A Machine Learning Based Approach to Detect Stealthy Cobalt Strike C&C Activities from Encrypted Network Traffic

Fabian Martin Ramos[1,2] and Xinyuan Wang[1]

[1] George Mason University, Fairfax, VA 22030, USA
fmartinr@gmu.edu; xwangc@gmu.edu
[2] ETSI Telecomunicación, Universidad Politécnica de Madrid, 28040 Madrid, Spain

Abstract. Cobalt Strike is a stealthy and powerful command and control (C&C) framework that has been widely used in many recent massive data breach attacks (e.g., the SolarWinds attack in 2020) and ransomware attacks. While detecting Cobalt Strike C&C network traffic is crucial to the protection our mission critical systems from many sophisticated cyberattacks, no existing intrusion detection systems have been shown to be able to reliably detect real world Cobalt Strike C&C traffic from encrypted traffic.
In this paper, we propose a machine learning based approach to detect stealthy Cobalt Strike C&C traffic. Based on the analysis of real world Cobalt Strike traffic, we have developed an approach using flow-level features that capture the inherent characteristics of Cobalt Strike C&C traffic. We have validated our machine learning based detection with five machine learning algorithms and evaluated them with Cobalt Strike traffic from real world cyberattacks. Our empirical results demonstrate that our random forest model can detect close to 50% of real world Cobalt Strike C&C traces in encrypted traffic with a 1.4% false positive rate.

Keywords: Cobalt Strike C&C · Intrusion Detection · Machine Learning

1 Introduction

As our society is increasingly dependent on digital technologies, cyberattacks have become a more serious threat to mission critical systems and infrastructures. For example, recent massive data breach attacks [1] on various business organizations and government agencies (e.g., Target, JP Morgan Chase, OPM, Anthem Inc., Equifax) have impacted tens or hundreds of millions of people. Now cybercriminals can launch sophisticated attacks and compromise mission critical systems via the Internet from virtually anywhere in the world. By using covert command & control (C&C) channels, cybercriminals can stealthily control the compromised systems from thousands of miles away and exfiltrate sensitive data for months. This type of sophisticated cyberattacks are called advanced persistent threats (APT) and they have caused prohibitive financial

losses to many businesses in recent times. According to the 2019 Cost of a Data Breach Report [2], the average damage cost of a data breach of 50 million records is $388 million. Specifically, the 2017 Equifax data breach has costed Equifax nearly $1.4 billion as of May 2019 [3]. Cybersecurity Ventures predicted that the annual global cybercrime damage would grow from $3 trillion in 2015 to $10.5 trillion by 2025 [4].

Cobalt Strike has been widely used in recent sophisticated cyberattacks [5, 6] due to its ability to establish stealthy C&C channels between the victim system and the attacker. According to Cisco Talos Incident Response (CTIR) Quarterly Report [7], ransomware has been the dominating threat in 2020, and 66% of all ransomware attacks in summer 2020 used Cobalt Strike. The 2020 SolarWinds supply chain attack [8] delivered a customized Cobalt Strike payload to 18,000 Orion customers that included many Fortune 500 organizations (e.g., Microsoft, Cisco) and various US government agencies (e.g., DoD, DHS, DOJ). Cobalt Strike stealthy in-memory persistence and C&C channels enabled the SolarWinds attack to surreptitiously explore, identify and exfiltrate highly sensitive information from some of the most secured information systems without being detected for over nine months. Specifically, the SolarWinds attacker gained access to and exfiltrated the emails of the highest-ranking officials in the Treasure Department [9] and stole gigabytes of source code of Microsoft, Cisco, SolarWinds and FireEye [10].

Detecting Cobalt Strike C&C channel is crucial to the protection of our mission critical cyber systems and infrastructures from many sophisticated cyberattacks. However, Cobalt Strike C&C is very stealthy and hard to detect. Specifically, Cobalt Strike C&C traffic is fully encrypted and can mimic legitimate network traffic using communication protocols such as HTTPS, which will defeat any content based detection. An independent study sponsored by IBM Security [2] shows that it took average 206 days to detect a data breach in 2019. DHS CISA spokesperson Sara Sendek acknowledged that none of deployed intrusion detection or prevention systems (IDS/IPS) – including the U.S. government's multi-billion dollar detection system, Einstein, was able to detect the 2020 SolarWinds breach [11]. In other words, no existing IDS/IPS was able to detect the stealthy Cobalt Strike C&C activities involved in the 2020 SolarWinds attack. To the best of our knowledge, there is no published result on reliable detection of real world stealthy Cobalt Strike C&C activities from encrypted traffic.

In this paper, we explore a new direction in detecting stealthy Cobalt Strike C&C activities from encrypted traffic. Instead of using the packet flow content information, we build our machine learning based detection upon the packet timing and flow duration information that are not changed by encryption. Based on analysis of real world Cobalt Strike C&C traces, we have generated machine learning features that capture the inherent flow level characteristics of encrypted Cobalt Strike C&C traffic. We have empirically validated our machine learning based detection with five popular machine learning models and real world encrypted Cobalt Strike C&C traffic mixed with normal traffic. Our empirical

results demonstrate that our our random forest model can detect close to 50% of real world Cobalt Strike C&C traces in encrypted traffic with a 1.4% false positive rate. Our naïve Bayes model achieves over 84% detection true positive rate with 13% false positive rate.

The rest of the paper is structured as follows: section 2 describes and analyze the characteristics of the Cobalt Strike framework and its C&C traffic. Section 3 details the threat model, flow-based features, machine learning models and metrics used in the generation of the model. Section 4 presents the experimental results using real world Cobalt Strike C&C traffic. Section 5 reviews related works. Finally, section 6 concludes the paper with potential future work directions.

2 Analysis of Cobalt Strike C&C

2.1 Cobalt Strike

Cobalt Strike is a powerful and stealthy command and control (C&C) framework that was originally developed by Raphael Smudge as a penetration testing tool in 2012 [12]. Cobalt Strike allows attackers to 1) perform target reconnaissance by identifying known vulnerabilities in software versions; 2) create trojans and malicious website clones that enable drive-by attacks; 3) deploy and inject malicious agents called Beacons into vulnerable targets; 4) perform tasks in the systems infected with a Beacon, such as log keystrokes, take screenshots, execute commands, download additional malware or inject a Beacon in other processes; and 5) disguise its C&C traffic using encryption and mimic normal network traffic using communication protocols such as HTTP, HTTPS or DNS to surpass cybersecurity defenses. Due to its post-exploitation and stealthy C&C features, Cobalt Strike has been widely used in sophisticated cyberattacks.

Cobalt Strike has two main components: the team server and the client. The team server is the C&C server that interacts with a victim that has been infected with a Beacon, and it also accepts client connections. The client is the system used by the attacker to interact with the team server to send commands to the Beacons. Cobalt Strike Beacon is the malware payload used by Cobalt Strike to create a backdoor on a victim system that connects to the team server and can be divided into two parts: the payload stage, and the payload stager. The payload stager is a smaller program that is used to download the payload stage on to a system, inject it into memory, and pass the execution to it. The payload stage is the actual backdoor that runs in memory and can establish a connection to the C&C server through different channels. Cobalt Strike contains many additional components [6], which allow it to be configured to bypass defense systems:

1. Listeners: the listeners define how the Beacon connects to the team server, such as the IP address of the C&C server, the ports and the protocol used. Cobalt Strike supports a great variety of protocols: HTTP, HTTPS and DNS are the most popular ones, while also supporting SMB, raw TCP, foreign listeners (using Metasploit's Meterpeter) and external C&C listeners.

2. Arsenal Kit: these kits allow additional customization into Cobalt Strike capabilities to evade antivirus products. Some of the most popular kits are:
   - Artifact kit: allows attackers to modify the template for all Cobalt Strike executables, DLLs and shellcode.
   - Elevate kit: allows attackers to include third-party privilege escalation scripts with Cobalt Strike Beacon.
   - Mimikatz kit: allows attackers to use and update the Mimikatz installation included with Cobalt Strike.
3. Malleable C&C profiles: part of the Arsenal Kit, it allows the attacker to customize the communications between the Beacon and the team server as well as the Beacon in-memory characteristics, determine how it does process injection, and influence post-exploitation jobs.

The Malleable C&C profile´s ability to customize the network traffic that the Beacon generates and receives, such as the interval between each Beacon callback to the team server, the URI of the HTTP/S requests, inserting additional data to masquerade the actual data payload size and more, is the main characteristic that makes Cobalt Strike such a powerful tool for penetration testers and malicious agents alike. It allows the attacker to blend the C&C traffic with the normal traffic, bypassing security defenses such as network firewalls and intrusion detection systems.

Cobalt Strike has become one of the favorite tools for attackers of all skill levels, from script kiddies to state-sponsored attackers, being used among other malware such as QBot or Emotet and phishing attacks or being actively used for all the phases of the attack lifecycle. Cobalt Strike has also been identified not only in ransomware attacks, but also part of the famous SolarWinds supply chain attack in 2020 and cyberespionage campaigns targeting the Ukrainian population.

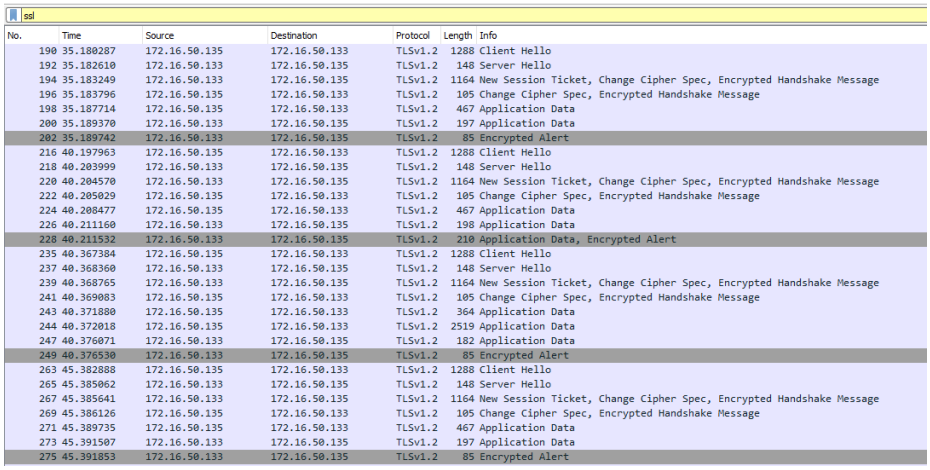
2.2 Cobalt Strike C&C communications

This paper focuses on the detection of the two most popular application level protocols used by attackers for Cobalt Strike C&C communications: HTTP and HTTPS. Per the Cobalt Strike documentation, a typical Cobalt Strike C&C HTTP/S transaction between an infected host and the C&C server proceeds as follows:

- First, the TCP connection is established via the TCP three-way handshake (SYN, SYN/ACK, ACK).
- If the HTTPS protocol is used, the TLS handshake is performed (Client Hello, Server Hello, Server Key Exchange, Client Key Exchange, Finished) to establish the HTTPS session.
- Then, the exchange of data takes place between the victim and the team server.
- Finally, the HTTPS session and TCP connection are closed.

The Beacon performs callbacks to the C&C server periodically to retrieve tasks to execute, sending the information of the infected system in a GET request after the connection has been established. The server responds to the request with the tasks, if any have been instructed by the attacker. The Beacon then executes the tasks and initiates another connection to send the output back to the C&C server using a POST request. The server then responds with data that is discarded by the Beacon. The packet payload is encoded and encrypted to complicate the analysis of its contents by security systems. Cobalt Strike uses this process to simulate a legitimate HTTP/S exchange between a client and a server.

Figure 1 shows several HTTPS transactions between a Beacon-infected host and a team server from a packet capture using the network protocol analyzer tool Wireshark. In the image, it is possible to observe that, after establishing the HTTPS session through the TLS Handshake, the victim and the C&C server exchange "Application Data" packets which are used to transmit the tasks that are to be executed to the Beacon or send the results from those tasks back to the C&C server. Once the data has been exchanged, the C&C server closes the HTTPS session using an "Encrypted Alert" packet. Each connection between the victim and the team server requires the client to open a new HTTPS session with the server.

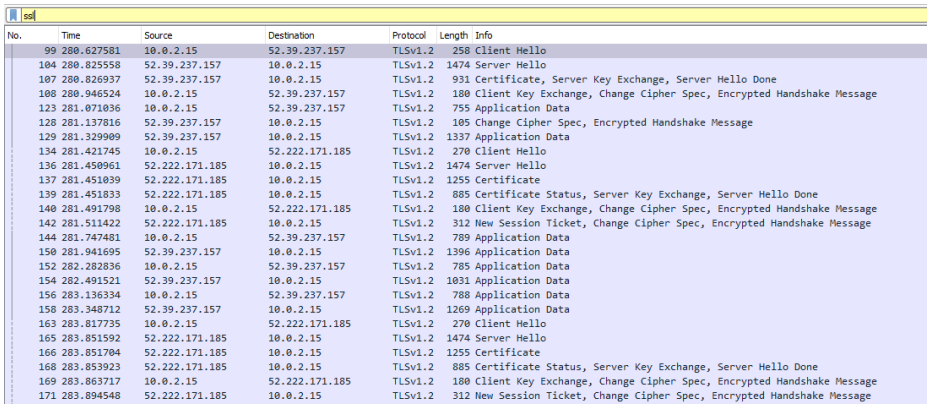

ssl

No.	Time	Source	Destination	Protocol	Length	Info
190	35.180287	172.16.50.135	172.16.50.133	TLSv1.2	1288	Client Hello
192	35.182610	172.16.50.133	172.16.50.135	TLSv1.2	148	Server Hello
194	35.183249	172.16.50.133	172.16.50.135	TLSv1.2	1164	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
196	35.183796	172.16.50.135	172.16.50.133	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
198	35.187714	172.16.50.135	172.16.50.133	TLSv1.2	467	Application Data
200	35.189370	172.16.50.133	172.16.50.135	TLSv1.2	197	Application Data
202	35.189742	172.16.50.133	172.16.50.135	TLSv1.2	85	Encrypted Alert
216	40.197963	172.16.50.135	172.16.50.133	TLSv1.2	1288	Client Hello
218	40.203999	172.16.50.133	172.16.50.135	TLSv1.2	148	Server Hello
220	40.204570	172.16.50.133	172.16.50.135	TLSv1.2	1164	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
222	40.205029	172.16.50.135	172.16.50.133	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
224	40.208477	172.16.50.135	172.16.50.133	TLSv1.2	467	Application Data
226	40.211160	172.16.50.133	172.16.50.135	TLSv1.2	198	Application Data
228	40.211532	172.16.50.133	172.16.50.135	TLSv1.2	210	Application Data, Encrypted Alert
235	40.367384	172.16.50.135	172.16.50.133	TLSv1.2	1288	Client Hello
237	40.368360	172.16.50.133	172.16.50.135	TLSv1.2	148	Server Hello
239	40.368765	172.16.50.133	172.16.50.135	TLSv1.2	1164	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
241	40.369083	172.16.50.135	172.16.50.133	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
243	40.371880	172.16.50.135	172.16.50.133	TLSv1.2	364	Application Data
244	40.372018	172.16.50.135	172.16.50.133	TLSv1.2	2519	Application Data
247	40.376071	172.16.50.133	172.16.50.135	TLSv1.2	182	Application Data
249	40.376530	172.16.50.133	172.16.50.135	TLSv1.2	85	Encrypted Alert
263	45.382888	172.16.50.135	172.16.50.133	TLSv1.2	1288	Client Hello
265	45.385062	172.16.50.133	172.16.50.135	TLSv1.2	148	Server Hello
267	45.385641	172.16.50.133	172.16.50.135	TLSv1.2	1164	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
269	45.386126	172.16.50.135	172.16.50.133	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
271	45.389735	172.16.50.135	172.16.50.133	TLSv1.2	467	Application Data
273	45.391507	172.16.50.133	172.16.50.135	TLSv1.2	197	Application Data
275	45.391853	172.16.50.133	172.16.50.135	TLSv1.2	85	Encrypted Alert

Fig. 1. Packet capture from HTTPS Beacon transaction

Comparing the previous figure with Figure 2, which shows a packet capture of legitimate HTTPS traffic when accessing a web pages such as Facebook (www.facebook.com), we can observe some notable differences between legitimate traffic and Beacon traffic.

First, the Beacon does not use the same TCP connection to contact the C&C server more than once. This event occurs for both the HTTP and HTTPS



ssl

No.	Time	Source	Destination	Protocol	Length	Info
99	280.627581	10.0.2.15	52.39.237.157	TLSv1.2	258	Client Hello
104	280.825558	52.39.237.157	10.0.2.15	TLSv1.2	1474	Server Hello
107	280.826937	52.39.237.157	10.0.2.15	TLSv1.2	931	Certificate, Server Key Exchange, Server Hello Done
108	280.946524	10.0.2.15	52.39.237.157	TLSv1.2	180	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
123	281.071036	10.0.2.15	52.39.237.157	TLSv1.2	755	Application Data
128	281.137816	52.39.237.157	10.0.2.15	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
129	281.329909	52.39.237.157	10.0.2.15	TLSv1.2	1337	Application Data
134	281.421745	10.0.2.15	52.222.171.185	TLSv1.2	270	Client Hello
136	281.450961	52.222.171.185	10.0.2.15	TLSv1.2	1474	Server Hello
137	281.451039	52.222.171.185	10.0.2.15	TLSv1.2	1255	Certificate
139	281.451833	52.222.171.185	10.0.2.15	TLSv1.2	885	Certificate Status, Server Key Exchange, Server Hello Done
140	281.491798	10.0.2.15	52.222.171.185	TLSv1.2	180	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
142	281.511422	52.222.171.185	10.0.2.15	TLSv1.2	312	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
144	281.747481	10.0.2.15	52.39.237.157	TLSv1.2	789	Application Data
150	281.941695	52.39.237.157	10.0.2.15	TLSv1.2	1396	Application Data
152	282.282836	10.0.2.15	52.39.237.157	TLSv1.2	785	Application Data
154	282.491521	52.39.237.157	10.0.2.15	TLSv1.2	1031	Application Data
156	283.136334	10.0.2.15	52.39.237.157	TLSv1.2	788	Application Data
158	283.348712	52.39.237.157	10.0.2.15	TLSv1.2	1269	Application Data
163	283.817735	10.0.2.15	52.222.171.185	TLSv1.2	270	Client Hello
165	283.851592	52.222.171.185	10.0.2.15	TLSv1.2	1474	Server Hello
166	283.851704	52.222.171.185	10.0.2.15	TLSv1.2	1255	Certificate
168	283.853923	52.222.171.185	10.0.2.15	TLSv1.2	885	Certificate Status, Server Key Exchange, Server Hello Done
169	283.863717	10.0.2.15	52.222.171.185	TLSv1.2	180	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
171	283.894548	52.222.171.185	10.0.2.15	TLSv1.2	312	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message

Fig. 2. Captured HTTPS Packets from Normal Traffic

Beacons, but it can be more clearly observed in the case of the HTTPS Beacon: the C&C server sends an "Encrypted Alert" packet indicating that the HTTPS session is closed, and consequently also closes the TCP connection shortly after that packet is sent. Therefore, as we can also observe in Figure 1, the Beacon opens a TCP connection for every call to the C&C server and closes it after retrieving the required tasks or sending the information back.

Thus, we can be certain that, unless more than one Beacon is used to infect a system and the Beacons are communicating with the same server simultaneously, it is not possible to observe the case of two consecutive TCP connections being made from the victim to the server without the previous one being closed, which is a common occurrence in legitimate traffic to have multiple TCP connections simultaneously between the client and the server.

Furthermore, the duration of a TCP connection between a legitimate client and a server tends to be longer than a Beacon connection. In a Beacon transaction only a few packets are exchanged, especially when the transaction is the Beacon receiving the tasks. The exfiltration of data from the infected system can also be used to detect the Beacon traffic, since the legitimate traffic rarely requires the client to send great quantities of data to the server.

2.3 Malleable C&C profiles

Cobalt Strike introduces the use of Malleable C&C profiles to customize the network indicators of the C&C traffic between the Beacon and the team server as an evasion measure, as they can be used to disguise the Beacon traffic to look like other malware or blend-in with legitimate traffic, making it difficult to detect. The Malleable C&C profiles are loaded onto the team server and modify the in-memory characteristics of the Beacon, how to transform and store the data in a transaction and post-exploitation functions.

The Malleable C&C profile is structured into several sections which are used to configure the global Beacon behavior or specific behavior depending on the

communication protocol used. The global configuration includes the *sleeptime* which is the Beacon callback interval, the *data jitter* which is a random length string up to the chosen value (in bytes) that is appended to the server value and the *useragent* which sets the User-Agent string used in the HTTP requests identifying the application, operating system, vendor and version of the requesting computer program.

The Malleable C&C profile can also be used to configure the HTTP headers, the SSL certificate used in the HTTPS sessions, the URI of the HTTP requests, the HTTP verb used in the transactions and the modifications performed to the payload of the packets sent from the server and the Beacon, including appended data and encoding.

Our analysis of the HTTP/S Cobalt Strike traffic and the Malleable C&C profiles has identified several network indicators that can be used to detect the Cobalt Strike C&C traffic: 1) the Beacon sleeps for an interval of time after not receiving tasks or after sending the output to the server; 2) most of the sessions have a short duration and few data packets are exchanged; 3) in most cases the victim sends more data to the C&C server than the opposite, especially when it is exfiltrating information about the victim.

3 A Machine Learning Based Detection

3.1 Threat Model

Our objective is to build a machine learning based detection system that can identify and detect stealthy Cobalt Strike C&C activity from encrypted network traffic in near real-time. We assume the following:

- The system has already been infected with a Cobalt Strike Beacon and that the initial infection has not been detected by IDS/IPS.
- The Cobalt Strike Beacon initiates the communications with the C&C server after completing the download of the Beacon payload.
- The attackers use encryption and customized Cobalt Strike Malleable C&C profiles to mimic legitimate traffic and evade detection. Consequently, the C&C packets exchanged do not have any fixed pattern.

3.2 Flow Based Features

In order to detect Cobalt Strike C&C traffic amongst legitimate traffic, we have used the traffic analysis software Zeek [13] to extract the network indicators of the individual flows or connections. Zeek produces a record for each connection that has occurred with a system in the log file *conn.log* in real-time, but it can also be used to analyze packet captures and output the connections within. While a connection is usually associated with the TCP protocol, Zeek can also track stateless protocols like UDP. The flow-related features that have been selected from the Zeek output are the following:

- **id.orig_h:** string. IP address of the host that initiated the connection.
- **id.orig_p:** integer. Port used by the host that initiated the connection.
- **id.resp.h:** string. IP address of the host that received the connection.
- **id.resp.p:** integer. Port used by the host that received the connection.
- **proto:** string. Transport layer protocol of the connection (TCP, UDP, ICMP).
- **service:** string. Identification of an application protocol sent over the connection (DNS, HTTP, HTTPS...).
- **duration:** double. Total duration of the connection.
- **orig_bytes:** integer. Payload bytes that the originator of the connection sent.
- **resp_bytes:** integer. Payload bytes that the responder sent.
- **conn_state:** string. State of the connection, some typical values include SHR (responder sent SYN ACK followed by SYN) and S0 (connection attempt seen).
- **history:** string. Records the state history of connections as a string of letters.
- **orig_pkts:** integer. Number of packets that the originator sent during the connection.
- **orig_ip_bytes:** integer. Number of IP level bytes that the originator sent during the connection.
- **resp_pkts:** integer. Number of packets that the responder sent during the connection.
- **resp_ip_bytes:** integer. Number of IP level bytes that the responder sent during the connection.

Features such as the IP addresses and ports of the sender and the receiver of the connection will be used to identify the connections, but cannot be used in the detection process. Other features such as the protocol, duration of the connection, packets and bytes exchanged will be used by the machine learning model to make the predictions on the traffic, based on the analysis of the Cobalt Strike C&C traffic.

3.3 Machine Learning Models

In order to select the machine learning algorithms that will be used to develop the model, it will be necessary to assess which algorithms best tackle the problem at hand. Since the goal of the model is to make predictions on the network traffic to determine if a system has been infected with a Cobalt Strike Beacon, we can identify it as a binary classification problem. However, since a system being infected with Cobalt Strike is considered an abnormal event due to its rare occurrence probability, it can also be identified as an anomaly detection problem, even if the Beacon traffic features do not differ significantly from the legitimate traffic, as its purpose it to disguise itself as such. The training dataset records will be labeled, thus both supervised and unsupervised anomaly detection analysis can be performed. However, as we will observe, the unsupervised model will obtain worse results due to the similarity in feature values that the Beacon and the legitimate traffic have.

The following machine learning algorithms will be evaluated to generate the model: random forest, artificial neural network, support vector machine and naïve Bayes for the supervised model, and K-Means clustering algorithm for the unsupervised model.

3.4 Evaluation Metrics

A variety of metrics will be used to objectively evaluate the different machine learning algorithms that will be used to generate the model that will make predictions on network traffic to discover if a system has been infected with a Cobalt Strike Beacon.

A confusion matrix is a table commonly used to describe the performance of a machine learning model when performing classification tasks over a test dataset whose labels (or classes) are known. It does so by establishing a relationship between the real label of a record and the predicted label of the same record, thus graphically exposing the number of records that have been correctly and incorrectly classified.

Independently of the number of labels in the dataset, the confusion matrix outputs four values for each label. For a specific label, the number of true positives (TP) is the number of records corresponding to that label that have been correctly classified as such. The number of false negatives (FN) is the number of records corresponding to the label that have been misclassified as belonging to other labels. The number of false positives (FP) indicates the number of records belonging to different labels that have been misclassified as belonging to the chosen label. Last, the number of true negatives (TN) is the number of records that have been correctly classified as belonging to other labels. In the case of our paper, the positive label will refer to the Beacon traffic, while the legitimate traffic will be labeled as negative.

Using the output of the confusion matrix, it is possible to calculate the following metrics to evaluate the machine learning algorithms:

- **Accuracy:** Ratio between the number of correct predictions and the total number of predictions made. It is the best indicator of the algorithm's performance only if the dataset has the same number of records for each class.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

- **Precision:** The precision is the ratio between the number of correct predictions for a class and the total number of predictions of said class. The weighted precision is used to evaluate the model for all classes, calculating a weighted average depending on the probability of occurrence of each class.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

- **Recall or True Positive Rate (TPR):** Proportion of positive records correctly classified as such compared to the total number of positive records.

It can be considered a percentage. Similarly to the precision, we will consider the weighted recall for all the labels.

$$TPR = \frac{TP}{TP + FN} \tag{3}$$

- **False Positive Rate (FPR):** Proportion of negative records incorrectly classified as positive records, with respect to all negative records. It can be considered a percentage.

$$FPR = \frac{FP}{FP + TN} \tag{4}$$

- **F1 score:** Measures the performance of a model by considering both its precision as well as its robustness. To do so, it uses the Precision and Recall values.

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{5}$$

When evaluating the model certain evaluation metrics will have more importance than others. That will be the case of the true positive rate or TPR, which will indicate the model's ability to correctly detect the Beacon traffic. The false positive rate or FPR will be used to evaluate the adequacy of the model to be deployed in a real environment, as most of the network traffic that an intrusion detection system analyzes is legitimate, a low rate of false alarms is required for its deployment. For example, if a detection system is deployed in a real environment that will potentially see hundreds of thousands to millions of connections, a high false positive rate will cause disruptions in the legitimate activity of the users of the network, as tens of thousands of legitimate connections will be detected as malicious. Finally, the F1 score will provide overall information about the model's accuracy and robustness.

4 Empirical Validation

4.1 Dataset Acquisition

The machine learning based detection requires a training dataset to develop the machine learning model and a testing dataset to evaluate the performance of the model. Both datasets contain legitimate traffic and Cobalt Strike C&C traffic, so that the model is developed and evaluated using traffic that is closely related with the traffic that would be found in a real life environment.

The legitimate traffic has been obtained from the popular cybersecurity dataset CICIDS17 [14] of the Canadian Institute of Cybersecurity and the public CTU-Normal-20 dataset [15] of the Stratosphere Research Laboratory. For the training dataset, we have generated and collected 31 Cobalt Strike C&C packet captures (PCAP) in our lab environment using different Malleable C&C profiles that emulate advanced penetration threats (APT), crimeware, normal traffic, or generated using randomizers such as [16]. The use of different Malleable C&C

profiles and commands reduces the possibility of overfitting the model to a specific profile, thus enabling the machine learning models to perform better in the presence of previously unseen real world Cobalt Strike traffic. The testing dataset, on the other hand, has been generated using 25 captured traces of real world cyberattacks that include Cobalt Strike C&C traffic from malware-traffic-analysis.net [17]. All the Cobalt Strike traffic in these traces has been manually labeled by cybersecurity teams, which enables us to measure the performance of our machine learning based detection.

Each record and its features in the dataset corresponds to an individual flow or connection in the network, which has been extracted from the acquired traffic using the network analysis tool Zeek. Each record has been labeled using an additional feature, *label*, which will take two values: 1 if the record corresponds to Cobalt Strike Beacon traffic, and 0 if the record corresponds to other types of traffic (legitimate or unknown, this is due to some of the test captures containing malicious traffic generated from other malware).

Table 1. Dataset Content

	Cobalt Strike (HTTP)	Cobalt Strike (HTTPS)	Legitimate HTTPS
Training dataset	5,500 records	4,000 records	391,500 records
Testing dataset	450 records	3,150 records	10,800 records

Table 1 shows the resulting record distribution of the datasets for each of the labels. Given that Cobalt Strike C&C traffic is very rare in real world, we deliberately build the training dataset with significantly more legitimate traffic records than Beacon traffic records. Such a training dataset not only enable the machine learning model to detect Cobalt Strike activities in real world scenarios, but also helps reduce potential detection false positives generated by the model. The percentage of Beacon records, which is around 2% of all the records in the training dataset, is a lower than average percentage for most cybersecurity datasets, but it is enough for most models to be able to detect the Beacon traffic generated using different Malleable C&C profiles and also correctly identify the legitimate traffic producing a low rate of false alarms.

4.2 Flow Based Machine Learning Detection

First, we selected the features from the raw dataset that will be used in the detection of the network traffic. Each record belonging to the dataset has 14 features out of which four of those features (*src_ip*, *src_p*, *dst_ip* and *dst_p*) will only be used for identification purposes. The IP addresses and ports used in the connection communications will not be used to detect the Beacon traffic, as they are network indicators that can easily be changed by the attacker, thus evading the detection system.

Since the machine learning model will focus on the detection of HTTP/S Beacons, which use the TCP protocol in the transport layer, the *proto* and

service features will be key in the identification of the Beacon traffic. The state of the connection feature, *conn_state*, will not be selected because it will not help in the identification of the Beacon traffic, since it only identifies if the connection has correctly finished or not and most if not all the records in the dataset have the same value for the connection. On the contrary, the duration of the connection will not be used in the detection of the Beacon traffic because it can vary due to external factors such as network latency and packet drops regardless of the type of traffic.

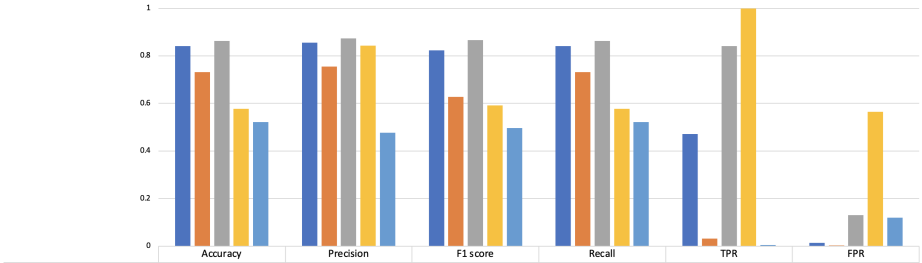
The *history* feature, extracted by Zeek, records the "history" of the transactions in the connection such as the type of packets that were transmitted, the order in which the packets were sent, and who send the packets by using letters. Therefore, based on the analysis of the Beacon traffic performed in previous sections, we can assume that the history feature will have similar values for the connections made by the Beacon traffic, as it always follows the same general schema of communications. The selection of the rest of the features (*orig_bytes*, *resp_bytes*, *orig_pkts*, *orig_ip_bytes*, *resp_pkts*, *resp_ip_bytes*) will be done alongside the evaluation of the model, depending on how they affect the performance of the model.

Since several features selected contain categorical data – *proto*, *service* and *history* more precisely – it will be necessary to perform feature encoding to transform their values to numerical data. To do so, we transformed the features' values from string to integer by assigning an integer value to each distinct string value according to the frequency of appearance of the value and generating an additional feature with "_index" in the name. If a string value appears for the first time in the testing dataset, it will be assigned the same value as the least frequent value.

The hyperparameter tuning phase has been performed using the results from the machine learning model evaluation for their respective algorithms. However, all the tested values for the tuning phase will not be explained, since most experiments regarding the hyperparameter tuning phase give the reader little information, as they consist in manual changes to the hyperparameter values which improve the performance of the models by less than 1% on most occasions.

The random forest model has been generated using *proto_index*, *service_index*, *history_index*, *orig_bytes*, *resp_bytes*, *orig_pkts*, *orig_ip_bytes* and *resp_pkts* as input features, and the hyperparameters values of a maximum depth of 15 and 30 trees created. As we can observe, the random forest model achieves a moderately good detection rate (around 50%) and a low false alarm rate of 1.4%. On the other hand, the neural network model achieves poor detection results with a 3% detection rate but a low false alarm rate of 0.3%. The neural network model has been built using a four-layer structure, using two hidden layers with 18 nodes each, taking the 9 selected features as inputs (*proto_index*, *service_index*, *history_index*, *orig_bytes*, *resp_bytes*, *orig_pkts*, *orig_ip_bytes*, *resp_pkts* and *resp_ip_bytes*) and utilizes the L-BFGS solver.

The naïve Bayes model and the linear support vector machine achieve higher detection rate than the random forest, at the expense of the false positive



	Accuracy	Precision	F1 score	Recall	TPR	FPR
Random forest	0.841	0.855	0.824	0.841	0.471	0.014
Neural network	0.731	0.755	0.627	0.731	0.031	0.003
Naive Bayes	0.863	0.874	0.866	0.863	0.842	0.13
Linear Support Vector Machine	0.5776	0.8426	0.5912	0.5776	1	0.5639
K-Means	0.521	0.476	0.497	0.521	0.004	0.12

Fig. 3. Comparison of the performance of the machine learning models

rate. The naïve Bayes model takes *proto_index*, *service_index*, *history_index*, *orig_bytes* and *orig_pkts* as input features and using a multinomial model. While obtaining a high true positive rate at 84%, it does not serve a real intrusion detection system due to having a 13% false positive rate. Similarly, the linear support vector uses three input features *proto_index*, *service_index*, *history_index* and an aggregation depth of 4 achieving a 100% detection rate but misclassifying 56% of the legitimate traffic. Finally, the unsupervised clustering machine learning model K-means attempts to group the Beacon and legitimate traffic in two separate clusters using the same input features as the linear support vector machine model, but ultimately fails to do so, obtaining a 0.4% true positive rate and a 12% false negative rate.

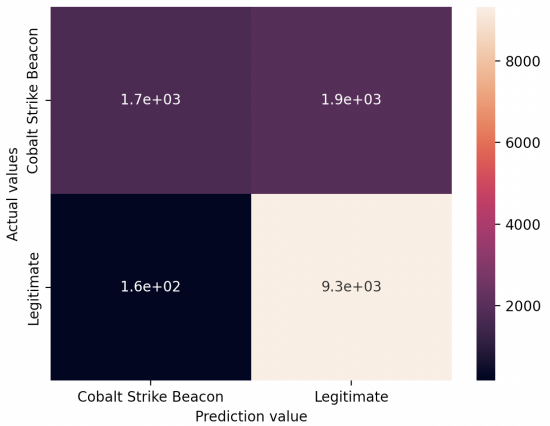


Fig. 4. Confusion Matrix of the Random Forest Model

As shown in Figure 3, the random forest and naïve Bayes models perform better than the rest of the models if we consider the F1 score, which considers both the precision and the recall of the models. While the naïve Bayes model achieves a higher F1 score of 86.6% because it detects the Beacon traffic with more accuracy, its high false alarm rate (13%) does not allow its deployment in a real environment that will see a really high percentage of legitimate connections. The random forest model, while having a lower F1 score of 82.4% and having more difficulty detecting Beacon traffic, makes for a better intrusion detection system due to having a false positive rate of 1.4%, as it will produce ten times less false alarms than the naïve Bayes model.

Figure 4 shows the confusion matrix of the random forest model when tested against real traffic from Cobalt Strike attacks that include HTTP and HTTPS Beacon traffic, normal traffic, and malicious traffic from other malware sources. Since the focus of this work is to detect Cobalt Strike C&C traffic, the malicious traffic from other sources has been labeled as legitimate.

5 Related Works

Machine learning techniques (e.g., decision trees, neural network) have a long history of being used in constructing Intrusion Detection Systems (IDS) [18–23]. Recent machine learning based IDS approaches have been shown to have good accuracy and acceptable efficiency in detecting/classifying attacks in the public datasets (e.g., UNSW-NB15, CICIDS17). Yet, machine learning based approaches have achieved far less success in real world intrusion detection than in other areas such as speech recognition. R. Sommer et al. [24] examined the fundamental differences between the network intrusion detection problem and those problems where machine learning regularly finds much more success, and argued that it is significantly harder to apply machine learning techniques effectively in intrusion detection.

Most proposed machine learning based IDS approaches have focused on detecting known exploits rather than stealthy C&C activities from encrypted traffic. Gardiner et al. [25] examined evasion techniques against machine learning based detection, and pointed out that many existing machine learning based C&C detection approaches are vulnerable to evasion.

Despite Cobalt Strike C&C has been used in almost all massive breaches [5], there are very few published results on how to detect Cobalt Strike C&C traffic. Navarrete et al. from Palo Alto performed an extensive analysis on the Cobalt Strike C&C traffic encoding [26] and encryption [27] of the payload as well as the Malleable C&C profiles [28], and explained why such versatility makes Cobalt Strike C&C difficult to detect.

B. Vennyk [5] suggested using beaconing characteristics to detect Cobalt Strike C&C communication without any empirical validation result. In addition, it did not consider the data jitter and the sleep jitter that the attackers can configure to significantly change the beaconing pattern.

N. Kanzig et al. [29] proposed a machine learning based approach to identify C&C channels using features extracted by CICFlowMeter. While they have shown that their random forest classifier can detect their lab generated Cobalt Strike C&C traffic, their work does not demonstrate if their classifier can detect any real world Cobalt Strike C&C traffic captured from real world attacks. In addition, they have not considered the data jitter that can be introduced into the server responses which would impact the packet flow features.

Van der Eijk et al. [30] proposed a threshold based approach detect Cobalt Strike C&C traffic. The proposed method was able to detect the Cobalt Strike C&C traffic generated with a few Malleable profiles in their lab environment.

In summary, all previous machine learning based Cobalt Strike C&C detection approaches used lab generated Cobalt Strike traffic with very few Malleable profiles, and none of them have been validated with real world Cobalt Strike C&C traffic. In contrast, our machine learning based detection has been trained with 31 different Malleable profiles of Cobalt Strike C&C and validated with Cobalt Strike C&C traffic captured from real world cyberattacks.

6 Conclusions

Given the wide spread use of Cobalt Strike C&C in recent massive data breach attacks and ransomware attacks, it is critically important to be able to detect the stealthy Cobalt Strike C&C traffic in order to effectively mitigate the these stealthy and damaging cyberattacks.

In this paper, we propose using flow based features to detect Cobalt Strike Beacon C&C traffic, and evaluate five machine learning algorithms to develop a model that can detect the Beacon traffic. To the best of our knowledge, our machine learning based detection is the first to be validated using Cobalt Strike C&C traffic captured from real world cyberattacks. Our experimental results show that it is feasible to detect real world, previously unseen Cobalt Strike C&C traffic with a reasonable true positive rate and low false alarm rate at the same time.

For future work, we plan to look for more effective machine features for detecting stealthy Cobalt Strike C&C traffic, and investigate how to improve the detection true positive rate and reduce the false positive rate at the same time by combining different machine learning models.

References

1. Swinhoe, D.: The 15 biggest data breaches of the 21st century (April 2020), https://www.csoonline.com/article/2130877/the-biggest-data-breaches-of-the-21st-century.html
2. Security, I.: Cost of a Data Breach Report 2019 (2019), https://f.hubspotusercontent40.net/hubfs/2783949/2019_Cost_of_a_Data_Breach_Report_final.pdf

3. Schwartz, M.J.: Equifax's Data Breach Costs Hit $1.4 Billion (May 2019), https://www.bankinfosecurity.com/equifaxs-data-breach-costs-hit-14-billion-a-12473
4. Morgan, S.: Cybercrime To Cost The World $10.5 Trillion Annually By 2025 (November 2020), https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/
5. Vennyk, B.: How to Detect CobaltStrike Command & Control Communication(March 2021), https://underdefense.com/guides/how-to-detect-cobaltstrike-command-control-communication/
6. Rahman, A.: Defining Cobalt Strike Components So You Can BEA-CONfident in Your Analysis (October 2021), https://www.mandiant.com/resources/blog/defining-cobalt-strike-components
7. Liebenberg, D., Huey., C.: Quarterly Report: Incident Response Trends in Summer 2020 (September 2020), http://blog.talosintelligence.com/2020/09/CTIR-quarterly-trends-Q4-2020.html
8. FireEye: Highly Evasive Attacker Leverages SolarWinds Supply Chain to Compromise Multiple Global Victims With SUNBURST Backdoor (December 2020), https://www.fireeye.com/blog/threat-research/2020/12/evasive-attacker-leverages-solarwinds-supply-chain-compromises-with-sunburst-backdoor.html
9. 2020 United States federal government data breach, https://en.wikipedia.org/wiki/2020_United_States_federal_government_data_breach
10. Abrams, L.: SolarLeaks site claims to sell data stolen in SolarWinds attacks(January 2021), https://www.bleepingcomputer.com/news/security/solarleaks-site-claims-to-sell-data-stolen-in-solarwinds-attacks/
11. Timberg, C., Nakashima, E.: The U.S. government spent billions on a system for detecting hacks. The Russians outsmarted it (February 2021), https://www.seattletimes.com/nation-world/the-u-s-government-spent-billions-on-a-system-for-detecting-hacks-the-russians-outsmarted-it/
12. Software for Adversary Simulations and Red Team Operations, https://www.cobaltstrike.com
13. An Open Source Network Security Monitoring Tool, https://zeek.org/
14. for Cybersecurity, C.I.: Intrusion Detection Evaluation Dataset (CIC-IDS2017), https://www.unb.ca/cic/datasets/ids-2017.html
15. Laboratory, S.R.: Malware Capture Facility Project, https://www.stratosphereips.org/datasets-normal
16. Malleable-C2-Randomizer, https://github.com/bluscreenofjeff/Malleable-C2-Randomizer
17. A source for packet capture (pcap) files and malware samples, https://www.malware-traffic-analysis.net/index.html
18. Sinclair, C., Pierce, L., Matzner, S.: An Application of Machine Learning to Network Intrusion Detection. In: Proceedings of the 5th Annual Computer Security Applications Conference (ACSAC 1999) (December 1999)
19. Sangkatsanee, P., Wattanapongsakorn, N., Charnsripinyo, C.: Practical Real-Time Intrusion Detection Using Machine Learning Approaches. Computer Communications **34**(18), 2227–2235 (December 2011)
20. Kulariya, M., Saraf, P., Ranjan, R., Gupta, G.P.: Performance Analysis of Network Intrusion Detection Schemes Using Apache Spark. In: Proceedings of the 2016 International Conference on Communication and Signal Processing (ICCSP 2016). pp. 1973–1977. IEEE (April 2016)

21. Toupas, P., Chamou, D., Giannoutakis, K.M., Drosou, A., Tzovaras, D.: An Intrusion Detection System for Multi-class Classification Based on Deep Neural Networks. In: Proceedings of the 18th IEEE International Conference On Machine Learning And Applications (ICMLA 2019). IEEE (December 2019)
22. Quinan, P.G., Traore, I., Gonhdi, U.R., Woungang, I.: Unsupervised Anomaly Detection Using a New Knowledge Graph Model for Network Activity and Events. In: Proceedings of the 4th International Conference on Machine Learning for Networking (MLN 2021). pp. 117–130. Springer LNCS 13175 (December 2021)
23. Tufan, E., Tezcan, C., Acartürk, C.: Anomaly-Based Intrusion Detection by Machine Learning: A Case Study on Probing Attacks to an Institutional Network. IEEE Access **9**, 50078–50092 (2021)
24. Sommer, R., Paxson, V.: Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In: Proceedings of the 2010 IEEE Symposium on Security and Privacy (S&P 2010). IEEE (May 2010)
25. Gardiner, J., Nagaraja, S.: On the Security of Machine Learning in Malware C&C Detection: A Survey. ACM Computing Surveys **49**(3), 1–39 (September 2017)
26. Navarrete, C., Sangvikar, D., Guan, A., Fu, Y., Jia, Y., Shibiraj, S.: Cobalt Strike Analysis and Tutorial: CS Metadata Encoding and Decoding (May 2022), https://unit42.paloaltonetworks.com/cobalt-strike-metadata-encoding-decoding/
27. Navarrete, C., Sangvikar, D., Guan, A., Fu, Y., Jia, Y., Shibiraj, S.: Cobalt Strike Analysis and Tutorial: CS Metadata Encryption and Decryption (July 2022), https://unit42.paloaltonetworks.com/cobalt-strike-metadata-encryption-decryption/
28. Navarrete, C., Sangvikar, D., Guan, A., Fu, Y., Jia, Y., Shibiraj, S.: Cobalt Strike Analysis and Tutorial: How Malleable C2 Profiles Make Cobalt Strike Difficult to Detect (March 2022), https://unit42.paloaltonetworks.com/cobalt-strike-malleable-c2-profile/
29. Känzig, N., Meier, R., Gambazzi, L., Lenders, V., Vanbever, L.: Machine Learning-based Detection of C&C Channels with a Focus on the Locked Shields Cyber Defense Exercise. In: Proceedings of the 11th International Conference on Cyber Conflict (CyCon 2019). pp. 1–19. IEEE (May 2019
30. van der Eijk, V., Schuijt, C.: Detecting Cobalt Strike Beacons in NetFlow Data, https://rp.os3.nl/2019-2020/p29/report.pdf