arXiv:1805.02684v1 [cs.CR] 7 May 2018

Improving Network Intrusion Detection Classifiers by Non-payload-Based Exploit-Independent Obfuscations: An Adversarial Approach

Ivan Homoliak[1,2], Martin Teknos[1], Martín Ochoa[2], Dominik Breitenbacher[1], Saeid Hosseini[2], and Petr Hanacek[1]

[1] Faculty of Information Technology,
Brno University of Technology,
Bozetechova 1/2, 612 66 Brno Czech Republic

[2] STE-SUTD Cyber Security Laboratory,
Singapore University of Technology and Design, Singapore
8 Somapah Road, Building 2 Level 3, 487372, Singapore

Abstract. Machine-learning based intrusion detection classifiers are able to detect unknown attacks, but at the same time they may be susceptible to evasion by obfuscation techniques. An adversary intruder which possesses a crucial knowledge about a protection system can easily bypass the detection module. The main objective of our work is to improve the performance capabilities of intrusion detection classifiers against such adversaries. To this end, we firstly propose several obfuscation techniques of remote attacks that are based on the modification of various properties of network connections; then we conduct a set of comprehensive experiments to evaluate the effectiveness of intrusion detection classifiers against obfuscated attacks. We instantiate our approach by means of a tool, based on NetEm and Metasploit, which implements our obfuscation operators on any TCP communication. This allows us to generate modified network traffic for machine learning experiments employing features for assessing network statistics and behavior of TCP connections. We perform evaluation on five classifiers: Gaussian Naïve Bayes, Gaussian Naïve Bayes with kernel density estimation, Logistic Regression, Decision Tree, and Support Vector Machines. Our experiments confirm the assumption that it is possible to evade the intrusion detection capability of all classifiers trained without prior knowledge about obfuscated attacks, causing an exacerbation of the TPR ranging from 7.8% to 66.8%. Further, when widening the training knowledge of the classifiers by a subset of obfuscated attacks, we achieve a significant improvement of the TPR by 4.21% – 73.3%, while the FPR is deteriorated only slightly (0.1% – 1.48%). Finally, we test the capability of an obfuscations-aware classifier to detect unknown obfuscated attacks, where we achieve over 90% detection rate on average for most of the obfuscations.

Key words: Classification-Based Intrusion Detection • Adversarial Classification • Non-Payload-Based Obfuscation • Evasion • NetEm • Network Normalizer

1 Introduction

Network intrusion attacks such as exploiting unpatched services continue to be one of the most dangerous threats in the domain of information security [2], [3]. Due to an

increasing sophistication in the techniques used by attackers, misuse-based/knowledge-based [11] 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 approaches in the context of intrusion detection. Anomaly-based approaches are based on building profiles of normal users and trying to detect anomalies deviating from these profiles [11], which might lead to detection of unknown intrusions, but on the other hand it might also generate many false positives. In contrast, the classification approaches take advantage of both misuse-based and anomaly-based models in order to leverage their respective advantages. The classification detection methods firstly build a model based on the labeled samples from both classes – intrusions and the legitimate instances. Secondly, 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.

Assumptions & Scope: Due to efficiency reasons as well as pervasive encryption, we assume in this work a classification-based network intrusion detection system that does not perform deep packet inspection and its model works with TCP connections objects, not single packets. Also, we assume an adversary who knows design details of such a system, but cannot modify its training data. The adversary can only modify the input of the system in a limited way that has to conform the protocol specification of the TCP/IP stack including victim's application. There are several ways how it can be achieved: exploit modification, adding padding at the application layer of exploit code, artificially influencing network or transport layer protocols. If an adversary wants to take advantage of huge database of existing exploits to make their obfuscated mutations and massively exploit targets, adding padding or various changes to exploit code may be time consuming and unsustainable with newly obtained exploits. Therefore, the easiest way for an adversary is to design non-payload-based obfuscation techniques working at network and transport layers, which will mutate instances of known intrusions in an *exploit-independent* way. This will make attacks similar to a legitimate traffic. We follow this idea in our paper and construct exploit-independent modifications of attacks at network and transport layers of TCP/IP. According to the taxonomy of adversarial attacks against IDS [10], our adversarial approach belongs to *evasions of measurement phase* of IDS. Considering *influence*, *security violation*, and *specificity* as dimensions of taxonomy of attacks against learning systems [5], our obfuscated attacks belong to: 1) *exploratory attacks*, which exploit misclassification but do not affect training data, 2) *integrity attacks*, which compromise assets via false negatives, and 3) *indiscriminate attacks*, which compromise wide class of instances.

Despite the fact that non-payload-based evasions and obfuscations of network attacks are not new research topics [14], [22], [23], they are still challenging subjects [7]. There exist several related works considering non-payload-based adversarial evasions of network attacks for payload-based intrusion detection [22], [24], [29]. However, to the best of our knowledge, there are no studies on *non-payload-based intrusion detection* and *obfuscation-based adversarial evasion*.

Problem statement: In this work we address the following questions:

a) Is it possible to evade the detection of a non-payload-based intrusion detection classifier by obfuscation techniques?
b) If so, is it possible to increase the resilience of such a classifier against obfuscated attacks, or even detect unknown ones?

Proposed solution: To address this problem, we define a set of obfuscation operators based on non-payload-based modifications of connection-oriented communications accomplished by *NetEm* utility [15] and *ifconfig* command. We argue that our obfuscation transformations, despite not being exhaustive, cover a wide range of network connection morphing possibilities that can influence the detection performance of a non-payload-based intrusion detection classifier. Subsequently, we propose several experiments to train a classifier using obfuscated attacks as well as obfuscated legitimate connections and compare it against another model of a classifier that is unaware of obfuscated attacks.

Contributions: The main contributions of this paper are as follows:

a) We define non-payload-based obfuscation techniques and their influence on a classification task in an intrusion detection classifier.
b) We implement several obfuscation techniques as part of our obfuscation tool and later conduct a data collection experiment that employs the obfuscation tool.
c) We perform an evaluation of non-payload-based obfuscation techniques using our dataset, and we reveal them as: **1)** successful in evading detection by five classifiers that leverage selected subset of network connection features designed in [16], as well as **2)** successful in an improvement of evasion resistance of the classifiers against unknown obfuscated attacks.
d) Moreover, we elucidate an alternative view on the outcome of our results, which is denoted as training data driven approximation of a network traffic normalizer.
e) The collected dataset is provided to the research community.

2 Background

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 such application data exchanges between client and server can be formulated, considering the TCP/IP stack up to the transport layer, by connection k that is constrained to connection oriented protocol TCP at L4, Internet protocol IP at L3 and Ethernet protocol at L2. The TCP connection k is represented by start and end timestamps, ports of the client and the server, IP addresses of the client and the server, sets of packets sent by the client P_c, and by the server P_s, respectively.

Features 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 k into space of features F: $f(k) \mapsto F$, $F = (F_1, F_2, \ldots, F_n)$ where n represents the number of defined features. Each feature f_i generating feature space F_i is defined as a function that maps the connection k into feature space F_i: $f_i(k) \mapsto F_i$, $i \in \{1, \ldots, n\}$, and each element of codomain F_i is defined as $e = (e_0, \ldots, e_n)$, $n \in \mathbb{N}_0$,

$e_i \in \mathbb{N} \mid e_i \in \mathbb{R} \mid e_i \in \Gamma^+,\ i \in \{0, \ldots, n\}$, $\Gamma = \{a-z, A-Z, 0-9\}$, where Γ^+ denotes positive iteration of the set Γ. In the context of this work, examples of such features are show in Table 8 of Appendix.

Intrusion Detection Classification Task. Referring to [19], let $X = V \times Y$ be the space of labeled samples,[1] 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 $x_i = (v_i \in V,\ y_i \in Y)$. Consider classifier C which maps unlabeled sample $v \in V$ to a label $y \in Y$: $y = C(v)$, and learning algorithm A which maps the given dataset D to a classifier C: $C = A(D_{tr})$. 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}. Now, all extracted features of the connection k can be used as an input of the trained classifier C that predicts the target label:$y_{predict} = A\big(D_{tr}, f(k)\big)$, where $y_{predict} \in \{Intrusion, Legitimate\}$.

3 Proposed Approach

Considering the background from the previous section, now we describe non-payload-based obfuscations that aim at modification of the behavioral characteristics of a remote attack connection, and thus can influence the outcome of the intrusion detection classification task.

Non-payload-based Obfuscations. Consider connection k_a representing a remote attack communication executed without any obfuscation. Then, k_a can be represented by features $f(k_a) \mapsto F^a = (F_1^a, F_2^a, \ldots, F_n^a)$, which are delivered to the previously trained classifier C. Assume that C can correctly predict the target label as an intrusive one, because its knowledge base is derived from training dataset D_{tr} containing intrusive connections having similar (or the same) behavioral characteristics.

Now, consider connection k'_a which represents intrusive communication k_a executed by employment of non-payload-based obfuscations aimed at modification of its network behavioral properties. The obfuscations can modify the P_c and P_s packet sets of the original connection k_a by insertion, removal and transformation of the packets. The modifications of P_c and P_s of the connection k_a can cause alteration of the original features' values F^a to new ones. Thus, features extracted over k'_a are represented by $f(k'_a) \mapsto F^{a'} = (F_1^{a'}, F_2^{a'}, \ldots, F_n^{a'})$ and have different values than features F^a of the connection k_a. Therefore, we conjecture that the likelihood of a correct prediction of k'_a-connection's features $F^{a'}$ by the previously assumed classifier C is lower than in the case of connection k_a. Also, we conjecture that classifier C' trained by learning algorithm A on training dataset D'_{tr}, containing some obfuscated intrusion instances, will be able to correctly predict higher number of unknown obfuscated intrusions than classifier C. These assumptions will be evaluated and analyzed later.

Obfuscation Tool. We designed a tool 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 com-

[1] A sample refers to the vector of the network features extracted over a connection.

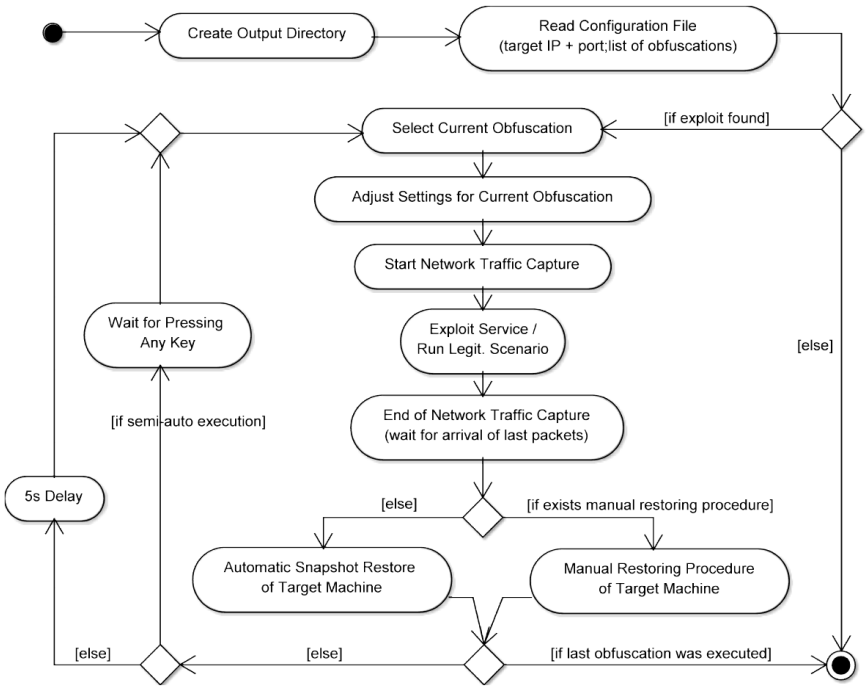

Fig. 1: Behavioral state diagram of the obfuscation tool

munications (non-obfuscated ones) is also supported by the tool as well as capturing network traffic related to a 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 communication, the output contains several network packet traces associated with pertaining obfuscations. The behavioral state diagram of the obfuscation tool is depicted in Figure 1.

Description of Data Collection. We applied the obfuscation tool for a specific set of vulnerable network services and obtained samples of network packet traces related to malicious as well as legitimate communications executed with the employment of particular obfuscations in a virtual network environment. Also, we collected network traffic dump samples of direct attacks for each vulnerable service. These network packet traces were passed to a feature extraction process that first identified all TCP connections and then extracted features per each TCP connection. The collection of these TCP connection-based feature vectors is referred to as dataset, which is analyzed in further machine learning experiments.

Description of Machine Learning Experiments. We performed several classification experiments in order to evaluate the effectiveness of the proposed obfuscation techniques as well as feedback of a classifier having obfuscated data included in its training process. All of our experiments considered two class prediction, discerning between legitimate and malicious TCP connections. Therefore, obfuscated and direct attacks were represented by the same class. We executed the following experiments:

1. For the purpose of finding the best subset of network connection features, we ran the Forward Feature Selection (FFS) method. FFS started to run with an empty set of features and in each iteration executing cross validation, it added a new feature contributing by the best improvement of average recall of all classes. In order to alleviate the possibility of the selection process becoming stuck in local extremes, we allowed acceptance of one iteration without improvement.
2. Considering the selected subset of features, we evaluated evasion resistance of a classifier trained on direct attacks and legitimate traffic only, while testing was performed on the whole dataset including obfuscated attacks.
3. Next, we widened the knowledge of a classifier by adding some obfuscated attacks into the training set and compared its evasion resistance with the previous case.
4. Another experiment tested capability of the classifier to detect unknown obfuscated attacks by customized leave-one-out validation.
5. Finaly, we analyzed the success rate of evasion per particular vulnerable service.

4 Evaluation

The proposed obfuscation techniques had been instantiated as part of the obfuscation tool designed and implemented in the Unix environment. Parametrized instances of these

Table 1: Experimental obfuscation techniques with parameters and IDs

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

techniques are present in Table 1. The selection of particular obfuscation techniques was primarily motivated by the need for achieving divergent behavior of obfuscated network attacks as well as by capabilities of Unix OS. We experimented with various parameters' values with the intention to cover a wide range of divergent behaviors and moreover for the case of attacks preserve their exploitation successful. Although they are not exhaustive, we believe such obfuscation operators are comprehensive enough to demonstrate a wide range of network connection morphing possibilities. The methodology presented in this paper allows for a straightforward extension of the proposed obfuscation set.

Implementation Notes and Network Infrastructure. The obfuscation tool is based on open source tools and is written in the Python and Ruby programming languages. For the purpose of an automatic attack execution an utility from *Metasploit* framework was used. *Tcpdump* tool was chosen to perform network traffic capture between the attacker's machine and the legitimate one. Most obfuscations were carried out by Linux *tc* utility and its extension *NetEm* [15], respectively. NetEm enabled us to add latency of packets, loss of packets, duplication of packets, reordering of packets, and other outgoing traffic characteristics of the selected network interface. The modification of MTU was performed by the Linux utility *ifconfig*.

We established a virtual network environment for vulnerability exploitation, where all virtual machines (VM) 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,[2] and Windows XP with SP 3.

Vulnerable Services. For proof-of-concept purpose, we aimed at selection of vulnerable services with the high severity of their successful exploitation leading to remote shell code execution through an established backdoor communication. Although there exist plethora of publicly available exploit-codes for contemporary vulnerabilities, the situation with corresponding available vulnerable SW is different due to understandable prevention reasons. Therefore, we selected older available high-severity vulnerable services that are outdated but may serve as a demonstration of our approach. The following listing contains an enumeration of vulnerable services involved in our experiments, complemented by brief description of their exploitation. Common Vulnerabilities and Exposures (CVE) IDs with Common Vulnerability Scoring System (CVSS) values are shown in square brackets:

- **Apache Tomcat 5.5**: [CVE-1999-0502: 7.5; CVE-2009-3843: 10.0] – firstly, a dictionary attack was executed in order to obtain access credentials into the application manager instance. Further, the server's application manager was exploited for transmission and execution of malicious code.
- **Microsoft SQL Server 2005**: [CVE-1999-0506: 7.2; CVE-2000-1209: 10.0] – a dictionary attack was employed to obtain access credentials of MSSQL user and then the procedure `xp_cmdshell` enabling the execution of an arbitrary code was exploited.
- **Samba 3.0.20-Debian**: [CVE-2007-2447: 6.0] – vulnerability in Samba service enabled the attacker of arbitrary command execution, which exploited MS-RPC functionality when configuration `username map script` was allowed. There was no need of authentication in this attack.

[2] `https://information.rapid7.com/metasploitable-download.html`

- **Server Service of Windows XP**: [CVE-2008-4250: 10.0] – the service enabled the attacker of arbitrary code execution through crafted RPC request resulting into stack overflow during path canonicalization.
- **PostgreSQL 8.3.8**: [CVE-1999-0502: 7.5; CVE-2007-3280: 9.0] – a dictionary attack was executed in order to obtain access credentials into the PostgreSQL instance. Standard PostgreSQL Linux installation had write access to `/tmp` directory and it could call user defined functions (UDF) that 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.
- **DistCC 2.18.3**: [CVE-2004-2687: 9.3] – vulnerability enabled the attacker remote execution of an arbitrary command through compilation jobs that were executed on the server without any permission check.

An example of a TCP sequence diagram comparing direct and obfuscated attacks on Samba service is depicted in Figure 2, where each arrow contains the timestamp of send event and short description with the size of transmitted data. TCP handshakes and end-shakes are represented by dashed arrows, exploitation of a vulnerability is depicted in red. The new and different transmissions of obfuscated attack against direct one are depicted in blue, while another difference can be seen in values of transmission time. These changes were caused by obfuscation (o) that generated a loss of one or more SYN/ACK packets at the 3-way handshake phase, loss of ACK packet after SMB Request and also addition of delay into delivery of all packets.

Collected Network Traffic Dataset. Our obfuscation tool was leveraged for automatic exploitation of the enumerated vulnerable services using the proposed obfuscations

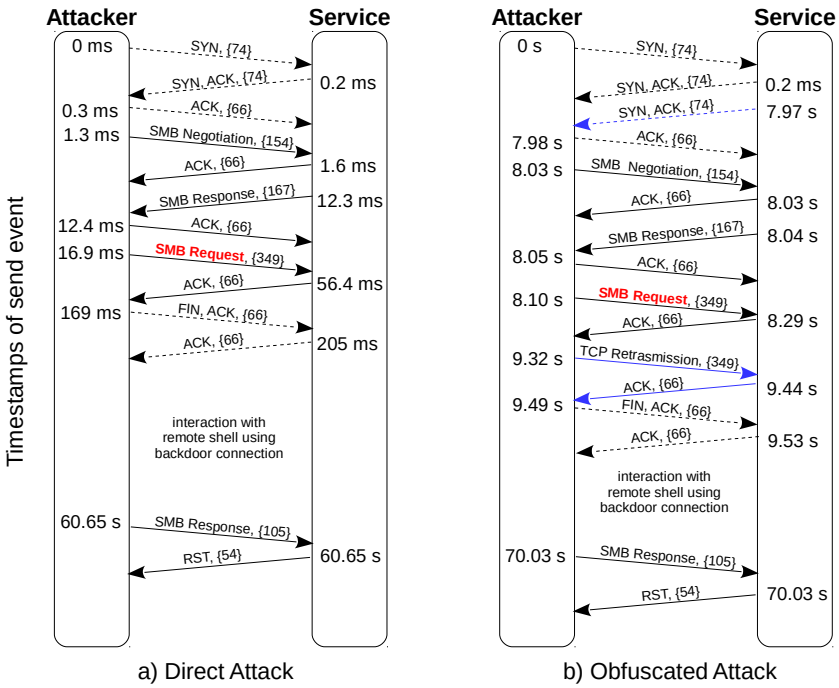

Fig. 2: Comparison of direct and obfuscated attacks on Samba service

and for the capturing of related malicious network traffic, which was further passed to TCP connection-level feature extractor. When an exploitation leading to a remote shell was successful, simulated attackers performed simple activities involving various shell commands (such as listing directories, opening and reading files). The 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 involved surroundings (context) of an analyzed TCP connection for their computation, and thus they captured relations to other TCP connections initiated from/to a service. On the other hand, legitimate network traffic was collected from two sources:

- Common usage of all previously mentioned services was captured in a real campus network and further anonymized. 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 Network Intrusion Detection Systems (NIDS) – Suricata and Snort – through Virus Total API [4]. Herein, the activities performed by real users of campus network are not known.
- 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 [10], 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.[3] Simulation of legitimate traffic was aimed at various *SELECT* and *INSERT* statements when interacting with

[3] 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 a resistance against the overstimulation attacks.

Table 2: Distribution of TCP connection objects in collected dataset

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 Legitimate Traffic	647	n/a	n/a	647
Summary	10805	162	478	11445

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.

The final dataset is summarized in Table 2 and is also available from `http://www.fit.vutbr.cz/~ihomoliak/asnm/ASNM-NPBO.html`.

4.1 Machine Learning Experiments

All machine learning experiments were performed in Rapid Miner Studio [1] using five different classifiers: two with *parametric models* – Gaussian Naïve Bayes and Logistic Regression; and three with *nonparametric models* – Gaussian Naïve Bayes with kernel density estimation, SVM with radial kernel function, and Decision Tree with maximal depth of 10 levels. Note that parametric models make assumptions about the data, which means that they use a finite set of parameters for modeling the data. This makes them simple and fast, but on the other hand they are not flexible in modeling of data that do not contain their assumed distribution. In contrast, non-parametric models have no assumptions about the data, and thus they may use unlimited number of parameters. The advantage of these models is their flexibility, but on the other hand they may overfit the training data. Across all of our experiments, network connection features were instantiated by FFS-selected subset of ASNM features (e.g., Table 8), whose full list is available in Appendix D of [17]. Note that some features of ASNM may lead to overfitting of training data due to laboratory conditions of VMs' setup where attacks were executed. Therefore, such features were removed from the dataset in the preprocessing phase of our experiments and consist of TTL-based features, IP addresses, ports, MAC addresses, occurrence of source/destination host in monitored network. Considering our current dataset's class distribution, we decided to select 5-fold cross validation, which creates big enough folds for binary classification. All cross validation experiments have been adjusted to employ stratified sampling during assembling of folds, which ensured equally balanced class distribution of each fold.

Forward Feature Selection. The experiment consisted of two executions of the FFS per each classifier. In each of executions, we optimized a few important parameters of the classifiers using grid approach. In the cases of both Naïve Bayes classifiers, we enabled Laplace correction in order to prevent models from high influence by zero probabilities of some values, and moreover in the kernel-based version we optimized the bandwidth of kernels. In SVM, we optimized: 1) parameter C that represents trade-off between a soft and hard boundary of the hyperplane and 2) parameter gamma of the Gaussian radial kernel that influences the variance of the Gaussian kernel. The regularization parameter lambda was optimized in the case of Logistic Regression – the parameter controls overfitting of the model at the expense of incorporating the bias. And finally in the case of the decision tree, we used gain ratio as a criterion for selection of attributes for splitting, while we optimized minimal gain required for splitting, which controls the number of splits.

The first execution set of FFS took as input just legitimate traffic and direct attack entries, and represented the case where intrusion detection classifiers were trained without knowledge about obfuscated attacks. We denote the selected features as **FFS DL** (**D**irect + **L**egitimate). The second execution set took as input the whole dataset of network

Table 3: Direct attacks & legitimate traffic cross validation (ordered by F_1)

Classifier	TPR	FPR	$\mathbf{F_1}$	Avg. Recall
N. Bayes (kernels)	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%
Log. Regression	70.99%	0.20%	76.92%	85.40%
N. Bayes	97.53%	8.14%	26.33%	94.70%

Table 4: Prediction of obfuscated/all attacks (ordered by TPR)

(a) Obfuscated attacks

Classifier	TPR	Δ TPR
N. Bayes	81.80%	-13.26%
Log. Regression	63.18%	-7.81%
N. Bayes (kernels)	52.30%	-45.85%
Decision Tree	36.61%	-59.07%
SVM	15.90%	-66.82%

(b) All attacks

Classifier	TPR	Δ TPR
N. Bayes	86.09%	-8.97%
Log. Regression	66.25%	-4.74%
N. Bayes (kernels)	64.38%	-33.77%
Decision Tree	52.03%	-43.65%
SVM	26.25%	-56.47%

traffic – consisting of legitimate traffic, direct attacks as well as obfuscated ones, and thus represented the case where classifiers were aware of obfuscated attacks. Here, we denote the selected features as **FFS DOL** (**D**irect + **O**bfuscated + **L**egitimate). We assume FFS DL features set as less informed (and thus less tuned) than FFS DOL features, therefore when FFS DL features are used we assume that classifiers do not have knowledge about obfuscated attacks, while FFS DOL features are used when we assume the opposite case. As the example of both FFS-selected feature sets see Table 8 of Appendix (columns FFS DOL and FFS DL).

Evasion of Intrusion Detection Classifiers. A 5-fold cross validation was performed using direct attacks with legitimate traffic considering FFS DL features. The performance measures of the classifiers validated by cross validation are shown in Table 3. Then the classifiers trained on all direct attacks and legitimate traffic instances were applied for the prediction of the obfuscated attacks and all attacks, respectively (see Table 4). 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.

Note that in the case of direct attacks and legitimate traffic cross validation, non-parametric classifiers achieved better performance than parametric classifiers, while in the case of obfuscated attacks non-parametric classifiers were more significantly deteriorated in TPR, which was caused by their property of overfitting known data.

Widening the Knowledge of the Classifiers. In order 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, it is justified to use FFS DOL features that consider knowledge about obfuscated attacks for updating not only the model of a classifier but also underlying feature set. Additionally, we show the results with FFS DL features, which consider updating model only. The results of this experiment are shown in Table 5. Comparing against the results from the previous experiment (see FPRs from Table 3

Table 5: Whole dataset cross validation (ordered by F_1)

(a) FFS DL features

Classifier	TPR	FPR	Δ TPR	Δ FPR	F_1	Avg. Recall
N. Bayes (kernels)	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%
Log. Regression	74.69%	0.33%	+20.00%	+0.13%	82.85%	87.18%
Decision Tree	67.34%	0.36%	+15.31%	+0.27%	77.65%	83.49%
N. Bayes	60.31%	1.87%	-34.07%	-6.27%	62.87%	79.22%

(b) FFS DOL features

Classifier	TPR	FPR	Δ TPR	Δ FPR	F_1	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%
N. Bayes (kernels)	98.75%	0.99%	+34.37%	+0.97%	91.66%	98.88%
Log. Regression	97.50%	1.68%	+31.25%	+1.48%	86.37%	97.91%
N. Bayes	98.59%	3.75%	+4.21%	-4.39%	75.30%	97.42%

and TPRs from Table 4b), most of the classifiers were significantly improved in TPR, while FPR was deteriorated only slightly. This confirms the fulfilled assumption that the classifiers trained with knowledge about some obfuscated attacks are able to detect the same or similar obfuscated attacks. The only exception is the Gaussian Naïve Bayes classifier when updating model only, not the underlying feature set (Table 5a). Here is important to note that this classifier makes strong assumptions about the modeled data and when we searched for the optimal feature set with direct and legitimate traffic (FFS DL), it was unable to further optimize FPR, which remained high in contrast to other classifiers. Therefore, when obfuscated attacks were added into the cross validation, the classifier was unable to use the same features and the same strong assumptions about the original data for fitting the different data. However, in the case of updating the feature set (Table 5b), both TPR and FPR of the classifier were improved.

Detection of Unknown Obfuscated Attacks. For the purpose of explicitly testing the classifiers' capability to detect new kinds of obfuscated attacks, we performed customized leave-one-out validation using FFS DOL features, where the classifier was step-by-step trained on all permutations of the whole dataset excluding only obfuscated attack samples created by a single obfuscation technique, or its instance, respectively; while it was validated on the excluded part of the dataset. Table 6 presents ordered ratios of correctly detected unknown obfuscated attacks per obfuscation technique as well as per its instance. Comparing detection performance of unknown obfuscated attacks, either per instance or per obfuscation technique, we concluded that in most of the obfuscations, there were achieved high detection rates that indicate the acceptable resistance of the obfuscations-aware classifiers against unknown obfuscated attacks. The only exceptions are obfuscation techniques that modify MTU. This can be explained by the fact that the majority of the features employed in our experiments is mostly sensitive to packet lengths,

Table 6: Ratios of correctly detected unknown obfuscated attacks

(a) Per instance

Unknown Instance of Obfuscation Technique	The Number of Samples	Ratio of Correctly Detected Samples					
		N. Bayes (Kernels)	N. Bayes	Decision Tree	SVM	Logistic Regression	Average of All Classifiers
(a)	28	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
(b)	22	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
(j)	27	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
(p)	33	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
(c)	30	96.67%	100.00%	100.00%	100.00%	100.00%	99.33%
(i)	27	100.00%	100.00%	100.00%	96.30%	100.00%	99.26%
(e)	26	100.00%	100.00%	96.15%	100.00%	100.00%	99.23%
(g)	26	100.00%	100.00%	100.00%	100.00%	96.15%	99.23%
(d)	30	100.00%	100.00%	96.67%	96.67%	100.00%	98.67%
(h)	30	100.00%	100.00%	96.67%	96.67%	100.00%	98.67%
(q)	35	100.00%	97.14%	97.14%	97.14%	100.00%	98.29%
(m)	27	92.59%	100.00%	92.59%	100.00%	100.00%	97.04%
(o)	28	100.00%	92.86%	92.86%	100.00%	96.43%	96.43%
(f)	28	92.86%	96.43%	92.86%	100.00%	96.43%	95.71%
(l)	27	81.48%	100.00%	92.59%	100.00%	100.00%	94.81%
(k)	27	66.67%	100.00%	100.00%	100.00%	100.00%	93.33%
(n)	27	70.37%	77.78%	48.15%	55.56%	74.07%	65.19%
Average		94.15%	**97.89%**	94.45%	96.61%	**97.83%**	
Std. Dev.		±10.81%	±**5.54%**	±12.30%	±10.68%	±**6.29%**	

(b) Per technique

Unknown Obfuscation Technique	The Number of Samples	Ratio of Correctly Detected Samples					
		N. Bayes (Kernels)	N. Bayes	Decision Tree	SVM	Logistic Regression	Average of All Classifiers
(a, b, c)	80	98.75%	100.00%	100.00%	100.00%	100.00%	99.75%
(i, j)	54	100.00%	100.00%	100.00%	94.44%	100.00%	98.89%
(d)	30	100.00%	100.00%	96.67%	96.67%	100.00%	98.67%
(h)	30	100.00%	100.00%	96.67%	96.67%	100.00%	98.67%
(o, p, q)	96	100.00%	96.88%	96.88%	96.88%	98.96%	97.92%
(e, f, g)	80	97.50%	98.75%	95.00%	100.00%	97.50%	97.75%
(k, l, m, n)	108	75.93%	92.59%	83.33%	61.11%	93.52%	81.30%
Average		96.03%	**98.32%**	95.51%	92.25%	**98.57%**	
Std. Dev.		±8.91%	±**2.78%**	±5.68%	±13.87%	±**2.41%**	

which are influenced by fragmentation-based obfuscations. This phenomenon is more significant in the cases of non-parametric classifiers due to their property of overfitting the training data. In general, parametric classifiers were more successful in detecting unknown obfuscated attacks and their correct predictions were more stable than in non-parametric classifiers. However, in the case of one parametric classifier – Gaussian Naïve Bayes – we also have to take into account its worse FPR in comparison to Logistic Regression (shown in Table 3).

Successful Evasions per Service. This experiment compares and analyzes success rate of evasion per vulnerable service. The results presented here originate from a binary classification experiment in which the classifier was trained without obfuscated attacks and validated on the whole dataset (Table 4) using FFS DL features. The obfuscations are considered successful if they are predicted as legitimate traffic; the situation represents evasion case. Ordered ratios of successfully obfuscated attacks per service are present in Table 7. The minimum achieved ratios of attacks that evaded detection are shown in bold and belong to parametric classifiers, which was already mentioned above. Most successful obfuscated attacks are those exploiting Apache service. From a detailed analysis of this service, we found out that instances of direct attacks had very flat value distribution of many features in comparison to other direct attacks. Examples of such features are the standard deviation of inbound and outbound packet sizes of the connection, and other features dependent on the packets' length variability. Therefore, obfuscated attacks caused more variability of the features that were in many cases similar to legitimate traffic. On the other hand, in the cases of Server, many features of the direct attacks were more divergent across their instances, and thus obfuscations contributed to the divergence only in a low scale. Therefore, most of the obfuscated attacks had similar characteristics like direct ones, which enabled their detection.

Table 7: Successfully obfuscated attacks (evasions) per service

Vulnerable Service	The Number of Samples	Ratio of Successfully Evaded Samples					
		N. Bayes (Kernels)	N. Bayes	Decision Tree	SVM	Logistic Regression	Average of All Classifiers
Apache	163	55.21%	4.91%	55.83%	88.34%	93.87%	59.63%
PostgreSQL	45	66.67%	0.00%	88.89%	100.00%	0.00%	51.11%
MSSQL	103	18.45%	23.30%	71.84%	100.00%	12.62%	45.24%
Samba	44	70.45%	0.00%	50.00%	100.00%	2.27%	44.55%
DistCC	23	39.13%	17.39%	95.65%	47.83%	17.39%	43.48%
Server	100	49.00%	0.00%	54.00%	44.00%	5.00%	30.40%
Average		49.82%	**7.60%**	69.37%	80.03%	**21.86%**	
Std. Dev.		19.17%	10.23%	19.35%	26.84%	35.88%	

5 Discussion

Impact of Obfuscations on Feature Divergence. To assess the impact of proposed obfuscations on the divergence of ASNM features, we compared values of each FFS-selected feature of obfuscated attacks with feature values obtained from direct attacks executed with the same exploit. In other words, we quantified the change that obfuscations

bring by computing the ratio of divergent single-feature obfuscated attacks against the closest single-feature direct attack using the same exploit on the same service. We found out that this ratio (averaged per all FFS DL features) is higher than 55% in the case of all classifiers, which can be viewed as the proposed obfuscations were able to influence the majority of features in obfuscated attacks. The example of this ratio computed per each input feature of Gaussian Naïve Bayes classifier is present in Table 8 of Appendix (the last column). Note that in the case of FFS DOL features, the average of this ratio per-feature is lower (66.33%) than in the case of FFS DL features (72.80%), as FFS DOL was aware of obfuscations and thus selected more obfuscation-resistant features.

Retraining. Although we had demonstrated that our adversarial classification approach to network intrusion detection can detect unknown obfuscated attacks with high performance, it is still possible to design and apply unknown network connection morphing techniques to bypass the detection. In order to keep this performance as high as possible, retraining of the classifier should be performed each time a new form of obfuscation is known to occur. However, such retraining of *generative classifiers* (both Naïve Bayes classifiers) relates to sub-model of the malicious class only, and therefore is faster than retraining monolithic models of *discriminative classifiers* such as SVM, decision tree, and logistic regression, where the whole model incorporating both classes has to be retrained. This favors generative classifiers over discriminative when a frequent update of a model is required. On the other hand, retraining of the legitimate sub-model of generative classifiers should be also performed once in a while in order to ensure that all new manners of using particular services are captured. Next aspect related to fast retraining of classifiers is whether they can be retrained with preserving the feature set and still provide high performance. From this point of view, we have found Naïve Bayes with Gaussian kernels as the most convenient classifier (see results in Table 5a).

High Rate of Attacks. Our dataset has the ratio of malicious to legitimate connections equal to 5.9%, while in practice this ratio is usually several orders of magnitude lesser. Although an arbitrary value of this ratio does not distort the performance of the classifier when correct performance measure is chosen (e.g., F_1*-measure*, average recall of classes), it might impact the accuracy of modeling the legitimate class whose high volume occurred in practice can result in high divergence of data, which might not be captured by models built from our dataset in sufficient manner. Therefore in practice, classifiers would require much more legitimate data than in our dataset.

Normalizers. If we would assume the existence of optimal network normalizer that would be able to completely eliminate the impact of proposed non-payload-based obfuscation techniques, then these obfuscation techniques would be useless. Nevertheless, if such optimal network normalizer would exist, then it would be still prone to state holding and CPU overload attacks [12], [21], [26]. Contrary, if we would not assume network normalizer as part of our system, then non-payload-based obfuscation techniques might be employed as *training data driven approximation of network normalizer* that would not be prone to previously mentioned attacks. The situation can be demonstrated by our binary classification experiments (Section 4.1). Consider intrusion detection classifier validated on direct attacks and legitimate traffic whose average recall is higher than 90% for each classifier (Table 3). Here training and testing data of the classifier were built upon normalized malicious network traffic represented by direct attacks. Then, the model trained

on the direct attacks and legitimate traffic was applied to prediction of the obfuscated attacks. In this case, obfuscated attacks may represent un-normalized malicious network traffic, and thus the classifier achieved worse performance than in the previous case: TPR was significantly decreased while FPR was preserved from the previous step (Table 4a). In order to alleviate negative performance impact of un-normalized malicious network traffic (represented by obfuscated attacks) on our system, we can include obfuscated attacks in the training process of the classifier. This case is interpreted by performance measured contained in Table 5b. There was achieved average recall over 97% for each classifier, primarily thanks to significant improvement of TPR in most of the classifiers. Thus, as an alternative outcome of our work, a network normalizer element may be omitted from classification-based intrusion detection infrastructure and can be approximated by appropriate training data.

6 Related Work

Using taxonomy of attacks against learning systems [5], we categorize our approach to the class of *exploratory indiscriminate* attacks *violating integrity via false negatives*. The same class of attacks was addressed for example in the field of *spam filtering* [6], [20], *malware detection* [6], *payload-based anomaly intrusion detection* [13], [27], *automatic speech recognition* [8], etc. However, to the best of our knowledge, this class of attacks had not been studied in the *non-payload-based intrusion detection* yet, including anomaly-based and classification-based approaches. Further, we aim at related work of evasive adversarial attacks against IDS, and we divide it into payload-based and non-payload-based approaches, plus their combination. Additionally, papers dealing with network traffic normalization are also described.

Payload-based Evasions. The first work dealing with payload-based evasions is described in [23] and presents a tool called Whisker. The author aims at anti-intrusion detection tactics by performing mutations of the HTTP request in a way that a web server is able to understand the request, but intrusion detection systems can be confused. Vigna et al. [28] proposed a framework generating exploit mutations to change the appearance of a malicious payload bypassing detection of NIDS. The proposed framework was evaluated on two well-known signature-based NIDSs – Snort and RealSecure. A similar approach was proposed by Fogla et al. [13] in their polymorphic blending attacks that change the payload of a network worm in order to look normal, and thus effectively evade a byte frequency-based anomaly NIDS. Other approaches use many different techniques for evading detection by changing the payload, e.g., obfuscation techniques such as malware morphism [31] and other attack tactics against IDSs [10]. All of these adversarial approaches are similar to our approach, but in contrast they deal only with evasions of payload-based NIDSs.

Non-payload-based Evasions. Previous methods can evade payload-based NIDS systems primarily by morphing the payload, but do not need to be efficient against non-payload-based network intrusion detectors, which are most sensitive on the attack morphing at the network and transport layers of the TCP/IP stack. Fragroute [22] is a tool that was written to test intrusion detection systems by using simple ruleset language

enabling interception and modification of egress traffic with minimal support for randomized or probabilistic behavior. Fragroute implements three classes of attacks – insertion, evasion, and denial of service. AGENT [24] uses several methods of altering network traffic by packet splitting, duplicate insertions, etc. Watson et al. [29] proposed a method called *Protocol Scrubbing* that represents active mechanisms for transparent removing of network attacks from protocol layers in order to allow passive IDS systems to operate correctly against evasion techniques. Wright et al. [30] proposed thwarting of network traffic classifiers by optimally morphing one class of traffic to look like another class with respect to a given set of features, while they employ padding or splitting the data into smaller parts. This is similar to our approach, but in contrast the authors aim at network traffic classification in general, rather than intrusion detection.

Combinations. Evasions based on modifications at each of the application, transport and network layers of the TCP/IP stack are described in [9] and [18]. Cheng et al. [9] described general evasion techniques and examined the detection performance of signature-based NIDS when performing mutation of known attacks. Juan et al. described framework called *idsprobe* [18] that is intended for evasion-resilience testing of NIDSs. Idsprobe can perform offline as well as live evasion test cases, while it supports payload-based and non-payload-based modifications of network attacks according to predefined transformation profiles. The authors of idsprobe differ from our approach in two points: 1.) they aim at payload-based intrusion detection, 2.) their test cases involve evasions at application layer of ISO/OSI.

Normalizers. In order to answer non-payload-based evasions of NIDS, the concept of network traffic normalizer was introduced by Handley et al. [14]. The authors proposed the implementation of normalizer called *norm*. Norm performs normalizations of ambiguities in the TCP traffic stream that can be seen by NIDS. However, introducing a network normalization brought problems related to platform dependent semantic of network ambiguities interpretation as well as throughput reduction. These problems lead Shankar et al. [25] to introduce the concept and implementation of *Active Mapping*, which eliminates them with minimal runtime cost by building profiles of the network topology including the TCP/IP policies of hosts on the network. A NIDS may then use the host profiles to disambiguate the interpretation of the network traffic on a per-host basis. However, because of the shortcomings of network normalizers, their usage in a network can result in side-effects and can even be prone to various attacks, e.g., state holding, and CPU overload [12], [21], [26].

7 Conclusion

The motivation behind our work is to strengthen non-payload-based intrusion detection classifiers in an *attack-independent* way, assuming an adversary who can massively mutate known exploits to attack huge amount of targets. With this in mind, we executed remote attacks and legitimate communications on selected vulnerable network services while utilizing various non-payload-based obfuscation techniques based on NetEm and MTU modifications with the intention to make behavioral characteristics of the attacks being similar to those of legitimate traffic, and thus cause evasion of our experimental non-payload-based intrusion detection classifiers. The summary of the presented results

revealed non-payload-based obfuscation techniques as partially successful in evading detection by five classifiers (two parametric and three non-parametric), which were trained without prior knowledge about them. On the other hand, if some of the obfuscated attacks were included in the training process of the classifiers, then they were able to detect other unknown obfuscated attacks with high performance. From the practical point of view, we discussed requirements on fast retraining of classifiers, where we identified Naïve Bayes classifier with Gaussian kernels as the most convenient one due to its capability to update the model of a single class independently of another class, and also it does not need to replace the feature set and still can provide a high performance. Note that we do not envision to use our obfuscation-aware non-payload-based classifiers as an independent security solution but as a complementary part of existing solutions, such as misuse-based and anomaly-based intrusion detection systems that perform deep packet inspection.

In our future work, we plan to perform experiments with existing implementations of network normalizers as well as verify the effect of non-payload-based obfuscation techniques on more contemporary vulnerabilities. Another option is to explore impact of proposed obfuscations on communication between bots and C&C server.

References

1. RapidMiner: RapidMiner Studio, `https://rapidminer.com/products/studio/`
2. Top Intrusion Attacks, `https://www.mcafee.com/threat-intelligence/ips/top-attacks.aspx`
3. Top Targeted Vulnerabilities, `https://www.us-cert.gov/ncas/alerts/TA15-119A`
4. VirusTotal - Virus, Malware and URL Scanner, `https://www.virustotal.com/`
5. Barreno, M., Nelson, B., Joseph, A.D., Tygar, J.: The security of machine learning. Machine Learning 81(2), 121–148 (2010)
6. Biggio, B., Corona, I., He, Z.M., Chan, P.P.K., Giacinto, G., Yeung, D.S., Roli, F.: One-and-a-half-class multiple classifier systems for secure learning against evasion attacks at test time. In: Schwenker, F., Roli, F., Kittler, J. (eds.) Multiple Classifier Systems. pp. 168–180. Springer (2015)
7. Boltz, M., Jalava, M., Walsh, J.: New Methods and Combinatorics for Bypassing Intrusion Prevention Technologies. Tech. rep., Stonesoft (2010)
8. Carlini, N., Wagner, D.: Audio adversarial examples: Targeted attacks on speech-to-text. arXiv preprint arXiv:1801.01944 (2018)
9. Cheng, T., Lin, Y., Lai, Y., Lin, P.: Evasion Techniques: Sneaking through Your Intrusion Detection/Prevention Systems. IEEE Communications Surveys and Tutorials, 14(4), 1011–1020 (2012)
10. Corona, I., Giacinto, G., Roli, F.: Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues. Information Sciences 239, 201–225 (2013)
11. Debar, H., Dacier, M., Wespi, A.: A revised taxonomy for intrusion-detection systems. Annals of Telecommunications 55(7), 361–378 (2000)
12. Dharmapurikar, S., Paxson, V.: Robust TCP Stream Reassembly in the Presence of Adversaries. In: 14th USENIX Security Symposium. pp. 65–80 (2005)
13. Fogla, P., Sharif, M.I., Perdisci, R., Kolesnikov, O.M., Lee, W.: Polymorphic Blending Attacks. In: USENIX Security Symposium. pp. 241–256 (2006)
14. Handley, M., Paxson, V., Kreibich, C.: Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics. In: USENIX Security Symposium. pp. 115–131 (2001)
15. Hemminger, S., et al.: Network Emulation with NetEm. In: Australia's 6th National Linux Conference. pp. 18–23. Citeseer (2005)

16. Homoliak, I., Barabas, M., Chmelar, P., Drozd, M., Hanacek, P.: ASNM: Advanced Security Network Metrics for Attack Vector Description. In: Conference on Security & Management. pp. 350–358 (2013)
17. Homoliak, I.: Intrusion Detection in Network Traffic. Dissertation, Faculty of Information Technology, University of Technology Brno (2016)
18. Juan, L., Kreibich, C., Lin, C.H., Paxson, V.: A Tool for Offline and Live Testing of Evasion Resilience in Network Intrusion Detection Systems. In: 5th International Conference Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA), pp. 267–278. Springer (2008)
19. Kohavi, R.: A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection. In: 14th International Joint Conference on Artificial Intelligence (IJCAI). vol. 2, pp. 1137–1145 (1995)
20. Lowd, D., Meek, C.: Adversarial learning. In: SIGKDD International Conference on Knowledge Discovery in Data Mining. pp. 641–647. ACM (2005)
21. Papadogiannakis, A., Polychronakis, M., Markatos, E.P.: Tolerating Overload Attacks Against Packet Capturing Systems. In: USENIX Annual Technical Conference. pp. 197–202 (2012)
22. Ptacek, T.H., Newsham, T.N.: Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection. Tech. rep., DTIC Document (1998)
23. Puppy, R.F.: A look at Whisker's Anti-IDS Tactics (1999), `http://www.ussrback.com/docs/papers/IDS/whiskerids.html`
24. Rubin, S., Jha, S., Miller, B.P.: Automatic Generation and Analysis of NIDS Attacks. In: 20th Annual Computer Security Applications Conference). pp. 28–38. IEEE (2004)
25. Shankar, U., Paxson, V.: Active Mapping: Resisting NIDS Evasion without Altering Traffic. In: Symposium on Security and Privacy (S&P). pp. 44–61. IEEE (2003)
26. Singh, A., Mora dos Santos, A., Nordstrom, O., Lu, C.: Stateless Model for the Prevention of Malicious Communication Channels. International Journal of Computers and Applications 28(3), 285–297 (2006)
27. Tan, K.M., Killourhy, K.S., Maxion, R.A.: Undermining an anomaly-based intrusion detection system using common exploits. In: International Workshop on Recent Advances in Intrusion Detection. pp. 54–73. Springer (2002)
28. Vigna, G., Robertson, W., Balzarotti, D.: Testing Network-based Intrusion Detection Signatures Using Mutant Exploits. In: 11th ACM conference on Computer and Communications Security (CCS). pp. 21–30. ACM (2004)
29. Watson, D., Smart, M., Malan, G.R., Jahanian, F.: Protocol Scrubbing: Network Security through Transparent Flow Modification. IEEE/ACM Transactions on Networking (TON) 12(2), 261–273 (2004)
30. Wright, C.V., Coull, S.E., Monrose, F.: Traffic morphing: An efficient defense against statistical traffic analysis. In: NDSS. vol. 9 (2009)
31. You, I., Yim, K.: Malware Obfuscation Techniques: A Brief Survey. In: 5th International Conference on Broadband and Wireless Computing, Communication and Applications, BWCCA. pp. 297–300. IEEE (2010)

Appendix

Table 8: TCP connection-level features selected by FFS (Naïve Bayes with kernels)

Feature ID	Description	FFS DOL	FFS DL	Ratio of Obfuscated Attacks Having Divergent Values of a Feature in Comparison to Direct Attacks Executed with the Same Exploit
SigPktLenOut	• Std. deviation of outbound (client to server) packet sizes.	X	X	99.44 %
MeanPktLenIn	• Mean of packet sizes in inbound traffic of a connection.	X	X	95.01 %
CntOfOldFlows	• The number of mutual connections between client and server, which started up to 5 minutes before start of an analyzed connection.	X	X	35.37 %
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	62.04 %
ModTCPHdrLen	• Modus of TCP header lengths in all traffic.	X		0.00 %
UrgCntIn	• The number of TCP URG flags occurred in inbound traffic.	X		0.00 %
FinCntIn	• The number of TCP FIN flags occurred in inbound traffic.		X	53.76 %
PshCntIn	• The number of TCP PUSH flags occurred in inbound traffic.		X	54.00 %
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	95.15 %
FourGonModulOut[1]	• The same as the previous one, but it represents the module of the 2nd coefficient of the FFT for outbound traffic.		X	99.50 %
FourGonAngleOut[1]	• The same as the previous one, but it represents the angle of the 2nd coefficient of the FFT.	X		99.51 %
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	99.69 %
FourGonAngleN[1]	• The same as the previous one, but it represents the angle of the 2nd coefficient of the FFT.	X		99.69%
FourGonModulN[0]	• The same as the previous one, but it represents the module of the 1st coefficient of the FFT.		X	98.80 %
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		66.21 %
PolyInd3ordOut[3]	• The same as the previous one, but it represents the 4th coefficient of the approximation.		X	99.50 %
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 to the interval of the packets' slice.	X		95.54 %
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 to the interval of the packets' slice.		X	52.87 %
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		14.69 %
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	38.20 %
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	35.83 %
Average of Divergent Obfuscated Attacks (FFS DL)				72.80 %
Average of Divergent Obfuscated Attacks (FFS DOL)				66.33 %