Mobile Authentication Subspace Travel

HITBSECCONF2015 AMSTERDAM

Markus Vervier
markus.vervier@lsexperts.de

May 28, 2015

Abstract

The nature of current mobile network devices makes active attacks feasible. Even when acquiring the permanent secret K_i from a SIM card is not possible, temporary authentication tokens can often be retrieved and forwarded. They suffice to successfully impersonate a mobile network user on a remote device. To impersonate a mobile user, we transfer network authentication to a remote device via different channels. Permanent access to a SIM card is not necessary as authentication tokens are valid for a certain amount of time. Different methods to acquire such tokens exist affecting millions of mobile devices. They range from SIM card access via USB and Bluetooth to exploiting baseband firmware and vendor API calls.

By modifying a baseband firmware the acquired authentication tokens can be used in a stock off-the-shelf phone using a virtual SIM. Other baseband modifications allow hardening of existing basebands by reducing attack surface and adding custom defensive functionality.

Contents

1 Attacking Mobile Network Authentication

Security in mobile networks should achieve several protection goals such as integrity and confidentiality of data transmitted over the air. Also authenticity should be established in order to correctly bill customers and prevent fraud.
In modern cellular 4G, 3G and 2G networks, algorithms of the A5/X family are used to implement these goals. A Subscriber Identity Module (SIM) plays a key role in this concept. It contains an unique identifier, the International Mobile Subscriber Identity (IMSI) as well as a secret key K_i that should be only present in the SIM and the Authentication Center (AuC) of the mobile network. Temporary authentication tokens are calculated using K_i. They are not as well protected as K_i and must be distributed to mobile network infrastructure and to the mobile device. The SIM as well as all other aspects of the mobile network connection are controlled by a specialized baseband firmware running on a separate baseband processor (BP) that is part of a mobile device. While there are open source implementations[1] for older mobile phones, there is no open baseband firmware available for modern smartphones so far.

2 Previous Work

Several attack vectors against mobile network authentication and privacy are known and have been discussed in the past.
Examples are attacks against the cryptographic algorithms of the *A5/X*-family. The *A5/1* and *A5/2* algorithms used in *GSM* for *OTA*-Privacy are considered broken[2]. For the algorithm *A5/3* (KASUMI) currently used in 3G-networks, no practical attack exists in public knowledge in context of a mobile network.
More recent attacks[3] focus on the exploitation of SIM cards to retrieve the secret key K_i. While such attacks lead to full permanent compromise of a mobile network identity, they exploit specific bugs introduced by vendors.

New Attack Vectors The widespread use of smartphones opens up new attack vectors that were not feasible before. This is mainly due to two reasons:

1. Smartphones have multiple communication channels besides a single mobile network connection (WiFi, Bluetooth, Dual-SIM).
2. Third-Party-Software is deployed onto phones on a massive scale.

Malware exists in the Android and IOS ecosystems that can exploit vulnerabilities in the operating system to elevate privileges and take control over the OS running on the Application Processor.
Based on these observations new attacks are possible that work on a general scale. By using legitimate features and vendor API calls it is possible to access SIM cards on different phones and send commands to them.

[1] `http://bb.osmocom.org/trac/`
[2] `https://en.wikipedia.org/wiki/A5/1#Attacks_on_A5.2F1_as_used_in_GSM`
[3] `http://securitywatch.pcmag.com/mobile-security/313914-encryption-bug-in-sim-card-can-`

AT-Commands It was discovered that sensitive data is exposed via AT commands on different devices. Most notably the possibility to execute unrestricted APDU-Commands on a SIM card and to receive the results is a long time underrated attack vector.
Mobile phones expose AT command interfaces to allow external devices to create dial up data connections, read SMS and use other mobile services. Especially older Android based devices expose AT command interfaces by default via USB and Bluetooth. Recent devices have better protections as for example asking for explicit user interaction before exposing dial up networking or replacing the dial up networking with USB and WiFi tethering. Still there are millions of vulnerable legacy devices in use worldwide. Also SIM card access is possible on the most recent devices where the attacker has root privileges on the AP-OS.
A stock Samsung Galaxy Y Duos (Model S6102b) device exposed an AT command interface as serial device via USB by default when tested. No user interaction was required and no notification was shown when connecting the device via USB to a Linux system.
By using the *AT+CSIM* command valid network authentication could be retrieved:

```
$ cu -h -l /dev/ttyACM0
Connected.
AT+CSIM=14,"A0A40000027F20"
+CSIM: "9F17"
OK
AT+CSIM=10,"A0B0000017" <= This is necessary
+CSIM: "9400"
OK
AT+CSIM=42,"A088000010123456789012345678901234567890FF"
+CSIM: "9F0C"
OK
AT+CSIM=10,"A0C000000C"
+CSIM: "3852AF1083C2D92A74E3A8009000" <= Authentication Result
OK
```

The serial modem interface is exposed by default via USB. This means a malicious device[4] hidden in an USB charger may use this feature to impersonate mobile network users.
On recent devices, the *AT+CSIM* command is not always available. To acquire authentication the *AT+EAUTH* and *AT+ESIMAUTH* commands do an authentication via the SIM card and return the result.
Additional methods of accessing the SIM card were found in Bluetooth *Dial up Networking* (DUN) and via local methods on older Android phones. On several devices various security related flaws have been identified. Most notably on older Alcatel OT-918D and OT-903D devices, permissions for pseudo terminal devices belonging to gsm0710d are set to world readable and writable. This allows even unprivileged apps to access the AT-Command prompt and execute commands on both SIM cards of the phone.

Bluetooth / BT-SAP Besides AT-Command access via Bluetooth-DUN (Dial Up Networking) there is another way that SIM cards may be exposed over Bluetooth: Bluetooth-SAP

[4] http://www.blackhat.com/us-13/briefings.html#Lau

(SIM Access Profile).
It is intended for cars that have their own mobile radio unit. By exposing the SIM card via Bluetooth it is possible to use it remotely in another device.
A malicious Bluetooth device may use this feature to forward SIM card access to remote devices.
To use the information retrieved from the SIM by the methods described above, firmware of a stock smartphone was modified.

3 Baseband and Application Processor

Modern phones are divided into two separate units. The Baseband Processor (BP) and the Application Processor (AP). They are located on logically separated systems with different memory spaces. While the AP is usually running a full general purpose OS (Android/Linux or IOS), the baseband processor is often a real time OS (e.g. Nucleus OS based) or a DSP with hardcoded functionalities.

Baseband Processor (BP) The Baseband Processor (BP) / modem takes care of mobile network communication. It controls the radio network connection and all other things related to the mobile network connection. The amount of executable code can be very large depending on the capabilities of the modem. Parsing of different protocols for OTA communication as well as TCP/IP and PPP stacks mean a large attack surface.

Application Processor (AP) On modern phones user interface and third party applications run on the Application Processor (AP). It runs an OS that is managed by the user to some degree. IOS and Android do not give the user full control over the OS (”root”) by default. Nevertheless many users root or jailbreak their phones. Also malware may exploit kernel vulnerabilities to gain kernel level privileges and usually full control over the operating system running on the AP.
Normally no direct interaction between an external application and the BP-OS should be possible. An exception of this are AT-Command-Prompts that are exposed via Bluetooth (Dial-Up-Networking), vendor APIs and local devices on the phone. The security permissions of modern Android and IOS devices try to prevent full access to the AT command prompt for unprivileged apps. For rooted phones this restriction does not apply as the interaction between AP and BP is still based on AT commands for most basebands.
In general external data may be sent to the BP and is processed by it using standard APIs. Examples are phone numbers, USSD codes and SMS. This poses an attack vector to gain code execution on the BP.

4 MediaTek Basebands

A MediaTek (MTK) based system was chosen to create a proof of concept. To understand more about the MTK baseband an analysis of the firmware of different phones based on the

MT6573, MT6577, MT6589 and MT6595 chipsets has been performed. The following information applies to all of these, while minor differences might be observed.

Firmware Image The baseband firmware has no reserved place on the phone flash memory. It is located in a normal file in the system.img Android filesystem image. All relevant firmware files are located in */etc/firmware/* and the modem image is called *modem*.img* - depending on the version of the modem.
During boot of the Android system the */system/bin/ccci_mdinit* utility triggers loading of the modem image into the BP memory using a Linux kernel driver interface called *Cross Core Communication Interface* (ccci). It was discovered that the BP memory is accessible from the AP and the image is directly written into the memory space and the load triggered. In case of a CPU exception the AP is able to reset the BP using a special hardware device inside the phone.
The permissions on the modem image are 0622 which means that all users can read the image. Modification is possible for the root user. Therefore even unprivileged apps can read the contents of the baseband image.

Image Structure The modem image is an uncompressed raw binary file. It is a partial image of the firmware memory space starting at address *0x00000000* and includes padding as well as a trailer with various metadata:

```
04e1ba8: efd8 0020 30bd 0000 ccfc adf0  ... 0.......
04e1bb4: 4348 4543 4b5f 4845 4144 4552  CHECK_HEADER
04e1bc0: 0100 0000 0200 0000 0200 0000  ............
04e1bcc: 4d54 3635 3733 5f53 3031 0000  MT6573_S01..
04e1bd8: 0000 0000 3230 3132 2f30 342f  ....2012/04/
04e1be4: 3233 2031 363a 3231 0000 0000  23 16:21....
04e1bf0: 0000 0000 0000 0000 0000 0000  ............
04e1bfc: 0000 0000 0000 0000 0000 0000  ............
04e1c08: 0000 0000 0000 0000 0000 0000  ............
04e1c14: 0000 0000 0000 0000 4d41 5549  ........MAUI
04e1c20: 2e31 3141 4d44 2e57 3131 2e33  .11AMD.W11.3
04e1c2c: 3700 0000 0000 0000 0000 0000  7...........
04e1c38: 0000 0000 0000 0000 0000 0000  ............
04e1c44: 0000 0000 0000 0000 0000 0000  ............
04e1c50: 0000 0000 0000 0000 0000 0000  ............
04e1c5c: 0000 0000 0000 0000 0000 0000  ............
04e1c68: 0000 0000 bc00 0000            ........
```

The trailer also contains a hash that was initially checked by the kernel driver which gives a warning message if it is incorrect. It seems to be only an integrity protection and signatures are not enforced on the hardware side. Functionality for cryptographic signatures of the firmware image is also present in the released source code of various devices. So far no device was discovered that actually used this functionality.

MTK baseband firmwares are based on an older version of *Nucleus RTOS*[5] running on ARM using Thumb as well as ARM mode. It does not use virtual memory. In case of an Alcatel OT-918D image available during research the entry point is located at offset *0x0*. Up to offset *0x1c* there is a table of exception handlers:

```
00000000        ldr     pc, = sub_a0
00000004        ldr     pc, = sub_368   ;
   XREF=rmmi_ebtsap_hdlr+150, sub_285cb0+38, sub_30f520+126,
   sub_43a6c0+18
00000008        ldr     pc, = sub_388   ; XREF=sub_1cb7c8+6
0000000c        ldr     pc, = sub_3c4   ; XREF=sub_1cb7c8+12,
   sub_215c48+78, sub_2bc100+96
00000010        ldr     pc, = 0x3e4     ; XREF=sub_1cb7c8+10,
   sub_215c48+178
00000014        ldr     pc, = EntryPoint
00000018        ldr     pc, = 0x5000e698
0000001c        ldr     pc, = 0x300
```

They are called when exceptions occur, e.g. *DATA ABORT*.
A lot of profiling code and assertions are present inside the image. This includes strings that give hints on the purpose of different functions:

```
00004b28        db      "hal/peripheral/src/drv_hisr.c", 0
```

MediaTek uses several proprietary tools[6] to debug the BP. To support this many parts of the firmware call trace functions. Debug and trace information are forwarded to these tools running on the AP side. Even without access to vendor debug tools they are very helpful when reverse engineering a firmware image. In case assertions are false, the error messages give hints about variable names and expected conditions:

```
001e787c        db      "t\xD1\x10PInvalid␣parameter:␣file=%s,␣
   line=%d.␣p1=%d,␣p2=%d,␣p3=%d.", 0
```

The different functions of the baseband such as radio communication, SIM card access and others run as user tasks of the RTOS. They process inter layer messages (ILM) from a dedicated message queue. The message format has the following structure:

```
00000000 ilm_struct     struc ; (sizeof=0x18, align=0x4,
   mappedto_805)
00000000 src_mod_id     DCD ?
00000004 dest_mod_id    DCD ?
00000008 sap_id         DCD ?
0000000C msg_id         DCD ?
00000010 local_para_ptr DCD ?     ; offset
00000014 peer_buff_ptr  DCD ?     ; offset
00000018 ilm_struct     ends
```

[5] http://www.mentor.com/embedded-software/nucleus/
[6] Catcher / META

The task will process the message depending on the *sap_id* and send the result back in another ILM-message to the task with id *src_mod_id*. If a task needs access to the SIM card it will send a message to the task responsible for SIM card communication.

BP Memory As stated before, the OS running on the BP does not use virtual memory. Instead there is a single global address space that the different tasks have access to. Besides usual CODE, BSS and DATA segments there are also other segments that are shared between different parts of the hardware. Most notable there are shared segments between BP and AP as well as different DSPs.

Communication between AP and BP Data exchange between AP and BP is implemented via two main methods:

- Serial UART
- Shared Memory (SRAM)

A Linux kernel driver for the ccci communication interface exposes various functionality via character devices and IOCTLs. Source code for different models and the kernel drivers is available from various vendors[7].
Baseband commands are sent using the AT-Command-Processor available on /dev/ttyC0. Serial multiplexing using *CMUX* as specified by 3GPP[8] is used via a *gsm0710muxd*.

5 Firmware Modification

MTK based devices have a broad userbase especially in Asia. Therefore there are a lot of useable devices from different vendors. Also several documents are available at different sources that describe details of interest to baseband development. The most interesting part is that the AP controls the BP and both systems are ARM based. Development is possible with standard Android/Linux toolchains. The usage of an ARM architecture makes development of patches easier in contrast to fully DSP based basebands.
Custom baseband modification can be divided into two main categories: Adding new features and hardening the baseband against attacks.
While it would be entirely possible to create a new baseband from scratch to replace the original firmware, custom patches allow to keep existing functionality.

ShadowSIM - Active Cloning

[7] `http://sourceforge.net/projects/alcatel/`
[8] `http://www.3gpp.org/ftp/Specs/archive/07_series/07.10/0710-720.zip`

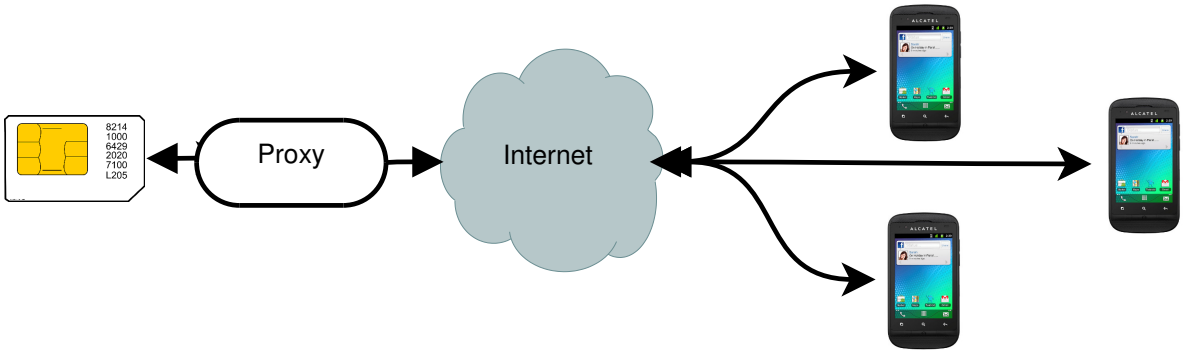

Figure 1: Active Cloning

As a proof of concept the ShadowSIM-Project[9] was developed. It implements a virtual SIM card on an Alcatel OT-918D phone. By patching the relevant functions in the original baseband firmware it is possible to redirect commands intended for a physical SIM card to the AP side of the phone.
This way a virtual SIM card is implemented. It can be shared between several phones which would allow an active cloning attack on mobile network identities. While current SIMs card can not be copied/cloned easily, the mobile network identity can. Devices are able register to mobile networks by using a remote SIM card. During network authentication temporary cryptographic tokens (e.g. XRES, CK, IK in 3G-networks) are generated from the secret key K_i that is located in the SIM card and the Authentication Center (AUC). As these temporary keys are valid and sufficient until the next authentication request, an active connection to the remote SIM card is only needed at the time of authentication.
Mobile networks have different requirements on re-authentication. Most networks require a new authentication (and therefore an active connection to the remote SIM card) each time a call is initiated from the mobile device. For data connections a new authentication is required on average every 24 hours, depending on the network.
Forwarding of SIM commands is done via TCP/IP over an internet connection. Therefore an active internet connection is required. Other communication channels may also be possible when the latency is low enough to not trigger authentication timeouts enforced by the mobile network side.

Hardening Another application of firmware patches is hardening and improving the security of the baseband against attacks.
An easy as well as effective measure to improve the security of a system is to reduce the attack surface. This is achieved by disabling unneeded functionality of the baseband firmware.
A prime example for this is the *SIM Application Toolkit* (often called STK or SAT). It allows so called "*value added services*" that may work outside of user control. Features of the STK can be used for surveillance and may be considered as undesired in hostile environment.
By disabling the relevant functions inside the baseband, attacks using STK can be prevented and detected.

[9] `https://github.com/shadowsim/shadowsim`

The same applies for network location tracking, which is a feature that may be used for surveillance. Location tracking on the baseband is mostly outside of user control. While on the AP location information is restricted by app permissions, network location information on the BP may be requested by the mobile network without the user noticing. By modifying the baseband firmware such attacks can be discovered, detected and counter measures implemented. As signatures, compression or other obfuscation methods are not present in the firmware image, changing code is possible using standard tools such as Hopper[10], IDAPro[11] or radare2[12]. As a proof of concept processing of proactive SIM toolkit commands on an Alcatel OT-918D is disabled by changing opcodes at the following offset:

```
ROM1:000F972C   MOVS r0, #0x0 ;          was: PUSH {R3-R7,LR}
ROM1:000F972E   BX LR ; <= do nothing    was: MOV R5, R1
ROM1:000F9730   MOV     R6, R0
ROM1:000F9732   MOV     R0, R1
```

6 Conclusion and Future Work

The possibility of remote SIM access on consumer smartphones are two-fold. On one hand this poses a security threat as remote attackers may impersonate mobile network users.
On the other hand the possibility to temporarily share a mobile network identity opens up new potentials. New commercial as well as social apps could be imagined where a user is able to share his mobile network identity or even borrow another identity for a limited amount of time. Key players such as Apple[13] as well as smaller startups[14] are already testing such applications. The foundation of all these systems is an open baseband that includes such functionality. As shown above it is possible to modify existing basebands of stock Android phones and enhance their functionality.
Future work should focus on creating a fully open baseband firmware for stock Android phones. As long as this goal is not achieved, parts of the baseband can be replaced with custom code to implement needed features that are not provided by baseband vendors yet.

[10] http://www.hopperapp.com
[11] https://www.hex-rays.com/products/ida/index.shtml
[12] http://radare.org/r/
[13] https://en.wikipedia.org/wiki/Apple_SIM
[14] http://www.cell-buddy.com/