CSM

Automated Confidence Score Measurement of Threat Indicators

by

Ajay Modi

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2017 by the
Graduate Supervisory Committee:

Gail-Joon Ahn, Chair
Ziming Zhao
Adam Doupé

ARIZONA STATE UNIVERSITY

August 2017

ABSTRACT

The volume and frequency of cyber attacks have exploded in recent years. Organizations subscribe to multiple threat intelligence feeds to increase their knowledge base and better equip their security teams with the latest information in threat intelligence domain. Though such subscriptions add intelligence and can help in taking more informed decisions, organizations have to put considerable efforts in facilitating and analyzing a large number of threat indicators. This problem worsens further, due to a large number of false positives and irrelevant events detected as threat indicators by existing threat feed sources. It is often neither practical nor cost-effective to analyze every single alert considering the staggering volume of indicators. The very reason motivates to solve the overcrowded threat indicators problem by prioritizing and filtering them.

To overcome above issue, I explain the necessity of determining how likely a reported indicator is malicious given the evidence and prioritizing it based on such determination. Confidence Score Measurement system (CSM) introduces the concept of confidence score, where it assigns a score of being malicious to a threat indicator based on the evaluation of different threat intelligence systems. An indicator propagates maliciousness to adjacent indicators based on relationship determined from behavior of an indicator. The propagation algorithm derives final confidence to determine overall maliciousness of the threat indicator. CSM can prioritize the indicators based on confidence score; however, an analyst may not be interested in the entire result set, so CSM narrows down the results based on the analyst-driven input. To this end, CSM introduces the concept of relevance score, where it combines the confidence score with analyst-driven search by applying full-text search techniques. It prioritizes the results based on relevance score to provide meaningful results to the analyst. The analysis shows the propagation algorithm of CSM linearly scales with larger datasets and achieves 92% accuracy in determining threat indicators. The evaluation of the result demonstrates the effectiveness and practicality of the approach.

To my parents, who taught me the first letter of my life,

To my brother, who made me believe in myself,

To my soulmate, for boundless support to make everything I do possible,

To God, for being companion throughout my journey so called life.

ACKNOWLEDGMENTS

Foremost, I express my sincerest gratitude to my advisor Dr. Gail-Joon Ahn for his continuous feedback giving me direction and focus to this project. I also thank him for encouraging me to pursue a thesis in the cyber security field.

I want to thank Dr. Ziming Zhao, for his continuous encouragement and support over relentless revision. I also thank him for all his help, and his valuable input on the changes to be made to make the project more impactful.

I wish to thank Dr. Adam Doupé, for all his insight on practical matters concerning implementation which was enlightening and fruitful.

Finally, I thank the entire Threat Intelligence Analytics team and SEFCOM Lab members for their support and guidance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

The threat intelligence domain creates many new challenges for security analysts as it is vast and heterogeneous. The volume and frequency of new cyber threats and variants targeting organizations and companies are continuously on the rise and have become critical concerns as more automated tools are available to anyone with the bitcoin account to conduct cyber attacks. Not only the government and other public sectors are facing unprecedented cyber attacks, which may potentially undermine national security and critical infrastructure [34], but also individuals and businesses are vulnerable to cyber threats constituting a persistent threat to privacy, finances, and the economy as a whole. It was estimated that the likely annual cost to the global economy from cybercrime is more than \$400 billion in 2014 [23]. And, the number is projected to reach \$2 trillion in 2019 [26].

The staggering number of cyber crimes are able to evade existing security measures because they have complicated workflows. In light of this, organizations are looking to increase their knowledge base of threat intelligence data to better equip their security teams with the latest information on new and existing attack methods and how to stop them. As one solution does not fit all, to avoid being victimized, respondent organizations rely on multiple threat intelligence feeds, including the community-driven and vendor-driven feeds for aggregation and analysis to tackle against the adversary's attacks. Though such subscriptions add intelligence and can help in taking more informed decisions to security incidents, it comes with the unique set of challenges.

By following subscription to multiple source feeds and current practices, organizations may face a challenge to turn threat intelligence data from multiple sources into actionable, contextual information. To ensure threat detection and remediation happens in a timely manner, organizations have to put considerable efforts in facilitating and analyzing a large number of indicators. The dilemma of entrusting prevails organizations to subscribe to many reputable yet overlapping threat feed sources in fear of not losing out on potential harmful threat indicators which results in overcrowded data [39].

The volume and frequency based problem worsens further, due to a large number of false positives detected as threat indicators by existing threat feed sources. It forces organizations to analyze each individual threat alert which remains neither practical nor cost-effective considering the staggering volume of indicators. Considering the criticality to monitor threat indicators, limited time availability in responding to the situation and volume of observations, the manual assessment is not sufficient enough to measure the assurance of systems. The ability to quickly triage malicious items is of vital importance and it requires automated assessment of suspected malicious indicators [46]. The very reason motivates to solve the overcrowded threat indicators problem by prioritizing and limiting them.

In order to determine the maliciousness of threat indicators, CSM introduces the concept of confidence score, where it defines how likely the reported indicator is malicious based on the evaluation of the indicator by different threat intelligence systems. An indicator propagates maliciousness to adjacent indicators based on relationship determined from behavior of an indicator. The propagation algorithm derives final confidence to determine overall maliciousness of the threat indicator. Further, an analyst may not be interested in the entire result set generated from the confidence score, so CSM has to narrow down the results based on the analyst-driven input. To this end, CSM introduces the concept of relevance score, a method for computing a ranking of every threat indicator by combining confidence score with analyst driven input. When the analyst queries to the system, it filters result based on full-text search techniques and combines pre-calculated confidence score in real time to generate relevance score of the threat indicators. The relevance score balances the user-driven input with confidence score to provide meaningful result to the analyst by prioritizing them.

To sum up, CSM provides an analysis framework that crawls and analyzes plenty of heterogeneous threat indicators. CSM exploits the domain knowledge of various analysis systems in a structured tuple format to store into the knowledge graph. It then applies an iterative propagation algorithm to propagate maliciousness to adjacent threat indicators and to derive final confidence score of a threat indicator through convergence. It further incorporates user search terms to derive relevance score based on search criteria provided by the analyst. With this refined scores, CSM provides useful information to help security analysts make an informed decision with further investigation.

Structure of document This thesis document is divided logically into the following sections:

- Chapter 2 discusses the challenges in threat intelligence domain and motivation to solve overcrowded threat indicators problem in that domain.
- Chapter 3 discusses the system design, the architecture, and the components of the system.

- Chapter 4 compares existing graph propagation techniques and their shortcomings. Then, it discusses the algorithm design through various scenarios and explains the algorithm in detail.
- Chapter 5 describes the full-text search techniques, and it's integration with derived confidence score to prioritize and limit the results according to the search of an analyst.
- Chapter 6 describes the experimental setups.
- Chapter 7 evaluates the system, presents our findings and our analysis of the results. It also describes limitation and scope of improvement of the research.
- Chapter 8 explores related work in the area.
- Chapter 9 concludes this thesis.

We hope that our research sheds some light on this relatively new problem faced by threat intelligence teams, and suggests one possible approach to solve the problem. In summary, we make the following contributions:

- We developed a novel graph propagation approach to calculate confidence and relevance score of the threat indicator.
- We designed and implemented a scalable architecture that can incorporate heterogeneous threat indicators from various threat intelligence systems, stores into the knowledge graph and propagates confidence throughout the graph to provide proof of concept for our approach.
- Our proof of concept shows the feasibility of our technique to limit and prioritize threat indicator based on an analyst input and confidence score.

Chapter 2

BACKGROUND

This chapter describes the current practices followed in cyber security domain. Next, it focuses into threat intelligence domain and goes into detail about the challenges with threat intelligence. It then provides motivation for our research in the relevant field by emphasizing the requirement of limiting and prioritizing the threat indicators.

2.1 Problem Background

To understand the challenges faced by threat intelligence teams, we need to first understand the current trends observed in the organizations. Current trends in the cyber security domain remain as follows:

- **Attackers use sophisticated technology and tactics with continuous attempts to breach security**: As Mcafee reports, they have seen a change during the past two years, with a significant increase in the number of technically sophisticated attacks. Many of these have been designed purely to evade advanced defenses. Attackers are infiltrating in pieces, hiding in seemingly inert code, and waiting for an unprotected moment to emerge. These threats also avoid the signature-based traps of their ancestors, employing encryption and dynamic code modification to change with each new deployment and hide incriminating data [24].

- **Enterprises lack the resources or skills to protect the assets**: A report from Cisco reports one million cybersecurity job openings at global level [9]. Symantec points out that demand is expected to rise to 6 million globally by 2019, with a projected shortfall of 1.5 million. It expects demand to further rise due to burgeoning cybersecurity market which is expected to grow to $170 billion by 2020 [48]. Moreover, statistics reports show that more than 209,000 cybersecurity jobs in the U.S. are unfilled, and postings are up 74% over the past five years, which emphasizes the lack of resources to protect the assets of organizations [37].

- **Analysis, enforcement, and mitigation is largely manual**: Today, 57% of cybersecurity professionals claim that their threat intelligence programs are "somewhat mature" or "immature". Collecting, processing, correlating, and analyzing threat intelligence is still a manual effort as cybersecurity professionals spend a lot of time cutting and pasting data from emails, transforming data formats, and writing code. Many organizations are still figuring out how to weave threat intelligence into things like communication, collaboration, risk scoring, and IT workflows. Enterprises are sharing internally-derived threat intelligence, but they are doing so on an ad-hoc and informal basis [14]. Such manual proactive steps or countermeasures lead to high response time as harsh reality statistics validate that for the vast majority of incidents (85%), attackers are able to compromise the victim very quickly (minutes or faster) but initial discovery of compromise happens only after few days [46].

- **Small organizations cannot afford to have separate personnel**: Small and mid-sized businesses are hit by 65 percent of all cyber-attacks and the last five years have shown a steady increase in attacks targeting businesses with less than 250 employees. As small organizations cannot afford to have a dedicated team looking after vulnerabilities, they become an easy, soft target to penetrate for cybercriminals [41].

2.2 Challenges with Threat Intelligence

As per trend, many organizations have some level of threat detection and incident response capabilities. But building out these capabilities to take a proactive stance against an evolving threat landscape is often expensive and challenging. It requires additional and significant investment in people, intelligence, technology, and analytics. In the past few years, it has become abundantly clear that enterprises leveraging threat intelligence have a distinct advantage in protecting their critical infrastructure [13]. In light of this, organizations are looking to increase their knowledge graph of threat intelligence data to better equip their security teams with the latest information on new and existing attack methods and how to stop them. As one solution does not fit all, to avoid being victimized, respondent organizations rely on multiple threat intelligence feeds, including the community-driven and vendor-driven feeds for aggregation and analysis to tackle against the adversary's attacks. Organizations normally consume threat intelligence from paid solutions, trusted partners, formal industry corporations, government and law enforcement agencies, and open source solutions, as well as their own analysis and detection processes. Organizations are increasingly integrating threat intelligence feeds into their security architecture. Though such subscriptions add intelligence and can better pinpoint threats to specific systems and help in focusing efforts on more informed responses to security incidents, it comes with the unique set of challenges. Before we address those challenges, letś understand what is threat intelligence.

According to [38] threat intelligence is the set of data collected, assessed and applied regarding security threats, threat actors, exploits, malware, vulnerabilities and compromise indicators. The challenges with threat intelligence are mentioned below:

- **Too many alerts.** As organizations consume threat intelligence from numerous sources, security teams are overwhelmed with threat feed alerts. They have limited staff members to investigate and triage all the alerts. It leads to the reason where the team struggles to filter out the noise because there were too many irrelevant alerts produced by internal sources or consumed from external sources.
- **Too little time to respond.** Time remains an extremely critical factor in threat intelligence environment. As reported by [46], the time between the initial attack to the initial compromise remains trivially small. Hence security team has very little time to investigate the compromise and provide mitigation or enforcement steps. With the volume of data breach directly proportional to the time taken to address compromise, too little time to respond remains an ardent challenge for threat intelligence team.
- **Too few resources.** In order to ensure threat detection and remediation happens in a timely manner, organizations have to put considerable efforts in facilitating and analyzing a large number of indicators. Often, reported potential threat indicators outnumber the capacity of investigating and triaging all the alerts resulting in a struggle for the team to filter out important alerts from irrelevant alerts.
- **Issues with false positives** The problem with the deluge of threat feeds becomes worse with false positives or irrelevant information provided by

sources. False positives may lead to conclude a wrong decision. As per the survey conducted, more than 83% security practitioners agreed to face a problem of false positives reported by threat feed sources [19].

2.3 Motivation

As overwhelmed by security alert volume, today's organizations are in dire need of smarter, faster and stronger solutions. They require an approach which helps them to filter out and limit irrelevant data and prioritize the threat indicators as per the need of an analyst.

To further strengthen our claim, we present a motivating example to show the overwhelming threat indicators provided by different threat intelligence systems, which clearly shows the necessity of an automated prioritizing and filtering system to accelerate the incident response life cycle.

On a daily basis, [6] [2] [28] [3] [33] such organizations published list of IP addresses, domains, URLs, ASN and other threat indicators. Considering each list contains more than 100 threat indicators, a threat intelligence team deals with more than a couple of 1000 indicators on a daily basis. Research by Anomali and the Ponemon Institute found that 70 percent of security professionals believe the data is too voluminous and complex to be actionable [40]. Further, all these lists provide isolated disjoint intelligence from heterogeneous sources which does not provide a holistic picture to understand cyberattack events. A careful analysis of such a scenario tells us the necessity to prioritize and limit the threat indicators as per the requirement of an analyst which remains the motivation for my thesis.

2.4 Research Questions

This motivation leads us to ask three research questions. First question is how likely the reported indicator is malicious? This question tries to understand whether reported indicator is malicious or not. CSM introduces the concept of confidence score to measure maliciousness of a threat indicator. However, with the limited amount of time and resources, an analyst may have time to investigate only 5 out of 500 indicators. This prompts us to ask second research question that is how to prioritize reported threat indicators? We may use confidence score to prioritize reported indicators but this provides an entire knowledge graph of indicators with measured confidence score of each indicator. However, there is a better way to filter and prioritize threat indicators. Since, different analyst have different perspectives, and interests, we need to integrate analyst's perspective into search results. This guides us to ask third research question that is how to prioritize and filter threat indicators based on analyst's inputs? To solve this research question, CSM introduces the concept of relevance score.

The confidence score is the value between 0 to 100 assigned to the threat indicator that represents the confidence we have in an indicator being malicious. 0 confidence suggests that we are not confident enough to consider an indicator as malicious. 100 confidence suggests we are extremely confident about malicious behavior of the indicator.

The relevance score is the value between 0 to 100 that shows how relevant a threat indicator to the analyst interest. 0 relevance suggests that a threat indicator is irrelevant to the analyst query. 100 relevance suggests that a threat indicator is extremely relevant to the analyst query.

Chapter 3

SYSTEM DESIGN

In this chapter, we present the design goals that CSM strives to meet. Then, we discuss the system design and explain the architecture and the components of the CSM in detail. We then proceed to enumerate the issues faced and the assumptions made during the building of the system.

3.1 System Design Goals Of CSM

To calculate the confidence score and relevance score we designed and implemented Confidence Score Measurement System abbreviated as CSM. CSM is designed with the following goals in mind:

- **Continuous Collection of Threat Feeds.** Discovering intelligence is a continuous process as we may find new evidence at any moment during the process. Not only that, sometimes, the discovered intelligence may get updated. CSM requires to continuously integrate newly discovered or updated intelligence into the knowledge graph by properly interlinking between threat indicators and propagating confidence among them.

- **Reusing Existing System.** Existing analysis tools can provide more details about reported threat indicators. For example, VirusTotal is the leading service that analyzes suspicious files and URLs and facilitates the quick detection of viruses, worms, trojans, and all kinds of malware by providing detailed reports including monitored system calls and static and dynamic analysis [47]. Safe Browsing is a Google service that lets client applications check URLs against Google’s constantly updated lists of unsafe web resources including social engineering sites (phishing and deceptive sites) and sites that host malware or unwanted software [36]. By reusing existing analysis systems, CSM will be able to establish interlinks between indicators in the light of evidence.

- **Automated Confidence and Relevance Score Calculation.** Given the ever increasing volume of threat indicators, the manual process of calculating confidence score and limiting and prioritizing the relevant data is impossible for human analysts in a timely manner. Therefore, automatic processes are desperately needed to help analysts utilize their time for value-added analysis.

3.2 System Workflow

Figure 3.1 provides a high-level overview of the workflow of our proposed CSM, which begins from the collection of data and ends at the generation of prioritized results.

CSM collects threat indicators from various threat intelligence systems which consist of autonomous crawlers and their corresponding parsers. For example, a crawler for VirusTotal [47] will continuously crawl threat indicator analysis reports for malicious hash values, URL, domain or IP address using VirusTotal API. Similarly, a crawler for ThreatCrowd [45] will continuously collect threat intelligence by scraping HTML structure for reported hash, IP address, and domain. Each threat intelligence system generates detailed analysis report about the threat indicator in JSON file format. JSON format makes it easier to extract relevant behavioral information of the threat indicator and store it in the knowledge graph. These analyzed reports are passed to individual parsers for further

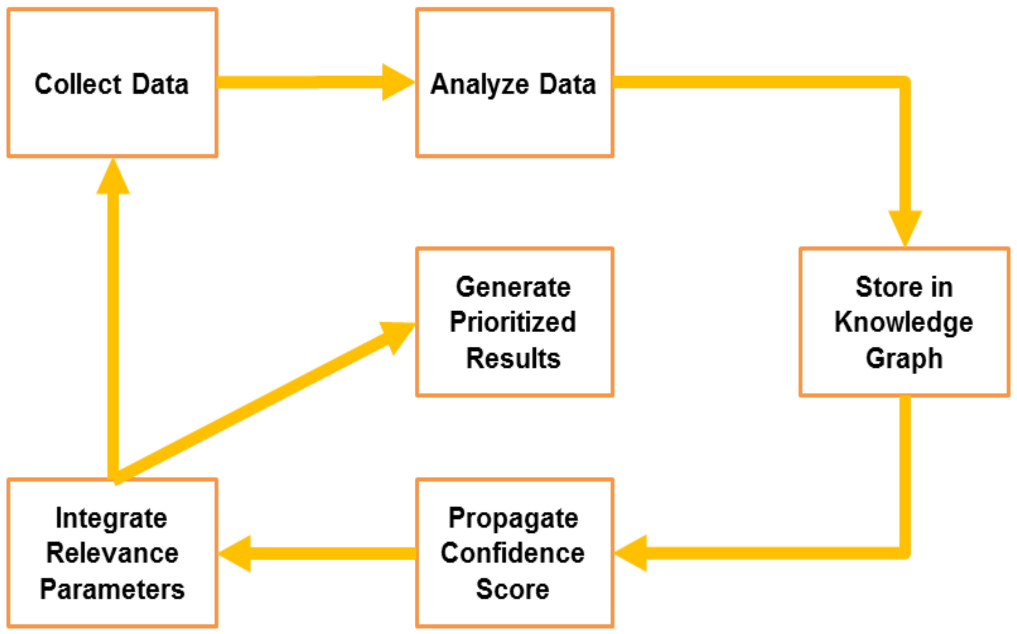

Collect Data
Analyze Data
Generate Prioritized Results
Store in Knowledge Graph
Integrate Relevance Parameters
Propagate Confidence Score

Figure 3.1: System Workflow Overview.

processing, where the parser converts these reports into a set of structured format to store into the knowledge graph, a graph database. The parser utilizes the behavior reported by threat feeds to establish relationships between threat indicators in the knowledge graph. The next step is to propagate the maliciousness to the neighbor threat indicators based on the relationship and derives confidence to determine overall maliciousness of the threat indicator. For example, a malware connecting to URL to receive further instructions propagates malicious behavior to URL. Lastly, CSM narrows down the scope of generation of the result, based on the analyst-driven input as an analyst may be interested in specific indicators. It prioritizes and filters the results based on relevance score to provide meaningful results to the analyst.

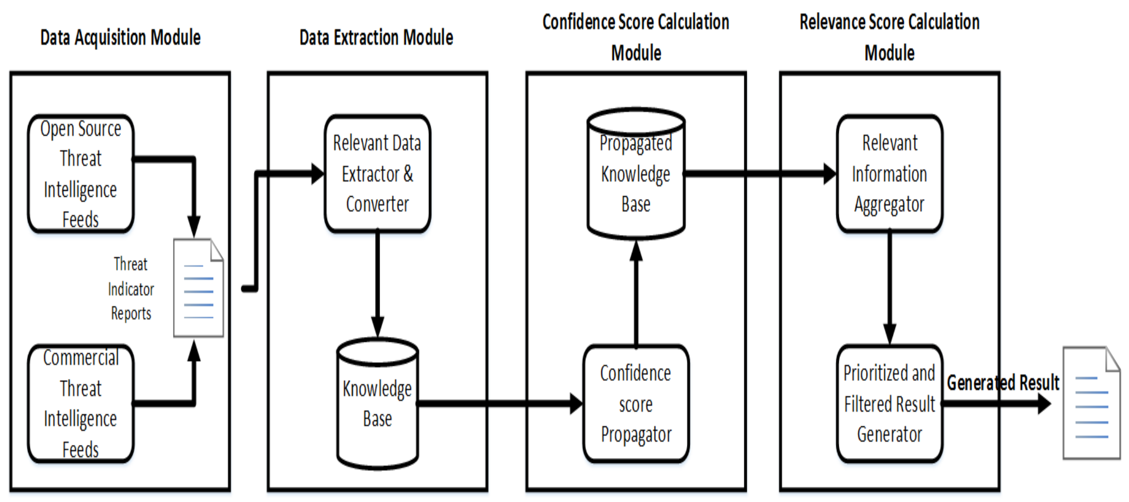

Data Acquisition Module
Data Extraction Module
Confidence Score Calculation Module
Relevance Score Calculation Module
Open Source Threat Intelligence Feeds
Threat Indicator Reports
Commercial Threat Intelligence Feeds
Relevant Data Extractor & Converter
Knowledge Base
Propagated Knowledge Base
Confidence score Propagator
Relevant Information Aggregator
Prioritized and Filtered Result Generator
Generated Result

Figure 3.2: System Architecture Overview.

3.3 System Architecture

The CSM architecture shown in Figure 3.2 consists of 4 modules from left to right: i) Data Acquisition Module ii) Data Extraction Module iii) Confidence Score Calculation Module iv) Relevance Score Calculation Module. This section describes in detail the functionality of each of the module.

3.3.1 Data Acquisition Module

This module collects information from various threat sources. It continuously collects open-source threat intelligence feeds as well as commercial feeds from various autonomous threat intelligence systems such as AlienVault Open Threat Exchange to collect indicators of compromise reported by thousands of active users on such platforms. These indicators are collected and stored in a relational database for future reference. More detail on the same is orchestrated in 3.3.1 database subsection. These indicators are further queried to various open-source threat intelligence systems, such as VirusTotal, PyWhoIs, ThreatCrowd, ThreatMiner, SafeBrowsing, and various other blacklist maintaining websites to fetch additional information of threat indicators. The crawler maintains threat indicators in a queue to further query to different threat intelligence system APIs respecting their usage limits.

Crawler

The crawler uses two different methods to collect information. i) API calling ii) Web scraping.

i) API calling. In general, threat intelligence feeds have an API for their customers. They provide a relatively easy way to extract information about the threat indicator.

However, many such threat intelligence feeds do not have an API or there are many limitations on the data that is available through the API. Even if the API provided access to all the data, we need to adhere to their rate limits. To overcome such shortcomings, we developed the second approach regarding web scraping.

ii) Web scraping. Our web scraper collected information from more than one hundred websites providing blacklist or threat intelligence feeds. We respected the rate limits in many such systems to not hammer the feeds with hundreds of concurrent requests.

For the scope of our research, we collected 70,402 threat indicators from the reported analysis. All these analyses are published between November 2016 and April 2017. Sample structure of a report crawled by the crawler is shown in the Listing 3.1.

```
{
        "report": [
                {
                        "intelligence": {
                                "type": "domain",
                                "information": "..."
                        },
                        "system": "ThreatCrowd",
                        "attribution": "example.com"
                },
                {
                        "intelligence": {
                                "type": "domain",
                                "information": "..."
                        },
                        "system": "VirusTotal",
                        "attribution": "example.com",
                }
                ...
        ]
}
```

Listing 3.1: Sample analysis report

In general, CSM subscribes and crawls multiple cyber threat generated feeds to accumulate collective intelligence reports.

Database

CSM collects and stores as much data as possible at each stage in the system. It is due to the two following reasons:

1. The data is used to validate our findings.

2. The data collected can be used for other research projects in this area.

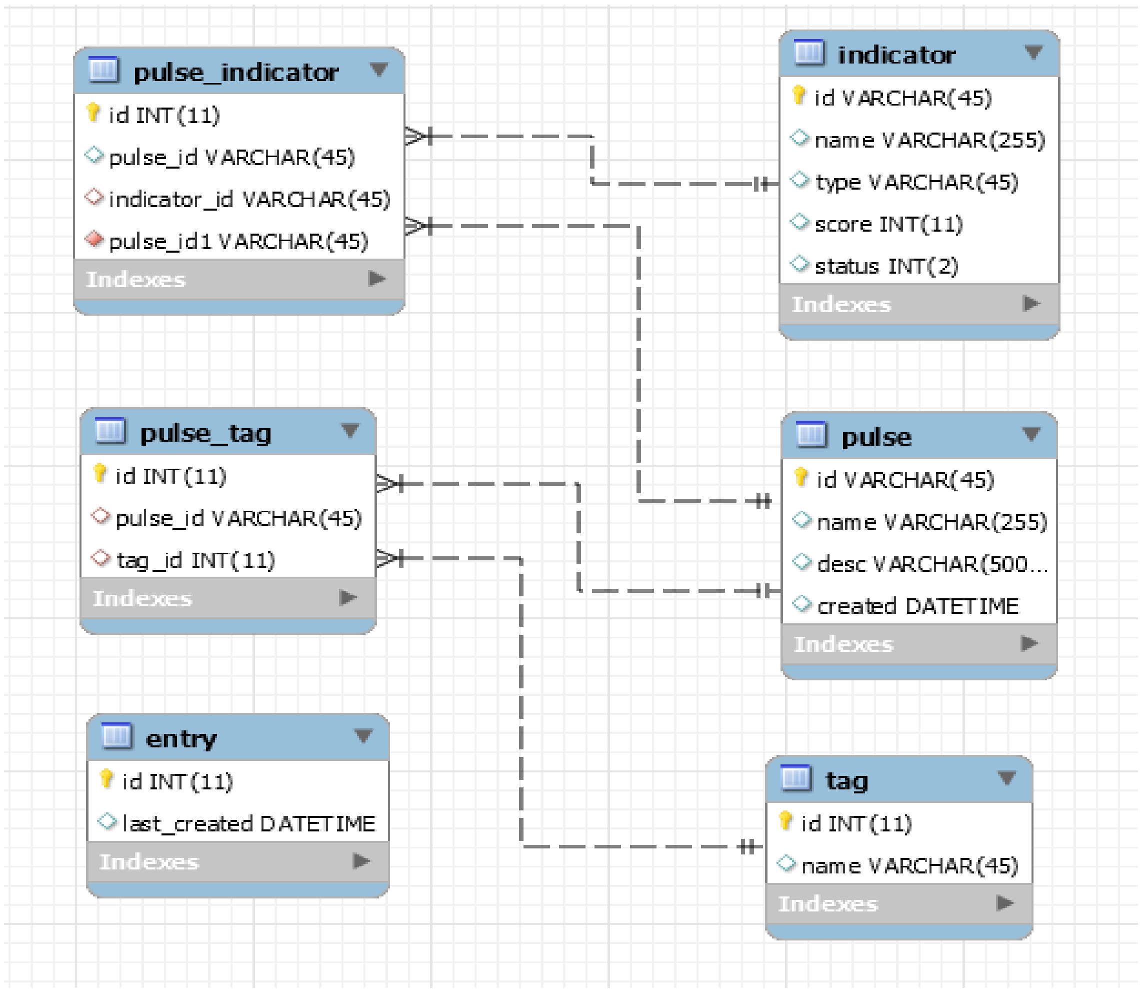


Figure 3.3: Database Schema.

While collecting the information, we found that many such threat indicator reports provide an additional and important set of information which can be used to decide whether a particular artifact is malicious, whether it is a part of any ongoing malicious campaign, or does it talk about any new attacks for which signatures has not been developed by existing security firms. Considering this information as highly important and relevant, we design a schema of the database shown in Figure 3.3. Each table in our database is listed in Table 3.1 along with the data it is designed to hold.

In CSM, data acquisition module collects key observations from threat intelligence sources. It collects such information in real-time and keeps the knowledge graph of CSM updated with real-time information. It keeps track of already scraped information to avoid duplication. All accumulated JSON reports from different data sources are then presented to data extraction module. Though we collected 70,402 threat indicators and analysis reports, it only constitutes a small part of collective threat intelligence available.

3.3.2 Data Extraction Module

After the collection of a dataset, the next step is to extract relevant information from the analysis report and store it into the knowledge graph. We followed a similar approach as mentioned in [25] to store threat indicators data into the knowledge graph. In this module, relevant data extractor parses analysis report to understand the behavior of attribution and establish the relationship between them. It uses indicator type and behavior to determine the relationship between them. It also extracts information in a set of tuple to conveniently store it into the knowledge graph (a graph Database).

S.No	Table Name	Purpose
1	pulse	Holds data about all the pulses (aka indicators of compromise) that we receive from the OpenSource communities such as description, creation time, author information etc.
2	indicator	Holds data about all the indicators that we receive from various pulses such as indicator value, type, description etc.
3	tag	Holds data about all the tags that we receive from the pulses such as malware, dropper, trojan.
4	pulse_indicator	Holds indicator and pulse mapping to establish interconnectivity between pulse and indicators.
5	pulse_tag	Holds tag and pulse mapping to establish interconnectivity between pulse and tags.
6	entry	Holds information about the last pulse fetched by the crawler.

Table 3.1: The different tables in our database.

Relevant Data Extractor

Each threat intelligence system is autonomous, and it only cares about analyzing certain types of data. Each autonomous crawler provides such analysis report in JSON format where it includes intelligence information, attribution details, and threat intelligence system information. Even though each standard threat intelligence system may have a structured format, its structures and attributes vary. Therefore, making a generic structure regarding representation of knowledge of threat indicators is required. To address this issue, relevant data extractor abstracts each analyzed report as a set of structured tuples. This store process is influenced by [25]. It established such relationship based on the type of incoming and outgoing threat indicators and their reported behavior. To understand structured tuple format, let us take examples.

$$\langle \texttt{"windows-exe"}, \texttt{"ip"}, \texttt{"has-associated-ip"} \rangle, \tag{3.1}$$

$$\langle \texttt{"url"}, \texttt{"windows-exe"}, \texttt{"has-detected-sample"} \rangle \tag{3.2}$$

Line (1) says if we take a windows executable file as an input, and outputs an IP address where the executable file contains IP address in the string of an executable. The relationship between the input and output remains as "has-associated-ip" due to uncertainty towards the context of behavior. Line (2) says if we found a malicious windows executable file gets downloaded from a url, our system can report the relationship between the domain and file as "has-detected-sample". Obviously, a comprehensive analysis system will be abstracted as a set of hundreds or even thousands of 3-tuples like this [25].

Table 3.3 provides more information of such tuples which CSM utilizes to establish the relationship between two indicators.

Input Type	**Output Type**	**Relationship**
Domain	Domain	HAS-ASSOCIATED-DOMAIN HAS-OTHER-DOMAIN
Domain	Email	HAS-ASSOCIATED-EMAIL
Domain	IP	HAS-IP
Domain	Hash	HAS-DETECTED-SAMPLE HAS-ASSOCIATED-SAMPLE
Email	Domain	HAS-OTHER-DOMAIN
URL	IP	HAS-IP HAS-ASSOCIATED-IP
URL	Domain	HAS-DOMAIN
URL	Hash	HAS-DETECTED-SAMPLE HAS-ASSOCIATED-SAMPLE
URL	Email	HAS-ASSOCIATED-EMAIL
IP	Domain	HAS-DOMAIN
IP	Hash	HAS-DETECTED-SAMPLE HAS-ASSOCIATED-SAMPLE
Hash	Domain	HAS-ASSOCIATED-DOMAIN HAS-COMMUNICATING-SAMPLE HAS-DETECTED-DOMAIN
Hash	IP	HAS-ASSOCIATED-IP HAS-COMMUNICATING-SAMPLE HAS-DETECTED-IP
Hash	URL	HAS-ASSOCIATED-URL HAS-COMMUNICATING-SAMPLE HAS-DETECTED-URL
Hash	Email	HAS-ASSOCIATED-EMAIL HAS-DETECTED-EMAIL

Table 3.2: Indicator types and relationship between them.

Relationship	**Context**
HAS-ASSOCIATED-DOMAIN	Associated sub-domain Connects to Analytics associated domain Adsense associated domain
HAS-ASSOCIATED-EMAIL	Has admin email Has registrant email Contains
HAS-OTHER-DOMAIN	Other TLD Owns
HAS-IP	Owns
HAS-ASSOCIATED-IP	Contains Connects to
HAS-DOMAIN	Maps to
HAS-DETECTED-SAMPLE	Downloads
HAS-ASSOCIATED-SAMPLE	Downloads
HAS-ASSOCIATED-DOMAIN	Contains Connects to
HAS-ASSOCIATED-URL	Contains Connects to
HAS-DETECTED-EMAIL	Attachments
HAS-COMMUNICATING-SAMPLE	Communicating
HAS-DETECTED-IP	Downloads from
HAS-DETECTED-DOMAIN	Downloads from
HAS-DETECTED-URL	Downloads from

Table 3.3: Relationship and their context.

Knowledge Graph

The knowledge graph stores collected intelligence into a graph. Given the heterogeneousness of the intelligence and the systems that generate them, the knowledge graph has to handle, store and represent knowledge containing information from various threat intelligence systems in a suitable manner so that a holistic picture of heterogeneous intelligence can be painted.

At a high level, the knowledge graph can be viewed as a system that collects and stores the intelligence in a structured format. CSM stores it into the graph format, where each vertex is a threat indicator that includes some information about it, such as the type, the value, and other properties. Each edge in this graph is labeled with a relationship and is assigned propagation factor to each relationship.

When parser extracts information about threat indicators, it also establishes local confidence or prior knowledge of the threat indicator. There are two ways CSM can determined prior knowledge of a threat indicator. 1) Set by experts - The prior knowledge of a threat indicator can be set by domain experts based on their experience and expertise. 2) Set through analysis reports - CSM uses reported analysis to determine the behavior of the threat indicator. It then cumulates the opinion about the threat indicator from various threat intelligence systems and sets a threshold value to determine whether the given threat indicator is malicious or not. If more than 5 analysis system reports an indicator as malicious, then local confidence of an indicator is set as malicious. This intuitively set threshold for CSM can be adjusted by the analyst. However, we believe there is a scope of improvement in determining prior knowledge of a threat indicator.

3.3.3 Confidence Score Calculation Module

After the data is stored into the knowledge graph, the next step is to propagate maliciousness to the adjacent nodes based on the behavior reported by analysis systems. The confidence score calculation module is responsible for propagating the maliciousness to determine the confidence of a threat indicator. Each vertex in the knowledge graph determines the category (type) and the relationship between them is assigned based on the reasoning and context about an existing or evolving cyber-attack. Each relationship is linked with a propagation factor that can be set by an expert in the domain or assigned based on prior experience, usability tests and surveys or through machine learning algorithms. The confidence score calculation module uses a proprietary algorithm to determine the global confidence of the indicator from the local confidence score, and the behavior spreads from the adjacent nodes. More information on the algorithm is described in section 4.5. If the threat intelligence system is uncertain about the behavior, in such cases, the indicator generalizes the relationship. The starting indicator propagates lowest possible weights to its neighbors in an uncertain situation. However, if the behavior provides enough reasoning, context, and implications of maliciousness, the indicator propagates highest possible weights to adjacent neighbors.

Table 3.4 refers the propagation factor associated with each relationship in CSM.

3.3.4 Relevance Score Calculation Module

The relevance score calculation module of CSM provides a way to extract relevant information from the knowledge graph. It is determined by the com-

Relationship	**Propagation Score**
HAS-ASSOCIATED-DOMAIN	0.3
HAS-ASSOCIATED-EMAIL	0.3
HAS-OTHER-DOMAIN	0.3
HAS-IP	0.6
HAS-ASSOCIATED-IP	0.3
HAS-DOMAIN	0.6
HAS-DETECTED-SAMPLE	1
HAS-ASSOCIATED-SAMPLE	0.3
HAS-ASSOCIATED-DOMAIN	0.3
HAS-ASSOCIATED-URL	0.3
HAS-DETECTED-EMAIL	1
HAS-COMMUNICATING-SAMPLE	0.5
HAS-DETECTED-IP	1
HAS-DETECTED-DOMAIN	1
HAS-DETECTED-URL	1

Table 3.4: Propagation factor assigned to relationships.

bination of search criteria and confidence score of the threat indicator. Considering the analyst may not be interested in the entire result set from the knowledge graph, CSM has to narrow down the scope of the results based on the analyst-driven input. An application can take analysts' input and search related information in the knowledge graph based on the search term and filter criteria, prioritize the output based on relevance score and retrieve the results. More information on the approach is being described in section 5.2.

3.4 Design Issues

This section will describe the issues we faced with the design decisions we made, and how we did our best to mitigate them, and their effect on the system.

- **Knowledge Representation**

  All discovered and collected intelligence from the analysis reports is stored in a graph database instead of a relational database like MySQL, as our dataset represents well in the form of connected data and majority of the time, an analyst is interested in close neighbors from a particular node. In graph databases, data connections take priority, hence the application doesn't have to infer relations through multiple join queries, which improves performance significantly.

- **Speed Up the System**

  To benefit from parallel processing and speed up the system, we implemented crawler with threading and a queuing system where each thread processes data from a global queue which holds crawled report and parses it to put the tuple formed data into the knowledge graph. To speed up the insertion of the records, we also inserted data using bulk insert.

- **Individual Crawler-Parser Construction**

  CSM contains several threat intelligence feeds to crawl information for threat indicators. Whereas these feeds provide additional information about the behavior or newer intelligence through analysis, each feed provides intelligence in its own format. In order to standardize the format, each crawled data requires to have specific parser based on response from the threat intelligence feed. We modularized the system in such a way that each plug and play crawler and parser generates the final result into the standard 3-tuple format which was described 3.3.2.

- **API Shortcomings**

  We found that some APIs do not provide a detailed result which is shown during visiting the website and providing the same parameters. To deal with this issue, we converted such approaches from the API crawling-based approach to the web scraper-based approach. We used libraries such as BeautifulSoup and Selenium web driver to scrap the details of the fields in which we are interested.

- **Bot Blockers**

  Because our data collection is fully automated, it is also susceptible to being stopped by 'bot-blockers' i.e. mechanisms built-in to a website to prevent automated crawls. Measures like CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) are often used to detect bots [30], [49].

  We did not have an anti-CAPTCHA functionality built into our system, in such cases, we requested permission to have higher usage of such resources for research purpose.

- **Handling Malformed or Erroneous Response of API**

  The parser assumes to have valid JSON object and it does not try to parse malformed JSON, and throws an exception on encountering malformed content. Thus, we have designed the system to exit gracefully on such occasions. A side-effect of this is that our system is unable to parse JSON object which contains bad markup.

- **Limitation of Crawling Websites**

  In contrast to bot blockers that try to prevent the automated systems from attacking them, some APIs define the upper limit on the number of queries that can be performed to limit the resource utilization. To honor the limit on utilization of such resources, we added sleep time in our queries to restrict our queries to the upper limit.

3.5 Assumptions

We made certain assumptions while building the system. This section describes the assumptions and explores the extent to which these hold true:

1. **Crawler is not blocked by firewalls**

   This is a requisite for our system to work. If the crawler is blocked for any reason, we do not get the data for our system, and without this input, it is almost impossible to set our system up. Having said that, we have adjusted our crawlers to honor daily quota limit of various threat intelligence feeds to not affect the performance of such services.

2. **Crawler feed is an ideal representation of the threat intelligence feed**

   We have collected a reasonable amount of data from the open source systems where more than 25000 security researchers have subscribed. However, considering the large magnitude of heterogeneous threat feeds will help us in representing the reliable sample of the entire population. A crawl of this large magnitude should give us a very distributed sample of the overall Threat indicators, eventually converging to the average of all threat indicators in existence.

3. **Weights assigned to calculate relevance score at various stages in the process represents the justifiable ideal scenarios**

   We assume that based on analysis reported by threat intelligence systems, each system requires assigning a different level of confidence. Further, reported indicators themselves show a variety of behaviors leading to assigned set of relationships. Each relationship conveys a weighted propagation factor that helps us to calculate a final confidence score of threat indicators.

   We believe that this is a reasonable assumption, as we tried to assign weights in a justifiable manner.

That concludes our discussion about the design of the system. To recap, we discussed our approach, the system architecture and how the components fit into our architecture. We also discussed the issues faced, and the assumptions that we made while building the system.

Chapter 4

PROPAGATION ALGORITHM

In this chapter, we discuss the logic behind adding the propagation module. Then, we compare the existing propagation algorithms and their shortcomings for CSM. Then, we discuss the algorithm design using various scenarios and explain the algorithm in detail.

4.1 Motivation

In confidence building we deal with shades of gray rather than simple black and white. Hence, determining the confidence level of our assessment of an entity’s reputation is essential. Security professionals rely on such confidence scores to make effective policy decisions based on known probabilities and score generated by confidence score measurement system. The more dimensions we take into consideration when calculating a score, the higher our confidence. A useful analogy could be putting together pieces of a puzzle. The more pieces are joined, more likely we are to guess the correct outcome of the puzzle. To reach a high confidence level, we may need to look at a dozen dimensions that, on their own, dont́ tell us much but, correlated with each other, offer us high confidence. However, many times, data for such dimensions are not available, and in the absence of that, we need to focus on correlation factor of the puzzle. We may consider how likely a piece of the puzzle is correlated with another to identify the links between two pieces of the puzzle. Taking the same analogy, we believe indicators when correlated with other indicators, propagate contextual behavior which helps us in propagating confidence throughout such a neighborhood. This belief remains the motivation for propagating confidence score between threat indicators.

4.2 Definitions

Before we look into propagation algorithms, let us understand certain terminology which will be widely used in explanation of those algorithms.

4.2.1 Confidence Score

In simple terms, the confidence score is a value between 0 to 100 assigned to the threat indicator which represents the confidence CSM has in a reported threat indicator being malicious. 0 confidence suggests that CSM is not confident enough to consider the node as malicious. 100 confidence suggests CSM is extremely confident about the malicious behavior of the node.

Let k be a threat indicator. Then let F_u be the set of threat indicators that points to k. Let $\zeta_{k_{(l)}}$ be the local confidence derived from the analysis report from the threat indicator system, then global confidence score of the threat indicator ($\zeta_{k_{(g)}}$) can be determined by summing up the propagated confidence of the neighbors and normalizing it by the normalization factor $\frac{(100-\zeta_{k_{(l)}})}{\sum_i 100}$

$$\zeta_{k_{(g)}} = \zeta_{k_{(l)}} + (100 - \zeta_{k_{(l)}}) * \sum_i \frac{(\zeta_{i_{(g)}} * Max(\psi_{ik}))}{100} \tag{4.1}$$

where

$\zeta_{k_{(g)}}$ = Global confidence score for threat indicator k

$\zeta_{k_{(l)}}$ = Local confidence score for threat indicator k

i = Neighboring node which have outgoing edges to k; i $\in F_u$

ψ = Propagation Factor

The reason that confidence score is interesting is that there are many cases where the simple graph propagation algorithm does not correspond to our common sense notion of importance. For example, if a threat indicator has an incoming link which says that a botnet downloads these samples in the victim machine to communicate, it may be just one link but it is a very important one which makes the threat indicator receive a higher confidence score than many other threat indicators which have more links but with obscure context or links which do not generate enough confidence to determine that given indicator is malicious. However sometimes, we have a threat indicator which may not have enough links due to lack of evidence or an indicator which is analyzed as malicious by threat intelligence system analysis, in such cases, inherent properties of the threat indicator itself helps in determining the higher confidence score. Overall, Confidence Score is an attempt to see how good an approximation can be obtained by combining inherent properties and link structure.

4.2.2 Propagation Factor

Let u,v be threat indicators. Let $E_{u,v}$ be the edge between threat indicator nodes that points from u to v, then propagation factor ψ_u determines the percentage of confidence that node u propagates to node v. The propagation factor is determined by relationship types. Section 4.4 provides detailed scenarios to understand more about the propagation factor and how it can propagate confidence from one node to another.

The propagation factor from u to v can be considered as a dependency of node v on u. If two nodes have an edge between them, then we cannot assume they are conditionally independent, meaning confidence score of the node v cannot be determined before confidence score of node u.

4.2.3 Key Observations

The key observation of CSM is that threat indicators' behavior and determination of maliciousness are not separated; instead, they are closely correlated and interdependent. Through interacting with each other, CSM can determine the confidence that the indicator can impact the neighbors through malicious behavior. At the same time, the threat indicator itself can be influenced by the behavior of its neighbors. For example, if a malicious file's frequent visits are suspicious due to phishing, malware sites, botnet C&C, then these visited threat indicators are likely to receive a higher confidence of being malicious from CSM. Similarly, if a malware is detected that downloads a file, it may also propagate bad influence on that file providing higher confidence on the downloaded file as being malicious. This mutually dependent relationship is more formally known as the mutual reinforcement principle.

Another key observation is that while analyzing threat indicators data, we found a linear ordering of vertexes such that for every directed edge $u \rightarrow v$, vertex u comes before v in the ordering. Through this property, we can reduce the number of iterations required to converge the graph drastically as the confidence score of the follower node v is only determined after the confidence score of the followee u.

4.3 Problem Background

In the section 3.3.2, we have seen that the knowledge graph is represented in graph form, where the graph can be viewed as a network of threat indicators, all broadcasting their activities to their neighbors, that propagate the information to neighboring nodes, which leads to the question of how to measure information spread through the network. We reviewed existing propagation algorithms, and this section describes the various properties and differences of each existing algorithm compared to CSM propagation algorithm.

PageRank

PageRank is a graph propagation algorithm, and it assigns a numerical weighting to each element of a hyperlinked set of documents, with the purpose of measuring its relative importance within the set. It works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites [51]. Some of the shortcomings of PageRank are listed below:

- The main factor used in determining a website's PageRank is the quantity of inbound links and the PageRank of the web pages providing the incoming links. If we apply similar analogy a malicious node has to receive large number of incoming edges. However, there is no guarantee that a malicious node will always receive large number of incoming edges.

- PageRank cumulates the outgoing neighboring probability to 1 by equally dividing or weighing each edge differently. By doing so, it obtains nice properties of stochastic matrix which helps it to converge. Let us take following scenario to understand the shortcoming of above property. In figure 4.1, if M represents a malicious node and N represents a non malicious node then due to the single outgoing connection a malicious node will propagate entire confidence to a non malicious node. So, if PageRank starts with equal initial value for all nodes in the graph, then a

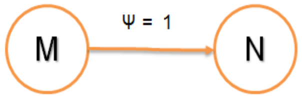


Figure 4.1: An Example Scenario of PageRank and Similarity Flooding Shortcoming.

non malicious node end up receiving higher confidence then a malicious node which remains counterproductive for our definition of the confidence score.

Similarity Flooding

The philosophy behind this iterative method is that "two nodes are similar if their neighborhoods are also similar." It attempts to find the correspondence between the nodes through similar neighborhoods and remains dependent on structural similarities. In this algorithm, humans check whether the matchings are correct, and the accuracy of the algorithms is computed based on the number of adaptations that have to be done in the solutions to get the right ones [18].

Though it is a different problem from ours, it is based on the notion that in each iteration, the nodes exchange similarity scores and this process ends when convergence is achieved.

CSM does not try to match similarity between two nodes; rather it tries to propagate confidence between two nodes through the relationships. CSM does not depend on human intervention except in the case where we determine the weight of the domain-specific relations. CSM does not stack the outgoing propagation to 1 otherwise it will suffer from similar problem as PageRank. It considers propagation to each node independently.

There is one more algorithm which works on the similar line with CSM propagation algorithm.

Belief Propagation

Belief Propagation is an iterative message passing(graph propagation) algorithm to answer conditional probability queries in a graphical model. It requires the subset of the graph as evidence nodes (observed variables E), and compute conditional probabilities on the rest of the graph (hidden variables X) [50]. It has been widely used to solve many graph inference problem [7], such as social network analysis [5], fraud detection [29], and computer vision [11].

Both Belief Propagation and CSM work on peer pressure. Belief Propagation is typically used for computing the marginal distribution for the nodes in the graph, based on the prior knowledge about the nodes and from its neighbors. A node X determines a final belief distribution by listening to its neighbors in Belief Propagation, where nodes in CSM determine local confidence score from the reputation of reporting threat indicator systems and monotonically increase this confidence through propagation from neighbors.

Belief Propagation multiplies message passing from neighboring nodes directly with the prior knowledge of a node. This creates problem if there is not enough context about the established relationship or a non malicious node propagates belief to a malicious node. In both the scenarios, it ends up negating confidence of high confident malicious sample.

In figure 4.2, if M represents a malicious node and N represents a non malicious node then due to the connection from a non malicious node to a malicious node, it will reduce our confidence in prior knowledge of a node being malicious. If a node M with prior knowledge of getting detected by 50 antivirus

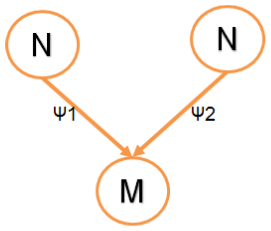


Figure 4.2: An Example Scenario of Belief Propagation Shortcoming.

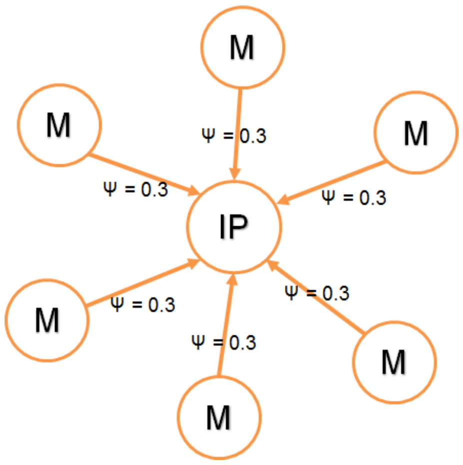


Figure 4.3: An Example of Confidence Score Normalization.

systems receives connection from a non malicious node or receives lower propagation factor due to unknown context, then it will reduce the confidence of the malicious node.

Next, Belief Propagation takes number of iterations to converge to a consensus that determines the marginal probabilities of all the nodes. In graphs with loops, it creates a problem as Belief Propagation may not converge and may perform poorly [18].

Based on above shortcomings, we design and implement a graph propagation algorithm which follows below listed properties.

- CSM takes into consideration node's inherent properties and their prior reported analysis to determine local confidence score of a node. It further determines the propagation factor to propagate confidence from one node to other node. It does this only based on reported behavior independent of considering count of outgoing edges from a node.

- CSM propagation algorithm propagates confidence between nodes based on relationship. Based on the relationship CSM determines propagation factor to spread maliciousness to adjacent nodes. If Two nodes A and B with relationship `connects-to` between them, represents different context than relationship `downloads` between them. Hence, a relationship `connects-to` propagates different confidence than `downloads` between two nodes.

- CSM normalizes propagated score to avoid confidence being saturated at 100 for the nodes having many incoming links. Figure 4.3 represents a scenario to explain why we need to normalize confidence score of a node. Let us say, we have 6 malicious nodes with 100 confidence propagates confidence to an IP address with 0.3 propagation factor. By summing all the incoming confidence, the receiving node, IP address may end up getting 180 confidence. The problem here is even though each node propagates only 0.3 of original confidence, the receiving node saturates confidence at 100. Hence, CSM requires to normalize the propagated score to avoid confidence being saturated at 100 for the nodes.

- To avoid being problem faced by belief propagation represented in figure 4.2, CSM considers propagated confidence score as complement to local confidence score (prior knowledge) of the node. It makes confidence score calculation function monotonically increasing function of local confidence score.

- CSM avoids propagation of confidence in cycle by taking advantage of linear ordering. It calculates confidence of a parent node before calculating confidence of a child node. During cycle, since the parent node is dependent on the child node and vice versa, it avoids calculation for all the nodes participating in forming the cycle. This indirectly solves the problem of nodes getting saturated at 100 confidence in cycle and CSM propagation algorithm converges even if there is a cycle in the graph.

4.4 Various Scenarios

To understand the working of our algorithm, we have orchestrated various scenarios and calculated the confidence score of the node during that scenario. The following images represent the scenarios:

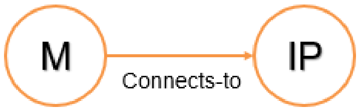
Figure 4.4(a) represents the confidence score calculation between two nodes, which are directly connected via a single edge.

Figure 4.4(b) represents the confidence score calculation between two nodes that have more than one edge connected to them. In such a scenario, the propagation algorithm takes the maximum propagation factor of the incoming edges to propagate the confidence score.

Figure 4.4(c) represents the confidence score calculation where two nodes have different incoming relationships generating from the same source.

Figure 4.4(d) represents the confidence score calculation where a node has multiple incoming edges from different sources.

Figure 4.4(e) represents the confidence score calculation where a node does not have any incoming edge. In this case, the local confidence score of the node becomes the global confidence score.



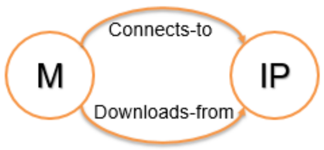
$\zeta_{M(g)} = 100$

$\psi_{Connects\text{-}to} = 0.2$

$\zeta_{IP(l)} = 50$

$\zeta_{IP(g)} = 50 + (100-50) * (0.2*100/100) = 60$

(a)



$\zeta_{M(g)} = 100$

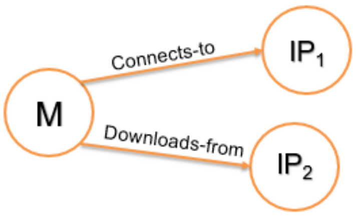
$\psi_{Connects\text{-}to} = 0.2$, $\psi_{Downloads\text{-}from} = 0.4$

$\zeta_{IP(l)} = 50$

$\zeta_{IP(g)} = 50 + (100-50)*(Max(0.2,0.4)*100/100)$

$= 70$

(b)



$\zeta_{M(g)} = 100$

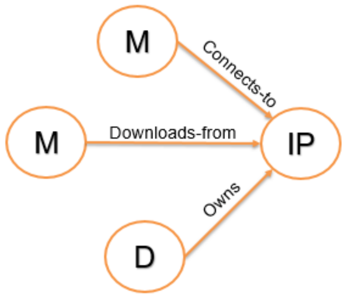
$\psi_{Connects\text{-}to} = 0.2$, $\psi_{Downloads\text{-}from} = 0.4$

$\zeta_{IP1(l)} = 50$ $\quad \zeta_{IP1(2)} = 50$

$\zeta_{IP1(g)} = 50 + (100-50)*(0.2*100/100) = 60$

$\zeta_{IP2(g)} = 50 + (100-50)*(0.4*100/100) = 70$

(c)

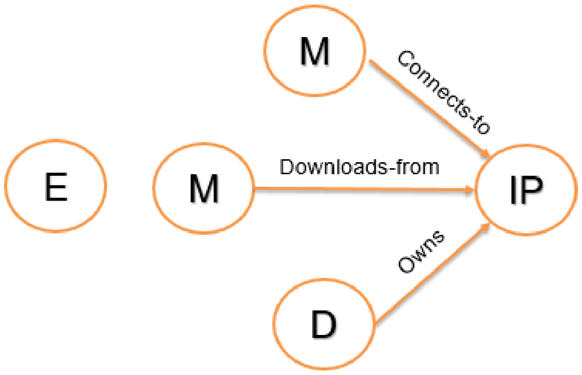


$\zeta_{M1(g)} = 100$ $\quad \zeta_{M2(g)} = 100$ $\quad \zeta_{D(g)} = 30$ $\quad \zeta_{IP(l)} = 30$

$\psi_{Connects\text{-}to} = 0.2$, $\psi_{Downloads\text{-}from} = 0.4$, $\psi_{Owns} = 0.1$

$\zeta_{IP(g)} = 30 + (100-30)*(0.2*100 + 0.4*100 + 0.1*30)/300$
$= 44.7$

(d)



$\zeta_{E(l)} = 50 = \zeta_{E(g)}$

(e)

Figure 4.4: Example Scenarios of Confidence Score Calculation.

4.5 Propagation Algorithm

This section complements Section 4.4, and discusses the propagation algorithm of our project.

In the knowledge graph, information forms a directed Graph G = (V, E) where V is a set of vertexes and E is a set of directed edges. The algorithm presented in Algorithm 1 propagates the confidence score between vertexes in the graph, where each vertex represents a threat indicator. This algorithm accepts, Directed Graph G, as an input parameter. It further uses our observation which states that the knowledge graph represents a linear ordering of vertexes such that for every directed edge $u \rightarrow v$, vertex u comes before v in the ordering. We have implemented a customized topological sorting algorithm to propagate confidence score throughout the graph.

The following steps describes Algorithm 1 in detail:

- Compute in-degree (number of incoming edges) for each of the vertex present in the graph and initialize the set of visited nodes as ϕ.
- Pick all the vertexes with in-degree as 0 and add them into a queue (Enqueue operation).
- Pick all the vertexes $v \in$ V with local confidence score $\zeta_{v_{(l)}}$ as 100, set global confidence score $\zeta_{v_{(g)}}$ of the indicator as 100 and add them into a queue (Enqueue operation).
- Repeat next step until queue is not empty.
- Remove a vertex v from the queue (Dequeue operation).
  Update visited set by adding visited node v.

Algorithm 1 Iterative Propagation algorithm

```
procedure Initialize(V)
    for each $v \in V$ do
        $v$ = The current vertex
        $indegree[v] \leftarrow 0$
        $score[v] \leftarrow \zeta_{v_{(l)}}$          ▷ $\zeta_{v_{(l)}}$ represents local confidence score
        $visited \leftarrow \phi$
    end for
end procedure
procedure ComputeInDegree(E)
    for each edge $u, v \in E$ do
        $indegree[v] \leftarrow indegree[v] + 1$
    end for
end procedure
procedure Enque(V)
    for each $v \in V$ do
        if $indegree[v] = 0$ then
            add $v$ to queue
        else if $\zeta_{v_{(l)}} = 100$ then
            $\zeta_{v_{(g)}}$ = 100
            add $v$ to queue
        end if
    end for
end procedure
procedure Enque($v$)
    add node $v$ to queue
end procedure
```

Algorithm 1 Iterative Propagation algorithm (continued)

```
procedure Propagate(Directed Graph G(V,E))
    call Initialize(V)
    call ComputeInDegree(E)
    call Enque(V)
    while queue is not empty do
        Remove a node $v$ from the queue.
        $visited.update(v)$
        calculate $\zeta_{v_{(g)}}$ based on $\zeta_{u_{(g)}} \in F_u$
              ▷ $F_u$ be set of the adjacent vertexes $u$ that has outgoing edges in $v$
        for each node $w \in F_w$ do
         ▷ $F_w$ be set of the adjacent vertexes $w$ that has incoming edges from $v$
            $indegree[w] \leftarrow indegree[w] - 1$
            if $indegree[w] = 0$ and w is unvisited then
                call Enque($w$)
            end if
        end for
    end while
end procedure
```

Calculate confidence score of the node v based on incoming neighboring nodes using equation 4.1.

Decrease in-degree by 1 for all its neighboring nodes.

If in-degree of a neighboring nodes is reduced to zero, then add it to the queue.

- At the point, when the queue becomes empty, the graph converges.

If in the future a new node is joined, a new edge is discovered or the weight of the node is changed, then the algorithm can work only on that subgraph to accommodate changes.

If an edge(u,v) is added, the confidence score of node v gets recalculated. These changes further propagated into the graph, where v propagates updated confidence score to nodes that have incoming edges from v. This propagation continues until the queue of updated node becomes empty.

If a node u changes its weight after the calculation of the confidence score, then the algorithm determines the outgoing edge(u,v) from the node u and propagates the updated weight to all neighboring nodes $v \in$ V, which has an incoming edge from node u. These changes propagated into graph until the queue of updated node becomes empty.

Chapter 5

RELEVANCE SCORE CALCULATION

After the confidence score propagation, the knowledge graph receives global confidence score of each node. However, an analyst still has to deal with all the reported and collected indicators. In this chapter, we introduce the technique to narrow down the scope of the results by combining search query with the confidence score of a threat indicator. We explain a full-text search technique used in the narrowing down the results of search query and the workflow of calculation of relevance score in search application of the CSM.

5.1 Background

During our analysis, we found that every threat pulse (set of indicators), when reported provides title, description, and tags for the group of threat indicators. These attributes linked with reported threat indicators give us information about the event that occurred or the operating system for which these indicators were detected, the location at where this incident was reported, etc. We found this great deal of information, when combined with the previously calculated confidence score, helps the analyst to perform relevant queries quickly. This section provides background on the technique used in the full-text search to filter and prioritize threat indicators based on the search term used by the analyst.

CSM uses the Query Model to find matching threat indicators, and applies boosting and filtering techniques in real time as the practical scoring function to calculate relevance. This formula borrows concepts from term frequency/inverse document frequency (TF/IDF) and the vector space model but adds more modern features like a coordination factor, field length normalization, and term or query clause boosting.

[42] provides description on the full-text search technique used in the CSM. These full-text search techniques can be used with Boolean Model or Fuzzy Model where the former simply applies the AND, OR, and NOT conditions expressed in the query to find all the terms that match, and the later calculates the edit distance to match relevant terms that are within the maximum edit distance specified in fuzziness. As both models rely on an underlying concept of TF/IDF, this section provides an overview of this technique.

5.1.1 Term Frequency/Inverse Document Frequency (TF/IDF)

When the analyst searches for the result, the algorithm searches for the list of matching indicators and ranks them by relevance. Not all indicators will contain all the terms, and some terms are more important than others. The relevance score of the indicator depends on the weight of each query term that appears in that indicator[42].

Term frequency

Term frequency (tf) stands for how often does the term appear in this indicator. The term frequency is calculated as follows:

$$tf = \sqrt{f} \tag{5.1}$$

The term frequency for term t in document d is the square root of the number of times the term appears (f) in the indicator attributes such as title, description, tags, etc.

Inverse document frequency

Document frequency stands for how often does the term appear in all the indicators in a collection. Inverting such document frequency reduces the importance of common terms and increases for uncommon terms[42]. The inverse document frequency (idf) is calculated as follows:

$$idf = 1 + log(ni/(nc + 1)) \tag{5.2}$$

The inverse document frequency of term t is the logarithm of the number of indicators in the index (ni), divided by the number of indicators that contain the term (nc).

Field-length norm

Field length norm normalizes the weight assigned to the term based on the length of the field. If a term appears in a short field, such as a title field, it is more likely that the content of that field is about the term than if the same term appears in a much bigger description field[42]. The field length norm is calculated as follows:

$$norm(d) = 1/\sqrt{nt} \tag{5.3}$$

The field-length norm (norm) is the inverse square root of the number of terms in the field (nt).

These three factors term frequency, inverse document frequency, and field-length norm are used together to calculate the weight of a single term in a particular indicator.

Vector Space Model

The vector space model provides a way of comparing a multiterm query against an indicator.

5.2 Relevance Score Calculation Application

In this section, we present the skeleton of the application that displays results based on relevance. As mentioned in the background, the search application lets an analyst specify the search term, applies full-text search techniques on nodes and relationships in the knowledge graph, combines with a confidence score and returns search results on the graph based on human analysts' inputs.

The searcher application provides a platform for human analysts to carry out research queries, allowing users to find what they are looking for very quickly, visualize relationships between heterogeneous data nodes and understand how they are correlated to get the insight of data, which is not eminent by just looking at the data.

Like existing search engine functionalities, the search application also serves results in real-time, sorted by decreasing relevance. However, apart from a search based on a string of characters, an analyst may also be interested in advance search such as a search based on category, time, etc. Hence, the application requires to have filtering parameters to provide personalized search results. Specifically, we provided following search techniques:

- **Combined Search**

  This query will apply conjunction to the words appear in the search term to find all the terms that match the partial or full input and return results.

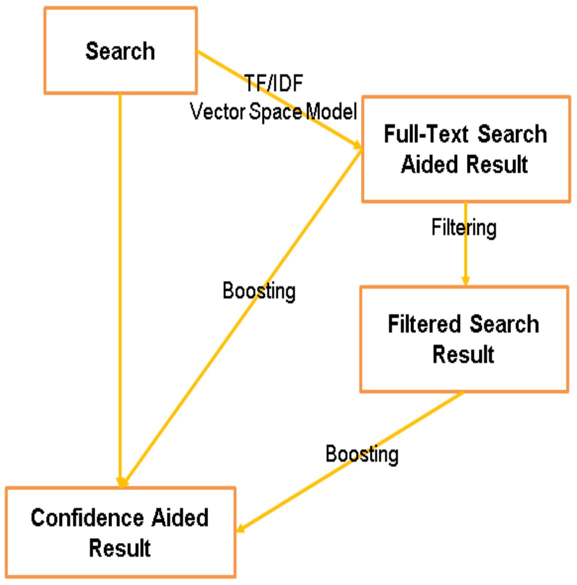


Figure 5.1: Full-Text Search Workflow.

- **Exact Search**

  This query will return results for the exact search term provided as input.

- **Category Filter**

  This query can be used very effectively to generate filtered results by restricting the search scope to the specific category such as IP, Domain, Hash, etc.

- **Time Filter**

  This query can be used very effectively to generate filtered results by restricting the search scope to the time range such as last 6 months, last 1 month, last 7 days, etc.

Further, the position of the word appearing in the collection derives different weights due to the significance of position. For example, a term appearing in the tag or title section provides direct relevance compared to the term that appears in the description section. Considering the same analogy, we need to assign different weights to different fields to establish the significance of the one field over other. It requires a full-text search engine to boost the fields, which are more important than others.

Hence, the application is required to build techniques on top of a full-text search engine, which allows the analyst to filter, order, and boost the results based on field weights. Figure 5.1 provides an overview of the entire workflow of the full-text search process. When an analyst does not provide search term, results are generated only based of the confidence score. If analyst provides a search term, the application leverages TF/IDF based full-text search technique, and calculates the results from graph data. If filtering parameters are provided, the application further filters the results based on selected parameters such as category and reported time. It calculates relevance score by boosting resulting indicators' full-text search score based on confidence and field weights and generates search results. The final result is returned to the analyst in the decreasing order of the relevance score.

Chapter 6

SYSTEM IMPLEMENTATION

This chapter describes the experimental setup for our project including the servers used, the software and platforms involved, and the languages used. We follow this up with our evaluation of the system, with a result section to describe our findings from a sample application.

6.1 System Configuration

We used a single system for the project, and its configuration is as follows:

- **Dell OptiPlex 9020**

  ```
  CPU: Intel Core i7 @ 3.6 GHz
  Cache size : 8 MB
  No. of Cores : 4
  Total Memory (RAM) : 16 GB
  Disk Space : 1 TB
  ```

6.2 Platforms and Software

We enumerate the platforms and the software used for our project in Table 6.1.

6.3 Languages Used

We used Python 2 and Java 8 to build the system. The following factors influenced our choice of language: Neo4j traversal capabilities specifically built

Operating system	Windows 7 Enterprise
Server	Apache Tomcat - 7.0.75 Flask Built-in Server
Database	Neo4j - 3.1.0 MySQL - 5.7.16
Other software used	ElasticSearch, PostMan, Redis

Table 6.1: Platforms and software used for our project.

in JAVA, Full-Text Search Integration Support and the numerous libraries for HTML Parsing, HTTP request generation, etc. We made use of the following major libraries (shown in Table 6.2) for our system.

Library	**Functionality**
Httplib Urllib2	HTTP Request Generation Url Reuqest Generation
Beautiful Soup Selenium	HTML Parsing HTML Parsing
Re	Regular Expression Operations
Py2Neo	Graph Database Support
Neo4j-driver	Graph Database Support
ElasticSearch	Full Text Search Operations

Table 6.2: Libraries that we used and their functions.

6.4 Implementation

We implemented a prototype CSM framework that integrates various open source analysis platforms including the VirusTotal, ThreatCrowd, ThreatMiner, PyWhoIS, etc. The autonomous crawling and analysis systems were implemented in Python language. These scripts were built using Python-based libraries like Beautifulsoup, and Selenium webdriver. The results from the crawler were stored in MySQL relational database and file reports. The Relevant Data Extractor parsed data using Python based library such as re for information extraction. This preprocessed information is converted into a structured tuple format and is stored as indexed data with source based identification. As to avoid storing duplicate data, we implemented unique indexing to de-duplicate data. All discovered intelligence is stored in a Neo4j graph database instead of relational database systems like MySQL since it has been found that graph databases work well on highly connected data. A propagation technique is developed to propagate maliciousness behavior between threat indicators throughout the graph to determine confidence score. This propagation algorithm developed in Java as to leverage traversal API of Neo4j and to easily parallelize the propagation in future. A web-based application was developed to show the potential of the system where we integrated full-text search technique to provide flexibility of searching to the analyst through various filters. It was implemented using Flask framework, Python, D3.js which queries to interface to provide required filtered and prioritized data.

Chapter 7

DATA ANALYSIS AND RESULTS

This chapter serves to present our findings. It reports various observation based conclusion to evaluate the effectiveness of the CSM. It describes comprehensive and analytical evaluation results for our system to demonstrate the feasibility and scalability of our approach.

7.1 Collected Data

At the time of writing, the CSM knowledge graph consists of more than 70,402 nodes from heterogeneous data sources and more than 51,418 relationships between nodes. Our perception of collecting more data concludes that if each received threat indicator also serves as a data point influencing the overall confidence score measurement, more data translates to more evidence to establish a higher degree of confidence in the system. Table 7.1 shows information about the gathered data. It confirms our data remains well-balanced and heterogeneous in nature to perform our analysis.

S.No	**Type of Data**	**Quantity**
1	Network Indicators (IP-Domain-URL)	49329
2	Other Indicators (Email)	203
3	System Indicators (File-Hash)	20870

Table 7.1: Collected Node Data.

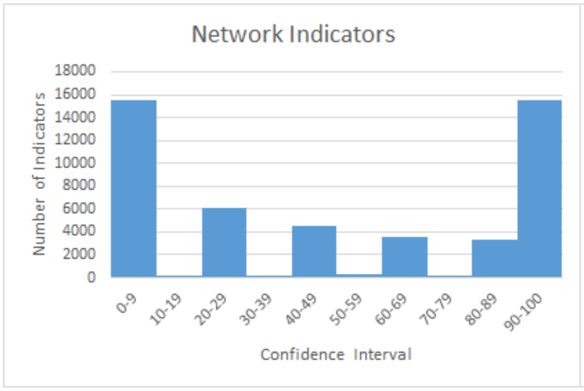
7.2 Analysis of Threat Indicators

Next, we wanted to analyze threat indicators based on category. We divided overall indicators into network-centric, system-centric indicators, and other indicators. Network-centric indicators may include IP, domain, URL, while system-centric indicators may include registry entries, file hashes, etc. Remaining indicators may belong to the other category. We applied the propagation algorithm with default propagation factors to the graph, and we tried to decipher the results.

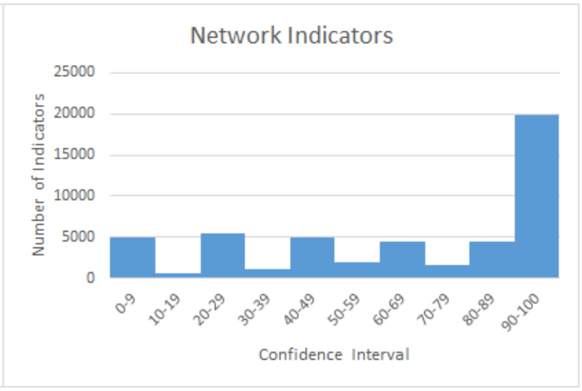
In network-centric indicators (IP-Domain-URL), we found that the confidence of indicators spread across the range from 0 to 100 and the distribution can be represented as multimodal distribution. The important point is 8.9% more indicators of overall network indicators were successfully able to derive confidence between 90-100 through propagation shows effectiveness of the propagation algorithm in network indicators. Figure 7.1a and 7.1b reveal more information about determined confidence score for network indicators.

System-centric indicators (files-hash) provides bimodal distribution of indicators where majority of indicators derive confidence between 0-9 or 90-100. Through propagation we have seen 0.8% more indicators of overall system indicators were successfully able to derive confidence between 90-100. This indicates, system-indicators have enough prior knowledge about threat indicators being malicious or not. Figure 7.1c and 7.1d reveal more information about determined confidence score for system indicators.

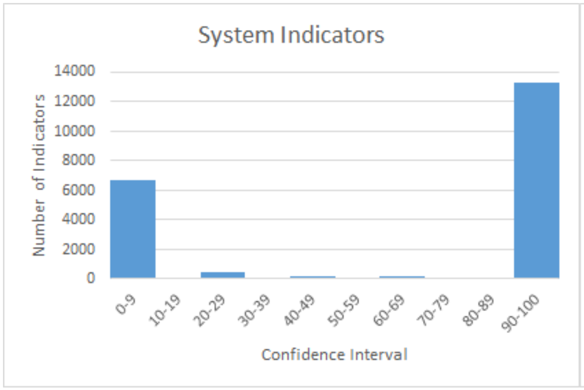
Plenty of other indicators such as email address can be considered as a network or system indicator based on whether they are the result of whois information linked with malicious domain or derived during dynamic analysis of



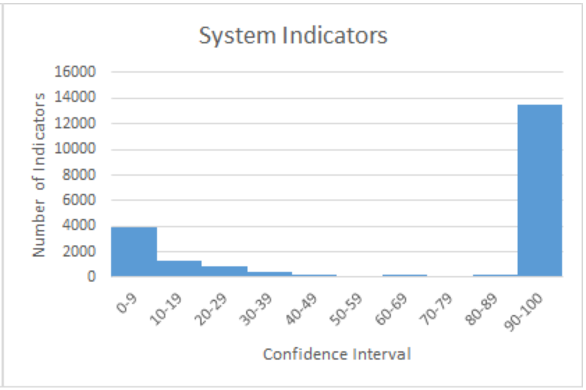
(a) Local Confidence spread of Network Indicators



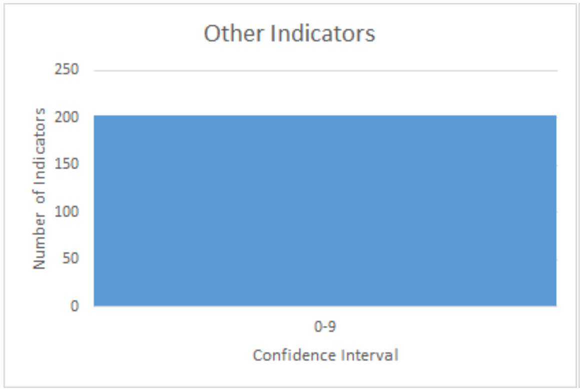
(b) Global Confidence spread of Network Indicators



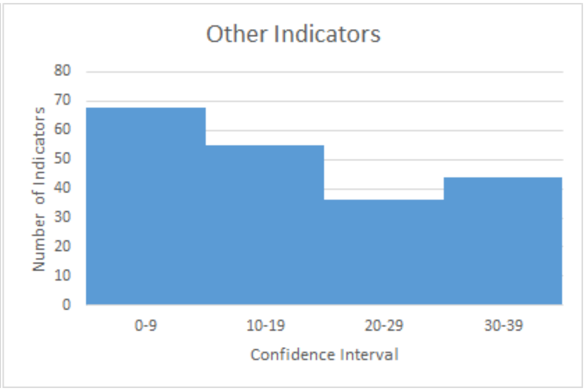
(c) Local Confidence spread of System Indicators



(d) Global Confidence spread of System Indicators



(e) Local Confidence spread of Other Indicators



(f) Global Confidence spread of Other Indicators

Figure 7.1: Threat Indicator Analysis Based on Category.

malware. We categorized such indicators into the other category. For other indicators, we did not have any prior knowledge, hence their prior knowledge remain 0. With the propagation algorithm and whois information, they were able to receive generalized context, which spread the confidence score of indicators in the range from 0-39. Figure 7.1e and 7.1f reveal more information about the determined confidence score for other indicators.

Based on the above observation, we conclude that majority time system-centric indicators have enough prior knowledge about threat indicator. We can confirm it through bimodal distribution of indicators. Propagation of confidence creates minor impact on system indicators. On the other hand, network-centric indicators receives major impact through propagation as 8.9% network indicators successfully derive confidence through propagation. The wide spread of network indicators confirms network indicators derive gray confidence. The reason for gray confidence could be some indicators either provide too generic context to deduce any concrete conclusion from their behavior or they are associated indicators that have participated in the event, yet do not perform any malicious actions. Hence, they propagate lower confidence towards being malicious.

7.3 Accuracy

To measure the accuracy in terms of percent false positives (%fp) and percent false negatives (%fn) we collected 130 threat indicators that are malicious and benign. For the non-malicious dataset, we selected Alexa top-20 domains and their IP address by assuming them to be non-malicious. We further selected 19 windows executables from `originaldll.com` [43] to make our dataset heterogeneous. For a malicious dataset, we used `virusshare` dataset [35].

We further used `malwaredomains` to accumulate malicious domains and IP address [21] [22]. Figure 7.2 shows how CSM algorithm fares with selected dataset. Non-malicious data received the confidence score between 0 to 64. The higher confidence was due to malicious indicators are communicating with the non-malicious domains and their IP address. We believe with more detailed context, such as the analysis of communication between threat indicators, can further improve the decision-making process of calculating confidence score. The majority of malicious data resided in the range from 60 to 100. However, some of the malicious threat indicators failed to provide enough context to receive high confidence on indicator being malicious. To statistically measure how the CSM performed, we considered percent false positives and percent false negatives. Percent false positives and percent false negatives can be formally defined as follows. Let μ_r be the number of non-malicious entities reported as malicious entities above β and let μ_o be the number of malicious entities in the original dataset then percent false positives is given as

$$\%fp = \frac{\mu_r}{\mu_o} \tag{7.1}$$

Let μ_{nr} be the number of malicious entities reported as non-malicious entities below β and let μ_o be the number of malicious entities in the original dataset then percent false negatives is given as

$$\%fn = \frac{\mu_{nr}}{\mu_o} \tag{7.2}$$

If threshold β is set too high, many hacker actions are missed and if β is set too low, too many entities are above the threshold β, indicating inaccurate threat predictions. Examining both $\%fp$ and $\%fn$ seems to be essential to de-

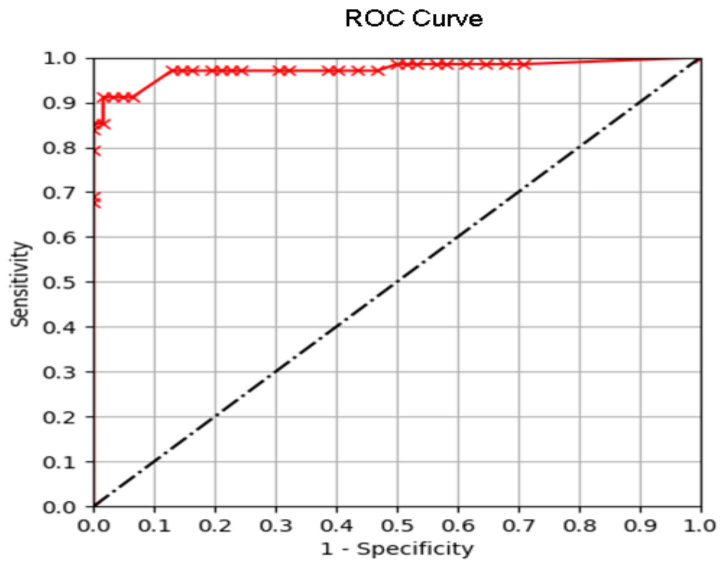


Figure 7.2: ROC Curve of an Example Dataset.

termine the accuracy of an algorithm. From the 7.2 we found, we can take β between 60-70 to get more accurate results. With the β at 65, we received $\%fn$ as 14.7% and $\%fp$ as 0%. If the number of indicators in a dataset is given as N then accuracy (accr) is given as

$$accr = \frac{N - (\mu_{nr} + \mu_r)}{N} \tag{7.3}$$

Accuracy for the CSM algorithm stands at 92%. The area under curve (AUC) remains 0.9736.

7.4 Scalability

To compare the scalability of CSM propagation algorithm, we collected 5 months of data to analyze the performance of the algorithm. We iteratively increased the size of the graph in terms of the number of nodes and edges to measure running time of the propagation algorithm. Figure 7.3, in which x-axis

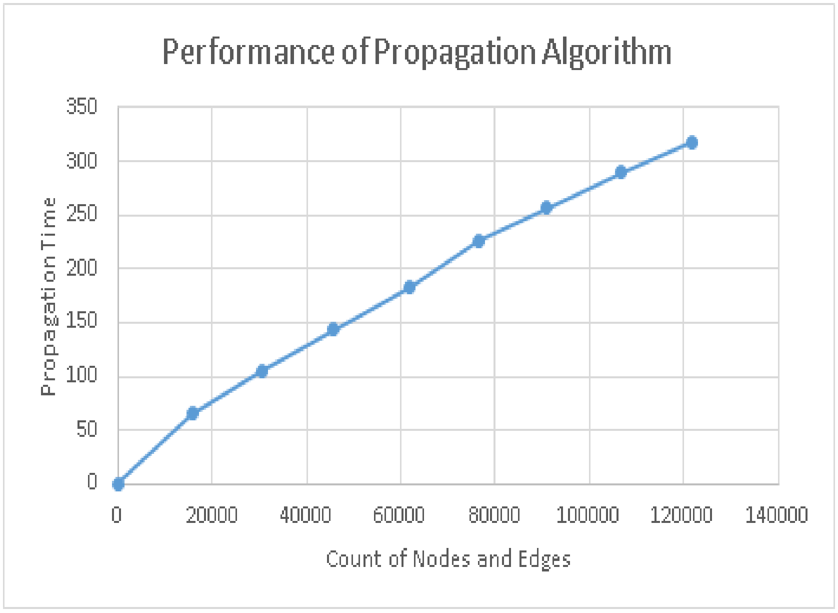


Figure 7.3: Propagation Time with Varying Size of the Knowledge Graph.

indicates number of nodes and number of edges in the graph and y-axis shows the propagation time of the algorithm. At peak of 5 months of data, the propagation algorithm took 318 seconds to propagate confidence score throughout the graph.

As shown in Figure 7.3, the running time of the algorithm linearly increased, which shows our algorithm can scale very well with large graphs. Further, given the data is organized in small clusters, propagation algorithm can be distributed and executed in parallel, providing more scalability by reducing the processing time of the algorithm.

7.5 Justification for Assigned Weights

Next, we wanted to verify the impact on the results and justify our assigned weights. We calculated the confidence score of threat indicators based on weights provided in 3.4. These weights were intuitively assigned based on the behavior from the description. However, we may derive such weights from the inputs of domain experts based on their prior experience, usability tests, and surveys or machine learning algorithms through collecting enough data.

Anything that screamingly tells about malicious behavior of the end indicator receives the relationship *DETECTED*. For example, an email attachment using exploitation of trust to lure to phish for sensitive data, distribute malware, promote scams, generate revenue from ads on parked domains, and drive monetizable traffic to other site considered as a malicious behavior receives has-detected-email relationship, which propagates 100% confidence.

A report from [44] suggests that from overall domain infringements, 26% infringements took place through subdomains. Further, through techniques such as domain shadowing, they noticed the risk of subdomain associated with malware and phishing increases 10 fold. This leads us to believe associated subdomains prove to be risky and requires to propagate 25-30% of confidence from parent domain. Some may argue to consider weighting through $total\ number\ of\ detected\ subdomains/total\ number\ of\ subdomains$. However, there are technical challenges, which require vast infrastructure and technical investment to collect large enough repository of data to be useful for this purpose [44]. Further, these domains maliciousness change over time and they require continuous adaptation of newer information and lastly, we need to restrict our scope to the reported subdomains. This lead us to believe for associated subdomains, the best way to provide propagation factor is to take the proportionate a subdomain involved in exploiting system compare to a domain.

A report from Cisco [16] reports domain registration relationship with whois information. The research concludes that registrant email address was associated with a mix of malicious and benign domains and some registered domains had no email addresses. Moreover, there are many instances where people have used altered or fake contact information when registering their domains as a step to ensure their privacy. These force us to put such whois information to the generalize category where there are further efforts required to verify the accuracy of registered details. Hence, we propagate 25-30% confidence to HAS-ASSOCIATED-EMAIL or HAS-REGISTRANT-EMAIL relationships.

[32] provides insights about various TLDs with their reputations for spam operations and the domains that offered free, greatly discounted and 2 for 1's are the top source for spamming according to this list. As there was no information available about the size of the individual TLD, we decided to take the median of the dataset after cleaning the TLDs with 0 score. To our surprise, we received the mean for such an entry as 5.40 and a median as 1 for 246 such TLD entries. Apparently, we decided to propagate the same confidence as to other generalize situation, which is 30% confidence for HAS-OTHER-DOMAIN relationship. However, it would be more prudent if we had assigned different propagation factor to each TLDs as such .science TLD scores 93.3% bad reputation in Spamhaus list, that might have been proved as more appropriate approach.

[15] reported a detailed observation of ASes (Autonomous systems) indulge in hosting malicious activities due to lax security measures. They reported more than 65% ASes had at least 10% but less than 50% of their IP addresses blacklisted. More precisely, 25% ASes had 20-30% of their IP addresses blacklisted, which remains the mode for the given ranges. Hence, we determined to use 20-30% confidence for HAS-ASSOCIATED-IP relationships. Overall, a relationship which represents generalized behavior for threat indicators was represented through *ASSOCIATED* and they achieved propagation between 20-30%.

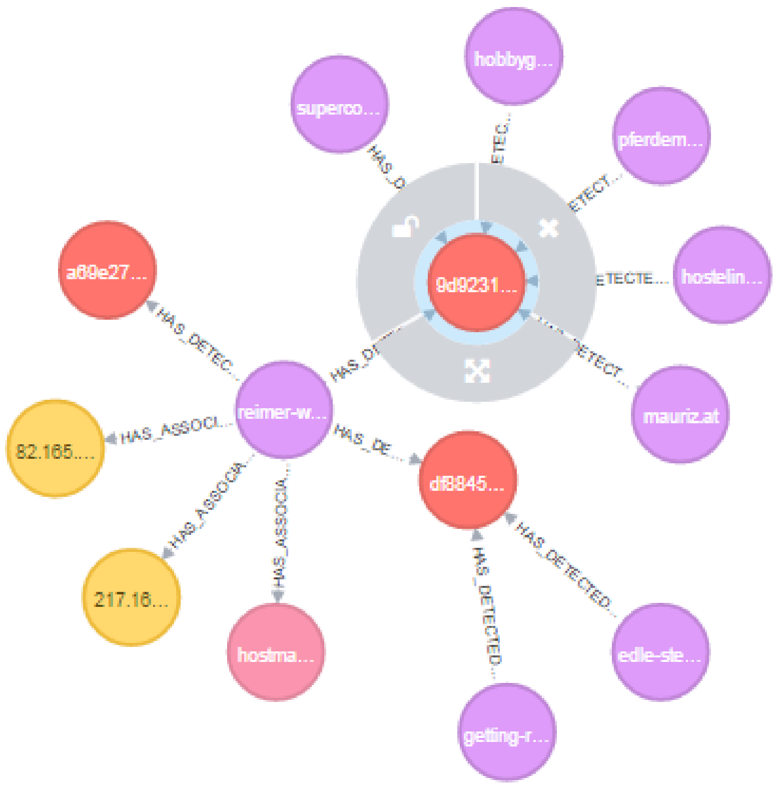
Communicating samples represent the samples that communicate with the domain/IP during sandbox analysis. Apparently, such samples often connect to benign domains to check network availability. Alternatively, ransomware connects to malicious domains to receive further commands. Due to lack of availability of the total number of such incidents or samples, we simply propagate 50% confidence score for HAS-COMMUNICATING-SAMPLE relationships due to bifurcation of possibilities.

In receiving whois information about IP domain resolution, if an IP is owned by a domain, it should normally propagate 100% confidence score. However, these resolutions may change over time, and that requires a massive historical resolution data set, which allows analysts to view which domains resolved to an IP address and vice versa, as well as in terms of time-based correlation. These factors negate the confidence of the system and required integrating time-based information involved to verify accurate resolution during the time period of threat incident. Due to such factors, we consider to propagate 60% confidence to the end indicator for HAS-DOMAIN and HAS-IP relationships.

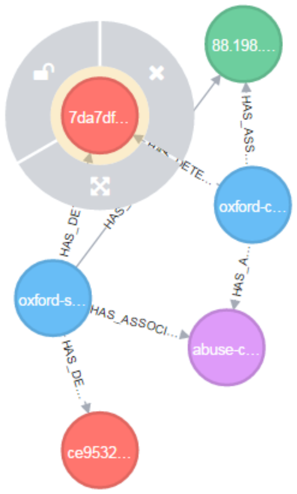
7.6 Example Scenarios

Note that there is no ground truth of measuring the confidence score of threat indicators. Therefore, we opt for providing example scenarios to evaluate the effectiveness of our approaches. We handpicked scenarios to understand what drives receiving higher and lower confidence scores.

First, let's take an example of how propagation of confidence helps us to determine the newer intelligence. In the first example, we found information reported for JSdropper architecture through pastebin. Our system quickly related the information to establish highest possible confidence for



(a)



(b)

Figure 7.4: An Example of New Intelligence Discovered via Propagation of Confidence Score.

9d923147baf5fe885c80cfea2bbc468bac9480822be09db5015bf70993e7513c. At current time, only 4 antivirus systems were able to report such a dropper instance as malicious. Figure 7.4a reveals more information about propagated behavior between
9d923147baf5fe885c80cfea2bbc468bac9480822be09db5015bf70993e7513c and adjacent nodes.

In another example, we found the analysis report where an Iranian threat agent OilRig has been targeting multiple organizations where the attackers set up two fake websites pretending to be a University of Oxford conference sign-up page and a job application website. In these websites, they hosted malware that was digitally signed with a valid, likely stolen, code signing certificate. Based on such behavior we found oxford-careers.com and oxford-symposia.com were set up to host malware
7da7df6b2ae25a2b32a494dacea2c51b02b173dcb020c79f4df47a92fb497274. At the current time, only 1 antivirus system was able to report such malware instance as malicious. Figure 7.4b reveals more information about propagated behavior between
7da7df6b2ae25a2b32a494dacea2c51b02b173dcb020c79f4df47a92fb497274 and adjacent nodes.

However, propagation of confidence from many nodes does not always saturates the confidence at 100. It depends on the behavior displayed by adjacent nodes as well as the indicator itself. To verify the same, we have taken the example of abuse@godaddy.com which stands for email-address from Godaddy's abuse department that monitors the complaint regarding spam. Figure 7.5 reveals that many incoming indicators generalizes relationship behavior to normalized confidence score of abuse@godaddy.com to 20.

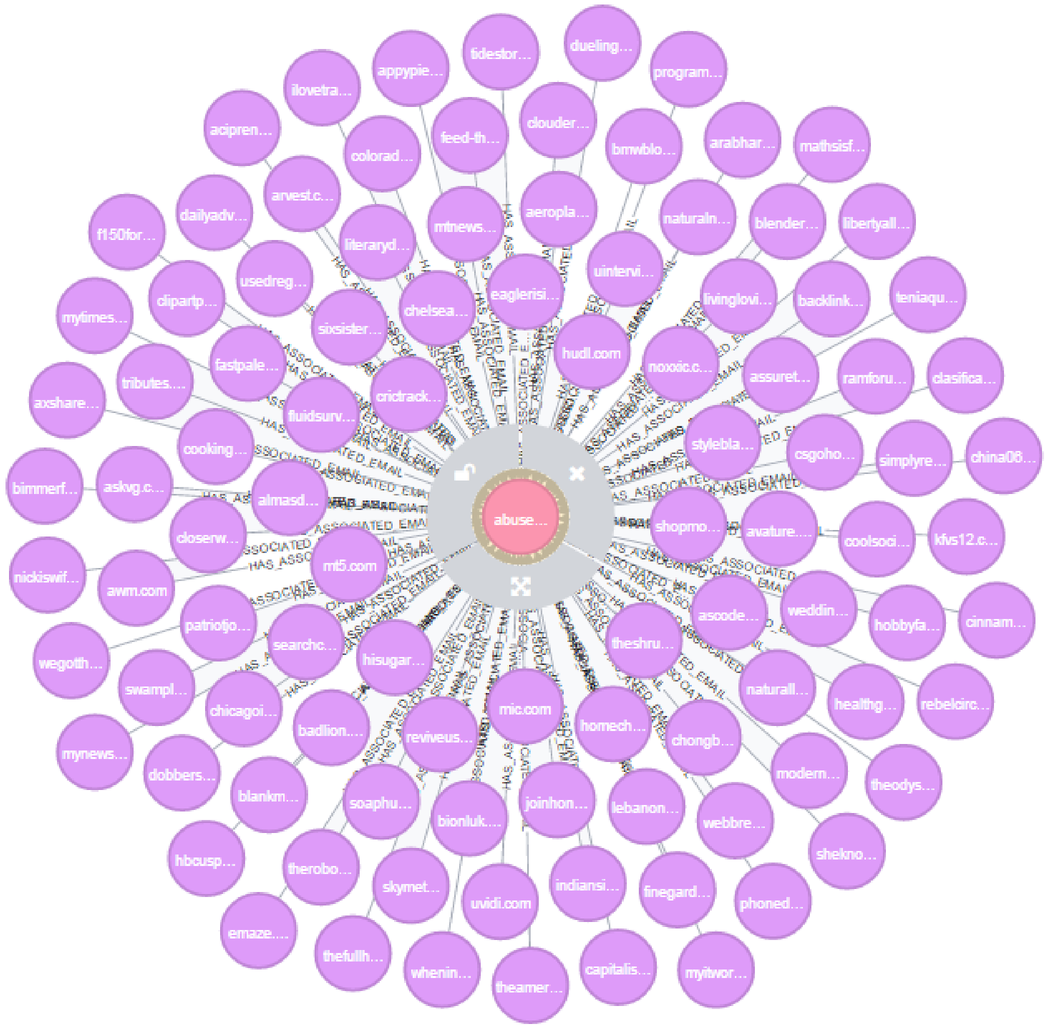
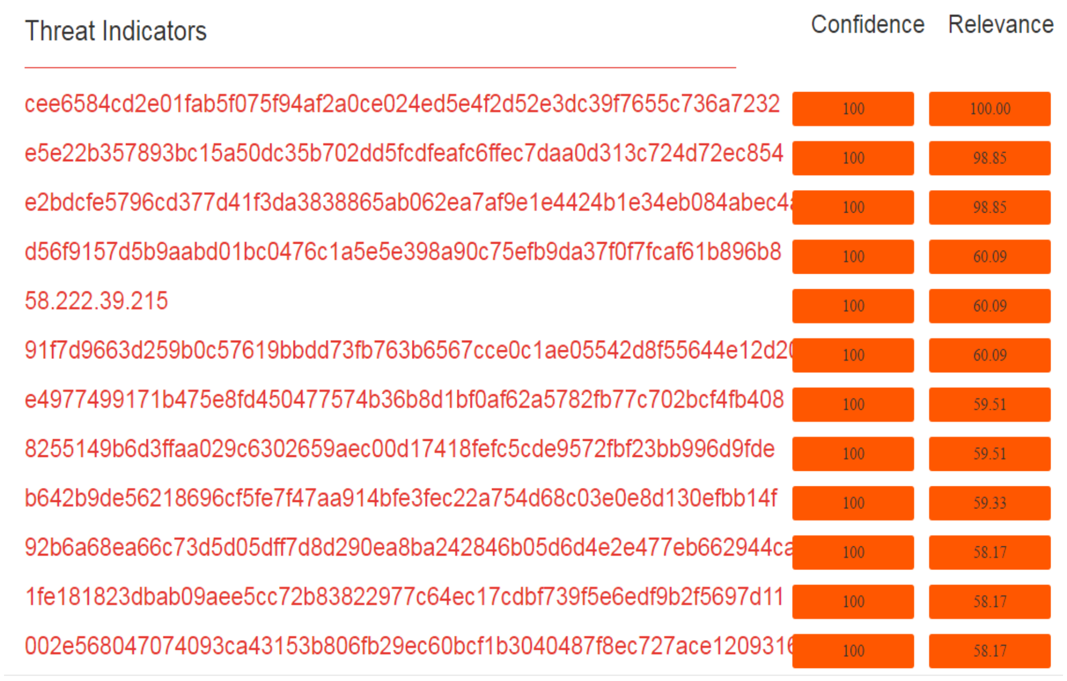


Figure 7.5: An Example of Low Confidence Propagation via Incoming Links.

We noticed that determining the confidence score of a threat indicator is not enough as it may still provide more than 33,000 results, which the analyst would have to analyze or understand for a confidence score greater than 90. We wanted to further narrow the scope of results for the analyst. Hence, we developed a searching application that uses title, description, tags, proximity and other related parameters. The benefits of the search application are the greatest for underspecified queries. For example, an analyst search for "Android Rootnik" may return any number of threat indicators related to Android, but results related to Android Rootnik is listed on top.

In the first example, if human analysts want to find out about the "Android Rootnik" related threat indicators, they can simply input the search term in the search bar. CSM application finds all the threat indicators that contain all or partial of the query words. Then, it sorts the results by relevance score. As shown in the figure 7.6, an application automatically prioritizes the results based on calculated relevance score based on the search term "Android Rootnik." Relevance score of threat indicators, which are related to "Android" but not with "Android Rootnik" gets score less than 60 and declining, while indicators with "Android Rootnik" and 100 confidence receives the score above 95.

In another example, if human analysts want to find out IP address related to "Banking Trojan," they can simply put search term "Banking Trojan" into the search bar and set a filter to a IP address. CSM application takes all IP address and filters them according to a search term. It further calculates relevance score based on the parametric equation that includes search terms, their appearance, and confidence score. It then prioritizes the results based on calculated relevance score. As shown in the figure 7.7, an application automatically filtered category as IP address and prioritized the results based on calculated

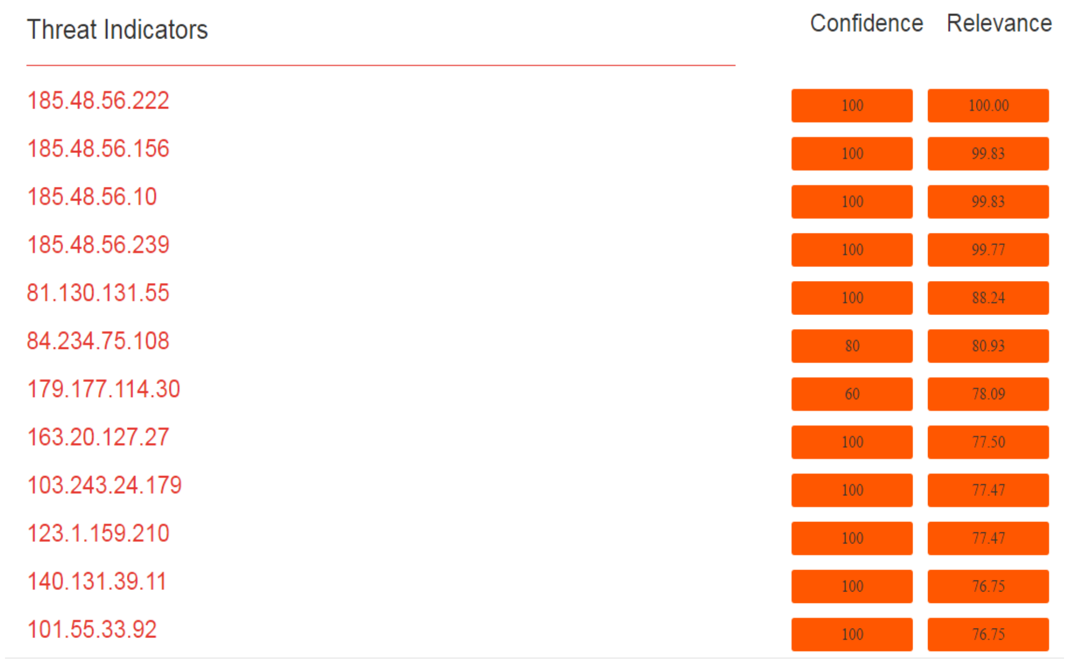


Threat Indicators	Confidence	Relevance
cee6584cd2e01fab5f075f94af2a0ce024ed5e4f2d52e3dc39f7655c736a7232	100	100.00
e5e22b357893bc15a50dc35b702dd5fcdfeafc6ffec7daa0d313c724d72ec854	100	98.85
e2bdcfe5796cd377d41f3da3838865ab062ea7af9e1e4424b1e34eb084abec4	100	98.85
d56f9157d5b9aabd01bc0476c1a5e5e398a90c75efb9da37f0f7fcaf61b896b8	100	60.09
58.222.39.215	100	60.09
91f7d9663d259b0c57619bbdd73fb763b6567cce0c1ae05542d8f55644e12d2	100	60.09
e4977499171b475e8fd450477574b36b8d1bf0af62a5782fb77c702bcf4fb408	100	59.51
8255149b6d3ffaa029c6302659aec00d17418fefc5cde9572fbf23bb996d9fde	100	59.51
b642b9de56218696cf5fe7f47aa914bfe3fec22a754d68c03e0e8d130efbb14f	100	59.33
92b6a68ea66c73d5d05dff7d8d290ea8ba242846b05d6d4e2e477eb662944ca	100	58.17
1fe181823dbab09aee5cc72b83822977c64ec17cdbf739f5e6edf9b2f5697d11	100	58.17
002e568047074093ca43153b806fb29ec60bcf1b3040487f8ec727ace120931	100	58.17

Figure 7.6: An Example Result for Android Rootnik Query.

relevance score. Note that some of the IP address though related to "Banking Trojan," does not have enough confidence receives lower relevance score and IP address related to partial search term receives lower relevance score.

The reason that this search mechanism works so well is that it ensures high precision and the confidence score ensures high quality. When matching a query like "Android," recall is not very important because there are more indicators available then an analyst can analyze. In such queries where precision is more important, it combines full-text and the confidence to calculate the relevance of the threat indicators. For specific searches, recall is more important. In such a scenario, the search mechanism combines a traditional information retrieval technique with a confidence score. We believe using confidence score as a factor in searching mechanism is quite beneficial.



Threat Indicators	Confidence	Relevance
185.48.56.222	100	100.00
185.48.56.156	100	99.83
185.48.56.10	100	99.83
185.48.56.239	100	99.77
81.130.131.55	100	88.24
84.234.75.108	80	80.93
179.177.114.30	60	78.09
163.20.127.27	100	77.50
103.243.24.179	100	77.47
123.1.159.210	100	77.47
140.131.39.11	100	76.75
101.55.33.92	100	76.75

Figure 7.7: An Example Result for Banking Trojan Query.

7.7 Limitations

This section discusses the limitation of our project. The following list goes into the limitation of our project in detail:

- We created a prototyped version to show the effectiveness of our method in calculating confidence score and relevance. For the same, we restricted our research to limited types of threat indicators, and to limited number of threat feeds. It may be possible to derive more meaningful intelligence by adding more niche threat intelligence feeds which provide a detailed results for certain categories, for example, Spamhaus provides insights about various Top Level Domains with their reputations for spam operations. The same can be improved by including diverse threat indicators such as CVE, registry entries, mutex, etc.

- Our system determines the propagation factor and threshold value to derive local confidence through knowledge available in public domain and intuition. Techniques based on natural language processing or machine learning combining with experts' insights can provide more meaningful way to approach such fixed weights and threshold values. We believe there is a scope of improvement in deriving such values in the system.

- We have established relationship between indicators based on context that are widely available as per our knowledge. At the current stage, any unknown context may miss the opportunity to determine proper propagation factor which may lead to receive lower confidence of an indicator. This set of context list needs to be exhaustive to provide full proof system to calculate confidence score of a threat indicator.

- Network-centric indicators change their behavior with the passage of time. Hence, such indicators require continuous reanalysis to update the confidence of them. Similarly, the prior knowledge of system indicators may change with time as many antivirus systems may change their reported analysis with passage of time. Currently, CSM does not provide ability to integrate such analysis in the system.

- Sometimes, an analyst may provide valuable insights about certain indicators, CSM does not provide ability to integrate such feedback in calculation of confidence score.

Chapter 8

RELATED WORK

The community interest in threat intelligence analysis and sharing platform has continuously grown throughout the years. Magee et al. presented a collective of threat intelligence gathering system [20]. Beaver et al. proposed a generic threat assessment approach that provides a computational means to draw conclusions about the probability of a threat [4]. MITRE also presented a system called Collaborative Research Into Threats, which combines an analytic engine with a cyber threat database. [10].

Similarly, there has been a great deal of work on calculating threat score/ risk score where they have published methods of how various parameters such as attacker rating, target rating, valid rating determines the threat score [8]. Another approach proposes a threat assessment algorithm to predict potential future attacker actions based on the attacker's capability and opportunity, and fuse the two to determine the attacker's intent [12]. The network threat risk assessment tool looks at multiple aspects of an IT threat, including both specific (traditional) IT threats and general (non-traditional) IT threats, and generates a threat score for each threat's overall potential to do harm [17].

Though these works on calculating a threat score and designing collective threat intelligence system are related to our work, it still leaves behind a gap to represent a different perspective where we need to assess the confidence score for each of the threat indicators. Risk/Threat score represents the relative or absolute maliciousness of reported malicious threat indicator for the given system. It is determined by the potential damage to the organization if the indicator is successful in carrying out the attack. However, given the community-driven reported threat indicators, we also need to assess the quality of reported threat indicators on the basis of ability to vet the threat intelligence and to confirm the observable as malicious. The confidence score assigns this value to the threat indicator between 0 to 100 where 0 confidence suggests that CSM does not confident enough to consider indicator as malicious, and 100 confidence suggests CSM is extremely confident about indicator being malicious. [31] throws more light on the difference between confidence score and risk score(impact) and derives the matrix to help in the decision making for where to apply the observable.

At the academic level, very little has been reported on how to evaluate confidence of the threat indicator because of the variety and constantly changing nature of cyber security threats. There has been some interest at the industry level on how to measure confidence score of the threat indicators. Rob et al. at Netflix presented Fully Integrated Defense Operation (FIDO) system [27]. They have provided an example scoring mechanism to determine the total score based on various parameters such as Threat Feeds, Other Detectors, Historical Information, User/Machine Previous Alerted, Machine Posture, User Posture, Asset Value by assigning a weight as to how much percentage it will make up of the score for machine/user/threat and total scores. However, there is no information available apart from one sample scoring example which remains insufficient to validate the feasibility of the technique.

Alex et al. at MLSec presented TIQ-Test that carries out a number of statistical tests against IP address threat data. They determined statistical inference based on how often the data changes, how much does feed overlap, and how do they compare with the population of IP addresses allocated to a particular country by performing novelty, overlap, and population test. These statistical test may represent the parameters in determining the confidence score of a threat indicator; however, only one type of data from the heterogeneous dataset may not paint a complete holistic picture of prioritizing and limiting the threat indicators [1].

We also believe utilizing graph properties to harness intelligence from reported threat indicators may provide a better result, which has not been reported to our best knowledge.

Chapter 9

CONCLUSION

We presented our automated Confidence Score Measurement system CSM as a novel system for calculating a confidence and relevance score of a threat indicator that automatically measures the confidence incurred by a graph propagation technique. Our approach starts from the collection of threat indicators from diverse threat intelligence feeds. Our system crawled 70,402 heterogeneous indicators over the span of 5 months and established 51,418 relationships. Then, it extracts relevant information from analysis reports to identify the behavior of a threat indicator and converts into a structured tuple format to conveniently store into the knowledge graph. The graph based propagation technique then propagates maliciousness to adjacent nodes to establish the confidence of an individual threat indicator. CSM further supports analysis by providing a relevant result by prioritizing and filtering the output according to the search criteria provided by an analyst. Also, we have described a proof-of-concept implementation of CSM, along with the extensive evaluation results of our approach. Our result shows the feasibility and effectiveness of calculating confidence and relevance score through scalable graph propagation algorithm and full-text search techniques.

We hope that our work sheds light on how current threat intelligence systems can leverage our technique to prioritize the threat feeds according to the needs of an analyst. To sum up, We developed a novel approach to calculate confidence and relevance score of a threat indicator. We designed and implemented a scalable architecture to provide proof of concept for our approach.

Our proof of concept shows the feasibility and effectiveness of our technique which can be used to prioritize the threat indicators according to the search criteria.

REFERENCES

[1] Alex Pinto, A. S., “Data-Driven Threat Intelligence: Metrics on Indicator Dissemination and Sharing”, URL `https://www.blackhat.com/us-15/briefings.html#data-driven-threat-intelligence-metrics-on-indicator-dissemination-and-sharing` (2015).

[2] alienvault.com, “alienvault”, `http://reputation.alienvault.com/reputation.data` (2017).

[3] autoshun.org, “shunlist”, `http://www.autoshun.org/files/shunlist.csv` (2017).

[4] Beaver, J. M., R. A. Kerekes and J. N. Treadwell, “An information fusion framework for threat assessment”, in “Information Fusion, 2009. FUSION’09. 12th International Conference on”, pp. 1903–1910 (IEEE, 2009).

[5] Bian, J., Y. Liu, D. Zhou, E. Agichtein and H. Zha, “Learning to recognize reliable users and content in social media with coupled mutual reinforcement”, in “Proceedings of the 18th international conference on World wide web”, pp. 51–60 (ACM, 2009).

[6] blocklist.de, “Ftp”, `http://www.blocklist.de/lists/ftp.txt` (2017).

[7] Chau, D. H. P., C. Nachenberg, J. Wilhelm, A. Wright and C. Faloutsos, “Polonium: Tera-scale graph mining and inference for malware detection”, in “Proceedings of the 2011 SIAM International Conference on Data Mining”, pp. 131–142 (SIAM, 2011).

[8] Church, C. A., M. Govshteyn, C. D. Baker and C. D. Holm, “Threat scoring system and method for intrusion detection security networks”, US Patent 7,594,270 (2009).

[9] Cisco, “Mitigating the cybersecurity skills shortage”, `http://www.cisco.com/c/dam/en/us/products/collateral/security/cybersecurity-talent.pdf` (2015).

[10] Community, C., “Collaborative Research Into Threats”, URL `https://github.com/crits/crits` (2016).

[11] Felzenszwalb, P. F. and D. P. Huttenlocher, “Efficient belief propagation for early vision”, International journal of computer vision **70**, 1, 41–54 (2006).

[12] Holsopple, J., S. J. Yang and M. Sudit, “Tandi: Threat assessment of network data and information”, in “Defense and Security Symposium”, pp. 62420O–62420O (International Society for Optics and Photonics, 2006).

[13] Jervis, S., “Cyber attack! 60six months”, `http://www.huffingtonpost.co.uk/shivvy-jervis/cyber-attacks-business_b_5083906.html` (2014).

[14] Jon Oltsik, J. G., Bill Lundell, "Threat intelligence and its role within enterprise cybersecurity practices", http://cdn2.hubspot.net/hubfs/299408/irp/ESG-Research-Report-Abstract-Threat-Intelligence-June-2015.pdf (2015).

[15] Kalafut, A. J., C. A. Shue and M. Gupta, "Malicious hubs: detecting abnormally malicious autonomous systems", in "INFOCOM, 2010 Proceedings IEEE", pp. 1–5 (IEEE, 2010).

[16] Kasza, A., "Visualizing Domain Registration Relationships by WHOIS Information", URL https://umbrella.cisco.com/blog/blog/2014/07/21/visualizing-domain-registration-relationships-whois/ (2014).

[17] Kelley, J., J. Lahann and D. Mackey, "Network threat risk assessment tool", US Patent App. 10/947,575 (2004).

[18] Koutra, D., A. Parikh, A. Ramdas and J. Xiang, "Algorithms for graph similarity and subgraph matching", in "Technical report", (Carnegie-Mellon-University, 2011).

[19] LLC, P. I., "The importance of cyber threat intelligence to a strong security posture", https://webroot-cms-cdn.s3.amazonaws.com/9114/5445/5911/ponemon-importance-of-cyber-threat-intelligence.pdf (2015).

[20] Magee, J., A. Andrews, M. Nicholson, J. James, H. Li, C. Stevenson and J. Lathrop, "Collective threat intelligence gathering system", URL http://www.google.com/patents/US20140007238, uS Patent App. 13/538,831 (2014).

[21] malware domain list team, "Malicious IP address.", URL https://www.malwaredomainlist.com/mdl.php?search=&colsearch=IP&quantity=50 (2017).

[22] malware domains Team, "Malicious Domain.", URL http://mirror1.malwaredomains.com/files/justdomains (2017).

[23] McAfee, "Net losses: Estimating the global cost of cybercrime. economic impact of cybercrime", http://www.mcafee.com/us/resources/reports/rp-economic-impact-cybercrime2.pdf (2014).

[24] McAfee, "Threats report", http://www.mcafee.com/us/resources/reports/rp-quarterly-threats-aug-2015.pdf (2015).

[25] Modi, A., Z. Sun, A. Panwar, T. Khairnar, Z. Zhao, A. Doupé, G.-J. Ahn and P. Black, "Towards automated threat intelligence fusion", in "Collaboration and Internet Computing (CIC), 2016 IEEE 2nd International Conference on", pp. 408–416 (IEEE, 2016).

[26] Morgan, S., "Cyber crime costs projected to reach $2 trillion by 2019", http://www.forbes.com/sites/stevemorgan/2016/01/17/cyber-crime-costs-projected-to-reach-2-trillion-by-2019/ (2016).

[27] Netflix, “Introducing FIDO: Automated Security Incident Response”, URL http://techblog.netflix.com/2015/05/introducing-fido-automated-security.html (2015).

[28] nothink.org, “blacklist”, http://www.nothink.org/blacklist/blacklist_ssh_day.txt (2017).

[29] Pandit, S., D. H. Chau, S. Wang and C. Faloutsos, “Netprobe: a fast and scalable system for fraud detection in online auction networks”, in “Proceedings of the 16th international conference on World Wide Web”, pp. 201–210 (ACM, 2007).

[30] Pope, C. and K. Kaur, “Is it human or computer? defending e-commerce with captchas”, IT Professional **7**, 2, 43–49 (2005).

[31] Poputa-Clean, P., “Automated defense using threat intelligence to augment security”, (2015).

[32] Project, T. S., “The World’s Most Abused TLDs”, URL https://www.spamhaus.org/statistics/tlds/ (2017).

[33] projecthoneypot.org, “listofips”, http://www.projecthoneypot.org/list_of_ips.php?rss=1 (2017).

[34] Robert P. Hartwig, C. W., “Cyber risks: The growing threat”, Insurance Information Institute (2014).

[35] Roberts, J.-M., “Virus share”, (2014).

[36] Safe Browsing, A., “Google developers”, (2012).

[37] SETALVAD, A., “Demand to fill cybersecurity jobs booming”, http://peninsulapress.com/2015/03/31/cybersecurity-jobs-growth/ (2015).

[38] Shackleford, D., “Who’s using cyberthreat intelligence and how?”, SANS Institute. Retrieved February **23**, 2016 (2015).

[39] Simkin, S., “Are You Getting the Most from Your Threat Intelligence Subscription?”, URL http://www.securityweek.com/are-you-getting-most-your-threat-intelligence-subscription (2016).

[40] Smith, M., “70 per cent of businesses overwhelmed by threat intelligence data”, https://www.anomali.com/news-events/in-the-news/70-per-cent-of-businesses-overwhelmed-by-threat-intelligence-data (2016).

[41] Symantec, “Symantec internet security threat report”, https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf (2016).

[42] Team, E., “Theory behind relevance scoring”, https://www.elastic.co/guide/en/elasticsearch/guide/current/scoring-theory.html (2017).

[43] Team, O., “Windows Executables.”, URL http://originaldll.com/search?q=.exe (2017).

[44] Team, R., “Subdomain infringement an unseen threat”, URL https://safe.riskiq.com/rs/455-NHF-420/images/RiskIQ_Subdomain_Infringement_White_Paper.pdf (2016).

[45] Team, T., “ThreatCrowd Community”, URL https://threatcrowd.org/ (2017).

[46] Team, V. R. *et al.*, “Data breach investigations report (2012)”, (2012).

[47] Total, V., “Vt community”, (2011).

[48] VENTURES, C., “Cybersecurity market report”, http://cybersecurityventures.com/cybersecurity-market-report/ (2016).

[49] von Ahn, L., M. Blum and J. Langford, “Telling humans and computers apart automatically”, Commun. ACM **47**, 2, 56–60, URL http://doi.acm.org/10.1145/966389.966390 (2004).

[50] Wikipedia, “Belief propagation — Wikipedia, the free encyclopedia”, http://en.wikipedia.org/w/index.php?title=Belief%20propagation&oldid=759708673, [Online; accessed 28-February-2017] (2017).

[51] Wikipedia, “PageRank — Wikipedia, the free encyclopedia”, http://en.wikipedia.org/w/index.php?title=PageRank&oldid=766217165, [Online; accessed 28-February-2017] (2017).

APPENDIX A

ACKNOWLEDGEMENT

This work was supported by grant from the Center for Cybersecurity and Digital Forensics at Arizona State University. The information reported here does not reflect the position or the policy of the funding agency or project sponsor.