Anomaly Detection in Cybersecurity Data

Outlier and Exploit Detection in DNS Logs

Rory Cox
MSc Data Anaytics
Queen's University, Belfast
Northern Ireland, UK
roderick.m.cox@googlemail.com

CONTENTS

Abstract

Anomaly detection in Domain Name System (DNS) Logs is a powerful tool for network administrators to gain insights into their network traffic, whether it be to understand the activity of their network's users and software, identify malfunctions in their network, or crucially, to identify malicious activity. An anomaly detection solution should provide a tool that an analyst can use to gain these insights and inform investigations into activity on their network. This paper proposes a two-part solution for anomaly detection; outlier detection and exploit detection, presented in the form of an analytical dashboard. Outlier detection aims to identify anomalous time-intervals, by engineering and aggregating features where outlying values of those features in a time-interval may be indicative of some change in DNS activity. Outlier-detection methods were compared and contrasted, and Median Absolute Deviation (MAD) and Autoencoders were seen to be most effective. Exploit detection aims to identify anomalous items, by computing a risk-score. Exploit-specific features are engineered, and the user determines value ranges for each feature which may indicate exploitative activity, assigning weights for feature importance. The risk-score for each item is the sum of all feature weights where the feature value for the item falls within the specified range. Inspection of high risk-score items validated this approach. While there are some limitations to its use, this proposed two-part solution to anomaly detection in DNS provides holistic anomaly detection capability, with the potential to be a powerful and flexible tool for network administrators.

I. INTRODUCTION

A. Inroduction, Objectives and Business Case

The Domain Name System (DNS) is a fundamental component of modern networks. It provides a means of resolving a server name and an IP address which is required for the transfer of data between client and server. DNS is essential for any machine that requires network access, and that necessity presents various exploitative opportunities for attackers. Gaining insights into DNS traffic is an important step in understanding the activity of a network, and can bring great advantages to a network's administrator.

This paper will propose a two-part approach to anomaly detection, providing an effective and holistic anomaly detection solution. The distinction between anomaly detection and threat-hunting should be noted. An anomaly detection solution should serve two purposes. Firstly, to identify deviations from regular DNS traffic. Secondly, to identify certain items that display symptoms of malicious activity. It would then be for an analyst to use their contextual knowledge of their specific network to select periods or items to investigate, monitor or white/blacklist.

The business case for such an anomaly detection solution is varied and extensive. The most obvious use-case would be to identify malicious activity on a network. However, a company may also be investigating deficiencies in their network, or malfunctions such as servers going down or excessive queries that fail to receive responses. They may be interested in the activity of their network's users or programs on their machines, or one-off events that might change their DNS traffic. A company like Allstate would be able to leverage this tool for gain these insights into their network, and would especially benefit from being alerted to malicious activity due to the sensitive client information that is held by insurance companies.

B. Proposed Solution

A two-part solution to anomaly detection in DNS logs will be proposed in this paper; outlier detection and exploit detection. Outlier detection will identify individual time-intervals as anomalous by their deviation from regular DNS traffic for some key features, as in Figure 1.

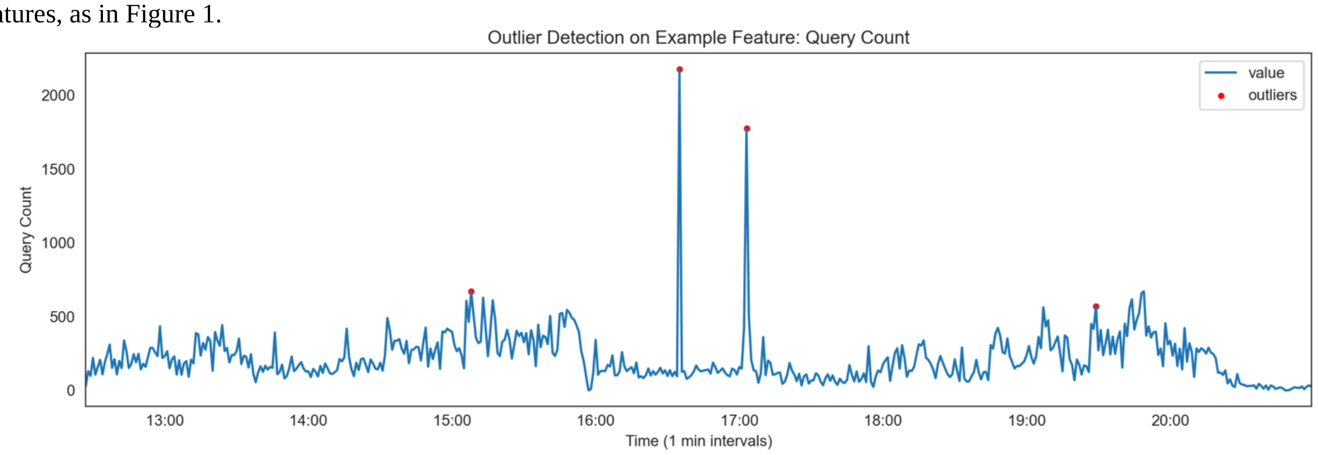
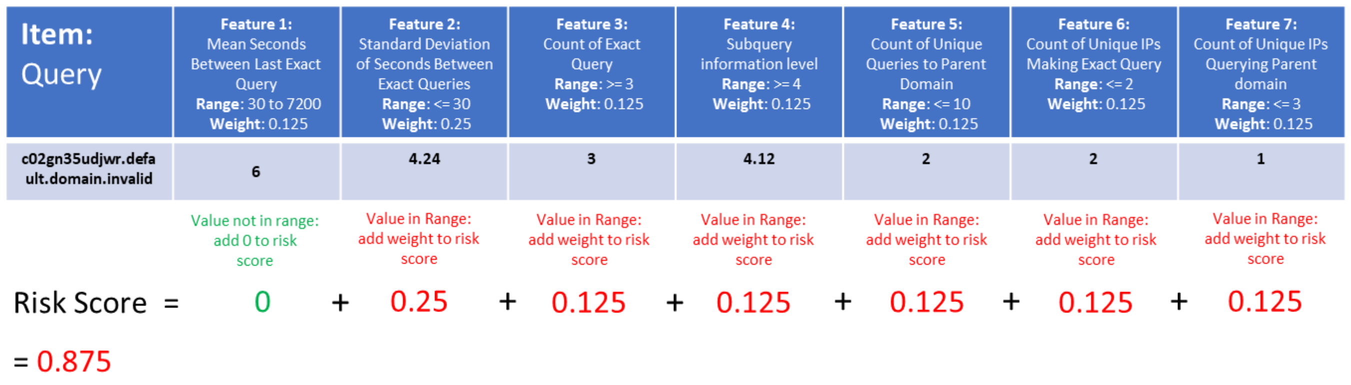


Figure 1 – Outlier Detection Example, Showing Potential Outliers in Query Count for each 1 Minute Time-Interval

For features where values are expected to remain static over time, Median Absolute Deviation (MAD) will be used to identify outliers. For features where values are expected to experience periodic trends, outliers will be identified using Autoencoders.

Exploit detection will identify items relevant to each exploit as anomalous by calculating a risk-score. Features will be engineered that capture each exploit's specific traits. The user will set value ranges for those features to help identify indicators of compromise or exploit. The user also assigns a weight to each feature to represent its importance, such that the risk-score for each item is the sum of feature weights where feature values fall within the predetermined ranges, as in Figure 2.



Item: Query	Feature 1: Mean Seconds Between Last Exact Query **Range**: 30 to 7200 **Weight**: 0.125	Feature 2: Standard Deviation of Seconds Between Exact Queries **Range**: <= 30 **Weight**: 0.25	Feature 3: Count of Exact Query **Range**: >= 3 **Weight**: 0.125	Feature 4: Subquery information level **Range**: >= 4 **Weight**: 0.125	Feature 5: Count of Unique Queries to Parent Domain **Range**: <= 10 **Weight**: 0.125	Feature 6: Count of Unique IPs Making Exact Query **Range**: <= 2 **Weight**: 0.125	Feature 7: Count of Unique IPs Querying Parent domain **Range**: <= 3 **Weight**: 0.125
c02gn35udjwr.default.domain.invalid	6	4.24	3	4.12	2	2	1
	Value not in range: add 0 to risk score	Value in Range: add weight to risk score	Value in Range: add weight to risk score	Value in Range: add weight to risk score	Value in Range: add weight to risk score	Value in Range: add weight to risk score	Value in Range: add weight to risk score

Risk Score = 0 + 0.25 + 0.125 + 0.125 + 0.125 + 0.125 + 0.125

= 0.875

Figure 2 – Illustration of the Risk-Score Calculation in Exploit Detection, Using One Item from the Botnet Heartbeat Check as an Example

A two-part approach has the advantage of incorporating two anomaly detection approaches, a macro approach which identifies anomalies by deviations from regular activity over time, and a micro approach which identifies anomalies by searching for specific activity symptomatic of exploits. In this way, the two-part approach is holistic, as the solution is not limited to identifying exploits, and certain exploits which may not be captured by outlier detection are captured by exploit detection.
As well as malicious activity and deficiencies in a network, there are many benign causes of anomalous traffic over short intervals. For example, antivirus software sending many lengthy and complex DNS queries to gather and verify information about certain domains, or Microsoft Office 365, which relies upon uncommon DNS records for user verification. A popular online sale may cause a spike in a network's outgoing queries. An effective anomaly detection solution would identify these examples if they caused inconsistencies in the regular flow of DNS traffic, and there is likely to be some overlap between the results of both approaches, as shown in Figure 3. The two approaches will be incorporated into a dashboard. For each approach, the user will be able to download and inspect the data and feature values.

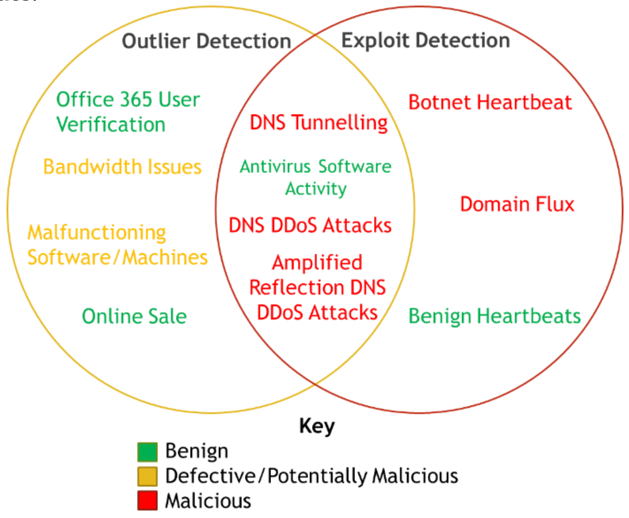

Outlier Detection
Exploit Detection
Office 365 User Verification
Bandwidth Issues
Malfunctioning Software/Machines
Online Sale
DNS Tunnelling
Antivirus Software Activity
DNS DDoS Attacks
Amplified Reflection DNS DDoS Attacks
Botnet Heartbeat
Domain Flux
Benign Heartbeats
Key
🟩 Benign
🟨 Defective/Potentially Malicious
🟥 Malicious

Figure 3 – Venn Diagram Giving Examples of Potential Results and Overlap in Outlier and Exploit Detection

C. Paper Structure

Section II outlines the project approach by describing the process that was undertaken to arrive at the proposed solution. It then provides an overview of DNS functionality and its key components. The dataset and data cleaning process are described, and key engineered features that are used in outlier and exploit detection are explained. For outlier detection, the methods and models to detect outliers are introduced, and for exploit detection, the features of that are engineered for each exploit are listed. Section III assesses the effectiveness of each outlier detection method and explains the choice of method for each feature. It then explains the results of exploit detection, acknowledging the nuance of an unlabelled dataset, and finally assesses the effectiveness of a Long Short Term Memory (LSTM) model to predict the parent domain randomness level for queries. Section IV discusses the practicalities of using the solution, and discusses alternative approaches to outlier and exploit detection. It also discusses the limitations of the solution. Section V offers concluding remarks on the solution.

II. Methods

A. Project Approach

To understand how anomalies could be identified in DNS, domain knowledge was gained through online research and discussion with Allstate's Cybersecurity Experts, which also included general understanding of networks and the cybersecurity space. This led to the understanding of 2 kinds of anomaly within DNS:

- Time-intervals whose values deviate from regular traffic for key features
- Specific items that share symptoms of a known DNS exploit

Accordingly, a two-part approach to anomaly detection was proposed: outlier detection and exploit detection. In this way, the project included – but was not constrained to – threat hunting within DNS. For outlier detection, researching and assessing the effectiveness of outlier detection methods led to the decision to use MAD and Autoencoders. For exploit detection, examination of the risk-scoring systems used by major cybersecurity providers informed the use of risk-scores in DNS, and detailed research into each DNS exploit was necessitated to engineer appropriate features to capture their indicators. Given the unlabeled nature of anomaly detection, neither the effectiveness of outlier detection's nor exploit detection's results could be measured numerically, therefore each approach demanded a more nuanced discussion of their results. Researching software available from major cybersecurity providers also led to the solution's implementation into a dashboard.

B. Understanding DNS

DNS is the protocol that converts a human-readable naming system of hosts or servers to Internet Protocol (IP) addresses, and vice-versa. In the early days of the internet, establishing a connection to another machine required knowledge of that machine's IP address in advance. This was not sustainable, given that IP addresses are not memorable for humans, and by the end of 2020 there are expected to be more than 30 billion devices connected to the internet [1]. DNS provides a directory service, linking domain-related information such as IP addresses to domain names. There are many record types that can be requested in a DNS query [2]. These records are queried by many applications that communicate with other machines on networks. These include games, database management applications, instant messaging and video call software and many more. DNS consists of a hierarchical query system, and caching. When a query is made from a machine, the machine, and the DNS resolver will first attempt to resolve it by checking their caches. If the record is not cached or has expired, the DNS resolver follows a hierarchical process of iterative queries to until the record is found. An example is given in Figure 4.

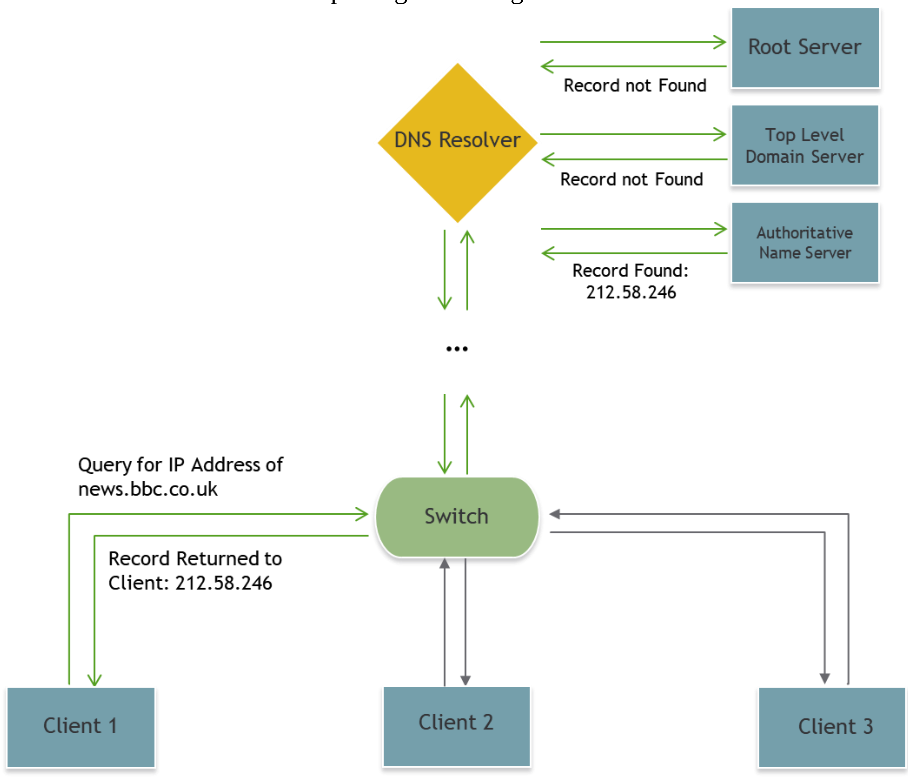


Figure 4 – Example of the Process of Resolving a Record to a Domain Name in DNS. Client 1 Makes a Query for a Domain Name's IP Address, the Query is Forwarded to the DNS Resolver. If the Record is not Cached, the DNS Resolver Makes Queries Iteratively Until the Record is Found and Returned to Client 1.

A broad definition of regular DNS traffic should be made when considering anomaly detection. Regular DNS traffic is traffic that maintains approximately consistent values for some key features over time. The following features are used in outlier detection to identify outliers in DNS traffic.

- Query count
- Count of queries for uncommon record types
- Count of queries that return uncommon response types
- Mean information level of subqueries (where high values can indicate encryption)
- Mean randomness level of each query's parent domain (where high values can indicate randomly generated domain names)

For a corporate network, some feature-values, such as mean information level of subqueries, are expected to be relatively static over time, while others such as query count may experience periodic trends such as for working hours.
The inherent characteristics of DNS provide desirable exploitative opportunities for attackers. DNS is a vital component of network infrastructure, and as such, DNS queries are not typically blocked by firewalls [3]. The huge volumes of DNS queries allow exploits to be easily concealed, and the communication protocols used by default in DNS allow for IP "Spoofing" [4] which creates further opportunities to obfuscate the source of malicious activity. The traffic volume of DNS queries is also often so high that manual inspection is impractical, and as such, DNS logs may not be subject to sufficient levels of scrutiny. The solution proposed in this paper will address the following exploits.

- Botnet Heartbeats
- DNS Tunnelling
- Outgoing DNS Distributed Denial of Service (DDoS) Attacks
- Outgoing Amplified Reflection DNS DDoS Attacks
- Domain Flux

These exploits occupy various points on the Cyber-Kill-Chain proposed by Lockheed-Martin in 2011 [5], and exhibit specific identifying traits that are reflective of their objectives. It should be noted that while some malicious activity, such as Outgoing DNS DDoS Attacks, do not represent a direct threat to the network owner, the presence of a compromised machine on a network risks its engagement in other malicious activity. It is therefore in the network owner's self-interest to identify and neutralise the threat.

C. Dataset and Data Cleaning

A third party open-source dataset of DNS logs from the Mid-Atlantic Cyber Defense Competition (MACCDC) 2012, gathered using Bro, was used for this paper [6]. It contained 338,003 observations, each of which represented a DNS packet containing a query and response, if applicable. The dataset is significantly different from what would be expected from a company network's DNS logs. The competition consisted of two teams competing to maintain or attack specified services on a network [7], though it was not specified that DNS exploits should be used to accomplish this. Although a team was tasked with maintaining the supply of various specified services, there is no reason to assume that the DNS queries generated for these services would be representative of a typical company's network's queries. Furthermore, the adversarial work of the attacker team convolutes the data. The attacker team appeared successful in bringing the DNS resolver offline on more than one occasion, resulting in a large proportion of DNS queries returned with no response, which is usually uncommon. There were also periods where no queries were made, potentially the result of the network or machines being compromised or simply turned off. The dataset was unlabeled, so measuring success of any anomaly detection methods could not be achieved through the usual metrics of accuracy, precision and recall, and classification algorithms could not be used. There were no obvious instances of the DNS exploits explored in this paper present in the dataset. It is possible that the attacker team used other means to perform cyber-attacks, since the attacker team was not tasked to use DNS, or achieve the objectives of the DNS exploits explored in this paper.
The DNS Log data collected using Bro did not contain missing or corrupted values. Approximately 20% of the queries were duplicated and were therefore removed. Query duplication was a known issue of Bro in 2012, and a fix was applied in 2014 [8]. A check was also made to find any queries made from one DNS resolver to another. The small number that were found were removed, as these were relayed queries which are effectively duplicates.

D. *Features of Interest*

Two key engineered features that were used in both outlier and exploit detection require detailed explanation.

I) Mean Subquery Information Level

The information level of a subquery can indicate whether a query contains encrypted information. Together with other indicators, it can contribute towards evidence for DNS Tunnelling or a Botnet Heartbeat. The measure used to calculate information level is the Shannon Entropy [9]; a measure of the level of information contained in a string. The Shannon entropy $H(X)$ is calculated for each subquery string, where N is the number of unique characters in the string, and x_i is each unique character in the string.

$$H(X) = -\sum_{i=1}^{N} P(x_i) log_2 P(x_i)$$

II) Mean Parent Domain Randomness Level

The randomness level of a query's parent domain can indicate whether the parent domain has been randomly generated, as in the case of the Domain Flux exploit. The domain names were distinguished from second-level domains (such as "co" in ".co.uk") using the root zone database from IANA [10]. The feature uses an LSTM trained on a sample of 50,000 of the top 1m domain names according to alexa.com [11] and 50,000 randomly generated strings of characters permitted in domain names [12] with lengths randomly sampled from the 50,000 domain names. Padding was used to ensure vectors were of equal length and could be used for training in batches. An LSTM is effective when there are long-term dependencies that can be learned from data, something which can be lost when using a basic Recurrent Neural Network, due to the vanishing gradient problem, or by using a Feed Forward Neural Network (FFNN), which does not use feedback connections. An LTSM will therefore be effective in distinguishing domain names in the dataset from random strings, if characters used in those domain names have long-term dependencies on previous characters. The structure of the LSTM used for this feature is illustrated in Figure 5.

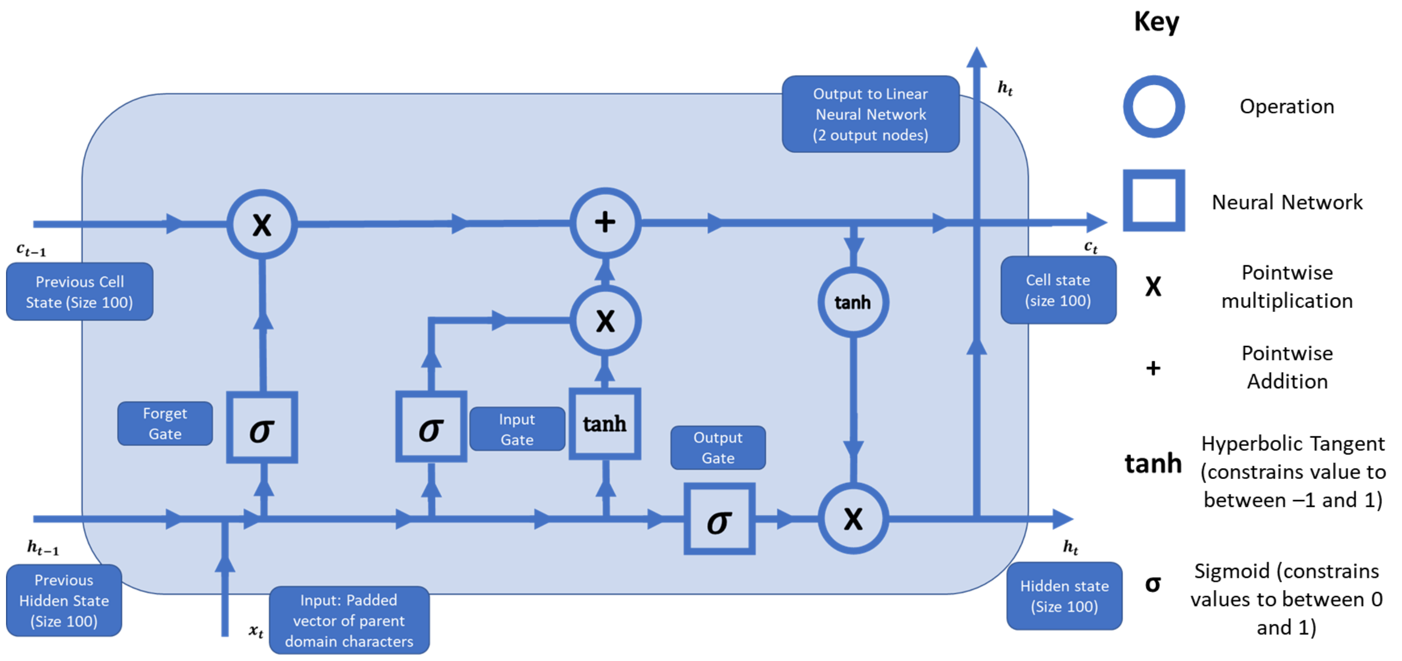

Key
Operation
Neural Network
X
Pointwise multiplication
+
Pointwise Addition
tanh
Hyperbolic Tangent (constrains value to between −1 and 1)
σ
Sigmoid (constrains values to between 0 and 1)
Output to Linear Neural Network (2 output nodes)
h_t
c_{t-1}
Previous Cell State (Size 100)
c_t
Cell state (size 100)
Forget Gate
Input Gate
Output Gate
tanh
σ
h_{t-1}
Previous Hidden State (Size 100)
x_t
Input: Padded vector of parent domain characters
h_t
Hidden state (Size 100)

Figure 5 – LSTM Cell Structure for Determining the Mean Parent Domain Randomness Level Feature Values

E. *Outlier Detection – Methods and Implementation*

To identify time-intervals with anomalous values for key features in outlier detection, an appropriate size of time-interval first needed to be determined. To obtain meaningful results for outlier detection, the interval size should be made as small as possible, for specificity, but not so small that time-intervals are subject to excessive noise. Figure 6 shows, for each feature used in outlier detection, a steep decrease in the standard deviation of values each second until an interval size of approximately 1 minute, and gradual decrease following that, implying that 1 minute is an appropriate interval size.

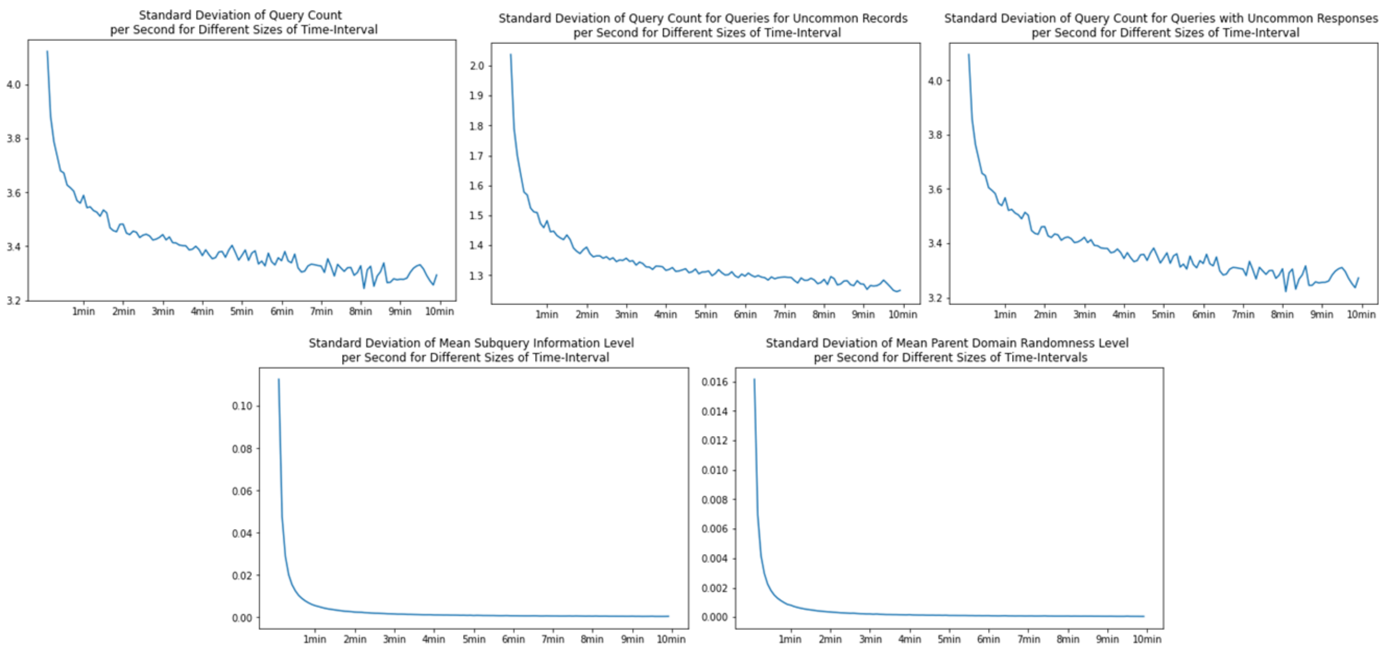


Figure 6 – Figure to Show the Standard Deviation of Values each Second for each Feature used in Outlier Detection Sharply Decreasing Until an Interval Size of Approximately 1 Minute.

To identify time-intervals with anomalous values for key features in outlier detection, the following methods are discussed in this paper. For each method, the dashboard allows anomalous time-intervals and logged DNS queries within those time-intervals to be downloaded as a csv file and inspected.

I) *Statistical Methods*

Two statistical methods, standard deviation and MAD, are examined. Each method identifies outliers by their respective measure of deviation from a statistical average; mean or median.

a) *Standard Deviation*

The standard deviation σ is calculated for a date-range, where N is the number of time-intervals in the date-range, x_i is the value of a certain feature at each time-interval, and μ is the mean value of that feature over all N time-intervals.

$$\sigma = \sqrt{\frac{\sum_1^N (x_i - \mu)^2}{N}}$$

Values for each time-interval are therefore identified as outliers by the value's multiple of the standard deviation from the mean, as shown in Figure 7. Using the dashboard, the user selects a feature and a period over which to compute the mean and standard deviation and identify outlying time-interval values. The user sets a value for the standard deviations from the mean that a time-interval's value must exceed to be flagged as an outlier.

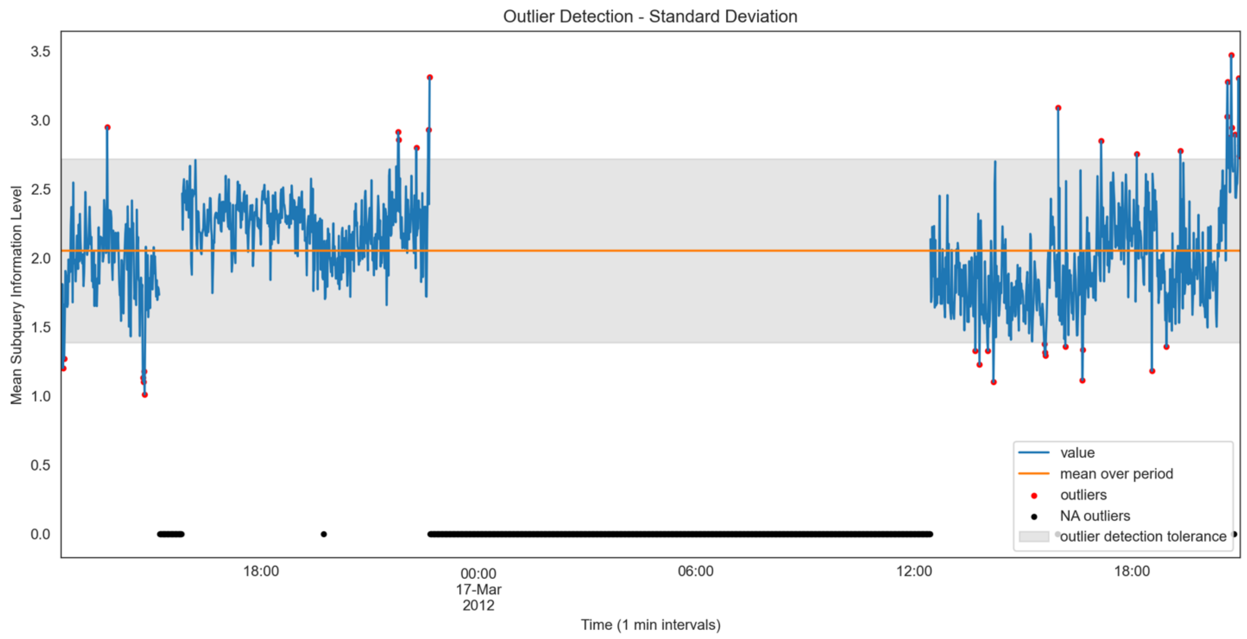


Figure 7 – Example of Using Standard Deviation for Outlier Detection on the Mean Subquery Information Level Feature

b) Median Absolute Deviation (MAD)

The MAD is calculated for a date-range, where N is the number of time-intervals in the date-range and x_i is the value of a certain feature at each time-interval.
$i \in \{1 \dots N\}$

$$d_i = |x_i - med([x_1 \dots x_n])|$$
$$MAD = med(d_i)$$

Values for each time-interval are therefore identified as outliers by the value's multiple of the MAD from the median, as shown in Figure 8. Using the dashboard, the user selects a feature and a period over which to compute the median and MAD and identify outlying time-interval values. The user sets a value for the MADs from the median that a time-interval's value must exceed to be flagged as an outlier.

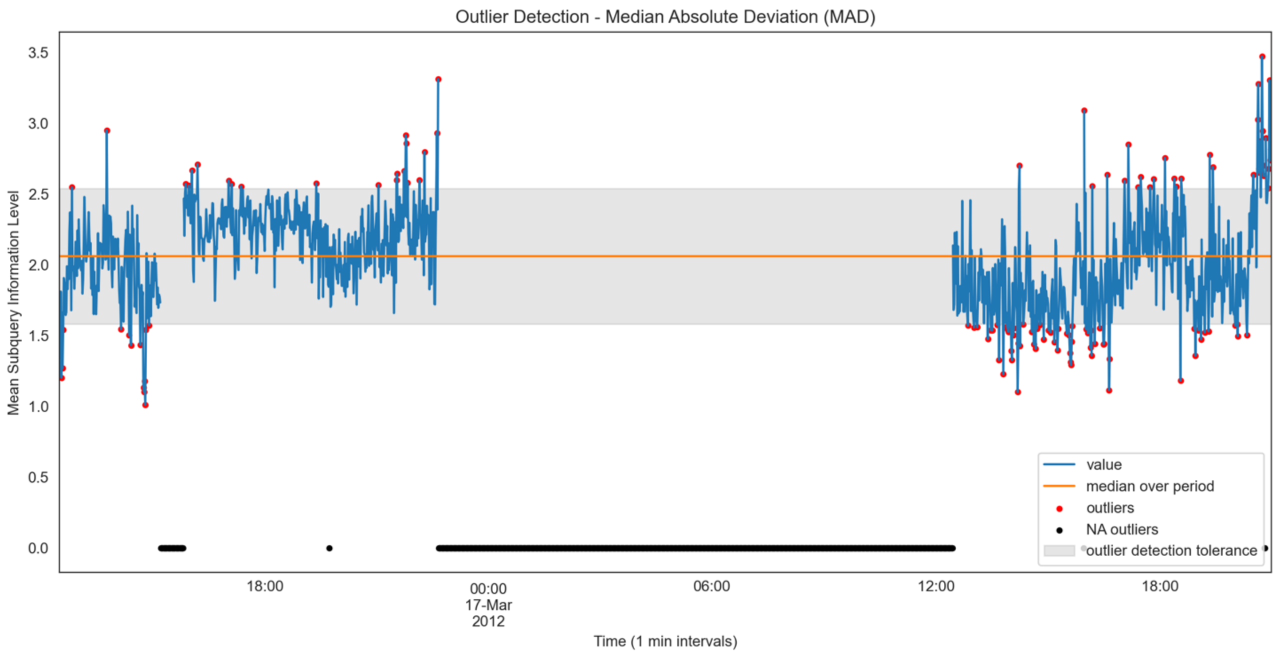


Figure 8 – Example of Using Median Absolute Deviation for Outlier Detection on the Mean Subquery Information Level Feature

II) Deep Learning Methods

A predictive approach can also be taken for outlier detection using deep learning models. The method used in this paper is based on the method proposed by Harsha Kumara Kalutarage and Siraj Ahmed Shaikh [13]. The user trains a deep-learning model on data prior to a prediction period to predict values for features at each time-interval. Outliers, by definition, are inconsistent with the shape, or "signal" of the data, and therefore an effective model will not be able to learn to predict outliers accurately. The user sets a confidence threshold for the predictions of the training data, which is 95% by default, and this translates to a threshold for the Root Mean Square Error (RMSE) that a time-interval's value must exceed to be flagged as an outlier. The process of converting a confidence threshold to a RMSE upper threshold is shown in Figure 9. First, the RMSE distribution is transformed using a cube root transformation and approximated using a normal distribution. Next, the transformed upper RMSE threshold is calculated using the confidence interval provided by the user. Finally, this value is reverse-transformed, and is used as the RMSE upper threshold. The model can then predict values over a specified prediction period, and any values with an RMSE greater than the previously calculated RMSE upper threshold, are flagged as outliers.

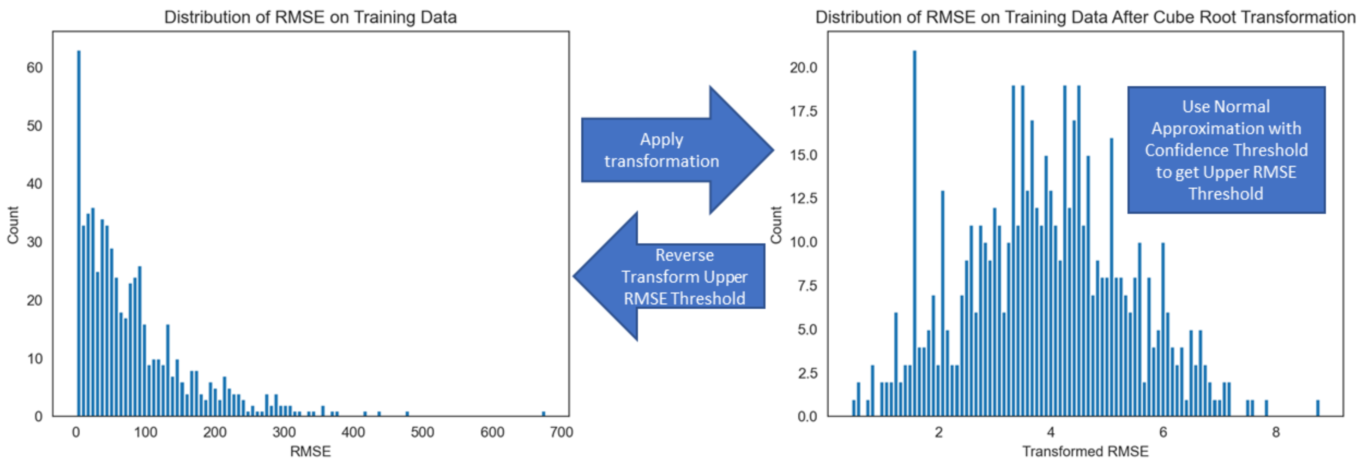


Figure 9 – Figure to Show the Process by which a Confidence Threshold is Converted to an RMSE Upper Threshold for Outlier Detection using Deep Learning Model Predictions

In order to preserve the integrity of the training data, the user may upload date indices that they have identified as anomalous to the dashboard. The deep-learning models will not include any time-interval in the training data that is contained within the specified indices. It is suggested that, once sufficient data is available, the deep-learning models are trained on the week of data leading up to the desired prediction-period. This will ensure that models stay up-to-date with network activity.

a) Feed Forward Neural Network (FFNN)

An FFNN is a supervised deep learning model that can be used to solve a regression problem. An FFNN can be trained to predict a value for a time-interval using values from a predefined number of previous time-intervals. The number of time-intervals to use for the input was determined by grid-search where the metric for accuracy was the MSE. Shuffling the dataset and splitting into a train and test set allowed for the training of the FFNN and assessment of its accuracy. An example of outlier detection by RMSE of predictions using an FFNN is shown in Figure 10.

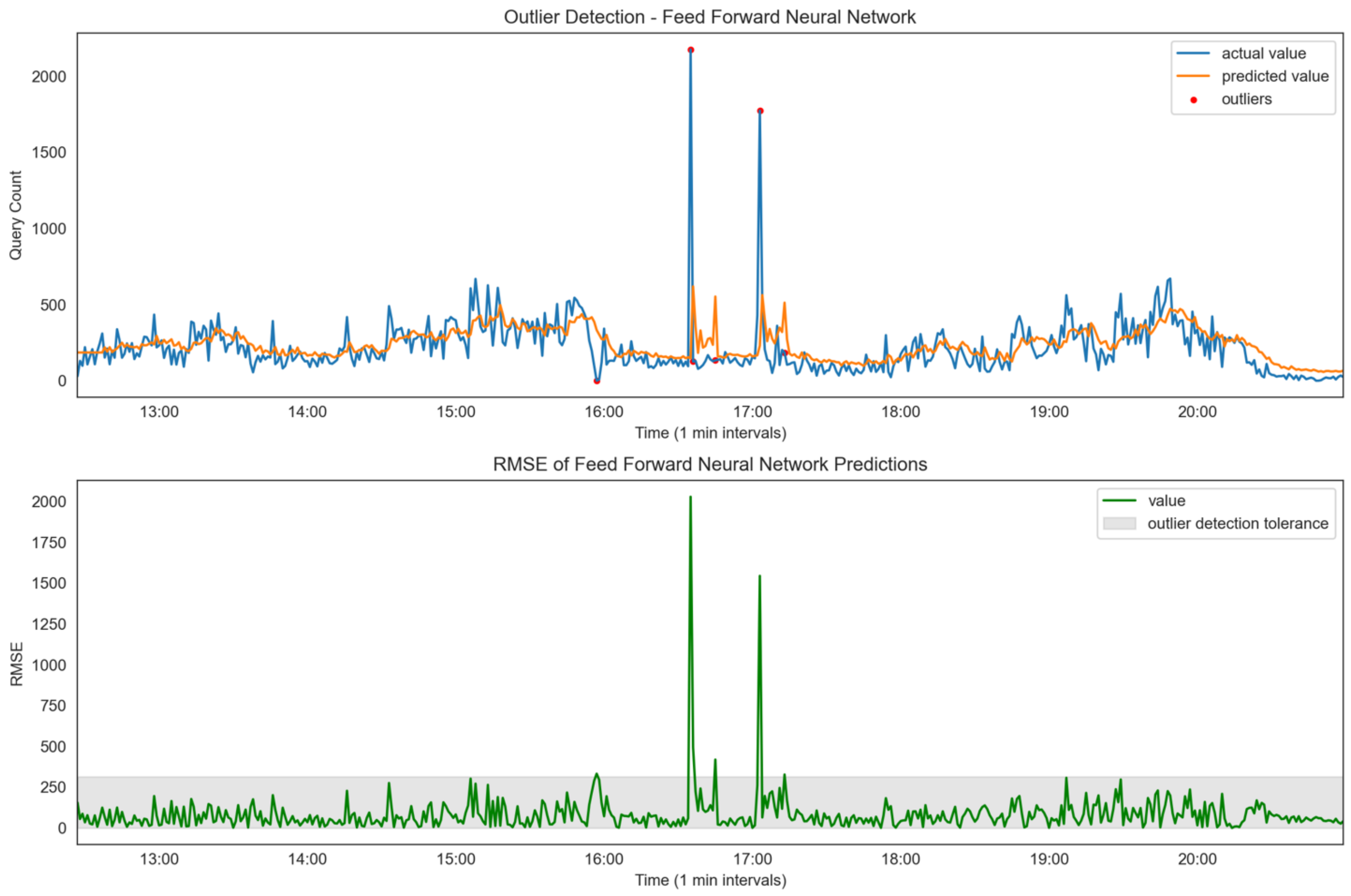


Figure 10 – Example of Outlier Detection by RMSE of Predictions Using a Feed Forward Neural Network on the Query Count Feature

b) Autoencoder

An autoencoder is a supervised deep-learning model which reconstructs a vector of inputs by learning an approximation to the identity function. Autoencoders first compress a vector input into a low dimensional representation, and then reconstruct that vector input from the low dimensional representation. In this way, an autoencoder learns the signal of the data. The number of time-intervals to use for the vector input was determined by grid-search where the metric for accuracy was the MSE. Shuffling the dataset and splitting into a train and test set allowed for the training of the autoencoder and assessment of its accuracy. This paper follows the method used by Harsha Kumara Kalutarage and Siraj Ahmed Shaikh [13] for outlier detection using an Autoencoder. This is to take the mean of the predictions made by the autoencoder that correspond to each time-interval, which results in a single predicted value for each time-interval. An example of outlier detection by RMSE of predictions using an autoencoder is shown in Figure 11.

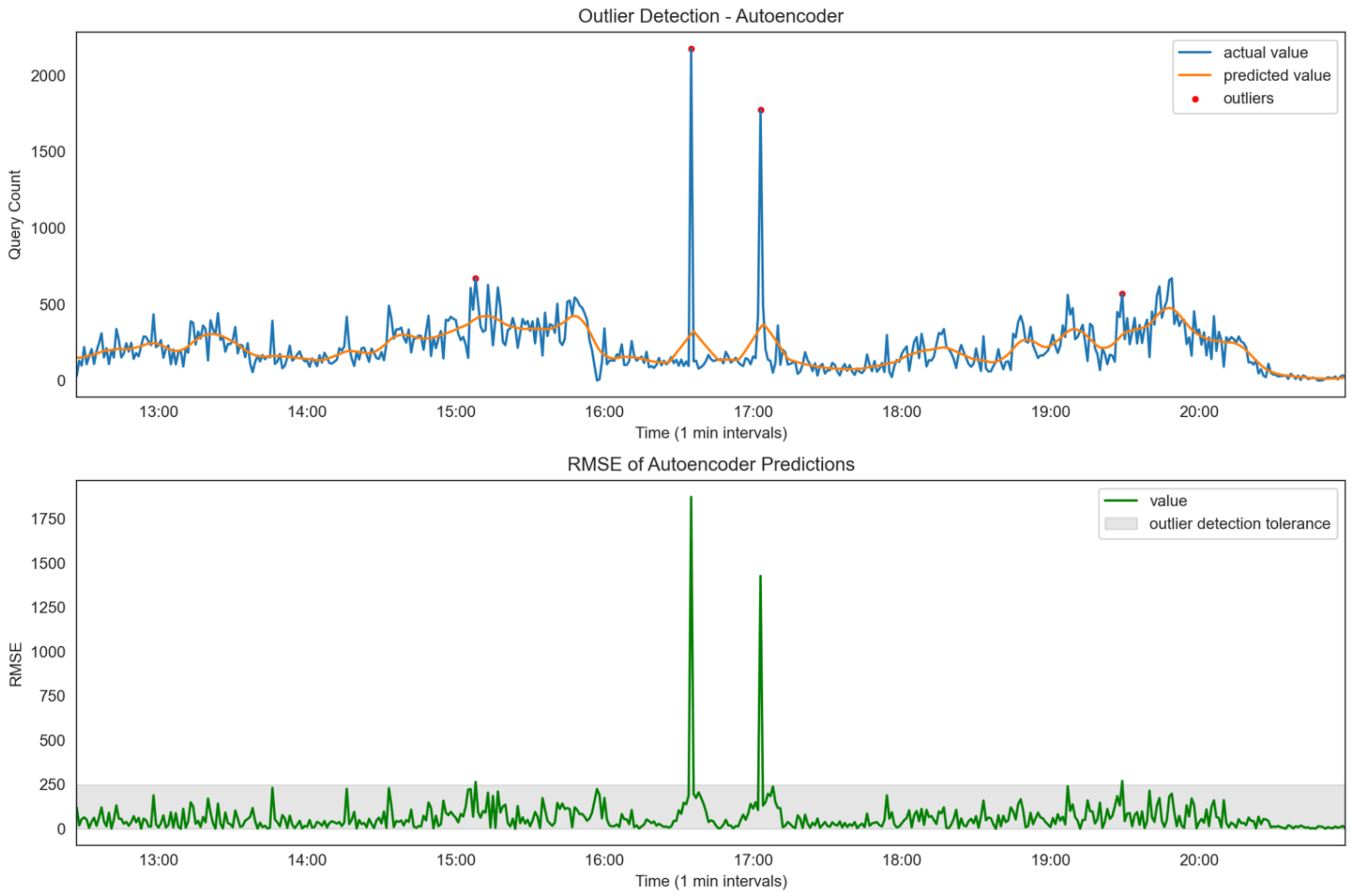
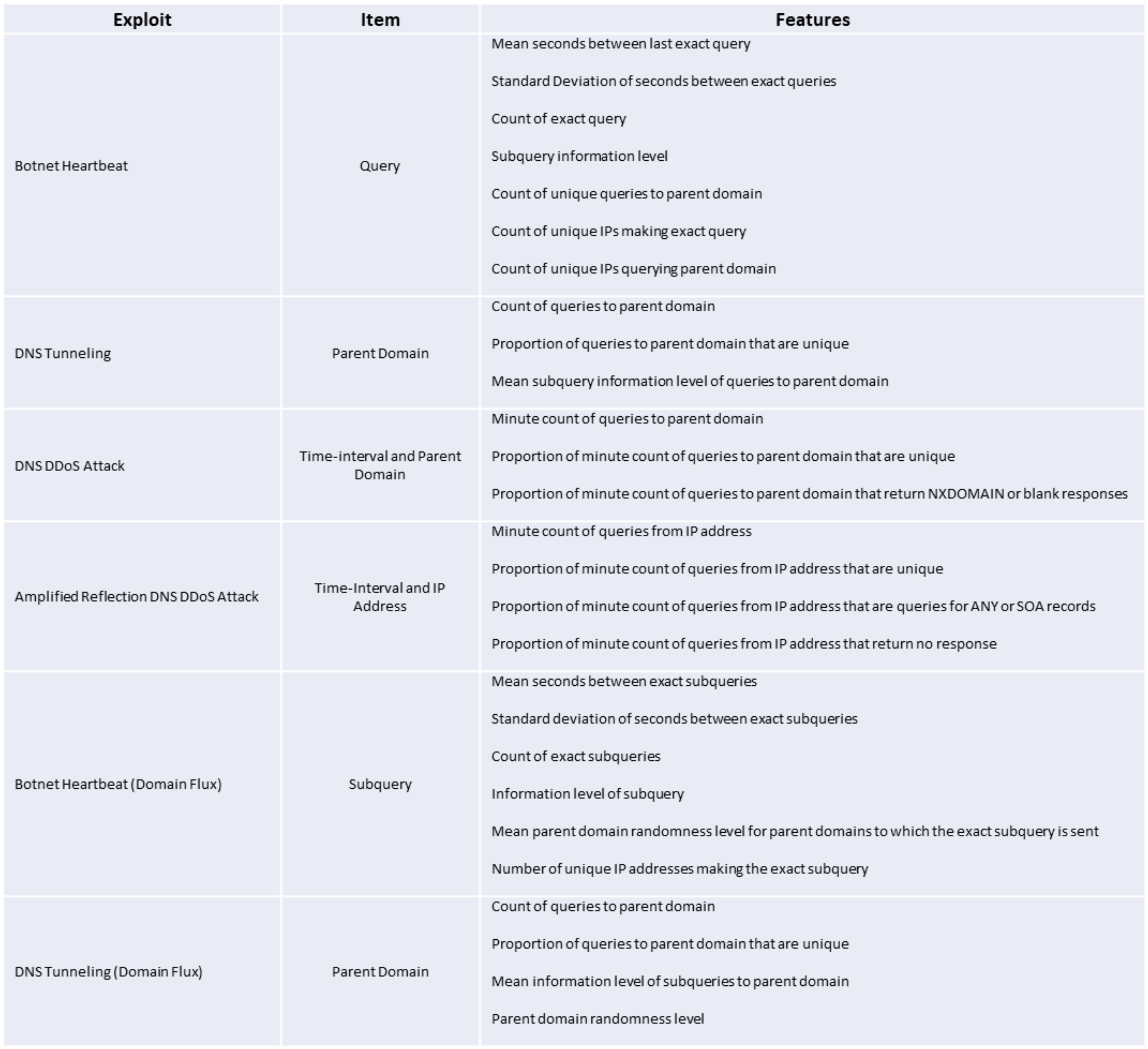


Figure 11 – Example of Outlier Detection by RMSE of Predictions Using an Autoencoder on the Query Count Feature

F. Exploit Detection – Methods and Implementation

To perform a check for each exploit and calculate a risk-score for each item that relates to the exploit, specific features were engineered to capture the traits of each exploit, where the feature's weight and value-range would determine the risk-score.
The relevant items and features used to identify symptoms of each exploit are given in Figure 12. The value ranges and feature weights are determined by the user. To ensure that feature values are calculated over an equal time period, the user sets a 24-hour window over which to calculate feature values for each item.



Exploit	Item	Features
Botnet Heartbeat	Query	Mean seconds between last exact query Standard Deviation of seconds between exact queries Count of exact query Subquery information level Count of unique queries to parent domain Count of unique IPs making exact query Count of unique IPs querying parent domain
DNS Tunneling	Parent Domain	Count of queries to parent domain Proportion of queries to parent domain that are unique Mean subquery information level of queries to parent domain
DNS DDoS Attack	Time-interval and Parent Domain	Minute count of queries to parent domain Proportion of minute count of queries to parent domain that are unique Proportion of minute count of queries to parent domain that return NXDOMAIN or blank responses
Amplified Reflection DNS DDoS Attack	Time-Interval and IP Address	Minute count of queries from IP address Proportion of minute count of queries from IP address that are unique Proportion of minute count of queries from IP address that are queries for ANY or SOA records Proportion of minute count of queries from IP address that return no response
Botnet Heartbeat (Domain Flux)	Subquery	Mean seconds between exact subqueries Standard deviation of seconds between exact subqueries Count of exact subqueries Information level of subquery Mean parent domain randomness level for parent domains to which the exact subquery is sent Number of unique IP addresses making the exact subquery
DNS Tunneling (Domain Flux)	Parent Domain	Count of queries to parent domain Proportion of queries to parent domain that are unique Mean information level of subqueries to parent domain Parent domain randomness level

Figure 12 – Table of the Relevant Items and Engineered Features Used to Check for Each Exploit in Exploit Detection

III. RESULTS

A. *Outlier Detection Results*

To achieve the best results, an appropriate method must be applied to each feature used in outlier detection. For the Mean Subquery Information Level and Mean Parent Domain Randomness Level features, values are not expected to experience periodic trends. It is therefore logical to identify outliers by deviation from a statistical average using statistical methods such as standard deviation or MAD. The advantage of using MAD over standard deviation is that MAD is a robust statistic, necessarily resistant to the effect of outliers. This is illustrated in Figure 13. MAD was therefore chosen for those 2 features.

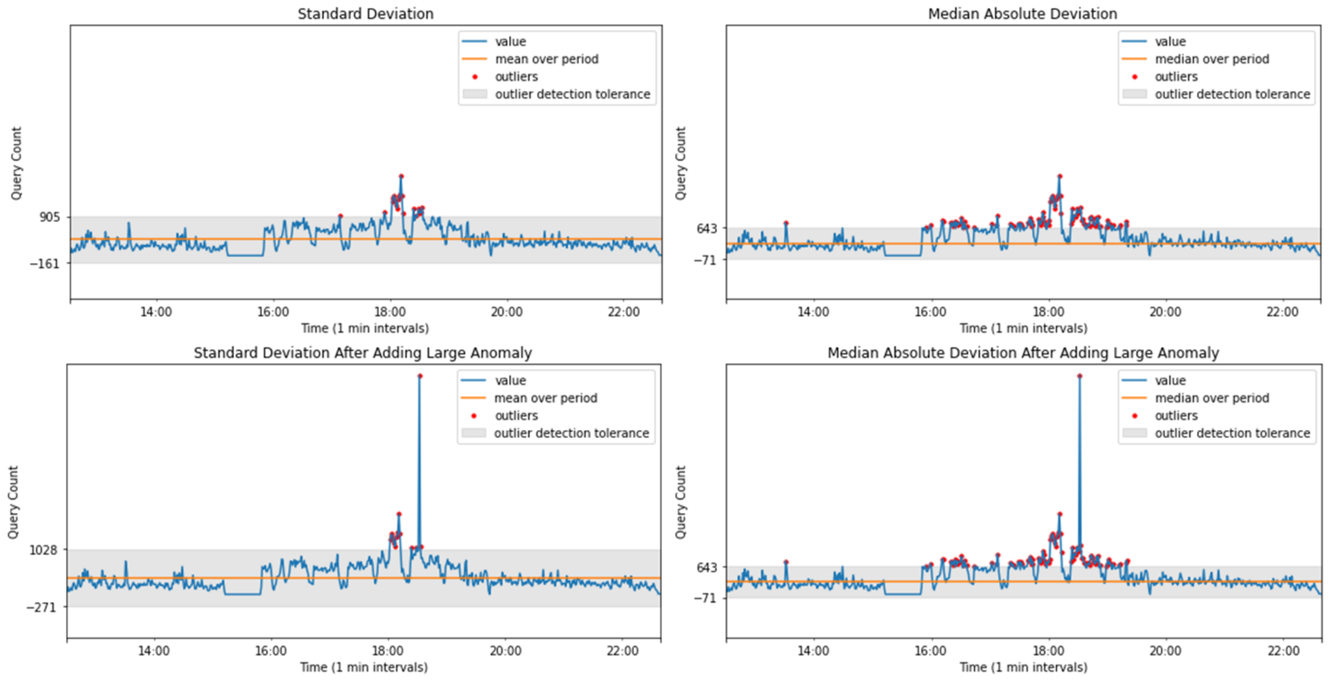


Figure 13 – Illustration to show the Robustness of MAD by Contrasting the Effect on the Standard Deviation and MAD Outlier Detection Tolerances by Introducing a Single Large Anomaly into the Query Count Feature. The Standard Deviation's Outlier Detection Tolerance is Significantly Changed while MAD's is Unchanged

For the Query Count, Count of Queries for Uncommon Record Types and Count of Queries with Uncommon Responses features, values are expected to experience periodic trends for working hours. Therefore, neither statistical measures nor their rolling variants are appropriate, as they fail to take the signal of the data into account. Using the predictive approach, an FFNN does learn the signal of the data by using previous values to predict a future value, though it has a clear deficiency. When an outlier does occur, the outlier has the effect of corrupting the input, and causing false positives in the subsequent predictions that use the outlying value in the input. The risk of misclassifying values around an outlier still exists when using the autoencoder, though the effect is much less pronounced, as shown in Figure 14.

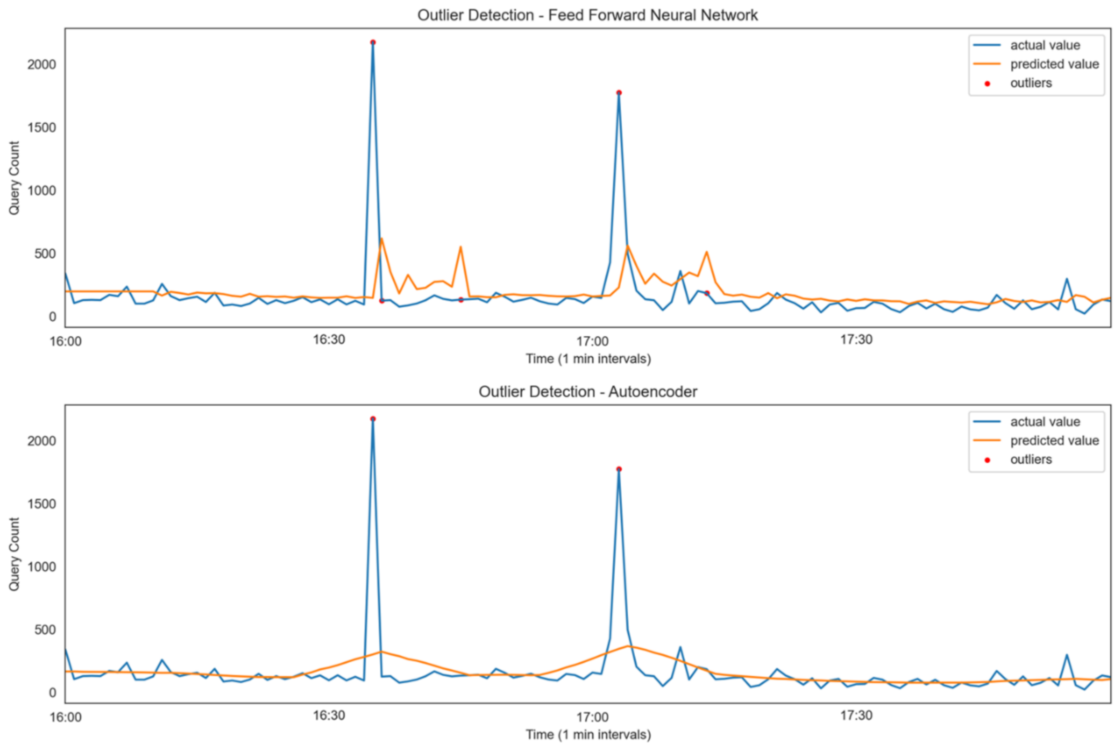
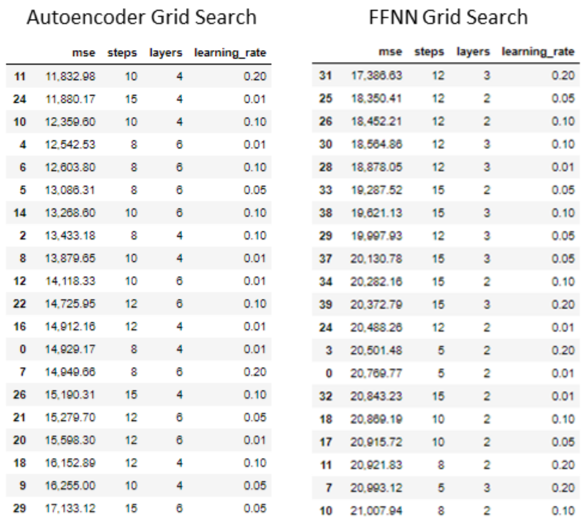


Figure 14 – Figure Comparing the Effect on Predictions that an Outlier has on a Feed Forward Neural Network to an Autoencoder. The Feed Forward Neural Network Makes Poor Predictions when an Outlier is Present in its Input, Creating False Positives, While the Autoencoder is Relatively Resilient to the Effect of an Outlier in its Input

Autoencoders also consistently achieved a higher accuracy than the FFNN models. The Autoencoder with the architecture that resulted in the lowest predictive MSE achieved an MSE of 11,832 for the query count feature, compared to the best FFNN's 17,386, as shown in Figure 15. Autoencoders were therefore chosen for those 3 features.



Autoencoder Grid Search

	mse	steps	layers	learning_rate
11	11,832.98	10	4	0.20
24	11,880.17	15	4	0.01
10	12,359.60	10	4	0.10
4	12,542.53	8	6	0.01
6	12,603.80	8	6	0.10
5	13,086.31	8	6	0.05
14	13,268.60	10	6	0.10
2	13,433.18	8	4	0.10
8	13,879.65	10	4	0.01
12	14,118.33	10	6	0.01
22	14,725.95	12	6	0.10
16	14,912.16	12	4	0.01
0	14,929.17	8	4	0.01
7	14,949.66	8	6	0.20
26	15,190.31	15	4	0.10
21	15,279.70	12	6	0.05
20	15,598.30	12	6	0.01
18	16,152.89	12	4	0.10
9	16,255.00	10	4	0.05
29	17,133.12	15	6	0.05

FFNN Grid Search

	mse	steps	layers	learning_rate
31	17,386.63	12	3	0.20
25	18,350.41	12	2	0.05
26	18,452.21	12	2	0.10
30	18,564.86	12	3	0.10
28	18,878.05	12	3	0.01
33	19,287.52	15	2	0.05
38	19,621.13	15	3	0.10
29	19,997.93	12	3	0.05
37	20,130.78	15	3	0.05
34	20,282.16	15	2	0.10
39	20,372.79	15	3	0.20
24	20,488.26	12	2	0.01
3	20,501.48	5	2	0.20
0	20,769.77	5	2	0.01
32	20,843.23	15	2	0.01
18	20,869.19	10	2	0.10
17	20,915.72	10	2	0.05
11	20,921.83	8	2	0.20
7	20,993.12	5	3	0.20
10	21,007.94	8	2	0.10

Figure 15 – Figure to Compare Grid Search Results for Different Autoencoder and FFNN Architectures, Showing the Lower MSEs Achieved by Autoencoders

B. *Exploit Detection Results*

Since the dataset contained no obvious DNS exploits, the success of exploit detection had to be determined by whether the items returned with high risk-scores shared symptoms of DNS exploits, even though they themselves were likely benign. Inspecting high-risk items for each exploit revealed desirable results. High-risk items in the Botnet heartbeat check included a benign heartbeat to Baltimore Community College servers, likely generated by software providing remote access. The DNS Tunnelling check revealed the domain "in_addr.arpa", which is used legitimately for performing reverse lookups. Large numbers of DNS queries with high subquery information levels were also seen requesting miscellaneous subdomains of Microsoft and Apple, possibly for error reporting. DNS DDoS attacks, being very specific and based on high-levels of traffic in an interval, resulted in a check that returned no items with a high risk-score, as would be expected. The check for Amplified reflection DDoS attacks was complicated by dataset issues. The fact that over 99% of the data were DNS queries that contained no response, raised item risk-scores for this check. The Domain Flux check did not return high risk-scores, and effectively differentiated itself from the DNS Tunnelling check, for example, by not classifying "in_addr.arpa" as high-risk, as a result of the parent domain not appearing to be randomly generated.

C. *Parent Domain Randomness Level Feature Results*

The results of the LSTM which determines the Parent Domain Randomness Level feature values, which is used in both outlier and exploit detection, showed it to be an effective model. The LSTM achieved 92.5% accuracy in classifying randomly generated strings and real domain names, compared to 91.7% from an FFNN, indicating that characters in domain names from the dataset used may have some long-term dependencies on previous characters, although the improvement was not major. Grid search found that an LSTM with a hidden size of 100 achieved the best accuracy. The confusion matrix showed no obvious deficiencies in classification, with misclassifications being made up fairly evenly of false positives and false negatives.

IV. Discussion

A. *Outlier Detection Discussion*

Analysis of the efficiencies and deficiencies of each model indicated that autoencoders and MAD were the most effective outlier detection approaches for their relevant key features. The implementation into a dashboard provides a flexible and accessible solution where outliers can be visualized and downloaded as a csv file. Furthermore, the analyst has control to customize the size of time-intervals, so that checks can be performed with different levels of precision if desired. With increased data, and the dashboard's feature for removing predetermined anomalous time-intervals from training data, the autoencoder is also expected to improve as an outlier detection tool. Both methods provide an anomalous-rating, by RMSE or by absolute deviation. This is desirable for an analyst, who can then prioritise their investigations. An alternative deep-learning model for outlier detection is an LSTM, but this was not deemed suitable for this task. LSTMs are effective when they can take advantage of long-term dependencies. In anomaly detection, they would therefore be best used in data with a strong temporal element. Over the 2 days of available data, however, there was no such temporal element, and while other DNS log data may exhibit such a pattern, this cannot be assumed.

B. *Exploit Detection Discussion*

The results of exploit detection validated the use of a risk-score and the effectiveness of the engineered features. There are benefits to using a risk-score for exploit detection, which is evident from the fact that a similar approach is used by cybersecurity providers such as Splunk [14]. It allows the user to prioritise certain items, and build a base of evidence for their investigations rather than relying on binary classification. Once a high-risk item is identified, items with lower risk-scores that share feature values with the high-risk item, such as time of day or parent domain, can be subsequently investigated. The value-ranges and weightings for each feature being determined by the user is both an advantage and disadvantage. It has the benefit of utilising the analyst's domain knowledge of their network, and provides flexibility, however it also risks that the chosen inputs may be inappropriate. To avoid placing the responsibility of setting value-ranges and weights on the user, an alternative approach could be to simulate attacks and incorporate them into data. Assuming that the dataset is guaranteed to be free of DNS exploits, models could be trained to classify items as exploits based on the training data. Models would need to be simple and interpretable so that the classification criteria could be verified by the analyst. However, this was not deemed to be a suitable approach, as training models on simulated values risks that models only learn limited classification rules that exclude key features. Assuming that a benign dataset can be provided by an analyst is also likely to be unrealistic for large networks, and furthermore, the many instances of benign DNS activity that display symptoms of exploits would be likely to corrupt the training data for the model.

Inspecting the features engineered for exploit detection, it might be suggested that engineering features based on IP addresses and Session IDs (UIDs) would prove even more effective than those chosen. However, these features were deliberately avoided due to the ease with which an attacker can evade detection if these features are relied upon. IP spoofing is a simple workaround for a competent attacker, and similarly, many UIDs can be created by opening many instances of the same application in an exploit like a DNS DDoS Attack, making the UID feature unreliable. The exceptions are Botnet Heartbeats, since the IP address would not be spoofed as the machine sending the DNS query requires a response, and an Amplified Reflection DDoS Attack, where the spoofed IP address is required to be a victim's IP address.

C. Limitations of Solution

There are certain limitations to this anomaly detection solution that should be noted. Both outlier and exploit detection use a feature based on the randomness level of a query's parent domain, which is a score determined by an LSTM. It should be considered that the LSTM has been trained on a dataset of the top 1 million sites from Alexa.com, though this dataset may be inappropriate for countries that do not use the English language. In exploit detection, the proposed features to detect a DNS DDoS Attack are largely based on counts of queries over time, which would be effective for a small botnet. However, if an attacker has control over a large botnet, many machines could send relatively small numbers of queries, which would not be captured. Finally, the solution may need to be updated as new DNS exploits are devised, or existing DNS exploits varied. For example, there is an evolution of the Domain Flux evasive exploit, where malware algorithmically generates dictionary words rather than randomly generated strings, which could evade the solution's anomaly detection methods in its current state.

V. Conclusion

This paper addresses the nuanced challenges of anomaly detection in DNS logs by proposing a flexible, two-part solution, leveraging powerful deep-learning solutions and robust statistical measures in one approach, and knowledge of DNS exploits combined with a risk-scoring system in another. The solution provides an effective, accessible and customisable tool for a user to gain insights into their network, and provides a base on which a user can compile evidence and monitor irregular network activity. Network technology, and particularly cybersecurity, are constantly evolving fields. Effective diagnosis and successful investigation are contingent on contextual knowledge and technical expertise; the tool's relative dependence upon the user's domain knowledge is reflective of this reality.

VI. References

[1] Security Today, 13 January 2020, *16 Internet of Things (IoT) Statistics*, viewed 26 August 2020, https://securitytoday.com/Articles/2020/01/13/The-IoT-Rundown-for-2020.aspx?Page=2

[2] IANA, 20 July 2020, *Domain Name System (DNS) Parameters*, viewed 26 August 2020, https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml

[3] EfficientIP, 21 November 2019, *Why Protecting DNS Requires More Than Firewalls*, viewed 26 August 2020, https://www.efficientip.com/protecting-dns-more-than-firewalls/

[4] CloudFlare, 2020, *What is IP Spoofing*, viewed 26 August 2020, https://www.cloudflare.com/learning/ddos/glossary/ip-spoofing/

[5] Lockheed-Martin, 2020, *The Cyber Kill Chain*, viewed 26 August 2020, https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html

[6] Mike Sconzo, 2012, *SecRepo.com - Samples of Security Related Data*, viewed 26 August 2020, https://www.secrepo.com/

[7] NETRESEC, 2012, *Capture files from Mid-Atlantic CCDC*, viewed 26 August 2020, https://www.netresec.com/?page=MACCDC

[8] The Bro Project, 1 October 2016, *Bro Documentation Release 2.4.1*, viewed 26 August 2020, http://www.ncsa.illinois.edu/People/jazoff/bro-2.4.1.pdf page 83

[9] C. E. Shannon, October 1948, *A Mathematical Theory of Communication*, viewed 26 August 2020, http://people.math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf

[10] IANA, 2020, *Root Zone Database*, viewed 26 August 2020, https://www.iana.org/domains/root/db

[11] Sid Ghodke, 16 January 2018, *Alexa Top 1 Million Sites*, viewed 26 August 2020, https://www.kaggle.com/cheedcheed/top1m

[12] Whogohost, 2020, *Valid Domain Name Characters*, viewed 26 August 2020, https://www.whogohost.com/host/knowledgebase/308/Valid-Domain-Name-Characters.html

[13] Harsha Kumara Kalutarage and Siraj Ahmed Shaikh, 2018, *Feature Trade-off Analysis for Reconnaissance Detection*, viewed 26 August 2020, https://www.worldscientific.com/doi/abs/10.1142/9781786345646_005

[14] Splunk Enterprise Security, 28 April 2020, *Analyze risk in Splunk Enterprise Security*, viewed 26 August 2020, https://docs.splunk.com/Documentation/ES/6.2.0/User/RiskScoring