Your Name: Sheng-Feng Lu

Slack Username: Sheng Feng

Email Address: cindejze412@gmail.com

Primary Spoken Language: Chinese

Top project choice (can be one of our project ideas or your own)

Quark Engine: The Storyteller of Android Malware

Are you willing and able to work on other projects instead?

Yes, I am willing to work on other projects if necessary.

Please describe your preferred coding languages and experience.

My preferred coding language is Python. I wrote Python for my independent study in college before, which mainly reproduces a telephony MITM attack from a paper. The attack targets smartphone calls via the net. It intercepts the encrypted packets and performs a decision tree to parse them. Then the program can significantly influence the quality of the call. I trained the decision tree and wrote the interception all by myself.

Please describe any Windows, Unix or Mac OS X development experience relevant to your chosen project. If your project does not require OS-specific expertise, feel free to leave it empty.

Please describe any previous usage with Honeynet Project tools or honeypots in general.

I used Quark-Engine to analyze suspicious android applications on my smartphone. I found that it is handy and easy to know what executables will do.

Please describe any previous Honeynet Project or honeypot related development experience, including details of any patches, code or ideas you may have previously submitted.

I submitted two patches to Quark-Engine and helped Quark-Engine integrate with Ghidra.

The first patch fixes the miscalculation in the scoring system (issue #118). The scoring system gives a wrong upper bound when the user provides a threshold. I

dived in and proposed some functions with problems to the team. With assistance from them, I submitted a pull request and fixed it.

The second one relates to the duplicated descriptions in the summary output (issue #119). I read through the reporting module source code and found a tiny bug in a section. Then, I fixed it by myself.

Finally, I developed a Ghidra plugin for Quark-Engine, which makes Quark-Engine work nicely in Ghidra. Also, I implemented a mechanism that allows users to quickly navigate to the corresponding source code when they click on malicious behavior descriptions. Now, the plugin is available on the Quark-Engine Github page.

Please describe any previous open source development experience, including projects you have worked on

I have no other open-source development experience in the past.

What school do you attend and what is your specialty/major at the school?

Department of computer science and engineering in National Sun Yat-sen University.

How many years have you attended there?

Three and a half years.

What city/country will you be spending this summer in?

Kaohsiung, Taiwan.

How much time do you expect to have for this project?

About 20 hours per week for this project.

Please list all jobs, summer classes, vacations and/or other commitments that you'll need to work around.

There is no other plan on which I need to work during the summer.

Have you participated in any previous Summer of Code projects? If so please describe your projects and experience, including what you liked or didn't like about the experience

No

Have you applied for (or intend to apply for) any other Google Summer of Code 2018 projects? If so, which ones?

No

If you have a URL for your resume/CV, please list it here.

If you wish to list any personal/blog URLs, do so here.

Please describe your proposed project in detail, including deliverables and expected timeline with milestones (this is the long answer, so spend most time here!).

In this proposal, I mainly focus on two dimensions provided by Quark-Engine, including resilience and performance. After discussing with the mentor, we need to solve some critical issues below.

1. The core library, Androguard, used by the Quark-Engine is no longer maintained(see issue #833 from Androguard).

2. After profiling, the most time-consuming sections in the analysis process are the usage of Androguard.

3. Androguard takes up too much memory while analyzing.

The mentor suggested replacing Androguard with Radare2, a great tool that can solve the problems above. Therefore, I list several goals below according to these dimensions.

Resilience

For maintainability, it is necessary to deprecate Androguard from the core library for Quark-Engine. Radare2 is an alternative tool recommended by the mentor. It is a well-known open-source reverse engineering tool and provides many analysis assists on code. Not only can it replace the functions of Androguard, but it also comes with a strong community. Thus, I have been working on this for a while. The proposal aims at continuously improving with the following goals.

1. Improve the Quark-Engine final stage analysis.
2. Port the behavior classification and expand it.
3. Port the call graph and expand it.
4. Write all the tests and documents related to the above modules.

Performance

According to the issues on Github (#82, #130), Quark-Engine takes large consumption of resources when analyzing a big file (size over 100MB) due to two reasons.

- The core library, Androguard, costs the memory and CPU resources a lot.
- Quark-Engine performs the analysis process sequentially.

We have found that replacing Radare2 can solve the first issue. And parallel computing is a possible solution to the second one. Quark-Engine currently uses a single thread to perform all works. However, we can speed up most of them in parallel, so my improvement strategy contains two goals.

1. Optimize the five-stage analysis algorithm.
2. Introduce the concept of parallel computing into Quark-Engine.

Benefits to the Community

For the Quark-Engine project

Radare2 has a promising community to support the entire project. Deploying Radare2 as a core library will improve the health of Quark-Engine. Also, the work-in-process replacement has shown that the analysis speed and memory consumption is greatly improved.

For the Honeynet project

The prosperity of Quark-Engine not only benefits its community but also benefits other projects under Honeynet. For example, IntelOwl, an OSINT tool, has used Quark-Engine as its analyzers. As Quark-Engine gets more prosperous, the IntelOwl project becomes more reliable. Also, a healthy project can contribute to the potential projects in Honeynet, inspiring more great tools in the future.

For the open-source community

For the open-source community, Quark-Engine integrates with many great projects, such as Jadx, Ghidra, Bazaar, IntelOwl, APKLab, Kali Linux, BlackArch. Contributions to Quark-Engine are also helpful for the upstream or downstream tools, which make the threat intelligence community stable and reliable. In general, it inspires more unique ideas and innovations to the whole open-source community.

Current Status of the Project

The core library replacement of Quark-Engine is open on Github and has been developed for a while. For now, it can perform several features listed below.

- The five-stage analysis theory.
- The summary report and JSON report.

Other useful features, such as call graph generation, behavior classification, and support for complex APK analysis, are not implemented yet. However, the replacement has proved that the solution works and can analyze malware 15% faster than the original Quark-Engine.

Implementation Plan

Replacement of core library

To avoid any unexpected error on Quark-Engine, I will enrich all the tests first and then continually work on the replacement. Also, I will introduce the listed features during development, which is encouraged by the mentor.

Five-stage analysis algorithm

1. Trace data types.
2. Support more bytecodes.
3. Make Quark's rules support API parameters.

Call graph

1. Support a more comprehensive call graph that draws all functions together.

Performance enhancement

There are two ways to enhance performance.

Algorithm optimization

The strategy mainly focuses on four steps to optimize algorithms.

1. Detecting the bottleneck of the process.
2. Listing the potential solutions that mitigate the impact
3. Evaluate the effectiveness of each method and find the best one.
4. Implement it and verify the result.

Parallel computing

Currently, the highest time consumption often occurs in the particular stage of the analysis process. And the second-highest one happens while loading rules. Therefore, to parallelly speed up the analysis, there are two dimensions to improve.

1. Parallelization of the five-stage algorithm.
2. Parallelization of the file I/O.

Deliverables

1. A more comprehensive call graph.
2. A Radare2-based core library of Quark-Engine.
3. All documents and tests for the mentioned module.
4. A strategy to optimize performance and its implementation.

Timeline

June 12 - June 28

1. Enrich all tests for Quark-Engine.
2. Adjust the behavior classification module.

June 28 - July 13

1. Add those features mentioned above to the five-stage analysis process.
2. Introduce the capability of multi-dex analysis.
3. Support a comprehensive call graph.

July 13 - July 27

1. Propose a strategy for bottleneck detection, mitigation, and evaluate the effectiveness.
2. Verify the optimization result.

July 27 - August 10

1. Continuously verify the analysis result.
2. Introduce the concept of parallel computing.

August 10 - August 24

1. To be in constant touch with the developer in Quark-Engine and adjust the code.
2. Bug fixes, error handling, and code readability improvement.
3. Fulfill the document and tests for all the above components.

Why are you well suited to perform this project and why are you interested in it?

There are two reasons that I am suited for this project.

- I have comprehensive knowledge about Quark-Engine, including the analysis theory and other components. I am ready to code without any investigation.
- I have developed this project for months. Therefore, I think I can surmount the further challenges and accomplish this project.

On the other hand, I want to take this project to learn the implementation of an analysis theory. Also, I want to have a better knowledge of the workflow of an open-source project. Last but not least, I want to improve my collaboration skill by discussing and coding with others.

Have any of our members met you face to face, such as at one of our recent public events (Paris, San Francisco Bay Area, Dubai, Warsaw, Stavanger, San Antonio and Canberra)? If so, please list who/where.

No