MalRank: A Measure of Maliciousness in SIEM-based Knowledge Graphs

Pejman Najafi
Hasso Plattner Institute
pejman.najafi@hpi.de

Alexander Mühle
Hasso Plattner Institute
alexander.muehle@hpi.de

Wenzel Pünter
Hasso Plattner Institute
wenzel.puenter@student.hpi.de

Feng Cheng
Hasso Plattner Institute
feng.cheng@hpi.de

Christoph Meinel
Hasso Plattner Institute
christoph.meinel@hpi.de

ABSTRACT

In this paper, we formulate threat detection in SIEM environments as a large-scale graph inference problem. We introduce a SIEM-based knowledge graph which models global associations among entities observed in proxy and DNS logs, enriched with related open source intelligence (OSINT) and cyber threat intelligence (CTI). Next, we propose MalRank, a graph-based inference algorithm designed to infer a node maliciousness score based on its associations to other entities presented in the knowledge graph, e.g., shared IP ranges or name servers.

After a series of experiments on real-world data captured from a global enterprise's SIEM (spanning over 3TB of disk space), we show that MalRank maintains a high detection rate (AUC = 96%) outperforming its predecessor, Belief Propagation, both in terms of accuracy and efficiency. Furthermore, we show that this approach is effective in identifying previously unknown malicious entities such as malicious domain names and IP addresses. The system proposed in this research can be implemented in conjunction with an organization's SIEM, providing a maliciousness score for all observed entities, hence aiding SOC investigations.

CCS CONCEPTS

• **Security and privacy** → *Intrusion detection systems*; *Malware and its mitigation*; • **Information systems** → *Data mining*.

KEYWORDS

Big data analytics in security, graph inference, graph mining, malicious domain detection, SIEM

ACM Reference Format:
Pejman Najafi, Alexander Mühle, Wenzel Pünter, Feng Cheng, and Christoph Meinel. 2019. MalRank: A Measure of Maliciousness in SIEM-based Knowledge Graphs. In *2019 Annual Computer Security Applications Conference (ACSAC '19), December 9–13, 2019, San Juan, PR, USA*. ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3359789.3359791

1 INTRODUCTION

The majority of today's medium to large sized organizations collect event logs generated by different components on the organization's premises. These event logs are shipped to a centralized system known as Security Information and Event Management (SIEM). Traditionally, this collection and storage has been mostly done for compliance. However, nowadays, the organizations monitor these events within their Security Operations Center (SOC), constantly seeking indicators of compromise such as a sudden spike in the number of requests made to a server, or access to an unknown port. More and more organizations are starting to realize the value and the potential of monitoring and analyzing these data.

SIEMs are expected to be the centralized repository for all events and information. If there is a threat that has managed to successfully bypass the perimeters of defense such as firewall, intrusion detection system, anti-virus, etc., it is quite likely that there are traces of its activities somewhere in these log-data shipped to the SIEM system.

There are numerous works introduced over the last decade that explore these dark data stored within SIEMs to derive security value, thus introducing concepts such as big data analytics, machine learning, data mining and pattern matching into cybersecurity [9].

The majority of those works study the application of machine learning and data mining for log analysis. Whilst machine learning has been successfully adopted in other domains, it has been extremely challenging to utilize it successfully in the cybersecurity domain. This is mostly due to the nature of security data. In contrast to other domains, there are no public, unbiased and up-to-date datasets that can be used to train a successful and realistic ML algorithm. Even if there exists such a perfectly labeled and unbiased dataset, there are still serious challenges in building a successful ML algorithm.

ML-based techniques such as anomaly detection assume that malicious events have a set of features that are distinguishable from those of legitimate events. However, this is a strong assumption as the majority of the features within these data are extremely volatile and temporal. All it takes is for the adversary to change a few lines to create a new set of values for a particular feature [17, 44]. Consider an ML algorithm that has learned to distinguish between legitimate and malicious URLs using number of subdomains and URL's entropy as distinct features. While this might work at first within its training set, it is wrong to assume it can classify under different circumstances, as all it takes, is to replace the URL string with a slightly altered string to defeat the ML classifier. Furthermore, if the ML algorithm tries to include broader patterns, the false positive rates will increase drastically as legitimate events will end up posing as malicious from the ML algorithm point of view. That is the biggest challenge in the application of generic data mining and machine learning algorithms in the security domain. In other words, ignoring the existence of an adversary that can constantly adapt to defeat the detection algorithm.

While the features utilized by the majority of previous efforts are relatively easy to change, there are other features that are harder to change. Meaningful associations among entities are such features, we would like to refer to those as global features. Local features in contrast are those that adhere to a single entity. For instance, in malicious domain detection, while URL structure as a feature is local to a single entity, IP address resolution or the ASNs/IP range mapping are global as they investigate the association among different entities. This concept of local/global features in regards to security analytics has been introduced by Khalil et al. [23] and Najafi et al. [35].

The hypothesis here is that an adversary's resources are limited. Therefore reuse of infrastructure is inevitable, e.g., usage of the same X.509 certificate, pool of IPs for domains, or even Tactics, Techniques and Procedures (TTPs). The other key intuition is that an external entity is less likely to be malicious when it is associated with a large number of benign entities, e.g., if a domain is visited by the majority of the workstations in a company, it is less likely to be a malicious domain [11][46]. Unsurprisingly, this reasoning (i.e., guilt-by-association [25]) is also adopted in SOC or forensic investigation. For example, investigating a potential malicious domain involves the investigation of open source intelligence and threat intelligence related to that domain, e.g., its registrar, subdomains, connected domains, TI feed observation, etc. In this regard, while an association with malicious entities does not necessarily imply maliciousness, it could be an indicator of a higher risk.

Ultimately, our approach is expected to utilize a small seed of threat intelligence to detect previously unknown malicious entities, therefore allowing us to increase the quality and quantity of our threat intelligence. It is also an effective threat detection techniques combating: malvertising, exploit kits' landing pages, rogue ASNs or registrars, fast-flux networks, domain shadowing, infrastructure reuse, malicious entities (e.g., malicious domains/IPs, rogue X.509 certificates).

The main distinct contributions of this paper can be summarized as follows:

- We propose the blueprint for a SIEM-based knowledge graph (section 2), emphasizing on the most important entities and relationships observed in DNS and proxy logs and the relevant Open-Source Intelligence (OSINT) as well as Cyber Threat Intelligence (CTI).
- We introduce, MalRank, a large-scale graph-based inference algorithm designed to infer maliciousness using the associations presented in the knowledge graph (section 3).
- We evaluate MalRank on a SIEM-based knowledge graph constucted from data collected by an international enterprise's SIEM (2-days of proxy, DNS, DHCP logs) enriched with related OSINT/CTI (IP ranges, ASN, DNS RRs, X.509 certificates) (section 4).
- We provide a comprehensive overview of all related graph-based inference algorithms particularly in the context of cybersecurity (section 5 and 3).

2 SIEM-BASED KNOWLEDGE GRAPH

2.1 Event Logs of Interest

SIEM systems within organizations are centralized repositories that are expected to hold all relevant security-related data. However, the amount and the variety of the data ingested into these SIEM systems vary drastically depending on the organization and its dedication to security practices, regulatory compliance, and analytics. These can include events and alerts generated by Intrusion Detection Systems (IDSs), firewalls, proxy servers, VPN servers, mail servers, workstations, authentication logs, NetFlow, HTTP/HTTPS traffic, DNS traffic, inventory, etc. Thus, it is important to determine a scope in which our knowledge graph is bound to.

In this regard for the purpose of this research, we are going to only focus on proxy, DNS, and DHCP logs.

Due to the fact that web traffic is typically allowed by the most of firewalls, HTTP, HTTPS, and DNS traffic are extensively abused by cybercriminals [1, 14] (e.g., bots communication with command-and-control servers), hence leading to the popularity of proxy and DNS log analysis in the security domain.

Oprea et al. [36] discuss the set of features extractable from proxy logs (e.g., domain connectivity, the referrer string, the user-agent string) that aid in the detection of malicious domains. Ma et al. [28, 29] address the same problem using URL' lexical and host-based features (e.g., number of dots) with the intuition that malicious URLs exhibit certain common distinguishing features. Zhang et al. [49] use term frequency/inverse document frequency (TF-IDF) algorithm to tackle malicious URL detection. Bilge et al. [7] introduce EXPOSURE, a system that employs large-scale passive DNS analysis to detect malicious domains using features such as the number of distinct IP addresses per domain, average TLL, the percentage of numerical characters, and etc. Antonakakis et al. [4] propose Notos, a similar system to EXPOSURE while distinguishing itself by incorporating complementary information such as the registration, DNS zones, BGP prefixes, and AS information. In a later research [5] Kopis is introduced, which separates itself from previous work by analyzing the DNS traffic at the upper level of DNS hierarchy rather than local recursive DNS servers.

Unlike these efforts which mostly target local features, we focus on global features extracted from proxy and DNS logs correlated with DHCP. Figure 1 shows the nodes and relationships extracted, and table 1 described each relationship.

2.2 OSINT and CTI

We would like to define Open Source Intelligence (OSINT) as any type of information gathered from publicly available sources (i.e., open-source) that provide context to those observed entities extracted from our SIEM-based data. OSINT has the potential to improve the inference and reasoning about maliciousness of an entity (e.g., IP range or ASN for an IP). Furthermore would like define Cyber Threat Intelligence (CTI) as subset of OSINT that can aid particularly with the threat detection tasks (e.g., list of Indicators of Compromise such as malicious domains) [12].

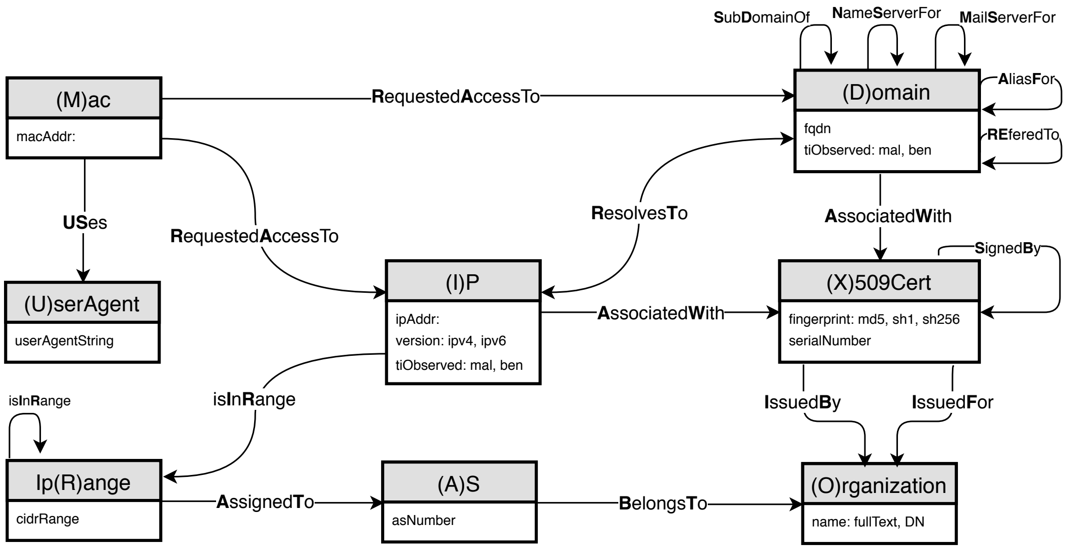

Figure 1: Knowledge graph schema, showing the extracted entities and relationships.

Table 1: The Description and the intuition behind the importance of each relationship used in the knowledge graph, as well as the data source used to extract such relationship.

Relationship	Description	Intuition	Data Source
subDomainOf	Dependency between different levels of a Fully Qualified Domain Name (FQDN), e.g., x.example.com, is a subdomain of example.com	Subdomain abusement (e.g., domain shadowing) is one of the simplest yet effective techniques utilized by cybercriminals to evade detection [35]	Proxy, DNS
requestedAccessTo	HTTP/HTTPS request or a DNS query from a client work station (MAC address) to/for domain/IP	Infected hosts are more likely to visit various malicious domains whereas user behavior on benign hosts should result in benign domain access [33]	Proxy, DNS (correlated with DHCP)
referedTo	Relationship between two domain/IP if one has referred to the other	The majority of the malware serving networks (e.g., exploit kits, drive-by-download, malvertising) are composed of a tree-like structure in which the victims are usually redirected through various hops before landing on the main distribution site [33]	Proxy
uses	The user agent used by an endpoint (MAC) for a specific HTTP/HTTPS query	If malware is trying to disguise itself as an innocent application (e.g., a browser) to reach out using HTTP, the user agent string might still differ from the major UA used by the workstation	Proxy
resolvesTo	DNS resolution of a domain name to an IPv4 address (DNS A records) or reverse DNS lookups for an IPv4 address (DNS PTR record)	In many cases, if a domain is listed as a malicious, intuitively we could assume that the IP address it resolves to is also malicious for the duration of that resolution [20, 23, 35, 50]	Proxy, DNS
nameServerFor and *mailServerFor*	Delegation of a domain name to a set of name servers (NS) or mail server (MX)	Infrastructure reuse by cybercriminals	DNS
aliasFor	Canonical names (CNAME) for a domain	domains connected by CNAME records share intrinsic relation and are likely to be in a homophilic state [39]	DNS
isInRange	Relationship between an IP and its IP address space	Cybercriminals tend to utilize almost an entire address range for their malicious purpose [30]	OSINT: IPR/ASN
assignedTo	Relationship between an IP range and its associated autonomous system (AS)	Malicious domains tend to be hosted on a pool of IPs hosted by specific hosting providers (i.e, ASs) [4, 23, 30]	OSINT: IPR/ASN
belongsTo	Relationship between an AS and the organization responsible for it	Same organization could be responsible for multiple ASs	OSINT: IPR/ASN
associatedWith	Association between an X.509 certificate and a DNS domain/IP, extraxtable from Subject Alternative Name, or Subject Common [13]	X.509 certificates reuse by cybercriminals. It could be costly and inefficient to register a certificate for each domain/IP under the attackers' control, and it is easier to reuse a pool of certificates	OSINT: X.509
signedBy	The validity of the certificate chain extractable from the chain's certificates fingerprints	Intended to counter malicious self-signed certificates and rogue intermediate certificates	OSINT: X.509
issuedFor	Relationship between the subject name (organization) and the X.509 certificate	Although the Subject's Distinguished Name (DN) can be a bogus name, it might still be useful to utilize with the intuition that a rogue organization might have more than one certificate	OSINT: X.509
issuedBy	Captures the issuer DN for a X.509 certificate	Set of issuer might be preferred by a group of cybercriminals due to an easier validation process [3], or a compromise [40]	OSING: X.509

Similar to event logs, OSINT and CTI could also pivot endlessly, therefore, it is important to also define a scope for the related OSINT. OSINT Framework[1] provides a good overview of all available OSINT sources, Enaqx[2] provides a comprehensive collection of OSINT tools and Slatman[3] provides a curated list of CTI. Furthermore, due to the fact that our event logs can reach up to 10 terabyte (TB) gernerated per day, it is also important to select those OSINT and CTI which can be collected/crawled at scale. Lastly, we would like to also distinguish between passive and active collection. We define active as those that require an active engagement with a server or an API for the collection, e.g., DNS RRs. While passives are those that can be collected as bulk without an active engagement (e.g., ASN).

Thus for the purpose of this research we limit our OSINT/CTI to *IPRanges*, *ASN*, *X.509 certificates*, *DNS Resource Records*, and our CTI to *malicious domains and IPs*. Please refer to Figure 1 for the related nodes and relationships extracted from those OSINT, and table 1 for the description and importance of each relationship.

3 MALRANK

3.1 Problem Definition and Requirements

At high level, we would like to reason about an entity based on its association with other entities, with the intuition that malicious entities tend to share some global properties. In this regard, graphs are ideal for this task due to their capability to preserve the correlation and association among different entities. That is why we formulate our problem as a graph-based inference problem. More specifically,

Given:

- A directed weighted graph $G(V, E)$ where V corresponds to the collection of entities (e.g., domains, IPs), and E corresponds to the set of relationships between those entities (e.g., resolvesTo).
- A Prior p (label) and prior confidence c defined over V, where $p \in \{0, 1\}$ and $c \in [0, 1]$. Where $p = 0$ represents a neutral node and $p = 1$ indicates a known malicious node. c represent the confidence in the label, 0 being no confidence ,and 1 absolute confidence in trustworthiness of that label (expected to be set according to the TI source).

Find:

- Maliciousness score s (MalRank) of a node x, i.e., $s(x) \in [0, 1]$. Higher MalRank score indicates a higher risk.

Graph-based inference has been studied widely in a variety of domains. Although it is referred to differently depending on the domain (e.g., influence, diffusion, propagation, classification), at the core, the problem can be simplified to the inference of nodes' properties based on their neighbors. In our case, inferring the maliciousness of a node based on the maliciousness of its neighbors. This is also known as guilt-by-association throughout the literature [46]. Before providing an overview of the most related algorithms, it is important to first define our main requirements for our use case that would allow us to better reason about the limitations of the previous algorithms.

3.1.1 *Single Diffused Label.*

Due to the fact that our graph (described in the previous section) is constructed from entities and relationships observed in an enterprise's SIEM, it is quite unlikely that the number of benign and malicious entities are proportional, i.e., majority of the entities are expected to be benign. This is due to the fact that the most traffic within an organization is expected to be benign. That is why it is important for us to consider only one label (maliciousness). Therefore the algorithm should be able to infer a maliciousness score for any given node based on the maliciousness scores of its neighbours while taking into consideration the number of neutral neighbours to reduce the maliciousness, i.e., if a node has a high degree with a large number of neutral neighbours, it is less likely to be malicious The intuition here is that, if an entity is observed many times, it is less likely to be a malicious entity (e.g., a malicious domain is more likely to be accessed by a few number of enterprise's workstations rather than the majority of them [32]). This requirement would also allow us to eliminate the super node issues (nodes with a high degree, e.g., content delivery networks, web hosting services, or advertising networks).

3.1.2 *Directed Weighted Propagation.*

The next important requirement is the ability to define edge weights. Since the knowledge graph is expected to consist of various types of nodes and edges, it is important that the algorithm is capable of considering how maliciousness should be propagated through a particular association. For instance, a *resolvesTo* edge should have a much higher influence than *requestedAccessTo*. Furthermore, although the majority of the relationships described in the previous section can be treated as bidirectional edges, the algorithm should be able to not only incorporate edge directions but also different edge weights on different directions. This would allow one to have much more control over not only the influence weights but also its directions. This is important as it can stop an adversary from defeating the algorithm by connecting to a large number of neutral nodes (e.g., referring to large number of benign domains, or adding CNAME record pointing to other legitimate domain). Although it's quite unlikely that this is happening at the moment, one must also consider this as part of the threat modeling.

3.1.3 *Maliciousness Influence Maximization.*

Maliciousness should be treated like a disease, i.e., the more malicious a node is, the higher its influence. This ensures that the maliciousness does not disappear within a graph of extreme bias towards benignness. Thus, the algorithm should have a mechanism to adjust the edge weights depending on the source's maliciousness score, i.e., if a node gets more malicious, the edge weights on the edges connecting that node to others should be increased accordingly thus allowing maliciousness to be propagated more effectively, and the opposite.

3.2 Background: Graph-based Inference Algorithms

As mentioned, there are various graph-based inference algorithms applied in different domains. While investigating all related graph-based inference algorithms in details is beyond the scope of this paper, it is still important to mention the influential related work

[1]http://osintframework.com/
[2]https://github.com/enaqx/awesome-pentest#osint-tools
[3]https://github.com/hslatman/awesome-threat-intelligence/

Table 2: Related graph-based inference algorithms and their shortcomings for the purpose of threat detection.

Algorithm	Description	Shortcomings
Belief Propagation (BP)	Also known as sum-product, one of the most popular and successful applications of label propagation used in probabilistic graphical models, e.g., Bayesian Networks and Markov Random Field. BP infers a node's label from some prior knowledge about that node and other neighboring nodes by iteratively passing messages between all pairs of nodes in the graph [38, 48]. BP is the most widely adopted graph based inference algorithm used for threat detection (further described in section 5).	First, BP is designed to work best with probabilistic graphical models which do not generally take into consideration the type of nodes/edges nor directions [16]. Second, BP expects a balance among labels which is not the case for us (i.e., extreme bias toward benignness). Hence, due to the numerical instability of multiplication, maliciousness ends up disappearing for the majority of the nodes that has a connection to a large number of benign nodes. Consider a node having 3 connection to neutral nodes with $P(x_{unknown}) = 0.5$ and 1 connection to a malicious node with $P(x_{mal}) = 1$, running BP until convergence will change the score of the node from originally 0.5 ($P(x_{mal}) = P(x_{ben}) = 0.5$) to $P(x_{mal}) = 0.508$, which is clearly, a low score for such a structure.
Random Walk with Restart (RWR)	RW-based algorithms emulate random walkers taking steps within a graph while having a small probability of teleporting to a random node, rather than following an out-edge. RWR has been successfully utilized in numerous related setting, e.g., Google's classic PageRank [8, 37], TrustRank [26], Distrust Rank [15] and SybilRank [10].	Inability to define different types of nodes and edges, or the ability to introduce weights on the edges. Although there have been a number of works tackling those specific issues, e.g., Personalized PageRank [6] and Topic-Sensitive PageRank [19] to incorporate the node's context (types), Biased Random Walks, Weighted PageRank [47] introducing the concept of edge weights, yet RWR are not adaptable for threat detection. RWR algorithms are designed to be a measure of importance and not beliefs, and importance is a relative measure which means, in the most of RWR-based algorithms the values are never created nor destroyed, rather it is passed from one node to another. This works great to measure importance, but not maliciousness. Maliciousness needs to be treated like a disease. Lastly, RWR-based algorithms assume a connected graph whereas our knowledge graph is extremely sparse [25].
Influence and Diffusion	Designed to study the influence and diffusion in social networks such as how a group of people might adopt an idea, or how information might spread. Linear Threshold (LT), and Independent Cascade (IC) [22] are among the most notable algorithms in this field.	These algorithms are extremely simple and require a major adjustment to support our main requirements, i.e., directional and weighted edges, echo cancelation, and influence maximization.
SimRank	A graph-based structural context similarity measure with the intuition that two objects are similar if they are related to similar objects, which intuitively can be adapted to measure influence [21].	Computational complexity, which makes it impossible to use considering the scale of our knowledge graph
Graph-based Semi-Supervised Learning	Also known as label propagation tackles the problem of unlabeled data with the principle idea that unlabeled data can be utilized to decide the metric between data points and improve models' performance [45]	Require a major adjustment to support our main requirements.
GraphSAGE	A node embedding algorithm that uses neural networks to learn embeddings for nodes in the graph structure while taking aggregated features from a node's local neighborhood [18]	Implementation challenges for the scale of our knowledge graph (i.e., challenges in parallel and scalable neural network).

and their shortcomings, hence leading us to the introduction of our MalRank algorithm, and how it is designed to fit our requirements the best. Table 2 provides a brief overview of the most relevant graph-based inference algorithms on their shortcomings. We would like to refer the reader to the references provided to learn more about the details of each algorithm.

3.3 MalRank Formulation

Let us denote the maliciousness score of a node $x \in V$ as $s(x)$, following our earlier intuition and definition, $s(x)$ can be calculated with:

$$s(x) = c_{s^o(x)} s^o(x) + (1 - c_{s^o(x)}) \frac{\sum\limits_{y \in N(x)} \sum\limits_{t \in T_{xy}} \hat{\omega}_{yx^{(t)}} . s(y)}{\sum\limits_{y \in N(x)} \sum\limits_{t \in T_{xy}} \hat{\omega}_{yx^{(t)}}} \tag{1}$$

where $s^o(x) \in [0, 1]$ refer to the prior of node x. If x is known malicious node $s^o(x) = 1$, and 0 otherwise (usually set if x is observed in a TI source). $c_{s^o(x)} \in [0, 1]$ is the prior strength of $s^o(x)$. This indicates the trust level of the prior. The value is decided according to the trust level for the corresponding TI source. This is introduced to control low quality threat intelligence, we shall discuss this later. $N(x)$ is the set of nodes neighboring node x, T_{xy} is the set of edge types between x and y. $\hat{\omega}_{xy^{(t)}}$ is the maximized/minimized edge weight on the edge type t directed from x to y.

3.3.1 *Maximized/Minimized Edge Weight, $\hat{\omega}_{xy^{(t)}}$.*

As discussed previously, there are three main requirements to control the propagation and influence: first, the ability to decay the influence differently on different edge types. This is achieved by introducing edge weights, $\omega_{xy^{(t)}}$ denoting the weight on the edge of type t between x and y. Second, the ability to have different weights on different directions of the edges. This is achieved by distinguishing the direction of the weight. i.e., $\omega_{xy^{(t)}} \neq \omega_{yx^{(t)}}$. It is worth to mention that this is how the algorithm sees the directions. Although our knowledge graph is a directed graph, from the algorithm perspective all edges are bidirectional, but the influence can be different on each direction. This way, one could define the $\omega_{xy^{(t)}} = 0$ and $\omega_{yx^{(t)}} = k$ if the edge type t between x and y is directed from y to x only. Lastly, the ability to adjust this decay based on the score of the influencer. Thus, introducing the maximized/minimized edge weight ($\hat{\omega}_{xy^{(t)}}$). This value is calculated by taking the weighted average of the original edge weight and the source maliciousness score:

$$\hat{\omega}_{xy^{(t)}} = \begin{cases} 0, & \text{if } \omega^o_{xy^{(t)}} = 0 \\ ks(x) + (1-k).\omega^o_{xy^{(t)}}, & otherwise \end{cases} \tag{2}$$

where $\omega^o_{xy^{(t)}}$ is the original edge weight on edge type t directed from node x to y and k is the maximizer factor which is expected to take a value between 0.5 and 0.8. The higher k values enforce a higher maximization for the new weight ($\hat{\omega}$) according to the influencer's score.

3.3.2 Iterative MalRank. MalRank can also be calculated iteratively as follows:

$$s^{i+1}(x) = c_{s^o(x)} s^o(x) + (1 - c_{s^o(x)}) \frac{\sum_{y \in N(x)} \sum_{t \in T_{xy}} m^i_{yx^{(t)}}}{\sum_{y \in N(x)} \sum_{t \in T_{xy}} \hat{\omega}^i_{yx^{(t)}}} \tag{3}$$

$$m^{i+1}_{yx^{(t)}} = [s^i(y) \underbrace{- (1 - c_{s^o(x)}) \frac{\sum_{t \in T_{xy}} m^i_{xy^{(t)}}}{\sum_{z \in N(x)} \sum_{t \in T_{zx}} \omega^i_{zx^{(t)}}}}_{\text{echo cancellation}}].\hat{\omega}^{i+1}_{yx^{(t)}} \tag{4}$$

$$\hat{\omega}^{i+1}_{yx^{(t)}} = \begin{cases} 0, & \text{if } \omega^o_{xy^{(t)}} = 0 \\ ks^i(y) + (1-k).\omega^o_{yx^{(t)}}, & otherwise \end{cases} \tag{5}$$

where $m^{i+1}_{yx^{(t)}}$ is the MalRank score sent from node y to node x in iteration $i+1$ over edge type t. $s^i(x)$ is the MalRank score of node x in iteration i.

4 EXPERIMENT: SIEM-BASED KNOWLEDGE GRAPH AND MALRANK

In this section, we present the details of our experiments running MalRank on a real-world SIEM-based knowledge graph.

4.1 Experiment Setup

4.1.1 Dataset Description. For the purpose of this research, we used two days of proxy, DNS, and DHCP logs (almost 3 billion events) collected by a large international enterprise SIEM, spanning over 3 TB. For details refer to appendix A.1 table 5.

The described event logs were later enriched with related OSINT as described in Section 2, i.e., ASN, X.509 certificates, and DNS RRs. In this regard, we used the sanitized version of the BGP prefixes, origin ASNs[4], and ASN to organization name mapping[5] available at thyme.apnic.net. These files span to approximately 20 MB total. For X.509 certificates we used censys[6] IPv4 snapshot which consists of the entire IPv4 address space scanned for all ports. This data spans over 1.2 TB of disk space. Note that, we were only interested in port 443 scans. Due to the enterprise's configuration for DNS servers to not log the DNS responses (DNS RRs), we had to pass all the DNS queries (logged by the DNS server) to our active OSINT-DNS enricher (scalable implementation of Gieben DNS library[7]) and log the responses ourselves.

Lastly, we utilized various sources (e.g., Google's Safe browsing, malwaredomains.com, etc.[8]) to collect our threat intelligence that was used as the ground truth trough out our experiments. Ultimately, we managed to passively collect a total of 1.5 million malicious indicators (domains and IPs) and 1 million benign domains (from Cisco's top 1 million domains, one can also use Alexa's tom 1 million domains). Note that our algorithm does not rely on benignness, and this list was collected only for the purpose of evaluation.

4.1.2 Hardware Setup. For the purpose of this research, we set up a big data processing cluster consisting of two Dell PowerEdge (R730, R820) and five Fujitsu Primergy RX600 with a total of 1,864 GB RAM, 24 CPUs (200 total cores), and 4 TB storage interconnected via 10 Gb optical fiber. In addition, the data was initially stored on an external Network Attached Storage (NAS) connected to the cluster via 3x 10Gb optical fiber. While the detailed description of the cluster setup is beyond the scope of this paper, we would like to mention that this cluster is backed by Kubernetes[9] for orchestration, Apache Spark[10] for distributed processing, and Apache Kafka[11] for distributed queueing. This allows us to scale our implementations both vertically and horizontally.

4.2 Implementation

The majority of graph algorithm libraries are designed for single machine use, thus making large-scale graph processing an extremely challenging task for today's big data. Pregel originally introduced by Malewicz et al. [31] brings graph algorithms into the map-reduce world by expressing graph algorithms as a sequence of iterations, in each of which a vertex can receive messages sent in the previous iteration, send messages to other vertices, and modify its own state and that of its outgoing edges or mutate graph topology. Using this vertex-centric intuition ("think like a vertex"), one can express a broad set of algorithms while parallelizing its computation across any number of nodes. GraphX is Apache Spark's API for parallel and fault-tolerant graph computation at scale. In this regard, we decided to implement the whole system (the knowledge graph and MalRank algorithm) with Pregel's computational model using Apache Spark GraphX. It is worth mentioning that throughout our experiments Apache Spark was configured to utilize a maximum of 72 CPU cores and 1.4TB of memory from the described hardware setup.

Figure 2, shows the high-level architectural design for the system implemented for the purpose of this research. As shown there are four main layers within the system: *Event logs PET* , *OSINT Enrichment*, *Loading*, and *MalRank*.

4.2.1 Event Logs PET. This layer is responsible to first, preprocess (e.g., prepare, clean, deduplicate, parse, validate, etc.) the raw event logs. Second, to extract entities and relationships of interest (as described in the Section 2 and Figure 1), and finally, transform

[4]http://thyme.apnic.net/current/data-raw-table
[5]http://thyme.apnic.net/current/data-used-autnums
[6]https://censys.io/
[7]https://github.com/miekg/dns
[8]https://github.com/hslatman/awesome-threat-intelligence/
[9]https://kubernetes.io/
[10]https://spark.apache.org/
[11]https://kafka.apache.org/

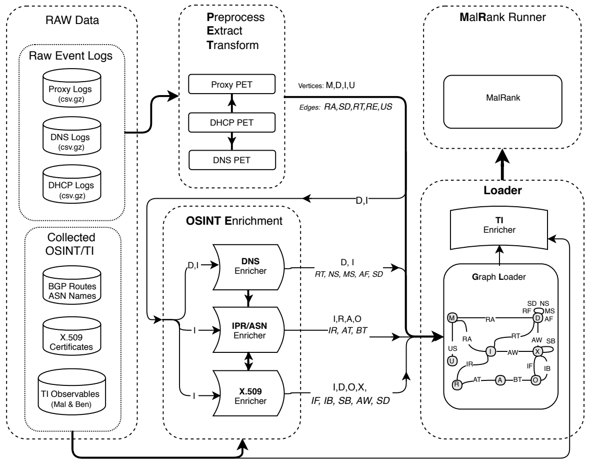

Figure 2: System architecture for the SIEM-based knowledge graph and MalRank algorithm implemented for the purpose of this research.

those into graph vertices and edges. The output of this layer is a set of independent vertices and edges which is then passed to the loading and the enrichment modules (as expressed in the Figure 2, where *D* represents a vertex type *Domain* and *RA* represents the relationship *RequestedAccessTo*, and so on).

Each vertex object has a *vid* (vertex identifier), *name*, *type*, *tiObservation*, and *mrScore*. Each edge has a *srcId*, *dstId*, *srcV* (the whole source vertex object), *dstV*, and *eType* (edge type). The intuition behind this specific design is to embrace micro service, stateless, and distributed design patterns. In this regard, despite, duplicating each vertex within each edge object, the system can scale-out more efficiently. This is due to the fact that the loading module can process the received vertices and edges independently no matter the order or distibution.

4.2.2 OSINT Enrichment. This layer consists of various enricher modules. Upon initialization, each enricher first loads and prepare the previously collected OSINT data (e.g., ASN mapping). Then, it subscribes to a repository (either a message queue or a file system directory) waiting for a batch of vertices. These vertices are provided as part of PET layer's output. Finally, the enrichers enrich those observed entities with their corresponding OSINT. For instance, ASN enricher listens for a batch of IPv4 vertices to enrich with IP range and ASN. The output of the OSINT enrichment layer is also a set of entities (vertices) and relationships (edges) which is passed to not only the loading module but also other enrichers for further enrichment as shown in Figure 2.

4.2.3 Loading. All extracted and processed entities and relationships arrive independently at the loading module. This module is responsible for de-duplicating, indexing, cleaning and combining all the vertices and edges. It is also responsible for labeling all vertices according to the TI collected previously while marking some for the purpose of evaluation. The output of this layer is the final labeled and processed distributed graph.

4.2.4 MalRank Runner. The output of the loading module is then passed to this layer which runs a distributed and iterative implementation of the MalRank algorithm. In each iteration for every edge in the graph, a map function calculates the MalRank msg to be sent to the destination vertex (according to MalRank Eq. (4)). Intuitively by the end of this mapper round, each vertex is going to receive a message for every incoming edge (from other vertices). Then the reduce function is used to combine all msgs at each vertex (MalRank Eq. 3). The reduce function itself is written to handle only two messages at a time, but it will be repeated until all of the messages have collapsed into a single message.

The described system is designed to work both in streaming and batch mode. However, in this research, we only utilized its batch mode. The current implementation of the MalRank does not support incremental updates. Therefore, one must re-run the MalRank algorithm to score newly added vertices. We would like to leave this to our future work, to implement the temporal incremental mode of MalRank which not only operates on streams but also takes time (first-seen and last-seen) into consideration.

4.3 Graph Structure

Before presenting our results it is important to understand some key characteristics of our final knowledge graph.

After passing the described data set through the event logs PET layer, 13 million vertices and 122 million edges were extracted. This process took approximately 4 hours on the described setup. Next, after passing through the enrichment layer an extra 6 million vertices and 12 million edges were added making a total of 19 million vertices and 134 million edges being passed to the loading module. The enrichment layer processing time was about 2 hours. After the loading module stage, the final graph was created with 15 million unique vertices and 132 million unique edges.

Figure 3 shows the degree and component distribution of the created knowledge graph. The distribution follows a power-law distribution which indicates an extremely sparse graph with very few edges between the majority of the node and minority with high degree connected clusters. This is understandable since the majority of our relationships enforce low degree when we have no global view of the association rather an enterprise-level view of only observed entities. The majority of high degree nodes were entities associated with the enterprise itself, e.g., enterprise domains, and workstations. For the details of vertex and edge types and their corresponding counts refer to table 7 in appendix A.3.

As mentioned before, the loading module is also responsible for labeling the nodes according to the collected TI as well as marking some for the purpose of evaluation. In this regard, out of 1.5 million combined TI collected in our experiment we had only 10 thousand matches (within 15 million vertices).

The loading module took almost 3 hours to complete, with the final graph spanning over 100 GB in memory across the cluster.

4.4 Evaluation

The evaluation in this section has two main objectives, first, evaluating the effectiveness of our approach and intuition as a threat detection technique. Second, evaluating the MalRank algorithm as a graph-based inference algorithm.

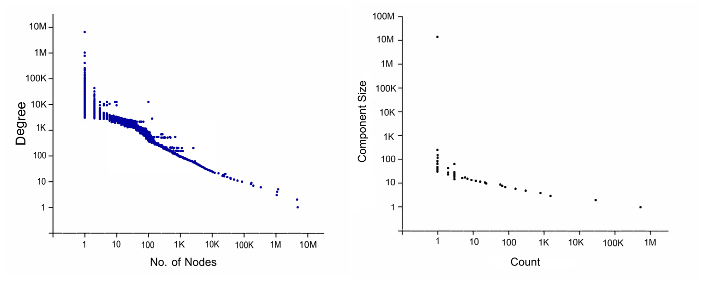

Figure 3: Node degree distribution and connected component size of the final graph in log-log scale.

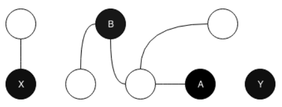

Figure 4: Pseudo-random sampling for the purpose of evaluation. In this regard to select the testing set we only consider connected nodes such as *A* and *B*.

4.4.1 Previously Known Malicious. Those are nodes that were known to be malicious at first (e.g., indicated by a TI source), however, they were marked as unknown when passed to the algorithm so that later they can be utilized to evaluate the algorithm detection capability (i.e., the testing set). Following the standard practices, in order to evaluate the detection capability of our approach, we decided to utilize a Receiver Operating Characteristic (ROC) curve as well as Precision and Recall (PR) curve.

It is worth to remember the testing set could only be derived from the 20K labeled data points that are connected. This is due to the fact that, first, the rest of the data points did not have any label in the first place that could be used for evaluation. Second, taking random samples in a sparse graph with low degree distribution could result in samples that have no connection to any other labeled nodes, thus eliminating the ability to evaluate the inference. Figure 4 illustrate this idea. If black nodes are previously known malicious nodes and white node are unknown (unlabeled) nodes, we only consider nodes such as *A* and *B* for our test samples as choosing *X* or *Y* will give us no value considering the fact that they are not connected to any other labeled nodes to allow effective maliciousness propagation.

More specifically, to select the testing samples for a class (e.g., maliciousness) in an evaluation run, the loading module, first, calculates the connected components (clusters) for that class, next, from each of those clusters that have more than one member, selects $\sqrt{k}$ nodes at random (where k is the number of labeled nodes in each cluster). For instance, if Figure 4 is our knowledge graph we would take only *A* or *B* at random. In our experiment this process led to the random selection of approximately 2,000 nodes (1,000 known malicious and 1,000 known benign nodes) in each evaluation run.

Figure 5 shows the ROC and PR plot for 9 iterations of MalRank with configuration described in Table 3 on the described data set with the described testing set. Note that the whole experiment (including the sampling) was repeated 4 times to flatten out the outliers. The results show a high accuracy (AUC = 96%, with the peak *F1-score* = 0.905, and *Accuracy* = 0.900). Please refer to appendix A.4 figure 7 for the details of the experiment on different number of iterations.

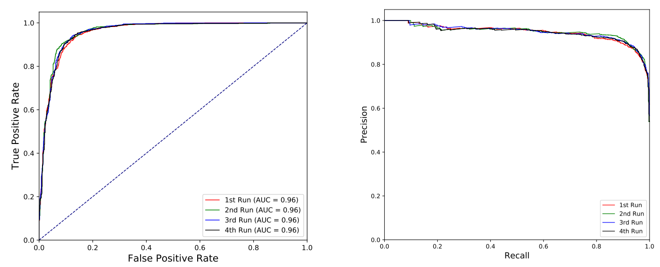

Figure 5: Receiver Operating Characteristic and Precision Recall curves of 9 iterations of MalRank ran 4 times.

In order to better understand the algorithm's results we investigated the false positives (FPs) and false negatives (FNs). In this regard, we had the following observations; first, in contrast to our original intuition and the common practice used in past efforts, top-ranked domains by Cisco Umbrella or Alexa did not necessarily reflect benign domains. In this regard, there were multiple instances of domains being marked as malicious after MR due to association with multiple malicious entities despite appearing on Umbrella top 1 million domains (therefore FP). This was also confirmed by [42]. A good example of this was *world.rickstudio.ru*, which appeared among the top 1 million domains under the umbrella while being malicious. This was also due to our random selection of benign samples from the entire 1 million, one should at least ensure that the samples are from the top k thousand.

Other legitimate false positives were due to the association of malicious IPs to benign domains, this could be explained by the web hosters that might share IPs among domains.

Majority of the false negatives were also due to bad quality Threat Intelligence. For instance, ingesting a TI source where *github.com* and *google.de* were marked as malicious by the TI. These were later marked as non-malicious by our algorithm due to their association with major neutral nodes. Other FNs were also due to content delivery network (CDN) in which a TI was reporting an IP malicious while it was also associated with a couple of legitimate domains through a proxy server.

4.4.2 Comparison with Belief Propagation. In order to evaluate MalRank's efficiency and accuracy, we decided to compare it with the Hewlett Packard's implementation of Belief Propagation implemented in Apache Spark [2]. BP is the most popular algorithm used throughout the literature as a graph-based inference algorithm in the context of security, See Section 5.

Figure 6 shows the ROC plot for 9 iterations of MR vs. 9 iterations of BP on the same knowledge graph with the same testing set.

Table 4 and 3 shows BP and MR algorithm configuration for this experiment respectively. It is also worth to note that due to the fact that MalRank utilizes only one class label (maliciousness) whereas BP requires at least two class labels (maliciousness and benignness), we decided to configure BP by initializing all unknown and benign nodes with 0.5 as the probability of maliciousness (i.e., $P(x_{mal}) =$

Table 3: MalRank configuration for the experiment

MR Parameter	Description
$\forall t \in T_{xy} : \omega^{o}_{xy^{(t)}} = \omega^{o}_{yx^{(t)}} = 0.5$	All edge weights initialized with 0.5 in both directions
$\forall_{x}, s_{m^{o}(x)} = 0.8$	Prior strength (TI source trust score) for all sources initialized with 0.8
$k = 0.7$	Maximizer factor for $\hat{\omega}_{(t)}$
$M^{o}(x) = \begin{cases} 0.9, & \text{if } x \in \{X_{mal}\} \\ 0.0001, & \text{otherwise} \end{cases}$	Node x initial score depending on whether it was observed in TI

Table 4: Node and edge Potential configuration for Belief Propagation experiment.

Edge Potential			Node Potential		
$\psi_{ij}(x_i, x_j)$	$x_j = ben$	$x_j = mal$	Node	$P(mal)$	$P(ben)$
$x_i = ben$	$0.5 + \epsilon$	$0.5 - \epsilon$	Malicious	0.99	0.01
$x_i = mal$	$0.5 - \epsilon$	$0.5 + \epsilon$	Benign	0.5	0.5

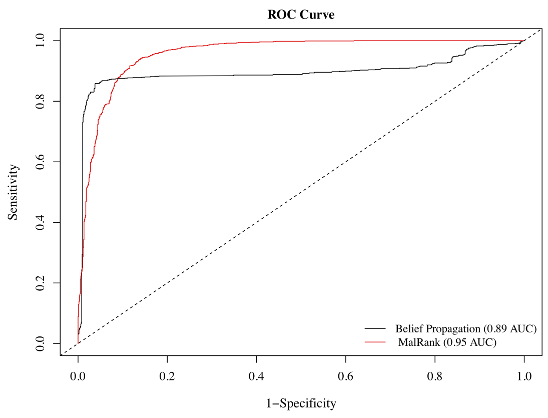

Figure 6: ROC curve for 9 iterations of MalRank vs. Belief Propagation.

$P(x_{ben}) = 0.5$) and 1 for those previously known malicious nodes (i.e., $P(x_{mal}) = 1$, $P(x_{ben}) = 0.5$) .

As shown in the Figure 6, MR is outperforming BP not only in terms of accuracy but also the run-time. In this regard with almost an identical implementation in GraphX, MalRank finishes 9 iterations within 20 minutes, whereas BP takes about 2 hours. It is worth to mention that BP memory utilization was almost 6 times higher than MalRank. It is also important to note that while BP shows high accuracy, this is not true in all cases. The main reason for this accuracy in this experiment is that the majority of the testing set were nodes with a low-degree. Due to numerical instability of multiplication, BP starts introducing errors when a node's degree increases. This was observed mostly in our next experiment when investigating previously unknown threats.

4.5 Case Studies of Previously Unknown Malicious

ROC and PR curves are useful to evaluate a threat detection technique. Nevertheless, plotting such curves requires a testing set, and as mentioned before the choice of the testing set for our approach is a challenging task. More specifically, our approach is not a generic classifier that can classify any arbitrarily given entity as malicious or non-malicious. Instead, it is an inference model designed to increase the quantity and the quality of our threat intelligence by discovering new malicious entities associated with previously known malicious entities. Therefore, the most relevant validation for us is the evaluation of previously unknown and inferred maliciousness. In this regard, we decided to manually investigate top high MalRank scored nodes which did not have a prior (not observed in our TI). For this manual investigation, we utilized *ThreatCrowd, VirusTotal, ThreatMiner, URLVoid, AlienVault, Robtex and MXToolBox*. We categorized our investigation depending on the type of the entity.

When investigating the top 200 Domains/IP, we were able to find an indicator for 67% of those. While the majority of those were result of maliciousness inference on *resolvesTo* relationship, there were those high degree nodes that were scored mostly due to *mailServerFor*, *isInRange*, and *referedTo*. As a result we were able to identify large number of previously unknown malicious domains and IPs. We were also able to identify surprising number of pornographic domains that were ranked high. We assume this is due to malvertising, clickjacking techniques widely adopted by such domains.

When investigating the top high scored X.509 certificates we were mostly capable of identifying parking domains (i.e., domains registered solely for the purpose of displaying web advertisements with typically no real content [27]) and rogue web hosters (e.g., **.000webhostapp.com* whom its subdomains are regularly misused by cybercriminals to host scams). Hence allowing us to capture further potentially unknown malicious Domains/IPs.

We had the same observation when investigating top malicious organization, as one of the top MalRanks scored ones, was the organization responsible for **.000webhostapp.com*. We were also able to identify number of self signed certificates associated with an organization which lead us to find associated Domains/IPs which were in fact classified as malicious by VirusTotal.

We didn't investigate MAC address, ASN nor User Agent (UA) as the majority of nodes did not come up with high scores (less than 0.2). This was reasonable considering we had only access to two days of data (i.e., low chance of major outliers)

Lastly, we investigated a set of malicious domains which was identified by the enterprise's SOC analysts to be associated with malware beaconing on a number of clients. More specifically these were domain starting with imp (i.e., `^imp\\..+`) such as *imp.searchlff.com*[12]. When we checked the MalRank score of these previously unknown malicious domain, we noticed that the algorithm scored them as malicious (0.6 - 0.7) due to association with TI (through IP address and range sharing).

Interestingly, when we investigated the BP score for the above findings, we could verify our initial intuition, i.e., BP's limitation to infer maliciousness for high-degree nodes with unbiased labeled

[12]https://www.threatcrowd.org/domain.php?domain=imp.searchlff.com

neighbours. In this regard, the majority above previously unknown malicious entities were scored between 0.51-0.56 which makes it susceptible to missclassification by BP.

In summary, although we were unable to validate all high score nodes, according to our investigations MalRank proved to be an effective method to increase the quality and the quantity of threat intelligence. While one could argue, that these were low-hanging-fruits, we still see the value in our approach. Furthermore, It is also worth to note that the SIEM logs used within this research were from an international enterprise that already utilizes various security measures and practices (e.g., IDS, AV, Proxy/DNS blacklisting, signature checking, and etc.) therefore making it rare to encounter various threats, yet MalRank was capable of detecting valuable previously unknown malicious entity, i.e., the detection of a potential malware beaconing case.

It is worth to note that, throughout our investigation, we came across a number of nodes and cases in which the nodes were scored high (malicious) but we could not validate the maliciousness as it seemed harmless (e.g., parked domains, link farms, and other dubious domains/IPs). Even though such entities seemed non-malicious (FPs), we argue that blocking them at the enterprise level should not have a drastic effect, as the main reason for their false classification was having a number of association with high scored nodes.

5 RELATED WORK

This section provides an overview of the most relevant and influential work in the context of graph-based inference for cybersecurity.

Chau et al. [11] introduce Polonium as one of the first and arguably the most successful works that tackles the problem of malware detection using large-scale graph inference with the intuition that good applications are typically used by many users, whereas, unknown (i.e., potentially malicious) applications tend to only appear on few computers. The authors achieve this by running an adopted version of belief propagation on an undirected, unweighted bipartite machine-file graph. In similar research, Tamersoy et al. [46] propose Aesop, which tackles the same problem using locality sensitive hashing to measure the similarity between files to eventually construct a file-file bipartite graph and running BP to infer files' goodness based on its neighbors.

Manadhata et al. [32] address the problem of detecting malicious domains by using enterprise HTTP proxy logs to construct a host-domain bipartite graph capturing workstations' connection to external domains, then running BP to discover malicious domains based on a set of seed malicious nodes. The intuition in this research is that infected hosts are more likely to visit various malicious domains whereas user behavior on benign hosts should result in benign domain access.

Khalil et al. [23] address the same problem using passive DNS data focusing on a domain-IP bipartite graph with the intuition that a domain/IP is malicious if it has strong association to a previously known malicious domain/IP. While the authors evaluate BP as part of their evaluation, their main proposal takes a different approach. In this regard, the authors formulate the problem as a similarity measure between a pair of domains based on the number of IPs shared to derive a domain-domain similarity graph and use a path-based algorithm to infer a maliciousness score for each domain according to their topological connection to known malicious domains. In a later research Khalil et al. [24] discuss the limitations of their previous work [23] which is the computational complexity leading them to adopt belief propagation again on an adjusted graph while emphasizing on ASN.

Zou et al. [50] takes a similar approach focusing on DNS logs. In this regard, the authors focus on three main relationships extractable from DNS logs: 1) connection request from an enterprise's workstation to a domain, 2) resolves to relationship (DNS record type A) which indicates a domain resolving to an IPv4 address, and 3) CNAME DNS RRs which indicates a domain being an alias for another domain.

Najafi et al. [35] also tackle the problem of malicious domain/IP detection using BP on a property graph focusing on domain to IP resolution (DNS record type A), domain to domain referral (proxy log referer header) and sub-domain relationship.

Other works include, Huang et al. [20] investigating the connection between domain, IP, and URL. Oprea et al. [36] addressing the early-stage APT detection using BP on host-domain graph extracted from proxy logs. Rahbarinia et al. [41] proposing Segugio to detect new malware-control domains based on DNS traffic analysis with a very similar intuition to Manadhata et al. [32]. Mishsky et al. [34] explore the same issues with the slightly different angle. Simeonovski et al. [43] approach the problem using taint-style techniques for propagation of labels in a property graph built from nodes consisting of domains, organizations, and ASNs. Finally, Peng et al. [39] build a domain-domain graph using DNS CNAME RRs with the intuition that domains connected by DNS CNAME RRs share intrinsic relations and are likely to be in a homophilic state.

While the majority of the previous works focus on single edge type isolated (i.e., a bipartite graph), we construct a comprehensive knowledge graph which incorporates various types of nodes and edges. To the best of our knowledge, this is the first work exploring knowledge graphs at this scale within the security domain. Furthermore, in contrast to other works, while we evaluate BP, we introduce a much more effective and efficient algorithm that allows us to better infer maliciousness in knowledge graphs.

6 LIMITATION AND FUTURE WORK

The biggest limiting factor in our work was the quality of the threat intelligence which was also the main factor for the high false positive rate. MalRank is designed to infer maliciousness using a small set of previously known malicious nodes as seeds. These seeds are expected to be validated TI. However, the majority of the publicly available TI sources are low quality with a large number of false positives, and inference based on false TI results in further false positives. Although MalRank has a mechanism to incorporate the quality of threat intelligence, throughout our experiments we did not have any approach to rate the sources of the TI. In this regard, the majority of our false positives were due to bad TI ingestion from a source. In our future work, we would like to utilize a better quality TI (perhaps at the cost of API limitation) and introduce an approach to evaluate our TI sources to derive trust scores for each source.

Despite the algorithms capability to support directional edge weights, throughout our experiments, we decided to rely on naive expert knowledge for edge weights by specifying all edges as bidirectional with 0.5 as their weight on each direction. Although, MalRank has a mechanism to adjust the weights within each iteration (depending on the source maliciousness score), therefore allowing us to not worry too much about the exact weights, it is expected that one defines the initial edge weights with more precision. However. deciding the initial edge weights is an extremely challenging task which we would like to explore in our future work.

We would also like to explore MalRank algorithmic improvements. More specifically, first, finding the closed formula in terms of matrix operation which should improve its efficiency and better reason about its convergence. Second, expansion to support incremental updates in streaming mode. Last experimenting with other aggregator functions such as LSTM and Pooling [18] as opposed to weighted average.

It is also interesting to combine our approach with previous works that focused on local features. In this regard, one can utilize those local features to derive a prior (initial score) for each node (e.g., 0.5) and then run MalRank and look back at the nodes. If the MR score was increased further, one can conclude maliciousness with a higher confidence as the node was marked malicious based on not only its local features but also global. Alternatively, one can also use MalRank score as a local feature to combine with other features to train another ML classifier to further improve detection accuracy. We expect that one could drastically reduce the false positive rate by ensembling MalRank with other approaches.

Lastly, we only covered three sources of event logs within SIEMs, however, today's SIEMs collect much more than just proxy, DNS, and DHCP. It would be also interesting to extend our knowledge graph with more event logs and perhaps more OSINT (e.g., registrar). In that case, it would also be interesting to experiment with different graph schemas.

7 CONCLUSION

In this paper, we first introduced the intuition behind global features for threat detection. Next, we presented the SIEM-based knowledge graph which is constructed from entities and relationships observed within data captured by an enterprise's SIEM. We also covered the most relevant OSINT and TI that can be collected at scale. We formulated threat detection as a large-scale graph inference problem. This led us to the introduction of our proposed algorithm named MalRank, a scalable graph-based inference algorithm designed to infer a node's maliciousness score based on its association to other nodes. We also discussed MalRank's unique characteristics that sets it apart from other graph-based inference algorithms.

For the purpose of evaluation, we implemented the proposed knowledge graph, as well as MalRank algorithm on a big data cluster consisting of 7 physical servers (on top of Apache Spark). We used two days of proxy and DNS logs collected from a large international organization's SIEM to construct the proposed knowledge graph. Next, we used publicly available OSINT to enrich entities observed, and TI to label small set of nodes as malicious. After running MalRank on the generated knowledge graph, we showed that our approach can achieve a high accuracy (AUC = 96%). Furthermore, proving effective to increase threat intelligence by discovering a large number of new malicious entities.

REFERENCES

[1] 2018. ATT&CK: Standard Application Layer Protocol. https://attack.mitre.org/wiki/Technique/T1071
[2] 2018. Loopy Belief Propagation. https://github.com/HewlettPackard/sandpiper. Accessed: 2018-08-10.
[3] Osama Almanna. 2016. StartSSL Domain validation (Vulnerability discovered). http://oalmanna.blogspot.com/2016/03/startssl-domain-validation.html
[4] Manos Antonakakis, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster. 2010. Building a Dynamic Reputation System for DNS. In *USENIX security symposium.* 273–290.
[5] Manos Antonakakis, Roberto Perdisci, Wenke Lee, Nikolaos Vasiloglou, and David Dagon. 2011. Detecting Malware Domains at the Upper DNS Hierarchy. In *USENIX security symposium*, Vol. 11. 1–16.
[6] Bahman Bahmani, Abdur Chowdhury, and Ashish Goel. 2010. Fast incremental and personalized pagerank. *Proceedings of the VLDB Endowment* 4, 3 (2010), 173–184.
[7] Leyla Bilge, Engin Kirda, Christopher Kruegel, and Marco Balduzzi. 2011. EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis. In *Ndss.*
[8] Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems* 30, 1-7 (1998), 107–117.
[9] Anna L Buczak and Erhan Guven. 2016. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials* 18, 2 (2016), 1153–1176.
[10] Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Tiago Pregueiro. 2012. Aiding the detection of fake accounts in large scale social online services. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation.* USENIX Association, 15–15.
[11] Duen Horng "Polo" Chau, Carey Nachenberg, Jeffrey Wilhelm, Adam Wright, and Christos Faloutsos. 2011. Polonium: Tera-scale graph mining and inference for malware detection. In *Proceedings of the 2011 SIAM International Conference on Data Mining.* SIAM, 131–142.
[12] D Chismon and M Ruks. 2015. Threat intelligence: Collecting, analysing, evaluating. *MWR InfoSecurity Ltd* (2015).
[13] David Cooper, Stefan Santesson, Stephen Farrell, Sharon Boeyen, Russell Housley, and William Polk. 2008. *Internet X. 509 public key infrastructure certificate and certificate revocation list (CRL) profile.* Technical Report.
[14] MITRE Corporation. 2018. ATT&CK: Commonly Used Port. https://attack.mitre.org/wiki/Technique/T1043
[15] Brian Davison. 2006. Propagating trust and distrust to demote web spam. (2006).
[16] Dhivya Eswaran, Stephan Günnemann, Christos Faloutsos, Disha Makhija, and Mohit Kumar. 2017. Zoobp: Belief propagation for heterogeneous networks. *Proceedings of the VLDB Endowment* 10, 5 (2017), 625–636.
[17] Maryam Feily, Alireza Shahrestani, and Sureswaran Ramadass. 2009. A survey of botnet and botnet detection. In *Emerging Security Information, Systems and Technologies, 2009. SECURWARE'09. Third International Conference on.* IEEE, 268–273.
[18] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems.* 1024–1034.
[19] Taher H Haveliwala. 2003. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE transactions on knowledge and data engineering* 15, 4 (2003), 784–796.
[20] Yonghong Huang and Paula Greve. 2015. Large scale graph mining for web reputation inference. In *Machine Learning for Signal Processing (MLSP), 2015 IEEE 25th International Workshop on.* IEEE, 1–6.
[21] Glen Jeh and Jennifer Widom. 2002. SimRank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 538–543.
[22] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 137–146.
[23] Issa Khalil, Ting Yu, and Bei Guan. 2016. Discovering malicious domains through passive DNS data graph analysis. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security.* ACM, 663–674.
[24] Issa M Khalil, Bei Guan, Mohamed Nabeel, and Ting Yu. 2018. A Domain is only as Good as its Buddies: Detecting Stealthy Malicious Domains via Graph Inference. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy.* ACM, 330–341.
[25] Danai Koutra, Tai-You Ke, U Kang, Duen Horng Polo Chau, Hsing-Kuo Kenneth Pao, and Christos Faloutsos. 2011. Unifying guilt-by-association approaches: Theorems and fast algorithms. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases.* Springer, 245–260.
[26] Vijay Krishnan and Rashmi Raj. 2006. Web spam detection with anti-trust rank.. In *AIRWeb*, Vol. 6. 37–40.
[27] Marc Kührer, Christian Rossow, and Thorsten Holz. 2014. Paint it black: Evaluating the effectiveness of malware blacklists. In *International Workshop on Recent Advances in Intrusion Detection.* Springer, 1–21.

[28] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. 2009. Beyond blacklists: learning to detect malicious web sites from suspicious URLs. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1245–1254.
[29] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. 2009. Identifying suspicious URLs: an application of large-scale online learning. In *Proceedings of the 26th annual international conference on machine learning*. ACM, 681–688.
[30] Dhia Mahjoub. 2013. Monitoring a fast flux botnet using recursive and passive DNS: A case study. In *eCrime Researchers Summit (eCRS), 2013*. IEEE, 1–9.
[31] Grzegorz Malewicz, Matthew H Austern, Aart JC Bik, James C Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. 2010. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM, 135–146.
[32] Pratyusa K Manadhata, Sandeep Yadav, Prasad Rao, and William Horne. 2014. Detecting malicious domains via graph inference. In *European Symposium on Research in Computer Security*. Springer, 1–18.
[33] Niels Provos Panayiotis Mavrommatis and Moheeb Abu Rajab Fabian Monrose. 2008. All your iframes point to us. In *USENIX Security Symposium. USENIX*. 1–16.
[34] Igor Mishsky, Nurit Gal-Oz, and Ehud Gudes. 2015. A topology based flow model for computing domain reputation. In *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 277–292.
[35] Pejman Najafi, Andrey Sapegin, Feng Cheng, and Christoph Meinel. 2017. Guilt-by-Association: Detecting Malicious Entities via Graph Mining. In *International Conference on Security and Privacy in Communication Systems*. Springer, 88–107.
[36] Alina Oprea, Zhou Li, Ting-Fang Yen, Sang H Chin, and Sumayah Alrwais. 2015. Detection of early-stage enterprise infection by mining large-scale log data. In *Dependable Systems and Networks (DSN), 2015 45th Annual IEEE/IFIP International Conference on*. IEEE, 45–56.
[37] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web.* Technical Report. Stanford InfoLab.
[38] Judea Pearl. 2014. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier.
[39] Chengwei Peng, Xiaochun Yun, Yongzheng Zhang, Shuhao Li, and Jun Xiao. 2017. Discovering Malicious Domains through Alias-Canonical Graph. In *Trustcom/BigDataSE/ICESS, 2017 IEEE*. IEEE, 225–232.
[40] J Ronald Prins and Business Unit Cybercrime. 2011. DigiNotar Certificate Authority breach 'Operation Black Tulip'. *Fox-IT, November* (2011). https://www.rijksoverheid.nl/ministeries/ministerie-van-binnenlandse-zaken-en-koninkrijksrelaties/documenten/rapporten/2011/09/05/diginotar-public-report-version-1
[41] Babak Rahbarinia, Roberto Perdisci, and Manos Antonakakis. 2015. Segugio: Efficient behavior-based tracking of malware-control domains in large ISP networks. In *Dependable Systems and Networks (DSN), 2015 45th Annual IEEE/IFIP International Conference on*. IEEE, 403–414.
[42] Paul Royal. 2012. Maliciousness in top-ranked alexa domains. *Online]. https://www. barracudanetworks. com/blogs/labsblog* (2012).
[43] Milivoj Simeonovski, Giancarlo Pellegrino, Christian Rossow, and Michael Backes. 2017. Who controls the internet?: Analyzing global threats using property graph traversals. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 647–656.
[44] Elizabeth Stinson and John C Mitchell. 2008. Towards Systematic Evaluation of the Evadability of Bot/Botnet Detection Methods. *WOOT* 8 (2008), 1–9.
[45] Amarnag Subramanya and Partha Pratim Talukdar. 2014. Graph-based semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 8, 4 (2014), 1–125.
[46] Acar Tamersoy, Kevin Roundy, and Duen Horng Chau. 2014. Guilt by association: large scale malware detection by mining file-relation graphs. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1524–1533.
[47] Wenpu Xing and Ali Ghorbani. 2004. Weighted pagerank algorithm. In *Communication Networks and Services Research, 2004. Proceedings. Second Annual Conference on*. IEEE, 305–314.
[48] Jonathan S Yedidia, William T Freeman, and Yair Weiss. 2003. Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium* 8 (2003), 236–239.
[49] Yue Zhang, Jason I Hong, and Lorrie F Cranor. 2007. Cantina: a content-based approach to detecting phishing web sites. In *Proceedings of the 16th international conference on World Wide Web*. ACM, 639–648.
[50] Futai Zou, Siyu Zhang, Weixiong Rao, and Ping Yi. 2015. Detecting malware based on DNS graph mining. *International Journal of Distributed Sensor Networks* 11, 10 (2015), 102687.

A EXPERIMENT DETAILS

A.1 SIEM Logs

The statistics of SIEM logs used for the purpose of this research

Table 5: The statistics of our SIEM logs for the experiment

Source	Size	#Events
DNS Logs	2TB (120GB gzip compressed)	2 billion
Prxy Logs	1TB (100GB gzip compressed)	755 million
DHCP Logs	12GB (800MB gzip compressed)	4m

A.2 Hardware Setup

Table 6: Description of the hardware used in this research

Server	Details	#Servers
Dell PowerEdge R730	2x Intel Xeon E5-2690 @ 2.6GHz (14 cores each), 768GB RAM, 1TB SSD	1x
Dell PowerEdge R820	2x Intel Xeon E5-4617 @ 2.90GHz (6 cores each), 328GB RAM, 540GB HDD	1x
Fujitsu Primergy RX600	4x Intel Xeon E7-4820 @ 2.60GHz (8 cores each), 256GB RAM, 2TB SSD	1x
Fujitsu Primergy RX600	4x Intel Xeon E7-4820 @ 2.60GHz (8 cores each), 128GB RAM and 1TB SSD	4x

A.3 Vertices and Edges

Table 7: The count of each vertex and edge type loaded into the final knowledge graph.

Edges		Vertices	
Type	#	Type	#
requestedAccessTo	103 million	Domain	12.4m
subDomainOf	10m	Ipv4	1.8m
resolvedTo	7m	Organization	0.28m
uses	3m	X509cert	0.27m
aliasFor	2.5m	Mac	0.12m
referedTo	2m	ipRange	0.08m
associatedWith	1.7m	Useragent	0.07m
isInRange	1.3m	Asn	0.02m
mailServerFor	1m		
issuedBy	0.26m		
issuedFor	0.26m		
signedBy	0.23m		
nameServerFor	0.12m		
assignedTo	0.08m		
belongsTo	0.03m		

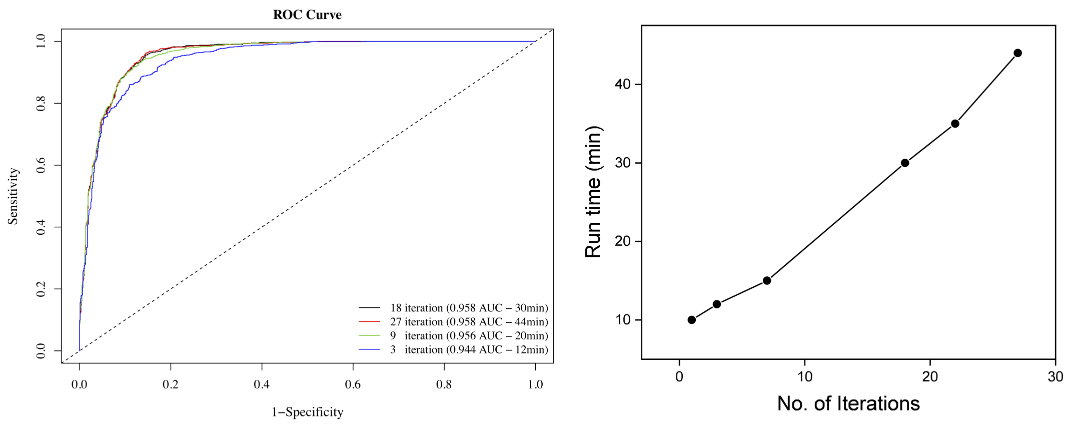

Figure 7: ROC Curve for different number of MalRank iteration based on the same testing/evaluation set and their corresponding run-time.

A.4 Number of Iterations

In the majority of our experiments, we chose 9 as maximum number of iterations. The main reason for this choice is that; within our knowledge graph, the inference from more than 4 hops aways does not make much sense. Consider the *requestedAccessTo* edge isolated from all the others. This edge captures the relationship between a MAC address and a set of domain/IP nodes (shaping a bipartite graph). In order to decide the label for a domain, it makes sense to traverse back to the MAC address that requested this domain and check the score for all other domains visited by that MAC (perhaps an indication of a malware trying to reach out to malicious IPs or Domain for C&C), i.e., inference from two hops away. It also makes sense to check another two hops, i.e., check whether there exist other workstations which connect to similar known malicious nodes. However, going deeper than that loses the intuition entirely. This could also be observed in Figure 7 which shows the ROC curve for a different number of iterations. As shown in the figure, while the results do not vary drastically after 7 iterations, the algorithm runtime increases drastically. For instance, when we change from 9 to 18 iterations we increase the accuracy by 0.2% and the run-time by 50%.