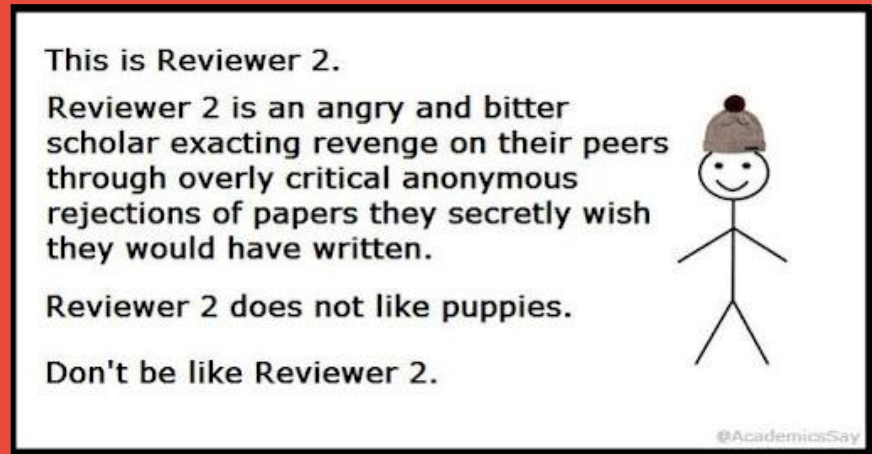


When Adversarial Perturbations meet Concept Drift: An Exploratory Analysis on ML-NIDS

Giovanni Apruzzese*, Aurore Fass§, Fabio Pierazzi‖

*University of Liechtenstein, §CISPA Helmholtz Center of Information Security, ‖King's College London

giovanni.apruzzese@uni.li*, fass@cispa.de §, fabio.pierazzi@kcl.ac.uk‖

Shoutout to our Reviewer 2, who wrote: "From my point of view, if all ML-NIDS papers were as well documented as this one, research in ML-NIDS would gain a great deal!"

Our Institutions

UNIVERSITÄT LIECHTENSTEIN — CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY — KING'S College LONDON

Abstract

We scrutinize the effects of "blind" adversarial perturbations against machine learning (ML)-based network intrusion detection systems (NIDS) affected by concept drift. There may be cases in which a real attacker – unable to access and hence unaware that the ML-NIDS is weakened by concept drift – attempts to evade the ML-NIDS with data perturbations. It is currently unknown if the cumulative effect of such adversarial perturbations and concept drift leads to a greater or lower impact on ML-NIDS. In this "open problem" paper, we seek to investigate this unusual, but realistic, setting—we are not interested in perfect knowledge attackers.

We begin by retrieving a *publicly available* dataset of documented network traces captured in a real, large (>300 hosts) organization. Overall, these traces include several years of raw traffic packets—both benign and malicious. Then, we adversarially manipulate malicious packets with problem-space perturbations, representing a *physically realizable attack*. Finally, we carry out *the first exploratory analysis* focused on comparing the effects of our "adversarial examples" with their respective unperturbed malicious variants in concept-drift scenarios. Through two case studies (a "short-term" one of 8 days; and a "long-term" one of 4 years) encompassing 48 detector variants, we find that, although our perturbations induce a lower detection rate in concept-drift scenarios, some perturbations yield adverse effects for the attacker in intriguing use cases. Overall our study shows that the topics covered are still an open problem which require a re-assessment from future research.

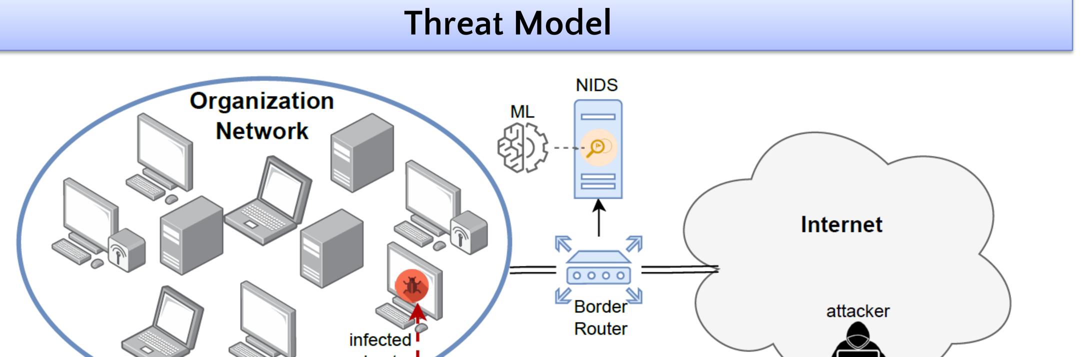
What do we do (and why do we do it)?

Contributions

We shed light on a problem that has never been investigated before in the ML-NIDS context: the combination of realistic (blind, realizable) adversarial perturbations with concept drift. We:

- pinpoint an open-source (and documented) dataset that can be used for concept-drift assessments in ML-NIDS (and which has been overlooked by most research);
- craft problem-space adversarial perturbations by manipulating raw network traffic simulating a simple and feasible attack;
- investigate the extent to which ML-NIDS are statistically significantly affected by realistic adversarial perturbations in concept drift contexts

Threat Model



The attacker is outside the organization's network, cannot access the NIDS (e.g., to query it), and only knows that an ML model analyzes network traffic. The attacker knows that the ML-NIDS is trained on datapoints "similar" to those used in their malicious activities: hence, the attacker is aware that there is a risk of being detected if nothing is done. However, **the attacker does not know if the ML-NIDS is affected (or not) by concept drift** (and neither do the defenders!)

Motivation

A real-attacker's dilemma. It is unknown whether it is beneficial to introduce blind adversarial perturbations in an attempt to bypass an ML-NIDS affected by concept drift. The attacker must make a choice—potentially a detrimental one!

Challenge: Finding the right Data

How to carry out meaningful concept-drift assessments?

Concept drift pertains to supervised ML, and defines the temporal "degradation" of an ML-model's performance due to naturally occurring phenomena. Hence, an assessment requires:

- Data that captures a "relatively long" timeframe
- Data that reflects "naturally occurring" phenomena
- Data that is "unambiguously" labeled.

Unfortunately, meeting such a threefold requirement is tough in the ML-NIDS context.

Data "pitfalls": what can existing datasets for ML-NIDS allow?

We surveyed the landscape of existing publicly available datasets for ML-NIDS, and we found that most do not allow a "meaningful" concept drift assessment. Some examples:

- NSL-KDD is flawed ☺
- CICIDS17 is captured over a timespan of 5 days (!), and the data was created via simulations
- UNSW-NB15 is collected over 15 hours (!)
- Kitsune also has barely 1 day of data.
- UGR'16 is captured over 100 days… but the labeling is not consistent and is not provided with PCAP

Intriguingly, most prior research on concept drift was evaluated on the abovementioned datasets!

*We are **not** pointing the finder, nor seek to invalidate prior work!*

The MCFP data: a practical solution (for our community)

We found that the best publicly-available dataset that meets all our requirements is the "Malware Capture Facility Project" (https://www.stratosphereips.org/datasets-malware), which is an extension of the popular CTU13 dataset.

Takeaway. The MCFP is a suitable solution for assessments of ML-NIDS under concept drift. It is *large*, entails *long* timespans, *ground truth* is provided, and includes various types of benign (from hundreds of hosts) and malicious (entailing recent attacks) traffic in PCAP format – hence useful for "problem-space" attacks.

Perhaps surprisingly, we found only one work ([49]) on "concept drift" that considers MCFP (but it is unclear how the temporal aspect was taken into account).

*We are **not** the creators of MCFP, nor are affiliated with them!*

A "snippet" of the MCFP dataset (which we will use for our analysis)

Table 5: Benign PCAP traces from MCFP, containing "background" and "active" traffic.

(a) Recent benign traces (active).

Trace (Link)	Date	PCAP Size	Flows Total
1	17 Dec 2013	400M	10K
2	17 Dec 2013	800M	10K
3	24 Mar 2015	1M	3K
4	13 Sept 2016	0.6M	40
5	13 Sept 2016	0.6M	50
6	13 Sept 2016	0.5M	40
7	18 Apr 2017	280M	18K
8	19 Apr 2017	200M	7.8K
9	25 Apr 2017	480M	7.8K
10	26 Apr 2017	110M	16K
11	30 Apr 2017	270M	21K
12	30 Apr 2017	420M	39K
13	[illegible]	[illegible]	[illegible]
14	[illegible]	[illegible]	[illegible]
15	[illegible]	[illegible]	[illegible]
16	[illegible]	[illegible]	[illegible]
17	[illegible]	[illegible]	[illegible]
18	[illegible]	[illegible]	[illegible]
19	[illegible]	[illegible]	[illegible]
20	[illegible]	[illegible]	[illegible]
21	[illegible]	[illegible]	[illegible]
22	[illegible]	[illegible]	[illegible]

(b) Traces from CTU13.

Trace (Link)	Date	PCAP Size	Flows Total	Nature
42	10 Aug 2011	6.1G	4M / 35K	Background / Active
43	11 Aug 2011	6.2G	2M / 10K	Background / Active
44	12 Aug 2011	14.5G	5M / 113K	Background / Active
45	15 Aug 2011	5.4G	1.4M / 30K	Background / Active
46	15 Aug 2011	0.4G	150K / 5K	Background / Active
54	16 Aug 2011	4.4G	2.1M / 28K	Background / Active
50	17 Aug 2011	9.8G	2.6M / 35K	Background / Active
52	18 Aug 2011	0.6G	113K / 2.8K	Background / Active

Table 6: Malicious PCAP traces from MCFP used in our assessment (we will perturb these). Note: the most recent traces in MCFP are collected in 2021. We provide an explanation of the method used to choose these traces in our supplementary document [3].

(a) More recent traces (part 1).

m	Trace (Link)	Date	PCAP Size	Flows Total
Wannacry	[illegible]	[illegible]	[illegible]	[illegible]
Dridex	[illegible]	[illegible]	[illegible]	[illegible]

(b) More recent traces (part 2).

m	Trace (Link)	Date	PCAP Size	Flows Total
Artemis	[illegible]	[illegible]	[illegible]	[illegible]
Trickster	[illegible]	[illegible]	[illegible]	[illegible]
Trickbot	[illegible]	[illegible]	[illegible]	[illegible]

(c) Traces from CTU13.

m	Trace (Link)	Date	PCAP Size	Flows Total
Neris	[illegible]	[illegible]	[illegible]	[illegible]
Rbot	[illegible]	[illegible]	[illegible]	[illegible]
Virut	[illegible]	[illegible]	[illegible]	[illegible]

Realistic Perturbations and Case Studies

Crafting "blind" and "realizable" adversarial perturbations

We only manipulate malicious packets (i.e., those that can be detected by the ML-NIDS). For each PCAP trace we consider, we create four "adversarial" traces by proceeding as follows:

- Take all UDP packets and append a small padding of [1-100] (random) bytes
- Take all TCP packets with the PSH flag active, and apply the [1-100] bytes random padding
- Repeat the process again to mitigate bias due to randomness.

We also ensure each packet does not exceed the maximum packet length and recreate the checksum. We do these operations via *scapy* and we release our "packet manipulator" tool.

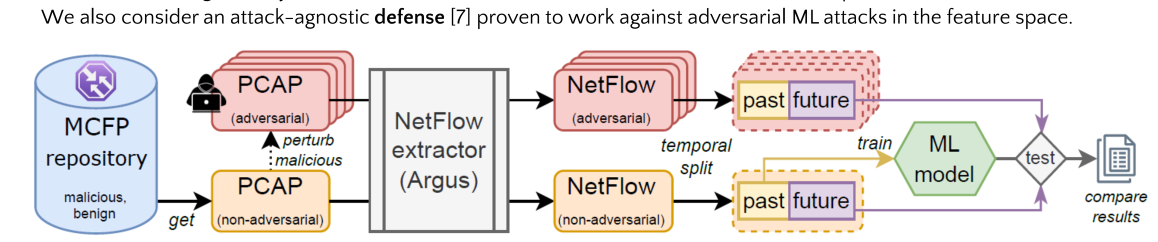
We consider ML-NIDS analysing NetFlows. Hence, the perturbations above will translate in the "feature space". We generate the NetFlows by using Argus, and we then label them by following the official documentation provided by MCFP's creators.

Moreover, in our "adversarial-evaluation" analyses, we will filter to only consider traffic generated from the attacker-controlled machine

*Our perturbations do **not** guarantee evasion! (ours is not a "white-box" attacker!)*

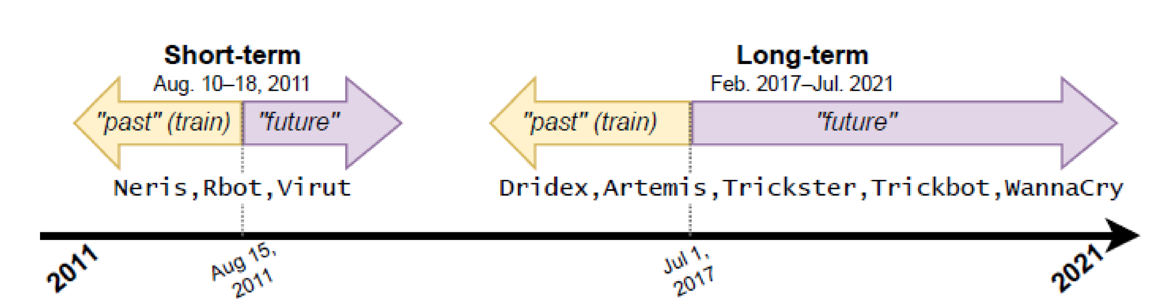
Workflow and ML-NIDS development

We do a "temporal split": the data before a certain data is considered "past data" and is used to develop the ML models used in the ML-NIDS. The data after a certain date is considered "future data" and is used to test our hypotheses during our analysis. Our models are binary classifiers (benign/malicious), and we consider 48 different ML-NIDS, varying the classification algorithm (Random Forest of Histogram Gradient Boosting) and detector architecture (a single binary classifier, or an ensemble of classifiers---each devoted to a specific malware variant). We also consider an attack-agnostic **defense** [7] proven to work against adversarial ML attacks in the feature space.



Summary. We get benign and malicious PCAP from MCFP and manipulate only the malicious traces—yielding problem-space adversarial perturbations. Next, we take all PCAPs and extract (and label) the corresponding NetFlows. Finally, we train ML models (on both benign and malicious "past" data) and test their effectiveness on "future" data (benign, malicious, and adversarial) in a concept-drift setting. We repeat our experiments 50x to ensure statistically sound comparisons.

Overview of our analysis



Case Studies. We consider two case studies, capturing different time periods:

- Short-term Case Study (SCS), of one week (Aug. 10→18, 2011).
  - The cutoff date (past/future data) is set to Aug. 15th 2011
- Long-term Case Study (LCS), of four years (Feb. 2017→Jul 2021).
  - The cutoff date (past/future data) is set to Jul. 1st 2017

For SCS, we consider three malware types (Neris, Rbot, Virut).
For LCS, we consider five (Trickster, Trickbot, Artemis, Wannacry, Dridex).

Preliminary Analysis: are our ML-NIDS good and is there Concept Drift?

Table 1: Pre-deployment results. We assess the performance of our ML-NIDS on the test set from "past" data. We report the average tpr (on malicious samples) and tnr (on benign samples). Cells in boldface are more relevant (they represent architectures denoting "generic" detectors). The defense is denoted with a ⊍.

		SCS: Aug.10th→Aug.18th, 2011					LCS: Feb.2017→Jul.2021						
		Full	Ens	Neris	Rbot	Virut	Full	Ens	Artemis	Dridex	Trickbot	Trickster	Wannacry
Benign	RF	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
	HGB	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Malicious	RF	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
	HGB	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Benign	⊍RF	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
	⊍HGB	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Malicious	⊍RF	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
	⊍HGB	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Table 2: Concept-drift results. We assess the performance of our ML-NIDS on the test set from "future" data. We report the tpr (malicious) and tnr (benign) averaged over 50 trials. Cells in red report cases in which the performance is statistically significantly ($p < 0.05$) worse than on the test set of "past" data (in Table 1). The defense is denoted with a ⊍.

		SCS: Aug.10th→Aug.18th, 2011					LCS: Feb.2017→Jul.2021						
		Full	Ens	Neris	Rbot	Virut	Full	Ens	Artemis	Dridex	Trickbot	Trickster	Wannacry
Benign	RF	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
	HGB	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Malicious	RF	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
	HGB	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Benign	⊍RF	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
	⊍HGB	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Malicious	⊍RF	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
	⊍HGB	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Answer: yes, our ML-NIDS are good; and yes, there is concept drift in both case studies!

There are no perturbations here (yet)

Major Findings and Recommendations

Do adv. perturbations help against ML-NIDS affected by concept drift?

Table 3: Non-adversarial results. We measure the average tpr on the "future" malicious NetFlows. We only consider UDP/TCP NetFlows starting from within the network. The defense is denoted with a ⊍: a ↑/↓ denotes when the defense is statistically significantly ($p < 0.05$) better/worse than the "vanilla".

		SCS: Aug.10th→Aug.18th, 2011					LCS: Feb.2017→Jul.2021						
		Full	**Ens**	Neris	Rbot	Virut	**Full**	**Ens**	Artemis	Dridex	Trickbot	Trickster	Wannacry
UDP	RF	**0.980**	**0.791**	0.941	0.151	0.384	**0.823**	**0.807**	0.000	0.785	0.832	0.000	0.166
	HGB	**0.824**	**0.816**	0.990	0.000	0.039	**0.754**	**0.893**	0.000	0.807	0.823	0.000	0.593
TCP	RF	**0.923**	**0.715**	0.366	0.436	0.773	**0.919**	**0.988**	0.000	0.937	0.922	0.996	0.997
	HGB	**0.943**	**0.861**	0.458	0.615	0.861	**0.983**	**0.993**	0.011	0.926	0.975	0.999	0.998
UDP	⊍RF	**0.892↓**	**0.304↓**	0.031↓	0.117	0.000↓	**0.698↓**	**0.908↑**	0.000	0.893↑	0.854↑	0.040↑	0.586↑
	⊍HGB	**0.921↑**	**0.478↓**	0.025↓	0.103↑	0.113↑	**0.705↓**	**0.909**	0.000	0.864↑	0.852↑	0.060↑	0.583↓
TCP	⊍RF	**0.880↓**	**0.814↑**	0.412↑	0.513↑	0.959↑	**0.979↑**	**0.958↓**	0.128↑	0.741↓	0.940	0.994	0.973↓
	⊍HGB	**0.867↓**	**0.810↓**	0.378↓	0.507↓	0.858	**0.982**	**0.976↓**	0.134↑	0.744↓	0.952↓	0.994↓	0.945↓

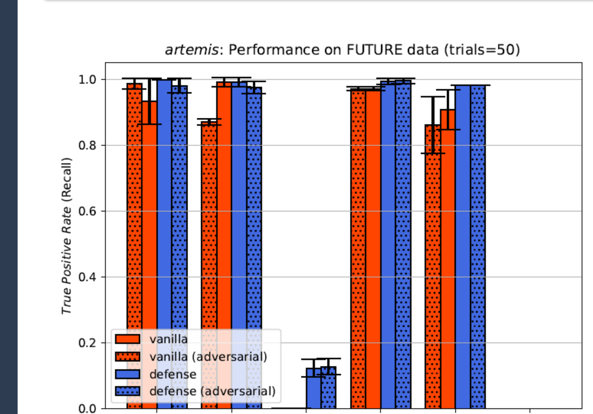
Table 4: Adversarial results. We compute the average tpr on "future" adversarial NetFlows. Cells in red denote cases in which the tpr is statistically significantly *worse* than the baseline in Table 3. A ↑/↓ denotes when the defense ⊍ is statistically significantly ($p < 0.05$) better/worse than the "vanilla".

		SCS: Aug.10th→Aug.18th, 2011					LCS: Feb.2017→Jul.2021						
		Full	**Ens**	Neris	Rbot	Virut	**Full**	**Ens**	Artemis	Dridex	Trickbot	Trickster	Wannacry
UDP	RF	**0.921**	**0.602**	0.470	0.011	0.032	**0.812**	**0.700**	0.000	0.009	0.587	0.000	0.128
	HGB	**0.824**	**0.803**	0.975	0.014	0.059	**0.740**	**0.899**	0.000	0.660	0.833	0.000	0.593
TCP	RF	**0.902**	**0.717**	0.381	0.292	0.769	**0.929**	**0.934**	0.000	0.869	0.778	0.841	0.991
	HGB	**0.934**	**0.861**	0.472	0.470	0.822	**0.967**	**0.980**	0.022	0.910	0.856	0.298	0.992
UDP	⊍RF	**0.873↓**	**0.334↓**	0.052↓	0.050↑	0.000↓	**0.698↓**	**0.908↑**	0.000	0.881↑	0.855↑	0.020↑	0.586↑
	⊍HGB	**0.889↑**	**0.389↓**	0.060↓	0.079↑	0.058	**0.707**	**0.909**	0.000	0.826↑	0.856↑	0.080↑	0.581
TCP	⊍RF	**0.863↓**	**0.831↑**	0.400	0.378↑	0.953↑	**0.922**	**0.963↑**	0.127↑	0.721↓	0.742↓	0.499↓	0.969↓
	⊍HGB	**0.868↓**	**0.834**	0.414↓	0.410↓	0.816	**0.931↓**	**0.974↓**	0.132↑	0.651↓	0.762↓	0.507↑	0.949↓

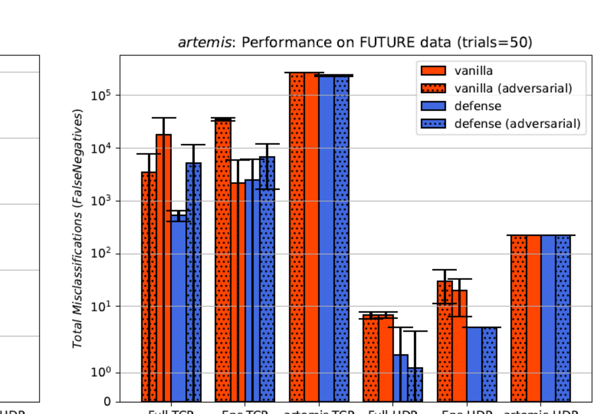
Answer: it depends. We derived three key observations:

- Our "blind" perturbations do cause a (mild, but statistically significant) performance degradation, but they have no effect on the Full-binary classifier on LCS (p=0.4).
- The defense has a smaller benefit than reported in [7].
- In some isolated cases, our perturbations have little effect, and concept drift is enough to defeat the ML-NIDS.

Low-level analysis of "intriguing phenomena"



(a) True-Positive Rate (Recall)



(b) Total Misclassifications (logarithmic scale)

We consider results on *Artemis* because it consistently yielded near-zero *tpr* by the *Artemis*-specific classifier.

- The Full-binary classifier is better against our adversarial perturbations on TCP packets: there are 18k misclassifications for non-adversarial NetFlows, and only 3.5k for adversarial NetFlows.
- The situation is inverted for the ensemble classifier, with 35k evasions due to adversarial NetFlows against only 2k for non-adversarial ones (making our perturbations very effective!).

Both of these claims are validated with a t-test ($p \ll 0.05$).

The UDP (malicious) NetFlows are always misclassified by the Artemis-specific classifier: interestingly, the ensemble (which relies also on this classifier) still retains at least 0.871 *tpr*. This phenomenon is due to the 4 other malware-specific classifiers (i.e., Dridex, Trickster, Trickbot, Wannacry—all of which have never seen any NetFlow from Artemis during training!), of which at least one correctly predicted the ground truth of these NetFlows in the ensemble.

Lessons Learned. We derive three relevant implications for future endeavours.

- The MCFP dataset and our custom resources can be used by future research for realistic concept-drift and/or problem-space assessments of adversarial perturbations in ML-NIDS contexts.
- Overall, blind perturbations (when applied on raw network traffic and in the presence of concept drift) can decrease the *tpr* of state-of-the-art ML-NIDS. However, some perturbations have no effect or can be detrimental to the attacker. We endorse future work to consider game-theory approaches.
- Statistical tests are pivotal to make sound claims. For instance, in some cases our perturbations lowered the *tpr*, but the impact was not statistically significant (i.e., $p > 0.05$). Yet, we are not aware of any prior work on concept drift in the NIDS context whose claims were validated via statistical tests.

We openly release everything!

All our resources (experimental source code, links to data source and explanation of data selection choices, configuration files, complete results, statistical tests, packet-manipulator) are publicly available at: https://github.com/hihey54/aisec24/