Deep Mining Port Scans from Darknet

Sofiane Lagraa, Yutian Chen, Jérôme François

Deep Mining Port Scans from Darknet

Sofiane Lagraa*[1] | Yutian Chen[3] | Jérôme François[1,2]

[1]SnT, University of Luxembourg, Luxembourg L-1855, Luxembourg
[2]Inria Nancy-Grand Est, 615 rue du Jardin Botanique, 54600 Villers-les-Nancy, France
[3]TELECOM Nancy - University of Lorraine, 193 Avenue Paul Muller, 54602 Villers-lès-Nancy, France

Correspondence
*Corresponding author name, Email: sofiane.lagraa@uni.lu

Summary

TCP/UDP port scanning or sweeping is one of the most common technique used by attackers to discover accessible and potentially vulnerable hosts and applications. Although extracting and distinguishing different port scanning strategies is a challenging task, the identification of dependencies among probed ports is primordial for profiling attacker behaviors, with as a final goal to better mitigate them. In this paper, we propose an approach that allows to track port scanning behavior patterns among multiple probed ports and identify intrinsic properties of observed group of ports. Our method is fully automated based on graph modeling and data mining techniques including text mining.

It provides to security analysts and operators relevant information about services that are jointly targeted by attackers. This is helpful to assess the strategy of the attacker, such that understanding the types of applications or environment she targets. We applied our method to data collected through a large Internet telescope (or Darknet).

KEYWORDS:
Port scanning, semantic port similarity, graph theory, graph clustering, data mining, darknet

1 | INTRODUCTION

Application port scanning (also known as sweeping) is a technique widely adopted by the attackers to discover open ports (TCP or UDP) as a prelude to an exploit or an intrusion. There are three main types of scans: vertical, horizontal and block scans. A vertical scan is described as a single IP address being probed on multiple ports, while a horizontal scan consists in targeting the same unique port for a group or IP addresses, or even the full Internet IPv4 addresses. Block scan is a combination of both of them when attacker tests several ports on several IP addresses.

Table 1 represents the top-10 of probed services in each month from our dataset described in Section 7.1. Based on the IANA (Internet Assigned Numbers Authority) port list, which associates standard services with port numbers, we can observe web servers and databases are the most targeted services. However, these statistical results (1) do not provide any relationship between the probed ports and (2) do not quantify the correlation of services being co-targeted frequently, and (3) do not provide deep similarities between co-targeted ports. For example, assuming the following ports successively probed, 1360, 1433, and 1434, our goals is to check whether there is a dependency between them and what is the reason of this dependency, i.e. what is the underlying semantics among the different services behind those port numbers. In the current example, these three ports are used for SQL database management. By experience, we also know that ports 80 and 443 are similar ports because they are typically used for web traffic (respectively HTTP and HTTPS).

This paper targets multiple goals. The first one is to make progress in developing a unified approach to measure the similarity of ports from official source descriptions using text mining techniques. The second one is to leverage deep mining of darknet

08-2016		09-2016		10-2016		11-2016	
Port	**Number**	**Port**	**Number**	**Port**	**Number**	**Port**	**Number**
telnet	106,891,746	telnet	122,793,736	telnet	183,601,219	telnet	229,054,687
ms-sql-s	7,788,167	3d-nfsd	9,615,076	3d-nfsd	19,775,823	pcanywherestat	28,614,678
ssh	3,921,299	microsoft-ds	8,285,083	microsoft-ds	7,975,819	3d-nfsd	26,786,260
ms-wbt-server	2,241,857	ms-sql-s	7,885,440	ms-sql-s	7,096,882	ms-sql-s	9,221,479
http	2,123,664	ssh	3,148,455	ssh	3,589,657	microsoft-ds	7,766,815
http-alt	1,257,436	ms-wbt-server	2,376,006	ms-wbt-server	2,479,072	ssh	5,290,784
https	1,238,662	http	1,729,630	http	2,019,860	cwmp	4,380,958
mysql	1,199,221	http-alt	1,052,039	https	1,121,846	http	3,262,717
dtserver-port	847,182	https	1,035,851	mysql	1,058,366	http-alt	1,111,872
smtp	464,722	mysql	950,356	http-alt	873,097	https	1,090,220

TABLE 1 Top-10 of scan probes

data for discovering new strategies of port scans performed over time in vertical scans as well as in horizontal scans. Deep mining means that as we use data mining techniques in our analysis, we perform another analysis over the discovered information by mining techniques. We refer to deep mining as we perform a second level of analysis over the discovered knowledge by mining techniques.

The present work is an extension of our previous work [1] by defining a port similarity, automatic port tagging, and performing a deep mining over darknet data.

The following list summarizes our main contributions:

- **Generic graph-based model of port scans relationships**. We develop a graph-based model dependencies between port scans. The generic model can be instantiated to a vertical port and horizontal port scanning.
- **Measurement and knowledge discovery of port scanning attacks**. We provide a method to analyze a large graph model using unsupervised graph mining approach. It groups the commonly scanned ports to track the behaviors of attackers over a large IP address range. Especially graph clustering algorithms can extract groups of commonly scanned ports being well-connected in the graph, *i.e.* detecting dense partitions.
- **New semantic port similarities**. We propose new semantic port similarities based on the port document descriptions existing on the web. We crawl, filter, and measure the distances between each pair of port numbers. The distance between each couple of ports helps us to analyze the coherency of the discovered groups of ports.
- **Mining the discovered knowledge of port scanning**. The different groups of scanned ports are automatically enriched with tags extracted from the semantic port similarities calculation. For each enriched group of ports, we mine the co-occurrence of tags of ports by finding the common shared services, *i.e.* the coherency and correlation between ports inside a group.
- **Interactive visualization of groups of scanned ports**. We use `D3.js` library with enriched groups of scanned ports for the interpretation of the discovered knowledge and making scanned ports dependencies analysis useful for practitioners. It allows visualizing the graphs, their group structures, and their enriched data. Additional properties of the groups of scanned ports like groups size, density or coherency can be highlighted as well as external tags of each port.
- **Application of our methodology on real darknet data**. Collected scans from a /20 darknet network (i.e., 4096 addresses) for 2 years (Nov.2014 $\sim$ Nov2016).

The rest of the paper is organized as follows. Section 2 reviews and discusses related works to darknet, port scanning, and analysis techniques. Section 3 provides an overview of our proposed framework for port-scan mining. Section 4 and Section 5 propose an intrinsic description of the port similarity and a generic knowledge discovery method from port scans, respectively. Section 6 instantiates the latter on vertical and horizontal port scanning problem. We then present the experimental results in Section 7. Section 8 concludes and gives some future research directions.

2 | BACKGROUND AND RELATED WORK

Darknet. Recently, darknet has been receiving significant attention from the research community for observing Internet activities and cyber-attacks via passive monitoring. In the meantime, darknet technologies have been developed such as the deployment of sensors infrastructure, traffic analysis, and visualization. In[2], the authors provide a survey of such darknet technologies. The darknet analysis research works concern threat profiling, anomalies, and malicious activities. For instance, in[3], the authors evaluated UDP-based network protocols if they are vulnerable to amplification attacks and proposed a method based on darknet monitoring to detect them. In[4], the authors proposed a formal probabilistic model that aims at analyzing activities targeting Cyber-Physical System (CPS) protocols. In[5], the authors passively monitoring close to 16.5 million darknet IP addresses from a /8 and a /13 network telescopes.They proposed a probabilistic model-based big data approach for preprocessing traffic and data sanitization by fingerprinting misconfiguration traffic. In[6], the authors performed a darknet monitoring using topological data analysis techniques applied on network packets. They analyzed and visualized a large number of IP packets in order to discover malicious activities patterns such as scanning activities easily observable by security analysts.

Port similarity. Some representative works include measuring the similarity for clustering or anomaly detection of flow packets. In[7], the authors are interested in behavioral analysis and semantic similarity metrics for common data types found within network data hosts. They proposed a method for measuring and clustering the host behaviors using time series analysis. In their semantic similarity measure concerning especially the distance between ports, they defined a distance hierarchy for the port type. They decomposed the space of port numbers into groups based on the IANA port listings: well-known (0-1023), registered (1024-49151), and dynamic (49151-65535) ports. The authors considered well-known and registered ports to be closer to one another than to dynamic ports and ports within the same group are closer than those in different groups, and the same port has a distance of zero. In this paper, we develop another approach completely different for semantic similarity measure. In fact, we define a dynamic semantic similarity of ports over a set of documents describing ports. The proposed semantic similarity measure between ports allows us to profile the similar common ports scanned frequently.

Port scanning. In[8], the authors presented empirical-based measurement and analysis of a 12-day worldwide cyber scanning campaign targeting VoIP (SIP) servers using a darknet. They discovered that the coordinated scanning campaign targeted the entire IPv4 address space by using darknet data. Similarly, in[9], the authors also analyzed a darknet to explore scanning behaviors and uncovering large horizontal scan operations. Different analysis has been performed such as the source of scans (IP, country), services are being targeted. In[10], the authors presented an animated 3-D scatter plot visualization of port scanning on darknet data. In[1], we presented the discovery of vertical port scans from Darknet. They performed a deep analysis of port scan based attacks by proposing a graph model-based approach to analyze/understand them. The author discovered the common dependencies among targeted ports using graph analytic techniques for grouping the scanned ports. However, the scanned ports need to be augmented automatically with additional information highlighting the common targeted services, type of services or applications.

Network Service Dependencies.

Network service dependencies are usually described in terms of IP addresses and port numbers. Several works[11,12,13,14,15] focused on coarse-grained dependencies between network application, host, traffic, and service components in modern networked systems. Different approaches have been developed using fuzzy inference engine[11], matrix factorization[14], Bayesian decision theory[13], inference technique[15], dependency graph model[12] for applications and the underlying IT infrastructure, such as servers and databases, correlations on neighboring links, mining unstructured logs, and network service dependencies, respectively. Our work is close to [15]. However, in[15], the authors discovered normal dependencies that exist in benign applications whereas ours establishes relations in attacker behaviors when performing TCP scans. To achieve that, we provide behavior pattern of port-scan through graph mining techniques which provide intrinsic analysis of scanned ports augmented automatically with additional information where the classical statistical tools and existing works do not allow to discover and provide.

The originality of our approach compared to the related works is that first, it focuses on a specific problem, which is a port scanning by extending the paper[1]. We aim to provide a generic and automatic port scanning profiling approach over darknet data. Second, our approach includes the combination of port scanning dependencies using graph mining techniques and semantic similarities between the scanned ports using natural language processing techniques and data mining techniques. Third, we provide an interactive visualization for the group of scanned ports tagged with significant labels. This approach helps security analysts to have a global and complete infrastructure of information concerning port-scans, and strategies of scans in different periods.

3 | SYSTEM DESIGN

In this section, we first present the framework for computing port similarity measures (Section 4) and build the generic port scan graph model (Section 5), by introducing key components and the analysis knowledge. We discuss several practical considerations in semantic similarity measure between port numbers and the instantiation of a generic port scan graph model on two types of scans: vertical (Section 6.1) and horizontal (Section 6.2) port scanning.

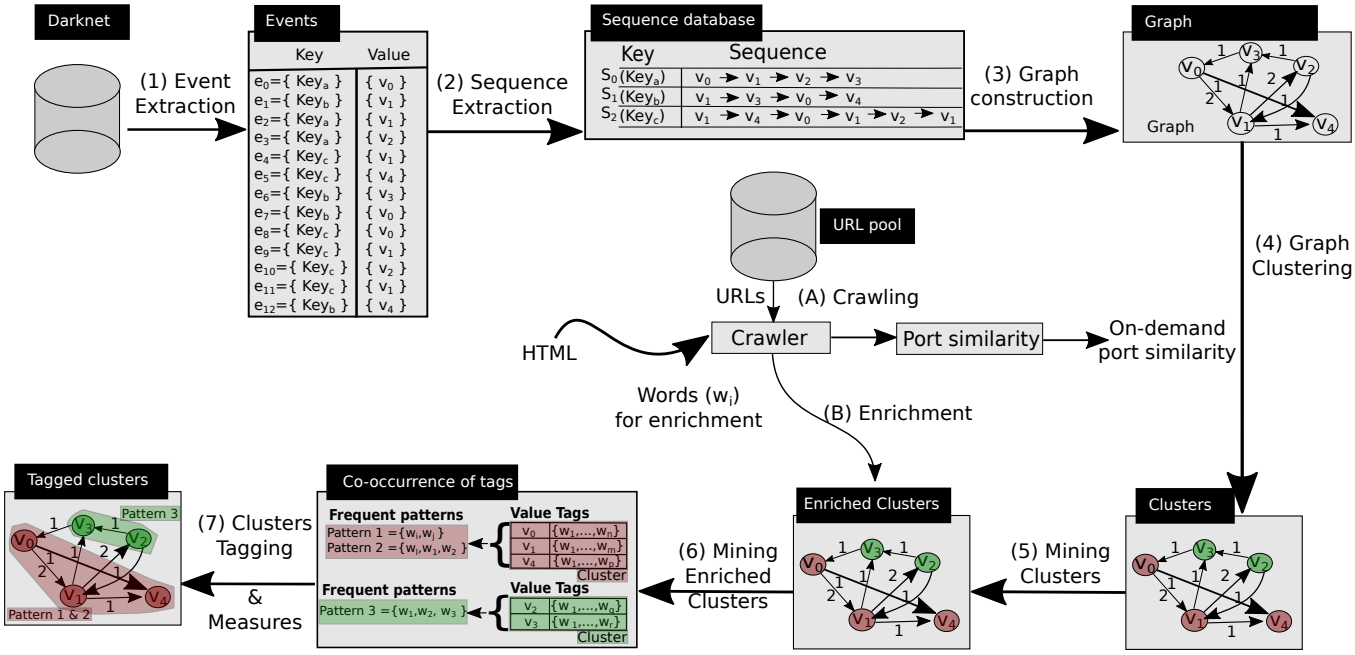


FIGURE 1 Port-scan mining: pipeline

Figure 1 highlights the components of our system, that actually constitutes a processing pipeline starting from network traffic data (from our Darknet):

1. **Event extraction**. This component extracts TCP packets from darknet source. Each individual TCP packet is transformed into an attribute-based *event* where each attribute is directly derived from regular traffic network features such as IP addresses or port numbers. The selected attributes are divided into two categories: keys and values. Every event is a set of key-value pairs. Both are composed of the event attributes. The key and values are fixed by the user depending on the problem. For example, if the user wants to analyze the access of IP addresses behaviors, then he/she can set the key to the source IP address and value to destination IP address.

2. **Sequence extraction**. Assuming a unique key, multiple events may thus share this same key with different associate values. So, for each key instance, a sequence of values can be created in this second step based on timestamp (originally retrieved from network traffic). In Figure 1, the sequences of keys key_a, key_b, and key_c are built from the values.

3. **Behavioral graph construction**. This component models all the sequences in a single graph, called a behavioral graph. This representation condenses information by merging vertices and labels over all the sequences, independently of the keys. Therefore, each unique key-based sequence is still represented by the graph, i.e. transitions observed in sequences are still observable in the a graph but new ones as well. We introduce this approximation to speed-up graph mining and also identify common patterns by weighting edges based on the number of occurrences of the represented transitions. As shown in Figure 1, the sequences have been reduced to four vertices that represent the values of sequences.

4. **Behavioral graph mining**. This component discovers the intrinsic properties of the constructed behavioral graph. Thanks to graph clustering, well-connected set of vertices are identified. For example, in vertical port scans, discovering the

frequently scanned ports with a high degree of dependency can be derived. In horizontal ports scans, discovering the commonly targeted subnetwork for scans can be inferred.

5. **Graph vertices enrichment**. Once the clusters of nodes are discovered, this component aims at enriching the clusters with meta-data. First, vertices representing port numbers are augmented with their textual descriptions using the component (8). Second, IP addresses are labeled with their country, city, domain, or organization. These steps allows respectively to enhance the expressiveness regarding vertical and horizontal scanning.

6. **Mining the enriched vertices**. In this component, a pattern mining algorithm is leveraged in order to discover the frequent co-occurrence of words within a single cluster. These words will thus constitute a representative *pattern* or *signature*.

7. **Graph clusters tagging**. The discovered patterns are used for tagging clusters. In vertical port-scans, the frequent patterns express the common shared services and technologies of probed ports. In horizontal port-scans, the frequent patterns express the common organizations and network characteristics of the targeted subnetwork. More details are given in Section 5.

8. **Crawling and processing documents**. This component is independent of the components described above. It allows crawling web pages describing services associated to the standard port numbers. Afterwards, we process the crawled documents, called *port-documents* by cleaning and keeping technical words. Those words are used as well as for port similarity measurement (9) and vertices ports enrichments (5). More detail of this component is given in Section 4.1.

9. **Port similarity measurement**. Port similarity is computed pair-wise between port descriptions from used documents. More detail of this component is given in Section 4.2.

4 | PORT SIMILARITY

A similarity measure between the port numbers is essential and can be used for clustering, classification, and anomaly detection in network traffic packets. In this paper, it facilitates investigations of the different scanned ports, and in the understanding of scan strategies. In this section, we present how to measure the similarity between ports.

4.1 | Application description crawling

In this section, we present an approach for crawling application port descriptions useful for port similarity calculation.

Port numbers and their related applications or protocols are described in different databases. For example, standard documents such as the RFCs produced by the IETF provide in-depth and technical information. However, there are also "de facto" standards, in particular from proprietary protocols, that are not documented in RFCs or any formal standards. Thus, we prefer to use the information collected in Wikipedia which has the following advantages:

- As community-based, the listed protocols and ports are not bounded to a specific standardization organization such as IETF. All applications can be thus described once there is a community large enough to agree on the content. Thanks to this scheme, it also benefits from very fresh updates.
- The database is well structured and especially a single page lists all known TCP and UDP ports[1] with the default protocol name and a hyperlink to the specific Wikipedia page if it exists.
- The description of the protocol is usually shorter than standards and will lead to a more efficient semantic analysis.
- Although a protocol can be defined though multiple documents, Wikipedia provides a summary within a unique page and so plays the role of the data-aggregation (again based on a community process)

[1] https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

Therefore, the list of port numbers is extracted from the aforementioned web page including for each listed port, the protocol name and eventually an hyperlink to an auxiliary page page with detailed content. This page constitutes the input in order to build our similarity between ports, which can only be computed for documented ports in Wikipedia. At the time of our experiments, this represents 4192 ports.

Formally, let $P = \{0, \dots, 65535\}$ a set of ports (TCP or UDP), each port $p_i \in P$ is represented by a 3-tuple, $(p_i, sname_{p_i}, doc_{p_i})$, where $p_i \in P$ is a port number, $Sname_{p_i}$ is a service name associated to that port number p_i, and $doc_{p_i} = \{doc_1, \dots, doc_n\}$ is a set of web document pages $doc_j \; \forall j \in \mathbb{N}$ crawled and associated to the latter. Thanks to this model, we are able to extract valuable information and words describing a port number, that is necessary for profiling scanned ports and so tagging clusters.

In order to obtain more accurate results, we filtered out *stopwords*. Stopwords are common and high-frequency words that generally do not contribute to the meaning of a sentence, used for information retrieval and natural language processing.

4.2 | Similarity

In this section, we introduce several measure definitions to evaluate the similarity between port number based on crawled documents describing their associated services. It will be used to find similarities between port numbers grouped into the same cluster.

We describe three classes of measures that can be used for identifying the similarity between ports: two syntactic-based similarities (Jaccard and Cosine similarity) and one semantic-based similarity (Doc2vec). We compare these similarity functions in order to have the best one in the clustering of port documents.

4.2.1 | Jaccard coefficient

Jaccard similarity coefficient (or index) is a classic measure used in information retrieval to infer the similarity or dissimilarity between sets of information. The Jaccard similarity measure between two sets is the ratio of sharing common items between the sets over all items of the merged two sets sets. The similarity in a set of port-related documents is computed as the intersection of their representative word sets.

Definition 1 (Jaccard similarity)**.** Given two ports p_i and p_j and their corresponding port documents with their word sets $doc_{p_i} = \{word_1, \dots, word_n\}$ and $doc_{p_j} = \{word'_1, \dots, word'_m\}$, respectively, Jaccard similarity between p_i and p_j is defined as:

$$Jaccard(p_i, p_j) = \frac{doc_{p_i} \cap doc_{p_j}}{doc_{p_i} \cup doc_{p_j}} \tag{1}$$

The Jaccard similarity measures the degree of the overlap between the two sets of words. It ranges from 0 to 1, with 0 for disjoint sets and 1 for identical sets.

Example 4.1. The Jaccard similarity between the port 80 and 443 (used for transferring web content) is 0.43 which is a higher value compared to the rest of the ports.

4.2.2 | Cosine

The cosine similarity is also very popular in the information retrieval and pattern recognition area[16]. The cosine similarity is computed based on vectors of attributes. These attributes are usually based on term frequencies. For term frequency computation, the Term Frequency-Inverse Document Frequency (*tf-idf*) scheme[17] is considered. *tf-idf* reflects the importance of a word in a document among collections of documents or corpus. The Term Frequency (*tf*) captures the number of occurrences of a given word in a document. The Inverse Document Frequency (*idf*) captures the rarity of the words that appear in a whole document collection.

Definition 2 (Term Frequency)**.** Term Frequency $tf(w, doc_{p_i})$ of the word w in *port document* doc_{p_i} is defined as the number of times w occurs in doc_{p_i} compared to the document length, *i.e.* number of words.

$$tf(w, doc_{p_i}) = \frac{f(w, doc_{p_i})}{\sum\limits_{w' \in doc_{p_i}} f(w', doc_{p_i})} \tag{2}$$

$f(w, doc_{p_i})$ is the number of occurrence of the word w in the document doc_{p_i}. Term frequency gives so a measure of the importance of the word w within a particular document.

Definition 3 (Inverse document frequency)**.** $DOC = \{doc_{p_0}, \dots, doc_{p_n}\}$ is the set of all documents associated to port descriptions.

$$idf(w, DOC) = log(\frac{|DOC|}{|doc_{p_i} \in DOC : w \in doc_{p_i}|}) \tag{3}$$

Where $|DOC|$ is the number of documents in the whole port documents collection. $|w \in doc_{p_i} : doc_{p_i} \in DOC|$ is the number of documents where the word w appears in ($tf(w, doc_{p_i}) \neq 0$). If the word is not in the port documents collection, this will lead to a division-by-zero. In this case, it is therefore common to use $1 + |w \in doc_{p_i} : doc_{p_i} \in DOC|$. Therefore IDF allows assess if a word frequently occurs in a document, and so if it is a discriminative word or very common.

Definition 4 (Term Frequency and Inverted Document Frequency)**.** Term Frequency and Inverted Document Frequency $tf - idf$ is the product of its *tf* weight and its *idf* weight.

$$tfidf(w, doc_{p_i}, DOC) = tf(w, doc_{p_i}) * idf(w, DOC) \tag{4}$$

A discriminative word of a document, i.e. a *keyword*, is actually a word that appears frequently in the considered document (high *tf*) but very rarely in others (high IDF), and so with a high TF-IDF value.

Definition 5 (Cosine similarity)**.** Given two ports p_i and p_j, their vectors of *tf-idf* values are $tf\text{-}idf_{p_i} = \{x_1, \dots, x_n\}$ and $tf\text{-}idf_{p_j} = \{y_1, \dots, y_n\}$, respectively. The *tf-idf* vectors must have the same fixed size, n by definition of the cosine similarity. We assume that whole vocabulary has a size of n words over the entire set of documents. For a particular document that contains m unique words, we thus consider $n - m$ zeros values in the *tf-idf* vector. The cosine similarity is then formally defined as:

$$Cosine(p_i, p_j) = \frac{\sum_{i=1}^{n} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \cdot \sqrt{\sum_{i=1}^{n} y_i^2}} \tag{5}$$

The resulting cosine similarity measure is always within the range of -1 and +1 meaning exactly the opposite and exactly the same, respectively. When the cosine similarity measure is 0, it indicates decorrelation between the two vectors.

It is worth to mention that Cosine similarity is for comparing two real-valued vectors, but Jaccard similarity is for comparing two sets.

4.2.3 | Doc2Vec

In [18], the authors propose *word2vec* in order to associate each word to a valued vector (embedding) based on the surrounding words. More specifically, they use the embedding layer of a neural network. At the end, words can be easily compared together by comparing their valued vectors. The embedding reduces the dimensionality of data while still preserving relationships among words. For instance, words having the same surrounding words will behave close values as well. Doc2Vec is very efficient to support similarity measurement between documents [19].

In [20], *Doc2Vec* extends *word2vec* for set of words (*e.g.* paragraphs, documents, etc.). Paragraphs are used in a similar manner as word2vec to predict the target word given context words. Doc2Vec differs from the classical existing methods such as Jaccard and Cosine. It incorporates an unsupervised learning algorithm to infer semantic relationships between words, sentences, paragraphs or entire documents.

5 | GENERIC PORT-SCAN GRAPH MODEL

Source and destination IP addresses and ports have been widely used in the intrusion detection domain [21,22]. They are good indicators of attack traffic flows and targeted services. In our context, we are interested in the scanning process to successive ports in vertical scanning ports and successive destination IP address in horizontal port scanning. Relying on a graph representation

will allow to condense information and thus to model global knowledge from multiple instances, *i.e.* multiple scans involving different IP addresses.

5.1 | Event extraction

The first step aims at preprocessing original packets records. It consists of extracting and keeping valuable header fields allowing to track IP addresses and their behaviors such as the used ports, and protocols. Thus, it eases the profiling of IP addresses and would be helpful to identify dependencies and correlations.

Let $PR = \{pr_0, \dots, pr_n\}$ be a set of packets records where each pr_i is described by a set of attributes including:

- The timestamp ts
- The source IP address sip
- The destination IP address dip
- The source port $sport$
- The destination port $dport$
- The protocol $proto$
- The type of service $typeOfService$
- The flags $flags$
- And some other technical data

Each event e_i corresponds a packet pr_i with its attributes. We develop key-value data model based on packet attributes. The key-value data model stores data as a collection of key/value pairs into a database. This is a simple method of storing data to support scalability[23]. Assuming $Attributes$ be the set of packet attributes, we define:

- $K \subset Attributes$ a set of attributes to be used as keys;
- $V \subset Attributes$ a set of attributes to be used as values;
- $K \cap V = \emptyset$.

Definition 6. For each pr_i, the event e_i is defined as $e_i = (ts_i, k_i, v_i)$ and consists of a the timestamp ts_i of the packet pr_i and a compound key k_i and value v_i composed respectively from the set of attributes K and V of the packet pr_i.

As an example, we consider the following IP packet pr_0 = `{1478708411.24; X.X.X.X; Y.Y.Y.Y ; ; ;18888;23;TCP;;;0x2}`

Its equivalent event representation with $K = sip, dip$ and $V = sport, dport$: K = `{X.X.X.X, Y.Y.Y.Y, TCP}` and V = `{18888, 23}`.

5.2 | Sequence construction

Let the set of events extracted from packets according to their keys and values. For each unique key, a sequence of event values is created based on the original timestamps. A sequence highlights the different steps of a potential scan by keeping the dependencies between each step. Therefore, a sequence representation allows characterizing the order of events values of a specific key.

Definition 7 (Sequence)**.** Let $E = \{e_0, \dots, e_n\}$ be the ordered list of events, a sequence S_i is built from a unique key k_i, $k_i \in \bigcup_{e_j \in E} k_j$ such that $S_i =< v_i^0, \dots, v_i^m >$ where the value $v_i^k \in E$ corresponds to the key k_i of the event e_k.

5.3 | Behavioral graph construction

In practice, the length of sequences can be very long, and can reach up to a million of events values by key. This makes the analysis more complex in space and time. In addition, in some cases, the event values can be redundant in a sequence. Therefore, we propose to group the set of sequences into a unique and global graph. A graph is a structure formed by a set of vertices (also called *nodes*) and the set of edges that are connections between pairs of vertices.

To characterize dependencies between successive events in all event sequences S_i for all keys k_i, we introduce the notion of the *event values graph* or *behavioral graph* model as an intuitive graph representation. It is a directed weighted graph that represents successive relationships between events. Specifically, each vertex represents an event value e_i and each edge (e_i, e_j) indicates that the event value e_j occurs after the event value e_i. An edge is weighted by the number of occurrences this particular transition happens in the whole set of sequences. Thus, a cycle in the graph indicates successive repetitive events.

Definition 8. Assuming all sequences S_i form a set S, a behavioral graph is an aggregated labeled weighted directed graph $G = (V_G, E_G, \alpha)$:

- V_G is the set of vertex of unique event value in the entire set of sequences, $\forall S_i \in S \forall e_j \in S_i \; V_G = \bigcup e_j$
- E_G is the edges of G. Let e_i and e_j be two event values in V_G, an edge exists, *i.e* $(e_i, e_j) \in E_G$, if and only if they appear successively in at least one sequence of S, *i.e.*
- α is a function that assigns the value l_{e_i,e_j} in each edge (e_i, e_j), the number of times the two events occurs consecutively.

The behavioral graph is easy to use and provide an easy interpretation of a set of long sequences of events based on their aggregation. Intuitively, a path from an event e_i to another event e_j in a behavioral graph indicates a potential "causal path", or dependency from e_i to e_j.

5.4 | Behavioral graph mining

Graph mining algorithms are analytic tools used to measure structural properties of the graph and, as a final objective in our case, to determine correlations and relationships between event values represented as a vertices in a graph. Among the graph mining approaches, we are interested in determining the groups of well-interconnected vertices in a behavioral graph. This process is called *graph clustering* and aims at finding dense parts, *i.e.* the number of edges between vertices of this part is higher than in the rest of the graph. There should be dense subgraphs with many edges within each cluster and relatively few between the clusters.

The ability to detect those clusters is helpful to discover ports or IP addresses which are commonly scanned together, and so may reveal a particular strategy of scanning.

In order to discover subgraph clusters, we use the *modularity clustering algorithm*. Modularity clustering aims to construct clusters in such a way the clusters with high modularity Q have dense connections between the vertices but sparse connections between vertices in different clusters. Modularity defines the density of the partition of a graph into subgraphs called *clusters* or *modules*. It measures the density of edges inside clusters compared to edges between clusters[24]. For computing modules, we use the algorithm developed in[25] for community structure discovery in large graphs.

Definition 9. Let M be the set of modules and e'_m be the number of edges inside the module $m \in M$. $|E_G|$ be the total number of edges in the graph G, and d_m is the total degree of the vertices in module m. The modularity of a partition of a graph is written as

$$Q = \sum_{m=1}^{M} [\frac{e'_m}{|E_G|} - (\frac{d_m}{2|E_G|})^2] \tag{6}$$

The modularity Q of the group is the difference between the number of edges within the group and the expected number of edges within the group in a random graph. The value of Q is between $-1 \leq Q \leq +1$. The more positive the value of Q the more significant the grouping. The entire graph (as one community) has $Q = 0$. Modularity algorithm iteratively selects and merges a pair of vertices to rise in modularity. It is a NP-complete problem and so requires high computing cost $O(|V_G|^2)$, where V_G is the number of vertices[26].

Once the clusters are discovered, the similarity measures are applied in each cluster for measuring the similarity between ports. It allows us to see the homogeneity or heterogeneity of services in a cluster.

5.5 | Mining the enriched vertices

Assumptions. We assume that the web pages are published by trusted sources and not distorted by malicious actors. In fact, malicious actors could inject fake web documents and publish them or inject them into the crawler. However, we expect that attacker represents a minority of users and so the introduced noise has a limited effect. We assume that the web pages are secure from exploits. We understand it is potentially not the case and could affect our crawler. However, the security of the crawler itself is out of scope of this paper.

Mining the enriched vertices. From the discovered clusters, we can extract the frequently targeted common services and machines. The discovered clusters support a fine-grained analysis. In vertical port-scans, the vertices representing the targeted ports are enriched with words describing them. In horizontal port-scans, the vertices represent the targeted IP addresses enriched with descriptions such as geolocation, organization, or domain. In each cluster, analyzing the meta-data of vertices by extracting the frequent co-occurrence of words is a combinatorial problem. For this, we describe the frequent pattern mining method that discovers the set of targeted services and accesses across the time. A critical problem is to find the frequent patterns describing the clusters well.

In this step, we use the meta-data related to a targeted problem and used keys and values in order to obtain a high level of granularity analysis. Meta-data enriches the graph vertices within clusters.

The data mining technique used to discover such sets of frequently occurring words is called *frequent itemset mining algorithm*[27],[28],[29 30].

Generally, in frequent itemset mining algorithms, the first input is a multiset of transactions $D = \{t_1, .., t_m\}$ defined over an alphabet of items $\Sigma = \{i_1, .., i_q\}$, where $\forall t_i \in D \quad t_i \subseteq \Sigma$, m is the number of transactions, and q is the number of items. The second input is a minimum threshold $\varepsilon \in [0, m]$. Frequent itemset mining algorithms then extract all the frequent itemsets called *patterns*, i.e. all the itemsets $X \subseteq \Sigma$ that appear in more than ε transactions of D. More formally, X must satisfy $support(X) \geq \varepsilon$, where $support(X) = |\{t_i \mid t_i \in D \wedge X \subseteq t_i\}|$.

In order to exploit this technique, we transform all the set of words that describe the ports in a cluster into a set of transactions D. Pattern mining algorithms are complex and combinatorial algorithms i.e the algorithms have an exponential time complexity according to the number of transactions, the number of items in a transaction, and the minimum threshold. Depending on the related problem, in Section 6, we provide examples of the use of frequent itemset mining algorithm. The discovered frequent patterns with their frequencies are used for labeling the graph clusters. They highlight the targeted clusters as well as services and accesses that should be investigated by the experts.

6 | PORT-SCAN GRAPH MODEL INSTANTIATION

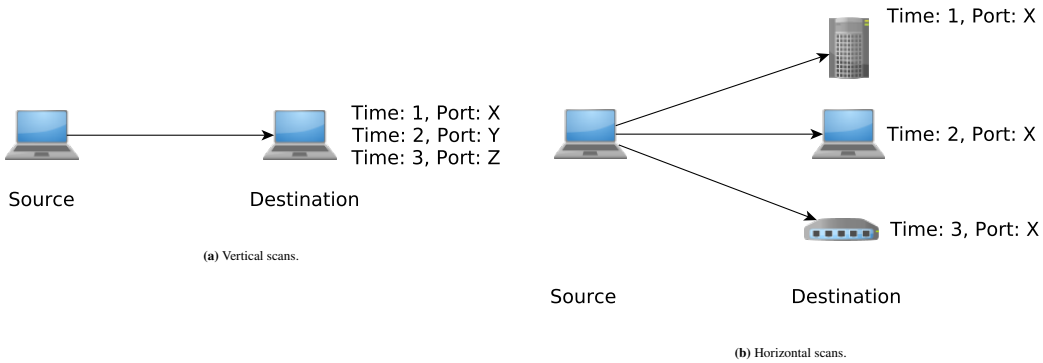
In this section, we propose two instantiations of our generic model developed in the previous section. The instantiations target two types of scans performed by the attackers: the vertical port-scan and the horizontal port-scan. Figure 2 shows a description of these two types of port-scans. Based on our proposed formalism, we instantiate our generic approach for these two types of scans. In a nutshell, a vertical scan is described as a single IP address being scanned for multiple ports. A horizontal scan is described as a scan against a group of IPs for a single port.

6.1 | Vertical port-scan graph model

TCP-SYN packets are transformed into graph describing TCP ports and their relationships from a scanning perspective. IP addresses are good indicators for profiling and monitoring traffics, attacks, and targeted services. In our case, we are interested in the representation and profiling the graph representation of the vertical scanning process to successive ports. A graph representation will allow to condense information and thus to model global knowledge from multiple scans involving different IP addresses.

6.1.1 | Event extraction

The event extraction consists of extracting a set of keys and values according to the need of the targeted problem and the dataset characteristics. For vertical scans, these attributes are the source and destination IP addresses, and source and destination ports. According to the generic model, we instantiate the key $k = \{sip, dip\}$ and the value $v = \{dport\}$.



(a) Vertical scans.

(b) Horizontal scans.

FIGURE 2 Vertical and Horizontal port-scans

6.1.2 | Sequence extraction

The use of the pairs of source and destination IP as the key naturally and accurately groups the targeted ports together. It keeps the port changes like in port scanning and provides the answer to the following questions: who is talking to who? Which are the group of ports commonly targeted? In another way, the time series of distinct source IP addresses per destination port is a better indicator than packet rate[31,1].

Definition 10. Let P be the set of all TCP ports. Assuming, a key $k_i = \{sip, dip\}$ with a source IP address sip and a destination IP address dip, we denote $S_i(k_i, T_s, T_e)$, the sequence of targeted ports by sip to dip between the starting time T_s and the ending time T_e, where $T_s < T_e$. S_i is thus a list of TCP destination ports of the key k_i ordered by time: $S =< (p_1, t_1), \dots, (p_n, t_n), \dots >$, where $p_j \in P$, $T_s \leq t_j \leq T_e$ such that $t_j < t_{j+1}$.

Figure 3 shows an example of two port sequences of two key-value pairs. In this case, the key is (sip, dip) and values is $dport$. Both port sequences have five targeted ports at different times. We note that given two IP addresses of a client and a server, two port sequences are constructed from the client to a server, i.e. client-server connection and from the server to the client, i.e. server-client connection.

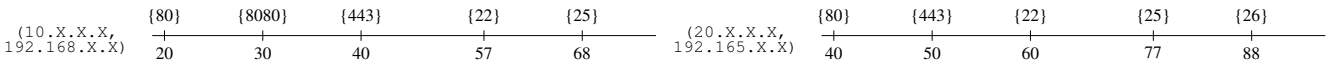


FIGURE 3 Port sequences

6.1.3 | Construction of a port-scan graph

To characterize causal relations between two successive TCP ports in a port sequence, we introduce the notion of the *vertical port-scan graph* model as an intuitive graph representation for successive scans in all port sequences.

A vertical port-scan graph is constructed from multiple successive scans. In another way, it is a weighted directed graph that represents successive relationships between targeted ports in port sequences of each key. Specifically, each vertex represents a port number p_i and each edge (p_i, p_j) indicates that a scan after to port p_j occurs after a scan to the port p_i, where $p_i \neq p_j$. $p_i \neq p_j$ means that we neglect successive probes to identical ports. The advantage of such an approach is to automatically discard DDoS attacks from the analysis which are the main type of mixed traffic with scans in darknet data. In addition, many scanning tools sends successive probes to the same ports for improving their accuracy.

The represented dependencies between port numbers represent the signature of all port sequences, i.e. the most and common behavior of IP source addresses targeting IP destination addresses. Figure 4 shows the vertical port-scan graph constructed from the two sequences in Figure 3

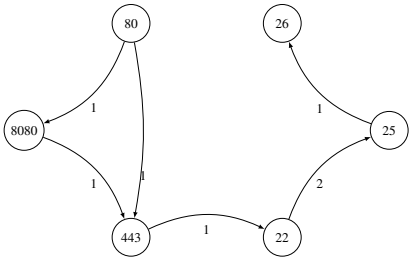


FIGURE 4 Vertical port-scan graph

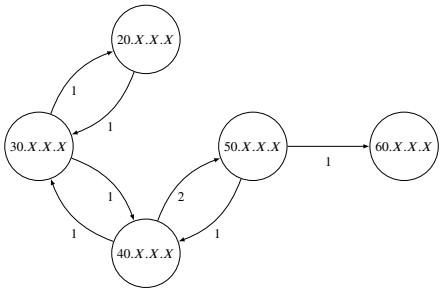


FIGURE 5 IP graph used in horizontal port-scanning

6.1.4 | Vertical port-scan graph analytic

The objective of the vertical port-scan graph analytic is to extract strongly connected targeted ports. The application of the modularity class algorithm described in Section 5.4 allows discovering the inherent clusters of commonly scanned ports. Detecting such clusters of vertices is significantly important for investigation, attack detection, and understanding new port-scans strategies in vertical port-scans. Applying graph clustering algorithm without further analysis is not sufficient for understanding the content of clusters and their characteristics.

In fact, the connected ports represent subgraphs that highlight the dependency of scans and the homogeneity or heterogeneity of clusters. A cluster is homogeneous if it contains a subgraph of scanned ports targeting specific services or vendors used for applications with apparent semantic relations. Otherwise, a cluster is heterogeneous. Determining the clusters of ports supposes discovering different underlying scanning strategies. However, profiling and discovering the content of clusters representing a semantic of ports of clusters is not an easy task and is a challenging problem. The semantic of ports describes if a cluster is homogeneous or not. It allows performing a deep understanding of clusters of ports behind their numbers by extracting the different scanning strategies.

6.1.5 | Automatic description of clusters using frequent pattern mining

For each discovered cluster, the goal is to discover potential correlations and common proprieties between port numbers. The automatic description of the cluster of ports with textual and semantic data means providing a tagging of clusters with frequent co-occurrence of words describing the ports.

In Section 4, we explored the words which tag each port. The challenge is to mine a set of port words in a cluster by discovering the common co-occurrence of ports words. These co-words can be used to describe a cluster. For this, the frequent pattern mining method is helpful to discover the set of common co-occurrence of words across ports in the cluster. The most important information to extract is the automatic description of clusters by the frequent co-occurrence of a set of words that describe the service associated to the port numbers of a cluster.

In our case, the alphabet Σ of transaction items should contain all the extracted words describing a port number, i.e. $\Sigma = \{word_1, .., word_q\}$. Given a vertical port-scan graph, their clusters $c_i \forall i \in [1, C]$, and the port $p_j \in c_i$ where C is the number of clusters. The set of transactions of ports $D_{c_i} = \{p_1, .., p_m\}$ is defined over words $\Sigma = \{word_1, .., word_q\}$ in the cluster c_i.

Example: Let a vertical port-scan graph $G_{Vertical}$, their clusters c_i, the ports in the cluster $c_1 = \{1433, 1434, 3306\}$, and a minimum threshold $\varepsilon = 50\%$. We assume the following port transactions with tags in Table 2. For this example, we limit the number of words describing a port to eight words. In this table, we have only a technical word related to computer science and no adverb or pronoun is kept. Table 3 shows the frequent co-occurrence of words describing the cluster c_1.

The discovered frequent co-words with their frequencies tag the cluster. Once we have the set of transactions by cluster, we can use a state of the art frequent itemset mining algorithm. We use LCM (Linear time Closed itemset Miner)[28], the most efficient one according to the FIMI (Frequent Itemset Mining Implementations) contest[32].

Table 4 shows closed frequent patterns of Table 2. We see that the number of patterns is reduced drastically without lost information. The closed frequent patterns are used to enrich the clusters with the frequent co-occurrence of tag occurring in their ports.

The automatic description of a cluster is given by the frequent patterns. The resulting frequent patterns are the co-occurrences of words that we are looking for. The discovered frequent tags with their frequencies, for each cluster, provide a high level of analysis and different services commonly targeted that should be investigated by the security expert.

Port	Tags
1433	$\{database, sql, relational, model, dbms, management, query, microsoft\}$
1434	$\{mssql, sql, database, management, microsoft, monitor, server, system\}$
3306	$\{mysql, oracle, workbench, sun, mariadb, innodb, database, percona\}$

TABLE 2 Port transactions with tags

Pattern	Frequency	Pattern	Frequency
$\{database\}$	(3/3)	$\{microsoft\}$	(2/3)
$\{sql\}$	(2/3)	$\{microsoft, management\}$	(2/3)
$\{sqldatabase\}$	(2/3)	$\{microsoft, management, sql\}$	(2/3)
$\{management\}$	(2/3)	$\{microsoft, management, sql, database\}$	(2/3)
$\{management, sql\}$	(2/3)	$\{microsoft, management, database\}$	(2/3)
$\{management, sql, database\}$	(2/3)	$\{microsoft, sql\}$	(2/3)
$\{management, database\}$	(2/3)	$\{microsoft, sql, database\}$	(2/3)
$\{microsoft, database\}$	(2/3)		

TABLE 3 Frequent patterns

Pattern	Frequency
$\{database\}$	(3/3)
$\{microsoft, management, sql, database\}$	(2/3)

TABLE 4 Closed frequent patterns

6.2 | Horizontal port-scan graph model

6.2.1 | Event extraction

For horizontal port-scan graph model, we need to track the carried out accesses *i.e.* the targeted hosts during the scans. Thus, we need to instantiate following keys and values: $k = \{sip, dport\}$ and the value $v = dip$. The keys are composed of source IP address and destination port, and the value is composed of the destination IP address.

6.2.2 | Sequence extraction

Figure 6 shows an example of two sequences from events enriched as described before. We construct only the event sequence of a pair of source IP address and destination port without the source port. For each pair, the targeted IP destination address is kept.

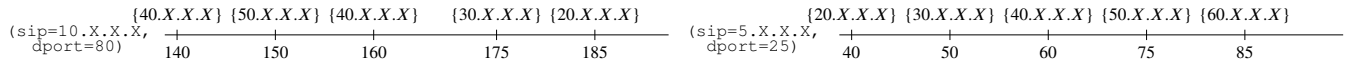


FIGURE 6 IP sequences

6.2.3 | Construction of a port-scan graph

To characterize causal relations and dependencies between two successive destination IP addresses in an IP sequence, similar to vertical port-scan graph model, we introduce the notion of the *horizontal port-scan graph* model as an intuitive graph representation for successive targeted accesses in all IP sequences.

A horizontal port-scan graph is constructed from multiple successive destination IP addresses. In another way, it is a weighted directed graph that represents successive relationships between targeted accesses in IP sequences of each key. Specifically, each vertex represents an IP address dip_i and each edge (dip_i, dip_j) indicates that a scan to IP dip_j occurs after a scan to the address dip_i. The advantage of such an approach is again to automatically discard DDoS attacks from the analysis. In fact, analyzing the targeted IP addresses allows us to investigate which (sub)networks or organizations are commonly targeted each other.

The dependencies between all targeted IP addresses represent the signature of all IP sequences, *i.e.* the most and common behavior of IP source addresses targeting IP destination addresses.

Figure 5 shows the horizontal port-scan graph constructed from two sequences in Figure 6. The vertices represent targeted IP addresses and the edges represent the successive scans performed by the whole keys with their weights.

6.2.4 | Horizontal port-scan graph analytic

The objective of horizontal port-scan graph analytic is to extract strongly concentrated target IP addresses. The application of graph clustering algorithm described in Section 5.4 allows discovering the inherent clusters of concentrated target IP addresses. Detecting such clusters of vertices is significantly important for investigation the strategies in horizontal port-scans. The discovered clusters of IP addresses are enriched with following properties: organization name, Internet Service Provider (ISP), domain name, county, city, and autonomous system organization. Other properties may be added such as approximate geolocation coordinates.

7 | EXPERIMENTAL RESULTS

In this section, we present experimental results of our proposed approach. We extract the knowledge from darknet data and perform evaluations of the feasibility and effectiveness of our proposed framework. In addition, we evaluate how much our semantic port similarity.

All experiments were conducted on a 2.80GHz Intel(R) Core (TM) i7-2640M 64 bits laptop with 8 GB main memory running on Ubuntu 16.04 LTS. All programs were implemented in Java.

7.1 | Dataset

Darknet data used in this paper is collected from a /20 darknet network or telescope (i.e., 4096 addresses) during 2 years (Nov.2014 $\sim$ Nov2016). A darknet consists of unused IP address space with no active host. So it collects silently incoming and unsolicited traffic. During this period 2,884,539,435 packets were captured representing around 500 GB of data with an increasing trend over the months as shown in Figure 7. The number of TCP packets represents more than 78% of packets.

7.2 | Results

7.2.1 | Vertical port-scans

To evaluate how relations among targeted ports evolved over time, our analysis is monthly-based. Hence, 24 unique graphs have been created from the two years of collected data. The built graphs contain up to 6284 vertices and 589117 edges and very low density as shown. Table 5 provides a summary of the vertical port-graph discovered in our experiments, showing for every graph discovered in each month, the number of vertices, the number of edges, the density, and the diameter of the graph. The port-scan graph density is defined as follows:

$$Density = \frac{\#Edges}{\#Vertices(\#Vertices - 1)} \tag{7}$$

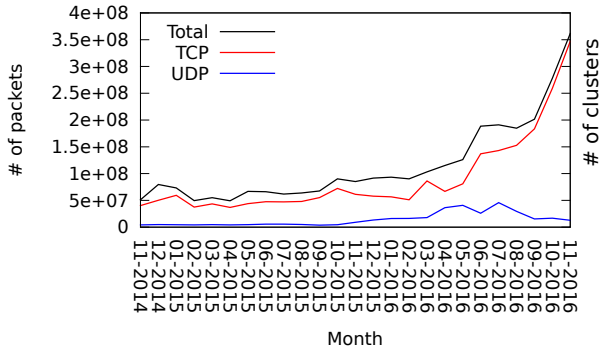


FIGURE 7 Number of packets according to the months

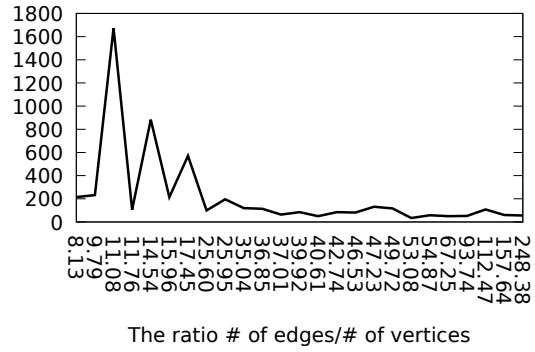


FIGURE 8 Number of clusters according to the ratio of the number of edges / the number of vertices of each vertical port-scan graph.

The density is based on the number of edges and vertices. The maximal density is 1 for complete graphs (fully connected graph) and the minimum density is 0 for a very sparse graph. The port-scan graph diameter is the maximum length of the longest path between any two graph port vertices. It allows showing the eccentricity of any port vertex in the port-scan graph.

Month-Year	#Vertices	#Edges	Density	Diameter	#Clusters
11-2014	1118	13151	0.0105	52.0	106
12-2014	1899	18607	0.0051	59.0	232
01-2015	1614	25772	0.0098	20.0	214
02-2015	1003	25680	0.0255	19.0	99
03-2015	811	34668	0.0527	15.0	85
04-2015	1169	290359	0.2126	15.0	56
05-2015	2465	277244	0.0456	304.0	108
06-2015	2294	40034	0.0076	110.0	571
07-2015	779	42750	0.0705	13.0	58
08-2015	2558	37211	0.0056	11.0	883
09-2015	1730	44897	0.0150	29.0	196
10-2015	6145	49966	0.0013	197.0	215
11-2015	898	44651	0.0554	30.0	117
12-2015	2659	97993	0.0138	10.0	114
01-2016	2051	71872	0.0170	25.0	120
02-2016	6816	75575	0.0016	20.0	1673
03-2016	1630	76989	0.0289	28.0	131
04-2016	1524	80899	0.0348	17.0	34
05-2016	1364	54457	0.0292	21.0	85
06-2016	1707	63185	0.0216	22.0	64
07-2016	1413	57393	0.0287	33.0	50
08-2016	1246	196421	0.1266	42.0	60
09-2016	1781	119779	0.0377	12.0	50
10-2016	6284	589117	0.0149	16.0	52
11-2016	2114	98374	0.0220	8.0	81

TABLE 5 A summary of the extracted vertical port-scan graphs from corresponding dataset of packets in Figure 7 used in our experiments. The table shows for every graph, the number of vertices (ports), the number of edges, the density, the diameter, and the number of founded clusters.

Figure 8 shows the discovered number of clusters according to the ratio of the number of edges vs. the number of vertices of each vertical port-scan graph. We see a correlation between the number of clusters and the vertical port-scan graph characteristics. When the number of the ratio increases then the number of clusters decreases. This means that the clusters are more compact describing common properties such as the co-targeted services via their discovered labels, and the number of scans between two successive ports.

Using pattern mining algorithm for discovering dependencies between ports within clusters, we discover different knowledge shown in Table 6 and Table 7.

Table 6 highlights a profile of five clusters among 81 from the graph 11-2016. We profile clusters based on their total number of vertices, the number of unassigned port vertices and the average of pairwise port similarities using Jaccard, Cosine, and Doc2Vec. When the distance between ports converges to 0, it means the ports are similar, otherwise, the ports are dissimilar. Table 7 shows frequent co-occurrence of words describing each cluster found by our automatic description of clusters. From Table 6 and Table 7, we discover the part of automatic following results:

- High correlation in clusters mainly contain vendors use or user ports or for applications (1024-49151) but few of system ports (0-1023) and dynamic and/or private ports (49152-65535).
- High correlation in clusters containing unassigned port services.
- Hybrid clusters with signed and unassigned port services.
- Cluster 1 contains the frequent patterns describing a set of probed ports related to a streaming server.
- Cluster 2 contains the frequent patterns describing a set of probed ports related to computer networking protocol such as *xns* and from *Xerox* and another pattern related to routers and router board such as *rb493g*.
- Cluster 3 contains the frequent patterns related to gaming software, remote control software for Microsoft Windows (*Vista*), and browser such as *Firefox* or *XeroBank*.
- Cluster 4 contains the frequent patterns related to gaming software such as *Quake*, remote communication protocols with server such as *Telnet* and *ssh*.
- Cluster 5 contains all vertices with unassigned port service which means the sources of the targeted ports perform the same scans on the specific unassigned port, hence, it does not exist semantic port similarity measures.

Figure 9 shows a sample of clusters. The interactive visualization is created using `D3.js` library allowing to highlight a group, a port, or zooming. The thick and darkness edges represent high interactions between ports and grouped in the same clusters having the same color. In Figure 9b, we can observe the specific and correlated services within a cluster. For example, the cluster in orange represents communication services. The blue cluster represents database services. However, the black cluster is representative of a iterative scan from port 1973 to port 1979.

Cluster ID	#Vertices	#Unassigned	Jaccard	Cosine	Doc2Vec
1	4065	3165	0.1283	0.0825	0.3523
2	30	10	0.0285	0.0048	0.3755
3	4	3	0.0847	0.0232	0.6336
4	8	3	0.0322	0.0095	0.5066
5	12	12	Not exist		

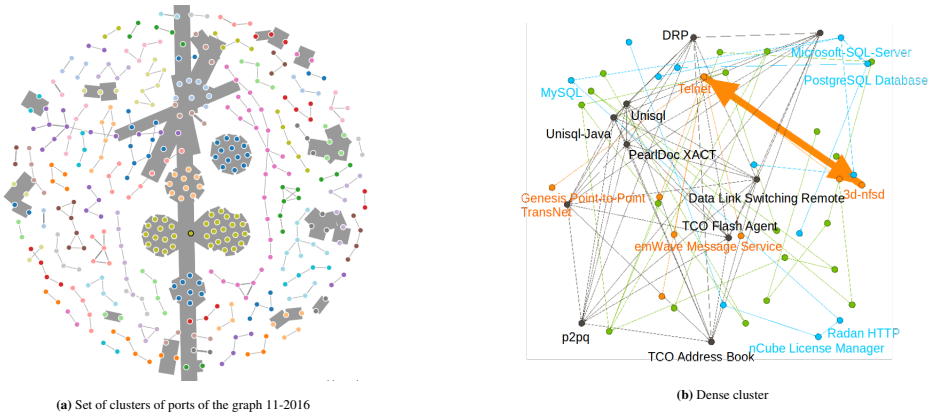
TABLE 6 Profiling clusters using semantic port similarities corresponding to the graph 11-2016.

7.2.2 | Horizontal port-scan

Table 8 provides a summary of the horizontal port-graph discovered in our experiments, showing for every graph discovered in each month, the number of vertices (IP addresses), the number of edges, the density, and the diameter of the graph. The number of vertices reaches up to the maximum size ranges /20 darknet network (i.e., 4096 addresses). Hence all addresses are targeted during scans. Hence, all IP addresses reach each other, and there is a successive access between all IP addresses as shown by the diameter of all graphs equal to 2. Thus, the scanner targets all subnetworks regardless of the IP addresses in this subnetwork. Due to the high density of graphs, the number of communities is lower than vertical port-scan graphs.

Cluster ID	Pattern (Frequency)	Pattern (Frequency)	Pattern (Frequency)
1	{*server*} (11.88%)	{*server*, *streaming*} (11.77%)	{*server*, *streaming*, *quicktime qtss*, *stub*} (11.55%)
2	{*wireless*, *mikrotik*, *routerboard*, *routeros*, *rb*493*g*} (5%)	{*authentication*, *xns*, *importance xerox*, *systems*} (5%)	{*security*, *cisco*} (5%)
3	{*remote*, *vista radminf amatech*} (100%)	{*games*, *gamasutra*, *mongodb*} (100%)	{*browser*, *xerobank f irefox*, *torpark*, *onion*} (100%)
4	{*rfc*, *telnet*, *ssh terminal*, *standards*} (20%)	{*dynamic*, *allocated hosts*, *requests*} (20%)	{*quake*, *doom*} (20%)
5	No pattern		

TABLE 7 Extended profiling clusters of Table 6 with frequent co-occurrence of words describing them.



(a) Set of clusters of ports of the graph 11-2016

(b) Dense cluster

FIGURE 9 Clusters of scanned ports.

The discovered knowledge clearly indicates, first, that our method is able to find the intrinsic correlations based on the semantics of the ports and, second, that attackers may leverage intelligent scanning to attack a dedicated type of systems. In vertical scans, the attacks are performed on specific ports sharing common properties and services. In horizontal scans, the attacks target subnetworks of an organization having a specific port number as well as all addresses space.

8 | CONCLUSION AND FUTURE WORK

In this paper, we have proposed a new approach based on graph analytic and data mining techniques for grouping the scanned ports by proposing a generic graph-based model of port scans relationships, discovering the knowledge of port scanning attacks using text mining techniques, and providing new similarity measure for port numbers. Using our approach, the method helps the security operators to understand what clusters of services are commonly being targeted in vertical and horizontal scans. We also highlight the similarity of the targeted services. Our experimental results, over real data collected in a darknet, highlight the ability of our method to discover unknown specific co-targeted ports belonging to the same or different types of services. Our future plan consists in predicting the future scanned ports.

ACKNOWLEDGMENTS

This work was partially funded by HuMa, a project funded by Bpifrance and Region Grand Est under the FUI 19 framework, and by the NATO Science for Peace and Security Programme under grant G5319 Threat Predict: From Global Social and Technical Big Data to Cyber Threat Forecast. It is also supported by the High Security Lab hosted at Inria Nancy Grand Est.

Month-Year	#Vertices	#Edges	Density	Diameter	#Clusters
11-2014	4096	8386560	0.500	2	17
12-2014	4096	8537518	0.509	2	15
01-2015	4096	8552439	0.509	2	15
02-2015	4096	9022087	0.537	2	13
03-2015	4096	9542053	0.568	2	12
04-2015	4096	9709785	0.578	2	12
05-2015	4096	9894012	0.589	2	11
06-2015	4096	10591460	0.631	2	11
07-2015	4096	10647918	0.634	2	11
08-2015	4096	10794145	0.643	2	10
09-2015	4096	11810241	0.704	2	10
10-2015	4096	11274119	0.672	2	9
11-2015	4096	10989759	0.655	2	10
12-2015	4096	12788926	0.762	2	10
01-2016	4096	9541004	0.568	2	11
02-2016	4096	10947117	0.652	2	9
03-2016	4096	11608717	0.692	2	8
04-2016	4096	12800785	0.763	2	9
05-2016	4096	2524037	0.150	2	13
06-2016	4096	14800785	0.882	3	16
07-2016	4096	16484032	0.982	2	16
08-2016	4096	6445034	0.384	2	9
09-2016	4096	16658806	0.993	3	4
10-2016	4096	16764346	0.999	2	5
11-2016	4096	16770106	0.999	3	7

TABLE 8 A summary of the extracted horizontal port-scan graphs from corresponding dataset of packets in Figure 7 used in our experiments. The table shows for every graph, the number of vertices (IP addresses), the number of edges, the density, the diameter, and the number of founded clusters.

References

1. Lagraa Sofiane, François Jérôme. Knowledge discovery of port scans from darknet. *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Lisbon, Portugal, May 8-12, 2017.* 2017;:935–940.

2. Fachkha C., Debbabi M.. Darknet as a Source of Cyber Intelligence: Survey, Taxonomy, and Characterization. *IEEE Communications Surveys Tutorials.* 2016;:1197-1227.

3. Rossow Christian. Amplification Hell: Revisiting Network Protocols for DDoS Abuse. In: ; 2014.

4. Fachkha Claude, Bou-Harb Elias, Keliris Anastasis, Memon Nasir, Ahamad Mustaque. Internet-scale Probing of CPS: Inference, Characterization and Orchestration Analysis. In: ; 2017.

5. Bou-Harb E., Husak M., Debbabi M., Assi C.. Big Data Sanitization and Cyber Situational Awareness: A Network Telescope Perspective. *IEEE Transactions on Big Data.* 2017;PP(99).

6. Coudriau Marc, Lahmadi Abdelkader, Francois Jerome. Topological Analysis and Visualisation of Network Monitoring Data: Darknet case study. *8th IEEE International Workshop on Information Forensics and Security - WIFS 2016.* 2016;.

7. Coull Scott E., Monrose Fabian, Bailey Michael. On Measuring the Similarity of Network Hosts: Pitfalls, New Metrics, and Empirical Analyses. In: ; 2011.

8. Dainotti Alberto, King Alistair, Claffy kc, Papale Ferdinando, Pescapè Antonio. Analysis of a "/0" Stealth Scan from a Botnet. *Proceedings of the 2012 ACM Conference on Internet Measurement Conference.* 2012;:1–14.

9. Durumeric Zakir, Bailey Michael, Halderman J. Alex. An Internet-wide View of Internet-wide Scanning. *Proceedings of the 23rd USENIX Conference on Security Symposium.* 2014;:65–78.

10. Riel Jean-Pierre, Irwin Barry. InetVis, a Visual Tool for Network Telescope Traffic Analysis. *Proceedings of the 4th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa.* 2006;:85–89.

11. Dechouniotis Dimitrios, Dimitropoulos Xenofontas, Kind Andreas, Denazis Spyros. Dependency Detection Using a Fuzzy Engine. In: DSOM'07:110–121; 2007.

12. Wang S., Capretz M. A. M.. A Dependency Impact Analysis Model for Web Services Evolution. In: :359-365; 2009.

13. Lou Jian-Guang, Fu Qiang, Wang Yi, Li Jiang. Mining Dependency in Distributed Systems Through Unstructured Logs Analysis. *SIGOPS Oper. Syst. Rev.*. 2010;44:91–96.

14. Ding Min, Singh Vishal, Zhang Yueping, Jiang Guofei. Application Dependency Discovery Using Matrix Factorization. In: IWQoS '12:6:1–6:4; 2012.

15. Peddycord Barry, Ning Peng, Jajodia Sushil. On the Accurate Identification of Network Service Dependencies in Distributed Systems. In: USENIX; 2012.

16. Salton Gerard, McGill Michael J.. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc.; 1986.

17. Salton Gerard, Fox Edward A., Wu Harry. Extended Boolean Information Retrieval. *Commun. ACM*. 1983;:1022–1036.

18. Mikolov Tomas, Chen Kai, Corrado Greg, Dean Jeffrey. Efficient Estimation of Word Representations in Vector Space. *CoRR*. 2013;.

19. Kusner Matt J., Sun Yu, Kolkin Nicholas I., Weinberger Kilian Q.. From Word Embeddings to Document Distances. In: ICML'15:957–966; 2015.

20. Le Quoc V., Mikolov Tomas. Distributed Representations of Sentences and Documents. In: :1188–1196; 2014.

21. Hang Huy, Wei Xuetao, Faloutsos Michalis, Eliassi-Rad Tina. Entelecheia: Detecting P2P botnets in their waiting stage. *IFIP Networking Conference, 2013, Brooklyn, New York, USA, 22-24 May, 2013*. 2013;:1–9.

22. Beigi Elaheh Biglar, Jazi Hossein Hadian, Stakhanova Natalia, Ghorbani Ali A.. Towards effective feature selection in machine learning-based botnet detection approaches. *IEEE Conference on Communications and Network Security, CNS 2014, San Francisco, CA, USA, October 29-31, 2014*. 2014;:247–255.

23. Carlson Josiah L.. *Redis in Action*. Manning Publications Co.; 2013.

24. Newman M E. Modularity and community structure in networks. *Proc Natl Acad Sci USA*. 2006;:8577-8582.

25. Blondel Vincent D, Guillaume Jean, Lambiotte Renaud, Lefebvre Etienne. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics Theory and Experiment*. 2008;.

26. Newman M. E. J., Girvan M.. Finding and evaluating community structure in networks. *Phys. Rev. E*. 2004;69:026113.

27. Agrawal Rakesh, Srikant Ramakrishnan. Fast Algorithms for Mining Association Rules in Large Databases. *Proceedings of the 20th International Conference on Very Large Data Bases*. 1994;.

28. Uno Takeaki, Kiyomi Masashi, Arimura Hiroki. LCM Ver.3: Collaboration of Array, Bitmap and Prefix Tree for Frequent Itemset Mining. *Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations*. 2005;.

29. Manku Gurmeet Singh. Frequent Itemset Mining over Data Streams. In: 2016 (pp. 209–219).

30. Sohrabi Mohammad Karim, Roshani Reza. Frequent itemset mining using cellular learning automata. *Computers in Human Behavior*. 2017;68:244–253.

31. Dainotti Alberto, King Alistair, Claffy Kimberly. Analysis of Internet-wide Probing Using Darknets. *Proceedings of the 2012 ACM Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*. 2012;:13–14.

32. *Workshop on Frequent Itemset Mining Implementations (FIMI'04). 2004.* `http://fimi.ua.ac.be/fimi04/`. `http://fimi.ua.ac.be/fimi04/`; 2004.