When Adversarial Perturbations meet Concept Drift: an Exploratory Analysis on ML-NIDS (Supplementary Document)

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 we covered are still an open problem which require a re-assessment from future research.

Appendix A. Selection of Malicious PCAP traces

Let us explain the procedure we followed to derive the set of 5 malicious classes (i.e., Artemis, Dridex, Trickbot, Trickster, Wannacry) from MCFP [1] considered in our assessment.

A.1. Preliminary Investigation

First, we observe that MCFP contains PCAP captured since 2011 (i.e., it extends CTU13 [2]). Each PCAP[1] is associated with a specific *code* (e.g., "malware" or "normal") and *number*—which is unique and progressive (with some gaps) for each code; as of Feb 14th, 2024, the most recent malicious capture occurred on July 1st, 2021 (having number=410). To resemble a realistic and contemporary scenario, *we consider malicious PCAP starting from 2016*, and specifically from capture #200. Table 7 shows the timespans covered by these malicious traces.

1. Complete list of traces: https://mcfp.felk.cvut.cz/publicDatasets/

TABLE 7: Timespans covered by the malicious PCAP traces in MCFP.

Numbers	Timespan
200→210	Nov.→Dec. 2016
211→220	Dec16→Feb17
221→230	Feb→Mar 2017
231→240	Mar 2017
241→250	Mar→Apr 2017
251→260	May 2017
261→270	Jun 2017
271→280	Jun 2017
281→290	Jul 2017
291→300	Jul 2017
301→310	Jul→Aug 2017
311→320	Aug→Dec 2017
321→330	Jan→Feb 2018
331→340	Feb→Mar 2018
341→350	Mar→May 2018
351→360	May 2018
361→368	May18→Oct18
371	Feb 17
372 onwards	Dec 2018+

A.2. Suitable Candidates

After our preliminary investigation, we proceeded to identify a subset of potential "candidate malicious classes" that could be used for a comprehensive assessment. Given that we are interested in concept drift, the *temporal aspect is crucial for our selection*. Hence, we inspected each malicious PCAP trace to determine *(i)* which malware class it captures, and *(ii)* when it was collected. Indeed, we must categorically exclude those classes for which there is a single capture, since they do not enable fair analyses. We report below the list of classes for which more than one PCAP exists in MCFP (alongside the specific number).

- Dridex.A: 218, 228, 248, 249, 251, 257, 259, 260, 263, 322, 326, 346
- TrickBot: 238, 239, 240, 241, 242, 243, 244, 247, 261.1, 261.2, 261.3, 261.4, 265, 266, 267, 273, 324, 325, 327.1, 327.2, 405
- WannaCry: 252, 253, 254, 256, 258, 270, 283, 284, 285, 286, 287, 290, 291, 292, 293, 294, 295, 296, 297
- Artemis: 275, 305, 306, 311, 316, 374
- Trickster: 277, 302, 309, 323
- Trojan.Yakes: 203, 310
- Pony: 223, 280
- Trojan.Wisdomeyes: 206, 210, 215.1, 215.2, 219.3
- OpenCandy: 208.1, 213
- Trojan.Locky: 214, 221, 222, 236
- Bladabindi: 230.1, 230.2
- TrojanSpyBanker: 235, 245
- Emotet: 264.1, 264.2, 268, 269, 271, 272.1, 272.2, 276.1, 276.2, 279
- TrojanStrictor: 281, 282
- NotPetya: 288, 289, 298, 299
- CCleaner Trojan: 320.1, 320.2

- HtBot: 348, 364, 369, 372, 373
- Sality: 319, 368.2, 368.3
- Simda: 353, 355
- CoinMiner: 329, 338, 342, 347, 351, 352, 367
- Tinba: 225, 233
- Nettool.Netcut: 211.1, 211.2
- Kovter.B: 219.1, 219.2

In contrast, the following classes have only one PCAP (trace # in parentheses) in MCFP, and are hence excluded: Dr.Autoit (200) BundleApp (201) PUP.Adware (202) Toolbar.Google (204) Trojan.MSDILInjector (207) PUA.Adtoolbar (209) Trojan.Agent (216) Trojan.BIKF (217) Worm.Netsky (226) Trojandownloader (227) Trojan.Dynamer (229) W32CoreBot (231) Win32/Taobao.PUA (232) Dryeza (234) TaoBao (237) Tagarep (255) Sennoma (262) Razy (274) TrojanDownloader (341) Trojan.Dynamer (371) Sathurbot (303) Trojan.Snojan (308) Zbot (312) Ursnif (313) Upatre (314) Graftor (315) MagicHound (318) AutoIt (328) WebCompanion (339) Ramnit (343) Cobalt (345) AdwareAdload (349) Mansabo (350).

A.3. Final Selection

Next, we further inspect the capture date of each malicious class, scrutinizing "how close" these PCAP traces are to each other. We found that most of these (e.g., Emotet) are captured within the same day/week, so we exclude these from our analysis. We report in Table 8 the shortlist of our remaining "candidate" classes.

TABLE 8: List of "candidate" malicious classes.

Malware	Traces	FirstSeen	LastSeen
Trickbot	21	Mar 2017	Jul 2021
WannaCry	17	May 2017	Jul 2017
Dridex	13	Feb 2017	Apr 2018
Artemis	5	Jun 2017	Aug 2017
Trickster	3	Jun 2017	Jan 2018
Pony	2	Feb 2017	Jun 2017
Yakes	2	Nov 2016	Sept 2017
HTbot	5	Mar 2018	Dec 2018
CoinMiner	7	Feb 2018	Oct 2018
Locky	5	Sept 2016	Mar 2017
WisdomEyes	5	Dec 2016	Mar 2017

The final selection stems from our **necessity to craft *realistic* adversarial perturbations in the "problem space"**: we need to preserve domain constraints when manipulating the PCAP traces, hence some packets cannot be manipulated without risking to create "unrealizable" NetFlows. As explained in our paper (in §4), we consider manipulations of TCP and UDP packets, which will be reflected in changes to UDP and TCP NetFlows. To provide a comprehensive analysis, we must hence ensure that our selected malicious classes have a *large-enough number of TCP and UDP NetFlows* for both the training/-validation (because the ML model should exhibit a good performance to justify its deployment) and inference (to comprehensively assess the impact of concept drift and adversarial perturbations) phases. Therefore, we take the PCAP traces of our "candidates", generate the corresponding NetFlows (via Argus [3]), and analyse how many UDP and TCP NetFlows are included in each trace.

We found that only five of these classes (i.e., Artemis, Dridex, Trickbot, Trickster, Wannacry) have a sufficiently high number for our evaluation—motivating our selection. We report in Table 9 (which is an extension of Table 6 in the main paper) the actual number of UDP and TCP NetFlows generated by processing each PCAP trace of our considered malicious classes: these NetFlows will be used as the basis to craft our adversarial examples. We stress that, in our assessment, the training/test phase of our ML models will consider *all* NetFlows (including, e.g., ICMP ones): the UDP and TCP NetFlows are merely the ones used for our adversarial evaluation.

TABLE 9: Low-level details of our chosen malicious classes.

Malware	Trace (Link)	**Date**	PCAP Size	Flows Total	Adv. Flows (udp — tcp)
Artemis	1	24 Jun 2017	37	23K	0 — 14K
	2	1 Aug 2017	336	30K	100 — 30K
	3	14 Aug 2017	772	226K	58 — 221K
	4	16 Aug 2017	153	11K	39 — 11K
	5	16 Aug 2017	146	10K	26 — 10K
Dridex	1	13 Feb 2017	79	102	4 — 10
	2	27 Feb 2017	57	2.5K	4 — 1.2K
	3	11 Apr 2017	31	51K	4 — 14K
	4	18 Apr 2017	66	30K	2 — 14K
	5	18 Apr 2017	47	35K	11 — 11K
	6	15 May 2017	7.4	43K	2 — 10K
	7	15 May 2017	33	48K	0 — 13K
	8	16 May 2017	52	63K	3 — 13K
	9	24 Jun 2017	16	11K	6 — 4K
	10	29 Jan 2018	310	73K	2K — 14K
	11	30 Jan 2018	193	37K	749 — 8K
	12	03 Apr 2018	223	52K	16 — 12K
Trickbot	1	29 Mar 2017	83	40K	43 — 15K
	2	30 Mar 2017	90	41K	48 — 15K
	3	30 Mar 2017	90	41K	78 — 15K
	4	30 Mar 2017	81	38K	43 — 15K
	5	12 Apr 2017	288	160K	5 — 114K
	6	12 Apr 2017	115	53K	85 — 24K
	7	17 Apr 2017	142	103K	4 — 83K
	8	8 May 2017	214	127K	2 — 106K
	9	15 May 2017	204	79K	174 — 45K
	10	7 Jun 2017	211	124K	0 — 104K
	11	15 Jun 2017	228	141K	6 — 101K
	12	24 Jun 2017	77	31K	10 — 25K
	13	24 Jun 2017	76	33K	2 — 24K
	14	24 Jun 2017	78	31K	0 — 25K
	15	24 Jun 2017	44	27K	0 — 16K
	16	30 Jan 2018	33	13K	6 — 11K
	17	30 Jan 2018	212	62K	37 — 42K
	18	2 Feb 2018	197	59K	18 — 39K
	19	27 Mar 2018	410	122K	6 — 56K
	20	30 Jul 2021	0.1	61	11 — 27
Trickster	1	24 Jun 2017	52	24K	2 — 16K
	2	3 Aug 2017	6.4	2K	2 — 2K
	3	29 Jan 2018	252	63K	22K — 0
WannaCry	1	14 May 2017	0.5	5K	2 — 5K
	2	14 May 2017	11	15K	6 — 15K
	3	15 May 2017	3.6	171	2 — 43
	4	15 May 2017	13	32K	4 — 30K
	5	24 Jun 2017	444	9K	2 — 2K
	6	11 Jul 2017	1.6	14K	0 — 14K
	7	11 Jul 2017	7.6	14K	2 — 43
	8	11 Jul 2017	7.3	13K	12 — 13K
	9	11 Jul 2017	7.1	11K	0 — 10K
	10	11 Jul 2017	6.8	9.3K	12 — 9K
	11	11 Jul 2017	3.1	35	0 — 16
	12	11 Jul 2017	6.3	4K	0 — 4K
	13	11 Jul 2017	14	17K	5 — 16K
	14	12 Jul 2017	6.1	3.6K	8 — 3K
	15	13 Jul 2017	6.2	210	69 — 4
	16	13 Jul 2017	6.8	11K	0 — 11K
	17	13 Jul 2017	6.7	10K	17 — 9K

Our malicious classes have been discussed in prior work: Artemis [4], Dridex [5], Trickster [6], Trickbot [7], Wannacry [8].

Appendix B. Experimental Results

We report here additional tables that we could not insert in the main paper due to page limitations.

B.1. Complete Tables

We report in Tables 10–13 the complete results of our evaluation. Specifically, all these tables report the *standard deviation* (not provided in our paper), thereby allowing one to derive statistical comparisons among individual groups of results (thanks to our experiments being carried out 50 times each). Moreover for Table 13 we also provide the results for the "secondary" adversarially-manipulated traces. For transparency and benchmarking, these tables are provided without any marker highlighting statistical validation (used in Tables 1–4 of the paper).

B.2. Statistical Validation

We use t-tests to statistically confirm some of our claims. First, concerning the validation discussed in §5.1 and §5.2 of the main paper (resulting in the red-cells and/or ↑s/↓ in Tables 2–4), we performed a pairwise comparison by using the average and standard deviation (all provided in Tables 10–13). Then, we carry out additional tests, revolving around four research questions (RQ):

[$RQ1$] does the performance in "future" decreases w.r.t. "past"? (concept-drift check)

[$RQ2$] on "future" data, do our perturbations have any impact? (perturbations+concept drift assessment)

[$RQ3$] does the defense [9] provide a benefit against our blind perturbations? (defense effectiveness)

[$RQ4$] do the TCP perturbations on the traffic generated by Artemis *improve* (*decrease*) the detection rate of the full-binary (ensemble) classifier using RF?

There are many ways to investigate these RQ via statistical tests. Here, we perform this verification by aggregating the results at the "architecture" level, and then comparing the results of the two considered "populations".

Method. We proceed as follows. For each architecture (i.e., full-binary, ensemble, and the various malware-specific classifiers—across both LCS and SCS), we consider two groups (X and Y) for each RQ. Specifically: [$RQ1$] we aggregate the tpr (*and* the tnr) of all ML algorithms (RF and HGB) on *all malicious NetFlows* (and all benign NetFlows), differentiating only among the results on the validation set from "past" (i.e., group X) and those on the test set from "future" (i.e., group Y); we will repeat this for both "vanilla" and "hardened" algorithms, and for both benign and malicious NetFlows. [$RQ2$] we aggregate the tpr of all (vanilla) ML algorithms (RF and HGB) on "future" data and, specifically, on the malicious UDP and TCP NetFlows originating from inside the network, differentiating only among the results for the non-adversarial NetFlows (i.e., group X) and the adversarial ones (i.e., group Y, which includes both "primary" and "secondary" batches of adversarial perturbations). [$RQ3$] similar to $RQ2$, but group Y is represented by the tpr achieved by the corresponding hardened variants of ML-NIDS. [$RQ4$] we aggregate the tpr of the full-binary classifier using the (vanilla) RF on the TCP NetFlows originating from inside the network and stemming from the Artemis malware: group X is for the non-adversarial ones, and group Y is for adversarial ones; then, we repeat this but by considering the ensemble classifier.

Results. After defining our groups, we carry out a t-test comparing the two groups: the output of the test is a statistic, t, which can be converted to a p-value: if p is found to be less than a given target α (which we set to 0.05 [10]), it means that the two groups are generated by a different stochastic process. Put simply: if $p < 0.05$, then X≠Y (and X=Y otherwise); plus, the *sign* of the statistic t is useful to determine whether X is lower/greater than Y. We report the results in Table 14. Here, cells report the resulting p-value (and, in a smaller font, the t statistic); boldface denotes that $p < 0.05$. We use $RQ1^v_b$, $RQ1^h_b$, $RQ1^v_m$, $RQ1^h_m$ to denote the variants of $RQ1$ corresponding to benign/malicious (b/m) and hardened/vanilla (h/v) groups. For $RQ4$ (not reported in Table 14): $\mathbf{p < 0.001}$ with $t = -5.23$ for the full-binary classifier; and $\mathbf{p < 0.001}$ with $t = +50.6$ for the ensemble.

Answers. By observing Table 14, it is apparent that most of the tests revealed that our compared groups can be considered as "statistically different". In other words: [$RQ1$]: concept drifts always causes a degradation to the defense; and it always affects the detection of malicious NetFlows for the vanilla detectors, and for all but one architecture (the Rbot-specific classifier in SCS) for benign NetFlows. [$RQ2$]: in the presence of concept drift, our blind perturbations always cause an increased drop to the tpr for SCS; for LCS, the Artemis- and Wannacry-specific binary classifiers are not further affected (but their performance was terrible to begin with due to concept drift) and, remarkably, the full-binary classifier appears to be robust to our perturbations (in the presence of concept drift). [$RQ3$]: in 3 architectures (full-binary and Virut-specific on SCS, and Trickster-specific on LCS) out of 12, the defense does not cause a statistically significant difference against our adversarial perturbations; however, the statistically significant differences reveal that the defense can be *worse* than the corresponding vanilla variant of the same architecture (as evidenced by the sign of t): this is the case for the full-binary classifier on LCS, and the Neris-specific and ensemble architectures on SCS. For the remaining 6 architectures (Rbot-specific classifier on SCS; and for the ensemble, Artemis-/Dridex-/Trickbot-/Wannacry-specific classifiers for LCS) the defense is better. [$RQ4$]: the full-binary classifier is *better* in the presence of perturbations, whereas the ensemble is *worse*. [these results can be appreciated in our repository by observing a dedicated notebook, or watching our demonstrative 35s video [11].]

B.3. Extra Experiments

We also assessed our perturbations when tested on "past" data. This allows one to appreciate the extent to which concept drift helps in decreasing the tpr. However, this scenario is not very realistic (an organization would sanitize these datapoints), which is why we did not consider these results in our main paper. These results are included in our repository (as figures and raw data) [11].

References

[1] "Mcfp," https://www.stratosphereips.org/datasets-malware.

[2] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Comput. Secur.*, 2014.

[3] QoSient, "Argus NetFlow," https://qosient.com/argus/argusnetflow.shtml, 2012.

[4] M. J. De Lucia and C. Cotton, "Detection of encrypted malicious network traffic using machine learning," in *MILCOM*, 2019.

[5] L. Rudman and B. Irwin, "Dridex: Analysis of the traffic and automatic generation of iocs," in *ISSA*, 2016.

[6] S. K. Katherasala, V. S. Manvith, A. Therala, and M. Murala, "Netmd-network traffic analysis and malware detection," in *ICAIIC*, 2022.

[7] A. Gezer, G. Warner, C. Wilson, and P. Shrestha, "A flow-based approach for trickbot banking trojan detection," *Comp. Secur.*, 2019.

[8] Q. Chen and R. A. Bridges, "Automated behavioral analysis of malware: A case study of wannacry ransomware," in *ICMLA*, 2017.

[9] G. Apruzzese, M. Andreolini, M. Colajanni, and M. Marchetti, "Hardening random forest cyber detectors against adversarial attacks," *IEEE TETCI*, 2020.

[10] G. Apruzzese, P. Laskov, and J. Schneider, "Sok: Pragmatic assessment of machine learning for network intrusion detection," in *EuroS&P*, 2023.

[11] "Our repository," https://github.com/hihey54/aisec24, 2024.

TABLE 10: **Validation results.** We assess the performance of our ML-NIDS on the test set from "past" data. We report the tpr (malicious) and tnr (benign) averaged over 50 trials (we also provide the standard deviation). Defense: $\cup$

Samples	CS / Arch.	SCS: Aug.10th→Aug.18th, 2011 Full	Ens	Neris	Rbot	Virut	LCS: Feb.2017→Jul.2021 Full	Ens	Artemis	Dridex	Trickbot	Trickster	Wannacry
Benign	RF	0.999±0.000	0.999±0.000	0.999±0.000	1.000±0.000	1.000±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000
	HGB	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000
Malicious	RF	0.994±0.001	0.992±0.002	0.993±0.000	0.998±0.000	0.966±0.011	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000
	HGB	0.992±0.002	0.982±0.010	0.995±0.000	0.999±0.000	0.763±0.151	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000
Benign	$\cup$RF	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000
	$\cup$HGB	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000
Malicious	$\cup$RF	0.993±0.002	0.992±0.002	0.992±0.000	0.998±0.000	0.953±0.014	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000
	$\cup$HGB	0.989±0.003	0.983±0.010	0.992±0.000	0.998±0.000	0.762±0.150	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000

TABLE 11: **Concept Drift assessment.** We assess the performance of our ML-NIDS on the test set from "past" data. We report the tpr (malicious) and tnr (benign) averaged over 50 trials (we also report the standard deviation). Defense: $\cup$

Samples	CS / Arch.	SCS: Aug.10th→Aug.18th, 2011 Full	Ens	Neris	Rbot	Virut	LCS: Feb.2017→Jul.2021 Full	Ens	Artemis	Dridex	Trickbot	Trickster	Wannacry
Benign	RF	0.989±0.004	0.993±0.005	0.993±0.005	1.000±0.000	1.000±0.000	0.969±0.001	0.986±0.000	0.999±0.000	0.993±0.000	0.991±0.000	0.999±0.000	0.998±0.000
	HGB	0.990±0.005	0.982±0.009	0.989±0.004	0.999±0.003	0.990±0.011	0.959±0.002	0.965±0.004	0.999±0.000	0.981±0.003	0.983±0.003	0.993±0.002	0.996±0.001
Malicious	RF	0.675±0.007	0.587±0.028	0.701±0.084	0.028±0.000	0.691±0.099	0.927±0.015	0.988±0.008	0.000±0.000	0.982±0.009	0.956±0.038	0.031±0.000	0.994±0.000
	HGB	0.673±0.007	0.757±0.154	0.768±0.068	0.020±0.012	0.663±0.206	0.859±0.068	0.991±0.015	0.010±0.031	0.977±0.016	0.970±0.032	0.031±0.000	0.995±0.000
Benign	$\cup$RF	0.990±0.007	0.996±0.001	0.996±0.001	0.999±0.000	0.999±0.000	0.955±0.002	0.951±0.004	0.995±0.000	0.965±0.000	0.986±0.002	0.992±0.001	0.997±0.000
	$\cup$HGB	0.985±0.012	0.991±0.007	0.995±0.001	0.998±0.003	0.995±0.006	0.955±0.003	0.946±0.004	0.995±0.000	0.962±0.003	0.984±0.001	0.992±0.001	0.996±0.000
Malicious	$\cup$RF	0.631±0.012	0.438±0.026	0.182±0.041	0.024±0.000	0.769±0.002	0.786±0.012	0.957±0.036	0.121±0.026	0.947±0.009	0.920±0.031	0.045±0.088	0.968±0.048
	$\cup$HGB	0.634±0.015	0.561±0.160	0.192±0.054	0.025±0.000	0.665±0.210	0.791±0.013	0.958±0.035	0.130±0.027	0.944±0.016	0.935±0.040	0.093±0.163	0.949±0.057

TABLE 12: **Non-adversarial results.** We measure the tpr (avg 50 trials; we also report the standard deviation) on "future" data on the non-adversarial NetFlows. We only consider UDP and TCP NetFlows starting from within the network. Defense: $\cup$

Proto	CS / Arch.	SCS: Aug.10th→Aug.18th, 2011 Full	Ens	Neris	Rbot	Virut	LCS: Feb.2017→Jul.2021 Full	Ens	Artemis	Dridex	Trickbot	Trickster	Wannacry
UDP	RF	0.980±0.034	0.791±0.268	0.941±0.141	0.151±0.094	0.384±0.325	0.823±0.295	0.807±0.284	0.000±0.000	0.785±0.132	0.832±0.021	0.000±0.000	0.166±0.056
	HGB	0.824±0.250	0.816±0.259	0.990±0.001	0.000±0.000	0.039±0.191	0.754±0.355	0.893±0.166	0.000±0.000	0.807±0.193	0.823±0.042	0.000±0.000	0.593±0.000
TCP	RF	0.923±0.099	0.715±0.228	0.366±0.032	0.436±0.071	0.773±0.075	0.919±0.127	0.988±0.024	0.000±0.000	0.937±0.041	0.922±0.073	0.996±0.001	0.997±0.000
	HGB	0.943±0.069	0.861±0.126	0.458±0.146	0.615±0.089	0.861±0.198	0.983±0.045	0.993±0.012	0.011±0.039	0.926±0.024	0.975±0.041	0.999±0.000	0.998±0.000
UDP	$\cup$RF	0.892±0.121	0.304±0.329	0.031±0.040	0.117±0.128	0.000±0.000	0.698±0.380	0.908±0.158	0.000±0.000	0.894±0.132	0.854±0.018	0.040±0.195	0.586±0.010
	$\cup$HGB	0.921±0.088	0.478±0.385	0.025±0.034	0.103±0.123	0.113±0.307	0.705±0.383	0.909±0.158	0.000±0.000	0.864±0.193	0.852±0.020	0.060±0.237	0.583±0.011
TCP	$\cup$RF	0.880±0.086	0.814±0.132	0.412±0.112	0.513±0.135	0.959±0.003	0.979±0.041	0.958±0.071	0.128±0.032	0.741±0.041	0.940±0.043	0.994±0.000	0.973±0.048
	$\cup$HGB	0.867±0.093	0.810±0.135	0.378±0.062	0.507±0.125	0.858±0.214	0.982±0.046	0.976±0.040	0.134±0.023	0.744±0.024	0.952±0.046	0.994±0.000	0.945±0.058

TABLE 13: **Adversarial results.** We measure the tpr (avg 50 trials); we also report the standard deviation) on "future" data on the adversarial NetFlows. We only consider UDP and TCP NetFlows starting from within the network. Defense: $\cup$. The "primary" results are those in the main paper, whereas "secondary" are reported only here (there is no statistically significant differences between "primary" and "secondary").

Pert. / Proto	CS / Arch.	SCS: Aug.10th→Aug.18th, 2011 Full	Ens	Neris	Rbot	Virut	LCS: Feb.2017→Jul.2021 Full	Ens	Artemis	Dridex	Trickbot	Trickster	Wannacry
(primary) UDP	RF	0.921±0.112	0.602±0.304	0.470±0.299	0.011±0.031	0.032±0.079	0.812±0.296	0.700±0.307	0.000±0.000	0.009±0.002	0.587±0.047	0.000±0.000	0.128±0.075
	HGB	0.824±0.248	0.803±0.265	0.975±0.094	0.014±0.061	0.059±0.233	0.740±0.363	0.899±0.157	0.000±0.000	0.660±0.308	0.833±0.032	0.000±0.000	0.593±0.000
(primary) TCP	RF	0.902±0.118	0.717±0.229	0.381±0.075	0.292±0.051	0.769±0.080	0.929±0.116	0.934±0.060	0.000±0.000	0.869±0.055	0.778±0.004	0.841±0.052	0.991±0.000
	HGB	0.934±0.080	0.861±0.135	0.472±0.166	0.470±0.102	0.822±0.252	0.967±0.047	0.980±0.023	0.022±0.046	0.910±0.030	0.856±0.071	0.298±0.095	0.992±0.001
(secondary) UDP	RF	0.923±0.108	0.599±0.307	0.457±0.312	0.015±0.043	0.054±0.104	0.807±0.302	0.715±0.307	0.000±0.000	0.010±0.004	0.628±0.061	0.000±0.000	0.138±0.073
	HGB	0.824±0.250	0.806±0.264	0.989±0.002	0.000±0.000	0.039±0.191	0.754±0.355	0.893±0.166	0.000±0.000	0.649±0.295	0.833±0.042	0.000±0.000	0.593±0.000
(secondary) TCP	RF	0.901±0.121	0.721±0.227	0.369±0.037	0.304±0.048	0.761±0.075	0.923±0.122	0.932±0.062	0.000±0.000	0.878±0.052	0.777±0.006	0.853±0.042	0.991±0.000
	HGB	0.935±0.077	0.875±0.122	0.461±0.146	0.493±0.111	0.854±0.198	0.969±0.050	0.981±0.022	0.017±0.046	0.919±0.026	0.846±0.070	0.289±0.076	0.992±0.001
(primary) UDP	$\cup$RF	0.873±0.147	0.334±0.360	0.052±0.030	0.050±0.078	0.000±0.000	0.698±0.380	0.908±0.158	0.000±0.000	0.881±0.002	0.855±0.015	0.020±0.139	0.586±0.010
	$\cup$HGB	0.889±0.128	0.389±0.369	0.060±0.038	0.079±0.122	0.058±0.231	0.707±0.385	0.909±0.158	0.000±0.000	0.826±0.308	0.856±0.011	0.080±0.271	0.581±0.011
(primary) TCP	$\cup$RF	0.863±0.087	0.831±0.119	0.400±0.096	0.378±0.110	0.953±0.003	0.922±0.119	0.963±0.050	0.127±0.024	0.721±0.055	0.742±0.043	0.499±0.062	0.969±0.048
	$\cup$HGB	0.868±0.087	0.834±0.124	0.414±0.113	0.410±0.077	0.816±0.253	0.931±0.104	0.974±0.031	0.132±0.024	0.651±0.030	0.762±0.042	0.507±0.115	0.949±0.058
(secondary) UDP	$\cup$RF	0.868±0.147	0.312±0.340	0.061±0.039	0.101±0.137	0.000±0.000	0.699±0.380	0.908±0.158	0.000±0.000	0.885±0.004	0.854±0.018	0.040±0.195	0.586±0.010
	$\cup$HGB	0.893±0.124	0.481±0.384	0.056±0.033	0.077±0.130	0.109±0.300	0.705±0.384	0.909±0.158	0.000±0.000	0.845±0.295	0.852±0.020	0.060±0.237	0.582±0.011
(secondary) TCP	$\cup$RF	0.867±0.085	0.834±0.119	0.414±0.112	0.387±0.084	0.953±0.003	0.907±0.130	0.954±0.062	0.132±0.029	0.704±0.052	0.739±0.043	0.495±0.063	0.971±0.048
	$\cup$HGB	0.862±0.088	0.832±0.121	0.382±0.062	0.387±0.068	0.854±0.218	0.919±0.129	0.972±0.036	0.135±0.020	0.657±0.026	0.751±0.045	0.470±0.052	0.943±0.058

TABLE 14: **Statistical Validation.** We validate our RQ with statistical t-tests. Cells report the resulting p-value of the test: numbers in boldface are when $p < 0.05$.

CS / Arch.	SCS: Aug.10th→Aug.18th, 2011 Full	Ens	Neris	Rbot	Virut	LCS: Feb.2017→Jul.2021 Full	Ens	Artemis	Dridex	Trickbot	Trickster	Wannacry
$RQ1_b^v$	< **0.001**+27.9	< **0.001**+13.1	< **0.001**+18.8	0.301+1.03	< **0.001**+6.19	< **0.001**+85.4	< **0.001**+31.8	< **0.001**+12.5	< **0.001**+26.9	< **0.001**+39.1	< **0.001**+13.8	< **0.001**+23.1
$RQ1_b^h$	< **0.001**+15.9	< **0.001**+9.11	< **0.001**+28.4	< **0.001**+2.70	< **0.001**+3.69	< **0.001**+201	< **0.001**+145	< **0.001**+75.7	< **0.001**+120	< **0.001**+61.2	< **0.001**+93.7	< **0.001**+49.2
$RQ1_m^v$	< **0.001**+539	< **0.001**+41.6	< **0.001**+45.1	< **0.001**+1.5k	< **0.001**+16.0	< **0.001**+26.1	< **0.001**+10.9	< **0.001**+549	< **0.001**+24.1	< **0.001**+14.2	< **0.001**+44k	< **0.001**+80.3
$RQ1_m^h$	< **0.001**+326	< **0.001**+53.6	< **0.001**+244	< **0.001**+19k	< **0.001**+12.5	< **0.001**+210	< **0.001**+16.9	< **0.001**+438	< **0.001**+92.7	< **0.001**+28.0	< **0.001**+98.1	< **0.001**+10.6
$RQ2$	< **0.001**+3.56	< **0.001**+4.38	< **0.001**+5.61	< **0.001**+6.13	< **0.001**+2.88	0.405+0.83	< **0.001**+6.78	0.157−1.41	< **0.001**+12.0	< **0.001**+18.1	< **0.001**+7.00	0.710+0.37
$RQ3$	0.496−0.68	< **0.001**+12.5	< **0.001**+19.3	< **0.001**−2.38	0.158−1.41	< **0.001**+4.25	< **0.001**−10.0	< **0.001**−16.7	< **0.001**−7.22	< **0.001**−5.45	0.542+0.61	< **0.001**−4.62