Identifying Encrypted Malware Traffic with Contextual Flow Data

Blake Anderson
Cisco
blake.anderson@cisco.com

David McGrew
Cisco
mcgrew@cisco.com

ABSTRACT

Identifying threats contained within encrypted network traffic poses a unique set of challenges. It is important to monitor this traffic for threats and malware, but do so in a way that maintains the integrity of the encryption. Because pattern matching cannot operate on encrypted data, previous approaches have leveraged observable metadata gathered from the flow, e.g., the flow's packet lengths and inter-arrival times. In this work, we extend the current state-of-the-art by considering a *data omnia* approach. To this end, we develop supervised machine learning models that take advantage of a unique and diverse set of network flow data features. These data features include TLS handshake metadata, DNS contextual flows linked to the encrypted flow, and the HTTP headers of HTTP contextual flows from the same source IP address within a 5 minute window.

We begin by exhibiting the differences between malicious and benign traffic's use of TLS, DNS, and HTTP on millions of unique flows. This study is used to design the feature sets that have the most discriminatory power. We then show that incorporating this contextual information into a supervised learning system significantly increases performance at a 0.00% false discovery rate for the problem of classifying encrypted, malicious flows. We further validate our false positive rate on an independent, real-world dataset.

Keywords

Encryption; Malware; Machine Learning; Transport Layer Security; Network Monitoring

1. INTRODUCTION

With an increasing amount of encrypted network traffic, the burden of determining its trustworthiness is becoming too onerous a task for most incident response teams. Traditional approaches of identifying threats, such as deep packet inspection and signatures, are not applicable on encrypted traffic. Solutions that decrypt network traffic weaken the privacy of the users, do not work for all encryption, and are computationally intensive. Furthermore, these solutions' reliance on the configuration of a cooperating endpoint makes it challenging to deploy, and limits its applicability.

In this paper, we do not propose decrypting the network traffic. We instead focus on identifying malware communication in encrypted traffic through passive monitoring, the extraction of relevant data features, and supervised machine learning based on a large set of sandbox malware samples and data collected at a large enterprise network. We make the following novel contributions:

1. We provide the first results that leverage contextual information, i.e., DNS responses and HTTP headers, to identify threats in encrypted traffic.
2. We demonstrate highly accurate, with respect to a 0.00% false discovery rate, machine learning algorithms that operate on this data.
3. Finally, we provide a real-world validation of the methods we develop, demonstrating that our results are not simply due to overfitting.

Given the unique set of challenges that threat detection on encrypted traffic presents, and our desire to develop machine learning models that are as robust as possible, it is natural to consider including all possible data views associated with an encrypted flow. We term this viewpoint the *data omnia* approach. Conceptually, this can be achieved by extending flow records to contain all metadata about a flow, such as the unencrypted TLS handshake information, and pointers to "contextual" flows. We define DNS contextual flows to be DNS responses correlated with the TLS flow based on destination IP address, and contextual HTTP flows to be HTTP flows originating from the same source IP address within a 5 minute window. This approach is distinct from existing multiple flow techniques because it uses detailed information about the flow and contextual flows, and not simply the flow metadata. For example, Figure 1 demonstrates how we can link a DNS flow with a TLS flow, and the type of additional information that this approach makes available.

We provide an in-depth analysis of the TLS handshake metadata and two contextual flow types that we found to be particularly relevant for identifying threats in encrypted traffic. Unlike previous work, we present a detailed analysis of the protocol feature values that differ between malicious and benign traffic. Our motivation for making these observations public was that any motivated threat actor, given the feature type, which in most cases has been published in the open literature, could observe benign traffic and attempt to modify their servers and clients to mimic the observed

AISec'16, October 28 2016, Vienna, Austria

DOI: http://dx.doi.org/10.1145/2996758.2996768

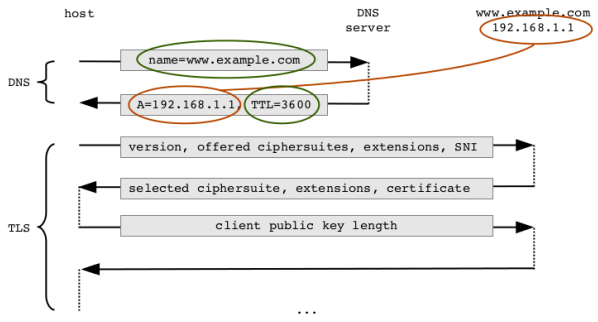

Figure 1: An illustration of a TLS flow and a DNS contextual flow, showing the data elements used to link the flows (in red), the data elements brought in as context (in green), and the TLS unencrypted header information collected as metadata (unmarked).

behavior. On the other hand, by obfuscating the feature values, we increase the difficulty for incident responders to write and deploy indicators of compromise.

The first type of contextual flow that we analyze is the DNS response that provides the address used by an encrypted flow, and the TTL associated with the name. Having the domain name for an IP address provides a lot of meaningful information on its own. Among TLS flows, this information can sometimes be gathered from the server name indication extension or the subject of the server certificate. But, the SNI extension is optional and there will be no server certificate in the case of TLS resumption. In these cases, the contextual DNS flow has the potential to provide information which would otherwise be unavailable. Additionally, as we outline in this paper, malicious DNS responses have characteristics that can distinguish them from benign DNS responses, and we can use this information to more accurately classify the corresponding TLS encrypted flow.

In addition to DNS flows, which can be directly correlated with the TLS encrypted flows, we analyze the HTTP headers of HTTP contextual flows. There have been several rule-based systems and machine learning classifiers that have been based on HTTP data [29, 33], and we build on these studies by taking advantage of HTTP header information to help classify encrypted flows. There are also several interesting rule-based inferences that can be made by correlating HTTP data with the unencrypted TLS handshake metadata. For instance, the TLS offered ciphersuite list and extensions can be used to infer the cryptographic library and version in use, which in turn can be used to infer the `user-agent` that initiated the flow. Finding discrepancies between the `user-agent` that a flow advertises in its HTTP fields and the `user-agent` inferred from the adjacent, encrypted flow's TLS parameters is a useful indicator of compromise.

We wrote a custom `libpcap`-based tool [25, 26] to capture our data features from live traffic, and to process malware packet capture files. This tool was run on malicious packet capture files collected from ThreatGRID [2], a commercial sandbox environment, between January 2016 and April 2016. This tool was also run on a large enterprise network's DMZ for a 5 day period in April 2016. This process resulted in tens-of-millions of malicious and benign flows. We realize that the DMZ traffic does contain a small amount of malware traffic, but for the purpose of this paper, we refer to this traffic as benign. Our analysis tools were based on Python and Scikit-learn [32].

In this paper, we show that the data omnia approach is both practical and valuable. For our machine learning applications, we take a bottom-up approach. From the data that we collected, we first identify data features of TLS, DNS, and HTTP that have discriminatory power. We then show that machine learning algorithms can be defined using these data features that can correctly classify their respective flow types. Finally, we leverage the context that the HTTP and DNS flows provide to help classify TLS encrypted network flows. When processing tens-of-millions of flows each day, high total accuracy with even a modest false discovery rate can overwhelm an analyst. For this reason, instead of total accuracy, we focus our results on an accuracy at a 0.00% false discovery rate, that is, an FDR of zero with four significant figures. To further defend the false positive rates presented in this paper, and to confirm that our results were not simply due to overfitting on the initial datasets and feature sets, we also ran experiments on an additional validation dataset collected ∼4 weeks after the initial benign dataset.

We embrace supervised machine learning as the best way to use previously observed malware communications to detect encrypted malware communications. A machine learning classifier provides the most direct way to build a detector, and it can also provide a probability estimate. Supervised learning provides grounded and easily interpretable results, unlike anomaly detection [36]. Importantly, regularization can be used during classifier training to select the data features that have the most discriminatory power [21], which is essential to our data omnia approach. Lastly, there are classifiers that are robust against overfitting [23]; by using these, we avoid that pitfall. In Section 6, we further defend these claims by demonstrating that efficient algorithms, whose results can be easily interpreted, have equivalent performance to inefficient, black-box models on this data.

The remainder of this paper is organized as follows: Section 2 details our in-depth TLS study, Section 3 details our in-depth DNS study, and Section 4 details our in-depth HTTP study. Section 5 reviews our datasets and extracted features, and Section 6 presents our classification results. Finally, Section 7 reviews background material and related work, and we conclude in Section 8.

2. MALWARE AND TLS

Malware and benign traffics' usage of TLS is quite distinct. In this section, we detail these differences from the perspective of the client by examining the offered ciphersuites, the advertised TLS extensions, and the client's public key length. We also look at the differences in the server's TLS implementation by examining the selected ciphersuite and information gathered from the server certificates. We collected 21,417 malicious TLS flows that had the full TLS handshake from ThreatGRID [2] between January 2016 and April 2016, and 1,130,386 benign TLS flows using the same criteria during a 5 day period in April 2016. Our tool was used to parse the TLS flows. It collected all of the information contained in the unencrypted TLS handshake messages.

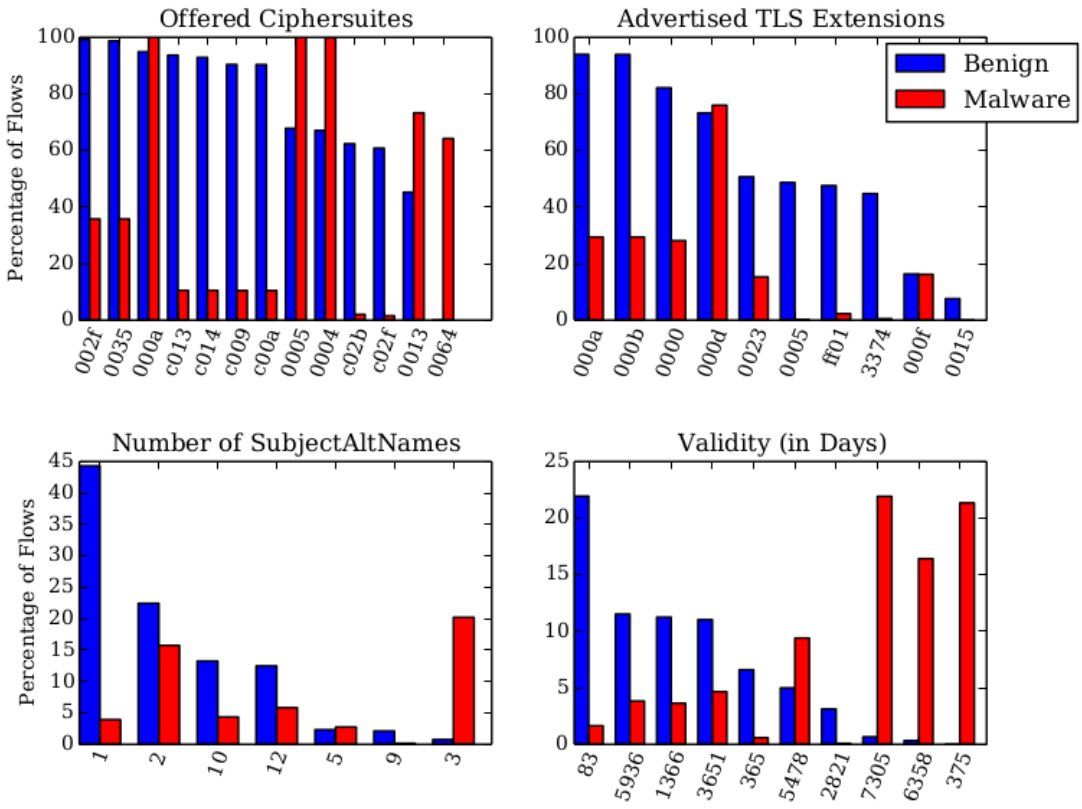

Figure 2: TLS Statistics, revealing the different prevalence of feature values for malware (red) and benign (blue).

The unencrypted TLS metadata contained in the `clientHello` and `clientKeyExchange` messages contains valuable information that can be used to make inferences about the client's TLS library. We have also observed that these features are quite different in our malware and benign datasets; the implication being that malware authors use a distinct set of TLS libraries and/or configurations. Figure 2 illustrates differences in two client-side TLS features: the offered ciphersuites and the advertised extensions. Using standard guidelines and industry recommendations [39], TLS ciphersuites can be segmented into acceptable and obsolete categories. We can see that malware usually offers a set of three obsolete ciphersuites in the `clientHello` message including `0x0004` (`TLS_RSA_WITH_RC4_128_MD5`). In the benign traffic we collected, the `0x002f` (`TLS_RSA_WITH_AES_128_CBC_SHA`) ciphersuite was the most offered. Malware also seems to have comparatively little diversity in the client-supported TLS extensions. `0x000d` (`signature_algorithms`) was the only TLS extension supported in the majority of TLS flows. ∼50% of the DMZ traffic also advertised the following extensions, which were rarely seen in the malware dataset:

- `0x0005` (status request)
- `0x3374` (next protocol negotiation)
- `0xff01` (renegotiation info)

Although not shown, the client's public key length was another client-based data feature that had significant differences. Most of the DMZ traffic used 256-bit elliptic curve cryptography for the public keys, but most of the malicious traffic used 2048-bit RSA public keys.

The `serverHello` and `certificate` messages can be used to gain information about the server. The `serverHello` message contains the selected ciphersuite and supported extensions. As one would expect given the type and diversity of the offered ciphersuites and the advertised extensions, the malicious traffic most often selected obsolete ciphersuites. The DMZ traffic contained a wider variety of supported TLS extensions by the servers.

The `certificate` message passes the server's certificate chain to the client. We observed that the number of certificates in the chain for the malware and DMZ data were roughly the same. But, if we restrict our focus on the length-1 chains, ∼70% were self-signed for malware and ∼.1% were self-signed for the DMZ traffic. The number of names in the SubjectAltName (SAN) X.509 extension also differed in the two datasets. For the DMZ traffic, the length of the list was 1 ∼45% of the time. This is in part because a number of Content Distribution Network (CDN) providers, e.g., Akamai, only have one entry. Length-10/12 lists were also common in the DMZ traffic due to some ad services.

Figure 2 also shows the distribution of the validity of the certificates rounded to the nearest day. Similar to the other data features, the period of validity for a server certificate has notable differences in the malicious and DMZ traffic. Combining the certificate information with information from

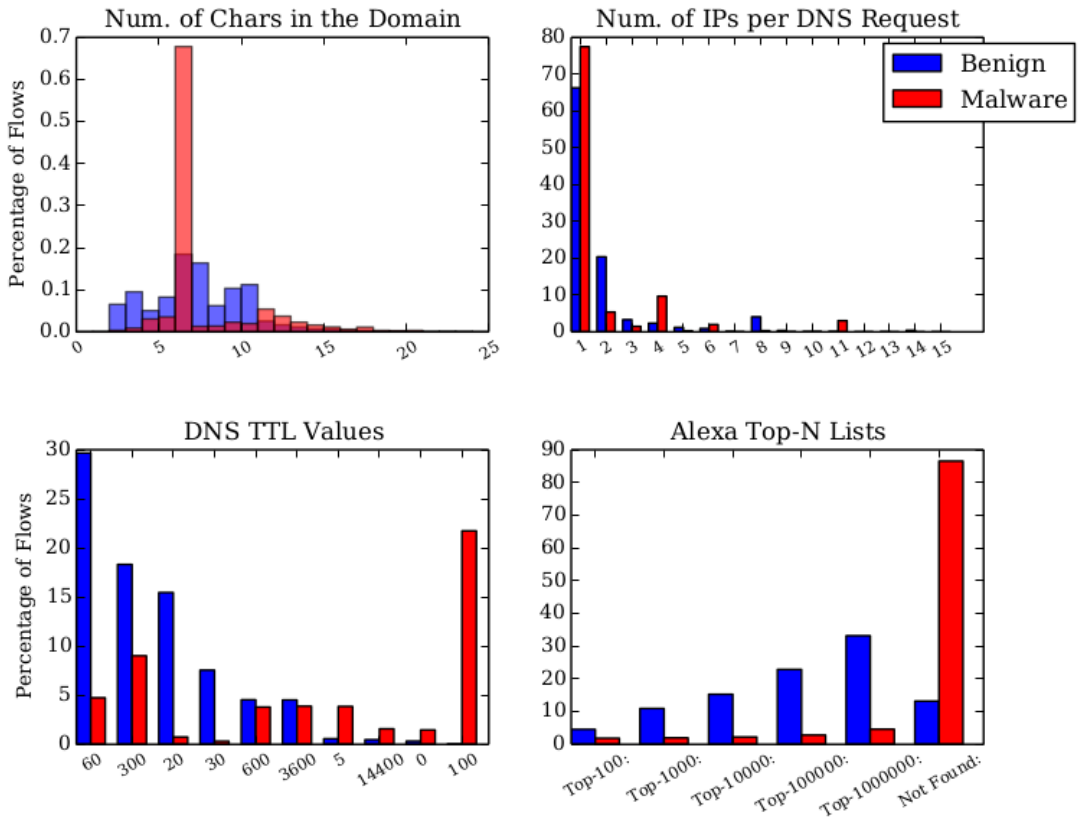

Figure 3: Statistics of DNS flows linked to TLS flows, revealing the different prevalence of feature values for malware (red) and benign (blue).

the client can help improve classifier performance, and help attributed encrypted flows to specific malware families [6].

3. MALWARE AND DNS

In this section, we systemically compare malware's use of DNS to that of benign's. Using the same data sources as the previous section, we collected 6,906,627 malicious and 8,060,064 benign DNS responses. Our tool was used to perform the DNS parsing of the malware packet capture files and live DMZ data.

From a signature perspective, domain names provided by DNS are less dynamic than the associated IP addresses, which allows for more robust blacklists. This information also provides visibility into encrypted flows that is missing in many cases. In some cases, the server name indication (SNI) TLS extension and the subject / subject alternative names (SANs) in the server certificate can provide this information. In the case of TLS session resumption, the certificate will not be present, and with the release of TLS 1.3, the server certificate will most likely be encrypted. Also, in our studies, we found that malware only took advantage of the SNI extension in ∼27% of the observed TLS flows. On the other hand, DNS responses were available for ∼78% of the malicious TLS flows in our dataset, and were available for ∼73% of the malicious TLS flows that lacked the SNI extension.

Domain Generation Algorithms (DGA) generate a large number of domains for the malware to try to communicate with. This process gives malware a more robust method of contacting its command and control server. It is often the case that these algorithms have implicit or explicit biases that make detecting a DGA domain possible. We examined some simple statistics about the domain name and the fully qualified domain name (FQDN) to aid in these inferences, and more generally, to determine the differences in benign and malicious domain names.

Figure 3 illustrates a histogram of the number of characters in the domain names in our dataset. This analysis was also done on the FQDN. For the domain name lengths, the benign domain names had a roughly Gaussian shape centered around 6/7. The malicious domain names and FQDNs had a sharp peak at 6 and 10, respectively. After manual inspection of the names and the associated pcaps, it was clear that this phenomena was due to DGA activity, i.e., there were many such random-looking DNS responses per sample with the same length. Contrary to our initial intuition, it seems that benign domains have a longer tail for the number of characters in the FQDN. This was because of how cloud-based services structure their FQDNs.

We examined two more metrics on the FQDNs: the percentage of numerical characters and the percentage of non-alphanumeric characters. In contrast to previous work [9], we found that the benign DNS responses had a higher per-

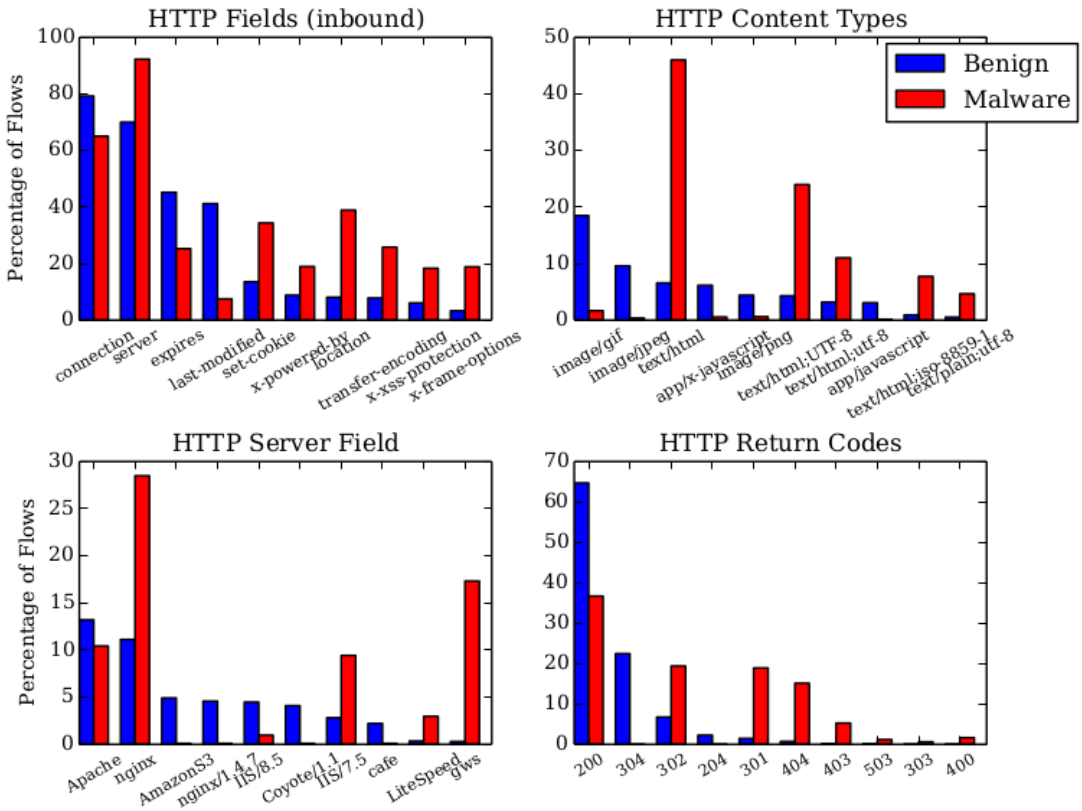

Figure 4: Statistics of HTTP flows contextual to TLS flows, revealing the different prevalence of feature values for malware (red) and benign (blue).

centage of numerical characters, 13.44% versus 0.85%, and a higher percentage of non-alphanumeric characters, 16.36% versus 9.61%. Non-alphanumeric characters include wildcards and periods. We observed a significant amount of benign traffic visiting cloud services and CDNs which could have caused this discrepancy between prior work.

In addition to the FQDN, the DNS response provides other interesting data elements. Figure 3 shows the number of distinct IP addresses returned. The majority of DNS responses return 1 IP address for both malicious and benign responses. Beyond those cases, there is some interesting structure where we see significantly more benign responses returning 2 and 8 IP addresses and significantly more malware responses returning 4 and 11 IP addresses.

Figure 3 also shows the prevalence of different TTL values between malware and benign DNS responses. The four most common TTL values for benign DNS responses are 60, 300, 20, and 30, in order. TTL value 300 is malware's second most common, but TTL values 20 and 30 are rarely observed. It is also interesting to note that ∼22% of malware DNS responses used a TTL value of 100, a value that was not observed in our DMZ data.

Alexa [1] ranks websites based on the number of page views and the number of unique IP addresses. We recorded the top-1,000,000 websites from Alexa in April 2016. We used this list to create 6 categories for the domain names: whether the target domain name was in the top-100, top-1,000, top-10,000, top-100,000, top-1,000,000 or not found in the Alexa list. The most prestigious category was chosen for each domain name, i.e., a domain name would belong to top-100 but not top-100 and top-1,000. Figure 3 shows the distributions of domain names with respect to the Alexa lists for the malware and DMZ traffic. As expected, roughly ∼86% of the domain names that the malware samples looked up were not found in the Alexa top-1,000,000 list. On the other hand, Figure 3 shows that the DMZ traffic had the majority of its domain names in the top-1,000,000.

Given the results presented in Figure 3, it is clear that DNS provides valuable, discriminatory information that can be correlated with encrypted flows to provide additional context for increased visibility, and can be used to create highly accurate machine learning classifiers.

4. MALWARE AND HTTP

Finally, we present the differences we found in the DMZ and malware traffic with respect to the HTTP headers. We again used the same data sources, and filtered on port 80 HTTP traffic. We collected 1,743,842 HTTP flows from the DMZ and 1,004,798 HTTP flows from ThreatGRID [2]. Our tool parsed all available header fields and values from the flows, thus leaving intact information that would be discarded by web proxy logs. Those logs only report on a fixed subset of all headers, and they normalize the values that are

reported, discarding valuable data. For instance, the only HTTP-specific data features available in the exported IIS 6.0 web server logs are the `Method`, `URI`, `code`, `User-Agent`, `Referer`, `Cookie`, and `Host`. Of these, only the first four are enabled by default [4]. Our tool also maintains the capitalization and ordering of the HTTP fields.

The appearance of an HTTP field or set of HTTP fields can be a good indicator for malicious activity. Figure 4 lists the prevalence of some interesting inbound HTTP fields. For instance, malicious HTTP is more likely to make use of the `Server`, `Set-Cookie`, and `Location` fields, while the DMZ HTTP traffic was more likely to make use of the `Connection`, `Expires`, and `Last-Modified` fields. For outbound HTTP fields, the DMZ HTTP was more likely to make use of the `User-Agent`, `Accept-Encoding`, and `Accept-Language` fields.

As Figure 4 demonstrates, the dominant HTTP `Content-Type` for the DMZ traffic was `image/*`, and the malware traffic was mostly `text/*`. `Content-Type` also serves as an example of why you should not normalize the values of these fields. The second and third most common content types for malware are `text/html;charset=UTF-8` and `text/html; charset=utf-8`. These subtle differences have an incredible amount of value in terms of detection and attribution, and should not be discarded.

Figure 4 also shows the value of the `Server` and `code` inbound HTTP fields. Malware most often says that it is using a version-less `nginx` server, and the benign traffic most often says that it is using either the version-less `Apache` or `nginx` server. In terms of interesting servers, `AmazonS3` and `nginx/1.4.7` were announced almost exclusively by the benign servers, and `LiteSpeed` and `gws` were announced almost exclusively by the malicious servers.

The outbound `User-Agent` field had a very long tail, several thousand unique strings in both datasets. For the malware data, the most common advertised `User-Agent` string was `Opera/9.50(WindowsNT6.0;U;en)`, followed by several variations of `Mozilla/5.0` and `Mozilla/4.0`. All of the top `User-Agent` strings in the DMZ data were Windows and OS X variants of `Mozilla/5.0`. This field also had the most diverse set of capitalizations that we observed:

- `User-Agent`
- `user-agent`
- `User-agent`
- `USER-AGENT`
- `User-AgEnt`

The `User-Agent` field is also another interesting example of the power of these fields in the context of encrypted traffic. Finding a discrepancy between what software is being used on an endpoint and what software is being advertised by an endpoint is an interesting indicator of compromise. By correlating HTTP and TLS data, specifically the `User-Agent` field with the TLS library, we can make useful inferences.

For instance, we can get a reasonable estimate of the browser in use by observing the TLS metadata, inferring the TLS library, and then correlating the TLS library with the probable browser, or set of browsers. As we will show in Section 6, it is quite interesting when the TLS browser estimate is different from what is being advertised in the HTTP `User-Agent` field.

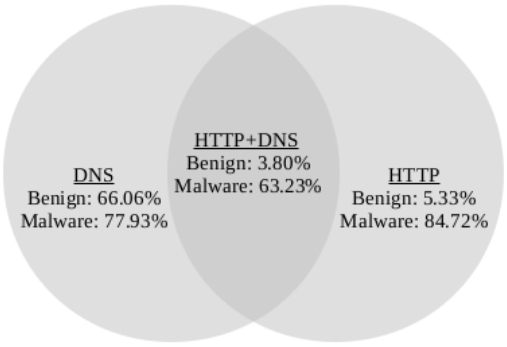

Figure 5: The prevalence of different contextual data, showing the percentage of TLS flows that had DNS and/or HTTP context, for both benign (DMZ) and malware flows.

5. DATA

In this section, we reiterate our strategy for data collection. We also describe how each of the data features we explored in the previous sections are represented to the machine learning algorithms. Joy [26] was used to transform all of the data, either from packet capture files or collected live, into a convenient JSON format.

5.1 Malware Data

The malicious dataset was collected from January to April 2016 from a commercial sandbox environment that receives user submissions. Each submission is allowed to run for 5 minutes, and all network activity is collected for analysis. From this set, there were 21,417 successful TLS flows. Each flow completed the full handshake, and the TLS data from the `clientHello`, `serverHello`, `certificate`, and `clientKeyExchange` messages was collected.

Figure 5 gives the percentages of the other contextual data features that were also present. 16,691 of the TLS flows had an associated DNS lookup, and 18,144 had an active HTTP session during the sandbox detonation. 13,542, or ∼63%, of the malicious TLS flows had all available data. To accurately compare the classifiers based on the different data views, this set of 13,542 flows was used in the results for Section 6.

5.2 Benign Data

The benign dataset was collected during a 5-day period in April 2016 from a large enterprise network's DMZ. We again collected all TLS flows that successfully completed the TLS handshake and had all relevant messages. There were 1,130,386 such flows.

As shown in Figure 5, 746,723, or ∼66%, of the TLS flows had an associated DNS response. But, only 60,285, or ∼5%, of the TLS flows had any HTTP flows from the same source IP address within a 5 minute window of the TLS flow. HTTP context by itself is an interesting indicator, and should remain interesting as more of the Alexa top-1,000,000 websites transition to HTTPS. There were 42,927 TLS flows that had both DNS and HTTP context, and these flows were used for the results of Section 6.

As an additional validation of our results, we collected another dataset from the same enterprise DMZ in May 2016: ∼4 weeks after the initial dataset was collected. The same data collection and filtering strategy was employed. There were 35,699 TLS flows that had both DNS and HTTP context during this time period.

5.3 Data Features

5.3.1 Observable Metadata

Features based on observable metadata, such as the sequence of packet lengths and inter-arrival times, were used, and were modeled as Markov chains. We exclude TCP retransmissions from our data by having our tool track the TCP sequence number. The packet lengths were taken to be the sizes of the UDP, TCP, or ICMP packet payloads. If the packet was not one of those three types, then the length was set to the size of the IP packet. The inter-arrival times had a millisecond resolution.

For both the lengths and times, the values were discretized into equally sized bins. The length data Markov chain had 10 bins of 150 bytes each. A 1500 byte MTU was assumed, and any packets observed with a size greater than 1350 bytes were put into the same bin. The timing data Markov chain used 50 millisecond bins and 10 bins for 100 total features. Any inter-packet time greater than 450ms fell into the same bin. The transition probabilities were used as features for the machine learning algorithm.

Another form of observable metadata, the byte distribution, was represented as a length-256 array that keeps a count for each byte value encountered in the payloads of the packets of the flow being analyzed. The byte value probabilities of a flow can be easily computed given the byte distribution by dividing the byte distribution counts by the total number of bytes found in the packet payloads. The features used by the machine learning algorithms are the 256 byte distribution probabilities.

5.3.2 TLS Data

The client-based TLS-specific features used in our classification algorithm were the list of offered ciphersuites, the list of advertised extensions, and the client's public key length. We observed 176 unique hex codes advertised in the lists of offered ciphersuites, and created a binary vector of length 176 where a one is assigned to each hex code in the sample's list of offered ciphersuites. More advanced representations taking the order of the list into account were considered, but did not lead to significantly improved results and were therefore not pursued in favor of simplicity. Similarly, a length-21 binary vector was used to represent the TLS extensions. Finally, a single integer value was used to represent the public key length.

The server-based TLS-specific features included the selected ciphersuite, supported extensions, number of certificates, number of SAN names, validity in days, and whether there was a self-signed certificate.

5.3.3 DNS Data

From the associated DNS response of the TLS flow, we collect the lengths of both the domain name and the FQDN. We also harvested a list of the 40 most common suffixes, and have a binary feature for each suffix and a binary feature for "other". Similarly, we harvested a list of the 32 most common TTL values and an "other" option. We also had features for the number of numerical characters, the number of non-alphanumeric characters, and the number of IP addresses returned by the DNS response. Finally, we had six binary features representing whether the domain name was in the top-100, top-1,000, top-10,000, top-100,000, top-1,000,000 or not found in the Alexa list. The most prestigious category was chosen for each domain name, i.e., the top-100 feature would be a one, but not the top-1,000 feature.

5.3.4 HTTP Data

For each TLS flow, we collect all HTTP flows from the same source address within a 5 minute window of the TLS flow. There is a single feature vector of binary variables representing all of the observed HTTP headers. If any of the HTTP flows have a specific header value, then that feature will be a 1 regardless of the other HTTP flows.

We used seven types of features from the HTTP data. For each feature, we selected all specific values used by at least 1% of either the malware or benign samples and an "other" category. The types were the presence of outbound and inbound HTTP fields, `Content-Type`, `User-Agent`, `Accept-Language`, `Server`, and `code`.

6. CLASSIFYING ENCRYPTED TRAFFIC

In this section, we outline the type of results that can be achieved when looking at encrypted data from a data omnia perspective. Not only do we show that using all available data leads to incredibly accurate classifiers at a 0.00% false discovery rate, we also show how these machine learning models are interpretable. We provide evidence for the claim that correlating unencrypted metadata can lead to useful and interesting indicators of compromise for encrypted flows. Finally, we validate our methods on more real-world data - data that was held out from the initial training and tuning of the models and features.

6.1 Classification Results

For these results, we used the set of 13,542 malicious and 42,927 benign TLS flows that contained DNS and HTTP context, as explained in Section 5. All results used 10-fold cross-validation and $l1$-logistic regression. We will show that this classifier performed extremely well for our task, and has the added benefit of being easily interpretable. This model also reports a probabilistic output, allowing one to easy change the threshold of the classifier. We did compare $l1$-logistic regression with a support vector machine (Gaussian kernel, width adjusted through CV), and found **no** statistically-significant improvement using a 10-fold paired t-test at a 5% significance level [14]. Because of the added computational resources needed to train the SVM and the loss of interpretability, we emphasis only the $l1$-logistic regression results. Finally, all data features were normalized to zero-mean and unit variance.

Table 1 gives 10-fold, $l1$-logistic regression classification results for different combinations of feature sets. SPLT/BD are the sequence of packet lengths and times and byte distribution. TLS, HTTP, and DNS are self-explanatory. Only using information from the encrypted flow itself, SPLT+BD +TLS, we were able to achieve a total accuracy of 99.933%. Given the amount of encrypted flows seen on even modestly sized networks, total accuracy means very little because even a classifier with 99.99% total accuracy could have tens of

Dataset	Total Accuracy	0.00% FDR	Avg # of Model Parameters
SPLT+BD+TLS+HTTP+DNS	**99.993%**	**99.978%**	189.7
SPLT+BD+TLS+HTTP	**99.983%**	**99.956%**	209.8
SPLT+BD+TLS+DNS	**99.968%**	98.666%	197.1
TLS+HTTP+DNS	**99.988%**	99.889%	129.3
SPLT+BD+TLS	99.933%	77.881%	250.7
HTTP+DNS	**99.985%**	**99.956%**	109.2
TLS+HTTP	99.955%	99.660%	109.3
TLS+DNS	99.883%	96.849%	89.4
HTTP	99.945%	98.996%	87.5
DNS	99.496%	94.654%	50.2
TLS	96.335%	62.857%	51.6

Table 1: Classification results for for different combinations of data features.

thousands of false positives. For this reason, we focus on a 0.00% false discovery rate.

With ∼55,000 samples in our dataset, it is impossible to give a true, robust 0% false discovery rate. With that caveat, we do report a 0%, or 0.00%, false discovery rate for our classifiers. Comparatively, the performance of the SPLT+BD+TLS classifier suffers under this metric with 77.881% accuracy. But, once we take advantage of the additional HTTP and DNS context, the 0.00% false discovery rate becomes 99.978%, a significant improvement. We further defend the false positives when we analyze a real-world, validation set.

The results presented in Table 1 clearly show the benefit of utilizing contextual DNS and HTTP information to classify TLS encrypted flows. And while HTTP context is absent in many of the TLS flows collected from the DMZ, 66% of the TLS flows did have DNS information. Only TLS and DNS data still provided a 98.666% accuracy at a 0.00% FDR.

6.2 Model Interpretability

In an operational setting, having a black box classifier that returns a 0/1 response is suboptimal. Having the ability to add context to the classification decision is invaluable in terms of performing the appropriate response. The $l1$-logistic regression classifier we use is highly accurate, and has easily interpretable parameters that can be used to help explain the classifier's decision to an incident responder. Furthermore, the $l1$ penalty shrinks most of the parameters to 0, creating sparser models that are even more interpretable and generalize better. The number of parameters of each learned model, averaged over the cross-validation folds, is shown in Table 1. It is interesting to note, that the model with all possible data features actually has less model parameters than some other models with fewer data features, e.g., all features has an average of 189.7 and SPLT+BD+TLS has an average of 250.7 parameters.

Table 2 presents the top-5 parameters that influence a positive malware conviction and the top-5 parameters that influence a negative malware conviction. This $l1$-logistic regression classifier was built using all available data sources. Interestingly, data features from TLS, DNS and HTTP all show up in one or both of the top-5 lists.

The majority of these values are easily understood. For instance, the DNS feature `Alexa: None` was one of the top-ten contributors to a malicious conviction, and the majority of the malicious samples that we processed did not talk to a domain in the Alexa top-1,000,000. Interestingly, the DNS feature `Alexa: top-1,000,000` was the only Alexa-based feature that contributed to a benign classification, i.e., `Alexa: top-100`, etc. had a weight of 0. This can be ex-

Weight	Feature
3.38	DNS Suffix `org`
2.99	DNS TTL `3600`
2.62	TLS Ciphersuite `TLS_RSA_WITH_RC4_128_SHA`
2.28	HTTP Field `accept-encoding`
1.95	TLS Ciphersuite `SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA`
1.78	HTTP Field `location`
1.38	DNS `Alexa: None`
1.21	TLS Ciphersuite `TLS_RSA_WITH_RC4_128_MD5`
1.12	HTTP Server `nginx`
1.11	HTTP Code `404`
-2.16	TLS Extension `extended_master_secret`
-1.65	HTTP Content Type `application/octet-stream`
-1.61	HTTP Accept Language `en-US,en;q=0.5`
-1.35	TLS Ciphersuite `TLS_DHE_RSA_WITH_DES_CBC_SHA`
-1.10	HTTP Content Type `text/plain;charset=UTF-8`
-0.97	HTTP Server `Microsoft-IIS/8.5`
-0.95	DNS `Alexa: top-1,000,000`
-0.91	HTTP User-Agent `Microsoft-CryptoAPI/6.1`
-0.88	TLS Ciphersuite `TLS_ECDHE_ECDSA_WITH_RC4_128_SHA`
-0.85	HTTP Content Type `application/x-gzip`

Table 2: The data features most relevant to the TLS/DNS/HTTP classifier.

	All	TLS +HTTP	TLS +DNS	TLS	All Available
0.0	35,699	51,113	655,906	988,105	988,105
.5	86	312	4,207	20,186	4,602
.9	25	130	982	4,718	2,004
.95	18	97	855	3,014	1,644
.99	4	67	621	465	1,056

Table 3: Alarms generated based on different feature sets and thresholds on a real-world validation dataset.

plained by a non-trivial percentage of the malware samples performing connectivity checks to popular websites such as `www.google.com`, but comparatively fewer malware samples connecting to websites in the 100,00-1,000,000 range.

The TLS ciphersuite `TLS_RSA_WITH_RC4_128_SHA` was correlated with malicious convictions. It has been observed that there are significantly more malware flows that offer this ciphersuite than enterprise flows [6]. Again, these model parameters and the feature vectors of the samples can be used to easily explain why the classifier classified a flow as malicious or benign.

6.3 Second-Order Correlations

In addition to creating machine learning models, the data that we collect is useful for rule-based indicators of compromise. When detecting malicious behavior, it is useful to identify unexpected discrepancies. The server name indication (SNI) TLS extension [16] is available in the `clientHello` message and indicates the hostname of the server that the client is trying to connect to. The subjectAltName (SAN) X.509 extension [35] is available in the `certificate` message and allows a server to list additional names including additional DNS names.

We analyzed the 21,417 malicious and 1,130,386 benign TLS sessions that had server certificates looking for instances where the SAN and SNI were present, had relevant information, and disagreed. ∼0.01% of the benign flows and ∼8.25% of the malicious flows had a discrepancy. The benign cases were mostly IP addresses listed in the SNI and `*.whatsapp.net` and variations listed as SANs. Not only was this discrepancy more prominent in the malicious dataset, in many cases it was also a very different type of discrepancy than the benign cases. The format of most of the malware differences included DGA-like behavior in the SNI and SAN, e.g., SNI: `'bbostybfmaa.org'` and SAN: `'giviklorted.at'`.

More advanced differences can also be analyzed with the data sources that we have collected. Finding discrepancies between the `User-Agent` that a TLS-adjacent HTTP flow advertises and the `User-Agent` inferred from the TLS offered ciphersuites and advertised extensions is a another useful indicator of compromise. As an example, we observed 125 malicious TLS flows that had associated HTTP flows with a `User-Agent` string = `Firefox/31.0`. All TLS flows in this set offered the prototypical Windows XP ciphersuite list [27].

We also found 97 DMZ TLS flows that had an associated HTTP context with a `User-Agent` string = `Firefox/31.0`. These TLS flows offered an ordered ciphersuite list that matches what is offered by real versions of `Firefox/31.0` [34]. This line of thinking is not immune to false positives, but it does offer a unique way to identify indicators of compromise that arise when correlating disparate types of data.

6.4 Real-World Results

In a research setting with limited data, it is often difficult to avoid overfitting machine learning algorithms to a specific dataset. Even with a proper cross-validation methodology, implicit biases can occur in the meta-parameters such as the choice of machine learning algorithm or data features [15]. In addition to the cross-validated results, we now present results on an additional validation dataset. This data was collected over a 4 day period from the same enterprise DMZ ∼4 weeks after the initial dataset. The machine learning algorithms were trained only on the original dataset, and then applied to this new data. No changes to the algorithms or features were made. There were 35,699 TLS flows with DNS/HTTP context, 51,113 TLS flows with only HTTP context, 655,906 TLS flows with only DNS context, and 988,105 total TLS flows in this validation set.

Table 3 lists the number of alarms for each set of data features at different thresholds. The "All" classifier classified TLS flows that had HTTP and DNS contextual information. The "All Available" classifier classified all TLS flows, but used any contextual flow information that was available. The $l1$-logistic regression classifier reports a probability of maliciousness, and we can use this probability to easily adjust how many alarms are raised. For instance, at the default `0.5` threshold, the TLS-only classifier had 20,186 alarms. Adjusting the threshold to `0.99`, there were only 465 alarms during this 4-day period.

Unsurprisingly, the classifier that used DNS and HTTP contextual information performed the best on the validation dataset. This classifier had 86 total alarms with 29 unique destination IP addresses and 47 unique source IP addresses. 42 of these alarms appeared to be false positives. In nearly all of these 42 cases, the TLS flow was communicating with either a Google or Akamai server, but had contextual HTTP flows that communicated with suspicious domains. "Suspicious" was determined by extracting the `Host` HTTP field, and checking whether VirusTotal [3] found convicted executables with that domain name embedded in their source code. In this context, convicted means more than 50% of the VirusTotal detectors flagged the executable as malicious. These `Host` HTTP fields did not show up in the Alexa lists.

Adjusting the TLS+HTTP+DNS classifier's threshold to `0.95` reduced the number of alarms during this four day period to 18, and the number of unique destination IP addresses to 7. We manually checked each of the IP addresses. These encrypted flows were considered to be true positives if executables, convicted by VirusTotal, communicated with the destination IP address during the same dates that we collected our data. 16/18 of the TLS+HTTP+DNS classifier's alarms at a threshold of `0.95` were confirmed to be malicious using this method. Again, we could not correlate these IP addresses with domain names listed in the Alexa top-1,000,000, e.g., no IP address resolved to `google.com`.

7. RELATED WORK

Research in network-based malware detection has two main directions. The first, vertical correlation, uses the network traffic of a single host to find evidence of a malware infection. The second, horizontal correlation, uses the traffic of two

or more hosts to find malicious communication. Horizontal correlation can, in some cases, detect large-scale, malicious communication graphs. Some techniques are content agnostic, while others rely on content inspection. Our approach uses vertical correlation, and it is not fully content-agnostic because of its use of DNS, TLS, and HTTP metadata, but it does not rely on the inspection of the encrypted payload.

7.1 Flow Metadata

Traditionally, flow-monitoring systems have been used to collect metadata about network communications such as the IP addresses, ports, number of bytes exchanged and number of packets. This data is then exported to a collector using a protocol such as IPFIX [12] or NetFlow [11]. This data is especially valuable when traffic is encrypted because deep-packet inspection is no longer viable. The simplest type of analysis on flow data takes advantage of the IP address in the flow records and blacklists. This type of data fusion is widely deployed, but is fragile and difficult to maintain. Additional protocol-agnostic features have also been studied, such as the packet lengths and distribution of byte values [19].

This data has also been used to perform application or malware detection [18, 19, 30, 40, 41, 42, 43, 44]. BotFinder [38] is an important vertical correlation technique that uses only NetFlow features. Unsupervised machine learning is used to identify clusters that are characteristic of malware communications collected from malware sandboxes. It detects periodic components in malware communications, and achieved a detection rate of 0.8 with a 0.0001 false positive rate. Because BotFinder is content agnostic, it can operate on encrypted traffic. By utilizing more data features, our approach achieves a significantly higher detection rate, and is robust against evasion techniques in which the malware is not periodic and does not have communication characteristics that show up in NetFlow data. Additionally, our approach works even when each malware instance is only observed for a short, e.g., 5 minute, time window. This is an important strength, because large-scale, labeled malware datasets are often limited by sandbox resources. The periodic behavior exploited by BotFinder, in contrast, would not show up in these short windows.

Another important vertical technique is BotHunter [19]. This algorithm tracks the communication between internal assets and external entities, and develops an evidence trail of communication events that match a state-based infection sequence model. BotHunter recognizes the infection and coordination dialog that occurs during a malware infection. It is content-dependent, and uses both Snort signatures and anomaly detection based on content inspection, the latter utilizing a sophisticated n-gram based Statistical payLoad Anomaly Detection Engine (SLADE).

One important horizontal correlation technique is BotSniffer [20], which detects spatial-temporal correlations and similarity patterns in network traffic that are characteristic of botnet command and control traffic. Other interesting horizontal correlation techniques include BotMiner [18] and [37]. These techniques have the limitation that they cannot detect single infected hosts with high efficacy.

7.2 DNS

The Domain Name System (DNS) [28] is a hierarchical, decentralized means to provide additional information about domain names, notably a domain name to IP address mapping. More recently, malware has taken advantage of DNS and domain generation algorithms (DGA) [8] to provide a robust way to operate its command and control channels. There have been many previous results on classifying DNS data as malicious or benign [7, 9, 24]. None of this work leverages the DNS data to make inferences about encrypted traffic. Our work also differs in that we illustrate the distributions of different data features of DNS, e.g., TTL values.

7.3 HTTP

The Hypertext Transfer Protocol (HTTP) [17] is an application level protocol used to communicate data on the World Wide Web. Similar to DNS, HTTP has also been used by threat actors as a command and control channel [29, 33]. There has been some work specifically targeting features present in HTTP data. In [33], the authors use statistics, e.g., the average length of the URL, and string matching methods on the URL to cluster malware. Again, concrete differences in malware and benign HTTP sessions are not highlighted. [22] specifically analyzed `User-Agent` field values. We give a detailed description of more HTTP fields, and use this information to create machine learning classifiers for encrypted traffic.

7.4 TLS

Transport Layer Security (TLS) [13] is a cryptographic protocol that provides privacy for applications. TLS is usually implemented on top of common protocols such as HTTP for web browsing or SMTP for email. HTTPS is the usage of TLS over HTTP, which is the most popular way of securing communication between a web server and client and is supported by most major web servers. We have observed malware increasing its usage of TLS for the past 10 months, from ∼7% of samples with active network communications in June 2015 to ∼18% in April 2016.

The features of certificates used by malware have been studied [5], but this work does not examine benign certificates or their differences from the certificates used by malware. While there has been little work analyzing malware's use of TLS, work analyzing only the flow metadata would be equally effective on encrypted flows. Man in the Middle (MITM) solutions [10] have also been proposed to find threats in encrypted traffic. In MITM, an enterprise will decrypt network traffic passing through a security appliance and typically apply signatures to the unencrypted traffic. This method does not respect the privacy of the users on the network, it is not always effective, and it requires a lot of resources to effectively deploy and maintain. For these reasons, flow-based solutions are generally preferred.

8. CONCLUSIONS

Identifying threats in encrypted network traffic, and doing so in a way that does not compromise the integrity of the encryption, is an important problem. In this paper, we have shown that the data omnia approach, specifically collecting and correlating TLS, DNS, and HTTP metadata, can be used to accurately classify malicious, TLS network flows without resorting to crippling the TLS protocol.

We began by identifying features of the TLS protocol, as well as DNS and HTTP features of contextual flows, that have interesting discriminatory information. We showed how finding discrepancies in the unencrypted metadata can be used to find interesting indicators of compromise, such as

mismatched SNI and SAN fields. We presented an $l1$-logistic regression model that was able to achieve an accuracy of 99.978% at a 0.00% FDR. This model also offers easily interpretable results that generalized extremely well to a new dataset collected ∼4 weeks after the initial dataset.

There are several research directions that could extend and improve on our work. Longer-term behavior could be observed, making it possible to use an FFT data feature to detect periodic communication patterns [38]. Training data could be extended to include honeypots and malware observed in the wild. Utilizing additional content-aware data features, such as SLADE [19], could be useful on non-TLS flows that are encrypted at the application layer. Like all vertical correlation systems, it could be extended by hybridizing it with a horizontal correlation system, such as one that analyzes the communication graph [31].

9. ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their constructive feedback. We would also like to thank Greg Akers for supporting this work. Finally, we would like to thank Rich West, Dave Schwartzburg, Brandon Enright, Craig Brozefsky, and the ThreatGRID team for their direction, insightful comments, and support in facilitating the data collection.

10. REFERENCES

[1] Alexa. http://www.alexa.com/, 2016.

[2] ThreatGRID. http://www.threatgrid.com, 2016.

[3] Virus Total. https://www.virustotal.com/, 2016.

[4] W3C Extended Log File Format (IIS 6.0). https://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/676400bc-8969-4aa7-851a-9319490a9bbb, 2016.

[5] O. Alrawi and A. Mohaisen. Chains of Distrust: Towards Understanding Certificates Used for Signing Malicious Applications. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 451–456. International World Wide Web Conferences Steering Committee, 2016.

[6] B. Anderson, S. Paul, and D. McGrew. Deciphering Malware's use of TLS (without Decryption). *ArXiv e-prints*, July 2016.

[7] M. Antonakakis, R. Perdisci, W. Lee, N. Vasiloglou II, and D. Dagon. Detecting Malware Domains at the Upper DNS Hierarchy. In *USENIX security symposium*, page 16, 2011.

[8] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon. From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware. In *USENIX security symposium*, pages 491–506, 2012.

[9] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi. EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis. In *Network and Distributed System Security Symposium (NDSS)*, 2011.

[10] F. Callegati, W. Cerroni, and M. Ramilli. Man-in-the-Middle Attack to the HTTPS Protocol. *IEEE Security & Privacy*, 7(1):78–81, 2009.

[11] B. Claise. Cisco Systems NetFlow Services Export Version 9, 2013. RFC 3954.

[12] B. Claise, B. Trammell, and P. Aitken. Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information, 2013. RFC 7011.

[13] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. 2008. RFC 5246.

[14] T. G. Dietterich. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural computation*, 10(7), 1998.

[15] C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, and A. Roth. The Reusable Holdout: Preserving Validity in Adaptive Data Analysis. *Science*, 349(6248):636–638, 2015.

[16] D. Eastlake. Transport Layer Security (TLS) Extensions: Extension Definitions. 2011. RFC 6125.

[17] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. 1999. RFC 2616.

[18] G. Gu, R. Perdisci, J. Zhang, and W. Lee. BotMiner: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection. In *USENIX Security Symposium*, volume 5, pages 139–154, 2008.

[19] G. Gu, P. A. Porras, V. Yegneswaran, M. W. Fong, and W. Lee. Bothunter: Detecting Malware Infection through IDS-Driven Dialog Correlation. In *USENIX Security Symposium*, volume 7, pages 1–16, 2007.

[20] G. Gu, J. Zhang, and W. Lee. BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic. 2008.

[21] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin. The Elements of Statistical Learning: Data Mining, Inference and Prediction. 27(2):83–85, 2005.

[22] N. Kheir. Behavioral Classification and Detection of Malware through HTTP User Agent Anomalies. *Journal of Information Security and Applications*, 18(1):2–13, 2013.

[23] K. Koh, S.-J. Kim, and S. P. Boyd. An Interior-Point Method for Large-Scale l1-Regularized Logistic Regression. *JMLR*, 8(8):1519–1555, 2007.

[24] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1245–1254. ACM, 2009.

[25] D. McGrew and B. Anderson. Enhanced Telemetry for Encrypted Threat Analytics. In *ICNP Workshop on Machine Learning in Computer Networks (NetworkML)*. IEEE, 2016.

[26] D. McGrew and B. Anderson. Joy. https://github.com/davidmcgrew/joy, 2016.

[27] Microsoft. Choose the right ciphersuites in SChannel. https://www.ssl.com/how-to/choose-the-right-cipher-suites-in-schannel-dll/, 2016.

[28] P. Mockapetris. Domain Names - Concepts and Facilities. 1987. RFC 1034.

[29] A. Mohaisen. Towards Automatic and Lightweight Detection and Classification of Malicious Web Contents. In *Hot Topics in Web Systems and Technologies (HotWeb)*, pages 67–72. IEEE, 2015.

[30] A. W. Moore and D. Zuev. Internet Traffic Classification Using Bayesian Analysis Techniques. In *ACM SIGMETRICS Performance Evaluation Review*, volume 33, pages 50–60. ACM, 2005.

[31] S. Nagaraja, P. Mittal, C.-Y. Hong, M. Caesar, and N. Borisov. BotGrep: Finding P2P Bots with Structured Graph Analysis. In *USENIX Security Symposium*, pages 95–110, 2010.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *JMLR*, 12:2825–2830, 2011.

[33] R. Perdisci, W. Lee, and N. Feamster. Behavioral Clustering of HTTP-Based Malware and Signature Generation using Malicious Network Traces. In *NSDI*, pages 391–404, 2010.

[34] Qualys. Qualys SSL Labs. https://www.ssllabs.com/ssltest/clients.html, 2016.

[35] P. Saint-Andre and J. Hodges. Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS), 2011. RFC 6125.

[36] R. Sommer and V. Paxson. Outside the Closed World: On using Machine Learning for Network Intrusion Detection. In *2010 IEEE Symposium on Security and Privacy*, pages 305–316. IEEE, 2010.

[37] W. T. Strayer, R. Walsh, C. Livadas, and D. Lapsley. Detecting Botnets with Tight Command and Control. In *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 195–202. IEEE, 2006.

[38] F. Tegeler, X. Fu, G. Vigna, and C. Kruegel. Botfinder: Finding Bots in Network Traffic without Deep Packet Inspection. In *The 8th international conference on Emerging networking experiments and technologies*, pages 349–360. ACM, 2012.

[39] A. Vassilev. Annex A: Approved Security Functions for FIPS PUB 140-2, Security Requirements for Cryptographic Modules. http://csrc.nist.gov/publications/fips/fips140-2/fips1402annexa.pdf, 2016.

[40] K. Wang, G. Cretu, and S. J. Stolfo. Anomalous Payload-Based Worm Detection and Signature Generation. In *Recent Advances in Intrusion Detection*, pages 227–246. Springer, 2006.

[41] L. Wang, K. P. Dyer, A. Akella, T. Ristenpart, and T. Shrimpton. Seeing through Network-Protocol Obfuscation. In *22nd ACM Conference on Computer and Communications Security*, pages 57–69, 2015.

[42] N. Williams, S. Zander, and G. Armitage. A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification. *Computer Comm. Review*, 30, 2006.

[43] P. Wurzinger, L. Bilge, T. Holz, J. Goebel, C. Kruegel, and E. Kirda. Automatically Generating Models for Botnet Detection. In *Computer Security–ESORICS 2009*, pages 232–249. Springer, 2009.

[44] S. Zander, T. Nguyen, and G. Armitage. Automated Traffic Classification and Application Identification using Machine Learning. In *The 30th IEEE Conference on Local Computer Networks*, pages 250–257. IEEE, 2005.

APPENDIX

A. CIPHERSUITE AND EXTENSION HEX CODES

Hex Code	Ciphersuite
0x0004	TLS_RSA_WITH_RC4_128_MD5
0x0005	TLS_RSA_WITH_RC4_128_SHA
0x000a	TLS_RSA_WITH_3DES_EDE_CBC_SHA
0x0013	TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
0x0015	TLS_DHE_RSA_WITH_DES_CBC_SHA
0x002f	TLS_RSA_WITH_AES_128_CBC_SHA
0x0035	TLS_RSA_WITH_AES_256_CBC_SHA
0x0064	SSL_RSA_EXPORT1024_WITH_RC4_56_SHA
0xc007	TLS_ECDHE_ECDSA_WITH_RC4_128_SHA
0xc009	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
0xc00a	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
0xc013	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
0xc014	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
0xc02b	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
0xc02f	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

Table 4: Hex code to ciphersuite mapping for ciphersuites used in figures.

Hex Code	Ciphersuite
0x0000	server_name
0x0005	status_request
0x000a	supported_groups
0x000b	ec_point_formats
0x000d	signature_algorithms
0x000f	heartbeat
0x0015	padding
0x0017	extended_master_secret
0x0023	SessionTicket TLS
0x3374	next_protocol_negotiation
0xff01	renegotiation_info

Table 5: Hex code to extension mapping for extensions used in figures.