2018

Comparing the Effectiveness of Different Classification Techniques in Predicting DNS Tunnels

Patrick Walsh
Technological University Dublin

Comparing the effectiveness of different classification techniques in predicting DNS tunnels

Patrick Walsh

A dissertation submitted in partial fulfilment of the requirements of
Dublin Institute of Technology for the degree of
M.Sc. in Computing (Security & Forensics)

September, 2018

I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Security & Forensics), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the test of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Dublin Institute of Technology and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

Signed: ________________________________

Patrick Walsh

Date:

ABSTRACT

DNS is one of the most widely used protocols on the internet and is used in the translation of domain names into IP address in order to correctly route messages between computers. It presents an attractive attack vector for criminals as the service is not as closely monitored by security experts as other protocols such as HTTP or FTP. Its use as a covert means of communication has increased with the availability of tools that allow for the creation of DNS tunnels using the protocol.

One of the primary motivations for using DNS tunnels is the illegal extraction of information from a company's network. This can lead to reputational damage for the organisation and result in significant fines – particularly with the introduction of General Data Protection Regulations in the EU.

Most of the research into the detection of DNS tunnels has used anomalies in the relationship between DNS requests and other protocols, or anomalies in the rate of DNS requests made over specific time periods. This study will look at the characteristics of an individual DNS requests to see how effective different classification techniques are at identifying tunnels. The different techniques selected are Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), and Support Vector Machine (SVM).

The effectiveness of the different techniques will be measured and compared to see if there are statistically significant differences between them using a Cochran's Q test. The results will indicate that DT, RF and SVM, are the most effective techniques at categorising DNS requests, and that they are significantly different to the other models.

Key Words: DNS Tunnel, Logistic Regression, Support Vector Machine, Decision Tree, Random Forest, Cochran's Q Test

ACKNOWLEDGEMENTS

I would like to thank my supervisor **Dr Seán O'Leary** for helping to put shape on the dissertation and for suggesting areas of research that the project could encompass, and **Dr Luca Longo** for providing me with the structure and process required in order to successfully complete a thesis.

I would also like to thank my family for their patience and understanding during the past two years, and in particular during the last couple of months while I completed this thesis.

Contents

TABLE OF TABLES

TABLE OF FIGURES

List of Acronyms

DNS – Domain Name System

HTTP – Hypertext Transfer Protocol

FTP – File Transfer Protocol

IP – Internet Protocol

TCP – Transmission Control Protocol

ARPANET – Advanced Research Projects Agency Network

AGD – Algorithmically Generated Domains

IDS – Intrusion Detection Systems

LR – Logistic Regression

DT – Decision Tree

RF – Random Forest

SVM – Support Vector Machine

RBF – Radial Basis Function

1 INTRODUCTION

The DNS protocol is used to resolve host names to IP addresses. It acts much like a telephone directory for the Internet, without it users would need to remember the IP address of every server they send messages to. Although not designed as a data communications channel, it can be used as such by individuals using a DNS tunnel to avoid paying for internet services, to access restricted sites, or to illegally extract information from a network. Identifying DNS tunnels can be a difficult task as attackers become more creative in their design and implementation of tunnels, and take steps to avoid detection.

This paper will briefly review the history of DNS and how it works, how it can be used as a covert channel, and the different techniques employed to identify DNS tunnels. A number of different classification models will then be developed that can predict that a DNS request is part of a tunnel by looking at the characteristics of the request. Comparisons between the different models developed will be made to establish which one is the most effective at detecting tunnels and to see if any differences between them are significant.

1.1 Background

1.1.1 TCP/IP

The TCP/IP protocol was first developed as part of ARPANET, and is used to reliably transmit messages between hosts on a network or networks. TCP is used to create channels of communication between hosts, and determine how messages are divided up into smaller packets before they are transmitted from a source to a destination host. IP is used as an addressing mechanism to ensure that messages are routed to the correct destination. Each router on the network checks the message's destination IP address to determine where it should be sent next (Mehta, 1999).

An IP address is made up of a series of numbers that create a unique address that is used to identify different hosts on a network. If someone wanted to transmit a message from host A to host B, they would need the IP address related to host B, and set that as the destination for the message. IP addresses can be difficult for people to remember, so instead host names are used which are automatically translated to IP addresses before the message is sent. E.g. www.dit.ie is translated to 147.252.25.70 before any messages are sent to the server. Initially, the host name translation was done by looking up a hosts.txt on the local filesystem. Every time a new host was added to the network, the hosts.txt file would need to be updated and sent to all other hosts so that they could communicate with the new host using the name. By the early 1980s, the number of hosts being added to the network was increasing exponentially, and modifying the hosts.txt file and updating the local file system on each existing server when a new host was added became impractical. This led to the creation of DNS, a distributed naming system which is used to resolve host names to IP addresses (Pope, Warkentin, Mutchler, & Luo, 2012).

1.1.2 Domain Name System

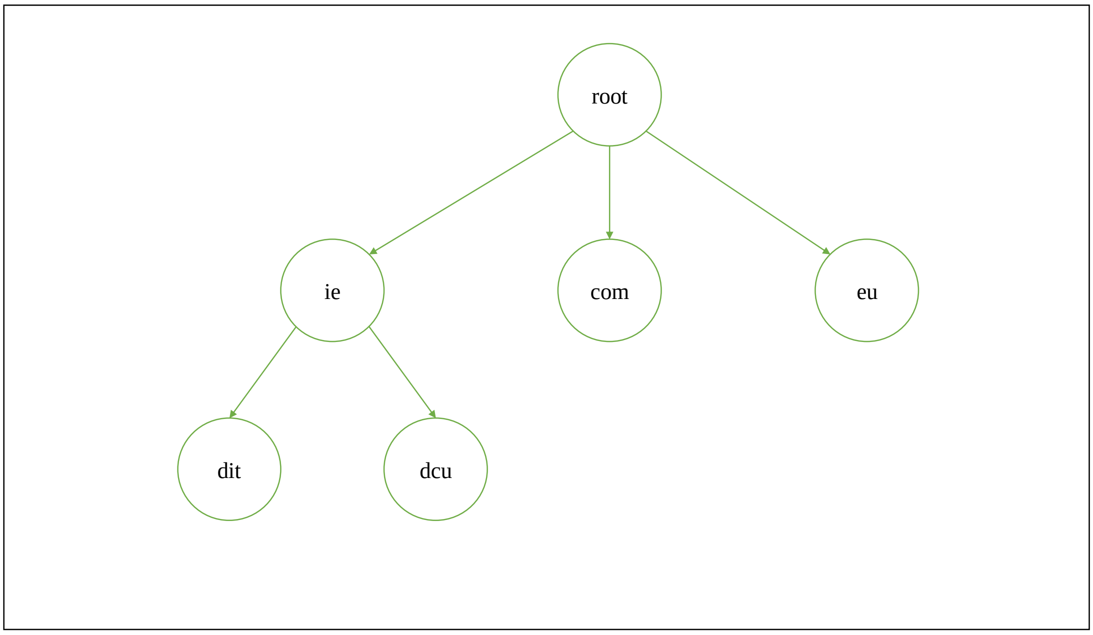
DNS is a fundamental protocol on the Internet used to translate domain names to IP addresses. It is a distributed hierarchical service with different DNS servers administering different parts of the database. It is formulated as a tree structure, with each node on the tree (except the root node) having a label and parent node. Figure 1.1 DNS Tree Structure illustrates the basic setup.

In this example, dit.ie is the fully qualified domain name of the "dit" node whose parent is "ie" and whose grand-parent is the root node. Different zones within the database are maintained by different organisations.

When a new node is added to a zone, the controlling organisation adds it to the database and updates multiple servers to make the node accessible to different clients on the network (Wright, 2012).

When a DNS server receives a request for a domain it doesn't know, it re-directs it to another DNS server to be resolved (Callahan, 2013). For example, if a client on a network requests the IP address of www.dit.ie, it will first make a request to a local DNS server. If the local DNS server does not have the IP address, it will send a message to another DNS server to resolve the request, e.g. the "root" DNS server.

Figure 1.1 - DNS Tree Structure

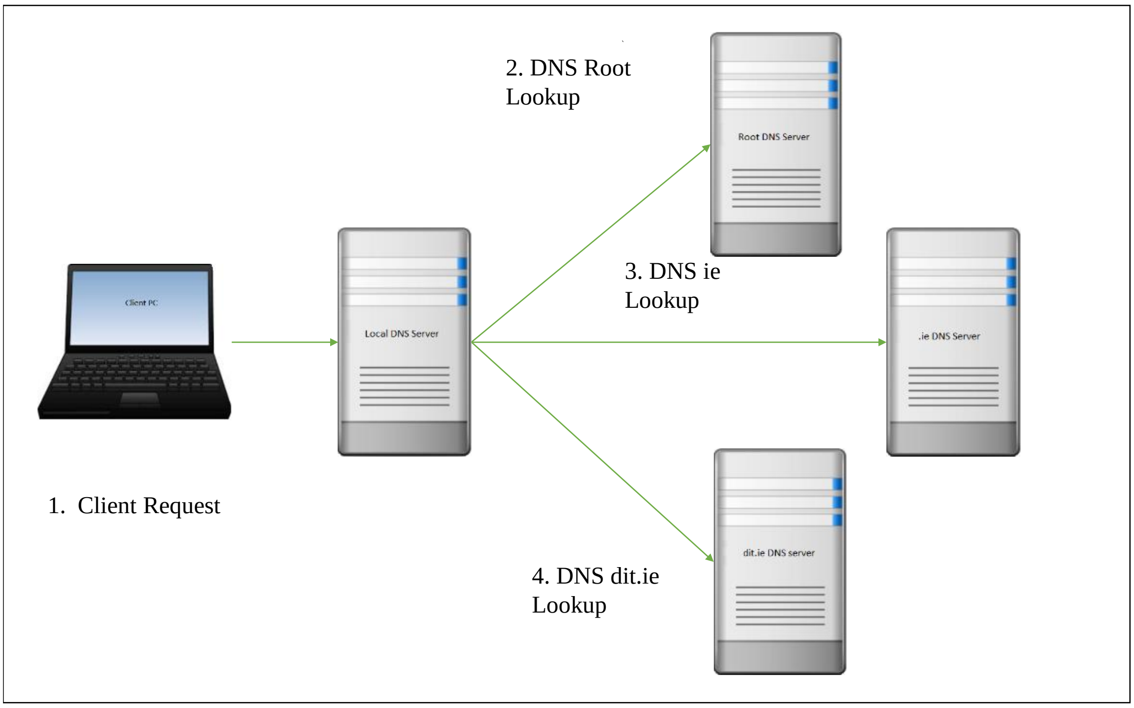


If this server does not have the IP address and is not the authority for the sub domain, it will either forward the request to the relevant authority (e.g. “ie” domain) or send a message back to the local DNS server to indicate that they should re-direct to the authoritative server. This continues until the request reaches the authoritative DNS server for the domain, at which point the IP address is returned to the client. This is illustrated in Figure 1.2 DNS Lookup.

Steps to Resolve IP Address:

1. A client requests the IP of www.dit.ie.
2. The local DNS server does not have the IP address and so forwards the request to the “root” DNS server. The “root” server responds to the local DNS server indicating that they should direct the query to the “ie” DNS server.
3. The local DNS server forwards the request to the “ie” DNS server. The response indicates that they should go to the “dit.ie” DNS server.
4. The local DNS server forwards request to the “dit.ie” DNS server that responds with the resolved IP address. The local DNS server then responds to the client with the IP address.

Figure 1.2 - DNS Lookup



1.1.3 DNS Tunnelling

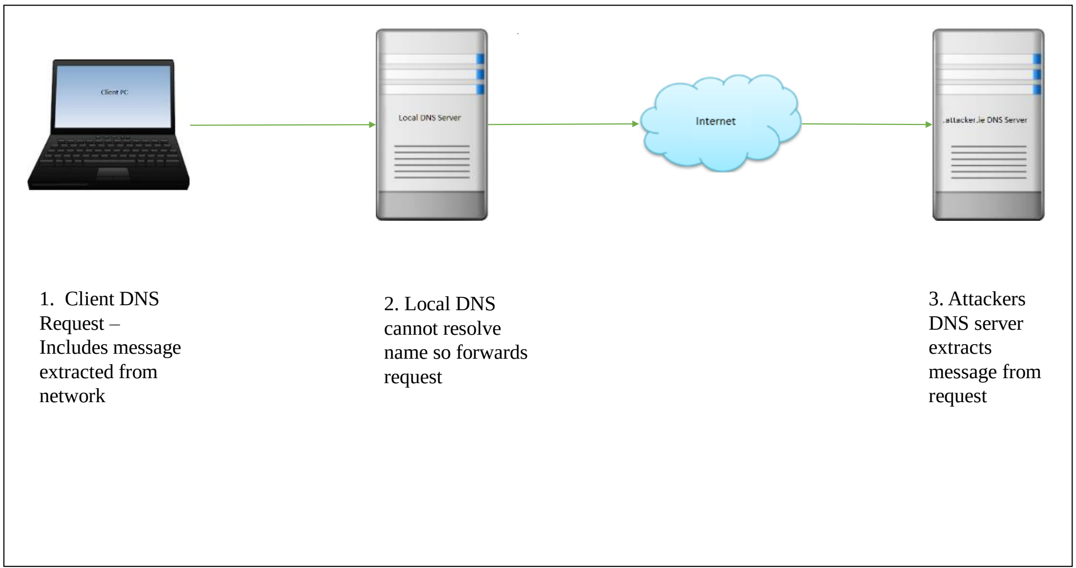
DNS was not built with security in mind and as a result it can be vulnerable to various attacks, such as DNS forgery and DNS tunnels, as the requests don’t usually receive the same amount of scrutiny from network security teams who are usually more focused on HTTP or FTP traffic. This can leave the protocol vulnerable to attack from un-authorised individuals looking to exploit the security weaknesses in the system (Wright, 2012).

The protocol cannot be simply switched off as it is ubiquitous in nature and is required by many other services to allow them to operate efficiently – for both legitimate and illegal activities. As such, DNS requests are rarely blocked and have become a target for use in covert channels that use it to exfiltrate data from a network by setting up a DNS tunnel (Tien & Kavakli, 2008).

1.1.4 Data Exfiltration

A DNS exfiltration attack works by installing malware on a computer that sends DNS requests for a domain controlled by an attacker. The requested domain name contains data from the network gathered by the malware. The network router does not recognise the domain, and so forwards the request to higher level DNS servers. The request eventually reaches the attackers DNS server, at which point the data is extracted and a response sent to the client (Van Antwerp, 2011). Only a relatively small amount of information can be extracted in each request but if left running over even a moderate amount of time, this can lead to significant data loss.

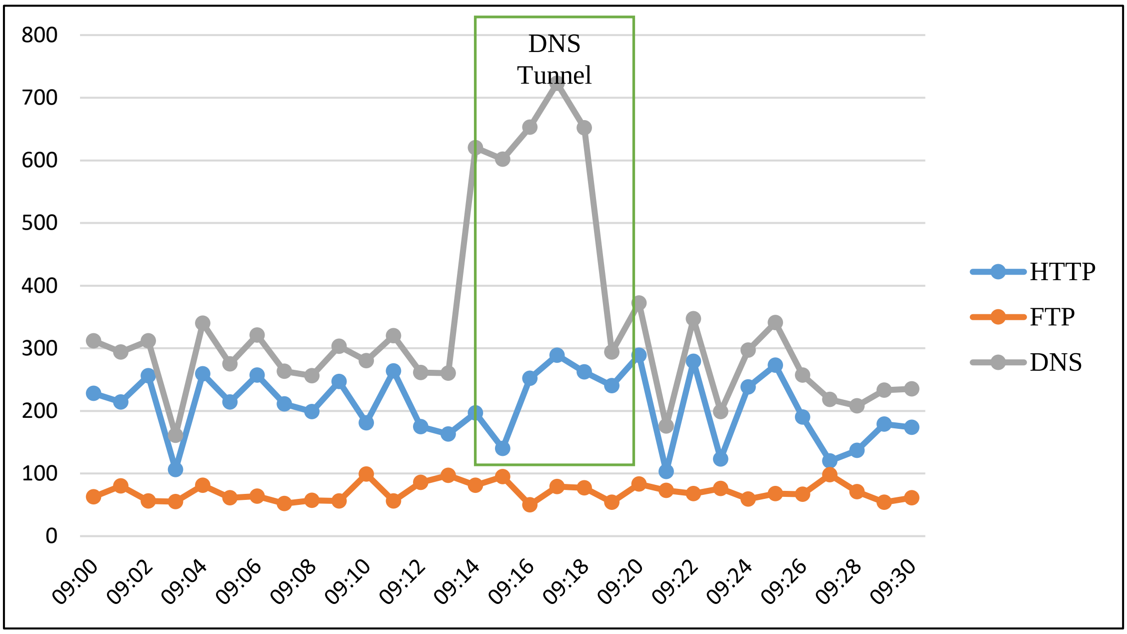
Figure 1.3 - Data Exfiltration



1.1.5 Identifying DNS Tunnels

Detecting DNS tunnels that have a high throughput can be easily achieved by observing a spike in DNS requests with no corresponding spike in HTTP or FTP traffic (Sheridan & Keane, 2015). Normally, DNS requests would be made as a pre-cursor to other types of requests, such as HTTP or FTP. Therefore, you would expect to see a correlation between DNS and other network traffic messages. This is illustrated (using example requests) in Figure 1.4 – Spike in DNS Traffic, where there is an obvious increase in DNS traffic between 9:14 and 9:20 with no corresponding increase in HTTP or FTP traffic. This should be of concern to network security analysts who would then need to identify the source and destination of the DNS requests and determine if they are genuine or part of a DNS tunnel. Once it is confirmed that they are part of a tunnel, rules can be put in place to prevent any further DNS requests being sent to that domain.

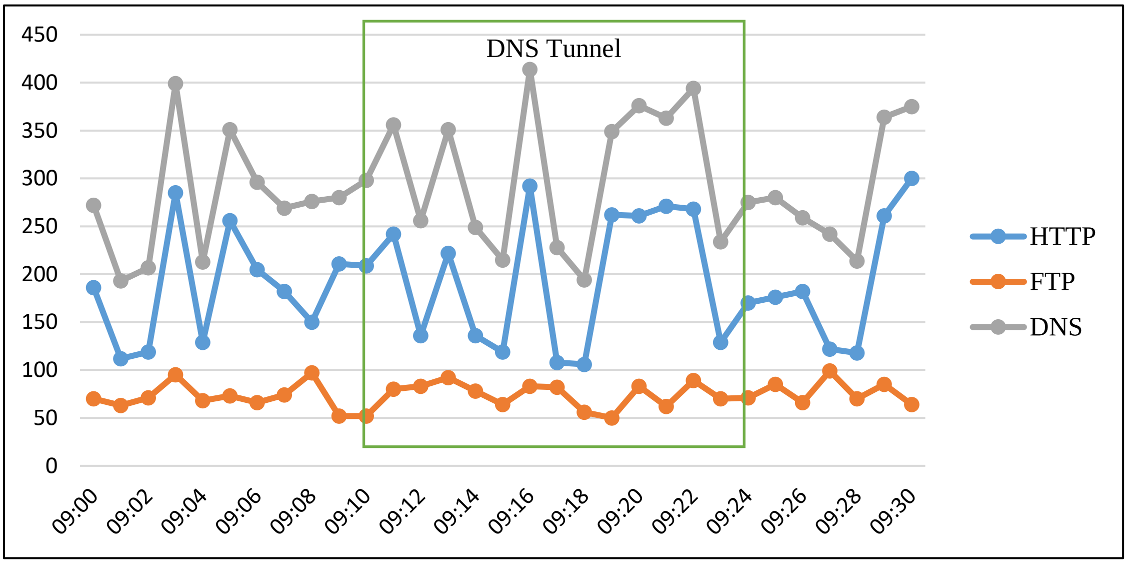
Figure 1.4 - Spike in DNS Traffic



Further investigation may then be required to determine what data was extracted from the network and if clients and regulatory bodies need to be notified e.g. if personal or bank details were extracted.

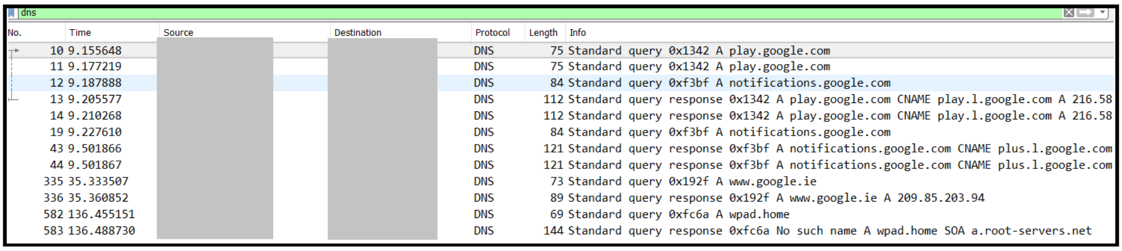
Identifying DNS tunnels with a high level throughput by comparing network traffic protocols can be very effective, however if the attackers adopt a Low and Slow approach to data exfiltration, then this kind of identification becomes much more difficult. For example, Figure 1.5 Low Level Throughput illustrates the existence of a tunnel between 9:10 and 9:24, but it is much more difficult to observe a spike in DNS traffic. In this instance, the increase in DNS traffic is much more subtle and reflects a Low and Slow approach to data exfiltration where the attacker keeps the number of DNS requests below a certain threshold to avoid detection. Additionally, if the malware implementing the tunnel matches existing network traffic patterns, then real time detection can be even more difficult. For example, if an attacker extracted information from a network over weeks or months, they could go undetected if they match existing traffic and stay below detection thresholds (Kalutarage, 2013). In these circumstances, simply monitoring DNS traffic and comparing it to HTTP and FTP traffic to identify tunnels can prove difficult. This is particularly true on an organisation's network that can generate large amounts of DNS and HTTP traffic. Recording and analysing this amount of traffic to identify minor changes can be costly and time consuming.

Figure 1.5 - Low Level Throughput



Even if spikes in DNS traffic are identified, they can often be false positives. For example, they can be caused by internet browsers that can generate DNS traffic without any corresponding HTTP traffic. If you open *Google Chrome* and monitor the DNS traffic in *Wireshark*, you will see DNS requests from the browser even when the session is inactive.

Figure 1.6 - Figure 1.6 - DNS Requests from Inactive Browser Session



dns

No.	Time	Source	Destination	Protocol	Length	Info
10	9.155648			DNS	75	Standard query 0x1342 A play.google.com
11	9.177219			DNS	75	Standard query 0x1342 A play.google.com
12	9.187888			DNS	84	Standard query 0xf3bf A notifications.google.com
13	9.205577			DNS	112	Standard query response 0x1342 A play.google.com CNAME play.l.google.com A 216.58
14	9.210268			DNS	112	Standard query response 0x1342 A play.google.com CNAME play.l.google.com A 216.58
19	9.227610			DNS	84	Standard query 0xf3bf A notifications.google.com
43	9.501866			DNS	121	Standard query response 0xf3bf A notifications.google.com CNAME plus.l.google.com
44	9.501867			DNS	121	Standard query response 0xf3bf A notifications.google.com CNAME plus.l.google.com
335	35.333507			DNS	73	Standard query 0x192f A www.google.ie
336	35.360852			DNS	89	Standard query response 0x192f A www.google.ie A 209.85.203.94
582	136.455151			DNS	69	Standard query 0xfc6a A wpad.home
583	136.488730			DNS	144	Standard query response 0xfc6a No such name A wpad.home SOA a.root-servers.net

The approach this paper will take to identify DNS tunnels is to look at the characteristics of the individual requests and then to use different classification techniques, Logistic

Regression (LR), Decision Tree (DT), Random Forest (RF), and Support Vector Machine (SVM), to make predictions on whether or not the request is part of a tunnel. The models will be built from existing datasets of DNS requests that contain both legitimate and tunnel related traffic. These datasets are publically available and represent typical DNS activity associated with the tunnelling tools *Iodine* and *Powercat*. The results from each model will then be compared to see if there are statistically significant differences in the predictions made between the various models.

1.1.6 Research Project

1.1.6.1 Research Question

Is there a statistically significant difference in the prediction of DNS tunnels using the classification techniques Logistic Regression, Decision Tree, Random Forest, and Support Vector Machine?

Comparisons between the various techniques will be made using the Sensitivity, Specificity, Accuracy, and Precision of the models produced. These values will be calculated by creating a confusion matrix for the results – True Positive, True Negative, False Positive, and False Negative. To compare the classification techniques used, the output of True Positive and True Negative will be examined using a Cochran's Q test to see if there are statistically significant differences in the results obtained. If the test indicates that there are significant differences between all of the models, then further tests between pairs of classification techniques will be carried out.

The first step in creating the classification models will be to examine the candidate characteristics in the table below to see if they are appropriate predictive variables for use in classification models. The justification for the inclusion of variables will be based on previous research and statistical analysis of the requests in the dataset.

Once the predictive variables are identified from the candidate characteristics, the different classification models will be built to predict which requests are part of a DNS tunnel.

Table 1.1 - DNS Request Characteristics

Name	**Description**
Length	The length of the domain being requested
Sub Domain Length	The length of the first sub-domain in the request
Sub Domain Count	The number of sub domains in the request
Character Distribution	The number of different characters used in the request
Entropy	The min number of characters required to encode the request

1.1.6.2 Logistic Regression

LR models calculate the probability of an outcome based on either categorical or continuous predictor variables. The first step is to create a boundary equation to separate the input data into multiple regions (two for binary logistic regression). The goal is to create an equation that maximises the likelihood that the training data will be placed into the correct region. From the boundary equation, the odds ratio is calculated using the exponential function e^x, and from this the probability of and individual record being in the target region can be determined (Peng, Lee, & Ingersoll, 2002).

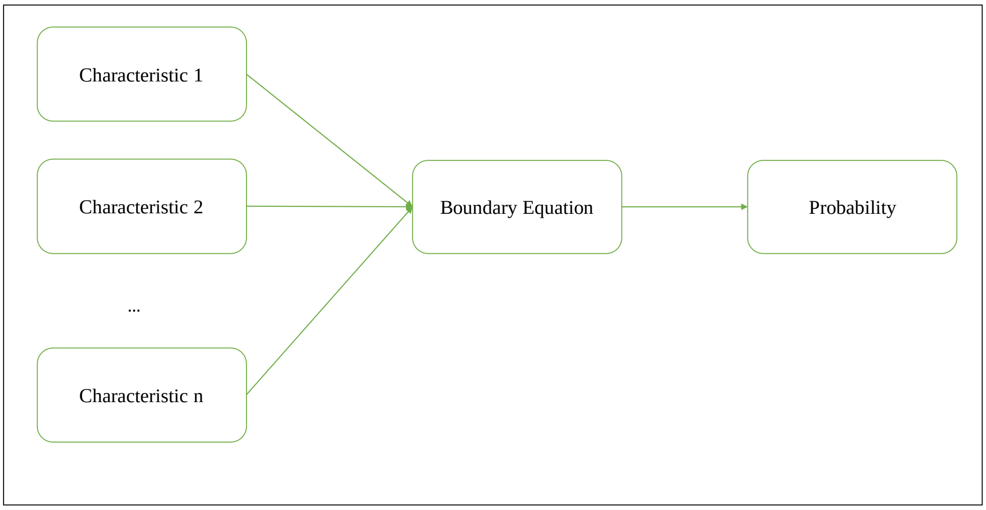
For this research, the models will be built using the DNS request characteristics to develop a boundary equation to predict the probability that the request is part of a DNS tunnel. The output from the model will be a boundary equation in the following form:

output = b0 + (b1 x c1) + (b2 x c2) + …. + (bn x cn)

- c1 to cn represent the score of the different characteristics of the DNS request being examined, e.g. length = 50
- b1 to bn represent the impact those characteristics have on whether the request is part of a tunnel as predicted by the model, e.g. the weighting applied to the length of the request
- b0 represents a constant for the likely outcome, e.g. the probability that a request is part of a tunnel regardless of the characteristics

The output of the boundary equation could be positive or negative, lying somewhere between $(-\infty, +\infty)$. If the output is between $(-\infty, 0)$, then the result is negative (i.e. the model will predict that the request is not part of a tunnel). If the output is between (0, +∞), then the results are positive (i.e. the model will predict that the request is part of a tunnel). If the output from the boundary equation is 0, then the model will not be able to predict if the request is part of a tunnel or not.

Figure 1.7 - Binary Logistic Regression



The odds ratio (OR) is associated with the output from the boundary equation and is used to map that output to the probability that the request is part of a tunnel. If P(X) is the probability of the request being tunnel related, then the odds ratio is calculated as follows:

$$OR(X) = \frac{P(X)}{1 - P(X)}$$

It is the ratio of the probability of a request being part of a tunnel, versus the probability of it not being part of a tunnel. From this, we get the following equation for the probability:

$$P(X) = \frac{OR(X)}{1 + OR(X)}$$

The relationship between the output from the boundary equation and the OR is defined as follows:

$$Log_e(OR(X)) = \text{ output}$$

so therefore

$$OR(X) = e^{output}$$

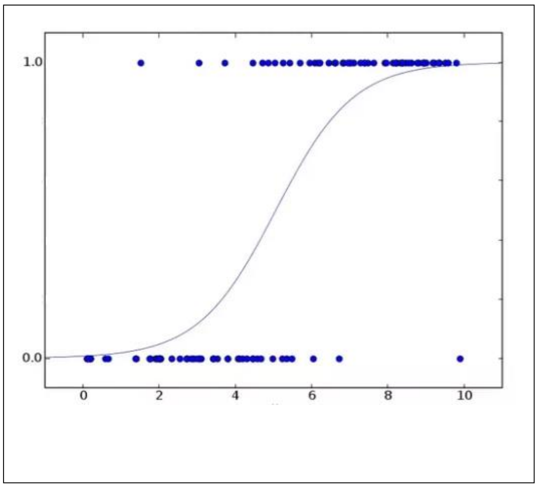
where e is the base of natural logarithms (approximately 2.7182818). This gives the following equation for the probability:

$$P(X) = \frac{1}{1 + e^{-output}}$$

The output from the boundary equation above will result in values between (-∞, +∞), the odds ratio will have values that are in the range (0, +∞), and the probability will have values in the range (0, 1). For this study, if a request has a probability above 0.5, then the prediction will be that it is part of a DNS tunnel.

A binary logistic regression curve can give a visual representation of the probability of a request being part of a tunnel. This can be useful in determining which predictive variables are likely to have the most impact on the model, and shows the cut-off point at which a request is more likely to be tunnel related. The curve maps the probability function defined above and should produce an s-type curve if the variable if effective at categorising the data. This can be useful to see how well a predictive variable separates the binary outcome regions in a logistic regression model.

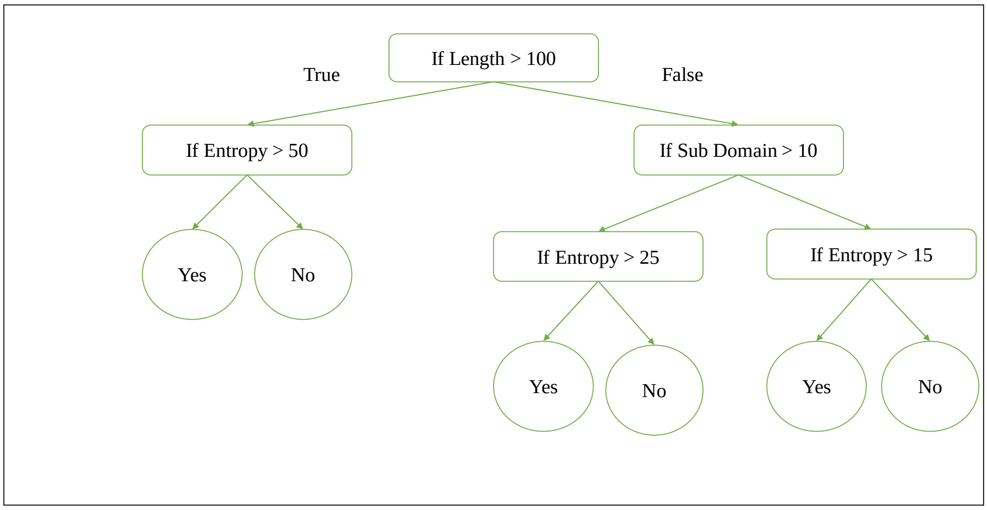
Figure 1.8 - DNS Curve



1.1.6.3 Decision Tree

A DT is another technique that can be used in binary classification to predict which category a set of predictive variables belong to. Each node on the tree represents a test on a characteristic (univariate), or a test on a set of multiple characteristics (multivariate), and each leaf represents a predicted outcome – tunnel Y/N. For example, in Figure 1.9 – Decision Tree Illustration, if the Length of a DNS request is greater than 100 and the Entropy is greater than 50, then the model would predict that the request is part of a tunnel. Similarly, if the Length is less than 100, the Sub Domain greater than 10, and the Entropy less than 15, then the model would predict that the request is legitimate.

Figure 1.9 - Decision Tree Illustration



The DT algorithm operates in two phases, the first is to build an initial tree, and the second to prune the tree (Garofalakis, Hyun, Rastogi, & Shim, 2000). During the build phase, the tree is grown until each node contains only a small number of child nodes. During the pruning stage, nodes that provide only redundant information are removed. This is done to improve the overall efficiency of the model and still maintain the accuracy of the predictions. Unlike LR, the output from a DT is a prediction rather than a probability, but comparisons can be made using the actual and predicted values to determine which model is most effective.

1.1.6.4 Random Forest

Overfitting can occur in a DT if too many variables are used with not enough sample data. For example, if ten characteristics are used to examine DNS tunnels with only one hundred samples, then there is a risk that the algorithm will over fit the training data to produce the model and make the results unreliable. If too few characteristics are used, then there may not be enough variance in them to produce an accurate model than can be used outside of the training set.

RF is one technique that can be used to get around this problem. The process uses multiple decision trees with different characteristics and training data, and then aggregates the results to produce an overall prediction. This can have the effect of reducing errors due to overfitting, but also maintain the accuracy by including all variance in the data. As with DT, the outcome from RF is a prediction rather than a probability.

1.1.6.5 Support Vector Machine

SVM can be used as a classification model based on a set of predictive variables. It is a machine learning algorithm that tries to find a hyperplane that divides data into two distinct categories. For example, Figure 1.10 SVM with Hyperplane (using example requests), illustrates how a scatter plot of length and entropy can be divided into legitimate and tunnel related requests. The green line represents the hyperplane and the points closest to it are the support vectors. The support vectors have the biggest impact on the hyperplane in terms of its slope and boundary. This is referred to as a linear SVM, as a straight line can be drawn between the two different groups. If no clear distinction between groups can be made, then it may not be possible to draw a linear hyperplane, as illustrated in the Figure 1.11 - SVM with no Linear Hyperplane.

In this case, non-linear transformations are performed to add extra dimensions to the model using the predictive variables as inputs - this is referred to as *Kernelling*. Gaussian Kernel, Polynomial, and Radial Basis Function (RBF) are typical examples of the technique (Ben-Hur & Weston, 2010) (Mongillo, 2011). Once the kernel is applied, the hyperplane is no longer a line but a plane - see Figure 1.12 - SVM with Kernelling.

Figure 1.10 - SVM with Hyperplane

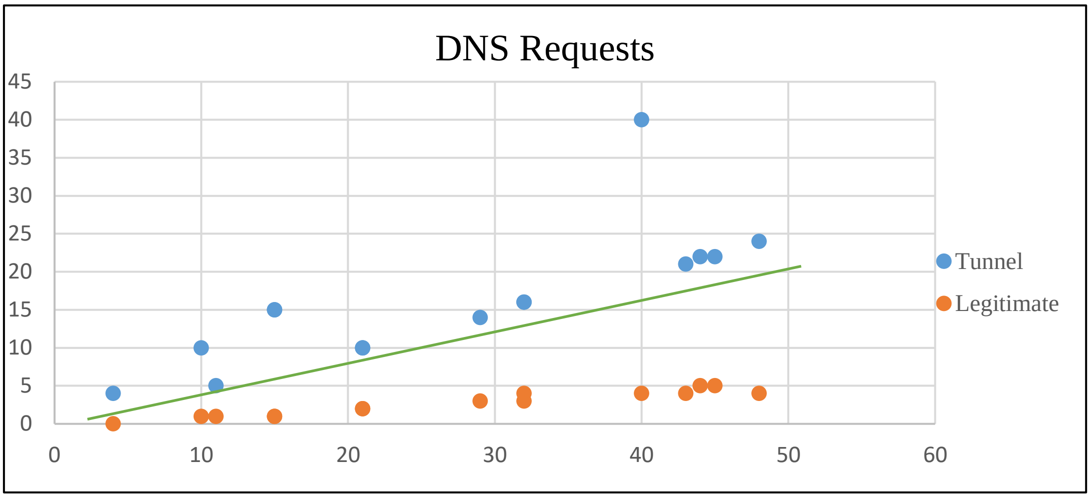


Figure 1.11 - SVM with no Linear Hyperplane

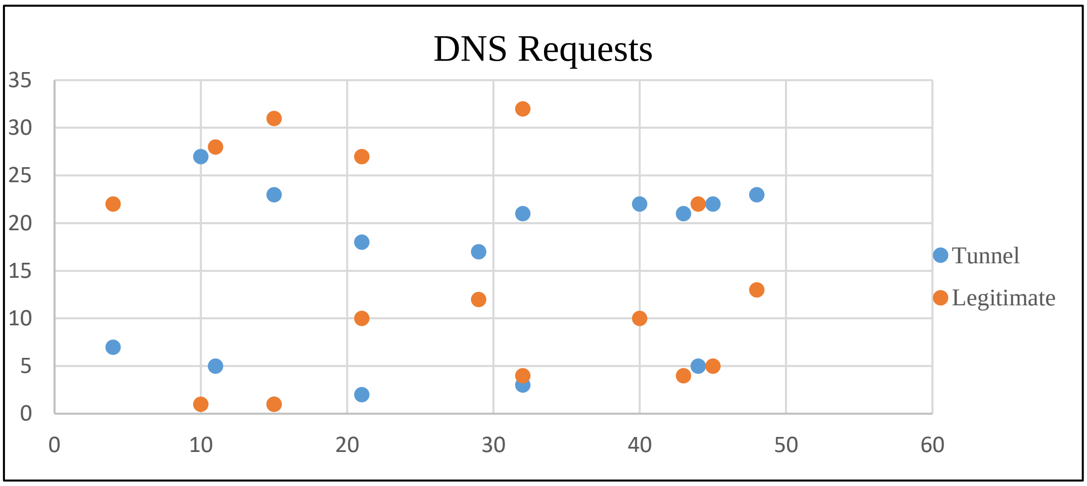
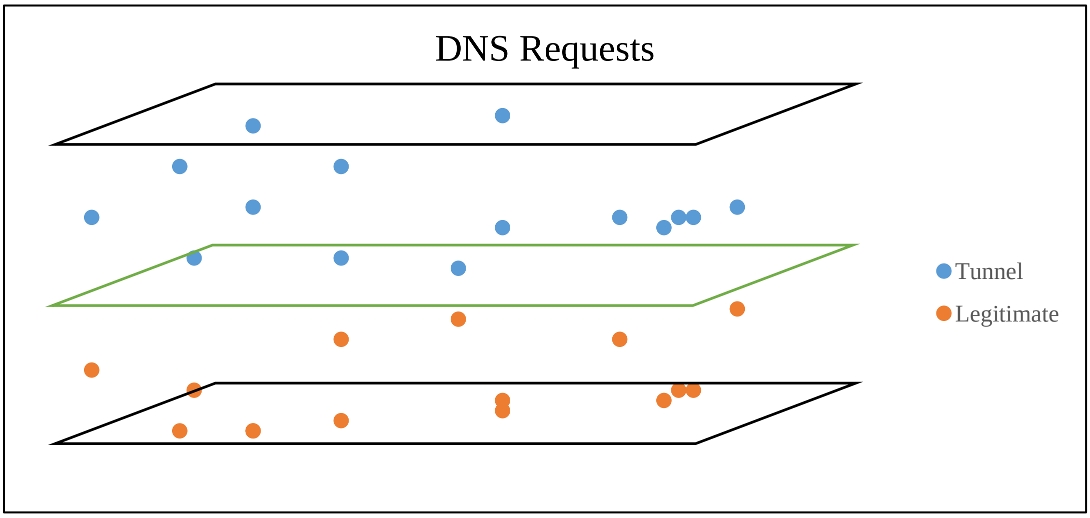


Figure 1.12 - SVM with Kernelling



Similar to DT and RF, SVM models place the observation in one of the two groups – tunnel or legitimate request. Comparisons between the different techniques will be made using the accuracy of the results obtained.

1.2 Research Objectives

1.2.1 Hypothesis

H_0 – There is no statistically significant difference ($p<0.01$) in the accuracy of the prediction of DNS tunnels using the classification techniques Logistic Regression, Decision Tree, Random Forest, and Support Vector Machine.

H_a – There is a statistically significant difference ($p<0.01$) in the accuracy of the prediction of DNS tunnels using the classification techniques Logistic Regression, Decision Tree, Random Forest, and Support Vector Machine.

1.2.2 Comparison of Classification Techniques

The objective of the research is to create classification models to make an effective prediction that a request is part of a DNS tunnel. Multiple classification models will be built and compared to each other to see which is the most effective at predicting tunnels. This comparison will be made by examining the True Positive and True Negative (accuracy) of each of the modes. The significance of the differences between the different models will be examined using a Cochran's Q test. The research will use publically available datasets of network traffic, with individual requests previously categorised as being legitimate or part of tunnel.

1.3 Research Methodologies

1.3.1 Data Gathering and Preparation

The research being undertaken is secondary in nature using existing datasets of DNS requests. The sources for the data are the GitHub PCAP Samples, Wireshark Sample Captures, Penetration Tester Lab, and TCP Replay samples, which contain PCAP files of DNS requests for legitimate and tunnel requests. The tunnel related traffic was generated using the *Iodine* and *Powercat* tools.

1.3.2 Validation of Inclusion of Characteristics in Model

The first phase in the solution will be to generate descriptive statistics for each of the candidate predictive variables to determine if they follow a normal distribution. The relationship between the predictive and the outcome variable will then be examined using an Independent t-test for parametric predictors, or Mann Whitney U test for non-parametric predictors. If statistically significant differences are not found for the predictor variable and the different outcome groups, then it will not be included in the model.

The second phase will look at the assumptions that need to be satisfied for classification models to produce reliable results. This will include looking at collinearity between the predictive variables using cross-tab reports. If a strong collinearity is found (i.e. one of the candidate predictive variables is closely linked with another), then one of the predictive variables will need to be removed or replaced. This can occur if two of the predictive variables are measuring the same variance in the data. If they were both included, then this variance would have an undue influence over the model.

1.3.3 Building Classification Model

The third phase of the solution will be to build classification models using each of the different techniques to predict if a request is part of a DNS tunnel. The first step will be to produce simple models with just one predictive variable as an input. The other predictive variables will then be added to the models to see if they can improve the results.

The different classification techniques being used are Logistic Regression, Decisions Tree, Random Forest, and Support Vector Machine. Statistics will be generated on the models to show how well they predict which category the request belongs to, DNS tunnel or legitimate request, and then comparisons will be made to see which technique produces the best results.

1.3.4 Evaluation of Classification Models

The models will be evaluated in terms of the Sensitivity, Specificity, Accuracy and Precision of the results obtained as calculated using the True Positive, True Negative, False Positive, and False Negative values.

True Positive (TP): The total number of DNS requests correctly categorised as being tunnel related.

True Negative (TN): The total number of DNS requests correctly categorised as not being tunnel related.

False Positive (FP): The total number of DNS requests incorrectly categorised as being tunnel related.

False Negative (FN): The total number of DNS requests incorrectly categorised as not being tunnel related.

Sensitivity: This tells us the percentage of tunnel related DNS requests that were correctly classified by the model and is calculated as TP / (TP + FN).

Specificity: This tells us the percentage of non-tunnel related DNS requests that were correctly classified by the model and is calculated as TN / (TN + FP).

Accuracy: This tells us the percentage of DNS request that were correctly classified as being tunnel or legitimate and is calculated as (TP + TN) / (TP + FP + FN + TN).

Precision: This tells us the percentage of predicted DNS tunnel request that were correct and is calculated as TP / (TP + FP).

Using these measures will allow the models to be evaluated during each step in the process (base model followed by additional predictors) and comparisons to be made between the techniques. To determine if the differences in accuracy are significant, a Cochran's Q test will be performed on the TP and TN results. The statistic for the test is defined as follows:

$$q = (k-1) * \frac{(k * \sum_{i=1}^{k} m_i^2) - (\sum_{i=1}^{k} m_i)^2}{k * \sum_{i=1}^{k} r_i - \sum_{i=1}^{k} r_i^2)}$$

where k is the number of models, m_i is the number of correctly predicted requests per model, and r_i is the number models that correctly predicted each request.

Table 1.2 - Illustration of Cochran's Q Test

Model / Request	**Model 1**	**Model 2**	**Model 3**	**Model 4**	**Total**
Request 1	0	0	1	1	2
Request 2	0	1	1	0	2
Request 3	1	1	0	1	3
Total	1	2	2	2	8

In this case q = 0.8181

$$q = (3) * \frac{(4 * (1^2 + 2^2 + 2^2 + 2^2) - (1 + 2 + 2 + 2)^2}{4 * (2 + 2 + 3) - (2^2 + 2^2 + 3^2)}$$

The statistic has a χ^2 (chi-squared) distribution. In this instance the p-value is approximately 0.85, which would mean we would fail to reject the null hypothesis that the predictions are the same.

1.4 Scope and Limitations

The scope of the study is to look at four classification techniques, Logistic Regression, Decision Tree, Random Forest, and Support Vector Machine, to see if there are statistically significant differences in their ability to predict DNS tunnel requests. Implementing the models in a production environment, or documenting the creation of DNS tunnels, is beyond the scope of this research.

It can be difficult to find datasets of DNS requests with known tunnels without generating them yourself. If you generate them yourself, then you have complete control over the nature of the request which could lead to invalid results as there would be a risk that you could control for the characteristics under investigation.

For this study, datasets available online with known DNS tunnels will be used. The volumes of data available are small (2221 legitimate requests and 3425 tunnel requests), and so will need to be taken into account when interpreting the results. Also, as the requests were produced in a test environment, the source IP address and responses were constant for tunnel and legitimate traffic, so these characteristics will not be used as predictive variables in the models. The tunnel requests were generated from the DNS tunnelling tools *Iodine* and *Powercat*, and so may not have the variety required to produce a general purpose model that could be reliably used to help identify DNS tunnels in a real production environment.

1.5 Document Outline

- Chapter 2 – Literature review and related work
  The literature review chapter will briefly review the history of DNS and the steps required to create a tunnel using the protocol. The various techniques used to detect tunnels in previous studies is also examined – in particular anomaly based systems and the detection of auto-generated domains. The remaining section will review the potential gap in the existing research around the prediction of an individual request being part of a DNS tunnel and the evaluation of classification techniques.

- Chapter 3 – Design and methodology
  The design and methodology used in the research followed the CRISP-DM (cross-industry process for data mining) process to help ensure reliable results. The steps followed were business understanding, data understanding, data preparation, modelling, evaluation, and deployment. The steps required to obtain & prepare the data is explained, along with process required to select the predictive variables. The different tuning methods used to improve the models are reviewed along with an explanation of how the effectiveness of the models will be measured.

- Chapter 4 – Implementation and results
  This chapter will present the results from the research that includes summary statistics on the data and visual representations of how the data is distributed. Decision around which characteristics to include as predictive variables is also

outlined. For each of the models produced with the different classification techniques, the results obtained at each stage are reviewed.

- Chapter 5 – Analysis, evaluation and discussion
  The analysis chapter will summarise the results obtained in the study and put some context on the reliability of the models to be used as general predictors of DNS tunnels. It will also review the strengths and weaknesses of the models and highlight any areas of concern with the results.

- Chapter 6 – Conclusion
  This chapter will outline the process to complete the study, present a summary of the results obtained, and determine which binary classification techniques are most suitable for categorising DNS traffic as being legitimate or tunnel related. It will also comment on the impact of the research and how it could be extended to be used in a production environment.

- Chapter 7 – Bibliography
  This chapter details all the previous research referenced in this study.

- Chapter 8 – Appendix
  This chapter contains of all the results obtained during the evaluation of the different classification techniques. This includes results obtained when using one, two, and three predictive variables.

2 LITERATURE REVIEW AND RELATED WORK

2.1 Introduction

This chapter will give a brief history of the domain name system, how it can be exploited as a covert channel using a tunnel, the different detection techniques available to identify them using anomaly based detection and identification of auto generated domains, and the gaps in research in relation to the comparison of different classification techniques and their ability to classify an individual request as being legitimate or tunnel related based on the characteristics.

2.2 Domain Name System

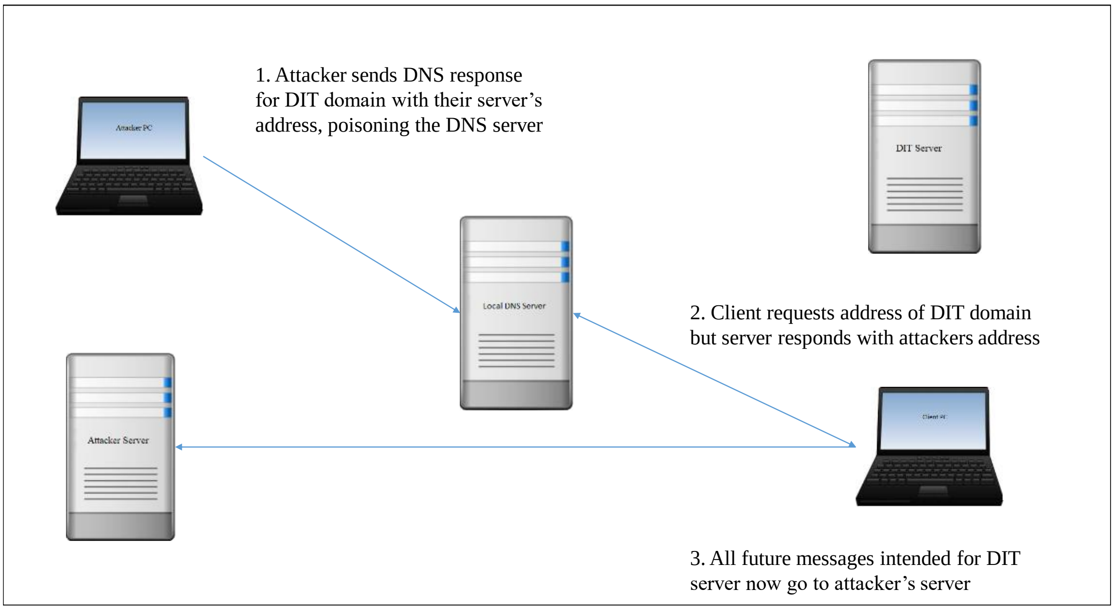
The DNS protocol is a hierarchical, distributed database used to translate domain names that are human readable into IP addresses that are then used to transmit messages between hosts on a network. Most security systems concentrate on preventing intruders gaining unauthorised access to systems and data on the organisations network. However, if malware gets installed onto a machine connected to the network, then it can be used to run commands on the network, spoof IP address, disrupt services, and extract information.

DNS can be the focus of many different cyber-attacks due to the perceived lack of security in the protocol. Wright (2012), in his paper on *DNS in Computer Forensics*, concluded that there were not enough security professionals who truly understand DNS and how vulnerable it can be to being exploited. He demonstrated many of the different vulnerabilities in DNS and suggested some techniques that could be used in a forensic investigation, including DNS logging and looking at the history and behaviour of clients on a network. However, logging DNS requests for even a moderate amount of time may be impractical as it would generate a large amount of data and could slow down the DNS service.

There are many different type of attacks that can be based on the security vulnerabilities in DNS. A denial of service attack (DoS) can be achieved by sending a large number of DNS requests that overwhelm a DNS server. Hudaib and Hudaib (2014) suggest that increasing the number of DNS servers can help in a DNS related DoS attack, but if the attacker employs DNS spoofing, then they can increase the volume of DNS requests and exhaust the servers. In this scenario, the attacker sends out a small request, but the spoofed target address receives a much larger DNS response. This type of attack can be blocked if the servers are capable of recognising a spoofed IP address.

Another example of an attacker vector involving DNS is cache poisoning. When a DNS server receives a request for a domain, it will search its own cache to find a response. If it doesn't find one, it can either forward the request to an authoritative server for the domain, or alternatively, it can respond to the client directing them to the authoritative server. If an attacker forges a response to the DNS server, which looks like it comes from the authoritative server, and this is accepted by the DNS server, then this would result in the cache for the DNS server being poisoned (Son & Shmatikov, 2010). Any future requests for the domain sent to the DNS server would result in the address defined by the attacker being returned to the client. This could result in HTTP or FTP messages intended for a legitimate target ending up on the attacker's machine. The attacker could then search these requests to find sensitive information to conduct further attacks against the client or third parties.

Figure 2.1 - DNS Poisoning



The motivation behind these attacks are not solely financial, they can also be political, religious, state sponsored, industrial espionage, or just individuals trying to make a name for themselves. Terrorism is another motivating factor that is sometimes considered a sub-class of religious or politically based attacks. Atkins (2013) in his thesis on *Cyber Espionage*, looked at the impact DNS vulnerabilities and internet protocols in general has had on terrorist activities. He concluded that terrorist groups have adapted to the internet to take advantage of some of the security weaknesses in the protocols used. He also looked at how technology could play a vital role in tracking the movements of terrorists, and highlighted the risks to privacy associated with such surveillance.

DNS can also be used as a command-and-control (C&C) channel for botnets allowing attackers to send updates to bots or launch attacks against networks or servers. The service is an ideal protocol for attackers to use as the high volume of DNS traffic on a network can help to hide the C&C requests, and the lack of security around DNS gives an attacker a potentially un-monitored median to send instructions to bots. Xu, Butler, Saha, and Yao (2013), present a C&C strategy that uses DNS tunnels that match existing traffic patterns to help avoid detection. They concluded that manipulating the DNS protocol was a very effective means to covertly create a C&C channel to control bots to conduct illegal activities.

The fundamental nature of DNS, coupled with the lack of security and monitoring, has resulted in DNS becoming a target for individuals who use it unlawfully in different attacks and with different goals. Those goals can sometimes include transmitting messages outside of an organisation’s network via a DNS tunnel.

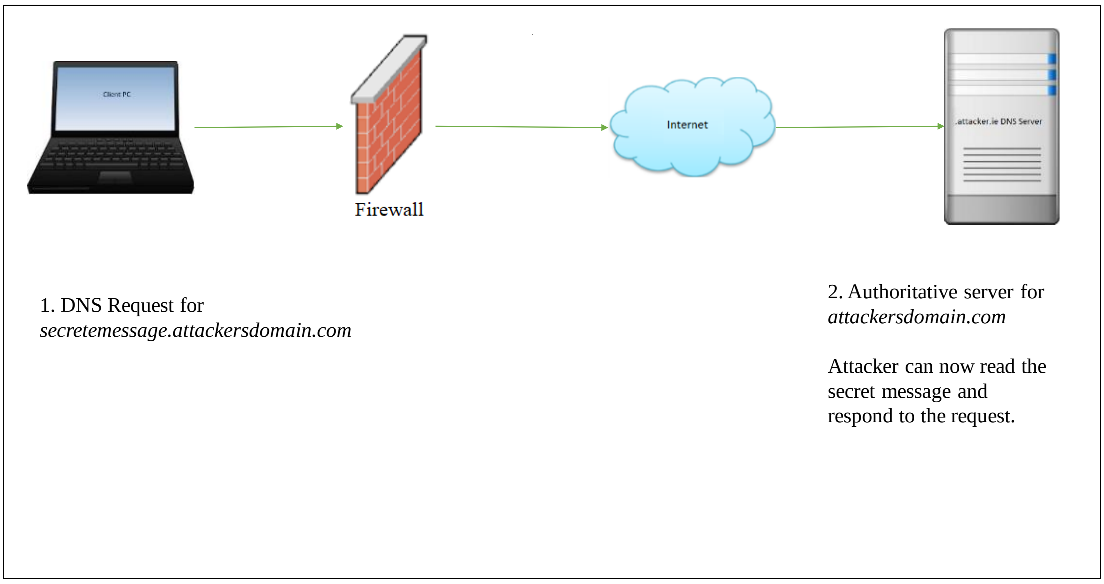
2.3 DNS Tunnels

2.3.1 Creating a DNS Tunnel

A DNS tunnel is the process by which the protocol is used to covertly communicate messages between computers. The motivation to create DNS tunnels can be to avoid internet paywalls, exfiltrate data, act as a C&C channel, or to get around censorship rules. Wolfgarten (2006), in his thesis on *Investigating large-scale Internet content filtering*, reviewed how DNS tunnels could be used to circumvent censorship laws as used in the People’s Republic of China. His conclusions at the time were that the process to create tunnels was too complicated for the average user and that alternatives were needed to avoid censorship.

Since then, tools such as *Iodine* and *Powercat* have become available to make the process of creating a tunnel more straightforward not only to avoid censorship laws, but also for users trying to inflict harm on individuals or organisations. This reduction in technical expertise required to create a DNS tunnel has made them much more accessible to the average user. Reyes (2014), in his paper on *Covert channel detection using flow-data*, describes the process of creating a DNS tunnel by embedding a message in the sub domain of a DNS request. If an attacker controls the authoritative server for a domain, then they will receive the DNS request and be able to extract the message. For example, if a host on a network sent a DNS request for the *secretmessage.attackersdomain.com*, the request would normally be allowed through the firewall as DNS requests are rarely blocked, and eventually be forwarded to the authoritative server for *attackersdomain.com.* This server is controlled by the attacker who could then extract the message from the request. The messages would normally be encrypted which can make it more difficult to detect that they are transmitting sensitive information.

Figure 2.2 - DNS Tunnel



Data exfiltration via DNS tunnels is generally implemented in this way by installing malware on the client PC that collects information from the local network and then transmits it to the server controlled by the attacker via a DNS request (Raman et al., 2013). Once the message reaches that DNS server, the attacker can read the message and respond with a fake IP that directs the malware on the client to take a specific action, or it can simply send a null response.

Llamas, Allison, and Miller (2005), in their overview of *Covert Channels in Internet Protocols*, described how DNS requests can be an attractive attack vector for anyone wishing to hide or disguise communication. The continued evolution of hidden and anonymous communication channels has made it increasingly difficult for security teams to detect that a covert channel exists. This is particularly true if the attacker is patient and has the time to adopt a Low and Slow approach to the communication. Middelesch (2015), in his thesis on *Anonymous and Hidden Communication Channels*, looked at the continued evolution of covert channels and discussed the use of DNS as a covert means of communication. He concluded that there were many different options for hidden communication but that the *Tor* browser was probably the best.

Hoffman, Johnson, Yuan, and Lutz (2012) describe a different technique to exploit the DNS protocol, by manipulating time-to-live (TTL) on DNS request, and to use that as a means to communicate covertly. Although the throughput on this type of tunnel would be relatively small, over time it could still be very damaging financially or to the reputation of an organisation.

As previously mentioned, the creation of DNS tunnels is now more straight forward than ever. As a consequence they have become much more prevalent, and therefore it is more important for organisations to be able to detect DNS tunnels on their network to prevent data leakage.

2.3.2 Detecting DNS Tunnel

The detection of DNS tunnels has become a focus for organisations and individuals trying to prevent the exfiltration of data from their network. Banking details are no longer the only target for criminal organisations trying to extract information from personal or corporate networks. Client data, personal data, intellectual property, or company secrets, are now all targets for criminals, which can cause both financial and reputational damage to organisations or individuals.

Giani, Berk, and Cybenko (2006), in their paper on *Data Exfiltration and Covert Channels*, discussed some of the different techniques available to attackers to extract information from an organisations network. They reviewed how organisations can sometimes find it difficult to know what information is leaving the network legitimately and what information is being leaked to unauthorised individuals or organisations. They also looked at how sometimes it can be difficult to know if communication is occurring at all.

As network security analysts develop different techniques to identify malware, attackers adapt their strategy to avoid being detected. For example, the *Conflicker* worm received updates by creating 250 algorithmically generated domains (AGD) every three hours. They used the current date and time as a seed, ensuring that all the bots would generate the same names every day. When the worm was reverse engineered, and the domains were identified and blocked, the later version of the malware switched to use any 500 of 50K different AGD. Identification of malware generated AGD is further compounded by the use of AGD for legitimate purposes by benign applications such as Internet browsers (Krishnan, Taylor, Monrose, & McHugh, 2013).

2.4 Network Anomaly Based Detection

Adapting a Low and Slow approach to data extraction can be an effective technique for attackers to use to avoid detection of their DNS tunnels. Most Intrusion Detection Systems (IDS) will be able to identify high throughput tunnels by the rapid increase in DNS traffic relative to other network traffic protocols, but if the attacker has enough patience and time, and adopts a Low and Slow approach, then they can go undetected by an IDS.

One of the main approaches to identify covert DNS channels adopting a Low and Slow approach is by using a network anomaly based IDS. For example, Reyes (2014), suggests taking a base line for different protocols (including DNS) to establish normal network traffic patterns, and using this as a comparison against future network traffic to identify suspicious activity. The analysis concluded that the ratio between the number of requests sent and received, and the ratio between the size of the message sent and received, were key differentiators between normal and legitimate DNS traffic flow, and that a DNS tunnel could be detected using this technique. The research looked at historic flow-data to identify the DNS requests and establish the normal behaviour thresholds.

Nadler, Aminov, & Shabtai (2017), also looked at ways to detect low throughput DNS tunnels using entropy, request type, uniqueness of requests, time-to-live, size of request, length of sub-domain, and longest word length, as predictive measures for their models. Principle component analysis (PCA) was used to reduce these dimensions to a set of uncorrelated values used in the model.

The approach was to use an SVM with an RBF kernel to build a one-class classifier, trained with legitimate DNS requests, to define what the normal behaviour was for the network. An anomaly based rather than classification based technique was used due to the limited availability of known DNS tunnel datasets. Nadler et al. (2017) viewed this lack of availability as having the potential to restrict the ability of a classification model to detect different patterns of DNS tunnel requests, and so developed an anomaly based system instead. For their research, the authors determined that any requests outside of the normal behaviour was to be classified as suspicious. As part of the process, they ignored DNS requests to what was deemed to be reliable services, in order to reduce the number of false positives. Using this technique, they achieved a success rate of over 80% for low throughput tunnels, with less than 1% false positives.

Systems using this approach usually include a training phase when the system observes what is deemed to be normal behaviour for comparison during the operational phase. Satam, Alipour, Al-Nashif, and Hariri (2015), developed an DNS-IDS system that was trained with normal DNS traffic and then used this information to detect anomalies during the operational phase. The process followed was to build a DNS finite state machine with legitimate DNS traffic, and then classify future DNS flows as abnormal if they didn’t follow the same pattern. They concluded that their model, using supervised machine learning, was capable of detecting malicious DNS activity, for both known and unknown attack vectors, with a 97% success rate and a false positive rate at around 0.01%.

Machine learning techniques can be applied to identify DNS tunnels with different goals – e.g. tunnels created for data exfiltration and ones created to avoiding paying for internet services. In his thesis on DNS tunnels, Skow (2016) used the statistical differences between the characteristics of tunnels and normal DNS traffic to identify suspicious requests. In this instance, the author was interested in identifying tunnels that were being used to avoid paying for services on mobile networks. Similar to Nadler et al., they used a one-class SVM with a number of different kernels (including RBF) to identify normal patterns of request. A K-Means classification model was also created to categorise the requests as either legitimate or being part of a tunnel. The data used in the research was generated by the author using a DNS tunnel application, with the requests captured in *Wireshark*. He concluded that the SVM model outperformed the K-Means classifier in its ability to detect DNS tunnels. The RBF kernel was recommended for use in a real network. As with Reyes, Skow identified the size of the DNS request and response messages as the key characteristics most likely to identify a tunnel.

During the operation phase of an anomaly based IDS, statistical analysis is conducted on the data to determine if the traffic is within the limits of normal behaviour. Sarmah, Dey, and Kumar (2014) used a kernel density estimation (KDE) model to smooth out the data, and Pearson's correlation coefficient, to help identify differences between normal and tunnel related DNS requests. In the learning phase, for each IP address on the network, they measured the number of bytes sent and received in each 15 second interval for 5 days. They then calculated the probability density function using a KDE model and compared the results in the operational phase using a Pearson correlation. They concluded that combining KDE with Pearson's correlation coefficient improves the success rate in identifying abnormal behaviour which may indicate data exfiltration on a network.

Using the relationship between network events (for example HTTP & DNS traffic) is another approach learning systems can use as a baseline for detecting anomalies. This was illustrated by Zhang, Yao, and Ramakrishnan (2014), in their paper on detecting malware that operates in stealth mode to avoid being picked up by an IDS. In their paper, the authors estimated that over 25% of computers are infected with malware, and that if it was operating in a stealth mode, then frequency analysis based IDS may fail to detect it. Instead, they looked at the relationship between different types of traffic as a means to identify malicious requests. The first step in the process was to collect the attributes associated with different network events and then try and pair events that have a potential causational relationship. The next step involved significant manual effort to label the data into different categories. The authors then trained different classification models, Naïve Bayes, Bayesian Network, and SVM, and then compared the results using precision and sensitivity (recall). Their conclusion was that relationships between events can be an effective means to identify abnormal behaviour on a network. SVM appeared to give the better results of the three models used.

Another strategy suggested by Paxson et al. (2013) in their paper on *Surreptitious Communication Over DNS*, is to limit the amount of data that can be extracted to an attackers domain by measuring the potential data leakage on all requests to a domain over a specified period. When the potential data leakage through DNS requests to a specific domain reaches an upper bound limit, then an alert is created for the security analyst to determine if the requests are genuine or not. The trade off with the approach is the risk of potential data leakage versus the number of false alerts generated for the analysts to look at. The higher the upper bound limit on DNS requests to a domain, the higher the potential for data leakage, but there is also a reduced number of alerts that need to be investigated. With only two parameters to tune, the model is relatively simple to implement, can be an effective way to reduce any potential data loss, and can also highlight specific domain requests that may need further investigation to determine if they are legitimate or not.

Research by Born & Gustafson suggest that examining the character frequency of DNS requests can help distinguish between legitimate requests and those that are being used in a DNS tunnel (2010). They examined strings of length one, two, and three, and compared the character frequency distribution of legitimate domains to randomly generated domains. The authors maintained that tunnel traffic, if properly encrypted, would appear as if randomly generated. The legitimate domains followed a distribution similar to natural languages, while the randomly generated domains where more evenly distributed. As the number of domains increased, the legitimate domains moved closer to natural language distribution, and the randomly generated domains moved further away. The authors also note that entropy and character frequency are closely related, as entropy is directly impacted by the character frequency distribution of a language. Using this method, the authors were able to create a fingerprint to identify anomalies in DNS traffic.

Ellens et al. setup a tunnel on a network using *Iodine* to examine ways of identifying DNS tunnels (2013). They looked at the use of DNS tunnels for data exfiltration, command & control, and web browsing, to give them a variety of attack vectors to examine. They primarily measured the number of bytes in DNS requests and the frequency of request over different time periods throughout the day. For legitimate DNS requests, they could identify distinct patterns between day time and night time traffic, for tunnel traffic, the patterns were not so obvious. They proposed three different techniques to identify abnormal DNS traffic. The first was by setting a threshold (for the time of day) above which the DNS activity would be considered suspicious. The second was to calculate the average number of bytes for a time period and compare it to an earlier time period. Any differences would again indicate suspicious activity. If the threshold or average test fail to detect anything, then a third method using a *Kolmogorov-Smirnov* normality test was used to identify differences. Using these techniques, the authors demonstrated that the size and frequency of DNS requests were appropriate metrics to identify tunnels.

The advantage anomaly based systems have over signature based systems is that they are capable of detecting zero-day attacks. Signature based system cannot fulfil this role as they require a known pattern of behaviour in order to identify an attack. The disadvantage for anomaly based systems is that they require a training phase and careful setting of thresholds in order to be effective. Jyothsna, Prasad, and Prasad (2011), in their *Review of Anomaly based Intrusion Detection Systems*, looked at the issue of classifying normal traffic and setting rules to identify abnormal behaviour. They concluded that once anomaly based systems define the rules of normal behaviour that they can be effective at identifying malicious traffic, but if an attacker stays within the limits of what is deemed to be normal, then their activities can go un-detected.

2.5 Auto-Generated Domains

Auto-Generated Domains (AGD) are often used in DNS tunnels as part of the covert communication between servers. If an AGD could be identified in a DNS request, it could indicate that the request is part of a DNS tunnel. One approach is to reverse engineer AGD to see what domains it uses and blacklist them from any future DNS requests. Liu et al. (2017) in their paper on *Shadowed Domains*, reported that attackers had started to use shadowed domains of legitimate apex domains to avoid this type of detection. To detect shadowed domains used by attackers, the authors compared the host IP of the apex domain with the host IP of the shadowed domain. When used legitimately, the hosts tended to be close together, when used maliciously, the host could be far apart, with the shadowed domain's host having fewer security restrictions. Maliciously created sub-domains also tended to be created in bulk rather than incrementally, as is the case with legitimate shadowed domains. The authors also observed similarities in the maliciously created shadowed domains across different apex domains. A combination of these factors could then be used to detect the maliciously created shadowed domains and block them from transmitting information outside the network.

Yadav, Reddy, Reddy, and Ranjan (2010), looked at the command & control of botnets using AGD as used in the *Conflicker*, *Kraken* and *Torpig* botnets. They considered the reverse engineering of malware to blacklist the set of domains was of limited use, as the malware adapted to use an increased number of potential domains. However, they observed that the sub-domains of legitimate DNS traffic exhibited a more irregular distribution of characters than those created with an AGD, and that legitimate domains were more likely to be made up of proper words. If an AGD used proper words, this would increase the chances of the domain already being registered which could restrict attacker's ability to control the bots. This allowed the authors to successfully identify an AGD using different techniques used to measure the character distribution. The most successful technique they found was to use the *Jaccard* index. This measures similarities between two sets of data, and was used to compare the set of *bigrams* (character pairs) from legitimate and malicious traffic. The *Jaccard* distance to measure the similarities between sets is related to the Entropy of a language used to determine the minimum number of characters required to encode a string (Parker, Yancey, & Yancey, 2016). Even if the AGD was modified to generate domains that more closely resembled words, the model still had a high success rate in identifying generated names.

Antonakakis et al. (2012) presented an alternative approach to detecting AGD by checking for null responses from the DNS requests. Like Yadav et al., they looked at ways to identify AGD without having to reverse engineer the malware to blacklist the potential domains. The difference with their paper was the premise that most domain generation algorithms (DGA) respond to the DNS request with a NXDOMAIN – domain does not exist message. They were able to classify these requests into different groups based on the similarity of the domain names and the source of the requests. From here they were able to identify previously known botnets in operation, as well as identifying new botnets. However, this technique may be vulnerable to attackers who return spoofed IP addresses to the client in order to avoid detection.

Butler, Xu, & Yao specifically looked at DNS tunnels being used as command & control for botnets (2011). They noted that DNS was a particularly useful means to conduct covert communications due to the high volume of traffic associated with the service and the general lack of security provisions in place to protect against abuse of the channel. They gave an overview of different techniques used to setup a botnet to communicate over DNS, and then described the different strategies to avoid detection by an anomaly based IDS. This included controlling the frequency of DNS requests to follow existing patterns, and piggybacking on the timing of legitimate requests by only sending covert messages when legitimate traffic has been detected. Based on the fact that legitimate domains tend to have some meaning (i.e. made up of real words), and auto generated domains being used for tunnelling did not, they were able to show that there was a difference with the entropy associated with each group. The authors observed that although this could allow suspicious DNS activity to be identified, it was unclear how effective it would be if an algorithm was created that could generate domains that had similar properties to natural language words.

2.6 Gaps in Research

The above techniques demonstrate the problems related to DNS tunnels and the options associated with using statistical analysis to identify them. The various strategies that can be used to improve the detection rate are also described along with the different properties that can be used as indicators of a tunnel.

The gap in the research relates to the identification of which classification technique is the most effective at identifying DNS tunnels using only the characteristics of the individual request as predictive variables. If an individual request could be identified in this manner, it could prevent tunnel related DNS requests ever leaving the network.

2.7 Summary

DNS is a lookup service used in the routing of messages between servers on the internet that converts domain names into an IP address. It is one of the fundamental protocols used in computer networks and is not usually monitored for security breaches to the same extent as other protocols such as HTTP or FTP. As a result, the DNS protocol has become a target to be used as a covert means of communication. In the past, the creation of a DNS tunnel would require a level of technical expertise not found in the average user, but now with the availability of tools such as *Iodine* and *Powercat*, the creation of DNS tunnels has become a lot easier.

The detection of DNS tunnels has become increasingly important for individuals and organisations as they try to prevent data from being exfiltrated from their network. It is not possible to simply switch off the protocol as this would impact on legitimate activities as well as illegal ones. One of the primary mechanism used to identify DNS tunnels is anomaly based detection systems. In these systems, the strategy is for the system to learn what normal traffic looks like, and then alert if any anomalies are detected during the operational phase. Character distribution, Entropy and Size of Requests, are common characteristics used in these models. One-class classification models using different techniques, such as Support Vector Machine (SVM), can be used to determine that abnormal behaviour is occurring on the network.

Another common approach to detecting DNS tunnels is to look at the domain to determine if it was algorithmically generated. Genuine domains tend to have some meaning and are made up of real words, whereas generated ones are more likely to be appear random. Examining *Jaccard* index or character frequency distribution can help distinguish between a legitimate domain and an AGD. Both of these attributes are closely related to the Entropy value.

The goal of this study is to combine the characteristics defined in Table 1.1 – DNS request characteristics to develop binary classification models to determine if a DNS request is part of a tunnel, and then determine if different techniques give different results that are statistically significant. Classification models using different techniques, LR, DT, RF and SVM, will be built and comparison made between the results. Previous studies have demonstrated how the characteristics of DNS tunnel requests can differ from legitimate requests and how classification models can be applied to detect these differences. In this study, the models will be trained with categorised DNS traffic (tunnel or legitimate request), and then tested on individual DNS requests to determine how effective they are at identifying tunnels.

Table 2.1 - Summary of previous work contributing to this study

Author(s)	**Description**
Nadler et al.	Using one-class SVM-RBF classifiers with Entropy, Length and Length of Sub Domain as predictor variables
Skow	Using one-class SVM-RBF classifiers with Length as a predictor variable
Born, Gustafson	Using Character Frequency / Entropy as differentiators between legitimate and tunnel related traffic
Yadav et al.	Using Logistic Regression techniques with Character Distribution / *Jaccard* Index as predictive variables
Zhang et al.	Comparison of classification techniques using the relationship between DNS traffic and other protocols

3 DESIGN AND METHODOLOGY

3.1 Introduction

This chapter will review the steps required to carry out the study. The CRISP-DM (cross-industry process for data mining) methodology will be used to structure the research in order to help achieve a successful outcome and ensure that the results are reliable and conclusions easy to follow and understand.

The different tools used throughout the research are:

- Wireshark
- Excel
- Databricks Community Edition – Python

3.2 Business Understanding

The first step in the process is to understand the goals of the project as defined in the Research Question. In this instance, the goal is to see if statistically significant differences exist between classification techniques and their ability to categorise a request as being legitimate or tunnel related based on the characteristics of an individual request. The different techniques being examined are:

- Logistic Regression (LR)
- Random Forrest (RF)
- Decision Tree (DT)
- Support Vector Machine (SVM)

An overview of these techniques can be found in the Research Project section. Each model produced will be capable of making predictions on whether a request is part of a tunnel, but the research should provide an indication of which technique is most effective in terms of the Sensitivity, Specificity, Accuracy, and Precision of the results obtained. To see if any differences are statistically significant, a Cochran's Q test will be performed on the accuracy of the models produced.

3.3 Data Understanding

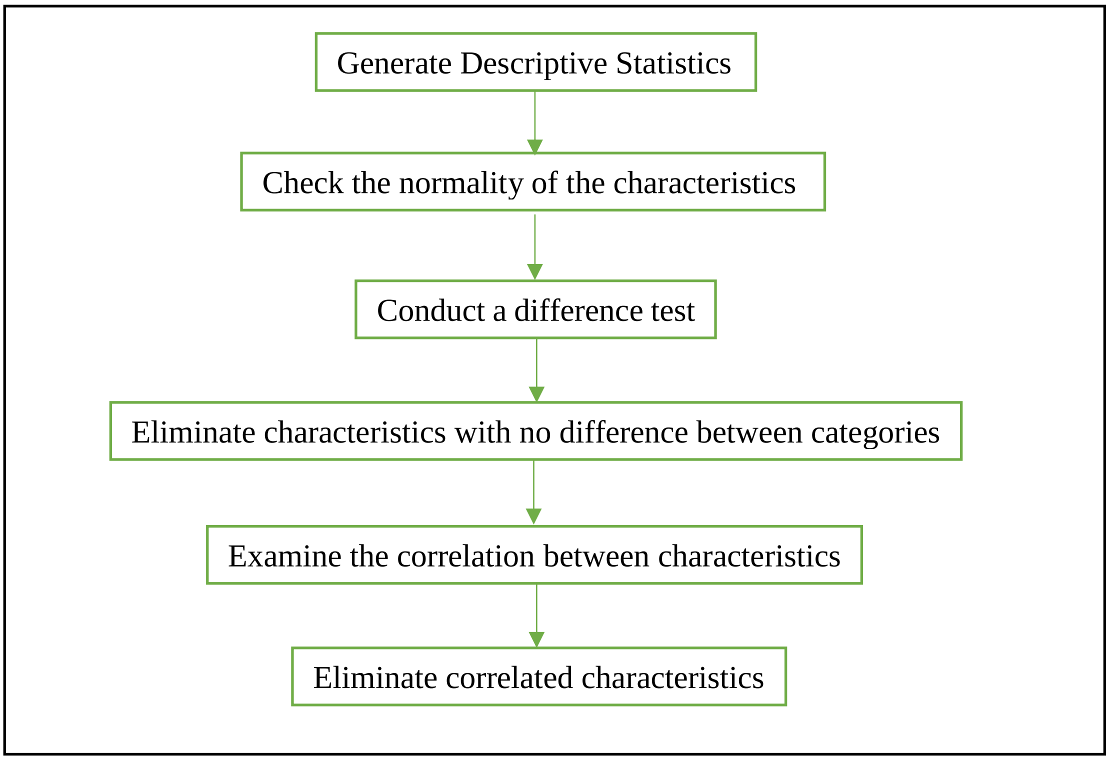
The candidate predictive variables to be used in building the different classification models are listed in Table 1.1 - DNS Request Characteristics. To justify their inclusion as predictive variables, a difference between legitimate and tunnel related requests should be demonstrated. To achieve this, descriptive statistics will be calculated for each characteristic highlighting the mean, standard deviation, median and inter quartile range for legitimate and tunnel related traffic. This should give an indication of any differences between the two different categories.

To determine if the differences are statistically significant, the normality of the distribution of the characteristics must first be tested. If the variables follow a normal distribution, then a parametric test for difference can be used (Independent t-test), if not, a non-parametric test will be required (Mann Witney U test). If no differences are found, then the characteristic should not be included as a predictive variable.

The last step in determining the predictive variables will be to look at the correlation between the characteristics. Multicollinearity between variables in a classification model can reduce the reliability of the results obtained and make the model very sensitive to small changes. To eliminate it, correlation between all of the request characteristics needs to be calculated. If a high correlation is found between characteristics, then only one should be included as a predictive variable in the models. This should not reduce the accuracy of the models as the variance associated with the eliminated variable should be included in the variable that remains.

For each of the selected predictive variables, a logistic regression curve will be created to help demonstrate the impact that each of them will have in determining that a request is part of tunnel. To achieve this, an individual LR model will be built and the boundary equation created for each of the variables. The probability function (using the output from the boundary equation) will then be graphed to demonstrate the difference in that variable between tunnel and legitimate traffic. The data will then be split into training (70%) and testing (30%) datasets. The training set will be used to fit the models and the testing set will be used in the predictions.

Figure 3.1 - Steps to Determine Predictive Variables



3.4 Data Preparation

This research is secondary in nature using publically available datasets of DNS requests already categorised as being legitimate or tunnel related. The sources of the PCAP files used during the research are listed in Table 3.1- PCAP files used in the analysis. Each of these PCAP files will need to be loaded in *Wireshark* and exported into CSV files. These files will then be loaded in *Excel* and merged into a single CSV file that will then be loaded into *Databricks*.

All the requests in the dataset were created in a test environment for analysis purposes, so there is a consistency to certain characteristics across the different categories. As such, these characteristics will not be used during the build of the classification models as they would yield unreliable results and not be representative of real traffic. Specifically, the DNS requests generated from *PowerCat* started with "*dnscat*", and all the requests generated from *Iodine* ended with "*pirate.sea*". These strings were removed from the DNS requests as this is unlikely to occur in a real DNS tunnel. Also, there is a consistency to the response messages for the tunnel traffic and so they will not be used during the analysis. The data from *Skype* and *TCPReplay* datasets had reverse DNS lookups included that also needed to be removed.

Table 3.1- PCAP files used in the analysis

URL	File(s)	Description
https://github.com/elastic/examples/tree/master/Security%20Analytics/dns_tunnel_detection	dns-tunnel-iodine.pcap	Tunnelled traffic generated using *Iodine*
https://wiki.wireshark.org/SampleCaptures#Sample_Captures	SkypeIRC.cap	Capture of legitimate DNS traffic generated from *Skype*
http://www.labofapenetrationtester.com/2015/05/	powercat_dns.pcapng	Tunnelled traffic generated using *PowerCat*
http://tcpreplay.appneta.com/wiki/captures.html	smallFlows.pcap bigFlows.pcap test.pcap	Capture of legitimate DNS traffic generated from various applications

3.5 Modelling

The goal of the research project is to determine how effective the different classification techniques, LR, DT, RF, and SVM, are at predicting that a DNS request is legitimate or part of a tunnel. For each technique, the first step will be to build a model with a single predictive variable. The remaining predictive variables will be added in subsequent steps to see what impact they have on the effectiveness of the model. After each model is built, it will be evaluated in terms of the Sensitivity, Specificity, Accuracy, and Precision scores achieved. To see if the differences are statistically significant, a Cochran's Q test will be performed on the Accuracy of the results. The test will only be completed when all predictive variables are included, as these models are likely to be the most stable and should account for the greatest amount of variance in the request characteristics.

For the LR, DT and RF classification techniques, the models produced will be tuned with a *cross validator* to try and improve the effectiveness and reliability of the results. The parameters used are listed in Table 3.2: Tuning parameters for LR, DT and RF models. The values for these parameters were chosen to try and reduce the amount of overfitting of the models with the known data, and to set a maximum on the number of iterations allowed to stabilise the models. Increasing the number of iterations can have a significant impact on the time required to create the models, but can also improve the reliability of the results. For the SVM classification technique, models will be built using the RBF and Linear kernel techniques.

Table 3.2 - Tuning parameters for LR, DT and RF models

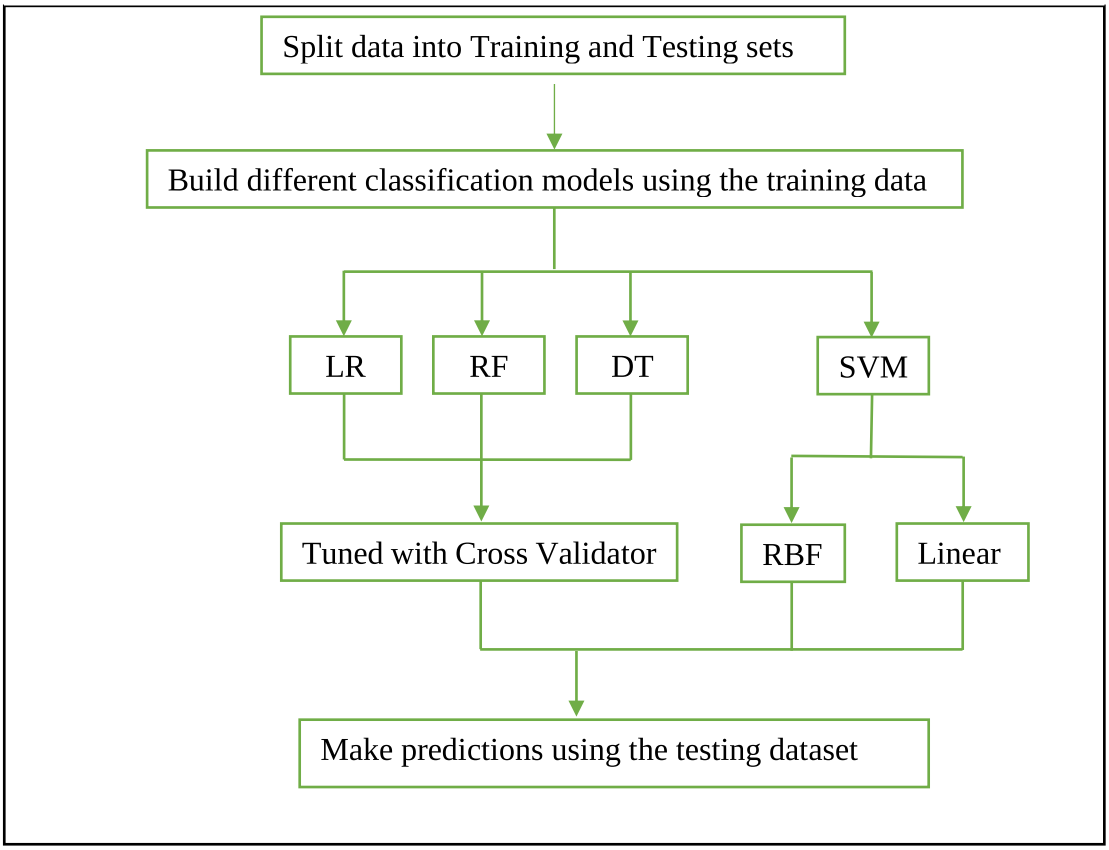
Parameter	**Description**	**Values**
regParam	Regularisation Parameter – This can help avoid overfitting of a model by reducing the influence of variables in the dataset that have a large impact	[0.01, 0.5, 1.0]
elasticNetParam	Add random noise to the dataset to lessen the risk of overfitting in the models	[0.0, 0.3, 0.5]
maxIter	Set a maximum on the number of iterations permitted to stabilise the values of the coefficients	[5, 7, 10]

3.6 Evaluation

To evaluate the models, the results will be examined to determine the Sensitivity, Specificity, Accuracy and Precision of each model.

Sensitivity will measure the percentage of tunnel requests correctly predicted by the model. Specificity will measure the percentage of legitimate requests correctly predicted by the model. Accuracy will measure the percentage of requests correctly predicted. Precision will measure the percentage of predicted tunnel requests that were correct.

Figure 3.2 - Steps to Build Classification Models



The results in each category will be calculated as follows:

$$Sensitivity = \frac{TP}{TP + FN}$$

$$Specificty = \frac{TN}{TN + FP}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

- TP (true positive) is the number of correctly predicted tunnel requests
- TN (true negative) is the number of correctly predicted legitimate requests
- FP (false positive) is the number of incorrectly predicted tunnel requests
- FN (false negative) is the number of incorrectly predicted legitimate requests

To calculate TP, TN, FP and FN, the results from the predictions from each model will be compared to the actual categories for all of the DNS requests in the test data. For the models with all predictive variables included, a Cochran's Q test will be performed on the TP and TN values to see if any differences between the techniques are significant.

For the LR, DT & RF models, the *AreaUnderROC* (Receiver Operating Characteristics) and the *AreaUnderPR* (Precision and Recall) values will also be examined. The *AreaUnderROC* curve tries to maximise the accuracy of the model by looking at the true positive rate and the false positive rate (note: this is equivalent to the sensitivity and 1 – specificity). *AreaUnderPR* tries to maximise accuracy by looking at the precision and recall (sensitivity) of the model. It is typically used in scenarios where there is a sizeable difference in the number of rows for each category in the sample. In this research, the number of tunnel and legitimate requests are relatively equal, so *AreaUnderROC* may be a better measure of accuracy.

3.7 Deployment

The final phase of the CRISP-DM methodology is the deployment phase. For this project, the deployment phase is the production of this report that will include the analysis of the results, any conclusions reached, and any future recommended research in the area to expand on the work carried out.

3.8 Summary

This research is designed to determine if binary classification techniques are equally effective at predicting if an individual DNS request is tunnel related, and if not, which technique gives the best results. To achieve this, existing datasets of DNS requests, that are known to be either tunnel related or legitimate, will be used. The characteristics of these requests will be examined to select suitable predictive variables to be used in four different classification techniques, Logistic Regression, Decision Tree, Random Forest, and Support Vector Machine. Each of these models will then be evaluated in terms of the Sensitivity, Specificity, Accuracy, and Precision scores obtained. Differences between the results obtained will be examined using a Cochran's Q test to see if they are statistically significant.

4 IMPLEMENTATION AND RESULTS

4.1 Introduction

This chapter will review the results obtained during the analysis of the dataset being used and the effectiveness of the classifications models produced. The differences in the average and variance across the various characteristics will be examined, along with the collinearity between them, in order to justify the inclusion or elimination of the characteristic as a suitable predictive variable for the models. The effectiveness will be measured for all models produced under the different classification techniques, starting with one predictive variable and then adding the remaining variables. The results show that each technique was effective at predicting tunnels with varying degrees of success across each of the measures. The difference in accuracy between the classification techniques in the final models was shown to be statistically significant using a Cochran's Q Test.

4.2 Data Analysis

4.2.1 Summary Statistics

The first step in the process was to produce summary statistics for all the candidate predictor variables available for the models. Table 4.1 - Summary Statistics shows the results of the analysis across all requests characteristics for legitimate and tunnel related traffic in the dataset. (Note: there were 2221 legitimate requests and 3425 tunnel related requests in the dataset). This helped indicate that differences existed for the value and distribution of the characteristics between the two categories. For example, the median and inter quartile ranges of the *length* variable differs between the tunnel and legitimate traffic.

4.2.2 Difference Test

A Kolmogorov-Smirnov test for normality was then conducted for each characteristic per category. The results in Table 4.2: Kolmogorov-Smirnov normality test indicate that none of the characteristics in either category followed a normal distribution. As a result, a non-parametric test, Mann–Whitney U, was used to show that the distributions of the characteristics was different between legitimate and tunnelled traffic. The results can be seen in Table 4.3: Mann-Whitney difference test. This suggests that each of the characteristics can be used to determine if a request is legitimate or part of a DNS tunnel, as there are significant differences between the categories.

Table 4.1 - Summary Statistics

Variable – Tunnel Y/N	**Min**	**Max**	**Mean**	**SD**	**Q1**	**Med**	**Q3**
Length - N	6	72	21.25	9.10	15	20	24
Length - Y	4	246	38.47	50.49	27	27	27
Length of Sub Domain - N	1	53	6.81	5.78	3	5	9
Length of Sub Domain - Y	4	62	4.91	6.27	4	4	4
Sub Domain Count - N	1	9	3.43	1.11	3	3	4
Sub Domain Count - Y	1	13	6.05	1.97	6	6	6
Character Distribution - N	5	30	13.11	3.77	11	13	15
Character Distribution - Y	4	107	11.34	5.78	10	11	12
Entropy - N	2.25	4.71	3.50	0.36	3.25	3.48	3.69
Entropy - Y	1.71	6.58	3.06	0.35	2.99	3.10	3.18

Table 4.2 - Kolmogorov-Smirnov Normality Test

Variable	**Tunnel**	**Statistic**	**P-Value**
Length	No	0.15	< 0.01
Length	Yes	0.52	< 0.01
Length of Sub Domain	No	0.18	< 0.01
Length of Sub Domain	Yes	0.49	< 0.01
Sub Domain Count	No	0.25	< 0.01
Sub Domain Count	Yes	0.46	< 0.01
Character Distribution	No	0.15	< 0.01
Character Distribution	Yes	0.34	< 0.01
Entropy	No	0.06	< 0.01
Entropy	Yes	0.22	< 0.01

Table 4.3 - Mann-Whitney Difference Test

Variable	**U-Value**	**P-Value**
Length	1589705.5	< 0.01
Length of Sub Domain	3125035.5	< 0.01
Sub Domain Count	595366.0	< 0.01
Character Distribution	2358067.0.0	< 0.01
Entropy	985841.0	< 0.01

4.2.3 Correlation

The results of the correlation tests between the request characteristics indicate a strong correlation between the *Length* and the *Sub Domain Count* variables, and between the *Entropy* and the *Character Distribution* variables – see Table 4.4: Correlation between characteristics. As a result, the *Sub Domain Count* and *Character Distribution* variables were eliminated from the model. This should not impact on the reliability of the model as the variance associated with the eliminated variables should be accounted for in the *Length* and *Entropy* characteristics. The remaining variables, *Length*, *Length of Sub Domain*, and *Entropy*, were all selected to be used as predictive variables in the model. Although only three predictor variables remain, they should contribute enough to the variance between categories to develop reliable classification models.

Table 4.4 - Correlation between Characteristics

Characteristics	**Length**	**Sub Domain Length**	**Sub Domain Count**	**Char. Dist.**	**Entropy**
Length	n/a	0.099	0.724	0.45	-0.035
Sub Domain Length	0.099	n/a	-0.096	0.139	-0.144
Sub Domain Count	0.724	-0.096	n/a	0.539	0.221
Char. Dist.	0.45	0.139	0.539	n/a	0.772
Entropy	-0.035	-0.144	0.221	0.772	n/a

4.2.4 Data Visualisation

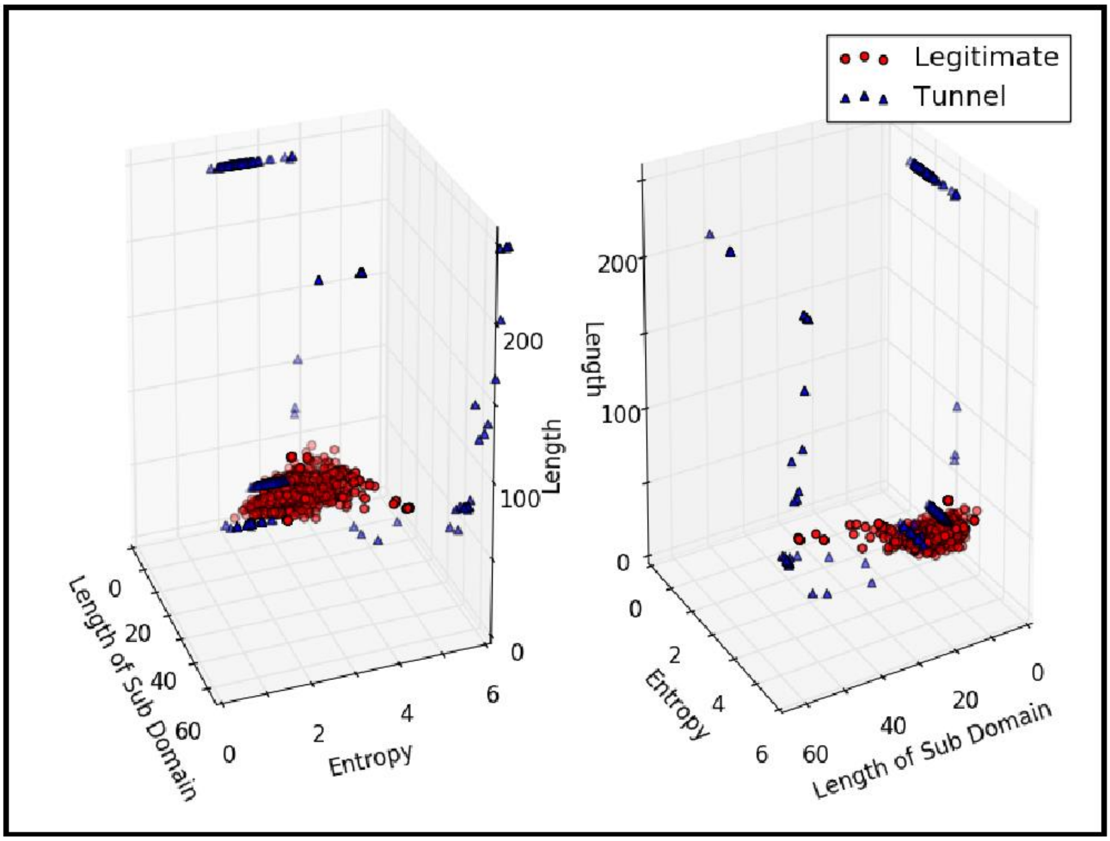
Classification models work by examining the differences in the predictive variables between different groups to derive a model that can accurately predict which category a set of variables is more likely to belong to. Visual representations of the differences between categories can give an indication of how the models can separate the groups.

For this study, a number of graphs were created that show the distribution of the predictive variables between legitimate and tunnel related traffic. Scatter plots of all the variables demonstrate how the different groups separate in a three dimensional graph, box plots and histograms give an indication of the distribution of individual variables between the categories.

4.2.4.1 Scatter Plots

This figure shows two scatter plots of the three characteristics to be used as input predictive variables in the classification models. They are two views of the same scatter plot (with one rotated) to help highlight the differences between the predictive variables for tunnel and legitimate requests.

Figure 4.1 - Scatter Plot of Predictive Variables



The goal of the classification models is to try and separate out the different categories into distinct regions to maximise the number of requests that are correctly predicted. The plots indicate distinct differences in the *Length* variable between legitimate and tunnel related requests, although some overlap does exist. The higher values for *Length of Sub Domain* appear to be mostly related to tunnel requests, but there is a lot of overlap with legitimate requests for the lower values. There does not appear to be any clear distinction for the *Entropy* variable between categories. This gives an indication that the *Length* variable will be the dominant predictor for each of the classification models produced and that *Entropy* is unlikely to have much of an impact.

4.2.4.2 Box Plots

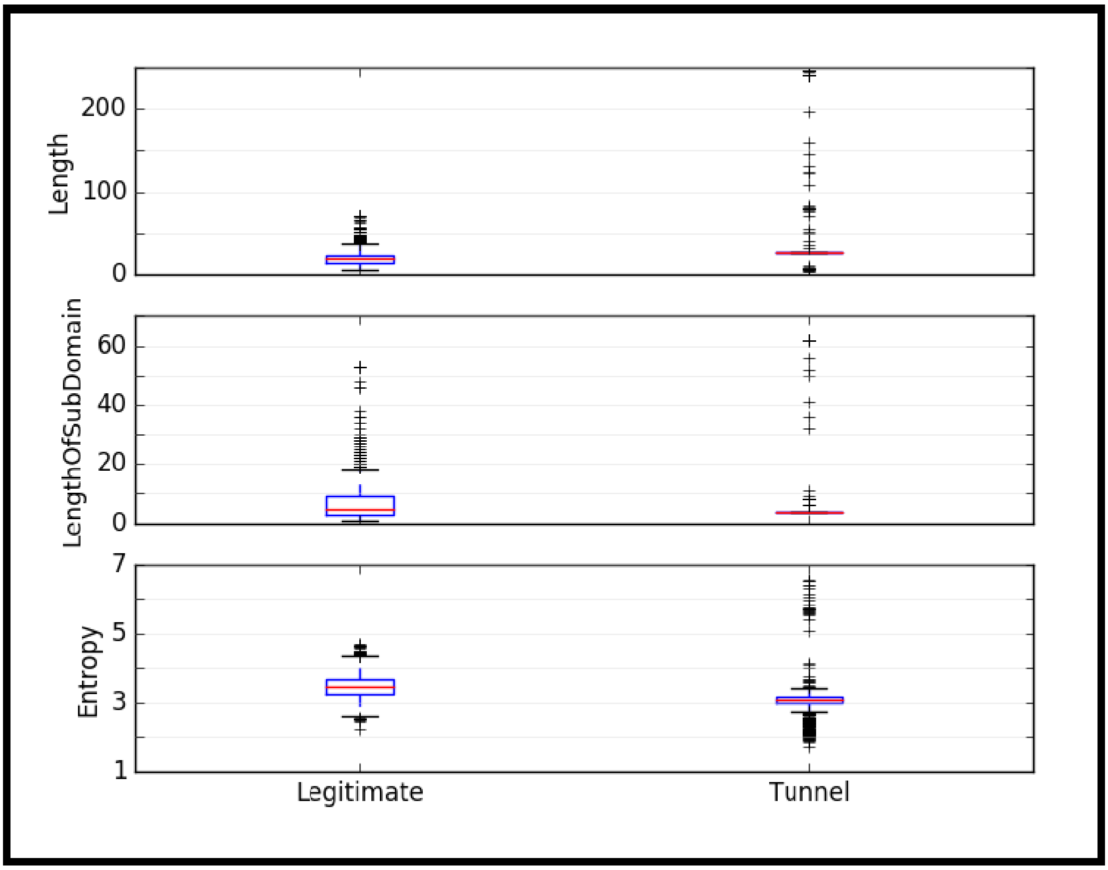
Box plots can give a visual representation of the median and inter-quartile range for small datasets and give an indication of how normally distributed it is. In Figure 4.2 - Box Plot of Predictive Variables, the box represents the 1st and 3rd quartile of the data, the red line in the middle of the box represents the median, the dashed lines represent the variation outside the quartiles, and the individual points represent the outliers. Values above the 3rd quartile indicate the data is positively skewed, ones below the 1st quartile indicate negative skewness, and an even spread indicate a normal distribution.

The number of outliers and position of the median line in the graphs suggests that the distribution of the predictive variables do not follow a normal distribution, although the *Entropy* values for legitimate traffic appears to be close to normal with some outliers. This supports the results of the Kolmogorov-Smirnov test presented earlier. Comparing box plots between legitimate and tunnel related traffic demonstrates the difference in distribution between the two categories, which helps indicate how they can be separated by the classification models.

Table 4.5 - Box Plot Distribution Results

Predictive Variable	**Legitimate**	**Tunnel**
Length	Positively Skewed	Positively Skewed
Length of Sub Domain	Positively Skewed	Positively Skewed
Entropy	Near Normal	Negatively Skewed

Figure 4.2 - Box Plot of Predictive Variables



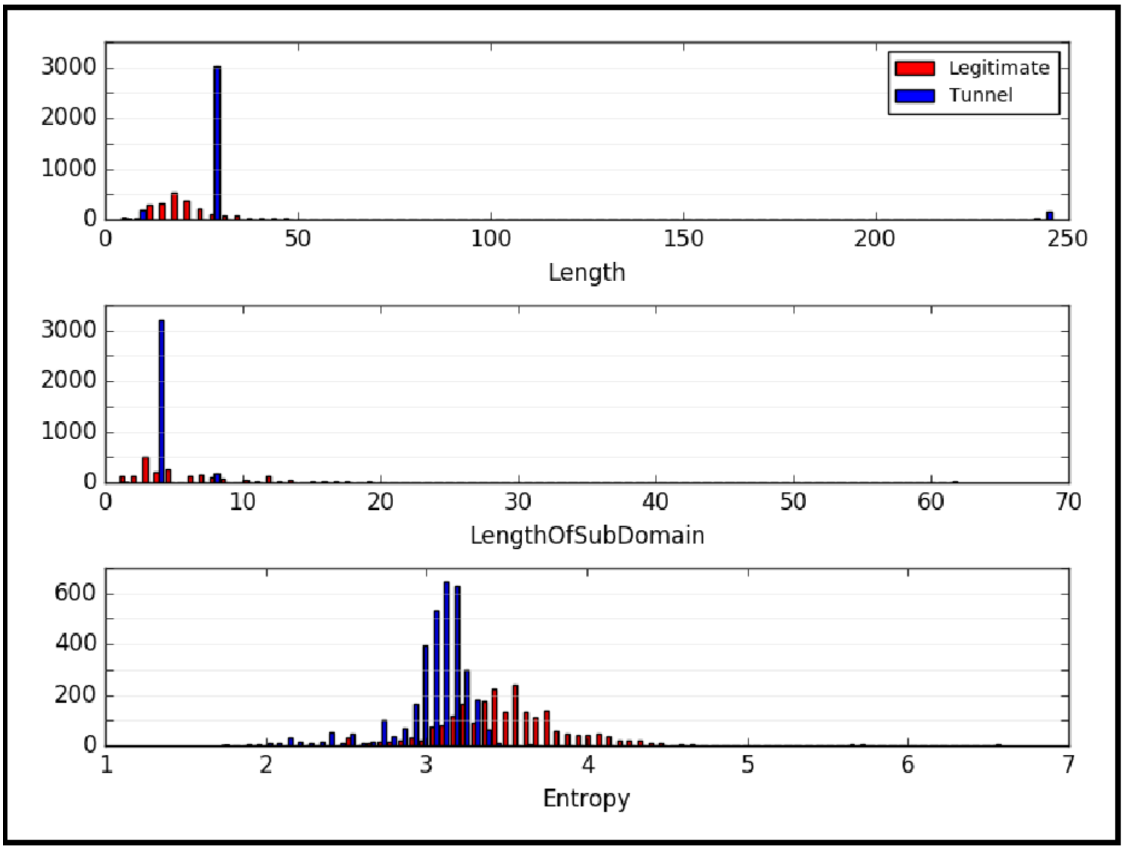
4.2.4.3 Histograms

Histograms can give an accurate visual representation of the distribution of medium to large datasets. If the data is normal distributed, you would expect to see a bell-shaped histogram, if not, it will be skewed to the left (negatively) or right (positively). The results in Figure 4.3 - Histogram of Predictive Variables are broadly in line with the boxplot and Kolmogorov-Smirnov test results. As with the boxplots, the distribution of the *entropy* value for legitimate requests appears to be approaching normal.

Table 4.6 - Histogram Distribution Results

Predictive Variable	**Legitimate**	**Tunnel**
Length	Positively Skewed	Positively Skewed
Length of Sub Domain	Positively Skewed	Positively Skewed
Entropy	Near Normal	Negatively Skewed

Figure 4.3 - Histogram of Predictive Variables



4.2.4.4 Individual Logistic Regression Curve

For each of the predictive variables selected, an individual logistic regression model (without tuning) was created to indicate how much it contributed to the variance in the outcome variable (tunnel or legitimate request). The output from the boundary equations equals the Intercept + Coefficient * Predictive Variable, and the probability of an individual request being part of a tunnel is defined as

$$P(X) = \frac{1}{1 + e^{-output}}$$

From this, individual regression curves were created showing the actual outcome along with the probability for each request. As with the classification models, the data was split between training and testing sets (70:30). Logistic regression models were then created for each predictive variable. The intercept and coefficients are in Table 4.7: Individual predictive variable logistic regression. For the dataset used in this analysis, only the *Length* variable came close to the s-type curve associated with logistic regression. As with the scatter plots, this suggests that *Length* will be the most effective in predicting tunnels.

Figure 4.4 - Logistic Regression Curve of Predictive Variables

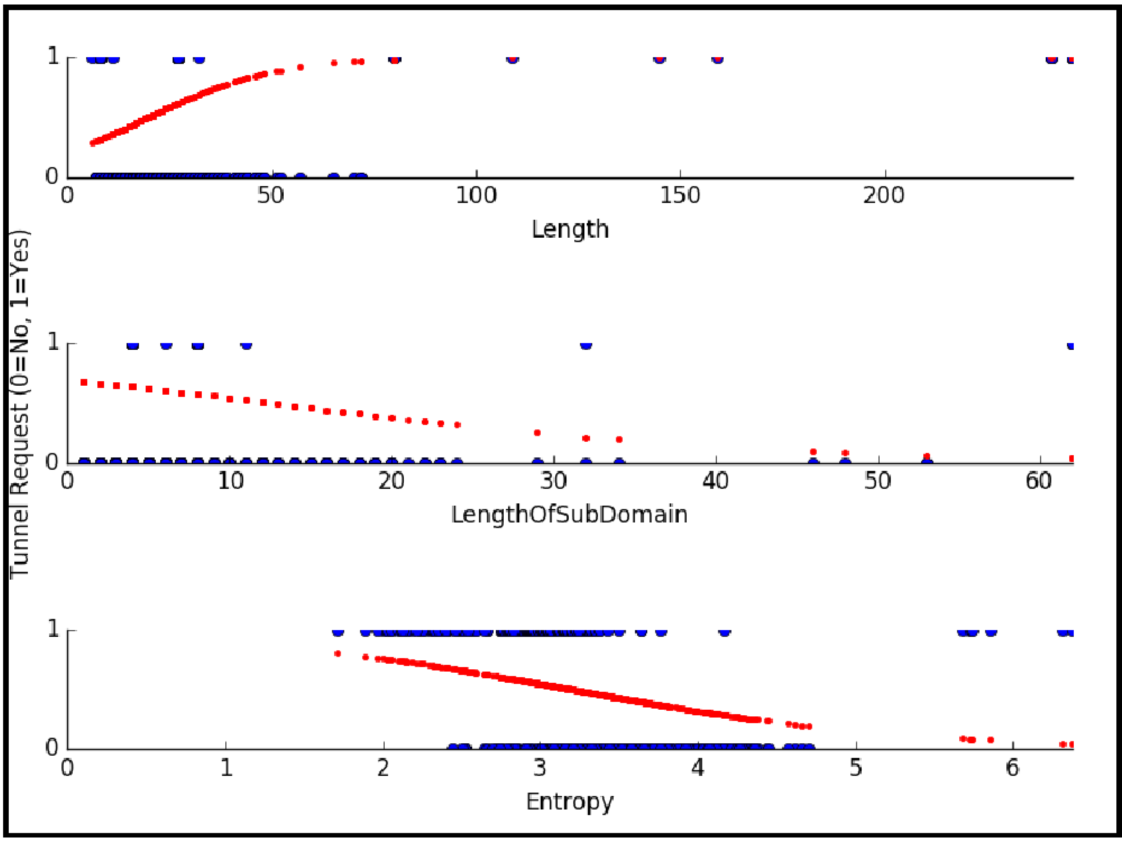


Table 4.7 - Individual Predictive Variable Logistic Regression

Characteristics	**Intercept**	**Coefficient**
Length	-1.2916	0.0653
Length of Sub Domain	0.8247	-0.0658
Entropy	3.0258	-0.9492

4.3 Classification Models

To create the models, the data was first split into *Training* and *Testing* sets (approximately 70:30). The total number of requests in the *Training* set was 4019, and in the *Testing* set it was 1627. All classification models will be evaluated by examining the Sensitivity, Specificity, Accuracy and Precision of the predictions as determined by the True Positive, True Negative, False Positive, and False Negative values. There are three predictive variables being used to create the models - *Length*, *Length of Sub Domain* and *Entropy*. Each classification technique will first create a model using only the *Length* variable used as a predictor. The remaining predictors will then be added, with the results for all models calculated at each stage.

The Logistic Regression, Decision Tree, and Random Forest models, will be tuned using the variables in Table 3.2: Tuning parameters for LR, DT and RF models to see if the results produced can be improved. For the Support Vector Machine, models using two different kernels will be created – Radial Basis Function and Linear, to see which one produces the best results.

4.3.1 Logistic Regression

LR calculates the probability of an outcome based on a set of predictive variables. The outcome required for this study is a binary value to indicate if a DNS request is predicted as being legitimate or part of a tunnel (0 = Legitimate Request, 1 = Tunnel Related). In this study, the probability threshold for predicting that a DNS request is tunnel related is set at 0.5.

For the base LR models, the results were between 50% and 94% across all measures depending on the number of variables used. When extra variables were added, the results increased by between 1% and 9% across all measures except sensitivity which dropped by 1%. The detailed results can be seen in the Base Logistic Regression Results appendix. When the LR models were tuned, the results with just one predictive variable were less than the base model, e.g. Specificity value was 16%. However, when all three predictive variables were used, the tuned model outperformed the base model with all measures in excess of 80% (Sensitivity was at 99%). See appendix Tuned Logistic Regression Results for detailed results.

4.3.2 Decision Tree

DT classifiers predict an outcome based on a series of input variables by creating a tree-like graph with each node on the tree representing a test, and each leaf a prediction. For example, a node might be "`if length > 10`" and the leaf (or prediction) could be "`tunnel=Y`". Each node can have either a child node or a leaf attached.

For this study, the DT techniques achieved close to 99% across all measures when all predictive variables were included. The *Length* predictive variable alone achieved a minimum of 96% in all measures. There was no material change when the model was tuned. See appendix Base Decision Tree Results and Tuned Decision Tree Results for details on the results.

4.3.3 Random Forest

Overfitting in a DT can impact on the reliability of the results obtained. It can occur if the sample size is small relative to the number of predictors. To fix these errors, a series of DTs are created using subsets of the available predictors and test data. The results are then combined into a single tree. This type of classification is called a Random Forest.

In this study, the base RF models and the tuned RF models gave very similar results as the DT models with no significant difference for any of the measures. See Base Random Forest Results and Tuned Random Forest Results for details.

4.3.4 Support Vector Machine

SVM classification creates a hyperplane that divides data into two distinct categories based on a set of predictive variables. The hyperplane is created using a kernel trick to map the data in different dimensions. For this research, a Radial Basis Function (RBF) kernel and a linear kernel were created.

The results for the RBF kernel were marginally better than the Decision Tree models, with every measure at 99% when all predictive variables were included. The linear kernel was slightly less successful with results between 92% and 96%. See SVM RBF Kernel Results and SVM Linear Kernel Results for details.

4.3.5 Statistical Significance

When all predictive variables are included, there is a significant difference (Q = 1407, p-value<0.01) in the prediction scores from the set of classification techniques tested. In order to see if the differences between individual techniques was significant, pairwise tests were performed. This demonstrated that the LR, Tuned-LR, and SVM-Linear models produced significantly different results to the other techniques (p-value<0.01). The differences between the DT, RF and SVM-RBF techniques were not significant. The following table highlights the q-test & p-value between pairs of classification models (note: tuned DT and tuned RF were excluded as they had the same results as the base DT and RF models):

Table 4.8 - Cochran Q Test Results

Models	**LR**	**T-LR**	**DT**	**RF**	**SVM-RBF**	**SVM-Linear**
LR	n/a	q=118 p<=0.01	q=301 p<=0.01	q=300 p<=0.01	q=309 p<=0.01	q=196 p<=0.01
T-LR	q=118 p<=0.01	n/a	q=123 p<=0.01	q=120 p<=0.01	q=108 p<=0.01	q=195 p<=0.01
DT	q=301 p<=0.01	q=123 p<=0.01	n/a	q=1 p=0.32	q=0.33 p=0.56	q=77 p<=0.01
RF	q=300 p<=0.01	q=120 p<=0.01	q=1 p=0.32	n/a	q=0.08 p=0.78	q=74 p<=0.01
SVM - RBF	q=309 p<=0.01	q=108 p<=0.01	q=0.33 p=0.56	q=0.08 p=0.78	n/a	q=63.75 p<=0.01
SVM - Linear	q=196 p<=0.01	q=195 p<=0.01	q=77 p<=0.01	q=74 p<=0.01	q=63.75 p<=0.01	n/a

4.4 Summary

The results from the data analysis indicated that differences existed between legitimate and tunnel related traffic across all request characteristics. Correlation between the characteristics led to the elimination of the *Sub Domain Count* and *Character*

Distribution characteristics. The three remaining characteristics, *Length*, *Length of Sub Domain*, and *Entropy*, were all selected as predictive variables for the classification models.

Using these three variables, models were built using four different classification techniques – Logistic Regression, Decision Tree, Random Forest, and Support Vector Machine. Three models were built using each technique, the first used only the Length variable, the second added the *Length of Sub Domain*, and the third added *Entropy*. The effectiveness of the models was then measured by calculating the Sensitivity, Specificity, Accuracy, and Precision, using a confusion matrix of True Positive, True Negative, False Positive, and False Negative scores obtained. The results achieved for each technique demonstrated their effectiveness at predicting DNS tunnels based on the set of characteristics chosen as predictive variables.

Overall, the difference in predictions between the classification techniques was statistically significant when all predictive variables were included. Comparisons between pairs of techniques showed that the differences between DT, RF and SVM-RBF models were not statistically significant. The differences in all the other models were statistically significant.

5 ANALYSIS, EVALUATION AND DISCUSSION

5.1 Introduction

This section will compare the results of each of the classification techniques and then look at the strengths and weaknesses of the study. The strength of the study is in the demonstration of classification models to effectively predict tunnel related traffic and the identification of which models produce the best results, the weakness is in the variability of the data used to build the models.

5.2 Comparison of Models

5.2.1 Logistic Regression

For the base LR model, the Specificity, Accuracy and Precision scores obtained increased as each predictive variable was added. The Sensitivity score remained largely the same. As more predictors were included and the variance in the data used to create the models increased, it is likely that the final model produced with all three predictive variables would be more reliable in a production environment.

When the models were tuned, the *Length* variable produced an unusually low score for Specificity (16%). This may be due to the noise added during the tuning. Once all the variables were added, the score increased to over 80% along with significant increases for the other measures. Tuning the model allowed the algorithm to try and avoid overfitting and increased the number of iterations in order to achieve a stable result. As with the base model, the effectiveness of the tuned models increased with more predictive variables, and again gave potentially more reliable results because of the increased variability produced by using more characteristics. Overall, the tuned LR models gave better results than the base LR models in predicting which category a request belonged to.

5.2.2 Decision Tree

The base DT models gave significantly better results than either LR model in predicting DNS tunnels. The effectiveness of the base DT model with a single predictive variable was above 96% across all measures, this increased to being above 98% with the introduction of the second and third predictive variables. This could be interpreted as demonstrating the effectiveness of the technique, but there is also the risk that model over fitted the data (due to the lack of variance in the test data) and that it would not be as effective in a production environment.

To try and reduce this risk of overfitting, the model was again tuned to add a level of random noise to the data and increase the number of iterations allowed to stabilise the results. In this case the tuning had no impact on the results obtained. This is potentially because the tuned model simply returned the best results which were obtained using the base model.

5.2.3 Radom Forest

The RF algorithm is used to avoid overfitting in decision trees by creating a series of trees with different predictive variables and test data, and then combining them to give an overall result. For this data, neither the base nor the tuned RF models produced significantly different results to the DT models. This was probably down to the number of predictive variables and the size of the dataset available. If there were more predictive variables and data available to the model, then the RF technique may be more useful at increasing the reliability of the results.

5.2.4 Support Vector Machine

Two different SVM models were built for this research – one using an RBF kernel and one using a linear kernel. The results for the RBF kernel were 99% across all measures once the second and third predictive variables were introduced. Although there was no change when the third variable was introduced, it may make the model more reliable as there is an increased variance in the data taken into account. These results were significantly better than the LR models, but equivalent to the DT and RF models.

The linear kernel was less effective than the RBF kernel at predicting tunnels which reflects the restrictive nature of a linear separation of the data. The results for a single predictive variable were between 62% and 94%. The increases in scores when the second and third variable were added to the model was around 10% across most of the measures. The linear SVM did achieve better results than the LR models, but was less effective than the DT, RF, and SVM-RBF models.

5.3 Strength of Results

The strength of this study is in the demonstration of which binary classification techniques are the most effective at predicting if a single DNS request is tunnel related using only the characteristics of the request as predictor variables. It describes how the selected techniques operate to categorise data and how their effectiveness can be measured and compared. It also shows how to justify the inclusion of predictive variables and explains under what circumstances they should be eliminated.

5.4 Limitations

The dataset used in the research contained only a limited number of legitimate and tunnel related DNS requests, which were retrieved from existing sets of publically available PCAP captures. Some consistencies in the test data meant that certain characteristics could not be used in the classification models – such as source IP address and DNS response. The sources of the legitimate traffic were limited and only two tools were used to create the tunnel related traffic. This will have restricted the variety of data in the research and could have had an impact on the results obtained.

5.5 Summary

The total number of DNS requests in the test set was 1627 (this excludes the data used to train the models). This was made up of 963 (59%) tunnel related requests and 664 (41%) legitimate requests. All four classification techniques are capable of predicting that a single request is part of a DNS tunnel, based on the request characteristics, with varying degrees of success.

The base LR model correctly classified 76% of requests with only the *Length* variable used as a predictor, the accuracy increased when more predictive variables were added. The tuned LR model achieved a 91% accuracy when all variables were added.

The DT models achieved a 97% accuracy with a single predictive variable, this increased to 99% when the second and third predictive variables were added. There was no change in results when the DT model was tuned. The results for the RF models were the same as the DT models.

The SVM RBF model achieved a 98% accuracy with just the *Length* variable, a 99% success rate was achieved when the extra variables were added. The SVM Linear models did not achieve the same success rate, 81% for a single variable, and 94% when all predictive variables were included.

The following highlights the results for each model, using the values from the confusion matrix to calculate the percentages in each category. The tuned DT models and RF models are excluded from the results as they matched the base DT model. See Table 5.1 - Confusion Matrix Percentages for Models for the results in a table format.

Figure 5.1 - Pie Charts of Confusion Matrix Percentages for Models

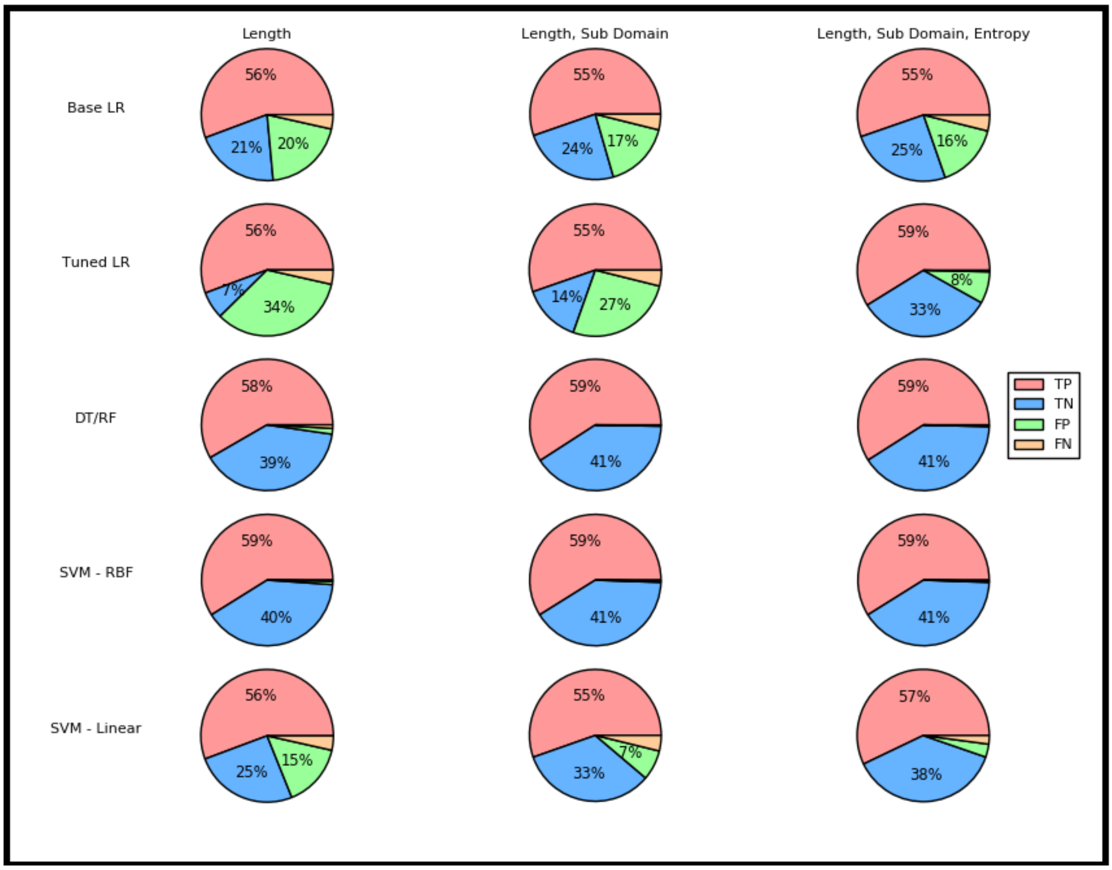


Table 5.1 - Confusion Matrix Percentages for Models

	Length	Length, Sub Domain	Length, Sub Domain, Entropy	
Base LR	55.69%	55.32%	55.32%	
	20.77%	24.15%	25.02%	
	20.04%	16.66%	15.80%	
	3.50%	3.87%	3.87%	
Tuned LR	55.69%	55.32%	58.82%	
	6.70%	14.26%	33.07%	
	34.11%	26.55%	7.74%	
	3.50%	3.87%	0.37%	
DT/RF	58.33%	59.13%	59.00%	TP
	39.40%	40.57%	40.57%	TN
	1.41%	0.25%	0.25%	FP
	0.86%	0.06%	0.18%	FN
SVM-RF	58.88%	58.88%	58.88%	
	40.01%	40.57%	40.57%	
	0.80%	0.25%	0.25%	
	0.31%	0.31%	0.31%	
SVM - Linear	55.69%	55.38%	57.10%	
	25.38%	33.44%	37.62%	
	15.43%	7.38%	3.20%	
	3.50%	3.81%	2.09%	

The high success rate in categorising a request as being tunnel related demonstrates the potential of classification models to be used to help prevent data exfiltration from a network. The DT, RF, and SVM-RBF models gave the highest success rate among all the techniques. This suggests that the differences in DNS request characteristics between tunnel and legitimate requests are not as easily separated using linear techniques, and that the outliers in the values of the predictive variables can be more easily managed using non-linear techniques. The differences between DT, RF, and SVM-RBF techniques are not statistically significant when all three predictive variables were included. The differences between them and the LR and linear SVM models were significant.

6 CONCLUSION

A Cochran's Q test on the accuracy of all the classification techniques indicate that there are statistically significant differences in accuracy between the models produced (Q = 5944, p-value<0.01). This allows us to reject the null hypothesis that there are no statistically significant differences in the prediction of DNS tunnels from the classification techniques used.

The results of the study indicate that all the binary classification techniques used are effective at predicting if an individual request is part of a DNS tunnel, with the Decision Tree, Random Forest, and Support Vector Machine (with an RBF kernel) techniques producing the best results. These techniques had statistically significant differences to the Logistic Regression and Linear SVM models.

6.1 Research Overview

This research looked at the problem of data exfiltration from a private network using a DNS tunnel. DNS is a fundamental protocol on the internet acting as an address book to transmit messages from one machine to another. It is rarely monitored to the same extent as other protocols which makes it a target for anyone trying to avoiding internet paywalls, circumvent censorship laws, or extract information from a network. The availability of tools such as *Iodine* and *Powercat* has made the creation of a DNS tunnel more accessible to individuals with only a limited technical know-how.

The research reviewed the history of DNS and how it can be used as a covert channel, and then looked at the various techniques available to identify that a DNS tunnel exists on a network – primarily anomaly based detection. Most of the existing research focuses on identifying tunnels by looking at anomalies in network traffic between protocols or over different periods of time. The basis of this research was to determine if differences existed between binary classification techniques in their effectiveness at categorising an individual DNS request as being legitimate or tunnel related based on the request characteristics, and to see which of them were the most effective.

6.2 Problem Definition

The problem this research tries to address is the ability of different classification techniques to identify individual requests as being related to a tunnel before the request leaves the network with the potential loss of data. There are a number of different classification techniques available to categorise data, for this study Logistic Regression, Decision Tree, Random Forest, and Support Vector Machine were selected. The classification techniques operated differently with some more suitable to non-linearly separable data than others. The research looked at the four different techniques to determine how successful they were at predicting DNS tunnels, and if differences existed between the results obtained from each technique.

6.3 Design/Experimentation, Evaluation & Results

The data used in the experiment was sourced from existing datasets of DNS traffic that had previously being identified as being either legitimate or tunnel related. From this, a number of characteristics were extracted and used as candidate predictive variables for the classification models. For each candidate predictive variable, summary statistics were generated which indicated that differences existed for each of them between tunnel and legitimate traffic. The correlation between the candidate variables was then examined in order to eliminate multicollinearity which can impact on the reliability of the models. This resulted in three predictive variables being selected for the classification models – *Length*, *Length of Sub Domain* and *Entropy*.

Models for each classification technique were first built using only the *Length* variable. The *Length of Sub Domain* and *Entropy* variables were added in subsequent models. For each model produced, the Sensitivity, Specificity, Accuracy, and Precision, were measured using a confusion matrix of True Positive, True Negative, False Positive, and False Negative scores obtained.

The results demonstrate that a Decision Tree, Random Forest, and Support Vector Machine with an RBF kernel, are the most effective techniques to classify individual DNS requests as being legitimate or tunnel related.

6.4 Contributions and impact

The study demonstrates the ability of LR, DT, RF, and SVM classification techniques at identifying tunnel related traffic using only the characteristics of an individual requests as predictor variables, and highlights the significant differences between them and how those differences can be measured. It also helps demonstrate the process of selecting characteristics to be used as predictive variables and the impact they have on the models.

6.5 Future work & recommendations

The data for this research was obtained from sample PCAP files of DNS requests. The only tools used for the tunnel related traffic were *Iodine* and *Powercat*. Although the data was enough to demonstrate the effectiveness of the different classification techniques at identifying tunnels, there may not have been the variance in the data to distinguish correctly between the models produced. This is particularly true of some of the request characteristics (such as source IP address) that needed to be excluded due to unrealistic consistencies between the two groups.

An extension of the research would be to work with an organisation that has previously identified a DNS tunnel related data exfiltration attack, and then train the different classification models with this data to see which of them produces the best results. This could potentially allow extra characteristics to be included and produce more stable and reliable models that could be compared to each other. Recognising real words in a DNS request may also help differentiate between legitimate and tunnel related requests, although this should be reflected in the value of Entropy. An increased number of legitimate and tunnel related requests available to train the classification techniques may show more significant differences in the values for some of the characteristics used as predictive variables.

7 BIBLIOGRAPHY

Adkins, G. (2013). *Utilizing-Cyber-Espionage-to-Combat-Terrorism-Adkins.pdf*. The University of Texas at El Paso, USA. Retrieved from https://www.utep.edu/liberalarts/nssi/_Files/docs/Theses1/Utilizing-Cyber-Espionage-to-Combat-Terrorism-Adkins.pdf

Antonakakis, M., Perdisci, R., Nadji, Y., Vasiloglou, N., Abu-Nimeh, S., Lee, W., & Dagon, D. (2012). From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware. In *Proceedings of the 21st USENIX conference on Security symposium (Security'12).* (Vol. 12, pp. 24–24). Bellevue, WA: USENIX Association, Berkeley, CA, USA. Retrieved from https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final127.pdf

Ben-Hur, A., & Weston, J. (2010). A User's Guide to Support Vector Machines. *Methods in Molecular Biology (Clifton, N.J.)*, *609*, 223–239. https://doi.org/10.1007/978-1-60327-241-4_13

Born, K., & Gustafson, D. (2010). Detecting DNS Tunnels Using Character Frequency Analysis. *ArXiv Preprint ArXiv:1004.4358*, 1–12.

Butler, P., Xu, K., & Yao, D. (2011). Quantitatively Analyzing Stealthy Communication Channels. In *Applied Cryptography and Network Security* (pp. 238–254). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-21554-4_14

Callahan, T. (2013). *Understanding Internet Naming: From the Modern DNS Ecosystem to New Directions in Naming*. Case Western Reserve University.

Ellens, W., Żuraniewski, P., Sperotto, A., Schotanus, H., Mandjes, M., & Meeuwissen, E. (2013). Flow-based Detection of DNS Tunnels. In *IFIP International Conference on Autonomous Infrastructure, Management and Security* (pp. 124–135). Springer. https://doi.org/10.1007/978-3-642-38998-6_16

Garofalakis, M., Hyun, D., Rastogi, R., & Shim, K. (2000). Efficient algorithms for constructing decision trees with constraints (pp. 335–339). ACM Press. https://doi.org/10.1145/347090.347163

Giani, A., Berk, V. H., & Cybenko, G. V. (2006). Data Exfiltration and Covert Channels. In *Proceedings for Sensors Command Control Commun. Intell. (C3I) Technol. Homeland Security Homeland Defense V* (Vol. 6201, pp. 1–11). https://doi.org/10.1117/12.670123

Hoffman, C., Johnson, D., Yuan, B., & Lutz, P. (2012). A Covert Channel in TTL Field of DNS Packets. In *Proceedings of the International Conference on Security and Management (SAM)* (pp. 1–6). Las Vegas, USA. Retrieved from http://scholarworks.rit.edu/cgi/viewcontent.cgi?article=1301&context=other

Hudaib, A. A. Z., & Hudaib, E. A. Z. (2014). DNS Advanced Attacks and Analysis. *International Journal of Computer Science and Security (IJCSS)*, *8*(2), 63–74.

Jyothsna, V., Prasad, V. R., & Prasad, K. M. (2011). A Review of Anomaly Based Intrusion Detection Systems. *International Journal of Computer Applications*, *28*(7), 26–35.

Kalutarage, H. K. (2013). *Effective Monitoring of Slow Suspicious Activities on Computer Networks.* Coventry University. Retrieved from https://curve.coventry.ac.uk/open/file/afdbba5c-2c93-41a7-90c3-2f0f3261b794/1/Kalutarage2013.pdf

Krishnan, S., Taylor, T., Monrose, F., & McHugh, J. (2013). Crossing the Threshold: Detecting Network Malfeasance via Sequential Hypothesis Testing. In *Dependable Systems and Networks (DSN), 2013 43rd Annual IEEE/IFIP International Conference on* (pp. 1–12). IEEE. Retrieved from http://ieeexplore.ieee.org/abstract/document/6575364/

Liu, D., Li, Z., Du, K., Wang, H., Liu, B., & Duan, H. (2017). Don't Let One Rotten Apple Spoil the Whole Barrel: Towards Automated Detection of Shadowed Domains (pp. 537–552). ACM Press. https://doi.org/10.1145/3133956.3134049

Llamas, D., Allison, C., & Miller, A. (2005). Covert Channels in Internet Protocols: A survey. In *Proceedings of the 6th Annual Postgraduate Symposium about the Convergence of Telecommunications, Networking and Broadcasting, PGNET* (Vol. 2005, pp. 1–5). Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.99.5859&rep=rep1&type=pdf

Mehta, N. (1999). Internetworking Technology Overview. Retrieved from http://fab.cba.mit.edu/classes/961.04/people/neil/ip.pdf

Middelesch, E. W. (2015). *Anonymous and Hidden Communication Channels: A Perspective on Future Developments*. University of Twente. Retrieved from http://essay.utwente.nl/66741/1/Middelesch_MA_DACS.pdf

Mongillo, M. (2011). Choosing Basis Functions and Shape Parameters for Radial Basis Function Methods. *SIAM Undergraduate Research Online*, *4*, 190–209. https://doi.org/10.1137/11S010840

Nadler, A., Aminov, A., & Shabtai, A. (2017). Detection of Malicious and Low Throughput Data Exfiltration Over the DNS Protocol. *ArXiv:1709.08395 [Cs]*, 1–11.

Parker, A. J., Yancey, K. B., & Yancey, M. P. (2016). Regular Language Distance and Entropy. *ArXiv:1602.07715 [Cs, Math]*. Retrieved from http://arxiv.org/abs/1602.07715

Paxson, V., Christodorescu, M., Javed, M., Rao, J. R., Sailer, R., Schales, D. L., … Weaver, N. (2013). Practical Comprehensive Bounds on Surreptitious Communication over DNS. In *USENIX Security Symposium* (pp. 17–32). Retrieved from http://www1.icsi.berkeley.edu/~mobinjav/publications/2013/DNS_USENIX13.pdf

Peng, C.-Y. J., Lee, K. L., & Ingersoll, G. M. (2002). An Introduction to Logistic Regression Analysis and Reporting. *The Journal of Educational Research*, *96*(1), 3–14. https://doi.org/10.1080/00220670209598786

Pope, M. B., Warkentin, M., Mutchler, L. A., & Luo, X. (Robert). (2012). The Domain Name System-Past, Present, and Future. *CAIS*, *30*. Retrieved from http://www.unm.edu/~xinluo/papers/CAIS2012.pdf

Raman, D., Sutter, B. D., Volckaert, S., Bosschere, K. D., Danhieux, P., & Buggenhout, E. V. (2013). *DNS Tunneling for Network Penetration* (Vol. 7839). Berlin, Heidelberg: Springer. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.727.1484&rep=rep1&type=pdf

Reyes, G. P. (2014). *Covert channel detection using flow-data*. University of Amsterdam. Retrieved from https://homepages.staff.os3.nl/~delaat/rp/2013-2014/p66/report.pdf

Sarmah, A., Dey, A., & Kumar, C. J. (2014). Analysis and Modeling of an Effective Anomaly Detection Techniques for Detecting Data Exfiltration of Network. *Analysis*, *2*(5), 90–93.

Satam, P., Alipour, H., Al-Nashif, Y. B., & Hariri, S. (2015). Anomaly Behavior Analysis of DNS Protocol. *J. Internet Serv. Inf. Secur.*, *5*(4), 85–97.

Sheridan, S., & Keane, A. (2015). Detection of DNS Based Covert Channels. In *European Conference on Cyber Warfare and Security* (pp. 267–275). Academic Conferences International Limited. Retrieved from https://www.researchgate.net/publication/282931276_Detection_of_DNS_based_covert_channels

Skow, T. K. (2016). *Protection Against DNS Tunneling Abuses on Mobile Networks*. Norwegian University of Science and Tecnology. Retrieved from https://brage.bibsys.no/xmlui/bitstream/handle/11250/2406856/15578_FULLTEXT.pdf?sequence=1

Son, S., & Shmatikov, V. (2010). The hitchhiker's guide to DNS cache poisoning. In *International Conference on Security and Privacy in Communication Systems* (Vol. 50, pp. 466–483). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-16161-2_27

Tien, D., & Kavakli, M. (Eds.). (2008). Fifth International Conference on Information Technology & Applications 2008: ICITA 2008 (pp. 1–7). Bathurst, N.S.W.: Macquarie Scientific Publishing. https://doi.org/10.1.1.496.6055

Van Antwerp, R. C. (2011). *Exfiltration Techniques: An Examination and Emulation*. University of Delaware. Retrieved from http://udspace.udel.edu/bitstream/handle/19716/10145/Ryan_VanAntwerp_thesis.pdf?sequence=1

Wolfgarten, S. (2006). *Investigating large-scale Internet content filtering*. Dublin City University, Ireland. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.133.5778&rep=rep1&type=pdf

Wright, N. (2012). DNS in Computer Forensics. *Journal of Digital Forensics, Security and Law*, *7*(2), 11–42. https://doi.org/10.15394/jdfsl.2012.1117

Xu, K., Butler, P., Saha, S., & Yao, D. (2013). DNS for massive-scale command and control. *IEEE Transactions on Dependable and Secure Computing*, *10*(3), 143–153. https://doi.org/10.1109/TDSC.2013.10

Yadav, S., Reddy, A. K. K., Reddy, A. L., & Ranjan, S. (2010). Detecting algorithmically generated malicious domain names. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement* (pp. 48–61). ACM. https://doi.org/10.1145/1879141.1879148

Zhang, H., Yao, D. D., & Ramakrishnan, N. (2014). Detection of stealthy malware activities with traffic causality and scalable triggering relation discovery (pp. 39–50). ACM Press. https://doi.org/10.1145/2590296.2590309

8 APPENDIX

8.1 Base Logistic Regression Results

Table 8.1 - Base Logistic Regression Confusion Matrix

N = 1627	**Predicted: No**	**Predicted: Yes**
Predictive Variables - Length		
Actual: No	TN = 338	FP = 326
Actual: Yes	FN = 57	TP = 906
Predictive Variables – Length and Length of Sub Domain		
Actual: No	TN = 393	FP = 271
Actual: Yes	FN = 63	TP = 900
Predictive Variables – Length, Length of Sub Domain, and Entropy		
Actual: No	TN = 407	FP = 257
Actual: Yes	FN = 63	TP = 900

Table 8.2 - Base Logistic Regression Evaluation Results

Predictors / Measures	**Length**	**Length & Length of Sub Domain**	**Length, Length of Sub Domain & Entropy**
ROC	0.79	0.88	0.93
PR	0.65	0.76	0.95
Sensitivity	94%	93%	93%
Specificity	50%	59%	62%
Accuracy	76%	79%	80%
Precision	73%	76%	77%

Table 8.3 - Base Logistic Regression Boundary Equation

Predictors	**Intercept**	**Length**	**Sub Domain**	**Entropy**
Length	-1.29	0.0654	N/A	N/A
Length & Sub Domain	-0.93	0.117	-0.289	N/A
Length, Sub Domain & Entropy	4.107	0.0959	-0.2445	-1.4412

8.2 Tuned Logistic Regression Results

Table 8.4 - Tuned Logistic Regression Confusion Matrix

N = 1627	**Predicted: No**	**Predicted: Yes**
Predictive Variables - Length		
Actual: No	TN = 109	FP = 555
Actual: Yes	FN = 57	TP = 906
Predictive Variables – Length and Length of Sub Domain		
Actual: No	TN = 232	FP = 432
Actual: Yes	FN = 63	TP = 900
Predictive Variables – Length, Length of Sub Domain, and Entropy		
Actual: No	TN = 538	FP = 126
Actual: Yes	FN = 6	TP = 957

Table 8.5 - Tuned Logistic Regression Evaluation Results

Predictors / Measures	**Length**	**Length & Length of Sub Domain**	**Length, Length of Sub Domain & Entropy**
ROC	0.79	0.88	0.96
PR	0.65	0.76	0.96
Sensitivity	94%	93%	99%
Specificity	16%	34%	81%
Accuracy	62%	69%	91%
Precision	62%	67%	88%

Table 8.6 - Tuned Logistic Regression Boundary Equation

Predictors	**Intercept**	**Length**	**Sub Domain**	**Entropy**
Length	-0.45	0.0341	N/A	N/A
Length & Sub Domain	-0.202	0.0559	-0.1378	N/A
Length, Sub Domain & Entropy	9.43	0.0898	0.0425	-3.483

8.3 Base Decision Tree Results

Table 8.7 - Base Decision Tree Confusion Matrix

N = 1627	**Predicted: No**	**Predicted: Yes**
Predictive Variables - Length		
Actual: No	TN = 641	FP = 23
Actual: Yes	FN = 14	TP = 949
Predictive Variables – Length and Length of Sub Domain		
Actual: No	TN = 660	FP = 4
Actual: Yes	FN = 1	TP = 962
Predictive Variables – Length, Length of Sub Domain, and Entropy		
Actual: No	TN = 660	FP = 4
Actual: Yes	FN = 3	TP = 960

Table 8.8 - Base Decision Tree Evaluation Results

Predictors / Measures	**Length**	**Length & Length of Sub Domain**	**Length, Length of Sub Domain & Entropy**
ROC	0.99	0.99	0.99
PR	0.99	0.99	0.99
Sensitivity	98%	99%	99%
Specificity	96%	99%	99%
Accuracy	97%	99%	99%
Precision	97%	99%	98%

8.4 Tuned Decision Tree Results

Table 8.9 - Tuned Decision Tree Confusion Matrix

N = 1627	**Predicted: No**	**Predicted: Yes**
Predictive Variables - Length		
Actual: No	TN = 641	FP = 23
Actual: Yes	FN = 14	TP = 949
Predictive Variables – Length and Length of Sub Domain		
Actual: No	TN = 660	FP = 4
Actual: Yes	FN = 1	TP = 962
Predictive Variables – Length, Length of Sub Domain, and Entropy		
Actual: No	TN = 660	FP = 4
Actual: Yes	FN = 3	TP = 960

Table 8.10 - Tuned Decision Tree Evaluation Results

Predictors / Measures	**Length**	**Length & Length of Sub Domain**	**Length, Length of Sub Domain & Entropy**
ROC	0.99	0.99	0.99
PR	0.99	0.99	0.99
Sensitivity	98%	99%	99%
Specificity	96%	99%	99%
Accuracy	97%	99%	99%
Precision	95%	98%	98%

8.5 Base Random Forest Results

Table 8.11 - Random Forest Confusion Matrix

N = 1627	**Predicted: No**	**Predicted: Yes**
Predictive Variables - Length		
Actual: No	TN = 641	FP = 23
Actual: Yes	FN = 14	TP = 949
Predictive Variables – Length and Length of Sub Domain		
Actual: No	TN = 660	FP = 4
Actual: Yes	FN = 1	TP = 962
Predictive Variables – Length, Length of Sub Domain, and Entropy		
Actual: No	TN = 660	FP = 4
Actual: Yes	FN = 4	TP = 959

Table 8.12 - Base Random Forest Evaluation Results

Predictors / Measures	**Length**	**Length & Length of Sub Domain**	**Length, Length of Sub Domain & Entropy**
ROC	0.99	0.99	0.99
PR	0.99	0.99	0.99
Sensitivity	98%	99%	99%
Specificity	96%	99%	99%
Accuracy	97%	99%	99%
Precision	97%	99%	99%

8.6 Tuned Random Forest Results

Table 8.13 - Tuned Random Forest Confusion Matrix

N = 1627	**Predicted: No**	**Predicted: Yes**
Predictive Variables - Length		
Actual: No	TN = 641	FP = 23
Actual: Yes	FN = 14	TP = 949
Predictive Variables – Length and Length of Sub Domain		
Actual: No	TN = 660	FP = 4
Actual: Yes	FN = 1	TP = 962
Predictive Variables – Length, Length of Sub Domain, and Entropy		
Actual: No	TN = 660	FP = 4
Actual: Yes	FN = 4	TP = 959

Table 8.14 - Tuned Random Forest Evaluation Results

Predictors / Measures	**Length**	**Length & Length of Sub Domain**	**Length, Length of Sub Domain & Entropy**
ROC	0.99	0.99	0.99
PR	0.99	0.99	0.99
Sensitivity	98%	99%	99%
Specificity	96%	99%	99%
Accuracy	97%	99%	99%
Precision	97%	99%	99%

8.7 SVM RBF Kernel Results

Table 8.15 - RBF Kernel SVM Confusion Matrix

N = 1627	**Predicted: No**	**Predicted: Yes**
Predictive Variables - Length		
Actual: No	TN = 651	FP = 13
Actual: Yes	FN = 5	TP = 958
Predictive Variables – Length and Length of Sub Domain		
Actual: No	TN = 660	FP = 4
Actual: Yes	FN = 5	TP = 958
Predictive Variables – Length, Length of Sub Domain, and Entropy		
Actual: No	TN = 660	FP = 4
Actual: Yes	FN = 5	TP = 958

Table 8.16 - RBF SVM Evaluation Results

Predictors / Measures	**Length**	**Length & Length of Sub Domain**	**Length, Length of Sub Domain & Entropy**
Sensitivity	99%	99%	99%
Specificity	98%	99%	99%
Accuracy	98%	99%	99%
Precision	98%	99%	99%

8.8 SVM Linear Kernel Results

Table 8.17 - Linear Kernel SVM Confusion Matrix

N = 1627	**Predicted: No**	**Predicted: Yes**
Predictive Variables - Length		
Actual: No	TN = 413	FP = 251
Actual: Yes	FN = 57	TP = 906
Predictive Variables – Length and Length of Sub Domain		
Actual: No	TN = 544	FP = 120
Actual: Yes	FN = 62	TP = 901
Predictive Variables – Length, Length of Sub Domain, and Entropy		
Actual: No	TN = 612	FP = 52
Actual: Yes	FN = 34	TP = 929

Table 8.18 - Linear SVM Evaluation Results

Predictors / Measures	**Length**	**Length & Length of Sub Domain**	**Length, Length of Sub Domain & Entropy**
Sensitivity	94%	93%	96%
Specificity	62%	81%	92%
Accuracy	81%	88%	94%
Precision	78%	88%	94%