Intelligent Analysis of Attacker Behaviour on the Dark Web

Nathan Andrew Deguara

MSc Cyber Security

Birmingham City University

Supervisor: Junaid Arshad

January 2022

Masters Project

CMP7200

Abstract

The number of cyber attacks on companies' IT systems are increasing in both number and complexity. The origins of many of these attacks can be traced to sites on the dark web. Hacker forums can be a key source of information as they serve as a virtual meeting place for hackers to discuss attacks and techniques. This project investigated how posts on these hacker forums could be automatically analysed to extract key information to produce cyber threat intelligence (CTI). In this project a system was created to analyse posts from hacker forums and use semantic analysis to generate a similarity score which could be used to determine how likely it was that a post contained information that related to a cyber attack. The system created made use of a Word2Vec machine learning model for semantic analysis. Once the information had been extracted from the posts the information was written to a STIX report so that the information found by the system could be uploaded to an open-source threat intelligence platform (OSINT). The reports generated were classed as being either good, bad or vague and analysis showed that most of the reports generated were vague, meaning they contained some useful information, but more analysis was needed to be able to act upon the intelligence found. The system analysed thousands of posts and generated a few hundred reports based on the information it had found.

Acknowledgements

I would like to thank all my family and friends for their continued support throughout my studies.

I would also like to thank my supervisor Junaid Arshad for all his time, dedication and support throughout the project.

Contents

Table of Figures

Table of Tables

1.0 Introduction

This project looked at identifying new and emerging cyber threats by carrying out analysis on data that has been gathered from the dark web and various other sources such as hacker forums. From this analysis it was hoped that information on new techniques and targets could be identified and turned into actionable intelligence.

1.1 Background

Cyber security is an ever-growing field with cyber attacks on companies becoming more common and the complexity of these attacks is constantly increasing. In the first six months of 2021 there was a 125% increase in cyber related incidents (Accenture, 2021). Many of these attack have origins in the dark web such as buying, selling or sharing malware source code (Kaspersky, 2020).

It should be noted that there is a notable difference between the dark web and the deep web. Although the terms are often used interchangeably the deep web is a range of websites and databases that cannot be accessed through conventional search engines (Hatta, 2020). Some examples of deep web uses are for a private intranet, emails, or a private database. Some estimates suggest that this accounts for 90% of all content on the internet (Kaspersky, 2020).

The dark web is a collection of websites that can only be accessed using a dark web browser, such as Tor. Users on the dark web are completely anonymised and connections between users and servers are never direct. Instead, each device connects through a series of nodes and relays and each device will meet at a set node where all communication will take place. This ensures there are several layers of protection between users making it nearly impossible to trace IP addresses (Hatta, 2020). While only a small portion of the deep web is actually the dark web, it is the latter in which most illegal activity including cyber crime takes place and is discussed.

1.2 Rational

This project is important because cyber security is a growing field with many companies in different sectors facing increased cyber threats. The use of open-source intelligence (OSINT) to prepare and counter these threats is a vital part of many organisations' cyber security strategy. The dark web provides an excellent source of intelligence about cyber attacks as it has numerous hacker forums where cyber criminals can discuss techniques of a cyber attack and share malware for use in such attacks. Carrying out monitoring and analysis of information shared on the dark web is something already being done by several different organisations such as various law enforcement agencies and it often relies on a mix of technology and analysists. (Photon Research Team, 2019). Finding this information and turning it into actionable intelligence that can be used by organisations to prevent a cyber attack is a core goal of this project which could help many organisations around the world.

1.3 Aims

The aim of this project was to develop a text mining system that uses semantic analysis to accurately identify new cyber threats from the most influential hackers within the CrimeBB dataset. The identification of these threats would then be turned into actionable intelligence that was in a format that could be used by major open-source threat intelligence platforms. This will allow organisations to understand and prepare for the threats posed. These threats could take the form of a new hacking technique, new malware, or a targeted cyber attack against a particular organisation.

1.4 Objectives

There are five major objectives that were identified for this project.

- Understand how to identify different types of cyber threats
- Identify cyber threats from the most influential hackers within the CrimeBB dataset
- Develop a text mining system using semantic analysis to accurately identify cyber threats from influential hackers within the CrimeBB dataset
- Test the accuracy and relevance of identified cyber threats
- Turn findings into actionable intelligence that is in a compatible format with open-source threat intelligence platforms

2.0 Literature Review

The first step in planning the project was to undertake extensive background research into the core topics of the project. The key topics of this project are cyber threats, open-source threat intelligence, cyber threat intelligence formats and semantic analysis. It was also important to identify and analyse any existing methods. This helped to accomplish objective 1.

2.1 Cyber Threats

The first theme of this project was cyber threats. The term cyber threat encompasses a diverse set of threats that are present to computers and often utilise the internet. Some of the threats included in this term are cyber attacks, cybercrime and misinformation (Choo, 2011). This project mostly focused on the threat of cyber attacks.

A cyber attack is an attack on an organisations use of cyberspace with the aim of disrupting, disabling, destroying or maliciously controlling a computing environment or stealing information or destroying the integrity of data (Rebecca M. Blank. Patrick D. Gallagher, 2012). Cyber attacks can take many forms such as hacking, malware, denial of service and phishing (Choo, 2011). Hacking is the process of taking over or compromising a computer system, network, or data. It is often combined with the use of malware and phishing (McClure, Scambray and Kurtz, 2012).

Malware (malicious software) is a type of software that causes harm to a computer (Sikorski and Honig, 2012). Malware is one of the most common cyber attacks in use and its purpose varies depending on what type of malware is being used (Choo, 2011). There are many different types of malware such as viruses, backdoors, botnets, information stealing malware and ransomware (Sikorski and Honig, 2012).

Denial of service (DoS) is where an attacker overloads a computer or network with information, often taking the form of user requests, with the aim of making that computer or network unavailable for other users. This can be done as a regular denial of service attack (one user) or as part of a distributed denial of service attack (DDoS) which often makes use of botnets (compromised computer networks) to flood the target with network traffic (Bhattacharyya and Kalita, 2016).

Phishing often takes place as part of a wider cyber attack. It is where an attacker attempts to steal information by tricking a legitimate user into performing some action. This could be done by sending an email to a user asking them to click on a link and then provide login details which would then be sent to the attacker (Jansson and Von Solms, 2013). Alternatively, the link could be used to install malware on the users device (Choo, 2011).

The number of cyber threats facing organisations are increasing year on year. Most organisations have become the target of cyber attacks. Some of these are automated attacks while others are specifically chosen (Runciman, 2020). Whether the attacks are coming from automated systems, random hackers or are part of an attack by advanced persistent threats (APTs), it is ever more important to have a strong cyber security strategy in place to defend against these threats.

There is no one solution to defend against cyber threats and it is a field that is constantly evolving. One of the key defences is to have a strong cyber security policy in place (Choo, 2011). It is also important to have several different layers of defence such as intrusion detection/prevention systems (IDS/IPS), network segregation and, anti-virus (O'Leary, 2019).

2.2 Open-Source Threat Intelligence

One of the key themes of this project was the topic of open-source threat intelligence (OSINT). Open-source threat intelligence forms an important part of cyber threat intelligence (CTI). The goal of CTI is to research the developments in cyber-attacks and analyse the latest trends (Tundis, Ruppert and Mühlhäuser, 2020). OSINT is the collection of information using publicly available data. This data could come from social media, websites or videos (Tundis, Ruppert and Mühlhäuser, 2020). The information gathered can then be used to prepare for or prevent cyber attacks.

Collecting and analysing open-source intelligence is a complex process. It often gathers vast quantities of data which then needs to be analysed for intelligence. Not all the data gathered will be useful and some of the information could be outdated or simply false (Gao *et al.*, 2021). One way of limiting the amount of data to analyse and find useful information quicker is by using topic detection. Topic detection can be used to identify keywords and themes within data. This can be done using semantic analysis (Li, Zhou and Xue, 2020). Li, Zhou and Xue go on to say how they used semantic analysis and machine learning algorithms for topic detection. This will be talked about further in section 2.4 Semantic Analysis.

An example of open-source threat intelligence being used is to identify cyber threats to critical infrastructure such as electricity grids. One way of identifying intelligence relating to a specific target is by searching for anything discussing that target in particular but another way is by searching for information regarding the particular systems in use by an organisation (Lee and Shon, 2017). Another use is to try to identify threat actors. These are the people and organisations that are carrying out cyber-attacks. This could be cyber criminals or hostile states. While it remains almost impossible to trace people over the dark web, the same threat actors could be associated with different cyber attacks and a profile of the attackers could be built up (Mavroeidis *et al.*, 2021).

There are several tools that exist for assisting in the gathering of open-source intelligence. One of these tools is MISP (Malware Information Sharing Platform). MISP is an open-source platform where users can share intelligence they have gathered. This is particularly useful because users can see what intelligence has been found for systems they are using. For example intelligence regarding threats to a particular operating system (MISP, 2021). Another platform for open-source intelligence is Echosec. This is a system that gathers huge quantities of data from various different sources for organisations to use (Echosec Systems, 2021). Another platform for open-source threat intelligence is Talos Intelligence. This is a platform from Cisco that highlights the biggest threats on a weekly basis (Cisco, 2019). A comparison of these platforms can be seen in *table 1*.

MISP	**Echosec**	**Talos Intelligence**
Regular updates	Real-time data	Weekly updates
Community run	Managed by Echosec team	Managed by Cisco team
Highly automated	Very large database	Updates sent to Cisco devices
CTI format – custom JSON files but has dependencies on STIX, CybOX and MAEC	CTI format – not stated	CTI format – not stated
Free to use	Paid service	Free to use

Table 1: Comparison of open-source threat intelligence platforms

An open-source threat intelligence platform that has been looked at in depth for this project is OpenCTI. OpenCTI is a collection of communities that share different types of cyber threat intelligence. Each of these communities is called an ecosystem (OpenCTI, 2018). One of the major advantages of OpenCTI is that it has a high level of data availability (Menges, Putz and Pernul, 2021). This means that data can easily be accessed, and the platform has measures in place to avoid significant system outages. This is important because if a system automatically posts cyber threat intelligence reports to the OpenCTI platform and cannot connect, it may cause the system to crash. Another advantage of OpenCTI is that it has strong authentication in place to help with non-reputation (Menges, Putz and Pernul, 2021). This means what reports have been posted and the user who posted the reports can be tracked to ensure that the reports are accountable. There are various types of communities on the OpenCTI platform including ones run by major cyber security companies such as Kaspersky, communities for specific types of vulnerabilities and many more (OpenCTI, 2018).

2.3 Cyber Threat Intelligence Formats

Any open-source threat intelligence that is found needs to be published to an open-source threat intelligence platform in order to be effective and used by the cyber security community. In order to make it easier for data to be shared and uploaded there are several universal cyber threat intelligence formats.

The first of these formats is structured threat information expression (STIX). This is an open-source project that aims to make it consistent and easy to share intelligence. There are nine key constructs including the indicators of compromise, types of exploit and how to respond (Barnum, 2014). STIX is written using a series XML files and Python scripts. It is built to work well with other CTI formats. It is built to be in a machine-readable format. STIX is specialised to characterise malicious activity using attack patterns (Barnum, 2014). Another CTI format is cyber observable expression (CybOX). CybOX has now been integrated with STIX. CybOX was built to be flexible across different vendors and is specialised to analyse event data (Barnum, 2014).

Malware Attribute Enumeration and Characterisation (MAEC) is specialised for malware reporting and aims to reduce the inaccuracy that exists when reporting malware based on its signature alone (The MITRE Corporation, 2017). MAEC data can be represented as a graph that shows the relationship between different objects. This is especially useful when determining if an instance of malware is part of a larger malware family and to identify connected attacks (The MITRE Corporation, 2017). Another format is trusted automated exchange of intelligence information (TAXII). TAXII is an application layer protocol that focuses on the sharing of CTI. Information is shared over HTTPS and works on a client-server architecture (Oasis Open, 2021).

When choosing which cyber threat intelligence format to use it is important to look at several factors such as what type of intelligence is being gathered and what open-source threat intelligence platforms it is compatible with (Cyware, 2021). Each CTI format has its own particular uses and they often work together. Some OSINT platforms may accept more than one type of CTI format in which case it is important to use the one that best describes the cyber threat intelligence that has been gathered and is contained within the reports generated (Barnum, 2014).

2.4 Semantic Analysis

Another theme of the project is semantic analysis. Semantic analysis is the process of using natural language processing and text analysis to understand the meaning of information (Omitola, Riós and Breslin, 2015). It understands information by calculating the degree of similarity between different words (Landauer, 2002). How the similarity is calculated differs depending on the algorithm used.

There are many different algorithms that are used for semantic analysis. One of the most common algorithms used is latent semantic analysis (LSA). Latent semantic analysis is used widely because it has proven to be highly accurate when understanding the meaning of text (Liu and Wang, 2012). Another commonly used algorithm is term frequency – inverse document frequency (TF-IDF). TF-IDF has proven to be an extremely useful algorithm especially when it comes to recommendation systems (Beel *et al.*, 2016). Some other more generic algorithms, such as support vector machine (SVM) and Naïve Bayes, can also prove useful when it comes to semantic analysis (Medhat, Hassan and Korashy, 2014).

Most semantic analysis algorithms work by building a decision graph or tree, where each word is a node on the graph (Steyvers and Tenenbaum, 2005). The distance between the nodes can be used to calculate the similarity between words. How these graphs are then used differs between algorithms and techniques. For example latent semantic analysis creates “bags” of words and assigns an importance to each word depending on how many times it appears in a passage of text (Liu and Wang, 2012). The meaning of the most common words is then used to understand the meaning of the text. TF-IDF also looks at how often a word occurs in a passage of text but that is used to assign a “weight” to each word that is proportional to the number of occurrences (Beel *et al.*, 2016). This helps to understanding the meaning of text by using the text as a whole but focusing on the more common words.

A relatively new set of algorithms being used for semantic analysis is Word2Vec. Word2Vec is a collection of model architectures and optimisations used to learn word embeddings from data (TesnorFlow, 2021). There are two main methods of Word2Vec. The first of these methods is the continuous bag-of-words model. This model predicts the middle word based on the surrounding words for context (TesnorFlow, 2021). The second model is the continuous skip-gram model which predicts words either before or after the current word. The number of words predicted depends on the parameters set (TesnorFlow, 2021). A key component of Word2Vec is the similarity score. This compares words to each other and generates a score depending on how similar they are. The higher the score the more closely related the words are (Church, 2017). One of the major downsides of Word2Vec is that it is still a work in progress meaning that some functions are not fully optimised and there may still be a few software bugs present. Since it is a relatively new semantic analysis tool it has some issues. One of the major issues is that it does not always correctly identify similar words and sometimes results can have a high similarity score but not actually be related to one another (Church, 2017). Despite this Word2Vec has already proven to be an effective semantic analysis tool and compared to other algorithms in use is relatively simple to implement (Di Gennaro, Buonanno and Palmieri, 2021).

The Word2Vec semantic analysis algorithms are very scalable and work well on both large and small datasets (Li *et al.*, 2019). This was an important consideration for the project as the posts in the dataset varied in length from just a few words to large paragraphs of text. The Word2Vec algorithms are pre-trained using data gathered online (Perambai, 2020). Despite this the Word2Vec model must still be trained on the particular dataset that is being used for analysis to ensure that it has the correct understanding of words in context and to both tokenise the words and calculate similarity scores for the words in the dataset being used (Perambai, 2020). While Word2Vec is a collection of machine learning algorithms rather than a single model, it makes use of neural networks (deep learning) for analysis on the data. It uses a series of hidden layers and word embeddings to calculate word similarities and depending on which method is being used and how it is being applied, it can use these techniques for word prediction as well as several other uses (Perambai, 2020).

An example of semantic analysis being used is to discover open-source threat intelligence by using topic detection by Li, Zhou and Xue (2020). They used a custom variation of the TF-IDF algorithm to help improve open-source data mining. They search various sources of open-source data to try and find cyber threat intelligence. The TF-IDF algorithm is searching for frequent mentions of several key terms such as malware and hacking with higher weights being assigned to words of interest. The findings showed that using an existing semantic analysis algorithm but modifying it to better work for cyber security provided some very useful information (Li, Zhou and Xue, 2020).

There are several sub-groups of semantic analysis. One of these groups is sentiment analysis. Whereas semantic analysis as a whole aims to understand the meaning of text, sentiment analysis aims to determine if the text has a positive, negative or neutral sentient towards something (Gupta, 2018). Sentiment analysis has become particularly popular with the rise of social media and online shopping where posts can be analysed to determine the users feelings towards a topic or product (Liu, 2012).

There are several different semantic analysis tools that are specialised for data mining. One of these tools is Weka (Waikato Environment for Knowledge Analysis). Weka is a collection of machine learning algorithms that are optimised for data mining (Witten *et al.*, 2017). It includes tools for data preparation and processing. Another tool used for semantic analysis is RapidMinor. RapidMinor is a data science platform that has a wide range of algorithms available for different data processing tasks. These tasks can range from preventing fraud to stopping a cyber attack (Cartes, 2020). Another one of the other tools used for semantic analysis is Node-Red. Node-Red is a programming tool that can be run either locally or in a cloud environment. It has in built libraries to help with data analysis and data mining (OpenJS Foundation & Contributors, 2013).

3.0 Methodology

It was important to identify how each stage of the project would be carried out in order to ensure that all the objectives set out for this project are achieved. The project was split into four distinct stages which each relate to one or more objectives.

3.1 Research Methodology

The first stage of the project was the research stage. This part of the project included carrying out the literature review and understanding the topics involved in the project. The research stage of the project followed the onion methodology by Saunders, Lewis and Thornhill (2019). The key stages of the onion research methodology are philosophy, approach, strategy, choices, time horizon and techniques (Crossley and Jansen, 2021).

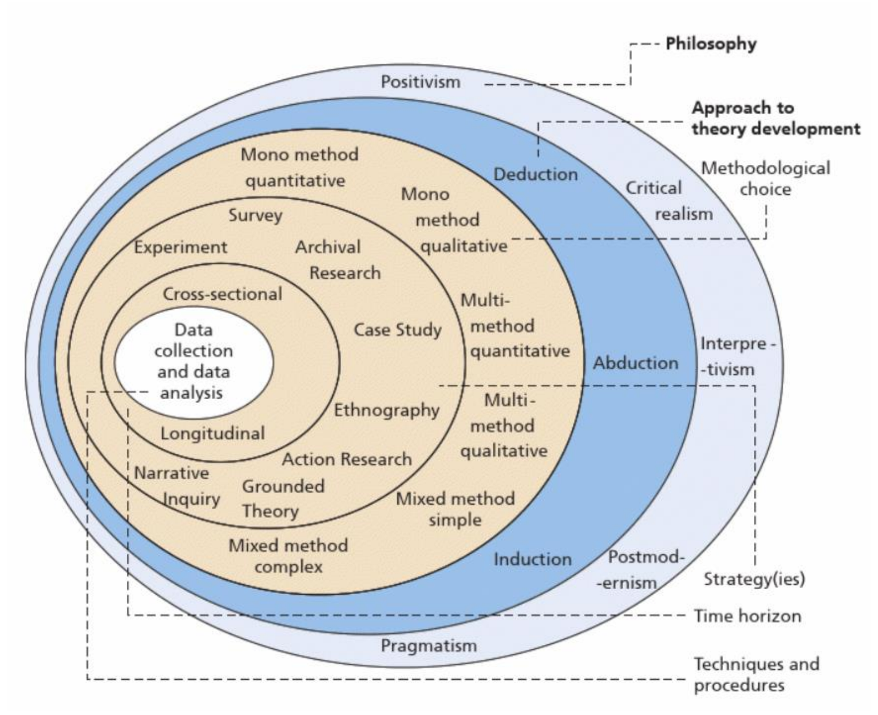

Figure 1: The onion methodology (Saunders, Lewis and Thornhill, 2019)

The philosophy stage looks at what is known and how it is known. It forms the foundation of the research (Crossley and Jansen, 2021). For this project a pragmatic research philosophy was used. This is because it both questions and interprets knowledge which is important in a field where information changes quickly such as computer science and cyber security. The approach is the method used for the research (Crossley and Jansen, 2021). For this project an inductive approach was used. This is because literature was used to research the key topics before a theory was created on how to implement the system created in the project. The strategy looks at how the research is conducted. This project made use of a grounded theory approach to the research strategy. This approach was chosen as it is data driven and choices are made based on the results gathered and data analysis (Crossley and Jansen, 2021).

The next stage of the research onion methodology is the choices. It is about choosing what data types will be used (Saunders, Lewis and Thornhill, 2019). This project made used of a mixed approach where both qualitative and quantitative data were used. The qualitative data was used when analysing the reports produced by the system to determine the quality. Quantitative data was used when getting an overall statistical view of the number of reports that were classed as either good, bad or vague. The quantitative data was based on the qualitative data.

The next stage in the onion research methodology is the time horizon. This stage describes when data will be collected and how many times. For this project a cross-sectional time horizon was used. This is because the data was already gathered, and it was not changing over time. The data was static (Crossley and Jansen, 2021). The final stage of the onion research methodology is the techniques. This stage is about how data will be collected and analysed. This stage is explained throughout the rest of this section. Gathering research and critically analysing it to identify important information helped to achieve objective 1.

3.2 Dataset Analysis

The next stage of the project was to understand the data being used. For this project the CrimeBB dataset by Pastrana *et al.* (2018) was selected. This is a large dataset with several subsets that have been gathered from dark web hacker forums. Initial analysis on the data needed to determine which subsets would be used for the project. The project would be using the posts from the most influential hackers using the hacker forums so where these posts reside in the dataset needed to be identified. This achieved objective 2.

3.3 Development

The next part of the project was to develop the system. The development of the system followed the waterfall software development methodology. Waterfall was used because it is sequential and splits development into distinct tasks (Balaji, 2012). This formed objective 3 and part of objective 5. A block diagram was created to show the outline of the system that would be created and how the parts would interact with one another. This can be seen in *figure 2*.

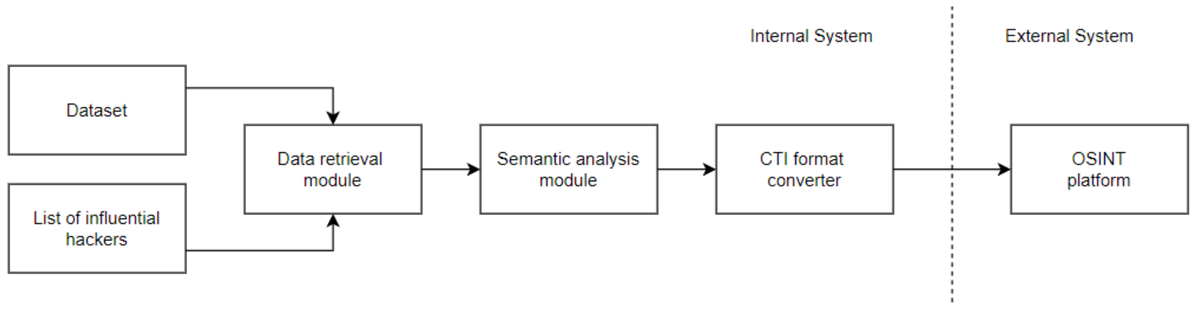

Figure 2: System block diagram

First the system being developed would need to retrieve the data. This required the dataset and a list of influential hackers within the dataset. The data retrieval module then gathered all the posts by the influential hackers from the dataset. This data would then be passed onto the semantic analysis (text mining) module. This is where the posts were searched for any information regarding cyber threats. Once all the data was analysed any possible threats would then be passed onto the CTI format converter. This would get the raw intelligence gathered through semantic analysis and convert it into a format that can be used by the OSINT platforms. Some formats that could have been used are STIX or Cybox. Finally, the system could have been developed further to automatically send information to the OSINT platforms.

3.4 Testing and Integration

The final stage of the project was to test the system and convert the results into a format that was compatible with open-source threat intelligence platforms. This would accomplish objectives 4 and 5 respectfully. Testing took three forms. Unit testing was carried out throughout development of the project to ensure that each individual section created works. This tested the quality of the code and checked for syntax or runtime errors but not necessarily the logic as that might require several other parts of the system. When the development phase was completed, and the testing phase had started then integration testing took place. This ensured that the system worked as a whole and that the logic of the system worked correctly. Finally, once all the tests were completed and any necessary alterations to the system had been made, regression testing was used to ensure that any changes made to the system worked correctly.

An additional objective of the project if there was time was to attempt to integrate any intelligence found to open-source threat intelligence platforms. This would be done by converting all intelligence found into a format that is compatible with CTI platforms such as Stix or Cybox. This would allow the intelligence gathered to be used by organisations to defend against cyber attacks.

3.5 Project Schedule

The project was split into six major stages. The first stage was the research stage. This involved gathering information and gaining an initial understanding of some of the key topics in the project. Then the first deliverable for the project was worked on. This was the project proposal. The next stage was to develop the code for the actual project. This was the largest task of the project and involved working with the dataset, designing the system and writing code. Once development of the system was completed it underwent testing and refinement. As part of this stage, it was also attempted to integrate the system with an OSINT platform. The final stages of the project were to write the dissertation and prepare a presentation. This marked the end of the project. A Gantt chart showing the timeline of the project can be seen in *figure 3*.

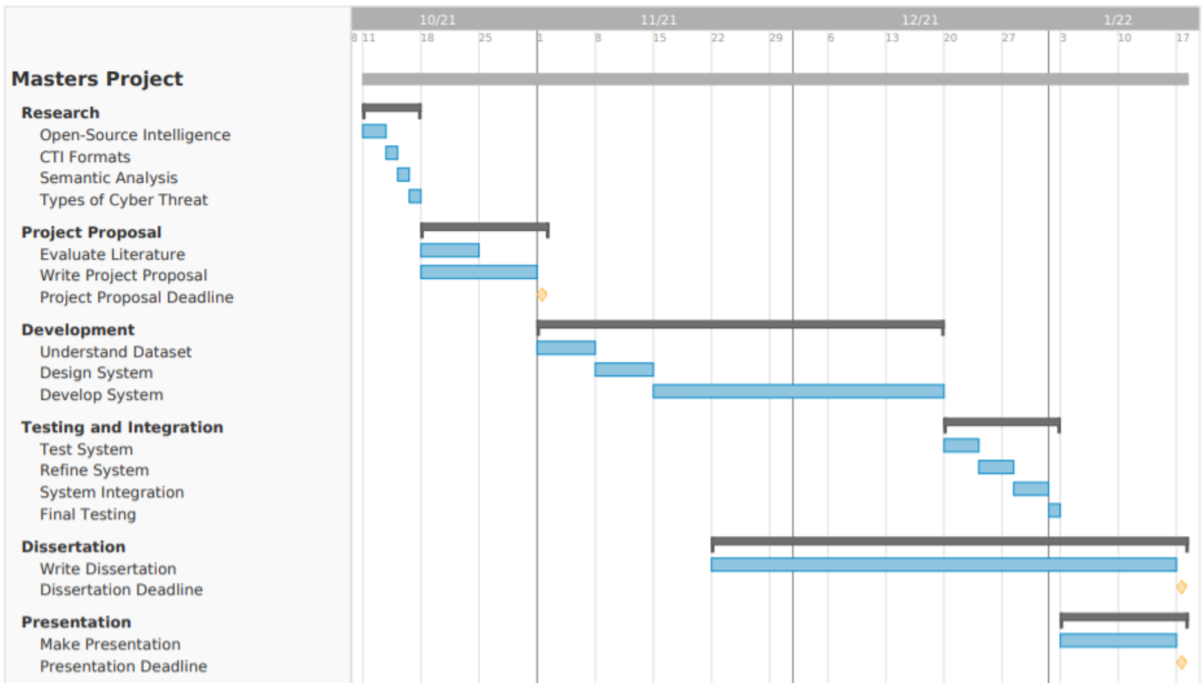

Figure 3: Gantt chart

Gantt Chart Legend
Black bar – Parent task duration Blue bar – Individual task duration Yellow Diamonds – Milestones

4.0 Development

The development phase of the project involved building, testing and refining a system to analyse posts on hacker forums and find possible cyber threat intelligence. The aim of this phase was to identify the influential hackers (objective 2), develop a system to accurately identify cyber threats (objective 3), test the accuracy and relevance of identified cyber threats (objective 4) and to turn the findings into actionable intelligence (objective 5).

4.1 Dataset

The dataset used for this project was the CrimeBB dataset by Pastrana *et al.* (2018) and provided by Cambridge Cyber Crime Lab. In particular the Hack Forums data was used for this project. The project was building upon work done in a previous project done by Atique, Arshad and Mubashir (2021). This previous project looked at influential users on the Hack Forums site and gave users a reputation based on various different factors. This reputation was used to identify users of interest in this project.

The dataset took the form of two CSV files. The first file was the original dataset which contained the posts on Hacker Forums along with the author ID of the user who made the post and the destination ID of the user who was being replied to (if the post was in response to another). The second file contained the users of the hacker forum and was the work by Atique, Arshad and Mubashir (2021) where each user had a reputation associated to them. This file contained user IDs along with a reputation and influencer score. A higher reputation meant that a user had more “social influence” on the forum. The values for a user’s reputation and influencer score were calculated using the number of posts a user made and the amount of interaction each of these posts received from the forum’s community (Atique, Arshad and Mubashir, 2021).

The ID columns were all numerical in value and did not contain the actual usernames. Each ID was unique to a particular user. The posts contained string data. The reputation was an integer and the influencer score was a floating-point number. The lowest reputation was -1046 and the highest reputation was 3634. The average reputation was 40 and the most common value for reputation was 1. In total there were 13,214 users and 17,719 posts in the dataset.

4.2 System Design

The next step was to design the system to be created. This involved selecting the algorithm to use for semantic analysis, creating system diagrams and developing protypes. For the project two prototypes were created. The first prototype used latent semantic analysis as the machine learning algorithm. This worked by reading the data and getting the posts from all users with a reputation greater than ten. This number was chosen as a starting point and further tests would be done in the development phase to work out the optimal value for the reputation. Then a term frequency-inverse document frequency (TF-IDF) statistical model was trained on the data. This statistical model was used to give each word in the dataset a weighting. The output of this function was then given to the latent semantic analysis algorithm through the use of truncated singular value decomposition (SVD). This returned a series of vectors relating to each word that was analysed by the algorithm.

The second prototype used a Word2Vec machine learning model for semantic analysis. This started the same as the first prototype by retrieving the data and getting all posts from users with a reputation greater than ten. Then all the posts had to go into the Gensim pre-processing tool. This got the unique words in each post and separated them into a series of strings called tokens. This ensured that each post had no duplicate words and that all the letters were in lower case. It also implements stop words which ensures that commonly used words that do not affect the overall meaning of a sentence are removed, such as “a” and “the” (TesnorFlow, 2021). Then the pre-processed data was sent to the Word2Vec algorithm for analysis. The output of the algorithm was a similarity score between different words in the dataset.

From the prototypes it was decided to use Word2Vec as the algorithm for this project. This is because the output of the algorithm was significantly more useful for the aims of this project than the output of latent semantic analysis. The similarity score provided by Word2Vec could immediately be used to search for posts containing terms similar to words related to hacking. If LSA was used, then additional code would be needed to interpret the vectors provided and calculate distances between words to determine how similar the words were to one another. Given the large size of the dataset, this would have a considerable overhead and require a large amount of system resources to carry out the necessary calculations for all words within the dataset. Additionally, research had shown that Word2Vec was a very good algorithm for semantic analysis. For these reasons Word2Vec was chosen as the algorithm used for this project.

Once that the algorithm had been chosen, system diagrams could be created to show all the components of the system and how they would work together. The UML diagram of the system created can be seen in *figure 4*. Where the data type is listed as DF the variable uses a data frame from the pandas library for Python created by the pandas development team (2020).

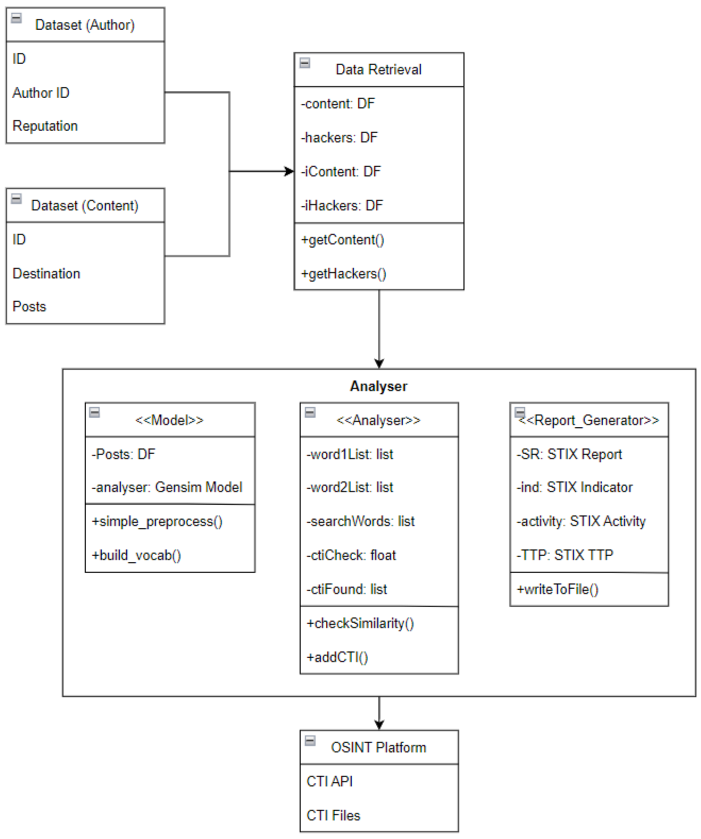

Figure 4: System UML diagram

The data retrieval module gets data from the two CSV files. The Model class creates the Word2Vec model by building its vocabulary and performs pre-processing on the data. The Analyser class searches through all the data to find possible CTI or IOC (indicators of compromise). The Report Generator class gets the results of the analyser class and generates a STIX report for each instance of CTI found.

4.3 System Development

Once the system had been designed it was time to build the system itself. Following the system diagram and using what had been learnt from the prototypes a CTI analyser system that searched posts for possible cyber threat intelligence and then automatically generated STIX reports was created.

The first step was to import all the external libraries that were needed. This included the Pandas library by the Pandas development team (2020), Gensim by TesnorFlow (2021), and the STIX libraries by OASIS Cyber Threat Intelligence Technical Committee (2021). Then each of the datasets were read into the program. Each CSV file was stored in a Pandas data frame. The content data file that contained the posts themselves was stored in a variable called content and used columns 0 (source), 1 (destination) and 2 (post). The columns which were required by the system was stated because some of the posts were split across several columns in the CSV file. The majority of the posts were in column 2 however there were a small number that used columns 2 and 3 with one using up to 33 columns. This was an issue with the dataset provided and could not be corrected without access to the original data and source code. Since this was such a small number the decision was made to ignore data in these extra columns. While that did result in some data being lost, the small number of posts that this affected ensured that the overall quality of the program did not suffer as a result.

The author file contained the ID, author ID, reputation and an influencer score. The ID was the graph generated ID and is the ID that would be used to identify users going forward. The influencer score was generated through the work of Atique, Arshad and Mubashir (2021) and related to the reputation of the user. The author CSV file was saved into a variable called hackers. This data was then reduced by getting all users with a reputation greater than 10 and storing the results into a variable called iHackers. This made up the list of influential hackers whose posts would be analysed. The value of ten was chosen as it provided the optimal number of results. Results for different values can be seen in *table 2*.

Reputation Value	Posts Returned
0	13861
10	7515
50	4451
100	1034
500	278
1000	127
2000	17

Table 2: Reputation values

As can be seen in the table, using a reputation of greater than 10 provided 7515 posts to be analysed by the system. Through testing of the system later in the project it was found that using a reputation greater than 50 did not provided enough results to effectively train the machine learning model on the dataset being used. Using a reputation value of 0 provided plenty of data to train the model but added a significant number of posts and users which were not considered influential and added noise to the data, thus the quality of the cyber threat intelligence found was greatly decreased. There was no significant performance difference noted between reputation values of 10 and 50 but 10 was chosen since it provides more data to train the system on which did slightly improve the results generated. It should be noted that it was found there were 3903 users with a reputation less than 0 and 9705 users with a reputation less than or equal to 10. This makes up 73.4% of users on the forum. There are 3509 users with a reputation greater than 10. Using a reputation of greater than 10 means only the top 26.6% of users' posts are included for analysis by the system.

The next step involved getting posts from all these influential hackers. For this the source header in the content file was renamed to ID. This then allowed the two data frames to be combined together using the merge function in Pandas library. The content and the iHackers data frames were merged based on the ID column. This meant that if the ID value was present in both the iHackers and the content data frames then all the information in both data frames for that record would be written to a new data frame called iContent. This new data frame represented a list of all the posts written by influential hackers.

Next the “Model” class was defined. This is where the Word2Vec model was created and the data was pre-processed. First the data was processed using the simple pre-process function within the Gensim library. This found the unique occurrences of words within a post and separated them into individual strings called tokens. This ensured that the model did not keep learning the same word repeatedly thus, improving the performance of the model. Word2Vec focuses more on the words around the word in question rather than how often a word occurs in text meaning that eliminating any reoccurrences of a word in the tokens would not have an effect on the outcome of the analysis. The pre-processing utility also converted all the letters in the words to lower case. Then the Word2Vec model itself was created by calling the Word2Vec algorithm from the Gensim library. The model was stored in a variable called analyser. The parameters set for the model was the window size and minimum count. The window parameter sets how many words should be looked at either side of the word in question. The value for this was set at 10. This value was chosen because research has shown that small values such as 1 or 2 often had little to no effect on the performance of the algorithm. Changes to performance were often found at window sizes of 10, 20, 30 and so forth (Di Gennaro, Buonanno and Palmieri, 2021). Due to the posts often not being very long in length, meaning window sizes of 20 or more would not be practical, it was seen appropriate to set the window size as 10. The minimum count parameter tells the algorithm the shortest word to include when training. For this parameter a value of 2 was chosen. This is because anything less than 2 letters is probably not significant in the meaning of a sentence but anything greater than 2 could be important. Then the model was called and used the build vocabulary function from the Gensim library. This function was given a list of tokenised posts from the iContent variable that had been pre-processed in the first part of this class. This trained the Word2Vec model on data specific to the dataset being used for the project. This is in addition to all pre-training that is included with the Word2Vec model as mentioned in section 2.4 Semantic Analysis. The Word2Vec model was now ready to be used for analysis on the data.

The next class to be defined was the “Analyser” class. This used the Word2Vec model created in the Model class and carried out analysis on the posts from the influential users stored in the iContent data frame. First two list variables were created. The first was called word1List and contained a series of technical terms related to hacking. These terms would form the search words and the Word2Vec model would aim to find words similar to these terms. There were ten search words. These were: SQL, virus, compromise, malware, script, XSS, phishing, spoofing, rat, and encryption. These were found to be some of the most popular hacking terms (Chauhan, 2021). The words hack and hacking was originally included in this list but not used in the final build of the system as later testing deemed that it returned too many false positives, where a post was flagged as cyber threat intelligence but provided no relevant information regarding a specific attack or any indicators of compromise. Removing these words from the word list greatly improved the quality of the reports generated by the system.

The second list variable was called word2List and contained all the vocabulary contained in the text that the Word2Vec model had learned from the posts in the dataset. Then an empty list called searchWords was created. This list would contain all the words that were found to be similar to the hacking terms in word1List and would form the final list of words to be searched for.

The next step involved iterating through both of the word lists using for loops. First word1List was iterated through and then nested within the first for loop was another for loop iterating through word2List. Within this nested loop was the similarity checker. Here the Word2Vec model created in the Model class was used to compare the words in both word lists. This worked by getting the first word in word1List and comparing it to each word in word2List using the Word2Vec similarity function. The output of this function was a score between zero and one. A value of zero meant the words were not at all similar and a value of one meant the words were the exact same. If this similarity score was greater than 0.9 (90%), then it would be added to the searchWord list created earlier. Once this process had compared all the words in word2List to the first word in word1List it would repeat the same process of comparing all the words in word2List to the second word in word1List. This cycle continued until both lists had been fully iterated through meaning that all the words had been compared and a full list of search terms had been generated. A similarity score of greater than 0.9 was used as a starting point and results for different values of similarity score can be seen in section 5.1 Results.

The next part of the Analyser class removed any duplicate words that were present in the searchWords list. These duplicate words occurred when a word in word2List was similar to more than one of the terms in word1List. They were removed by converting the list into a dictionary since dictionaries use discrete (unique) values. The list was then converted back into a list for further analysis.

The next step was to search the original posts for any occurrences of the search terms stored in the searchWords list. The reason that individual words were checked for similarity to the technical hacking terms rather than the entire posts themselves is because Word2Vec is only capable of analysing the similarity of two words at a time and so could not check the entire posts similarity to a particular word or set of words. First an empty list called ctiFound was created. This would contain all the posts which were found to contain one or more of the search words. Then the searchWords list was iterated through to find all the posts in the iContent data frame that contained one of the search words. If there was a positive match and a search word was found in the post, then that post, along with the user ID of the author would be added to the ctiFound list.

The ctiFound list was a list of data frames, where each data frame had the posts relating to a particular search word. This data needed to be combined in order to allow further processing which would generate a STIX report. For this to happen the data was placed into a new data frame called ctiList. All of the data in the ctiFound list was merged into a single data frame using the concatenate function in the Pandas library. The final part of the Analyser class removed any duplicates from the ctiList data frame using the drop duplicates function in the Pandas library. The drop duplicates function checked for duplicate posts and did not use the author ID. This ensured that if more than one post by the same user was flagged as CTI then all the relevant posts would still be included in the ctiList. The ctiList data frame now contained all the posts that were deemed to contain cyber threat intelligence, along with the posts author user ID.

The final class of the system was the “Report Generator” class. This class got the results of the analyser and generated cyber threat intelligence reports in STIX format. For this the STIX Python library by the OASIS Cyber Threat Intelligence Technical Committee (2021) was used. Each instance of CTI found by the analyser was transformed into its own report. It was experimented generating a single report for the entire dataset however this was found to be too large and complex for any real understanding to be gained from it, hence the creation of individual reports for each instance. In order to generate a report for each instance of CTI found in the ctiList data frame a for loop was used to iterate through each row.

The first step in generating the report was to set a namespace. This is a unique way of identifying STIX reports and the creator of the report. For this project a simple name of CTI report was chosen. The next step was to create a STIX package. This used the STIX package function from the STIX external library. This was the overall container for the report to go in and contained all information regarding the cyber threat intelligence being discussed in that report. Then the report was created. The report contained a header and description. The header was defined using the default value contained in the STIX library. The header description was CTI report.

The next step of generating the STIX report was to add an indicator. The indicator contained the actual intelligence of the report. It was used to describe the indicator of compromise or the cyber threat intelligence that had been discovered and was the subject of the report. The indicator title was set as the content of the post. With more time to work on the project this could have been further expanded to actually extract the key information rather than just use the entire post. This would have allowed the information contained in the report to be more relevant and direct. Next the TTP title was set. This is specific information that characterises the behaviour of the attacker. Due to the limited data that was available in the posts of the hacker forums this is hard to define as most people would not discuss attacks in detail in a single post. It was decided that this would be used to contain the user ID of the posts author. The reasoning behind this was that this allowed that user to be investigated further to find more information regarding a particular threat or to find who they were interacting with. More information on how this could be used can be found in section 7.0 Future Work. Then the TTP was added to an activity container. Then the indicator and activity objects were added to the STIX package. The final step of generating the STIX report was to write the STIX package to an XML file. For this the XML function in the STIX external library was used to write the generated STIX package into an XML format. Then this XML was written to a file. The index of each post in the original dataset was used as the filename. For example, the filename of “CTI10.xml” would be used for the post flagged as containing CTI at index 10.

The last part of the program was to write a main function that called each of the classes and provided them with the necessary data. The Model class took data read in from the two CSV files, the Analyser class used data from the Model class and the Report Generator class used data from the Analyser class. The analyser class also needed to call the Word2Vec model that had been created. Once the program had finished running it displayed an analysis complete message on the command line interface to signify that the program had finished running and had carried out its analysis of the data.

4.4 Testing

Once the system was built it underwent testing to ensure that there were no errors and that the logic of the system worked as intended. Due to the work done when creating the prototypes there were no major syntax or runtime errors discovered in the final build of the system. Unit testing was carried out while making the prototypes on individual sections of code to check for any errors and to ensure that the functions worked as intended. Functional testing ensured there were no logical errors present within the program. This was done for both the prototypes and the final build and check key areas where data analysis and manipulation were carried out. The key areas that were checked was when the data was being read into the system, when data was being written to lists or data frame and the output of any loops in the system. One area in particular that had to be checked was the for loops when iterating through values. It was vital to ensure that all values had been checked and included in the system processes. The analysis of the system for logical errors was done by manual analysis by looking at individual processes and examining the input and then determining the expected outputs and seeing if the actual output resembled the expected output. Finally, regression testing was carried out on the final build to ensure that the entire system worked together and that any changes made to the system worked correctly. There were only a few minor refinements made and all these worked correctly. The final product was a fully functioning system and was deemed to have no major errors.

4.5 Connection to OSINT Platforms

The reports generated by the system were in STIX format. This meant that the cyber threat intelligence gathered by the system created could be uploaded to open-source threat intelligence platforms. One platform that could be used to upload the cyber threat intelligence gathered is OpenCTI. This platform accepts cyber threat intelligence in STIX format and uses a series of connectors to act as external APIs (application programming interface). The system created in this project did not automatically connect to an OSINT platform such as OpenCTI, but it could be made to do so. The OpenCTI platform has various communities with open connections where possible cyber threat intelligence can be uploaded to or alternatively a user can create their own community to host cyber threat intelligence (OpenCTI, 2018). These communities allow STIX reports to be uploaded either manually or automatically through the use of their respective connectors. This allows the cyber threat intelligence found to be used by the cyber security community and can help to protect organisations from cyber attacks.

4.6 Experiments

The final part of the project, once the system had been built and fully tested, was to run experiments and gather results from the system. Ultimately, the system was judged based on the quality of the reports produced and whether the cyber threat intelligence it had found was actually intelligence or not. If the report did not provide useful information, it was a false positive as it had been incorrectly classed as intelligence by the system.

In order to determine the quality of the reports they had to be manually reviewed. Since hundreds of reports were generated from the data, a subset of 50 reports were reviewed to provide a representation of the results. The results were chosen at random. This was to ensure the chance of a fair representation of the data but also to avoid looking at only one type of intelligence found e.g., not just using reports about malware. Using this method did have its limitations as not all the data was analysed and there was a chance valuable insight would be lost however due to the amount of data present it was deemed to be appropriate and the best way to gather the results. With more time to carry out the project the data could be analysed in more depth or an automated system could be developed to gather the results. More on this can be found in section 7.0 Future Work.

Each report that was analysed would be given a classification of good, bad or vague. A report that was classed as being good would provide clear information about a specific cyber threat or indicator of compromise. It would be relevant and could be used by organisations to help inform them about or protect them from either a specific cyber attack or provide information about a specific technique being used by cyber adversaries. A report that was classed as being bad would provide no information about a cyber attack or technique and would be completely irrelevant. A report that was given a classification of vague would provide some insight into a cyber attack or technique however the report alone would not provide enough information to be acted upon. Further analysis would be needed to determine if the intelligence gathered provided useful information. More on how this could be built upon is discussed in section 7.0 Future Work.

In addition to classifying the quality of the reports generated, further experiments were carried out to determine the ideal values of hyperparameters. The main parameter experimented with was the similarity score in the analyser class. Different values of similarity would affect how many words were present in the word list and consequently affect the number of posts flagged as containing cyber threat intelligence. It was hypothesised that a higher value for the similarity score would result in less posts being flagged and also lead to an increase in the quality of the reports being generated. The results gathered can be found in section 5.1 Results.

5.0 Results and Evaluation

The results gathered from the project can be seen in section 5.1 Results. The results have been collated and put into a table. The results will be briefly discussed and explained in this subsection. Then in section 5.2 Evaluation, the results will be examined in detail and the effectiveness of the system will be discussed. Finally, in section 5.3 Discussion, the project as a whole will be evaluated and the objectives will be analysed to determine the successes of the project.

5.1 Results

The results gathered from the project can be seen in *table 3, table 4* and *table 5* below. The results were gathered for three different values of similarity score. The metrics gathered for each value of the similarity score were the total number of reports generated, the number of good reports, the number of bad reports and the number of vague reports. To help with analysis of the results the ratio of good to bad to vague reports was calculated. The number of reports generated for each value of similarity score can be seen in *table 3*.

Similarity Score Value (%)	Total Number of Reports
90	257
80	2810
70	3629

Table 3: Results

A breakdown of the results showing the number of good, bad and vague reports can be seen in *table 4* and *table 5*. It should be noted that this analysis was carried out on a subset of 50 reports.

Similarity Score Value (%)	Number of Good Reports	Number of Bad Reports	Number of Vague Reports	Ratio (Good/Bad/Vague)
90	16	12	22	8:6:11
80	11	14	25	11:14:25
70	14	9	27	14:9:27
Average	**13.6**	**11.6**	**18.0**	**13.6:11.6:18.0**

Table 4: Results breakdown

Similarity Score Value (%)	Good Reports (%)	Bad Reports (%)	Vague Reports (%)
90	32.0	24.0	44.0
80	22.0	28.0	50.0
70	28.0	18.0	54.0
Average	**27.3**	**23.3**	**49.3**

Table 5: Results breakdown as percentage

The analysis done when determining the number of good, bad and vague reports was based on 50 randomly selected reports. This was done due to the large number of reports generated for each value of similarity score. The ratio was calculated using these 50 reports and aims to provide a representation of the entire dataset.

5.2 Evaluation

The results show that a higher number of reports is generated the lower the value of the similarity score. This is as expected and is in line with the hypothesis stated in section 4.6 Experiments. The hypothesis was that a higher value for the similarity score would result in less posts being flagged and also lead to an increase in the quality of the reports being generated. The reason more results are generated with lower values of similarity score is because more words will be included in the words list used by the Word2Vec model. Having more words in the words lists means there are more matches for possible cyber threat intelligence. One interesting point from looking at the results in *table 3* is the significant jump in reports generated between similarity scores of 90% and 80%. There is an increase of 2553 reports yet the difference between similarity scores of 80% and 70% is only 819. The sharp increase between 90% and 80% could be explained as it is possible the majority of the words present in the dataset have a similarity of greater than 80% and there are less words that have a similarity score of less than 80%. Furthermore, since most posts are quite short and only contain a few sentences, it could be possible that most words are given a higher value similarity score than would be the case in a longer passage of text. If this is the case it would be a limitation of the Word2Vec model and could be solved by increasing the amount of data used to train the model.

Analysing the data further to look at the percentage breakdowns shown in *table 5* show some more interesting trends in the results. As predicted in the hypothesis the largest number of good reports were generated when using a similarity score of greater than 90%. What is perhaps more interesting though is the fact that a similarity score of greater than 70% produces more good results than a similarity score of greater than 80%. This can be explained by the method used to analyse the results. Since only 50 results were analysed and they were chosen at random this meant that some of the good reports would have been missed and there is a chance that more of the bad and vague reports were picked up. This combined with the fact that a similarity score of greater than 70% produced more reports than a similarity score of greater than 80% meaning there was more reports to randomly choose from. This is a major limitation of the method of analysis used and if a full analysis of all the reports generated had been carried out it is expected that a similarity score of greater than 80% would have a higher percentage of good reports than a similarity score of greater than 70%.

Another interesting trend in the results is the high proportion of vague results. On average 49.3% of the reports generated were classed as being vague. A similarity score of greater than 90% produced the lowest percentage of vague reports with 44% of reports being classed as vague. Across all three similarity scores tested, the majority of reports generated were classed as being vague. Further analysis into the causes of this has determined two major factors.

The first major factor is the posts in the dataset. Since these posts are written by people on what is essentially a hackers social media platform, many people talk in the same manner as they would on other social media and not necessarily in a structured format. Posts are often filled with abbreviations and slang and some people may use words in different ways to their original meaning. This can make it harder for semantic analysis algorithms to correctly understand the way a term is being used and affect the interpretation of words (Kolajo *et al.*, 2020). Furthermore, posts tend to be relatively short in length containing just a few sentences. Semantic analysis algorithms often work best when there are given large passages to analyse. Using shorter passages of text can lead to words being incorrectly identified as being similar. This is especially the case with Word2Vec since it uses words in relative proximity to word being analysed depending on the size of the window parameter set (Di Gennaro, Buonanno and Palmieri, 2021).

The second major factor for the high number of reports being classed a vague is the system created, specifically the way posts are flagged as containing cyber threat intelligence and also the way reports are created. The way the system works is that posts that contain a word that either is a hacking term or is similar to one will get flagged as CTI. Then a report is generated where the post that has been flagged is set as the indicator. There are two ways this process could be improved to produce less vague reports. The first method is to only flag posts as CTI if they contain several words that are similar to hacking terms. For example, if the requirement is for at least two words to be over 90% similar to a hacking term then a post that says “The hack” would not be flagged unless the word “the” is over 90% similar. A post that says “Hack the server” would be flagged provided either the words “the” or “server” are over 90% similar to a hacking term. The exact number of words in a post that would have to be over a set similarity score would need some experimentation to find the optimal number but using this method may help to reduce the number of both bad and vague reports produced. The second method to improve the system is to change the way that reports are produced. Rather than just using the entire post as the indicator in the report, the key information could be extracted from the post. Combined with the first method to improve the system previously mentioned, the indicator could be set to just include the terms with a set similarity score. One potential downside of this method could be that posts may lose some context and if just several words are stated it could actually increase the number of vague reports. Ultimately further testing and investigation is required to discover what methods work best to reduce the number of vague reports produced by the system.

Some examples of generated reports that were classed as good include a user who was explaining in detail how they were going to spoof their MAC address and was then looking for guidance on getting past Windows Defender without being blocked. While no specific target was mentioned they did go into detail of how they were carrying out the attack. Another example of a good report is a user talking about using safe mode on Windows XP to bypass the login screen and then carry out an EOP (elevation of privileges) attack. It looks like it could be referencing the known exploit CVE-2004-2339 but the post does not give enough detail to say for certain if it is that vulnerability. There are also reports from posts explaining how to create certain scripts for XSS (cross site scripting) and the best way to deliver malware payloads. Some examples of reports that were classified as being bad include users talking about the best web browser, someone saying they did not read the code and a post that just says congratulations. The presence of these reports suggest that the system needs further improvements to avoid producing these types of report.

Some examples of reports that were classified as being vague include users talking about carrying out a DDos (distributed denial of service) attack but not specified either the target or technique. Another report has a post talking about a user that can supposedly hack any email. This could be the identity of a professional cyber-criminal however the username cannot be verified and even if it could be it would still be extremely difficult to trace to the actual person the account belongs to. There are also several posts claiming to have links to a website where users can get malware. The authenticity of these links cannot be verified without a secure sandbox environment. Even if the site is legitimate, it is likely that the malware itself is behind a paywall. These types of posts could prove extremely useful if further investigation is carried out however they could also prove to be false leads. Without more information it is impossible to tell how useful the cyber threat intelligence gathered from these posts really is.

A factor that has not been measured in the results but is worth noting is that using a similarity score value of less than 90% significantly increases the runtime of the program. This is because significantly more results are found for the lower values of similarity score meaning more data is being processed and more reports are generated. When generating cyber threat intelligence, it is unlikely that this increase in runtime would be an issue but if it was being used for a time sensitive issue then it could prove a problem. To resolve this issue using a higher value of similarity score should ensure the system completes its analysis within seconds.

Overall, the results gathered are very interesting and provide an insight into the type of posts on hacker forums. While the cyber threat intelligence reports generated by the system could be improved, some of the reports are already very useful to identify potential cyber threats. The system created has achieved the objective of identifying cyber threats from influential hackers (objective 3). Although the accuracy of the system needs improvement, the basis is there to be built upon. The system also turns its findings into actionable intelligence that is in a compatible format with open-source threat intelligence platforms (objective 5).

5.3 Discussion

The project overall has achieved what it set out to accomplish and provided an interesting set of results. The system created can produce cyber threat intelligence reports using the STIX format. However, there is significant work to be done in terms of the system accuracy and the quality of the reports generated. In order to improve the system more work needs to be done on how a post is classed as cyber threat intelligence and what information is then used to produce the report.

Further investigation could make use of additional datasets within the CrimeBB database. The dataset used in this project was one of many provided by Cambridge Cyber Crime Lab. Using additional datasets would test how adaptable the system is and could find more cyber threat intelligence. The dataset used for this project has proven to work very well. The system created has successfully analysed the dataset for cyber threat intelligence and the project as a whole has accomplished all the objectives set out at the start of the project.

6.0 Conclusion

This project has found that a system can be created to intelligently analyse attacker behaviour on the dark web. Specifically, a system was created that makes use of artificial intelligence and machine learning to automatically produce cyber threat intelligence reports based on analysis of data from dark web hacker forums. The cyber threat intelligence reports generated from this system vary in their usefulness, but the initial results are promising. With further work the results could be improved further to make the system more resilient to noise in the data and to produce less false positive results. Almost half of the reports generated are classed as being vague and work to improve the quality of these reports is the main area that warrants further investigation. Despite this an average of 27.3% of reports were classed as being good with the best percentage of good reports present when a high value of similarity score was used for the analyser. At higher values of similarity score 32% of the reports created were classed as being good.

The project has completed the aim and the objectives set out at the start of the project. It has examined how to identify different types of cyber threat (objective 1) through research carried out into topic discussed in the literature review. Cyber threats from the most influential hackers within the CrimeBB dataset have been identified through the use of the user reputation and influencer score created by Atique, Arshad and Mubashir (2021) which was then used to narrow down the number of users and posts that were analysed by the system (objective 2). A text mining system using semantic analysis was created which then identified potential cyber threats (objective 3). The accuracy of the system has been calculated during the results gathering process (objective 4). Finally, the findings of the system were converted into a format that is compatible with most major open-source threat intelligence platforms (objective 5). This took the form of STIX reports. The aim has been accomplished by fulfilling all of the objectives and by the fact that the system works as intended. While there are some improvements that can be made, the system works and has produced some good results.

Ultimately this project has been a success. It has accomplished the aim and all the objectives set out at the start of the project. Furthermore, the results gathered by the system offer an interesting insight into what is happening on hacker forums and the dark web. The types of cyber attack and the techniques used to carry out these attacks can be discovered by analysing the reports generated by the system created in this project. Information gathered by this system either on this dataset or using other data could prove vital in helping to understand cyber attacks and cyber adversaries and could possibly help prevent cyber attacks in the future.

7.0 Future Work

The work carried out in this project could be expanded upon by further study and development. One of the key areas that could be improved upon by future work is reducing the number of vague reports. This could be done by only flagging posts as cyber threat intelligence if they contain several words that have a high similarity score to hacking terms. Another area where more work could be carried out is for a more in-depth analysis on the reports generated. For the results of this project only a subset of the overall reports generated were analysed. By analysing more of the reports, a better and more accurate representation of the data could be gathered. Another interesting area for future research could be to do more analysis on the users. This could look at what users interact with each other and generate a list of threats with high amounts of CTI generated. This could prove an interesting source of information and allow more cyber threat intelligence to be gathered and with more context. These suggestions have the potential to improve the work carried out in this project even further to ensure that the system created is the best it can be.

8.0 References

Accenture (2021) *Triple digit increase in cyberattacks: What next?* Available at: https://www.accenture.com/us-en/blogs/security/triple-digit-increase-cyberattacks (Accessed: 14 October 2021).

Atique, A., Arshad, J. and Mubashir, M. (2021) 'Identifying Influencers on Dark Web Forums'.

Balaji, S. (2012) 'Waterfall vs v-model vs agile : A comparative study on SDLC', *WATEERFALL Vs V-MODEL Vs AGILE : A COMPARATIVE STUDY ON SDLC*, 2(1), pp. 26–30.

Barnum, S. (2014) 'Standardizing Cyber Threat Intelligence Information with the Structured Threat Information eXpression'.

Beel, J. *et al.* (2016) 'Research-paper recommender systems: a literature survey', *International Journal on Digital Libraries*. Springer Berlin Heidelberg, 17(4), pp. 305–338. doi: 10.1007/s00799-015-0156-0.

Bhattacharyya, D. K. and Kalita, J. K. (2016) *DDoS attacks : evolution, detection, prevention, reaction, and tolerance*. Boca Raton.

Cartes, R. F. (2020) *Data science meets cybersecurity to protect your web application from bots.* Available at: https://rapidminer.com/resource/data-science-for-cybersecurity-identifying-and-mitigating-threats-with-rapidminer/ (Accessed: 21 October 2021).

Chauhan, B. (2021) *20 Must-Know Hacking Terminologies To Safeguard Your Online Business from Hackers*. Available at: https://www.getastra.com/blog/knowledge-base/hacking-terminologies/ (Accessed: 30 January 2021).

Choo, K. K. R. (2011) 'The cyber threat landscape: Challenges and future research directions', *Computers and Security*, 30(8), pp. 719–731. doi: 10.1016/j.cose.2011.08.004.

Church, K. W. (2017) 'Emerging Trends: Word2Vec', *Natural Language Engineering*, 23(1), pp. 155–162. doi: 10.1017/S1351324916000334.

Cisco (2019) 'Talso Intelligence', pp. 1–7.

Crossley, J. and Jansen, D. (2021) *Saunders' Research Onion – What is it?* Available at: https://gradcoach.com/saunders-research-onion/ (Accessed: 7 January 2022).

Cyware (2021) *Considerations for Choosing the Right Threat Intelligence Platform*. Available at: https://cyware.com/educational-guides/cyber-threat-intelligence/considerations-for-choosing-the-right-threat-intelligence-platform-034c (Accessed: 10 January 2022).

Echosec Systems (2021) *Echosec*. Available at: https://www.echosec.net/ (Accessed: 19 October 2021).

Gao, P. *et al.* (2021) 'A System for Automated Open-Source Threat Intelligence Gathering and Management', *Proceedings of the ACM SIGMOD International Conference on Management of Data*, (1), pp. 2716–2720. doi: 10.1145/3448016.3452745.

Di Gennaro, G., Buonanno, A. and Palmieri, F. A. N. (2021) 'Considerations about learning Word2Vec', *Journal of Supercomputing*. Springer US, 77(11), pp. 12320–12335. doi: 10.1007/s11227-021-03743-2.

Gupta, S. (2018) *Sentiment Analysis: Concept, Analysis and Applications*. Available at: https://towardsdatascience.com/sentiment-analysis-concept-analysis-and-applications-6c94d6f58c17 (Accessed: 20 October 2021).

Hatta, M. (2020) ‘Deep web, dark web, dark net: A taxonomy of “hidden” Internet’, *Annals of Business Administrative Science*, 19(6), pp. 277–292. doi: 10.7880/abas.0200908a.

Jansson, K. and Von Solms, R. (2013) ‘Phishing for phishing awareness’, *Behaviour and Information Technology*, 32(6), pp. 584–593. doi: 10.1080/0144929X.2011.632650.

Kaspersky (2020) *What is the Deep and Dark Web?* Available at: https://www.kaspersky.com/resource-center/threats/deep-web (Accessed: 13 October 2021).

Kolajo, T. *et al.* (2020) ‘A framework for pre-processing of social media feeds based on integrated local knowledge base’, *Information Processing and Management*. Elsevier, 57(6), p. 102348. doi: 10.1016/j.ipm.2020.102348.

Landauer, T. (2002) ‘Latent Semantic Analysis : Theory , Method and Application’, *CSCL ’02: Proceedings of the Conference on Computer Support for Collaborative Learning: Foundations for a CSCL Community*, pp. 742–743. Available at: https://dl.acm.org/doi/abs/10.5555/1658616.1658815.

Lee, S. and Shon, T. (2017) ‘Open source intelligence base cyber threat inspection framework for critical infrastructures’, *FTC 2016 - Proceedings of Future Technologies Conference*. IEEE, (December), pp. 1030–1033. doi: 10.1109/FTC.2016.7821730.

Li, B. *et al.* (2019) ‘Scaling Word2Vec on Big Corpus’, *Data Science and Engineering*, 4(2), pp. 157–175. doi: 10.1007/s41019-019-0096-6.

Li, D., Zhou, X. and Xue, A. (2020) ‘Open source threat intelligence discovery based on topic detection’, *Proceedings - International Conference on Computer Communications and Networks, ICCCN*, 2020-Augus. doi: 10.1109/ICCCN49398.2020.9209602.

Liu, B. (2012) *Sentiment Analysis and Opinion Mining Mining. Synthesis Lectures on Human Language Technologies. [Draft]*, *Synthesis Lectures on Human Language Technologies*. Available at: http://www.morganclaypool.com/doi/abs/10.2200/S00416ED1V01Y201204HLT016.

Liu, C. and Wang, Y. M. (2012) ‘On the connections between explicit semantic analysis and latent semantic analysis’, *ACM International Conference Proceeding Series*, pp. 1804–1808. doi: 10.1145/2396761.2398521.

Mavroeidis, V. *et al.* (2021) ‘Threat Actor Type Inference and Characterization within Cyber Threat Intelligence’, *International Conference on Cyber Conflict, CYCON*, 2021-May(303585), pp. 327–352. doi: 10.23919/CyCon51939.2021.9468305.

McClure, S., Scambray, J. and Kurtz, G. (2012) *Hacking Exposed 7 : Network Security Secrets and Solutions*. McGraw-Hill Publishing.

Medhat, W., Hassan, A. and Korashy, H. (2014) ‘Sentiment analysis algorithms and applications: A survey’, *Ain Shams Engineering Journal*. Faculty of Engineering, Ain Shams University, 5(4), pp. 1093–1113. doi: 10.1016/j.asej.2014.04.011.

Menges, F., Putz, B. and Pernul, G. (2021) ‘DEALER: decentralized incentives for threat intelligence reporting and exchange’, *International Journal of Information Security*. Springer Berlin Heidelberg, 20(5), pp. 741–761. doi: 10.1007/s10207-020-00528-1.

MISP (2021) *User guide of MISP intelligence sharing platform*.

O’Leary, M. (2019) *Cyber Operations Building, Defending, and Attacking Modern Computer Networks*. Berkeley.

OASIS Cyber Threat Intelligence Technical Committee (2021) ‘STIX^{TM} Version 2.1 OASIS Standard’, (June). Available at: https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.pdf.

Oasis Open (2021) 'TAXII Version 2.1'.

Omitola, T., Riós, S. A. and Breslin, J. G. (2015) *Social Semantic Web Mining*, *Synthesis Lectures on the Semantic Web: Theory and Technology*. doi: 10.2200/S00623ED1V01Y201412WBE010.

OpenCTI (2018) *OpenCTI Public Knowledge Base*. Available at: https://www.notion.so/OpenCTI-Public-Knowledge-Base-d411e5e477734c59887dad3649f20518 (Accessed: 10 January 2022).

OpenJS Foundation & Contributors (2013) *Node-Red*. Available at: https://nodered.org/ (Accessed: 21 October 2021).

Pastrana, S. *et al.* (2018) 'CrimeBB: Enabling cybercrime research on underground forums at scale', *The Web Conference 2018 - Proceedings of the World Wide Web Conference, WWW 2018*, pp. 1845–1854. doi: 10.1145/3178876.3186178.

Perambai, A. (2020) *Theory behind Word Embeddings in Word2Vec*. Available at: https://medium.com/analytics-vidhya/theory-behind-word-embeddings-in-word2vec-858b9350870b (Accessed: 10 January 2022).

Photon Research Team (2019) *Dark Web Monitoring: The Good, The Bad, And The Ugly*. Available at: https://www.digitalshadows.com/blog-and-research/dark-web-monitoring-the-good-the-bad-and-the-ugly/ (Accessed: 7 January 2022).

Rebecca M. Blank. Patrick D. Gallagher (2012) 'NIST Special Publication 800-30 Revision 1 - Guide for Conducting Risk Assessments', *NIST Special Publication*, (September), p. 95.

Runciman, B. (2020) 'Cybersecurity report 2020', *Itnow*, 62(4), pp. 28–29. doi: 10.1093/ITNOW/BWAA103.

Saunders, M., Lewis, P. and Thornhill, A. (2019) *Research Methods for Business Students*. Pearson Education.

Sikorski, M. and Honig, A. (2012) *Practical malware analysis the hands-on guide to dissecting malicious software*. San Francisco : No Starch Press.

Steyvers, M. and Tenenbaum, J. B. (2005) 'The large-scale structure of semantic networks: Statistical analyses and a model of semantic growth', *Cognitive Science*, 29(1), pp. 41–78. doi: 10.1207/s15516709cog2901_3.

TesnorFlow (2021) *Word2Vec*. Available at: https://www.tensorflow.org/tutorials/text/word2vec (Accessed: 22 December 2021).

The MITRE Corporation (2017) 'MAEC 5.0 Specification'.

The pandas development team (2020) *pandas-dev/pandas: Pandas*, *Zenodo*. Available at: https://doi.org/10.5281/zenodo.3509134 (Accessed: 25 November 2020).

Tundis, A., Ruppert, S. and Mühlhäuser, M. (2020) 'On the automated assessment of open-source cyber threat intelligence sources', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12138 LNCS, pp. 453–467. doi: 10.1007/978-3-030-50417-5_34.

Witten, I. H. *et al.* (2017) *Data Mining: Practical Machine Learning Tools and Techniques*. Fourth Edi. Morgan Kaufmann.