Global Information Assurance Certification Paper

Malware Detection in Encrypted TLS Traffic Through Machine Learning

GIAC (GCIA) Gold Certification and ISE 5901

Author: Bryan Scarbrough, bryan.scarbrough@gmail.com

Advisor: *Dr. Johannes Ullrich*

Accepted: February 18, 2021

Abstract

The proliferation of TLS across the Internet leads to a safer environment for the end user but a more obscure setting for the network defender. This research demonstrates what can be learned using Machine Learning analysis of TLS traffic without decryption. It applies a novel approach to TLS analysis by analyzing data available in the unencrypted portion of the handshake combined with Open-source Intelligence (OSINT) data about Internet Protocol (IP) addresses and domain names. The metadata is then analyzed using three different machine learning algorithms: Support Vector Machine (SVM), One-Class SVM (OC-SVM), and an Autoencoder Neural Network. This research also addresses the imbalanced data distribution between malicious and benign traffic with the OC-SVM and the Autoencoder Neural Network. Finally, this research demonstrates that when using the correct header data the SVM and OC-SVM classify malware with a more than 99% F2 score and the Autoencoder approximately 95% F2.

1. Introduction

The use of Secure Sockets Layer/Transport Layer Security (SSL/TLS) is quickly on the rise across the Internet. According to the Google Transparency Report ("Google Transparency Report," n.d.) and Let's Encrypt statistics ("Let's Encrypt Stats," n.d.), in September of 2020, 95% of all web pages loaded by the Chrome browser and 92.2% of pages loaded by the Firefox browser in the United States used Hypertext Transfer Protocol Secure (HTTPS). Alongside the growth of secure web traffic is the increased use of enterprise cloud solutions such as Office365, Microsoft Azure, Amazon Web Services (AWS), user Virtual Private Network (VPN) connectivity, and the introduction of bring your own device (BYOD) to encrypted mobile application protocols. Along with the increase in regular SSL/TLS, malware's use of TLS encryption to hide in the noise of day-to-day operations has also increased.

Given TLS's effectiveness in protecting data and its widespread use for legitimate Internet-based services, it is no wonder that many malware authors use it to hide their activities. According to a security researcher at Sophos, in February of 2020, nearly a quarter of malware analyzed used TLS to conceal Command and Control (C2), installation, payload delivery, and even data exfiltration (Nagy, 2020). Also, according to a Zscaler report, "attacks involving the use of SSL/TLS encryption jumped 260% in the first nine months of 2020 compared to the same period last year" (Vijayan, 2020). Thus, detecting threats embedded within encrypted traffic has never been more critical. There are three primary options available to security analysts to identify malicious activity in encrypted traffic: manual packet analysis, inline decryption, and machine learning.

Manual packet analysis bears the cost of time, training, required expertise, and a high potential of overlooking anomalies. Gaining a complete understanding of a cybersecurity incident often requires manual packet and log analysis (Sikos, 2020). However, most forensic network analysts will spend years honing their skills through advanced training and daily inspection of network events. Even when using advanced visualization tools and analysis platforms, a seasoned professional can overlook data correlations that signal malicious intent, especially in encrypted traffic. Thus, a manual review of network traffic should not be a company's primary means of analysis and other methods considered.

Inline decryption raises many concerns around potential privacy and security issues for the analysis of encrypted network traffic. According to an NSA issued Cyber Advisory warning from November 2019, organizations have multiple concerns with inline decryption (NSA, 2019). Risks such as proxy device misconfiguration, certificate trust abuse, single point of failure, and breach of privacy regulations are among the list of potential problems (NSA, 2019). The organization should thoughtfully consider each risk and weigh it against the benefits of inline decryption and inspection. Companies should consider machine learning to help solve encrypted traffic analysis problems before adopting such invasive technology.

Finally, the statistical nature of network data and advances in computing make machine learning (ML) a viable option to solve problems like encrypted traffic classification. Omar Yaacoubi, CEO of Barac, stated that organizations do not need to rely on "traditional anti-malware scanners that can't support encrypted traffic, they can now utilize machine learning techniques that are able to inspect encrypted traffic without ever having to decrypt it" (Yaacoubi, 2020). A properly trained ML algorithm can provide insight into communication intent by measuring the correlation between various network events or analyzing network metadata. ML algorithms can identify network anomalies by analyzing metadata such as protocols used, packet density and size, or even communication direction (Anderson, Paul, & McGrew, 2016). To identify these anomalies, however, the algorithm must thoroughly understand the structure of the analyzed data.

2. TLS Protocol

2.1 Background

It is essential to begin a conversation about Transport Layer Security (TLS) analysis by understanding what it is and how it secures communications. The primary purpose of TLS "is to provide privacy and data integrity between two communicating applications" (Dierks & Allen, 1999). Initially introduced in 1999, TLS is the successor of the now deprecated Secure Sockets Layer (SSL) protocol and provides additional security and privacy over SSL without providing backward compatibility (Dierks & Allen, 1999). Today's primary use is web-based

communications to secure Hypertext Transfer Protocol (HTTP) traffic and form the HTTPS (Secure) version of the protocol.

2.2 TLS 1.2 Handshake

A TLSv1.2 session requires two round trips of messages between a client and a server to establish a secure tunnel and follows a well-defined process (Driscoll, n.d.). First, a client application such as a web browser sends a ClientHello message that specifies its supported TLS version, cryptographic algorithms (cipher suites), and any additional supported features. The server responds with a series of messages that perform functions such as selecting the cipher suite and TLS version, sending the server's certificate, and providing information necessary for encryption key generation. The client validates the provided certificate, sends its key generation data to the server, and provides a shared encryption key. Finally, the server responds with a ChangeCipherSpec message, after which all communication is encrypted, and the handshake completes (Figure 1).

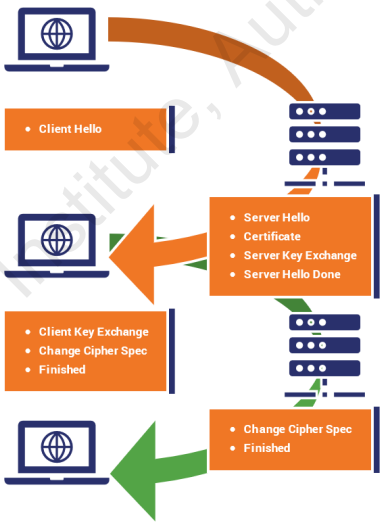

Figure 1: TLS 1.2 Handshake (Nohe, 2019)

2.3 TLS 1.3 Handshake

A TLSv1.3 handshake is a more streamlined, efficient, and secure version of the TLSv1.2 process. TLSv1.3 prioritizes speed and security by removing obsolete ciphers and hashing algorithms such as SHA1, MD5, and DES and reducing the handshake to only three messages.

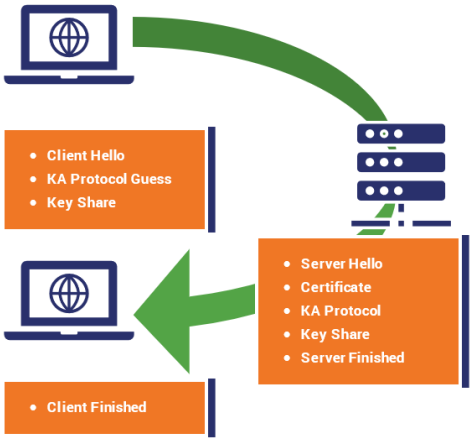

Figure 2: TLS 1.3 Handshake (Nohe, 2019)

The TLSv1.3 handshake begins just like TLSv1.2 with the same ClientHello message; however, this message indicates the desire to use TLSv1.3 to communicate, and the client provides a public key to the server. The server responds with a ServerHello message and includes a public key. The server's public key is used with the client's public key to derive a shared key that encrypts the rest of the handshake. Following the ServerHello message is a ChangeCipherSpec message that informs the client that all future handshake communications will be encrypted using the derived key. Finally, the server sends an encrypted wrapper containing the remainder of its handshake data. TLS is a proven and secure standard of data encryption and protection, which makes it a valuable tool for hiding malware communications.

2.4 Malware's use of TLS

As malware authors consider how to evade detection and protect their interests, well-established, proven, and versatile protocols such as TLS quickly become a valid option. A brief look at the top malware families of recent months confirms that assumption, with several using TLS for various operations. One report by Check Point Software Technologies lists the most prevalent malware for November of 2020, and nearly all of those listed use TLS ("November 2020's most wanted malware: Notorious Phorpiex Botnet returns as most impactful infection," 2020). Thus, it is incumbent upon the security analyst to determine the best method of network traffic analysis. Since TLS obscures both benign and malicious data equally, the introduction of machine learning provides a unique opportunity to analyze and classify traffic.

3. TLS Analysis Using Machine Learning

3.1 What is Machine Learning, and why use it for TLS Analysis?

Machine learning (ML) is becoming a ubiquitous part of our daily lives, yet many do not understand what it is or how it can help solve day-to-day problems. One definition of ML states, "Machine learning is a branch of artificial intelligence (AI) focused on building applications that learn from data and improve their accuracy over time without being programmed to do so" (IBM Cloud Education, 2020). In this definition, applications usually refer to computer algorithms designed around specific methods of statistical analysis. These algorithms examine large sets of data and attempt to predict outcomes based on the information received.

For a simple example, consider a recommendation list for a video streaming service. The ML algorithm, often called a model, reviews the movies or television shows a user indicates they like or prefer. These preferences are usually obtained directly through a question-and-answer session at account creation. They are then continually updated as the user watches shows and movies to help improve the algorithm's understanding of what the user likes and dislikes. Some platforms even allow a user to rate a movie or show and incorporates this rating into its predicted preference rank. The ML algorithm uses this data for training, making predictions about what the user will like. Based on these predictions, the algorithm presents similar content to the user. The understanding an ML algorithm derives from a user's activity is similar to that of understanding TLS handshakes, thus the abundance of existing research on this topic.

3.2 Existing Machine Learning and TLS Research

There is currently a significant amount of research surrounding the analysis of encrypted TLS traffic. This research does not intend to rehash all the information on this topic but intends to review some of the proposed ML-based TLS analysis approaches and discuss this research's differentiating factors. Many teams have analyzed encrypted data through ML, using both regression and classification to reveal underlying processes. Below are a few of those studies and a discussion of their outcomes.

One of the more prevalent papers, written by a team from Cisco concerning encrypted TLS analysis, is titled "Deciphering Malware's use of TLS (without Decryption)" (Anderson, Paul, & McGrew, 2016). This seminal work focused on differences between malware and benign

traffic using combinations of NetFlow and TLS handshake metadata. It evaluated both malware detection and malware family attribution without decrypting the data. Their analysis used four different feature sets: flow metadata, the sequence of packet lengths and times, byte distribution, and TLS header information. The researchers used an L1 Logistic Regression classifier model to ingest various combinations of this data to generate an overall 98.5%-99.6% malware classification rate[1] (see Figure 3 for classification statistics).

	All Data		No SChannel	
Dataset	Total Accuracy	0.01% FDR	Total Accuracy	0.01% FDR
Meta+SPLT+BD+TLS+SS	99.6%	92.6%	99.6%	87.4%
Meta+SPLT+BD+TLS	99.6%	92.8%	99.6%	87.2%
TLS	98.2%	63.8%	96.7%	59.1%
Meta+SPLT+BD	98.9%	1.3%	98.5%	0.9%

Figure 3: L1 Logistic regression classification rate listed by feature set (Anderson, Paul, & McGrew, 2016)

Expanding upon the research conducted by Blake and McGrew, another team sought to build a robust encrypted malware detection classifier. This team created a tool called MalDetect, which leverages a robust Online Random Forest classifier that is trained in online mode to avoid any retraining or redeployment when new samples are detected (Liu et al., 2019). This research yielded a platform capable of classifying multiple traffic types from legitimate to adware and malware; however, its false-negative rates are around 20% and 50% for adware and dynamic routing protocols, respectively (Liu et al., 2019). This research's use of a dynamically trainable model in the Online Random Forest introduced a novel means of analysis and focused on long-term usability and sustainability rather than merely hypothetical use cases.

Finally, a project sponsored by Lastline focused on TLS metadata to classify TLS flows (Roques, 2019). This study used five different models, Logistic Regression, Random Forest, K-Nearest neighbors, Linear Discriminant Analysis, and a Linear Support Vector Classifier, each with varying results. Overall, this study yielded a model with a 97.6% accuracy of malware classification, but that is not necessarily where this research shines. This research stands out for its side-by-side comparisons of malicious and benign TLS handshake metadata, directly influencing this paper's research and analysis. These feature comparisons yielded potentially significant differences in metadata characteristics and offered considerable insight into the

[1] It is important to note that the logistic regression classifier was trained with benign traffic and only tested using malicious traffic. Thus, the research did not figure false-positive rates into the overall classification rate (Anderson, Paul, & McGrew, 2016).

potential differences between malware and benign traffic data. This research, and others before it, also helped rule out various models for analysis as they have already proven either successful or unsuccessful.

Based on the previous research, many models demonstrated success using TLS handshake metadata; however, some, like the MalDetect research team, used outdated datasets, potentially leading to misrepresented results. There are many different ML models available to analyze data problems that may arise. This paper chose three models with varying degrees of success to classify TLS connections as malicious or benign. The models used in this research are the Support Vector Machine, One-Class (or single-class) Support Vector Machine, and an Autoencoder Neural Network. Each of these models offers a different approach to data analysis, and they provide a reasonable expectation of effectiveness for meeting this challenge.

3.3 Support Vector Machines (SVM)

The first model used in this research is the SVM. An SVM "is a generalization of a simple and intuitive classifier called the maximal margin classifier" (James, Witten, Hastie, & Tibshirani, 2013). The maximal margin classifier offers a binary classification of datasets by calculating a hyperplane (a line in two-dimensional space and a plane in multi-dimensional space) between the two data classifications. However, the potential problem of a simple hyperplane is determining exactly how to separate the data. As the image in Figure 4 shows, many different hyperplanes separate the two data classifications, but which one is the best one?

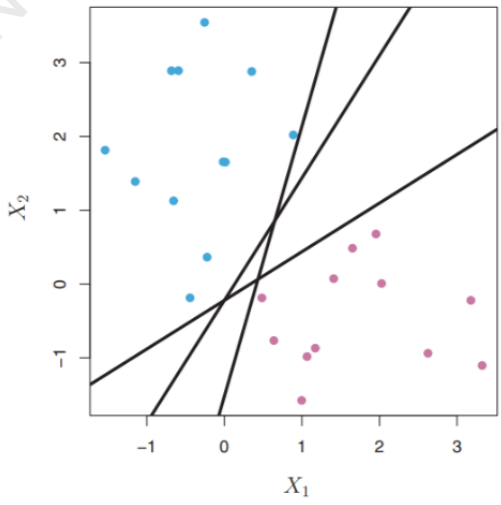

Figure 4: Hyperplanes separating a binary classified dataset (James, Witten, Hastie, & Tibshirani, 2013)

The maximal margin classifier classifies data by calculating the "separating hyperplane that is farthest from the training observations" (James, Witten, Hastie, & Tibshirani, 2013). The classifier calculates each observation's distance to a given hyperplane, and those with an equal distance determine the upper and lower limits of the boundary. The boundary lines work together with the hyperplane to create the margin, and the observations that support the margin are called the support vectors. Figure 5 shows what this margin looks like for a dataset. The solid line is the hyperplane separating data classifications, and the area between the dashed lines and the solid line represents the margin. The blue and orange observations that reside on the dashed line are all equal distance to the hyperplane, and they are the support vectors or the observations that support the maximal margin's size. Thus, this hyperplane with this margin best defines, separates, and accurately classifies each class of observations below.

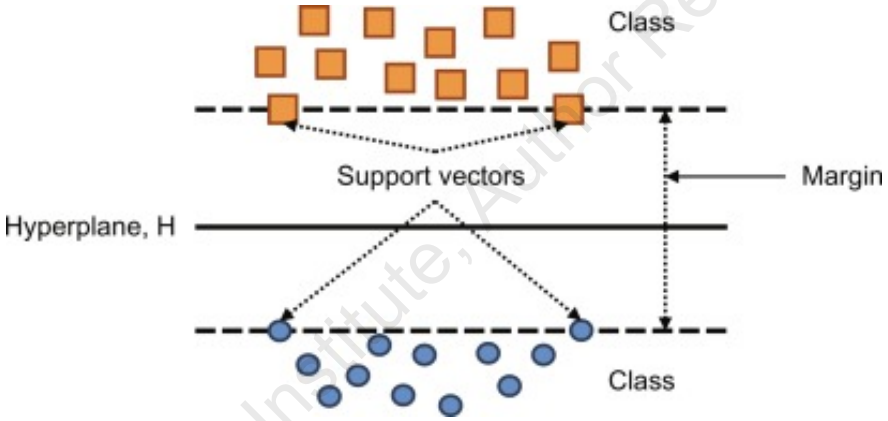

Figure 5: Hyperplane, margin, and support vectors (Thanki & Borra, 2019)

The maximal margin classifier works well with linearly separable data; however, analysis of the data used in this research (Figure 6, left image) revealed a non-linear relationship. To overcome this limitation, the SVM introduces what it calls kernels to support more complex data relationships. SVM kernels such as polynomial or radial bias function (RBF) provide non-linear flexibility for classifying datasets. The left image in Figure 6 represents the non-linear relationship of the dataset used in this research; however, the right picture in Figure 6 demonstrates the SVM's versatility using the RBF kernel to classify the data. The versatility of SVM kernels makes it a sure choice as an evaluation algorithm for this problem.

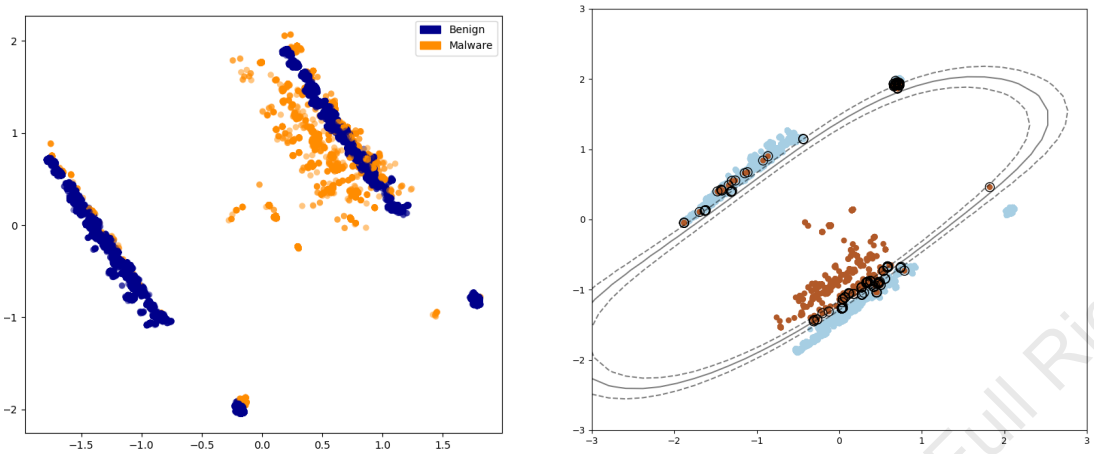

Figure 6: Component analysis of data used in this research showing the non-linear relationship (Left) and SVM with RBF function to overcome the non-linearity (Right)

3.4 One-Class Support Vector Machines (OC-SVM)

One of the intrinsic limitations of an SVM, and most ML models, is its requirement for a relatively balanced dataset. Balancing the dataset means that the ratio of positive (malware) to negative (benign) cases in the dataset must be nearly equal for the best results. Without this class equality, the SVM struggles to locate the maximal margin as represented in Figure 7. The SVM margin error happens because there are not enough positive samples compared to the large number of negative observations to calculate the hyperplane or maximal margin. Thus, classifiers such as the OC-SVM overcome this limitation.

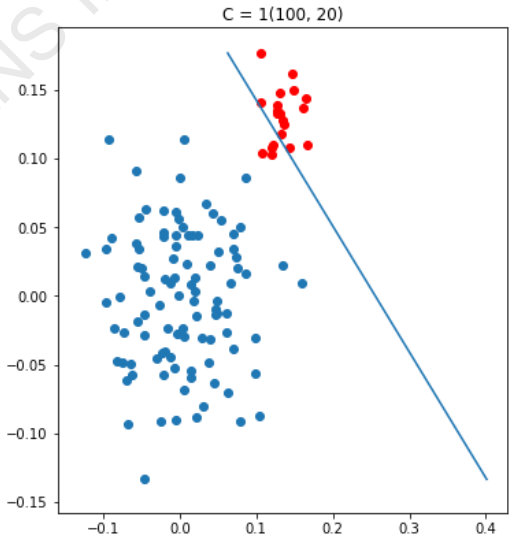

Figure 7: SVM Struggling to locate the hyperplane (Deepthi, 2019)

OC-SVMs capture the majority class's density and classify outliers as anything that resides beyond the majority class's extremes. That means that the model is "trained to learn what is 'normal' so that when new data is shown, the algorithm can identify whether it should belong to the group or not. If not, the new data is labeled as out of [the] ordinary or [an] anomaly" (Alam, 2020). Another term for this is outlier detection. For example, using a 1,000-sample dataset, the model is trained with 80% benign data (800 samples). Then, the remaining 20% sample composed of 99% benign (198 samples) and 1% malicious (2 samples) is used to test the model's classification capability. Using such a small proportion of the malware ensures the model detects benign data very effectively and that malware exercises no influence over SVM margin decisions or classification measurements.

Another benefit of the OC-SVM is that it is unsupervised, meaning training, or fitting, occurs without requiring pre-labeled data. Many ML algorithms, such as the SVM, require training data to be correctly labeled before processing. In this research, each record was annotated as benign or malicious and was used to train the SVM. The OC-SVM and Autoencoder models also used labels, but only to verify classification accuracy after training. Thus, for the OC-SVM, a dataset with around 98%-99% benign traffic was used to train the model. Once trained, the model processed additional datasets of similar malware ratios to verify efficacy.

3.5 Autoencoder Neural Network (Autoencoder)

The last ML model used for this research is the Autoencoder. An Autoencoder is an unsupervised model designed to reduce and encode data from many features down to a lower, compressed representation of those features. The "Autoencoder, by design, reduces data dimensions by learning how to ignore the noise in the data" (Badr, 2019). As the Autoencoder ignores the noise, it looks for unique correlations between various features and extracts those correlations for data reconstruction. While the Autoencoder creates a compressed representation of a dataset, it attempts to maintain enough variance to reconstruct the original data (Figure 8). Thus, "In the context of anomaly detection and condition monitoring, the basic idea is to use the autoencoder network to 'compress' the [data] to a lower-dimensional representation, which captures the correlations and interactions between the various variables" (Flovik, 2018). The Hidden Layer segments in Figure 8 represent the compressed and correlated data values.

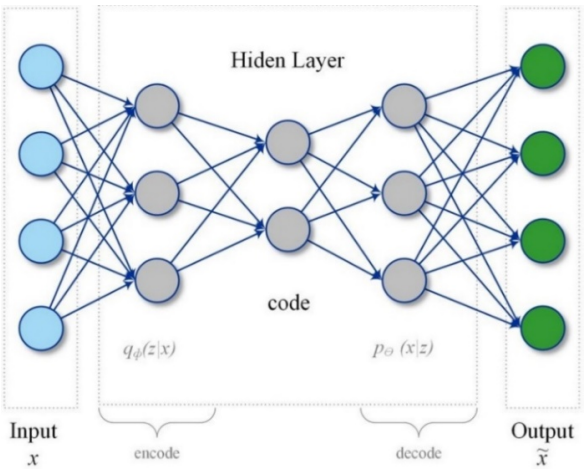

Figure 8: Autoencoder network (Flovik, 2018)

Just like the OC-SVM, the Autoencoder is an unsupervised model. The Autoencoder learns by reading large amounts of benign traffic and finding correlations between those data points. Once it understands the benign data, it detects anything that falls outside of that data standard and marks it as an anomaly or an outlier. Thus, it classifies data by finding correlations between its hidden layer components (Figure 8). It uses these correlations to reconstruct the data based on threshold calculations, then flags any values above the threshold as anomalous. Understanding how ML models work is a vital part of model selection and implementation, but just as vital is knowing how to measure a model's efficacy.

3.6 Measuring the Success of a Machine Learning Model

There are many measurement criteria available to evaluate a model's effectiveness, such as accuracy, precision, recall, and F-scores. Each formula involves calculations surrounding true-positive (TP), true-negative (TN), false-positive (FP), and false-negative (FN) predictions. A typical representation of these measurements is a diagram called a confusion matrix represented in Figure 9. TP and TN values are those where the ML model's predictions correctly align with the test data values. FP and FN measurements are those where the ML model's predictions incorrectly align with the data values.

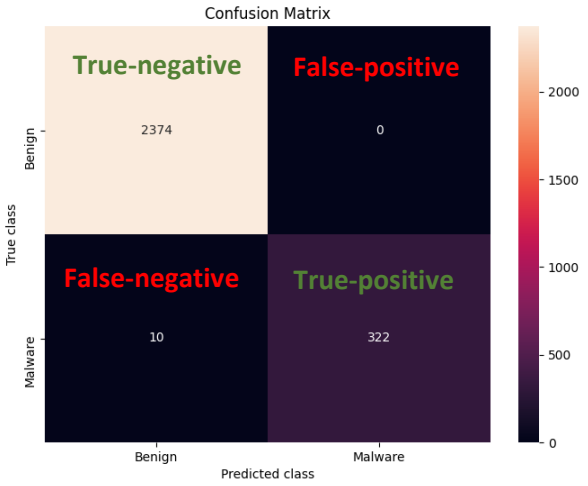

Figure 9: Confusion matrix diagram

Model accuracy is the percentage of correct measurements of a test dataset. Accuracy is calculated by dividing the number of correct predictions by the total number of predictions.

$$accuracy = \frac{\text{correct predictions}}{\text{all predictions}}$$

Precision measurements favor FP errors and are the proportion of positive values that are correct. They calculate this value by dividing the number of TP values by the sum of TP and FP values.

$$precision = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall favors FN errors and is the proportion of positive measurements out of all possible positive predictions. It calculates this proportion by dividing the number of TP by the sum of TP and FN.

$$recall = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Finally, the F score, also called F-measure or Fbeta-measure, computes a weighted average of the precision and recall values and is a better calculation of the incorrectly classified cases than accuracy. Three primary measurements are the F0.5, F1, and F2 scores. The F0.5 score applies more weight to the precision measurement, the F1 score provides balanced weight to both precision and recall, and the F2 score applies more weight to recall. This research chose

the F2 score for its emphasis on recall and the FN measurement, and the potential severity of an FN prediction of malicious traffic to an organization. The F2 score's formula is five times precision times recall divided by four times precision plus recall.

$$f2\ measure = \frac{5 * \text{precision} * \text{recall}}{4 * \text{precision} + \text{recall}}$$

There are many other formulas for calculating various ML model effectiveness values, but the accuracy, precision, recall, and F2 score are the primary measurements used in this research. Each value provides a different view of the model's performance and helps formulate a complete picture of efficacy. These measurements were used along with the mean output of a K-fold cross-validation process (Figure 10) to abide by industry best practices. After determining the appropriate measurement statistics and models for analysis, it is time to gather the data and format it appropriately.

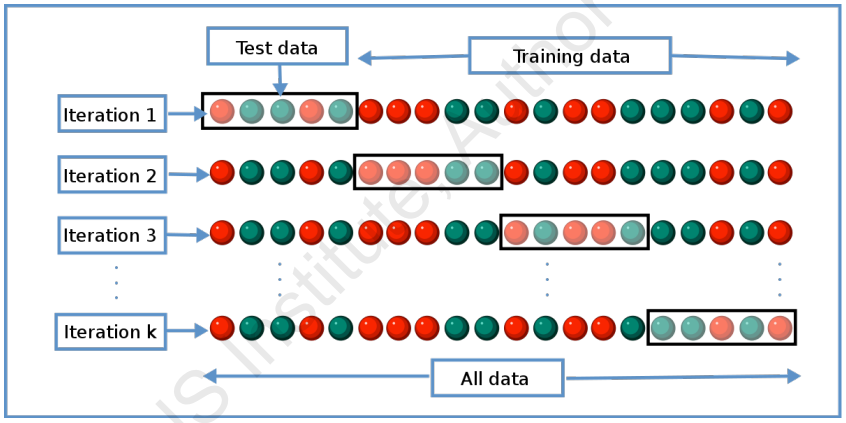

Figure 10: K-fold cross-validation process

4. Capturing Metadata for Analysis

4.1 Selecting a Dataset

This research chose to use a dataset provided by a group from the Canadian Institute for Cybersecurity (CIC) sponsored by the Canadian Internet Registration Authority for the discovery of malicious DNS over HTTPS (DoH) traffic (MontazeriShatoori, Davidson, Lashkari, & Habibi, 2020) as its benign dataset. The dataset generated by this group was created over several weeks in early 2020 and provides labeled datasets containing both malicious and benign network capture files. This research used four benign network capture files from the Google Chrome and

Mozilla Firefox datasets using Cloudflare and Google DNS for analysis[2]. A network metadata capture tool called NetCap (Mieden, 2018) processed each file and extracted TLS client and server handshakes. A custom Python program correlated the handshakes and wrote the resulting metadata to a comma-separated values (CSV) file for ML processing resulting in 110,490 benign samples.

The malicious network capture files were all obtained from the website malware-traffic-analysis.net (Duncan, n.d.). This site hosts hundreds of network capture files containing many different, relevant, and recent malware samples and their associated infection, payload delivery, and C2 traffic profiles. The dataset included 255 capture files across many malware families from Dridex, TrickBot, Emotet, Zbot/Zloader, IcedID, Quakbot, and more. The NetCap (Mieden, 2018) network analysis tool processed these files in the same manner as the benign samples and extracted the TLS handshake metadata. A custom Python program correlated the handshakes resulting in 6422 malicious samples. Figure 11 shows the relationship of benign to malicious data samples.

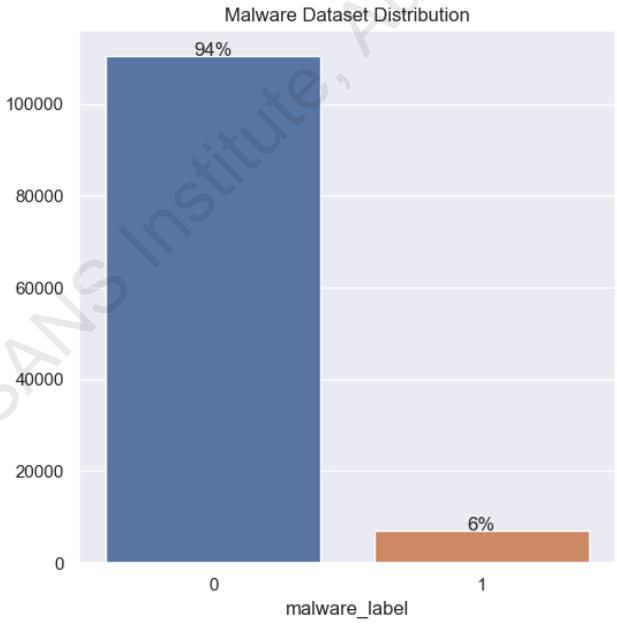

Figure 11: Malicious vs. benign data (1 is malware, 0 is benign). This diagram also represents the intentionally imbalanced dataset used to ensure model efficacy given a small sample of the positive case.

[2] There may be some inconsistencies in the benign data samples given their focus on DoH traffic flows; however, this research did not consider inconsistencies. Future research is necessary to determine if DoH skewed the analysis results.

4.2 TLS Handshake Metadata

Previous research of TLS using ML discovered a measurable variance between malicious and benign TLS handshakes (Roques, 2019). Thus, this research focuses only on the handshake metadata and a few OSINT resources without using packet metadata such as NetFlow or sequence of packet lengths and times to influence ML outcomes. The metadata obtained was available during the plaintext portion of the handshake, which led to the capture of 510 TLS features outlined in Table 1.

Feature	Size	Data Type
Source Port	1	Int
Destination Port	1	Int
TLS Record Type	1	Int
Client TLS Version	1	Int
Message Length	1	Int
Cipher Suite Length	1	Int
Cipher Suites	351	Float
Extension Length	1	Int
Handshake Type	1	Int
Handshake Length	1	Int
Handshake Version	1	Int
Signature Algorithms	36	Float
Supported Groups	49	Float
Supported Points	3	Int
Server OCSP Stapling	1	Int
Server TLS Version	1	Int
Server Supported TLS Version	1	Int
Server Extensions	59	Float
Total	510	

Table 1: TLS Handshake Metadata Features

4.3 Adding OSINT Metadata

Further processing incorporated multiple Open-source Intelligence (OSINT) sources to determine if the additional data offered any significant insight concerning intent. Five different OSINT resources provided understanding, and, in most cases, a "1" indicated the existence of a record, and a "0" denoted the record's absence. Additionally, record age provided precedence using the number of days between the entry's first and last reported timestamps.

Feature	Size	Data Type
DGA Intel	1	Float
AlienVault OTX	2	Float
URL Haus	2	Int
SSL Blacklist JA3	2	Int
Tranco	1	Float
Total	8	

Table 2: OSINT Metadata Features

The first OSINT tool used is a Python library called dgaintel (Mallarapu, 2019). This tool uses deep learning to determine whether a domain name is genuine or created by a domain generation algorithm (DGA) as a malicious C2 server. Dgaintel uses a Convolutional Neural Network with Long Short-Term Memory (CNN-LSTM) to calculate the degree of confidence that a human-generated the domain name. This tool's output is a floating-point number between "0", indicating a human-generated domain, and a "1" indicating a DGA domain.

Next, the domain name was compared against entries in the Tranco Top 1 Million domains list. The Tranco list is a security-focused list of the most popular websites across the Internet, similar to the Alexa Top 1 Million. It offers a pre-built Python lookup library and a combined list capability, offering a 30-day aggregated ranking of the most common domain lists (Le Pochat, Van Goethem, Tajalizadehkhoob, Korczynski, & Joosen, 2019). Thus, for this research, a comparison of domain names gathered from packet metadata and DNS lookups across 15 days provided insight into the domain's popularity and legitimacy. Each domain entry received a "1" for every day of appearance, and the script added the mean value to the connection metadata CSV file.

The next OSINT resource used was the AlienVault Open Threat Exchange (OTX) online database of malicious URLs ("AlienVault - Open Threat Exchange," n.d.). AlienVault OTX is a free, API driven, community-supported threat intelligence service where security analysts and organizations can contribute to the security community in multiple ways. Analysts can submit malware samples for analysis, add detected malicious IP addresses or domain names, or monitor systems for specific indicators of compromise. Organizations can also scan internal endpoints with the OTX Endpoint Scanner and perform individual IP or hostname lookups against the OTX system database. For this research, an IP or hostname matching a database entry received a "1" along with a count of the number of days between the first reported and last reported date.

Along with the AlienVault OTX resource were the two final OSINT resources available through abuse.ch. The first resource is called URL Haus, and its goal is "sharing malicious URLs that are being used for malware distribution" ("URLhaus," n.d.). This resource is like AlienVault in that it is a free, community-supported resource, but it contains a very different list of domains. Also, instead of trying to be a one-stop-shop like AlienVault, it specializes in malicious URL

detection. This research gathered the same data from URL Haus as AlienVault, the domain or IP's existence in the malware database, and its record age.

Finally, abuse.ch offers another service called the SSL Blacklist, a project whose goal is to identify and create a blocklist for SSL/TLS certificates used by botnets and C2 servers ("SSLBL," n.d.). The certificate blocklist is a list of SHA1 hashes of suspect TLS certificates gathered from across the Internet. However, due to TLS certificate caching, the certificate is only sometimes exchanged between clients and servers. Abuse.ch also offers a JA3 fingerprint database of botnet and C2 services leveraging TLS to overcome potential certificate analysis limitations. Thus, to ensure consistency across data samples, this research analyzed the JA3 values obtained during NetCap analysis against the abuse.ch database. Entries appearing in the database received a "1" along with a count of the number of days between the first reported and last reported date.

5. Feature Correlation and Influence

5.1 Top 10 Features

An essential first step in any ML project is understanding the dataset and any underlying correlations that may cause future problems or steer an analyst away from one methodology or toward another. TLS metadata analysis yielded many unique correlations that became apparent when graphing data relationships. The investigation began by comparing the 10 most important features of each classification. In Figure 12, the left image represents malware's 10 most important features, while the right image signifies benign traffic. There is minimal overlap between the two, proving the likelihood that this problem is solvable through ML. Table 3 maps the feature name used in this research to its associated TLS handshake value (refer to Appendices B-E for the full list of TLS features and the Internet Assigned Numbers Authority (IANA) to dataset name associations).

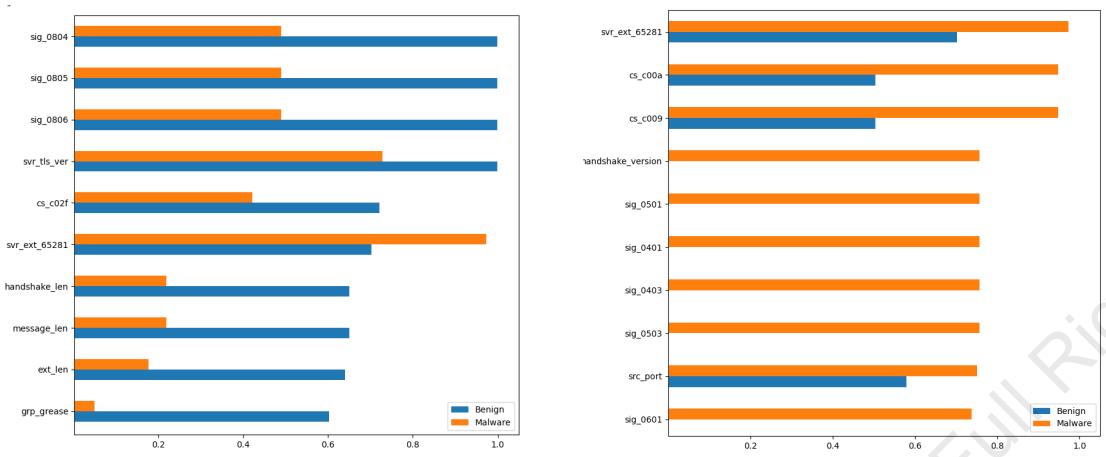

Figure 12: Comparison of the top 10 TLS handshake features by traffic class. The left image is sorted by benign importance, and the right image by malicious importance.

Feature Short Name	Actual Value	Traffic Class
sig_0804	rsa_pss_rsae_sha256	
sig_0805	rsa_pss_rsae_sha384	
sig_0806	rsa_pss_rsae_sha512	
ssl_tls_ver	TLS Client Version	
cs_c02f	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	Benign
svr_ext_65281	renegotiation_info	
handshake_len	TLS Handshake length	
message_len	TLS Client message length	
ext_len	Extension length (number of client extensions offered)	
grp_grease	Generate Random Extensions And Sustain Extensibility	
svr_ext_65281	renegotiation_info	
cs_c00a	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	
cs_c009	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	
handshake_version	TLS Client Handshake Version	
sig_0501	rsa_pkcs1_sha384	Malware
sig_0401	rsa_pkcs1_sha256	
sig_0403	ecdsa_secp256r1_sha256	
sig_0503	ecdsa_secp384r1_sha384	
src_port	TCP Source port	
sig_0601	rsa_pkcs1_sha512	

Table 3: Feature name to real name of top 10 features

5.2 Source and Destination Ports

Expanding the filter criteria beyond the top 10 features indicated significant differences in other areas, such as source and destination ports. The distribution of source and destination ports between the two data classes proved highly variable. While all the benign data samples had a zero percent variance in the destination port (TCP port 443), the malware samples demonstrated a 6.6% variance across the dataset (Figure 13). This variation was more considerable for source ports where the benign data had a 21.4% variance, and malware had a 51.8% differential (Figure 13). TCP port 443 was the most common source and destination port

for the malware samples. In contrast, benign traffic appears with evenly distributed source ports spread across the random high ephemeral range, typically above 30,000 (Figures 14 and 15). This activity is expected due to how TCP typically makes connections between random high ephemeral client ports and lower, IANA designated server ports.

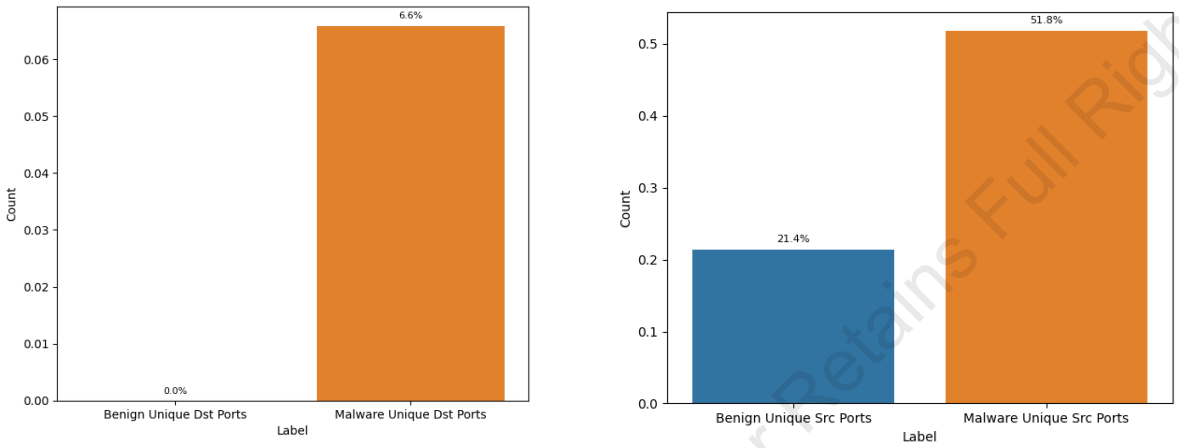

Figure 13: Source (right) and destination (left) port distribution across data classes

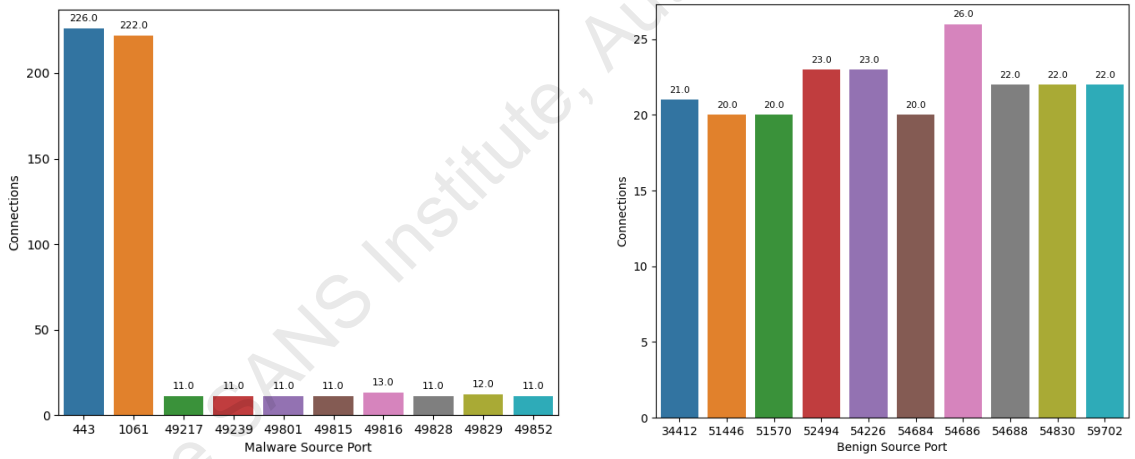

Figure 14: Top 10 malware source ports (left) and top 10 benign source ports (right)

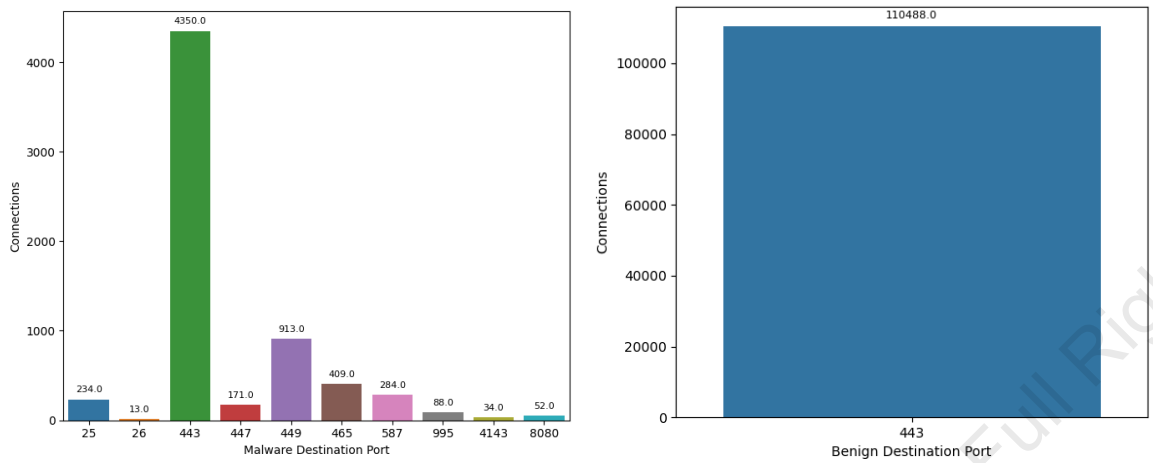

Figure 15: Top 10 malware destination ports (left), single benign destination port (right)

5.3 Cipher Suites and Signature Algorithms

Next, this research reviewed cipher suites and signature algorithms. As with ports, the first step was to examine the number of unique cipher suites and signature algorithms offered by the client. What stood out are the number of different values used by malware in both scenarios. Even though the malware has only a six percent data distribution in the dataset, it still presents seven more signature algorithms and 100 more cipher suites than the benign traffic (Figure 16). Figure 17 also shows that not only did malware use a wider variety of cipher suites and signatures, but it also preferred very different versions than those of benign processes.

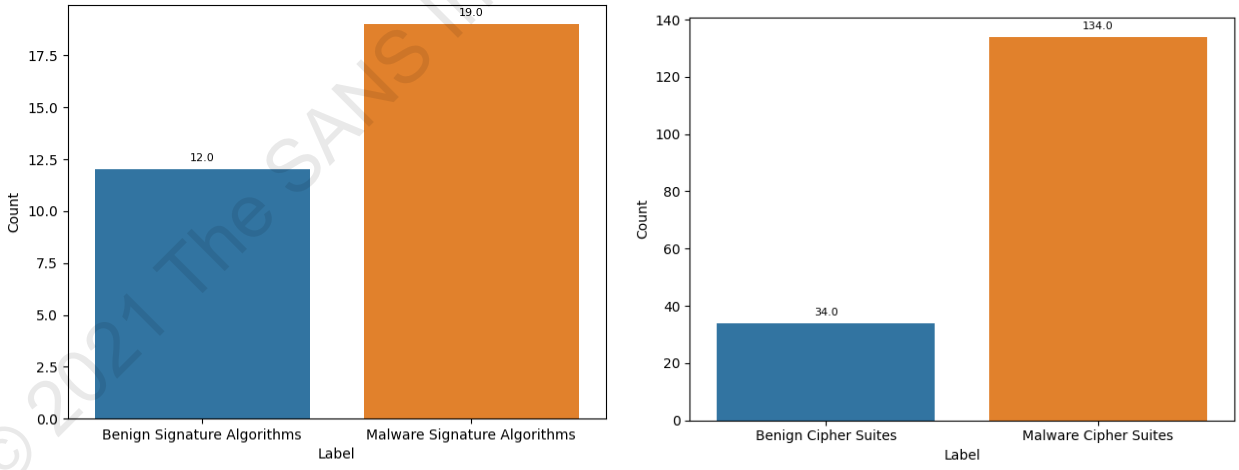

Figure 16: Cipher Suite (left) and Signature Algorithm (right) distributions represented by traffic class

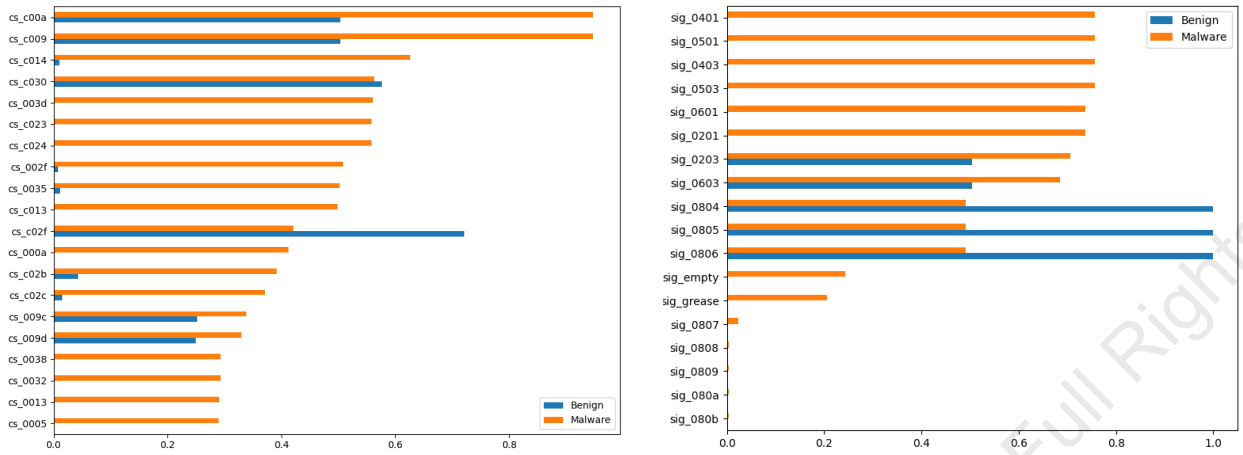

Figure 17: Most common cipher suites (left) and most common signature algorithms (right) sorted by malware preference

5.4 Metadata Sizes, Server Responses, and OSINT Data

Finally, the last groups of feature data represent areas where malicious actors have the least influence and are more difficult to modify than the previous fields. Up until this point, all the data features reviewed can be directly influenced by the malware author. They can easily change the source or destination ports, modify the client libraries used for development and completely change the distribution of cipher suites, signature algorithms, or extensions offered to align more readily with benign data samples. However, metadata fields such as the handshake length and message length are a bit more complicated. Server responses are more difficult to change since rewriting a C2 infrastructure may be required. Finally, the OSINT sources are community managed and maintained[3]. While malware authors may influence some of these features, many are often either standardized by available tooling or managed by groups beyond their reach. Thus, malware authors have a diminished ability to affect these values compared to the previous features.

Several unique relationships arise while analyzing the metadata sizes, server responses, and OSINT data. Firstly, the mean of the benign length values is significantly larger than that used by malware. Based on the initial analysis, it appears that while malware families use more

[3] Future analysis on this project might lead to applying additional weights to these features' measurements due to the malware author's inability to change or influence them. Using a weighted scale based on the difficulty of influence might improve the efficacy of the models.

cipher suites or extensions, the overall consistency of the benign clients presents itself with a much higher mean value (Figure 18).

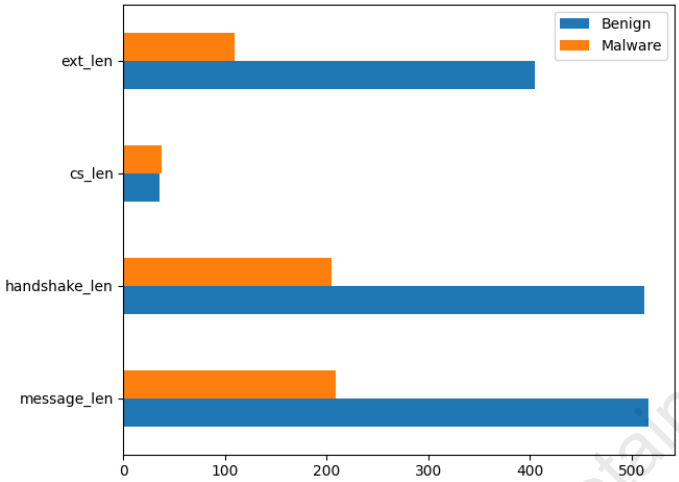

Figure 18: Mean of metadata length fields compared

Server response values appear evenly distributed, but they are spread across different values, as shown in Figure 19. According to measurements, only benign servers indicate a supported TLS version different than that used during the handshake. Other measurements stand out as being particularly important to benign servers, such as Online Certificate Status Protocol (OCSP) stapling, a protocol used to check certificate revocation status, and several server extensions such as key_share, connection_id, and application_layer_protocol_negotiation ("Transport Layer Security (TLS) Extensions," n.d.).

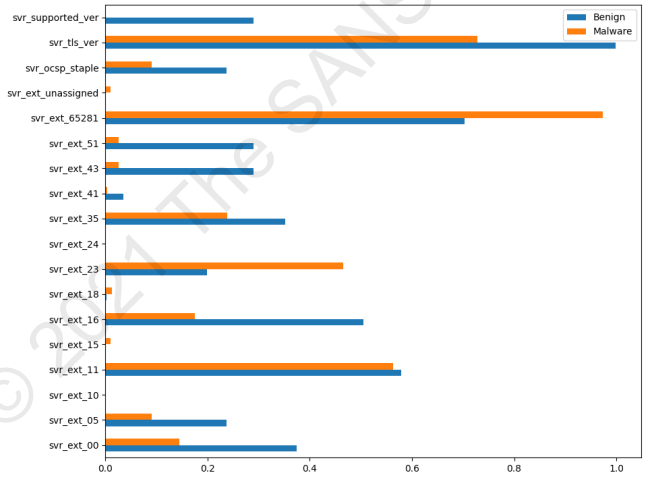

Figure 19: Server response relationship

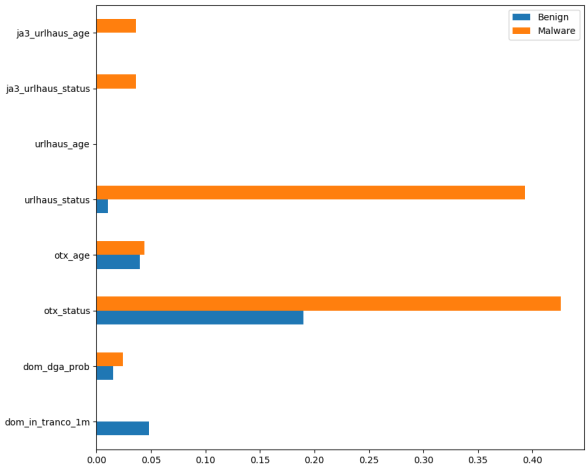

Figure 20: OSINT Data relationship

Finally, the OSINT metadata fields round out the features of this dataset. Figure 20 indicates that AlienVault OTX is the most effective in detecting malware samples, followed closely by URL Haus. However, while the OTX platform successfully detected many malware samples, it also detected many benign samples. This detection is due to malware's use of popular web services such as Dropbox, Google Drive, Twitter, etc., for C2. When this happens, the community reports those services to the OTX platform, and those domains become flagged even though they are quite common for normal, benign network traffic.

6. Machine Learning Results

Analysis of the previously described dataset consisted of multiple phases and iterations to determine the most accurate classification methodology. First, even though the SVM is efficient in processing highly dimensional datasets, this research began the analysis by evaluating multiple feature reduction techniques to increase model efficiency. Reduced feature datasets proved ineffective during prediction compared to the full dataset, and performance was not affected using the complete 518 feature dataset. Thus, this research abandoned feature reduction techniques early in experimentation. The results presented here are available at the link in Appendix A, and represent model fitting and prediction using all 518 features and various distribution sizes of the dataset as indicated in Tables 4-9.

6.1 SVM Analysis

The first model analyzed was the SVM. This model continually predicted outcomes with a mean recall of around 97.7%. The left image of Figure 19 shows the confusion matrix representing an accuracy score of approximately 99%, a recall of 96.9%, and a 100% precision score using a 25,000-sample training dataset. The right image of Figure 21 shows this model's efficacy using the full dataset after training with a randomly selected 25,000-sample batch. The right image in Figure 21 shows that the model effectively detected all but 68 malware instances out of the 6,422 samples present in the dataset, providing very high scores across the board, as demonstrated in Tables 4 and 5.

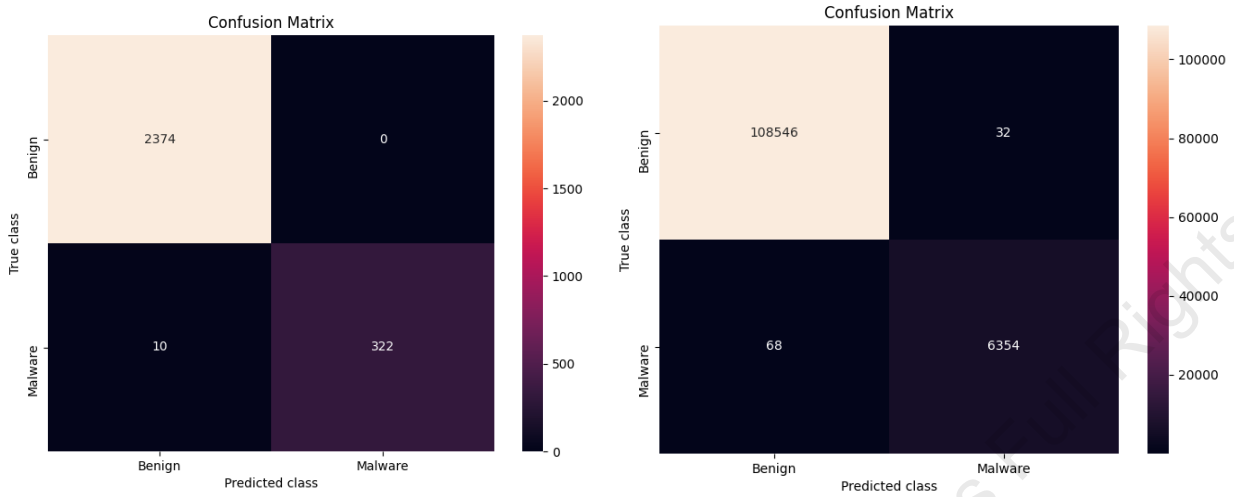

Figure 21: Confusion matrix of SVM early modeling (left) Confusion matrix using full validation dataset (right)

Further analysis of the SVM and cross-validation of settings yielded high scores for this model. The mean F2 score when trained with a 25,000-sample dataset with 20% malware distribution is 98.7%. Exporting this model and processing the full validation dataset yielded a 99.91% F2 score (Figure 21, right), meaning that of all the data samples in this dataset, incorrect classification occurred for approximately 100 samples.

Tables 4 and 5 represent the various training and validation scores across the SVM. Even though the 5,000-sample measurements are high, the data diversity proved too small to classify data samples across larger datasets. Once the training dataset reached 25,000 samples, the SVM required three times the data samples to increase accuracy by 0.001. Thus, the 25,000-sample dataset trained the model for all additional testing in Table 5. The SVM demonstrated measurable success when classifying the data within this dataset, and, even though the methodology is different, the SVM's implementation influenced the configuration of the OC-SVM.

Accuracy	Precision	Recall	F2 Score	Dataset Size in samples
0.9987	1.0	0.9791	0.9832	5000
0.9976	0.9914	0.9683	0.9728	10000
0.9989	0.9975	0.9847	0.9872	25000

Table 4: SVM training iterations using varying data distributions and a 20% Malware distribution

Accuracy	Precision	Recall	F2 Score	Dataset Size in samples
0.9974	0.9898	0.9659	0.9706	5000
0.9983	0.9937	0.9790	0.9819	10000
0.9989	0.9948	0.9876	0.9891	25000
0.9992	0.9979	0.9897	0.9991	Full Dataset

Table 5: Pre-trained SVM from 25000 sample size evaluating datasets of varying data size (approximately 6% Malware distribution)

6.2 OC-SVM Analysis

The OV-SVM was the most performant model evaluated. Its F2 score consistently measured above 99% across both training and evaluation datasets. The confusion matrices in Figure 22 and Tables 6-7 represent the OC-SVM classification capabilities and demonstrate its effectiveness and potential. The OC-SVM consistently measured a more than 99.5% F2 score across varying data sample sizes and an increasing F2 score as the training data size increased. The F2 score tapered after around 25,000 samples and required approximately 75,000 data samples to increase by an additional 0.001. Thus, the 25,000-sample dataset was used to train the model and evaluate larger dataset distributions. A much larger data sample size is necessary to determine if the OC-SVM's effectiveness might grow to an even higher F2 and if the overall model accuracy scales along with the dataset size. This model will benefit from an additional evaluation using a more diverse dataset to ensure overfitting has not occurred.

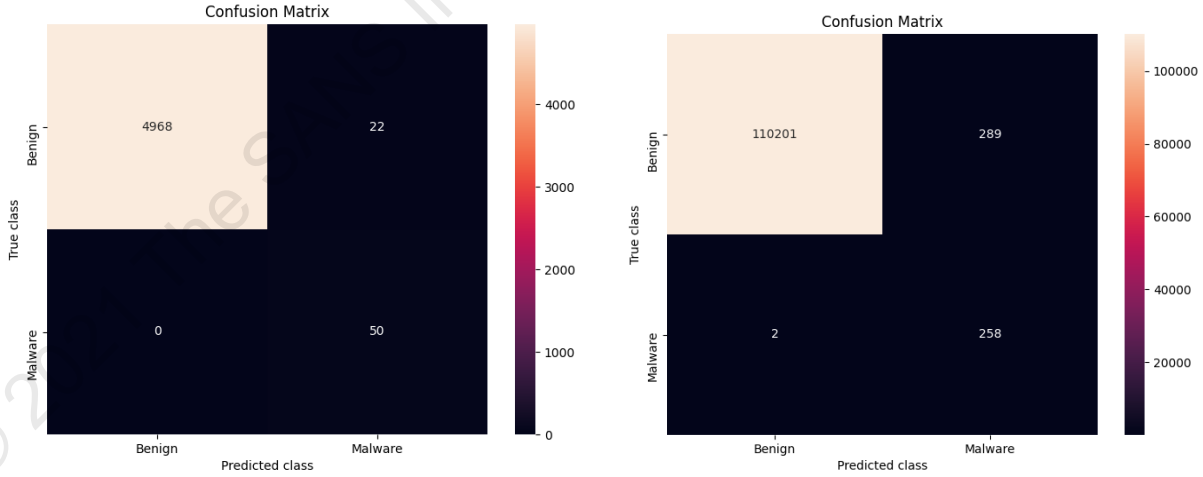

Figure 22: OC-SVM confusion matrix trained with 20,000-sample benign data and tested with 5,000-sample mixed data (left), OC-SVM showing previously trained model analyzing 115,000 data sample size (1% malware distribution)

Accuracy	Precision	Recall	F2 Score	Dataset Size in samples
0.9930	0.9989	0.9941	0.9950	5000
0.9910	0.9943	0.9966	0.9961	10000
0.9974	0.9996	0.9978	0.9981	25000
0.9987	0.9996	0.9990	0.9992	125000

Table 6: OC-SVM training iterations using varying data distributions (1% Malware distribution)

Accuracy	Precision	Recall	F2 Score	Dataset Size in samples
0.9966	0.9990	0.9976	0.9979	5000
0.9968	0.9990	0.9979	0.9981	10000
0.9974	0.9988	0.9986	0.9986	25000
0.9975	0.9986	0.9916	0.9933	Full Dataset

Table 7: Pre-trained OC-SVM from 25000 sample size evaluating datasets of varying size (1% Malware distribution)

6.3 Autoencoder Analysis

The final ML model used was the Autoencoder neural network. Due to the non-linear nature of the dataset (as previously demonstrated in Figure 6), the Autoencoder was a suitable fit for this type of model analysis. However, this model was not handled as a standard Autoencoder since the problem set is still to perform outlier detection. An article by Vegard Flovic concerning its use for anomaly detection and condition monitoring strongly influenced the Autoencoder implementation in this research (Flovik, 2018). The blog article presented a problem of predicting bearing failure in a factory and used only the "known good" data to train the Autoencoder. The model evaluates its accuracy and graphs a curve representing the error rate during training, also called the mean absolute error. The point where the model no longer learns the curve reaches zero, and this distance is used as a threshold to derive positive and negative samples (Figure 23).

Figure 23 shows the loss curve for the trained data model using a purely benign TLS dataset, then the mean absolute error is calculated against malware data from the same dataset. The point where the line stops decreasing (Figure 23, left image) is the point at which the Autoencoder has effectively learned or modeled the data. Since the only data it has modeled is benign, in theory, malicious measurements will be outside the standard loss curve's bounds and happen outside the trained data boundary. The distribution of the loss function graph (Figure 23,

right image) shows how many standard deviations it takes for the learning curve to drop from its peak to zero. The standard deviation measurement provides a "noise" threshold or boundary above which all anomalies should appear. Thus, using the loss distribution function in Figure 23 (right image), the data threshold is between 0.0125 and 0.0150.

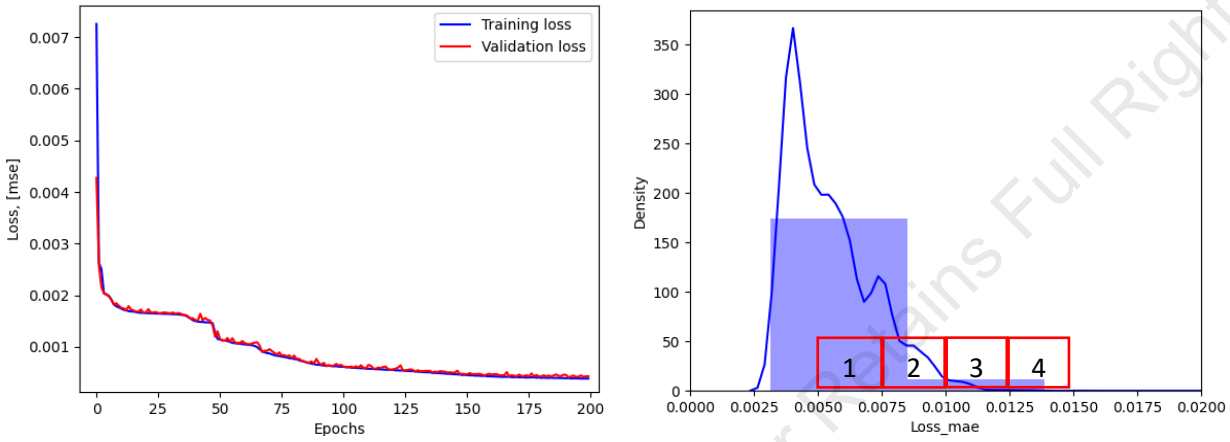

Figure 23: Loss curve of Autoencoder (left) with Loss distribution (right) annotating standard deviations to near 0 loss for threshold calculations

After calculating the threshold, the model processes malware cases, and measurements provide the selected boundary's efficacy. Using the limit from the curve in Figure 23, the chart in the left image of Figure 24 demonstrates the model's classification capability. The scatter plot in Figure 24 shows the model's classification capability using a 10,000-sample dataset. The anomalies become evident both by color and their presence above the blue threshold line. Benign data consistency is also apparent in this graph because it is a tightly clustered mass below the threshold. The confusion matrix below (the right image in Figure 24) demonstrates the same success level recognized in the threshold diagram and indicates five false positives from the dataset and 48 of 700, or approximately 7.4%, false negatives.

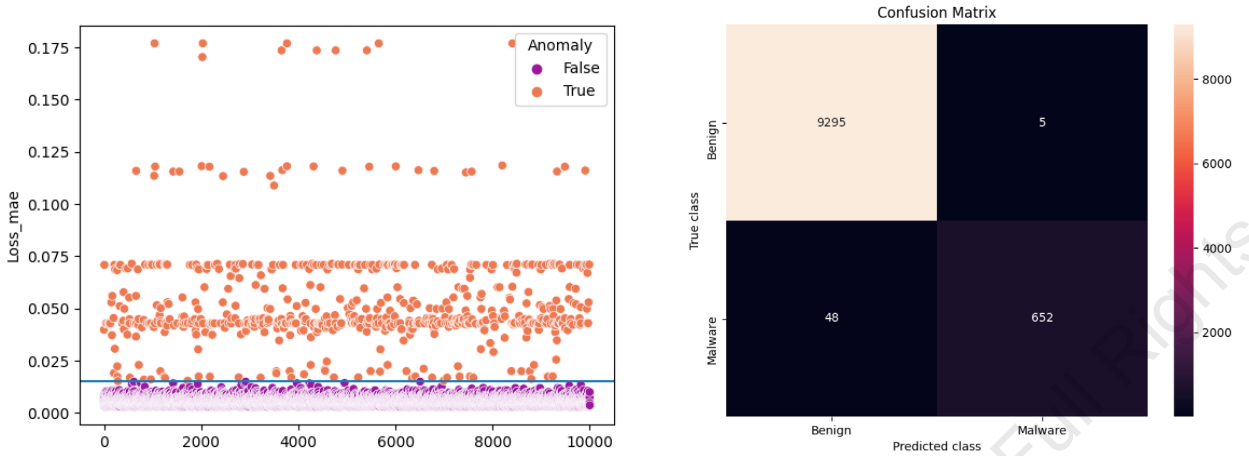

Figure 24: Autoencoder scatter plot (left), and confusion matrix of training sample (right)

A more extensive validation dataset was evaluated after training, containing varying data distributions as represented in Table 9. The scatter plot in Figure 25 (left image) describes the full dataset measurements after training with the 25,000-sample dataset. The highly clustered data below the threshold are benign, while data points spread above the boundary represent malware. The confusion matrix demonstrates approximately the same ratios as the smaller dataset in Figure 25 with a 4.7% false-negative and 4.4% false-positive rate.

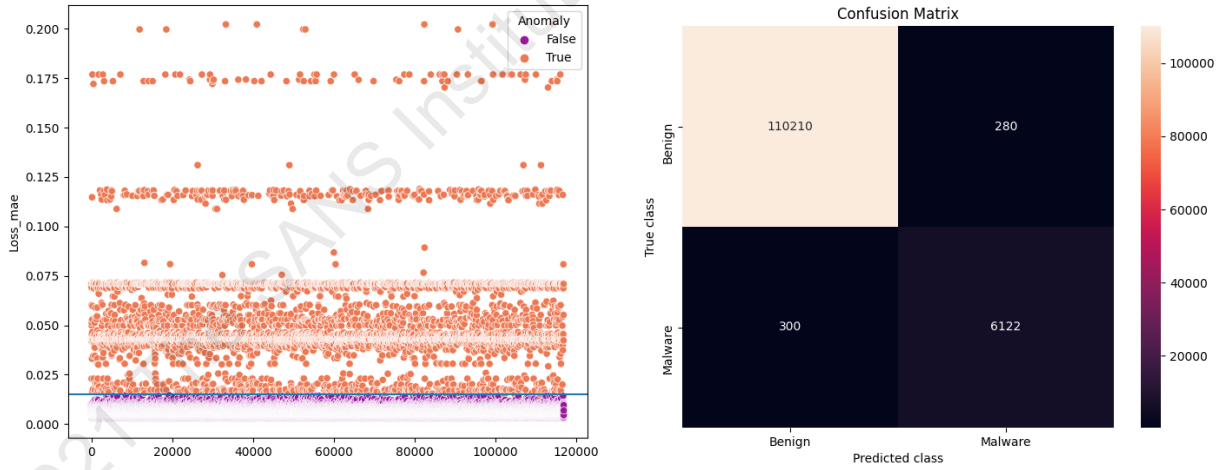

Figure 25: Pre-trained Autoencoder evaluating large validation dataset

Accuracy	Precision	Recall	F2 Score	Dataset Size in samples
0.05	0.05	1.0	0.2083	5000
0.5731	0.1040	0.99	0.3662	10000
0.9919	0.9906	0.8472	0.8725	25000

Table 8: Random data sample sizes used to train the model using a 5% malware ratio

Accuracy	Precision	Recall	F2 Score	Dataset Size in samples
0.9964	0.9640	0.9640	0.9640	5000
0.9967	0.9642	0.97	0.9688	10000
0.9952	0.9528	0.952	0.9521	25000
0.9948	0.9532	0.9501	0.9507	Full Dataset

Table 9: Random data sample sizes evaluated against model trained with the 25,000-sample training set

7. Future Research and Conclusions

The proliferation of TLS and other encrypted protocols is likely to continue to increase over the coming years. As such, the security community must work together to solve analysis and malicious intent detection of encrypted traffic. ML grants a step in that direction, and research such as this offers a means to promulgate that objective. This research analyzed three different ML models, each with very different results.

Overall, the OC-SVM model performed the best, demonstrating a more than 99.5% accuracy of malware classification given the dataset used in this research. Once the OC-SVM was better understood, adequately trained, and evaluated correctly, the model's efficacy was consistently over 99%. After baselining, the OC-SVM can immediately detect malicious activity, and the analyst can fine-tune the model to match any environmental differences as they arise. Additionally, pre-training may suit this model and make drop-in implementation an option since it ignores site-specific data such as source IP. More research is necessary to ensure that these results are accurate by gathering more robust datasets, processing a more extensive diversity of data samples, and processing live enterprise network traffic, and using the trained model to classify live connections.

The next performant model was the SVM. It performed almost as well as the OC-SVM, making it a strong contender to solve the TLS analysis problem using a dataset like the one in this research. Demonstrating a consistent 0.98-0.99 F2 score bodes well for the SVM; however, it falls short of the OC-SVM in its recall scores, measuring between 0.96-0.98. Such small variations may not seem significant, but it is vital to bear in mind that it only takes a single instance for a threat actor to be successful and bring an entire network down. A four percent false-negative ratio over 100,000 TLS samples equates to 4,000 malicious connections, and all are marked as benign by the ML model. Measurements such as this can lead to a lack of

confidence in the model's efficacy. Like the OC-SVM, this model requires additional research using more diverse datasets and processing live network traffic to verify the measured results.

Finally, the Autoencoder performed well even though it did not appear as though it would during initial training. This model struggled with some false negatives, measuring around 300 against the full 117,000-sample dataset, giving it a lower mean recall score (approximately 0.95) than anticipated. This Autoencoder may not be suited for a problem such as TLS analysis; however, additional research can still be conducted to ensure all configuration options and analysis techniques are addressed across a more extensive and diverse dataset. Each of the ML models reviewed as a part of this research deserves additional consideration to determine their feasibility over the long term and usefulness to other statistical analysis forms.

There is still much work remaining to solve the problem of malware detection in encrypted TLS traffic. There remain significant opportunities for statistical analysis of data in this vein to understand what is happening on the network. For example, future research could include NetFlow for a more holistic view of the handshake process. Additional OSINT resources could round out analysis, and visualization tools can provide visual correlation context for an event. A more extensive dataset needs to be used with all three models to train against a greater diversity of TLS clients and servers before this research can make any definitive statements concerning these models' effectiveness. Despite this, however, they all show promise for dealing with the imbalanced dataset issue and potentially solving the concern of classifying malicious data in encrypted traffic.

References

Alam, M. (2020, October 17). Support Vector Machine (SVM) for anomaly detection. Retrieved January 6, 2021, from Towards Data Science website: https://towardsdatascience.com/support-vector-machine-svm-for-anomaly-detection-73a8d676c331

AlienVault - Open Threat Exchange. (n.d.). Retrieved January 4, 2021, from Alienvault.com website: https://otx.alienvault.com/

Anderson, B. (2020, February 5). Overcoming the limitations of TLS fingerprinting for malware detection - Blake Anderson. Retrieved January 2, 2021, from https://vimeo.com/389483062

Anderson, B., & McGrew, D. (2017). Machine learning for encrypted malware traffic classification: Accounting for noisy labels and non-stationarity. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM.

Anderson, B., Paul, S., & McGrew, D. (2016). Deciphering malware's use of TLS (without decryption). Retrieved from http://arxiv.org/abs/1607.01639

Badr, W. (2019, April 22). Auto-encoder: What is it? And what is it used for? (part 1). Retrieved January 5, 2021, from Towards Data Science website: https://towardsdatascience.com/auto-encoder-what-is-it-and-what-is-it-used-for-part-1-3e5c6f017726

Can I use... Support tables for HTML5, CSS3, etc. (n.d.). Retrieved December 29, 2020, from Caniuse.com website: https://caniuse.com/tls1-3

Deepthi, A. R. (2019, December 18). Support vector machines & imbalanced data - towards data science. Retrieved January 5, 2021, from Towards Data Science website: https://towardsdatascience.com/support-vector-machines-imbalanced-data-feb3ecffbb0e

Dierks, T., & Allen, C. (1999). *The TLS Protocol Version 1.0*. Retrieved from https://tools.ietf.org/html/rfc2246

Driscoll, M. (n.d.). The illustrated TLS connection. Retrieved December 5, 2020, from Ulfheim.net website: https://tls.ulfheim.net/

Duncan, B. (2019, December 23). Malware-Traffic-Analysis.net - 2019-12-23 - Rig EK sends malware payload I cannot identify. Retrieved December 31, 2020, from Malware-traffic-analysis.net website: https://www.malware-traffic-analysis.net/2019/12/23/index3.html

Duncan, B. (n.d.). Malware-Traffic-Analysis.net. Retrieved January 20, 2021, from Malware-traffic-analysis.net website: https://www.malware-traffic-analysis.net/

Flovik, V. (2018, December 31). How to use machine learning for anomaly detection and condition monitoring. Retrieved January 5, 2021, from Towards Data Science website: https://towardsdatascience.com/how-to-use-machine-learning-for-anomaly-detection-and-condition-monitoring-6742f82900d7

Google Transparency Report. (n.d.). Retrieved September 25, 2020, from Google.com website: https://transparencyreport.google.com/https/overview?hl=en

IBM Cloud Education. (2020, July 15). What is machine learning? Retrieved January 5, 2021, from Ibm.com website: https://www.ibm.com/cloud/learn/machine-learning

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning*. New York, NY: Springer New York.

Le Pochat, V., Van Goethem, T., Tajalizadehkhoob, S., Korczynski, M., & Joosen, W. (2019). Tranco: A research-oriented top sites ranking hardened against manipulation. *Proceedings 2019 Network and Distributed System Security Symposium*. Reston, VA: Internet Society.

Let's Encrypt Stats. (n.d.). Retrieved September 25, 2020, from Letsencrypt.org website: https://letsencrypt.org/stats/

Liu, J., Zeng, Y., Shi, J., Yang, Y., Wang, R., & He, L. (2019). MalDetect: A structure of encrypted malware traffic detection. *Computers, Materials & Continua*, *60*(2), 721–739.

Mallarapu, R. (2019, November 21). dgaintel. Retrieved January 3, 2021, from Pypi.org website: https://pypi.org/project/dgaintel/

Mercury. (2019). Retrieved from https://github.com/cisco/mercury

Mieden, P. (2018, December 18). Implementation and evaluation of secure and scalable anomaly-based network intrusion detection. Retrieved January 3, 2021, from Netcap.io website: https://www.researchgate.net/publication/329815346_Implementation_and_evaluation_of_secure_and_scalable_anomaly-based_network_intrusion_detection

MontazeriShatoori, M., Davidson, L., Lashkari, G. K., & Habibi, A. (2020). *Detection of DoH Tunnels using Time-series Classification of Encrypted Traffic*. Calgary, Canada: The 5th IEEE Cyber Science and Technology Congress.

Nagy, L. (2020, February 18). Nearly a quarter of malware now communicates using TLS. Retrieved December 30, 2020, from Sophos.com website: https://news.sophos.com/en-us/2020/02/18/nearly-a-quarter-of-malware-now-communicates-using-tls/

Nohe, P. (2019, July 16). TLS 1.3 Update: Everything you need to know. Retrieved December 29, 2020, from Thesslstore.com website: https://www.thesslstore.com/blog/tls-1-3-everything-possibly-needed-know/

November 2020's most wanted malware: Notorious Phorpiex Botnet returns as most impactful infection. (2020, December 9). Retrieved December 31, 2020, from Checkpoint.com website: https://blog.checkpoint.com/2020/12/09/november-2020s-most-wanted-malware-notorious-phorpiex-botnet-returns-as-most-impactful-infection/

NSA. (2019, November 19). NSA releases Cyber Advisory: Managing risk from Transport Layer Security Inspection. Retrieved February 2, 2021, from Cisa.gov website: https://us-cert.cisa.gov/ncas/current-activity/2019/11/19/nsa-releases-cyber-advisory-managing-risk-transport-layer-security

Rescorla, E., Oku, K., Sullivan, N., & Wood, C. (2020, December 16). draft-ietf-tls-esni-09 - TLS Encrypted Client Hello. Retrieved December 30, 2020, from Ietf.org website: https://datatracker.ietf.org/doc/draft-ietf-tls-esni/?include_text=1

Roques, O. (2019, September). Detecting Malware in TLS Traffic. The IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05). doi:10.1109/lcn.2005.35

Sikos, L. F. (2020). Packet analysis for network forensics: A comprehensive survey. Forensic Science International: Digital Investigation, 32(200892), 200892.

SSL Pulse. (n.d.). Retrieved December 29, 2020, from Ssllabs.com website: https://www.ssllabs.com/ssl-pulse/

SSLBL. (n.d.). Retrieved January 4, 2021, from Abuse.ch website: https://sslbl.abuse.ch/

Thanki, R., & Borra, S. (2019). Application of machine learning algorithms for classification and security of diagnostic images. In N. Dey, S. Borra, A. S. Ashour, & F. Shi (Eds.), Machine Learning in Bio-Signal Analysis and Diagnostic Imaging (pp. 273–292). San Diego, CA: Elsevier.

Tranco. (n.d.). Retrieved January 3, 2021, from Tranco-list.eu website: https://tranco-list.eu/

Transport Layer Security (TLS) Extensions. (n.d.). Retrieved January 6, 2021, from Iana.org website: https://www.iana.org/assignments/tls-extensiontype-values/tls-extensiontype-values.xhtml

URLhaus. (n.d.). Retrieved January 4, 2021, from Abuse.ch website: https://urlhaus.abuse.ch/

Vijayan, J. (2020, November 10). Malware hidden in encrypted traffic surges amid pandemic. Retrieved December 30, 2020, from Dark Reading website: https://www.darkreading.com/attacks-breaches/malware-hidden-in-encrypted-traffic-surges-amid-pandemic/d/d-id/1339420

Yaacoubi, O. (2020, January 17). Is this the beginning of the end for transport layer security inspection techniques? Retrieved February 2, 2021, from Infosecurity-magazine.com website: https://www.infosecurity-magazine.com/opinions/transport-layer-inspection/

Appendix A
Project Source Code and Help Docs

To access the source code used for this project, refer to the GitHub repository below. The code is documented, and the repository contains installation instructions. If you have any problems, feel free to raise an issue on GitHub.

https://github.com/1computerguy/tls-mal-detect

Appendix B
TLS Cipher Suites List

Dataset Value	IANA Value	Description
cs_0000	0x00,0x00	TLS_NULL_WITH_NULL_NULL
cs_0001	0x00,0x01	TLS_RSA_WITH_NULL_MD5
cs_0002	0x00,0x02	TLS_RSA_WITH_NULL_SHA
cs_0003	0x00,0x03	TLS_RSA_EXPORT_WITH_RC4_40_MD5
cs_0004	0x00,0x04	TLS_RSA_WITH_RC4_128_MD5
cs_0005	0x00,0x05	TLS_RSA_WITH_RC4_128_SHA
cs_0006	0x00,0x06	TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5
cs_0007	0x00,0x07	TLS_RSA_WITH_IDEA_CBC_SHA
cs_0008	0x00,0x08	TLS_RSA_EXPORT_WITH_DES40_CBC_SHA
cs_0009	0x00,0x09	TLS_RSA_WITH_DES_CBC_SHA
cs_000a	0x00,0x0A	TLS_RSA_WITH_3DES_EDE_CBC_SHA
cs_000b	0x00,0x0B	TLS_DH_DSS_EXPORT_WITH_DES40_CBC_SHA
cs_000c	0x00,0x0C	TLS_DH_DSS_WITH_DES_CBC_SHA
cs_000d	0x00,0x0D	TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA
cs_000e	0x00,0x0E	TLS_DH_RSA_EXPORT_WITH_DES40_CBC_SHA
cs_000f	0x00,0x0F	TLS_DH_RSA_WITH_DES_CBC_SHA
cs_0010	0x00,0x10	TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA
cs_0011	0x00,0x11	TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
cs_0012	0x00,0x12	TLS_DHE_DSS_WITH_DES_CBC_SHA
cs_0013	0x00,0x13	TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
cs_0014	0x00,0x14	TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
cs_0015	0x00,0x15	TLS_DHE_RSA_WITH_DES_CBC_SHA
cs_0016	0x00,0x16	TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
cs_0017	0x00,0x17	TLS_DH_anon_EXPORT_WITH_RC4_40_MD5
cs_0018	0x00,0x18	TLS_DH_anon_WITH_RC4_128_MD5
cs_0019	0x00,0x19	TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA
cs_001a	0x00,0x1A	TLS_DH_anon_WITH_DES_CBC_SHA
cs_001b	0x00,0x1B	TLS_DH_anon_WITH_3DES_EDE_CBC_SHA
cs_001e	0x00,0x1E	TLS_KRB5_WITH_DES_CBC_SHA
cs_001f	0x00,0x1F	TLS_KRB5_WITH_3DES_EDE_CBC_SHA
cs_0020	0x00,0x20	TLS_KRB5_WITH_RC4_128_SHA
cs_0021	0x00,0x21	TLS_KRB5_WITH_IDEA_CBC_SHA
cs_0022	0x00,0x22	TLS_KRB5_WITH_DES_CBC_MD5
cs_0023	0x00,0x23	TLS_KRB5_WITH_3DES_EDE_CBC_MD5
cs_0024	0x00,0x24	TLS_KRB5_WITH_RC4_128_MD5
cs_0025	0x00,0x25	TLS_KRB5_WITH_IDEA_CBC_MD5
cs_0026	0x00,0x26	TLS_KRB5_EXPORT_WITH_DES_CBC_40_SHA
cs_0027	0x00,0x27	TLS_KRB5_EXPORT_WITH_RC2_CBC_40_SHA
cs_0028	0x00,0x28	TLS_KRB5_EXPORT_WITH_RC4_40_SHA
cs_0029	0x00,0x29	TLS_KRB5_EXPORT_WITH_DES_CBC_40_MD5

cs_002a	0x00,0x2A	TLS_KRB5_EXPORT_WITH_RC2_CBC_40_MD5
cs_002b	0x00,0x2B	TLS_KRB5_EXPORT_WITH_RC4_40_MD5
cs_002c	0x00,0x2C	TLS_PSK_WITH_NULL_SHA
cs_002d	0x00,0x2D	TLS_DHE_PSK_WITH_NULL_SHA
cs_002e	0x00,0x2E	TLS_RSA_PSK_WITH_NULL_SHA
cs_002f	0x00,0x2F	TLS_RSA_WITH_AES_128_CBC_SHA
cs_0030	0x00,0x30	TLS_DH_DSS_WITH_AES_128_CBC_SHA
cs_0031	0x00,0x31	TLS_DH_RSA_WITH_AES_128_CBC_SHA
cs_0032	0x00,0x32	TLS_DHE_DSS_WITH_AES_128_CBC_SHA
cs_0033	0x00,0x33	TLS_DHE_RSA_WITH_AES_128_CBC_SHA
cs_0034	0x00,0x34	TLS_DH_anon_WITH_AES_128_CBC_SHA
cs_0035	0x00,0x35	TLS_RSA_WITH_AES_256_CBC_SHA
cs_0036	0x00,0x36	TLS_DH_DSS_WITH_AES_256_CBC_SHA
cs_0037	0x00,0x37	TLS_DH_RSA_WITH_AES_256_CBC_SHA
cs_0038	0x00,0x38	TLS_DHE_DSS_WITH_AES_256_CBC_SHA
cs_0039	0x00,0x39	TLS_DHE_RSA_WITH_AES_256_CBC_SHA
cs_003a	0x00,0x3A	TLS_DH_anon_WITH_AES_256_CBC_SHA
cs_003b	0x00,0x3B	TLS_RSA_WITH_NULL_SHA256
cs_003c	0x00,0x3C	TLS_RSA_WITH_AES_128_CBC_SHA256
cs_003d	0x00,0x3D	TLS_RSA_WITH_AES_256_CBC_SHA256
cs_003e	0x00,0x3E	TLS_DH_DSS_WITH_AES_128_CBC_SHA256
cs_003f	0x00,0x3F	TLS_DH_RSA_WITH_AES_128_CBC_SHA256
cs_0040	0x00,0x40	TLS_DHE_DSS_WITH_AES_128_CBC_SHA256
cs_0041	0x00,0x41	TLS_RSA_WITH_CAMELLIA_128_CBC_SHA
cs_0042	0x00,0x42	TLS_DH_DSS_WITH_CAMELLIA_128_CBC_SHA
cs_0043	0x00,0x43	TLS_DH_RSA_WITH_CAMELLIA_128_CBC_SHA
cs_0044	0x00,0x44	TLS_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA
cs_0045	0x00,0x45	TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA
cs_0046	0x00,0x46	TLS_DH_anon_WITH_CAMELLIA_128_CBC_SHA
cs_0067	0x00,0x67	TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
cs_0068	0x00,0x68	TLS_DH_DSS_WITH_AES_256_CBC_SHA256
cs_0069	0x00,0x69	TLS_DH_RSA_WITH_AES_256_CBC_SHA256
cs_006a	0x00,0x6A	TLS_DHE_DSS_WITH_AES_256_CBC_SHA256
cs_006b	0x00,0x6B	TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
cs_006c	0x00,0x6C	TLS_DH_anon_WITH_AES_128_CBC_SHA256
cs_006d	0x00,0x6D	TLS_DH_anon_WITH_AES_256_CBC_SHA256
cs_0084	0x00,0x84	TLS_RSA_WITH_CAMELLIA_256_CBC_SHA
cs_0085	0x00,0x85	TLS_DH_DSS_WITH_CAMELLIA_256_CBC_SHA
cs_0086	0x00,0x86	TLS_DH_RSA_WITH_CAMELLIA_256_CBC_SHA
cs_0087	0x00,0x87	TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA
cs_0088	0x00,0x88	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA
cs_0089	0x00,0x89	TLS_DH_anon_WITH_CAMELLIA_256_CBC_SHA
cs_008a	0x00,0x8A	TLS_PSK_WITH_RC4_128_SHA

cs_008b	0x00,0x8B	TLS_PSK_WITH_3DES_EDE_CBC_SHA
cs_008c	0x00,0x8C	TLS_PSK_WITH_AES_128_CBC_SHA
cs_008d	0x00,0x8D	TLS_PSK_WITH_AES_256_CBC_SHA
cs_008e	0x00,0x8E	TLS_DHE_PSK_WITH_RC4_128_SHA
cs_008f	0x00,0x8F	TLS_DHE_PSK_WITH_3DES_EDE_CBC_SHA
cs_0090	0x00,0x90	TLS_DHE_PSK_WITH_AES_128_CBC_SHA
cs_0091	0x00,0x91	TLS_DHE_PSK_WITH_AES_256_CBC_SHA
cs_0092	0x00,0x92	TLS_RSA_PSK_WITH_RC4_128_SHA
cs_0093	0x00,0x93	TLS_RSA_PSK_WITH_3DES_EDE_CBC_SHA
cs_0094	0x00,0x94	TLS_RSA_PSK_WITH_AES_128_CBC_SHA
cs_0095	0x00,0x95	TLS_RSA_PSK_WITH_AES_256_CBC_SHA
cs_0096	0x00,0x96	TLS_RSA_WITH_SEED_CBC_SHA
cs_0097	0x00,0x97	TLS_DH_DSS_WITH_SEED_CBC_SHA
cs_0098	0x00,0x98	TLS_DH_RSA_WITH_SEED_CBC_SHA
cs_0099	0x00,0x99	TLS_DHE_DSS_WITH_SEED_CBC_SHA
cs_009a	0x00,0x9A	TLS_DHE_RSA_WITH_SEED_CBC_SHA
cs_009b	0x00,0x9B	TLS_DH_anon_WITH_SEED_CBC_SHA
cs_009c	0x00,0x9C	TLS_RSA_WITH_AES_128_GCM_SHA256
cs_009d	0x00,0x9D	TLS_RSA_WITH_AES_256_GCM_SHA384
cs_009e	0x00,0x9E	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
cs_009f	0x00,0x9F	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
cs_00a0	0x00,0xA0	TLS_DH_RSA_WITH_AES_128_GCM_SHA256
cs_00a1	0x00,0xA1	TLS_DH_RSA_WITH_AES_256_GCM_SHA384
cs_00a2	0x00,0xA2	TLS_DHE_DSS_WITH_AES_128_GCM_SHA256
cs_00a3	0x00,0xA3	TLS_DHE_DSS_WITH_AES_256_GCM_SHA384
cs_00a4	0x00,0xA4	TLS_DH_DSS_WITH_AES_128_GCM_SHA256
cs_00a5	0x00,0xA5	TLS_DH_DSS_WITH_AES_256_GCM_SHA384
cs_00a6	0x00,0xA6	TLS_DH_anon_WITH_AES_128_GCM_SHA256
cs_00a7	0x00,0xA7	TLS_DH_anon_WITH_AES_256_GCM_SHA384
cs_00a8	0x00,0xA8	TLS_PSK_WITH_AES_128_GCM_SHA256
cs_00a9	0x00,0xA9	TLS_PSK_WITH_AES_256_GCM_SHA384
cs_00aa	0x00,0xAA	TLS_DHE_PSK_WITH_AES_128_GCM_SHA256
cs_00ab	0x00,0xAB	TLS_DHE_PSK_WITH_AES_256_GCM_SHA384
cs_00ac	0x00,0xAC	TLS_RSA_PSK_WITH_AES_128_GCM_SHA256
cs_00ad	0x00,0xAD	TLS_RSA_PSK_WITH_AES_256_GCM_SHA384
cs_00ae	0x00,0xAE	TLS_PSK_WITH_AES_128_CBC_SHA256
cs_00af	0x00,0xAF	TLS_PSK_WITH_AES_256_CBC_SHA384
cs_00b0	0x00,0xB0	TLS_PSK_WITH_NULL_SHA256
cs_00b1	0x00,0xB1	TLS_PSK_WITH_NULL_SHA384
cs_00b2	0x00,0xB2	TLS_DHE_PSK_WITH_AES_128_CBC_SHA256
cs_00b3	0x00,0xB3	TLS_DHE_PSK_WITH_AES_256_CBC_SHA384
cs_00b4	0x00,0xB4	TLS_DHE_PSK_WITH_NULL_SHA256
cs_00b5	0x00,0xB5	TLS_DHE_PSK_WITH_NULL_SHA384

cs_00b6	0x00,0xB6	TLS_RSA_PSK_WITH_AES_128_CBC_SHA256
cs_00b7	0x00,0xB7	TLS_RSA_PSK_WITH_AES_256_CBC_SHA384
cs_00b8	0x00,0xB8	TLS_RSA_PSK_WITH_NULL_SHA256
cs_00b9	0x00,0xB9	TLS_RSA_PSK_WITH_NULL_SHA384
cs_00ba	0x00,0xBA	TLS_RSA_WITH_CAMELLIA_128_CBC_SHA256
cs_00bb	0x00,0xBB	TLS_DH_DSS_WITH_CAMELLIA_128_CBC_SHA256
cs_00bc	0x00,0xBC	TLS_DH_RSA_WITH_CAMELLIA_128_CBC_SHA256
cs_00bd	0x00,0xBD	TLS_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA256
cs_00be	0x00,0xBE	TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA256
cs_00bf	0x00,0xBF	TLS_DH_anon_WITH_CAMELLIA_128_CBC_SHA256
cs_00c0	0x00,0xC0	TLS_RSA_WITH_CAMELLIA_256_CBC_SHA256
cs_00c1	0x00,0xC1	TLS_DH_DSS_WITH_CAMELLIA_256_CBC_SHA256
cs_00c2	0x00,0xC2	TLS_DH_RSA_WITH_CAMELLIA_256_CBC_SHA256
cs_00c3	0x00,0xC3	TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA256
cs_00c4	0x00,0xC4	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA256
cs_00c5	0x00,0xC5	TLS_DH_anon_WITH_CAMELLIA_256_CBC_SHA256
cs_00c6	0x00,0xC6	TLS_SM4_GCM_SM3
cs_00c7	0x00,0xC7	TLS_SM4_CCM_SM3
cs_00ff	0x00,0xFF	TLS_EMPTY_RENEGOTIATION_INFO_SCSV
cs_1301	0x13,0x01	TLS_AES_128_GCM_SHA256
cs_1302	0x13,0x02	TLS_AES_256_GCM_SHA384
cs_1303	0x13,0x03	TLS_CHACHA20_POLY1305_SHA256
cs_1304	0x13,0x04	TLS_AES_128_CCM_SHA256
cs_1305	0x13,0x05	TLS_AES_128_CCM_8_SHA256
cs_5600	0x56,0x00	TLS_FALLBACK_SCSV
cs_c001	0xC0,0x01	TLS_ECDH_ECDSA_WITH_NULL_SHA
cs_c002	0xC0,0x02	TLS_ECDH_ECDSA_WITH_RC4_128_SHA
cs_c003	0xC0,0x03	TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA
cs_c004	0xC0,0x04	TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA
cs_c005	0xC0,0x05	TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA
cs_c006	0xC0,0x06	TLS_ECDHE_ECDSA_WITH_NULL_SHA
cs_c007	0xC0,0x07	TLS_ECDHE_ECDSA_WITH_RC4_128_SHA
cs_c008	0xC0,0x08	TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
cs_c008	0xC0,0x09	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
cs_c00a	0xC0,0x0A	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
cs_c00b	0xC0,0x0B	TLS_ECDH_RSA_WITH_NULL_SHA
cs_c00c	0xC0,0x0C	TLS_ECDH_RSA_WITH_RC4_128_SHA
cs_c00d	0xC0,0x0D	TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA
cs_c00e	0xC0,0x0E	TLS_ECDH_RSA_WITH_AES_128_CBC_SHA
cs_c00f	0xC0,0x0F	TLS_ECDH_RSA_WITH_AES_256_CBC_SHA
cs_c010	0xC0,0x10	TLS_ECDHE_RSA_WITH_NULL_SHA
cs_c011	0xC0,0x11	TLS_ECDHE_RSA_WITH_RC4_128_SHA
cs_c012	0xC0,0x12	TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA

cs_c013	0xC0,0x13	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
cs_c014	0xC0,0x14	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
cs_c015	0xC0,0x15	TLS_ECDH_anon_WITH_NULL_SHA
cs_c016	0xC0,0x16	TLS_ECDH_anon_WITH_RC4_128_SHA
cs_c017	0xC0,0x17	TLS_ECDH_anon_WITH_3DES_EDE_CBC_SHA
cs_c018	0xC0,0x18	TLS_ECDH_anon_WITH_AES_128_CBC_SHA
cs_c019	0xC0,0x19	TLS_ECDH_anon_WITH_AES_256_CBC_SHA
cs_c01a	0xC0,0x1A	TLS_SRP_SHA_WITH_3DES_EDE_CBC_SHA
cs_c01b	0xC0,0x1B	TLS_SRP_SHA_RSA_WITH_3DES_EDE_CBC_SHA
cs_c01c	0xC0,0x1C	TLS_SRP_SHA_DSS_WITH_3DES_EDE_CBC_SHA
cs_c01d	0xC0,0x1D	TLS_SRP_SHA_WITH_AES_128_CBC_SHA
cs_c01e	0xC0,0x1E	TLS_SRP_SHA_RSA_WITH_AES_128_CBC_SHA
cs_c01f	0xC0,0x1F	TLS_SRP_SHA_DSS_WITH_AES_128_CBC_SHA
cs_c020	0xC0,0x20	TLS_SRP_SHA_WITH_AES_256_CBC_SHA
cs_c021	0xC0,0x21	TLS_SRP_SHA_RSA_WITH_AES_256_CBC_SHA
cs_c022	0xC0,0x22	TLS_SRP_SHA_DSS_WITH_AES_256_CBC_SHA
cs_c023	0xC0,0x23	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
cs_c024	0xC0,0x24	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
cs_c025	0xC0,0x25	TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256
cs_c026	0xC0,0x26	TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384
cs_c027	0xC0,0x27	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
cs_c028	0xC0,0x28	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
cs_c029	0xC0,0x29	TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256
cs_c02a	0xC0,0x2A	TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384
cs_c02b	0xC0,0x2B	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
cs_c02c	0xC0,0x2C	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
cs_c02d	0xC0,0x2D	TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256
cs_c02e	0xC0,0x2E	TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384
cs_c02f	0xC0,0x2F	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
cs_c030	0xC0,0x30	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
cs_c031	0xC0,0x31	TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256
cs_c032	0xC0,0x32	TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384
cs_c033	0xC0,0x33	TLS_ECDHE_PSK_WITH_RC4_128_SHA
cs_c034	0xC0,0x34	TLS_ECDHE_PSK_WITH_3DES_EDE_CBC_SHA
cs_c035	0xC0,0x35	TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA
cs_c036	0xC0,0x36	TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA
cs_c037	0xC0,0x37	TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256
cs_c038	0xC0,0x38	TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA384
cs_c039	0xC0,0x39	TLS_ECDHE_PSK_WITH_NULL_SHA
cs_c03a	0xC0,0x3A	TLS_ECDHE_PSK_WITH_NULL_SHA256
cs_c03b	0xC0,0x3B	TLS_ECDHE_PSK_WITH_NULL_SHA384
cs_c03c	0xC0,0x3C	TLS_RSA_WITH_ARIA_128_CBC_SHA256
cs_c03d	0xC0,0x3D	TLS_RSA_WITH_ARIA_256_CBC_SHA384

cs_c03e	0xC0,0x3E	TLS_DH_DSS_WITH_ARIA_128_CBC_SHA256
cs_c03f	0xC0,0x3F	TLS_DH_DSS_WITH_ARIA_256_CBC_SHA384
cs_c040	0xC0,0x40	TLS_DH_RSA_WITH_ARIA_128_CBC_SHA256
cs_c041	0xC0,0x41	TLS_DH_RSA_WITH_ARIA_256_CBC_SHA384
cs_c042	0xC0,0x42	TLS_DHE_DSS_WITH_ARIA_128_CBC_SHA256
cs_c043	0xC0,0x43	TLS_DHE_DSS_WITH_ARIA_256_CBC_SHA384
cs_c044	0xC0,0x44	TLS_DHE_RSA_WITH_ARIA_128_CBC_SHA256
cs_c045	0xC0,0x45	TLS_DHE_RSA_WITH_ARIA_256_CBC_SHA384
cs_c046	0xC0,0x46	TLS_DH_anon_WITH_ARIA_128_CBC_SHA256
cs_c047	0xC0,0x47	TLS_DH_anon_WITH_ARIA_256_CBC_SHA384
cs_c048	0xC0,0x48	TLS_ECDHE_ECDSA_WITH_ARIA_128_CBC_SHA256
cs_c049	0xC0,0x49	TLS_ECDHE_ECDSA_WITH_ARIA_256_CBC_SHA384
cs_c04a	0xC0,0x4A	TLS_ECDH_ECDSA_WITH_ARIA_128_CBC_SHA256
cs_c04b	0xC0,0x4B	TLS_ECDH_ECDSA_WITH_ARIA_256_CBC_SHA384
cs_c04c	0xC0,0x4C	TLS_ECDHE_RSA_WITH_ARIA_128_CBC_SHA256
cs_c04d	0xC0,0x4D	TLS_ECDHE_RSA_WITH_ARIA_256_CBC_SHA384
cs_c04e	0xC0,0x4E	TLS_ECDH_RSA_WITH_ARIA_128_CBC_SHA256
cs_c04f	0xC0,0x4F	TLS_ECDH_RSA_WITH_ARIA_256_CBC_SHA384
cs_c050	0xC0,0x50	TLS_RSA_WITH_ARIA_128_GCM_SHA256
cs_c051	0xC0,0x51	TLS_RSA_WITH_ARIA_256_GCM_SHA384
cs_c052	0xC0,0x52	TLS_DHE_RSA_WITH_ARIA_128_GCM_SHA256
cs_c053	0xC0,0x53	TLS_DHE_RSA_WITH_ARIA_256_GCM_SHA384
cs_c054	0xC0,0x54	TLS_DH_RSA_WITH_ARIA_128_GCM_SHA256
cs_c055	0xC0,0x55	TLS_DH_RSA_WITH_ARIA_256_GCM_SHA384
cs_c056	0xC0,0x56	TLS_DHE_DSS_WITH_ARIA_128_GCM_SHA256
cs_c057	0xC0,0x57	TLS_DHE_DSS_WITH_ARIA_256_GCM_SHA384
cs_c058	0xC0,0x58	TLS_DH_DSS_WITH_ARIA_128_GCM_SHA256
cs_c059	0xC0,0x59	TLS_DH_DSS_WITH_ARIA_256_GCM_SHA384
cs_c05a	0xC0,0x5A	TLS_DH_anon_WITH_ARIA_128_GCM_SHA256
cs_c05b	0xC0,0x5B	TLS_DH_anon_WITH_ARIA_256_GCM_SHA384
cs_c05c	0xC0,0x5C	TLS_ECDHE_ECDSA_WITH_ARIA_128_GCM_SHA256
cs_c05d	0xC0,0x5D	TLS_ECDHE_ECDSA_WITH_ARIA_256_GCM_SHA384
cs_c05e	0xC0,0x5E	TLS_ECDH_ECDSA_WITH_ARIA_128_GCM_SHA256
cs_c05f	0xC0,0x5F	TLS_ECDH_ECDSA_WITH_ARIA_256_GCM_SHA384
cs_c060	0xC0,0x60	TLS_ECDHE_RSA_WITH_ARIA_128_GCM_SHA256
cs_c061	0xC0,0x61	TLS_ECDHE_RSA_WITH_ARIA_256_GCM_SHA384
cs_c062	0xC0,0x62	TLS_ECDH_RSA_WITH_ARIA_128_GCM_SHA256
cs_c063	0xC0,0x63	TLS_ECDH_RSA_WITH_ARIA_256_GCM_SHA384
cs_c064	0xC0,0x64	TLS_PSK_WITH_ARIA_128_CBC_SHA256
cs_c065	0xC0,0x65	TLS_PSK_WITH_ARIA_256_CBC_SHA384
cs_c066	0xC0,0x66	TLS_DHE_PSK_WITH_ARIA_128_CBC_SHA256
cs_c067	0xC0,0x67	TLS_DHE_PSK_WITH_ARIA_256_CBC_SHA384
cs_c068	0xC0,0x68	TLS_RSA_PSK_WITH_ARIA_128_CBC_SHA256

cs_c068	0xC0,0x69	TLS_RSA_PSK_WITH_ARIA_256_CBC_SHA384
cs_c06a	0xC0,0x6A	TLS_PSK_WITH_ARIA_128_GCM_SHA256
cs_c06b	0xC0,0x6B	TLS_PSK_WITH_ARIA_256_GCM_SHA384
cs_c06c	0xC0,0x6C	TLS_DHE_PSK_WITH_ARIA_128_GCM_SHA256
cs_c06d	0xC0,0x6D	TLS_DHE_PSK_WITH_ARIA_256_GCM_SHA384
cs_c06e	0xC0,0x6E	TLS_RSA_PSK_WITH_ARIA_128_GCM_SHA256
cs_c06f	0xC0,0x6F	TLS_RSA_PSK_WITH_ARIA_256_GCM_SHA384
cs_c070	0xC0,0x70	TLS_ECDHE_PSK_WITH_ARIA_128_CBC_SHA256
cs_c071	0xC0,0x71	TLS_ECDHE_PSK_WITH_ARIA_256_CBC_SHA384
cs_c072	0xC0,0x72	TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_CBC_SHA256
cs_c073	0xC0,0x73	TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_CBC_SHA384
cs_c074	0xC0,0x74	TLS_ECDH_ECDSA_WITH_CAMELLIA_128_CBC_SHA256
cs_c075	0xC0,0x75	TLS_ECDH_ECDSA_WITH_CAMELLIA_256_CBC_SHA384
cs_c076	0xC0,0x76	TLS_ECDHE_RSA_WITH_CAMELLIA_128_CBC_SHA256
cs_c077	0xC0,0x77	TLS_ECDHE_RSA_WITH_CAMELLIA_256_CBC_SHA384
cs_c078	0xC0,0x78	TLS_ECDH_RSA_WITH_CAMELLIA_128_CBC_SHA256
cs_c079	0xC0,0x79	TLS_ECDH_RSA_WITH_CAMELLIA_256_CBC_SHA384
cs_c07a	0xC0,0x7A	TLS_RSA_WITH_CAMELLIA_128_GCM_SHA256
cs_c07b	0xC0,0x7B	TLS_RSA_WITH_CAMELLIA_256_GCM_SHA384
cs_c07c	0xC0,0x7C	TLS_DHE_RSA_WITH_CAMELLIA_128_GCM_SHA256
cs_c07d	0xC0,0x7D	TLS_DHE_RSA_WITH_CAMELLIA_256_GCM_SHA384
cs_c07e	0xC0,0x7E	TLS_DH_RSA_WITH_CAMELLIA_128_GCM_SHA256
cs_c07f	0xC0,0x7F	TLS_DH_RSA_WITH_CAMELLIA_256_GCM_SHA384
cs_c080	0xC0,0x80	TLS_DHE_DSS_WITH_CAMELLIA_128_GCM_SHA256
cs_c081	0xC0,0x81	TLS_DHE_DSS_WITH_CAMELLIA_256_GCM_SHA384
cs_c082	0xC0,0x82	TLS_DH_DSS_WITH_CAMELLIA_128_GCM_SHA256
cs_c083	0xC0,0x83	TLS_DH_DSS_WITH_CAMELLIA_256_GCM_SHA384
cs_c084	0xC0,0x84	TLS_DH_anon_WITH_CAMELLIA_128_GCM_SHA256
cs_c085	0xC0,0x85	TLS_DH_anon_WITH_CAMELLIA_256_GCM_SHA384
cs_c086	0xC0,0x86	TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_GCM_SHA256
cs_c087	0xC0,0x87	TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_GCM_SHA384
cs_c088	0xC0,0x88	TLS_ECDH_ECDSA_WITH_CAMELLIA_128_GCM_SHA256
cs_c089	0xC0,0x89	TLS_ECDH_ECDSA_WITH_CAMELLIA_256_GCM_SHA384
cs_c08a	0xC0,0x8A	TLS_ECDHE_RSA_WITH_CAMELLIA_128_GCM_SHA256
cs_c08b	0xC0,0x8B	TLS_ECDHE_RSA_WITH_CAMELLIA_256_GCM_SHA384
cs_c08c	0xC0,0x8C	TLS_ECDH_RSA_WITH_CAMELLIA_128_GCM_SHA256
cs_c08d	0xC0,0x8D	TLS_ECDH_RSA_WITH_CAMELLIA_256_GCM_SHA384
cs_c08e	0xC0,0x8E	TLS_PSK_WITH_CAMELLIA_128_GCM_SHA256
cs_c08f	0xC0,0x8F	TLS_PSK_WITH_CAMELLIA_256_GCM_SHA384
cs_c090	0xC0,0x90	TLS_DHE_PSK_WITH_CAMELLIA_128_GCM_SHA256
cs_c091	0xC0,0x91	TLS_DHE_PSK_WITH_CAMELLIA_256_GCM_SHA384
cs_c092	0xC0,0x92	TLS_RSA_PSK_WITH_CAMELLIA_128_GCM_SHA256
cs_c093	0xC0,0x93	TLS_RSA_PSK_WITH_CAMELLIA_256_GCM_SHA384

cs_c094	0xC0,0x94	TLS_PSK_WITH_CAMELLIA_128_CBC_SHA256
cs_c095	0xC0,0x95	TLS_PSK_WITH_CAMELLIA_256_CBC_SHA384
cs_c096	0xC0,0x96	TLS_DHE_PSK_WITH_CAMELLIA_128_CBC_SHA256
cs_c097	0xC0,0x97	TLS_DHE_PSK_WITH_CAMELLIA_256_CBC_SHA384
cs_c098	0xC0,0x98	TLS_RSA_PSK_WITH_CAMELLIA_128_CBC_SHA256
cs_c099	0xC0,0x99	TLS_RSA_PSK_WITH_CAMELLIA_256_CBC_SHA384
cs_c09a	0xC0,0x9A	TLS_ECDHE_PSK_WITH_CAMELLIA_128_CBC_SHA256
cs_c09b	0xC0,0x9B	TLS_ECDHE_PSK_WITH_CAMELLIA_256_CBC_SHA384
cs_c09c	0xC0,0x9C	TLS_RSA_WITH_AES_128_CCM
cs_c09d	0xC0,0x9D	TLS_RSA_WITH_AES_256_CCM
cs_c09e	0xC0,0x9E	TLS_DHE_RSA_WITH_AES_128_CCM
cs_c09f	0xC0,0x9F	TLS_DHE_RSA_WITH_AES_256_CCM
cs_c0a0	0xC0,0xA0	TLS_RSA_WITH_AES_128_CCM_8
cs_c0a1	0xC0,0xA1	TLS_RSA_WITH_AES_256_CCM_8
cs_c0a2	0xC0,0xA2	TLS_DHE_RSA_WITH_AES_128_CCM_8
cs_c0a3	0xC0,0xA3	TLS_DHE_RSA_WITH_AES_256_CCM_8
cs_c0a4	0xC0,0xA4	TLS_PSK_WITH_AES_128_CCM
cs_c0a5	0xC0,0xA5	TLS_PSK_WITH_AES_256_CCM
cs_c0a6	0xC0,0xA6	TLS_DHE_PSK_WITH_AES_128_CCM
cs_c0a7	0xC0,0xA7	TLS_DHE_PSK_WITH_AES_256_CCM
cs_c0a8	0xC0,0xA8	TLS_PSK_WITH_AES_128_CCM_8
cs_c0a9	0xC0,0xA9	TLS_PSK_WITH_AES_256_CCM_8
cs_c0aa	0xC0,0xAA	TLS_PSK_DHE_WITH_AES_128_CCM_8
cs_c0ab	0xC0,0xAB	TLS_PSK_DHE_WITH_AES_256_CCM_8
cs_c0ac	0xC0,0xAC	TLS_ECDHE_ECDSA_WITH_AES_128_CCM
cs_c0ad	0xC0,0xAD	TLS_ECDHE_ECDSA_WITH_AES_256_CCM
cs_c0ae	0xC0,0xAE	TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8
cs_c0af	0xC0,0xAF	TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8
cs_c0b0	0xC0,0xB0	TLS_ECCPWD_WITH_AES_128_GCM_SHA256
cs_c0b1	0xC0,0xB1	TLS_ECCPWD_WITH_AES_256_GCM_SHA384
cs_c0b2	0xC0,0xB2	TLS_ECCPWD_WITH_AES_128_CCM_SHA256
cs_c0b3	0xC0,0xB3	TLS_ECCPWD_WITH_AES_256_CCM_SHA384
cs_c0b4	0xC0,0xB4	TLS_SHA256_SHA256
cs_c0b5	0xC0,0xB5	TLS_SHA384_SHA384
cs_c100	0xC1,0x00	TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC
cs_c101	0xC1,0x01	TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC
cs_c102	0xC1,0x02	TLS_GOSTR341112_256_WITH_28147_CNT_IMIT
cs_c103	0xC1,0x03	TLS_GOSTR341112_256_WITH_KUZNYECHIK_MGM_L
cs_c104	0xC1,0x04	TLS_GOSTR341112_256_WITH_MAGMA_MGM_L
cs_c105	0xC1,0x05	TLS_GOSTR341112_256_WITH_KUZNYECHIK_MGM_S
cs_c106	0xC1,0x06	TLS_GOSTR341112_256_WITH_MAGMA_MGM_S
cs_c1a8	0xCC,0xA8	TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
cs_c1a9	0xCC,0xA9	TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256

cs_c1aa	0xCC,0xAA	TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256
cs_c1ab	0xCC,0xAB	TLS_PSK_WITH_CHACHA20_POLY1305_SHA256
cs_c1ac	0xCC,0xAC	TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256
cs_c1ad	0xCC,0xAD	TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256
cs_c1ae	0xCC,0xAE	TLS_RSA_PSK_WITH_CHACHA20_POLY1305_SHA256
cs_d001	0xD0,0x01	TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256
cs_d002	0xD0,0x02	TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384
cs_d003	0xD0,0x03	TLS_ECDHE_PSK_WITH_AES_128_CCM_8_SHA256
cs_d005	0xD0,0x05	TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256

Appendix C
TLS Signature Algorithms List

Dataset Value	IANA Value	Description
sig_0201	0x0201	rsa_pkcs1_sha1
sig_0203	0x0203	ecdsa_sha1
sig_0401	0x0401	rsa_pkcs1_sha256
sig_0403	0x0403	ecdsa_secp256r1_sha256
sig_0420	0x0420	rsa_pkcs1_sha256_legacy
sig_0501	0x0501	rsa_pkcs1_sha384
sig_0503	0x0503	ecdsa_secp384r1_sha384
sig_0520	0x0520	rsa_pkcs1_sha384_legacy
sig_0601	0x0601	rsa_pkcs1_sha512
sig_0603	0x0603	ecdsa_secp521r1_sha512
sig_0620	0x0620	rsa_pkcs1_sha512_legacy
sig_0704	0x0704	eccsi_sha256
sig_0705	0x0705	iso_ibs1
sig_0706	0x0706	iso_ibs2
sig_0707	0x0707	iso_chinese_ibs
sig_0708	0x0708	sm2sig_sm3
sig_0709	0x0709	gostr34102012_256a
sig_070a	0x070A	gostr34102012_256b
sig_070b	0x070B	gostr34102012_256c
sig_070c	0x070C	gostr34102012_256d
sig_070d	0x070D	gostr34102012_512a
sig_070e	0x070E	gostr34102012_512b
sig_070f	0x070F	gostr34102012_512c
sig_0804	0x0804	rsa_pss_rsae_sha256
sig_0805	0x0805	rsa_pss_rsae_sha384
sig_0806	0x0806	rsa_pss_rsae_sha512
sig_0807	0x0807	ed25519
sig_0808	0x0808	ed448
sig_0809	0x0809	rsa_pss_pss_sha256
sig_080a	0x080A	rsa_pss_pss_sha384
sig_080b	0x080B	rsa_pss_pss_sha512
sig_081a	0x081A	ecdsa_brainpoolP256r1tls13_sha256
sig_081b	0x081B	ecdsa_brainpoolP384r1tls13_sha384
sig_081c	0x081C	ecdsa_brainpoolP512r1tls13_sha512

Appendix D
TLS Supported Groups List

Dataset Value	IANA Value	Description
grp_01	1	sect163k1
grp_02	2	sect163r1
grp_03	3	sect163r2
grp_04	4	sect193r1
grp_05	5	sect193r2
grp_06	6	sect233k1
grp_07	7	sect233r1
grp_08	8	sect239k1
grp_09	9	sect283k1
grp_10	10	sect283r1
grp_11	11	sect409k1
grp_12	12	sect409r1
grp_13	13	sect571k1
grp_14	14	sect571r1
grp_15	15	secp160k1
grp_16	16	secp160r1
grp_17	17	secp160r2
grp_18	18	secp192k1
grp_19	19	secp192r1
grp_20	20	secp224k1
grp_21	21	secp224r1
grp_22	22	secp256k1
grp_23	23	secp256r1
grp_24	24	secp384r1
grp_25	25	secp521r1
grp_26	26	brainpoolP256r1
grp_27	27	brainpoolP384r1
grp_28	28	brainpoolP512r1
grp_29	29	x25519
grp_30	30	x448
grp_31	31	brainpoolP256r1tls13
grp_32	32	brainpoolP384r1tls13
grp_33	33	brainpoolP512r1tls13
grp_34	34	GC256A
grp_35	35	GC256B
grp_36	36	GC256C
grp_37	37	GC256D
grp_38	38	GC512A
grp_39	39	GC512B
grp_40	40	GC512C

grp_4	41	curveSM2
grp_256	256	ffdhe2048
grp_257	257	ffdhe3072
grp_258	258	ffdhe4096
grp_259	259	ffdhe6144
grp_260	260	ffdhe8192
grp_65281	65281	arbitrary_explicit_prime_curves
grp_65282	65282	arbitrary_explicit_char2_curves
grp_grease		Used for "random" values provided as a part of TLS GREASE

Appendix E
TLS Extensions List

Dataset Value	IANA Value	Extension Name
svr_ext_00	0	server_name
svr_ext_01	1	max_fragment_length
svr_ext_02	2	client_certificate_url
svr_ext_03	3	trusted_ca_keys
svr_ext_04	4	truncated_hmac
svr_ext_05	5	status_request
svr_ext_06	6	user_mapping
svr_ext_07	7	client_authz
svr_ext_08	8	server_authz
svr_ext_09	9	cert_type
svr_ext_10	10	supported_groups (renamed from "elliptic_curves")
svr_ext_11	11	ec_point_formats
svr_ext_12	12	srp
svr_ext_13	13	signature_algorithms
svr_ext_14	14	use_srtp
svr_ext_15	15	heartbeat
svr_ext_16	16	application_layer_protocol_negotiation
svr_ext_17	17	status_request_v2
svr_ext_18	18	signed_certificate_timestamp
svr_ext_19	19	client_certificate_type
svr_ext_20	20	server_certificate_type
svr_ext_21	21	padding
svr_ext_22	22	encrypt_then_mac
svr_ext_23	23	extended_master_secret
svr_ext_24	24	token_binding
svr_ext_25	25	cached_info
svr_ext_26	26	tls_lts
svr_ext_27	27	compress_certificate
svr_ext_28	28	record_size_limit
svr_ext_29	29	pwd_protect
svr_ext_30	30	pwd_clear
svr_ext_31	31	password_salt
svr_ext_32	32	ticket_pinning
svr_ext_33	33	tls_cert_with_extern_psk
svr_ext_34	34	delegated_credentials
svr_ext_35	35	session_ticket (renamed from "SessionTicket TLS")
svr_ext_36	36	TLMSP
svr_ext_37	37	TLMSP_proxying
svr_ext_38	38	TLMSP_delegate
svr_ext_39	39	supported_ekt_ciphers

svr_ext_40	40	Unassigned
svr_ext_41	41	pre_shared_key
svr_ext_42	42	early_data
svr_ext_43	43	supported_versions
svr_ext_44	44	cookie
svr_ext_45	45	psk_key_exchange_modes
svr_ext_46	47	certificate_authorities
svr_ext_48	48	oid_filters
svr_ext_42	49	post_handshake_auth
svr_ext_50	50	signature_algorithms_cert
svr_ext_51	51	key_share
svr_ext_52	52	transparency_info
svr_ext_55	55	external_id_hash
svr_ext_56	56	external_session_id
svr_ext_65281	65281	renegotiation_info
svr_ext_unassigned		Used for "random" values provided by client