Visual Analysis of Spam Campaigns based on Network Modelling

by

Chitrarth Patel

B.Eng. Gujarat Technological University, 2019

A Project Report Submitted in Partial Fulfillment of the Requirements for

the Degree of

MASTER OF ENGINEERING

in the Department of Electrical and Computer Engineering at the University of Victoria,

Victoria, British Columbia, Canada

University of Victoria

Supervisory Committee

Visual Analysis of Spam Campaigns based on Network Modelling

by

Chitrarth Patel

B.Eng., Gujarat Technological University, 2019

Dr. Issa Traore,

Supervisor (Department of Electrical and Computer Engineering)

Dr. Waleed Yousef,

Departmental Member (Department of Electrical and Computer Engineering)

Table of Contents

List of Tables

List of Figures

Abstract

With the growing Internet use, spamming methods have evolved, and attackers have modernized the attack strategies, making them more scalable using botnets. Botnets play a crucial role in spreading these spam email campaigns. A single individual or a group usually controls botnets. However, the same attacker or group can run different campaigns in many cases. Therefore, detecting the campaigns run by the same entity is crucial. Furthermore, it helps the analyst to capture stronger evidence against the attacker.

The report proposes an approach for exposing coordinated spam campaigns initiated by single controlling entities. It uses network modelling and creates network graphs based on different behavioural traces for spam campaigns. Campaigns have a strong connection among them if they have similar behaviours. The proposed approach can also be used in investigating other cybersecurity attacks.

Chapter 1 Introduction

1.1 Context

Email is a well-acknowledged medium of exchanging messages, which is used to send instant messages, media, and documents around the world. This growing popularity also leads to the increasing number of cyber-attacks via emails. The share of spam email was on average 46.56% in the second quarter of 2021 in global email traffic [1]. Spam emails are irrelevant and unwanted emails sent by attackers via email with the motive of advertising, releasing malware, stealing the credentials of the victim or phishing. Moreover, a tremendous amount of spam in the mailbox causes a performance issue, storage space waste and inconvenience while using the email service.

Spamming methods have evolved over time, and attackers have modernized the attack strategies; they usually send countless spam emails to numerous Internet users like a flash. Botnets play a crucial role in spreading these spam email campaigns. Bots can be referred to as compromised computers on the Internet. The network of these compromised computers is called botnets. Botnets are usually controlled by a single individual or a group. There are multiple unethical uses of these botnets. They are used to perform Distributed Denial-of-Service (DDoS) attacks, run spam email campaigns, steal users' data, and many more. Multiple pieces of evidence are there to show that spam email campaigns are a driving force in monetizing these botnets.

In many cases, the same botnet can be used to run different campaigns, and these campaigns are usually executed by the same controlling entity. Therefore, it is crucial to detect the campaigns run by the same entity. Furthermore, it helps the analyst capture stronger evidence against the attacker.

This report aims to elaborate on techniques for detecting coordinated spam campaigns initiated by single controlling entities. A framework is also proposed to create a network graph for spam campaigns based on behavioural traces. Campaigns have a strong connection among them if they have similar behaviours. These behavioural traces to develop a network, can be extracted from the available metadata. The proposed framework of network graph creation can also be applied to other cybersecurity investigation problems.

1.2 Objectives and Approach

Daily numerous malicious spam campaigns get detected delivering malware or stealing victims' information across different regions. These spam campaigns usually are run by botnets. This report shows how the unsupervised approach using graph analytics and visualization based on graphs [2] and network theory [3] can help us gain insights into the strategic dynamics of spam campaigns and provides a better understanding of large-scale spam operations operated by botnets controlled by the same entity.

The unsupervised approach proposed here is more general in allowing multiple similarity criteria to detect human coordination control in multiple different spam campaigns. Proposed method is based on association network construction, which follows a rule that the more frequent the same attributes are shared among spam campaigns, the more chances are there of the same human association. Here, different networks are created based on attributes/traces like IP, domains, timelines, and many more. It helps in investigating campaigns where the attacker uses some obfuscation techniques on common attributes used by multiple campaigns to inject more variance into the different campaigns. The generated network data is stored in a database and handled through REST API Services [4] for cyber security investigational and visualization purposes.

1.3 Report Outline

The remaining chapters in this report are structured as follows. In Chapter 2, background on security visualization and the Campaign Data Model is discussed. Chapters 3 presents the working of proposed method in detail. Chapter 4 presents the efficiency, scalability, and one case study of investigation using the framework. Chapter 5 contains the concluding remarks and an outline of future work.

Chapter 2 Background

2.1 Security Visualization

Legg proposed in [5] visual analytics technique to gain insights about insider threat activity based on user profiling and derived features. In the paper, visual analytics is used for exploratory visualization of detection results and user activity. It also provides explanatory visualization to support the detection technique results. It uses interactive Principal Component Analysis (iPCA) for the assessment of the anomaly detection results.

Nance and Marty proposed in [6] a method of using bipartite graphs to visualize insider activity threats. In this approach, the bipartite graph has two types of nodes. One represents a user, and another means the activity. An edge between nodes represents the action performed by the user node on the activity node. But this approach is not scalable to detect a large number of entities and characteristics. Moreover, it is helpful in investigating the data rather than predicting the potential threats.

In [7] , Zhuang et al. extracted specific characteristics from spam email data, and used network analysis techniques to characterize the botnets of email spam campaigns (a bunch of emails having common characteristics). The authors provided a detailed study in tracing the botnets behind email spam campaigns and opened a new direction in understanding botnet activities.

The approach presented in this report uses different dimensions of coordination (i.e., IP, domains, timelines, and many more) to develop a generic approach to detecting coordinated campaigns initiated by the same entity. In contrast to other approaches, network graph modelling based on spam email campaign properties is used to find clusters of related spam email campaigns. Moreover, investigating different dimensions of coordination can help in gathering strong evidence of having a common attacker behind the totally different looking campaigns.

2.2 Campaign Data Model and Dataset

The Information Security and Object Technology (ISOT) Research Lab provided the spam email campaigns dataset obtained from one of its industry partners. All the records containing campaigns and emails data are stored in a central document database accessible through an API [8]. Each document includes general information, DNS records, WHOIS and WHOWAS and geographical information about the campaign. Along with all the information about campaigns, each campaign document also consists of emails belonging to the spam campaign. The email data in the campaign document includes data about the email header, text, attachments and URLs in the email. Table 2.1 shows all the feature description provided in each campaign document and table 2.2 describes the features included in email record documents.

Feature	**Description**
General Info	Describes campaign name, type, threat type, source information
DNS Info	Describes IP, domain, name servers' information

WhoIs Info	Contains registrant's information (i.e., Name, City, Country etc.)
WhoWas Info	Contains WhoWas snapshot of registrant's information
Geolocation Info	Contains Geolocation information of spammer
Message Info	Email records data of the campaign

Table 2.1. Spam Campaign document description

Feature	**Description**
Header Info	Data stored in spam email header
Readability Info	Key information about body and subject data (i.e., length, content type)
Email Layout Info	Data about the structure of the email (i.e., HTML and text related data)
Language Info	Data about top bigrams, top words, language used in the email
Attachments	Email attachments data (i.e., name, format, size, hash etc.)
URLs	Information about the URLs in the email (i.e., URL, Redirect URL etc.)

Table 2.2. Spam email document (Message Info) description

Each spam campaign record of the dataset is characterized by several features. First, the campaign data is bifurcated into the campaign metadata and the data of emails belonging to the campaign. The campaign metadata contains all the information about campaigns such as DNS Information, Information, WhoIs and WhoWas Information, and the geographic location including country, ISP, and ASN lookup data. In addition, all the email information such as sender and receiver email address, email header information, body and attachments data, and any embedded URLs in the emails are stored in the email data section. All the extracted features available in the dataset are described in Tables 2.1 and 2.2.

The dataset contains a total of 51,784 spam campaigns. Each of them is described by about 35 different features. Each spam campaign record consists of an average of eight spam emails.

The raw spam campaign data is complex. To perform the network modelling, we need to understand what information the features hold and how it can be used to uncover discrete connections among spam campaigns. Identifying the behaviour similarity is really important in linkage analysis. It means finding the properties commonly shared among various campaigns. The relevant features that can potentially demonstrate high behavioural similarity were extracted for this multi-criteria analysis. The selected features were:

IP Address: Source IP address of the email is a crucial identity trace. Assuming that campaigns originated from a specific region can demonstrate IP and other geographical metadata similarities.

Domain: Like Source IP Addresses, the domain from which the emails were sent was also included.

Sender email: The sender email address in the email header helps reveal specific patterns in spam campaigns. Common keywords are usually found in emails belonging to different campaigns.

Time: The distribution of spam emails from different campaigns sent over the same period of time can help detect the temporal dynamics of the emails. Here, the time frame of one day is used to define the distribution of emails.

Any individual feature from above can be helpful in detecting commonality in emails, but alone they are not enough for detecting linkage or coordination among campaigns. Thus, the combined outcome, including results from all the features, is used in decision making.

Chapter 3. Proposed Framework

3.1 Bipartite Graph Generation

After extracting all the features from the dataset, the next step is to build a bipartite network graph [9] consisting of campaigns with the features mined from the emails belonging to those campaigns. From the dataset, a list of tuples is generated. Every tuple has the campaign id and a feature used in the email of that campaign. Four types of bipartite graphs are created: IP, domain, source email, and timeline bipartite graphs. Each Bipartite graph represents two disjoint sets of nodes. One is campaigns IDs, and another is a property (i.e., one of the extracted features: IP, Domain, Sender Email, Day). There will be an edge between a campaign ID (node) and a property (node), if at least one email from the given campaign has that property. The bipartite graph used in the modelling is unweighted and undirected. Thus, if the same campaign has one property used in multiple emails, it will not strengthen the association. It is not suspicious if a campaign uses the same property in many of its emails [10]. Algorithm 1 depicts the steps involved in the bipartite network generation. Algorithm 1 requires two arguments. The first one is $L_{campaigns}$, which contains all the campaign data records. The second parameter is P_{name}, which denotes the property field name to be extracted from the campaign records. Here, four features, (IP, domain, source email, and time) are used as input to create four different bipartite networks. The algorithm will iterate through each email document in every campaign documents. If the property specified in the input is there in the email document, a tuple with that property value and the campaign ID will be added to the edge list. The output edge list will be the input for the association generation algorithm. Using this 2-step framework, four different networks corresponding to IP, domain, source email, and active time are created. In Figure 3.1, The unweighted bipartite network is described. The set of nodes V in the right represents all property value nodes and the set U represents all the campaign ID nodes.

Algorithm 1: Bipartite Network Generation

Input: $L_{campaigns}$, P_{name}
Output: G

```
L_campaigns = list containing campaigns data
P_name = property field name
G : edge list containing tuples
for each campaign C_i in L_campaigns do
    for each email E_i in C_i do
        if P_name exist in E_i then
            add (C_i, value of P_name) in G
V_c = set of unique campaign IDs
P_v = set of unique property values
G = { (i, j) | i ∈ V_c, j ∈ P_v }
return G
```

Listing 1. Bipartite network generation algorithm

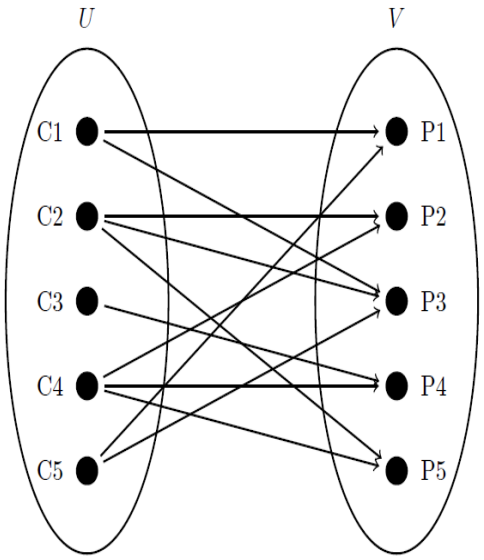

Figure 3.1 Creating the bipartite network of campaign nodes and property nodes (i.e. IP address, domain etc.)

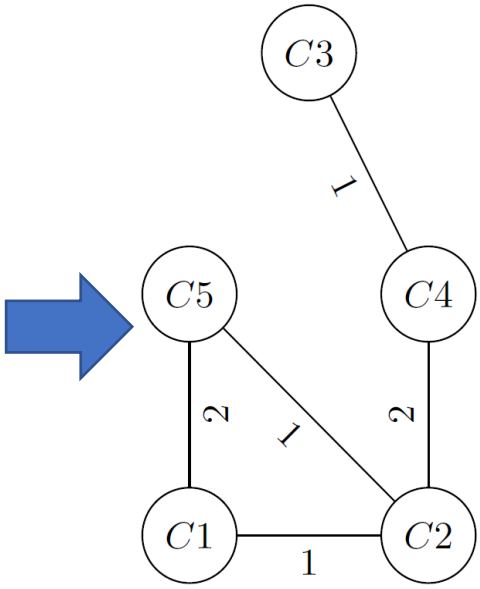

Figure 3.2 Association network of campaigns based on bipartite network.

3.2 Association Graph Generation

After creating the bipartite network graph for each feature, the next step is to transform it into the association graph of campaigns. For each pair of the campaigns connected to common property in the bipartite graph, an edge is added between those campaigns in the association graph [11]. For example, two campaigns C1 and C2 have a shared IP address in one of their emails; an edge between them will be added in the association graph. Moreover, if any other email in both of these campaigns shares the same IP, the weight of the edge between these campaigns will be increased. The weight of an edge in the association graph is computed through co-occurrence [12]. The edge weight in this network helps quantify the strength of behavioural similarity among campaigns. A high edge weight corresponds to more substantial indicator of a single entity controlling these campaigns. Algorithm 2 depicts the association network generation steps. The algorithm takes the incoming bipartite network's edge-list and creates key-value pairs. The key in the key-value pairs is the property value, and the value is a list of campaign IDs that include that property value in any of their emails. Then, the algorithm processes all the key-value pairs to generate the weighted association network of campaign IDs. Here, the weight between two campaign IDs denotes the number of common property values between two campaigns. This algorithm generates a weighted network as described in Figure 3.2. The following listing describes association network generation algorithm.

Algorithm 2: Association Network Generation

Input: G
Output: G_A

```
G = edge list of tuples containing campaign ID and property
G_A = edge list containing campaign nodes and weights
V = set of unique campaign IDs in G
W = set of edge weights between campaigns
L = hashmap of key value pairs
for each property p in Input do
    l_k = list of campaigns connected to property p
    add p as a key and l_k as a value in L
for each l_k in L do
    for all pair of campaigns (i,j) in l_k do
        if w_ij not in W then
            set w_ij = 1
            add w_ij to W
        else
            w_ij = old w_ij + 1
E = { (i, j, w_ij) | w_ij > 0 , w_ij ∈ N }
return G_A = (V, E)
```

Listing 2. Association network generation algorithm

3.3 Clustering

While the association network generated can provide great insights to the investigator, finding clusters of campaigns that are strongly related to each other is also important [13]. With this objective, a clustering algorithm was applied to the association network. Any general community detection algorithm to find highly connected components can be used to find clusters or cliques in the network. Here, k-clique community detection using the percolation method was used to find campaign communities in association networks. Moreover, in one of the case studies, Girvin-Newman algorithm is used to detect clusters in the association network.

3.4 RESTful API Interface

All the association network data is stored in the MySQL database. To allow the investigator to utilize the association network data stored in the database for the investigation, the association property networks data is put over the RESTful APIs. RESTful APIs enable quick and secure access of the data over the HTTP endpoints. The APIs are developed in the Flask framework. Flask is a micro web framework based on Python. Flask helps create REST APIs to allow sending the

request and receiving the network data as a response. The network data was returned as a JSON object for request.

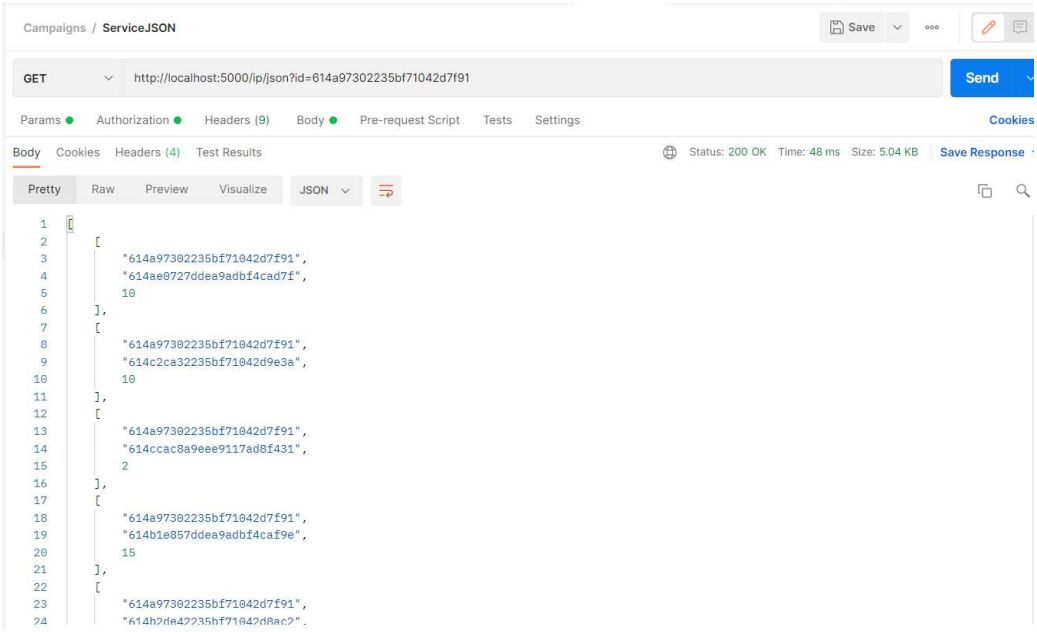

Figure 3.3 JSON response containing all the campaigns ID node and edge weights related to campaign with ID 614a97302235bf71042d7f91 as a result of REST API request

Other programs can also be plugged into RESTful APIs to enhance the functionality or to create more advanced interfaces and real-time dashboards. Here, I made a straightforward dashboard utilizing the REST APIs to display neighbour campaigns connected to a campaign given as the input. For example, Figure 3.4 shows the dashboard screen displaying all the campaigns related to campaign with ID "614a97302235bf71042d7f91" in the IP property association network. In this dashboard, Campaign IDs based on a central database are used to identify campaigns uniquely.

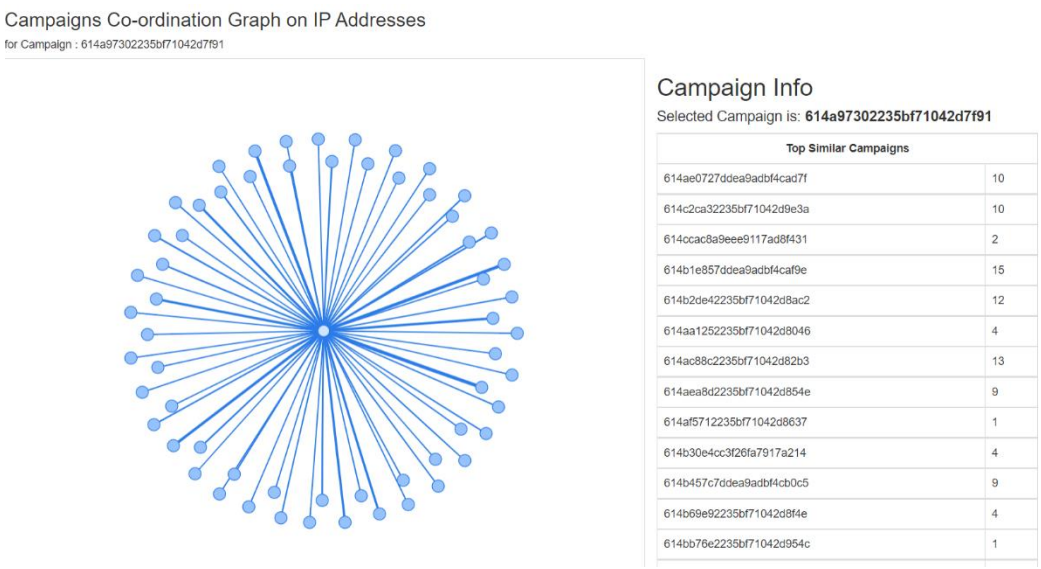

Figure 3.4 A dashboard working on top of REST API to display the connected campaigns and other relevant information

Chapter 4. Experimental Evaluation

4.1 Scalability and Efficiency

The spam campaign dataset obtained through one of ISOT Lab's industry partners was stored in an API-accessible secured NoSQL database. Our modelling used a subset of spam campaign data collected from 2016 to 2019.

Each record is collected via APIs. The pulled data is then fed into the bipartite network generation algorithm, and the association network is generated from its output. Both bipartite and association network generation algorithms process the data and generate the results in real-time. In addition, the algorithms work in such a way that distributed systems can also use them to accommodate a higher volume of data.

The bipartite network generation algorithm is written in python and executed in a single multi-core machine with four 2.7-GHz cores and 16 GB memory. It pulls the data from the central campaign database and processes it to create edge lists for the bipartite networks for each property. These bipartite graphs act as the input for association network generation algorithms. Processing time overhead were noted for all the whole process of graph generation and takes about 20 to 25 minutes depending on no email records in each campaign. The association network generation algorithm projects bipartite graphs into association graphs. The algorithm converts an unweighted bipartite network into a weighted association network. The association algorithm creates a weighted graph G = (V, E) consisting of a set of unique campaign nodes V and a set of weighted edges E for each property. Each of these generated association networks is stored in a separate table in the MySQL database. The association network graph is represented as an edge list. All four association graphs contain in total 97 million edges and 0.2 million nodes. The benefit of using an edge list over the other structures is that it is straightforward to generate and store the edge lists. The techniques like indexing and partitioning work well with edge list structure and help to improve data retrieval.

There are a few reasons to store the graph structures in relational databases. First, SQL is a popular query language, and it would be easy to develop more services on top of it. Moreover, the relational database provides ACID compliance [14] and flexibility for ad-hoc data access and migrations. Each graph is stored as a table in the relational database. The table consists of three columns, two for storing nodes connected by an edge and the third column stores the edge weight. One table was created to store all the campaign node IDs and extra information about campaigns. In this case, the further normalization of tables was avoided as it may increase the join operation overheads. Each table has over 30 million rows, and when the scale of the database is this large, the choice of data types to store the data and operations to be done on the table is crucial. 66 % of data in each edge-list table consists of campaign IDs and is represented as a string of 24 alphanumeric characters.

Thus, it is essential to store the IDs correctly and efficiently. These campaign IDs are stored as VARCHAR(24) for better storage efficiency and utilization of the CPU cache. VARCHAR uses only 1 or 2 extra bytes to record the length value. For example, in the latin1 charset, VARCHAR(24) will use up to 25 bytes of storage. An example of a SQL query to create a table to store the association network corresponding to IP address:

```
CREATE TABLE `IPservice` (
        `campaign_x` varchar(24) NOT NULL,
        `campaign_y` varchar(24) NOT NULL,
        `edge_weight` int NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Listing 3. Table creation query to store network data

The database is using the InnoDB engine, which helps in processing large data volumes and it is designed for efficient CPU usage and maximum performance. Two columns storing campaign IDs in the table are indexed. All the columns have NOT NULL specifications, meaning each record must contain values for all the columns.

```
CREATE INDEX node1idx ON campaigndb.IPService(campaign_node1);
CREATE INDEX node2idx ON campaigndb.IPService(campaign_node2);
```

Listing 4. Query to create B-tree indexes on table for faster read operations

The objective of fast lookup of table records was also considered when designing the database. To reduce the reading time of the records and quicker query execution, indexes are used in the tables. Choosing the right indexes for establishing better performance is gradual. Indexes are used to speed up the process of finding rows with specific values in the columns [15]. The database needs to iterate through all the rows in the entire table without using indexes to find particular records. Thus, to get better reading efficiency in the large tables, it is necessary to use the indexes. Here, indexes speed up queries with WHERE clause on campaign IDs to allow fast lookups on campaign_x and camapign_y columns. Indexes use a B-tree structure [15] underneath to store all the records. B-tree helps organize and keep data in such a way that reading specific records from the table is 100 times faster. In B-tree, nodes are arranged in sorted order. The worst-case performance is O(log n) for selecting a single record [15], better than the normal iterative lookup process. Although extra disk space and a bit of CPU overhead on each INSERT, UPDATE and DELETE query are sacrificed, indexes make other data reading-related queries much faster. Thus, it is a trade-off between time and space. Listing 4 shows a query to create indexes on two columns of the table that store the association network corresponding to IP address. Using the indexes on the column significantly reduces the read query execution time on the table from 58 seconds to 2.5 seconds. That further helped in improving REST APIs performance significantly.

While dealing with such a large volume of data in MySQL, the InnoDB is optimal as a storage engine, and it provides ACID compliance [14] and high-performance simultaneously. Here, to cut data writing costs, one of the essential settings in InnoDB's design [16] is the InnoDB buffer pool size. Its primary function is to cache the table data and indexes. The higher data and indexes this buffer can cache, the better, especially when dealing with enormous data sets. The default value of this is 8 MBs [16]. It was increased to 128 MBs to achieve good performance in caching the important data. Optimally in a dedicated database server, the value of innodb_buffer_pool_size is roughly 70% of available memory (RAM). Tuning up innoDB parameters help improve the ingesting time of processed data from the algorithm by 20%.

4.2 Case Studies

4.2.1 Case Study 1

This subsection describes a case study of investigating spam campaigns using the proposed framework. In this case study, I used the REST API platform to pull the network data for different properties and run some clustering methods to find cliques in the network. A clique is a subset of nodes in a network such that every two distinct nodes in the clique are adjacent.

This subsection describes a case study of investigating spam campaigns using the proposed framework. In this case study, I used the REST API platform to pull the network data for different properties and run some clustering methods to find cliques in the network. A clique is a subset of nodes in a network such that every two distinct nodes in the clique are adjacent.

All the graphs have high complexity and sometimes might be insufficient to efficiently produce understandable network visualizations. Here is a technique that will result in brief layouts and increased readability. It also enables the analysis of large networks and helps detect coordinated campaigns by revealing complex relationships and structures. To reduce the volume of edges and nodes some filtering techniques are used. The first step is filtering out the edges with low values. the intuition behind it is that the edges of the low value do not provide high confidence in coordination between campaigns; thus it is not adding any significant value to the resulting graph visualization. Another important aspect is to determine the threshold edge value. Here, the average edge value of the network is used as the threshold edge value. The next step is to use an appropriate layout for rendering the graph. The layout defines the precise position of its vertices on a 2D plane or in 3D space. Layouts use algorithms that calculate vertex positions based on the properties of the graph. Different layouts work for different visualization purposes. For example, the *Fruchterman-Reingold* force-directed layout (spring layout in NetworkX) tends to bring clusters of highly connected vertices nearer to each other, making the result of visualization more meaningful. If you are working on a small sub-portion of the graph, a circular layout or shell layout can help highlight local structures and convey centrality information of campaign nodes easily. Here, the spring layout is used in rendering, as it is easier to inspect various spam campaign

communities in the graph using the spring layout. Utilizing these visualization techniques, four similar cliques were found from four networks (each representing a different property).

These cliques consist of a few nodes representing campaigns connected to each other. For example, Figures 4.1, 4.2 and 4.3 show all the network node subsets and a suspicious clique found in IP, Domain, and Timeline networks.

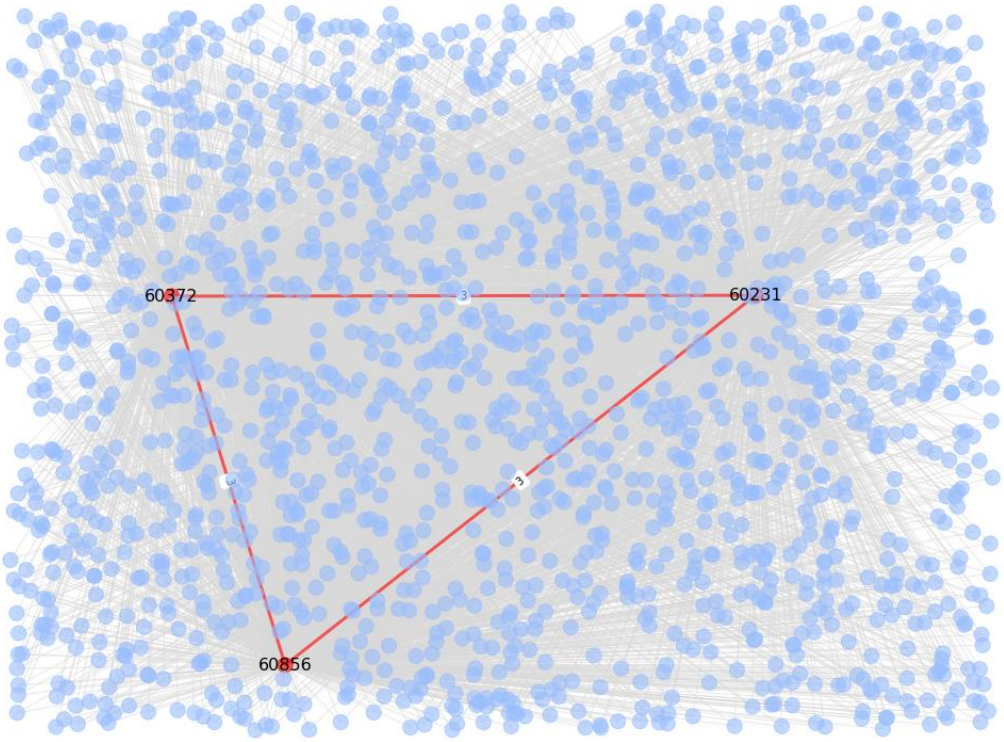

Figure 4.1 Association Network describing common IPs among campaigns. A node represents a spam campaign, and the weight of an edge between them is the number of unique IP addresses shared by both campaigns. Here the suspicious clique is highlighted by red colour. The campaign IDs and the number of shared IPs are labelled on respective nodes and edges. All three campaigns have 3 same IP Addresses with each other.

It is discovered that campaigns with key ID 60231, 60856 and 60372 are strongly connected to each other on every property. Corresponding to the IP network, we can see that, campaigns 60231 and 60856 share 3 of the same IPs. Campaign 60856 also shares 4 IPs with campaign 60372.

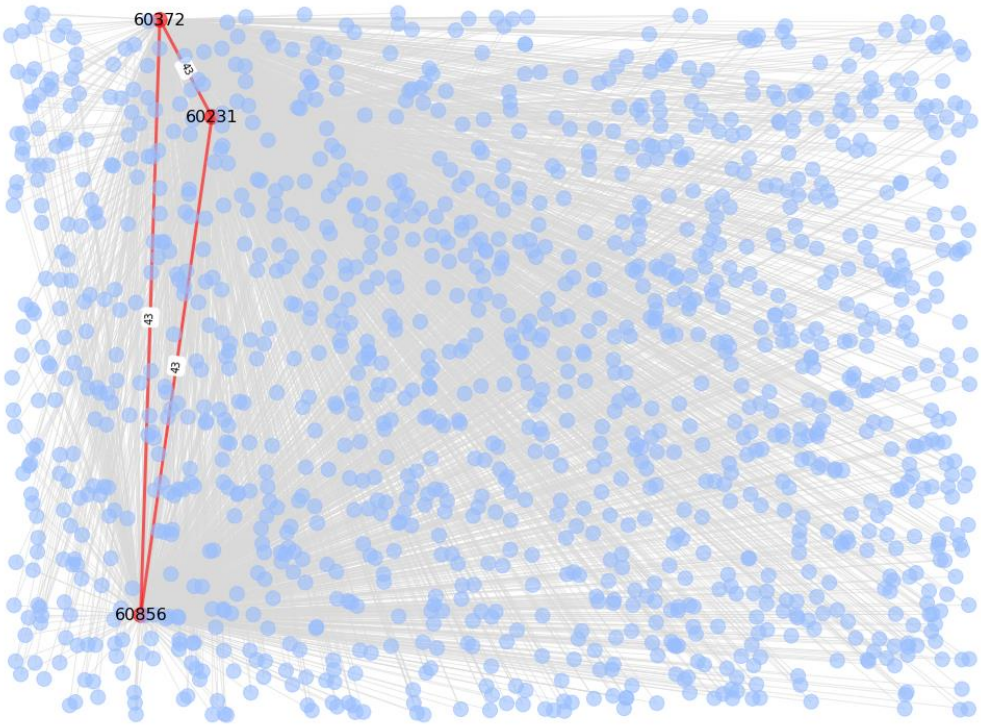

Figure 4.2 Association Network describing common domains among campaigns. A node represents a spam campaign, and the weight of an edge between them is the number of unique domains shared by both campaigns. Here the suspicious clique is highlighted by red colour. The campaign IDs and the number of shared domains are labelled on respective nodes and edges.

A similar pattern can also be seen in the other networks where campaigns 60231 and 60856 have 43 common domains in their WhoIs information. Moreover, they have 42 same source emails that send spam emails. All three campaigns also demonstrate similar temporal behaviour.

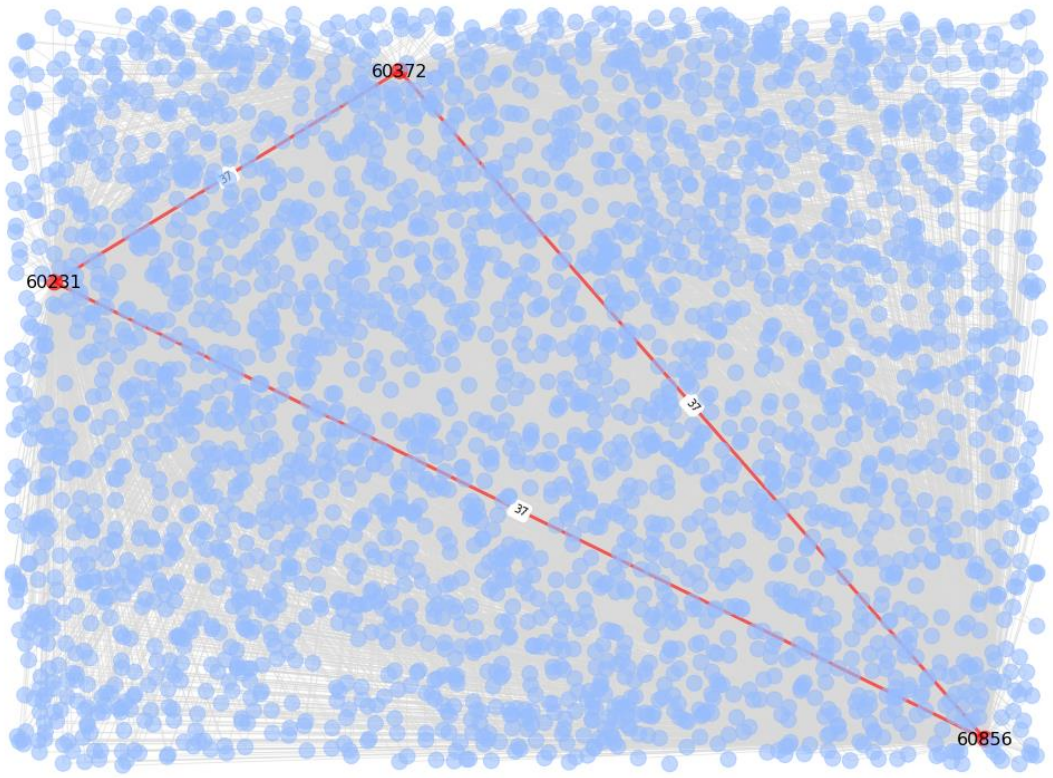

Figure 4.3 Association Network describing common timelines among campaigns. A node represents a spam campaign, and the weight of an edge between them is the number of unique timelines shared by both campaigns. Here the suspicious clique is highlighted by red colour. The campaign IDs and the number of shared timelines are labelled on respective nodes and edges. Here, the campaigns in the shown subset are densely connected to each other.

Figure 4.4 shows that all three campaigns were active in the first few months of 2018. Also, campaigns 60856 and 60372 are highly correlated in terms of operational and deactivated days. However, campaign 60372 was also active on the few days of November and December 2017. Investigating the source email property network further, it is discovered that both campaigns 60231 and 60856 do not share any common source email address with campaign 60372.

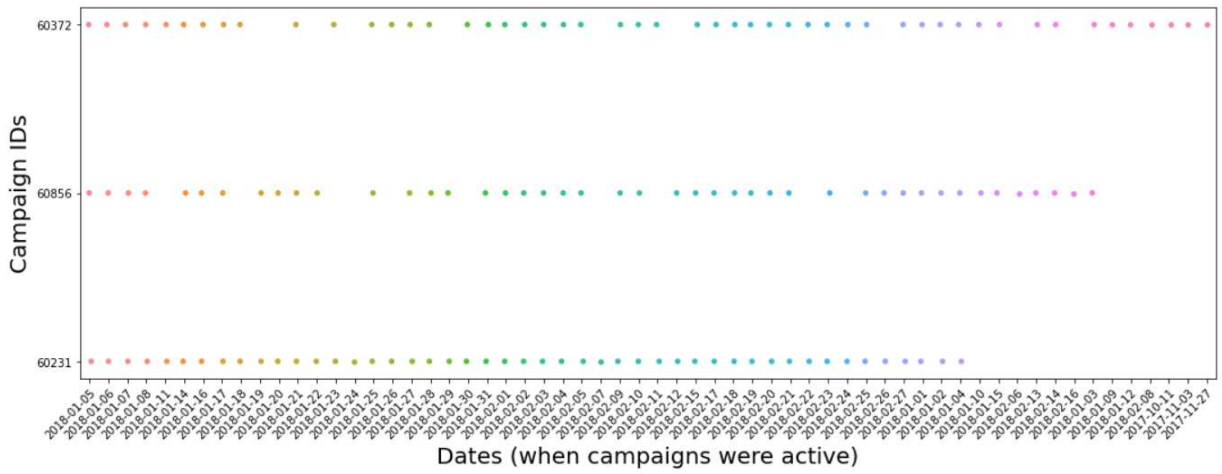

Figure 4.4 Comparing the active days of three campaigns. Here, coloured dots represent the days a campaign was actively sending the emails. Here, the correlation between campaigns 60372 and 60856 is high.

Based on this initial investigation utilizing different property networks, the connection among these campaigns becomes more evident. Campaigns 60231 and 60856 have the same Japanese word "インフォメーション" in the from-name section of the email header. In addition to this, all three campaigns have the same full name, "hisako utiyama" in their DNSinfo field. Another hit, many emails have common string "3a8d23fd99e50e7f1b0de4bac339b62d" in campaigns 60372 and 60856. It is suspected that it might be some cryptocurrency wallet address. We can conclude from all these insights that these campaigns are initiated in Japan, targeting the Japanese audience. Despite not having similar source emails in the campaign emails, all other evidence suggests a strong connection among these campaigns and makes it more strongly that the same entity runs these campaigns.

In this case study, there is no campaign of interest, and we are directly trying to find some similar campaigns. If you are given a campaign of interest and want to investigate and find other spam campaigns related to it, the following approach can be used.

The intuition behind this approach is that if the attacker is reusing the infrastructure for running different campaigns along with a campaign of interest, these campaigns will show up as the neighbors of the campaign of interest in all the graphs. And it is easier to spot these common neighbor campaigns with the visual inspection. Using the dashboard shown in figure 3.4, all the campaigns adjacent to the campaign of interest are rendered on the screen for all the graphs (i.e., IP, domain, source email, and time-coordination). Then, compare all the neighboring campaigns in all the graphs. Filter those neighboring campaigns which are common in all the graphs and have high edge value with the campaign of interest. Now, you have all the campaigns demonstrating high behavioral similarity with the campaign of interest. This way, it is easier to find similar campaigns potentially controlled by the same entity and using the same infrastructure.

4.2.2 Case Study 2

This case study demonstrates the use of the time coordination network to investigate the spam campaigns that involve synchronized actions or send spam emails within the same time frame. In the time coordination network proposed here, two campaigns are connected if they send spam emails within the same 24-hour period. The spam emails are then binned based on the time interval (24 hours) in which they are sent. These bins are used to construct the bipartite network of campaign IDs and time periods. The projected time coordination network is weighted, and edge weights are based on co-occurrence. For this case study, a subset of the network where campaigns have at least 7 common days of sending spam emails is shown in Figure 4.5.

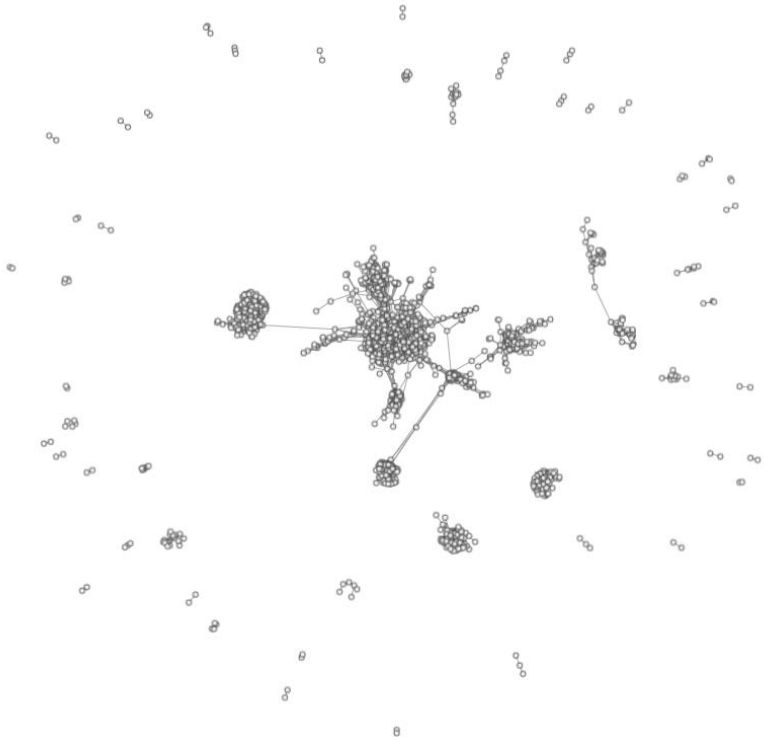

Figure 4.5 Visually the network is divided into two groups. First, the densely connected campaigns in the center and another one creates the outer boundary in the network and is made of numerous small groups of individually connected campaigns

As the network gets larger and more complex, it becomes challenging to analyze it as a whole. Therefore, the network is broken down into clusters and analyzed on a lower level. To break the time coordination network further, the Girvin-Newman algorithm is used. Girvan-Newman algorithm is a divisive clustering algorithm; at every iteration, an edge with the highest edge betweenness centrality is removed. Three communities are detected in the subset of the time coordination network as shown in Figure 4.6. The three communities were separated by colors blue, green, and red.

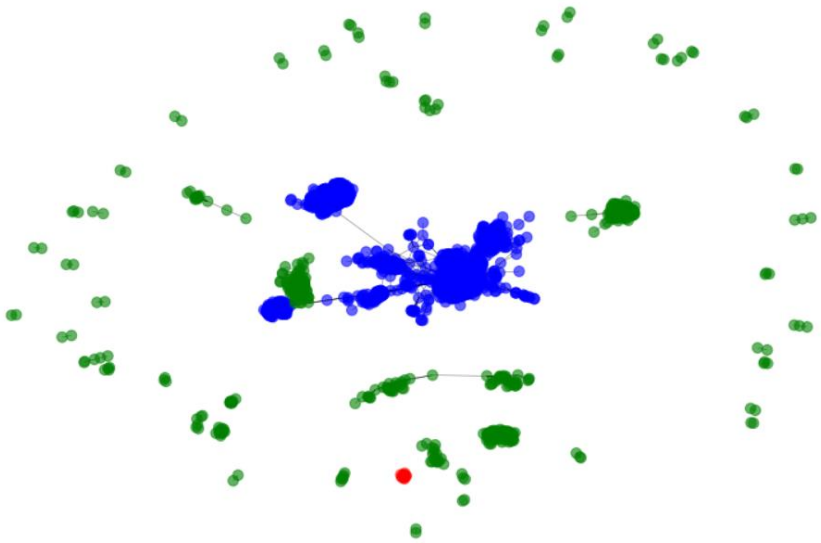

Figure 4.6 The Algorithm divided the time-coordination network into three clusters. First, the densely connected community in the center is coloured in blue; another one is the outer boundary of the in green colour; a small group of campaigns is also creating a community in red.

The Girvan–Newman algorithm detects communities by progressively removing edges from the original network, and the connected components of the remaining network are the communities. Instead of trying to construct a measure that tells us which edges are the most central to communities, the Girvan–Newman algorithm focuses on edges that are most likely "between" communities.

Inspection of dense blue cluster reveals many campaigns with high edge values. This means in the blue cluster, campaigns demonstrate highly similar temporal dynamics. A small cluster was found in the blue community where the number of common days when the campaigns were actively sending the emails was highest. Campaigns in this small cluster have at least 30 common days among them and it is shown in Figure 4.7.

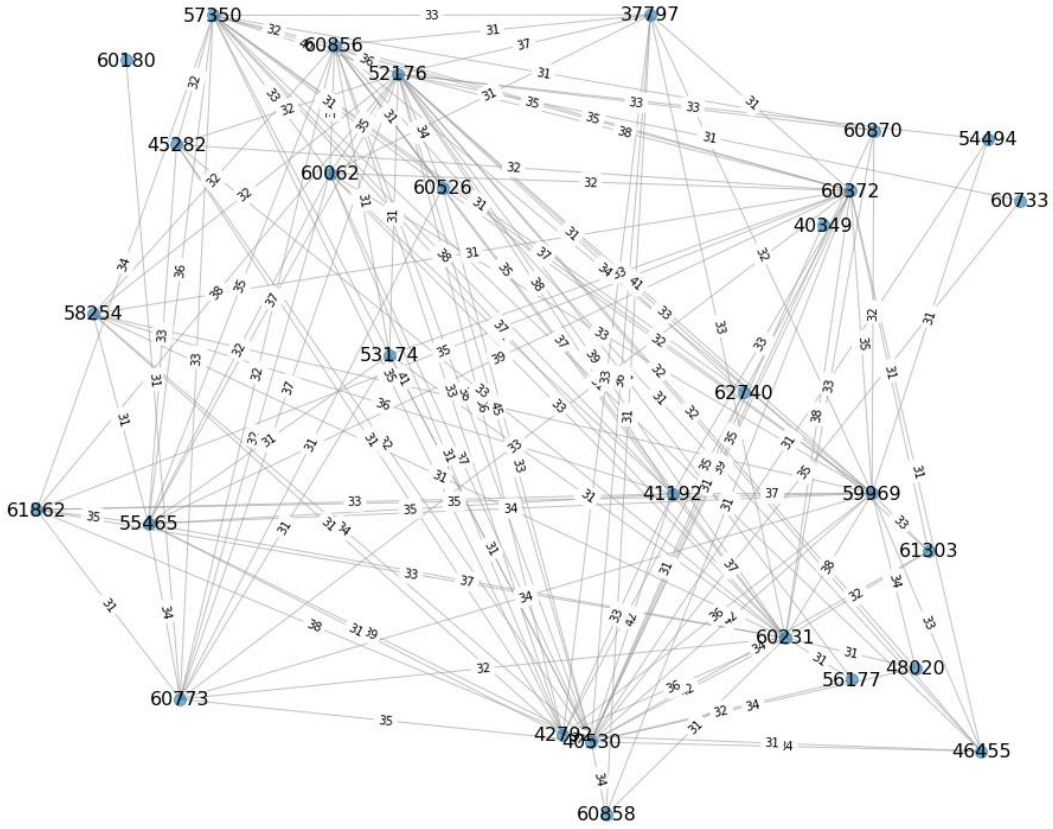

Figure 4.7 Visualization of a small portion of the time coordination association graph. Each node represents a campaign. The value of the edge between two nodes is wi,j defined in Algorithm 2. It shows the number of the same active days between two campaigns. The graph represents a densely connected cluster of campaigns that demonstrate high coordination of active days.

All the campaigns were actively sending emails from the last quarter of 2017 to starting of 2018. from Figure 4.8 it is seen that the spamming was less active in starting few months and gained momentum from September of 2017; many of the campaigns have similar patterns of gaining and losing momentum in sending spam emails.

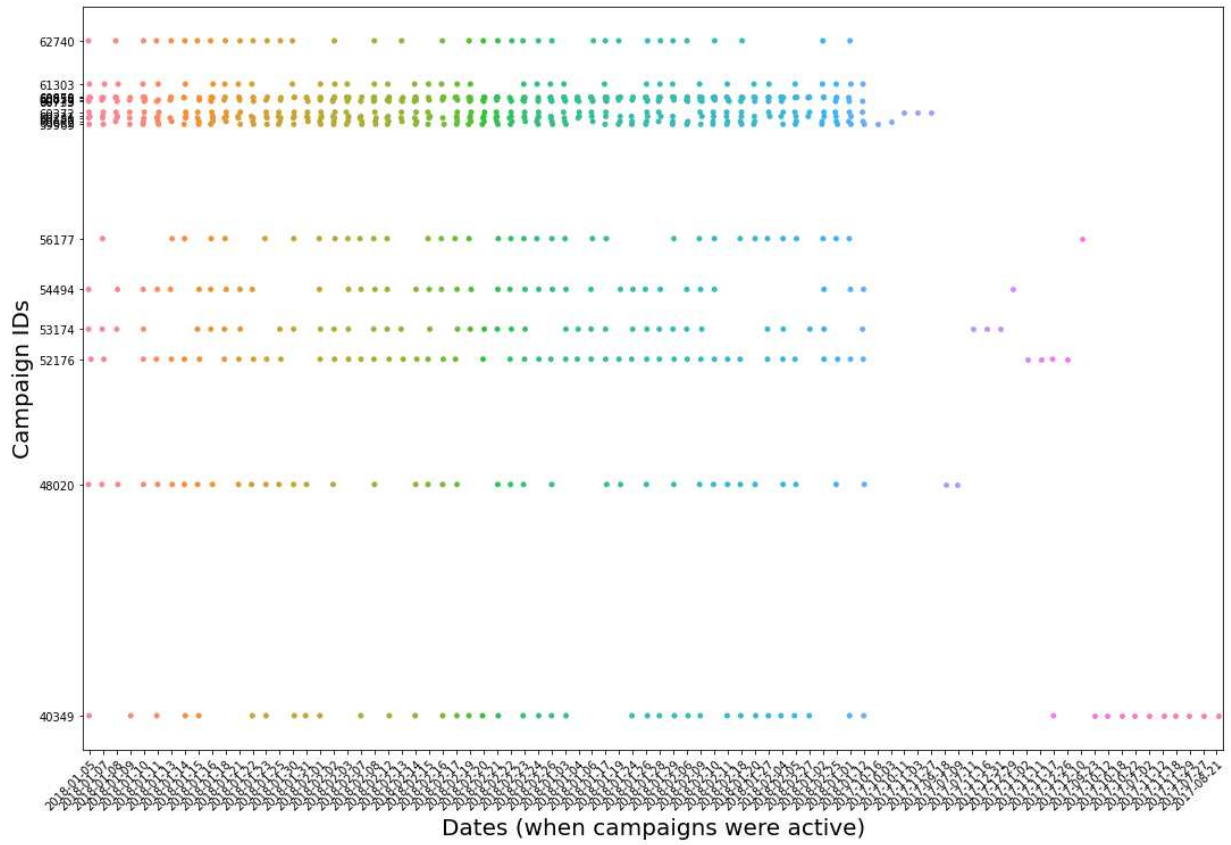

Figure 4.8 Comparing the active days of all the campaigns in the cluster. Here, the coloured dots represent the days campaigns were actively sending the emails.

All the campaigns in this small cluster demonstrate highly overlapping active days and along with that, also have common IPs, domains, with similar email headers. But another interesting insight found was campaign 59969 and 52176 have many similar kinds of emails; many emails from both campaigns have similar content, structure, and subjects. For example, emails in campaign 52176 have a string "※ご入金がありましたので至急ご確認下さい※" as the subject. Which means "There was a payment, so please check immediately". On the other hand, emails in campaign 59969 have "現在無料でご利用可能となっております" as the subject, that translates to "It is currently available for free". Although the emails have different subject lines in emails, these emails have the same IP address, language, similar content and same string "お知らせ" in their FromName field. Furthermore, many of these emails also have common domains.

In addition, while looking at the campaign's DNS records, it was found that all the registrants in the campaign's DNS records registered under the same email address. For instance, campaigns 59969 and 53174 have "Kazuko Konndou" as Administrative Contact Name with the email "nekohausuyamaneko@yahoo.co.jp" . The same email is registered with the name "Haruna Satou" in campaign 52176 and with the name "Sanae Ueda" in campaign 61303.

So, with this information, it can be concluded that the attacker with the email "nekohausuyamaneko@yahoo.co.jp" registered different domains under different names which were responsible for several campaigns across a period of 6 months in 2017-18 with banking and money related scams.

Chapter 5. Conclusion and Future work

This report presents a framework for using association network modelling for detecting coordinated spam campaigns controlled by a single entity. Four different association networks are used to represent links among spam campaigns on four different properties. I also presented one case study demonstrating the use of the proposed framework in real-world spam campaign investigation scenarios. Utilizing insights from all the property networks in decision-making leads to more accurate results. Effective data storing and retrieval strategies can efficiently handle computational considerations resulting from the large volume of data. The proposed network generation algorithm allows for numerous sources of information to be effectively combined and utilized.

Compared to the other existing approaches, the proposed framework has the following features:

1) The framework provides a solution to spam campaign linkage analysis and provides more generic solutions to detect links in other cyber security threats that involve the same controlling entity.

2) The proposed method generates different networks based on many properties, which help investigators make more accurate decisions and get more evidence against attackers.

The future work consists of a few tasks. Using more advanced similarity methods in constructing edge weights to get more information, a graph database to store association network data, and a real-time dashboard can also be plugged into the RESTful APIs for real-time observation and detection of spam campaigns.

In the proposed framework, weight values of the association network are computed via simple co-occurrences of the property. More advanced similarity methods like mutual information, the chi-square method or Jaccard similarity can be used to get more details about how campaigns share the properties rather than only giving the number of co-occurring properties. A filtering mechanism can also be included to filter out any low-value edges.

Although relational database used in the proposed method is better when operating on many records. Graph databases make network modelling and running complex network queries more efficient. Moreover, it leads to faster development of new features on top of the current system. In addition to that, a graph database executes queries that include graph traversal more efficiently than a relational database.

References

[1] T. Kulikova, "Spam and phishing in Q2 2021," 2021. [Online]. Available: https://securelist.com/spam-and-phishing-in-q2-2021/103548/. [Accessed November 2021].

[2] Wikipedia contributors. (2021, July 3). Graph theory. In *Wikipedia, The Free Encyclopedia*. Retrieved 07:01, July, 2021, from https://en.wikipedia.org/w/index.php?title=Graph_theory&oldid=1076773117.

[3] Wikipedia contributors. (2021, July 2). Graph (abstract data type). In *Wikipedia, The Free Encyclopedia*. Retrieved 07:04, July, 2021, from https://en.wikipedia.org/w/index.php?title=Graph_(abstract_data_type)&oldid=1074929950

[4] "RESTful APIs using flask-Python," [Online]. Available: https://flask-restful.readthedocs.io/. [Accessed 2021].

[5] P. A. Legg, "Visualizing the insider threat: challenges and tools for identifying malicious user activity," 2015 IEEE Symposium on Visualization for Cyber Security (VizSec), 2015, pp. 1-7, doi: 10.1109/VIZSEC.2015.7312772.

[6] K. Nance and R. Marty, "Identifying and Visualizing the Malicious Insider Threat Using Bipartite Graphs," 2011 44th Hawaii International Conference on System Sciences, 2011, pp. 1-9, doi: 10.1109/HICSS.2011.231.

[7] Zhuang, Li, John Dunagan, Daniel R. Simon, Helen J. Wang, Ivan Osipkov and J. Doug Tygar. "Characterizing Botnets from Email Spam Records." *LEET* (2008).

[8] W. Briguglio, *Campaign Detection Service,* 2021.

[9] "Bipartite Graph," [Online]. Available: https://en.wikipedia.org/wiki/Bipartite_graph. [Accessed July 2021].

[10] "Bipartite matching and vertex covers," [Online]. Available: http://www.princeton.edu/~aaa/Public/Teaching/ORF523/ORF523_Lec6.pdf. [Accessed August 2021].

[11] Neo4J, "Graph Data Modelling Guide," [Online]. Available: https://neo4j.com/developer/guide-data-modeling. [Accessed August 2021].

[12] L. Nanni, S. Brahnam, S. Ghidoni, E. Menegatti and T. Barrier, "Different Approaches for Extracting Information from the Co-Occurrence Matrix," 2013.

[13] Liu, Z., Barahona, M. Graph-based data clustering via multiscale community detection. Appl Netw Sci 5, 3 (2020). https://doi.org/10.1007/s41109-019-0248-7

[14] MySQL, "InnoDB and ACID Model," [Online]. Available: https://dev.mysql.com/doc/refman/5.6/en/mysql-acid.html. [Accessed December 2020].

[15] MySQL, "Optimization and indexes in MySQL," [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/optimization-indexes.html. [Accessed December 2020].

[16] MySQL, "Optimizing for InnoDB Tables," [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/optimizing-innodb.html. [Accessed December 2021].

[17] Wikipedia, "Representational State Transfer," [Online]. Available: https://en.wikipedia.org/wiki/Representational_state_transfer. [Accessed November 2021].

[18] flask-python, "Flask-RESTful User Guide," [Online]. Available: https://flask-restful.readthedocs.io/en/latest/. [Accessed November 2021].