Journal of Network and Computer Applications 193 (2021) 103212

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

CREME: A toolchain of automatic dataset collection for machine learning in intrusion detection

Huu-Khoi Bui [a], Ying-Dar Lin [a], Ren-Hung Hwang [b,*], Po-Ching Lin [b], Van-Linh Nguyen [b,c], Yuan-Cheng Lai [d]

[a] Department of Computer Science, National Chiao Tung University, Hsinchu City 300, Taiwan
[b] Department of Computer Science and Information Engineering, National Chung Cheng University, Chiayi County 621, Taiwan
[c] Department of Information Technology, TNU-University of Information and Communication Technology, Thai Nguyen, Viet Nam
[d] Department of Information Management, National Taiwan University of Science and Technology, Taipei City 106, Taiwan

ARTICLE INFO

Keywords:
Dataset toolchain
Intrusion detection
Machine learning
Security dataset
Dataset generation
Dataset evaluation
Multiple data sources

ABSTRACT

Intrusion detection is one of the most common approaches for addressing security attacks in modern networks. However, given the increasing diversity of attack behaviors, efficient detection becomes more challenging. Machine learning (ML) has recently dominated as one of the most promising techniques to improve detection accuracy for intrusion detection systems(IDS). With ML-based approaches, a quality dataset for training holds the key to gain high detection performance. Unfortunately, there are few methods to assess the dataset quality, and specifically for ML training. This work presents an automated toolchain, termed CREME (**C**onfiguration, **RE**production, **M**ulti-dataset, and **E**valuation), to generate a dataset and measure its quality and efficiency. CREME integrates various tools to automate all stages of configuration, attack and benign behavior reproduction, data collection, feature extraction, data labeling, and evaluation. CREME can also automatically collect and generate a dataset from multiple sources such as accounting, network traffic, and system logs. Compared with the available datasets in the same category, experiment results show that the datasets generated by CREME contribute up to 20% better performance to ML-based IDS in terms of coverage. They also have significantly better efficiency than most other datasets. The CREME source code is available at https://github.com/buihuukhoi/CREME.

1. Introduction

Dataset is the key to the success of a machine-learning-based intrusion detection system (ML-based IDS). Without a suitable dataset, ML-based IDS detection performance can be even worse than conventional approaches such as signature-based IDS. Choosing correct data sources when generating a dataset is vital for increasing the behavioral diversity and further enriching data inputs for ML training — which is the key to high detection performance. Data sources can broadly be classified into three types: network packets, accounting, and system logs. Several recent papers (Singh et al., 2015; Song et al., 2011; Moustafa and Slay, 2015; Kang et al., 2019; Koroniotis et al., 2019) have proposed to generate datasets for specific training requirements and applications. For example, Zhang et al. (2019) are in favor of using system logs for host-based IDS while Shi et al. (2019) prefer network traffic for network-based IDS. Both of these approaches have their shortcomings. Given the increase in the use of secure protocols in communications, decoding encrypted packets in the network and classifying them is already a challenge (Xing and Wu, 2020). On the other hand, detecting malware running in a host cannot rely only on inspecting the network packets (Hwang et al., 2020).

Combining multiple types of datasets to enhance attack detection performance is an emerging approach. The data from numerous sources can significantly contribute to exposing the entire path of an attack. For example, papers by Beer et al. (2017) and Turcotte et al. (2017) propose a simple combination of network packets and system logs for generating synthetic datasets. These authors found that a system log can record relevant information about malware activities that is impossible to find in network traffic. Moustafa (2019) collected a dataset from both network packets and accounting sources and found that accounting log entries such as resource usage statistics are critical for exposing mining and DoS attacks. Unfortunately, most of the collecting methods are

* Corresponding author.
E-mail addresses: buihuukhoi@cs.nctu.edu.tw (H. Bui), ydlin@cs.nctu.edu.tw (Y. Lin), rhhwang@cs.ccu.edu.tw (R. Hwang), pclin@cs.ccu.edu.tw (P. Lin), nvlinh@ictu.edu.vn (V. Nguyen), laiyc@cs.ntust.edu.tw (Y. Lai).

https://doi.org/10.1016/j.jnca.2021.103212
Received 20 March 2021; Received in revised form 8 August 2021; Accepted 27 August 2021
Available online 7 September 2021

difficult to reproduce. Critically, there have also been few studies on providing a toolchain to validate the quality of the generated datasets.

This work presents our first attempt at creating an automated toolchain for generating and validating datasets from multiple sources. The research addresses three challenges as follows. (1) *Correct combination of dataset types*. The goal is to increase information about attacks or legitimate activities that benefit the ML-based IDS. For many available datasets, it is unclear how to quantify the amount of data per type and specify the combination order when designing a testbed. Mixed data, without a correct configuration, will probably decrease the accuracy of ML training. (2) *Reproducible Datasets*. Generally, most studies evaluate the ML models with public datasets. Unfortunately, there is a disparity between a public network/host's activities and internal sites (e.g., in enterprise companies). Reproducing a dataset from the sources of the internal networks/hosts, such as protection targets, will increase the detection accuracy of the ML and reduce the negative impact of the disparity. (3) *Automation*. Data collection, feature extraction, data labeling, and evaluation are often time-consuming tasks, let alone the tremendous challenge of carrying them at scale. In this case, the automation of doing such tasks is vital.

Addressing these challenges, our built-in toolchain can automatically carry out data collection, feature extraction, data labeling, and assessing a dataset's quality from various sources. This toolchain consists of a centralized controller that collects and aggregates reports from the clients at the network branches or distributed hosts. The central server can also decode periodic messages from monitoring software and statistical programs, e.g., *Atop*. In a typical configuration, a testbed runs on a honeynet of virtual machines. Target servers are under many types of notorious attacks, e.g., DDoS attacks or brute force. A data logger server is then designed to collect data from all these target servers periodically. This research aims to answer the following essential questions: (1) What is the quality of a generated dataset in terms of quantified metrics, such as coverage and efficiency? (2) Whether ML can work effectively on different data sources to detect specific security attacks? (3) Which are the critical factors and lessons learned to build a useful dataset for ML-based IDS?

1.1. Contributions

The main contributions of this work are summarized as follows. First, we provide an open-source automated framework for collecting multiple-sources datasets that the other researchers can use to do in their enterprise environment for specific research. Second, this work provides a basic validation to evaluate the generated datasets. The evaluation results demonstrate the advantages of our collection method in terms of coverage and efficiency, compared to several typical IoT datasets and conventional IT datasets. The datasets can partially enrich data for causality-inspired ML/DL-based IDS research. To the best of our knowledge, the toolchain is the first attempt of its kind.

1.2. Organization of the paper

The rest of the paper is organized as follows. In Section 2, we present earlier work on building a dataset for the ML-based IDS. In Section 3, we describe the background of attack models and ML techniques for detection. In Sections 4 and 5, the challenges of doing automated dataset generation and evaluation frameworks are described. In Section 6, we describe our design of the toolchain and the implementation of a testbed in detail. In Section 7, the experiment results are presented. Finally, conclusions and future work are given in Section 8.

2. Related work

This section discusses several datasets that have been specifically designed for ML-based IDS and the methods to validate their effectiveness. To this end, a comparison of the performance of several datasets is also given.

There are two approaches to collect datasets. The first one is to collect data packets in conventional IP-based networks. For example, the study of Singh et al. (2015) provides a synthetic network-traffic dataset with two labeling levels for the IDSs. The dataset is generated based on NSLKDD dataset's basic characteristics that do not contain activity-based labeling. Kyoto 2006+ dataset (Song et al., 2011) is collected with three years of real traffic data from honeypots. All traffic data are labeled by inspecting them using security software. Recently, Moustafa and Slay (2015) presented a comprehensive approach to collect a UNSW-NB15 dataset by using the IXIA PerfectStorm tool. The newer approach to collect IoT datasets where data row can be IP or non-IP traffic. For IP-based traffic, the data are collected through capturing packets at the network between the IoT gateways and the central IoT applications, e.g., from MQTT, DNS, FTP, HTTP, and SSH protocols. 5-tuple (a source IP address/port number, destination IP address/port number, and the protocol) is then used as the unique field to identify traffic from different users. For non-IP traffic, the data are often collected through capturing packets at the IoT gateways. As designing for low-cost devices, the data from these non-IP protocols (e.g., MQTT) often support several fields only, such as device address, timestamp, state, value. In this case, the device address in the IoT packets will be used as the unique field to separate the traffic from different devices. As a typical example in the work of Anagnostopoulos et al. (2020), the data row of a Bluetooth packet includes several simple fields (source address, destination address, data length, method of protocol, service, value (*0x00000004, 0x00000001, 1570733035.823057, 43, man, 4, 62*)).

For dataset collection, the authors in Kang et al. (2019) present an IoT dataset that consists of various network attacks on two typical smart home devices. The Bot-IoT dataset (Koroniotis et al., 2019; Khraisat et al., 2019) combines legitimate and attack traffic from a real testbed environment. The authors present a simple approach to quantify the quality of the dataset using Correlation Coefficient and Entropy. The IoT-23 dataset (Laboratory, 2020) contains both attack behavior (20 malicious scenarios from 12 malware on Raspberry Pi) and benign activities (collected from 3 smart IoT devices). Since the studies focus on the network-based IDS, that datasets (Singh et al., 2015; Song et al., 2011; Moustafa and Slay, 2015; Koroniotis et al., 2019; Laboratory, 2020) consists of network traffic only. With recent developments of machine learning (ML) and deep learning (DL) technologies, many scholars start eyeing collecting multi-source datasets to enrich data for generative ML/DL learning models. For example, the studies in Beer et al. (2017), Turcotte et al. (2017), Hassan et al. (2020), Cinque et al. (2020) present the methods to generate a dataset from both traffic data and system logs. In the other work, Moustafa (2019) presented a Ton_IoT dataset that combines both network traffic and accounting data.

However, researchers need to identify whether their built-in datasets are effective in specific measurements and how they are competitive with many available datasets. For this issue, Haider et al. (2017) presents the first attempt to evaluate an NGIDS-DS dataset in terms of attack and benign coverage. The shortcoming is that the work did not assess and correlate the dataset's quality with ML-based IDS systems. Gharib et al. (2016) and Sharafaldin et al. (2018) introduce several more evaluation metrics such as network configuration, traffic coverage, labeled dataset, available protocols, attack diversity, anonymity, heterogeneity, feature set, and metadata. Table 1 shows the existing datasets for a security domain, where *nw* is network traffic; *sl* is syslog; *ac* is accounting; *cov* is coverage; *eff* is efficiency; *acc*

Table 1
Comparison of various datasets and ours.

Platform	Dataset	Data sources			Problem	Solution	Metrics			Tool-chain
		nw	sl	ac			cov	eff	acc	
Traditional network	PU-IDS (Singh et al., 2015)	O	X	X	Labeling traffic activity	Synthetic dataset based on NSL KDD	X	X	X	X
	Kyoto 2006+ (Song et al., 2011)				Common security attacks	Real traffic from honeypots	X	X	X	
	UNSW-NB15 (Moustafa and Slay, 2015)				Mixed benign and attacks behaviors	IXIA Perfect Storm framework	X	X	X	
	NGIDS-DS (Haider et al., 2017)				Realistic dataset	IXIA Perfect Storm	O	X	X	
	CICIDS (Sharafaldin et al., 2018)				Assess existing datasets	Cover 11 criteria	O	X	O	
	NDSec-1 (Beer et al., 2017)	O	O	X	Find attack composition	Framework to analyze	X	X	X	
	Unified Host and Network (Turcotte et al., 2017)				Collect enterprise networks dataset	Combination of network traffic, windows log	X	X	X	
IoT network	IoT-NID (Kang et al., 2019)	O	X	X	Collect network attacks traffic	A framework to collect	X	X	X	
	Bot-IoT (Koroniotis et al., 2019)				Realistic botnet traffic dataset	Set up realistic testbed	X	O	O	
	IoT-23 (Laboratory, 2020)				Collect real malware dataset	12 malwares on raspberry PI	X	X	X	
	Ton_IoT (Moustafa, 2019)	O	X	O	Collect heterogeneous datasets	Build a new architectural IoT testbed (3 tiers)	X	X	X	
	Our dataset (CREME)	O	O	O	Assess dataset quality, multi-data sources, automated dataset generation	A automated toolchain	O	O	O	O

X: Not supported; O: Supported.

is accuracy. The *cov* and *eff* metrics are used to evaluate a dataset's quality, and the *acc* metric is used to evaluate the ML models.

For building large-scale datasets, data collection and processing automation are vital. Al-Hadhrami and Hussain (2020) proposed a dataset auto-generation platform in IoT networks. However, the platform is not open to independent validation. Another promising framework is CALDERA,[1] which is open-source. CALDERA can enable data collection and processing automation. CALDERA also supports MITRE ATT&CK, a globally accessible knowledge base of adversary tactics and techniques. But CALDERA is not designed to generate a dataset. Inspired by the CALDERA approach, this work presents an automated toolchain to support security dataset generation and validation. The toolchain enables automation for multi-tasks: data collection, attack replay, and data processing. Further, the platform also supports generating a dataset from multiple data sources, e.g., network traffic, system log, accounting. Unlike prior work, the generated datasets are also validated in terms of coverage and efficiency in detail.

In summary, compared with the state-of-the-art studies, this work aims to collect multiple-sources datasets in heterogeneous environments, which few scholars weigh in. This dataset is particularly useful for IDS in detecting several attacks. For example, network traffic may reveal little evidence of a real threat of malware spread if the attacker compresses payload or transfers malicious content in many fragments. The abnormal behavior of the scripts (from the transferred traffic) only becomes apparent if they are active to participate in an attack, e.g., modify a specific file, open a port or send a large volume of DDoS traffic. As a result, monitoring all three data sources can benefit an IDS to figure out which data passed through the traffic filter/deep packet inspection but become malicious entities after storing in the IoT devices. Our preliminary results showed that different data source is more critical for intrusion detection at different attack stage and a combined dataset of multiple data sources can achieve the best F1 score of the intrusion detection by applying machine learning algorithms (Wang, 2021).

3. Attack model and scenarios for generating dataset

This section covers the principle concepts of advanced persistent threat attacks. Through the lens of attack models, we outline five scenarios that are used in this work to generate a dataset.

3.1. Attack model

An attack model is a combination of different attack stages to reach a specific goal. The attack model can provide a better understanding of the goal of the attacks and the behavior of adversaries before and after attacks occur. Further, understanding the attack model can benefit defenders in choosing suitable security mechanisms and identifying potential vulnerabilities in a system (Al-Mohannadi et al., 2016) There are various attack models, such as the pyramid model and graph model. MITRE ATT&CK Matrix[2] one of the best model. MITRE ATT&CK uses 12 tactics: initial access, execution, persistence, privilege escalation, defense evasion, credential access, discovery, lateral movement, collection, command and control, ex-filtration, and impact. To match the matrix with practical attacks, the MITRE Corporation created adversary simulation plans.[3] APT3 is one of the most common plans to test security issues in IT networks. However, currently, APT3 does not support IoT networks. Based on the well-organized plan in APT3, we develop a general attack scenario for IoT networks with several components borrowed from the APT3 architecture.

[1] https://github.com/mitre/caldera.
[2] https://attack.mitre.org/matrices/enterprise/.
[3] https://attack.mitre.org/resources/adversary-emulation-plans/.

3.2. Attack scenarios

Our attack scenario includes three attack stages: initial access, compromise and propagation, and complete mission. The initial access stage aims to get access to a system. Reconnaissance, scanning, and exploitation are common steps in this stage. Several other steps such as privilege escalation and defense evasion can be carried out in the compromise and propagation stage. We design five attack scenarios: Mirai botnet, ransomware, disk wipe, resource hijacking, and end-point DoS. Since launching large-scale attacks from non-IP IoT devices is a challenge as a result of their simplicity and low energy, in this work, we simulate the attacks from IP-enabled devices only. The researchers can update our attack scripts to support non-IP devices in the future. The details of the attack scenarios are as follows.

Mirai botnet: Mirai is a worm-like malware family that exploits default accounts in IoT devices via telnet and SSH services (Kolias et al., 2017; Antonakakis et al., 2017). The source code of Mirai is available on the Internet. Initially, Mirai scans all the hosts that are running a telnet service. Using a hardcoded dictionary of username/password, Mirai launches brute force attacks to find the victims' telnet accounts. After intruding on the devices successfully, the cracked username and password are reported to a CnC server. The information will later be used to login and update malicious programs to the injected devices. The targets actually become zombies in the botnet, which was commanded by the CnC server. In a volumetric attack such as DDoS, the attacker often use the CnC server to command all the live bots sending a large amount of traffic to the victim server.

Ransomware: Ransomware is one of the most damaging attacks that many cybercriminals have used to blackmail a victim. In this case, the hackers encrypt data or folders on the victim's computers or workstations. In our testbed, a server is set up with a vulnerability in the IRC service. Ransomware can exploit the vulnerability to gain access to the server with the root privilege. In order to maintain future access, a backdoor can be installed on the server. Finally, the data of the server will be encrypted by using a Bash-ransomware open-source tool.

Disk wipe: Other than encrypting the data, an attacker can choose to destroy the data on a server. The server can consequently not provide clients services or even reboot if the boot folder is wiped. For this scenario, the setup server runs a web application that contained a Ruby on Rails vulnerability for exploitation. After compromising the target, the attacker can freely install a backdoor and access the server in the future. Note that, even if the vulnerability is patched, the attacker can still connect to the server using the backdoor and reinstall other malware to wipe out data.

Resource hijacking: Instead of destroying the system, an attacker can use the system's resources for mining cryptocurrency. To simulate this scenario, a server is set up with a vulnerability on the Apache continuum service. A backdoor is then installed after exploiting the vulnerability successfully to establish a backdoor connection for further attacks.

End-point DoS: This attack simulates taking down all the services of a server by consuming all the server resources. In this scenario, the IRC service's vulnerability is exploited to intrude on the server and create a hidden root account for future access. Using the root account, the attacker can remotely install and execute the malicious code.

4. Automated dataset generation and evaluation

This section covers the measurement metrics used to evaluate the quality of a generated dataset. Table 2 shows the notations used in this work, which are classified into four categories. The attack and benign category mean two types of programs used to generate abnormal and benign behavior. The category can also denote the machines that are used to represent entities in the system. There are six types of machines: vulnerable client, non-vulnerable clients, controller, data logger server, attacker servers, malicious client. Raw data, extracted data, sub-datasets, and dataset denote data types.

Table 2
Notations used in this work.

Category	Notation	Name
Attacks and programs	$a_n, n \in [1, N]$, N: number of predefined attacks	Attacks
	$b_m, m \in [1, M]$, M: number of predefined benign programs	Benign programs
	$ser_g, g \in [1, G]$, G: number of servers	Servers
	$v_h, h \in [1, H]$, H: number of vulnerable clients	Vulnerable clients
	$nv_l, l \in [1, L]$, L: number of non-vulnerable clients	Non-vulnerable clients
	c	Controller
	dls	Data logger server
	$aser$	Attacker server
	mc	Malicious client
Data types	$r_i, i \in [1, S]$, S: number of data sources	Raw data
	$et_i, i \in [1, S]$	Extracted data
	$d_i, i \in [1, S]$	Sub-datasets
	$D, \Sigma_{i=1}^{S} d_i, \{d_i, i \in [1, S]\}$	Dataset
Metrics	cov^D	Coverage
	$eff^{d_i}, i \in [1, S]$	Efficiency
	$acc^{d_i}, i \in [1, S]$	Accuracy
	$f1score^{d_i}, i \in [1, S]$	F1 score
Parameters	$w_j, j \in [1, C]$, C: number of criteria	Weight of criteria
	$v_j, j \in [1, C]$	Value of criteria
	$maxv_j, j \in [1, C]$	Max value of criteria
	$nif^{d_i}, i \in [1, S]$	Number of important features
	$naf^{d_i}, i \in [1, S]$	Number of all features

4.1. Measurement metrics for evaluating the quality of a generated dataset

There are two metrics used in this research to validate the quality of the generated dataset: coverage and efficiency. Details of the metrics are listed as follows.

Coverage. This metric aims to measure the attack diversity, traffic completion, available protocols, features, metadata, etc. The higher coverage a dataset is, the richer data or closer to the real situation the dataset supports. In fact, with an ML-based IDS, rich data for training is crucial. The dataset should include a broad range of different attack scenarios to cover real situations. *Attack types*, *staged attack scenarios*, *data sources*, *labeled data*, *feature set*, and *metadata* criteria are the main parameters for validating the coverage metric in this work. The *attack types* criteria represent the number of attack types covered in a dataset. The attack type denotes the kind of security attacks such as DoS or port scanning. The staged attack scenario means a sequence of different attack types used by the attacker to achieve the intrusion target. For example, the Mirai scenario contains attack types such as scanning, brute force, backdoor, DDoS. The *data sources* criteria mean the number of data sources used to collect data. More data sources in consideration can give a better chance to detect sophisticated attacks. For example, if an IDS fails to detect scanning attacks with network packets, the engine still has another chance to find out the attacks by analyzing system logs. Correct *labeled data*, like a ground truth, is another important criterion for validating the detection result. For an ML-based IDS, the *feature sets* are also crucial, and extracting them is quite time-consuming, given the diversity of large-scale raw data. Other than the criteria mentioned above, *metadata* is an important tag to describe a dataset's detail for later reference. In summary, inspired by the work of Gharib et al. (2016), the coverage metric in our work is the total of weighted criteria's score. The coverage score is measured as follows:

$$cov^D = \Sigma_{j=1}^{C} \frac{w_j * v_j}{max_{v_j}} \tag{1}$$

where C is the number of criteria; w_j a weight for each criterion, and can be set to different values based on the requirements at the running time. The sum of w_j is equal to 1. v_j the value of each criterion and a non-negative integer that is not greater than max_{v_j}; max_{v_j} is the maximum value of each criterion; D is the dataset as noted in Table 2. Suppose that w_j and max_{v_j} are the sets based on the requirement of researchers.

Let consider an example by calculating the coverage scores for two datasets Kyoto 2006+ Song et al. (2011) and NDSec-1 Beer et al. (2017). As attack types and staged attack scenarios have the highest value at around ten in previous work, we set its max value to twenty to have room for more attacks. Similarly, a staged attack scenario can consist of a set of exploitation and attacks up to 20. Generally, the highest value of data sources, labeled data, and features set is set at 3 in default. However, we double the maximum value (e.g., 6) for these three metrics to enrich and open the room for future usage. The maximum value of metadata is to present a binary set, i.e., 1 (yes) or 0 (no). In summary, the considered maximum values for different criteria are $[20, 20, 6, 6, 6, 1]$. The weights are then calculated by the importance of criteria [0.4, 0.2, 0.1, 0.1, 0.1, 0.1]. Based on the values of two datasets from Table 6, the coverage scores of the two datasets were calculated as follows: $cov^{Kyoto} = \frac{0.4*8}{20} + \frac{0.2*0}{20} + \frac{0.1*1}{6} + \frac{0.1*1}{6} + \frac{0.1*1}{6} + \frac{0.1*1}{1} = 0.310$; $cov^{NDSec-1} = \frac{0.4*9}{20} + \frac{0.2*3}{20} + \frac{0.1*2}{6} + \frac{0.1*1}{6} + \frac{0.1*1}{6} + \frac{0.1*1}{1} = 0.377$. In this example, the NDSec-1 has a higher coverage score as a result of rich attack types and features.

Efficiency. Given a dataset with many features, it is essential to figure out which features in the dataset are useful for training. Generally, the ML-based IDS can work at best by using various *important* features. Because of the importance of these features, any failure to remove one of them can negatively impact on the performance of the ML-based IDS (Chandrashekar and Sahin, 2014). Selecting the correct features is still by far the biggest challenge. Efficiency is then a metric that quantifies the sufficiency of important features in the generated dataset. The efficiency metric is calculated as the ratio between the number of useful features and the number of all features in the dataset. The efficiency score is measured as

$$eff^{d_i} = \frac{nif^{d_i}}{naf^{d_i}} \tag{2}$$

where nif is the number of important features; naf is the number of all features, and d_i is the notation noted in Table 2.

Accuracy. The accuracy acc is defined by :

$$acc = \frac{TP + TN}{TP + TN + FP + FN}, \tag{3}$$

where TP is true positive, TN is true negative, FP is false positive, FN is false negative. Accuracy is a critical metric for measuring the efficiency of the ML-based IDS.

F1 score. F1 score is also used to examine the ML result on our generated datasets. The score $F1$ is defined as:

$$score = \frac{2 * precision * recall}{precision + recall}, \tag{4}$$

where *precision* measures the percentage of TP among the total of TP and FP, *recall* measures the percentage of TP among the total of TP and FN. *precision* and *recall* are defined as follows.

$$precision = \frac{TP}{TP + FP} \tag{5}$$

$$recall = \frac{TP}{TP + FN} \tag{6}$$

The accuracy metric can be used to support finding the number of useful features in a dataset or evaluate the efficiency. The idea to find the number of useful features is described as follows. Features are considered important features if they can achieve acceptable accuracy with fewer features. The acceptable accuracy is a threshold provided by the user. Let consider an example: a dataset has 100 features. We try to find the efficiency of different feature combinations (from 1–100). Let say the acceptable accuracy is 0.98 (the user can accept to lose 0.02 of accuracy, but they want to reduce the number of used features to accelerate the training or detection speed). Then we found a combination of 40 features that can achieve at least 0.98 of accuracy. So, the efficiency is $\frac{40}{100} = 0.4$.

4.2. Data collection, assessment, and challenges

The first task of generating a dataset D is to set up configuration parameters: vulnerable clients v_h, non-vulnerable client nv_l, servers ser_g, a controller c, a data logger server dls, an attacker server $aser$, a malicious client mc, a set of multiple data sources (traffic, syslog, and accounting), a set of support attacks a_n and benign programs b_m. After collecting this data, the measurement metrics (coverage cov and efficiency eff) are used to quantify the quality of D. The dataset D can include several sub-datasets, such as d_i and ground truth. The second objective is to maximize the dataset quality in terms of two metrics: cov^D, eff^{d_i}. Subject to diversity constraints, the attacks a_n need to be uncontrollably replayed, and the data of a_n must be distinguished from the data of b_m. For simplicity, we separate the collecting and measurement problem into various sub-problems, *auto-configuration*, *behavior reproduction*, *multi-data sources*, and *dataset quality*. The sub-problems are described as follows.

Auto-configuration. The goal of this task is to automatically configure for attacks a_n and benign programs b_m or the other information, e.g., hostname, ip, username, password about v_h, nv_l, ser_g, c, dls, $aser$, and mc (defined in Table 2). Performing an auto-configuration for all the programs in distributed workstations is a challenge, particularly if doing that from a remote server.

Behavior reproduction. Given attacks a_n and benign programs b_m, the task determines a method to generate malicious behaviors for a_n and benign behaviors for b_m, to reproduce the behaviors for a_n and b_m automatically. The challenge is to connect different attack stages and combine the collected data with building a consistent attack path or benign traffic. For this, we build a graphic interface to support adjusting system parameter configurations visually.

Multi-data sources. Given the behavior of attack a_n and benign programs b_m, this phase proposes a method of generating the datasets along with the ground truth from the three data sources, network traffic, system log, and accounting statistics. The challenge is to distinguish and label the attack and benign data.

Dataset quality assessment. Given a dataset D generated from different data sources, this phase — designs a toolchain to validate the output values of cov^D and eff^{d_i}, and improve the quality of the dataset D. The challenge is to calculate cov^D and eff^{d_i} metrics accurately.

5. CREME - a toolchain with dataset generation and quality assessment

The CREME (**C**onfiguration, **RE**production, **M**ulti-dataset, and **E**valuation) is a framework of multi-built-in components. The CREME can automatically generate datasets and quantify the datasets' quality. This section presents the principles of the framework and the key components in detail.

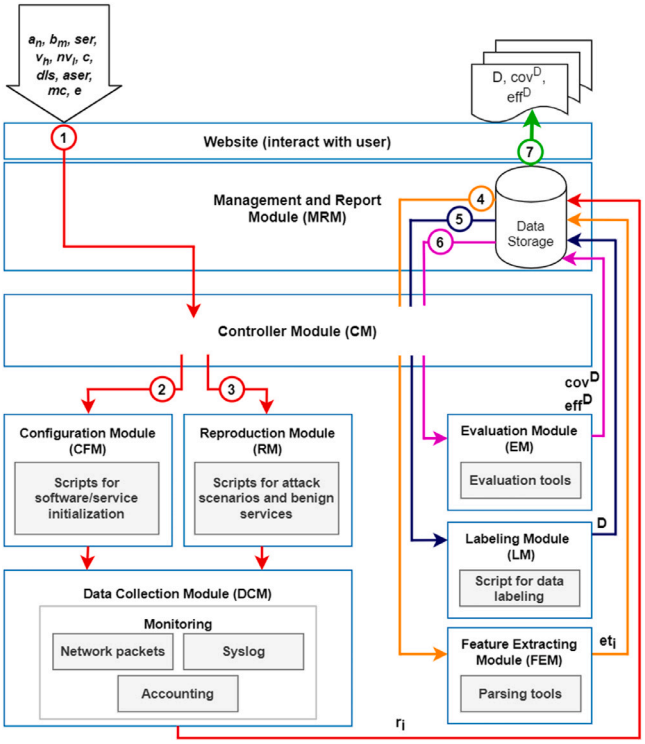

Fig. 1. CREME's architecture design.

5.1. System architecture overview

Fig. 1 shows CREME's architecture. There are seven key processing steps. In the first step, the control module collects the setting configuration (a_n, b_m, ser_g, v_h, nv_l, c, dls, $aser$, and mc) from the user/researcher (via website GUI). Based on the demand (setting), the control module calls to run software and service scripts (Step 2) and then launch attack scenario (Step 3). The control module is set at the central server and can connect with the service scripts/attack tools on the distributed workstation through gRPC/SCP tool. During the collection period, the data collected r_i at the workstations are then uploaded into the centralized server for storing. In the next steps, the data r_i are then extracted (Step 4), labeled (Step 5), and evaluated (Step 6) to retrieve the final results D for the performance analysis statistic in terms of the coverage and efficiency metrics (Step 7). Every step runs automatically. The modules are open-source code for customizing. A *centralized controller* can remotely connect to the virtual machines and execute these modules.

5.2. Centralized controller

The centralized controller consists of two independent modules: configuration and data reproduction. The objectives of each module are detailed as follows.

Controller configuration. The purpose of this module is to automate the configuration and process flow. As illustrated in Fig. 2, a controller C can connect to a data logger server dls, non-vulnerable clients nv_l, vulnerable clients v_h, servers ser_g, an attacker server $aser$, a malicious client mc to install and configure the parameters (hostname, IP address, username, password). Theoretically, the controller C can remotely connect to every machine to install and configure related services/programs without a privilege escalation attack. For examples, an Atop[4] program can be installed on the machine ser_g, vulnerable

[4] https://linux.die.net/man/1/atop.

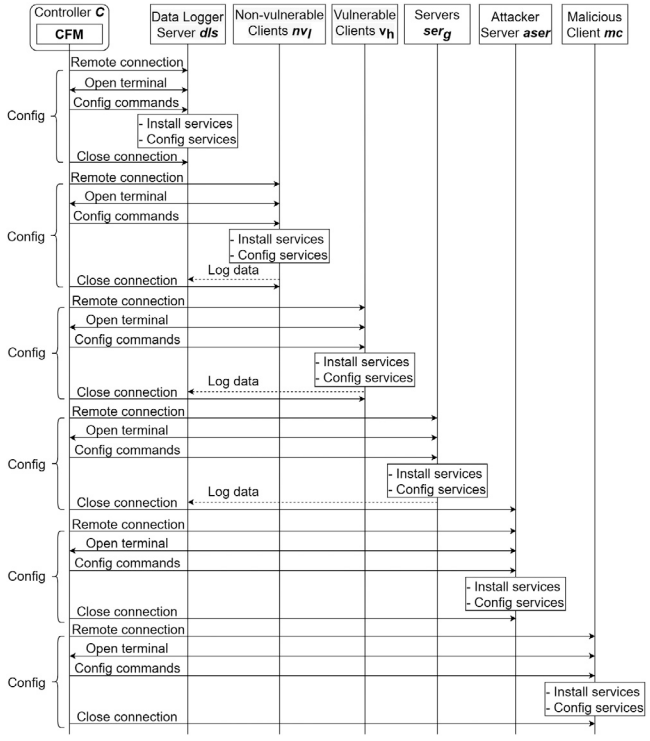

Fig. 2. The process flow of controller driven configuration.

client v_h, and non-vulnerable nv_l to collect accounting data. After the controller c completes the machine's configuration, the data log server will receive and keep the reports from other machines.

Data reproduction. This module is to reproduce attack and benign behavior, given the configured testbed. The module consists of a set of scripts for launching attack scenarios and benign services. Each attack scenario's scripts are designed to run independently but aware of what stage they are performing on, such as network or system log. Fig. 3 illustrates data reproduction for the Mirai attack scenario where the controller C commands the machines of malicious client m_c to scan all machines (non-vulnerable and vulnerable clients). The malicious client m_c can report found successfully injected machines to the attack server (CnC server) for future exploitations (i.e., exploit, transfer malware, execute malware). The controller specifically runs a process on the attacker server to periodically check the number of connected bots that will trigger a DDoS attack command when reaching a certain number of bots (predefined via the configuration). The attacker server will continuously keep track of the control command and switch to new attack targets if required.

5.3. Feature extraction and data labeling

This stage details feature extraction and labeling for the raw data collected from three sources in the previous step. The top of Fig. 4 illustrates the processing flow of two modules. Initially, the controller C commands the feature extraction module (FEM), which will get raw data from data storage and extract the features. Then the attack and benign data in the dataset are also classified by data sources (network traffic, system log, accounting statistic) at this step. For network traffic data, the pcap file packets are grouped into network flows by using the 5-tuple (source IP address, source port, destination IP address, destination port, and transport protocol). With long network flows, e.g., downloading files, they can be split into multiple sub-flows using a time window. The labeling uses 5-tuple and timestamps to index. We specifically use the timestamps for identifying the attack stage.

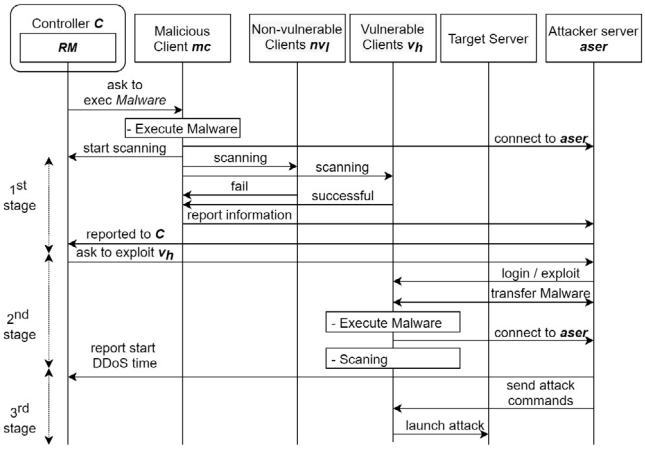

Fig. 3. The processing flow of Mirai attack-based scenario.

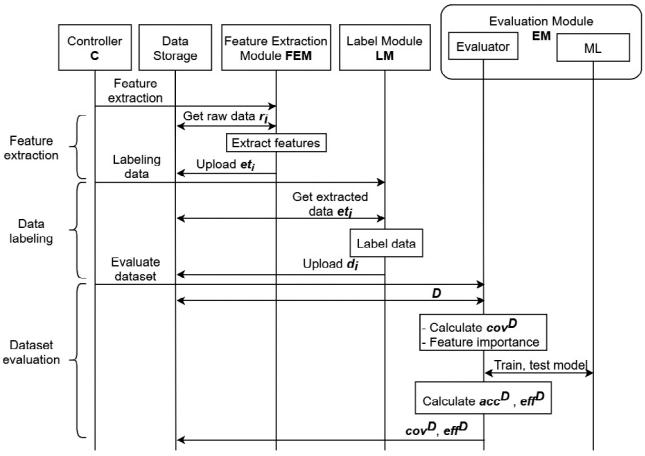

Fig. 4. The process flow of multi-dataset generation and auto-evaluation.

Labeling sub-flows is primarily based on the source and destination IP addresses. If the traffic comes from predefined legitimate machines, they are labeled as normal. In our experiment, we assume that the benign programs in vulnerable clients/injected devices will not access the target server during the attack period. As a result, we can label the abnormal traffic by filtering out the traffic flows from 5-tuple with the destination IP address of the target server. For system log data, log messages are separated into the log template and parameter list. The machine ID or process ID is the index to separates the records of different processes or malware. Similar to network traffic, to label logs, the attack programs or machines are predefined. The log flows are then labeled with their source machine/process ID. For accounting data, the *cmd* name of each process is used as the reference index for labeling data.

5.4. Auto-evaluation

Auto-evaluation is to validate the dataset quality. This study uses two metrics (coverage and efficiency) for the measurement. The process flow of the module is illustrated at the bottom of Fig. 4, i.e., auto-evaluation block. As defined in Section 4.1, the coverage metric *cov* includes the number of attack types, the number of attack scenarios, the number of data sources, the labeled data amount, the number of feature sets, and metadata. The efficiency metric *eff* is measured by a pair of bound. The higher bound is the highest accuracy from the combinations of the selected features. The lower bound is the acceptable accuracy

Table 3
Open sources and tools.

Category	Name	Functionality
Data collection	Tcpdump	capturing network packets
	Rsyslog	collecting logs
	Atop	monitor activity of processes
Features extraction	Argus	extract features for packets
	Drain	parsing raw log messages
	Atop	extract accounting's features
Attacks	Mirai	emulate Mirai botnet
	MySQL	database service
	Metasploit	hacking tool
	Metasploitable 3	vulnerable Linux VM

based on a given ML-based IDS and a provided threshold. If the ML-based IDS uses only a list of few features to achieve a certain accuracy as doing with all features, a number of the features in the list will contribute to calculating *eff*.

6. Implementation

This section covers the implementation of the dataset generation/evaluation module, and summarizes the open source and tools used in this work. To the end, we show the configuration of the testbed and overview the setting up process.

6.1. Open sources and tools

Table 3 summarizes the most important open-sources and tools used in this work. The main stages of implementation are categorized as follows.

Management and Report. We used Django (a high-level Python Web framework) to build a website that acts as a centralized GUI-based controller. The researcher can configure all the testbeds, e.g., number of machines, attack scenarios, through this portal. The website provides a dashboard and a summary report to track the launched modules' progress and results.

Configuration. *Expect*[5] is the script to connect ssh to each machine and configure the tasks, e.g., installing malware, launching an attack. *Expect* scripts are specified for the requirements of each machine. *SCP*[6] is the main tool to support uploading/downloading configured files for the tools which do not support a remote report to the centralized server, *Atop*.

Attack reproduction. For easy reproduction, we open-source tools to carry out attack scenarios in this work. For example, Mirai botnet's source code and MySQL database are collectible from the Internet. Metasploitable and Metasploit are also the available frameworks on the Internet for downloading. Note that Metasploitable3 is different from Metasploit. Metasploitable 3 is a virtual machine containing many security vulnerabilities, while Metasploit is a security tool to perform penetration tests and vulnerability exploitation.

Data collection. In this work, we use three tools, *Tcpdump*,[7] *Rsyslog*,[8] *Atop*, to collect data from network traffic, system logs, and accounting. *Tcpdump* is a common command-line packet analyzer that many experts have used to capture a packet's contents on a specific network interface. For specific domain, *Iptables*[9] is a tool for updating the routing rules and forwarding all the outgoing packets to the data log server dls where *Tcpdump* can capture and extract. *Rsyslog* is a robust and rocket-fast module for log processing. *Rsyslog* supports forward log messages to a remote server for aggregation. For accounting statistics, *Atop* is a Linux-based interactive program that monitors the occupation of sharing resources on the system level, such as CPU, memory, and disk.

Table 4
Features of network traffic and accounting.

Data sources	Features
Network packet	flgs, pkts, bytes, state, dur, mean, sbytes, dbytes, spkts, dpkts, stddev, sum, min, max, rate, srate, drate
Accounting	PID, RDDSK, WRDSK, WCANCL, DSK, CMD, MINFLT, MAJFLT, VSTEXT, VSIZE, RSIZE, VGROW, RGROW, MEM, TRUN, POLI, TSLPI, TSLPU, NICE, PRI, RTPR, CPUNR, ST, EXC, S, CPU

Feature extraction. *Argus*,[10] *Drain* (He et al., 2017), and *Atop* are the three selected tools for extracting features from the collected raw data of network traffic, logs, and accounting. *Argus*, specifically, is an open-source network activity auditing tool that can process packets at the network flow level using 5-tuple. *Argus* consists of two agents: *Argus server* and *Argus client*. *Argus server* supports reading and translating *pcap* files to the sub-flow level with a particular duration time, while *Argus client* extracts features of the flows to a CSV file. Other than recording the raw data, *Atop* is a tool for extracting the accounting data features. Each process's features can be CPU utilization, the average size of disk reading and writing, memory utilization, and so on. Features of network traffic and accounting are shown in Table 4. For the system log, *Drain* is used to parse the unstructured log messages to the structured format, e.g., including the timestamp, the source service, the log template, and the parameter list. *Drain* is the advanced version of the log parser tool demonstrated in the study (Zhu et al., 2019).

Labeling. Labeling is performed based on the prior understanding of the attack programs or injected machines defined in the configuration step. For example, a flow is labeled as an attack for the network traffic data if they come from the predefined malicious machines. Similarly, the logs from the predefined attack machines are labeled as abnormal.

Evaluation. We use the Scikit-Learn library algorithm, namely Recursive Feature Elimination and Cross-Validation Selection (RFECV), to eliminate irrelevant features based on validation scores. The goal is to figure out the best combinations of the extracted features. The variety of essential features is a combination with a minimum number of selected features while still achieving acceptable accuracy (with a provided threshold). The efficiency is scored by a division of the number of important features nif^{d_i} and the number of all features naf^{d_i}. The coverage score is provided with the ground truth information in the dataset. For further assessment, datasets are put into ML-based IDS to test and verify the detection performance.

6.2. Testbed

Fig. 5 illustrates our testbed environment. The hosts running in the testbed are based on Virtual Machines (VMs). The VM hosts can run Windows (10) or Linux (Ubuntu) kernels. The benign servers and the target servers run on Metasploitable 3 with Ubuntu 14.04. The benign servers host several legitimate services such as DNS, Web, and FTP. The attack clients run on Kali Linux. Table 5 shows the parameter setting of the machines. Without loss of generality, we set all the passwords to the same value. For data collection and pre-processing, both the accounting collection duration and time windows of sub-flows are set to 1 s. For filtering out redundant features, the features that have correlation scores of less than 0.1 with the label will be removed.

Fig. 6 is a screenshot of the toolchain dashboard. The dashboard provides an overview of the current toolchain's operation with seven phases. The dashboard provides a summary of the operations performed in each phase.

[5] https://linux.die.net/man/1/expect.
[6] https://linux.die.net/man/1/scp.
[7] https://www.tcpdump.org/.
[8] https://www.rsyslog.com/.
[9] https://linux.die.net/man/8/iptables.
[10] https://openargus.org/.

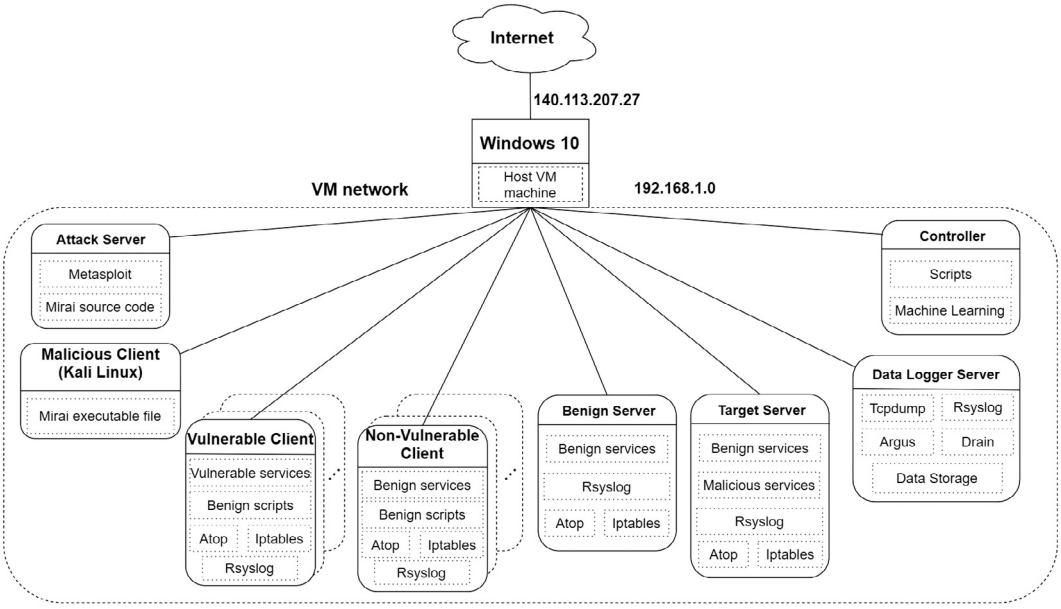

Fig. 5. The illustration of our testbed environment. The hosts are virtual machines and can run Windows (10) or Linux (Ubuntu) kernels.

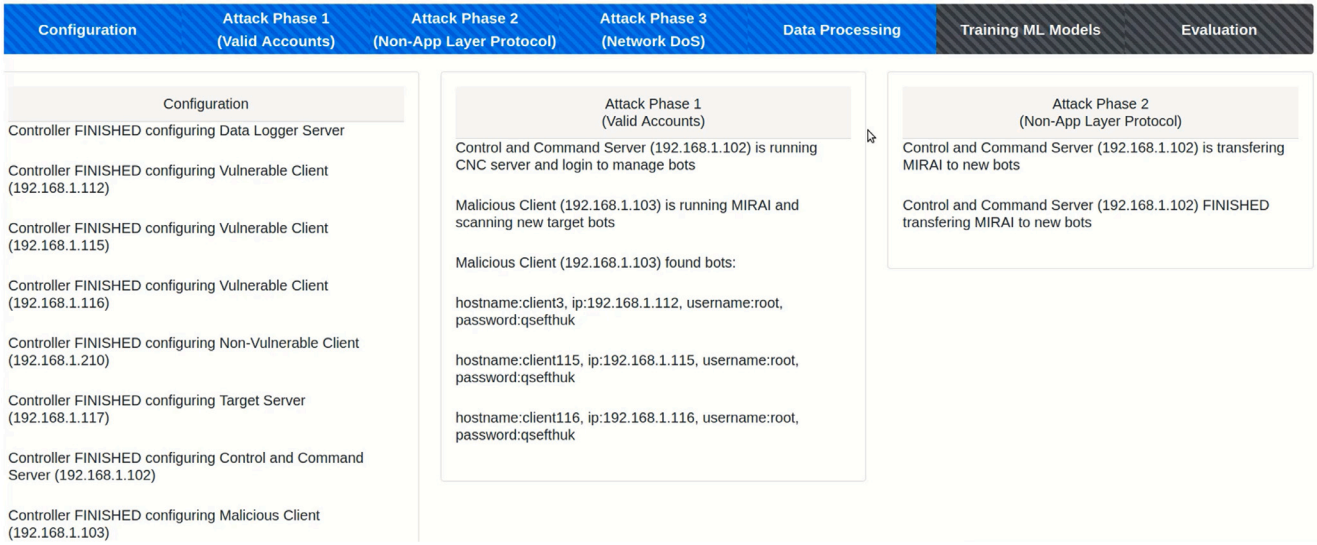

Fig. 6. The illustration of the toolchain dashboard. The dashboard has two basic parts. The upper part is a progress bar to track the steps of the toolchain. Each phase has a black color at the beginning that will change to a blue color when the toolchain run in the corresponding stage. The below part is the details of each phase.

Table 5
Configuration setting for machines in the testbed.

Hostname	IP address	Username	Password
controller	192.168.1.200	controller	qsefthuk
data-logger-server	192.168.1.164	root	qsefthuk
benign-server-1	192.168.1.21	root	qsefthuk
target-server-1	192.168.1.11	root	qsefthuk
vul-client-1	192.168.1.111	root	qsefthuk
vul-client-2	192.168.1.112	root	qsefthuk
vul-client-3	192.168.1.113	root	qsefthuk
non-vul-client-1	192.168.1.211	root	qsefthuk
non-vul-client-2	192.168.1.212	root	qsefthuk
non-vul-client-3	192.168.1.213	root	qsefthuk
non-vul-client-4	192.168.1.214	root	qsefthuk
attacker-server	192.168.1.102	root	qsefthuk
malicious-client	192.168.1.103	root	qsefthuk

7. Experiment result

This section covers the evaluation performance of the generated datasets on coverage and efficiency metric.

7.1. Dataset coverage

Table 6 shows main criteria covered by different datasets. These criteria are used as a basis for calculating the coverage scores. The calculation is based on Eq. (1). The weight step is set by any of the values in $[0.4, 0.2, 0.1, 0.1, 0.1, 0.1]$ and max_{v_j} to [20, 20, 6, 6, 6, 1] (as described in Section 4.1). The training and testing ratio is 80% and 20%, respectively. 5-fold cross-validation is also used. The results of the coverage scores is presented in Fig. 7.

We found that the CREME dataset has a roughly 20% higher coverage score than the others. CREME's dataset coverage is 0.52 compared to the past average of 0.334 in the other datasets. These positive results

Table 6
Coverage evaluation comparison of various datasets and ours.

Dataset	Attack types	Staged attack scenarios	Data sources	Labeled data	Feature set	Metadata
Kyoto 2006+	DoS, scanning, trojan, worm, vulnerability, backscatter, phising, shell code	None	Traffic	Traffic	Traffic	Yes
UNSW-NB15	Backdoors, DoS, Exploits, spam, fuzzers, generic, port scans, reconnaissance, shellcode, worms	None	Traffic	Traffic	Traffic	Yes
NGIDS-DS	Backdoors, DoS, exploits, worms, generic, reconnaissance, shellcode	None	Traffic, Audit	Traffic, Audit	Audit	Yes
NDSec-1	Botnet, brute force, spoofing, DDoS, exploits, probe, XSS, SSL proxy, SQL injection	Bring your own device, watering hole, botnet	Traffic, Syslog	Traffic	Syslog	Yes
CICIDS	Botnet (Ares), DoS, DDoS, heartbleed, infiltration, XSS, SSH brute force, SQL injection	None	Traffic	Traffic	Traffic	Yes
IoT-NID	MITM, DoS, Scanning, DDoS, brute force	Mirai	Traffic	Traffic	None	Yes
Bot-IoT	Scanning, Dos, DDoS, brute force, vulnerability, data exfiltration, key logging	Data theft, keylogging	Traffic	Traffic	Traffic	Yes
Ton-IoT	Scanning, DoS, DDoS, MITM, Ransomware, Backdoor, Injection, XSS, Password	None	Traffic, Account-ing, Telemetry	Traffic, Account-ing, Telemetry	Traffic, Account-ing, Telemetry	Yes
IoT-23	C&C, FileDownload, DDoS, Scanning, HeartBeat, brute force, injection	Mirai, Torii, Trojan, Gagfyt, Kenjiro, Okiru, Hakai, IRCBot, Muhsik, Hide and Seek	Traffic	Traffic	None	Yes
CREME	Scanning, brute force, backdoor, DDoS, privilege escalation, vulnerability, resource hijacking, ransomware, end-point DoS, data destruction	Mirai, Ransomware, End-point DoS, Mining, WipeDisk	Traffic, Syslog, Account-ing	Traffic, Syslog, Account-ing	Traffic, Syslog, Account-ing	Yes

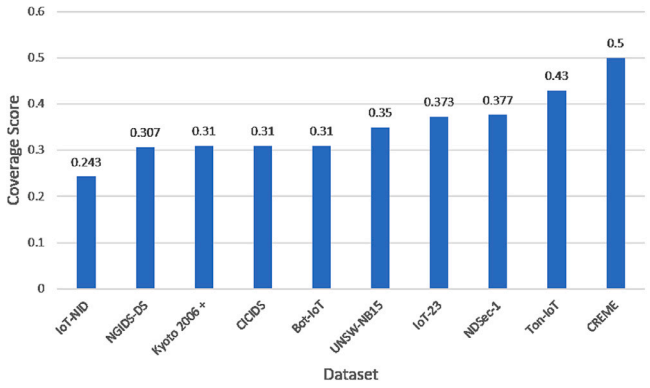

Fig. 7. The results for validating coverage score of the datasets.

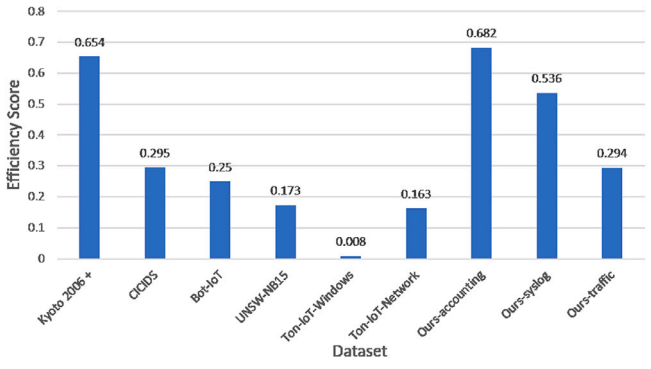

Fig. 8. The results for validating efficiency score of the datasets.

come from the fact that the CREME collects data from multiple data sources (traffic, syslog, accounting) along with the optimal feature set. If a single data source is used, the coverage score degrades to 0.4. The low score highlights the importance of having rich data for training. Data from multiple sources can give a better chance to detect the footprint of the attacks. Also, as listed in Table 6, CREME provides a rich set of features and well-labeled data that few studies can. CREME also covers more attack scenarios than most existing datasets.

7.2. Dataset efficiency

Fig. 8 shows the efficiency score of different datasets. Similarly, we use the ratio of 80% data for training, 20% data for testing, and 5-fold cross-validation. For example, the data amount for training in Kyoto 2006+ dataset, CICIDS dataset, Bot-IoT dataset, UNSW-NB15 dataset, Ton-IoT-Windows dataset, Ton-IoT-Network dataset, CREME-accounting dataset, CREME-Syslog dataset, and CREME-traffic dataset are 7615547, 4497544, 7575629, 4467745, 21104, 461043, 231576, 5381, and 297162 data points. For the dataset with a significant

Table 7
Top 14 ranked features of datasets.

Dataset	Top 14 ranked features
UNSW-NB15	sttl, ct_state_ttl, service_dns, state_CON, proto_tcp, ct_dst_sport_ltm, sloss, proto_arp, dbytes, sbytes, ct_srv_dst, proto_unas, Sintpkt, synack
Ton-IoT-Network	proto_tcp, dns_rejected_T, conn_state_REJ, src_pkts, dns_RA_T, dst_pkts, conn_state_OTH, conn_state_SF, dst_bytes, src_ip_bytes, dst_ip_bytes, src_bytes, conn_state_S3, duration
CICIDS	Bwd Packet Length Min, Total Fwd Packets, Average Packet Size, Bwd Packet Length Std, PSH Flag Count, Init_Win_bytes_backward, Bwd Header Length, Packet Length Std, Max Packet Length, Destination Port, Fwd Packet Length Max, Idle Max, Fwd Header Length, Total Backward Packets
Bot-IoT	N_IN_Conn_P_DstIP, state_number_5, state_number_2, seq, state_number_3, stddev, drate, state_number_4, state_number_1, srate, min, mean, state_number_9, N_IN_Conn_P_SrcIP
Kyoto 2006 +	Service, Flag_OTH, Dst_host_same_src_port_rate, Destination_bytes, Flag_S0, Dst_host_srv_count, Flag_RSTOS0, Dst_host_count, Source_bytes, Duration, Flag_SHR, Count, Flag_REJ, Same_srv_rat
Ours-Accounting	VSTEXT, DSK, TSLPI, POLI_-, EXC_1, CPUNR, RSIZE, ST_NE, ST_NS, MINFLT, MEM, VSIZE, PRI, EXC_0
Ours-Syslog	apache-access, postfix/submission/smtpd, vsftpd, postfix/qmgr, avahi-daemon, in.telnetd, login_FAILED, apache-access, login_pam_unix, login_FAILED, postfix/lmtp, postfix/qmgr, login_pam_securetty, proftpd
Ours-Traffic	Rate, State_INT, Dport, SrcPkts, Flgs_ e s, Sport, TotPkts, State_CON, Proto_tcp, DstRate, SrcRate, State_sSEfF, Proto_udp, DstPkts

Table 8
Examination results of datasets of each data source.

Algorithms	Accounting			Traffic			Syslog		
	Accuracy	F1-score	Execution time (s)	Accuracy	F1-score	Execution time (s)	Accuracy	F1-score	Execution time (s)
Decision Tree	0.9970	0.9967	0.443	0.9998	0.9998	0.590	0.9846	0.9838	0.0139
Naive Bayes	0.9395	0.9372	0.194	0.9774	0.9781	0.212	0.9833	0.9822	0.0106
Extra Tree	0.9970	0.9967	0.144	0.9987	0.9987	0.154	0.9859	0.9851	0.0116
KNN	0.9791	0.9767	37.893	0.9976	0.9977	167.873	0.9857	0.9849	0.1419
Random Forest	0.9969	0.9966	7.496	0.9998	0.9998	12.585	0.9864	0.9857	0.2475
XGBoost	0.9968	0.9965	5.380	0.9998	0.9998	5.521	0.9853	0.9845	0.1232

imbalanced problem, the data of rare classes are increased by using repetition. CREME's efficiency scores are 0.682 for accounting and 0.504 on average compared with the best performance at 0.654 and an average of 0.257 of other studies. The positive performance results show the significant contributions of the correlation of labeled data and the removal of redundant features in the data generation process. The evaluation results indicate that Kyoto 2006+ with an efficiency score (0.654) likely competing with ours. Unfortunately, Kyoto 2006+ only includes data from a single source (network traffic).

Fig. 9 shows the accuracy score of training the ML-based IDS (running XGBoost Classifier) with all and only important features per dataset. The results indicate that the accuracy of training a system with the important features is quite competitive with using all features. Even better, using full features can significantly cost the system in computation and training time if the data is large. This fact reveals one of the best advantages of identifying important features in a dataset. The ML-based IDS can save training time substantially while still maintaining the detection accuracy at best.

Fig. 10 shows the detection accuracy of selecting various combinations of features in different datasets. The system can achieve high accuracy (97% average) with only 14 features in all the datasets. Table 7 shows these top-ranked features in our testing. An important feature means it contributes more meaningful information for the learning training. Several features may have few contributions, e.g., a flow appears once or short time in the dataset. These features can be removed without a significant impact on the prediction performance. On the other hand, the cost to conduct those data can waste the detection system's resources. By contrast, a lack of enough data can contribute to an imbalance in the dataset, which can negatively impact a system's performance.

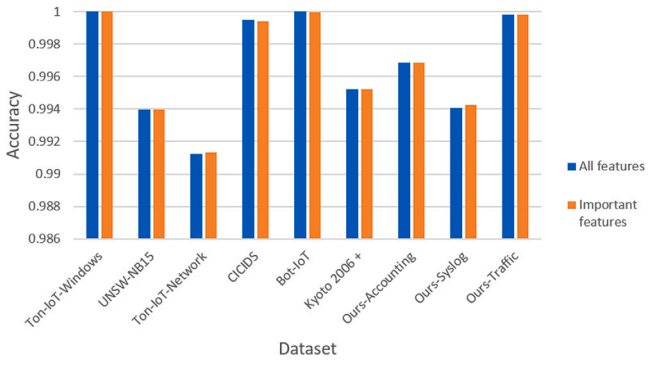

Fig. 9. Detection accuracy of ML-based IDS for two configurations: with important features vs with all features.

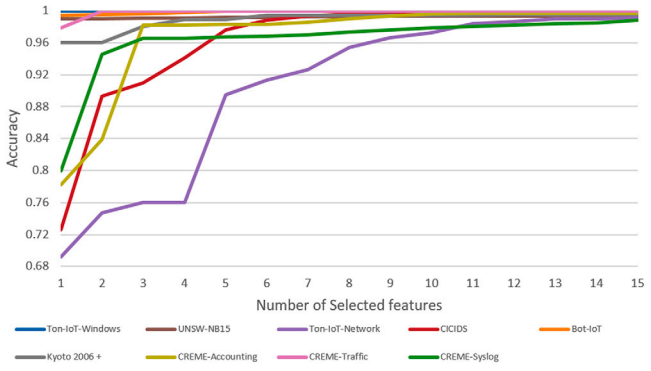

Fig. 10. Accuracy performance of the system with various combinations of selected features.

7.3. Detection accuracy performance

Table 8 shows the evaluation results of the CREME dataset after testing six common ML algorithms: Decision Tree, Naive Bayes, Extra Tree,

KNN, Random Forest, and XGBoost. We used two measurement metrics to compare six ML algorithms, accuracy, and F1 score. The results highlight the Decision Tree-based detection engine has the best performance with the measurement metrics and execution time. The Decision Tree-based IDS in particular gave an accuracy/F1 score/execution time of 0.9970/0.9967/0.443s for the accounting, 0.9998/0.9998/0.590s for the traffic, and 0.9846/0.9838/0.0139s for the syslog. By contrast, the KNN algorithm required the longest execution time for large-scale accounting and network traffic datasets, at 37.893s and 167.873s, respectively. The negative results were sourced from the high-computing mechanism of KNN, where calculating the distance to each data point is highly inaccurate.

8. Conclusions and future work

This work proposes a toolchain that can automatically perform the tasks: feature extraction, data labeling, and assessment of generated dataset quality from multiple sources. The sources vary, such as network traffic, system log, or reports of the monitoring software. Unlike the prior framework, besides automation, the toolchain supports dataset validation and customization. The experiment results show that the dataset collected through our framework can yield better performance for the ML-based IDS in terms of coverage and detection efficiency. The dataset outputs included precious information about the list of important features, the best configuration of data sources, and their order, which can contribute to future research in the field. The administrator can visually follow the progress of all the processing stages at the centralized portal.

Since our work is the first attempt in the field, there are several potential targets for further enhancement. First, to increase the diversity of attack behaviors for a specific environment; more security attacks and benign services can be added to the testing. A broad range of evaluating the systems on different network/host environments can provide a better overview of the dataset ranking in terms of effectiveness. Another potential topic is to support more IoT device types. Besides, despite the promising results of our first attempt at dealing with multiple data sources, the training and testing still require substantial time. A compression or novel approach to shorten the inter-learning time can also be a rewarding topic for further study. Finally, an end-to-end learning model with the help of an autoencoder to parse data from multiple sources or reinforcement learning with decision trees to conduct a comprehensive evaluation on the combined dataset deserves further studies.

CRediT authorship contribution statement

Huu-Khoi Bui: Methodology, Software, Data curation, Writing – original draft. **Ying-Dar Lin:** Conceptualization, Methodology, Writing – original draft, Supervision. **Ren-Hung Hwang:** Methodology, Writing – review & editing, Supervision. **Po-Ching Lin:** Review, Technical suggestions. **Van-Linh Nguyen:** Methodology, Software, Writing – review & editing. **Yuan-Cheng Lai:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the Ministry of Science and Technology (MOST) of Taiwan under Grant No 110-2811-E-194-501-MY2, MOST 109-2221-E-194-025-MY2, 108-2221-E-194-022-MY3, 108-2221-E-194-019-MY3, and in part by the Advanced Institute of Manufacturing with High-Tech Innovations (AIM-HI) through the Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) in Taiwan, and in part by the Ministry of Education and Training (MOET) of Vietnam and Thai Nguyen University under Grant No B2021-TNA-02.

References

Al-Hadhrami, Y., Hussain, F.K., 2020. Real time dataset generation framework for intrusion detection systems in IoT. Future Gener. Comput. Syst..

Al-Mohannadi, H., Mirza, Q., Namanya, A., Awan, I., Cullen, A., Disso, J., 2016. Cyber-attack modeling analysis techniques: An overview. In: 2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW). IEEE, pp. 69–76.

Anagnostopoulos, M., Spathoulas, G., Viaño, B., Augusto-Gonzalez, J., 2020. Tracing your smart-home devices conversations: A real world IoT traffic data-set. Sensors 20 (22), https://www.mdpi.com/1424-8220/20/22/6600.

Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J.A., Invernizzi, L., Kallitsis, M., et al., 2017. Understanding the mirai botnet. In: 26th {$USENIX$} Security Symposium ({$USENIX$} Security 17). pp. 1093–1110.

Beer, F., Hofer, T., Karimi, D., Bühler, U., 2017. A new attack composition for network security. In: 10. DFN-Forum Kommunikationstechnologien. Gesellschaft für Informatik eV.

Chandrashekar, G., Sahin, F., 2014. A survey on feature selection methods. Comput. Electr. Eng. 40 (1), 16–28.

Cinque, M., Della Corte, R., Pecchia, A., 2020. Contextual filtering and prioritization of computer application logs for security situational awareness. Future Gener. Comput. Syst. 111, 668–680, https://www.sciencedirect.com/science/article/pii/S0167739X19306454.

Gharib, A., Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A., 2016. An evaluation framework for intrusion detection dataset. In: 2016 International Conference on Information Science and Security (ICISS). IEEE, pp. 1–6.

Haider, W., Hu, J., Slay, J., Turnbull, B.P., Xie, Y., 2017. Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling. J. Netw. Comput. Appl. 87, 185–192.

Hassan, W.U., Noureddine, M.A., Datta, P., Bates, A., 2020. OmegaLog: High-fidelity attack investigation via transparent multi-layer log analysis. Netw. Distrib. Syst. Secur. (NDSS).

He, P., Zhu, J., Zheng, Z., Lyu, M.R., 2017. Drain: An online log parsing approach with fixed depth tree. In: 2017 IEEE International Conference on Web Services (ICWS). IEEE, pp. 33–40.

Hwang, R., Peng, M., Huang, C., Lin, P., Nguyen, V., 2020. An unsupervised deep learning model for early network traffic anomaly detection. IEEE Access 8, 30387–30399. http://dx.doi.org/10.1109/ACCESS.2020.2973023.

Kang, H., et al., 2019. IoT Network intrusion dataset. IEEE Dataport, https://doi.org/10.21227/q70p-q449 (Accessed: 30 May 2021).

Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J., 2019. Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity 2.

Kolias, C., Kambourakis, G., Stavrou, A., Voas, J., 2017. Ddos in the IoT: Mirai and other botnets. Computer 50 (7), 80–84.

Koroniotis, N., Moustafa, N., Sitnikova, E., Turnbull, B., 2019. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. Future Gener. Comput. Syst. 100, 779–796.

Laboratory, S., 2020. A Labeled Dataset with Malicious and Benign IoT Network Traffic, January 22th. Agustin Parmisano, Sebastian Garcia, Maria Jose Erquiaga. IEEE.

Moustafa, N., 2019. ToN_IoT datasets. IEEE Dataport https://doi.org/10.21227/fesz-dm97 (Accessed 30 May 2021).

Moustafa, N., Slay, J., 2015. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: 2015 Military Communications and Information Systems Conference (MilCIS). IEEE, pp. 1–6.

Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A., 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: ICISSP. pp. 108–116.

Shi, Z., Li, J., Wu, C., Li, J., 2019. DeepWindow: An efficient method for online network traffic anomaly detection. In: 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS). IEEE, pp. 2403–2408.

Singh, R., Kumar, H., Singla, R., 2015. A reference dataset for network traffic activity based intrusion detection system. Int. J. Comput. Commun. Control 10 (3), 390–402.

Song, J., Takakura, H., Okabe, Y., Eto, M., Inoue, D., Nakao, K., 2011. Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation. In: Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, pp. 29–36.

Turcotte, M.J., Kent, A.D., Hash, C., 2017. Unified host and network data set. arXiv preprint arXiv:1708.07518.

Wang, Z.-Y., 2021. Multi-datasource Machine Learning in Intrusion Detection: Packet Flows, System Logs and Host Statistics. National Yang Ming Chiao Tung University, Hsinchu, Taiwan, pp. 1–68.

Xing, J., Wu, C., 2020. Detecting anomalies in encrypted traffic via deep dictionary learning. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, pp. 734–739.

Zhang, L., Lv, Z., Zhang, X., Chen, C., Li, N., Li, Y., Wang, W., 2019. A novel approach for traffic anomaly detection in power distributed control system and substation system. In: International Conference on Network and System Security. Springer, pp. 408–417.

Zhu, J., He, S., Liu, J., He, P., Xie, Q., Zheng, Z., Lyu, M.R., 2019. Tools and benchmarks for automated log parsing. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP). IEEE, pp. 121–130.

Huu-Khoi Bui received the M.S. degree in electrical engineering and computer science from the National Chiao Tung University, Hsinchu, Taiwan, in 2021. He was an associate researcher at High Speed Network Lab, NCTU, in 2019–2021. His research interests include network security, machine learning, and application security.

Ying-Dar Lin is a Chair Professor of computer science at National Chiao Tung University (NCTU), Taiwan. He received his Ph.D. in computer science from the University of California at Los Angeles (UCLA) in 1993. He was a visiting scholar at Cisco Systems in San Jose during 2007–2008, CEO at Telecom Technology Center, Taiwan, during 2010–2011, and Vice President of National Applied Research Labs (NARLabs), Taiwan, during 2017–2018. He was the founder and director of Network Benchmarking Lab (NBL) in 2002–2018, which reviewed network products with real traffic and automated tools, and has been an approved test lab of the Open Networking Foundation (ONF). He also cofounded L7 Networks Inc. in 2002, later acquired by D-Link Corp, and O'Prueba Inc. a spin-off from NBL, in 2018. His research interests include network security, wireless communications, network softwarization, and machine learning for communications. His work on multi-hop cellular was the first along this line, and has been cited over 1000 times and standardized into IEEE 802.11s, IEEE 802.15.5, IEEE 802.16j, and 3GPP LTE-Advanced. He is an IEEE Fellow (class of 2013), IEEE Distinguished Lecturer (2014–2017), ONF Research Associate (2014–2018), and received in 2017 Research Excellence Award and K. T. Li Breakthrough Award. He has served or is serving on the editorial boards of several IEEE journals and magazines, including Editor-in-Chief of IEEE Communications Surveys and Tutorials (COMST, 1/2017–12/2020). He published a textbook, Computer Networks: An Open Source Approach, with Ren-Hung Hwang and Fred Baker (McGraw-Hill, 2011).

Ren-Hung Hwang received his Ph.D. degree in computer science from the University of Massachusetts, Amherst. He joined the Department of Computer Science and Information Engineering, National Chung Cheng in 1993, where he is now a distinguished professor and the chief information technology officer. He has published more than 250 international journal and conference papers. He served as the Dean of the College of Engineering during 2014~2017. He received the IEEE Best Paper Award from IEEE Ubi-Media 2018, IEEE SC2 2017, and IEEE IUCC 2014. His current research interests include Internet of Things, network security, cloud/edge/fog computing, and 5G V2X networks.

Po-Ching Lin received his Ph.D. degree in computer science from National Chiao Tung University, Hsinchu, Taiwan, in 2008. He joined the faculty of the Department of Computer Science and Information Engineering, CCU, in August 2009. He is currently an associate professor. His research interests include network security, network traffic analysis, and performance evaluation of network systems.

Van-Linh Nguyen received his Ph.D. degree in computer science and information engineering from National Chung Cheng University (CCU), Taiwan, in 2019. He joined the Department of Information Technology, Thai Nguyen University of Information and Communication Technology (TNU-ICTU) in 2012, where he is now an assistant professor. His research interests include cybersecurity, vehicular networks, autonomous driving, and edge computing.

Yaun-Cheng Lai received his Ph.D. degree in the Department of Computer and Information Science from National Chiao Tung University in 1997. He joined the faculty of the Department of Information Management at National Taiwan University of Science and Technology in August 2001 and has been a distinguished professor since June 2012. His research interests include performance analysis, software-defined networking, wireless networks, and IoT security.