QSEE TrustZone Kernel Integer Overflow Vulnerability

Dan Rosenberg
dr@azimuthsecurity.com

July 1, 2014

1 Introduction

This paper discusses the nature of a vulnerability within the Qualcomm QSEE TrustZone implementation as present on a wide variety of Android devices. The actual exploitation mechanisms employed in the proof-of-concept exploit are not covered in this document.

2 Software Description

"ARM TrustZone technology is a system-wide approach to security for a wide array of client and server computing platforms, including handsets, tablets, wearable devices and enterprise systems. Applications enabled by the technology are extremely varied but include payment protection technology, digital rights management, BYOD, and a host of secured enterprise solutions." [2]

3 Background

TrustZone segments both hardware and software into "secure" and "non-secure" domains, referred to as "worlds". The non-secure world includes the traditional operating system kernel and its corresponding userland applications, while the secure world often includes a trusted operating system referred to as a Trusted Execution Environment (TEE). Software running in the secure world has privileged access to all hardware and the non-secure world, but the non-secure world is restricted from accessing device peripherals and memory regions designated by TrustZone as protected. The non-secure world may issue requests to the secure world via a number of mechanisms, including the privileged Secure Monitor Call (SMC) ARM instruction. Further documentation on TrustZone may be found on ARM's website [1].

4 Affected Devices

The vulnerability described in this document affects Qualcomm's implementation of the Trusted Execution Environment ("QSEE") as present on a wide variety of Android mobile devices. At the time of this writing, the vulnerability is present on all known Android devices that support TrustZone and utilize a Qualcomm Snapdragon SoC, with the exception of the Samsung Galaxy S5 and HTC One M8, which have been patched. It is expected that by the time the details of this document have been made public, a more extensive list of devices will have been patched by their respective vendors.

This vulnerability is known to affect a wide range of flagship devices, including the LG Nexus 4, LG Nexus 5, LG G2, HTC One series, Moto X, Samsung Galaxy S4, and Samsung Galaxy Note 3.

5 Vulnerability Overview

Due to a flaw in bounds-checking Secure Monitor Call (SMC) requests, an attacker with kernel-level privileges (SVC mode) may issue specially crafted SMC requests to cause QSEE to write controlled data to arbitrary secure memory. This may be exploited to execute arbitrary code in the context of QSEE.

6 Vulnerability Description

Privileged non-secure code (i.e. the Linux kernel) may request services of QSEE by issuing a Secure Monitor Call (SMC) instruction. QSEE supports two calling conventions when using the SMC instruction: a call-by-registers convention, and a second convention that uses a command structure to provide arguments, as depicted in Figure 1. Included in the command structure is a request header which is populated by the non-secure world, and a response header which is populated by QSEE on completion of the SMC request.

When QSEE receives an SMC request using a command structure, it performs several checks on the request header to ensure validity. In particular, the following conditions must be satisfied for a command header to be considered valid (reverse engineered from a TrustZone image):

1. `req.len` >= 16 (*"Is the command length larger than the fixed size of the request header?"*)
2. `req.buf_offset` < `req.len` (*"Does the request input buffer reside within the command buffer?"*)
3. `req.buf_offset` >= 16 (*"Does the request input buffer begin after the request header?"*)
4. `qsee_is_ns_memory(req, req.len)` == `true` (*"Does the entire command buffer reside in non-secure memory?"*)

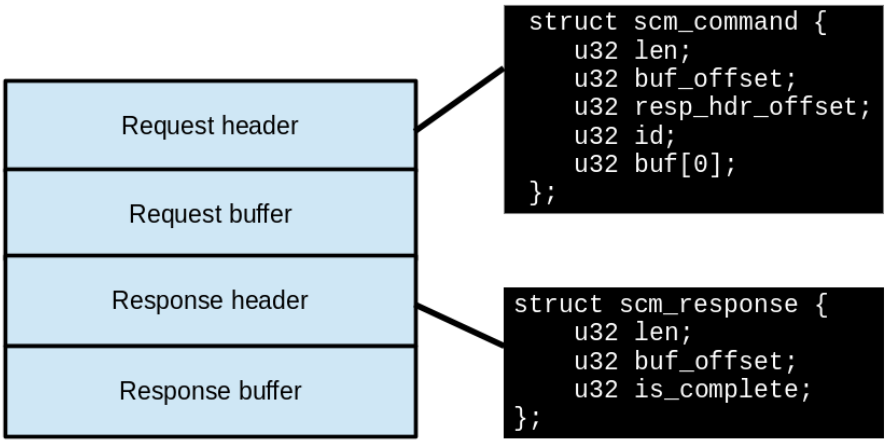

Figure 1: QSEE command buffer.

5. `req.resp_hdr_offset` <= `req.len` - 16 (*"Does the entire response header reside inside the command buffer?"*)

After sanity-checking the command structure, QSEE identifies and invokes the desired SMC handler (if it exists). On completion, if the input flags for the request indicate output is required, QSEE populates the output structure as follows:

```
rsp.len = 12;
rsp.buf_offset = 12;
rsp.is_complete = 1;
```

In this case, the address of the response header `rsp` is calculated as `req + req.resp_hdr_offset`.

The `qsee_is_ns_memory()` function (this is not the real name of this function, since source code is not available) is designed to check whether a range of memory has been marked as "secure" and is accessible only to TrustZone, or if it is "non-secure" and should be accessible to the Linux kernel. This function has a deficiency where it fails to handle integer overflows properly, and in fact explicitly allows for them by reversing the order of the arguments when one is unexpectedly greater than another. The following is approximate pseudocode of the involved functions:

```
int qsee_is_ns_memory(long addr, long len)
{
    return qsee_not_in_region(&region_list, addr, addr+len);
}
```

```
int qsee_not_in_region(void *list, long start, long end)
{

    if (end < start) {
        tmp = start;
        start = end;
        end = tmp;
    }

    // Perform validation
    ...
}
```

Note that if `qsee_not_in_region()` is invoked with a `start` address greater than the `end` address, this function will reverse the order of these arguments. As a result, the following request header will cause QSEE to write the three words of the populated response header, `0x0000000c 0x0000000c 0x00000001`, to arbitrary secure memory:

```
req.len = 0xfffff000
req.buf_offset = 0xffffe000
req.resp_hdr_offset = target - req
```

Glancing back at the validation, each sanity-checking condition is satisfied:

1. `0xfffff000` $>$ 16
2. `0xffffe000` $<$ `0xfffff000`
3. `0xffffe000` $>=$ 16
4. `qsee_is_ns_memory(req, 0xfffff000)` $==$ `true`
5. `target` - `req` $<$ `0xfffff000` - 16

In particular, (4) is satisfied because an integer overflow will occur when adding the address of the command buffer (`req`) and its supposed length (`0xfffff000`) in the invocation of `qsee_not_in_region()`, but this function will then reverse the order of the arguments such that the entire region being validated lies in non-secure memory and passes the check.

The ability to write these three words to arbitrary secure memory can be leveraged to execute arbitrary code in the context of QSEE.

7 Impact

The ability to execute arbitrary code in the context of QSEE results in the complete compromise of any applications leveraging TrustZone for security guarantees. In particular, this vulnerability may be used to compromise DRM

schemes, leak sensitive key materials, defeat operating system protection mechanisms, and in some cases (e.g. on some Motorola and HTC devices) manipulate software-programmable fuses to defeat secure boot.

References

[1] ARM Ltd. TEE reference documentation. http://www.arm.com/products/processors/technologies/trustzone/tee-reference-documentation.php, 2014.

[2] ARM Ltd. Trustzone. http://www.arm.com/products/processors/technologies/trustzone/index.php, 2014.