TDDE21 - Project end report

Carl Folkesson, Gunnar Grimsdal, Emil Gustafsson,
Daniel Månsson, Patrik Lundgren, Hampus Viken

December 2018

1 Introduction

In this report we will present our results from 2018's project. We will also give some advice to next years students, that hopefully will help them in getting started easier than we did.

2 Results

In this section we will present what we have successfully implemented in HIP version 2.

2.1 Variable puzzle and solution length

In HIP version 1 the length of the puzzle should always be 12 bytes. This was changed in HIP version 2 to be of variable length. This variable length is dependent on the length of the Responders hash function, RHASH_len, in bits, as the total length of the puzzle in HIP version 2 should be 4 + RHASH_len/8 bytes. This change comes from the change of the random #I parameter from the length 8 bytes to RHASH_len/8 bytes.

As a result of this, the length of the solution to the puzzle was also changed in HIP version 2. The new length becomes 4 + RHASH_len/4 bytes instead of the old 20 bytes. This change comes from the change of the random #I parameter in the same way as in the puzzle and the change of the puzzle solution parameter #J from the length 8 bytes to RHASH_len/8 bytes.

This change was implemented during the project. It was also discovered that in the RFC 7401, that describes HIP version 2, the length of the puzzle and solution parameters are inconsistent. On page 47 and 48, where the parameters are described in detail, the length of the puzzle and the solution is as described above. However, on page 41, where all the different HIP parameters are listed, the length of the puzzle and the solution is the same as it was in HIP version 1. This is considered an error in the RFC and should probably be changed.

2.2 HIT Suite

In version 2 of HIP a way to choose HMAC and HASH algorithm was introduced. One of our challenges has been to implement this further. Since HMAC-SHA-256 and SHA-256 respectively was standard in HIP version 1 there was many function and length-constants that had to be changed. With the new HIT suites came new signature algorithm families to be supported. We started by changing the hitgen.c file so that the hipgen executable could create configuration-files for elliptic curve digital signature algorithm (ECDSA) as well as change which suite ID to write in the configuration-file. After getting ECDSA, with 256-bit key length working, we started implementing ECDSA with 384-bit key. However, since HIP had not yet been executed with a key and hash algorithms bigger then 256-bits some memory problem occurred. These problems should now be solved. Another problem we encountered during implementation of suite ID was the suite ID parameter not being present in functions that now needed to know the ID, we therefore needed to forward the ID parameter to all functions using the HMAC or HASH algorithm. Lastly we changed the HIT suite list to not be static and instead read which suite ID:s we have in our configuration file.

All the changes are for suite ID are located in the *v2_ecdsa* [1] branch. One thing to think about while looking at the suite ID:s are that there exists 8-bit and 4-bit representations of the ID:s. In the code we use the 4-bit representation, even if the variable is stored in a 8-bit char (the 4 most significant bit are all 0). However, when sending the ID:s the 8-bit representation is used (the 4 least significant bit are all 0).

2.3 OpenSSL 1.1.0

In the beginning of the project we thought that it was odd that OpenHIP was based on an old version of OpenSSL (v1.0.2g), which is only supported until the end of 2019. Therefore, we decided that we would try to upgrade it to version 1.1.0. This showed to be a bigger task than we had expected and in addition to that, a new version of OpenSSL (v1.1.1)[2] was released during the course. This version is decided to be the next long-term-support which is going to be supported to at least 2023 and it contains some important features such as support for TLS v1.3 and SHA-3. We would therefore recommend the next years students to upgrade to this version instead, not knowing exactly how much our work on branch *SSL-1.1.0* will help. According to the OpenSSL blog[3] "most applications that work with 1.1.0 can gain many of the benefits of TLSv1.3 simply by dropping in the new OpenSSL version". We have not found a document that describes any breaking changes that OpenSSL v1.1.1 brings with it when upgrading from v1.1.0, but the changes from v1.0.2 to v1.1.0 is clearly described on the wiki[4].

[1] `https://bitbucket.org/openhip/openhip/branch/v2_ecdsa`
[2] `https://en.wikipedia.org/wiki/OpenSSL`
[3] `https://www.openssl.org/blog/blog/2018/09/11/release111/`
[4] `https://wiki.openssl.org/index.php/OpenSSL_1.1.0_Changes`

3 Future work

This section discusses the changes that still have to be made to make fulfill the requirements for HIP 2.0.

3.1 I2 receival and keymat generation

While the keymat generation has been updated so that it uses the RHASH value from the HIT-suite to decide the hash function, this has to be verified for edge cases and invalid values as well. It currently falls back to use SHA256 as a hash function in the case of an invalid HIT-suite. This functionality has to be verified that it really is a wanted behaviour.

The keymat generation is also still using the old method for generating keying material. This should be updated to be an negotiable part of the protocol, where the function used for keymat generation should be chosen based on the DH GROUP ID. For the moment the only valid function for keymat generation, HKDF, is not implemented in the code, because it is dependent on OpenSSL 1.1.0.

3.2 HIT suite negotiation

There is still need for proper HIT suite negotiation. In the current implementation, the HIT suite is picked from the reponders list of HIT suites, where it is the first suite of the list.

The first problem is when the initiator should verify the responders HIT when parsing the R1 package, the initiator dose not know the suite at this point. The next problem occur when parsing the I2 package, the responder must somehow know which suite the initiator chose after parsing the HIT_SUITE_LIST.

Based on our understanding of the problem and the RFC, we have some suggested solutions. **We do not promise that they will work in practice.** The solutions are as follows:

1. Using the RSA or DSA with SHA256 for the packets I1 and R1. After the HIT_SUITE_LIST has been parsed, pick the first suite in the list that the initiator supports.

2. When parsing the I2 packet, the responder could try to match the received HIT to one in the known_host_identities. This way, it would be possible to decide which HIT_SUITE that the initiator used to contact the responder (which is required to decrypt the I2 packet and verify the HIT). The responder can then drop the packet if there is no matching HIT found (meaning that a HIT_SUITE which is not supported was used). This approach would require the different HITs for the responder to be loaded into memory, together with the corresponding HIT_SUITE that it uses – otherwise there would be a possibility for DDOS attacks.

3. An alternative solution could be that the initiator tests all of its supported HIT suites in parsing of R1 when validating the tag. In the same way, the responder would have to do the same when parsing I2.

3.3 OpenSSL 1.1.0 or OpenSSL 1.1.1

As previously mentioned, modifying HIP to support OpenSSL 1.1.0 was not completed. In addition, in order to implement the correct KEYMAT generation for HIP version 2, the key derviation function HKDF needs to be supported. This is not the case with OpenSSL 1.0.2g, but is supported by OpenSSL 1.1.0 and above. Therefore, support for OpenSSL 1.1.0 and prefereably OpenSSL 1.1.1 as well, should be implemented in HIP and be seen as a priority.

The main changes from OpenSSL 1.0.2 to version 1.1.0 are how data is accessed through the API. Instead of accessing and modifying objects and their members directly, it is done through new getter and setter functions. Therefore, alot of the work to implement support for the newer OpenSSL version is to go through the code base and simply change how SSL objects and data is modified and accessed. It is not very hard work, but takes time since the code base is quite large. Make sure to read the wiki[5] on how these changes should be done.

3.4 ICMP response on version mismatch

When a HIP packet using a different HIP version than the one running on the Responder is received, the Responder should respond with a ICMP packet with type Parameter Problem, with the Pointer pointing to the Version/RES byte in the HIP header. This is not implemented, but a check for a different HIP version is done in the code(file hip_input.c search for HIP_PROTO_VER). If a different HIP version is detected, a NOTIFY packet is sent instead of a ICMP packet. This is incorrect behaviour.

4 How to get started

This section describes how to get started with developing for OpenHIP by discussing system requirements and how to setup various tools for testing and the OpenHIP itself.

4.1 Installing CORE

Gurtov has a nice guide how to install CORE on Ubuntu. Ether follow it or you could install the core-daemon[6] and core-gui[7].

[5] https://wiki.openssl.org/index.php/OpenSSL_1.1.0_Changes

[6] https://downloads.pf.itd.nrl.navy.mil/core/packages/4.8/core-daemon_4.8-0ubuntu1_precise_amd64.deb

[7] https://downloads.pf.itd.nrl.navy.mil/core/packages/4.8/core-gui_4.8-0ubuntu1_precise_all.deb

4.2 Ubuntu version

When we started, this project could not be compiled or executed with OpenSSL 1.1.0, this meant that for example Ubuntu 18.04 (or any other updated OS) would not work. Therefore, during our development, we used Ubuntu 16.04 in a virtual environment.

4.3 Configure files

We have created multiple config files for different suites and algorithms used. We have sent the config-files to Gurtov, but they are also uploaded to Pastebin (DSA-256 [8], RSA-256 [9], ECDSA-384 [10], DSA-256 & ECDSA-384 [11]) You can also create your own config-file by following Gurtovs guides.

4.4 The OpenHIP repository

In the BitBucket repository the branch we have developed from is the hipv2 branch. This is the branch you probably should proceed implementing from. The update to OpenSSL 1.1.0 has been worked on in the SSL-1.1.0 branch.

4.5 Pitfalls of doom

1. Make sure the configuration file is correct.
2. Check the Type and Length parameters in **wireshark**, if any of them are incorrect it may result in fairly unexpected program flow.
3. When modifying packets, make sure you have space in the send buffer for your change.
4. Don't blindly trust comments, they are quite frequently outdated.

4.6 Generate additional documentation

When you first starts to work with the code, it may be hard to understand how the different parts of the code is related. The documentation linked from the repository is not that helpful, because it only shows which files and functions that exists, not how they are related. Fortunately, more documentation can be generated. This is done using Doxygen, which is a tool that generates documentation for source code. To install Doxygen on ubuntu run:

```
sudo apt-get install doxygen
```

[8] https://pastebin.com/pbfK5pVm
[9] https://pastebin.com/NAfhb1gz
[10] https://pastebin.com/mkmcchQh
[11] https://pastebin.com/WehkcWGP

A Doxygen config file already exists in the OpenHIP repository in the folder docs, however, this config file seems to be wrong or very minimal. Therefore, to get better documentation you either need to change in this config file or use the doxygen gui(called doxywizard) to generate new documentation. If you decide to use the gui you can install it on ubuntu using:

```
sudo apt-get install doxygen-gui
```

In the GUI you can choose which source files to generate documentation about and a lot of settings for the generation. Some settings that can be useful is CALL_GRAPH and CALLER_GRAPH which generates graphs showing how different functions interacts. You can also chose the format of the documentation, where HTML or LaTeXprobably is the better options. It should, however, be noted that there has been some problems with displaying the generated HTML files on firefox in ubuntu, so there may be an good idea to open the documents in another browser.

4.7 In the distance

When HIP version 2 is fully implemented the extensions should be implemented. These extensions include mobility (RFC8046), multihoming (RFC8047), certificates (RFC8002), registration (RFC8003), rendezvous (RFC8004), and DNS (RFC8005). Lastly, please document what you have done and what problems you have found. The students that did the course the year before us did not do this and a document like this would have helped us a lot :)