2019 11th International Conference on Cyber Conflict:
Silent Battle
T. Minárik, S. Alatalu, S. Biondi,
M. Signoretti, I. Tolga, G. Visky (Eds.)
2019 © NATO CCD COE Publications, Tallinn

Machine Learning-based Detection of C&C Channels with a Focus on the Locked Shields Cyber Defense Exercise

Nicolas Känzig
Department of Information Technology and Electrical Engineering
ETH Zürich
Zürich, Switzerland
kaenzign@student.ethz.ch

Roland Meier
Department of Information Technology and Electrical Engineering
ETH Zürich
Zürich, Switzerland
meierrol@ethz.ch

Luca Gambazzi
Science and Technology
armasuisse
Thun, Switzerland
luca.gambazzi@armasuisse.ch

Vincent Lenders
Science and Technology
armasuisse
Thun, Switzerland
vincent.lenders@armasuisse.ch

Laurent Vanbever
Department of Information Technology and Electrical Engineering
ETH Zürich
Zürich, Switzerland
lvanbever@ethz.ch

Abstract: The diversity of applications and devices in enterprise networks combined with large traffic volumes make it inherently challenging to quickly identify malicious traffic. When incidents occur, emergency response teams often lose precious time in reverse-engineering the network topology and configuration before they can focus on malicious activities and digital forensics.

In this paper, we present a system that quickly and reliably identifies Command and Control (C&C) channels without prior network knowledge. The key idea is to train a classifier using network traffic from attacks that happened in the past and use it to identify C&C connections in the current traffic of other networks. Specifically, we leverage the fact that – while benign traffic differs – malicious traffic bears similarities across networks (e.g., devices participating in a botnet act in a similar manner irrespective of their location).

To ensure performance and scalability, we use a random forest classifier based on a set of computationally-efficient features tailored to the detection of C&C traffic. In order to prevent attackers from outwitting our classifier, we tune the model parameters to maximize robustness. We measure high resilience against possible attacks – e.g., attempts to camouflaging C&C flows as benign traffic – and packet loss during the inference.

We have implemented our approach and we show its practicality on a real use case: Locked Shields, the world's largest cyber defense exercise. In Locked Shields, defenders have limited resources to protect a large, heterogeneous network against unknown attacks. Using recorded datasets (from 2017 and 2018) from a participating team, we show that our classifier is able to identify C&C channels with 99% precision and over 90% recall in near real time and with realistic resource requirements. If the team had used our system in 2018, it would have discovered 10 out of 12 C&C servers in the first hours of the exercise.

Keywords: *malware, botnets, machine learning, digital forensics, Locked Shields, network defense*

1. INTRODUCTION

Large enterprise or campus networks handle data from a vast set of different applications, protocols, and devices. Identifying malicious traffic in such networks is similar to the figurative problem of finding a needle in a haystack, raising the need for effective tools to automate this process and to support defenders such as computer emergency response teams (CERTs) in their operation. As network traffic is not only voluminous but also very diverse, these tools need to adapt to different contexts.

Recent alarming examples of malicious software exploiting a remote infrastructure in order to issue directives to steal or modify data or performing distributed denial-of-service attacks include CryptoLocker [1] or the Mirai botnet [2].

Machine learning-based models have repeatedly been proven to outperform humans in tasks involving large data volumes and high-dimensional feature spaces. However, training these models to detect malicious activity in networks is a particularly challenging task, because the methods used by modern threat actors are continuously evolving. Moreover, the profiles of legitimate background traffic can vary strongly among different networks and their users. Consequently, such solutions might perform well in the environment they have been trained in, while failing in new deployments.

In this paper, we focus on one particular type of malicious traffic: communication between compromised hosts and their Command and Control (C&C) servers. C&C traffic only depends on the botnet (i.e. the communication scheme between the C&C server and the bots) and is invariant to the networks to which the bots are connected. This makes the development of machine learning-based models that perform reliably in different contexts more feasible. We argue that identifying this type of traffic is fruitful because it means that compromised hosts can be identified (and eventually blocked, isolated or patched) before an actual attack is launched.

The work that we present in this paper is based on data from Locked Shields [3], the world's largest cyber defense exercise. While Locked Shields is only an exercise, it reproduces critical infrastructure under the intense pressure of severe cyberattacks. Moreover, it provides a setting that closely matches the real world: in practice, defenders have limited resources to protect a large, heterogeneous network against unknown attacks. And because it is an exercise, we obtained a ground-truth of logs from the attackers describing when and where they were active, something which is hardly possible for real incidents.

Problem statement: Given the constraints (e.g. in terms of computational resources and lack of familiarity with the network) that defending teams face during the Locked Shields exercise, we aim to design a system that can identify C&C traffic and compromised hosts.

Challenges: Solving this problem is challenging for the following reasons:

- Benign and malicious traffic profiles can vary considerably between different Locked Shields exercises.
  This requires a solution with high generalization and robustness.

- Defenders have a very limited budget for computational resources. This requires an efficient classification technique.
- Defenders have a small amount of storage capacity. This prevents them from storing large amounts of network traffic.
- Defenders have a small bandwidth to access the attacked network. This makes it impossible to send large amounts of data to an external system.

Our approach: Our key idea is to use data from past iterations of Locked Shields to efficiently identify similar-looking C&C traffic in future exercises. We do this by creating a labeled dataset containing flow-based features extracted from raw Locked Shields traffic captures, which we then use to train a supervised classifier (random forest) to flag C&C traffic. Our approach is efficient enough to be deployed during future Locked Shields exercises.

Novelty and related work: Detecting C&C traffic has been the focus of many research papers in recent years (cf. surveys in [4] [5]), many of which also pursue classifier-based approaches using machine learning algorithms. [6] proposes a two-stage system for identifying P2P C&C traffic using a decision tree and a random forest classifier. To train a random forest classifier, [7] leverages the fact that malware-related domains are likely to have an inconsistent pool of requesting hosts. [8] develops a system for classifying malicious C&C servers using NetFlow data, extracting features related to flow sizes, client access patterns and temporal behavior.

In contrast to these approaches, we use a new set of flow-based features and evaluate our models on two new and completely labeled datasets (Locked Shields 2017 and 2018). While most studies train and evaluate their models on different parts of the same dataset, we use train- and test-sets that have been acquired independently in different setups. This provides strong evidence for the ability of our system to perform in new environments. Moreover, a minority of the solutions proposed in past investigations claim to run in real time [4]. In our approach, we combine quickly computable features (e.g. number of packets per flow) with an efficient random forest algorithm, which makes real-time calculation feasible.

Contributions: The main contributions of this paper are:

- A selection of features that allow identifying C&C channels while being fast and efficient to compute.
- An efficient random forest model that classifies between C&C traffic and normal traffic with high accuracy.
- An implementation of the system that is suitable for deployment in future Locked Shields exercises.

- An evaluation based on real data from Locked Shields 2017 and 2018, which shows that our system allows defenders to identify C&C traffic, C&C servers and compromised hosts.

Organization: The remainder of this paper is organized as follows. In Section 2, we provide background information on the Locked Shields exercise and define the attacker model. In Section 3, we present our system to identify C&C traffic before we evaluate it in Section 4. In Section 5, we discuss the outcome and finally, we conclude in Section 6.

2. BACKGROUND ON LOCKED SHIELDS

In this section, we explain how Locked Shields is organized and give details about the roles of defenders and attackers.

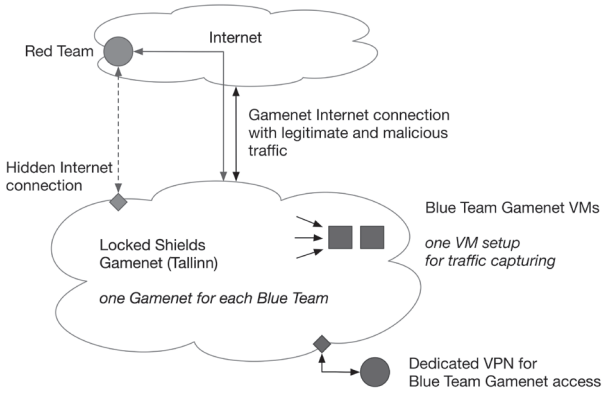
A. Exercise Organization

Locked Shields is the largest and most complex live-fire global cyber defense exercise, with more than 1000 participating cyber experts from 30 nations [9]. It takes place every year and is organized by the NATO Cooperative Cyber Defence Centre of Excellence (CCDCOE) in Tallinn (Estonia) [3].

For the exercise, participating countries send *Blue Teams*, which represent response teams whose main task is to secure and protect the network infrastructure. Whereas each Blue Team operates in an isolated instance of the network (*Gamenet*), a *Red Team* runs attacks against all these networks in order to compromise or degrade the performance of the connected systems.

In Figure 1, we illustrate the environment during an exercise.

FIGURE 1. LOCKED SHIELDS ENVIRONMENT OVERVIEW.



The environment simulated during the exercise changes every year. In this paper, we focus on the last two occurrences of Locked Shields (2017 and 2018). In 2017, the Blue Teams had to maintain the services and networks of a military air base; in 2018, a major civilian Internet service provider, a military base and other critical infrastructures of a fictional country were targeted in cyber attacks.

B. Environment and Constraints for the Defenders

Prior to the exercise, the defenders (Blue Teams) receive an architecture scheme of the original Gamenet that shows the topology and connected devices. However, the scheme does not show changes put in place by the Red Team (e.g. additional connections between the Gamenet and the Internet to bypass the main gateways).

In addition, each Blue Team obtains two virtual machines (VMs) inside the Gamenet, which it can use during the exercise to install its own tools (e.g. to perform forensics or deploy patches). Moreover, the traffic exchanged in the Gamenet is forwarded to one VM in order to allow the Blue Team to perform on-site analysis and detection. However, the performance of this VM is limited and access to it is only possible via a low-bandwidth VPN tunnel. In order to rapidly counter Red Team activity, the Blue Team has to deploy efficient analysis tools (given the constraints on computation and bandwidth), intrusion detection systems, and to avoid sending voluminous data to an external infrastructure. The system that we present in this paper is designed to work in such a restricted environment.

After the exercise, the Red Team delivers reports to the Blue Teams summarizing their malicious activities.

C. Attacker Model

The attackers (Red Team) perform their activities according to a tight schedule of missions and goals. Waves of attacks hit the Blue Team for the entire duration of the exercise. Some attacks are limited to a specific phase of the exercise while others are repeated during the entire exercise.

Prior to the exercise, the Red Team knows the configuration of the entire Gamenet and can use this knowledge to prepare suitable attacks (e.g. leveraging outdated systems).

In order to systematically orchestrate the large number of attacks on all Gamenets, the Red Team uses Cobalt Strike as a C&C framework. This allows automatizing injections, deployment of malicious code and C&C datalink management.

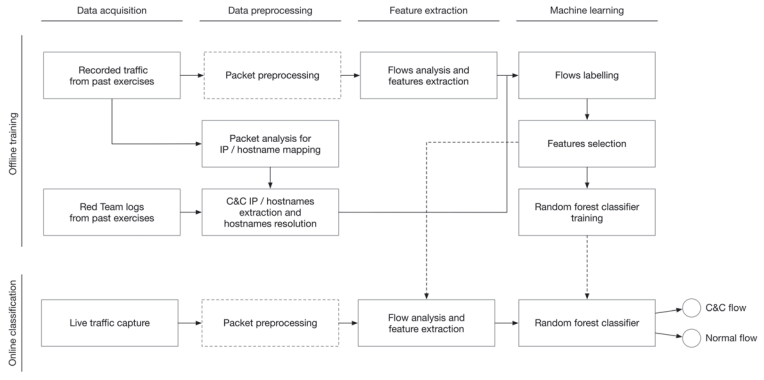
3. SUPERVISED MACHINE LEARNING FOR DETECTING C&C CHANNELS

In this section, we explain how we use supervised machine learning to identify C&C channels in the Locked Shields exercise. First, we provide an overview of our approach. Afterwards, we describe the data and labeling that we used. Finally, we explain how we selected the features and the machine learning model for this task.

A. Overview

Our system consists of two basic phases (Figure 2): offline training and online classification. In the offline training phase (which was done prior to the exercise), we used data from past Locked Shields exercises and processed them in order to obtain a labeled dataset to train a supervised classifier that could be used for live classification of C&C flows during the exercise.

FIGURE 2. SYSTEM OVERVIEW.



B. Data Analysis and Enrichment

In this section, we describe the data sources we used for labeling and training and the preprocessing steps we applied.

1) Available Data Sources

We built our labeled dataset from two sources: raw traffic captures and Red Team logs.

a) Raw Traffic Captures

We obtained pcap traffic traces containing the Gamenet activity recorded during Locked Shields 2017 and 2018 (LS17, LS18) from a participating country (Switzerland). The packets are not sampled, anonymized, or truncated. We extracted the features used to train our models from this data.

b) Red Team Logs

The activities of the Red Team are logged in different documents generated by the Cobalt Strike framework [10]. Among others, these documents contain *indicators of compromise* (e.g. IP addresses and domain names of C&C servers) and an *activity report*, which contains a timeline of all Red Team activities (e.g. commands that were executed on compromised machines). We used these log files to label the C&C flows.

2) Data Preprocessing

Before extracting features, we preprocessed the dataset in three ways: *(i)* we truncated packets to reduce the size of the dataset; *(ii)* we aggregated packets to flows; and *(iii)* we mapped domain names and IP addresses in the traffic capture.

a) Truncating Packets

Since capturing and analyzing full packets in real time is difficult for the Blue Team during the exercise, our approach does not require packet payloads. We used only the first 96 bytes (enough to capture everything up to the header of the transport layer) of each packet, which reduced the size of our dataset by approximately 75 percent. The performance of our final models did not decrease due to the truncation.

b) Flow Extraction

We aggregated the packets from the packet trace into flows, since our model operates at the flow level. A flow is defined by its 5-tuple (source IP, destination IP, source port, destination port, transport layer protocol). It starts with a TCP SYN packet and ends when the first TCP FIN packet is sent or after a timeout of 15s. We used CICFlowMeter [11] to extract flow-based features from the raw traffic traces.

c) Domain Name Resolution

The Red Team logs list some devices only by their domain name, thus we needed a mapping from these domain names to the associated IP addresses. We used Bro [12], a network analysis framework, to resolve the domain names to IP addresses from the packet traces, using information contained in the HTTP (host header), TLS (with server name indication), or DNS.

C. Data Labeling

After extracting a list of IP addresses and domain names of C&C servers from the Red Team logs, the labeling process was straightforward: we labeled all flows where at least one endpoint was a C&C server (i.e. listed in the Red Team logs) as malicious and all other flows as benign. The intuition behind this approach was that there was no benign reason for any device to contact a C&C server. It is safe to assume that any device communicating with a C&C server is compromised.

D. Feature Selection and Extraction

In this section, we explain how we selected and extracted the features that our classifier would use to identify C&C flows.

1) Feature Extraction

For computing the features, we used CICFlowMeter [13] (version 3.0), an open source tool for extracting flows from packet traces and computing large sets of features. CICFlowMeter focuses on time-related features such as the inter-arrival time of packets, active and idle times separately for packets in each direction, while including minimum, maximum, mean and standard deviation [11]. These features are suitable for our purposes because they can be extracted with little computational effort.

To capture the fact that C&C servers are typically located outside the internal network, we added an additional feature (Int/Ext Dst IP), indicating whether the destination IP address of a flow is within the internal address space.

Table I lists all features that we considered in our selection process.

TABLE I: COMPLETE LIST OF FEATURES CONSIDERED IN THE FEATURE SELECTION. ONE ROW CAN DESCRIBE MULTIPLE FEATURES (E.G. THE MINIMUM, MAXIMUM, MEAN AND STANDARD DEVIATION OF A PROPERTY)

Nr	Feature	Description
1	Flow Duration	Duration of the flow in microseconds
2-3	Tot Fwd/Bwd Pkts	Total packets in the fwd/bwd direction
4-5	TotLen Fwd/Bwd Pkts	Total size of packets in fwd/bwd direction
6-13	Fwd/Bwd Pkt Len	Min, Max, Mean, Std size of packet in fwd/bwd direction
14-23	Fwd/Bwd IAT	Total, Min, Max, Mean, Std time between two packets sent in the fwd/bwd direction
24-35	Flag Counts	Flag Counts PSH, URG, SYN, FIN, RST, ACK, URG, CWE, ECE in Fwd/Bwd and both directions. (0 for UDP)
36-37	Fwd/Bwd Header Len	Total bytes used for headers in the fwd/bwd direction
38-40	Fwd/Bwd/Tot Pkts/s	Number of fwd/bwd/tot packets per second
41	Flow Byts/s	Number of flow bytes per second
42-45	Pkt Len	Min, Max, Mean, Std packet length of a flow
46-49	Flow IAT	Min, Max, Mean, Std packet inter-arrival time in fwd/bwd direction
50	Down/Up Ratio	Download and upload ratio
51	Pkt Size Avg	Average size of packet
52-53	Fwd/Bwd Seg Size Avg	Average size observed in the fwd/bwd direction
54-55	Fwd/Bwd Byts/b Avg	Average number of bytes bulk rate in the fwd/bwd direction
56-57	Fwd/Bwd Pkts/b Avg	Average number of packets bulk rate in the fwd/bwd direction
58-59	Fwd/Bwd Blk Rate Avg	Average number of bulk rate in the forward direction
60-61	Subflow Fwd/Bwd Pkts	Average number of packets in a subflow in the fwd/bwd direction
62-63	Subflow Fwd/Bwd Byts	Average number of bytes in a subflow in the fwd direction
64-67	Active Time	Min, Max, Mean, Std time a flow was active before becoming idle
68-71	Idle Time	Min, Max, Mean, Std time a flow was idle before becoming active
72-73	Init Fwd/Bwd Win Byts	TCP window size in the fwd/bwd direction
74	Fwd Act Data Pkts	Count of fwd packets with at least 1 byte of TCP payload
75	Fwd Seg Size Min	Minimum segment size in the forward direction
76	Int/Ext Dst IP	0 if Dst-IP of a flow belongs to Blue Teams subnet, 1 if external
77	L3/L4 Protocol	0 for TCP, 1 for UDP, 2 for ICMP

2) Feature Selection

To identify the best set of features, we removed correlating and irrelevant features by applying a recursive feature elimination scheme based on random forest Gini importance scores [14].

In each iteration, we trained a random forest classifier with the dataset from LS17 and all the considered features. Afterwards, we removed the feature with the lowest score from the set of considered features. Thus, we obtained a feature ranking, where the one that is first removed has the lowest rank. Eliminating features one by one is crucial, as importance scores can spread over multiple features with redundant information (i.e. if multiple important features are strongly correlated, their scores can all be low in a particular iteration).

The 20 most important features according to our feature selection are listed below in descending order of importance (except for the last two features, which we included in the feature set based on preliminary evaluations).

Tot Fwd Pkts, Flow IAT Mean, Fwd IAT Max, Flow Pkts/s, Bwd Pkt Len Min, FIN Flag Cnt, Init Fwd Win Byts, Active Mean, Bwd IAT Mean, Bwd Pkt Len Std, Fwd Seg Size Min, Fwd Pkt Len Std, Tot Bwd Pkts, Bwd Header Len, Subflow Fwd Byts, Subflow Bwd Pkts, Fwd IAT Tot, Flow IAT Max, Int/Ext Dst IP, L3/L4 Protocol

E. Model Selection

We tested a variety of different supervised models on our data: Artificial Neural Network, Support Vector Machine, Logistic Regression, Naive Bayes, K-Nearest Neighbors and Random Forest (RF). The main difficulty in our task was that the distribution of the background traffic was different in the LS17 and LS18 data, as benign and attack traffic profiles change every year. However, the distribution of the C&C session features hardly varies, due to the fact that the same tool (Cobalt Strike) is used to maintain these sessions. We found that RF performed best under these circumstances. Furthermore, RF models are highly efficient and require low training and inference times, which is decisive for real-time deployments.

1) Model Configuration

As a baseline model, we used an RF classifier with default configurations from scikit-learn [15] (i.e. an ensemble of 10 fully expanded trees). However, this resulted in large trees (30,000 nodes for the model trained on LS17, 70,000 for LS18) and we found that constraining the maximal tree-depth significantly increased the robustness of our model. We empirically found that a maximum tree-depth of 10 drastically reduced the node count (to 700 for LS17 and 900 for LS18). However, reducing the depth further had a negative impact on the performance. Moreover, we found that increasing the number of trees to 128 further improved the robustness and prediction quality with negligible impact on computational cost. In the following, we refer to configurations with a maximum depth of 10 and 128 trees as “tuned” configurations.

2) Robustness Against Camouflage

In the following, we analyze possible attack vectors against our model, assuming a white-box scenario where the attacker has full knowledge of the model and the features we deploy. We focus on two strategies that the attacker can follow: modifying Cobalt Strike’s C&C configuration, and altering the C&C flows by other means (e.g. by changing the network stack on the infected machines).

a) Changing the appearance of the C&C sessions using Cobalt Strike

As our model detects C&C sessions maintained using Cobalt Strike, we first analyze the options this framework provides to alter their appearance. The two main parameters the Red Team can use during the exercise are the sleep-period and jitter of a C&C session. The sleep-period defines the time interval used to periodically contact the C&C server. The jitter configures the deviation from this periodicity. Our features are

invariant to both of these parameters, as they focus on timing statistics within single connections and do not depend on the time elapsed between the periodic connections of a C&C session.

Cobalt Strike's Malleable C2 tool [16] allows the custom design of the HTTP headers of the packets exchanged within C&C sessions to avoid detection. However, our model does not rely on features extracted from HTTP headers.

We conclude that bypassing detection of our model by altering Cobalt Strike's C&C configurations is infeasible as our features are invariant to the options the framework provides.

b) Identifying attack vectors for manipulating feature values
Our classifier identifies flows that look like Cobalt Strike C&C channels. To avoid this, an attacker might attempt to camouflage these C&C flows as normal traffic for the given network.

We observe that most of the feature values can be altered either by injecting additional packets (to manipulate statistics such as inter-arrival time or packet counts) or by altering the packet sizes (which affects features such as the download size). Many of these tampering attempts could be prevented by additional checks in the feature extraction phase (e.g. sequence number checking for packet injections). However, since this is computationally expensive, we assume that the defenders cannot do this.

To simulate the robustness of our model in such scenarios, we conducted experiments involving tampering with the feature values, as described in Section 4.D.

4. EVALUATION

In this section, we evaluate our classifiers based on data recorded by the Swiss Blue Team from Locked Shields 2017 and 2018. After providing more details about the methodology (Subsection A), we evaluate precision and recall (Subsection B), runtime (Subsection C), robustness against camouflaging (Subsection D) and incomplete traffic captures (Subsection E).

A. Methodology
In this section, we summarize the datasets that we used for the evaluation, the environment in which we conducted the experiments and the parameters that we used.

1) Datasets

To evaluate the performance of our models, we used the complete LS17 dataset for training and the LS18 dataset for testing and vice versa. Therefore, our evaluation corresponds to a case where our classifier is used for classifying previously unseen data in a different network. In the following, we will refer to models trained on the full LS17 or LS18 datasets as LS17-models and LS18-models, respectively (cf. Table III).

In Table II, we summarize the baseline information about the datasets that we used for the evaluation.

TABLE II: BASELINE INFORMATION ABOUT THE DATASETS USED.

Dataset	Size	Packets	Flows	C&C Flows
LS17	114 GB	288'940'662	9'070'828	1'239'041 (13.7%)
LS18	216 GB	557'783'930	16'379'346	1'818'006 (11.1%)

2) Environment

We conducted all experiments and calculations on a virtual machine running Ubuntu 16.04 (64 bit), with 10 Intel Xeon E5-2699 cores and 16 GB RAM. The implementation was based on Python 3.6 and scikit-learn (0.19.2) [17].

3) Parameters and Models

We evaluated two configurations of our classifier: one with the default scikit-learn parameters [15], and the other with the tuned parameters described in Section 3.E. We refer to these configurations as "baseline" and "tuned" and summarize them in Table III. We trained all models using the 20 features obtained from the recursive feature elimination scheme described in Section 3.D.

TABLE III: CHARACTERIZATION OF MODELS USED IN OUR EVALUATION.

Model	Training data	Testing data	RF size	RF depth
LS17-baseline	LS17	LS18	10 trees	unconstrained
LS17-tuned	LS17	LS18	128 trees	10
LS18-baseline	LS18	LS17	10 trees	unconstrained
LS18-tuned	LS18	LS17	128 trees	10

B. Precision/Recall

We used widespread metrics precision (i.e., the percentage of reported C&C flows that are actual C&C flows) and recall (i.e., the ratio between the correctly identified C&C flows and all the C&C flows present in the dataset) to measure the prediction quality

of our models. High precision is particularly important in the given task because a high number of false positives would mislead the defenders during their operation.

Table IV lists the precision and recall scores for all models. We repeated the evaluation ten times with different random seeds to train the models, and we report the medians of the results. The results show that all models achieve high precision and recall while tuned configuration clearly outperforms the baseline configuration.

TABLE IV: THE TUNED MODELS ACHIEVE HIGH PRECISION AND RECALL (MEDIANS)

Model	Precision	Recall
LS17-baseline	0.94	0.98
LS17-tuned	0.99	0.98
LS18-baseline	0.98	0.86
LS18-tuned	0.99	0.90

C. Runtime

In this experiment, we evaluate the runtime of three phases:

1. Extracting features from the training dataset
2. Training the model
3. Applying the model on the testing dataset

In Table V, we report the time it takes to extract features from both datasets (using CICFlowMeter). We note that the feature extraction tool extracts all 77 features from Table I. The runtime could be significantly improved by calculating only the 20 selected features and by using a more efficient implementation.

TABLE V: FEATURE EXTRACTION TAKES LESS THAN 45 MIN (LS17) AND LESS THAN 90 MIN (LS18) FOR DATASETS CONTAINING ABOUT 38 HOURS OF NETWORK TRAFFIC.

Dataset	Runtime	Extracted Flows
LS17	42 min	9'070'828
LS18	85 min	16'379'346

In Table VI, we report the time it takes to train and test the model on both datasets. As above, we point out that the training phase is not time-critical as it is done prior to the exercise. As the results show, running predictions on the whole dataset takes less than one minute. In a practical deployment, the inference would be performed on much smaller sets of samples, which makes real-time detection feasible.

TABLE VI: TRAINING AND INFERENCE TIMES OF THE BASELINE AND TUNED MODEL CONFIGURATIONS

Model	Training time	Inference time
LS17-baseline	120 s	6 s
LS17-tuned	1117 s	50 s
LS18-baseline	390 s	4 s
LS18-tuned	2828 s	30 s

D. Robustness Against Camouflaging

In this experiment, we simulate an attacker attempting to camouflage C&C flows as normal traffic. To model an attack against a particular feature, we replace the feature values in the malicious samples (i.e. the C&C flows) with values randomly subsampled from benign samples. As a result, this feature no longer helps in distinguishing C&C flows from normal flows.

In Figure 3 (LS17) and Figure 4 (LS18), we plot the precision and recall of the respective models depending on the number of tampered features. The results hold under the assumption that an attacker that attacks n features would target the n most relevant features according to Section 3.D. (which is a promising strategy). We evaluate the impact of tampering with 5 to 14 features on each model with 10 different random seeds and plot the median as well as the 95% confidence interval.

The results show that the tuned model reacts much less sensitive to camouflaging attempts and achieves high performance even if many features are tampered with (precision falls below 90% when manipulating >12 features). Recall of the LS18 model drops sharply when attacking more than 5 features, however, its precision remains high, meaning that the predictions the model makes are still reliable. Further, we observe that the variance among the tuned models is much lower than that of the baseline models.

FIGURE 3. ACHIEVED PRECISION AND RECALL FOR LS17 IF AN ATTACKER TRIES TO CAMOUFLAGE C&C FLOWS. OUR TUNED MODEL IS ROBUST AGAINST TAMPERING, FOR UP TO 10 FEATURES.

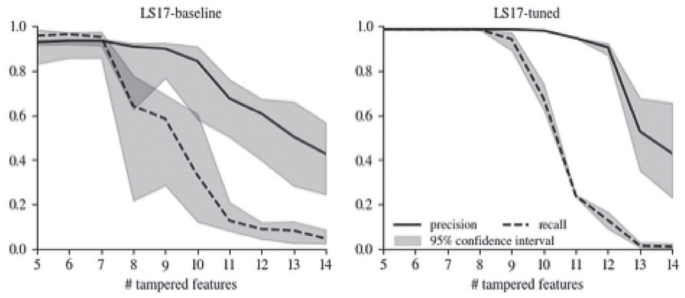
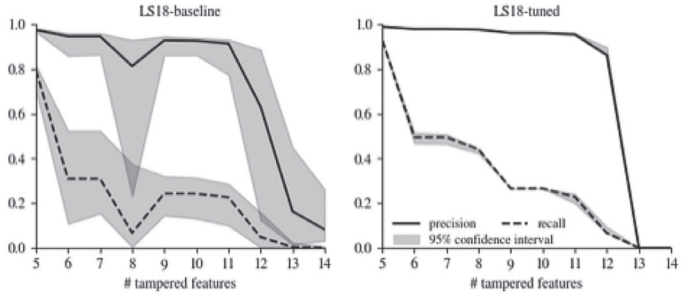


FIGURE 4. ACHIEVED PRECISION AND RECALL FOR LS18 IF AN ATTACKER TRIES TO CAMOUFLAGE C&C FLOWS. OUR TUNED MODEL ACHIEVES A HIGH PRECISION EVEN IF 12 FEATURES ARE ATTACKED BUT THE RECALL DROPS.

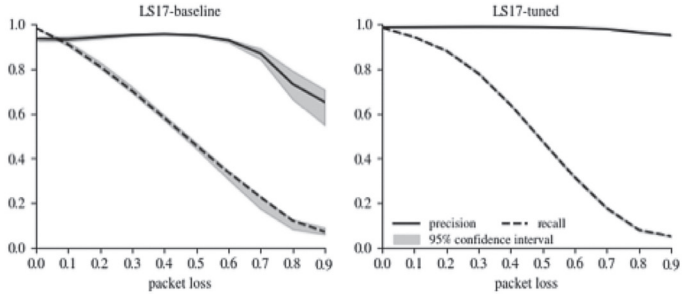


E. Robustness Against Packet Loss

In this experiment, we evaluate the impact of packet loss, which could occur due to the limited resources of the defenders to capture packets in real time during the exercise. We simulate this by randomly dropping between 10 and 90 percent of the packets.

The results in Figure 5 show that the tuned models achieve high precision (> 95%) even for 90% packet loss. This means that even for high losses, the raised alerts stay accurate. However, the recall decreases approximately linearly with the packet loss. Presumably, this is because C&C flows with too many dropped packets are no longer recognized as such, while the model still detects less affected flows.

FIGURE 5. IMPACT OF PACKET LOSS ON THE LS17-MODEL. THE CURVES SHOW THE MEAN VALUES OVER 10 MEASUREMENTS. IN OUR TUNED MODEL, PACKET LOSS HARDLY IMPACTS PRECISION.



5. DISCUSSION

In this section, we discuss the outcomes of the experiments conducted in this paper as well as details of possible real-world deployments and potential extensions.

A. Identifying C&C Servers

The ability to detect individual C&C flows can obviously be used to identify C&C servers (the destinations of such flows) and compromised hosts (the sources of the flows). In an additional experiment, we observed that running our system for a short time period of 30 minutes at the beginning of the exercise (11am-12 pm in Locked Shields 2018) was enough to identify most of the C&C servers (10 out of 12 listed in the Cobalt Strike reports). We further observed 5 different source IP addresses from the Blue Team's network communicating with these servers, suggesting that these hosts had been compromised at this point in time.

B. Running Multiple Models in Parallel

In this paper, we used datasets from two occurrences of Locked Shields: one to train the model, and the other to test it. In the future, when more datasets are available, we suggest training multiple models and conducting live classification during the exercise on all of them. This would make it even harder for the Red Team to camouflage C&C traffic as benign flows, because it needs to match the features of benign flows in multiple different models (while the features of C&C flows are similar in each model). Performing the inference only slightly increases the computational cost and is thus feasible during the exercise. Since we have data from only two iterations of Locked Shields, we could not evaluate this approach.

C. Practical Deployment for Future Locked Shields Exercises

In order to use our system in the next Locked Shields exercise, a Blue Team needs to perform three steps:

1. Train one or multiple models with labeled data from past exercises
2. Prepare the VM to record network traffic and compute the features
3. Run the trained models with the recorded features during the exercise

Step 1 is not time-critical and can be done at any time prior to the exercise. To counteract camouflaging attempts by the Red Team, we suggest using data from different years and training multiple models (cf. Section 5.B).

For Step 2, the Blue Team can use any tool to capture the traffic (no payloads required) and calculate the flow features. In our experiments, we used CICFlowMeter; however, more efficient implementations are possible.

Step 3 consists of feeding the extracted features to one or more models. Information about detected C&C flows can be passed to an intrusion alert system used by the defenders to coordinate security responses.

As our evaluation shows, our classifier is able to predict C&C flows with 99% precision and over 90% recall. By evaluating the system on two datasets originating from two different occurrences of Locked Shields (2017 and 2018), we provided strong evidence for the success of a deployment in future exercises on previously unseen data.

In an additional experiment, we simulated a real-case deployment, where we applied our system for a short 30 minutes time interval in the first phase of the LS18 exercise. There, our system unveiled almost the complete C&C infrastructure used by the Red Team (10 out of 12 C&C Servers).

D. Challenges and Deployment in Other Environments

In this paper, we have focused on a very specific use case for C&C detection (Locked Shields, Cobalt Strike). One of the main limitations of supervised-learning-based systems is that while they are highly effective in detecting anomalies that were labeled in the training set, they fail to detect new and unknown attacks. A further challenge is that the distribution of the legitimate background traffic may strongly vary among different networks.

By expanding the training data with more C&C traffic types and including a wider range of legitimate traffic profiles, our approach could be adapted for deployment in other environments. Moreover, data augmentation techniques such as domain randomization – currently applied with great success in the deep learning domain – are other promising paths towards broader generalization. For instance, OpenAI recently developed a human-like robotic hand to manipulate physical objects with unprecedented dexterity [18]. The training was performed solely in a simulated environment, but by randomizing the physical properties in the simulation, the final model generalized well enough to be deployed on a real physical hand. Although our application is very different, the same concepts could be applied to network traffic data to obtain richer training sets leading to more robust detection systems.

6. CONCLUSION

In this paper, we present a system for identifying C&C channels using supervised machine learning. As a typical use case for such a system, we focus on Locked Shields, the world's largest cyber defense exercise. Our evaluation shows that the system could

be deployed by defenders in this exercise and that it identifies C&C traffic with high precision and recall. We use real data from one participating Blue Team and show that if this team had trained the classifier with the data from 2017, it would have identified C&C channels in Locked Shields 2018 with 99% precision and 98% recall. Further, running the system during a time interval of just 30 minutes in LS18 would have been enough to identify 10 out of 12 C&C servers used by the Red Team.

Acknowledgments

We thank the Swiss Blue Team for sharing their data and expertise with us and for their constant support throughout this project.

REFERENCES

[1] "How a British SMB survived a nightmarish cryptolocker ransom attack | Security | Computerworld UK," [Online]. Available: https://www.computerworlduk.com/security/how-british-smb-survived-nightmarish-cryptolocker-ransom-attack-3677593/.

[2] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis and others, "Understanding the mirai botnet," in *USENIX Security Symposium*, 2017.

[3] "Locked Shields 2017," [Online]. Available: https://ccdcoe.org/ locked-shields-2017.html.

[4] A. H. Lashkari, G. D. Gil, J. E. Keenan, K. Mbah and A. A. Ghorbani, "A Survey Leading to a New Evaluation Framework for Network-based Botnet Detection," in *Proceedings of the 2017 the 7th International Conference on Communication and Network Security*, 2017.

[5] M. Feily, A. Shahrestani and S. Ramadass, "A survey of botnet and botnet detection," in *Third International Conference on Emerging Security Information, Systems and Technologies, 2009. SECURWARE'09*.

[6] B. Rahbarinia, R. Perdisci, A. Lanzi and K. Li, "Peerrush: Mining for unwanted p2p traffic," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2013.

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

[8] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda and C. Kruegel, "Disclosure: detecting botnet command and control servers through large-scale netflow analysis," in *Proceedings of the 28th Annual Computer Security Applications Conference*, 2012.

[9] "CCDCOE News (26 April 2018)," [Online]. Available: https://ccdcoe.org/more-1000-cyber-experts-30-nations-took-part-locked-shields.html.

[10] "Cobalt Strike Reporting," [Online]. Available: https://www.cobaltstrike.com/help-reporting.

[11] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun and A. A. Ghorbani, "Characterization of Encrypted and VPN Traffic using Time-related," in *Proceedings of the 2nd international conference on information systems security and privacy (ICISSP)*, 2016.

[12] "The Bro Network Security Monitor," [Online]. Available: https://www.bro.org/.

[13] "CICFLOWMETER A network traffic Biflow generator and analyzer," [Online]. Available: http://www.netflowmeter.ca/.

[14] G. Louppe, L. Wehenkel, A. Sutera and P. Geurts, "Understanding variable importances in forests of randomized trees," in *Advances in neural information processing systems*, 2013.

[15] "sklearn RandomForestClassifier," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[16] "Cobalt Strike 3.11 Manual," [Online]. Available: https://www.cobaltstrike.com/downloads/csmanual311.pdf.

[17] "scikit-learn Machine Learning in Python," [Online]. Available: https://scikit-learn.org.

[18] OpenAI, "Learning dexterous in-hand manipulation," 2018.