Evading botnet detectors based on flows and Random Forest with adversarial samples

Giovanni Apruzzese, Michele Colajanni
Department of Engineering "Enzo Ferrari"
University of Modena and Reggio Emilia
Modena, Italy
{giovanni.apruzzese, michele.colajanni}@unimore.it

***Abstract*—Machine learning is increasingly adopted for a wide array of applications, due to its promising results and autonomous capabilities. However, recent research efforts have shown that, especially within the image processing field, these novel techniques are susceptible to adversarial perturbations. In this paper, we present an analysis that highlights and evaluates experimentally the fragility of network intrusion detection systems based on machine learning algorithms against adversarial attacks. In particular, our study involves a random forest classifier that utilizes network flows to distinguish between botnet and benign samples. Our results, derived from experiments performed on a public real dataset of labelled network flows, show that attackers can easily evade such defensive mechanisms by applying slight and targeted modifications to the network activity generated by their controlled bots. These findings pave the way for future techniques that aim to strengthen the performance of machine learning-based network intrusion detection systems.**

***Index Terms*—Adversarial samples, machine learning, random forest, intrusion detection, flow inspection, botnet**

I. Introduction

Machine learning algorithms are being integrated in an ever increasing set of domains [1], such as image processing, speech recognition, social-media marketing and cybersecurity. Indeed, the appreciable results obtained by these methods, alongside their autonomous capabilities, led to their successful adoption for a wide array of applications, supporting or even replacing human operators in their tasks. Despite these promising achievements, several research papers [2]–[5] have shown that these novel techniques are susceptible to adversarial perturbations, which involve the manual or automatic creation of adversarial samples that are specifically designed to thwart the machine learning algorithm and generate misclassifications. This vulnerability is especially relevant for the cybersecurity domain because the implicit cost of undetected attacks is dramatically higher than misclassifications in other fields, as it can lead to the compromise of an entire organization [6]. Moreover, in the cybersecurity field adversaries trying to evade detection are highly skilled and motivated.

In this paper, we explore this problem from the perspective of a network intrusion detection system based on machine learning algorithms. More specifically, we analyze and expose the fragility against adversarial attacks of a botnet detector based on a random forest classifier that inspects network flows[1]. Despite extensive research efforts [7], botnets remain one of the most serious threats to modern enterprises. Several research papers address this issue by devising botnet detectors relying on machine learning [8], involving both supervised (e.g.: [9]) and unsupervised (e.g.: [10]) algorithms; however, the problem still persists [11]. Our study, evaluated on a public dataset of labelled network flows collected from a real large organization, shows that such detectors are easily evaded by introducing tiny and targeted perturbations in the malicious samples. The problem is further aggravated by the fact that similar alterations can be introduced at a very low cost for the adversary: they do not require to change the communication scheme nor the control logic of the botnet; can be easily introduced without the need of deeper compromise and privilege escalation on the infected bots; and the attacker does not need advanced knowledge of the defensive scheme. Indeed, attackers can evade detection with high probability just by introducing slight changes in the network communications of the controlled bots, such as inserting small delays or adding a few bytes to the network packets. Our extensive experimental evaluation highlights a critical issue affecting botnet detectors based on network flows and random forest classifiers, and pave the way to future proposals aiming at strengthening network intrusion detection systems relying on machine learning techniques.

The remainder of the paper is structured as follows. Section II compares our study with related literature.

978-1-5386-7659-2/18/$31.00 ©2018 European Union

[1]Network Flow: https://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html

Section III describes the threat model considered in this work. Section IV presents the details and results of the experimental evaluation. Section V concludes the paper with final remarks and outlines future work.

II. Related Work

This paper highlights and evaluates the fragility of a flow-based botnet detector built on a random forest classifier against adversarial attacks. Therefore, we identify two main areas of related work: *network intrusion detection* focused on botnets and based on machine learning; and *adversarial attacks* against machine learning.

The scientific literature on network intrusion detection has been applying techniques and algorithms borrowed from the machine learning domain for several years [12], and detection schemes involving machine learning have been integrated to some extent even in many recent cyberdefense platforms and commercial cybersecurity products. Many machine learning algorithms have been proposed to specifically address the problem of botnet detection [6], ranging from supervised to unsupervised techniques and even to deep learning. A meaningful and direct comparison among different detection methods is difficult to achieve due to known reasons [6]. However, many research papers [13]–[16] adopt ensemble classifiers based on random forests due to their appreciable results, which have been empirically shown to outperform several other machine learning methods for network intrusion detection tasks [17]. Among these, the authors of [15] devised a network intrusion detection system for identifying network traffic related to communications between infected hosts and their Command and Control infrastructure. This approach is based on the analysis of network flows, rather than full network packets, and applies a classifier based on random forests as suggested by related literature. We remark that inspecting network flows for network intrusion detection is a common practice (e.g.: [10], [18], [19]) because of the following advantages with respect to full packet traces including communication payloads: reduced privacy concerns; smaller storage space and lower computational requirements; the pervasive adoption of end-to-end encryption that prevents collection and analysis of the content of network packets. These reasons motivated us to focus our experiments on a flow-based botnet detector built upon a random forest classifier as a relevant, realistic and representative example of the best practices currently adopted by network security researchers and industry.

The increasing pervasiveness of machine learning led to question its performance in adversarial settings. Recent research papers have mostly addressed this problem from an image processing perspective [2], [4], [5], providing clear use-cases of adversarial attacks. However, literature on evasion attempts against cybersecurity detectors based on machine learning is more focused on the theoretical vulnerabilities of these techniques rather than showing and evaluating the actual effectiveness and consequences of adversarial attacks [20], [21].

Among the few examples of documented adversarial attacks against applications of machine learning to the cybersecurity domain, we highlight: [3], which analyzes the performance in adversarial settings of classifiers based on Simple Vector Machines (SVM) and on Neural Networks for malware detection in PDF files; [22], in which the authors successfully evaded two recent PDF malware classifiers based on either random forests or SVM; and [16], where a random forest classifier devoted to Domain Generation Algorithm (DGA) detection was thwarted with adversarial samples created by a generative adversarial network (GAN). While some of these papers involve evasion attempts towards random forest classifiers, they do not focus on the analysis of network flows, nor on the detection of botnets; moreover they do not consider the effects of adversarial examples against random-forest classifiers. To the best of our knowledge, this is the first work that analyzes the fragility of a flow-based botnet detector based on a random forest classifier against realistic adversarial attacks.

III. Scenario

In this section we describe the realistic scenario and threat model considered in our work. We begin by modeling the target network, and then present the characteristics and strategy of the attacker.

A. Defensive model

The defensive model is represented by a large enterprise network of over a thousand of internal hosts. At its edge, the network presents a border router connected to a network flow collector. Despite the existence of several software products aimed at collecting network flows, each presenting diverse characteristics and allowing to capture different pieces of data, we assume that the generated flows only contain the following essential information:

- source and destination *IP address*;
- flow Start- and End-time (flow *duration*);
- source and destination *ports*;
- *protocol*;
- source and destination *Type of Service (ToS)*;
- source and destination exchanged *bytes*;
- Total *packets* transmitted.

The produced network flows are then inspected by a botnet detector that relies on a random forest classifier to distinguish between legitimate and botnet samples. The classifier is trained to identify the malicious flows produced by specific botnet variants.

B. Attacker model

We assume an attacker that has already established a foothold in the internal network by compromising one or more machines and deploying a bot that communicates with a Command and Control infrastructure. We describe the attacker model accordingly to the approach proposed in [3].

1) Attacker Goal: The attacker aims to evade detection so that they can maintain access to the internal network.

2) Attacker Knowledge: The attacker has partial knowledge on the defenses adopted by the target network. They know that network communications might be monitored by a network intrusion detection system based on supervised machine learning; however, they do not possess knowledge on the algorithm itself (e.g.: they do not know the parameters, the weights or the feature set). We assume that the attacker knows that this detector is trained over a dataset containing malicious flows generated by the very same malware variant deployed on the infected machines. Thus, the attacker is aware that they need to quickly devise some countermeasure to evade the botnet detector.

3) Attacker Capabilities: We assume that the attacker does not have full control on the infected machines. The attacker is limited to issuing commands to the bots through the Command and Control infrastructure, possibly modifying their behavior. Finally, we assume that the attacker cannot interact with the detector in any way.

4) Attacker Strategy: To avoid detection, the attacker leverages their limited knowledge and capabilities to produce a *targeted exploratory integrity attack* [3]. More specifically, they insert some slight modifications in the communication sequences between bots and their Command and Control. The purpose is inducing the botnet detector to misclassify the network flows generated by bot communications due to their different characteristics with respect to the malicious flows contained in the training dataset. These alterations include slight increases of flow *duration*, exchanged *bytes* and exchanged *packets*. We remark that similar modifications can be applied without interfering with the application logic of the bots, which can continue to operate as designed by the attacker.

IV. Evaluation

For our original experiments, we evaluate the performance of a flow-based botnet detector relying on a random forest classifier in adversarial settings. We begin by describing the experimental environment, and then illustrate the results of our experiments.

A. Experimental Setup

We first present the *dataset* used as baseline for our analyses; then, we outline the characteristics of the random forest *classifier*. Finally we explain the methodology used to reproduce a *realistic adversarial setting*.

1) Dataset: Our analyses are based on the *CTU Dataset* [23], a public and labeled dataset collected in a real and large organization and containing both legitimate and botnet traffic. This dataset includes over $20M$ network flows corresponding to over $850M$ packets, and spans over several days. The netflow data is generated with Argus[2] and, besides the basic netflow information provided in Section III-A, contains three additional fields: flow *direction*; *state* of the connection; flow *label*. The labels categorize each flow into four classes: *normal* and *background*, corresponding to benign samples; *botnet* and *CnC-channel*, corresponding to malicious samples. The dataset is split into 13 different traces, each containing the network traffic generated by a specific botnet variant, along with the legitimate traffic produced by the monitored network. The overall number of malware variants included in the dataset is 7: `Neris`, `Rbot`, `Virut`, `Menti`, `Sogou`, `Murlo`, `NSIS.ay`. Some meaningful metrics of the dataset are summarized in Table I.

Table I: CTU Dataset metrics [23].

Trace	Packets	Netflows	Malicious flows	Benign Flows	Malware
1	71,971,482	2,824,637	40,959	2,783,677	`Neris`
2	71,851,300	1,808,122	20,941	1,787,181	`Neris`
3	167,730,395	4,710,638	26,822	4,683,816	`Rbot`
4	62,089,135	1,121,076	1,808	1,119,268	`Rbot`
5	4,481,167	129,832	901	128,931	`Virut`
6	38,764,357	558,919	4,630	554,289	`Menti`
7	7,467,139	114,077	63	114,014	`Sogou`
8	155,207,799	2,954,230	6,126	2,948,104	`Murlo`
9	115,415,321	2,753,884	184,979	2,568,905	`Neris`
10	90,389,782	1,309,791	106,352	1,203,439	`Rbot`
11	6,337,202	107,251	8,164	99,087	`Rbot`
12	13,212,268	325,471	2,168	323,303	`NSIS.ay`
13	50,888,256	1,925,149	39,993	1,885,156	`Virut`

2) Classifier: We evaluate a machine learning classifier based on random forests. The features used by the classifier are based on the feature set used in [9], which exhibits good detection performance on the same dataset used in this paper. We experimentally verified that – for this particular scenario – our classifier produces better

[2]Argus software: http://nsmwiki.org/Argus

results by expanding the feature set with the following information:

- source/destination *IP address type*: it can be either *internal* or *external*;
- connection *state*: included in the dataset;
- flow *direction*: included in the dataset.

We report in Table II the list and type of features considered by our classifier.

Table II: Features used by the random forest classifier.

#	Feature Name	Feature Type
1,2	source/destination IP address type	Boolean
3,4	source/destination port	Numerical
5	flow direction	Boolean
6	connection state	Categorical
7	duration (seconds)	Numerical
8	protocol	Categorical
9,10	source/destination ToS	Numerical
11,12	outgoing/incoming bytes	Numerical
13	total transmitted packets	Numerical
14	total transmitted bytes	Numerical
15	bytes per second	Numerical
16	bytes per packet	Numerical
17	packets per second	Numerical
18	ratio of outgoing/incoming bytes	Numerical

To allow easier reproduction of our experiment, we report the attributes and parameters of the classifier in Table III, where F denotes the number of features provided as input.

Table III: Attributes of the random forest classifier

Attribute name	Value
Number of estimators	10
Quality function	Gini
Features for best split	$\sqrt{F}$
Random state	0

This classifier is trained on the CTU Dataset. More specifically, we create 7 instances of the same classifier, and we train each instance to recognize the malicious network flows generated by a single botnet variant. This design choice is motivated by the fact that machine learning classifiers yield superior results when they focus on a specific problem instead of being used as a "catch-all" solution [6]. Thus, for each instance of the classifier we create a training dataset containing benign and malicious samples with a $20:1$ ratio. In particular, the malicious samples represent ~80% of the botnet flows generated by a single malware variant; benign samples are chosen randomly among the complete set of legitimate network flows. We validate each instance through 10-fold cross validation; the average number of malicious samples used to train each instance of the classifier is included in Table IV.

Table IV: Amount of malicious flows in the training datasets of each instance of the classifier.

Instance	Malware	Malicious flows in Training set
1	`Neris`	$197,611$
2	`Rbot`	$115,209$
3	`Virut`	$32,766$
4	`Menti`	$3,709$
5	`Sogou`	57
6	`Murlo`	$4,952$
7	`NSIS.ay`	$1,712$

To provide a baseline for our experiments, we present in Table V the performance of our classifiers. These results are obtained when each classifier is tested against a dataset containing the malicious flows generated by its specific malware variant, along with legitimate network flows. For each instance of the classifier, we report the most accepted evaluation metrics in the machine learning field: the rate of false positives (FP) and false negatives (FN); the *precision* and detection rate (DR, or *recall*), which are computed as follows:

$$Precision = \frac{TP}{TP + FP} \qquad DR = \frac{TP}{TP + FN}$$

Where TP denotes true positives. For the sake of clarity, we consider a "positive" to be a malicious sample.

Table V: Baseline performance for each instance of the classifier.

Malware	FP rate	FN rate	Precision	DR
`Neris`	0.0014	0.0472	0.9624	0.9528
`Rbot`	< 0.0001	0.0015	0.9999	0.9985
`Virut`	0.0003	0.0525	0.9871	0.9475
`Menti`	0	0.0015	1	0.9967
`Sogou`	0	0.8571	1	0.1429
`Murlo`	0	0.0162	1	0.9838
`NSIS.ay`	< 0.0001	0.1557	0.9872	0.8443

These results denote that our botnet detector obtains appreciable performance under "normal" circumstances, as each classifier exhibits low FP and FN rates, while retaining high precision and recall scores. The only exception is the instance devoted to the `Sogou` botnet, due to the limited sample size available (of only 63 malicious flows) which hinders the training process. Therefore, we exclude this instance from the evaluation.

3) Adversarial samples generation: To generate realistic adversarial samples we produce multiple adversarial datasets by manipulating combinations of up to 4 features of the original malicious flows. The altered features are:

the duration of the flows; the total number of transmitted packets; the number of outgoing(Src) or incoming(Dst) bytes. We remark that an attacker can easily increase flow duration by introducing a small latency, while the number of bytes and packets can be increased just by adding random, junk data. Hence all these modifications can be applied automatically without altering the logic of the bots.

The *groups of altered features* are shown in Table VI. As an example, adversarial samples belonging to group `1a` alter only the flow *duration*, while samples of group `3c` include modifications to the *duration*, *dst_bytes* and *tot_packets* features. For each group, every feature is increased through 9 *increment steps*; the steps are fixed for all the possible combinations. Thus, we produce a total of 135 adversarial datasets samples from the original set of malicious network flows[3]. We report in Table VII the relationship between each step and the corresponding feature increments (*Duration* is measured in seconds). As an example, the adversarial dataset obtained with the `VI` step of group `1b` has the all values of its flow outgoing bytes increased by 128; on the contrary, the adversarial dataset obtained with the `II` step of group `3c` has the all values of its flow duration, incoming bytes and total packets increased by 2. We remark that these perturbations can be achieved easily and resemble a realistic scenario: an excessive increase to the amount of exchanged data may trigger detection by other defensive mechanisms [19], whereas increasing the duration of each flow above 120 seconds may exceed the duration limits of the flow collector (e.g.: [24]).

Table VI: Groups of altered features.

Group	Altered features
1a	Duration (s)
1b	Src_bytes
1c	Dst_bytes
1d	Tot_pkts
2a	Duration, Src_bytes
2b	Duration, Dst_bytes
2c	Duration, Tot_pkts
2e	Src_bytes, Tot_pkts
2d	Src_bytes, Dst_bytes
2f	Dst_bytes, Tot_pkts
3a	Duration, Src_bytes, Dst_bytes
3b	Duration, Src_bytes, Tot_pkts
3c	Duration, Dst_bytes, Tot_pkts
3d	Src_bytes, Dst_bytes, Tot_pkts
4a	Duration, Src_bytes, Dst_bytes, Tot_pkts

B. Results

We evaluate the botnet detector with the crafted adversarial samples. We remark that each instance of the classifier is tested only with the adversarial samples of its corresponding malware variant. Since we are interested in determining how many adversarial samples are classified as negatives, we base our evaluation on the *detection rate* (DR). The results of the evaluation are presented in Tables VIII, where each table reports the detection rates obtained by a specific instance of the classifier. In every table, columns designate the *group of altered features* (described in Table VI), whereas rows represent the *increment steps* (described in Table VII). Detection rates below 50% are denoted with a gray background, while those below 10% are written in bold. The baseline detection rate is provided in the caption of each table.

Table VII: Increment steps of each feature for generating realistic adversarial samples.

Step	Duration	Src_bytes	Dst_bytes	Tot_pkts
I	$+1$	$+1$	$+1$	$+1$
II	$+2$	$+2$	$+2$	$+2$
III	$+5$	$+8$	$+8$	$+5$
IV	$+10$	$+16$	$+16$	$+10$
V	$+15$	$+64$	$+64$	$+15$
VI	$+30$	$+128$	$+128$	$+20$
VII	$+45$	$+256$	$+256$	$+30$
VIII	$+60$	$+512$	$+512$	$+50$
IX	$+120$	$+1024$	$+1024$	$+100$

Consider as an example Table VIIIb, which refers to the detection rate for the `Rbot` botnet variant. The baseline detection rate is 0.9985, however the result of column `1d` and row `I` shows that the attacker only has to increase the duration of network communications by just 1 second to cause a drop of the detection rate from 0.9985 to a clearly unacceptable 0.1922. To reduce the detection rate below 1% an attacker only has to add 128 bytes of useless data and generate additional 20 network packets, as shown by the value 0.0094 in column `2e` and row `VI`.

As expected, we observe that the performance tends to decrease for higher increment steps and for groups of multiple altered features. Intuitively, higher increments of multiple features imply larger modifications to the adversarial samples with respect to the original samples, which negatively impact the performance of the botnet detector. However, we highlight that even small perturbations cause a severe drop to the detection rate: for example, the results in the second row of column `3b` in all Tables VIII show that by simply increasing the *Duration*, *Src_Bytes* and *Tot_Pkts* by 2 units, it is possible to evade all instances with a good confidence (~60% to ~98%).

To provide a more intuitive representation of these results, we report in Figures 1 the results obtained by

[3] 15(combinations of up to 4 features) $* 9$(steps) $= 135$

Table VIII: Detection rates on the adversarial datasets obtained by each instance of the classifier.

Neris	1a	1b	1c	1d	2a	2b	2c	2d	2e	2f	3a	3b	3c	3d	4a
I	0.7368	0.6664	0.6471	0.5927	0.4717	0.4454	0.6504	0.6070	0.5190	0.5090	0.4091	0.4875	0.4941	0.4715	0.4617
II	0.5792	0.6220	0.6396	0.4532	0.2877	0.2612	0.2243	0.5884	0.3466	0.3558	0.2413	0.1433	0.1564	0.3220	0.1436
III	0.4864	0.5036	0.5788	0.3192	0.1733	0.1996	0.1775	0.5106	0.1348	0.1723	0.1674	**0.0788**	**0.0989**	0.1299	**0.0620**
IV	0.4773	0.4495	0.5739	0.2534	0.1556	0.1760	0.1281	0.4728	**0.0971**	0.1253	0.1550	**0.0541**	**0.0417**	**0.0945**	**0.0415**
V	0.4749	0.2251	0.5712	0.2497	**0.0614**	0.1743	0.1209	0.2238	**0.0571**	0.1475	**0.0680**	**0.0394**	**0.0416**	**0.0506**	**0.0385**
VI	0.4695	0.1407	0.5584	0.2447	**0.0332**	0.1767	0.1220	0.1312	**0.0502**	0.1179	**0.0293**	**0.0364**	**0.0404**	**0.0425**	**0.0345**
VII	0.4685	0.1009	0.5173	0.2409	**0.0586**	0.2002	0.1184	0.1579	**0.0381**	**0.0993**	**0.0538**	**0.0333**	**0.0354**	**0.0363**	**0.0325**
VIII	0.4656	**0.0825**	0.4057	0.2346	**0.0481**	0.1631	0.1142	**0.0911**	**0.0332**	**0.0824**	**0.0239**	**0.0726**	**0.0321**	**0.0315**	**0.0309**
IX	0.4650	**0.0611**	0.3265	0.1899	**0.0199**	0.1119	0.1061	**0.0768**	**0.0272**	**0.0726**	**0.0211**	**0.0223**	**0.0261**	**0.0276**	**0.0253**

(a) Detection rates of the `Neris` instance of the classifier (Baseline DR: 0.9528).

Rbot	1a	1b	1c	1d	2a	2b	2c	2d	2e	2f	3a	3b	3c	3d	4a
I	0.9918	0.8456	0.8457	0.1922	0.8208	0.8208	0.1867	0.8418	0.1751	0.1754	0.8197	**0.0203**	**0.0204**	0.1751	**0.0202**
II	0.9917	0.8410	0.8420	0.1899	0.8182	0.8191	0.1846	0.8379	0.1538	0.1546	0.8157	**0.0182**	**0.0185**	**0.0253**	**0.0182**
III	0.9846	0.8080	0.8157	0.1896	0.8033	0.8139	0.1845	0.8079	**0.0188**	**0.0192**	0.8031	**0.0175**	**0.0175**	**0.0178**	**0.0173**
IV	0.9848	0.8082	0.8131	0.1896	0.8028	0.8030	0.1840	0.8059	**0.0175**	**0.0179**	0.8026	**0.0169**	**0.0168**	**0.0172**	**0.0056**
V	0.9852	0.8049	0.8116	0.1892	0.8118	0.7898	0.1831	0.7911	**0.0168**	**0.0166**	0.2391	**0.0169**	**0.0028**	**0.0160**	**0.0046**
VI	0.9853	0.8027	0.7979	0.1892	0.8004	0.2392	0.1835	0.7902	**0.0094**	**0.0141**	0.2386	**0.0054**	**0.0015**	**0.0146**	**0.0036**
VII	0.9850	0.8024	0.7944	0.1892	0.2419	0.2388	0.1834	0.7896	**0.0073**	**0.0139**	0.2377	**0.0050**	**0.0015**	**0.0121**	**0.0014**
VIII	0.9850	0.8023	0.7904	0.1891	0.8003	0.2479	0.1834	0.7852	**0.0025**	**0.0136**	0.2373	**0.0098**	**0.0031**	**0.0049**	**0.0013**
IX	0.9847	0.8022	0.7856	0.1888	0.8003	0.2377	0.1834	0.7856	**0.0017**	**0.0045**	0.2380	**0.0024**	**0.0013**	**0.0047**	**0.0012**

(b) Detection rates of the `Rbot` instance of the classifier (Baseline DR: 0.9985).

Virut	1a	1b	1c	1d	2a	2b	2c	2d	2e	2f	3a	3b	3c	3d	4a
I	0.6412	0.7175	0.7433	0.7326	0.6095	0.6122	0.2400	0.7072	0.6334	0.1488	0.2556	**0.0789**	**0.0747**	0.1376	**0.0740**
II	0.6115	0.7000	0.7275	0.6825	0.3345	0.3321	0.5607	0.1938	**0.0922**	**0.0874**	**0.0700**	**0.0516**	**0.0549**	**0.0740**	**0.0500**
III	0.6048	0.1319	0.1876	0.5815	**0.0600**	**0.0530**	0.5458	0.1215	0.4844	**0.0664**	**0.0541**	**0.0397**	**0.0401**	**0.0528**	**0.0371**
IV	0.6009	0.1097	0.1824	0.5628	**0.0449**	**0.0506**	0.5410	0.1099	**0.0915**	**0.0492**	**0.0463**	**0.0353**	**0.0361**	**0.0476**	**0.0353**
V	0.5898	**0.0696**	0.1616	0.5560	**0.0364**	**0.0430**	0.5398	**0.0692**	**0.0392**	**0.0421**	**0.0343**	**0.0332**	**0.0329**	**0.0367**	**0.0316**
VI	0.5834	**0.0546**	0.1612	0.5519	**0.0297**	**0.0339**	0.5340	**0.0552**	**0.0348**	**0.0376**	**0.0313**	**0.0295**	**0.0278**	**0.0317**	**0.0261**
VII	0.5704	**0.0498**	0.1407	0.5488	**0.0263**	**0.0269**	0.5315	**0.0500**	**0.0297**	**0.0347**	**0.0333**	**0.0229**	**0.0245**	**0.0283**	**0.0222**
VIII	0.7052	**0.0365**	**0.0772**	0.5418	**0.0210**	**0.0248**	0.5285	**0.0376**	**0.0211**	**0.0267**	**0.0204**	**0.0143**	**0.0167**	**0.0235**	**0.0178**
IX	0.6999	**0.0316**	**0.0563**	0.5294	**0.0155**	**0.0156**	0.5199	**0.0304**	**0.0128**	**0.0171**	**0.0161**	**0.0098**	**0.0101**	**0.0173**	**0.0118**

(c) Detection rates of the `Virut` instance of the classifier (Baseline DR: 0.9475).

Menti	1a	1b	1c	1d	2a	2b	2c	2d	2e	2f	3a	3b	3c	3d	4a
I	0.5852	**0.0445**	**0.0434**	0.8903	**0.0358**	**0.0380**	0.4300	**0.0434**	0.8219	0.7872	**0.0380**	0.4235	0.4278	0.4289	**0.0347**
II	0.9870	**0.0445**	**0.0434**	0.7524	**0.0337**	**0.0380**	0.8284	**0.0380**	0.4267	0.3985	**0.0315**	0.4311	0.4365	0.4278	0.4311
III	0.9870	**0.0380**	**0.0380**	0.7524	**0.0054**	**0.0293**	0.7904	**0.0380**	0.3540	0.3985	**0.0054**	0.3985	**0.0597**	0.3540	**0.0597**
IV	0.9870	**0.0380**	**0.0380**	0.7524	**0.0054**	**0.0228**	0.4517	**0.0271**	0.3540	0.3985	**0.0033**	**0.0597**	**0.0000**	0.3540	**0.0000**
V	0.9870	**0.0000**	**0.0380**	0.7524	**0.0000**	**0.0000**	0.4517	**0.0000**	0.3540	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
VI	0.9870	**0.0000**	**0.0098**	0.7524	**0.0000**	**0.0000**	0.4517	**0.0000**	0.3540	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
VII	0.9870	**0.0000**	**0.0000**	0.7524	**0.0000**	**0.0000**	0.4517	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
VIII	0.9870	**0.0000**	**0.0000**	0.7524	**0.0000**	**0.0000**	0.4517	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
IX	0.9870	**0.0000**	**0.0000**	0.7524	**0.0000**	**0.0000**	0.4517	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**

(d) Detection rates of the `Menti` instance of the classifier (Baseline DR: 0.9967).

Murlo	1a	1b	1c	1d	2a	2b	2c	2d	2e	2f	3a	3b	3c	3d	4a
I	0.2247	0.2221	0.2204	0.9694	0.2221	0.2247	0.2119	0.2204	0.2085	0.2094	0.2213	0.1974	0.1966	0.2051	0.1957
II	0.2170	0.2221	0.2204	0.9651	0.2153	0.2153	0.2085	0.2170	**0.0570**	0.2068	0.2077	**0.0604**	0.2060	**0.0502**	**0.0545**
III	0.2034	0.2170	0.2196	0.9302	0.1787	0.1779	**0.0289**	0.2119	**0.0340**	0.1660	0.1864	**0.0221**	**0.0306**	**0.0315**	**0.0332**
IV	**0.0536**	0.2085	0.2136	0.9115	**0.0289**	**0.0264**	**0.0196**	0.2043	**0.0196**	0.1583	**0.0315**	**0.0196**	**0.0196**	**0.0196**	**0.0196**
V	**0.0511**	0.2017	0.2043	0.9106	**0.0187**	**0.0196**	**0.0187**	0.1957	**0.0187**	0.1685	**0.0187**	**0.0187**	**0.0187**	**0.0187**	**0.0187**
VI	**0.0434**	0.1923	0.1966	0.9106	**0.0187**	**0.0187**	**0.0187**	**0.0340**	**0.0187**	0.1779	**0.0187**	**0.0187**	**0.0281**	**0.0187**	**0.0187**
VII	**0.0434**	**0.0357**	**0.0357**	0.9098	**0.0187**	**0.0187**	**0.0179**	**0.0196**	**0.0179**	0.1702	**0.0187**	**0.0179**	**0.0179**	**0.0179**	**0.0179**
VIII	**0.0417**	**0.0187**	**0.0298**	0.9064	**0.0179**	**0.0289**	**0.0128**	**0.0289**	**0.0162**	0.1660	**0.0289**	**0.0153**	**0.0153**	**0.0170**	**0.0170**
IX	**0.0409**	**0.0179**	**0.0315**	0.9047	**0.0128**	**0.0289**	**0.0077**	**0.0281**	**0.0128**	**0.0136**	**0.0213**	**0.0077**	**0.0077**	**0.0136**	**0.0085**

(e) Detection rates of the `Murlo` instance of the classifier (Baseline DR: 0.9838).

NSIS.ay	1a	1b	1c	1d	2a	2b	2c	2d	2e	2f	3a	3b	3c	3d	4a
I	0.8004	0.8026	0.8333	0.5768	0.7785	0.8026	0.6228	0.8114	0.5263	0.5482	0.7873	0.5921	0.6096	0.5329	0.6031
II	0.7610	0.7632	0.8246	0.5307	0.6601	0.7171	0.4298	0.7237	0.4561	0.4934	0.5570	0.3399	0.3750	0.4605	0.3465
III	0.7061	0.5504	0.7456	0.4364	0.4561	0.6382	0.2895	0.5132	0.3377	0.4013	0.4715	0.2259	0.2544	0.3421	0.2346
IV	0.6842	0.5241	0.7149	0.3640	0.3838	0.5417	0.2303	0.4715	0.2632	0.3070	0.3904	0.2237	0.2215	0.2303	0.2281
V	0.6689	0.4342	0.6184	0.3465	0.3575	0.5088	0.2193	0.4539	0.2346	0.2346	0.3333	0.2346	0.2061	0.2039	0.2193
VI	0.6272	0.3662	0.5614	0.3311	0.2873	0.3618	0.1886	0.3947	0.2149	0.2083	0.3136	0.1908	0.1623	0.1732	0.1667
VII	0.6118	0.3289	0.4956	0.3268	0.2675	0.3816	0.1513	0.3004	0.2018	0.1864	0.3092	0.1579	0.1272	0.1535	0.1382
VIII	0.6053	0.3048	0.4232	0.2917	0.2719	0.3443	0.1228	0.2741	0.1776	0.1425	0.2325	0.1294	**0.0833**	**0.0877**	**0.0614**
IX	0.5724	0.2895	0.2873	0.2763	0.2610	0.1974	**0.0811**	0.2478	0.1557	0.1294	0.1886	**0.0570**	**0.0373**	**0.0702**	**0.0351**

(f) Detection rates of the `NSIS.ay` instance of the classifier (Baseline DR: 0.8443).

a single instance of the classifier for three different increment steps. In every figure, histograms represent the detection rates for each group of altered features, and the dotted horizontal line represents the baseline detection rate. For space limitations, we only represent results of the classifier for the `Neris` botnet. This choice is motivated by its higher number of malicious samples with respect to all other bot variants, thus leading to a more representative training dataset. Moreover we only include the `I`, `IV` and `IX` steps. From Figure 1a we notice that even the very small perturbations of the first increment step (combinations of: one second, one byte, one packet) reduce the detection rate of more than 20%, and up to 50% for some feature groups. On the other hand, the greater (but still realistic) perturbations reported in Figure 1b and Figure 1c cause almost all adversarial samples to be classified as benign flows, with detection rates always below 60% and even below 10% in most cases.

These results demonstrate the fragility of random forest classifiers to adversarial examples, and evidence the great problem posed by adversarial attacks against these types of security technologies.

V. Conclusions

Machine learning techniques are being increasingly integrated into cybersecurity-related defensive platforms. However, these novel algorithms are susceptible to adversarial settings, in which attackers avoid detection by introducing tiny modifications in their malicious samples. In this paper, we explore this issue from a network intrusion detection perspective. More specifically, we highlight the fragility of a flow-based botnet detector that relies on a random forest classifier. We perform extensive original experiments by leveraging a public dataset of labeled network flows, which is used as base for the creation of realistic adversarial samples in our evaluation. Our analysis shows that attackers can easily thwart such detection schemes by slightly modifying the network communication patterns of their controlled bots in ways that do not alter their logic and do not limit their activities. As an example, for some malware families increasing the duration of network communications by just 1 second causes the detection rate to drop from over 99% to less than 20%. Moreover in many cases it is possible to easily produce adversarial samples that cause a detection rate below 1%. These results evidence a critical vulnerability of machine learning applied to cybersecurity, and pave the way for future improvements aimed at making these techniques more secure.

References

[1] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, 2015.

[2] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE, 2016.

[3] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2013.

[4] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "Towards the science of security and privacy in machine learning," *arXiv preprint arXiv:1611.03814*, 2016.

[5] J. Su, D. V. Vargas, and S. Kouichi, "One pixel attack for fooling deep neural networks," *arXiv preprint arXiv:1710.08864*, 2017.

[6] G. Apruzzese, M. Colajanni, L. Ferretti, A. Guido, and M. Marchetti, "On the effectiveness of machine and deep learning for cybersecurity," in *International Conference on Cyber Conflicts (CyCon), 2018 NATO*. NATO, 2018.

[7] S. S. Silva, R. M. Silva, R. C. Pinto, and R. M. Salles, "Botnets: A survey," *Computer Networks*, 2013.

[8] J. Gardiner and S. Nagaraja, "On the security of machine learning in malware c&c detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 49, no. 3, p. 59, 2016.

[9] M. Stevanovic and J. M. Pedersen, "An analysis of network traffic classification for botnet detection," in *Cyber Situational Awareness, Data Analytics and Assessment (CyberSA), 2015 International Conference on*. IEEE, 2015.

[10] G. Apruzzese, M. Marchetti, M. Colajanni, G. G. Zoccoli, and A. Guido, "Identifying malicious hosts involved in periodic communications," in *Network Computing and Applications (NCA), 2017 IEEE 16th International Symposium on*. IEEE, 2017.

[11] "Internet Security Threat Report 2018." https://www.symantec.com/security-center/threat-report, visited in Jun. 2018.

[12] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Security and Privacy (SP), 2010 IEEE Symposium on*. IEEE, 2010.

[13] R. Perdisci, I. Corona, and G. Giacinto, "Early detection of malicious flux networks via large-scale passive dns traffic analysis," *IEEE Transactions on Dependable and Secure Computing*, 2012.

[14] M. Antonakakis, R. Perdisci, W. Lee, N. Vasiloglou, and D. Dagon, "Detecting malware domains at the upper dns hierarchy." in *USENIX security symposium*, vol. 11, 2011, pp. 1–16.

[15] M. Stevanovic and J. M. Pedersen, "An efficient flow-based botnet detection using supervised machine learning," in *Computing, Networking and Communications (ICNC), 2014 International Conference on*. IEEE, 2014.

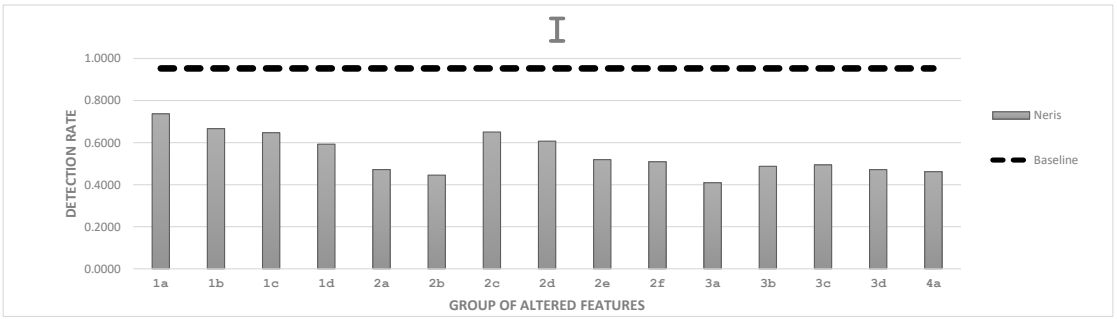
[16] H. S. Anderson, J. Woodbridge, and B. Filar, "Deepdga: Adversarially-tuned domain generation and detection," in *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*. ACM, 2016.

[17] S. Choudhury and A. Bhowal, "Comparative analysis of machine learning algorithms along with classifiers for network intrusion detection," in *Smart technologies and management for computing, communication, controls, energy and materials (ICSTM), 2015 International conference on*. IEEE, 2015.

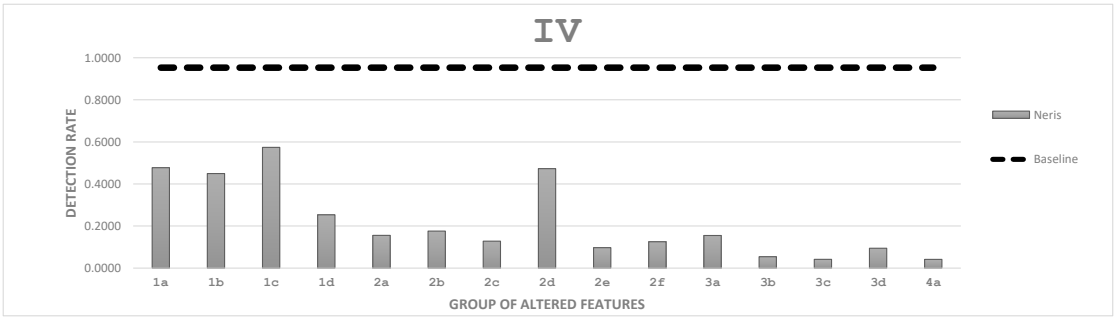
[18] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel, "Disclosure: detecting botnet command and control servers through large-scale netflow analysis," in *ACSAC*, 2012.

[19] F. Pierazzi, G. Apruzzese, M. Colajanni, A. Guido, and M. Marchetti, "Scalable architecture for online prioritisation of cyber threats," in *IEEE CyCon*, 2017.

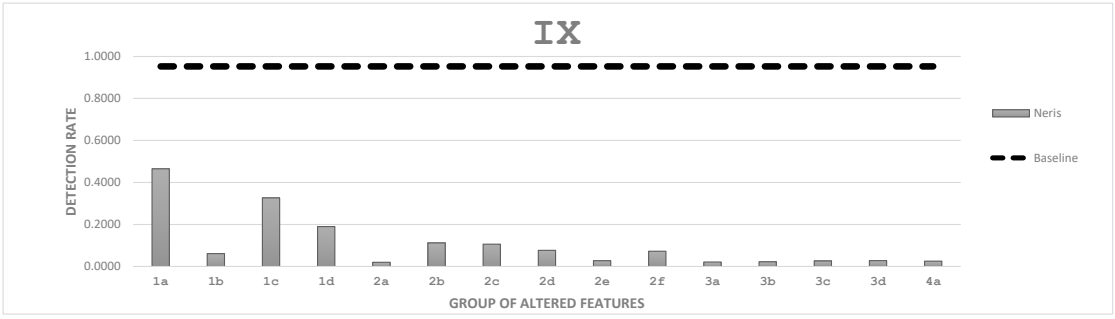
[20] J. Gardiner and S. Nagaraja, "On the security of machine learning in malware c&c detection: A survey," *ACM Computing Surveys (CSUR)*, 2016.



(a) Results on the `I` increment step.



(b) Results on the `IV` increment step.



(c) Results on the `IX` increment step.

Fig. 1: Detection rates of the `Neris` instance on the adversarial datasets obtained with three steps of every group of altered features.

[21] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. Tygar, "Adversarial machine learning," in *Proceedings of the 4th ACM workshop on Security and artificial intelligence*. ACM, 2011.

[22] W. Xu, Y. Qi, and D. Evans, "Automatically evading classifiers," in *Proceedings of the 2016 Network and Distributed Systems Symposium*, 2016.

[23] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *computers & security*, 2014.

[24] G. Apruzzese, F. Pierazzi, M. Colajanni, and M. Marchetti, "Detection and threat prioritization of pivoting attacks in large networks," *IEEE Transactions on Emerging Topics in Computing*, 2017.