Anomaly Detection for DNS Servers Using Frequent Host Selection

Akira Yamada†‡, Yutaka Miyake†, Masahiro Terabe†, Kazuo Hashimoto† and Nei Kato†
† Network Security Laboratory, KDDI R&D Laboratories Inc., Saitama, Japan.
‡ Graduate School of Information Sciences, Tohoku University, Miyagi, Japan.
{yamada.akira,miyake}@kddilabs.jp

Abstract

DNS is one of the internet’s fundamental building blocks, used by various applications such as web and mail transfer. Therefore, monitoring DNS traffic has potential to detect host anomalies such as spammers and infected hosts in a network. However, previous works assume a small number of hosts or target on domain name anomalies, so that they cannot be applied to a large-scale networks due to performance issues. A large number of hosts and long-term tracing consume computational resources and make real-time analysis difficult. In this paper, we propose anomaly detection for DNS servers using frequent host selection, which selects only potential hosts and does not depend on the number of hosts. We evaluate the proposed system using DNS traffic for 6 months of tracing, and show that the system can feasibly handle hosts in the dataset and detect anomalies, such as mail servers suffering from spam and DNS servers are configured incorrectly.

1. Introduction

DNS (Domain Name System) is a fundamental protocol for the other applications, such as web or mail, in the internet. For example, URL, which indicates the location of a web resource, contains a domain name string. When a host accesses to a web resource, the host has to resolve the IP address of the name. An e-mail address is composed of a user name and a domain name, which is used by the mail exchanger. Therefore, DNS queries are closely related to hosts’ behavior, such as web browsing or sending e-mails.

Since the volume of traffic is increasing year by year, it is becoming difficult to monitor individual hosts. DNS is useful for monitoring host anomalies in a network management. Alternative methods that monitor only DNS traffic rather than whole traffic have been proposed. Jung et al. proposed that DNS traffic be available to observe anomalies of SMTP clients [1].

On the other hand, a DNS server itself is also involved in attacks, such as a DDoS (Distributed Denial of Service) attack and a cache poisoning[2] attack. DNS is a protocol on UDP, which the source IP address can be spoofed by an attacker so that the attacker can exploit DNS. For example, a DNS server can be a reflector of a DDoS (Distributed Reflected DoS) attack [3] queried snooped packets with the targeted host.

In this paper, we target on individual host anomalies, which include both internal/external and clients/servers, monitoring DNS servers in a network. There are previous works [4], [5], [6] that detect host anomalies monitoring DNS servers; however, these proposals cannot work well in a network where many hosts are observed or for long-term tracing. Since they record data for every host, the amount of computational resource required depends on the number of hosts.

There are some other researches that target DNS servers in a large network [7], [8], [9], [10] however they do not target individual hosts but domain names. They analyze statistics of queries, and find out abnormal domain names controlled by an attacker. For example, they detect domain names used by a bot network, which is a network of compromised PCs, and employ DNS resource records to control them.

In this paper, we propose anomaly detection for DNS servers using frequent host selection. The proposed system selects only potentially abnormal hosts, using frequent host selection [11], [12]. Therefore, it is applicable to a network where a million of hosts exist. The fundamental idea is that hosts send large numbers of queries and tend to involve anomalies, which affect the network more than other hosts.

To evaluate the proposed system, we applied it to a dataset captured at a real DNS server for 6 months. In the network, we observed more than about 1,000,000 hosts during the evaluation period. As a result, we detected several anomalies, for example SMTP servers suffering from spam mail, and misconfigured DNS servers.

This paper is organized as follows: Sec. 2 gives a brief introduction to the DNS protocol and related works. Sec. 3
presents the proposed system. Sec. 4 evaluates the proposed system and discusses its accuracy. Finally, Sec. 5 concludes.

2. Background

2.1. Mechanism of DNS

DNS is a mechanism to administer resource records on hierarchical and distributed servers, originally maintaining records of IP addresses for domain names. Now DNS has been extended to deal with various kinds of records, such as IP addresses, domain names, mail servers, public keys, and digital signatures. Moreover, DNS is employed by contents delivery networks \(^1\) and is used as an interface to check spammers' IP addresses as DNSBL (DNS Blacklist).

The resource records of DNS are managed by authoritative servers hierarchically. The root authoritative severs delegate parts of administrative domains to TLD (Top Level Domain) authoritative servers, and these TLD servers delegate parts of domains to lower servers. Therefore, a DNS client queries these servers recursively from the root to the lower servers.

There are 2 types of DNS servers: the authoritative and the cache server. Most clients query a cache server, which is located at each organization’s network. The cache server substitutes the clients' queries to authoritative servers, and caches the answers until the TTL (Time to Live) of records expires. On the other hand, an authoritative server manages one or more administrative domains, and responds records about the domains to the cache server.

Fig. 1 illustrates a sequence for resolving an IP address related to a domain name. When a client queries a cache server about a domain name “www.example.com,” (1) the server queries authoritative servers until the client’s query is resolved (2) (3) (4), and responds to the client with the answer (5). At first, the cache server queries to the “.” root authoritative server, and then the “com” and “example” authoritative servers recursively. In this case, the authoritative server of “example.com” administers the record of “www.example.com” and responds “192.168.1.1.”

2.2. Related Works

There are several related works that detect anomalies by monitoring a DNS server. White et al. showed that DNS is available for monitoring anomalies in a network \([4], [13]\). They relied on correlation between DNS’s and the other protocol’s traffic in an enterprise network, and detected the propagation of worms. However, it needs the other protocols’ traffic also, and cannot detect anomalies without other traffic.

Matsumoto et al. and other successors proposed host anomalies detection monitoring only DNS servers \([5], [6]\). Although the approaches are similar to our proposal, their works cannot support a DNS server where large amount of hosts exist. Because they record hosts’ behaviors individually in a database, they require computational power and memory depending on the number of hosts.

Alternatively, Ishibashi et al. proposed mass-mailing host detection monitoring the ISP's DNS server \([14]\). Although the scheme detects malicious hosts, it uses a list of spammers’ domains and addresses. The scheme has to maintain the list, and is vulnerable to domain tricks such as fast flux or double flux domain \([15]\). In addition, the scheme cannot detect other anomalies such as cache poisoning, propagation of worms and so on.

There are several proposals \([7], [8], [9], [10]\) that target large network. However, they detected domain name anomalies rather than individual host anomalies. These approaches analyze statistics of domain names, and detect domains controlled by malware such as viruses, or bot networks.

3. Anomaly Detection for DNS Servers Using Frequent Host Selection

3.1. Overview of Proposed System

In this paper, we target the detection of not a single attack but various kinds of host anomalies monitored at DNS servers. Previous works \([4], [5], [6]\) are not applicable to long term tracing or a network where a number of hosts exist, because a number of hosts consume computational resources, and make real-time monitoring difficult.

We propose anomaly detection for DNS servers using frequent host selection, where potential hosts are selected and only these statistics are recorded. The basis of the proposal is that a host sending a number of queries tends to involve anomalies, which affects a network more than others.

Fig. 2 shows an overview of the proposed system. (1) First, the proposed system extracts packet parameters by protocol analysis. (2) After that, the system selects frequent hosts and records their statistics in a buffer. Because the buffer contains only data of frequent hosts, the required buffer size is smaller than data for all hosts. (3) The system

---

reads the data in the buffer periodically, such as every hour, or minute, and inspects each host’s behavior.

3.2. Protocol Analysis

At protocol analysis, the system extracts protocol parameters from each packet. Although DNS is a protocol on both TCP and UDP, most queries and responses are on UDP. For this reason, we target DNS query/responses on UDP packets. Tab. 1 lists the set of parameters that we extract from a packet. The names of the parameters follow as a definition of a network protocol analyzer, Wireshark [16].

A DNS query contains a pair of a name and its type; the response contains the query and answers, which are multiple pairs of a name and its type. Note that dns.resp.type is recorded if the packet is a response. If the parameters do not exist in a packet, “NA” is set as the value. In case of multiple answers in a response, the type of the first resource record in the answer section is set as the value.

3.3. Frequent Host Selection

There are plenty of algorithms [11], [12] that extract frequent hosts from a datastream. Because of easy implementation, we employ a modified FIFO (First in First out) algorithm. FIFO is a memory queue, in which the first entry is first deleted. In order to record statistics of frequent hosts, we modify the FIFO algorithm.

When a sequence of hosts is pushed into a queue, it is highly possible that frequent hosts remain in the queue. As the management of hosts in the queue, the frequency of their hosts is also recorded as one of the statistics. When a host is popped out from the queue, its statistics are deleted.

Fig. 3 illustrates a function of the modified FIFO algorithm. When host \( c \) is pushed into queue \( A \), its frequency count[c] starts to be recorded. At that time, the number of host \( c \) in queue \( A \) is also recorded in qcount[c]. The input of the function is host \( p \); its parameters count[p] and qcount[p] are updated during the function. Host \( q \) is a host popped out from queue \( A \).

3.4. Host Statistics

At frequent host selection, we record not only the frequency but also other statistics listed in Tab. 2. The number of query types a, ptr, mx, ns and o, and error responses error are useful for describing the hosts’ behavior. If a DNS client works as MTA (Mail Transfer Agent), the client asks MX to resolve the IP addresses of the other MTAs. Alternatively, it is expected that a host that receives many errors is involving some anomalies.

In addition to the above statistics, we add 4 statistics to categorize hosts; \( tgyr, treq, rgyr \) and \( resrs \) are the number of transferred/received queries/responses. As we describe in Sec. 2.1, observed hosts are categorized into 3 types: client, cache and authoritative server. \( tgyr \) and \( resrs \) are observed for a host, \( rgyr \) and \( trex \) are observed for an authoritative server, and \( tgyr, treq, rgyr \) and \( resrs \) are observed for a cache server. Using the 4 statistics, we can identify the type of host.

These statistics are recorded for each host individually. A set of statistics for host \( c \) is \( c.total, c.tgyr, c.treq, \cdots \).
Table 2. Statistics for Anomaly Detection.

<table>
<thead>
<tr>
<th>Statistics</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>total</td>
<td>The number of queries/responses for a host.</td>
</tr>
<tr>
<td>tqr, tres, rqr, rres</td>
<td>The number of transferred/received queries/responses for a host.</td>
</tr>
<tr>
<td>a, ptr, mx, n, o</td>
<td>The number of each type for a host. The system records types “A,” “PTR,” “MX,” and “NS,” respectively. The other types are categorized into “Other” type.</td>
</tr>
<tr>
<td>error</td>
<td>The number of error responses for a host.</td>
</tr>
</tbody>
</table>

c-error. For example, when an MX query from host c is observed, c-total, c-tqr, and c-a are updated. In addition, we record 3 types of statistics: sum, square sum and sum for a period. Using sum and square sum, we calculate the mean and the standard deviation.

3.5. Anomaly Detection

The proposed system detects host anomalies by using the following 5 heuristics:
1) the number of queries/responses
2) balance of queries and responses
3) the number of MX queries
4) the number of PTR queries
5) the number of error responses

First, the number of queries/responses is useful for monitoring status of the host. Even if the number of queries depends on human activity, such as 24 or 7, there is little deviation for a long period. Second, in the protocol specification, the number of queries and responses is balanced. If a source IP address is spoofed by an attacker, who attempts cache poisoning [2], then 2 or more responses would be observed for 1 query.

As the type field in a query, we pick up PTR and MX. PTR is a type to resolve a domain name related to an IP address, so the portion of PTR queries is smaller than the other types. MX is a type to resolve mail exchanger corresponding to a mail address. Therefore, MX queries are efficient for measuring SMTP servers. If a compromised host turns into a spam sender, then the host starts to query MX. In addition, we monitor the error code, which indicates errors in a protocol sequence.

Using host statistics, the system can calculate parameters that describes the above 5 heuristics. After that, the system detects host anomalies based on a set of thresholds for the parameters. The thresholds are decided by the statistics of each parameter for a long period. This means that a host is regarded as an anomaly if the host has a statistically abnormal parameter.

The number of queries or responses: We set threshold \( T_{total} \) for the number of a host’s queries/responses for a time period as Eq. (1). \( \mu_{c, total} \) and \( \sigma_{c, total} \) represent the mean and the standard deviation of host c’s queries/responses respectively. c-total(t) is the number of queries/responses observed for time period t.

\[
T_{total} \leq \frac{c_{total}(t) - \mu_{c, total}}{\sigma_{c, total}} \quad (1)
\]

Balance of queries and responses: We set threshold \( T_{balance} \) for the difference between the number of queries and responses as Eq. (2). c-tqr(t), c-tres(t), c-rqr(t) and c-rres(t) represent the number of transferred or received queries or responses for time period t.

\[
T_{balance} \leq \frac{|c_{tqr}(t) - c_{tres}(t)| + |c_{rqr}(t) - c_{rres}(t)|}{c_{total}(t)} \quad (2)
\]

The number of PTR queries: For types of queries, we define 2 thresholds, rate \( T_{PTRRate} \) and deviation \( T_{PTR} \) as Eq. (3) and (4). \( \mu_{c, ptr} \) and \( \sigma_{c, ptr} \) represent the mean and the standard deviation of host c’s PTR queries. c.ptr(t) is the number of PTR host c queries for time period t.

\[
T_{PTR} \leq \frac{c_{ptr}(t) - \mu_{c, ptr}}{\sigma_{c, ptr}} \quad (3)
\]
\[
T_{PTRRate} \leq \frac{c_{ptr}(t)}{c_{total}(t)} \quad (4)
\]

The number of MX queries: We set thresholds \( T_{MX} \) and \( T_{MXRate} \) for the number of MX queries as Eq. (5) and (6). \( \mu_{c, mx} \) and \( \sigma_{c, mx} \) represent the mean and the standard deviation of host c’s queries. c.mx(t) is the number of MX that host c queries for time period t.

\[
T_{MX} \leq \frac{c_{mx}(t) - \mu_{c, mx}}{\sigma_{c, mx}} \quad (5)
\]
\[
T_{MXRate} \leq \frac{c_{mx}(t)}{c_{total}(t)} \quad (6)
\]

The number of error responses: We set threshold \( T_{Error} \) for the number of error responses. \( \mu_{c, error} \) and \( \sigma_{c, error} \) are the mean and standard deviation, and c.error(t) is the number of response for which RCODE is not 0 for time period t.

\[
T_{Error} \leq \frac{c_{error}(t) - \mu_{c, error}}{\sigma_{c, error}} \quad (7)
\]
\[
T_{ErrorRate} \leq \frac{c_{error}(t)}{c_{tres}(t) + c_{rres}(t)} \quad (8)
\]

4. Evaluation

4.1. Dataset

Tab. 3 shows a dataset that we used for the evaluation. The dataset contains 6 months of tracing captured at a gateway of a DNS server in an enterprise network. Both authoritative and cache DNS servers exist in the network. Compared to
Table 3. Dataset for Evaluation.

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007-10-01/2008-03-31</td>
<td>Packets (UDP port 53)</td>
</tr>
<tr>
<td>5,247,193,774</td>
<td>Average packets/hour</td>
</tr>
<tr>
<td>1,247,746</td>
<td>Average IP addresses/hour</td>
</tr>
<tr>
<td>7,296</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4. Distribution of Hosts.

Figure 5. The Number of Observed Hosts.

4.2. The Number of Hosts in the Dataset

Fig. 4 shows the distribution of DNS hosts for the first 24 hours. The x-axis is hosts sorted by frequency order; y-axis is frequency of the hosts and fraction of the hosts. The graph indicates that the distribution of hosts follows Pareto distribution, in which a few hosts dominate most portions. A total of 88.7% of queries/responses are transferred or received by only the top 30 hosts.

Fig. 5 shows that (a) the number of unique hosts in the dataset and (b) unique hosts selected by frequent host selection. The x-axis is time; y-axis is the cumulative sum of unique hosts. At frequent host selection, we select the top 30 IP addresses for an hour.

The number of unique IP addresses increase for 6 months constantly, and reaching about 1,000,000. If the system records statistics for all IP addresses, the data occupy 1,000,000 entries in the database. In order to update an entry in the database, it is required to seek from these entries. On the other hand, the number of selected IP addresses is \( \frac{1}{1000} \) of all IP addresses. Although it increases continually, the number of hosts is feasible to update these entries after 6 months.

In general, increase of observed IP addresses stops after a certain duration, because the total number of IP addresses is \( 2^{32} \). However, there is no indication that the increase is stopping in Fig. 5. This means that more IP addresses will appear in the network in the future. The reason is that attackers would have plenty of zombie PCs, and could dispose of the IP addresses after they are listed on the blacklist.

4.3. Thresholds

Fig. 6 illustrates the distributions of parameters for anomaly detection: (a) shows the number of queries/responses, (b) the balance between query and response, (c) the number of PTR queries, (d) the rate of PTR queries, (e) the number of MX queries, (f) the rate of MX queries, (g) the number of error responses, and (h) shows the rate of error responses.

In graphs (a), (c), (e), and (g), most values are spread around 0, because the parameters are based on standard deviation. The reason for the spike in the graph is that some host statistics are insufficient for calculating standard deviation. At frequent host selection, the system starts to record a host’s statistics after the host is selected. Due to the spike, these distributions are more flat than normal distribution, so we set thresholds \( T_{total}, T_{PTR}, T_{MX} \) and \( T_{Error} \) as 5.

In graphs (b), (d), (f) and (h), parameters are distributed between 0 and 1. Most values are concentrated around 0, because the portions of MX and PTR are smaller than A queries. Usually queries and responses are balanced in a normal DNS protocol. However, there are some hosts where the parameters are near 1. In order to detect such abnormal hosts, we set thresholds \( T_{balance}, T_{PTR \text{ Rate}}, T_{MX \text{ Rate}} \) and
Figure 6. Distribution of Parameters for Anomaly Detection.

Table 4. Details of Alerts.

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>Alerts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>5,469</td>
</tr>
<tr>
<td>PTR query</td>
<td>4,094</td>
</tr>
<tr>
<td>MX query</td>
<td>1,158</td>
</tr>
<tr>
<td>Error response</td>
<td>2,798</td>
</tr>
</tbody>
</table>

\( T_{\text{ErrorRate}} \) as 0.9.

4.4. Detection Accuracy

At first, we conducted an experiment using the thresholds defined in Sec. 4.3. A total of 1,141 IP addresses are selected by frequent host selection, and 45,947 alerts are given at the anomaly detection. Because the system executes anomaly detection every hour for each selected host, more than one anomaly is detected. Details of alerts are listed in Tab. 4.

In order to examine whether these alerts are true or false, we analyze their details. It is difficult to investigate all alerts deeply; we select 10 samples for each heuristic. We curve out all packets related to the sampled alerts, and determine whether the alerts are true or false.

Tab. 5 shows the number of true alerts. All suspect alerts were due to unusual queries; however, we could not discover all causes of the occurrence of these queries. We define true alerts as the cases where the causes are also discovered.

Table 5. True Positive Rate.

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>True alerts</th>
<th>Samples</th>
<th>TPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>7</td>
<td>10</td>
<td>0.70</td>
</tr>
<tr>
<td>PTR query</td>
<td>9</td>
<td>10</td>
<td>0.90</td>
</tr>
<tr>
<td>MX query</td>
<td>10</td>
<td>10</td>
<td>1.00</td>
</tr>
<tr>
<td>Error response</td>
<td>3</td>
<td>10</td>
<td>0.30</td>
</tr>
<tr>
<td>Total</td>
<td>54</td>
<td>80</td>
<td>0.68</td>
</tr>
</tbody>
</table>

4.5. Details of Anomalies

We investigate details of discovered anomalies: backscatter of spam, misconfigured DNS, and PTR scanning.

Spam backscatter: On receiving a spam message, a SMTP server scatters a flood of DNS queries. When a SMTP server receives a message, it queries PTR of the sending server. In addition, a SMTP server is equipped with anti-spam mechanism, which requires PTR lookups and references of DNSBL.

Usually, spammers send messages from zombie PCs, which are home or office PCs controlled by them. Therefore, spammers’ domains and IP addresses are not held by a cache server, and cause new queries. Furthermore, a spammer sends many messages at the same time from multiple domains and multiple IP addresses.

Most messages accepted by the SMTP server are returned to the original sender, spammer. Because the spammer sends messages to unknown users, the mail addresses are not
In the process of returning the messages to the sender, the SMTP server must query another record. Therefore, massive DNS queries are observed when a SMTP server suffers from a spam. Fig. 7 shows observed pairs of PTR lookups and DNSBL references.

**Misconfigured DNS**: Misconfigured DNS records cause a flood of queries. Fig. 8 shows an example of flooding queries. When a cache server queries a PTR record related to “124.226.114.0,” “ns.lzptt.gx.cn” and “ns.gxnnptt.net.cn” are given as the answer. However, authoritative DNS servers “ns.lzptt.gx.cn” and “ns.gxnnptt.net.cn” do not respond to the correct answer, but the delegated authoritative servers, which are themselves. For this reason, the cache server queries both servers repeatedly. There are many such DNS servers configured incorrectly, which causes floods of queries.

**PTR Scanning**: As a first step of attacking a network, an attacker probes whether a host exists or not on an IP address. PTR scanning is an efficient measure to identify hosts. Usually, hosts have a domain name; SMTP servers in particular must have a name. In querying the domain name related to an IP address, attackers can distinguish whether a host exists or not. In addition, a part of a domain name string usually has information on the host; a web server has a name that contains the string “www.”

Fig. 9 shows observed PTR scanning. An attacker queries an authoritative server managing a range of IP addresses 59.88.0.0/16. If the host of an IP address is assigned a name, the authoritative server leaks the name to the attacker.

**4.6. Discussion**

Before the evaluation, we anticipate that attacks such as cache poisoning, DoS (Denial of Service) attack and PCs infected by mass-mailing worms. However, we could find SMTP servers suffering from spam and incorrect configuration. In order to detect the former attacks, we should monitor more vulnerable networks rather than enterprise networks.

It is a large proportion that 836 out of a selected 1,114 IP addresses were alerted at least 1 time for the monitoring period. The reason is that the frequent hosts are potential IP addresses, which tend to involve anomalies. The system alerts not only attackers but also the related hosts such as victims or victims’ cache servers. In addition, anomalies such as spam mail or incorrect configuration happen very often.

In fact, all sampled alerts indicate suspicion of anomaly, and 68% of alerts indicate a true anomaly that we ascertain to be the root cause. In the randomly selected IP from all addresses, fewer than 1% of hosts involve an anomaly. This means that frequent host selection works well to extract potential host anomalies.
In Tab.5, a heuristic, error response $T_{Error}$, caused many false positives. We investigated the false positives and found out that a DNS server was not stable and was highly likely to send error responses. The accuracy could be improved using a blacklist or whitelist that contains hosts that frequently cause false positives or negatives.

As for future work, we are planning to examine the relationships between combinations of alert and accuracy or details of anomaly. Accuracy would be improved by combinations of alert, because most detected hosts are alerted by multiple heuristics. In addition, the details of anomaly may depend on combinations of alert, because the type of query highly depends on the application. For example, wrongly configured DNS servers are alerted by heuristics of PTR, balance, and frequency. All hosts alerted by heuristics of MX alerts involve spam anomaly such as sending or suffering from spam messages.

5. Conclusion

In this paper, we proposed anomaly detection by monitoring DNS servers where large number of hosts exist. The proposed system employs frequent host selection based on modified FIFO algorithm, so that it extracts only potentially abnormal hosts. Compared to record statistics for all individual hosts, the proposed system reduces required computational power and memory.

Using a dataset captured for 6 months at an enterprise network, we evaluated our system. We verified that the proposed system can handle more than 1,000,000 hosts and can detect several kinds of anomalies, such as mass-mailing hosts or wrongly configured DNS servers.

References


