# DETECTION OF MALICIOUS DNS AND WEB SERVERS USING GRAPH-BASED APPROACHES

*Jinyuan Jia*[^,*], *Zheng Dong*[±], *Jie Li*[±], *Jack W. Stokes*[†]

[^] Duke University, Durham, NC 27708 USA
[±] Microsoft Corp., One Microsoft Way, Redmond, WA 98052 USA
[†] Microsoft Research, One Microsoft Way, Redmond, WA 98052 USA

## ABSTRACT

The DNS hijacking attack represents a significant threat to users. In this type of attack, a malicious DNS server redirects a victim domain to an attacker-controlled web server. Existing defenses are not scalable and have not been widely deployed. In this work, we propose both unsupervised and semi-supervised defenses based on the available knowledge of the defender. Specifically, our unsupervised defense is a graph-based detection approach employing a new variant of the community detection algorithm. When the IP addresses of several compromised DNS servers are available, we also propose a semi-supervised defense for the detection of compromised or malicious web servers which host the web content. We evaluate our defenses on a real-world attack. The experimental results show that our defenses can successfully identify these malicious web servers and/or DNS server IPs. Moreover, we find that a deep learning-based algorithm, i.e., node2vec, outperforms one which employs belief propagation.

***Index Terms***— DNS, Hijacking, Community, Detection, node2vec

## 1. INTRODUCTION

The DNS (Domain Name System) [1] lives at the foundation of the modern Internet. With its capability in IP address resolution, most Internet users can enjoy the more user-friendly domain names during their web browsing. Furthermore, DNS has enabled many operations on websites, such as load balancing amongst servers, to be transparent to visitors. The security of the native DNS infrastructure relies on two important assumptions. First, each DNS server is trustworthy. This does not only apply to the DNS server to which end users send their domain queries, but also to each DNS server that is involved in the recursive process if the previous server does not have a response to the query. Second, the communication channel between the end user and the DNS server is not tampered with. An attacker cannot forge or modify a DNS query or response. With a strong adversary (sometimes even state-sponsored), meeting these two assumptions can often be challenging.

To address the issue in the native DNS, the IETF proposed the use of DNS Security Extension (DNSSEC) [2], utilizing digital signatures for the end user and the DNS server. However, DNSSEC is not currently widely deployed [3]. In addition, this extension only addresses the authenticity part of the issue (the second assumption above). A DNS server can be run or compromised by an adversary, and the end users are still vulnerable to malicious websites if the DNS returns a malicious IP address. More recently, DNS hijacking has become a more widespread behavior of malware. As an example, the Novidade malware campaign [4] exploited vulnerable routers, which led to any devices (infected or not) connecting to the network being impacted.

In this work, we present our analysis on identifying DNS hijacking attacks using an internal Microsoft dataset. We propose utilizing an innovative graph-based approach to convert the dataset into a bipartite graph. Given the constructed graph, we build both unsupervised and semi-supervised defenses. Specifically, our unsupervised defense is based on a variant of community detection. We have two semi-supervised defenses. The first defense is based on Node2vec [5], which is a deep learning-based graph embedding algorithm. The second defense is based on belief propagation [6]. We use a Novidade malware campaign, a real-world DNS hijacking attack, to validate the effectiveness of the proposed defenses. We have the following findings from our experimental results:

1. With disjoint training and evaluation sets, our approach can detect the confirmed malicious web servers (i.e., IPs) involved in the Novidade with high precision.
2. On top of the previous approach and with manual confirmation, our graph-based approach can identify additional, compromised or malicious DNS and web servers from the dataset that were not published in the original blog post.
3. When applying our graph-based approach on unlabeled browsing records, we can identify anomalous web servers with high precision.

This paper is organized as follows: Section 2 introduces our approach to address the research problem. Section 3 discusses the dataset we used in our experimentation. Section 4 demonstrates the results from our experiments. Section 5 illustrates prior research in DNS hijacking detection. Section 6 summarizes our findings and concludes the paper.

## 2. METHODOLOGY

In this section, we introduce an unsupervised method, which is based on community detection, and then consider two semi-supervised approaches including a graph embedding-based method which builds upon the node2vec algorithm and belief propagation.

### 2.1. Unsupervised Defense

Our community detection-based unsupervised method is applicable to the scenario where we do not have any labels. In other words, we do not any know compromised or malicious DNS or web server IPs that are used to mount an attack. We have the following goals when designing our unsupervised method:

[*] The first author performed the work while at Microsoft Research.

1. The method should be parallelizable and efficient since: a) there are many domains on the Internet, b) each domain on the Internet may own many web servers, and c) there are many DNS servers on the Internet.
2. The method should be accurate in detecting attack IPs.

To fulfill the first goal, we propose to design a method that can be executed for each domain independently. Then, we can parallelize our method for each domain when deploying it in practice. To reach the second goal, our first intuition is that the attack IPs are usually "novel" for the victim domain. In other words, attack IPs do not frequently serve as the web server IPs for the victim domain in the past. Our second intuition is that the attack IPs are usually connected. For instance, when launching the attack for a victim domain, an attacker will either compromise a DNS server or deploy a new DNS server and use it to translate the victim domain to an attacker-controlled web server IP. Based on these intuitions, we propose a community detection-based method to detect attack IPs. In particular, we first construct a *base graph* $G_c$ that can capture the relationship between DNS server IPs and web server IPs for the victim domain. Specifically, each DNS server IP (or web server IP) represents a node in the graph and there is an edge between a DNS server IP and a web server IP if the DNS server IP redirects the victim domain to the web server IP. We construct the bipartite graph by leveraging historical DNS resolution data for the victim domain. Then, given a new DNS resolution graph $G_a$, which we call an *attack graph*, that may contain attack IPs, our goal is to identify IPs that are most likely to be attacks IPs based on $G_c$. Our idea is to divide $G_a$ into multiple communities. Then, we assign a community score for each community and this community score is further used to assign the score to each IP in the community.

**Dividing $G_a$ into communities.** Our first step is to divide $G_a$ into multiple communities by a community detection algorithm. For simplicity, we use $\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_\gamma$ to denote these individual communties. In this work, we use the Louvain method [7] since it is efficient. Roughly speaking, given a graph, it aims to find a division that maximizes the modularity, which measures the density of edges within communities compared to edges between communities. Note that our framework is general and any community detection method can be adopted.

**Assigning a community score.** Given the communities in the previous step, we assign a community score for each community. For simplicity, given a DNS server IP or a web server IP in graph $G_a$, we say the IP is a "novel" IP if it does not appear in $G_c$. Based on our intuition that the attack IPs are "novel" for the victim domains, we consider the following factors:

- **F-I:** Given base graph $G_c$ and a community $\mathcal{C}_i$, we use $w_i$ to denote the fraction of "novel" web server IPs in $\mathcal{C}_i$ .
- **F-II:** An attacker can either compromise an existing DNS server or deploy a new DNS server to redirect the victim domain to an attacker-controlled web server. Therefore, given base graph $G_c$, we use $d_i$ and $s_i$ to denote the fraction of DNS server IPs and "novel" DNS server IPs that redirect the victim domain to a "novel" web server IP.

Given these two factors, we define the following reputation score for a community $\mathcal{C}_i$:

$$RC(\mathcal{C}_i) = \frac{1}{1+e^{-w_i}} \cdot \frac{1}{1+e^{-s_i-d_i}}. \quad (1)$$

**Estimating a reputation score for each IP.** Given an arbitrary web (or DNS) server $IP \in G_a$, we can compute its reputation score as follows:

$$RP(IP) = \sum_i RC(\mathcal{C}_i) \cdot \mathbf{I}(IP \in \mathcal{C}_i), \quad (2)$$

where $\mathbf{I}$ is the indicator function. We note that other information like the traffic of an IP can also be incorporated into our framework. Specifically, we define the traffic score of an IP as the ratio between the traffic associated with the IP and the overall traffic of the entire domain. For simplicity, given an arbitrary web (or DNS) server $IP$, we use $T_{IP}$ to denote the traffic score. Then, we can define the new reputation score for the web (or DNS) server $IP$ as follows:

$$RP(IP) = \sum_i RC(\mathcal{C}_i) \cdot \frac{1}{1+e^{T_{IP}}} \cdot \mathbf{I}(IP \in \mathcal{C}_i). \quad (3)$$

Intuitively, an IP tends to have a larger reputation score if it has a smaller traffic score. In other words, it is more likely to be an attack IP in this case.

**Ranking IPs based on Reputation Scores.** Given the reputation score of each IP in the graph, we can rank these IPs based on their scores. Then among all IPs that we are interested in, we can predict the Top-$N$ IPs that have the largest scores as attack IPs.

### 2.2. Semi-supervised Defenses

#### *2.2.1. Node2vec*

We next develop a semi-supervised method based on node2vec [5] if we have some groundtruth attack IPs. Similar to the proposed unsupervised method, our semi-supervised method can also be applied to each domain such that it is scalable and efficient. Node2vec is a graph representation learning algorithm which can learn low-dimensional representations for nodes in the graph. Such low-dimensional representations contain the graph structure information which can be used for many downstream tasks, e.g., link prediction and node classification. We omit the details of the node2vec due space limitations. Similar to our unsupervised method, we use a graph $G_a$ to model the relationships between web server IPs and DNS server IPs for the victim domain. Specifically, each web server IP (or DNS server IP) represents a node in the graph and there is an edge between a DNS server IP and a web server IP if the DNS server IP redirects the victim domain to the web server IP. Our goal is to detect attack IPs in $G_a$. Given a graph $G_a$, an IP address $IP \in G_a$, and the node2vec algorithm $\mathcal{A}$, we have the following:

$$\mathcal{A} : IP \longrightarrow \mathbb{R}^d. \quad (4)$$

In other words, the algorithm $\mathcal{A}$ maps an arbitrary $IP$ to a low-dimensional embedding vector.

**Estimating a reputation score for each IP.** Similar to our unsupervised method, our intuition is that the attack IPs are usually connected. The reason is that an attacker needs to compromise some DNS servers or deploy some new DNS servers to redirect the victim domain to an attacker-controlled web server. Therefore, given an arbitrary $IP$, we propose to estimate its reputation score by computing its similarity with the groundtruth attack IPs. Formally, given an arbitrary $IP$ and a set of $K$ groundtruth attack IPs, which are denoted as $\mathcal{T} = \{IP_1, IP_2, \cdots, IP_K\}$, we can compute the reputation score for the given $IP$ as follows:

$$RP(IP) = \frac{\sum_{i=1}^{K} Sim(\mathcal{A}(IP), \mathcal{A}(IP_i))}{K}, \quad (5)$$

where $Sim$ measures the similarity between two embedding vectors $\mathcal{A}(IP)$ and $\mathcal{A}(IP_i)$. We note that any similarity metric can be used in the equation above. In this work, we use the *cosine similarity* function which measures the angle between two embedding vectors.

**Ranking IPs based on reputation scores.** Given these reputation scores and the set of IPs that we are interested in, we can predict the Top-$N$ IPs that have the largest scores as attack IPs.

#### 2.2.2. *Belief Propagation*

Recall that we use a graph $G_a$ to model the relationships between web server IPs and DNS server IPs for the victim domain. Then, given such graph $G_a$ and a set of groundtruth attack IPs, we can also leverage belief propagation [6] to obtain a reputation score for each remaining IP in the graph. For simplicity, we use $\mathcal{B}$ to denote the belief propagation algorithm. Then, given a set of IPs $\mathcal{T} = \{IP_1, IP_2, \cdots, IP_t\}$, which contains both benign IPs and attack IPs, the reputation score for a given $IP$ is as follows:

$$RP(IP) = \mathcal{B}(G_a, \mathcal{T}, IP). \tag{6}$$

We adopt the SybilScar [8] belief propagation approach in this work. The reason is that it is more accurate and efficient compared with the original belief propagation algorihm. Similarly, we can also rank the reputation scores and predict the Top-$N$ IPs that have the largest scores as attack IPs.

## 3. DATA

To evaluate the performance of the proposed methods, we collect historical network traffic from Microsoft Defender SmartScreen for the compromised IP addresses used in the Novidade campaign and identifed two domains which we call Novidade Domain 1 and Novidade Domain 2. From the SmartScreen logs, we can get the timestamp, domain and its associated DNS server IP and web server IP from real-world traffic. For each domain, we construct a bipartite graph based on the DNS server and web server IPs. In the unsupervised method, the base graph and attack graph are generated separately using a different timeframe's traffic. We receive the label for the attack DNS and web server IPs from the published Novidade campaign reports [4] and manual analysis from Microsoft security analysts.

To improve the data quality, we conducted two steps. 1) Remove the DNS server and web server IPs which belongs to multicast, private, reserved and local IPs from the traffic telemetry that are associated with the compromised domain. 2) Extract the largest connected component for the bipartite graph to remove the unconnected nodes. After those steps, the graph details for the two compromised domains include:

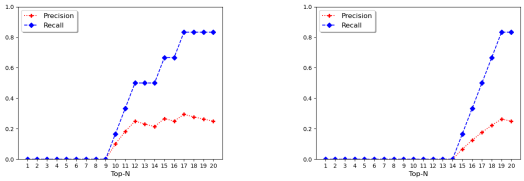
1. *Novidade Domain 1*
   (a) Base graph $G_c$: Network traffic before the attack 09/01/2018 - 11/30/2018
   (b) Attack graph $G_a$: Network traffic during the attack 12/01/2018 - 01/31/2019
      i. 1,814 nodes (164 web servers and 1,650 DNS servers) and 3,390 edges
      ii. 6 attack DNS IPs, and 4 malicious web server IPs used in the pharming attack
2. *Novidade Domain 2*
   (a) Base graph $G_c$: Network traffic before the attack 09/01/2018 - 11/30/2018
   (b) Attack graph $G_a$: Network traffic during the attack 12/01/2018 - 01/31/2019
      i. 1,985 nodes (414 web servers and 1,571 DNS servers) and 3,057 edges
      ii. 6 attack DNS IPs, and 1 malicious web server IPs used in the pharming attack

## 4. NUMERICAL EVALUATION

We provide an evaluation of the proposed methods for the task of detecting compromised DNS and malicious web servers in this section. We begin by providing a summary of the implementation steps used to conduct these experiments. Next, we consider the unsupervised learning case using the community detection algorithm. Two semi-supervised approaches, including belief propagation and node2vec, are then used to detect malicious web servers.

**Implementation.** The algorithms were implemented in either Python or C++. For the proposed community detection variant, we modified the Louvain algorithm which is implemented in Python [9]. Belief propagation is implemented in C++ [10]. The node2vec algorithm is implemented in Python [11] using the Tensorflow deep learning backend [12], and the experiments were run on an Azure virtual machine (VM) which included an NVIDIA P100.

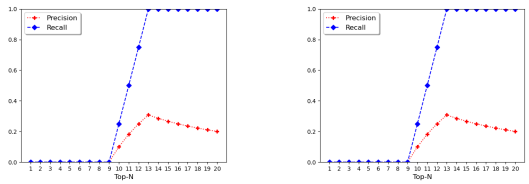
**Community Detection.** Because it is unsupervised, we can detect both compromised *DNS servers* in Figure 1 as well as compromised *web servers* in Figure 2. We compare the results for including the proposed traffic term in Figure 1a compared to the standard community detection algorithm in Figure 1b. In the figures, the precision and recall are plotted for increasing values of $N$ (i.e., Top-N) where $N$ indicates the number DNS or web servers with the largest reputation score (Eq (2) without traffic and Eq (3) with traffic). These figures indicate that including the proposed traffic term provides significant benefit to the analyst - including the traffic term begins to increase the precision and recall beginning with $N = 10$ whereas the values start to increase starting at $N = 15$ without the traffic term. The highest precision is reached in relatively short lists with $N = 16$ where the traffic term is included and $N = 18$ without the traffic term.




(a) With Traffic (b) Without Traffic

**Fig. 1**: Community Detection: Precision and recall for DNS servers using the Novidade Domain 1.

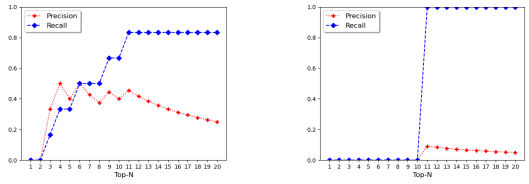
Next, Figure 2 considers the detection of compromised or malicious web servers which are the final destination of the attack using the community detection algorithm. We observe that including the traffic term in Figure 2a offers no improvement in the precision or recall compared to the standard community detection algorithm in Figure 2b. For both approaches, the precision and recall begin to improve with $N = 10$, and the precision reaches a value of 1.0 at $N = 13$.




(a) With Traffic (b) Without Traffic

**Fig. 2**: Community Detection: Precision and recall for web servers using in the Novidade Domain 1.

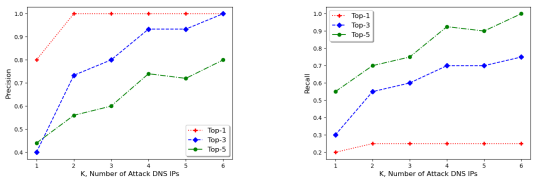
Figure 1 and 2 analyze the performance of the community detection algorithm for DNS and web servers using the Novidade Domain 1. Our data also includes compromised DNS and web servers from the Novidade attack which use the Novidade Domain 2. The performance of the community detection algorithm is provided for the DNS servers in Figure 3a and the web servers in Figure 3b. The precision and recall begin to increase for smaller values of $N$ for both DNS servers (Figure 3a) and web servers (Figure 3b) compared to the Novidade Domain 1.




(a) DNS Servers (b) Web Servers

**Fig. 3**: Community Detection: Precision and recall for DNS servers (a) and web servers (b) using the Novidade Domain 2 and including the traffic term.

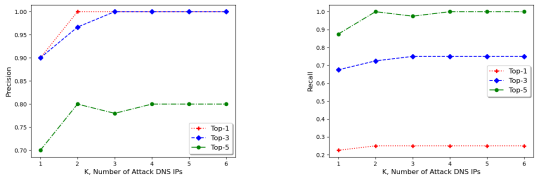
**Belief Propagation.** Next, we consider a semi-supervised approach where we have a few labeled DNS server nodes, and we would like to discover additional compromised or malicious web servers based on belief propagation [6]. The precision and recall for this task are provided in Figures 4a and 4b, respectively, for increasing numbers of known and labeled DNS servers $K$ in the Novidade Domain 1. We plot these results for the Top-1, -3, and -5 ranked lists based on the reputation score (Eq (6)). These results indicate that the precision and recall both tend to increase as we reach the maximum value of the 6 known compromised DNS servers using the Novidade Domain 1.




(a) Precision (b) Recall

**Fig. 4**: Belief Propagation: Precision and recall for web servers using the Novidade Domain 1.

**Node2vec.** Finally, we utilized the node2vec algorithm [5], again in a semi-supervised manner, to detect compromised or malicious web servers based on DNS servers which are known to be compromised. The precision and recall for the Top-N reputation scores (Eq (5)) are depicted in Figures 5a and 5b, respectively, for the Novidade Domain 1. Comparing the results in Figure 5 for the node2vec approach with those from Figure 4 for belief propagation, we see that node2vec offers superior performance for this attack.




(a) Precision (b) Recall

**Fig. 5**: Node2vec: Precision and recall for web servers using the Novidade Domain 1.

## 5. RELATED WORK

Many DNS hijacking attacks [13, 14, 15, 16, 17, 18, 19, 20] have been proposed. For instance, Stamm [16] proposed the Drive-by Pharming attack where an attacker can change the DNS server settings when a victim visits the attacker-controlled web page. Akritidis et al. [19] showed that an attacker can redirect users to spoofed web sites via providing free Internet access by malicious wireless routers. Karlof et al. [17] proposed dynamic pharming attacks which exploit DNS rebinding vulnerability in browsers and the name-based, same-origin policy to hijack the victim's authenticated session.

To mitigate the attacks, many defenses [21, 22, 23, 24] have been proposed in the literature. Cao et al. [21] proposed maintaining a white-list of user's all familiar Login User Interfaces of web sites to mitigate pharming attacks, which is not scalable and may negatively impact user experience. Gastellier-Prevost et al. [23] proposed combining IP checks and web page content analysis to detect pharming attacks. However, it requires the knowledge about the IP address. DNSSEC [2], which leverages digital signatures for the end user and the DNS server, has also been used to mitigate the attacks. However, it has not been widely deployed at this time [3]. All above defenses are different from our proposed graph-based defenses.

## 6. CONCLUSION

DNS hijacking attacks pose severe security and privacy threats to users both at home and when connecting to the internet using public wifi. This work demonstrates the effectiveness of both unsupervised and semi-supervised defenses for detecting compromised DNS and web servers via the Novidade malware campaign, which is a real-world DNS hijacking attack. Both approaches can be utilized for other campaigns where the attacker utilizes compromised DNS servers. The community detection defense, an unsupervised defense, can be used to discover previously undetected attacks. In addition, the community detection algorithm can generalize to different domains. Semi-supervised defenses can improve the performance if some comprised DNS servers have been previously labeled. For the Novidade campaign, the deep learning node2vec algorithm offers better performance than belief propagation.

## 7. REFERENCES


[1] Paul V Mockapetris, "Rfc1034: Domain names-concepts and facilities," 1987.

[2] Roy Arends, Rob Austein, Matt Larson, Dan Massey, and Scott Rose, "Dns security introduction and requirements," Tech. Rep., RFC 4033 (Proposed Standard), 2005.

[3] Wilson Lian, Eric Rescorla, Hovav Shacham, and Stefan Savage, "Measuring the practical impact of dnssec deployment," in *Proceedings of the 22nd USENIX Conference on Security*, USA, 2013, SEC'13, p. 573–588, USENIX Association.

[4] Joseph C Chen, "Exploit kit novidade found targeting home routers," `https://www.trendmicro.com/en_us/research/18/l/new-exploit-kit-novidade-found-targeting-home-and-soho-routers.html`, 2018.

[5] Aditya Grover and Jure Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[6] Judea Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*, Elsevier, 2014.

[7] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, pp. P10008, 2008.

[8] Binghui Wang, Le Zhang, and Neil Zhenqiang Gong, "Sybilscar: Sybil detection in online social networks via local rule based propagation," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 2017, pp. 1–9.

[9] Thomas Aynaud, "Louvain community detection," `https://github.com/taynaud/python-louvain`.

[10] Binghui Wang, "Sybildetection," `https://github.com/binghuiwang/sybildetection`.

[11] StellarGraph, "Stellargraph: Software for network graph analytics," `https://github.com/stellargraph/stellargraph`.

[12] Google, "Tensorflow," `https://www.tensorflow.org/`.

[13] Gunter Ollmann, "The pharming guide," *Retrieved October*, vol. 2, pp. 2005, 2005.

[14] Venugopalan Ramasubramanian and Emin Gün Sirer, "Perils of transitive trust in the domain name system," in *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, 2005, pp. 35–35.

[15] Alex Tsow, Markus Jakobsson, Liu Yang, and Susanne Wetzel, "Warkitting: the drive-by subversion of wireless home routers," *Journal of Digital Forensic Practice*, vol. 1, no. 3, pp. 179–192, 2006.

[16] Sid Stamm, Zulfikar Ramzan, and Markus Jakobsson, "Drive-by pharming," in *International Conference on Information and Communications Security*. Springer, 2007, pp. 495–506.

[17] Chris Karlof, Umesh Shankar, J Doug Tygar, and David Wagner, "Dynamic pharming attacks and locked same-origin policies for web browsers," in *Proceedings of the 14th ACM conference on Computer and communications security*, 2007, pp. 58–71.

[18] Joe Stewart, "Dns cache poisoning-the next generation," 2003.

[19] Periklis Akritidis, Wee-Yung Chin, Vinh The Lam, Stelios Sidiroglou, and Kostas G Anagnostakis, "Proximity breeds danger: Emerging threats in metro-area wireless networks.," in *USENIX Security Symposium*, 2007.

[20] Vasile C Perta, Marco V Barbera, Gareth Tyson, Hamed Haddadi, and Alessandro Mei, "A glance through the vpn looking glass: Ipv6 leakage and dns hijacking in commercial vpn clients," *Proceedings on Privacy Enhancing Technologies*, vol. 2015, no. 1, pp. 77–91, 2015.

[21] Ye Cao, Weili Han, and Yueran Le, "Anti-phishing based on automated individual white-list," in *Proceedings of the 4th ACM workshop on Digital identity management*, 2008, pp. 51–60.

[22] Hao Yang, Eric Osterweil, Dan Massey, Songwu Lu, and Lixia Zhang, "Deploying cryptography in internet-scale systems: A case study on dnssec," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 5, pp. 656–669, 2010.

[23] Sophie Gastellier-Prevost, Gustavo Gonzalez Granadillo, and Maryline Laurent, "A dual approach to detect pharming attacks at the client-side," in *2011 4th IFIP International Conference on New Technologies, Mobility and Security*. IEEE, 2011, pp. 1–5.

[24] Pratik Satam, Hamid Alipour, Youssif Al-Nashif, and Salim Hariri, "Dns-ids: Securing dns in the cloud era," in *2015 International Conference on Cloud and Autonomic Computing*. IEEE, 2015, pp. 296–301.