DoH Report

David Chitiz, Alon Perlmuter, Yishay Garame

September 12, 2021

Contents

1 Introduction

In order to understand this article in the best way, here are a few terms that are necessary to progress.

1.1 Terms

1.1.1 HTTP 1.1

HTTP 1.1 is a version of Hypertext Transfer Protocol (also known as HTTP) published in 1997, the World Wide Web application protocol that runs on top of the Internet's TCP/IP suite of protocols. HTTP 1.1 provides fast delivery for Web pages and reduces Web traffic. Here is a summary of how HTTP 1.1 makes information flow faster than the original:

• Instead of opening and closing a connection for each application request, HTTP 1.1 provides a persistent connection that allows multiple requests to be batched or pipelined to an output buffer. The underlying Transmission Control Protocol layer can put multiple requests (and responses to requests) into one TCP segment that gets forwarded to the Internet Protocol layer for packet transmission. Because the number of connection and disconnection requests for a sequence of "get a file" requests is reduced, fewer packets need to flow across the Internet. Since requests are pipelined, TCP segments are more efficient. The overall result is less Internet traffic and faster performance for the user. Persistent connection is similar to Netscape's HTTP 1.0 extension called KeepAlive but provides better handling of requests that go through proxy servers.

• When a browser supporting HTTP 1.1 indicates it can decompress HTML files, a server will compress them for transport across the Internet, providing a substantial aggregate savings in the amount of data that must be transmitted. (Image files are already in a compressed format, so this improvement applies only to HTML and other non-image data types.) In addition to persistent connections and other performance improvements, HTTP 1.1 also provides the ability to have multiple domain names share the same Internet address (IP address). This simplifies processing for Web servers that host several Web sites in what is sometimes called virtual hosting.

1.1.2 HTTP 2

HTTP/2 (originally named HTTP/2.0), published in 2015 as a revision of the HTTP protocol and was derived from the earlier experimental SPDY protocol, originally developed by Google. HTTP/2 provides an optimized transport for HTTP semantics. HTTP/2 supports all the core features of HTTP/1.1 but aims to be more efficient in several ways.

The basic protocol unit in HTTP/2 is a frame. Each frame type serves a different purpose. For example, HEADERS and DATA frames form the basis of HTTP requests and responses; other frame types like SETTINGS, WINDOW_UPDATE, and PUSH_PROMISE are used in support of other HTTP/2 features. Multiplexing of requests is achieved by having each HTTP request/response exchange associated with its own stream. Streams are largely independent of each other, so a blocked or stalled request or response does not prevent progress on other streams.

Flow control and prioritization ensure that it is possible to efficiently use multiplexed streams. Flow control helps to ensure that only data that can be used by a receiver is transmitted. Prioritization ensures that limited resources can be directed to the most important streams first. HTTP/2 adds a new interaction mode whereby a server can push responses to a client. Server push allows a server to speculatively send data to a client that the server anticipates the client will need, trading off some network usage against a potential latency gain.

The server does this by synthesizing a request, which it sends as a PUSH_PROMISE frame. The server is then able to send a response to the synthetic request on a separate stream. Because HTTP header fields used in a connection can contain large amounts of redundant data, frames that contain them are compressed. This has especially advantageous impact upon request sizes in the common case, allowing many requests to be compressed into one packet

1.1.3 DNS

The Domain Name System (known as DNS) is the phonebook of the Internet. Humans access information online through domain names, like facebook.com or google.com. Web browsers interact through Internet Protocol (IP) addresses. DNS translates domain names to IP addresses so browsers can load Internet resources. Each device connected to the Internet has a unique IP address which other machines use to find the device. DNS servers eliminate the need for humans to memorize IP addresses such as 192.168.1.1 (in IPv4), or more complex newer alphanumeric IP addresses such as 2400:cb00:2048:1:c629:d7a2 (in IPv6). The process of DNS resolution involves converting a hostname (such as www.example.com) into a computer-friendly IP address (such as 192.168.1.1). An IP address is given to each device on the Internet, and that address is necessary to find the appropriate Internet device - like a street address is used to find a particular home. When a user wants to load a webpage, a translation must occur between what a user types into their web browser (example.com) and the machine-friendly address necessary to locate the example.com webpage.

There are 4 DNS servers involved in loading a webpage:

- DNS recursor - The recursor can be thought of as a librarian who is asked to go find a particular book somewhere in a library. The DNS recursor is a server designed to receive queries from client machines through applications such as web browsers. Typically, the recursor is then responsible for making additional requests to satisfy the client's DNS query.
- Root nameserver - The root server is the first step in translating (resolving) human readable host names into IP addresses. It can be thought of like an index in a library that points to different racks of books - typically it serves as a reference to other more specific locations.

- TLD nameserver - The Top-Level Domain server (known as TLD) can be thought of as a specific rack of books in a library. This nameserver is the next step in the search for a specific IP address, and it hosts the last portion of a hostname (In example.com, the TLD server is "com").
- Authoritative nameserver - This final nameserver can be thought of as a dictionary on a rack of books, in which a specific name can be translated into its definition. The authoritative nameserver is the last stop in the nameserver query. If the authoritative name server has access to the requested record, it will return the IP address for the requested hostname back to the DNS Recursor (the librarian) that made the initial request.

1.1.4 TXT records

type of Domain Name System (DNS) record that contains text information for sources outside of your domain. You add these records to your domain settings. You can use TXT records for various purposes. Google uses them to verify domain ownership and to ensure email security.

1.1.5 DoH

is a protocol for performing remote Domain Name System (DNS) resolution via the HTTPS protocol. A goal of the method is to increase user privacy and security by preventing eavesdropping and manipulation of DNS data by using the HTTPS protocol to encrypt the data between the DoH client and the DoH-based DNS resolver. In other simple words, DoH grants privacy between two parties, meaning it is per-hop privacy. Your communication might be private between your web browsers and your ISP, but it may not be between your ISP and its upstream DNS server. Privacy may sound like a good idea for end users, but when used in a controlled environment such as corporate network, it may cause more concern than benefits. Running a rogue DoH client in a corporate setting means that the IT or security team is unable to look into the DNS queries the client makes, be it that the client is visiting a known malware-infected domain, or is using (encrypted) DNS to exfiltrate sensitive data out of the corporate network.

1.1.6 C&C

A command-and-control (also known as C&C) server is a computer controlled by an attacker or cybercriminal which is used to send commands to systems compromised by malware and receive stolen data from a target network. Many campaigns have been found using cloud-based services, such as webmail and file-sharing services, as C&C servers to blend in with normal traffic and avoid detection.

C&C servers are the headquarters or command centres where malware related to targeted attacks report back to so stolen data or download malicious commands can be stored. Establishing C&C communications is a vital step for attackers to move laterally inside a network.

C&C servers also serve as the headquarters for compromised machines in a botnet (a software array that exists on one or more computers connected to each other). It can be used to disseminate commands that can steal data, spread malware, disrupt web services, and more. C&C systems used by botnets may follow any of these three models: the centralized model, the peer-to-peer [P2P] model, and the random model. Aside from allowing attackers to steal data, the presence of C&C software on a server may also disrupt legitimate applications and cause the misuse of future resources.

1.1.7 Tunnelling

– In the physical world, tunnelling is a way to cross terrain or boundaries that could not normally be crossed. Similarly, in networking, tunnels are a method for transporting data across a network using protocols that are not supported by that network. Tunnelling works by encapsulating packets: wrapping packets inside of other packets. (Packets are small pieces of data that can be re-assembled at their destination into a larger file.) Tunnelling is often used in virtual private networks (VPNs). It can also set up efficient and secure connections between networks, enable the usage of unsupported network protocols, and in some cases allow users to bypass firewalls. Data traveling over a network is divided into packets. A typical packet has two parts: the header, which indicates the packet's destination and which protocol it uses, and the payload, which is the packet's actual contents.

An encapsulated packet is essentially a packet inside another packet. In an encapsulated packet, the header and payload of the first packet goes inside the payload section of the surrounding packet. The original packet itself becomes the payload.

All packets use networking protocols to get to their destinations. However, not all networks support all protocols. Imagine a company wants to set up a wide area network (WAN) connecting Office A and Office B. The company uses the IPv6 protocol, which is the latest version of the Internet Protocol (IP), but there is a network between Office A and Office B that only supports IPv4. By encapsulating their IPv6 packets inside IPv4 packets, the company can continue to use IPv6 while still sending data directly between the offices.

1.1.8 Flow Metadata

A group of sessions between two network addresses (IP pair) during the aggregation period. The aggregation period can be specified by an algorithm as the accurate period of time from the start of the first session in the flow, until the maximum idle time between two sessions. A new flow starts if the time between the end of a session (the last packet) and the start of a new session (first packet) is more than the defined idle time. The new session is then part of the new flow

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 or NetFlow. 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. This data has also been used to perform application or malware detection.

1.1.9 TLS

Transport Layer Security (known as TLS) is a cryptographic protocol that provides end-to-end security of data sent between applications over the Internet. It is mostly familiar to users through its use in secure web browsing, and in particular the padlock icon that appears in web browsers when a secure session is established. However, it can and indeed should also be used for other applications such as e-mail, file transfers, video/audioconferencing, instant messaging, and voice-over-IP, as well as Internet services such as DNS and NTP.

TLS evolved from Secure Socket Layers (SSL) which was originally developed by Netscape Communications Corporation in 1994 to secure web sessions. SSL 1.0 was never publicly released, whilst SSL 2.0 was quickly replaced by SSL 3.0 on which TLS is based.

TLS was first specified in 1999 as an applications independent protocol, and whilst was not directly interoperable with SSL 3.0, offered a fallback mode if necessary. However, SSL 3.0 is now considered insecure and was deprecated in 2015, with the recommendation that TLS 1.2 should be used. TLS 1.3 is also currently under development and will drop support for less secure algorithms.

It should be noted that TLS does not secure data on end systems. It simply ensures the secure delivery of data over the Internet, avoiding possible eavesdropping and/or alteration of the content. TLS is normally implemented on top of TCP in order to encrypt Application Layer protocols such as HTTP, FTP, SMTP and IMAP, although it can also be implemented on UDP, DCCP and SCTP as well (e.g. for VPN and SIP-based application uses).

1.1.10 Congestion control

Congestion control is a method used for monitoring the process of regulating the total amount of data entering the network so as to keep traffic levels at an acceptable value. This is done in order to avoid the telecommunication network reaching what is termed congestive collapse.

There's a few types to control this congestion for example there's the TCP method: TCP uses a congestion window and a congestion policy that avoid congestion.Previously, we assumed that only receiver can dictate the sender's window size. We ignored another entity here, the network. If the network cannot deliver the data as fast as it is created by the sender, it must tell the sender to slow down. In other words, in addition to the receiver, the network is a second entity that determines the size of the sender's window.

Congestion policy in TCP –

1. Slow Start Phase: starts slowly increment is exponential to threshold
2. Congestion Avoidance Phase: After reaching the threshold increment is by one
3. Congestion Detection Phase: Sender goes back to Slow start phase or Congestion avoidance phase

1.2 Tools & Libraries -To Be Continued

1.2.1 DNSExfiltrator

1.2.2 DoHC2

1.2.3 Godoh

2 Detection of DoH Tunnels using Time-series Classification of Encrypted Traffic

2.1 What is the problem the paper tried to solve and how?

This article tries to solve a very common security concern which is the abuse of DNS protocol in order to create hidden channels by tunneling data through DNS packets. It is well known that Doh makes DNS tunnels harder to detect. The first reason for this is that the DoH wraps the DNS traffic in HTTPS and therefore it transfers to the network without being notice and examining the DNS packets are impossible. Secondly, HTTP/2 is the minimum version that DoH recommends for use which allows the user to send a few DoH requests with only one connection.

The solution of the article to this problem is to classify between malicious DoH and harmless DoH using a two layered approach in a passive way which means without accessing the data itself.

2.2 Conclude the innovation of the paper

Although encrypted and DNS tunnelled traffic characterization has received large in the past due to its implications on traffic shaping, flow control and congestion control, DoH traffic characterization is still an evolving concept. The research presented in this paper attempts to detect and characterize DoH traffic in an online environment. The main advantage of the proposed method is that it needs small amount of traffic as input (less than 1 second) to competitively detect and characterize DoH traffic.

2.3 Did the authors cite additional important papers? If so, what does paper do?

The authors of the article cited approx. 30 articles which were divided to two categories: Encrypted Traffic Characterization and DNS Tunnelled Traffic Characterization.

In the first category that is mentioned, one of the outstanding articles is the one made by Zhang et al which used HTTP/2 based website fingerprinting technique to classify encrypted TLS traffic based on local request and response sequence feature set.

Another article which was mentioned a couple of times in this category was C. Patsakis, F. Casino, V. Katos, Encrypted and covert DNS queries for botnets: Challenges and countermeasures, Computers & Security, 88, 2020. They investigated the use of Domain Generation Algorithms to hide malicious DNS queries in a covert channel. One of the variables which helps detect DNS tunnels was TXT records.

In the second category DNS Tunnelled Traffic Characterization, a few important articles were mentioned as well. One of them was by Al-kasassbeh, Mouhammd, and Tariq Khairallah , Winning tactics with DNS tunnelling, Network Security, Vol 12, 2019 which is mainly about packet-based method of classification which is gaining importance since 2019. Moreover, Mouhammd and Tariq presented a thorough review of available tools developed specifically for DNS tunneling and other tools that have other primary purposes but utilise DNS tunneling as a data exfiltration method. Details of experimentation with Iodine DNS tunnels have been given and it was shown how DNS tunneling can encapsulate other protocols such as SSH

2.4 What are the learning features? What are the advantages of those features?

The learning features were divided to three categories,

1. Time-series features
2. Payload inspection features
3. Statistical features

Advantages and a brief explanation of each category:

Time-series features - Used to model the network according to the nature of the traffic flow which is encrypted by TLS protocol in the form of a series of packets transmitted over the period. The nature of network traffic is a series of packets transmitting over time, so we can model a network flow using a time-series representation of captured traffic. Since the traffic used in this work is encrypted by TLS protocol, the payloads(data) on the packets would not leak any useful information about the nature of underlying traffic however, we can use other traffic shape parameters such as the packet size, packet direction, and the time difference between packets, to infer some information about the underlying traffic. They defined a Clump C which has five features as follows:

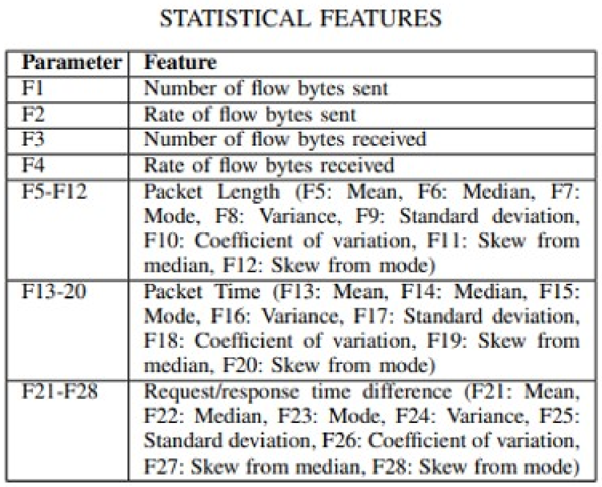
$$C = \langle size, pktCount, direction, duration, interarrival \rangle$$

$$S = (C_1, \ldots, C_n)$$

$$F_\ell = \{(C_i, ..., C_{i+\ell}) \mid 1 \leq i < |S| - \ell\}$$

Payload inspection features – Due to the fact that TLS traffic is encrypted, we can only try and access the handshake packets. Just by inspecting the initial query, we can identify a shape or a fingerprint to help classify the data according to the article of J. Yang et al, Bayesian Neural Network Based Encrypted Traffic Classification using Initial Handshake Packets. Although this fingerprinting method can be used to classify flows by clients, it is not helpful for detecting malicious tunneling traffic because it is limited to known client fingerprints only.

Statistical features – Based on statistical features (which considered very useful for performing an analysis) of L. Arash et al, Characterization of Tor Traffic Using Time-Based Features & the study of D. Gerard et al, Characterization of Encrypted and VPN Traffic Using Time-Related Features 28 features were selected to create the flow of DoH.



STATISTICAL FEATURES

Parameter	Feature
F1	Number of flow bytes sent
F2	Rate of flow bytes sent
F3	Number of flow bytes received
F4	Rate of flow bytes received
F5-F12	Packet Length (F5: Mean, F6: Median, F7: Mode, F8: Variance, F9: Standard deviation, F10: Coefficient of variation, F11: Skew from median, F12: Skew from mode)
F13-20	Packet Time (F13: Mean, F14: Median, F15: Mode, F16: Variance, F17: Standard deviation, F18: Coefficient of variation, F19: Skew from median, F20: Skew from mode)
F21-F28	Request/response time difference (F21: Mean, F22: Median, F23: Mode, F24: Variance, F25: Standard deviation, F26: Coefficient of variation, F27: Skew from median, F28: Skew from mode)

2.5 What are the paper data sources? How they were created? Can you access the dataset /reproduce it?

The data set is public at https://www.unb.ca/cic/datasets/dohbrw-2020.html and may be downloaded.

To generate the representative dataset, HTTPS (benign DoH and non-DoH) and DoH traffic is generated by accessing top 10k Alexa websites, and using browsers and DNS tunneling tools that support DoH protocol respectively.

The tools that assist to capture the data set was made by python and Selenium library to communicate with web browsers. They accessed Firefox by using GeckoDriver which is an intermediary between Firefox and tools that interacts with Firefox.

To create the malicious records of the dataset they deployed a network that can be used to simulate DoH tunneling scenarios. This is achieved by setting up a domain, setting an authoritative server as the nameserver for that domain, and using DoH requests/responses to carry data between client and server.

Since the DoH protocol is compatible with DNS authoritative nameservers, there was a usage of existing DNS tunneling tools to setup the authoritative server. For the client, they used a DoH proxy to encapsulate every DNS requests received by the client into a DoH connection with a DoH Server and relay the response it received from the DoH server to the client as DNS records. Data is captured between the DoH proxy and DoH server. This set up allows to simulate DoH tunnels using various DNS tunneling tools that are used in the relevant studies [30]. To label the training dataset, these simulations were done separately in an isolated network so that any traffic captured can be associated with the correct label. To generate enough data, the clients used in the simulation were run simultaneously on 10 servers.

2.6 What AI learning algorithms were used?

For classifying the data, the AI learning algorithms that were used are;

- Random Forest (RF)
- Decision Tree
- Support Vector Machines (SVM)
- Navie Bayes (NB)
- Deep Neural Network (DNN)
- Convolutional Neural Network (CNN)

3 Theoretical Background

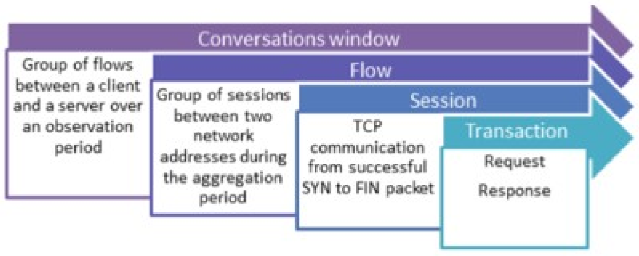
Following are summaries of several articles with high relevance to the project.

3.1 Unknown Malware Detection Using Network Traffic Classification by Dmitri Bekerman, Bracha Shapira, Lior Rokach, Ariel Bar

There is an issue of malware programs which are capable of hiding themselves in systems or disabling their activity as soon as they discover attempts to detect them. This article presents a passive system for detecting malware by analyzing network traffic without accessing them. This system is based on cross-layers and cross protocols traffic classification, using supervised learning method. The proposed method extracts many features from different protocols and network layers and the most meaningful features are selected in order to reduce the data so it can be tractable as of a its reduced size.

Through analyzing DNS, HTTP and SSL protocols the method gets a better understanding of the difference between normal network traffic and malicious activities.

This solution is unique because the system observes data stream analysis in four resolutions, based on Internet transport and Application layers as follows in this data stream observation module:


Conversations window
Group of flows between a client and a server over an observation period
Flow
Group of sessions between two network addresses during the aggregation period
Session
TCP communication from successful SYN to FIN packet
Transaction
Request
Response

The results of this study are very impressive, by learning from one environment they managed to classify a different environment and receiving accurate results with detecting malicious traffic as appose to benign traffic. Overall, this study managed to improve the time it takes to detect malware in addition of detecting unknown malware due to its passive system.

The practical significance of this research is the opportunity to develop highly accurate and good performing network intrusion detection system (NIDS) which apply machine learning algorithms and can detect most modern malicious software as well as new and unknown malware. Since the proposed method analyses only traffic behaviour and not its content, it is effective even for encrypted traffic and for malware that uses legitimate network resources, such as C&C or proxy toward C&C.

3.2 Identifying Encrypted Malware Traffic with Contextual Flow Data

This article tries to monitor malicious traffic in a passive way using the omnia approach. Due to the difficulties of accessing encrypted data, this research extends the usage of the metadata collected from the flow (packet length, inter-arrival times etc.) and adds unique features such as; TLS handshake, 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 by using a machine learning model.

The research starts with by exhibiting the differences between malicious and benign traffic's use of TLS, DNS, and HTTP on millions of unique flows in order to find the best feature sets that will give us the best distinction between benign and malware flow. Their machine learning model has a 0.00% false discovery rate for this classification, a tremendous success.

This paper contributes us in two major aspects, firstly, providing the first results that leverage contextual information, i.e., DNS responses and HTTP headers, to identify threats in encrypted traffic. Secondly, they demonstrate highly accurate machine learning algorithms which with validation proves that's its not due to an overfitting model.

Eventually, going through the protocol's metadata, the logistic regression model achieved an accuracy of 99.978%. for future work, longer-term behaviour could be observed, making it possible to use an FFT data feature to detect periodic communication patterns

3.3 Encrypted Malware Traffic Detection Using Incremental Learning

Many attackers use TLS protocol to avoid detection which makes detecting threats from the encrypted traffic a hard task.

In this paper they use an encrypted malware traffic detection method that use machine learning that updates itself all the time and keeps a high-performance level. This method uses features from DNS TLS and HTTP protocols and the results show that using incremental Support Vector Machine with Stochastic Gradient Descent algorithm is suitable for the detection method amongst three algorithms, by off-line and on-line accuracy at a low false discovery rate.

Old methods like deep package and port based are not suitable for identify threats in the encrypted traffic as they have no access through TLS protocol.

In addition, this paper proves that incremental learning algorithm is more suitable than batch learning algorithm because the batch method requires re-training the previously trained data. This characteristic makes incremental learning is more proper for easy model updates. In this paper they evaluate three incremental algorithms by the accuracy with a false discovery rate of 0.001% and consider intermediate models to provide a deeper insight.

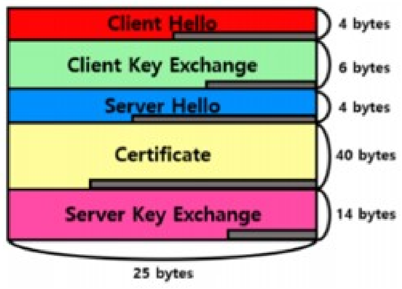
Finally, in their work they showed TLS-encrypted malware traffic detection method, that uses incremental learning and trains 31 contextual flow features.

In addition, the results of the research showed that the incremental SGD-based method with accuracy of 0.001% FDR in off-line and on-line effectively detect encrypted anomalies.

3.4 Feasibility of Malware Traffic Analysis

As of a growing rate of malware's use of TLS this paper proposes a visualize TLS-encrypted flow metadata as an image for better malware traffic analysis and classification.

recent studies [1], [2], show that malware's use of TLS is also rapidly increasing. This trend of TLS-encrypted flows is problematic in threat detection since traditional approaches such as deep packet inspection and signatures have limited effectiveness for encrypted traffic. The article suggests a rough feature selection approach for TLS flow metadata called TLS-encrypted flow visualization which is useful for malware traffic analysis. The first stage is to classify the malware from the TLS flow metadata at the beginning of the connections. Then, an assigment of colors to different messages for better visualization.



Last, generating images according to the metadata TLS flow and analyzing and classifying the images to malware and benign using CNN & SVM (machine learning classifies). To conclude, different malware families have different patterns in their images. In addition, malware family classification via images using SVM and CNN models have high accuracy. In future work, they will discuss why such high accuracy could be possible.

[1] L. Nagy. Nearly a quarter of malware now communicates using TLS. [Online]. Avail-able: https://news.sophos.com/en-us/2020/02/18/nearly-a-quarter-of-malware-now-communicates-using-tls [2] B. Anderson and D. McGrew, "Identifying encrypted malware traffic with contextual flow data," in Proc. of AISec'16, Vienna,

3.5 Encrypted HTTP/2 Traffic Monitoring: Standing the Test of Time and Space

Http/2 has been adopted in the last years and its goal is to keep user's privacy against traffic interception.

It's been proven that machine learning combined with some features are able to identify the incriminated traffic even when it is encrypted with h2. The disadvantage of this method is that with encrypted malware traffic, user's privacy are being disturbed so it's important that security practitioner will understand the efficiency of such a technique and its limits.

This paper addresses these challenges by defining an experimental methodology applied on more than 3000 different websites over four months continuously. The results show that this method is applicable to many websites but needs weekly training to keep the model accurate.

The robustness of a classification model is always questionable and is mainly impacted by two main factors:

1. The traffic generated for a same request can change over time and thus impacting the validity of models learned.
2. The generalization of a model can only be validated thanks to a large number of different instances to be tested.

In this technique they use Random Forest (RF) algorithm and they explain that by set of services S and by learning their actions, their methods will automatically learn a classification model related to particular actions defined a priority.

In conclusion, although H2Classifier has been specifically designed to monitor user activities in h2-based services (HTTP2 over TLS), the designed methodology and outcomes(results and discussions) are helpful for other encrypted traffic classification or fingerprinting techniques leveraging machine learning and is relevant for many websites.

3.6 Ransomware Network Traffic Analysis for Pre-Encryption Alert

The motivation of this research is to join the malware battle and providing more knowledge to help victims against ransomware. This article's goal is also to distinguish between benign and malicious traffic however, Instead of a real time solution, the work here is more similar to a research where they are trying to understand if ransom notes and encrypted data may be detected at a certain time before the start of the encryption.

The mechanism which helps evaluating the paper's goal is divided to three parts where the first part is data collection using Wireshark and Process Monitor for PCAP & PML respectively. The second part is the filtering of the data and session reconstructing and the last part is the analysis and the development of the model.

The experiments prove that machine learning classifiers are able to flag ransomware network traffic for both UDP and TCP records as in signature-based detection. In addition, the research found that the majority of ransomware behave in similar ways however they have evolved and don't only communicate via non encrypted HTTP traffic (TCP requests), they (ransomware) use GET requests and TLS protocol for ccommunication. There are some drawbacks with detecting the ransomware, in order to solve those issues, they recommend backing up Network Alerts with system data. In the future, the researches will gather additional information from the file system to present a multi-layer alert strategy to detect ransomware's payload as early as possible.

3.7 Detection of Encrypted Malicious Network Traffic using Machine Learning

The proliferation of encrypted network traffic necessitates an innovative machine learning traffic analysis approach which does not rely on pattern matching or the payload content of the packets to detect malicious / suspicious communications. Encryption of Internet traffic has increasingly become a typical best practice, making network packet content analysis yield diminishing returns. A majority of internet traffic is now protected using TLS protocol.

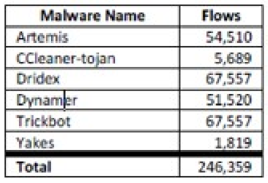
Malware authors have also followed this trend with the use of TLS to hide malicious network communications. This article proposes a malicious communication detection mechanism using a Support Vector Machine (SVM) and an alternative with a Convolutional Neural Network (CNN). Both methods achieve respectable results and a low False Positive Rate (FPR). However, the SVM method outperforms the CNN method in all evaluation metrics presented.

The contributions of this paper are as follows:

- An advancement of their previous method in "Identifying and detecting applications within TLS traffic" to include the application of an SVM, hash vector, and the frequency of TLS record sizes and direction for detection of TLS encrypted malware using a public dataset.
- An application of a one-dimensional CNN and the sequence of TLS record size and direction for detection of TLS encrypted malware using a public dataset.
- A comparison between the performance of the developed SVM and CNN classifiers. Dataset used: Malware Capture Facility Project, S. Garcia, 2016-2018. [Online]. Available: https://stratosphereips.org [NEED TO BE IN

REFERNCES]

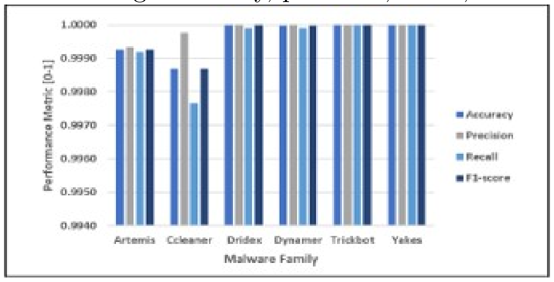
Dataset malware family and flows:



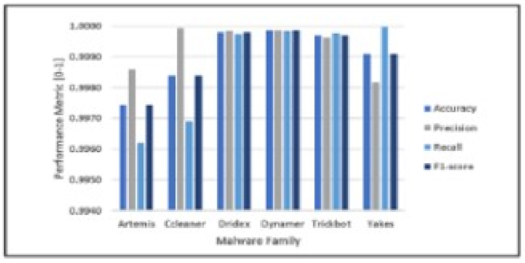
Malware Name	Flows
Artemis	54,510
CCleaner-tojan	5,689
Dridex	67,557
Dynamer	51,520
Trickbot	67,557
Yakes	1,819
Total	246,359

Results:

SVM average accuracy, precision, recall, and f1-score per malware family:



CNN average accuracy, precision, recall, and f1-score per malware family:


Performance Metric [0-1]
Artemis
Ccleaner
Dridex
Dynamer
Trickbot
Yakes
Malware Family
■ Accuracy
■ Precision
■ Recall
■ F1-score