Security Vulnerability Notice

SE-2019-01-ORACLE-2

[Security vulnerabilities in Java Card, Issues 20-25]

DISCLAIMER

Security Explorations discovered additional security vulnerabilities in Java Card [1] technology used in financial, government, transportation and telecommunication sectors among others. A table below, presents their technical summary:

ISSUE #	TECHNICAL DETAILS	
20	origin	`javacard.framework.Util` class
	cause	insufficient type check in `arrayCompare` method implementation (*src* argument)
	impact	compromise of memory safety / arbitrary read access of card memory (via oracle)
	status	verified
21	origin	`javacard.framework.Util` class
	cause	insufficient type check in `arrayCompare` method implementation (*dst* argument)
	impact	compromise of memory safety / arbitrary read access of card memory (via oracle)
	status	verified
22	origin	`javacard.framework.Util` class
	cause	insufficient type check in `arrayFill` method implementation
	impact	compromise of memory safety / arbitrary write access to card memory
	status	verified
23	origin	`javacard.framework.Util` class
	cause	insufficient type check in `arrayFillNonAtomic` method implementation
	impact	compromise of memory safety / arbitrary write access to card memory
	status	verified
24	origin	`javacard.framework.Util` class
	cause	insufficient type check in `setShort` method implementation
	impact	compromise of memory safety / arbitrary write access to card memory
	status	verified
25	origin	`javacardx.framework.util.intx.JCint` class
	cause	insufficient type check in `setInt` method implementation
	impact	compromise of memory safety / arbitrary write access to card memory
	status	verified

Issues 20-25 were successfully verified in the environment of the most recent Oracle Java Card 3.1 SDK from Jan 2019 incorporating reference implementation of Java Card VM [2].

All of newly reported issues are due to the missing type check when handling array arguments. If a specially crafted ordinary object is provided instead of an array of bytes, Java Card VM can be tricked to treat it is an array of a very large size. This type confusion condition is explained in a more detail in our previous report (SE-2019-01-ORACLE, Issues 1-2).

In this report, instead of a direct array copying functionality, other array operations are signaled. These are the following:

- Issues 20 and 21 make use of array comparison methods. They exploit the possibility to compare an array with a known content with an array of a completely unknown content. An unknown and malicious array of bytes A (spoofed by an ordinary object instance) can be scanned and for each of its elements a comparison operation can be conducted with an array of bytes B containing 1 element only. If the comparison operation is done for the length of 1 and the whole range of values of the only element of array B, one of them will trigger a match with a byte of an unknown array at given offset. As a result, the content of array A can be discovered (card memory can be read).

- Issues 22 and 23 exploit the possibility to use array filling methods as a gadget capable to store a given 1 byte value at target array offset (1 byte store operation),
- Issues 24 and 25 rely on the possibility to directly write short and int values into an array of bytes.

Table below provides more details with respect to APDU commands implemented by our Proof of Concept code illustrating the above issues.

POC	INS	TYPE	DESCRIPTION
arrayops	0x10	READ_MEM	Read memory by the means of an oracle available through an array comparison operation *REQ APDU:* 00-01: offset to start reading data 02: length of data to read 03: type `(unused)` *RESP APDU:* 00-len: bytes of data read (discovered) from a table of bytes starting from given offset
	0x11	WRITE_MEM	Write memory by the means of various array operations *REQ APDU:* 00-01: offset to start writing data 02: length of data to read 03: type 00 - use `arrayFill` method as a 1 byte store gadget 01 - use `arrayFillNonAtomic` method as a 1 byte store gadget 02 - use `setShort` method 03 - use `setInt` method 04-len: data bytes to write *RESP APDU:* 00-len: bytes of data written to a table of bytes starting from given offset

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/`

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.