GSoC Proposal for syslog-ng: Crypto Parser

About Me

I am Ujjawal Sharma, currently a student at the Johns Hopkins University, Baltimore, MD, USA. I am pursuing my masters in Information Security and would be completing my education in December 2018. Before beginning my education at JHU, I had worked for 3 years as a Security Consultant with Deloitte. Apart from this, I also have an experience of close to 1.5 years working as an e-Commerce developer with Sapient Corporation. Mostly, my Development experience of 4 years has been in Java and Python, but I have a good knowledge and understanding of C as well. (I have a professional training of C and Data Structures from Hewlett Packard).

LinkedIn: https://www.linkedin.com/in/ujjawal-sharma-87361b37/
GitHub: https://github.com/uzzzval
Facebook: https://www.facebook.com/sharma.ujjawal

syslog-ng

As per the documentation of syslog-ng and the problem statement for this project, I have understood that the requirement is to implement an encryption scheme inside the already existing framework of syslog-ng, which can accommodate the encryption and decryption of whole/part of the log message. Currently, only hashing has been implemented, and as hashing is irreversible (as we mostly compress the data to fit into the certain fixed size of hash), it is impossible to figure out the actual content behind the hash, once hashing is performed on the data.

Why the project will be useful?

Like any other application available on the internet (paid or open-source), the one thing, which drives their performance, is their ability to withstand adverse situations. No application will be used in practice if it is not secure and dependable enough to perform certain functionalities without being vulnerable. The aim of this project will be useful for the organizations, which will deploy this tool or include it as part of their main development suite for solving the purpose of logging. This project will make them more secure in terms of transferring the logs over the network in a secure manner. Logs are the very important component of any web application, as they solve the purpose of auditing. The more secure the transfer of logs is, the more the organization will be confident of its handling. There are following few test cases which I believe will be covered under this implementation:

1. There can be an organization which would be dealing with storing a huge number of files, for them transferring path in requests will be common, but for the web application attackers(hackers), it is a very crucial info which they can obtain to hack into the system. But, there can be a situation when files could not get stored may be because there was not enough space on disk and the path of destination got hashed in the request/log. Now, to maintain the consistency of the storage process, the organization will love to store it at the correct location, but they cannot because they don't have the original path and the hashed key cannot be reversed.

2. There can be a scenario when the logs have to show the key-value pair of the attributes being passed in the request but they being in a clear-text format can be dangerous. However, if suppose the request did not go properly, and the support team needs to fix the issue manually, they will require a method to decrypt the cryptic text to get the actual value. In short, the purpose of the decryption here will be to help the developers/supporters of the application in debugging.

3. Under many situations, error logs expose a lot more than the logs of the system under proper running conditions. But, an error log does not have any fixed form. Hashing them will make it completely useless forever. Error logs are the primary useful thing which developers love to see in case of any critical failure of the system.

4. There can also be a scenario, where the name of the file which we are interacting with during a particular request (viewing/editing) needs to be encrypted when it is getting saved on the disk (to ensure it becomes non-reachable to the attackers), but when user wants to fetch it, it should get fetched with its proper name. Here, first encryption and the decryption of the cryptic name of the file have to be performed and the transaction info may get saved in the logs. If the transaction info remains in a clear text format, it would be susceptible to an attack. Encrypting the info and on a requirement, decrypting it will be a useful add-on.

Being in the field of Information Security for some time, I know how much important logs are and how much secure it becomes when a proper encryption-decryption mechanism is added to it. I have implemented encryption for passwords and payment information in my professional career, but to implement it on such a large scale that should fit all scenarios any log can observe, keeping in mind how code changes can occur in future so that frequent modifications in the implementation of encryption should not cause any threat to the overall framework will definitely be an interesting work for me. Implementing the encryption without affecting the efficiency of the system will be more interesting and challenging for me.

Goal of the project?

The goal of the project will be to come up with an encryption scheme, which is suitable to accommodate the sending requests and receiving a response of various different kinds of systems such as HTTP, HDFS, Kafka, and Graphite etc. As of now, there are two methods available in the syslog-ng OSE for the manipulation of messages - rewriting and replacement, both require some or the other form of configuration and they utilize the rewrite rules for this purpose. We also need to keep the management of keys for encryption and decryption in such a way, so that the configuration files and key management system are in sync. For reaching these goal, we will have to compare each and every available encryption scheme (though super challenging, we can make one of our own!) keeping in mind our requirement. For example,

1. If we are mostly dealing with strings of very short length, to make it a larger string, many a times salting is applied in addition to the encryption scheme, but it makes the end of the cryptic text predictable which is not good.
2. For shorter length string, a recommended method is to use AES encryption scheme in CTR mode where it uses the key and a counter to generate a byte stream, which is XOR'd with the bytes of the string. It would leave our encrypted string at the same length as the input plaintext string.
3. For a medium length string such as 256, 512 or 1024 encryption schemes like Twofish, RSA and Serpent are considered as good options. Although there has been a successful partial recovery of their bits, they are still widely used and trusted.
4. For a larger length string, RSA is definitely the unbreakable encryption scheme to date. However, it is also true that they are considered relatively slower than other encryption schemes.
5. We will also need to find out what to select, a symmetric key encryption scheme or an asymmetric key encryption scheme.
6. Also, there are web applications employing different encryption schemes for different modules of the project, thus, having a hybrid environment of encryption. We also need to identify what configuration of key management will best suit the future changes in encryption schemes.

I will consider an implementation a successful one, when it is capable of handling all kinds of variations in the request, coming from different applications with which syslog-ng is capable of integrating. We can start off with the requests of one kind of application at a time and can cover all of them one after the other. The project completion will largely depend on how quickly we are able to prepare the metrics of the requirement, which the encryption scheme should fulfill at any given time.

Knowledge Areas:

I believe for the success of this project, following things will be very much important:

- C (Comfortable), Python (Proficient)
- Bison and Flex (New for me)
- Overall working knowledge of syslog-ng (Already started understanding)
- Knowledge of different cryptographic techniques. (Comfortable)
- The ways how the messages are getting manipulated currently (rewrite and template)(Already started understanding)
- Writing Efficient Algorithms (Proficient)
- Legacy BSD syslog, JSON etc. (I know JSON have used it a lot)

Time Schedule

April 23 - 14th May (Community Bonding and Introduction with the Project)

- Meeting with the team and knowing each other.
- Understanding the requirements in detail.
- Understanding the architecture of syslog-ng and coding practices.
- Setting up the environment for development.
- Analyzing the module to be changed and the possible modifications.
- A debate on which kind of encryption will be fruitful.

15th May to 31th May (First Phase of Work)

- A basic code development of the encryption scheme selected.
- Testing the algorithm with possible known test cases.
- Observing the success/failure of the algorithm for our requirement.
- Analyzing the completed work and observing for any roadblocks.

1st June to 15th June (Second Phase of Work)

- Analyzing the first phase of work and the effect of inclusion of new code.
- Identifying the modules, which should undergo changes to fit in the encryption scheme.
- Changes developed.
- Observing performance of the algorithm after including it in the main framework for a scenario.
- Unit Testing of the modules developed, bug tracking if any.
- First Complete review of developed/modified modules.

15th June to 20th June (Analyzing the work till now)

- Discussing the work till now.
- Understanding any new proposals/changes.
- First phase of Documentation (Draft work).
- Observing the metrics of bugs from the unit testing and planning for next phases of development.
- Time to thoroughly check the framework and identifying the remaining cases.
- Observing the performance effect of the changes.

20th June to 5th July (Third Phase of work)

- Designing the configuration changes to accommodate other scenarios.
- Preparing list of what all has to be changed for different scenarios.
- Fixing bugs of previous phase and Development of this phase to include all scenarios one at a time.
- Unit Testing of the modules already modified, bug tracking and carrying the remaining modules for next phase.

5th July to 20th July (Final Phase of work)

- Bug Fixing of the previous phase.
- Development of remaining modules.
- Unit Testing of the overall work.
- Observing the performance effects of the changes.
- Modifying the draft.

20th July to 3rd August (Last Modifications)

- Thorough testing of the modifications.
- Verifying the effects of changes on other modules.
- Doing final changes, if required.
- Doing thorough testing of the developed work.
- Repeating the above 2 steps until bugs become zero.
- Final commit of developed work.

4th August - 6th August (Wrapping Up)

- Preparing Final Document.
- Preparing Presentation.
- Presentation
- Last meet with the team!