IMPE: Intelligence Malware Processing Engine

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Dennis Tian

Spring, 2024

Technical Project Team Members

None

Briana Morrison, Department of Computer Science

IMPE: Intelligence Malware Processing Engine

CS4991 Capstone Report, 2023

Dennis Tian
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
dt7bbu@virginia.edu

ABSTRACT

Security analyst and incident response (IR) teams play a key role in preventing, detecting, and mitigating cybersecurity attacks; however, many of the necessary steps are repetitive and time consuming. The team I worked with for the past two summers was in charge of the development of a web application (IMPE) that enables submission of a piece of suspected malware and returns valuable metadata and behavioral information obtained from various static and dynamic analysis tools. The project was divided into two main teams: those in charge of the web application itself and those overseeing the reverse engineering (RE) tools that the application uses to process samples. It was written using mostly Python 3 and JavaScript and had extended support for API integrators. We implemented IMPE's backend using a combination of Celery, RabbitMQ, and Redis for parallelized worker/queue management, and we wrote the frontend using the React framework. The frontend and API are currently undergoing a major facelift that will allow integration into a larger "Analyst Tools" portal that contains a suite of other tools. The end goal of this portal is to have everything polished and tested to be ready for commercial distribution.

1. INTRODUCTION

In the realm of cybersecurity, speed and efficiency are key, whether it be when responding to a live incident or when researching signatures of newly-found malware. It is relatively obvious why it is critical during incident response: data, money, and perhaps even lives are in danger, so teams need to be able to quickly figure out what kind of malware they are dealing with.

For everyday operations, efficiency is also crucial – new, potent malware is developed every single day. Normally, after obtaining a novel malicious sample, analysts write signatures to help detect other samples in the same family. However, the process of recognizing previously-recorded signatures in new samples is very time consuming and error prone. Here, automation helps tremendously because it allows analysts to focus on new classes of malware rather than having to spend time determining whether something has been seen before.

Additionally, in the case of a previously-unseen malware, analysts need to decipher what it does and how it does it. This process takes significantly more time than signature matching, and automation is effective here as well. For example, extracting metadata often takes unexpectedly long to do manually due to analysts having to delve into the bytes of the file, but a program can be written to do all of it automatically. Moreover, numerous third-party software exist to help deduce what a sample does, and they can be leveraged in an application to further aid analysts.

2. RELATED WORKS

In their paper, Islam, et al. (2010) investigated ways to optimize static analysis by integrating pattern recognition algorithms.

These algorithms replaced the manual process of writing signatures for each new piece of malware, resulting in faster and more accurate detection. This relates to IMPE in that the goal was also to extend and improve from basic pattern identification and matching. However, IMPE took a different route: instead of equipping the static analysis programs with pattern recognition, it incorporated a suite of dynamic analysis sandboxes (allowing the code to actually run, rather than simply looking at the file) for more accurate classification.

Botacin, et al. (2022) developed a real-time malware identification engine based on existing antiviruses. It used hardware-assisted pattern matching to increase efficiency and was able to accurately classify thousands of samples. The application was designed to be both memory and storage efficient to reduce performance overhead. This ties into IMPE, as IMPE also uses signatures from antivirus databases as part of one of its tools. However, the target audience is different – instead of end users, the user base is more geared toward security analysts and incident responders, so there is more functionality beyond simply classification. Additionally, local hardware acceleration is not needed because most of the processing is done remotely on cloud servers.

3. PROJECT DESIGN

IMPE is a web application that centers on the automation of static and dynamic analysis to expedite the malware analysis process. Several user and design requirements to be followed are detailed in the following subsection. The rest of this section focuses on the design and implementation of the project and related projects that we also worked on.

3.1. Requirements

To determine the requirements for the application, we consulted with our end users, the security analysts and incident in adjacent teams. One of the main points brought up was speed. If the application takes too long to process a sample, it defeats the primary goal, as they will simply do it by hand. Furthermore, the system had to be able to scale seamlessly with larger amounts of users, because usage would not be limited to one person at a time. This requirement is especially critical given the end goal of commercial distribution. Additionally, the system would have to be developed in a modular fashion, since some tools would inevitably become deprecated and in need of replacement; new tools also should be able to be developed without affecting the rest of the application. Finally, the results produced would naturally need to be accurate and relevant, so they should be of the same or higher quality and scope of a manual analysis.

3.2. Design & Implementation

The main user base consisted of typical analysts, who mainly use the application's web page (UI) to submit samples, and API integrators, people who interact with the application through the REST API rather than the UI. We designed the web application with this in mind, providing both a frontend and backend interfaces.

The typical sample submission process started in the UI, where a user submits a file, hash, or URL to be analyzed. We wrote this frontend using the React JavaScript framework and integrated it with the company's existing authentication system to support user logins. Users were also able to view a history of their past submissions, so they would not have to go back and find the files to view the generated results again.

After the sample was submitted through the web page form, it was sent to the main IMPE worker through the REST API designed in Node.js; this worker was where we spent most of our time. We needed to design this backend component, which we called the "core," very carefully, since the speed and scalability of the entire application hinged on its performance. I will first discuss the overall pipeline, ignoring scaling solutions.

First, the core sent each submitted sample in its queue to all of the analysis tools, and those tools relayed the results. Next, the core aggregated those results to produce a report easily digested by analysts. Finally, the results and report were sent back to the UI, where the user was notified that their submission had finished processing and the results were formatted in tables.

Clearly, though, sending samples one by one to all of the tools was not conducive to scalability. We used three technologies to help us process multiple submissions at once: RabbitMQ, Redis, and Celery. Using Celery, we created multiple workers, units capable of processing one submission, for each separate analysis tool. Now, instead of sending a submission to the singular worker for each tool, we sent it to the least busy ones. We utilized RabbitMQ and Redis in conjunction to manage the queues for each one of these workers: they handled the logic of determining which worker was the least busy (had the shortest queue) so we could focus on the actual data being sent back and forth.

3.3. Related Projects

To further aid our end users in their work, we developed two related projects that both leverage the capabilities of IMPE.

First, we developed a tool called CronIngest. The basis of this tool is that new, unseen malware is distributed every day, and a critical part of a security analyst's job is to find and deconstruct these new threats. One of the most time-consuming tasks of this process is actually finding these novel samples in the midst of thousands, if not millions, of other files. CronIngest offloads this responsibility from the analysts; it is a program that automatically ingests new samples from VirusTotal's database using their Retrohunt feature. This feature allows CronIngest to query for samples in a time range that match on a provided set of rules, written in YARA. After the samples are obtained, they are then relayed over to IMPE for processing. This information is crucial to analysts because it allows them to keep an eye on new and upcoming vulnerabilities without having to dedicate the time for manual research.

The other project we worked on was a slackbot for IMPE. The primary form of communication at the company is Slack, so having the ability to use IMPE directly from Slack was an oft-requested feature. We were able to integrate the IMPE and Slack APIs to create an app that could respond to slash commands like **`/impe search <hash>`**. This bot aimed to increase the productivity of analysts, as switching back and forth between Slack and a web browser to access IMPE was both tedious and prone to errors when copying information.

4. RESULTS

From its first release, IMPE has gradually become the go-to resource for analysts whenever they need information about a sample that would otherwise take hours or days to compile by hand. It has actually changed what analysts do on a daily basis, since much more time can be dedicated to areas where human examination is required. IMPE has been able to support heavy usage without problems, which has peaked at tens of thousands of submissions per day. It is now a core component of numerous adjacent projects, providing accurate results in a timely fashion that are accessible both by humans and by other applications connecting to it.

Furthermore, CronIngest has helped analysts find a multitude of novel and dangerous malwares, some of which may not have been found at all by hand. These results were crucial, as they provided the necessary information to produce reports about these widely-unknown threats.

5. CONCLUSION

The day-to-day work of a security analysts often involve repetition to some degree, whether it is deconstructing files to find keywords or determining the filetype of a sample. What they are most valuable for, however, is human analysis. Everything mentioned previously is able to be automated, allowing analysts to focus on where their expertise is actually needed. IMPE integrates a wide variety of tools into one user-friendly web application that accomplishes this. What differentiates this from existing tools is the wide scope; it provides automatic static and dynamic analysis from numerous sources and combines all of the results into a concise, coherent report.

6. FUTURE WORK

IMPE is already a full-fledged, independent application capable of providing analysts with a wealth of information. However, there are many other resources and tools they need to consult in the process. So, IMPE is currently being refactored to fit into an existing analyst tools portal. With the addition of IMPE, we hope the portal's suite of tools will be the central hub that analysts use, eliminating the need to pivot back and forth between various sites.

A more long-term and overarching goal is to have IMPE evolve into a commercial product. Currently, it is an internal tool used by one team of analysts, but it can feasibly expand to support much greater demand. As an effective tool to improve the efficiency of security analysts, it would likely prove valuable to many clients when it reaches that stage of maturity.

REFERENCES

[1] Marcus Botacin, Marco Zanata Alves, Daniela Oliveira, and André Grégio. 2022. Heaven: A hardware-enhanced antivirus engine to accelerate real-time, signature-based malware detection. *Expert Systems with Applications* 201 (2022), 117083. DOI:http://dx.doi.org/10.1016/j.eswa.2022.117083

[2] Rafiqul Islam, Ronghua Tian, Lynn Batten, and Steve Versteeg. 2010. Classification of malware based on string and function feature selection. *2010 Second Cybercrime and Trustworthy Computing Workshop* (2010), 9-17. DOI:http://dx.doi.org/10.1109/ctc.2010.11