/2.0.52 (Red Hat)
Last-Modified: Tue, 07 Dec 2010 09:30:02 GMT
ETag: "9b57-a6295a80"
Accept-Ranges: bytes
Content-Length: 39767
Connection: close
Content-Type: application/octet-stream

.......k(.....W.M.8...M.;......%.....

(40kb of encoded or encrypted data)
```

Following this we print out the raw data of a connection to *www.google.com*, which seems to

be coming from the YouTube application. This application has been started automatically by Android and not by the user and is reporting to Google the phone type, kernel versions and which version of the YouTube app and Android OS the device is running, as shown below.

```
000  50 4f 53 54 20 2f 6d 2f  61 70 70 72 65 71 2f 6d POST /m/ appreq/m
010  6f 62 69 6c 65 76 69 64  65 6f 20 48 54 54 50 2f obilevid eo HTTP/
020  31 2e 31 0d 0a 43 6f 6e  74 65 6e 74 2d 54 79 70 1.1..Con tent-Typ
030  65 3a 20 61 70 70 6c 69  63 61 74 69 6f 6e 2f 62 e: appli cation/b
040  69 6e 61 72 79 0d 0a 43  6f 6e 74 65 6e 74 2d 4c inary..C ontent-L
050  65 6e 67 74 68 3a 20 31  36 39 0d 0a 48 6f 73 74 ength: 1 69..Host
060  3a 20 77 77 77 2e 67 6f  6f 67 6c 65 2e 63 6f 6d : www.go ogle.com
070  0d 0a 43 6f 6e 6e 65 63  74 69 6f 6e 3a 20 4b 65 ..Connec tion: Ke
080  65 70 2d 41 6c 69 76 65  0d 0a 55 73 65 72 2d 41 ep-Alive ..User-A
090  67 65 6e 74 3a 20 41 6e  64 72 6f 69 64 2d 59 6f gent: An droid-Yo
0A0  75 54 75 62 65 2f 32 20  28 70 61 73 73 69 6f 6e uTube/2  (passion
0B0  20 46 52 47 38 33 44 29  3b 20 67 7a 69 70 0d 0a  FRG83D) ; gzip..
0C0  0d 0a 00 02 00 00 7b 79  6f 75 74 75 62 65 2c 32 ......{y outube,2
0D0  2e 30 2e 32 36 2c 5b 4e  65 78 75 73 20 4f 6e 65 .0.26,[N exus One
0E0  5d 5b 67 6f 6f 67 6c 65  2f 70 61 73 73 69 6f 6e ][google /passion
0F0  2f 70 61 73 73 69 6f 6e  2f 6d 61 68 69 6d 61 68 /passion /mahimah
100  69 3a 32 2e 32 2e 31 2f  46 52 47 38 33 44 2f 37 i:2.2.1/ FRG83D/7
110  35 36 30 33 3a 75 73 65  72 2f 72 65 6c 65 61 73 5603:use r/releas
120  65 2d 6b 65 79 73 5d 2c  6d 76 61 70 70 2d 61 6e e-keys], mvapp-an
130  64 72 6f 69 64 2d 67 6f  6f 67 6c 65 2c 65 6e 5f droid-go ogle,en_
140  55 53 00 00 00 00 00 00  00 00 00 01 67 00 00 00 US...... ....g...
150  17 00 01 00 00 00 00 08  67 3a 6c 6f 67 3a 65 76 ........ g:log:ev
160  00 01 00 00 00 05 08 04  18 e9 07               ........ ...
```

Ten microseconds after the SYN-connection to port 80 at *www.google.com* given above (0.00001s immediately following the SYN and before there is any reply), there is a SYN-packet to port 443 also on *www.google.com* setting up an HTTPS connection to a server with a certificate from Thawte. This connection seems to be downloading news, but is related to the startup of Google Maps. These results were gathered before we had sslsniff working and the phone has since been used for testing in a multiple settings.

And after 580s there is a startup of Google Maps to *mobilemaps.clients.google.com* which is an alias for *clients.l.google.com* with the IP-address 74.125.77.100. One of the communication blocks is given below.

```
POST /glm/mmap HTTP/1.1
Content-Type: application/binary
```

```
Content-Length: 216
Host: mobilemaps.clients.google.com
Connection: Keep-Alive
User-Agent: GoogleMobile/1.0 (passion FRG83D); gzip

............en_US..android:HTC-passion-Nexus One..4.7.0.07..\
gmm-android-google>...N... .*.GMM...SYSTEM...-17dd8aefd4eb7e2a\
.........8...GMM_3119427614516133278....K....
...
...
.....,.2........4.7.0..H.......O...O...O

HTTP/1.1 200 OK
Content-Type: application/binary
Content-Length: 68
Date: Tue, 07 Dec 2010 09:38:48 GMT
Expires: Tue, 07 Dec 2010 09:38:48 GMT
Cache-Control: private, max-age=0
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
Server: GSE

..>....
.409.p.....h.K...
....w.^.....".H.

...6.......*.
H.......O
```

The always occurring versioning information including the Android version is also here present. But there are two parameters which we found no explanation to, the strings:

- “17dd8aefd4eb7e2a”
- “GMM_3119427614516133278”

These are suspected to be connected to some address identifier and the Google Mobile Maps program, but we have not been able to confirm or reject this theory. The latter identifier also occurs in later Google Maps connections.

After this (at 587 seconds) there is another connection to *mobilemaps.clients.google.com*. As we explained earlier we specifically told the Android OS not to give away location using neither

GPS nor WLAN, but this is not respected by the installed apps. If the application tricks the user to accept “every” security exception before installing - like Google Maps does - the client is able to get and submit information anyway. The following connection to fetch maps and location gives away more information than most users are aware of:

```
000  50 4f 53 54 20 2f 67 6c  6d 2f 6d 6d 61 70 20 48 POST /gl m/mmap H
010  54 54 50 2f 31 2e 31 0d  0a 43 6f 6e 74 65 6e 74 TTP/1.1. .Content
020  2d 54 79 70 65 3a 20 61  70 70 6c 69 63 61 74 69 -Type: a pplicati
030  6f 6e 2f 62 69 6e 61 72  79 0d 0a 43 6f 6e 74 65 on/binar y..Conte
040  6e 74 2d 4c 65 6e 67 74  68 3a 20 34 30 36 0d 0a nt-Lengt h: 406..
050  48 6f 73 74 3a 20 6d 6f  62 69 6c 65 6d 61 70 73 Host: mo bilemaps
060  2e 63 6c 69 65 6e 74 73  2e 67 6f 6f 67 6c 65 2e .clients .google.
070  63 6f 6d 0d 0a 43 6f 6e  6e 65 63 74 69 6f 6e 3a com..Con nection:
080  20 4b 65 65 70 2d 41 6c  69 76 65 0d 0a 55 73 65  Keep-Al ive..Use
090  72 2d 41 67 65 6e 74 3a  20 47 6f 6f 67 6c 65 4d r-Agent:  GoogleM
0A0  6f 62 69 6c 65 2f 31 2e  30 20 28 70 61 73 73 69 obile/1. 0 (passi
0B0  6f 6e 20 46 52 47 38 33  44 29 3b 20 67 7a 69 70 on FRG83 D); gzip
0C0  0d 0a 0d 0a                                     ....
0C4  00 17 70 09 7f 9c 1e 0c  68 d4 00 02 65 6e 00 1d ..p..... h...en..
0D4  61 6e 64 72 6f 69 64 3a  48 54 43 2d 70 61 73 73 android: HTC-pass
0E4  69 6f 6e 2d 4e 65 78 75  73 20 4f 6e 65 00 08 34 ion-Nexu s One..4
0F4  2e 37 2e 30 2e 30 37 00  12 67 6d 6d 2d 61 6e 64 .7.0.07. .gmm-and
104  72 6f 69 64 2d 67 6f 6f  67 6c 65 3e 00 00 00 58 roid-goo gle>...X
114  0a 03 34 30 39 18 f2 01  20 01 2a 03 47 4d 4d 92 ..409...  .*.GMM.
124  01 06 53 59 53 54 45 4d  9a 01 11 2d 31 37 64 64 ..SYSTEM ...-17dd
134  38 61 65 66 64 34 65 62  37 65 32 61 b0 01 03 b8 8aefd4eb 7e2a....
144  01 cd ac 4f c8 01 01 da  01 01 38 e2 01 17 47 4d ...O.... ..8...GM
154  4d 5f 33 31 31 39 34 32  37 36 31 34 35 31 36 31 M_311942 76145161
164  33 33 32 37 38 e8 01 00  10 00 00 00 00 29 00 00 33278... .....)..
174  00 a5 08 01 12 a0 01 0a  2e 32 1a 08 ff ff ff ff ........ .2......
184  ff ff ff ff ff 01 12 08  56 45 4e 54 45 4c 4f 20 ........ VENTELO
194  20 01 28 f2 01 3a 10 12  0c 32 30 2e 31 39 2e 31  .(..:.. .20.19.1
1A4  38 2e 31 37 32 18 02 12  04 18 07 30 05 22 68 0a 8.172... ...0."h.
1B4  22 0a 19 08 dd a1 01 10  f2 e7 be 06 18 01 20 f2 ".....  ...... .
1C4  01 28 ff ff ff ff ff ff  ff ff ff 01 10 bd a8 ca .(...... ........
1D4  81 cc 25 12 32 08 d4 a8  ca 81 cc 25 12 29 0a 11 ..%.2... ...%.)..
1E4  30 30 3a 39 30 3a 34 63  3a 37 65 3a 30 30 3a 36 00:90:4c :7e:00:6
1F4  34 12 09 41 6e 6f 6e 79  6d 6f 75 73 20 dd ff ff 4..Anony mous ...
204  ff ff ff ff ff ff 01 1a  0e 0a 0a 0d aa 04 8c 15 ........ ........
214  15 e4 91 c8 c6 40 01 49  00 00 00 15 0a 04 08 01 .....@.I ........
224  10 00 0a 04 08 02 10 00  0a 04 08 03 10 00 10 e3 ........ ........
```

```
234  24 39 00 00 00 20 22 1e  0a 10 08 91 b4 9e 11 10 $9... ". ........
244  ca 83 9d d2 ff ff ff ff  ff 01 10 94 a8 ff 16 18 ........ ........
254  ec f5 8e 14 20 05                                 .... .
```

Google Maps is here giving away the MAC-address of the access-point (00:90:4c:7e:00:64) we are connecting to together with the WLAN's SSID (Lab network was called "Anonymous"), which is a direct attempt to try to identify our location. What should be unnecessary in every situation is the internal IP address (172.18.19.20) in the encrypted wireless network and the GSM operator of the SIM card. The IP address might be quite handy for malicious attacks on the local wireless network giving more information to an attacker. Even if the encryption key is not sent over the plain text communication channel, the included information may assist others in an attack on WEP and WPA networks using Google maps only once. And keep in mind that this was plain text HTTP communication.

4 Conclusion

We have tried to give an overview of what information a smartphone is leaking through normal usage. The experiment intercepted all traffic, both plain text and encrypted traffic, and was able to decrypt the HTTPS traffic and analyze this for early versions of Android. We did this after setting up that no identifiers were to be used for giving away location information, e.g. from base stations, GPS- or WLAN-usage.

We found that many identifying elements of the smartphone were being transmitted back to Google and to the phone and software manufacturers. Most critical parameters include IMEI, WLAN-name, MAC-addresses, phone network operator and country, phone name and version, serial numbers, operating system version, kernel version and debugging information with system identificators.

Maybe the most critical result was a recommendation to install a phone operating system update from an unsecure site, on an unsecure connection, with the option to force the update to take place. By not having this over a secure channel most local users would be able to force the phone to install a new (or old) version og the operating system or a software package addon. Injection of JavaScript into this unencrypted communication, will enable the attacker to have a much wider range of attack vectors available for attacking known vulnerbilities in new attempts to break in.

Other critical findings were to ignore the setting of private location information and submitting both WLAN SSID, WLAN MAC address, and even WLAN internal IP address range.

During the long experimentation period for this report many of the discussed vulnerabilities have now been improved in newer versions of the Android operating system and in the phone manufacturer's software. HTC no longer sends the "checkin" informatino in clear text and it is

still quite some work to install new trusted certificates. We will be looking for more improvements in future work.

References

[1] Eirik Bae, Kjetil Gardåsen, and David Ueland. Personvern og datalekkasje. *Gjøvik University College - BSc Thesis*, May 2012.

[2] Big Brother Mobile Blog. First privacy test : Android/samsung galaxy tab. `http://bigbrothermobile.com/blog/?p=13`, January 2011. [Online; accessed 29 May 2012].

[3] ChainsDD. Androidsu.com - home of superuser. `http://androidsu.com/superuser/`, 2010. [Online; accessed 29 May 2012].

[4] CLShortFuse. shortfuse.org - official home of superoneclick. `http://shortfuse.org/`, 2011. [Online; accessed 29 May 2012].

[5] CyanogenMod Community. Cyanogenmod aftermarket firmware distribution. `http://www.cyanogenmod.com/`. [Online; accessed 29 May 2012].

[6] The Android Exploid Crew. yummy yummy, gingerbreak! `http://c-skills.blogspot.no/2011/04/yummy-yummy-gingerbreak.html`, April 2011. [Online; accessed 29 May 2012].

[7] Google Android Developers. Android platform versions. `http://developer.android.com/resources/dashboard/platform-versions.html`. [Online; accessed 5 June 2012].

[8] T. Dierks and E. Rescorla. The tls protocol - version 1.2. IETF RFC 5246, August 2008.

[9] Alexander Downer. We must not miss out on china's great innovations. `http://www.theaustralian.com.au/national-affairs/opinion/we-must-not-miss-out-on-chinas-great-innovations/story-e6frgd0x-1226328173404`, April 2012. [Online; accessed 29 May 2012].

[10] Koushik Dutta. Rom manager - roms and recovery images. `http://www.clockworkmod.com/rommanager`. [Online; accessed 29 May 2012].

[11] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – http/1.1. IETF RFC 2616, June 1999.

[12] The Squid Software Foundation. Squid: Optimising web delivery. `http://www.squid-cache.org/`. [Online; accessed 29 May 2012].

[13] Roee Hay and Yair Amit. Android browser cross-application scripting. `http://blog.watchfire.com/files/advisory-android-browser.pdf`, July 2011. [Online; accessed 29 May 2012].

[14] Bastian Könings, Jens Nickels, and Florian Schaub. Catching authtokens in the wild: The insecurity of google's clientlogin protocol. http://www.uni-ulm.de/en/in/mi/staff/koenings/catching-authtokens.html, May 2011. [Online; accessed 29 May 2012].

[15] Mary Lenninghan. India's equipment ban a blow to chinese vendors. http://www.totaltele.com/view.aspx?ID=455099, April 2010. [Online; accessed 29 May 2012].

[16] John Leyden. Security takes a backseat on android in update shambles. http://www.theregister.co.uk/2011/11/22/android_patching_mess/, November 2011. [Online; accessed 29 May 2012].

[17] Moxie Marlinspike. Internet explorer ssl vulnerability 08/05/02. http://www.thoughtcrime.org/ie-ssl-chain.txt, August 2002. [Online; accessed 29 May 2012].

[18] Moxie Marlinspike. Defeating ocsp with the character '3'. http://www.thoughtcrime.org/papers/ocsp-attack.pdf, July 2009. [Online; accessed 29 May 2012].

[19] Moxie Marlinspike. sslsniff version 0.8. http://www.thoughtcrime.org/software/sslsniff/, July 2011. [Online; accessed 29 May 2012].

[20] Paul O'Brien. Modaco. http://www.modaco.com/. [Online; accessed 29 May 2012].

[21] Paul O'Brien. Introducing.... superboot! http://www.modaco.com/topic/298782-08mar-superboot-erd79-gri40-rooting-the-nexus-one/, December 2009. [Online; accessed 29 May 2012].

[22] National Institute of Standards and Technology. Vulnerability summary for cve-2009-2417. http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2009-2417, August 2009. [Online; accessed 29 May 2012].

[23] National Institute of Standards and Technology. Vulnerability summary for cve-2011-2357. http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2011-2357, August 2011. [Online; accessed 29 May 2012].

[24] Ryan Pearl, Joshua Wise, Eric Smaxwill, Matthew Fogle, Matt Mastracci, Koush Dutta, Michael Sullivan, and Adam Glasgall. Unrevoked - set your phone free. http://unrevoked.com/, January 2011. [Online; accessed 29 May 2012].

[25] The OpenSSL Project. http://www.openssl.org/. [Online; accessed 29 May 2012].

[26] RyanZA. [APP] z4root. http://forum.xda-developers.com/showthread.php?t=833953, November 2010. [Online; accessed 29 May 2012].

[27] Steven J. Vaughan-Nichols. Can you trust chinese computer equipment? http://www.itworld.com/security/95398/can-you-trust-chinese-computer-equipment, February 2010. [Online; accessed 29 May 2012].

[28] XDA-developers. Android & windows phone: Tablets, apps, & roms xda-developers. http://www.xda-developers.com/. [Online; accessed 29 May 2012].

Appendix A Google checkin

This the communication for the Google “checkin” functionality. The data is reformatted to make them more readable, but the changes are only in the layout, not the content.

A.1 checkin POST and submitted data

```
1300198703 DEBUG sslsniff :
Read from Client (15:18:23.255012-*.google.com) :
POST /checkin HTTP/1.1
Content-type: org/x-json
Content-Length: 9794
Host: android.clients.google.com
Connection: Keep-Alive
User-Agent: Android-Checkin/1.0 (bravo ERE27)
```

```
1300198703 DEBUG sslsniff : Read from Client (15:18:23.263793-*.
    google.com) :
{"checkin":
 {"sim_operator":"24202",
  "build":
   {"timestamp":1269508994,
    "product":"bravo",
    "id":"htc_wwe/htc_bravo/bravo/bravo:2.1-update1/ERE27/155070:
        user/release-keys",
    "revision":"129",
    "radio":"32.42.00.32U_5.09.00.08",
    "carrier":"htc_wwe",
    "bootloader":"0.75.0000",
    "serialno":"HT03NPL04777"
   },
  "roaming":"notmobile-notroaming",
  "stat":
   [{"count":3,"sum":0,"tag":"CRASHES_REPORTED"},
    {"count":0,"sum":91.522,"tag":"ELAPSED_REALTIME_SEC"},
    {"count":0,"sum":91.522,"tag":"ELAPSED_UPTIME_SEC"},
    {"count":0,"sum":0,"tag":"NETWORK_RX_MOBILE"},
    {"count":0,"sum":0,"tag":"NETWORK_TX_MOBILE"},
    {"count":1,"sum":0,"tag":"PHONE_GSM_REGISTERED"}
   ],
```

```
  "event":
   [{"value":"","tag":"SYSTEM_BOOT","time_msec":1300198611137},
    {"value":"Linux version 2.6.29-82821fb5 (htc-kernel@and18-2) (
       gcc version 4.4.0 (GCC) ) #1 PREEMPT Mon Apr 26 22:40:49
       CST 2010

...
12.565368] msm_i2c msm_i2c.0: Error during data xfer (-5)
[   12.565612] ds2482 0-0018: i2c write c3 87 failed, -5, retries
   left 2
[   12.566040] msm_i2c msm_i2c.0: error, status 43c8
[   12.566284] msm_i2c msm_i2c.0: Error during data xfer (-5)
[   12.566406] ds2482 0-0018: i2c write c3 87 failed, -5, retries
   left 1
[   12.566955] msm_i2c msm_i2c.0: error, status 43c8
[   12.567077] msm_i2c msm_i2c.0: Error during data xfer (-5)
[   12.567321] ds2482 0-0018: i2c write c3 87 failed, -5, retries
   left 0
[   12.568267] device-mapper: uevent: version 1.0.3
[   12.568786] device-mapper: ioctl: 4.17.0-ioctl (2010-03-05)
   initialised: dm-devel@redhat.com
[   12.569030] Bluetooth: HCI UART driver ver 2.2
[   12.569244] Bluetooth: HCI H4 protocol initialized
[   12.569366] Bluetooth: HCILL protocol initialized
[   12.570617] mmc0: Qualcomm MSM SDCC at 0x00000000a0300000 irq
   24,0 dma 8
[   12.570739] mmc0: 4 bit data mode enabled
[   12.570953] mmc0: MMC clock 144000 -> 50000000 Hz, PCLK
   64000000 Hz
[   12.571075] mmc0: Slot eject status = 1
[   12.571197] mmc0: Power save feature enable = 1
[   12.571441] mmc0: DM non-cached buffer at ffa04000, dma_addr 0
   x3aa7e000
[   12.571563] mmc0: DM cmd busaddr 0x3aa7e000, cmdptr busaddr 0
   x3aa7e300
[   12.572479] mmc1: Qualcomm MSM SDCC at 0x00000000a0400000 irq
   26,0 dma 8
[   12.572692] mmc1: 4 bit data mode enabled
[   12.572814] mmc1: MMC clock 144000 -> 50000000 Hz, PCLK
   64000000 Hz
[   12.572937] mmc1: Slot eject status = 0
```

```
[   12.573181] mmc1: Power save feature enable = 1
[   12.573303] mmc1: DM non-cached buffer at ffa05000, dma_addr 0
   x3aa7f000
[   12.573516] mmc1: DM cmd busaddr 0x3aa7f000, cmdptr busaddr 0
   x3aa7f300
[   12.574401] logger: created 64K log 'log_main'
[   12.574676] logger: created 256K log 'log_events'
[   12.574859] logger: created 64K log 'log_radio'
[   12.575134] logger: created 64K log 'log_system'
[   12.575531] GACT probability NOT on
[   12.575653] Mirror/redirect action on
[   12.575866] u32 classifier
[   12.575988]     Actions configured
[   12.576110] Netfilter messages via NETLINK v0.30.
[   12.576385] nf_conntrack version 0.5.0 (6501 buckets, 26004 max
   )
[   12.576721] CONFIG_NF_CT_ACCT is deprecated and will be removed
    soon. Please use
[   12.576934] nf_conntrack.acct=1 kernel parameter, acct=1
   nf_conntrack module option or
[   12.577056] sysctl net.netfilter.nf_conntrack_acct=1 to enable
   it.
[   12.577362] ctnetlink v0.93: registering with nfnetlink.
[   12.577606] xt_time: kernel timezone is -0000
[   12.577972] ip_tables: (C) 2000-2006 Netfilter Core Team
[   12.578186] arp_tables: (C) 2002 David S. Miller
[   12.578521] TCP cubic registered
[   12.578643] NET: Registered protocol family 17
[   12.578765] NET: Registered protocol family 15
[   12.579071] Bridge firewalling registered
[   12.579223] Bluetooth: L2CAP ver 2.14
[   12.579345] Bluetooth: L2CAP socket layer initialized
[   12.579559] Bluetooth: SCO (Voice Link) ver 0.6
[   12.579681] Bluetooth: SCO socket layer initialized
[   12.580108] Bluetooth: RFCOMM TTY layer initialized
[   12.580230] Bluetooth: RFCOMM socket layer initialized
[   12.580474] Bluetooth: RFCOMM ver 1.11
[   12.580596] Bluetooth: BNEP (Ethernet Emulation) ver 1.3
[   12.580718] Bluetooth: HIDP (Human Interface Emulation) ver 1.2
[   12.580993] ThumbEE CPU extension supported.
[   12.581634] clock_late_init() disabled 18 unused clocks
```

```
[   12.581909] bravo_wifi_init: start
[   12.582244] VFP support v0.3: implementor 51 architecture 0
   part 0f variant 0 rev 1
[   12.583679] regulator_init_complete: incomplete constraints,
   leaving ldo2 on
[   12.584808] regulator_init_complete: incomplete constraints,
   leaving ldo1 on
[   12.586029] regulator_init_complete: incomplete constraints,
   leaving dcdc3 on
[   12.587249] regulator_init_complete: incomplete constraints,
   leaving dcdc2 on
[   12.588348] regulator_init_complete: incomplete constraints,
   leaving dcdc1 on
[   12.589935] rs30000048:00010000 rs30000048:00010000: setting
   system clock to 2011-03-15 14:15:28 UTC (1300198528)
[   12.590240] Warning: unable to open an initial console.
[   12.590393] Freeing init memory: 112K
[   12.642669] init: Unable to open persistent property directory
   /data/property errno: 2
[   12.646087] enabling adb
[   12.646820] adb_open
[   12.665130] crc16: version magic '2.6.29-97da29ed preempt
   mod_unload ARMv7 ' should be '2.6.34.4-cyanogenmod preempt
   mod_unload ARMv7 '
[   12.699615] jbd2: version magic '2.6.29-97da29ed preempt
   mod_unload ARMv7 ' should be '2.6.34.4-cyanogenmod preempt
   mod_unload ARMv7 '
[   12.734954] ext4: version magic '2.6.29-97da29ed preempt
   mod_unload ARMv7 ' should be '2.6.34.4-cyanogenmod preempt
   mod_unload ARMv7 '
[   12.828765] lcdc_unblank: ()
[   12.828887] samsung_oled_panel_unblank: +()
[   12.830291] yaffs: dev is 32505860 name is \"mtdblock4\" rw
[   12.830474] yaffs: passed flags \"\"
[   12.830596] yaffs: Attempting MTD mount of 31.4,\"mtdblock4\"
[   12.839233] block 27 is bad
[   12.842590] block 46 is bad
[   12.848876] mmc1: host does not support reading read-only
   switch. assuming write-enable.
[   12.849121] mmc1: new high speed SDHC card at address e980
[   12.849761] mmcblk0: mmc1:e980 SU04G 3.69 GiB
```

```
[    12.850097]  mmcblk0:
[    12.852294] block 89 is bad
[    12.855865]  p1 p2 p3
[    12.873138] block 135 is bad
[    12.880676] block 162 is bad
[    12.915954] block 289 is bad
[    12.918823] block 299 is bad
[    12.940429] yaffs_read_super: isCheckpointed 0
[    13.075714] save exit: isCheckpointed 1
[    13.102691] batt:  55%, 3815 mV, -460 mA (-347 avg), 32.5 C,
   755 mAh
[    13.119110] samsung_oled_panel_unblank: -()
[    13.119995] batt: charging OFF
[    20.438995] report_key: send OJ action key, pressed: 0
[    20.683715] report_key: send OJ action key, pressed: 1
[    21.890014] sudden key ignore
[    22.656799] sudden key ignore
[    23.674011] report_key: send OJ action key, pressed: 0
[    23.895721] report_key: send OJ action key, pressed: 1
[    25.110687] save exit: isCheckpointed 1
[    25.534423] EXT4-fs (mmcblk0p2): warning: checktime reached,
   running e2fsck is recommended
[    25.538024] EXT4-fs (mmcblk0p2): mounted filesystem with
   ordered data mode
[    25.679016] FAT: invalid media value (0x00)
[    25.679229] VFS: Can't find a valid FAT filesystem on dev
   mmcblk0.
[    25.740661] yaffs: dev is 32505860 name is \"mtdblock4\" rw
[    25.740905] yaffs: passed flags \"\"
[    25.741027] yaffs: Attempting MTD mount of 31.4,\"mtdblock4\"
[    25.791351] block 27 is bad
[    25.796295] block 46 is bad
[    25.807281] block 89 is bad
[    25.818969] block 135 is bad
[    25.825927] block 162 is bad
[    25.858367] block 289 is bad
[    25.860931] block 299 is bad
[    25.866546] yaffs_read_super: isCheckpointed 0
[    25.959014] save exit: isCheckpointed 1
[    30.608489] report_key: send OJ action key, pressed: 0
[    30.741699] save exit: isCheckpointed 1
```

```
[    30.765411] report_key: send OJ action key, pressed: 1
[    30.816284] Restarting system.
[    30.816497]
[    30.816619] Restarting Linux version 2.6.34.4-cyanogenmod (
    android@giulio-laptop) (gcc version 4.4.0 (GCC) ) #49 PREEMPT
    Wed Aug 18 01:33:56 CEST 2010
[    30.816650]

No errors detected
","tag":"SYSTEM_LAST_KMSG","time_msec":1300198611341},
  {"value":"I:Set boot command \"\"

-- Wiping data...
Formatting DATA:...
mtd: not erasing bad block at 0x00060000
mtd: not erasing bad block at 0x00900000
mtd: not erasing bad block at 0x03460000
mtd: not erasing bad block at 0x03d20000
mtd: not erasing bad block at 0x057e0000
mtd: not erasing bad block at 0x05be0000
mtd: not erasing bad block at 0x05dc0000
mtd: not erasing bad block at 0x06ba0000
mtd: not erasing bad block at 0x077c0000
Formatting CACHE:...
mtd: not erasing bad block at 0x00340000
mtd: not erasing bad block at 0x005a0000
mtd: not erasing bad block at 0x00b00000
mtd: not erasing bad block at 0x010c0000
mtd: not erasing bad block at 0x01420000
mtd: not erasing bad block at 0x02400000
mtd: not erasing bad block at 0x02540000
Formatting SDEXT:...
rm: can't remove '.' or '..'
rm: can't remove '.' or '..'
Formatting SDCARD:/.android_secure...
Data wipe complete.
I:Set boot command \"\"
",
  "tag":"SYSTEM_RECOVERY_LOG",
  "time_msec":1300198611421},
 {"value":"",
```

```
  "tag":"NETWORK_DOWN",
  "time_msec":1300198651336},
 {"value":"NetworkInfo: type: WIFI[], state: CONNECTED/CONNECTED,
reason: (unspecified), extra: (none), roaming: false, failover:
   false,
isAvailable: true",
  "tag":"NETWORK_UP",
  "time_msec":1300198701821},{"value":"245000 3909
384000 318
460800 172
499200 109
576000 169
614400 78
652800 101
691200 48
768000 115
806400 73
844800 39
883200 70
998400 9148
",
   "tag":"CPUFREQ_STATS",
   "time_msec":1300198702728}
  ],
  "operator":"24202"
 },
 "digest":"",
 "logging_id":-7439885170310449089,
 "mac_addr":"002376D5FF84",
 "imei":"357841030259226",
 "locale":"en_NO"
}
```


A.2 checkin response from server


```
1300198703 DEBUG sslsniff : Read from Server (15:18:23.487204-*.
   google.com) :
HTTP/1.0 200 OK
Content-Type: application/json; charset=UTF-8
Date: Tue, 15 Mar 2011 14:18:23 GMT
Expires: Tue, 15 Mar 2011 14:18:23 GMT
```

```
Cache-Control: private, max-age=0
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
Server: GSE


1300198703 DEBUG sslsniff : Read from Server (15:18:23.487762-*.
    google.com) :

{"setting":
 [{"name":"android_id", "value":"230620085XXXXXXXXX"},
  {"name":"c2dm_auth_token","value":"1"},
  {"name":"checkin_interval","value":"164613"},
  {"name":"feedback_tos_pretty_url","value":"https://tools.google.
     com/userfeedback/external/android/tos.html"},
  {"name":"feedback_tos_url","value":"https://tools.google.com/
     userfeedback/external/android/tos.html"},
  {"name":"gmail_discard_error_uphill_op","value":"1"},
  {"name":"gmail_discard_error_uphill_op_new","value":"1"},
  {"name":"gmail_host","value":"gmail.com"},
  {"name":"gmail_max_preview_image_size","value":"2097152"},
  {"name":"gmail_num_retry_uphill_op","value":"45"},
  {"name":"gmail_use_multipart_protobuf","value":"1"},
  {"name":"gmail_wait_time_retry_uphill_op","value":"129600"},
  {"name":"google_login_generic_auth_service","value":"mail"},
  {"name":"google_login_public_key","value":"AAAAgMom/1a/
     v0lblO2Ubrt60J2gcuXSljGFQXgcyZWveWLEwo6prwgi3iJIZdodyhKZQrNWp5nKJ3srRXcUW
     +
     F1BD3baEVGcmEgqaLZUNBjm057pKRI16kB0YppeGx5qIQ5QjKzsR8ETQbKLNWgRY0QRNVz34kMJ
     /LgHax/6rmf5AAAAAwEAAQ=="},
  {"name":"google_services:ah","value":"Google Apps||Allows
     applications to sign in to Google Apps using the account(s)
     stored on this phone."},
  {"name":"google_services:finance","value":"Google Finance||
     Allows
 applications to sign in to Google Finance using the account(s)
    stored
 on this phone."},
  {"name":"google_services:geowiki","value":"Google Map
```

```
maker||Allows applications to sign in to the Google Map maker
    service
using the account(s) stored on this
phone."},
  {"name":"google_services:goanna_mobile","value":"Google
Tasks||Allows applications to sign in to the Google Tasks service
using the account(s) stored on this
phone."},
  {"name":"google_services:grandcentral","value":"Google
Voice||Allows applications to sign in to Google Voice using the
account(s) stored on this phone."},
  {"name":"google_services:local","value":"Google Maps||Allows
applications to sign in to Google Maps using the account(s) stored
    on
this phone."},
  {"name":"google_services:notebook","value":"Google
Notebook||Allows applications to sign in t

1300198703 DEBUG sslsniff :
Read from Server (15:18:23.488206-*.google.com) :
o Google Notebook using the account(s) stored on this phone."},
  {"name":"google_services:panoramio","value":"Panoramio||Allows
     applications to sign in to the Panoramio service using the
     account(s) stored on this phone."},
  {"name":"google_services:reader","value":"Google Reader||Allows
     applications to sign in to the Google Reader service using
     the account(s) stored on this phone."},
  {"name":"google_services:speechpersonalization","value":"
     Personalized Speech Recognition||Allows applications to sign
      in to the Personalized Speech Recognition service using the
      account(s) stored on this phone."},
  {"name":"gtalk_active_heartbeat_ping_interval_ms","value
     ":"1680000"},
  {"name":"gtalk_auth_sasl","value":"false"},
  {"name":"gtalk_binary_protocol","value":"true"},
  {"name":"gtalk_clock_skew_threshold_ms","value":"300000"},
  {"name":"gtalk_compress","value":"false"},
  {"name":"gtalk_flickr_photo_info_url","value":"http://api.flickr
     .com/services/rest/?method=flickr.photos.getinfo&photo_id=%s
     &api_key=%s"},
```

```
{"name":"gtalk_flickr_photo_url","value":"http://farm%s.static.
    flickr.com/%s/%s_%s_m.jpg"},
{"name":"gtalk_heartbeat_ping_interval_ms","value":"1680000"},
{"name":"gtalk_hostname","value":"mtalk.google.com"},
{"name":"gtalk_idle_timeout_ms","value":"300000"},
{"name":"gtalk_max_server_heartbeat_time","value":"900000"},
{"name":"gtalk_nosync_heartbeat_ping_interval_ms","value
    ":"1680000"},
{"name":"gtalk_picasa_album_url","value":"http://picasaweb.
    google.com/data/feed/api/user/%s/album/%s"},
{"name":"gtalk_rmq_ack_interval","value":"10"},
{"name":"gtalk_secure_port","value":"5228"},
{"name":"gtalk_ssl","value":"true"},
{"name":"gtalk_sync_heartbeat_ping_interval_ms","value
    ":"1680000"},
{"name":"gtalk_terms_of_service_url","value":"www.google.com/
    talk/terms.html"},
{"name":"gtalk_url_scraping_for_jpg","value":"true"},
{"name":"gtalk_youtube_video_url","value":"http://gdata.youtube.
    com/feeds/api/videos/%s"},
{"name":"latin_ime_voice_input_supported_locales","value":"en
    en_US en_GB en_AU en_CA en_IE en_IN en_NZ en_SG en_ZA zh
    zh_CN zh_TW zh_HK zh_SG ja ja_JP
1300198703 DEBUG sslsniff :
Read from Server (15:18:23.488577-*.google.com) :
 de de_CH de_DE de_AT de_LI es es_ES es_US fr fr_FR fr_BE fr_CH
    fr_CA it it_IT it_CH ko ko_KR pl pl_PL cs cs_CZ ru ru_RU tr
    tr_TR pt pt_BR nl nl_NL af af_ZA en_ZA zu zu_ZA"},
{"name":"maps_enable_friend_finder","value":"1"},
{"name":"maps_enable_navigation","value":"1"},
{"name":"market_force_checkin","value":"1"},
{"name":"mms_maximum_message_size","value":"307200"},
{"name":"mms_x_wap_profile_url","value":"http://www.google.com/
    oha/rdf/ua-profile-kila.xml"},
{"name":"mobile_transcoder_url","value":"http://www.google.com/
    gwt/n?u=%s"},
{"name":"parental_control_timeout_in_ms","value":"604800000"},
{"name":"perform_market_checkin","value":"true"},
{"name":"search_allow_voice_search_hints","value":"false"},
{"name":"search_voice_app_install_uri","value":"market://search?
    q=pname:com.google.android.voicesearch"},
```

{"name":"secure:latin_ime_voice_input_supported_locales","value":"en en_US en_GB en_AU en_CA en_IE en_IN en_NZ en_SG en_ZA zh zh_CN zh_TW zh_HK zh_SG ja ja_JP de de_CH de_DE de_AT de_LI es es_ES es_US fr fr_FR fr_BE fr_CH fr_CA it it_IT it_CH ko ko_KR pl pl_PL cs cs_CZ ru ru_RU tr tr_TR pt pt_BR nl nl_NL af af_ZA en_ZA zu zu_ZA"},
{"name":"secure:ssl_session_cache","value":"file"},
{"name":"settings_contributors_pretty_url","value":"https://www.google.com/mobile/android/basic/phone-contributors.html"},
{"name":"settings_contributors_url","value":"https://www.google.com/mobile/android/basic/phone-contributors.html"},
{"name":"settings_tos_pretty_url","value":"https://www.google.com/intl/%s/mobile/android/basic/phone-legal.html"},
{"name":"settings_tos_url","value":"https://www.google.com/intl/%s/mobile/android/basic/phone-legal.html"},
{"name":"ssl_session_cache","value":"file"},
{"name":"url:block_crash_reports","value":"http://android.clients.google.com/crash block"},
{"name":"url:calendar_sync_http_proxy","value":"http://www.google.com/calendar rewrite http://android.clients.google.com/proxy/calendar"},
{"name":"url:calendar_sync_https_proxy","value":"https://www.google.com/calendar rewrite https://android.clients.google.com/
1300198703 DEBUG sslsniff :
Read from Server (15:18:23.488931-*.google.com) :
proxy/calendar"},
{"name":"url:contacts_sync_http_proxy","value":"http://www.google.com/m8/feeds rewrite http://android.clients.google.com/proxy/contacts"},
{"name":"url:contacts_sync_https_proxy","value":"https://www.google.com/m8/feeds rewrite https://android.clients.google.com/proxy/contacts"},
{"name":"url:gmail_sync_http_proxy","value":"http://mail.google.com rewrite http://android.clients.google.com/proxy/gmail"},
{"name":"url:gmail_sync_https_proxy","value":"https://mail.google.com rewrite https://android.clients.google.com/proxy/gmail"},
{"name":"url:google_location_server","value":"http://www.google.com/loc/m/api rewrite https://www.google.com/loc/m/api"},

```
{"name":"url:redirect_youtube_from_sandbox_ytbs","value":"http
    ://jmt17.google.com/proxy/ytbs rewrite http://android.
    clients.google.com/proxy/ytbs"},
{"name":"url:redirect_youtube_from_sandbox_ytbt","value":"http
    ://jmt17.google.com/proxy/ytbt rewrite http://android.
    clients.google.com/proxy/ytbt"},
{"name":"url:vending_machine_billing_ssl_url","value":"https://
    android.clients.google.com/vending/billing/ rewrite https://
    android.clients.google.com/market/billing/"},
{"name":"url:vending_machine_efe_url","value":"https://
    android_efe.clients.google.com/vending/api/efe rewrite https
    ://android.clients.google.com/market/efe/"},
{"name":"url:vending_machine_ssl_url","value":"https://android.
    clients.google.com/vending/ rewrite https://android.clients.
    google.com/market/"},
{"name":"url:vending_machine_url","value":"http://android.
    clients.google.com/vending/ rewrite http://android.clients.
    google.com/market/"},
{"name":"url:vending_suggestion_url","value":"http://android.
    clients.google.com/vending/suggest/SuggRequest rewrite http
    ://android.clients.google.com/market/suggest/SuggRequest"},
{"name":"use_msisdn_token","value":"false"},
{"name":"vending_carrier_cred_buf_ms","value":"1800000"},
{"name":"vending_carrier_ref_freq_ms","value":"3888000000"},
{"name":"vending_hide_content_rating","value":"1"},
{"name":"vending_hide_warning_message","value":"false"},
{"name":"vending_remote_sugg

1300198703 DEBUG sslsniff :
Read from Server (15:18:23.489295-*.google.com) :
estion_type","value":"2"},
{"name":"vending_require_sim_for_purchase","value":"true"},
{"name":"vending_show_similar_tab_in_app_info_page","value
    ":"1"},
{"name":"vending_support_url","value":"http://market.android.com
    /support/"},
{"name":"vending_sync_frequency_ms","value":"172800000"},
{"name":"vending_tos_url","value":"https://www.google.com/intl/%
    locale%/mobile/android/market-tos.html"},
{"name":"vending_tos_version","value":"1.0.0"},
{"name":"vending_use_checkout_qa_service","value":"false"},
```

```
{"name":"voice_search:advanced_features_enabled","value":"1"},
{"name":"voice_search:alternate_backoff_languages","value":"zh-
    CN:cmn-Hans-CN zh-TW:cmn-Hant-TW zh-HK:yue-Hant-HK zh-SG:cmn
    -Hans-CN zh:cmn-Hans-CN ja:ja-JP de-CH:de-DE de-AT:de-DE de-
    LI:de-DE de:de-DE es-AR:es-ES es-BO:es-ES es-CL:es-ES es-CR:
    es-ES es-CO:es-ES es-DO:es-ES es-EC:es-ES es-GT:es-ES es-HN:
    es-ES es-NI:es-ES es-MX:es-ES es-PA:es-ES es-PE:es-ES es-PR:
    es-ES es-PY:es-ES es-SV:es-ES es-US:es-ES es-UY:es-ES es-VE:
    es-ES es:es-ES fr-BE:fr-FR fr-CH:fr-FR fr:fr-FR fr-CA:fr-FR
    it-CH:it-IT it:it-IT ko:ko-KR pl:pl-PL cs:cs-CZ ru:ru-RU tr:
    tr-TR pt-PT:pt-BR pt:pt-BR pt-AO:pt-BR nl-BE:nl-NL"},
{"name":"voice_search:help_video_url","value":"http://www.
    youtube.com/watch?v=tPPcTN5sdX4"},
{"name":"voice_search:personalization_countries","value":""},
{"name":"voice_search:personalization_v2_countries","value":"310
     311 312 313 314 315 316"},
{"name":"voice_search:supported_actions","value":"en-US
    :0,1,2,3,4 en-CA:0,1,2,3,4 en-GB:0,1,2,3,4 en-AU:0,1,2,3,4
    en-NZ:0,1,2,3,4 en-IN:0,1,2,3,4 en-001:0,1,2,3,4 cmn-Hans-CN
    : cmn-Hant-TW: ja-JP: de-DE: es-ES: fr-FR: it-IT: ko-KR: pl-
    PL: cs-CZ: ru-RU: tr-TR: pt-BR: nl-NL: af-ZA: en-ZA: zu-ZA:
    yue-Hant-HK:"},
{"name":"voice_search:supported_actions_new_numbering_scheme","
    value":"en-US:14,18,2,12,13,15,4,17,1,6,3 en-CA:2,3,4,1 en-
    GB:2,3,4,1 en-AU:2,3,4,1 en-NZ:2,3,4,1 en-IN:2,3,4,1 en
    -001:2,3,4,1 cmn-Hans-CN: cmn-Hant-TW: cmn-Hans-SG: ja-JP:
    de-DE: es-ES: fr-FR: it-IT: ko-KR: pl-PL: cs-CZ: ru-RU: tr-
    TR: pt-BR: nl-NL: af-ZA: en-ZA: zu-ZA: yue-Hant-

1300198703 DEBUG sslsniff :
Read from Server (15:18:23.489481-*.google.com) :
HK:"},
{"name":"voice_search:supported_languages","value":"af-ZA cmn-
    Hans-CN cmn-Hans-HK cmn-Hant-TW yue-Hant-HK cs-CZ nl-NL en-
    AU en-CA en-IN en-NZ en-ZA en-GB en-US en-001 fr-FR de-DE zu
    -ZA it-IT ja-JP ko-KR pl-PL pt-BR ru-RU es-ES tr-TR"},
{"name":"voice_search:unsupported_action_market_url_set_alarm","
    value":"http://www.google.com/support/mobile/bin/answer.py?
    answer=187559"},
{"name":"youtube_use_proxy","value":"true"}
],
```

```
"stats_ok":true,
"digest":"uSQsVWb3n5DThSfkPJCFtg==",
"time_msec":1300198703799,
"intent":[{"action":"android.server.checkin.FOTA_CANCEL"}]}
```

Appendix B HTC checkin

This the communication for the HTC phones “checkin”. It is modified to make it more readable, but the changes are only in the layout, not the content.

```
POST /android/checkin HTTP/1.1
Content-type: org/x-json
Content-Length: 421
Host: andchin.htc.com
Connection: Keep-Alive
User-Agent: Android-Checkin/2.0

{"checkin":
 {"build":
  {"product":"bravo",
   "id":"htc_wwe/htc_bravo/bravo/bravo:2.1-update1/ERE27/155070:
       user/release-keys",
   "revision":"129",
   "firmware_version":"1.15.405.4 CL155070 release-keys",
   "radio":"32.42.00.32U_5.09.00.08",
   "carrier":"htc_wwe",
   "bootloader":"0.75.0000",
   "build_type":"user",
   "changelist":"155070",
   "serialno":"HT03NPL04777"
  }
 },
 "model_number":"HTC Desire",
 "digest":"",
 "imei":"357841030259226",
 "locale":"en_NO"
}


HTTP/1.1 200 OK
Connection: close
Date: Thu, 17 Mar 2011 07:28:04 GMT
Accept-Ranges: bytes
Server: Noelios-Restlet-Engine/1.1.6
Content-Length: 1034
Content-Type: application/json; charset=UTF-8
```

```
X-Powered-By: Servlet/2.5 JSP/2.1

{"time_msec":"1300346884247",
 "stats_ok":true,
 "digest":"fc121695254dba9224698fa3f1acdfd",
 "intent":[{
  "extra":[{
   "name":"promptVersion",
   "value":"System upgrade1.21.405.2 (27.07 MB)We would recommend
      using a free Wi-Fi
hotspot or an unlimited data plan to apply this update. If not,
standard data connection charges may apply. Any questions? Contact
    us
via http://www.htc.com/www/CA_Hotline.aspx"},
  {"name":"promptMessage",
   "value":"System Upgrade"},
  {"name":"promptSize",
   "value":"27.07 MB"},
  {"name":"promptFeature",
   "value":"Update:HTC application improvement"},
  {"name":"promptMinutes",
   "value":"30,..."},
  {"name":"timeoutSeconds",
   "value":"120"},
  {"name":"force_update",
   "value":"false"}],
  "action":"android.server.checkin.FOTA_UPDATE",
  "data_uri":"http://fotadl.htc.com/OTA_Bravo_HTC_WWE_1
     .21.405.2-1.15.405.4_release_P4ohim1mtp6zggcyon.zip"}],
 "setting":[
  {"name":"upload_crash",
   "value":"http://63.241.57.68/crashblock"},
  {"name":"checkin_interval",
   "value":"604800"},
  {"name":"upload_event",
   "value":"false"},
  {"name":"update_stats",
   "value":"false"}
 ]
}
```

Appendix C Nexus One data

The Google Nexus One sent a lot of data. These are a couple of data logs found during our experiments. Some lines of text has been changed and we have inserted a new line to make the lines fit onto this report format. The changed lines are ending with a '↵

C.1 Nexus One Amazon connection

Sample data from the music player starting a connection to Amazon in December 2010.

```
GET/gp/dmusic/client-config.html?androidVersionCode=unknown&
    clientVersionName=1.8&androidVersionName=2.2.1&
    androidVersionSdkName=8&deviceManufacturer=HTC&locale=en_US&
    androidVersionSdkCode=8&clientVersionCode=800014&deviceModel=
    Nexus+One&carrier=Ventelo&clientVersionFull=1.8.14 HTTP/1.1
Host: www.amazon.com
Connection: Keep-Alive
User-Agent: Apache-HttpClient/UNAVAILABLE (java 1.4)

HTTP/1.1 200 OK
Date: Tue, 07 Dec 2010 09:30:44 GMT
Server: Server
x-amz-id-1: 0TZ6CDNRHYKHNT0E72QM
p3p: policyref="http://www.amazon.com/w3c/p3p.xml",CP="CAO DSP LAW
     CUR ADM IVAo IVDo CONo OTPo OUR DELi PUBi OTRi BUS PHY ONL
    UNI PUR FIN COM NAV INT DEM CNT STA HEA PRE LOC GOV OTC "
x-amz-id-2:
    Wib0kiSIGWrsXKTyAh3ne7avGJSZ5IPRIkmhmaRuIq8wMjWYweL7OGzzriK4vBfL

Vary: Accept-Encoding,User-Agent
Content-Type: text/html; charset=ISO-8859-1
Set-cookie: session-id-time=20827872011; path=/; domain=.amazon.
    com; expires=Tue Jan 01 08:00:01 2036 GMT
Set-cookie: session-id=000-0000000-0000000; path=/; domain=.amazon
    .com; expires=Tue Jan 01 08:00:01 2036 GMT
Transfer-Encoding: chunked

{"config":
{"associate_tag_ota":"androidmarket01-20",
 "associate_tag_wifi":"androidmarket00-20",
```

```
"update_check_latest_version":"1.8.29",
"update_check_critical_version":"0.0.0",
"update_check_website_url":"market://details?id=com.amazon.mp3",
"update_check_market_url":"market://details?id=com.amazon.mp3",
"browse_by_subgenre":true,
"claim_code_server_host":"www.amazon.com",
"high_speed_download_only":false,
"home_mshop_market_uri":"market://search?q=pub:\"Amazon Mobile
    \"",
"home_mshop_package_name":"com.amazon.mShop.android",
"home_mshop_show_advert":true,
"marketplace_id":"1",
"ota_enabled":true,
"purchase_server_device_type_id":"A1DL2DVDQVK3Q",
"purchase_server_host":"atv-ext.amazon.com",
"purchase_server_link_device_path":"/cdp/link/LinkDevice",
"purchase_server_port":443,
"purchase_server_purchase_path":"/cdp/library/Purchase",
"purchase_server_use_https":true,
"split_buy_button_price":false,
"url_account_setup":"http://www.amazon.com/gp/help/customer/
    display.html?ie=UTF8&nodeId=200443800&pop-up=1#acct",
"url_album_detail":"http://www.amazon.com/gp/dmusic/aws/
    lookupAlbum.html",
"url_artist_albums":"http://www.amazon.com/gp/dmusic/aws/
    artistAlbums.html",
"url_artist_tracks":"http://www.amazon.com/gp/dmusic/aws/
    artistTracks.html",
"url_base":"http://www.amazon.com",
"url_claim_code_terms":"http://www.amazon.com/gp/product/features
    /dv-gcpc-toc.html",
"url_contact_us":"http://www.amazon.com/gp/help/customer/display.
    html?ie=UTF8&nodeId=200443800&pop-up=1#contactmp3",
"url_genre_hierarchy":"http://www.amazon.com/gp/dmusic/aws/
    lookupGenreHierarchy.html",
"url_help":"http://www.amazon.com/gp/help/customer/display.html?
    ie=UTF8&nodeId=200443800&pop-up=1",
"url_lookup":"http://www.amazon.com/gp/dmusic/aws/lookup.html",
"url_merchandising":"http://www.amazon.com/gp/dmusic/aws/
    getMerchandisingCampaigns.html",
```

```
 "url_new_customer":"http://www.amazon.com/gp/help/customer/
    display.html?ie=UTF8&nodeId=200443800&pop-up=1#acct",
 "url_sample_track":"http://www.amazon.com/gp/dmusic/aws/
    sampleTrack.html",
 "url_search":"http://www.amazon.com/gp/dmusic/aws/search.html",
 "url_top_albums":"http://www.amazon.com/gp/dmusic/aws/topAlbums.
    html",
 "url_top_tracks":"http://www.amazon.com/gp/dmusic/aws/topTracks.
    html",
 "url_tos":"http://www.amazon.com/gp/dmusic/help/device-eula.html/
    ref=dm_android_eula",
 "auth_server_host":"firs-ta-g7g.amazon.com",
 "auth_server_port":"443",
 "auth_server_link_device_path":"/FirsProxy/registerDevice",
 "auth_server_use_https":"true",
 "auth_use_purchase_server":"false",
 "claim_code_server_port":"443",
 "claim_code_server_path":"/gp/dmusic/aws/redeemCode.html",
 "claim_code_server_use_https":"true",
 "allow_self_signed_certs":"false",
 "device_event_disable":"true",
 "device_event_domain":"IndigoAndroidRetailClient",
 "device_event_submission_interval_sec":"86400",
 "device_event_max_entries":"10000",
 "purchase_server_allow_partner_svc":"false"},
 "filename":"default.json",
 "responseHeader":{"contentType":"application/json"}
}
```


C.2 Nexus One Google Maps data


```
POST /glm/mmap HTTP/1.1
Content-Type: application/binary
Content-Length: 216
Host: mobilemaps.clients.google.com
Connection: Keep-Alive
User-Agent: GoogleMobile/1.0 (passion FRG83D); gzip

.............en_US..android:HTC-passion-Nexus One..4.7.0.07..gmm-
    android-google>...N... .*.GMM...SYSTEM...-17dd8aefd4eb7e2a
    .........8...GMM_3119427614516133278....K....
```

```
...
...
.....,.2........4.7.0..H.......O...O...O

HTTP/1.1 200 OK
Content-Type: application/binary
Content-Length: 68
Date: Tue, 07 Dec 2010 09:38:48 GMT
Expires: Tue, 07 Dec 2010 09:38:48 GMT
Cache-Control: private, max-age=0
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
Server: GSE

..>....
.409.p.....h.K...
....w.^.....".H.

...6.......*.
H.......O

POST /glm/mmap HTTP/1.1
Content-Type: application/binary
Content-Length: 210
Host: mobilemaps.clients.google.com
Connection: Keep-Alive
User-Agent: GoogleMobile/1.0 (passion FRG83D); gzip

..p.....h...en_US..android:HTC-passion-Nexus One..4.7.0.07..gmm-
    android-google>...X
.409...  .*.GMM...SYSTEM...-17dd8aefd4eb7e2a.......O......8...
    GMM_3119427614516133278...'..."" en/strings_remote_1343902255.
    dat


HTTP/1.1 200 OK
Content-Type: application/binary
Content-Length: 12302
Date: Tue, 07 Dec 2010 09:38:48 GMT
Expires: Tue, 07 Dec 2010 09:38:48 GMT
```

```
Cache-Control: private, max-age=0
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
Server: GSE

..>....'.............}Ks.W.....Xx./.%.H.@H‘Q.%M.4...w7..(P.M..gU
   %
(+some kb of data)


POST /glm/mmap HTTP/1.1
Content-Type: application/binary
Content-Length: 332
Host: mobilemaps.clients.google.com
Connection: Keep-Alive
User-Agent: GoogleMobile/1.0 (passion FRG83D); gzip

..p.....h...en..android:HTC-passion-Nexus One..4.7.0.07..\
gmm-android-google>...X
.409...  .*.GMM...SYSTEM...-17dd8aefd4eb7e2a.......O......8...
   GMM_3119427614516133278.....'...GA..'...GA
    ..........................
/.....................................................................


HTTP/1.1 200 OK
Content-Type: application/binary
Content-Length: 111695
Date: Tue, 07 Dec 2010 09:38:49 GMT
Expires: Tue, 07 Dec 2010 09:38:49 GMT
Cache-Control: private, max-age=0
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
Server: GSE

..>.......S.............,..PNG
.
...
(+150kb of data)
```

Appendix D Rooting earlier versions of Android

This is a listing of the procedures required to root and install superuser priveleges on the earliest Android phones. The section is included to demonstrate some of the manual work previously was necessary to achieve root level access and that has now been made more automatic with new tools like SuperOneClick[4].

The described process here was found online at Various sites during the fall of 2010. Credits have been given where possible, but most of the information was found at *MoDaCo.com*, *xda-developers.com* and *androidpolice.com*.

The last part of the section describes how to use the Superboot utility at this early stage of rooting phones.

D.1 Rooting the HTC Legend

Many sites give the information and the scripts to root the HTC Legend. The working ones all require you to run different code and scripts downloaded from Internet and may be a security risk already in the first script produced. Two sites with working rooting descriptions are at AndroidPolice[12] and MoDaCo[13].

Rooting details

If the phone is branded and locked we need a special gold card to unlock this specific phone from the provider. This is not necessary in our case, but images are loaded on a MicroSD-card inserted into the phone.

After downloading the memory card content - often referred to as a **gold card** and able to load the recovery and flashing modes - we start the process of rooting the phone by:

- inserting the memory card into the phone
- holding down the "back"-key while powering on
- waiting a couple of seconds for the FASTBOOT screen to appear
- connecting the Legend via USB to a computer with adb installed (Android SDK+drivers)
- device state changes to "FASTBOOT USB"
- erase cached data on the phone using fastboot

```
$ fastboot --help
usage: fastboot [ <option> ] <command>

commands:
```

[12] http://www.androidpolice.com/2010/04/29/htc-legend-rooting-instructions-now-available/

[13] http://android.modaco.com/content/htc-legend-legend-modaco-com/307487/24-may-r4-htc-legend-rooting-guide-now-with-1-31-x/

```
  update <filename>                          reflash device from update.zip
  flashall                                   flash boot + recovery + system
  flash <partition> [ <filename> ]           write a file to a flash partition
  erase <partition>                          erase a flash partition
  getvar <variable>                          display a bootloader variable
  boot <kernel> [ <ramdisk> ]                download and boot kernel
  flash:raw boot <kernel> [ <ramdisk> ]      create bootimage and flash it
  devices                                    list all connected devices
  reboot                                     reboot device normally
  reboot-bootloader                          reboot device into bootloader

options:
  -w                                         erase userdata and cache
  -s <serial number>                         specify device serial number
  -p <product>                               specify product name
  -c <cmdline>                               override kernel commandline
  -i <vendor id>                             specify a custom USB vendor id

$ fastboot devices
HT03MNX03946 fastboot

$ fastboot erase cache
erasing 'cache'... OKAY

$ fastboot oem rebootRUU
... OKAY
```

We have to write the downloaded testimage.zip that allows us to contact the phone in recovery mode and call our own (well for now someone else's) recovery script.

```
$ fastboot flash zip testimage.zip
sending 'zip' (121756 KB)... OKAY
writing 'zip'... INFOadopting the signature contained in this image...
INFOsignature checking...
INFOzip header checking...
INFOzip info parsing...
INFOchecking model ID...
INFOchecking custom ID...
INFOchecking main version...
INFOstart image[hboot] unzipping for pre-update check...
INFOstart image[hboot] unzipping & flushing...
```

```
INFO[RUU]UZ,hboot,0
INFO[RUU]UZ,hboot,100
INFO[RUU]WP,hboot,0
INFO[RUU]WP,hboot,100
INFOstart image[radio] unzipping & flushing...
INFO[RUU]UZ,radio,0
INFO[RUU]UZ,radio,5
INFO[RUU]UZ,radio,12
INFO[RUU]UZ,radio,17
INFO[RUU]UZ,radio,26
INFO[RUU]UZ,radio,34
INFO[RUU]UZ,radio,43
INFO[RUU]UZ,radio,51
INFO[RUU]UZ,radio,60
INFO[RUU]UZ,radio,68
INFO[RUU]UZ,radio,73
INFO[RUU]UZ,radio,81
INFO[RUU]UZ,radio,86
INFO[RUU]UZ,radio,94
INFO[RUU]UZ,radio,99
INFO[RUU]UZ,radio,100
INFO[RUU]WP,radio,0
INFO[RUU]WP,radio,6
INFO[RUU]WP,radio,11
INFO[RUU]WP,radio,20
INFO[RUU]WP,radio,28
INFO[RUU]WP,radio,33
INFO[RUU]WP,radio,42
INFO[RUU]WP,radio,53
INFO[RUU]WP,radio,100
INFOstart image[rcdata] unzipping & flushing...
INFO[RUU]UZ,rcdata,0
INFO[RUU]WP,rcdata,0
INFO[RUU]WP,rcdata,100
INFOstart image[boot] unzipping & flushing...
INFO[RUU]UZ,boot,0
INFO[RUU]UZ,boot,43
INFO[RUU]UZ,boot,84
INFO[RUU]UZ,boot,100
INFO[RUU]WP,boot,0
INFO[RUU]WP,boot,44
```

```
INFO[RUU]WP,boot,89
INFO[RUU]WP,boot,100
INFOstart image[recovery] unzipping & flushing...
INFO[RUU]UZ,recovery,0
INFO[RUU]UZ,recovery,20
INFO[RUU]UZ,recovery,47
INFO[RUU]UZ,recovery,99
INFO[RUU]UZ,recovery,100
INFO[RUU]WP,recovery,0
INFO[RUU]WP,recovery,22
INFO[RUU]WP,recovery,44
INFO[RUU]WP,recovery,66
INFO[RUU]WP,recovery,89
INFO[RUU]WP,recovery,100
INFOstart image[system] unzipping & flushing...
INFO[RUU]UZ,system,0
INFO[RUU]UZ,system,9
INFO[RUU]UZ,system,14
INFO[RUU]UZ,system,23
INFO[RUU]UZ,system,28
INFO[RUU]UZ,system,37
INFO[RUU]UZ,system,46
INFO[RUU]UZ,system,51
INFO[RUU]UZ,system,60
INFO[RUU]UZ,system,65
INFO[RUU]UZ,system,70
INFO[RUU]UZ,system,79
INFO[RUU]UZ,system,84
INFO[RUU]UZ,system,93
INFO[RUU]UZ,system,100
INFO[RUU]WP,system,0
INFO[RUU]WP,system,100
INFOstart image[userdata] unzipping & flushing...
INFO[RUU]UZ,userdata,0
INFO[RUU]UZ,userdata,100
INFO[RUU]WP,userdata,0
INFO[RUU]WP,userdata,100
INFOstart image[sp1] unzipping & flushing...
INFO[RUU]UZ,sp1,0
INFO[RUU]UZ,sp1,100
INFO[RUU]WP,sp1,0
```

```
INFO[RUU]WP,sp1,100
OKAY
```

And for (a yet) unknown reason it is recommended to flash twice. Have not tested on a branded phone, so the reason for this is unknown. It works with only one flash on the tested unlocked Legend phone.

```
$ sudo ./fastboot-linux flash zip testimage.zip
sending 'zip' (121756 KB)... OKAY
writing 'zip'... INFOadopting the signature contained in this image...
INFOsignature checking...
INFOzip header checking...
INFOzip info parsing...
INFOchecking model ID...
INFOchecking custom ID...
INFOchecking main version...
INFOstart image[hboot] unzipping for pre-update check...
INFOstart image[hboot] unzipping & flushing...
INFO[RUU]UZ,hboot,0
INFO[RUU]UZ,hboot,100
INFO[RUU]WP,hboot,0
INFO[RUU]WP,hboot,100
INFOstart image[radio] unzipping & flushing...
INFO[RUU]UZ,radio,0
INFO[RUU]UZ,radio,5
INFO[RUU]UZ,radio,12
INFO[RUU]UZ,radio,17
INFO[RUU]UZ,radio,26
INFO[RUU]UZ,radio,34
INFO[RUU]UZ,radio,43
INFO[RUU]UZ,radio,51
INFO[RUU]UZ,radio,60
INFO[RUU]UZ,radio,68
INFO[RUU]UZ,radio,73
INFO[RUU]UZ,radio,81
INFO[RUU]UZ,radio,86
INFO[RUU]UZ,radio,94
INFO[RUU]UZ,radio,99
INFO[RUU]UZ,radio,100
INFO[RUU]WP,radio,0
INFO[RUU]WP,radio,6
```

```
INFO[RUU]WP,radio,11
INFO[RUU]WP,radio,20
INFO[RUU]WP,radio,28
INFO[RUU]WP,radio,33
INFO[RUU]WP,radio,42
INFO[RUU]WP,radio,53
INFO[RUU]WP,radio,100
INFOstart image[rcdata] unzipping & flushing...
INFO[RUU]UZ,rcdata,0
INFO[RUU]WP,rcdata,0
INFO[RUU]WP,rcdata,100
INFOstart image[boot] unzipping & flushing...
INFO[RUU]UZ,boot,0
INFO[RUU]UZ,boot,43
INFO[RUU]UZ,boot,84
INFO[RUU]UZ,boot,100
INFO[RUU]WP,boot,0
INFO[RUU]WP,boot,44
INFO[RUU]WP,boot,89
INFO[RUU]WP,boot,100
INFOstart image[recovery] unzipping & flushing...
INFO[RUU]UZ,recovery,0
INFO[RUU]UZ,recovery,20
INFO[RUU]UZ,recovery,47
INFO[RUU]UZ,recovery,99
INFO[RUU]UZ,recovery,100
INFO[RUU]WP,recovery,0
INFO[RUU]WP,recovery,22
INFO[RUU]WP,recovery,44
INFO[RUU]WP,recovery,66
INFO[RUU]WP,recovery,89
INFO[RUU]WP,recovery,100
INFOstart image[system] unzipping & flushing...
INFO[RUU]UZ,system,0
INFO[RUU]UZ,system,9
INFO[RUU]UZ,system,14
INFO[RUU]UZ,system,23
INFO[RUU]UZ,system,28
INFO[RUU]UZ,system,37
INFO[RUU]UZ,system,46
INFO[RUU]UZ,system,51
```

```
INFO[RUU]UZ,system,60
INFO[RUU]UZ,system,65
INFO[RUU]UZ,system,70
INFO[RUU]UZ,system,79
INFO[RUU]UZ,system,84
INFO[RUU]UZ,system,93
INFO[RUU]UZ,system,100
INFO[RUU]WP,system,0
INFO[RUU]WP,system,100
INFOstart image[userdata] unzipping & flushing...
INFO[RUU]UZ,userdata,0
INFO[RUU]UZ,userdata,100
INFO[RUU]WP,userdata,0
INFO[RUU]WP,userdata,100
INFOstart image[sp1] unzipping & flushing...
INFO[RUU]UZ,sp1,0
INFO[RUU]UZ,sp1,100
INFO[RUU]WP,sp1,0
INFO[RUU]WP,sp1,100
OKAY
```

And if we make our own testimage.zip we may install whatever we like also at this level.

Then we must reboot to the new bootloader

```
$ fastboot reboot-bootloader
rebooting into bootloader... OKAY
```

HERE WE MUST REMEMBER TO SWITCH MicroSD-card with a normal (FAT32-formatted) one...(!)

The uSDcard may already contain the ROM or software you want to flash/update already and you can then skip the "adb push recoveryimage.zip /sdcard" line below.

Hold "Back" key in while powering on the device.

Press the "Power" button on top of the device and wait for HBOOT screen to appear

Then select RECOVERY (VOL UP/DOWN + Power for select)

A recovery screen appears with a picture of a phone with a red exclamation mark inside a triangle appears.

Then we can transfer the files needed to the phone after killing any old adb-server still running.

```
$ adb kill-server
$ adb push files /
* daemon not running. starting it now *
* daemon started successfully *
push: files/system/lib/libcutils.so -> /system/lib/libcutils.so
push: files/system/lib/libc.so -> /system/lib/libc.so
push: files/system/lib/liblog.so -> /system/lib/liblog.so
push: files/system/lib/libstdc++.so -> /system/lib/libstdc++.so
push: files/system/lib/libm.so -> /system/lib/libm.so
push: files/system/bin/sh -> /system/bin/sh
push: files/system/bin/linker -> /system/bin/linker
push: files/sbin/sdparted -> /sbin/sdparted
push: files/sbin/log2sd -> /sbin/log2sd
push: files/sbin/parted -> /sbin/parted
push: files/sbin/um -> /sbin/um
push: files/sbin/backuptool.sh -> /sbin/backuptool.sh
push: files/sbin/fs -> /sbin/fs
push: files/sbin/dump_image -> /sbin/dump_image
push: files/sbin/flash_image -> /sbin/flash_image
push: files/sbin/ums_toggle -> /sbin/ums_toggle
push: files/sbin/e2fsck -> /sbin/e2fsck
push: files/sbin/unyaffs -> /sbin/unyaffs
push: files/sbin/mke2fs -> /sbin/mke2fs
push: files/sbin/wipe -> /sbin/wipe
push: files/sbin/toolbox -> /sbin/toolbox
push: files/sbin/reboot -> /sbin/reboot
push: files/sbin/fix_permissions -> /sbin/fix_permissions
push: files/sbin/tune2fs -> /sbin/tune2fs
push: files/sbin/busybox -> /sbin/busybox
push: files/sbin/adbd -> /sbin/adbd
push: files/sbin/mkyaffs2image -> /sbin/mkyaffs2image
push: files/sbin/recovery -> /sbin/recovery
push: files/sbin/nandroid-mobile.sh -> /sbin/nandroid-mobile.sh
push: files/etc/fstab -> /etc/fstab
push: files/etc/mtab -> /etc/mtab
31 files pushed. 0 files skipped.
1215 KB/s (3734509 bytes in 2.999s)
```

Then we install the busybox toolset with superuser application and *su* binary to have some tools to work with...

```
$ adb shell busybox --install /sbin
```

Then we can list the newly created /sbin "busybox tools"

```
$ adb shell ls  /sbin
[                   hwclock             renice
[[                  ifconfig            reset
adbd                ifdown              resize
arping              ifup                rm
ash                 insmod              rmdir
awk                 install             rmmod
backuptool.sh       ip                  route
basename            ipaddr              run-parts
bbconfig            ipcalc              sdparted
bunzip2             iplink              sed
busybox             iproute             seq
bzcat               iprule              setconsole
bzip2               iptunnel            setkeycodes
cat                 kbd_mode            setlogcons
catv                kill                setsid
chattr              killall             sh
chgrp               killall5            sha1sum
chmod               last                showkey
choice_fn           length              sleep
chown               less                sort
chroot              ln                  split
chrt                loadfont            stat
chvt                loadkmap            strings
cksum               log2sd              stty
clear               losetup             sum
cmp                 ls                  swapoff
cp                  lsattr              swapon
crond               lsmod               switch_root
crontab             makedevs            sync
cut                 md5sum              sysctl
date                mdev                tac
dc                  mkdir               tail
dd                  mkdosfs             tar
deallocvt           mke2fs              tcpsvd
depmod              mke2fs_recvy        tee
detect_key          mkfifo              telnet
```

```
devmem
df
dhcprelay
diff
dirname
dmesg
dnsd
dnsdomainname
dos2unix
du
dump_image
dumpkmap
dumpleases
e2fsck
echo
egrep
env
ether-wake
expr
false
fbset
fbsplash
fdisk
fgrep
find
fix_permissions
flash_image
fold
free
freeramdisk
fs
fsck
fuser
getopt
grep
gunzip
gzip
head
hexdump
hostname
mkfs.vfat
mknod
mkswap
mktemp
mkyaffs2image
modprobe
more
mount
mountpoint
mv
nameif
nandroid-mobile.sh
nc
netstat
nice
nmeter
nohup
nslookup
od
offmode_charging
openvt
parted
patch
pidof
ping
pipe_progress
pivot_root
power_test
printenv
printf
ps
pscan
pwd
rdate
rdev
readlink
readprofile
realpath
reboot
recovery
telnetd
test
tftp
time
toolbox
top
touch
tr
traceroute
true
tty
tunctl
tune2fs
udhcpd
udpsvd
um
umount
ums_toggle
uname
uncompress
uniq
unix2dos
unyaffs
unzip
uptime
usleep
uudecode
uuencode
vconfig
vi
watch
wc
wget
which
who
whoami
wipe
xargs
yes
zcat
```

Giving us lots of opportunities to work with. Then we mount the new file system SDcard and transfer the update we want to write to the system to the sdcard.

```
$ adb shell mount /sdcard
$ adb push rootedupdate.zip /sdcard
1235 KB/s (110689022 bytes in 87.474s)
```

And then we wipe the phone and install rootedupdate.zip from /sdcard to the new system.

```
$ adb shell nohup /sbin/recovery &
[1] 3171
$ nohup: appending output to nohup.out
```

Now we get the "REAL" system recovery screen to select wipe/partition/flash/etc.

D.2 Superboot user instructions for Nexus One

These are the installation instructions found on Superboot in August 2010 at MoDaCo[14].

```
How to use Superboot - Windows, Linux and OSX

- Download the Superboot zip file above and extract to a directory
- Put your device in bootloader mode - Turn off the phone then press
  and hold the trackball to enter the bootloader
- WINDOWS - double click 'install-superboot-windows.bat'
- MAC - Open a terminal window to the directory containing the files,
  and type 'chmod +x install-superboot-mac.sh' followed by
  './install-superboot-mac.sh'
- LINUX - Open a terminal window to the directory containing the
  files, and type 'chmod +x install-superboot-linux.sh' followed by
  './install-superboot-linux.sh'

Note: If you are using a retail Nexus One, you may need to unlock the
bootloader first, as detailed here.

That's it, job done! Enjoy!

P
```

[14] http://www.modaco.com/topic/298782-08mar-superboot-erd79-gri40-rooting-the-nexus-one/

Unlock the bootloader

Retail devices DO have locked bootloaders.

But you can unlock them.

You see, in the bootloader of the Nexus One there is a secret command. A command that, when issued via fastboot, gives you the warning you see here, after which you can merrily unlock your bootloader and flash anything you want onto your device (including our superboot rooting image).

Nice eh?

It's easy too! Google WANTS you to be able to do whatever you please on your Nexus One, but they also want you to be aware of the consequences.

To unlock your Nexus One's bootloader simply...

- Download and extract fastboot from here (Windows, Linux and Mac included)
- Run a command prompt / terminal at the directory you just extracted
- Type 'fastboot-windows oem unlock' or './fastboot-mac oem unlock' or
  './fastboot-linux oem unlock' (as appropriate)
- Proceed on your merry bootloader unlocked way

Enjoy! Big nod to packetlss for the heads up!

P