DoITrust: Dissecting On-chain Compromised Internet Domains via Graph Learning

Shuo Wang*†, Mahathir Almashor*†, Alsharif Abuadbba*†, Ruoxi Sun*,
Minhui Xue*†, Calvin Wang*†, Raj Gaire*†, Surya Nepal*† and Seyit Camtepe*†
*CSIRO's Data61, Australia
†Cybersecurity CRC, Australia

Abstract*—Traditional block/allow lists remain a significant defense against malicious websites, by limiting end-users' access to domain names. However, such lists are often incomplete and reactive in nature. In this work, we first introduce an expansion graph which creates organically grown Internet domain allow-lists based on trust transitivity by crawling hyperlinks. Then, we highlight the gap of monitoring nodes with such an expansion graph, where malicious nodes are buried deep along the paths from the compromised websites, termed as "on-chain compromise". The stealthiness (evasion of detection) and large-scale issues impede the application of existing web malicious analysis methods for identifying on-chain compromises within the sparsely labeled graph. To address the unique challenges of revealing the on-chain compromises, we propose a two-step integrated scheme, DoITrust, leveraging both individual node features and topology analysis: (*i*) we develop a semi-supervised suspicion prediction scheme to predict the probability of a node being relevant to targets of compromise (*i.e.*, the denied nodes), including a novel node ranking approach as an efficient global propagation scheme to incorporate the topology information, and a scalable graph learning scheme to separate the global propagation from the training of the local prediction model, and (*ii*) based on the suspicion prediction results, efficient pruning strategies are proposed to further remove highly suspicious nodes from the crawled graph and analyze the underlying indicator of compromise. Experimental results show that DoITrust achieves 90% accuracy using less than 1% labeled nodes for the suspicion prediction, and its learning capability outperforms existing node-based and structure-based approaches. We also demonstrate that DoITrust is portable and practical. We manually review the detected compromised nodes, finding that at least 94.55% of them have suspicious content, and investigate the underlying indicator of on-chain compromise.

I. Introduction

Allow-list expansion based on transitivity of trust. Malicious website detection has been a longstanding focus for cybersecurity vendors. The aim is to stem the tide of attacks such as phishing and malware. The prevailing practice is to add the domain names of malicious websites to deny-lists, blocking access to such sites as and when they are found [1], [2]. The diametrically opposed approach is the restrictive allow-list: only trusted domains (such as "[.]gov") are added to a pool of authorized sites [3], [4]. However, block/allow lists are generally incomplete, passive, and their updates are time and labor-intensive [5]. The inherent research question is ***"how to develop a dynamic mechanism for expanding and monitoring the allow/block lists"***. To that end, we begin with a graph expansion algorithm initialized with 56,805 seed domains (initial allow-list). We visit the *default main page* for each seed domain to gather outgoing links from their HTML. These links are then distilled to their basic domain names (*e.g.*, `example[.]org` for `example[.]org/login`) and the crawling is repeated recursively. Consequently, we collected an organically grown graph comprised of 1.7M nodes and 10M edges, which acts as an expanded allow-list based on the relationships between each domain's default main web page. The motivation of the expansion is based on the hyperlink relationships as *transitivity of trust*, allowing the trust between two parties to be extended further. Namely, if domain A trusts B, and B trusts C, then A would trust C as well [6], [7]. The benefits of the expansion graph are twofold: (*i*) providing the resource to build an extensive and *dynamic* allow-list; (*ii*) providing extra transitivity of trust information to further *monitor* the trust of the allow-list.

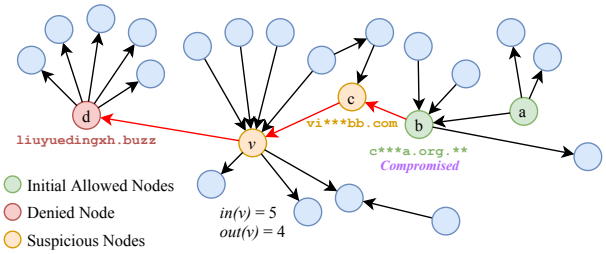


Fig. 1: An example of **On-chain Compromise**.

Research gap of domain monitoring: On-chain Compromise. After obtaining the expansion graph, the successive procedure is to monitor the expanded nodes from the initial list. All nodes in the expansion graph are supposed to be benign domains as they are all accessible from trusted seed domains in the initial list. Surprisingly, we find malicious nodes in the graph, indicating the risk that compromised and malicious domains may inadvertently be included.

Existing studies on domain monitoring mostly focus on identifying malicious patterns through features extracted from a pile of malicious and benign webpages, rather than exploring how benign webpages are compromised and eventually linked to malicious websites. Obviously, it is risky to include compromised nodes in an allow-list. For example, malicious domains may steal the reputation of a ".org" domain through a chain linked by compromised nodes, especially when the

Network and Distributed System Security (NDSS) Symposium 2023
27 February - 3 March 2023, San Diego, CA, USA
ISBN 1-891562-83-5
https://dx.doi.org/10.14722/ndss.2023.24322
www.ndss-symposium.org

embedded hidden hyperlinks in reputable websites are capable of bypassing security checks. Therefore, the goal of monitoring in this work is ***"to identify and filter the stealthy suspicious nodes (including compromised nodes) along the entire chain from the allowed nodes to the target of compromise"***.

Accordingly, we define such a new type of compromise as ***On-chain Compromise***, where a node in an allow-list could be directly or indirectly connected to nodes in a deny-list through stealthy methods such as nested hidden HTML links, resulting in potential reputation stealing, malware injection, or data theft threats. The ***property*** that distinguishes On-chain Compromise from existing malicious or compromise attacks is its two-fold stealthiness for evasion of detection: (*i*) Content stealthiness. Unlike malicious webpages that have specific patterns, the content of compromised webpages could be utterly "clean", *e.g.*, may only contain hyperlinks that are identified as "benign" by third-party security vendors but lay on the path to malicious targets. Such hyperlinks could be invisible and constantly updated, and thus hard to recognize in real-time. Such attacks are always more stealthy and silent than active attacks, making it easier for them to remain undetected for a longer period of time [8]. (*ii*) Topology (intent) stealthiness. We find the On-chain Compromises are always conducted in a *diluent* manner to evade detection, *i.e.*, the compromised nodes are buried deeply along the paths away from the malicious targets (nodes in deny-list, as intent of the compromise). Particularly, there are many intermediate suspicious nodes existing between the compromised nodes and denied nodes, which are commonly identified as "clean" by third-party security vendors. Generally, they are intended to attack the ranking of a web page in search engine results by building a hyperlink topology (demonstrated in § VI). A real-world example is presented in Figure 1, where nodes a and b are initially allowed nodes; nodes c and v are intermediate suspicious nodes that are not detected as malicious domains by existing detection tools; a node d is a malicious domain in deny-list. We found that, the node b (`c***s.org.**`, an anti-child abuse non-profit website) is compromised by nodes c and v, and is finally connected with the denied node d.

Technical challenges. Unlike one-step malicious/benign URL prediction, monitoring the expansion graph involves the identification of on-chain compromise in stealthy and deep connections. Due to the content and topology stealthiness of the on-chain compromise, as well as the lack of ground truth, none of the existing web malicious analysis methods can address this new problem. We summarize the technical challenges of detecting on-chain compromise as follows:

C1: ***Sparse labels and limited supervision information.*** Due to content stealthiness, there is no supervision information available for extracting malicious patterns. Additionally, only a small percentage of nodes could be detected as denied nodes, resulting in inaccurate predictions as a result of insufficient supervision information.

C2: ***Efficacy.*** Existing domain examination methods are either based on individual node features or only focus on the topology (*e.g.*, applying PageRank [9] and TrustRank [10] as a semi-automatic technique), resulting in a high false positive rate and low accuracy (especially when compromised nodes are considered). Additionally, lightweight feature extraction is necessary for fast inference.

C3: ***Scalability and portability.*** Graph Neural Networks (GNNs) excel at a variety of network mining tasks by considering both individual and global structural information [11]. However, existing graph learning approaches are facing constraints of performance and scalability in large graphs. Additionally, web-compromising behaviors have a typical long chain of influence and are stealthy in nature, which is beyond the 2-hops scope of ordinary graph learning schemes.

Our integrated scheme. As on-chain compromises cannot be detected directly due to their stealthiness, a ***two-step process*** is proposed: (*i*) measuring the suspicion of each domain node in the expanded graph to be denied nodes; and (*ii*) post-processing for the recognition and analysis of compromised nodes. To conquer the aforementioned challenges, we present an integrated learning scheme, DOITRUST, leveraging the strength of both machine learning (based on individual node features) and web topology analysis (label propagation in the global view) to uniquely solve the challenge of the on-chain compromised websites. Particularly, DOITRUST is a fast and scalable graph neural network-based semi-supervised node classification model for domain compromise threat measurement, consisting of two components: (*i*) *Suspicion Prediction* to evaluate the learning capability to detect compromise relevant behaviors, and (*ii*) *Compromise Pruning and Warning* as post-processing to confirm the correlation between highly suspicious nodes and on-chain compromised nodes, and to analyze the underlying indicator of compromise.

To make our scheme more practical and flexible, the crux of it is to customize suspicion (denied nodes). Malicious URLs/domain nodes that could be detected by third-party security vendors represent only one type and a small fraction of denied nodes (178 malicious nodes are identified out of 1.7M nodes). Other denied nodes should be customized according to specific scenarios. For example, a deny-list for government organizations should also contain the domains that lead to gambling and pornography web pages. Such customized denied nodes are always sparsely labeled and evade the detection of many threat intelligence vendors. Therefore, our graph learning based semi-supervised node classification utilizes the global structure information to propagate label information and globally determine the probability that a domain node is a denied one. The prediction confidence is used as a suspicion score. Finally, an enhanced allow-list could be acquired by *pruning high suspicion nodes from the graph* as per the predicted suspicion, while the compromised nodes are the original "benign" nodes that have high suspicious scores.

Contributions. The main contributions are as follows:

- We construct a Website Domain Graph baseline dataset, collating 1.7 million Internet domains in the wild, demonstrating the pipeline of allow-list expansion based on transitivity of trust. Further, a new type of domain compromise, On-chain Compromise, is identified as the research gap in monitoring nodes of the expansion graph. We also reveal two-fold stealthiness as its distinguishable properties.
- We propose a two-step integrated scheme to leverage both individual node features and topology analysis to solve the lack of labels and stealthy contents, addressing C1.
- Considering compromising web behaviors as a social engineering problem, we propose a new node ranking approach,

IncredulityRank, and its top-k approximation algorithm, as an efficient global propagation scheme to incorporate the topology information. We demonstrate that suspicion prediction is efficient even with simple individual features extracted from URL and HTML, addressing C2.
- To address C3, we implement a graph learning scheme to separate the global propagation from the training of the local prediction model, solving the scalability issue for semi-supervised node classification. We further propose four strategies of portability for the proposed scheme.
- We extensively evaluate DOITRUST's performances of denied nodes classification as suspicion prediction. DOITRUST obtains approximately 90% prediction accuracy with less than 1% labeled nodes, outperforming the state of the art. On a graph with 1.7M nodes, the training and inference time of our model only takes less than 2 minutes on a single machine. Furthermore, we manually review the detected compromised nodes, report that 94.55% detected compromised nodes have suspicious information found in the HTML, and investigate the underlying indicator of compromise.

To the best of our knowledge, DOITRUST is the *first* work tailored to identify on-chain compromised nodes that are stealthy and hidden deeply, while providing efficiency, scalability, and portability in the scenarios of limited information.

Responsible disclosure and ethical considerations. All results have been discussed at length and then reported to relevant stakeholders. DOITRUST is in the process of industry deployment, pending for public use. The crawled dataset purports to study Internet measurement in good faith, and is secured privately with access granted only to the authors' affiliations. The dataset does not include any personally identifiable information, with all examples in this work anonymized so as to prevent linking sensitive activities to specific users.

II. PRELIMINARIES

In this section, we introduce allow-list expansion graph as preliminary knowledge, along with the problem statement and our design goals.

A. Allow-list Expansion Graph

To achieve a larger and more realistic expansion, we started with 56,805 domain names from a self-compiled list of trusted domains. These domains were gathered from a range of credible public sources, such as *[Redacted]* Stock Exchange, the *[Redacted]* Charity Registry, non-profit commissions, and various government directories. These were then seeded into a custom web crawler that *only* followed links listed on the landing page of a domain name. Further, it *only* visits the domain name portion of any URLs found on the main page. For example, for `https://www.nytimes.com/section/world`, the crawler will extract and visit its default main page `nytimes.com`. The Breadth-First Search (BFS) strategy is used to extend the graph until the 6-depth level is reached. The crawler recursively expands the graph of trusted domains to 1.7 million nodes, which are interconnected via 10 million edges. By way of trust transitivity, all 1.7M nodes should be considered as "clean". However, security vendors, such as GSB [12], discovered 178 malicious domains within the graph. This means some domains in the initial allow-list and/or intermediate nodes are compromised. The spectra of likely compromised sites cast doubt on the trustworthiness of large-scale extended allow-lists and even the seed list. Namely, it is necessary to identify and filter the highly suspicious nodes in the extended graph before use.

B. Problem Statement and Design Goals

We divide the compromise detection into *Suspicion Prediction* and *Compromise Pruning and Warning*. Given the landscape of website domain names, we model the web as a graph $G = (V, E)$, where V is a set of vertices including both labeled and unlabeled domain names (represented by the default main page of the domain), and E is a set of directed hyperlinks (edges) that connect vertices. Practically, for each domain node v, we collect the HTML of its default main page, Web_v, which contains multiple hyperlinks $\{url_1, url_2, \ldots, url_i\}$ to other web pages. Figure 1 presents a graph generated by layer-by-layer crawling. The number of incoming links of a domain v is its in-degree $in(v)$, whereas the number of outgoing links is its out-degree $out(v)$. The representation x for each domain node is a D-dimensional numeric feature vector that is extracted from the composition of the main web page's URL, and the page's HTML contents. We summarize the definitions and notations used throughout the paper in Table I below. We define *suspicion* as the possibility of a domain v being denied node-relevant or not, which could be represented as a numerical value $Z_v \in [0, 1]$ (*e.g.*, prediction confidence). In general, we consider compromising intents as malicious (*e.g.*, malware or phishing venues recognized via third-party security vendors), pornography, and gambling websites (customized compromising intents). Evaluation of domain suspicion is therefore a suspicion prediction, depending on node features and/or graph topology. The goals of this classification can be summarized as follows:

(1) **High performance under resource scarcity.** The classifier must perform well when only a few sparse labels and limited supervision information are available.
(2) **Low false positive rate and high accuracy.** Given the relative scarcity of labels, the false positive rate of the classifier must be low. Additionally, the classifier must achieve high accuracy for both allowed and denied domains.
(3) **High scalability and portability.** The classifier must process a large graph with low latency, *i.e.*, must be trained and kept up with a load of millions of pages to examine in a short time. In addition, the overhead for industry deployment should be not large.

III. DESIGN OF DOITRUST

The DOITRUST consists of the Suspicion Prediction (§ III-A to § III-D) and Compromise Pruning and Warning (§ III-E). Motivated by separated graph learning [13]–[15] (see Appendix D), we propose a graph-based semi-supervised denied node classification approach for suspicion prediction. Local prediction and global propagation are decoupled so as to handle large graphs with few and sparse labels. Concretely, suspicion prediction is composed of *Individual Feature Extraction*, *Local Prediction*, *Personalized Incredulity Ranking*, and *Global Propagation*, as shown in Figure 2.

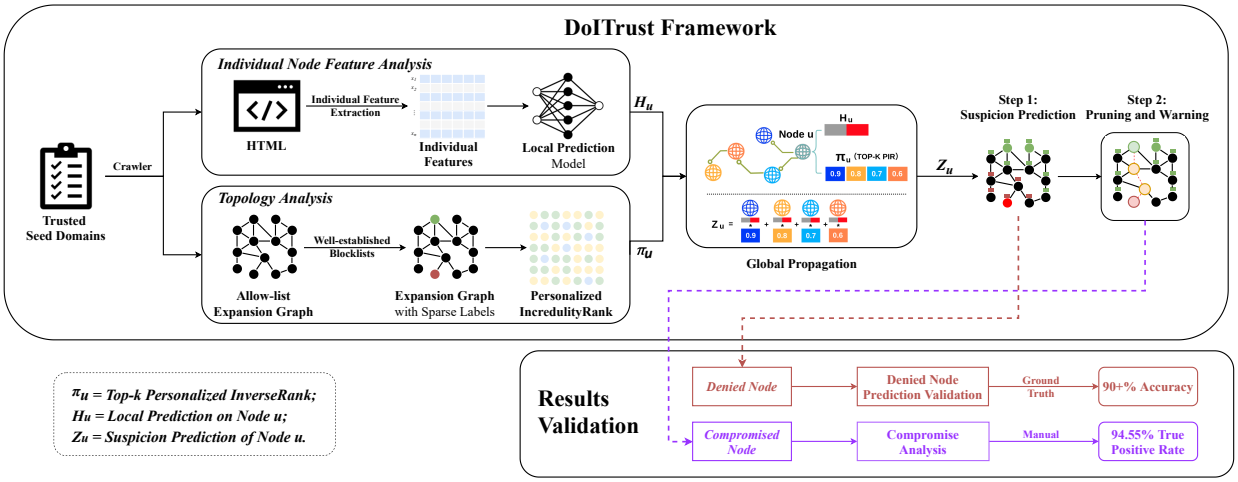


Fig. 2: An overview of DOITRUST.

TABLE I: Summary of terms and notations.

Term	Description
Initial allowed node	A seed node labeled as "allowed".
Denied node	The destination target (intent) of compromise.
Suspicion	The possibility of a domain being denied node-relevant or not.
Intermediate node	A node lies between compromised nodes and denied nodes, commonly identified as "clean" by third-party security vendors.
Compromised node	An original benign node that has a high suspicious score.
Obvious-positive alarms	Highly suspicious nodes with obviously suspicious information found.
Subtle-positive alarms	Highly suspicious nodes without obviously suspicious information found.
Notation	**Description**
$G = (V,\ E)$	Domain name graph, V is the set of domain names, E is a set of hyperlinks that connect domains.
$X,\ x_i$	Node features X consists of D-dimension vector x_i for node v_i.
$in(v_i), out(v_i)$	in-degree and out-degree for a domain node v_i.
f_θ, H_u, Z_u	Local predictor, local embedding and prediction value of node u.
$\Pi, \vec{\pi}^k(u)$	Personalized IncredulityRank matrix, top-k personalized IncredulityRank vector for node u.
α, ρ	Teleport probability, approximation indicator.
γ	Discount factor for trust evaluation.
δ, ϵ, p_f	PIR threshold , estimation error bound and failure probability.
$\hat{\pi}, r_{max}, \tilde{\pi}, r, \pi'$	Estimated PIR, residual threshold, reverse vector, residue vector, random walk estimation vector.

The first step is to apply a flexible and low-cost feature extractor to produce node features relevant to malware, phishing, spamming and sexual information by extracting the content of HTML and URL-based features for each node in G. After individual feature extraction, we produce local predictions via a neural network f_θ for each node, where θ is the trainable parameters. Feature extraction and local prediction are detailed in § III-A. The local prediction outputs a probability vector $H_i = f_\theta(x_i)$ on each node's features independently, allowing for parallelisation and flexibility of local prediction approaches.

During the global propagation, a message passing scheme such as the PageRank (PR) [9] or personalized PageRank (PPR) [13], [15]–[17] can be used to find trusted neighbors from each node in the whole graph vision, instead of in a limited 2-hop vision as existing graph neural networks do (background is provided in the Appendix A and B). The advantage of such a scheme enables infinitely many neighborhood aggregation layers effectively. Inspired by the unique characteristics of web compromising attacks (§ II), we propose Personal IncredulityRank (PIR) as a new variant of PPR, specified for the suspicion evaluation on the website domain graph. The difference between PIR and PPR is that PIR also accounts for the unsuspicion of a website domain node's children inversely, thereby penalizing the node if it recommends, by way of a hyperlink, an untrustworthy node (detailed in § III-B). Further, we apply top-k approximation mechanism to achieve an efficient PIR approximation for a large website domain graph, as detailed in § III-C. The PIR approximation may be viewed as a pre-processing that operates prior to or in parallel with the local predictor training.

The final prediction for node s is aggregated with its local prediction and the predictions propagated from its top-k important neighbors $N^k(s)$ associated with importance score $\pi(s, v_j)$ via the personalized IncredulityRank approximation scheme, denoted by

$$Z_s = \text{softmax}\left(\sum_{j \in N^k(s)} \pi(s, v_j) \cdot H_j\right), \tag{1}$$

where H_j is the local prediction probability vector for node j. For training, the gradient flows along with the PIR-based propagation scheme during backpropagation, which significantly increases the model's accuracy due to implicitly considering infinitely many neighborhood aggregation layers.

We evaluate three strategies to apply the PIR-based global propagation: (*i*) Training Incorporation (TI): The propagation is only implemented when training the local neural network for prediction. During inference, only the local f_θ is used to predict the node labels; (*ii*) Inference Incorporation (II): the local neural network f_θ is trained without considering information from any neighbors. During inference, the prediction of a given node is aggregated by its neighbors' prediction probability vector derived from the pre-trained f_θ networks with fixed weights, according to its PIR vector, and (*iii*) End-to-end Incorporation (EI): the propagation is implemented during both training and inference phases. In addition, for some use cases, the propagation may not be available, and therefore we also evaluate the strategy without propagation applied in neither training nor inference (Without Incorporation (WI)). After evaluation of these strategies, we exploit two possible implementations: static inference and real-time inference (even without global propagation), detailed in § V-C. We apply the

End-to-end Incorporation as the default strategy in our proposed model, *i.e.*, the PIR-based propagation during training and inference.

The output of DOITRUST is the prediction of the suspicion value of each node, which indicates the compromise intent. Extensive allow/deny-lists could be obtained via pruning nodes of the graph according to the threshold of the suspicion value. Additionally, we provide two pruning strategies to further achieve a clean extended allow-list with compromised nodes pruned as detailed in § III-E.

A. Individual Feature Extraction and Prediction

The first step is to extract individual features that are relevant to malware, phishing, spamming, and sexual information. For lightweight feature engineering, we adopt simple statistical and lexical features that are automatically and directly extracted from URL and HTML. In order to assess the performance of our integrated scheme with the worst case of training information and model complexity, we chose simple local features and local predictors.

1) Statistical features: The first type of individual feature is the statistical features in terms of the HTML tags and URLs. We select some easily accessed ones from the commonly used features in existing studies [18]–[21]. Considering dynamic web page development is the main source of injecting malicious code into web pages, statistical properties in the web page content are used to detect web pages that are malicious, such as the number of HTML tags, iframes, scripts tags, and href tags. These tags could be the sources to inject external code or for phishing the website by redirecting to malicious servers. In addition, we also consider the statistical URL features for each domain, such as the length of a URL, the number of special characters in URLs (*e.g.*, dots '.', hyphens '-', forward slashes '/', underscores '_', and equal signs '='), and whether an IP address or re-direction is present in a URL. We provide the list of statistical features in detail in Appendix E.

2) Lexical features: We also consider the content of the HTML extracted from the domain's default main page. The key point is to use the vectorization of textual content. Considering that the textual content of a web page is often too large for deep neural network-based embedding, we adopt a flexible representation mechanism inspired by the idea of bag-of-words. We vectorize HTML contents with bag-of-maliciousness (BoM), a numerical representation approach we proposed for domain suspicion evaluation. Specifically, we first build a token dictionary that consists of denied words or symbols summarized from 10K denied web pages (collected from blacklists and customized category websites such as pornography and gambling). Then, 2400 most frequent tokens are selected to build the token dictionary (demonstrated in Figure 15 in the Appendix). We then convert the set of textual content of each HTML to a frequency distribution matrix, where each HTML document is a row, each token from the denied dictionary is a column, and the corresponding (row, column) value is the frequency of occurrence of each token in that HTML document. Namely, each denied token is a column name, with the corresponding value being the frequency of that word in the document. We treat each token individually, regardless of the order of words.

3) Local prediction: With the embedding node feature vectors, the local node prediction can be simplified as a binary classification task. A fully connected network (FCN) is applied to classify the node domain, composed of two layers with 32 hidden sizes and ReLU activation functions. Given the labeled domain set, the parameters of local prediction FCN is trained with the Adam optimizer on binary cross-entropy loss.

B. Incredulity Rank

Given PageRank does not consider any knowledge about the quality of a domain, nor explicitly punish badness, in practice, it is possible for a skilled adversary to manipulate PageRank results through link exchange to achieve a falsely high PageRank score. Therefore, as the interplay of multiple factors relevant to the respectability and credibility of a website, domain suspicion is hard to evaluate by PageRank. TrustRank provides a biased PageRank, based on the assumption that benign sites rarely have links to spam sites. However, TrustRank is also easily manipulated via creating outbound links to, or secure back-links from, high-ranking and reputable websites. In addition, totally "benign" website domain nodes are always hard to define and find, while totally "malignant" ones are easier to recognize.

Compromising behaviors on the web, such as phishing or spamming, are more prioritized as a social engineering problem rather than a technical one. Like in society, distrust is propagated backward on the web. Phishing, spamming, or compromising behaviors aim to provide untrustworthy recommendations for end-users. The recognized untrustworthy recommendation provides an indicator to review the suspicion of the recommender. The untrustworthy recommenders are those who strongly support an untrustworthy recommendation. Therefore, given the identified untrustworthy recommendation, it is feasible to find a recommender who strongly supports the recommendation after a few iterations of distrust backward propagation. Additionally, it is more feasible to recognize a web page that is suspicious than completely trusted.

In this section, we investigate and develop the IncredulityRank, as an algorithmic way of evaluating compromising behaviors in hyperlink networks. Namely, we seek a quantitative approach to automatically evaluate the suspicion of web neighborhoods for a given node. We first perform a breadth-first-search over the incoming links from each denied node in the crawled domain graph, associated with the suspicion score 1. Then, the suspicion value is inversely split among in-neighbors that link to a given suspicion seed equally, layer by layer. The intuitive heuristic involved here is that a page is likely to be compromised node, if it has out-links to other denied domains. The shorter the distance away from denied pages, the higher confidence a page is a compromised node.

A discount function is further applied to split and dampen the suspicion score based on the steps away from the denied node. The IncredulityRank proposes an inverse approach to attenuate distrust values, starting from the denied nodes, then splitting and dampening distrust values upwards through the tree. The algorithm determines the suspicion score of a given node according to the number of steps away from a denied node and the number of denied domains it leads to. Only incoming links to denied nodes are considered, as the algorithm

aims to trace paths from the given node to each denied node. Outgoing links from suspicious nodes may direct to legitimate domains, so the algorithm will avoid penalizing domains with un-reciprocated incoming links from denied domains. Formally, if a page node v has a suspicion score of c_v and it has $in(v)$ incoming pages, each of the incoming pages will receive a score fraction $c_v/in(v)$ from node v. The IncredulityRank score of a page is then the sum of the score fractions received through its out-links and multiplied by a discount factor γ. We then normalize summed scores to the standard range of $[0, 1]$ among all nodes. Intuitively, the higher the suspicion score a page accumulates from other pages, the more likely it is compromised. The IncredulityRank score $IR(v)$ for a node v is then defined as one minus its accumulated distrust score. It is possible to determine a suspicion score for each node with respect to the entire graph according to the $IR(v)$.

$$IR(v) = \gamma \times \sum_{i \in N_{outlinks}(v)} IR(c_i)^{distrust} \tag{2}$$

C. Approximation of Personalized IncredulityRank

Similar to personalized PageRank (PPR), the personalized IncredulityRank score $\pi(s, t)$, reveals the relative (dis)trust score of a target node t with respect to a source node s in a graph. The challenges when applying existing PPR in our crawled graph are: sparse initialization values and low efficiency on a large-scale graph. To address these challenges, we propose the Personalized IncredulityRank (PIR), which refines the existing PPR approximation approaches [16] with adaptive initialization and termination thresholds introduced.

Initialization of PIR values. To initialize trust values, the existing approaches equally distribute the initial trust values among trusted seed nodes, or distrust values among denied nodes, *e.g.*, 1 divided by the number of trusted or denied nodes. However, this is not applicable in our scenario, where the initial seeds or denied nodes are rare and sparse, which may lead to hard-to-converge issues and low quality of trustworthy evaluation performance. Therefore, we first find all suspicious paths (shortest only) identified in the graph from seeds to denied nodes in the crawled domain graph, followed by initializing the PIR values along the suspicious paths according to Equation 2. Based on the adaptive initialization, we expand the non-zero initialized value from 178 (0.01% of all nodes) in existing methods to 283,117 (16% of all nodes) in our crawled domain graph using the extended from the same 178 denied nodes. This fraction could increase when considering more denied nodes. Our initialization strategy could enhance the scale of initialized values with reasonable values towards fast convergence and high quality of suspicion evaluation.

Approximation of personalized IncredulityRank with top-k nodes. For an inversed web graph $G = (V, E)$, the propagation scheme requires the Personalized IncredulityRank matrix Π, in which the u^{th} row is the PIR vector of the given node u, denoted by $\vec{\pi}(u) = \{\pi(u, q_1), \ldots, \pi(u, q_{|V|})\}$. Each entry $\pi(u, q_i)$ measures the importance of a target node q from the perspective of source u. The global IncredulityRank of a vertex q can be viewed as the sum of the contributions from all other vertices, *i.e.*, the sum of the q^{th} column of the matrix Π. The q^{th} column of the matrix Π is also called the contribution vector of q. Calculation of personalized PageRank or PIR values on an entire graph consumes massive computing and storage resources. Thus, approximation solutions are commonly adopted. Furthermore, for strongly connected graphs, the PIR matrix is non-zero for all nodes, resulting in burdensome computing and storage needs. Therefore, a more efficient approximation is via selecting top-k elements of the PIR matrix and truncating the rest to zero.

Two common personalized PageRank estimation approaches are Forward Push [22] and Monte-Carlo (MC) [17] (backgrounds are in the Appendix B). Forward Push is too expensive to obtain an exact estimation, which can be stopped earlier, but the tail term cannot guarantee an approximate quality. Monte-Carlo (MC) can guarantee that good approximate quality is obtained, but the efficiency is low. Inspired by FORA [16], [23], BiPPR [24], and Top-PPR [25], we implement an adaptive top-k PIR estimation algorithm. Formally, given a source node s, desired estimation amount k, a PIR threshold δ, an estimation error bound ϵ, and a failure probability p_f, the approximate top-k PIR query outputs a set of k nodes $\{v_1, \ldots, v_i, \ldots, v_k\}$ for a node s, with their estimated PIR scores $\hat{\pi}(s, v_i)$. This top-k PIR estimation combines four stages:

- Step I **Forward Push for coarse estimation.** Forward Push from a source node s is conducted to obtain coarse estimations for the first stage of one iteration using adaptive initialization and with early termination.
- Step II **Random walks for refining estimation.** Random walks are used to refine the accuracy of approximation for the nodes with non-zero residues for the final stage of one iteration.
- Step III **Top-k PIR values selection.** Top-k PIR selection is applied to terminate early the iterations of estimation when satisfying the accuracy criteria of estimation.
- Step IV **Adaptive threshold for efficiency and scalability.** To further improve the efficiency and scalability of PIR estimation, an adaptive residue threshold strategy will be applied to maintain an individual residue threshold for each node $v_i \in V$, denoted by $r_{max}^{v_i}$, instead of the common threshold for all nodes.

Finally, the estimated PIR values are used for the training and inference procedures described in Equation 1. The details of the top-k IncredulityRank approximation are in the Appendix C.

D. Incorporation Strategies of Global Propagation and Real-time Inference

Real-time and efficient inference is important for practical applications, especially for dynamic scenarios. Ideally, the model is trained once while providing continuous inference after implementation. Therefore, we evaluate the impact of PIR-based propagation on the performance of the node classification according to four strategies in Table II. We compare the accuracy of the classification using Training Incorporation (TI), Inference Incorporation (II), and End-to-end Incorporation (EI) and Without Incorporation (WI), to investigate the significance of the PIR-based propagation. For WI, the structural information of nodes is assumed to be unavailable. Thus DOITRUST can only classify a node based on its individual features, which is the worst case for DOITRUST application since

TABLE II: PIR-based propagation strategies.

	Training		Inference	
	LP	GP	LP	GP
End-to-end Incorporation (EI)	●	●	●	●
Training Incorporation (TI)	●	●	●	○
Inference Incorporation (II)	●	○	●	●
Without Incorporation (WI)	●	○	●	○

LP: Local propagation; **GP**: Global propagation.
●: the related propagation is involved in the strategy;
○: the related propagation is not involved in the strategy.

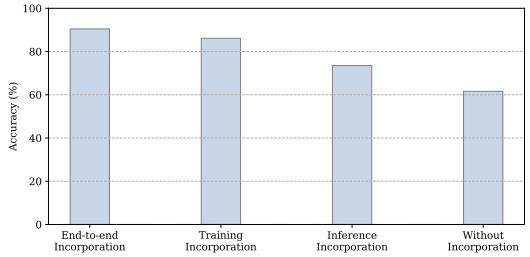


Fig. 3: Performance of Training Incorporation (TI), Inference Incorporation (II), and End-to-end incorporation (EI) and Without Incorporation (WI).

structural information is not difficult to obtain. The results are demonstrated in § V-C and Figure 3, which inspires real-time inference pipeline.

Accordingly, we could provide two use cases, as shown in Figure 4. ***(i)* One-shot static inference.** Given the website domain graph G associated with feature vectors as the input, this case aims to learn the semi-supervised node classification model using only a small fraction of nodes of G to be labeled. After applying EI training strategy, we obtain local classifiers, suspicious prediction of nodes in G, and the pruned graph G' from G. Here G' could be used as an extensive allow-list and deny-list. ***(ii)* Real-time Inference.** Given a domain of a website or a URL x, the pre-trained local classifier, graph G, and node predictions, this case aims to predict the label for x in a real-time manner. We demonstrate that the End-to-end Incorporation (EI) strategy has good accuracy (more than 90%). Therefore, it is possible to provide accurate predictions for $x \in G$ in real-time. If $x \notin G$, then we investigate all hyperlinks H_x in the HTML content of x. If $\exists x' \in H_x$ & $x' \in G$, we figure out the PIR for x, and perform the global propagation using the top-k important neighbors of x to get the final prediction of x on the aggregated feature embeddings via the EI or Inference Incorporation (II) strategy. As we demonstrate the Training Incorporation (TI) strategy could also have reasonable accuracy (better than WI), it is possible to make a prediction for node x with no connection to G. Namely, if no candidate in H_x belongs to G, then we give the prediction using the pre-trained local classifier using the feature of x only. When this is a data drift (a new disconnected graph appears), we also demonstrate it is feasible to re-train a new model with less overhead in § V-D.

E. Compromise Pruning

Based on the output label and compromise confidence value (suspicion prediction), the extensive allow/deny lists

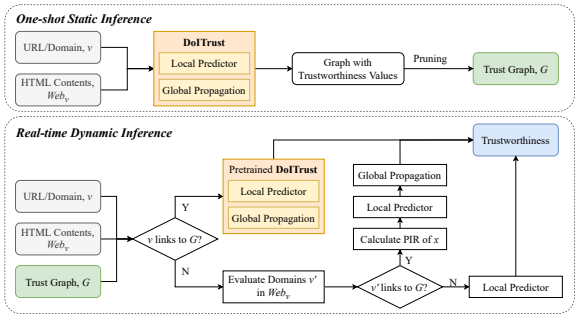


Fig. 4: One-shot static inference and real-time inference.

could be obtained via pruning nodes of the graph. In this section, we provide two pruning strategies to achieve almost 100% clean results.

1) Shortest path-based pruning: The first strategy is to identify all the shortest paths from the benign nodes to denied nodes, and remove all the nodes along this path to cut out parts of the graph that were likely compromised. The aim is to isolate clusters of suspicious and malignant domains by removing all paths that lead to these clusters. Specifically, Dijkstra's shortest path algorithm [26] is applied, followed by removing every node in the found path, excluding the benign node and the denied node.

2) Flow-based pruning: The second pruning strategy to find the most compromised domains in a path is to compute the flow of domains, which is defined as the number of shortest paths from the benign nodes to every denied domain passing through that given domain. A higher flow means that more paths to denied domains pass through that node in the graph, and thus represents the domain's importance in reaching malignant domains from the benign. We find the deepest node with the highest flow for each iteration and remove them from the graph. When no further nodes remain with the flow, unreachable nodes in the graph can be considered as part of a suspicious cluster. Specifically, this strategy identifies the most important nodes in the paths to denied nodes and removes these from the graph, which will reduce the reachability of suspicious sites. We prune the deepest node in a path constructed from the highest flow nodes, which often turns out to be a compromised domain with multiple hidden links to suspicious sites. The algorithm can identify and prune these nodes that lead to the most benign domains while preserving the legitimate domains that are higher on the same path. Through pruning, compromised nodes are pruned and a clean extended allow-list is obtained.

IV. Experiment Setups

A. Data and Settings

Ground truth labeling and scope of investigated website. We labeled allowed and denied domain names to validate the denied node classification for suspicion evaluation, according to the following ground truth sources: (*i*) For denied nodes, labeling resources are derived from malicious domains detected by GSB (178 labeled), authoritative blacklists, and customized improper domains according to domain category list, including PhishTank [27], MalwareURL [28], malwaredomains.com [29], Zeustracker [30], malwaredomainlist.com [31], and UT1 blacklists [32]. 53,168 denied domain

nodes (approximately 3%) were found in the graph, consisting of malicious domains and customized domains (*e.g.*,gambling and pornography); (*ii*) For allowed domains labeling: to obtain benign domains, we combine the collected domains that appear in Alexa 1M Global Sites 2020 [33] together with the initial trusted seed domains (56,805) as potentially allowed domains. We further use third-party tools and manual analysis to drop the malicious and improper domains. Finally, we obtained 54,112 allowed domains. Note we exclude popular interactive websites, such as Facebook and Reddit, as they may contain numerous user-submitted links that can lead to the noisy node. Due to the limitations on crawling time and storage capacity, in the current stage, we focus our investigation on top-level domains and the company's default homepage. We plan to conduct a measurement research with finer granularity, applying our approach on a larger scale in the future to detect more compromised domains.

Small and large datasets. To validate the performance of DOITRUST on different scales, we set up two datasets: (*i*) a Small dataset of 10,000 domain nodes (5K allowed and 5K denied nodes) randomly sampled from the ground truth nodes in graph G, and (*ii*) a Large dataset containing 1M domain nodes (including 50K allowed nodes and 50K denied nodes) randomly sampled from the ground truth nodes. We use 80% of labeled nodes as a training set and the remaining 20% labeled nodes as a testing set. The rest of the indeterminable nodes in the graph serve as the network resource but are not involved in the training and validation due to lack of ground truth for these two size datasets.

B. Validation Settings

The validation of DOITRUST is two-fold: denied node prediction validation with full ground truth (automatically quantitative evaluation), and compromise analysis without ground truth (post-processing and analysis).

1) Denied node prediction validation: To evaluate the capability of the semi-supervised denied nodes classification, we conduct validation on both 2K and 20K validation sets. The prediction accuracy and false positive rates are used as the evaluation metrics. Comparisons with learning benchmark baselines are conducted for this validation, in terms of learning capability (§ V-A). Additionally, we also compare DOITRUST with structural ranking baselines (§ V-B) in terms of Precision, Recall, F1, and Accuracy. Further, we evaluate the portability of DOITRUST under real-time inference settings in § V-C and efficiency in real-world deployment in § V-D.

2) Compromise analysis as post-preprocessing: After suspicion prediction, the next step is to confirm the correlation between highly suspicious nodes and on-chain compromised nodes (highly suspicious but with benign labels), and to analyze the underlying indicators of compromising. Here, high suspicious scores are considered as for warning on-chain compromise. We manually validate 110 nodes (randomly sample 10%) from the detected on-chain compromised nodes. Here, we report the accuracy of *obvious-positive alarms* (highly suspicious nodes with obviously suspicious information found) and *subtle-positive alarms* (highly suspicious nodes without obviously suspicious information found), detailed in § V-E. The manual check considers following criteria: *(i)* a compromised node is on the path from allow nodes to denied nodes; *(ii)* a compromised node is with benign label but predicted with high suspicion score; *(iii)* the detected compromised node is obvious-positive alarm if suspicious information found in the HTML; otherwise, subtle-positive alarm.

C. Benchmark Baselines

To demonstrate the performance of DOITRUST, we adopt four types of baselines for comparison.

- *Individual Machine Learning Baselines (IML)*: supervised machine learning approaches (such as Support Vector Machine and multi-layer perceptron (MLP)) on the individual node feature only.
- *Graph Neural Network Baselines (GNN)*: we compare the DOITRUST to state-of-the-art graph neural network approaches, *i.e.*, graph convolutional network (GCN) [34] (including vanilla version without early stopping and hyperparameter optimization, V-GCN) and graph attention networks (GAT) [35]. We set our method using the same number of parameters with GCNs, *i.e.*, two layers with 64 hidden units.
- *Scale Graph Neural Network Baselines*: we also compare the DOITRUST to state-of-the-art scale GNNs, *i.e.*, Cluster-GCN [36], APPNP [15], and PPRGO [13].
- *Structural Processing Only Baselines (SPO)*: we further compare our results with structural processing only baselines, such as PageRank [9], TrustRank [10], and two application strategies of TrustRank, *i.e.*, Step Function and Trust Discount Scoring. For domain suspicion evaluation using SPOs, the individual node features are not considered.

V. EVALUATION RESULTS

According to the two-fold validation setting in § IV-B, we evaluate the performance of our DOITRUST model for the semi-supervised denied node classification on the domain graph and illustrate its learning capability (§ V-A), and comparison with structural ranking baselines (§ V-B). We also show the scalability (§ V-C) and feasibility (§ V-D) of industry deployment and summarize the recommendation for stakeholders. Further, we conduct the validation for the compromised nodes in § V-E and indicate the underlying reasons of on-chain compromises in § V-F.

A. Learning Capability Evaluation for Suspicion Prediction

To the best of our knowledge, no existing tools can detect compromised domains, but we can still evaluate the learning capability of the proposed approach and compare DOITRUST with benchmarks listed in § IV-C, with respect to the accuracy of classification based on the initial ground truth. Figure 5 displays the node classification accuracy of each model on small and large datasets using the default settings. We set the suspicion threshold as 0.5 to classify allowed and denied nodes. Since suspicion prediction is a binary classification, it is natural to select 0.5 as the threshold to determine whether a node is suspicious. The threshold also holds true to the compromise analysis and pruning. For other implementations, this threshold could be elaborated to reduce the subtle-positive alert and false positive alert. On the Small dataset, our model achieves 92.50% accuracy (93.2% precision, 91.91% recall and 92.55% F1), outperforming the supervised machine learning

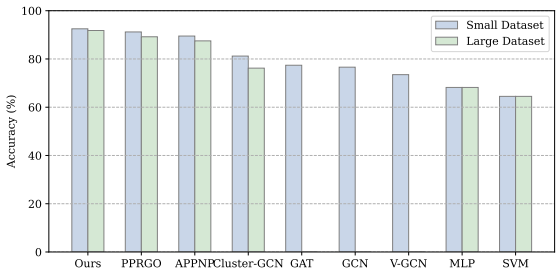


Fig. 5: Overall accuracy of of different models on two size of datasets.

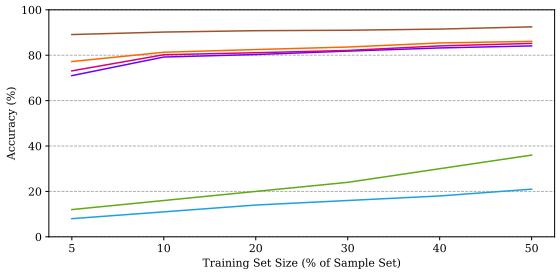


Fig. 6: Accuracy for different training set sizes on Small dataset.

approaches (68.20% accuracy for MLP and 64.50% accuracy for SVM), state-of-the-art graph neural networks (with accuracy ranging from 73.50% to 81.20%), and scale graph neural network baseline models (81.20%-91.20% accuracy). DoITrust exceeds the other baseline models on the Large dataset as well (with 91.8% accuracy, 93.6% precision, 90.35% recall and 91.94% F1). Note that the commonly adopted GNNs (*i.e.*, GAT, GCN, and V-GCN) are out of memory when being applied to the Large dataset. The accurate performance indicates that the DoITrust's learning capability outperforms other approaches. We also report the DoITrust's accuracy using K-Fold Cross-validation ($K = 2, 5, 10$). For both Small and Large settings, the accuracy of our approach increases from fold 2 to fold 10 cross-validations (91.45%, 92.36%, and 92.82% for the Small setting, and 90.26%, 91.25%, and 92.16% for Large).

As the well-labeled nodes are limited to obtain in real-world applications, we further evaluate the performance of our model with various labeling rates, *i.e.*, the proportion of labeled training samples in the dataset. We particularly mimic such a lack of labeled nodes by involving fewer training samples in the training phase. Figure 6 reports the accuracy of DoITrust on different training configurations (varying the percentage of training samples used during training from 5% to 50%), in comparison with SVM, MLP, V-GCN, GCN, and GAT. Note that we only perform the comparison on a Small dataset as GCN and GAT cannot work well on the Large dataset. The experimental results indicate that our DoITrust outperforms all GNN-based approaches, especially in the more sparsely labeled scenarios (*e.g.*, using 5% and 10% training samples during training). The reason is that the PIR-based global propagation benefits from obtaining important information from remote neighbors, instead of the 2-hop neighbors for most of the existing GNNs. Similarly, without global propagation, the performance of individual machine learning baselines drops down significantly when fewer training samples are provided.

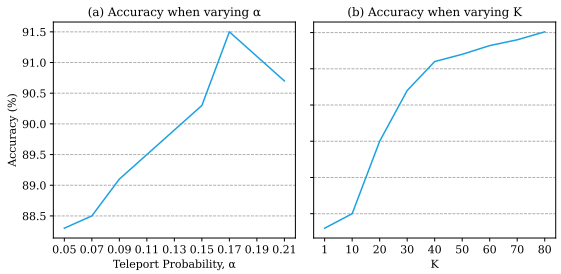


Fig. 7: Accuracy when varying teleport probability α and k.

TABLE III: Trust evaluation using structural ranking only.

Method	Threshold	Precision	Recall	F1-score	Accuracy
PageRank	0.1	53.06%	66.67%	59.09%	54.43%
	0.2	50.41%	78.21%	61.31%	51.27%
	0.3	50.40%	80.77%	62.07%	51.27%
	0.4	50.71%	91.03%	65.14%	51.90%
	1.0	49.67%	97.44%	65.80%	50.00%
TrustRank	0.001	52.58%	65.38%	58.29%	53.80%
	0.005	49.67%	96.15%	65.50%	50.00%
	0.100	49.67%	97.44%	65.80%	50.00%
	0.200	49.68%	98.72%	66.09%	50.00%
	0.300	49.49%	99.23%	66.04%	49.62%
	0.400	49.46%	99.36%	66.04%	49.56%
	1.000	49.40%	99.62%	66.04%	49.43%
Step	3	52.16%	97.89%	68.06%	54.05%
	4	53.20%	97.44%	68.82%	55.85%
	5	53.20%	88.38%	66.41%	55.31%
	6	51.09%	67.76%	58.26%	51.45%
	7	47.55%	36.26%	41.14%	48.13%
Discount	0.4	51.07%	97.99%	67.14%	52.05%
	0.5	51.06%	92.77%	65.87%	51.93%
	0.6	47.51%	75.23%	58.24%	46.06%
	0.7	43.40%	53.93%	48.09%	41.80%
	0.8	36.88%	24.75%	29.62%	41.20%
	0.9	28.73%	5.71%	9.53%	45.77%

We then investigate how hyperparameters α and top-k affect the accuracy of the node classification. As shown in Figure 7, the node classification accuracy increases with an increasing hyperparameter α until it reaches its maximum at around 0.18. When the hyperparameter top-k value is increased, the accuracy of node classification increases sharply before 40 and then levels off. It is possible to empirically find an effective α and top-k to achieve the best accuracy.

B. Comparisons with Structural Ranking Baselines

The aforementioned invisible link injection attack aims to increase the PageRank score or similar web ranking algorithm scores of suspicious sites to fool search engines, so it is called ranking attacks. Therefore, existing ranking-based evaluations can be manipulated by adversaries to boost their scores, and their performances are reduced significantly. We further demonstrate these ranking-only approaches result in low precision, recall, and F1-score, as shown in Table III. Each node starts with a score of 1/(number of nodes), and then is divided by its children equally. The converged ranking values are normalized between 0 and 1.

As we have 50% of allowed and 50% of denied samples in the dataset, we first select the threshold that splits the dataset equally. For example, in PageRank, about half of the samples (46.20%) have ranking scores higher than 0.1. We further vary the threshold to other values, *e.g.*, 0.2, 0.3, 0.4, and 1.0 in PageRank; as in practice, it is difficult to know in advance how the positive and negative samples are distributed in the dataset.

According to the results presented in Table III, regardless of the varying thresholds, the PageRank only achieves precision around 50% (49.67% to 53.06%), which indicates roughly half of positive predictions are incorrect. The recall presents how many positive samples are correctly detected as positive. Note that, even when the threshold is set to 1, there are 2.56% of positive samples not detected as they have a PageRank score equal to 1.

TrustRank-based filtering is also evaluated, with the selected seeds from the allow-list nodes. The results are similar to the PageRank method. We further apply two strategies, Step Function, and Trust Discount Scoring, used for TrustRank, to distinguish the allowed and denied domains based on our crawled domain graph. As a trust attenuation strategy, Step Function limits the maximum path length from any nodes to the allow-list nodes and assigns a trust score of 1 to every node within n steps from the allow-list nodes, and a trust score of 0 otherwise. In a range from 3 to 7 steps, the precision and accuracy stay remain 50% (47.55% to 53.20% and 48.13% to 55.85%, respectively). When the step threshold is set to 3, recall reaches 97.89% while precision is 52.16%. This indicates the results have high false positives even while most positive samples are captured. Based on the F1-score, we could infer that setting the step threshold as 4 will result in the best but still not ideal performance. For the Trust Discount algorithm, each node is assigned a trust score equal to the sum of its parent's trust scores multiplied by a discount factor to dampen the trust score the further it gets from the allow-list nodes. Trust Discount runs with a discount factor from 0.4 to 0.9 yielded similar accuracy results to the Step Function (around 50%). This suggests scores simply scale based on the percentage of domains above the threshold and are mainly based on distance from the root node, with limited improvement in separating benign from malignant domains. These independent structure checks are therefore not suited for the large-scale graph with sparsely labeled nodes.

As demonstrated in § V-A and § V-B, after applying our integrated scheme which combines local features with global structural information together via the well-designed IncredulityRank-based propagation scheme, the accuracy of denied node classification derived from machine learning models is enhanced from 60% (using simple statistic features only) and 50% (structural analysis only) to 90% (with only 0.01% labeled nodes).

C. Static and Real-time Inference Evaluation

We compare performances of different incorporation schemes (*i.e.*, End-to-end Incorporation, Inference Incorporation, Training Incorporation, and Without Incorporation) of the PIR-based propagation according to the accuracy of the classification, as shown in Figure 3. For the Without Incorporation strategy, we treat each node independently and train the local prediction neural network using only the node feature. As shown in in Figure 3, our proposed method with End-to-end Incorporation achieves the best accuracy performance (more than 90%) compared to other strategies. The End-to-end Incorporation strategy, *i.e.*, the complete implementation of DOITRUST, enhances the accuracy by nearly 30%, compared to the WI scenario where the DOITRUST only considers individual features of the nodes.

As an ablation study, we further compare the other two strategies, Training Incorporation and Inference Incorporation (involving propagation only during the inference phase), with the Without Incorporation case. By involving structural information only during the training phase, DOITRUST achieves an 86.12% accuracy, which still outperforms individual machine learning baselines, graph neural network baselines, and structural processing only baselines, as reported in § V-A and § V-B. Compared to some of the large-scale graph neural network baselines (*i.e.*, APPNP and PPRGO), the TI strategy shows a slight gap in accuracy (around 3% to 5%). However, considering the time efficiency of our approach, which is demonstrated in later sections, it is feasible and efficient to implement DOITRUST with a TI strategy in practice. Finally, even for the Inference Incorporation (involving structural information only during the inference phase), the accuracy increases by 12% compared to without propagation case. Considering that the training time and complexity are reduced by removing the propagation during training, it is feasible to consolidate our method to pre-trained classifiers without neighbors' information taken into account, achieving a notable accuracy enhancement.

D. Efficiency in Real-world Deployment

In this section, we discuss how we could deploy the DOITRUST in practice and address the challenges of scalability and efficiency when processing massive data (*e.g.*, a large-scale domain graph with a million nodes increasing per hour) within a reasonable amount of time (*e.g.*, training and inference of a newly updated graph in less than one minute). In addition, we demonstrate an optimization of DOITRUST, which increases the execution speed to support the training on billions of nodes in an industrial environment. Acknowledgments from our industrial collaborators indicate that DOITRUST is practical to be used in real-world scenarios.

1) Time efficiency: We evaluate the average training time per epoch for the DOITRUST, as well as other baselines, deployed on a real-world server (Ubuntu OS, with NVIDIA Quadro RTX 4000 8GB GPU and i7 9900 32G CPU), as demonstrated in Figure 8. According to the results, DOITRUST is around two orders of magnitude faster than GCN and GAT on the Small dataset by avoiding the iteratively adjacent matrix processing and global propagation, and 10x faster than APPNP via avoiding a higher number of matrix multiplications. For the Large dataset, the training time of DOITRUST is less than 5s and faster than other state-of-the-art graph learning variants baseline models. Therefore, DOITRUST is practical to be deployed in the real world.

2) Memory efficiency.: We further evaluate the memory efficiency of DOITRUST in comparison to other baseline models in the single machine setting. For the Small dataset, DOITRUST needs 1.5GB of memory, compared to more than 2.0GB for Cluster-GCN and APPNP. The memory consumption scales with graph size, while DOITRUST has relevant less consumption of memory, increasing to 10.0GB on the Large dataset, compared to more than 20.0GB for PPRGO and out-of-memory for APPNP and Cluster-GCN. Our PIR estimation mechanism effectively bypasses pre-processing and space overheads, enabling the global propagation scheme to be tailored for large graphs with frequent updates.

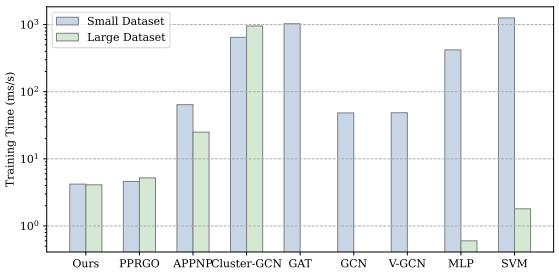


Fig. 8: Training time of different models on two size of datasets.

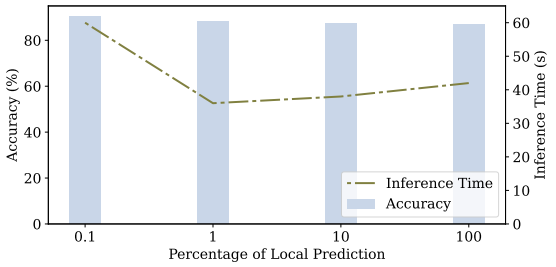


Fig. 9: Demonstration of the transferability of local prediction.

3) System optimization.: The training on a very large-scale graph with billions of nodes may face significant overheads. To address this challenge, we explore system optimization by reducing the scale of nodes involved in the training procedures while maintaining accuracy. We examine the performance of the node classification model when training is conducted on only a subset of nodes, *i.e.*, *transferability* of the node classification model. We find the accuracy of the model has little change when the update of the local prediction neural network is conducted for only a small, random subset of nodes. As shown in Figure 9, the accuracy only decreases by about 0.6 percent when reducing the amount of inferred nodes by a factor of 10. Additionally, the inference time does not change significantly when varying the percentage of location prediction.

4) Efficiency of pre-processing: We also evaluate the efficiency of the pre-processing step, *i.e.*, the top-k neighbor estimation. We evaluate the performance of top-k neighbor estimation between top-k PIR and MC ($\sqrt{1-\alpha}$ random walks) and FORA ($\delta = p_f = 1/n, \epsilon = 0.5$). We conduct 100 times top-1000 queries for each method, and report the average query time. As demonstrated in Figure 10, our top-k PIR estimation outperforms all benchmark methods in terms of query time for both datasets (less than 1 second on average to handle the huge graph). The memory consumption of top-k PIR for index and graph are only 411.9MB and 130.8MB, respectively, on the huge graph. Additionally, for the precision over these top-1000, the top-k PIR indeed achieves precision at 100%. We also evaluate the effects on the performance of our proposed method when varying hyperparameters r_{max} and k at other fixed parameters, using the averaged accuracy after ten times repetitions. We find the average accuracy consistently raises when increasing k or decreasing r_{max}. This means a more accurate approximation of the PIR vector or considering more top-k neighbors could improve the accuracy. Additionally, the difference between the highest accuracy and lowest accuracy is negligible ($< 2\%$), which means the algorithm is not sensitive to the k and r_{max}. Furthermore, the performance of the model

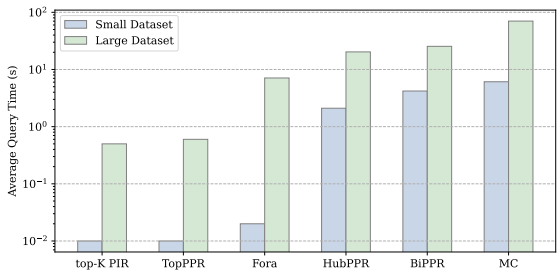


Fig. 10: Top-k PIR performance evaluation.

starts to be stable from the $k = 32$, for any r_{max} setting. Therefore, it is possible to find suitable hyper-parameters with a smooth trade-off between accuracy and computation cost in real-world deployment.

E. Validation and Findings on Compromised Nodes

DoITrust has pruned 1,138 allowed nodes as compromised nodes, which stand around 2.10% of 54,112 nodes in the extended allow-list, *i.e.*, 2.10% of allowed nodes (that are determined by third-party tools and manual analysis in § IV-B) are directly or indirectly connected to nodes in deny-lists and should not be involved in an allow-list. We further conducted a manual review on the detected compromised nodes. Specifically, four expert researchers (two co-authors and two external experts) manually checked the nodes lying on the path from detected compromised nodes to denied nodes, analyzing the HTML source code, the content of web page, and the service provided by the website. The four experts are separated into two groups. Each group reports a node as compromised once the evidence of connectivity between a detected compromised node and a denied node is found. If the decision on a node does not match, the two groups will exchange opinions and have sufficient discussions until an agreement is reached. Due to the large scale of the graph, which leads to significant manual efforts, we randomly sampled 110 (10%) detected compromised nodes in the manual review. Finally, 104 out of the 110 sampled nodes are validated as true positives, *i.e.*, the true positive rate of our DoITrust is 94.55%. We further investigate the underlying indicators of compromise in § V-F, and present several real-world cases of compromised domains in § VI.

F. Aggregated Measurement and Discussion

At a high level, the typical 2-hop investigative scope of existing GNNs would be bypassed by such stealthy attacks. This can be seen in Figure 11, where we aggregated the number of hops in paths from supposed trusted nodes (*i.e.*, compromised sites) to confirmed denied nodes. We see the mean number of hops stands at 6.94, with a σ of 1.69. In fact, the 25th percentile stands at 6 hops, meaning the majority of denied nodes are buried deep along the paths from the compromised sites. This alludes to the need for deeper investigative scopes in future approaches.

Takeaway: Searching for indicator of compromise within HTML is a mature but time- and labor-intensive undertaking. DoITrust aids in this regard, as it considers specific web compromising propagation in a global view against the plausible hyperlink injection. Finally, an obvious conclusion

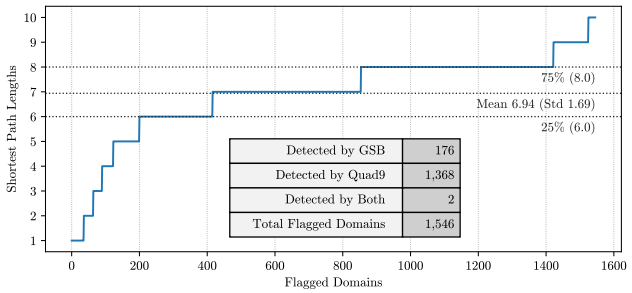


Fig. 11: Aggregation of the number of hops from supposed trusted nodes to all discovered denied nodes.

is that developers must maintain vigilance when using web templates. While these provide ease of use, most come with substantial security risks. Additionally, major web platforms such as WordPress should establish much stricter reviews of the content, themes and plug-ins they offer.

Broader implication: We further examine the unlabeled intermediate nodes along the compromise chain from compromised nodes to denied nodes. We find most of them have suspicion prediction values higher than compromised nodes and lower than denied nodes, and they tend to contain compromise relevant contents. For example, a number of portal websites (intermediate nodes) could be maintained by a pornography website to drive traffic to the main website (the denied node). In the stealthy compromise attack, the victim is injected with these portal websites, which is more stealthy than directly injecting the main website, as these portal websites will not trigger the alert. Our solution provides a feasible approach to enable the identification of such stealthy on-chain compromise by localizing the starting points along the compromise chain.

VI. Case Studies

We further investigate the underlying reasons for on-chain compromises. One possible reason could be the breached open source content management system (*i.e.*, WordPress CMS). CMS widely utilizes content, themes, and plug-ins provided by third parties, where plausible hyperlinks can be injected into any elements of HTML and scripts. These plausible hyperlinks pretend to be benign by manipulating patterns in order to evade security detection. The criteria of the on-chain compromise recognition used by our approach mainly focus on the fact that suspicious nodes are located in the path to denied nodes and are close to them, accumulating higher suspicion signals. Subtle-positive alarms result from higher suspicious scores, even when they have not been compromised yet but are at a high risk of being compromised. For example, subtle-positive nodes are developed using themes provided by open-source content management systems where similar themes are detected as positive. In another example, the detected node contains no suspicious contents, but some of its out-links finally direct to highly suspicious clusters or campaigns. Based on these inspections, we would argue that subtle-positive alerts are not false positives, since high suspicion scores can be used as risk alarms to trigger web security checks or warn that the web domain is at risk of being compromised. We report the following typical cases.

Non-Profit website with hidden malicious links. Here, we identified `c***a.org.**` as a compromised domain, despite initially being an allowed seed. Cursory checks show the site belonging to an anti-child abuse non-profit founded in 1963, which raised concerns when it led to various adult and malicious domains. For example, a downstream path contained a series of porn sites that ultimately led to the GSB-detected `liuyuedingxh.buzz`. The question then turns to *how did a supposed trusted site become compromised?* Accordingly, its HTML was manually reviewed, with all URLs within it passing checks by various third-party security vendors. However, a URL for `vi***bb.com` was discovered in a hidden hyperlink tag, as shown in Figure 13(a), which ultimately led to `liuyuedingxh.buzz`. Interestingly, the position of the hyperlink has been set off the viewable screen, via the attribute `left:-1909px`. Moreover, we quickly found that it contained explicit pornographic material. These checks were performed initially on 22 Jan 2021, and the link persists as of this writing. Delving further, we see the site was made using a WordPress theme, which indicates possible malicious injection via insecure- and/or unmaintained plug-ins. We note the threat is twofold. On one hand, it is common to use new domain names for adult, gambling and other inappropriate content. The stealthy compromise attack allows the attacker to exploit the reputation and ranking of the new domain through search engine optimization (SEO), by siphoning off the reputation score endorsed by the in-link connection injected into the compromised high-reputation website. Thus, it is possible when searching for children's charities, the results may also include pornography websites, as the website of the children's charity has been compromised to links to pornography sites. On the other hand, the injection of hyperlinks indicates the possibility of injecting malware or data theft attacks can also be conducted by the attacker.

Government linked services provider site with hidden popups. `ca***es.com.**` is a commercial entity providing facility management, cleaning, and disaster recovery services to government agencies. Its domain and URLs were recognized as clean by security vendors, but a hidden popup contains multiple links leading to malicious and inappropriate domains, as seen in Figure 13(b). The `display` attribute of the modal was set to `none` by default, so there was no apparent way to trigger its visibility. Interestingly, the malicious links were randomized on each page load, implying a level of sophistication beyond surface-level HTML injection, and that an external malicious database was dynamically involved. Upon closer inspection, we noticed that the top-level container for the hidden popup had a unique ID and class: `shbNet-PaddingWr` and `shbNetPopupWr`, respectively. A simple online search subsequently surfaced other WordPress sites that contain similar hidden popups, such as: `hu***en.com.**` and `ve***or.com.**`. We manually edited in the browser tools to make the dynamically loaded popup window visible for inspection, as shown in Figure 12(b) in the Appendix.

Private high school website with inappropriate links in invisible containers. In yet another example, `'s**w.edu.**'` is the official website for a private high school, and contains a highly trusted top-level domain. While not a WordPress site, it still hosted suspicious links to pornographic websites that were only found through DoITrust. As shown in Figure 13(c) and Figure 12(b) in the Appendix, `<div>` objects with heights and widths set to 0 were used to hide links to suspicious adult or gambling sites. While invisible to the human eye, such

links would be picked up by web-crawlers, which could be the motivation. We further present a screenshot that indicates the hidden hyperlink object in Figure 12(c) in the Appendix.

VII. Related Work

This section discusses supervised webpage classification, topology analysis for malicious node detection, and graph learning approaches for web security evaluations.

Supervised webpage classification. Existing studies mostly focus on collecting statistical features from a pile of malicious and benign samples from URLs and HTML to identify malicious patterns [18]–[21], [37]. Expressive features and sufficient labeling supervision information are two key components for the success of the classification. The commonly adopted features are the on-page features, *i.e.*, information directly located on the page, including textual content of HTML, tags, hyperlinks, and anchor text. Statistics on such information could be conducted automatically and directly to be the features for the classification. The compromised and bridging (from compromised victim to target) nodes are always stealthy to hide the malicious tracks, in order to avoid detection. Additionally, the node classification in our work is characterized by a lack of labeling information. Stealthy patterns and insufficient supervision information lead to inaccurate predictions, indicating that existing supervised learning approaches were ineffective in recognizing on-chain compromises. Besides the automatically selected features, other hand-selected features are also used in existing malicious webpage classification, such as the IP/host-name, or registrar information from third-party vendors. Such information can be used as additional features to enhance the feature representation capacity, which will be exploited as one of the future works.

Topology analysis for malicious recognition. Existing studies use structural information only to identify malicious behaviors, such as by applying PageRank [9] and TrustRank [10] as a semi-automatic technique to identify malicious websites from benign websites. They manually identified seed domains as benign websites and then used the web's link structure to discover other likely benign pages. Additionally, Li *et al.* [38] applied PageRank on the Host/IP graph connected by the redirection activity, which is characterized by static and shallow connections from fixed domains (such as the driver download website), resulting in relatively fixed malicious patterns. Alternatively, the hyperlink topology is a straightforward resource to obtain and observe, unlike other topologies that are often obtained in controlled environments and needed to trigger malicious activities. We also solely apply the PageRank or TrustRank analysis in our graph, resulting in low precision, recall, and accuracy (detailed in § V-B). The reason is ranking-based approaches only focus on label propagation and fail to capture malicious behavior propagation when labels are sparse in the graph. In addition, fully "benign" domains are always hard to define, which serve as a crucial and sensitive starting point for the existing structure evaluation approaches. With these gaps in mind, our idea to enhance detection accuracy is to integrate individual domain features with global structural information derived from hyperlinks.

Graph learning for web security. Numerous studies apply graph analysis to web security evaluations, such as the trustworthiness of user reviews on social networks [39], the detection of Sybil attacks [40]–[43], and exposing the exploitation of malicious URLs or accounts in social media [44]–[46], malicious reuse of taken-down domains [47] and even malware distribution [48]. Our research is the first study that focuses on the on-chain compromise of web domains. Unlike these existing approaches conducting separated graph analysis and independent classification, our design integrates the graph analysis into the learning progress. Several works apply graph neural networks for malicious domain detection, such as HG-Dom [49] using a heterogeneous graph convolutional network method on traffic information from the Domain Name System (DNS). Graph convolutional network (GCN) is also used for evaluating the maliciousness of domains [50]. However, existing GNNs face prevailing constraints of performance and scalability in large graphs, *e.g.*, over-smoothing when increasing the size of the neighborhood and expensive neighborhood expansion when adding additional layers. Additionally, web-compromising behaviors have a typical long chain of influence and are stealthy in nature, which is beyond the 2-hop scope of ordinary GNNs. Existing approaches, such as PageRank [9] and its associated graph learning schemes, focusing on the outgoing links from benign websites, cannot withstand such attacks. Existing general graph convolutional networks are not feasible in the on-chain compromise scenario due to scalability restrictions, especially when the labeling information is minimal and sparse. To the best of our knowledge, DoITrust is the first work to enable scalable and portable graph learning on large-scale graphs, and provide a more practical solution to the industry setting.

VIII. Conclusion

We introduced an integrated scheme to conduct on-chain compromise analysis called DoITrust, providing a scalable and accurate semi-supervised method of compromised domain measurement and pruning solutions. We created a large-scale Website Domain Graph baseline dataset associated with ground truth node labels and statistical features derived from HTML. By introducing separated learning, the proposed DoITrust bypasses expensive message-passing overheads, and can scale easily to graphs with millions of nodes in consideration of the influence of relevant nodes located multiple hops away. Towards the efficient propagation specified for the website domain scenario, we develop the Personalized IncredulityRank, as an algorithmic way of evaluating compromising behaviors in propagandistic networks. In addition, we provide an efficient approximation of the PIR approach on a large graph. We demonstrate the performance of the DoITrust in terms of false positive rate, learning capability, scalability, and portability compared to state-of-the-art approaches. We also show the flexibility of DoITrust via incorporating any pre-trained machine learning models without any additional training, as a practical solution in industry settings. Although we acknowledge DoITrust may not be amenable for a strong adaptive adversary to bypass the newly-designed personalized IncredulityRank, more robust approximations of personalized IncredulityRank based propagation schemes would benefit from this methodology in the future. Additionally, effectiveness of DoITrust can be further enhanced by combining it with other domain evaluation techniques, such as traffic analysis, or by incorporating elaborate HTML features extraction. We hope

the system DOITTRUST could provide an opportunity for broad Internet service providers, or any stakeholders, to curb the rampant emergence of such hidden attacks.

ACKNOWLEDGEMENT

We thank the anonymous reviewers for their valuable comments. The work has been supported by the Cyber Security Research Centre Limited whose activities are partially funded by the Australian Government's Cooperative Research Centres Programme.

REFERENCES

[1] Rania Zaimi, Mohamed Hafidi, and Mahnane Lamia. Survey paper: Taxonomy of website anti-phishing solutions. In *2020 Seventh International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pages 1–8, 2020.

[2] Steve Sheng, Brad Wardman, Gary Warner, Lorrie Cranor, Jason Hong, and Chengshan Zhang. An empirical analysis of phishing blacklists. In *Proceedings of the Sixth Conference on Email and Anti-Spam (CEAS 2009)*, 2009.

[3] JungMin Kang and DoHoon Lee. Advanced white list approach for preventing access to phishing sites. In *2007 International Conference on Convergence Information Technology (ICCIT 2007)*, pages 491–496, 2007.

[4] Gunikhan Sonowal and KS Kuppusamy. Phidma–a phishing detection model with multi-filter approach. *Journal of King Saud University-Computer and Information Sciences*, 32(1):99–112, 2020.

[5] Yury Zhauniarovich, Issa Khalil, Ting Yu, and Marc Dacier. A survey on malicious domains detection through DNS data analysis. *ACM Computing Surveys (CSUR)*, 51(4):1–36, 2018.

[6] Sana Hamdi, Alda Lopes Gancarski, Amel Bouzeghoub, and Sadok Ben Yahia. Tison: Trust inference in trust-oriented social networks. *ACM Transactions on Information Systems (TOIS)*, 34(3):1–32, 2016.

[7] Paulo Shakarian, Jana Shakarian, and Andrew Ruef. Chapter 9 - losing trust in your friends: Social network exploitation. In *Introduction to cyber-warfare: A multidisciplinary approach*, pages 171–183. Syngress, Boston, 2013.

[8] John R Vacca. *Computer and information security handbook*. Newnes, 2012.

[9] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[10] Zoltan Gyongyi, Hector Garcia-Molina, and Jan Pedersen. Combating web spam with TrustRank. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB)*, 2004.

[11] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[12] Google. Google safe browsing. https://safebrowsing.google.com/, 2022.

[13] Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. Scaling graph neural networks with approximate pagerank. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2464–2473, 2020.

[14] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized PageRank graph neural network. In *International Conference on Learning Representations*, 2021.

[15] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized PageRank. In *International Conference on Learning Representations*, 2018.

[16] Sibo Wang, Renchi Yang, Xiaokui Xiao, Zhewei Wei, and Yin Yang. Fora: Simple and effective approximate single-source personalized PageRank. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 505–514, 2017.

[17] Dániel Fogaras, Balázs Rácz, Károly Csalogány, and Tamás Sarlós. Towards scaling fully personalized PageRank: Algorithms, lower bounds, and experiments. *Internet Mathematics*, 2(3):333–358, 2005.

[18] Xiaoguang Qi and Brian D Davison. Web page classification: Features and algorithms. *ACM Computing Surveys (CSUR)*, 41(2):1–31, 2009.

[19] Min-Yen Kan and Hoang Oanh Nguyen Thi. Fast webpage classification using url features. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pages 325–326, 2005.

[20] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. 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*, pages 1245–1254, 2009.

[21] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. Identifying suspicious urls: an application of large-scale online learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 681–688, 2009.

[22] Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using PageRank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 475–486. IEEE, 2006.

[23] Sibo Wang, Renchi Yang, Runhui Wang, Xiaokui Xiao, Zhewei Wei, Wenqing Lin, Yin Yang, and Nan Tang. Efficient algorithms for approximate single-source personalized PageRank queries. *ACM Transactions on Database Systems (TODS)*, 44(4):1–37, 2019.

[24] Peter Lofgren, Siddhartha Banerjee, and Ashish Goel. Personalized PageRank estimation and search: A bidirectional approach. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 163–172, 2016.

[25] Zhewei Wei, Xiaodong He, Xiaokui Xiao, Sibo Wang, Shuo Shang, and Ji-Rong Wen. Topppr: Top-k personalized PageRank queries with precision guarantees on large graphs. In *Proceedings of the 2018 International Conference on Management of Data*, pages 441–456, 2018.

[26] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

[27] PhishTank. Join the fight against phishing. http://phishtank.org/, 2022.

[28] MalwareURL. Fighting malware and cyber criminality. https://www.malwareurl.com/, 2022.

[29] RiskAnalysis. Community projects. https://riskanalytics.com/community/, 2022.

[30] Zeustracker. Tracker amd advanced traffic filtering. https://zeustrack.io/, 2022.

[31] Malwaredomainlist. Malware domain list. http://www.malwaredomainlist.com/, 2022.

[32] The Université Toulouse 1 Capitole. Blacklists ut1. https://dsi.ut-capitole.fr, 2020.

[33] Alexa. The top sites on the web. https://safebrowsing.google.com/, 2020.

[34] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR*, 2017.

[35] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[36] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 257–266, 2019.

[37] Abubakr Sirageldin, Baharum B Baharudin, and Low Tang Jung. Malicious web page detection: A machine learning approach. In *Advances in Computer Science and its Applications*, pages 217–224. Springer, 2014.

[38] Zhou Li, Sumayah Alrwais, Yinglian Xie, Fang Yu, and XiaoFeng Wang. Finding the linchpins of the dark web: a study on topologically dedicated hosts on malicious web infrastructures. In *2013 IEEE Symposium on Security and Privacy*, pages 112–126. IEEE, 2013.

[39] Gang Wang, Manish Mohanlal, Christo Wilson, Xiao Wang, Miriam Metzger, Haitao Zheng, and Ben Y Zhao. Social turing tests: Crowdsourcing sybil detection. In *Proceedings of the 22nd Network and Distributed System Security Symposium (NDSS)*, 2013.

[40] Gang Wang, Tristan Konolige, Christo Wilson, Xiao Wang, Haitao Zheng, and Ben Y Zhao. You are how you click: Clickstream analysis for sybil detection. In *22nd USENIX Security Symposium (USENIX Security 13)*, pages 241–256, 2013.

[41] Haizhong Zheng, Minhui Xue, Hao Lu, Shuang Hao, Haojin Zhu, Xiaohui Liang, and Keith Ross. Smoke screener or straight shooter: Detecting elite sybil attacks in user-review social networks. In *Proceedings of the 27th Network and Distributed System Security Symposium (NDSS)*, 2018.

[42] Suibin Sun, Le Yu, Xiaokuan Zhang, Minhui Xue, Ren Zhou, Haojin Zhu, Shuang Hao, and Xiaodong Lin. Understanding and detecting mobile ad fraud through the lens of invalid traffic. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 287–303, 2021.

[43] Tong Zhu, Yan Meng, Haotian Hu, Xiaokuan Zhang, Minhui Xue, and Haojin Zhu. Dissecting click fraud autonomy in the wild. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 271–286, 2021.

[44] Beliz Kaleli, Brian Kondracki, Manuel Egele, Nick Nikiforakis, and Gianluca Stringhini. To err. is human: Characterizing the threat of unintended URLs in social media. In *Proceedings of the 28th Network and Distributed System Security Symposium (NDSS)*, 2021.

[45] Sangho Lee and Jong Kim. Warningbird: Detecting suspicious urls in twitter stream. In *Proceedings of the 21st Network and Distributed System Security Symposium (NDSS)*, volume 12, pages 1–13, 2012.

[46] Manuel Egele, Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. Compa: Detecting compromised accounts on social networks. In *Proceedings of the 22nd Network and Distributed System Security Symposium (NDSS)*, 2013.

[47] Eihal Alowaisheq, Peng Wang, Sumayah Alrwais, Xiaojing Liao, XiaoFeng Wang, Tasneem Alowaisheq, Xianghang Mi, Siyuan Tang, and Baojun Liu. Cracking the wall of confinement: Understanding and analyzing malicious domain. In *Proceedings of the 28th Network and Distributed System Security Symposium (NDSS)*, 2019.

[48] Luca Invernizzi, Stanislav Miskovic, Ruben Torres, Christopher Kruegel, Sabyasachi Saha, Giovanni Vigna, Sung-Ju Lee, and Marco Mellia. Nazca: detecting malware distribution in large-scale networks. In *Proceedings of the 23rd Network and Distributed System Security Symposium (NDSS)*, volume 14, pages 23–26, 2014.

[49] Xiaoqing Sun, Jiahai Yang, Zhiliang Wang, and Heng Liu. Hgdom: Heterogeneous graph convolutional networks for malicious domain detection. In *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*, pages 1–9. IEEE, 2020.

[50] Yuta Kazato, Yoshihide Nakagawa, and Yuichi Nakatani. Improving maliciousness estimation of indicator of compromise using graph convolutional networks. In *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–7. IEEE, 2020.

[51] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[52] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. Adaptive graph convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[53] Hanjun Dai, Zornitsa Kozareva, Bo Dai, Alex Smola, and Le Song. Learning steady-states of iterative algorithms over graphs. In *International Conference on Machine Learning*, pages 1106–1114. PMLR, 2018.

[54] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1416–1424, 2018.

[55] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[56] Konstantin Avrachenkov, Nelly Litvak, Danil Nemirovsky, and Natalia Osipova. Monte carlo methods in PageRank computation: When one iteration is sufficient. *SIAM Journal on Numerical Analysis*, 45(2):890–904, 2007.

[57] Bahman Bahmani, Abdur Chowdhury, and Ashish Goel. Fast incremental and personalized pagerank. *Proceedings of the VLDB Endowment*, 4(3), 2010.

[58] Glen Jeh and Jennifer Widom. Scaling personalized web search. In *Proceedings of the 12th international conference on World Wide Web*, pages 271–279, 2003.

[59] Atish Das Sarma, Anisur Rahaman Molla, Gopal Pandurangan, and Eli Upfal. Fast distributed pagerank computation. In *International Conference on Distributed Computing and Networking*, pages 11–26. Springer, 2013.

[60] Reid Andersen, Christian Borgs, Jennifer Chayes, John Hopcraft, Vahab S Mirrokni, and Shang-Hua Teng. Local computation of PageRank contributions. In *International Workshop on Algorithms and Models for the Web-Graph*, pages 150–165. Springer, 2007.

[61] Jieming Shi, Renchi Yang, Tianyuan Jin, Xiaokui Xiao, and Yin Yang. Realtime top-k personalized PageRank over large graphs on gpus. *Proceedings of the VLDB Endowment*, 13(1):15–28, 2019.

[62] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[63] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pages 5453–5462. PMLR, 2018.

APPENDIX

A. Graph Neural Networks

GNNs aim to represent a node combining both node features and the structural graph information [11]. Given a graph G with an optimized graph structure A, the target of the GNN is to learn state embedding Z for each node considering its features X and the aggregated features from neighborhood nodes, which is used to refine the representation of the node. Generally, the message-passing of GNN approaches is conducted with two stages: (1) messages are propagated along with the neighbors, and (2) the messages are aggregated to obtain the updated representations [13]. The parametric propagation function is shared among all nodes, updating the node state according to its neighborhood. The local output function is used to produce the output.

The variants of GNN apply various aggregators to gather information from the neighborhood, mainly divided into Convolution and Attention. Graph Convolutional Network (GCN) approaches are mainly divided into two taxonomies: spectral-based and spatial-based GCN. Spectral based GCNs [34], [51], [52] extends the convolution using spectral graph theory via the layer-wise propagation:

$$h^{(l+1)} = \sigma(D^{-1/2}\tilde{A}D^{-1/2}h^{(l)}W^{(l)}) \tag{3}$$

Here, $h^{(l)}$ and $W^{(l)}$ are the output and the trainable parameters of the l^{th} hidden layer, respectively. σ denotes an activation function. $\tilde{A}$ is the adjacent matrix with self-connections, D is a diagonal matrix with $D_{ii} = \sum_j \tilde{A}_{ij}$. To address the decomposition of the Laplacian matrix and necessary operation on the whole graph, spatial-based [53]–[55] is conducted in a more dynamical manner. Graph convolutions are redefined as feature aggregation with neighbor nodes and improve efficiency and flexibility with sampling strategies. The attention mechanism has been successfully used to propagation step in neural graph networks, such as Graph attention network (GAT) [35]. The

hidden states of each node are figured out by attending over its neighbors, following a self-attention strategy.

$$h^i = \sigma(\sum_{j \in N_i} \alpha_{ij} W h^j), \tag{4}$$

where α_{ij} is the attention coefficient of node j to i, N_i denotes the neighborhoods of node i in the graph.

B. Personalized PageRank and Approximation

Personalized PageRank (PRR) is a standard definition for finding vertices in a graph that are most relevant to a given node, recursively modeling the importance of nodes on a directed graph. Given a source node s whose point of view we take, a node is important if its in-neighbors are important, *i.e.*, the PPR $PR_s(u)$ reflecting the importance of each node u with respect to s. To keep the importance scores bounded, we normalize the importance given from a node u to a node v though an edge (u, v) by dividing by u's out-degree. In addition, we choose a decay term α, and transfer a fraction $1 - \alpha$ of each node u's importance to u's out-neighbors. PPR has high complexity, especially for large graph, so there are some approximate estimation algorithms for PPR values [56]–[59]. Formally, the Personalized PageRank vector with respect to source node s is the solution to the recursive equation

$$PPR_s(u) = \alpha \sum_{v \in N_{in}(u)}^{N} \frac{1}{N_{out}(v)} PPR_s(v) + (1 - \alpha)\frac{1}{n} \tag{5}$$

Given the adjacent matrix A and degree matrix D of graph G, the PPR matrix is also defined as the

$$\Pi^{PPR} = \alpha(I_n - (1 - \alpha)D^{-1}A)^{-1} \tag{6}$$

Here, each row of Π^{PPR} is the PPR vector for each node.

One common PPR estimation approach is Monte-Carlo (MC) [17] via producing $rw(s)$ random walks from a given source node s. The estimated PPR $\hat{\pi}(s, v)$ from s to v is represented as $\frac{rw(s,v)}{rw(s)}$, where $rw(s, v)$ is the random walks from s and terminated at v. Monte-Carlo (MC) is a classic solution for PPR estimation. MC generates $rw(s)$ random walks from source node s. For each node v, there are $rw(s, v)$ random walks stopping at v. MC has guarantees on the approximate solution, with the cost of the efficiency.

Forward Push (FP) [22] is also proposed to approximate the PIR scores $\hat{\pi}(s, v)$. With respect to a source node s, every node v is associated with two values: reserve value $\tilde{\pi}(s, v)$ and residue value $r(s, v)$. Initially, $\tilde{\pi}(s, s) = 0$, $r(s, s) = 1$ and $r(s, v) = \tilde{\pi}(s, s) = 0, \forall s \neq v$. Given a global residual threshold r_{max} , the FP iteretively pushes the residues of all nodes with $\frac{r(s,v)}{d_{out}(v)} > r_{max}$ (frontiers) to their reserves and the residues of their out-neighbors, where $d_{out}(v)$ is the number of outgoing neigbours of v. Specifically, for each node u in the set of out-neigbours of v, denoted by $N_{out}(v)$, the residue is updated via

$$r(s, u) = r(s, u) + (1 - \alpha)\frac{r(s, v)}{d_{out}(v)}, \tag{7}$$

and the reserve value of v is updated via

$$\tilde{\pi}(s, v) = \tilde{\pi}(s, v) + \alpha r(s, v), \tag{8}$$

followed by resetting the residue of v to 0, until no node to push. Finally, the $\tilde{\pi}(s, v)$ is used to estimate the $\pi(s, v)$, with time complexity $O(1/r_{max})$ but without guarantee. Similarly, [60] proposed backward search for PPR estimation.

FORA [16] combines both Forward Push and Monte-Carlo for the top-k PPR queries via iteratively conducting FP from s at first, then producing random walks from the nodes with non-zero residues. Formally, the estimation of PPR is given as

$$\hat{\pi}(s, v) = \tilde{\pi}(s, v) + \sum_{v \in V} r(s, v)\pi'(s, v), \tag{9}$$

where $\tilde{\pi}(s, v)$ and $r(s, v)$ are from FP, and $\pi'(s, v)$ is from MC. The total time complexity is minimized at $r_{max} = \frac{\epsilon}{\sqrt{m}}\sqrt{\frac{\delta}{(2\epsilon/3+2)log(2/p_f)}}$.

C. Approximation of Top-k Personalized IncredulityRank Values

(i) Forward Push for coarse estimation. The Forward Push algorithm considers maintaining two values for a given node s: reserve $\tilde{\pi}(s, t)$, and residue $r(s, t)$. We apply an adaptive initialization strategy, where the reserve and residue of all nodes are 0, except for $r(s, s)$, which is set to the initialized PIR from the aforementioned initialization step. Specifically, given a graph G, a source node s, teleport probability α, and a residue threshold r_{max}, Forward Push is conducted to obtain $\tilde{\pi}(s, t)$ and $r(s, t)$ for each target node $t \in V$. Forward Push conducts Push operation to process the residue for each node, until no node is active (*i.e.*, $\forall t \in V,\ r(s, t) > d_t \times r_{max}$), where d_t is the degree of the target node t. During each iteration, α portion of t's residue is converted to the reverse, *i.e.*, $\tilde{\pi}(s, t) \leftarrow \tilde{\pi}(s, t) + \alpha r(s, t)$, the rest $(1 - \alpha)$ portion of $r(s, t)$ is evenly distributed to the residues of t's out-neighbors. After the residue is processed, $r(s, t) \leftarrow 0$. Formally, the Forward Push of PIR is given as

$$\hat{\pi}(s, t) = \tilde{\pi}(s, t) + \sum_{v \in V} r(s, v)\pi'(v, t), \tag{10}$$

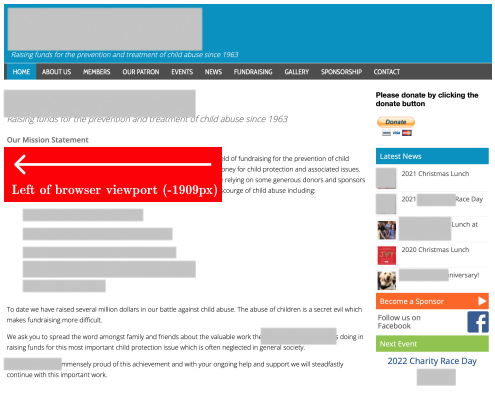
where $\tilde{\pi}(s, t)$ and $r(s, v)$ are from Forward Push, and $\pi'(v, t)$ is from MC.

(ii) Random walks for refining estimation. For every node v_i whose residue is larger than zero, we conduct $rw_i = r(s, v_i) \times rw/r_{sum}$ random walks from it. The number of random walks rw from each node is decided via

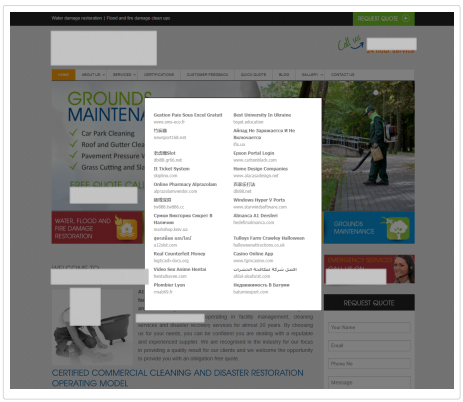
$$rw = d_{out} \times r_{sum} \times \frac{(2\epsilon/3 + 2) \times log(2/p_f)}{\epsilon^2 \times \delta}, \quad r_{sum} = \sum_{v_i \in V} r(s, v_i), \tag{11}$$

where r_{sum} is the sum of residue for all nodes [16], ϵ is the estimation error bound, and p_f is the failure probability. When a random walk terminates at a node t, then the $\pi'(v, t)$ grows by $(\frac{r(s,t)}{r_{sum}} \times \frac{rw}{rw_i}) \times \frac{r_{sum}}{rw}$.

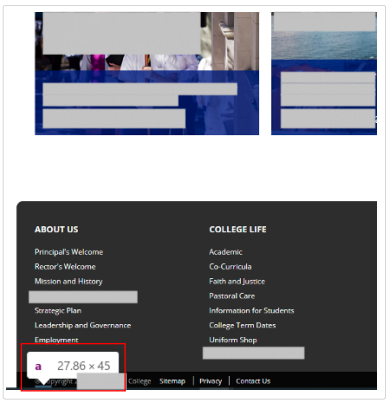
For the random walk, α part of them is expected to stop at the current node, and we can immediately record the portion of such random walks within $\mathcal{O}(1)$ time and hence avoid exactly simulating these random walks. Therefore, the estimation of



(a) Hyper link hidden off-screen in c***a.org.**



(b) Screenshot of hidden popup (forced visible) from ca***es.com.**



(c) Hidden hyperlink in footer from s**w.edu.**

Fig. 12: Screenshots from compromised domains initially labeled as benign.

TABLE IV: URLs/HTML statistic features.

Notation	Explanation
Length of URLs	Several parts are protocol, domain name or IP address, optional part, directory file, if HTTP GET request is used then a question mark followed by "key=value" pairs. In the data set we collected average domain character string length is less in benign web pages and more in malicious web pages.
Special Characters in URLs	The special characters that appears in URLs are number of dots ('.'), number of hyphens ('-'), number of forward slashes ('/'), number of underscores ('_'), number of equal signs ('=') and number o f client and/or server words in the URL. Average special characters appear less in benign web pages and more in malicious web pages in our collected data set.
URLs that Contain Address	This feature is indicative of malicious URLs because instead of domain name the IP address are used in URLs to compromise the victim.
Re-directions	Whether destination and an original URL are in the same domain. Number of redirection to different pages is more in malicious than in benign web pages.

```
<div id="dynamic-content" class="outline template-region-wrap
pl-area-container">
<!-- Canvas Area | Section Template -->
    <section id="pl_areau62e7a" ...></section>
    <a href="http://vi***bb.com" sytle="position:absolute;
    left:-1909px; top:0">vi***bb.com</a>
</div>
```

(a) Malicious link hidden off screen.

```
<div id="shbNetPaddingWr" class="shbNetPopupWr" style="display: none;">
    <table id="shbNetPaddingTable" class="shbNetPopupTable"
    style="display: none; position: fixed; margin: 0px; padding: 0px;
    left: 0px; top: 0px; width: 100%; height: 100%; direction: ltr;
    z-index: 9999999999; background: none;" width="100%" height="100%"
    cellspacing="0" cellpadding="0">...</table>
</div>
```

(b) Multiple malicious links in a hidden popup.

```
<div class="footer__copyright">
    "Copyright"
    <span id="js-copyrightDate">2021</span>
    "***** College"
</div>
<div style="overflow: auto; position: absolute; height: 0pt; width: 0pt;">
    <a href="http://it***ra.com" title="escort ankara">escort ankara</a>
</div>
<div style="overflow: auto; position: absolute; height: 0pt; width: 0pt;">
    <a href="http://se***iz.com" title="sex hatti">sex hatti</a>
</div>
```

(c) Malicious links in invisible containers.

Fig. 13: Code samples from compromised domains.

PIR is renewed as

$$\hat{\pi}(s,t) = \tilde{\pi}(s,t) + \alpha \times r(s,t) + \sum_{v \in V} r'(s,v)\pi''(v,t) \quad (12)$$

Here $\pi''(v,t) = (1-\alpha)E(X_t^v)$ and X_t^v is defined as: when sampling a random walk from each node v, one of its out-neighbor u will be randomly chosen, followed by conducting a random walk from u. $X_t^v = 1$ means the random walk terminates at t, and otherwise $X_t^v = 0$. Consequently, $r'(s,v_i) = (1-\alpha)r(s,v_i)$ and $rw_i = r(s,v_i)(1-\alpha) \times rw/r_{sum}$.

(iii) Top-k PIR values selection. We first perform the aforementioned estimation procedure using $\delta = 1/(k \times 2^{i-1}), \epsilon' = \epsilon/2$ and $p'_f = p_f/(nlog_2(n/k))$ for at most $log_2(n/k)$ iterations [23], where i presents the i^{th} iteration. We then select the top-k PIR estimated value for each node and assess the accuracy via evaluating whether k^{th} estimation estimation $\hat{\pi}(s,v_k) \geq (1+\epsilon) \times \delta$. If the accuracy of the top-k estimation is not satisfied, we continue the iteration with the halved value of δ. The global residue threshold r_{max} could be set at $\frac{\epsilon}{\sqrt{m}}\sqrt{\frac{\delta}{(2\epsilon/3+2)log(2/p_f)}}$ [16], and δ is set at $1/n$.

(iv) Adaptive threshold for efficiency and scalability. We use the output PIR values from the first iteration of the Forward Push to estimate the global PageRank, denoted by $pr(u)$, as

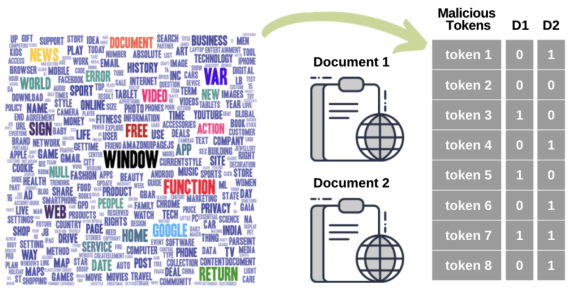


Fig. 15: Malicious dictionary and bag-of-maliciousness

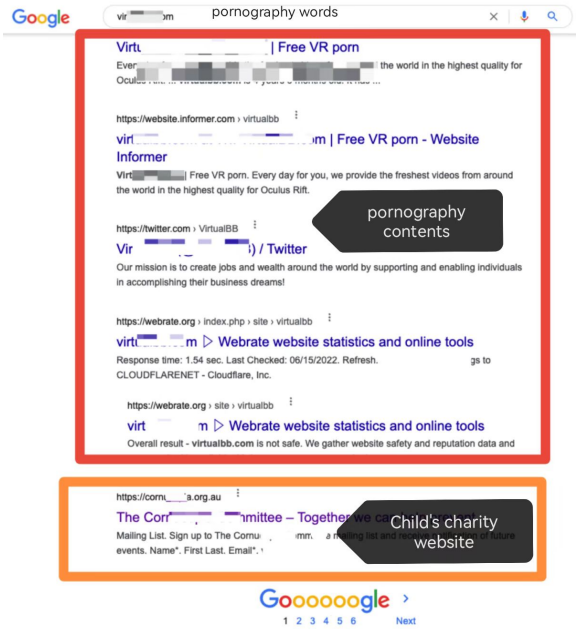


Fig. 14: Screenshots of associated search results for children's charities and pornography/paedophilia websites when searching the pornography words.

a measurement of the global importance of a node u with respect to the whole graph. Generally, a node u with larger $pr(u)$ tends to accumulate higher residue, and then the adaptive $r_{max}^{v_i}$ strategies could be applied [61].

D. Separate Learning for Scalability of GNNs

For each layer of the typical GNNs, the feature transformation is coupled with the propagation (aggregation) of messages passing among the neighborhoods. This leads to limitations in terms of the efficiency and scalability of the existing GNNs. For example, expanding the number of layers is beneficial to incorporate information from more distant neighbors, resulting in the over-smoothing and computational prohibition of recursive neighborhood expansion on large graph [13], [62], [63]. To address these limitations, Separate Learning based on decoupled feature transformation from the propagation is proposed [13]–[15]. Namely, the prediction on features for each node is produced locally and independently at first, followed by propagation/aggregation of local predictions via personalized PageRank. Formally, the node embedding is defined as

$$\begin{aligned} Z &= \mathrm{softmax}(\alpha(I_n - (1-\alpha)D^{-1/2}\tilde{A}D^{-1/2})^{-1}H), \\ H_i &= f_\theta(x_i) \end{aligned} \tag{13}$$

Here, α is a teleport probability, H is the local prediction matrix for a specific node generated via a neural network f_θ, and Z is the node embedding matrix after propagation. The key point of decoupling learning is how to efficiently calculate the dense propagation matrix $\alpha(I_n-(1-\alpha)D^{-1/2}\tilde{A}D^{-1/2})^{-1}$ and pre-computation of personalized PageRank, such as a variant of power iteration in [15] or sparse approximation in [13].

E. Statistic features derived from URLs/HTML

We summarize the statistic features derived from URLs and HTML in Table IV.