Name Server Switching: Anomaly Signatures, Usage, Clustering, and Prediction

Aziz Mohaisen[1], Mansurul Bhuiyan[2], Yannis Labrou[1], and Hyoungshick Kim[3]

[1] Verisign Labs, Reston, VA, USA
[2] Indiana University—Purdue University, Indianapolis, IN, USA
[3] Sungkyunkwan University, Suwon, Republic of Korea

Abstract. There exists a significant number of domains that have frequently switched their name servers for several reasons. In this work, we delved into the analysis of name-server switching behavior and presented a novel identifier called "NS-Switching Footprint" (NSSF) that can be used to cluster domains, enabling us to detect domains with suspicious behavior. We also designed a model that represents a time series, which could be used to predict the number of name servers that a domain will interact with. We performed the experiments with the dataset that captured all `.com` and `.net` zone changing transactions (i.e., adding or deleting name servers for domains) from March 28 to June 27, 2013.

1 Introduction

The Domain Name System (DNS) is an essential component in the operation of the Internet. While domain names are easy to remember by human, Internet Protocol (IP) addresses are numerical labels for identifying network entities on the Internet. The process of finding an IP address associated with a domain name is called the DNS name resolution, and is the first step required for navigating on the Internet. For example, a user entering "`http://www.verisign.com`" in the browser would be unaware of complexities and procedures performed in translating the domain name `verisign.com` into the IP address (`69.58.181.89`) for the Verisign web server. The resolution process is recursive in nature, meaning that a request for an IP address of `verisign.com` will propagate to the authoritative name server through intermediaries until a resolution happens successful by reaching such authoritative name server. Figure 1 illustrates the process of DNS resolution for an example of `abc.com`.

In the DNS ecosystem, the life cycle of a domain starts with its registration under a top level domain (TLD; such as `.com`, `.net`, etc). TLDs are maintained by registries, and as part of the registration process a domain name is paired with a name server, which serves as the gate keeper of the domain name. Name servers, specially the authoritative ones, are a significant entity of a domain's operation as they will tell users where to look for the domain. A domain name may have one or more name servers to resolve it.

While it is natural to expect a one-to-one mapping of a domain name to a name server, it is often desirable to maintain multiple name servers for a domain name to facilitate reliability, availability, tolerance to failure, and geographical diversity, among other desirable performance metrics and features. For example, when multiple name servers are associated with one domain name, they will enable connectivity to the domain name by other users even when some of the name servers fail. Additionally, having multiple name servers can enable load balancing [1]: a DNS resolver aware of multiple requests for the same domain name will be able to intelligently distribute the incoming requests to individual servers, and to provide different translation for the same domain name, thus serving the contents of the resolved domain from arbitrarily many servers in a uniform fashion at fairly well-balanced loads. The same idea of load balancing when couple with the geographical location of the requesting users can be further utilized to enable geographical diversity: the requests for a domain name can be resolved to an IP address that is located closer to the user, based on the user's location. An example of a technology that utilizes DNS for enabling geographical diversity is the content distribution network (CDN) [2], which have seen a great adoption in delivering content to users more reliably and efficiently. To that end, using multiple name servers can improve reliability, scalability, and utility in IP networks.

Although multiple name servers can be associated with the same domain, this association does not imply any form of dynamics. The set of multiple name servers is usually assigned to a domain name at the registration time of the domain name, and an update of those name servers happens less often once the domain name is set up and operated. Natural causes for updating the name servers associated with the domain name can include transferring between service providers, retiring hardware used as the name server, among others, which all are naturally less frequent and hard to observe in a short term.

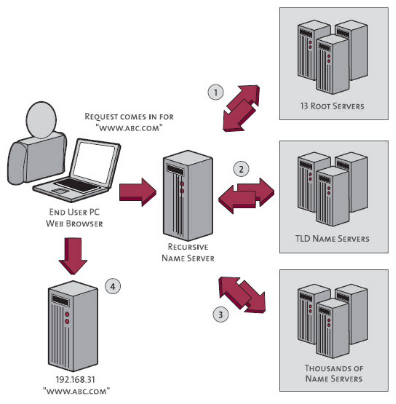


Fig. 1: DNS Resolution of `abc.com`

It is however noted that there exists a nontrivial number of domain names that perform frequent updates to their name servers, even by switching among name servers that belong to multiple service providers. Such transactions might not be reasonable in many situations because they demand interference with the existing DNS services. Accordingly, several studies in the literature were set out to understand this phenomenon and its implications. In [3], the author interprets name servers switching as a hiding mechanism of a domain's intended usage, and shows that domains with such behavior tend to display unsavory behavior, including the hosting of malware, pornography, and the sales of unauthorized pharmaceuticals drugs, among other illicit activities. In [4], researchers used the number of name servers of a domain within a period of time as a feature to develop a classifier for detecting malicious domain names. In [5], researchers developed an inference system to build proactive blacklist of domains where they used the name servers information of exiting blacklisted domains. They showed that future blacklisted domains tend to follow the same trend on name servers selection.

All of the prior work has looked at the aggregate feature of association between domain names and name servers to infer a better understanding of the use of domain names. A limiting aspect in the prior literature is that nobody had access to a continuous stream of data representing the dynamics of association between name servers and domain names over time. To this end, while motivated by the prior work in [3, 5, 4], we look deeper into the subject. Our unique position on the topic is facilitated by a unique data-point: we have considered not only the total number of name servers used by a domain or how much switching has happened within a period of time but also the evolution of the name servers associated with a domain name. For example, we looked at the changes in the number of name servers associated with a domain name: in a time slot t how many name servers are added to the pool of servers associated with a given domain name, and in time $t + 1$ how many name servers got deleted. We also try to understand how the sequence of addition and detection of name servers associated with a domain name proceeds over time.

We use this name server evolution of a domain to build an identifier called "NS-Switching-Footprint" (NSSF). Following [5], we hypothesize that NSSF can be used as a vital artifact to characterize domain names' intended usage. We argue that such an artifact can be used as a building block in a security system that associates the name servers to potentially check whether a domain name is malicious or benign. The NSSF is not only a feature that can be used to determine if a domain name is malicious or not, but can be further extended to highlight many intended uses of domain names, like advertising, traffic redirection, search engine optimization, etc. Our design of NSSF is intended to be robust, and is not limited to exact footprint matches. Rather, we use a partial match that makes comparison on various substrings within the fingerprint to achieve the same goal of understanding the intended characterization and usage of a domain name. To show the feasibility of NSSF, we experimentally analyzed the relationship between the characteristics of domains and NSSF and observed that domains of similar types (e.g., malicious domains and advertisement) tend to have similar footprints.

Building on preliminary findings on the power of NSSF, we also developed a prediction model that can estimate how many name servers a domain might interact with in the future given historical interactions. The prediction model serves two purposes. First, it enables us to identify a complete NSSF (within the certainty guarantees of the prediction model) for interactions that did not happen yet. Second, it enables us to probabilistically identify intended uses for domain names, using the predicted NSSF, before the intended use happen. This latter feature would enable proactive actions to be done in case, for example, of intended misuse of a domain name. Experimental results of the proposed prediction model unveil its power and potential use.

Contributions. First, we introduced the NSSF, a feature for characterizing the use of domain names based on the dynamics of association between the domain name and their name servers. Experimental results on `.com` and `.net` zone files show that the proposed feature is capable of capturing the intended use of various domain names. Second, we proposed a method for clustering domain names based on their NSSF structures, which captures their usages with applications to anomalies. Third, we examined the power of an-off-the-shelf method for predicting the NSSF.

Organization. The background is in §2, the proposed method for characterizing domain names by their name server switching patterns is proposed in §3. The observations on the proposed method for understanding domain name servers are in §4. Clustering of domain names using their NSSFs is presented in §5. A model for predicting NSSF is proposed in §6. The related work is in §7, and concluding remarks are in §8.

2 Background

In this section we will introduce preliminaries required for understanding the rest of the paper. First we elaborate on the interactions between domain names and name servers, and their book-keeping, while emphasizing on the terminology in that field. Then, we elaborate on time series data analysis used for predicting the intended use of domains.

2.1 Domain and Name Server Interaction

The domain name ecosystem consists of three entities: a registry, registrar, and registrant. The registrant is the entity that has the right to use the domain name. The registry is the organization responsible for maintaining a database of information about domain names and their name server mapping (that database is also called "registry"). Each TLD is associated with a registry, like VeriSign, which maintains such information about the TLD. Registrants and the registry are separated by a registrar, an entity that reserves domain names on behalf of registrants.

When a registrant wants to register a domain, one of the ICANN (Internet Corporation for Assigned Names and Numbers) accredited registrars of the registrant's choice creates a lease document with the consent of the relevant TLD registry (i.e., Verisign for `.com` and `.net`) for the intended use period. Upon the activation of the domain, the registry stores the DNS information of the domain in the corresponding DNS *zone file*. In the the zone file, all the authoritative name-servers of a domain are listed. In this work, we only considered `.net` and `.com` domains which in fact constitute almost 50% of all domains [6]. Any action that results in the addition or deletion of a domain name, name server, or the association of a name server with a domain name is called a *transaction* and is logged into the zone file. Atomically, there are three types of transactions: *add*, *update* and *delete* that are considered zone-impacting transactions. Actions that use those transactions are propagated in the zone file.

2.2 Time Series Data Analysis and Forecast

Time series analysis is a well-established field in statistics which provides systematic approaches to model data with time correlations. We focus in this paper on the (discrete) time domain approach as it is typically more appropriate for dealing with (possibly) non-stationary, shorter time series with a focus on predicting future values [7]. As we alluded to earlier, and discussed at length later, the problem of predicting the number of name-servers of a domain given the history of interactions can be modeled as a time series prediction problem. To this end, we specifically focus on the multiplicative models, represented by a systematic class called autoregressive integrated moving average (ARIMA) models [8]. These models assume that the observed data results from products of factors involving differential operators responding to a white noise input. The ARIMA model is a generalization of the more widely used autoregressive moving average models (ARMA), which found many applications in statistical process control [9], financial forecasting [10], biomedical dynamics modeling [11], and web traffic modeling and forecasting [12]. In this study, we focused on using the standard ARIMA [8] model for time series name-server prediction. Let X_t be a time series, where y is an integer index and X_t is a real number for t. The ARIMA model is defined as $Y_t = (1-L)^d X_t$, where Y_t is the predicted variable, L is the lag operator (or backshift; defined as an operator on the time series to produce the previous element), and d

is a multiplicity factor. For more details on the model and its operation, see [8]. Other techniques from the literature that we use as tools are *hierarchical clustering* methods. For a review of the technique and the different parameters in a similar context, see [13].

3 Proposed Measure of Switching Behavior

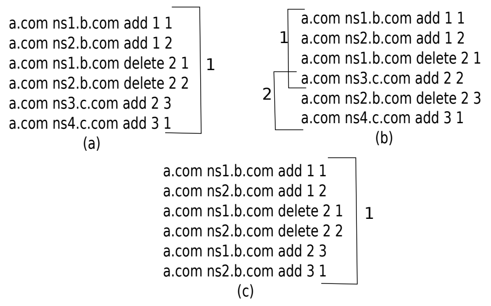
In this section we turn our attention to explaining the method used for characterizing the behavior of a domain name using name server switching.

3.1 Name Server Switching

To characterize the dynamics of name servers associated with domain names, we consider a discretizing process of the time domain. Let w be a window of events defined as a succession of at least one *add* operation followed by at least one *delete* operation followed by at least one *add* operation. Let a transition be defined as the set-theoretic difference of the set of name servers of a window (NSw) and the set of name servers of the previous window (NSw′). A name server switching occurs when the transition set is non-empty (i.e., $\mathrm{NSw}\setminus\mathrm{NSw}' \neq \phi$). The intuition is that by ignoring individual successive additions and deletions of name servers, and instead focusing on aggregate changes, we will capture significant changes in the state of a domain's NS providers.

Example. The above definition of switching is explained by a simple toy example shown in Figure 2, and discussed as follows.

In Figure 2(a) and Figure 2(c) there is only 1 window but in Figure 2(b) there are 2 windows, which can be manually vetted. At the end of each window we decide whether a switching of name server occurs or not. In Figure 2(a), the first two *add* operations cause the domain "`a.com`" to have two name servers. Then following two *delete* operations of the previously added name server cause domain "`a.com`" to have no name servers associated with it, but the last two *add* operations make the domain "`a.com`" associated with two new name servers than before, thus we consider this scenario as switching of name serves. At the end of the first window in Figure 2(a), the domain "`a.com`" has two servers.



```
a.com ns1.b.com add 1 1
a.com ns2.b.com add 1 2
a.com ns1.b.com delete 2 1
a.com ns2.b.com delete 2 2
a.com ns3.c.com add 2 3
a.com ns4.c.com add 3 1
```

```
a.com ns1.b.com add 1 1
a.com ns2.b.com add 1 2
a.com ns1.b.com delete 2 1
a.com ns3.c.com add 2 2
a.com ns2.b.com delete 2 3
a.com ns4.c.com add 3 1
```

```
a.com ns1.b.com add 1 1
a.com ns2.b.com add 1 2
a.com ns1.b.com delete 2 1
a.com ns2.b.com delete 2 2
a.com ns1.b.com add 2 3
a.com ns2.b.com add 3 1
```

Fig. 2: A toy example of the NS-Switching

In Figure 2(b), we can see that there are two windows. At the end of the first window, the domain "`a.com`" has two name servers where one is a new name server ("`ns3.c.com`") added to the previously allocated name servers in the set of "`a.com`" by the end of the first two *add* operations — hence one switching of name server is recorded. Then at the end of the second window, domain "`a.com`" has two name servers where one ("`ns4.c.com`") is new to what "`a.com`" had at the end of the first window, and hence the second switching name server is observed. Finally, in Figure 2(c), we only have one window. At the end of the window, domain "`a.com`" has two domains which are the same as it had at the beginning of the window, and hence no switching of name server is observed or recorded.

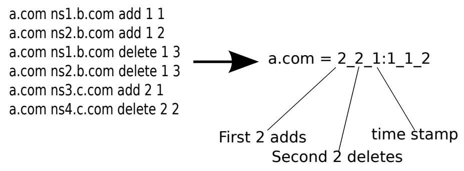
Since the concept of name servers switching is now made clear, we move on to the description of the name server switching footprint, which is the main feature used for identifying the use of domain names in the rest of the paper.

3.2 NSSF: Name Server Switching Footprint.

The NSSF is a domain name's unique identifier for characterizing the pattern of name servers switching over time. In Figure 3, we present the pseudo code for building the NSSF for a domain. In the footprint, we incorporate the number of name servers added or deleted along with the time period of these operations. Since we have a fixed number of time units of data (at the level of days) — in which name zone impacting transactions are observed and recorded, we set the length of each time period to one day. According to the algorithm in Figure 3, given a domain, the corresponding NSSF is generated as a string representing the number of additions, followed by the number of deletions of the name servers associated with the domain name, followed by a time index. Figure 4 illustrates a simple example on NSSF building.

```
# input = domain d
# output = Footprint of d
# T = Total time period = 90
**Build_NSSF**(domain $d$):
1. **for** $d$ exists in $t$ time period where $t \in (1, T)$:
2.     NSSF = concate(NSSF, concate(#ofAdded-NS(d),
           #ofDeleted-NS(d) , t,sep="_"), sep=":")
```

Fig. 3: Footprint Building Algorithm



```
a.com ns1.b.com add 1 1
a.com ns2.b.com add 1 2
a.com ns1.b.com delete 1 3
a.com ns2.b.com delete 1 3
a.com ns3.c.com add 2 1
a.com ns4.c.com delete 2 2
```

Fig. 4: NS-Switch Footprint(NSSF)

4 Experiments Observations

Next, we use the NSSF to detect anomalies in the domain name usages based on name server interactions. We rely in our study on a large corpus of name zone alerts associated with the registry of `.com` and `.net` which are operated by VeriSign.

Dataset. The dataset used in this study belongs to the registry operation of the `com` and `net` TLDs operated by VeriSign. As outlined in §2, upon the activation of a domain name the registry stores the DNS information of the domain in the corresponding DNS *zone file*. Verisign, as the registry of the `com` and `net` TLDs, has created the method called Domain Name Zone Alerts (DNZA) to capture all zone impacting transactions in a specific format in a special extension file called `.rzu` (rapid zone update). While the format and extension create a platform of such information to interested parties (for data usability), they capture an interesting aspect relevant to our study: the DNZA maintains an order of the transactions as they happen in reality, and log them in the format.

Using the DNZA files generated for the `com` and `net` TLDs, and for a given domain name, we can extract a completely ordered set of events that impact the zone file since the creation of the domain name until its retirement. For its operations, and at any point in time, VeriSign maintains DNZA files for the past 90 days, and makes it available for interested parties (through data agreements in place). We used this dataset during the 90 days covering the period between March 28, 2013 to June 27, 2013 in this study.

We only considered the domains that are registered within these 90 days since they are likely to be of interest and revealed their intended usages within that period of time. For that same period, we had approximately 31 million transactions, averaging about 350 thousand transactions per day, with 7.9 million unique domain names, and about 480 thousand unique name servers. These statistics are summarized, more precisely, in Table 1.

# of transactions	Avg transactions/day	# of domains	# of Name servers
31,586,839	350,964	7.9 mil	480K

Table 1: Statistics of DNZA dataset

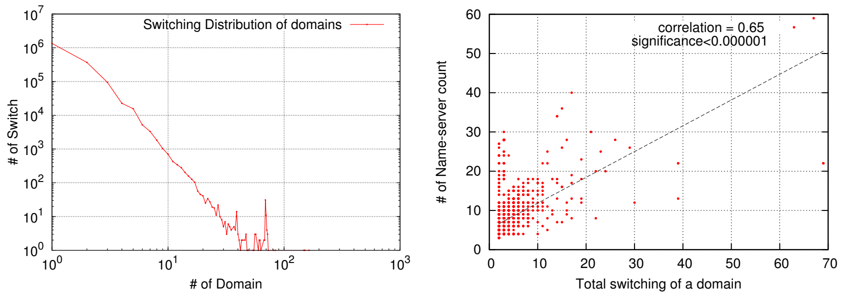


Fig. 5: (a) Log-log plot of switching. (b) Scatter plot of switching of all domain

Analysis of Name Server Switching. In the following, we first analyze the basic name server switching characteristics of domain names added to the zone of `.com` and `.net` in the 90 days covered by the data in this study. We ran our algorithm to compute the total number of switching of name servers for each domain in the DNZA dataset based on the definition in §3.1. We notice that 25% of all domains performed at least one NS switching. Figure 5(a) shows the switching distribution of domains, where the distribution exhibits a power law characteristic. Figure 5(b) shows the total number of switching versus the number of unique name servers of a domain name.

We also fitted a straight line to capture and demonstrated the positive correlation between the number of name servers and the count of overall switching associated with a domain. We also observed that most of the domains with a higher switching count tend to exhibit unusual behaviors and types as discussed in Section 1.

Table 2 shows some of the major findings by highlighting some examples of such domain names. For example, we see that "`amazingweb007.com`" is an adult dating website which does not follow mainstream dating website concepts; "`teknotigr.com`" is an empty blog that has a lot of

domain	Switchings	Type
`amazingweb007.com`	164	Adult Dating
`teknotigr.com`	151	Empty Blog
`climate13.com`	148	Fake Conference
`zqbifen8.com`	84	Advertisement
`dxsmalvn.com`	81	Page Not found

Table 2: Top 5 switching domains

NS activity; "`climate13.com`" is a fake conference website mentioned in "`scamwarners.com`"; and "`dxsmalvn.com`" was not loaded in the browser. All of the above are *only examples* of those domain names with an unfavorable behavior, which were spotted by using the name server switching as a side channel information.

5 Domain Clustering

One limitation of the findings so far is that a highly supervised process is needed to make use of the triggers made by the NSSF signature, and to identify the intended usage of domain names. While a great part of this process can be automated (e.g., by automatic crawling, analysis, etc.), for every domain name with a number of switchings, a manual vetting might be necessary to facilitate this process and ensure a high accuracy.

To this end, and in search for alternatives to this seemingly costly process, in the following we look at trading the cost for less supervision. In this section, we present in detail an experimental analysis of domain clustering based on the NSSF as the main feature. Initially, we cluster domains based on the exact matching of a footprint, then we introduce a more robust clustering method based on the n-grams created from the NSSF. The intuition for using this process of clustering is that one may have labels for a certain number of domain names, indicating their usage, and would want to extrapolate those labels for the rest of unlabeled domains. To that end, groups of domain names based on their NSSF similarity would greatly facilitate this operation. Next, we discuss the clustering approach to the problem with various settings.

5.1 Cluster Domains by Exact Footprint Matching

Given the strict definition of the NSSF, one would think that it is unlikely for two domain names to have the same footprint using the whole NSSF as a signature over the entire period of 90 days where domain names are observed in this study. However, to motivate for a partial footprint, we first tried this extreme scenario: we clustered domain names based on the exact match of their total footprint. To do that, we have followed the following steps. First, we computed the NSSF of each domain name from the DNZA files, as described in Figure 3. Then, using a simple counting, we grouped the domain names by the exact match of the footprint.

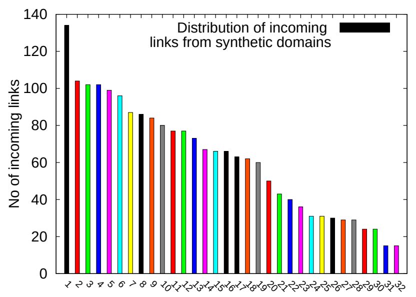


Fig. 6: Distribution of incoming links

To manually vet the resulting clustering, we selected clusters with at least 10 domain names, with footprints of length greater than five. Using those settings, we ended up with 2604 domains in 84 clusters with the median and maximum sizes of 23 and 99, respectively. After analyzing these clusters, we identified two special types of groups of domains: take over domains and domains that work collaboratively to increase the page rank of third party domains. Informally speaking, take over domain names are domains with registrations that expired and registered by another registrant in the hope of attracting traffic and utilizing it based on the previous usage of the domain name.

We found 31 takeover domains in 2 clusters of size 18 and 13 with NSSF of length 40. All of these domains used DNS providers that have the service of domain parking to generate revenue. By examining their WHOIS information, we found that all of these domains are owned by the same entity.

In the second case, and by statically analyzing the content of the web sites for the identified domain names, we found a set of domains that maintain links (referral urls) to the target domains to increase their page rank. After analyzing the contents hosted on these domains, we found that all of these domains are synthetically generated; we also found 343 domains that work together to increase the page rank of 32 third party domains, where all of them are owned by one entity and maintain a similar pattern of NS switching. In Figure 6, we show the distribution of incoming links from 343 domain to the 32 third party domains.

Note that, while those experiments are mainly manual, they are easy to automate using off-the-shelf tools and in-house datasets. As a registry of the studied TLDs, VeriSign also runs operations of crawling certain level of contents of those TLDs, and identify their usage by analyzing such contents. Also, information such as WHOIS contents, while are not totally under the control of the registry, are easy to obtain automatically to identify associations between domain names based on that additional dimension. To this end, findings in the work can be easily automated operationally.

5.2 The n-Gram Based Clustering

The exact matching outlined above is promising in identifying certain groups of domain names based on the whole NSSF, however the major drawback of the technique is that it will not capture associations between domains which have partial similarities. To work around this drawback, we consider clustering domains based on n-grams, defined as the reoccurring substrings within the footprint. The main challenging part of this clustering model is in identifying the n-gram given a domain name's footprint. Particularly, defining the proper distance metrics and measures is a significant challenge given the special nature of the NSSF. These distance measures will be used to cluster domains using any off-the-shelf clustering algorithm. In this experiment, we adopt the agglomerative hierarchical clustering. In the following sections, first we introduce two n-gram based distance metrics and two heuristics on building n-grams from the footprints.

n-gram based distance metric. After building n-grams from a footprint, we want to compute a pairwise distance measure between domains. We use two metrics suitable for that purpose: I) Tribased_Dist [14] and II) Keselj [15] distance

$$Tribased\_Dist(x,y) = 1 - \frac{1}{1 + |tri(x)| + |tri(y)| - 2 * |tri(x) \cap tri(y)|}$$

$$Keselj\_dist(x,y) = \frac{1}{|tri(x)| + |tri(y)|} \sum_{w \in tri(x) \cap tri(y)} \left(\frac{f_x(w) - f_y(w)}{f_x(w) + f_y(w)} \right)^2$$

Now we explain two heuristics for identifying the n-grams from a footprint.

Heuristic I. Suppose that a domain "`abc.com`" has footprint $2\_0\_1 : 0\_2\_2 : 4\_3\_3$ which we denote as $NSSF_{abc.com}$ (this footprint can be read as domain "`abc.com`" adds 2 name servers at time 1, deletes both of the name servers at time 2, and adds 4 then delete 3 name servers at time 3). To build the n-gram, we first split the $NSSF_{abc.com}$

across time and consider each part of the footprint as an item of a set from which we will construct the n-gram. $NSSF_{abc.com}$ spans over 3 time-stamps, so if we split the footprint across time, we get a set of three items $\{2\_0\_1, 0\_2\_2, 4\_3\_3\}$. We then apply the n-gram construction method over the set. In all of our experiments, we set $n = 3$.
Heuristic II. In heuristic II, we relax the rules of splitting the footprint. We split the footprint based on events (add, delete) and time. For footprint $2\_0\_1 : 0\_2\_2 : 4\_3\_3$, we gather the added name server event which construct a set $\{2, 0, 4\}$. The same thing is done for deletion of name servers as well as the time-stamps, and build a set $\{0, 2, 3\}$ and $\{1, 2, 3\}$ respectively. We then construct 3 n-grams from these sets. Each of these n-grams will compute a distance measure (Tribased or Keselj) for a domain pair and final distance is taken by averaging these distances.
Empirical evaluation. To empirically evaluate the performance of both of the heuristics described above, we select a dataset of $7,830$ domains with footprint length 6 and higher. Then we compute a pairwise distance metric as mentioned in section 5.2 between a domain pair and analyze the distribution of the measure. Figures are omitted for the lack of space. We observe that the distribution of the keselj distance using heuristic I as described in §5.2 is very skewed. Such skewness has an impact on the clustering by producing a smaller number of large size clusters and large number of small size clusters. On the other hand, we notice that heuristic-II reduces the skewness, with a similar insight on the resulting cluster size.

Next, we performed a hierarchical (with complete linkage; defined as $d(A, B) = \max_{a\in A, b\in B} d(a, b)$) clustering algorithm with an appropriate threshold k on pairwise distance. Setting the threshold affects the way clusters are formed: this means that the algorithm will cluster two domains if the distance between them is less than k. A threshold $k = 0.4$ is empirically found to be appropriate in our dataset. Table 3 shows basic statistics of cluster analysis for both the trigram and keselj distance metric while using heuristic-II. Table 4, shows the quantile summary on the size of the top 50 clusters.

# domains	# clusters	Avg size	# dom (top-50)
7,830	1893	4.1	5631
7,830	1774	4.4	5763

Table 3: Cluster statistics of the result when using heuristic-II.

Distance	min	25%	50%	75%	max
Trigram	8	12	19	50	3536
Keselj	8	11	19	54	3535

Table 4: Quantile summary of the top 50 clusters NSSF and heuristic-II.

Clusters vetting. To analyze the identity of the resulting clustering by understanding the use of the domain names, we again selected the top 50 clusters in size and obtained the registration information of those domain names. We extracted the registration information for the domain name (registrant and registrar) from the whois database. We found that all domain names within each cluster belong to the same registrant, and are registered via the same registrar. For legal reasons, we could not present further details on the registrants, however Table 5 shows the quantile distribution of the percentage of domains that belong to the same owner in a cluster.

Finally, we performed a similar analysis with host information to check whether domain names that were grouped within the same cluster also were hosted on the same host, or using the same hosting providers. Our analysis shows that it is almost always

Median	min	25%	50%	75%	max
60	2.83	22.01	58.05	99.50	100

Table 5: Quantile summary (%) of domains hosted by a same owner per heuristic-II.

Median	min	25%	50%	75%	max
50	11.11	28.57	59.35	100	100

Table 6: Quantile summary (%) of domains hosted by the same service per heuristic-II.

the case that such domains would be hosted using the same hosting infrastructure. For legal reasons, we could not name those individual domains and their hosting providers, however general statistics on the quartile distribution of the percentage of domain names hosted by the same hosting provider residing in the same cluster are shown in Table 6.

6 Prediction model

Given the value of the NSSF as discussed above, we looked further into the predictability of this feature. As discussed in § 1, there are many interesting applications that can be built on such predictability. The final interesting aspect that we examined is the power of an off-the-shelf prediction algorithm in predicting the NSSF. For that, we built a time series based prediction model to estimate the number of name servers that a domain will interact with, given the history of interaction of the respective domain.

We used Autoregressive Integrated Moving Average (ARIMA) [8] model for prediction. In this model, instead of time, we consider ordered events to build the time series data (i.e., the index used in the NSSF is used as for the time series). We trained the model with populated data series and predicted the number of interactive name server for future events. To validate our model, we omitted the data points for the last 2 events, predicted them, and computed the errors using the mean sum of squares (MSE; defined as $1/n \sum_{i=1}^{n}(a_i - b_i)^2$). Table 7, presents prediction results for the top three switching domains in Table 2. By rounding the predicted values, we can marginalize the error resulting from the prediction, and achieve a high accuracy of the results.

Domain	True value	Predicted value	MSE
amazingweb007.com	2 , 2	2.47 , 2.23	0.136
teknotigr	2 , 1	2.4 , 1.3	0.125
climate13.com	2 , 2	2.35 , 2.15	0.072

Table 7: Prediction accuracy for top domains

7 Related Work

To the best of our knowledge, there is no prior work in the literature that looked at the dynamics of associations between name servers and domain names at the level of granularity we use in this work to understand domain names' usage. However, there are several works in the literature on using aggregates of the name servers as a feature for identifying the intended use of domain names, as pointed out earlier. For example, the work in [3, 5, 4], identifies the number of name servers associated with a domain name over a period of time as a feature for extrapolating the label of malicious or benign and show that this feature is very effective.

Timeseries are used in the literature, in the context of intrusion detection systems, to character and process intrusion alerts at an aggregate level [16, 17]. Other security applications of time series have found in detection of distributed of denial services [18, 19], and authentication [20], among others.

The use of machine learning techniques in security, including clustering, is not new. There has been a large body of work in the matter, summarized in [21] and [13].

8 Conclusion

We analyzed the name server switching patterns for domains to potentially use this information for security applications. We used the evolution of name servers to build an identifier for domains that can be used to group domains of similar behavior. We clustered domains based on footprint and observed that the majority of domains in a cluster have the same owner and are hosted by the same provider. We have also presented a time series analysis to estimate number of name servers that a domain will interact with.

References

1. K. Salchow, "Load balancing 101: Nuts and bolts," *White Paper, F5 Networks, Inc*, 2007.
2. E. Nygren, R. K. Sitaraman, and J. Sun, "The akamai network: A platform for high-performance internet applications," *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 3, 2010.
3. T. Snoke, "Watching domains that changes dns servers frequently," CERT/CC Blog, 2013.
4. Y. He, Z. Zhong, S. Krasser, and Y. Tang, "Mining dns for malicious domain registrations," in *CollaborateCom*, 2010.
5. M. Felegyhazi, C. Kreibich, and V. Paxson, "On the potential of proactive domain blacklisting," in *LEET*, 2010.
6. F. Lardinois, "More than 250m domain names have now been registered, almost half are .com and .net," http://tcrn.ch/1i3G0Fh, April 2013.
7. R. Shumway and D. Stofer., *Time Series Analysis & its Applications*. Springer Verlag, 2000.
8. G. Box and G. Jenkins, *Time series analysis: Forecasting and control*, 1970.
9. L. C. Alwan and H. V. Roberts, "Time-series modeling for statistical process control," *Journal of Business & Economic Statistics*, 1988.
10. S. Porter Hudak, "An application of the seasonal fractionally differenced model to the monetary aggregates," *J of the American Stats Association*, 1990.
11. R. Shumway and D. Stoffer, "Dynamic linear models with switching," *Journal of the American Statistical Association*, vol. 86, no. 415, pp. 763–769, 1991.
12. J. chrysostome Bolot and P. Hoschka, "Performance engineering of the world wide web: Application to dimensioning and cache design," 1996, pp. 1397–1405.
13. A. Mohaisen and O. Alrawi, "Amal: Highfidelity, behavior-based automated malware analysis and classification," Verisign Labs, Tech. Rep., 2013.
14. D. Lin, "An information-theoretic definition of similarity." in *ICML*, 1998.
15. Y. Miao, V. Kešelj, and E. Milios, "Document clustering using character n-grams: a comparative evaluation with term-based and word-based clustering," in *CIKM*, 2005.
16. J. Viinikka, H. Debar, L. Mé, A. Lehikoinen, and M. Tarvainen, "Processing intrusion detection alert aggregates with time series modeling," *Information Fusion*, 2009.
17. S. Axelsson, "Intrusion detection systems: A survey and taxonomy," BTH, Tech. Rep., 2000.
18. J. B. Cabrera, L. Lewis, X. Qin, W. Lee, R. K. Prasanth, B. Ravichandran, and R. K. Mehra, "Proactive detection of distributed denial of service attacks using mib traffic variables-a feasibility study," in *IEEE IM*, 2001.
19. H. Liu and M. S. Kim, "Real-time detection of stealthy ddos attacks using time-series decomposition," in *IEEE ICC*, 2010.
20. R. Mayrhofer and H. Gellersen, "Shake well before use: Authentication based on accelerometer data," in *Pervasive computing*. Springer, 2007, pp. 144–161.
21. R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *IEEE Security and Privacy*, 2010, pp. 305–316.