ASNM Datasets: A Collection of Network Traffic Features for Testing of Adversarial Classifiers and Network Intrusion Detectors

Ivan Homoliak*
ihomoliak@fit.vutbr.cz

Petr Hanacek*
hanacek@fit.vutbr.cz

*Faculty of Information Technology, Brno University of Technology

***Abstract*—In this paper, we present three datasets that have been built from network traffic traces using ASNM features, designed in our previous work. The first dataset was built using a state-of-the-art dataset called CDX 2009, while the remaining two datasets were collected by us in 2015 and 2018, respectively. These two datasets contain several adversarial obfuscation techniques that were applied onto malicious as well as legitimate traffic samples during "the execution" of particular TCP network connections. Adversarial obfuscation techniques were used for evading machine learning-based network intrusion detection classifiers. Further, we showed that the performance of such classifiers can be improved when partially augmenting their training data by samples obtained from obfuscation techniques. In detail, we utilized tunneling obfuscation in HTTP(S) protocol and non-payload-based obfuscations modifying various properties of network traffic by, e.g., TCP segmentation, retransmissions, corrupting and reordering of packets, etc. To the best of our knowledge, this is the first collection of network traffic metadata that contains adversarial techniques and is intended for non-payload-based network intrusion detection and adversarial classification. Provided datasets enable testing of the evasion resistance of arbitrary classifier that is using ASNM features.**

I. Introduction

Network intrusion attacks such as exploiting unpatched services are one of the most dangerous threats in the domain of information security [1], [2]. Due to an increasing sophistication in the techniques used by attackers, misuse-based/knowledge-based [3] intrusion detection suffers from undetected attacks such as zero-day attacks or polymorphism, enabling an exploit-code to avoid positive signature matching of the packet payload data. Therefore, researchers and developers are motivated to design new methods to detect various versions of the modified network attacks including the zero-day ones. These goals motivate the popularity of Anomaly Detection Systems (ADS) and also the classification-based approaches in the context of intrusion detection. Anomaly-based approaches are based on building profiles of normal users, and they try to detect anomalies deviating from these profiles [3], which might lead to detection of unknown intrusions, but on the other hand it might also generate many false positives. In contrast, the classification-based approaches take advantage of both misuse-based and anomaly-based models in order to leverage their respective advantages. The classification-based detection methods first build a model based on the labeled samples from both classes – intrusions and the legitimate instances. Second, they compare a new input to the model and select the more similar class as the predicted label. Classification and anomaly-based approaches are capable to detect some unknown intrusions, but at the same time they may be susceptible to evasion by obfuscation techniques.

In this paper, we present ASNM datasets, a collection of malicious and benign network traffic data. ASNM datasets include records consisting of several features that express miscellaneous properties and characteristics of TCP communications (i.e., aggregated bidirectional flows). These features are called Advanced Security Network Metrics (ASNM) and were designed in our previous work [4] with the intention to distinguish between legitimate and malicious TCP connections (i.e., intrusions and C&C channels of malware). ASNM features are extracted from tcpdump [5] traces and do not perform deep packet inspection during their computation, which makes them suitable for passive monitoring of (potentially encrypted) network traffic.

To this end, we performed ASNM feature extraction over three different subsets of network traffic collections, resulting in three sub-datasets that we provide to the community:

- **ASNM-CDX-2009 Dataset**: was created from tcpdump traces of CDX 2009 dataset [6]. The dataset misses a few newer ASNM features and does not contain any obfuscations of the network traffic (see details in Section IV-A).
- **ASNM-TUN Dataset:** was created with the intention to evade and improve machine learning classifiers, and besides legitimate network traffic samples, it contains tunneling obfuscation technique [7] applied onto malicious network traffic, in which several vulnerable network services were exploited (see details in Section IV-B).
- **ASNM-NPBO Dataset:** like the previous dataset, the current dataset was created with the intention to evade and improve machine learning classifiers, and it contains non-payload-based obfuscation techniques (modifying the properties of network flows) applied onto malicious traffic and onto several samples of legitimate traffic (see details in Section IV-C).

All ASNM datasets are available for download at http://www.fit.vutbr.cz/∼ihomoliak/asnm/. In the following, we will describe ASNM features, detail particular datasets, and finally benchmark several supervised classification methods used for non-payload-based[1] network intrusion detection in ASNM datasets. We conduct a few experiments aimed at the adversarial classification, and we demonstrate that proposed obfuscations are able to evade an intrusion detection of employed classifiers. Consequently, we show that after partially augmenting the training data by obfuscated attacks, we can significantly improve the performance of the classifiers.

The rest of the paper is organized as follows. In Section II, we define the classification problem in intrusion detection and describe preliminaries and terms used throughout the paper. In Section III, we formally define ASNM features and describe them. Next, we introduce particular ASNM datasets in Section IV and consequently perform their benchmarking in Section V. In Section VII we discuss limitations of the proposed datasets. Then, in Section VI, we mention existing network datasets and network features, compare them to the ASNM datasets, and finally in Section VIII we conclude the paper.

II. Problem Definition and Preliminaries

First, we define the scope of our work by introducing the network connection as an elementary data object that is used for building our datasets. Second, we describe the feature extraction process over a network connection object, which forms a sample/data record in our datasets. Then, we describe the intrusion detection classification task, representing the problem that is addressed by an arbitrary binary classifier given a dataset containing 2-class labels. This problem represents the main challenge of ASNM datasets, but the application of ASNM datasets can be straightforwardly extended to a multi-class classification problem in sub-datasets containing multi-class labels.

A. TCP Connection

Consider a session of a protocol at the application layer of the TCP/IP stack that serves for data transfer between the client/server based application. The interpretation of application data exchanges between client and server can be formulated, considering the TCP/IP stack up to the transport layer, by connection c that is constrained to the connection-oriented protocol TCP at L4, Internet protocol IP at L3, and Ethernet protocol at L2. The TCP connection c is represented by the tuple

$$c = (t_s, t_e, p_c, p_s, ip_c, ip_s, P_c, P_s),$$

which consists of the start and end timestamps t_s and t_e, ports of the client and the server p_c and p_s, IP addresses of the client and the server ip_c and ip_s, sets of packets sent by the client P_c, and by the server P_s, respectively (see details in Table XXIII of Appendix). Sets P_c and P_s contain a number of packets, where each of them can be interpreted by the packet tuple

$$p = (t, size, eth_{src}, eth_{dst}, ip_{off}, ip_{ttl}, ip_p, ip_{sum},$$
$$ip_{src}, ip_{dst}, ip_{dscp}, tcp_{sport}, tcp_{dport}, tcp_{sum},$$
$$tcp_{seq}, tcp_{ack}, tcp_{off}, tcp_{flags}, tcp_{win}, tcp_{urp}, data).$$

The symbols of the tuple are described in Table XXIV of Appendix. We assume that the payload of P_s and P_c is encrypted, and thus data of these packet sets are not accessible.

Each TCP connection has its beginning that is represented by a three way handshake, in which, three packets that contain the same IP addresses (ip_s, ip_d), ports (p_s, p_d), and sequence/acknowledgment numbers (tcp_{seq}, tcp_{ack}) conforming the specification of RFC 793[2] must be found. Similarly, each TCP connection has its end, which is defined by a three-way-endshake or by an inactivity timeout.[3]

B. Feature Extraction

At this time, we can express characteristics of a TCP connection by network connection features. The features extraction process is defined as a function that maps a connection c into space of features F:

$$\begin{gathered} f(c) \mapsto F, \\ F = (F_1,\, F_2, \ldots, F_n), \end{gathered} \tag{1}$$

where n represents the number of defined features. Each function f_i that extracts feature i is defined as a mapping of a connection c into feature space F_i:

$$f_i(c) \mapsto F_i, \quad i \in \{1, \ldots, n\}, \tag{2}$$

and each element[4] of codomain F_i is defined as

$$\begin{gathered} e = (e_0, \ldots, e_n),\ n \in \mathbb{N}_0, \\ e_i \in \mathbb{N} \ \mid\ e_i \in \mathbb{R} \ \mid e_i \in \Gamma^+,\ i \in \{0, \ldots, n\}, \\ \Gamma = \{a - z, A - Z, 0 - 9\}, \end{gathered} \tag{3}$$

where Γ^+ denotes positive iteration of the set Γ. Note that for demonstration purposes, we abstract in our formalization from the fact that some features of a network connection c can be extracted not only from c itself but in addition from metadata of c that are not part of c. For example, such metadata may represent "neighboring" network connections of c, which we later refer to as *a context* of c (see Section III).

In general, network connection features can be instantiated, for example, by discriminators of A. Moore [8], Kyoto 2006+ features [9], basic and traffic features[5] of KDD Cup'99 dataset [10], NetFlow features [11], or ASNM features [4], CICFlowMeter features [12], multi-layered network traffic features from BGU [13], or connection-less features [14].

[1] Not performing deep packet inspection.

[2] URL http://www.ietf.org/rfc/rfc793.txt, page 30.

[3] E.g., in Unix-based systems, such a timeout is equal to five days.

[4] Representing a particular dimension of a feature.

[5] Not content features, which work over payload of the network data.

C. Intrusion Detection Classification Task

A data sample of the dataset D_{tr} refers to the vector of the network connection features, defined in Section II-B. Then, referring to [15], let $X = V \times Y$ be the space of labeled samples, where V represents the space of unlabeled samples and Y represents the space of possible labels. Let $D_{tr} = \{x_1, x_2, \ldots, x_n\}$ be a training dataset consisting of n labeled samples, where

$$x_i = (v_i \in V,\ y_i \in Y). \tag{4}$$

Consider classifier C which maps unlabeled sample $v \in V$ to a label $y \in Y$:

$$y = C(v), \tag{5}$$

and learning algorithm A which maps the given dataset D to a classifier C:

$$C = A(D_{tr}). \tag{6}$$

The notation $y_{predict} = A(D_{tr}, v)$ denotes the label assigned to an unlabeled sample v by the classifier C, build by learning algorithm A on the dataset D_{tr}. Then, all extracted features $f()$ of an unknown connection c can be used as an input of the trained classifier C that predicts the target label:

$$y_{predict} = A\big(D_{tr}, f(c)\big), \tag{7}$$

where

$$y_{predict} \in \{Intrusion, Legitimate\}. \tag{8}$$

D. Adversarial Obfuscations & Evasion of the Classifier

Assume a connection c_m representing a malicious communication executed without any obfuscation. Then, c_m can be expressed by network connection features

$$f(c_m) \mapsto F^m = (F_1^m, \quad F_2^m, \ldots, F_n^m) \tag{9}$$

that are delivered to the previously trained classifier C. Assume that C can correctly predict the target label as a malicious one, because its knowledge base is derived from training dataset D_{tr} containing features of malicious connections having similar (or the same) behavioral characteristics as c_m.

Now, consider connection c'_m that represents the malicious communication c_m executed by employment of an obfuscation technique that is aimed at modification of network behavioral properties of the connection c_m. An obfuscation technique can modify P_c and P_s packet sets of the original connection c_m as well as IP addresses (ip_s, ip_d) and ports (p_s, p_d) of the original connection c_m.

Hence, network connection features extracted for c'_m are represented by

$$f(c'_m) \mapsto F^{m'} = (F_1^{m'}, \quad F_2^{m'}, \ldots, F_n^{m'}) \tag{10}$$

and have different values than features F^m of the connection c_m. Therefore, we conjecture that the likelihood of a correct prediction of of c'_m-connection's features $F^{m'}$ by the previously assumed classifier C is lower than in the case of connection c_m, which might cause an evasion of the detection. Also, we conjecture that the classifier C' trained by learning algorithm A on training dataset D'_{tr}, containing obfuscated malicious instances, will be able to correctly predict higher number of unknown obfuscated malicious connections than classifier C. We will demonstrate the correctness of these assumptions in Section V on two of our datasets.

III. ASNM Features and Context Analysis

ASNM features [4] are network connection features that describe various properties of TCP connections and were designed with the intention to distinguish between legitimate traffic and remote buffer overflow attacks.[6] We studied behavioral characteristics of remote buffer overflow attacks in our previous work [17], and our findings inspired the design of ASNM features. We can interpret ASNM features like an extended protocol NetFlow [11] but describing more than statistical properties of network connections. In addition to NetFlow features, ASNM features represent dynamical, localization, and, most importantly, the behavioral properties of network connections. Moreover, some of the features utilize a context of an analyzed connection c, which represents "neighboring" connection objects (see Section III-A).

In the following, we assume an input dataset of network traffic traces, which is used for identification of network TCP connection objects $C = \{c_1, \ldots, c_M\}$, where M is a count of TCP connections in the dataset.

A. Context Definition

We assume a dataset of TCP connection objects (as described in Section II) Considering analyzed TCP connection c_k, we define a sliding window sw of length τ as a set of TCP connections W_k that are delimited by $\pm\frac{\tau}{2}$:

$$\begin{aligned} sw(c_k, \tau) &= W_k \\ W_k &\subseteq C, \\ W_k &= \{c_j\}, \end{aligned} \tag{11}$$

where each TCP connection c_j must satisfy the following:

$$\begin{aligned} c_j[t_s] &> c_k[t_s] - \frac{\tau}{2}, \\ c_j[t_e] &< c_k[t_s] + \frac{\tau}{2}. \end{aligned} \tag{12}$$

The next fact about each particular TCP connection c_k is an unambiguous association of it to particular sliding window W_k. We can interpret the start time t_s of the TCP connection c_k as a center of the sliding window W_k. Then, we can denote a shift of the sliding window $\Delta(W_j)$ which is defined by start time differences of two consecutive TCP connections in C:

$$\begin{aligned} \Delta(W_j) &= c_{j+1}[t_s] - c_j[t_s], \\ j &\in \{1, \ldots, |C| - 1\}. \end{aligned} \tag{13}$$

[6] See Appendix D of [16] for the full list of ASNM features.

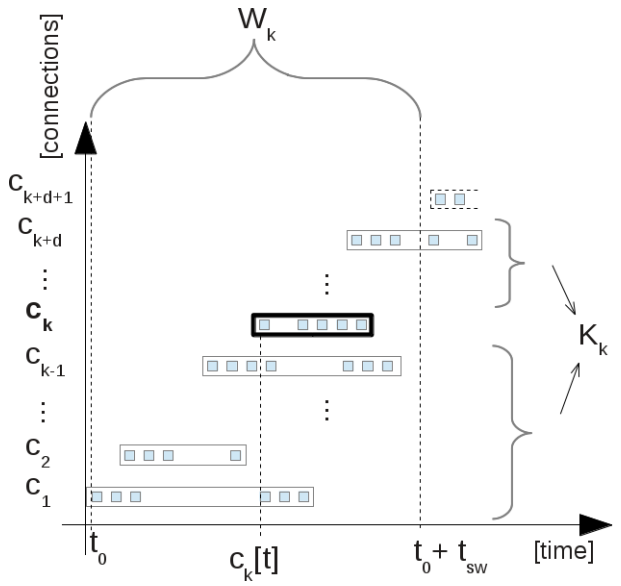

Figure 1: Sliding window and the context of the connection c_k [4].

Next, we define the context K_k of the TCP connection c_k, which is a set of all connections in a particular sliding window W_k excluding analyzed TCP connection c_k:

$$K_k = \{c_1, \ldots, c_n\} = \{W_k \setminus c_k\}. \tag{14}$$

Defined terms are shown in Figure 1. In the figure, the x axis displays time, and the y axis represents TCP connections, which are shown in the order of their occurrences. Packets are represented by small squares, and TCP connections are represented by a rectangular boundary of particular packets. A bold line and bold font are used for depicting an analyzed TCP connection c_k, which has an associated sliding window W_k and context K_k. TCP connections, which are part of the sliding window W_k, are drawn by full line boundary, and TCP connections, which are not part of this sliding window, are drawn by a dashed line boundary. We note that only a few features from the ASNM datasets utilize context; these features belong to dynamic and behavioral categories (see Section III-C).

B. ASNM Feature Extraction

In addition to a general definition of network connection feature extraction (see Section II-B), we incorporate the context of a TCP connection into the extraction process of ASNM features. The ASNM feature extraction is thus defined as a function that maps a connection c_k with its context $K_k = sw(c_k, \tau)$ into feature space F:

$$\begin{aligned} f(c_k, K_k) &\mapsto F, \\ F &= (F_1, F_2, \ldots, F_n), \end{aligned} \tag{15}$$

where n represents the number of defined features, while the rest of the definition is inherited from Section II-B.

Category of ASNM Features	#
Statistical	77
Dynamic	32
Localization	8
Distributed	34
Behavioral	43

Table I: Categorization of ASNM features.

C. Categorization of ASNM Features

The list of original proposed ASNM feature set contains 167 features and is present Master's thesis [18] and formally described in [4]. However, the ASNM feature set was later extended [16], resulting in 194 features. These 194 features are in many cases a result of reasonable parametrization of the base feature functions $f_i()$, We depict a categorization of our feature set in Table I together with their counts. We decided to determine the naming of particular categories of features according to their principles, not according to their data representation. In the following, we briefly describe each category.

Statistical Features. In this category of ASNM features, the statistical properties of TCP connections are identified. All packets of the TCP connection are considered in order to determine count, mode, median, mean, standard deviation, ratios of some header fields of packets, or the packets themselves. This category of features partially uses a time representation of packets occurrences, in contrast to the dynamic category (see below). Therefore, it includes particularly dynamic properties of the analyzed TCP connection, but without any context. Most of the features in this category also distinguish inbound and outbound packets of the analyzed TCP connection.

Dynamic Features. Dynamic features were defined with the aim to examine dynamic properties of the analyzed TCP connection and transfer channel such as a speed or an error rate. These properties can be real or simulated. Eighteen of the features consider the context of an analyzed TCP connection. The difference between some of the statistical and dynamic features from a dynamic view can be demonstrated on two instances of the same TCP connection, which performs the same packet transfers, but in different context conditions and with different packet retransmission and attempts to start or finish the TCP connection. Many of the defined features distinguish between the inbound and outbound direction of the packets and consider the statistical properties of the packets and their sizes, as mentioned in statistical features.

Localization Features. The main characteristic of the localization features category is that it contains static properties of the TCP connection. These properties represent the localization of participating machines and their ports used for communication. In some features, the localization is expressed indirectly by a flag, which distinguishes whether participating machines lay in a local network or not. Features included in this category do not consider the context of the analyzed TCP connection, but they distinguish a direction of the analyzed TCP connection.

Distributed Features. The characteristic property of the distributed features category is the fact that they distribute packets or their lengths to a fixed number of intervals per the unit time specified by a logarithmic scale (1s, 4s, 8s, 32s, 64s). A logarithmic scale of fixed time intervals was proposed as a performance optimization during the extraction of the features. The next principal property of this category is vector representation. All these features are supposed to work within the context of an analyzed TCP connection.

Behavioral Features. Behavioral features represent properties associated with the behavior of an analyzed TCP connection. Examples include legal or illegal connection closing, the polynomial approximation of packet lengths in a time domain or an index of occurrence domain, count of new TCP connections after starting an analyzed TCP connection, coefficients of Fourier series with the distinguished direction of an analyzed TCP connection, etc.

IV. ASNM Datasets

In this section, we detail three different datasets that have been built using ASNM features. The first of them was built using an existing dataset of network traffic traces, while the remaining two were collected by us, and they contain several adversarial obfuscation techniques that were applied onto malicious as well as legitimate samples during "the execution" of particular network connections.

A. ASNM-CDX-2009 Dataset

ASNM-CDX-2009 dataset was build from CDX-2009 dataset [19], which was introduced by Sangster et al. [6] and it contains data in tcpdump format as well as SNORT [20] intrusion prevention logs, as relevant sources for our purpose.

The CDX 2009 dataset was created during the network warfare competition, in which one of the goals was to generate a labeled dataset. By labeled dataset, the authors mean tcpdump traces of all simulated communications and SNORT log with information about occurrences of intrusions, deemed as the expert knowledge. Network infrastructure contained four servers with four vulnerable services (one per each server), while the authors provided two collections of network traces: 1) network traces captured outside the West Point network border and 2) network traces captured by National Security Agency (NSA). The services that run on the hosted servers together with IP addresses of the servers are listed in Table II. Two types of IP addresses are shown in this table:

Service	OS	Internal IP	External IP
Postfix Email	FreeBSD	7.204.241.161	10.1.60.25
Apache Web Server	Fedora 10	154.241.88.201	10.1.60.187
OpenFire Chat	FreeBSD	180.242.137.181	10.1.60.73
BIND DNS	FreeBSD	65.190.233.37	10.1.60.5

Table II: A list of vulnerable servers in CDX 2009 dataset.

- **Internal IP** addresses – corresponding to the SNORT log,
- **External IP** addresses – corresponding to a TCP dump network captured outside the West Point network border.

Network Service	Count of TCP Connections		
	Legitimate	Malicious	Summary
Apache	2911	37	2948
Postfix	179	7	186
Other Traffic	2637	–	2637
Summary	5727	44	5771

Table III: ASNM-CDX-2009 dataset distribution.

Note that specific versions of services described in [6] were not announced. We found out that SNORT log can be associated only with data capture outside of West point network border and only with significant timestamps differences – approximately 930 days. We have not found any association between SNORT log and data capture performed by NSA. We focused only on buffer overflow attacks found in SNORT log, and we performed a match with the packets contained in the West point network border capture.

Despite all the efforts, we matched only 44 buffer overflow attacks out of 65 entries in SNORT log. To correctly match SNORT entries, it was necessary to remap external IP addresses to internal ones, because SNORT detection was performed in external network and TCP dump data capture contains entries with internal IP addresses. We found out that in CDX 2009 dataset, buffer overflow attacks were performed only on two services – Postfix Email and Apache Web Server.

The buffer overflow attacks that were matched with data capture have their content only in two TCP dump files:

- *2009-04-21-07-47-35.dmp*
- *2009-04-21-07-47-35.dmp2*

Due to the high count of all packets (approx. 4 mil.) in all dump files, we decided to consider only these two files for the purpose of extraction both malicious and legitimate samples (which together contain $1,538,182$ packets). We also noticed that network data density was increased in the time when attacks were performed. Consequently, we made another reduction of all packets consider so far, which filtered enough temporal neighborhood of all attacks occurrences, and at the same time, included a high enough number of legitimate TCP connections. In the result, we used 204 953 packets for the extraction of ASNM features. A distribution of malicious and legitimate samples across obtained dataset is presented in Table III. Beside two services that contained buffer overflow vulnerabilities, our dataset also contains samples representing other network traffic, which we consider as legitimate since no match of its metadata with SNORT log was determined.

Labeling. ASNM-CDX-2009 dataset contains two types of labels that are enumerated by increasing order of their granularity in the following:

- **label_2**: is a two-class label, which indicates whether an actual sample represents a network buffer overflow attack or legitimate traffic.

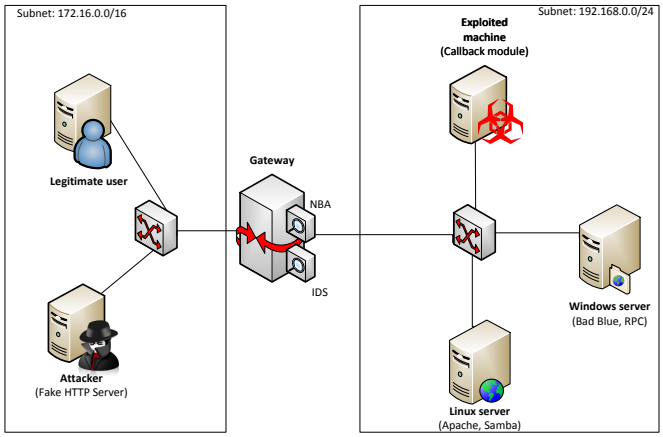

Figure 2: A setup of virtual network used in ASNM-TUN dataset.

Service	CVE	CVSS
Apache Tomcat	2002-0082	7.5
BadBlue	2007-6377	7.5
DCOM RPC	2003-0352	7.5
Samba	2003-0201	10.0

Table IV: A list of vulnerable services in ASNM-TUN dataset.

- **label_poly**: is composed of two parts that are delimited by a separator: (a) a two-class label where legitimate and malicious communications are represented by symbols 0 and 1, respectively, and (b) an acronym of network service. This label represents the type of communication on a particular network service.

This dataset was for the first time used and evaluated in [4].

B. ASNM-TUN Dataset

ASNM-TUN[7] dataset was build in laboratory conditions[8] using a custom virtual network architecture (see Figure 2), where we simulated malicious TCP connections on a few selected vulnerable network services. The selected vulnerabilities are presented in Table IV, which also contains Common Vulnerabilities and Exposures (CVE) IDs and Common Vulnerability Scoring System (CVSS) values. A selection of the vulnerable services was aimed at a high severity of their successful exploitation, namely a presence of buffer overflow vulnerabilities that led to a remote shell code execution through an established backdoor communication, while as a consequence of successful exploitation, the attacker was able to get the root access. The details about each vulnerability and its exploitation are briefly described in the following listing:

- **Apache web server with `mod_ssl` plugin 2.8.6:** This attack exploits a buffer overflow vulnerability in `mod_ssl` plugin of the Apache web server. The plugin does not correctly initialize memory in the `i2d_SSL_SESSION` function, which allows a remote attacker to exploit a buffer overflow vulnerability in order to execute arbitrary code via a large client certificate that is signed by a trusted Certification Authority, which produces a large serialized session [21]. This allows remote code execution and modification of any file on a compromised system [22]. The vulnerable versions of the plugin are in range 2.7.1-2.8.6.
- **BadBlue web server 2.72b:** The second attack exploits a stack-based buffer overflow vulnerability in `PassThru` functionality of `ext.dll` in BadBlue 2.72b and earlier [23]. In the attack performing phase, the specially crafted packet with a long header is sent, which leads to an overflow of processing buffer [24]
- **Microsoft DCOM RPC:** The third attack exploits a vulnerability in Microsoft Windows DCOM Remote Procedure Call (DCOM RPC) service of Microsoft Windows NT 4.0, 2000 (up to Service Pack 4), Server 2003, and XP [25]. This vulnerability allows a remote attacker to execute an arbitrary code after a buffer overflow in the DCOM interface. The vulnerability was originally found by the Last Stage of Delirium research group and has been widely exploited since then [26]. The vulnerability is well documented, and it was used, for example, by Blaster worm.
- **Samba service 2.2.7:** The last attack exploits a buffer overflow vulnerability in `call_ trans2open` function in `trans2.c` of Samba 2.2.x before 2.2.8a, 2.0.10, earlier versions than 2.0.x and Samba-TNG before 0.3.2 [27]. This vulnerability allows a remote attacker to execute arbitrary code. An exploit code sends malformed packets to a remote server in batches [28]. Packets differ in one shell-code address only because the return address depends on versions of Samba and host operating systems.

Adversarial Modifications. We employed tunneling of malicious network traffic inside of HTTP and HTTPS protocols, serving as obfuscation techniques when exploiting vulnerable services. The tunneling obfuscation modifies P_c and P_s packet sets (see Section II-A) of the original malicious connection c_m by wrapping each original packet into a new one. Assuming the background from Section II-D, the tunneling (i.e., wrapping) may cause fragmentation of IP packets, and thus it can also modify the number of packets in both packet sets P_c and P_s. Also, the obfuscation modifies IP addresses (ip_c, ip_s) and ports (p_c, p_s) of the original connection. Symbols of the packet tuple whose values are sensitive to the obfuscation include all defined fields, as tunneling obfuscation creates new TCP/IP stack with unique values of L2, L3, L4 headers as well as new content of application layer data. All these modifications, especially modifications of P_c and P_s of the connection c_m, cause alteration of the original network connection features' values (see Section II-D).

For the purpose of simulating real network conditions, we executed each malicious and legitimate network communication four times with four different network traffic modifica-

[7]The name is derived from **TUN**elling obfuscations.

[8]Note that part of the legitimate connections was extracted from anonymized metadata collected from a real network.

Network Service	Count of TCP Connections			
	Legitimate	Direct Attacks	Obfuscated Attacks	Summary
Apache	38	102	61	201
BadBlue	95	4	10	109
DCOM RPC	4	4	8	16
Samba	15	20	8	43
Other Traffic	25	–	–	25
Summary	177	130	87	394

Table V: ASNM-TUN dataset distribution.

tions. Network traffic modifications differ in the alteration degree of the network traffic, and we divide them into four categories:

(a) **No Modification:** The first category represents reference output without any modification. All experiments ran on the same host machine to minimize deviations among different tests.
(b) **Traffic Shaping:** The second category is dedicated to simulation of traffic shaping. Therefore, all packets were forwarded with higher time delays. For this purpose, the special gateway machine with a limited processor's performance was used. This machine was also fully loaded to emulate slower packets processing than in the first scenario.
(c) **Traffic Policing:** The third category is supposed to simulate traffic policing when some of the packets were dropped during the processing on the network gateway node. In this case, a custom packet dropper was used on the gateway node, and 25% of packets were dropped, resulting in output which contained re-transmitted packets.
(d) **Corrupted Traffic:** The fourth category represents transmission on an unreliable network channel; thus, 25% of packets were corrupted during processing on the network gateway node.

Legitimate Network Traffic. Legitimate samples of the dataset were collected from two sources. The first source represents a legitimate traffic simulation in our virtual network architecture and also employed network traffic modifications for the purpose of simulating real network conditions. As the second source, common usage of selected services was captured in the campus network in accordance with policies in force. In the obtained data, no content of packets was captured, and all collected metadata was anonymized. Further, we filter out data matched on high severity alerts by signature-based Network Intrusion Detection Systems (NIDS) Suricata [29] and SNORT [20] through Virus Total API. This step assured that legitimate traffic does not contain any malicious data. Note that SNORT was equipped with *Sourcefire VRT* ruleset, and SURICATA utilized *Emerging Threats ETPro* ruleset. The final composition of the dataset after extraction of ASNM features is depicted in Table V.

Labeling. ASNM-TUN dataset contains four types of labels that are enumerated by increasing order of their granularity in the following:

- **label_2:** is a two-class label, which indicates whether an actual sample represents a network buffer overflow attack or a legitimate communication.
- **label_3:** is a three-class label, which distinguishes among legitimate traffic (symbol 3), direct attacks (symbols 1), and obfuscated network attacks (symbol 2).
- **label_poly:** is a label that is composed of 2 parts: (a) a three-class label, and (b) acronym of a network service. This label represents a type of communication on a particular network service.
- **label_poly_s:** is composed of 3 parts: (a) a three-class label, (b) an acronym of network service, and (c) a network modification technique employed. This label has almost the same interpretation as the previous one, but in addition, it introduces a network modification technique employed (identified by a letter from the previous listing).

Testing with Signature-Based NIDS. To investigate the effect of the tunneling obfuscation on signature-based NIDSs, we performed detection by SNORT and SURICATA through VirusTotal API [30]. SNORT was equipped with Sourcefire VRT ruleset, and SURICATA utilized Emerging Threats ET-Pro ruleset. The results of direct attacks' detection by both NIDSs are shown in Table VI. Note that high priority rules detected 93 direct attacks on Apache service in both NIDSs, but 4 undetected direct attacks occurred almost at the same time as some of the detected attack instances, and hence, we consider them as a part of other detected direct attacks. Also, we can see that five instances of direct attacks were not detected by SNORT nor SURICATA. These five instances utilized network traffic modifications (c) and (d), which likely influenced the detection rate of both NIDSs; hence, they give an intuition for the adversarial obfuscation techniques utilized in the last ASNM dataset (see Section IV-C). The resulting detection rates of direct attacks look the same in both NIDSs, but there were differences in fired alerts during the exploitation of Apache service. Unlike SNORT, SURICATA had not detected any occurrence of buffer overflow, nor shellcode, nor remote command execution but instead fired high priority alerts related to potential corporate privacy violation:

- `ET POLICY Possible SSLv2 Negotiation in Progress Client Master Key SSL2_RC4_128_WITH_MD5,`

which we decided to consider as correct detection. If we would not consider them as correctly detected, then SURICATA was not detecting any direct attack on the Apache service.

Next, we have performed exploitation of each vulnerable service using the tunneling obfuscation, while scanning the network by aforementioned NIDSs. The obtained results are depicted in Table VII, which distinguishes between tunneling obfuscation performed through HTTP and HTTPS protocols.

We can see that an average detection rate per service is significantly lower for obfuscated attacks than in the case of direct attacks, and thus tunneling obfuscation was partially

	Direct Attacks		
	Detected	**Total**	**%**
Apache	93 +4	102	95.10%
BadBlue	4	4	100.00%
DCOM RPC	4	4	100.00%
Samba	20	20	100.00%
Overall Detection	125	130	96.15%
ADR* per Service			98.77%

*Average detection rate.

(a) SNORT

	Direct Attacks		
	Detected	**Total**	**%**
Apache	93 +4	102	95.10%
BadBlue	4	4	100.00%
DCOM RPC	4	4	100.00%
Samba	20	20	100.00%
Overall Detection	125	130	96.15%
ADR* per Service			98.77%

*Average detection rate.

(b) SURICATA

Table VI: Detection of direct attacks in ASNM-TUN dataset by SNORT and SURICATA.

capable of evading detection by utilized NIDSs. Regarding tunneling through the HTTP protocol, both SNORT and SURICATA achieved the same low detection rate for all classes of attacks.

The situation is slightly different for the case of tunneling through the HTTPS protocol. The SNORT achieved an average detection rate (ADR) per class equal to 68.75% and SURICATA only 23.25%. We found out the same fact about high priority rules fired by SURICATA on exploitation of Apache service as in the case of direct attacks detection – neither buffer overflow, nor shellcode, nor remote command execution rules were matched, and thus we decided to accept the previously mentioned potential corporate privacy violation alert as correct detection again. If we would not accept it, then SURICATA were not detected any tunneled attack on Apache service. Also note that SURICATA fired one non-high-priority alert classified as potentially bad traffic in several instances of attacks tunneled through HTTPS, which exploited BadBlue, DCOM and Samba services:

- `ET POLICY FREAK Weak Export Suite From Client (CVE-2015-0204).`

But we have not considered it as correct detection due to the low priority of the alert as well as the scope of corresponding CVE-2015-0204 is only related to the client code of OpenSSL. The plus notation in Table VII, alike in Table VI, denotes undetected attacks that occurred almost at the same time as some other correctly detected attacks, and thus are considered as their parts. Concluding the results of NIDSs detection, we can state that the proposed tunneling obfuscation technique was successful in evading the NIDSs used since a high number of obfuscated attacks were not detected in comparison to the case where obfuscations were not employed. On the other hand, we emphasize that SNORT has detected the most of direct attacks on Apache service even though it was encrypted. This indicates that VirusTotal may utilize a very paranoid rule set, which causes false positives. Hence, the results of the analysis through VirusTotal API are arguable.

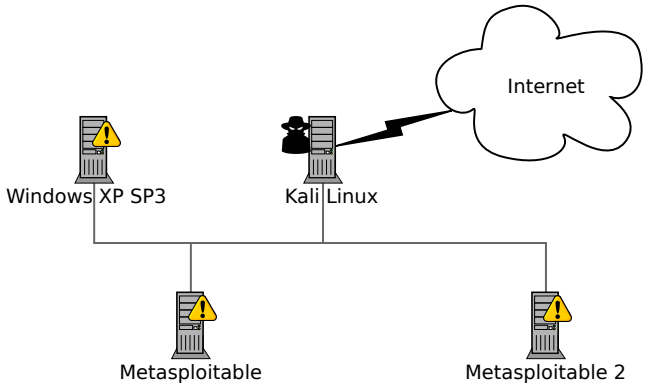

Figure 3: A setup of virtual network used in ASNM-NPBO dataset.

C. ASNM-NPBO Dataset

ASNM-NPBO[9] dataset was built in laboratory conditions using a virtual network architecture (see Figure 3) consisting of three vulnerable machines and the attacker's machine. All virtual machines were configured with private static IP addresses in order to enable easy automation of the whole exploitation process. Our testing network infrastructure consisted of the attacker's machine equipped with Kali Linux and vulnerable machines that were running Metasploitable 1, 2 [31], and Windows XP with SP 3. We aimed at the selection of vulnerable services with the high severity of their successful exploitation leading to remote shell code execution through an established backdoor communication. All selected vulnerable services are depicted in Table VIII, which also contains CVE IDs and CVSS severity score values. The details about each vulnerability and its exploitation are briefly described in the following:

- **Apache Tomcat 5.5:** First, a dictionary attack was executed in order to obtain access credentials into the application manager instance [32]. Further, the server's application manager was exploited for the transmission and execution of malicious code [33].
- **Microsoft SQL Server 2005:** A dictionary attack was employed to obtain access credentials of MSSQL user [34] and then the procedure `xp_cmdshell` enabling the execution of an arbitrary code was exploited [35].
- **Samba 3.0.20-Debian:** A vulnerability in Samba service enabled the attacker of arbitrary command execution, which exploited MS-RPC functionality when `username_map_script` [36] was allowed in the configuration. There was no need for authentication in this attack.

[9]The name is derived from **N**on-**P**ayload-**B**ased **O**bfuscations.

Service	Obfuscated Attacks								
	HTTP			HTTPS			All		
	Detected	Total	%	Detected	Total	%	Detected	Total	%
Apache	0	4	0.00	51 +6	57	100.00	57	61	93.40
BadBlue	3	6	50.00	2	4	50.00	5	10	50.00
DCOM	0	4	0.00	3	4	75.00	3	8	37.50
Samba	0	4	0.00	2	4	50.00	2	8	25.00
Summary	3	18	16.67	64	69	92.75	67	87	77.01
ADR*			12.50			68.75			51.49

*Average detection rate per class.

(a) SNORT

Service	Obfuscated Attacks								
	HTTP			HTTPS			All		
	Detected	Total	%	Detected	Total	%	Detected	Total	%
Apache	0	4	0.00	50 +3	57	92.98	53	61	86.89
BadBlue	3	6	50.00	0	4	0.00	3	10	30.00
DCOM	0	4	0.00	0	4	0.00	0	8	0.00
Samba	0	4	0.00	0	4	0.00	0	8	0.00
Summary	3	18	16.67	53	69	76.81	56	87	64.37
ADR*			12.50			23.25			29.22

*Average detection rate per class.

(b) SURICATA

Table VII: Detection of obfuscated attacks in ASNM-TUN dataset by SNORT and SURICATA.

- **Server Service of Windows XP:** The server service enabled the attacker of arbitrary code execution through crafted RPC request resulting in stack overflow during path canonicalization [37].
- **PostgreSQL 8.3.8:** A dictionary attack was executed in order to obtain access credentials into the PostgreSQL instance [38]. Standard PostgreSQL Linux installation had write access to `/tmp` directory, and it could call user-defined functions (UDF). UDFs utilized shared libraries located on an arbitrary path (e.g., `/tmp`). An attacker exploited this fact and copied its own UDF code to `/tmp` directory and then executed it [39].
- **DistCC 2.18.3:** A vulnerability enabled the attacker remote execution of an arbitrary command through compilation jobs that were executed on the server without any permission check [40].

Adversarial Modifications. We proposed several non-payload-based obfuscation techniques [41] when exploiting vulnerable network services as well as during the execution of legitimate communications on the services. The proposed non-payload-based obfuscation techniques are described in Table IX, Assuming the background from Section II-D, the proposed non-payload-based obfuscation techniques can modify P_c and P_s packet sets of the original connection c_m by insertion, removal and transformation of the packets. Symbols of the packet tuple (see Table XXIV) whose values are sensitive to the obfuscations include: $t, size$, ip_{off}, ip_{sum}, tcp_{sum}, tcp_{seq}, tcp_{ack}, tcp_{off}, tcp_{flags}, tcp_{win}, tcp_{urp} and $data$.[10] The modifications of P_c and P_s of the connection c_m can cause alteration of the original network connection features' values F^m to new ones (see Section II-D).

Service	CVE	CVSS
Apache Tomcat	2009-3843	10.0
DistCC service	2004-2687	9.3
MSSQL	2000-1209	10.0
PostgreSQL	2007-3280	9.0
Samba service	2007-2447	6.0
Server service of Windows	2008-4250	10.0

Table VIII: A list of vulnerable services in ASNM-NPBO dataset.

Then we built an obfuscation tool [42] that morphs network characteristics of a TCP connection at network and transport layers of the TCP/IP stack by applying one or a combination of several non-payload-based obfuscation techniques. Execution of direct communications (non-obfuscated ones) is also supported by the tool as well as capturing network traffic related to communication. The tool is capable of automatic/semi-automatic run and restoring of all modified system settings and consequences of attacks/legitimate communications on a target machine. After the successful execution of each desired obfuscation on the selected service, the output of the tool contains several network packet traces associated with pertaining obfuscations. The behavioral state diagram of the obfuscation tool is depicted in Figure Figure 4.

We applied our obfuscation tool for automatic exploitation of all enumerated vulnerable services while using the proposed obfuscations. When exploitation leading to a remote shell was successful, simulated attackers performed simple activities involving various shell commands (such as listing

[10]Note the $data$ field is sensitive to the obfuscations only in the manner of damaging or splitting the original packet's data.

Technique	Parametrized Instance	ID
Spread out packets in time	• constant delay: 1s • constant delay: 8s • normal distribution of delay with 5s mean 2.5s standard deviation (25% correlation)	(a) (b) (c)
Packets' loss	• 25% of packets	(d)
Unreliable network channel simulation	• 25% of packets damaged • 35% of packets damaged • 35% of packets damaged with 25% correlation	(e) (f) (g)
Packets' duplication	• 5% of packets	(h)
Packets' order modifications	• reordering of 25% packets; reordered packets are sent with 10ms delay and 50% correlation • reordering of 50% packets; reordered packets are sent with 10ms delay and 50% correlation	(i) (j)
Fragmentation	• MTU 1000 • MTU 750 • MTU 500 • MTU 250	(k) (l) (m) (n)
Combinations	• normal distribution delay ($\mu = 10ms$, $\sigma = 20ms$) and 25% correlation; loss: 23% of packets; corrupt: 23% of packets; reorder: 23% of packets • normal distribution delay ($\mu = 7750ms$, $\sigma = 150ms$) and 25% correlation; loss: 0.1% of packets; corrupt: 0.1% of packets; duplication: 0.1% of packets; reorder: 0.1% of packets • normal distribution delay ($\mu = 6800ms$, $\sigma = 150ms$) and 25% correlation; loss: 1% of packets; corrupt: 1% of packets; duplication: 1% of packets; reorder 1% of packets	(o) (p) (q)

Table IX: Non-payload-based obfuscation techniques with parameters and IDs.

directories, opening, and reading files). An average number of issued commands was around 10, and text files of up to 50kB were opened/read. Note that we labeled each TCP connection representing dictionary attacks as legitimate ones due to two reasons: 1) from the behavioral point of view, they independently appeared just as unsuccessful authentication attempts, which may occur in legitimate traffic as well, 2) more importantly, we employed ASNM features whose subset involves context of an analyzed TCP connection for their computation – i.e., ASNM features capture relations to other TCP connections initiated from/to a corresponding service.

Legitimate Network Traffic. The legitimate samples of this dataset were collected from two sources:

- A common usage of all previously mentioned services was obtained in an anonymized form, excluding the payload, from a real campus network in accordance with policies in force. Analyzing packet headers, we observed that a lot of expected legitimate traffic contained malicious activity, as many students did not care about up-to-date software. Therefore, we filtered out network connections yielding high and medium severity alerts by signature-based NIDS – Suricata and SNORT – through Virus Total API [30].
- The second source represented legitimate traffic simulation in our virtual network architecture and also employed all of our non-payload-based obfuscations for the purpose of partially addressing overstimulation in adversarial attacks against IDS [43], and thus making the classification task more challenging. However, only 109 TCP connections were obtained from this stage, which was also caused by the fact that services such as Server and DistCC were hard to emulate.[11] Simulation of legitimate traffic was aimed at various *SELECT* and *INSERT* statements when interacting with the database services (i.e., PostgreSQL, MSSQL); several *GET* and *POST* queries to our custom pages as well as downloading of high volume data when interacting with our HTTP server (i.e., Apache Tomcat); and several queries for downloading and uploading small files into Samba share.

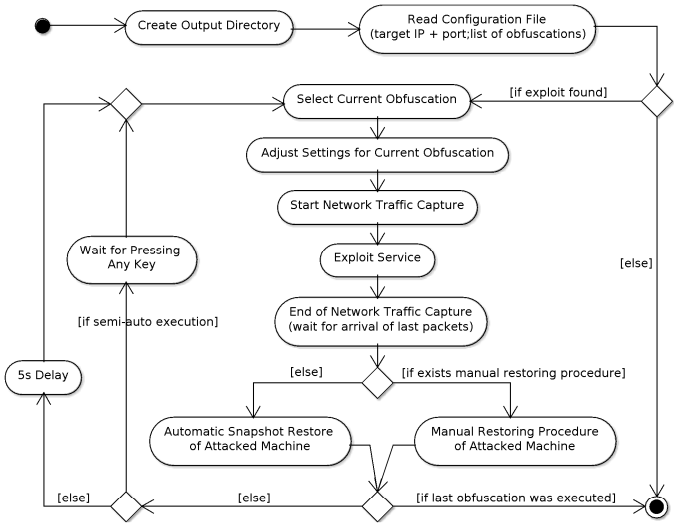

Figure 4: Behavioral state diagram of the obfuscation tool.

The class distribution of the final dataset after extraction of ASNM features is summarized in Table X

Labeling. ASNM-NPBO dataset contains four types of labels that are enumerated by increasing order of their granularity in the following:

- **label_2:** is a two-class label, which indicates whether an actual sample represents a network buffer overflow attack or a legitimate communication.
- **label_3:** is a three-class label, which distinguishes among legitimate traffic (symbol 3), direct attacks (symbols 1), and obfuscated network attacks (symbol 2).
- **label_poly:** is a label that is composed of 2 parts: (a) a three-class label, and (b) acronym of a network service.

[11]Note that additionally to those 109 TCP connections that were explicitly simulated, other 2252 TCP connections from obfuscated dictionary attacks were also considered as legitimate, and thus also helped in achieving a resistance against the overstimulation attacks.

This label represents a type of communication on a particular network service.

- **label_poly_o** is the last label, which is composed of 3 parts: (a) three-class label, (b) employed obfuscation technique, and (c) acronym of network service. The label has almost the same interpretation as label_poly but moreover introduces obfuscation technique employed (identified by ID from Table IX) into all obfuscated attack samples.

Testing with Signature-Based NIDS. To investigate the effect of the proposed non-payload-based obfuscations on signature-based NIDSs, we performed detection by SNORT and SURICATA in a similar manner as we did in the case of the tunneling obfuscations (see Section IV-B), while the same ruleset was employed.

First, we let NIDSs inspect direct attacks that exploit the current network vulnerabilities. The results of the inspection summarize the detection properties of SNORT and SURICATA, and are depicted in Table XI.We can see in the tables that SNORT overcame SURICATA and correctly detected 100.00% of direct attacks. However, only 33 direct attacks on Apache service were detected by high priority rules of SNORT, and 24 attacks were undetected. Despite it, we considered these attacks as correctly detected, as they occurred almost at the same time as other correctly predicted direct attacks, and thus might be a part of their execution. In the case of SURICATA, the only one such undetected direct attack occurred. Nevertheless, unlike SNORT, SURICATA did not fire any alert representing buffer overflow, shellcode, or remote command execution, but instead fired combination of high priority alerts related to potential corporate privacy violation:

- `ET POLICY Incoming Basic Auth Base64 HTTP Password detected unencrypted`
- `ET POLICY Outgoing Basic Auth Base64 HTTP Password detected unencrypted`
- `ET POLICY HTTP Request on Unusual Port Possibly Hostile`
- `ET POLICY Internet Explorer 6 in use Significant Security Risk,`

Network Service	Count of TCP Connections			
	Legitimate	Direct Attacks	Obfuscated Attacks	Summary
Apache Tomcat	809	61	163	1033
DistCC	100	12	23	135
MSSQL	532	31	103	666
PostgreSQL	737	13	45	795
Samba	4641	19	44	4704
Server	3339	26	100	3465
Other Traffic	647	–	–	647
Summary	10805	162	478	11445

Table X: ANSM-NPBO dataset distribution.

	Direct Attacks		
	Detected	Total	%
Apache Tomcat	33 +28	61	100.00
DistCC	12	12	100.00
MSSQL	31	31	100.00
PostgreSQL	13	13	100.00
Samba	19	19	100.00
Server	26	26	100.00
Overall Detection	162	162	100.00
ADR* per Service			100.00

*Average detection rate.

(a) SNORT

	Direct Attacks		
	Detected	Total	%
Apache Tomcat	56 +5	61	100.00
DistCC	0	12	0.00
MSSQL	31	31	100.00
PostgreSQL	0	13	0.00
Samba	0	19	0.00
Server	26	26	100.00
Overall Detection	118	162	72.84
ADR* per Service			50.00

*Average detection rate.

(b) SURICATA

Table XI: Detection of direct attacks in the ASNM-NPBO dataset by SNORT and SURICATA.

which we decided to consider as correctly detected. If we would not consider them as correctly detected, then SURICATA were not detected any attack on the Apache service.

Next, we analyzed detection capabilities of both NIDSs on obfuscated attacks and the results are depicted in Table XII. Comparing the detection rate of SNORT and SURICATA on obfuscated attacks, we can conclude that SNORT overcame SURICATA again and the ratio of their correct detection was almost the same as in the case of direct attacks (see Table XI). The only difference occurred during the exploitation of a vulnerability in Server service, where two instances of obfuscated attacks were not detected by any NIDS. These two instances utilized obfuscations with IDs (f) and (g), both from a category of unreliable network traffic channel simulation techniques (see Table IX). There were also several undetected obfuscated attacks on Apache service in both NIDSs, but we were able to track their occurrences and associate them as part of other correctly detected attacks; hence, the detection rate for Apache service achieved 100.00% for both NIDSs. Regarding Apache service, SURICATA once again did not fire any alert detecting malicious content, but instead, it fired the previously mentioned combination of high priority alerts stating corporate privacy violation, which we, once again, considered as a correct detection. Also, note that SURICATA fired one non-high-priority alert classified as potentially bad traffic in all instances of direct and obfuscated attacks exploiting PostgreSQL service:

- `ET POLICY Suspicious inbound to PostgreSQL port 5432.`

	Obfuscated Attacks		
	Detected	**Total**	**%**
Apache Tomcat	128 +36	164	100.00
DistCC	23	23	100.00
MSSQL	103	103	100.00
PostgreSQL	45	45	100.00
Samba	44	44	100.00
Server	98	100	98.00
Overall Detection	478	480	99.58
ADR* per Service			99.67

*Average detection rate.

(a) SNORT

	Obfuscated Attacks		
	Detected	**Total**	**%**
Apache Tomcat	162 +1	163	100.00
DistCC	0	23	0.00
MSSQL	103	103	100.00
PostgreSQL	0	45	0.00
Samba	0	44	0.00
Server	98	100	98.00
Overall Detection	364	478	76.15
ADR* per Service			49.67

*Average detection rate.

(b) SURICATA

Table XII: Detection of obfuscated attacks in ASNM-NPBO dataset by SNORT and SURICATA.

However, we did not consider it as a correct detection due to the low priority of the alert. As discussed in Section IV-B, VirusTotal likely uses a paranoid rule set, and thus fired alerts may contain false positives. Comparing fired alerts before and after obfuscation, we can see that utilized NIDSs detected most of the obfuscated attacks by non-payload-based, but there were also a few cases where they failed, and thus, evasion was successful.

V. Benchmarking the Datasets

In the previous research [4], [16], [7], [44], [41], [42], we conducted several machine learning experiments with ASNM datasets, and we summarize them in the current section.

A. ASNM-CDX-2009 Dataset

Forward Feature Selection. First, we used 5-fold cross-validation and forward feature selection (FFS) on top of the Naive Bayes classifier with kernel functions for the estimation of density distribution, which represents a non-parametric estimation method. In FFS, we accepted one iteration without improvement as we wanted to avoid the selection process to get stuck in local extremes. The maximal number of selected features was limited to 20 (although it was never reached). We used the binary label of the dataset (i.e., $label\_2$), and we obtained $F_1\text{-}measure$ over 90% and an average recall of both classes equal to 92%.

Additionally, we compared the performance of ASNM features with discriminators of A. Moore [8] in the same setting, and we concluded that both feature sets yielded similar results. Moreover, when we merged both feature sets and rerun FFS, $F_1\text{-}measure$ reached 98.87% [16].

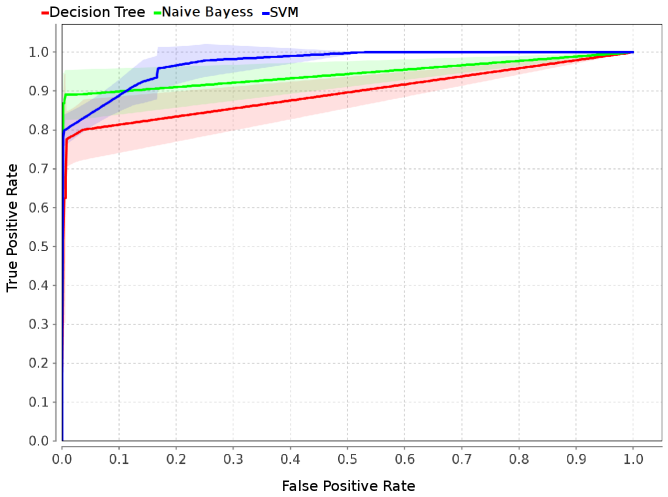

Figure 5: ROC diagram comparing a few classifiers on the ASNM-CDX-2009 dataset.

Comparison of Several Classifiers. Next, we compared three non-parametric classifiers while using a subset of ASNM features obtained by FFS with the Naive Bayes classifier – the selected features are enumerated and described in Table XXV of Appendix. The individual confusion matrices that we obtained are presented in Table XIII (Naive Bayes with kernel density estimation), Table XIV (Decision Tree with Gini index as a selection criterion for splitting of attributes), and Table XV (SVM with radial basis function). Finally, all classifiers were compared using ROC curves, and a comparison is depicted in Figure 5. Note that ROC curves also depict a variance coming from a cross-validation method, which is shown by line-adjacent transparent areas.

B. ASNM-TUN Dataset

Forward Feature Selection. Alike the case of the previous dataset, we again started with the FFS method using the same Naive Bayes classifier and 5-fold cross-validation, while we allowed acceptance of one FFS iteration without improvement to avoid the selection process becoming stuck in local extremes. All cross-validation experiments have been adjusted to employ stratified sampling during assembling of folds, which ensured equally balanced class distribution of each fold. We performed two-class prediction (i.e., using the label denoted as $label\_2$). Some features existed, which were inconvenient for comparison of synthetic attacks with legitimate traffic captured in a real network; therefore, such features were removed from the dataset in the pre-processing phase of our experiment. The examples include TTL-based features, IP addresses, ports, MAC addresses, the occurrence of source/destination host in the monitored local network, some context-based features, etc. The experiment consisted of two executions of FFS. The first took as an input just legitimate traffic and direct attack entries and represented the case where the classifier

Classification Accuracy: 99.86% ±0.07		True Class: Legit. Flows	True Class: Attacks	Precision
Predicted Class	**Legit. Flows**	5726	7	99.88%
	Attacks	1	37	97.37%
	Recall	99.98%	84.09%	$F_1 = 90.24\%$

Table XIII: Performance of the Naive Bayes classifier on the ASNM-CDX-2009 dataset.

Classification Accuracy: 99.71% ±0.07		True Class: Legit. Flows	True Class: Attacks	Precision
Predicted Class	**Legit. Flows**	5721	11	99.81%
	Attacks	6	33	84.62%
	Recall	99.90%	75.00%	$F_1 = 79.52\%$

Table XIV: Performance of the decision tree classifier on the ASNM-CDX-2009 dataset.

Classification Accuracy: 99.81% ±0.06		True Class: Legit. Flows	True Class: Attacks	Precision
Predicted Class	**Legit. Flows**	5726	10	99.83%
	Attacks	1	34	97.4%
	Recall	99.98%	77.27%	$F_1 = 86.07\%$

Table XV: Performance of the SVM classifier on the ASNM-CDX-2009 dataset.

was trained without knowledge about obfuscated attacks. The second execution took as input the whole dataset of network traffic – consisting of legitimate traffic, direct attacks as well as obfuscated ones, and therefore represented the case where the classifier was aware of obfuscated attacks. The selected features of both executions are depicted in Table XXVI of Appendix. The penultimate column of the table (i.e., FFS DOL) denotes the selected features where the whole dataset was utilized for the FFS, and the last column (i.e., FFS DL) denotes the case where only direct attacks and legitimate traffic were taken into account.

Several mutual features were selected in both cases, which means they provided a value regardless of whether obfuscation was performed or not. Almost all of the following experiments will use the feature set gained from the second execution (i.e., FFS DOL), as we consider them as more appropriate for general behavior representation of both kinds of attacks.

Evasions. First, we executed an experiment that performed detection of malicious obfuscated attacks by the classifier trained on all direct attacks and legitimate traffic samples. It represented the situation when the classifier had no previous knowledge about obfuscated attacks, and therefore we used FFS DL feature set. As a result, only 35.63% of obfuscated attacks (i.e., 31 of 87) were correctly detected by the classifier, and thus an average recall and $F_1\text{-}measure$ of the classifier were equal to 67.53% and 52.10%, respectively. An associated confusion matrix is depicted in Table XVI. We realized that 64.36% of obfuscated attacks were incorrectly predicted as legitimate traffic, and thus caused an evasion of the classifier.

Training Data Augmentation. Our second binary classification experiment considered explicit information about obfuscated attacks in the training phase of the classifier. Therefore, we used direct and obfuscated attacks labeled as one class while using 5-fold cross-validation. FFS DOL feature set was used for the purpose of this experiment. The resulting confusion matrix with performance measures is shown in Table XVII. The outcome of this experiment indicates a high class recall and $F_1\text{-}measure$ of the classifier trained with knowledge about some obfuscated attacks.

Classification Accuracy: 78.41%		True Class: Legit. Flows	True Class: Obfus. Attacks	Precision
Predicted Class	**Legit. Flows**	176	56	75.86%
	Obfus. Attacks	1	31	96.88%
	Recall	99.44%	35.63%	$F_1 = 52.10\%$

Table XVI: Detection of unknown obfuscated attacks by the Naive Bayes classifier trained on all direct attack samples and legitimate traffic samples from the ASNM-TUN dataset.

Comparison of Several Classifiers. For the purpose of performance comparison of various classifiers, we executed 5-fold cross-validation on the other two non-parametric classifiers – decision tree and SVM. FFS DOL feature set was used in this experiment as the input for the classifiers working with two class prediction (i.e., using $label\_2$). At first, we evaluated the performance of the SVM classifier, which utilized a radial basis function as the non-linear kernel. The adjacent confusion matrix is depicted in Table XVIII. The next experiment was performed with the decision tree classifier, which utilized gini index as selection criterion for splitting of attributes. The adjacent result is represented by confusion matrix in Table XIX. The results of both performance evaluation experiments can be compared to the result of the Naive Bayes classifier represented by the confusion matrix from Table XVII. Considering an average recall of all classes and $F_1\text{-}measure$, we can say that the Naive Bayes classifier achieved the best results, following by the decision tree, and finally by SVM.

All the classification models were compared by ROC method (see Figure 6). Note that comparison of ROC ran above the cross-validation method, and thus generated certain variability, which is shown by line-adjacent transparent areas. For more experiments with this dataset, including tri-nominal and multi-nominal labels and individual feature analysis, we refer the reader to [7], [44], and [16].

Classification Accuracy: 99.49% ± 0.62%		True Class: Legit. Flows	True Class: All Attacks	Precision
Predicted Class	**Legit. Flows**	176	1	99.44%
	All Attacks	1	216	99.54%
	Recall	99.44%	99.54%	$F_1 = 99.54\%$

Table XVII: Performance of the Naive Bayes classifier on the ASNM-TUN dataset using the binary label (i.e., $label\_2$).

Classification Accuracy: 80.96% ± 3.51%		True Class: Legit. Flows	True Class: All Attacks	Precision
Predicted Class	**Legit. Flows**	176	74	70.40%
	All Attacks	1	143	99.31%
	Recall	99.44%	65.90%	$F_1 = 79.22\%$

Table XVIII: Performance of the SVM classifier on the ASNM-TUN dataset.

Classification Accuracy: 95.93% ± 2.47%		True Class: Legit. Flows	True Class: All Attacks	Precision
Predicted Class	**Legit. Flows**	169	8	95.48%
	All Attacks	8	209	96.31%
	Recall	95.48%	96.31%	$F_1 = 96.31\%$

Table XIX: Performance of the decision tree classifier on the ASNM-TUN dataset.

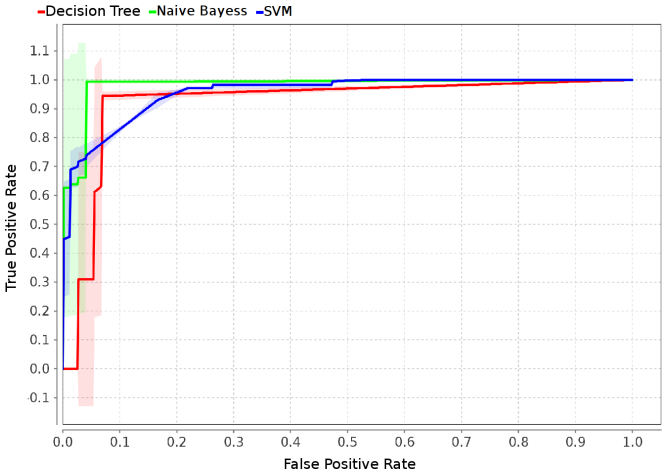

Figure 6: ROC diagram comparing a few classifiers on the ASNM-TUN dataset.

C. ASNM-NPBO Dataset

Forward Feature Selection. Alike the case of the previous datasets, we again started with the FFS method using the same Naive Bayes classifier and 5-fold cross-validation, while we allowed acceptance of one FFS iteration without improvement, and we excluded the same inconvenient features as in Section V-B. We performed two-class prediction (i.e., using $label\_2$) in two executions of FFS using the Naive Bayes classifier – the first execution did not contain obfuscated attack samples (i.e., FFS DL) and the another one included these samples (i.e., FFS DOL). The selected features of both executions are depicted in Table XXVII of Appendix.

Evasions. 5-fold cross-validation with FFS DL features was performed using all direct attack samples and legitimate traffic samples. The performance measures of three classifiers validated by the cross-validation are shown in Table XX. Then the classifiers trained on all direct attacks and legitimate traffic

Classifier	TPR	FPR	F_1 (↑)	Avg. Recall
Naive Bayes	98.15%	0.02%	98.45%	99.07%
Decision Tree	95.68%	0.09%	94.80%	97.80%
SVM	82.72%	0.01%	90.24%	91.36%

Table XX: Direct attacks and legitimate traffic cross validation on ASNM-NPBO dataset.

Classifier	TPR (↑)	Δ TPR
Naive Bayes	52.30%	-45.85%
Decision Tree	36.61%	-59.07%
SVM	15.90%	-66.82%

(a) Obfuscated attacks

Classifier	TPR (↑)	Δ TPR
Naive Bayes	64.38%	-33.77%
Decision Tree	52.03%	-43.65%
SVM	26.25%	-56.47%

(b) All attacks

Table XXI: Prediction of obfuscated attacks and all attacks in the ASNM-NPBO dataset by classifiers trained without knowledge about obfuscated attacks.

samples were applied for the prediction of the obfuscated attacks and all attacks, respectively (see Table XXI).[12] Here TPRs were deteriorated for all classifiers, which means that some obfuscated attacks were successful – they were predicted as legitimate traffic, and thus caused evasion of the classifiers.

Training Data Augmentation. To improve the resistance of the classifiers against evasions, we widened their knowledge about different mixtures of obfuscated attack instances, which was accomplished by random 5-fold cross-validation of the whole dataset. In this experiment, we use FFS DOL features that consider knowledge about obfuscated attacks for updating not only the model of the classifier but also the underlying feature set (in contrast to the previous experiment). Additionally, we show the results with FFS DL features, which consider updating the model only. The results of this experiment are shown in Table XXII. Comparing against the results from the previous experiment (see FPRs from Table XX and TPRs from Table XXIb), most of the classifiers were significantly improved in TPR, while FPR was deteriorated only slightly. Hence, the classifiers trained with knowledge about some obfuscated attacks were able to detect the same and similar obfuscated attacks later.

Comparison of Several Classifiers. From the previous experiments, we can say that the Naive Bayes classifier was the least sensitive to evasions by non-payload-based obfuscations (see Table XX), while SVM was the most sensitive classifier. This might be caused by overfitting of the training data. Note that all classifiers used the feature sets selected by FFS with the Naive Bayes classifier. However, we also rerun FFS with

[12]Note that we do not depict FPRs in the tables since no changes to legitimate traffic was made, hence FPRs remain the same as in Table XX.

Classifier	TPR	FPR	Δ TPR	Δ FPR	$F_1(\uparrow)$	Avg. Recall
Naive Bayes	93.28%	0.73%	+28.90%	+0.71%	90.73%	96.28%
SVM	80.31%	0.05%	+54.06%	+0.04%	88.70%	90.13%
Decision Tree	67.34%	0.36%	+15.31%	+0.27%	77.65%	83.49%

(a) FFS DL features

Classifier	TPR	FPR	Δ TPR	Δ FPR	$F_1(\uparrow)$	Avg. Recall
SVM	99.53%	0.13%	+73.28%	+0.12%	98.68%	99.70%
Decision Tree	98.44%	0.19%	+46.41%	+0.10%	97.60%	99.13%
Naive Bayes	98.75%	0.99%	+34.37%	+0.97%	91.66%	98.88%

(b) FFS DOL features

Table XXII: Cross validation of the whole ASNM-NPBO dataset, representing the situation where classifiers were aware of some obfuscated attacks, and therefore they brought performance improvement in contrast to classifiers aware only about direct attacks (see Table XXI).

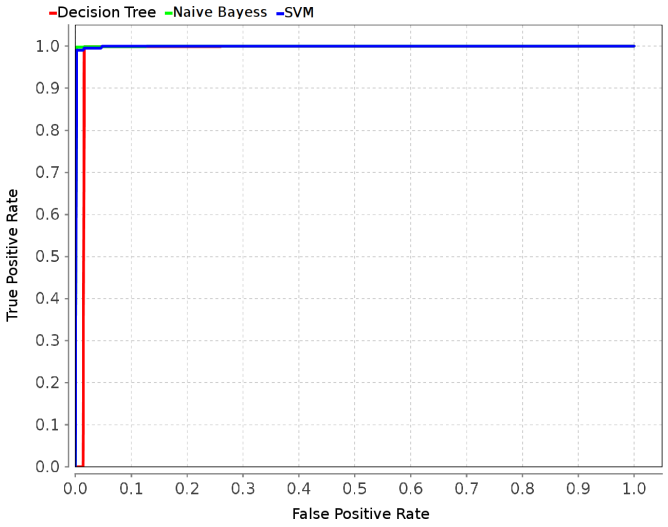

Figure 7: ROC diagram comparing a few classifiers on the ASNM-NPBO dataset.

individual classifiers, but obtained results were much worse than using the features selected by the Naive Bayes classifier.

After augmentation of a training data without updating the feature set (see Table XXIIa), we observe that the Naive Bayes classifier is the most robust one. However, when making a training data augmentation with updating the feature set (see Table XXIIb), other classifiers perform better than Naive Bayes, which might be again caused by overfitting of them.

Finally, we compared the classification models by ROC method (see Figure 7). The best results were achieved in the case of the Naive Bayes classifier and SVM. For more experiments with this dataset, including tri-nominal and multi-nominal labels, detection of unknown obfuscations by a custom leave-one-out validation, and individual feature analysis, we refer the reader to [42] and [16].

VI. Related Work

In this section, we summarize public datasets intended for the evaluation of network intrusion detection solutions. We partition all datasets into two categories. The first category represents datasets containing *raw network traffic traces*, and the second category represents datasets containing *high-level features* extracted from underlying network traces.

A. Datasets of Network Traffic Traces

Datasets from this category have one property in common: they contain network traffic traces with optional data serving for labeling purposes. The first four representatives of this category are large collections of datasets and are referred to as projects – MWS [45], PREDICT [46], CAIDA [47], NETRESEC [48]. The next four examples of this category represent just one specific collection of network data and are referred to as datasets – DARPA [49], CCRC [50], CDX [6], CONTIAGO [51]. We describe them in the following.

*1) **Project MWS**:* The project MWS represents a collection of various types of datasets that are primarily intended for use in anti-malware research [45], but some of them are also applicable in network intrusion detection. A summary of the MWS datasets is available in Japanese [52], [53], [54], [55], [56], and it covers three categories of datasets, which is based on phases of attacks: (1) probing, (2) infection, and (3) malware activities after infection.

From the perspective of network intrusion detection, we consider PRACTICE, D3M, CCC, and NICTER as related datasets of MWS. However, for network intrusion detection is also important to have a ground truth, which can be inferred from the MWS datasets called FFRI, IIJ MITF, D3M, and CCC. In the following, we briefly describe these datasets.

Cyber Clean Center (**CCC**) dataset consists of a malware sample, honeypot packet trace, and malware collection log. The dataset was collected from server-side, high-interaction honeypots operated by the CCC in a distributed manner. Over a hundred of honeypots gathered attacks and collected malware through multiple ISPs. These honeypots were based on Windows 2000 and Windows XP SP1 virtual machines. Drive-by Download Data by Marionette (**D3M**) dataset is a set of packet traces collected from the web-client-based high-interaction honeypot system Marionette [57], [58], which is built upon Internet Explorer with several vulnerable plugins, such as Adobe Reader, Flash Player, WinZip, QuickTime. The datasets contain packet traces for the two periods: at infection and after infection. The **IIJ MITF** dataset is collected by server-side, low interaction honeypots based on the open-source honeypot Dionaea [59]. This dataset contains attack communication and malware collection logs from a hundred honeypots between July 2011 and April 2012 in order to discover the trends of bot and botnets. The **PRACTICE** dataset contains the packet traces obtained during long-term dynamic analysis of five malware samples (Zbot, SpyEye, etc.)

and their metadata, while the focus was put on network activity of malware, using the dynamic analysis system proposed in [60]. The analysis period of the dataset is one week in the middle of May 2013. The **FFRI** dataset focuses on the internal activities that occurred at a host by the influence of malware and are generated by dynamic analysis systems – Cuckoo sandbox [61] and FFR Yarai Analyzer Professional [62]. The **NICTER** darknet dataset is a set of packet traces collected from April 2011 to 2014 using the darknet monitoring system NICTER [63]. The packet traces contain scan packets to explore the reachable hosts by worms, backscatter packets caused by source IP address spoofing, distributed reflection denial of service (DRDoS) attacks using DNS and NTP, etc.

*2) **Project PREDICT**:* The project PREDICT [46] provides 430 datasets in 14 categories contributed by several data providers. From all 14 categories, just three of them are relevant to the network intrusion detection and could be used for evaluation purposes:

Blackhole Address Space Data: is collected by monitoring routed but unused IP address space that does not host any networked devices. Systems that monitor such unoccupied address space have a variety of names, including darkspace, darknets, network telescopes, blackhole monitors, sinkholes, and background radiation monitors. Packets observed in the darkspace can originate from a wide range of security-related events, such as scanning in search of vulnerable targets, backscatter from spoofed denial-of-service attacks, automated spread of the Internet worms or viruses, etc. The related subcategory of this category is UCSD Archived Network Telescope Data. The archived files are in PCAP format. Source IP addresses are not anonymized.

IP Packet Headers: these datasets are comprised of headers of network data, containing information such as anonymized source and destination IP addresses and other IP and transport header fields. No payload is included. Depending on the specific dataset, this category of data can be used for characterization of typical internet traffic, or of traffic anomalies such as distributed denial of service attacks, port scans, or worm outbreaks.

Synthetically Generated Data: are generated by capturing information from a synthetic environment, where benign user activity and malicious attacks are emulated by computer programs. In this category, full network packets, as well as firewall logs, application logs, and malicious attacks are available, without any risk of compromising the privacy of real people. In this category, one can know and document complete "ground truth". Therefore, this category is well suited for the evaluation of NIDS systems.

Note that **IDS and Firewall Data** category contains large collection of logs submitted in a standard format but generated from a diverse set of hardware and software systems. It does not contain any PCAP files, therefore it could not be used for IDS evaluation purposes. If we were to consider categorization of datasets from the MWS project, then mentioned datasets of PREDICT would represent probing and infection categories.

*3) **Project CAIDA**:* Center for Applied Internet Data Analysis (CAIDA) [47] collects several different network data types at geographically diverse locations. The data are provided by various organizations, for whose data CAIDA guarantees anonymity and privacy.

The CAIDA datasets are dived into three categories that reflect a status of a collection process:

Ongoing: the data collection for such dataset is still active, while collections are added regularly,

One-Time Snapshot: the dataset comes from a single collection event that only occurred once. Future events will have a different dataset names,

Complete: a formerly ongoing data collection that is finished, and will not be resumed.

From the network intrusion detection perspective, CAIDA includes datasets containing e.g. DDoS attacks [64], [65], botnet traffic [66], dumps of various well known worms (Conficker [67], Code-Red [68], Witty [69]). These datasets could be utilized for the evaluation of intrusion detection approaches after a further analysis followed by labeling where it is not available. If we were to consider a categorization of datasets from MWS project (see Section VI-A1), then CAIDA would belong to probing and infection categories.

*4) **Project NETRESEC**:* Network Forensics and Network Security Monitoring (NETRESEC) [48] is an independent software vendor with focus on the network security field. NETRESEC specializes in software for network forensics and analysis of network traffic. In addition, NETRESEC maintains a comprehensive list of publicly available PCAP files that can be used for the evaluation of network intrusion detection approaches as well. The datasets are divided into six categories:

Cyber Defence Exercises: this category includes network traffic from exercises and competitions, such as Cyber Defense Exercises and red-team/blue-team competitions.

Capture the Flag Competitions: it contains files from capture-the-flag (CTF) competitions and challenges.

Malware Traffic: it contains PCAP files of captured malware traffic from honeypots, sandboxes, and intrusions.

Network Forensics: Network forensics training, challenges and contests.

SCADA/ICS Network Captures: files with attacks against Industrial Control Systems; files captured at Industrial Control System Village (4SIC, CTF, DEF CON 22).

Uncategorized PCAP Repositories: various captures that often represent data for intrusion detection purposes.

If we were to consider the categorization of datasets from MWS project (see Section VI-A1), then NETRESEC datasets would represent probing and infection categories.

*5) **DARPA 1998 and 1999 Datasets**:* The Cyber Systems and Technology Group [49] of MIT Lincoln Laboratory has collected the first standard corpora for evaluation of network intrusion detection systems in 1998 and 1999. There were

collected two datasets DARPA 1998 and 1999, and later there were released three datasets marked as DARPA 2000, which address specific network scenarios. If we were to consider the categorization of datasets from Section VI-A1, then DARPA datasets would represent probing and infection categories.

*6) **CCRC 2006 Dataset**:* The authors F. Massicotte et al. [50] developed a framework for automatic evaluation of intrusion detection systems, and they collected an examplar dataset consisted of several network attack simulations. We denote this dataset as CCRC 2006, because of the main author was, at the time of the article was written, an employee of Canada Communication Research Center in Ottawa.

The dataset is specific to signature-based network intrusion detection systems and contains only well-known attacks, without background traffic. The purpose of the dataset is testing and evaluation of the detection accuracy of IDS in the case of successful and failed attack attempts. The paper also reports an initial evaluation of the framework on two well-known IDS, namely SORT [70] and Bro [71]. In the proposed framework, the authors are able to automatically generate a large dataset, with which it is possible to automatically test and evaluate intrusion detection systems. Note that the framework also contains a mutation layer that is able to perform various L2 and L3 protocol based obfuscations using tools Fragroute [72] and Whisker [73]. If we were to consider the categorization of datasets from Section VI-A1, then CCRC 2006 dataset would represent the infection category.

*7) **CDX 2009 Dataset**:* The CDX 2009 dataset was introduced by Sangster et al. [6] and it contains data in tcpdump format as well as SNORT [20] intrusion prevention logs. We used this dataset in our research, and it is described in Section IV-A. If we were to consider the categorization of datasets from the MWS project (see Section VI-A1), then CDX 2009 dataset would represent probing and infection categories.

*8) **Twente 2009 Dataset**:* The Twente 2009 dataset [74] consists of 14.2M network flows (i.e., 155M packets) collected during a period of 9 days in 2008, where 7.6M of intrusion alerts were generated. The flows in this dataset were assembled by a modified version of *softflowd* utility, and 98% of them have been labeled by the authors. The authors collected dataset by a honeypot installed on virtual host Citrix XenServer 5. The deployed honeypot run with three opened services: OpenSSH, Apache web server, and FTP server *proftp*.

*9) **ISCX 2012 Dataset**:* The authors of [75] presented guidelines for the generation of benchmark dataset consisting of creating a malicious and benign profile that were later executed during dataset generation. The authors generated their own dataset of network traffic (including the payload) for various network services such as HTTP, SMTP, SSH, IMAP, POP3, FTP. In sum, they collected 2.5M of network flows, consisting of 125M of packets.

*10) **Contagio 2015 Dataset**:* Contagio dataset [51] contains a collection of PCAP files from malware analysis. The authors collected almost 1000 malicious PCAPs from various public sources. The collection is irregularly updated with new PCAP files. PCAPs in the Contagio dataset include implicit expert knowledge about the occurrence of attacks/malware. If we were to consider the categorization of datasets from the MWS project (see Section VI-A1), then the Contagio dataset would belong to categories representing an infection and malware activities after an infection.

B. Datasets Consisting of High-Level Features

The current category of datasets contains representatives that were built from network traffic traces, hence it can be interpreted as a post-processed version of the former category. The current category of datasets contains five representatives: KDD Cup '99 [10], NSL KDD '99 [76], Moore's 2005 [8], Kyoto 2006+ [9], and OptiFilter 2014 [77].

*1) **KDD Cup '99**:* In 1999, KDD Cup '99 [10] dataset was created, and it is based on the DARPA 1998 dataset of network dumps. It has been used for evaluating intrusion detection methods that analyze features extracted from network traffic and host machine data. The training dataset consists of approximately 4,9M single connection samples from seven weeks of network traffic, each labeled as either normal or attack, containing 41 features per connection sample. Similarly, the two weeks of testing data yielded around two million connection samples. The datasets contain a total number of 24 training attack types, with additional 14 types in the testing dataset. The simulated attacks fall into four main categories [10], [76]:

Denial of Service Attack (DOS): is an attack in which the attacker makes some computing or memory resource too busy or too full to handle legitimate requests, or denies legitimate users access to a machine, e.g., SYN flood.

Remote to Local Attack (R2L): occurs when an attacker who has the ability to send packets to a machine over a network, but who does not have an account on that machine, exploits some vulnerability to gain local access as a user of that machine, e.g., guessing password, remote buffer overflow attacks.

User to Root Attack (U2R): is a class of attacks where the attacker begins with access to a normal user account on the system (e.g., a dictionary attack) and then is able to exploit some vulnerability to gain superuser access to the system, e.g. local buffer overflow attacks.

Probing: is an attempt to gather information about a network of computers for the purpose of circumventing its security controls, e.g. port scanning for vulnerable services.

The features of the KDD '99 dataset are, according to [10], divided into three categories:

- **Basic Features:** of individual communications. This category encapsulates all the attributes that can be extracted from TCP or UDP communications.
- **Content Features:** are extracted within a connection suggested by domain knowledge. Unlike most of the DoS

and Probing attacks, the R2L and U2R attacks cannot be described by any volumetric or frequency pattern. This is because the DoS and Probing attacks involve many connections to some hosts in a very short period of time, while the R2L and U2R attacks are embedded in the data portions of the packets associated with a single connection. To detect such attacks, features that inspect application-level behavior are employed, e.g., the number of failed login attempts. These features parse the payload of packets regardless of it is encrypted or not. Hence, they cannot be extracted from only network data.

- **Traffic Features:** (a.k.a., time-based features) calculate statistics related to protocol behavior, service, etc., and they are computed using a two-second time window. This category of features is further divided into two subcategories [78]:
  - **Same Host Features:** examine only the connections in the past two seconds that have the same destination host as the current connection.
  - **Same Service Features:** examine only the connections in the past two seconds that have the same service as the current connection.

Stolfo et al. [78] criticize time-based features since there exist several slow probing attacks that scan host using a much larger time interval than two seconds. Rather than using a time window of two seconds, Stolfo et al. [78] use a window of 100 connections, and constructed a mirror set of host-based traffic features, replacing original time-based traffic features.

*2) **NSL KDD '99**:* Deficiencies of the KDD Cup '99 dataset were discussed in [76]. The main deficiency of original dataset relates to redundant replicated entries (78% in the training set and 75% in the testing set). The original dataset was modified, reduced, and release as the NSL KDD '99 dataset. The training dataset contains about 130 thousand entries and the testing one about 23 thousand. In NSL KDD '99 dataset, all samples are sorted into the original 24 classes as well as into two classes. Complete NSL KDD '99 dataset is available at [79].

*3) **Moore's 2005**:* The Moore's 2005 datasets [8] are primarily intended to aid in the assessment of network traffic classification. A number of datasets are described; each dataset consists of a number of objects, and each object is described by a group of features (a.k.a., discriminators [8]). Each object within each dataset represents a single flow of TCP packets between a client and a server. Features for each object consist of processed input data by discriminators extraction, and these features serve as the input for probabilistic classification techniques. Input data is obtained by the Network Monitor tool designed in [80]. In contrast to previously described KDD datasets, Moore's dataset is based purely on network traffic traces, and there is not utilized any information from host machines during the extraction of the features.

*4) **Kyoto 2006+**:* J. Song et al. [9] presented an evaluation dataset for NIDS, which was built from the 3 years of real-network traffic (since Nov. 2006 to Aug. 2009) that was collected by various types of honeypots. The total number of honeypots used for collection is 348, including two black hole sensors with 318 unused IP addresses. The most of honeypots were rebooted and restored original HDD image immediately after a malicious packet was observed. For inspection of captured traffic, the authors use three independent security SW: SNS7160 IDS system [81], Clam AntiVirus software [82], and Ashula [83]. Later on, the authors have deployed SNORT [70] to their infrastructure. The dataset contains over 50 millions of normal sessions and over 43 millions of attack sessions. The authors regarded all traffic data captured from their honeypots as attack data and all traffic data captured at their legitimate mail and DNS server as normal data. Also, among the attack sessions, there were observed over 425 thousands of sessions that were related to unknown attacks, because they did not trigger any IDS alerts, but they contained shellcodes detected by Ashula.

The Kyoto 2006+ dataset consists of 14 statistical features taken from the KDD Cup '99 dataset as well as 10 additional features that can be used for further analysis and evaluation of NIDSs. The authors have not used any content-based features (extracted from host data) and focused only on network traffic data. In addition to statistical features, the authors extracted other 10 features that enable them to investigate more effectively what kinds of attacks occurred (e.g., reflecting granularity of ground truth). The Kyoto 2006+ dataset is available at [84].

*5) **OptiFilter 2014 – Persistent Dataset Generation**:* Salem et al. proposed OptiFilter [77], a framework that on-the-fly constructs connection vectors from data flows. The framework collects network packets and host events continuously in real-time, parses them to a queue of dynamic windows, and then it generates connection vectors. Datasets generated by the framework can be utilized for the evaluation of NIDSs.

OptiFilter handles ARP, ICMP, IP/TCP, and IP/UDP protocols. Moreover, it utilizes a finite state machine on TCP and UDP connections for monitoring of their state until a connection is closed or a certain condition is satisfied. All host-based features are collected using SNMP traps, a mechanism that allows systems to send messages to a trap receiver. Within Windows machines, the Windows Management Instrumentation is used to filter events and send them as SNMP traps via WMI SNMP-Provider. In contrast, the Linux systems use syslog daemon to generate SNMP traps using the NetSNMP agent. The extracted features of OptiFilter framework are influenced by KDD Cup 99 [10] and Kyoto 2006 [9] datasets and consist of three categories:

Network-based Features: timestamp, source and destination IPs, ports, protocol type, service, transferred Bytes, connection state (using BRO [71]), the number of fragmentation errors.

Traffic Features: are statistical and are derived from the basic features. They are divided into two types, time-based traffic features, and connection-based traffic features. Both types are distinguished and treated differently.

The former type is calculated based on a dynamic time window, e.g., the last five seconds, while the latter type is calculated on a configurable connection window, e.g., the last 1000 connections.

Content Features: the features are obtained directly from a monitored host using SNMP. Examples are the number of failed login attempts, the indication of a successful login, and the indication of obtaining a root shell.

In the evaluation, the authors generated a dataset called SecMonet, in which 17 common services were captured (e.g., FTP, SSH, telnet, SMTP, SMB, NFS, etc.). However, it is not clear whether the dataset contains a self-collected malicious traffic, or it is only substituted from KDD Cup '99.

*6) **CICIDS 2017 Dataset**:* CICIDS 2017 dataset [85] consists of network attacks such as DoS, DDoS, Brute force, XSS, SQL injection, Heartbleed, infiltration through the scam, and port scanning. The authors generated benign data based on the extracted profile from an analysis of 25 users, which is in line with the approach proposed in [75]. The infrastructure used for the data collection consisted of 15 vulnerable Linux-based & Windows-based machines and 4 attacker machines. Further, the authors extracted 80 features using CICFlowMeter tool [12] and provided them along with the network traffic traces.

VII. Discussion

Age of Vulnerabilities. Although there exist a plethora of publicly available exploit-codes for contemporary vulnerabilities, the situation with corresponding available vulnerable SW is more difficult due to understandable prevention reasons imposed by SW vendors. Therefore, we were able to contain only older available high-severity vulnerable services that are outdated. However, we conjecture that from the point of view of non-payload-based network intrusion detection (not inspecting the payload of packets), the behavioral characteristics of simulated high-severity attacks are similar regardless of the age of vulnerabilities. In particular, we refer to the buffer overflow attacks, which are executed in a few stages involving a repeated transfer of one or more packets with the maximum payload.

Cross-Dataset Evaluation. In this paper, we provided only a basic benchmarking of several supervised classifiers on ASNM datasets. Nevertheless, it is worth to note that different benchmarking techniques can be used as well. One example is cross-dataset evaluation, where the target classifier is trained on the input data of one dataset, and then it is evaluated on data taken from another dataset. We leave these tasks as an open challenge for the community.

VIII. Conclusion

In this paper, we presented three datasets consisting of extracted high-level network features (ASNM features). These datasets are intended for non-payload-based network intrusion detection and adversarial classification, enabling to test evasion resistance of machine learning-based classifiers. In detail, ASNM-CDX-2009 dataset might serve for basic benchmarking of machine learning-based classifiers, while ASNM-TUN and ASNM-NPBO datasets might serve for more advanced benchmarking of these classifiers, such as testing the classifiers on evasion resistance. In future work, we will extend ASNM datasets with data collected from other experiments.

References

[1] "Top Intrusion Attacks." [Online]. Available: https://www.mcafee.com/threat-intelligence/ips/top-attacks.aspx

[2] "Top Targeted Vulnerabilities." [Online]. Available: https://www.us-cert.gov/ncas/alerts/TA15-119A

[3] H. Debar, M. Dacier, and A. Wespi, "A revised taxonomy for intrusion-detection systems," *Annals of Telecommunications*, vol. 55, no. 7, pp. 361–378, 2000.

[4] I. Homoliak, M. Barabas, P. Chmelar, M. Drozd, and P. Hanacek, "ASNM: Advanced Security Network Metrics for Attack Vector Description," in *Conference on Security & Management*, 2013, pp. 350–358.

[5] "Tcpdump.org: tcpdump." [Online]. Available: http://www.tcpdump.org/

[6] B. Sangster, T. O'Connor, T. Cook, R. Fanelli, E. Dean, W. J. Adams, C. Morrell, and G. Conti, "Toward Instrumenting Network Warfare Competitions to Generate Labeled Datasets," in *2nd Workshop on Cyber Security Experimentation and Test (CSET)*, 2009.

[7] I. Homoliak, D. Ovšonka, M. Grégr, and P. Hanáček, "NBA of Obfuscated Network Vulnerabilities' Exploitation Hidden into HTTPS Traffic," in *9th International Conference for Internet Technology and Secured Transactions (ICITST)*. IEEE, 2014, pp. 311–318.

[8] A. W. Moore, D. Zuev, and M. Crogan, "Discriminators for Use in Flow-based Classification," Technical report, Intel Research, Cambridge, Tech. Rep., 2005.

[9] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, "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*. ACM, 2011, pp. 29–36.

[10] "KDD Cup 99," 1999. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[11] I. Cisco Systems, "NetFlow," 2019.

[12] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of tor traffic using time based features." in *ICISSP*, 2017, pp. 253–262.

[13] D. Bekerman, B. Shapira, L. Rokach, and A. Bar, "Unknown malware detection using network traffic classification," in *2015 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2015, pp. 134–142.

[14] I. Homoliak, L. Sulak, and P. Hanacek, "Features for behavioral anomaly detection of connectionless network buffer overflow attacks," in *International Workshop on Information Security Applications*. Springer, 2016, pp. 66–78.

[15] R. Kohavi, "A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection," in *14th International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 2, 1995, pp. 1137–1145.

[16] I. Homoliak, "Intrusion Detection in Network Traffic," Dissertation, Faculty of Information Technology, University of Technology Brno, 2016.

[17] M. Barabas, I. Homoliak, M. Kacic, and H. Petr, "Detection of network buffer overflow attacks: A case study," in *2013 47th International Carnahan Conference on Security Technology (ICCST)*. IEEE, 2013, pp. 1–4.

[18] I. Homoliak, "Metrics for Intrusion Detection in Network Traffic," Master's thesis, University of Technology Brno, Faculty of Information Technology, Department of Intelligent Systems, 2011, In Slovak Language.

[19] "Cyber Research Center: Data sets," 2011. [Online]. Available: https://westpoint.edu/centers-and-research/cyber-research-center/data-sets

[20] Cisco, "SNORT," 2019. [Online]. Available: https://www.snort.org/

[21] "CVE-2002-0082: Buffer overflow vulnerability of mod_ssl and Apache-SSL," NIST. [Online]. Available: https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2002-0082

[22] "Rapid7: Remotely exploitable buffer overflow in mod_ssl," 2019. [Online]. Available: https://www.rapid7.com/db/vulnerabilities/HTTP-MODS-0003

[23] "Rapid7: Badblue 2.72b passthru buffer overflow," 2019. [Online]. Available: https://www.rapid7.com/db/modules/exploit/windows/http/badblue_passthru

[24] "CVE-2007-6377: Stack-based buffer overflow vulnerability in BadBlue." NIST. [Online]. Available: https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2007-6377

[25] "CVE-2003-0352: Buffer overflow in DCOM interface for RPC in MS Windows NT 4.0, 2000, XP and Server 2003," NIST. [Online]. Available: https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2003-0352

[26] "Rapid7: MS03-026 Microsoft RPC DCOM interface overflow," 2019. [Online]. Available: https://www.rapid7.com/db/modules/exploit/windows/dcerpc/ms03_026_dcom

[27] "CVE-2003-0201: Buffer overflow in the Samba service." NIST. [Online]. Available: https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2003-0201

[28] "Rapid7: Samba trans2open overflow (Linux x86)," 2019. [Online]. Available: https://www.rapid7.com/db/modules/exploit/linux/samba/trans2open

[29] Open Information Security Foundation, "Suricata IDS," 2019. [Online]. Available: http://suricata-ids.org/

[30] "VirusTotal - Virus, Malware and URL Scanner." [Online]. Available: https://www.virustotal.com/

[31] "Rapid7: Metasploitable – Virtual machine to test Metasploit," 2019. [Online]. Available: https://information.rapid7.com/metasploitable-download.html

[32] "Rapid7: Tomcat application manager login utility," 2019. [Online]. Available: http://www.rapid7.com/db/modules/auxiliary/scanner/http/tomcat_mgr_login

[33] "Rapid7: Apache Tomcat manager application deployer authenticated code execution," 2019. [Online]. Available: http://www.rapid7.com/db/modules/exploit/multi/http/tomcat_mgr_deploy

[34] "Rapid7: MSSQL login utility," 2019. [Online]. Available: https://www.rapid7.com/db/modules/auxiliary/scanner/mssql/mssql_login

[35] "Rapid7: Microsoft SQL server payload execution," 2019. [Online]. Available: http://www.rapid7.com/db/modules/exploit/windows/mssql/mssql_payload

[36] "Rapid7: Samba username map script command execution," 2019. [Online]. Available: http://www.rapid7.com/db/modules/exploit/multi/samba/usermap_script

[37] "Rapid7: Microsoft Server service relative path stack corruption," 2019. [Online]. Available: http://www.rapid7.com/db/modules/exploit/windows/smb/ms08_067_netapi

[38] "Rapid7: PostgreSQL login utility," 2019. [Online]. Available: http://www.rapid7.com/db/modules/auxiliary/scanner/postgres/postgres_login

[39] "Rapid7: PostgreSQL for Linux payload execution," 2019. [Online]. Available: http://www.rapid7.com/db/modules/exploit/linux/postgres/postgres_payload

[40] "Rapid7: DistCC daemon command execution," 2019. [Online]. Available: http://www.rapid7.com/db/modules/exploit/unix/misc/distcc_exec

[41] I. Homoliak, M. Teknos, M. Barabas, and P. Hanacek, "Exploitation of netem utility for non-payload-based obfuscation techniques improving network anomaly detection," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2016, pp. 770–773.

[42] I. Homoliak, M. Teknøs, M. Ochoa, D. Breitenbacher, S. Hosseini, and P. Hanacek, "Improving network intrusion detection classifiers by non-payload-based exploit-independent obfuscations: An adversarial approach," *EAI Endorsed Transactions on Security and Safety*, vol. 5, no. 17, 12 2018.

[43] I. Corona, G. Giacinto, and F. Roli, "Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues," *Information Sciences*, vol. 239, pp. 201–225, 2013.

[44] I. Homoliak, D. Ovšonka, K. Koranda, and P. Hanáček, "Characteristics of Buffer Overflow Attacks Tunneled in HTTP Traffic," ser. International Carnahan Conference on Security Technology. IEEE, 2014, pp. 188–193.

[45] M. Hatada, M. Akiyama, T. Matsuki, and T. Kasama, "Empowering anti-malware research in japan by sharing the mws datasets," *Journal of Information Processing*, vol. 23, no. 5, pp. 579–588, 2015.

[46] "PREDICT dataset: Protected Repository for the Defense of Infrastructure against Cyber Threats," 2019. [Online]. Available: https://www.dhs.gov/publication/dhsstpia-006-protected-repository-defense-infrastructure-against-cyber-threats

[47] "CAIDA: the Cooperative Association for Internet Data Analysis," 2019. [Online]. Available: http://www.caida.org/

[48] "Netresec – publicly available PCAP files," 2019. [Online]. Available: http://www.netresec.com/?page=PcapFiles

[49] "DARPA Intrusion Detection Evaluation," [Online], Cited 2014-01-13. [Online]. Available: http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/

[50] F. Massicotte, F. Gagnon, Y. Labiche, L. Briand, and M. Couture, "Automatic evaluation of intrusion detection systems," in *Proceedings of the 22nd Annual Computer Security Applications Conference, ACSAC'06*. IEEE, 2006, pp. 361–370.

[51] "Contagio malware dump: Collection of PCAP files from malware analysis," 2015. [Online]. Available: http://contagiodump.blogspot.cz/2013/04/collection-of-pcap-files-from-malware.html

[52] M. Akiyama, M. Kamizono, T. Matsuki, and M. Hatada, "Datasets for anti-malware research – mws datasets 2014," IPSJ SIG, Tech. Rep., 2014.

[53] M. Hatada, I. Nakatsuru, and M. Akiyama, "Datasets for anti-malware research – mws 2011 datases," *IPSJ Malware Workshop, MWS'11*, 2011, in Japanese language.

[54] M. Hatada, Y. Nakatsuru, M. Akiyama, and S. Miwa, "Datasets for anti-malware research – mws 2010 datasets," in *IPSJ Malware Workshop, MWS'10*, 2010, pp. 1–5.

[55] M. Hatada, Y. Nakatsuru, M. Terada, and Y. Shinoda, "Dataset for anti-malware research and research achievements shared at the workshop," in *Proceedings of the Computer Security Symposium*, 2009, pp. 1–8.

[56] M. Kamizono, M. Hatada, M. Terada, M. Akiyama, T. Kasama, and J. Murakami, "Datasets for anti-malware research – mws datasets 2013," in *Proceedings of IPSJ Computer Security Symposium*, 2013, pp. 1–8.

[57] M. Akiyama, M. Iwamura, and Y. Kawakoya, "Design and implementation of high interaction client honeypot for drive-by-download attacks," *IEICE Transactions on Communications*, vol. 93, no. 5, pp. 1131–1139, 2010.

[58] M. Akiyama, Y. Takeshi, Y. Kadobayashi, T. Hariu, and S. Yamaguchi, "Client honeypot multiplication with high performance and precise detection," *IEICE Transactions on Information and Systems*, vol. 98, no. 4, pp. 775–787, 2015.

[59] "Dionaea – A malware capturing honeypot," 2014. [Online]. Available: https://www.div0.sg/single-post/dionaea-malware-honeypot

[60] K. Aoki, T. Yagi, M. Iwamura, and M. Itoh, "Controlling malware http communications in dynamic analysis system using search engine," in *Proceedings of the 3rd International Workshop on Cyberspace Safety and Security*. IEEE, 2011, pp. 1–6.

[61] C. Guarnieri, A. Tanasi, J. Bremer, and M. Schloesser, "The cuckoo sandbox," Cited 2016-03-07. [Online]. Available: http://www.cuckoosandbox.org

[62] "Ffr yarai analyzer professional," 2019, in Japanese language. [Online]. Available: www.ffri.jp/products/yarai_analyzer_pro/

[63] D. Inoue, M. Eto, K. Yoshioka, S. Baba, K. Suzuki, J. Nakazato, K. Ohtaka, and K. Nakao, "Nicter: An incident analysis system toward binding network monitoring with malware analysis," in *Workshop on Information Security Threats Data Collection and Sharing, WISTDCS'08*. IEEE, 2008, pp. 58–66.

[64] "The UCSD CAIDA Backscatter dataset," 2019. [Online]. Available: http://www.caida.org/data/passive/backscatter_dataset.xml

[65] "The CAIDA UCSD "DDoS Attack 2007" dataset," 2019. [Online]. Available: http://www.caida.org/data/passive/ddos-20070804_dataset.xml

[66] A. Dainotti, A. King, F. Papale, A. Pescape *et al.*, "Analysis of a/0 stealth scan from a botnet," in *Proceedings of the ACM Internet Measurement Conference, IMC'12*. ACM, 2012, pp. 1–14.

[67] "The CAIDA UCSD network Telescope "Three Days Of Conficker"," 2019. [Online]. Available: http://www.caida.org/data/passive/telescope-3days-conficker_dataset.xml

[68] "The UCSD CAIDA Dataset on the Code-Red Worms," 2019. [Online]. Available: http://www.caida.org/data/passive/codered_worms_dataset.xml

[69] "The CAIDA UCSD dataset on the Witty worm," 2019. [Online]. Available: http://www.caida.org/data/passive/witty_worm_dataset.xml
[70] M. Roesch *et al.*, "Snort: Lightweight intrusion detection for networks," in *LISA*, vol. 99, no. 1, 1999, pp. 229–238.
[71] V. Paxson, "Bro: A system for detecting network intruders in real-time," *Computer Networks*, vol. 31, no. 23, pp. 2435–2463, 1999.
[72] Song, D., "Fragroute." [Online]. Available: http://www.monkey.org/~dugsong/fragroute/
[73] R. F. Puppy, "A look at Whisker's Anti-IDS Tactics," 1999. [Online]. Available: http://www.ussrback.com/docs/papers/IDS/whiskerids.html
[74] A. Sperotto, R. Sadre, F. Van Vliet, and A. Pras, "A labeled data set for flow-based intrusion detection," in *International Workshop on IP Operations and Management*. Springer, 2009, pp. 39–50.
[75] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *computers & security*, vol. 31, no. 3, pp. 357–374, 2012.
[76] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A Detailed Analysis of the KDD Cup 99 Data Set," in *Proceedings of the 2nd IEEE International Conference on Computational Intelligence for Security and Defense Applications*. IEEE Press, 2009, pp. 53–58.
[77] M. Salem, S. Reissmann, and U. Buehler, "Persistent dataset generation using real-time operative framework," in *Proceedings of International Conference on Computing, Networking and Communications, ICNC'14*. IEEE, 2014, pp. 1023–1027.
[78] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, "Cost-based modeling for fraud and intrusion detection: Results from the JAM project," in *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings*, vol. 2. IEEE, 2000, pp. 130–144.
[79] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "Nsl-kdd dataset," 2009. [Online]. Available: https://web.archive.org/web/20150205070216/http://nsl.cs.unb.ca/NSL-KDD/
[80] A. Moore, J. Hall, C. Kreibich, E. Harris, and I. Pratt, "Architecture of a Network Monitor," in *Proceedings of the Passive & Active Measurement Workshop, PAM'03*, 2003.
[81] "Symantec network security 7100 series," 2019. [Online]. Available: http://eval.symantec.com/mktginfo/enterprise/fact_sheets/ent-factsheet_network_security_7100_series_01-2005.en-us.pdf
[82] "ClamAV: Open source antivirus engine for detecting trojans, viruses, malware & other malicious threats," 2019. [Online]. Available: http://www.clamav.net
[83] "Ashula," 2019. [Online]. Available: https://sites.google.com/a/secure-ware.com/securewareproducts/ashula
[84] "Kyoto 2006+ Dataset," 2006. [Online]. Available: http://www.takakura.com/Kyoto_data/
[85] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." in *ICISSP*, 2018, pp. 108–116.

APPENDIX

A. Tuples of TCP Connection and Packet

We detail the TCP connection tuple in Table XXIII. In the table, the superscript at $\mathbb{R}^*$ represents a set of positive real numbers, including zero. Next, we present a detailed description of items from the packet tuple in Table XXIV. The description contains assignment to a particular layer of ISO/OSI stacked model together with supported instances of the protocols – placeholder $*$ represents an arbitrary protocol instance. Also note that in the case of *data* field, superscript in X^* represents an iteration of the set X.

Symbol	Description
$t_s \in \mathbb{R}^+$	Timestamp of the connection's start.
$t_e \in \mathbb{R}^+$	Timestamp of the connection's end.
$p_c \in \{0, \ldots, 2^{16}-1\}$	Port of the client within the TCP connection.
$p_s \in \{0, \ldots, 2^{16}-1\}$	Port of the server within the TCP connection.
$ip_c \in \{0, \ldots, 2^{32}-1\}$	IPv4 address of the client.
$ip_s \in \{0, \ldots, 2^{32}-1\}$	IPv4 address of the server.
P_c	Set of packets sent by client to server.
P_s	Set of packets sent by server to client.

Table XXIII: Symbols of the TCP connection tuple.

Symbol	Description
$t \in \Re_0^+$	Relative time of the packet capture (L1, $*$).
$size \in \aleph$	Size of the whole Ethernet frame including Ethernet header (L1, $*$).
$eth_{src} \in \{0, \ldots, 2^{48}-1\}$	Source MAC address of the frame (L2, Ethernet).
$eth_{dst} \in \{0, \ldots, 2^{48}-1\}$	Destination MAC address of the frame (L2, Ethernet).
$ip_{off} \in \{0, \ldots, 2^{13}-1\}$	Offset field (L3, IPv4).
$ip_{ttl} \in \{0, \ldots, 2^{8}-1\}$	Time to live field (L3, IPv4).
$ip_{p} \in \{0, \ldots, 2^{8}-1\}$	Protocol field (L3, IPv4).
$ip_{sum} \in \{0, \ldots, 2^{16}-1\}$	Checksum of the header (L3, IPv4).
$ip_{src} \in \{0, \ldots, 2^{32}-1\}$	Source IP address of the packet (L3, IPv4).
$ip_{dst} \in \{0, \ldots, 2^{32}-1\}$	Destination IP address of the packet (L3, IPv4).
$ip_{dscp} \in \{0, \ldots, 2^{8}-1\}$	Differentiated services code point field (L3, IPv4).
$tcp_{sport} \in \{0, \ldots, 2^{16}-1\}$	Source port of the packet (L4, TCP).
$tcp_{dport} \in \{0, \ldots, 2^{16}-1\}$	Destination port of the packet (L4, TCP).
$tcp_{sum} \in \{0, \ldots, 2^{16}-1\}$	Checksum of the header (L4, TCP).
$tcp_{seq} \in \{0, \ldots, 2^{32}-1\}$	Sequence number of the packet (L4, TCP).
$tcp_{ack} \in \{0, \ldots, 2^{32}-1\}$	Acknowledgment number of the packet (L4, TCP).
$tcp_{off} \in \{0, \ldots, 2^{8}-1\}$	Offset and reserved fields together (L4, TCP).
$tcp_{flags} \in \{0, \ldots, 2^{8}-1\}$	Control bits (L4, TCP).
$tcp_{win} \in \{0, \ldots, 2^{16}-1\}$	Window field (L4, TCP).
$tcp_{urp} \in \{0, \ldots, 2^{16}-1\}$	Urgent pointer field (L4, TCP).
$data \in \{0, \ldots, 2^{8}-1\}^*$	Payload of the packet (L7, $*$).

Table XXIV: Symbols of the packet tuple.

B. FFS Selected Features

As part of benchmarking ASNM datasets, in this section, we enumerate subsets of the ASNM features selected by the FFS method with Naive Bayes classifier running over 5-fold cross-validation. We present ASNM features selected using: (1) the ASNM-CDX-2009 dataset in Table XXV, (2) the ASNM-TUN dataset in Table XXVI, and (3) the ASNM-NPBO dataset in Table XXVII. Note that the FFS DL set denotes features selected when only legitimate samples and direct attacks were included in the FFS experiment. The FFS DOL set denotes selected features when, in addition to the previous case, obfuscated attacks were included in the FFS experiment.

Feature ID	Description
PolyInd3ordOut[0]	• Approximation of outbound communication (from client to server) by polynomial of 3rd order in the index domain of packet occurrences. The feature represents the 1st coefficient of the approximation,
PolyInd3ordOut[3]	• The same as the previous one, but the feature represents the 4th coefficient of the approximation,
PolyInd8ordOut[6]	• The same as the previous ones, but the feature represents the 7th coefficient of the approximation,
InPkt1s10i[7]	• Lengths of inbound packets occurred in the first second of a connection which are distributed into 10 intervals. The feature represents totaled inbound packet lengths of the 8th interval,
InPkt1s10i[0]	• The same as the previous one, but it represents the 1st interval,
InPkt1s10i[1]	• The same as the previous one, but it represents the 2nd interval,
GaussProds8All[7]	• Normalized products of all packet sizes with 8 Gaussian curves. The feature represents a product of the 8th slice of packets with a Gaussian function which fits the interval of the packets' slice.

Table XXV: ASNM features selected by FFS with the Naive Bayes classifier using ASNM-CDX-2009 dataset.

Feature ID	Description	FFS DOL	FFS DL
SigPktLenIn	• Std. deviation of inbound (server to client) packet sizes.	X	
ConTcpFinCntIn	• The number of TCP FIN flags occurred in inbound traffic.	X	X
ConTcpSynCntIn	• The number of TCP SYN flags occurred in inbound traffic.	X	X
InPktLen32s10i[0]	• Lengths of inbound packets occurred in the first 32 seconds of a connection which are distributed into 10 intervals. The feature represents totaled inbound packet lengths of the 1st interval.	X	
InPktLen1s10i[2]	• The same as the previous one, but computed above the first second of a connection. The feature represents totaled inbound packet lengths of the 3rd interval.	X	
InPktLen8s10i[7]	• The same as the previous one, but computed above the first 8 seconds of a connection. The feature represents totaled inbound packet lengths of the 8th interval.	X	
OutPktLen1s10i[0]	• Lengths of outbound (client to server) packets occurred in the first second of a connection which are distributed into 10 intervals. The feature represents totaled outbound packet lengths of the 1st interval.	X	
FourGonAngleN[9]*	• Fast Fourier Transformation (FFT) of all packet sizes. The feature represents the angle of the 10th coefficient of the FFT in goniometric representation.	X	X
InPktLen8s10i[1]	• Lengths of inbound packets occurred in the first 8 seconds of a connection which are distributed into 10 intervals. The feature represents totaled inbound packet lengths of the 2nd interval.		X
PolyInd8ordOut[5]	• Approximation of outbound packet lengths in index domain by polynomial of 8th order. The feature represents 6th coefficient of the polynomial.		X
PolyInd8ordIn[5]	• Approximation of inbound packet lengths in index domain by polynomial of 8th order. The feature represents 6th coefficient of the polynomial.		X

*Sizes of inbound and outbound packets are represented by negative and positive values, respectively.

Table XXVI: ASNM features selected by FFS with the Naive Bayes classifier using ASNM-TUN dataset.

Feature ID	Description	FFS DOL	FFS DL
SigPktLenOut	• Std. deviation of outbound (client to server) packet sizes.	X	X
MeanPktLenIn	• Mean of packet sizes in inbound traffic of a connection.	X	X
CntOfOldFlows	• The number of mutual connections between client and server, which started up to 5 minutes before start of an analyzed connection.	X	X
CntOfNewFlows	• The number of mutual connections between client and server, which started up to 5 minutes after the end of an analyzed connection.	X	X
ModTCPHdrLen	• Modus of TCP header lengths in all traffic.	X	
UrgCntIn	• The number of TCP URG flags occurred in inbound traffic.	X	
FinCntIn	• The number of TCP FIN flags occurred in inbound traffic.		X
PshCntIn	• The number of TCP PUSH flags occurred in inbound traffic.		X
FourGonModulIn[1]	• Fast Fourier Transformation (FFT) of inbound packet sizes. The feature represents the module of the 2nd coefficient of the FFT in goniometric representation.	X	X
FourGonModulOut[1]	• The same as the previous one, but it represents the module of the 2nd coefficient of the FFT for outbound traffic.		X
FourGonAngleOut[1]	• The same as the previous one, but it represents the angle of the 2nd coefficient of the FFT.	X	
FourGonAngleN[9]	• Fast Fourier Transformation (FFT) of all packet sizes, where inbound and outbound packets are represented by negative and positive values, respectively. The feature represents the angle of the 10th coefficient of the FFT in goniometric representation.	X	X
FourGonAngleN[1]	• The same as the previous one, but it represents the angle of the 2nd coefficient of the FFT.	X	
FourGonModulN[0]	• The same as the previous one, but it represents the module of the 1st coefficient of the FFT.		X
PolyInd13ordOut[13]	• Approximation of outbound communication by polynomial of 13th order in the index domain of packet occurrences. The feature represents the 14th coefficient of the approximation.	X	
PolyInd3ordOut[3]	• The same as the previous one, but it represents the 4th coefficient of the approximation.		X
GaussProds8All[1]	• Normalized products of all packet sizes with 8 Gaussian curves. The feature represents a product of the 2nd slice of packets with a Gaussian function that fits the interval of the packets' slice.	X	
GaussProds8Out[7]	• The same as the previous one, but computed above outbound packets and represents a product of the 8th slice of packets with a Gaussian function that fits the interval of the packets' slice.		X
InPktLen1s10i[5]	• Lengths of inbound packets occurred in the first second of a connection, which are distributed into 10 intervals. The feature represents totaled outbound packet lengths of the 6th interval.	X	
OutPktLen32s10i[3]	• The same as the previous one, but computed above the first 32 seconds of a connection. The feature represents totaled outbound packet lengths of the 4th interval.		X
OutPktLen4s10i[2]	• The same as the previous one, but computed above the first 4 seconds of a connection. The feature represents totaled outbound packet lengths of the 3rd interval.		X

Table XXVII: ASNM features selected by FFS using the Naive Bayes classifier.