Vulnerability of Deep Network Intrusion Detectors to Evasion Varies with the Training Seed.

Abstract: Evaluations of deep network intrusion detectors usually rest on one trained model, a random train–test split and an attacker that learns nothing from one flow to the next. We drop these three habits on the corrected CIC-IDS-2017. A chronological split lowers the macro F1 of a 1D CNN by only 0.042 but raises its false-positive rate from 0.003% to about 2%, mostly on one benign service seen on the last day. Nine CNNs that differ only in their seed reach a test macro F1 of 0.87 to 0.95, yet a decision-based attacker that may only pad, delay or add packets to its own flows gets through 0% to 99% of a family's flows depending on the detector, which explains 96% to 100% of the sum of squares across 15 attack conditions. Thirty-three MLPs behave alike, and a budget five to ten times smaller keeps the ranking of the detectors. Detection metrics do not predict this vulnerability, whereas attacking validation flows does (r_S = 0.60 to 1.00). Vulnerability should be measured over several training runs and used to choose the deployed model.

Index Terms: Intrusion detection, deep learning, adversarial evasion, decision-based attack, temporal evaluation, underspecification.

1. Introduction

On CIC-IDS-2017, flow-based deep detectors often exceed an F1 of 0.98 [1], [2]. These scores rest on three habits. The split mixes past and future traffic, which inflates results [3], [4], and the false-positive rate, which sets the analysts' workload, is seldom given [5], [6]. Evasion is studied with access to the model, or with changes an attacker could not make to real traffic [7], [8]. Finally, one trained model stands for the whole method, although a training pipeline can produce models that score alike on held-out data and differ elsewhere [9], [10].

We drop the three habits at once. Our main finding is that detectors trained the same way, with only the seed changed, differ almost entirely in how easily the same attack evades them; their detection scores do not tell them apart, but an attack on validation flows does. We also locate the false alarms that a chronological split reveals and state which flow statistics the attacker's actions change. Code, models, flow sets and per-flow results are public.

2. Related work

CIC-IDS-2017 and CSE-CIC-IDS-2018 were relabelled and their flows rebuilt by Engelen et al. [11] and Liu et al. [12]; Lanvin et al. [13] and Flood et al. [14] measured how dataset defects bias results. Pierazzi et al. [7] separate feature-space perturbations from attacks that can be carried out on real objects, and Apruzzese et al. describe realistic NIDS adversaries [8] and evasion under concept drift [15]. Label-only attacks such as HopSkipJumpAttack [16] spend many queries on each input; statefuldefences watch for such query streams [17], and adaptive attackers get around them [18]. Underspecification [9] predicts that equally accurate models differ off the validation distribution; we test it for evasion, keeping the attack identical across models.

UNDER PEER REVIEW IN IJAR

3. Setting

3.1 Flows and chronological split

CICFlowMeter groups the packets of a connection into a flow and summarises the sizes, gaps and flags of its packets by counts, sums, extremes, means, deviations and rates; after cleaning, a flow is a vector of 82 numbers. With m forward packets of sizes s_j, m' backward packets and $K = m + m'$ packets,

$$D = t_K - t_1, \qquad B_{\text{fwd}} = \sum_{j=1}^{m} s_j\,, \qquad B_{\text{bwd}} = \sum_{j=1}^{m'} s'_j \tag{1}$$

$$\text{Flow Bytes/s} = \frac{10^6\,(B_{\text{fwd}} + B_{\text{bwd}})}{D}, \qquad \text{Flow Packets/s} = \frac{10^6\,K}{D} \tag{2}$$

where the duration D is in microseconds, as in the dataset. Every feature goes through a signed logarithm and a min-max scaling fitted on training data. Each attack family of CIC-IDS-2017 was captured on a single day, so the split is made inside each class c: oldest 70% for training, next 15% for validation, last 15% for test,

$$\max_{x\in\text{Train}(c)} t(x) \;\le\; \min_{x\in\text{Val}(c)} t(x), \qquad \max_{x\in\text{Val}(c)} t(x) \;\le\; \min_{x\in\text{Test}(c)} t(x) \qquad \forall c \tag{3}$$

This tests later traffic of known classes; it does not simulate a single deployment date.

3.2 Threat model and actions

The attacker owns the source of a malicious flow that the detector flags and learns only the predicted label of each query. It cannot touch the server's answers and may only add to its forward traffic, through an action $a = (b, \delta, \kappa)$: b padding bytes per forward packet, a delay δ between forward packets, and κ dummy packets per forward packet. Actions are sampled in $[0,1]^3$ and squared, so small changes are tried more finely:

$$a(u) = \left(u_1^2\,\bar{b}(\beta),\; u_2^2\,\bar{\delta}(\beta),\; u_3^2\,\bar{\kappa}(\beta)\right), \qquad \mathcal{A}(\beta) = \left[0,\bar{b}(\beta)\right] \times \left[0,\bar{\delta}(\beta)\right] \times \left[0,\bar{\kappa}(\beta)\right] \tag{4}$$

Table 1. Upper limits of the three actions, per forward packet, at each budget.

Budget β	Padding $\bar{b}$	Delay $\bar{\delta}$	Dummy-packet ratio $\bar{\kappa}$
Low	100 bytes	10 ms	0.25
Medium	500 bytes	100 ms	1.0
High	1,000 bytes	1 s	2.0

A flow counts as evaded when it is classified benign within $L = 1{,}000$ queries. We modify the feature vector, not packets: every original forward packet is padded, the delay precedes each original forward packet but the first, and dummy packets carrying $\bar{s} + p$ bytes fall inside existing forward gaps. With $P_{\max} = 1{,}460$ bytes,

$$p = \max\left(0,\; \min(b,\; P_{\max} - s_{\max})\right), \qquad k = \text{round}(\kappa\, m), \qquad B'_{\text{fwd}} = B_{\text{fwd}} + m\,p + k\,(\bar{s} + p) \tag{5}$$

UNDER PEER REVIEW IN IJAR

$$m^{\star} = m + k, \qquad D' = D + (m-1)\,\delta, \qquad \bar{s}' = \bar{s} + p, \qquad \text{Down/Up}' = \frac{m'}{m+k} \quad \text{(3)}$$

These conventions give closed forms for 22 features, which we recompute from the stored raw flow; on 2,000 synthetic packet traces they match the statistics of the modified traces to a relative error below 10^{-5}. Six position-dependent features follow rules of thumb, 28 that real traffic would change stay as they are, and 26 are set by the server or the protocol. This checks our algebra, not CICFlowMeter itself, so we speak of feature-space evasion under traffic constraints.

4. Attacker and protocol

The adaptive attacker keeps a memory $\mathcal{M}$ of up to 20 actions that worked on earlier flows of the campaign (Algorithm 1). On a new flow it first replays them, then spends half of what is left on Gaussian perturbations of stored actions, and the rest on random draws whose scale grows with the query index. Once an action works, six bisection steps on its amplitude bring it close to the edge of the evading region.

Algorithm 1. Adaptive attacker on one flow

1: candidates ← stored actions, then their mutations, then escalating random draws, L in total
2: try the candidates in order; if none is classified benign, **return** failure
3: shrink the winning action u by six bisection steps on $t \in [0,1]$, keeping the smallest t that still evades
4: put $\min(\mathbf{1}, 1.25\,t\,u)$ first in $\mathcal{M}$, keep 20 entries, **return** success

A campaign attacks one family's flows in a set order, starting with an empty memory. To compare detectors fairly we fix four things. The flows: 100 test flows per family (21 for Bot), drawn once for all detectors. The success criterion: only a benign decision is an evasion, so every flow that raises an alert is attacked, and flows already judged benign count as evaded at no cost. The randomness: flow i gets its own generator, seeded by attack seed, attacker, family and i, so all detectors see the same random numbers on the same flow. The order: chronological, or one of two fixed shuffles. Five attack seeds and three orders make 15 attack conditions. Before each campaign we check that the null action leaves every decision unchanged.

5. Experiments

We use the corrected CIC-IDS-2017 [11], [12] with *Attempted* flows labelled benign, and keep 20% of each day (419,976 flows, 7 classes, 76.0% benign). The CNN stacks convolutions with 64, 128 and 256 filters, a global average and two dense layers (157,319 weights); the MLP has hidden layers of 256, 128 and 64 units (62,855 weights). Training uses Adam with a learning rate of 10^{-3} and batches of 512, and keeps the epoch with the best validation macro F1. Two models of the same kind differ only in their seed, which sets the initial weights, the batch order and the dropout masks. We attack nine CNNs (seeds 1–9), 33 MLPs (seeds 0–32) and three CNNs without the destination port, in 3,870 campaigns (Fig. 1).

UNDER PEER REVIEW IN IJAR

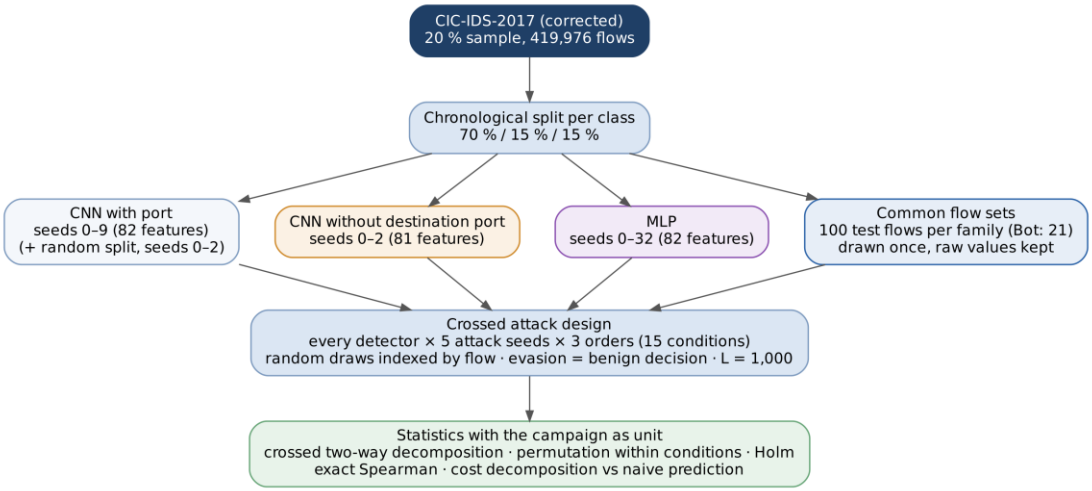


Fig. 1. Overview of the design: one sample, one chronological split, one set of attack flows shared by all detectors.

Because flows within an adaptive campaign depend on each other, the unit of analysis is the campaign. For one family, the evasion rate E_{kj} of detector k in condition j is modelled as

$$E_{kj} = \mu + \alpha_k + \gamma_j + \varepsilon_{kj}, \qquad \alpha_k \sim (0, \sigma_D^2), \quad \gamma_j \sim (0, \sigma_C^2), \quad \varepsilon_{kj} \sim (0, \sigma_E^2) \qquad (7)$$

and the detector's share of variance is

$$\hat{\sigma}_E^2 = \mathrm{MS}_E, \quad \hat{\sigma}_D^2 = \max\left(0, \frac{\mathrm{MS}_D - \mathrm{MS}_E}{J}\right), \quad \hat{\sigma}_C^2 = \max\left(0, \frac{\mathrm{MS}_C - \mathrm{MS}_E}{K}\right) \qquad (8)$$

$$\hat{\eta}_D = \frac{\hat{\sigma}_D^2}{\hat{\sigma}_D^2 + \hat{\sigma}_C^2 + \hat{\sigma}_E^2} \qquad (9)$$

We report it with a bootstrap over detectors and conditions, and test whether detectors differ by shuffling their labels inside each condition (10,000 shuffles). Links between detection metrics and evasion use Spearman's r_S with permutation p-values, and the tests behind one claim are corrected with Holm's method [19].

6. Results

6.1 Detection and false positives

Moving the CNN from a random to a chronological split lowers the macro F1 from 0.982 to 0.940 and raises the false-positive rate (FPR) from 0.003% to 2.01%. All 16 originally trained detectors have an FPR between 1.92% and 2.17%. On the last capture day, 832 benign test flows go to port 8080; every detector calls 95% of them PortScan, and they make up 77% to 85% of all false alarms (Fig. 2). Without them the FPR drops to 0.31–0.52%. This is a concentration of errors on one service, not a proven cause, and allow-listing the service would also let attacks through it.

UNDER PEER REVIEW IN JAR

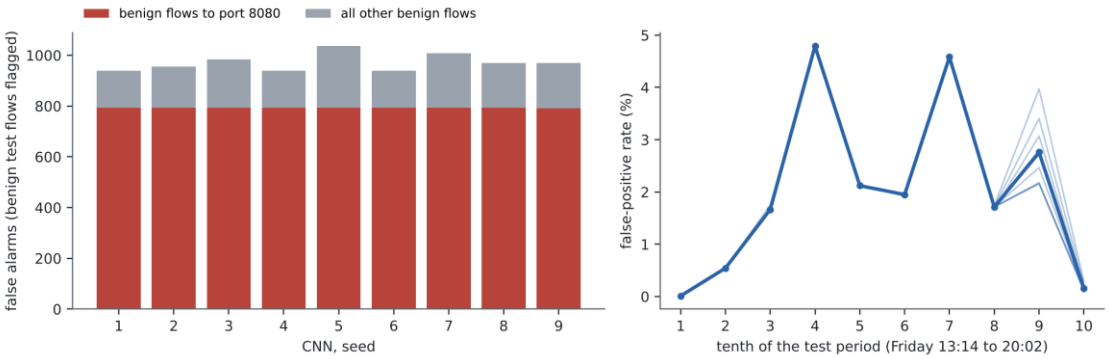


Fig. 2. False alarms of the nine CNNs: (a) share due to port 8080; (b) FPR in each tenth of the benign test period.

6.2 Same attack, same flows, different detectors

The nine attacked CNNs have an FPR of 1.96% to 2.17% and, seed 3 apart, a test macro F1 of 0.93 to 0.95. Under the same 15 conditions and on the same flows (Fig. 3, Table 2), their evasion rates span 5–100% for Bot, 14–100% for BruteForce, 9–100% for DDoS, 8–93% for DoS and 0–99% for PortScan. Repeating the attack on one detector moves its rate by 3.8 to 15 points at most, except for BruteForce on the CNN of seed 7 (1% to 41%), the one case where the attacker's luck matters.

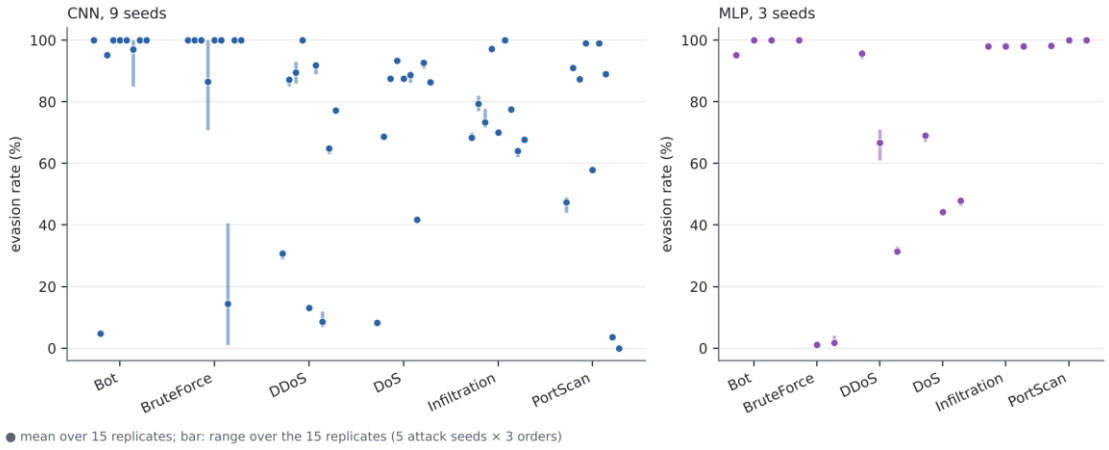


Fig. 3. Evasion at the medium budget on the same flows: nine CNNs (left) and three MLPs (right); a dot is a detector's mean over 15 attack conditions, its bar their range.

Table 2. For each of the nine CNNs: validation and test macro F1, FPR, and mean evasion rate per family (medium budget, 15 conditions).

CNN seed	Val. F1	Test F1	FPR	Bot	Brute Force	DDoS	DoS	Infiltration	Port Scan
1	0.983	0.944	1.96%	100%	100%	31%	3%	58%	47%
2	0.976	0.950	2.00%	5%	100%	37%	59%	79%	91%
3	0.982	0.874	2.05%	95%	100%	90%	38%	73%	37%
4	0.981	0.942	1.96%	100%	36%	100%	93%	97%	99%
5	0.982	0.933	2.17%	100%	100%	13%	38%	70%	58%
6	0.980	0.937	1.96%	100%	100%	92%	39%	100%	99%
7	0.970	0.939	2.10%	97%	14%	9%	42%	78%	39%
8	0.981	0.946	2.03%	100%	100%	55%	93%	54%	4%
9	0.965	0.937	2.03%	100%	100%	77%	36%	58%	0%

Table 3 splits this variation. The detector carries 13 to 39 points of standard deviation, the attack condition at most one point and the residual 0.6 to 5.3 points; the detector holds 96–100% of the sum

of squares, and shuffling detector labels never produces a gap as large ($p < 0.001$ for each family, Holm). Counting already-benign flows as evaded, or keeping only flows flagged by all nine CNNs, shifts no share by more than 0.1 point.

Table 3. Variance of the evasion rate over nine CNNs and 15 attack conditions: standard deviations by source, detector share of the sum of squares, $\hat{\eta}_D$ with bootstrap interval and Holm-corrected permutation p-value.

Family	Range of the detector means	SD detector	SD condition	SD interaction and residual	Sum of squares, $\hat{\eta}_D$ detector	[95% interval]	p (Holm)
Bot	5–100%	31.5	0.0	1.4	99.8%	[0.26, 1.00 1.00]	< 0.001
BruteForce	14–100%	28.3	1.0	5.3	96.3%	[0.47, 0.97 1.00]	< 0.001
DDoS	9–100%	35.7	0.0	1.1	99.9%	[0.99, 1.00 1.00]	< 0.001
DoS	3–93%	29.2	0.0	0.6	100.0%	[0.99, 1.00 1.00]	< 0.001
Infiltration	54–100%	12.9	0.0	0.9	99.5%	[0.94, 1.00 1.00]	< 0.001
PortScan	0–99%	39.4	0.0	0.6	100.0%	[1.00, 1.00 1.00]	< 0.001

The 33 MLPs, with a test macro F1 of 0.895–0.946, spread just as widely: for BruteForce, 16 let 90% or more of the flows through and 17 at most 5%; DDoS spans 5–100% and DoS 4–91%; the detector holds at least 99.8% of the sum of squares. Dropping the destination port leaves detection unchanged (macro F1 0.928) and makes the three CNNs without port somewhat more vulnerable at the low budget, a trend that three models per group cannot establish (Holm-corrected $p = 0.30$).

6.3 A smaller budget

At the low budget, on all 42 detectors and the same flows (Fig. 4), no detector is more vulnerable than at the medium budget, but the drop depends on the family and the architecture. Infiltration and PortScan barely move. Bot separates the architectures: the median CNN now lets none of its alerted Bot flows through, against all of them at the medium budget, while the median MLP still lets 95% through. BruteForce, let through on more than half of the flows by 24 detectors at the medium budget, is let through that much by only five. DDoS and DoS lose 20 to 25 points in median. The detector still explains at least 98% of the variation of each family, and the two budgets rank the detectors alike (r_S from 0.56 for BruteForce to 0.96 for PortScan), except for Bot (0.28), which has only 21 test flows.

UNDER PEER REVIEW IN JAR

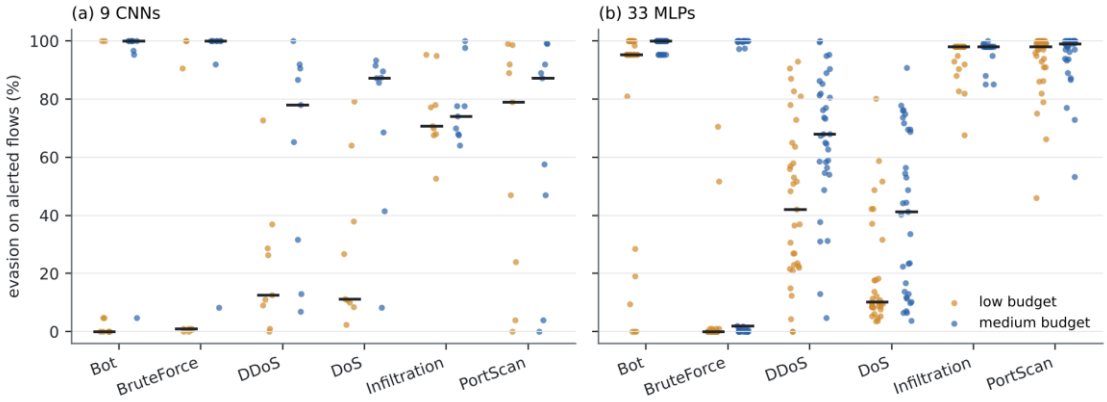


Fig. 4. Evasion of the 42 detectors at the low budget (orange) and at the medium budget (blue) on the alerted test flows, three attack seeds; a dot is a detector, a bar the median.

6.4 Does memory help?

Against a blind search with growing scale, the adaptive attacker is cheaper by a median factor of 39 for BruteForce, 15 for Bot and 1.4 to 3.1 elsewhere. This flatters memory, because the blind schedule wastes queries on tiny actions: over the nine CNNs, a uniform random search costs 2.57 million queries, as much as the adaptive attacker. Memory pays only when one action suits many flows, as for BruteForce, where an adaptive variant with uniform draws needs 58,000 queries against 229,000–253,000 for the other attackers.

6.5 Can vulnerability be measured before deployment?

Detection metrics do not reveal it. Across the nine CNNs, none of 18 correlations between a detection metric and an evasion rate is significant, and for the 33 MLPs validation macro F1 and FPR stay within $|r_S| \leq 0.38$. An attack does. We attacked every detector on 100 validation flows per family (Fig. 5). Ranked by validation evasion and by test evasion, the detectors come out in nearly the same order: r_S = 0.60–0.99 for the MLPs ($p < 0.001$ in each family, Holm) and 0.72–1.00 for the CNNs (Holm $p \leq 0.04$), and in every family the CNN with the lowest validation evasion is also the least vulnerable on the test flows. A cheaper probe, the same 20 random actions on each validation flow (2,000 queries per family), already correlates with test evasion at 0.64–0.95 for the MLPs.

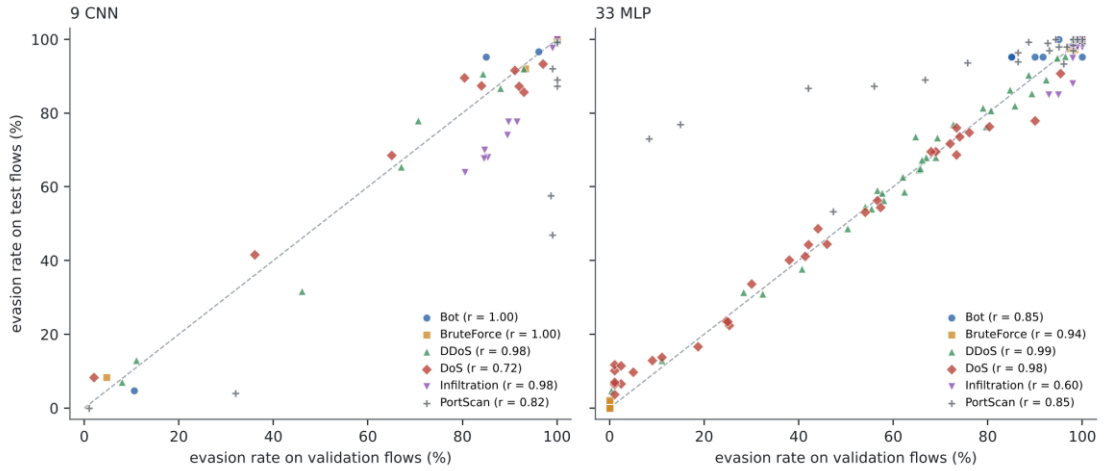


Fig. 5. Validation against test evasion, one point per detector and family; nine CNNs (left) and 33 MLPs (right).

UNDER PEER REVIEW IN IJAR

7. Discussion and limits

Two models with the same accuracy may differ by almost 100 points in how often one attack gets through, and their validation scores will not say which. Attacking candidate models on a separate set of validation flows, or at least probing them with random actions, is a cheap way to choose; the final test set must stay out of that choice. For statefuldefences, an adaptive campaign stands out at its start, when one flow receives hundreds of queries, but later sends about one query per flow.

The results come from one laboratory dataset of five days, split per class, and validation and test flows come from the same days. The attacks act on features: 28 features that would change in real traffic are not updated and no modified flow was replayed through CICFlowMeter, so the measured evasion is neither an upper nor a lower bound for live traffic. A failure within L queries does not prove that no evading action exists, and we measure one attack policy at two budgets.

8. Conclusion

A chronological split barely moves the macro F1 of a deep detector but multiplies its false-positive rate by more than six hundred. Nine CNNs and 33 MLPs that differ only in their seed reach similar accuracy, yet the same attack on the same flows gets through almost none of them or almost all, depending on the model and the family, at two attack budgets. Detection scores miss this; an attack on validation flows does not. Evaluations of evasion should cover several training runs with a fixed attack, and use that attack to pick the deployed model.

Data availability. The corrected CIC-IDS-2017 is available athttps://www.kaggle.com/datasets/ernie55ernie/improved-cicids2017-and-csecicids2018. The 45 trained detectors used in the attacks are published at https://www.kaggle.com/datasets/boriszannou/ids-modeles and the training and attack notebooks, flow sets, per-flow results and manifests (SHA-256 digests and software versions) at https://www.kaggle.com/boriszannou/code.

UNDER PEER REVIEW IN IJAR

References

[1] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy (ICISSP)*, 2018, pp. 108–116, doi: 10.5220/0006639801080116.

[2] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019, doi: 10.1109/ACCESS.2019.2895334.

[3] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro, "TESSERACT: Eliminating experimental bias in malware classification across space and time," in *Proc. 28th USENIX Secur. Symp.*, 2019, pp. 729–746.

[4] D. Arp et al., "Dos and don'ts of machine learning in computer security," in *Proc. 31st USENIX Secur. Symp.*, 2022, pp. 3971–3988.

[5] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proc. IEEE Symp. Secur. Privacy (SP)*, 2010, pp. 305–316, doi: 10.1109/SP.2010.25.

[6] G. Apruzzese, P. Laskov, and J. Schneider, "SoK: Pragmatic assessment of machine learning for network intrusion detection," in *Proc. IEEE 8th Eur. Symp. Secur. Privacy (EuroS&P)*, 2023, pp. 592–614, doi: 10.1109/EuroSP57164.2023.00042.

[7] F. Pierazzi, F. Pendlebury, J. Cortellazzi, and L. Cavallaro, "Intriguing properties of adversarial ML attacks in the problem space," in *Proc. IEEE Symp. Secur. Privacy (SP)*, 2020, pp. 1332–1349, doi: 10.1109/SP40000.2020.00073.

[8] G. Apruzzese, M. Andreolini, L. Ferretti, M. Marchetti, and M. Colajanni, "Modeling realistic adversarial attacks against network intrusion detection systems," *Digit. Threats: Res. Pract.*, vol. 3, no. 3, pp. 1–19, 2022, doi: 10.1145/3469659.

[9] A. D'Amour et al., "Underspecification presents challenges for credibility in modern machine learning," *J. Mach. Learn. Res.*, vol. 23, no. 226, pp. 1–61, 2022.

[10] X. Bouthillier et al., "Accounting for variance in machine learning benchmarks," in *Proc. Mach. Learn. Syst. (MLSys)*, vol. 3, 2021.

[11] G. Engelen, V. Rimmer, and W. Joosen, "Troubleshooting an intrusion detection dataset: The CICIDS2017 case study," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, 2021, pp. 7–12, doi: 10.1109/SPW53761.2021.00009.

[12] L. Liu, G. Engelen, T. Lynar, D. Essam, and W. Joosen, "Error prevalence in NIDS datasets: A case study on CIC-IDS-2017 and CSE-CIC-IDS-2018," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, 2022, pp. 254–262, doi: 10.1109/CNS56114.2022.9947235.

[13] M. Lanvin, P.-F. Gimenez, Y. Han, F. Majorczyk, L. Mé, and É. Totel, "Errors in the CICIDS2017 dataset and the significant differences in detection performances it makes," in *Risks and Security of Internet and Systems (CRiSIS 2022)*, ser. Lecture Notes in Computer Science. Cham, Switzerland: Springer, 2023, pp. 18–33, doi: 10.1007/978-3-031-31108-6_2.

[14] R. Flood, G. Engelen, D. Aspinall, and L. Desmet, "Bad design smells in benchmark NIDS datasets," in *Proc. IEEE 9th Eur. Symp. Secur. Privacy (EuroS&P)*, 2024, pp. 658–675, doi: 10.1109/EuroSP60621.2024.00042.

[15] G. Apruzzese, A. Fass, and F. Pierazzi, "When adversarial perturbations meet concept drift: An exploratory analysis on ML-NIDS," in *Proc. Workshop Artif. Intell. Secur. (AISec)*, 2024, pp. 149–160, doi: 10.1145/3689932.3694757.

[16] J. Chen, M. I. Jordan, and M. J. Wainwright, "HopSkipJumpAttack: A query-efficient decision-based attack," in *Proc. IEEE Symp. Secur. Privacy (SP)*, 2020, pp. 1277–1294, doi: 10.1109/SP40000.2020.00045.

[17] S. Chen, N. Carlini, and D. Wagner, "Stateful detection of black-box adversarial attacks," in *Proc. 1st ACM Workshop Secur. Privacy Artif. Intell. (SPAI)*, 2020, pp. 30–39, doi: 10.1145/3385003.3410925.

[18] R. Feng, A. Hooda, N. Mangaokar, K. Fawaz, S. Jha, and A. Prakash, "Stateful defenses for machine learning models are not yet secure against black-box attacks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2023, doi: 10.1145/3576915.3623116.

[19] S. Holm, "A simple sequentially rejective multiple test procedure," *Scand. J. Statist.*, vol. 6, no. 2, pp. 65–70, 1979.

UNDER PEER REVIEW IN IJAR