SECURITY
EXPLORATIONS

Security Vulnerability Notice

SE-2019-01-GEMALTO

[Security vulnerabilities in Java Card, Issues 19 and 33]

SECURITY
EXPLORATIONS

DISCLAIMER

INFORMATION PROVIDED IN THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW NEITHER SECURITY EXPLORATIONS, ITS LICENSORS OR AFFILIATES, NOR THE COPYRIGHT HOLDERS MAKE ANY REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OR THAT THE INFORMATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS, OR OTHER RIGHTS. THERE IS NO WARRANTY BY SECURITY EXPLORATIONS OR BY ANY OTHER PARTY THAT THE INFORMATION CONTAINED IN THE THIS DOCUMENT WILL MEET YOUR REQUIREMENTS OR THAT IT WILL BE ERROR-FREE.

YOU ASSUME ALL RESPONSIBILITY AND RISK FOR THE SELECTION AND USE OF THE INFORMATION TO ACHIEVE YOUR INTENDED RESULTS AND FOR THE INSTALLATION, USE, AND RESULTS OBTAINED FROM IT.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL SECURITY EXPLORATIONS, ITS EMPLOYEES OR LICENSORS OR AFFILIATES BE LIABLE FOR ANY LOST PROFITS, REVENUE, SALES, DATA, OR COSTS OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, PROPERTY DAMAGE, PERSONAL INJURY, INTERRUPTION OF BUSINESS, LOSS OF BUSINESS INFORMATION, OR FOR ANY SPECIAL, DIRECT, INDIRECT, INCIDENTAL, ECONOMIC, COVER, PUNITIVE, SPECIAL, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND WHETHER ARISING UNDER CONTRACT, TORT, NEGLIGENCE, OR OTHER THEORY OF LIABILITY ARISING OUT OF THE USE OF OR INABILITY TO USE THE INFORMATION CONTAINED IN THIS DOCUMENT, EVEN IF SECURITY EXPLORATIONS OR ITS LICENSORS OR AFFILIATES ARE ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS.

Security Explorations discovered two security vulnerabilities in the implementation of Java Card technology [1] used in some Gemalto products. A table below, presents their technical summary:

ISSUE #	TECHNICAL DETAILS	
19	**origin**	Java Card runtime
	cause	unmanaged object references
	impact	compromise of memory safety / arbitrary read, write and native code execution access on a target card
	status	verified
33	**origin**	Java Card runtime
	cause	unprotected / leaking memory references
	impact	compromise of memory safety / arbitrary read and write access on a target card
	status	verified

Vulnerabilities details

Both vulnerabilities are similar in the way that they make it possible to access arbitrary memory regions of a target Gemalto card such as SIM. Issue 19 is caused by no references' management mechanism of a target card (no checking for legitimate reference values). As a result, fake[1] reference values can be used, so that they can point into specially crafted object header data mimicing instances of large arrays. Issue 33 is about the abuse of existing, managed object references and the possibility to leak arbitrary card data with their use. Below, more details are provided with respect to both issues.

Issue 19

Reference implementation of Oracle Java Card technology relies on object references instead of direct pointers. Each object reference needs to be resolved prior to any instance field access conducted by the means of `getfield` / `putfield` group of instruction. If given object reference cannot be resolved (no object was created in the environment with given reference value), the runtime throws an exception.

We have observed that Gemalto implementation of Java Card VM makes use of object references, which almost directly correspond to real pointers. In the environment of GemXplore3G card, 16-bit Java reference values are used. Their format is illustrated on Fig. 1.

[1] not allocated by card OS or Java Card runtime environment.

Fig. 1 Java reference format (GemXplore3G case).

Chunk id identifies a 64KB chunk of memory (ROM or EEPROM). Chunk values 0, 2, 4 and 6 correspond to EEPROM / Flash memory.

Memory visible to the Java Card runtime is addressable with the granularity of 8 bytes. These bytes can be used as a header identifying the object (its type and size in particular). As such, every object allocated by the card is preceded by a header.

The problem with this particular approach is that one can use a custom reference value pointing into the middle of a given specially crafted object (A) so that, it could be used as a completely legitimate reference (B). Additionally, if header preceding the custom reference value is appropriately constructed, one can come up with a Java array object of overlong size that could be used to access card memory beyond the size of object A. This is illustrated on Fig. 2.

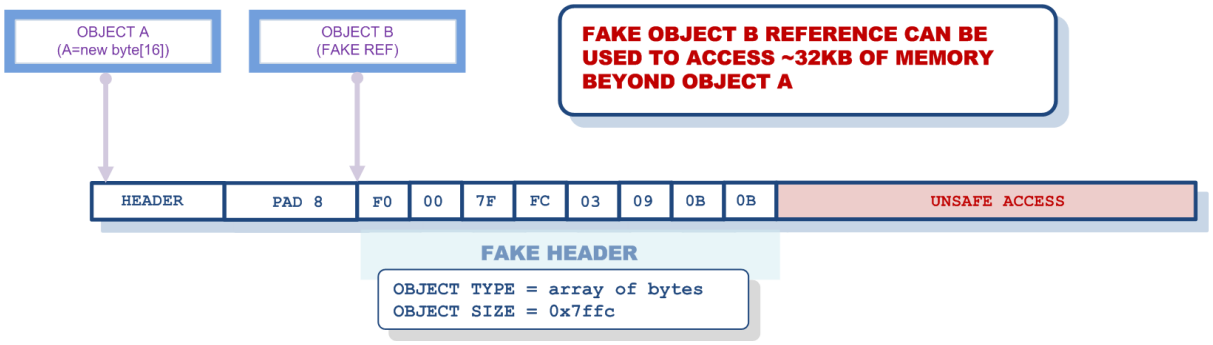

Fig. 2 Illustration of the vulnerability (exploit setup condition).

We verified that the above attack scenario in the environment of GemXplore3G card.

As for the construction of a custom reference value, this can be achieved by the means of simple type casts operations (casts between reference and integer values).

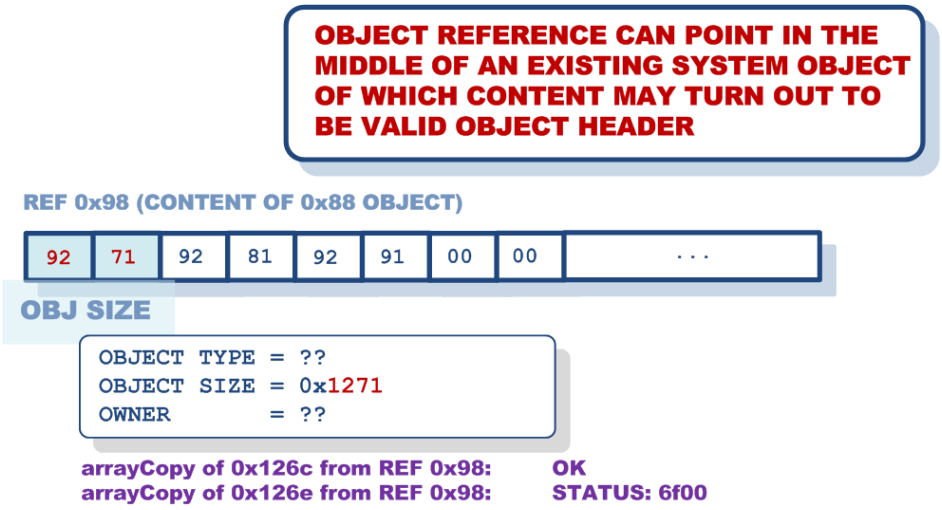

Fig. 3 An abuse of a reference value pointing into system memory content for arbitray memory access (GemXplore 3G case).

Finally, unmanaged object references can be also exploited in a more direct way (Fig. 3). Their values occasionally turn out to be valid Java objects (ordinary objects or arrays). In such a case, arbitrary memory regions can be read or written by scanning the whole range of possible Java reference values (13 bits offsets and all visible chunks) and treating them as an array of bytes (as an argument to `arrayCopyNonAtomic` call or `baload` / `bastore` instructions [2]).

Issue 33

Gemalto USimera Prime SIM card making use of managed object references can be also successfully used to gain access to arbitrary card memory regions.

Managed Java references in use by USimera Prime card have a format as illustrated on Fig. 4.

Fig. 4 Format of a managed Java reference (USimera Prime case).

Managed object references differ from the unmanaged ones in that they are virtual IDs instead of real pointers. For instance, consecutive object instance allocations return the following references values in the environment of USimera Prime card:

```
7c05
7c06
7c07
7c08
...
```

For object content access, Java Card VM conducts translation of such reference values to real memory pointers with the help of translation tables associated with every Java application as indicated below:

```
shell> applist
...
[4]
- addr     fb5e85
- aid      a000000030000011000200689
- state    04
- flags    03
- type:    java
- inst     6c04 class 5404
           addr fb5f01

- ptr_tab fb6d6e                    <--- TRANSLATION TABLE ADDR
```

We have observed, that for USimera Prime card there are occasional references (such as `6c01` associated with AID `a000000030000011000200689`) that point to a memory region encompassing installed CAP file, Java heap and pointer translation table among others (Fig. 5).

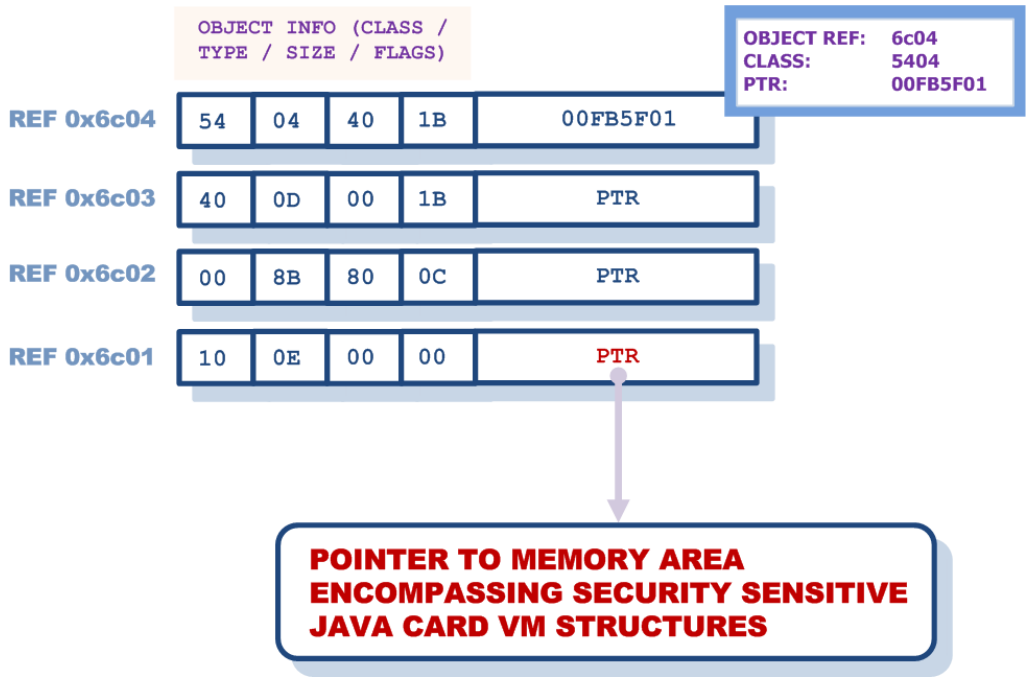

Fig. 5 Sample pointer translation table (USimera Prime case).

As a result, arbitrary card memory region can be successfully read or written through a Java reference by an unprivileged applet instance. All that is required to accomplish that is to treat such a reference value as denoting an array of bytes (use it as an argument to `arrayCopyNonAtomic` call or `baload` / `bastore` instructions).

Vulnerabilities impact

Discovered vulnerabilities make it possible to break memory safety of the underlying Java Card VM. As a result, full access to smartcard memory could be achieved, applet firewall could be broken or native code execution could be gained.

While our exploit code cannot successfully pass off-card verification process, the vulnerability should be still perceived in terms of a significant weak point. It paves the way for an in-depth analysis of vulnerable cards, which can potentially result in a discovery of far more serious issues.

Security Explorations conducted initial reverse engineering of both GemXplore3G and USIMERA cards by exploiting the reported weakness. As a result, significant information about Gemalto SIM cards implementation / their environment and security could be obtained as briefly shown in [3].

Affected cards

Our Proof of Concept code was successfully tested in the environment of several Gemalto SIM cards.

Issue 19 was verified to affect GemXplore 3G card:

- *GemXplore 3G V3.0-256K*
  ATR 3b9f95801fc78031e073fe211b63e208a8830f900089

Issue 33 was verified in the environment of the following card:

- *3G USIMERA Prime*
  ATR 3b9e96801fc78031e073fe211b66d0017a7b0e000e

REFERENCES

[1] JAVA CARD TECHNOLOGY
https://www.oracle.com/technetwork/java/embedded/javacard/overview/index.html

[2] JAVA CARD CLASSIC PLATFORM SPECIFICATION 3.0.5
https://www.oracle.com/technetwork/java/embedded/javacard/downloads/index.html

[3] Reverse engineering Java SIM card
http://www.security-explorations.com/materials/javasim-reversing.pdf

---

About Security Explorations

Security Explorations (http://www.security-explorations.com) is a security company from Poland, providing various services in the area of security and vulnerability research. The company came to life as a result of a true passion of its founder for breaking security of things and analyzing software for security defects. Adam Gowdiak is the

company's founder and its CEO. Adam is an experienced Java Virtual Machine hacker, with over 100 security issues uncovered in the Java technology over the recent years. He is also the Argus Hacking Contest co-winner and the man who has put Microsoft Windows to its knees (the original discoverer of MS03-026 / MS Blaster worm bug). He was also the first expert to present a successful and widespread attack against mobile Java platform in 2004.