Next Generation Malware Detection, Clustering and Heuristics

Proofpoint Staff:

Brad Woodberg

MSU Team:

Graham Thomas

George Zhao

Brad Doherty

Crystal Lewis

Yash Patel

Table of Contents

Executive Summary

Proofpoint is one of the leading next-generation cybersecurity companies in the world. Proofpoint protects the software that people use in their daily life like email, mobile applications, and social media. The company was originally founded in 2002 and is currently headquartered in Sunnyvale, CA. Every day, they detect and block threats in more than 600 million emails, 7 million mobile apps and hundreds of social media accounts. Additionally, they also help people send large files securely, and protect sensitive information. In order to do all this, they have malware analysts look over hundreds and thousands of malware files per day. With the threat of malware growing faster than malware analysts can scale, it would be more efficient if they spent more time looking over malware that is new and more malicious. That is why we are introducing “Next Generation Malware Detection, Clustering, and Heuristics”.

This dashboard is going to work in real time by analyzing all the provided malware and provide information about which ones need further inspection and which malware is already known. The main purpose of the dashboard is to help an analyst spend less time analyzing the known malware and more time analyzing the unknown malware.

Functional Requirements

The landscape of malware is ever-changing, with millions of malware samples being created and distributed daily. Proofpoint has proposed a way to reduce the number of malware an analyst must analyze by creating a tool which automates the malware analysis process as much as possible. This tool flags unknown malware, so that the analyst can focus their time determining what this new malware does and how it does it. The tool also shows the analysts information about the malware that will increase the speed of signature generation. Additionally, the analysis tool clusters malware together, based on similarity between information about the malware, so that it is easily displayed on the web application. The web application dashboard will display relevant information about the batch of malware being processed such as flagged files, total number of analyzed files, and specific information about a selected file.

The primary goal of this project is to reduce the amount of malware that must be analyzed and cluster the analyzed malware together. By automating the process, the workload of an analyst is reduced significantly. The tool processes the malware corpus hosted remotely, statically analyzes it, determines if dynamic analysis is appropriate and runs it through a clustering tool. The clustering tool will group similar types of malware together so that the web application can display the information generated by the tool.

The web application dashboard for analysts is a secondary goal for this project. This dashboard will display all relevant information pertaining to the malware that has been analyzed. The web app will interact with an API that handles all relevant information gathering. The dashboard display contains a graphical representation of the batch of malware, as well as a table containing every file that has been processed. Clicking on a malware entry navigates to a webpage with specific information about the file (e.g. file size, file hashes, time needed to analyze, etc.).

The third goal of this project is to provide a framework to develop signatures for malware after dynamic analysis. If, after dynamic analysis, the malware matches no known signatures, the analysis tool provides a way for an analyst to easily create a signature for the new malware. It collects information that is important for an analyst to be able to develop a new signature for it.

Design Requirements

Overview

Next Generation Malware Detection, Clustering, and Heuristics is an application designed to decrease the amount of malware a human analyst needs to manually examine by automating the process wherever possible. Using malware analysis techniques such as static and dynamic analysis, this tool filters out malware that is not interesting to a human analyst and prioritize malware that requires further human analysis. When possible, the system automates the job of the analysts by generating new signatures for malware that has not been seen before.

The web-based dashboard gives human analysts real-time information about malware currently being processed by the system. This information is organized in a manner that will allow the analysts to quickly pick up on any trends or other information which will assist them in their job. Users can filter malware based on Yara Rule matches, flagged status, and more in an effort to increase their efficiency as much as possible.

Figure one shows the flow of malware through our back-end analysis tool, a piece of malware is first run through static analysis. The results of static analysis determines whether or not dynamic analysis is run on the malware file. If dynamic analysis is not run the results are sent immediately to the database. However, if the file requires dynamic analysis the results of dynamic analysis will also be sent to the database, after it is analyzed. The tool clusters these results based on similar based on metrics such as file signatures, behavior, network signatures, malware type, hash, etc. This system provides a framework which assists malware analysts in signature generation, by gathering all relevant data related to signature writing and displaying the results. This data is accessed using an API, and the front-end dashboard will display these clusters of malware graphically. Additionally, the dashboard displays details about files which have been analyzed by the tool.

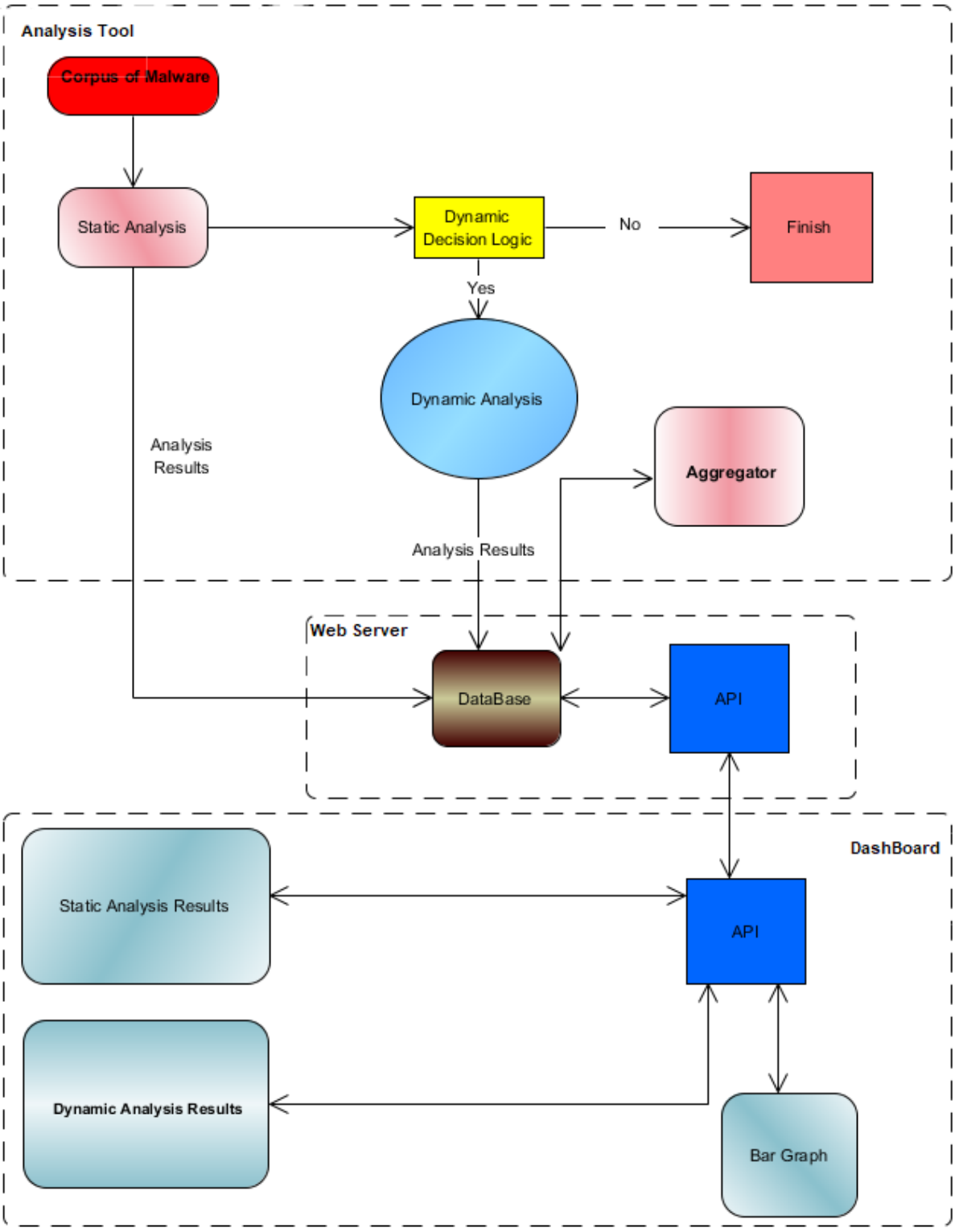

Figure 1: Process Flow

Web Application

The dashboard is accessible by malware analysts working at Proofpoint. Analysts are presented with a graph which displays a count of all the Yara rules that are matched for that batch. Malware may be determined to be similar based on metrics such as file signatures, behavior, network signatures, malware type, hash, etc. This graph will be updated in real time with malware that the system has finished processing.

The dashboard includes a web page for each file in the system. This web page shows information pertaining to each file, such as file hashes, file statistics, similar malware, Yara output, etc. The similar malware table contains links to the webpages of the malware that has been deemed similar by the system.

Dashboard Home Page

The dashboard home page contains a bar graph that shows the total number of malware that have matched a specific Yara rule, as well as a table that has an entry for each file that has been processed by our system.

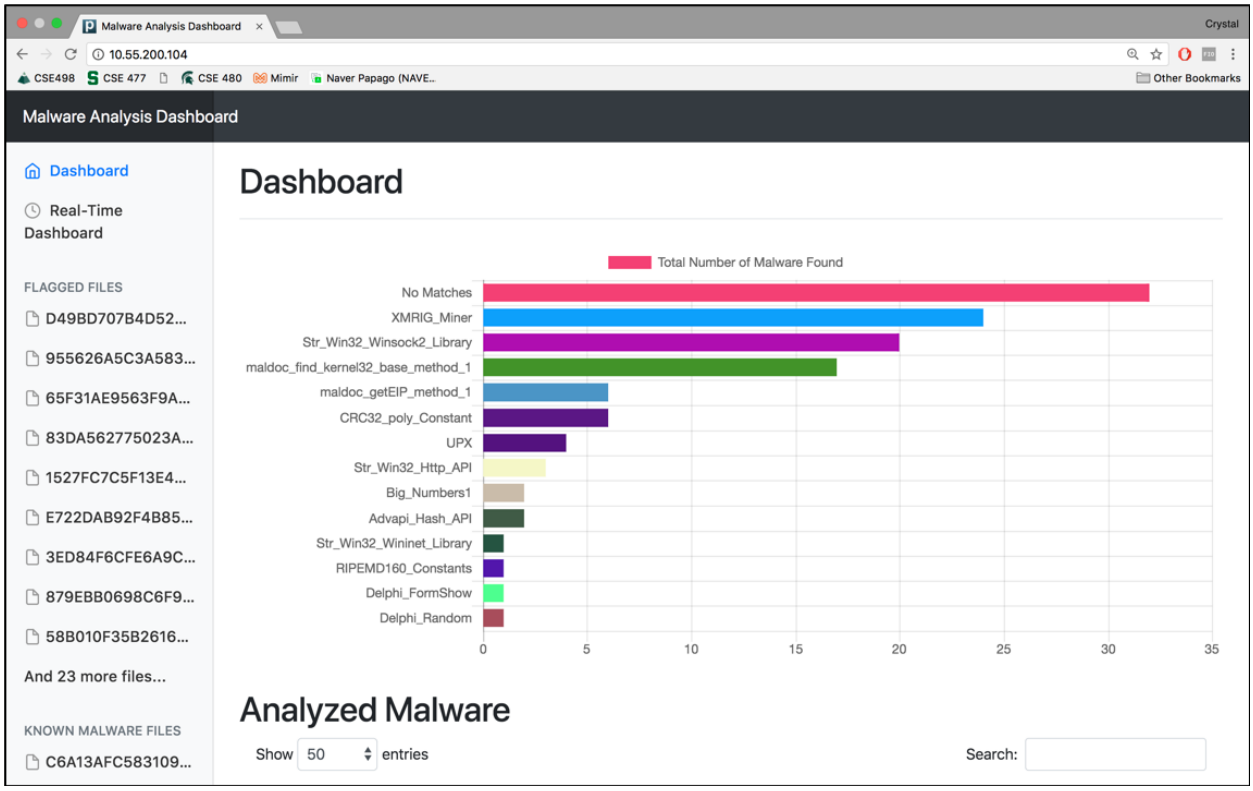

Figure 2: Bar graph showing number of malware matching the Yara rule

Specific File Page

The specific file page contains information about the specific file. It will contain file statistics, such as run time, file size, and the file extension. Additionally it has information about the file hashes, similar malware, and the output of the Yara rule it matched, if any.

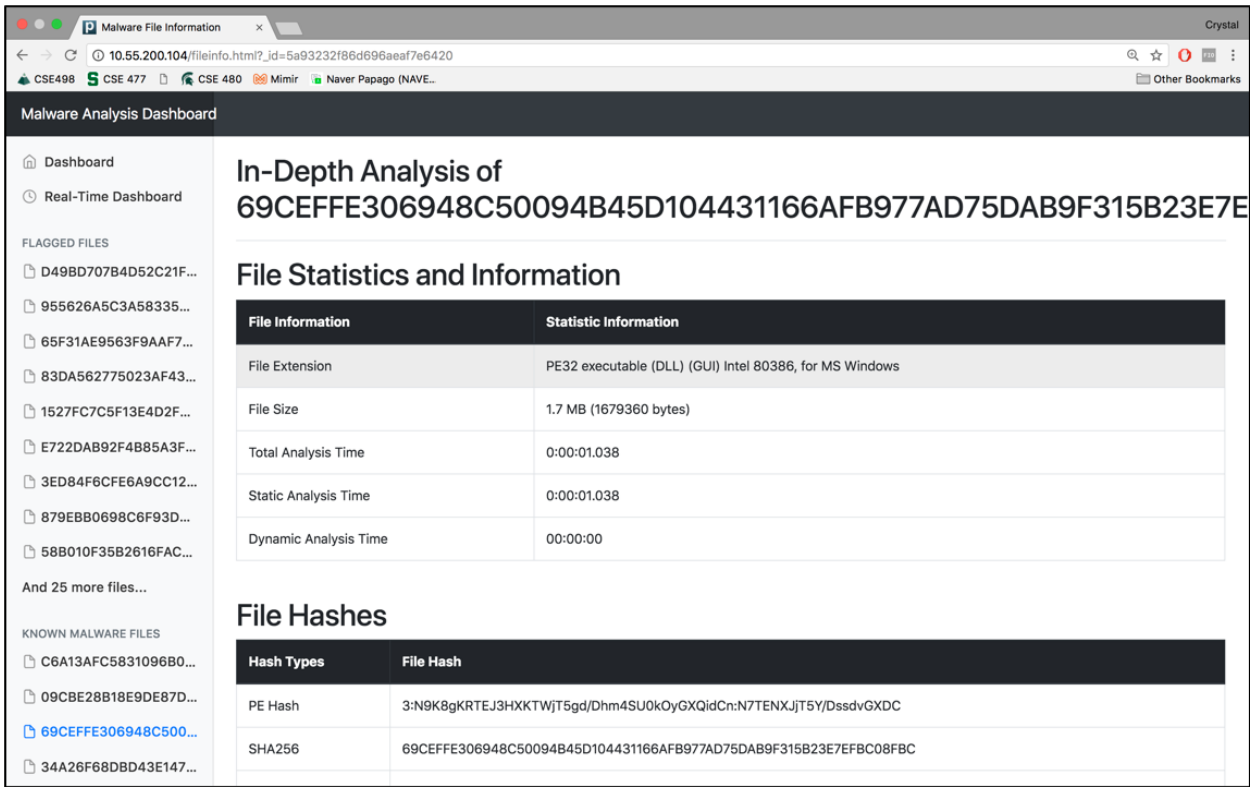

Figure 3: In-depth analysis of a file

Technical Specifications

System Architecture

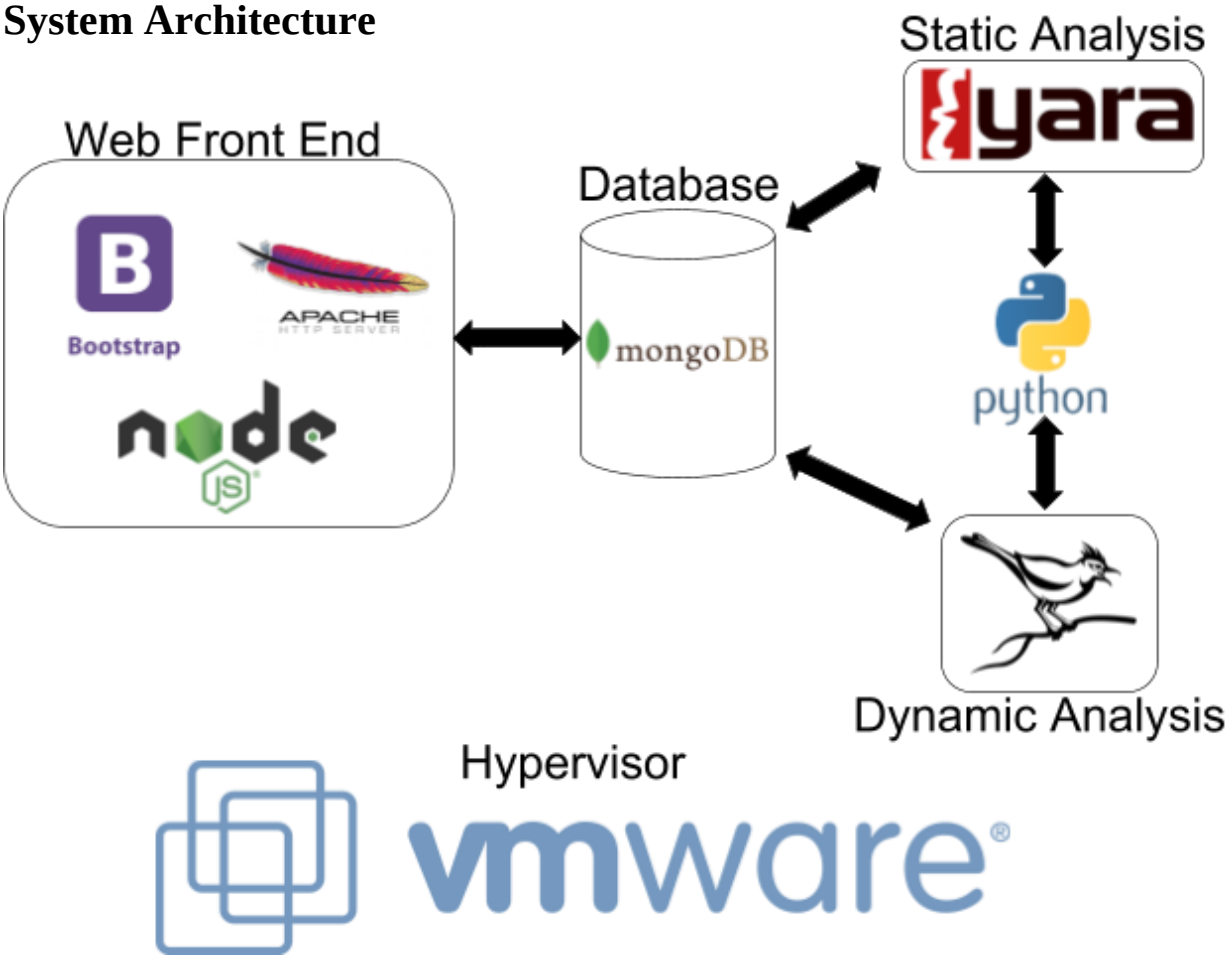

Figure 4: System Architecture Diagram

The system is hosted on a cluster managed by VMWare ESXi hypervisor. On top of the hypervisor, five virtual machines running Ubuntu operating systems run the malware corpus, the static malware analysis tool, the dynamic malware analysis tool, the Mongo database, and the Apache web server. When a piece of potential malware is fed into the system it first undergoes static analysis by tools such as ClamAV antivirus and Yara Rules. From here, a report is generated that the decision logic can use to determine if the malware should undergo further dynamic analysis. If the decision logic determines a need for further analysis, the malware is executed in Cuckoo sandbox. In both cases, however, information from static analysis and sandboxing, if applicable, is parsed into a format that provides information for signature writing. After the piece of malware has been processed, it passes through clustering logic to determine what

other pieces of malware it is similar to. All this information is then written to the Mongo database server.

When interacting with the web interface, a user can see details on malware that has been passed through the system through a Bootstrap framework web page. The front end interacts with the back using API calls to the Apache web server that passes the call to NodeJs for call processing.

Backend Analysis

The backend analysis of a piece of malware begins with the parsing of file type and intended architecture. This along with clustering, tool output processing, and the driver for the backend system is written in Python 2.7, a language familiar to the development team with support for software tools like YARA and Cuckoo Sandbox.

The next step in the process is automated analysis using static methods. The piece of malware is analyzed against Yara rules, ClamAV virus scanner, and PE, SHA256, MD5, SHA1 hashes. Hashes provide static binary matches and near matches to known malware file patterns, while Yara rules and ClamAV look for particular static patterns. The information from the static analysis is then processed by the logic responsible for flagging malware for dynamic analysis for attributes that may indicate an interesting file.

If a piece of malware has interesting static characteristics, it will be passed to a sandboxing environment for dynamic analysis. A sandbox is a piece of software that leverages virtualization technology to execute the malware safely and record its behavior. In our case, we will be using Cuckoo sandbox because of it features built in IDS support. Cuckoo will provide a detailed account of malware behavior for further analysis.

After dynamic analysis, static analysis information such as hashes, Yara Rule hits, and ClamAV signatures are combined with dynamic analysis information such as screen-shots, PCAP information, network traffic, Suricata alerts, and dropped files into a format that is easily navigable for the purpose of signature assignment. Some of this relevant signature information is used to automate some signature generation (e.g. coinbots). All of the contents of the report are added into a database that can be used for cluster analysis and front end information population.

Front End Web Interface

The user facing web application is built on the Bootstrap JavaScript framework, incorporating HTML models and CSS style descriptions. Bootstrap was chosen for its robust dynamic front end application support. Apache web server serves the webpage elements to the browser, and the API calls to get data pertaining to instances of malware and the statistics of the malware set is handled through NodeJs. Finally, the database server for the data delivered by the analysis tool is Mongo database, a database familiar to the development team.

Server System

The webserver, database server, and malware analysis systems are run on a cluster. The allocated cluster resources are managed by VMWare ESXi hypervisor, and the five previously mentioned features are run atop Ubuntu distributions of Linux operating system. Inside the analysis tool, virtual machines required by the sandbox software Cuckoo are powered by VirtualBox.

Software Technologies

- Yara Rules
- ClamAV
- Python 2.7
- Cuckoo
- MongoDB
- NodeJs
- Apache Server
- Linux OS
- Bootstrap(JavaScript)
- ESXi Hypervisor

Development Environments

The backend logic, written in Python 2.7, will be created in Visual Studio Code. The backend logic will be tested using the PyTest module. The front end web-page will be built in the JetBrains PHPStorm. The production, development and testing phases will all take place on the cluster.

Test Plan

The system will be first tested on a corpus of pre-processed malware provided by Proofpoint. Later the system will be used on unknown samples to test real world viability.

Database

The information Malware file hashes and file type, Networking Traffic (DNS, SSL, TSL, and Suricata Alerts), Registry keys, mutex, and created files will all need persistent storage in order to for the front end to access relevant information. Proofpoint has provide a hardware cluster, in which we will be running MongoDB Server with a Mongo database.

Risk Analysis

Many risks face this application and development team. These risks will affect many different facets of the product, but through proper identification and prioritization, they can be mitigated effectively.

Malware Categorization and Clustering

Difficulty: Low
Importance: High
Description: The clustering of different malware will be based off a set of characteristics that we have to identify. These characteristics have not been defined yet and if they are identified incorrectly the clustering will not be accurate.
Mitigation: Conduct research to understand the behavior, origin and defining characteristics of different types of malware. Conduct research to understand how clustering algorithms work and what kind of information from malware groups are needed to use clustering algorithm.

Scalability and Speed

Difficulty: High
Importance: Medium
Description: The volume of malware that will be analyzed in a given timeframe isn't known and will vary. The web application needs to be able to handle large volumes of malware and be time efficient when analyzing these files as well.
Mitigation: Establish a basis for the type of volume of malware to be analyzed per given time period. After this basis is set, the resources for the analysis software being used can be allocated and managed appropriately. This includes calculations for the probability for how often Dynamic Analysis will need to be conducted.

Understanding Dynamic and Static Analysis Tools

Difficulty: Low
Importance: High
Description: Static and Dynamic analysis will be conducted using a collection of software tools the team is unfamiliar with. Each of this software requires different input and have different output formats. Understanding each of these for the software is important so that each piece of malware can be properly analyzed and clustered.
Mitigation: Running different malware samples and testing different scenarios will provide insight into inputs and outputs. Additionally, subject matter expert advice from a Malware analyst will provide more accurate high level views for how each tool can play a role in analysis.

Web API

Difficulty: Medium
Importance: High
Description: API is necessary for information between the analysis and clustering process on the backend and the dashboard on the front end. Development team is unfamiliar with this technology.
Mitigation: Prototype different test API to understand development process. This will be done with research and trial and error.

Signature Generation Framework

Difficulty: High
Importance: High
Description: The malware analysts need to be able to easily create signatures for files we analyze, especially involving new malware.
Mitigation: Determine what analysis information is relevant for a signature, so that it can be easily displayed for the analyst on the dashboard.

Schedule

Product Stages

P0: Research and test malware software, configure tools and virtual machines

P1: Identify file types for static analysis, begin dynamic analysis logic, basic web layout

P2: Clustering implementation, Database implementation, Dynamic and Static Analysis

P3: API and Database and Web App integration, Signature Generation

P4: Real time cluster updates, Dashboard hyperlinks and more signature generation

Week 1/8-1/14

-Team assignment

-Initial team meeting

-Established slack, point of contact and team dynamics

-Initial client meeting

Week 2: 1/15 - 1/21

-Research malware tools Cuckoo, ClamAV, and Suricata

-Establish outline for Dynamic Analysis

-Establish outline for Static Analysis

Week 3: 1/22 - 1/28

-Project Plan Presentation and Document complete

-Website Server and Database Server deployed

-Meet with Malware Analyst

Week 4: 1/29 - 2/4

-Read malware determine file type

-Run PE Hash

-Sandboxing environment set up

-Website skeleton implemented

Week 5: 2/5 - 2/11

-Basic API functionality mapped out

-Database set up and populated with Dummy data

-Parsing output for Dynamic and Static analysis

Week 6: 2/12 - 2/18

-Web functionality UI complete

-Feed info into database

-Begin clustering

Week 7: 2/19 - 2/25

-API complete

-Practice Alpha Presentation

-Finish clustering

-Alpha Presentation (2/12 - 3/1)

Week 8: 2/26-3/4

-Integration of API and website and clustering

-Begin decision logic for dynamic analysis

Spring Break: 3/5 - 3/11

-Framework relevant to signature info

-Finish decision logic for dynamic analysis

-Status Report

Week 9: 3/12 - 3/18

-Web Fully Functional

-Signature generation support

Week 10: 3/19 - 3/25

-Testing deployed malware feed

-Quality assurance

Week 11: 3/26 - 4/1

-Practice Beta presentations

-Beta Presentation (4/3 - 4/12)

Week 12: 4/2- 4/8

-Video outline plan

-Film video

Week 13: 4/9-4/15

-Video filmed, edited

-Status report

-Testing of the product

Week 14: 4/16-4/22

- Finalize video

-Testing of the product

Week 15: 4/23-4/29

-Design day

-All deliverables due