Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Robustness of automated intrusion responders in software-defined networks to spoofed evidence and responder starvation

Amr H. Abdelhaliem
Irbid National University
Mohamed Heshmat
Ajloun National University
Saddam Askar
Saddam.kamel@su.edu.ye
Sana'a University
Mohammed Tawfik
Ajloun National University

Research Article

Keywords:

Posted Date: October 26th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11245064/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Robustness of automated intrusion responders in software-defined networks to spoofed evidence and responder starvation

Abstract

Context. Automated responders in software-defined networks turn intrusion alerts into blocking rules within seconds. **Gap.** Because these responders act on identifiers carried by the evidence, an adversary who forges those identifiers may aim the response at legitimate hosts, yet no evaluation of automated or language-model responders has considered such an adversary. **Method.** We define this mitigation-induced denial of service with four adversary objectives and three harm metrics. We measure it in 2088 trace-driven closed-loop runs on six topologies against seven responders, with an adversary controlling one compromised host and its IPv4 source address. We then evaluate four defenses for robustness and for their cost against genuine attacks. **Results.** A classic "block the source" responder was turned against legitimate hosts in 144 of 216 runs. It destroyed up to 173.8 bytes of legitimate traffic per adversary byte and cut protected-service availability to 65.8%. A fine-tuned language-model responder was fooled in 7 of 216 runs, but severely when it was. Pre-deployment verification removed those cases but not starvation: rotating spoofed identities kept the slow synthesizer busy, and a concurrent genuine attack was never mitigated. Ingress-port binding, alert corroboration and rule budgets eliminated measurable harm at a 1.08× time-to-mitigate cost; starvation was halved. **Implication.** Automated responders need an explicit harm model for the evidence they act on.

Keywords: software-defined networking; intrusion response; spoofing; denial of service; adversarial robustness; large language models; OpenFlow; Internet of Things

1 Introduction

Problem. Automated intrusion response is attractive in software-defined networks (SDN). The controller can install a blocking or rate-limiting rule within milliseconds of an alert [1, 2], and recent systems couple language-model detectors and rule generators to exactly this capability [3–5]. The risk is older than SDN: an automated response acts on evidence, and evidence can be forged. When a responder blocks "the source" of an attack, an adversary who controls what the source looks like controls whom the network blocks. Source-address spoofing remains measurable on the Internet [6] despite long-standing ingress-filtering recommendations [7], and inside a network a compromised host can usually forge its IPv4 source address freely.

Gap. Two decades ago, Toth and Kruegel argued that response actions carry a cost to legitimate users that must be weighed against their benefit [8], and cost-sensitive response models followed [9]. The current generation of SDN responders evaluates only how fast and how completely they stop an attack. Three gaps remain:

- none of these responders has been evaluated against an adversary who targets the responder itself;
- adversarial machine learning against intrusion detection studies *evasion* [10–14], not the opposite case of an adversary who *wants* to be detected, with evidence that points at somebody else;
- SDN control-plane attack research [15–18] treats the controller as the target, not the response application.

We therefore do not know how much harm a forged-evidence adversary can cause through an automated responder, which responder designs are vulnerable, or what it costs to defend them.

Research question. *Once a defender installs blocking rules automatically, can an adversary exploit the responder to make the network deny service to its own legitimate hosts and services, and at what cost can this be prevented?* We call this *mitigation-induced denial of service* and study it as a defensive evaluation.

Contributions.

- **A formal threat model of mitigation-induced denial of service.** The adversary is characterized by directly and indirectly controllable fields [14]. It pursues four objectives against the responder: **O1** collateral induction, **O2** protected-service isolation, **O3** responder exhaustion and **O4** oscillation (Section 3, Fig. 1).
- **Harm metrics with per-rule attribution** (Algorithm 1): amplification, legitimate-host downtime per adversary packet, and protected-service availability. Every lost byte is traced to the rule and the incident that caused it, which separates adversary-induced harm from the harm caused by detector false positives.
- **A controlled factorial evaluation** of seven responders that reproduce the state-of-the-art designs: port blocking, classic templates, a verified template ladder, and a language-model synthesizer with and without pre-deployment verification. The evaluation spans four objectives, three adversary budgets and six topologies in 2088 trace-driven closed-loop runs, with pre-registered hypotheses, mixed-effects models, and validation of the data-plane predictions on a real Containernet/Open vSwitch/Ryu network.
- **Four defenses with measured robustness and cost** (Algorithm 2): provenance-aware scoping, evidence thresholding, cost-sensitive gating, and hysteresis with rule budgets. We show formally and empirically that provenance-aware scoping confines adversary-induced cuts to the adversary's own ingress port and fails safe under stale bindings. We give single-defense ablations, a binding-staleness study, and design guidelines.

Attack traffic exists only inside the emulated testbed, and no exploit code against real controllers or devices is released.

Roadmap. Section 2 reviews related work. Section 3 formalizes the system, the adversary, the harm metrics and the defenses. Section 4 describes the experimental setup. Section 5 reports and interprets the results. Section 6 analyses the defenses individually. Section 7 discusses threats to validity, and Section 8 concludes.

2 Related work

Attacks on the SDN control plane. SDN controllers maintain state that data-plane hosts can influence:

- host-location hijacking and link fabrication poison the controller's view of hosts and topology [15], and SPHINX detects the resulting inconsistencies [16];
- topology tampering and its defenses were systematized by Skowyra et al. [17];
- data-to-control-plane saturation and flow-table attacks are addressed by AVANT-GUARD [19] and FloodGuard [20];
- races triggered by external events are shown by Xu et al. [21].

Broader surveys systematize the attack surface [18, 22]. In all these works the controller is the target and the defense is a fixed mechanism. A responder that *reacts* to alerts adds a new path from data-plane input to control-plane action, and that path is our subject.

Adversarial machine learning against intrusion detection. Realistic attacks on network intrusion detection must respect what the attacker can change [10, 12]. Adversarial examples against such detectors [11], including in SDN [13], aim at evasion. Li separates directly, indirectly and uncontrollable features for language-model traffic classifiers [14]. We adopt that separation but invert the goal: our adversary keeps the detector's verdict "attack" and changes the *identity* attached to it. Evasion defenses such as robust training do not help against this adversary, because the detector is right about the traffic and wrong only about whom to blame.

Costs of automated response. Several lines of work argue that automated blocking needs a harm model:

- response cost to legitimate users [8];
- cost-sensitive detection and response [9];
- the base-rate fallacy, which makes even accurate detectors produce many false alarms [23].

Crossfire shows that adversaries can exploit a network's own behaviour to amplify their impact [24]. When a language model is in the loop, attacker-controlled data reaches its input, a setting studied as indirect prompt injection [25]. These works motivate a harm model but do not quantify the harm an adversary can extract from a specific responder design.

Automated and language-model mitigation in SDN. Swileh and Zhang block the attacker's port [3]. ShieldGPT suggests DDoS mitigations [4]. El Hachimi et al. generate flow rules and retry on deployment rejection [5]. Network verifiers [26, 27] and rule-conflict analysis [28] can check a candidate rule before installation. None of these systems was evaluated against an adversary that targets the responder.

Gap. Table 1 summarizes the published coverage of the closest work. No prior study does all of the following:

1. models an adversary who forges evidence to aim the response;
2. treats the responder's own capacity as an attack target;

Table 1. Coverage of the closest prior work, from published descriptions (✓: yes; ∘: partial; –: no). (a) adversary targets the responder; (b) spoofed or forged evidence; (c) responder exhaustion or starvation; (d) harm to legitimate traffic quantified; (e) defenses with cost against genuine attacks; (f) language-model responders evaluated.

Line of work	(a)	(b)	(c)	(d)	(e)	(f)
SDN topology poisoning / saturation [15, 16, 19, 20]	–	✓	∘	–	✓	–
Adversarial evasion of intrusion detection [10, 13, 14]	–	–	–	–	∘	∘
Cost-sensitive intrusion response [8, 9]	–	–	–	∘	∘	–
Language-model SDN mitigation [3–5]	–	–	–	–	–	✓
This work	✓	✓	✓	✓	✓	✓

3. measures harm to legitimate traffic with attribution to responder rules;
4. evaluates defenses for both robustness and cost against genuine attacks, across classic and language-model responders.

3 Methodology

Figure 1 gives an overview. The adversary injects attack-class traffic with forged identifiers (top). The detection and response pipeline turns alerts into rules, optionally through the defenses D1–D4 and a pre-deployment verifier (middle). The SDN data plane and the evaluation harness measure induced harm, protected-service availability, responder load and defense cost (bottom).

3.1 System model

An OpenFlow network has switches $\mathcal{S}$, hosts $\mathcal{H}$, legitimate hosts $\mathcal{H}_\ell \subseteq \mathcal{H}$ and protected services $\mathcal{P} \subseteq \mathcal{H}$ (DNS, MQTT broker and gateway uplink). Each host h is attached to an access port $\mathrm{port}(h) = (s, i)$. A port is *shared* if several hosts attach to it through a legacy segment; we write $\mathcal{H}(s, i)$ for the hosts behind port (s, i).

A reactive controller performs LLDP discovery, MAC host tracking and shortest-path forwarding. A responder installs rules $r = (m, a, \sigma, \pi, \tau)$ in a security table evaluated before forwarding. Here m is an exact-match conjunction over header fields (including `in_port`, `eth_src`, `ip_src`, `ip_dst`, `ip_proto` and `tp_dst`), $a \in \{\texttt{drop}, \texttt{meter}(\rho), \texttt{redirect}\}$ is the action, $\sigma \subseteq \mathcal{S}$ the scope, π the priority and τ the time-to-live.

A detector labels flow records. Alerts are aggregated per key $\kappa = (\hat{y}, s, i)$ (predicted class, ingress switch, ingress port) into *incidents*, each with an evidence bundle E. The bundle contains the observed identifiers $\mathrm{ids}(E)$ (source IP and MAC addresses as seen on the wire), the confidence, and the targets. A *responder* is a map $\mathcal{R} : E \mapsto c$ from evidence to a candidate rule set. All responders share the detector, the aggregation and a single-server incident queue, so incidents are handled one at a time.

3.2 Adversary model

Position and knowledge. One compromised internal host h_a at port (s_a, i_a), as in LAN-SDN-NIDS [29]. The adversary has grey-box knowledge: it knows that an automated responder exists

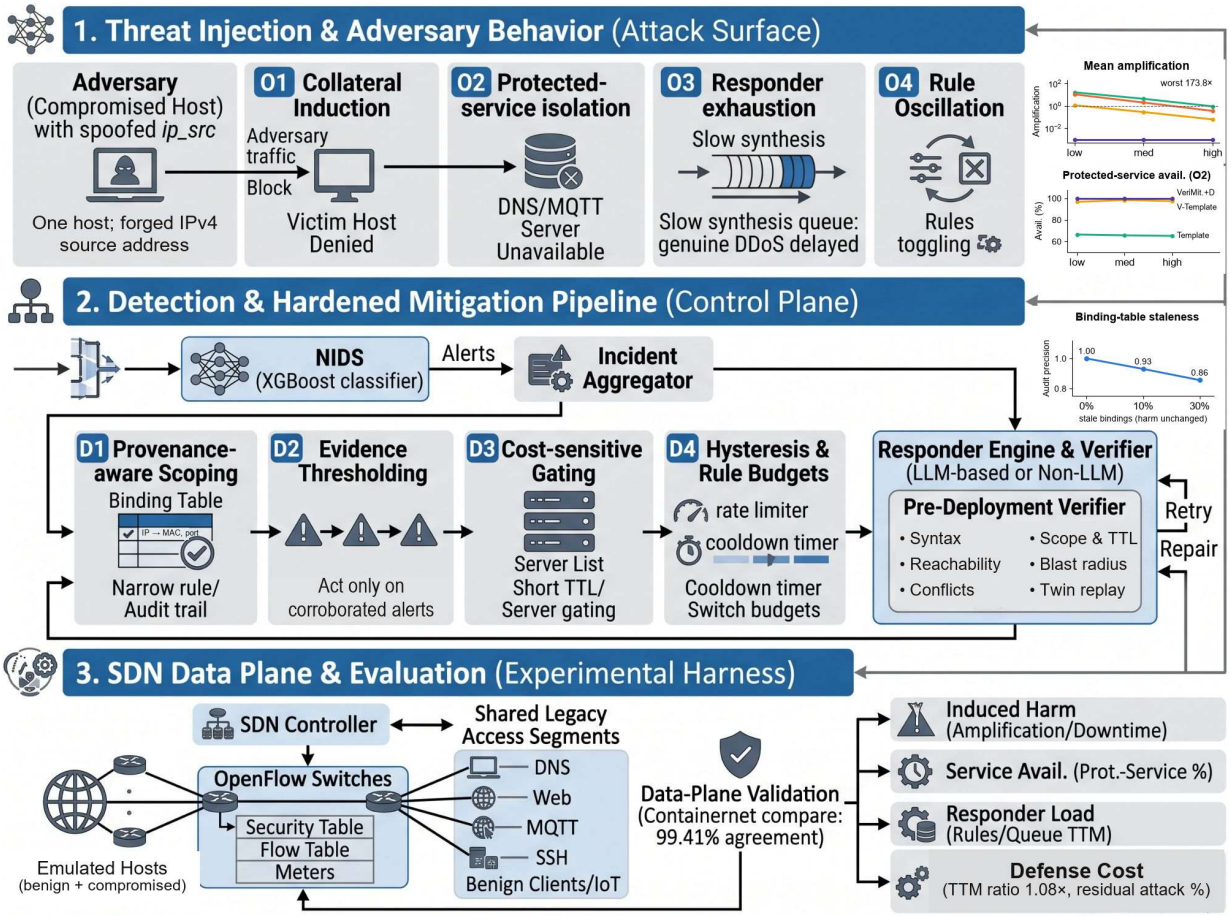

Figure 1. Study overview. (1) Threat injection: one compromised host forges its IPv4 source address to pursue objectives O1–O4. (2) Detection and hardened mitigation: a trace-driven XGBoost detector and an incident aggregator feed the responder engine. The engine is template-based or language-model-based, optionally behind a pre-deployment verifier, and is hardened by D1 provenance-aware scoping, D2 evidence thresholding, D3 cost-sensitive gating and D4 hysteresis with rule budgets. (3) Data plane and evaluation: OpenFlow switches with a security table, flow table and meters, a shared legacy access segment, and the four outcome families. The inset charts show measured results: mean amplification by adversary budget, O2 protected-service availability, and audit precision under stale bindings (Section 5).

and the general class of its policy, but not the detector or synthesizer weights, the protected-service list or the thresholds.

Capability. The adversary sends traffic from h_a only. Following the controllability taxonomy [14], it controls:

- the *directly controllable* field $F_D = \{\texttt{ip_src}\}$;
- the *indirectly controllable* properties $F_I = \{\text{rate } \lambda_a,\ \text{on/off schedule } \omega(t)\}$.

It cannot modify controller-side features, other hosts' traffic or the switches. Its frames carry its own MAC address and enter at (s_a, i_a). Every adversary flow has the statistical character of a real attack class: each is paired with a real LAN-SDN-NIDS DDoS or port-scan record, so the detector reacts as it would to a real attack.

Objectives. Let v be the forged source address. The four objectives are:

O1 *Collateral induction*: $v = \text{ip}(h_v)$ for a legitimate victim $h_v \in \mathcal{H}_\ell$.

Table 2. Adversary objectives, the controllable field each uses, and the primary metric.

Objective	Adversary behaviour (inside the emulated testbed)	Primary metric
O1 Collateral induction	attack-class traffic with `ip_src` set to a legitimate victim host	amplification $\mathcal{A}$; downtime $\mathcal{D}$
O2 Protected-service isolation	as O1, with `ip_src` set to a protected service	availability $\mathcal{V}_{\mathcal{P}}$
O3 Responder exhaustion	`ip_src` rotated every second through unused addresses; a genuine DDoS by other hosts starts 20 s later	rules installed; loop latency; TTM of the genuine attack
O4 Oscillation	as O1, active 6 s in every 40 s	amplification $\mathcal{A}$

O2 *Protected-service isolation*: $v = \mathrm{ip}(p)$ for $p \in \mathcal{P}$.

O3 *Responder exhaustion*: v_t rotates every second through unused addresses, while an independent set of compromised hosts starts a genuine DDoS 20 s later. The adversary aims to delay its mitigation.

O4 *Oscillation*: as O1 with $\omega(t) = [\,(t - t_0) \bmod 40\,\mathrm{s} < 6\,\mathrm{s}\,]$, so that rules expire and re-trigger.

The adversary budget λ_a is swept over three levels:

- *low*: 5 Mbit/s floods or 50 probes/s;
- *medium*: 20 Mbit/s or 200 probes/s;
- *high*: 60 Mbit/s or 800 probes/s.

Table 2 summarizes the objectives.

3.3 Harm metrics and attribution

Every installed rule r records its originating incident $\mathrm{inc}(r)$. Every incident records the ground-truth labels of the flow records that raised its alert. A rule is *attack-triggered* if at least one triggering record of its incident is attack traffic, and *false-positive-triggered* otherwise. Let $\mathcal{R}_{\mathrm{A}}$ be the set of attack-triggered rules. For each failed benign flow f, the harness identifies the causing rule $\mathrm{cause}(f)$: a drop or redirect on the flow's path, a meter that starved it, or a suppressed discovery link that removed its route. Let B_{adv} and N_{adv} be the bytes and packets the adversary sent.

Definition 1 (Harm metrics)**.**

$$\textit{Amplification:}\quad \mathcal{A} = \frac{L_{\mathrm{adv}}}{B_{\mathrm{adv}}}, \qquad L_{\mathrm{adv}} = \sum_{f \in \mathcal{F}_{\mathrm{ben}}} \mathrm{bytes_lost}(f)\,[\mathrm{cause}(f) \in \mathcal{R}_{\mathrm{A}}], \tag{1}$$

$$\textit{Downtime:}\quad \mathcal{D} = \frac{1000}{N_{\mathrm{adv}}} \sum_{h \in \mathcal{H}_\ell} \int_{T_{\mathrm{adv}}} \big[h \textit{ cannot resolve names due to some } r \in \mathcal{R}_{\mathrm{A}}\big]\, dt, \tag{2}$$

$$\textit{Protected availability:}\quad \mathcal{V}_{\mathcal{P}} = \frac{|\{\textit{successful probes to } \mathcal{P} \textit{ during } T_{\mathrm{adv}}\}|}{|\{\textit{probes to } \mathcal{P} \textit{ during } T_{\mathrm{adv}}\}|}. \tag{3}$$

$\mathcal{A} > 1$ means that the responder destroys more legitimate traffic than the adversary spends.

Algorithm 1 Per-rule harm attribution for one run

Input: installed rules with origin incidents; incidents with triggering records and ground truth; benign flow outcomes; adversary bytes B_{adv} and packets N_{adv}
Output: $\mathcal{A}$, $\mathcal{D}$, collateral, false-positive harm

$\mathcal{R}_{\text{A}} \leftarrow \{r : \exists x \in \text{trig}(\text{inc}(r)),\ \text{truth}(x) \neq \text{Normal}\}$
$L_{\text{adv}} \leftarrow 0;\ L_{\text{FP}} \leftarrow 0;\ n_{\text{coll}} \leftarrow 0$
for each benign flow f started during the attack period **do**
    if f failed **and** cause(f) is a responder rule r **then**
        $n_{\text{coll}} \leftarrow n_{\text{coll}} + 1$
        if $r \in \mathcal{R}_{\text{A}}$ **then** $L_{\text{adv}} \leftarrow L_{\text{adv}} + \text{bytes_lost}(f)$
        else $L_{\text{FP}} \leftarrow L_{\text{FP}} + \text{bytes_lost}(f)$
        end if
    end if
end for
for each 1-s step $t \in T_{\text{adv}}$ and host $h \in \mathcal{H}_\ell$ **do**
    if the DNS probe of h fails and its causing rule $\in \mathcal{R}_{\text{A}}$ **then** $down \leftarrow down + 1$
    end if
end for
return $\mathcal{A} = L_{\text{adv}}/B_{\text{adv}};\ \mathcal{D} = 1000 \cdot down/N_{\text{adv}};\ \text{collateral} = n_{\text{coll}}/|\mathcal{F}_{\text{ben}}|;\ L_{\text{FP}}$

Harm from false-positive-triggered rules is excluded from $\mathcal{A}$ and $\mathcal{D}$ and reported separately (Supplementary Table S2). Responder-attributable collateral counts benign flows broken by *any* responder rule.

For exhaustion we report the rules installed, the peak table size, the 95th-percentile loop latency, and the time to mitigate (TTM) of the concurrent genuine attack:

$$\text{TTM} = \inf\{t \geq t_0 : a_{\text{del}}(t) \leq \tfrac{1}{2} a_{\text{off}}(t)\} - t_0, \tag{4}$$

where t_0 is attack onset and a_{off}, a_{del} are the offered and delivered attack rates. TTM is censored at the 60-s attack duration.

Defense cost is the change in TTM and in the residual attack ratio $\int a_{\text{del}} / \int a_{\text{off}}$ under genuine attacks without an adversary. *Auditability* is the precision and recall of the spoof-suspected flag (D1) against ground truth. Algorithm 1 gives the attribution procedure.

3.4 Responders under test

The seven responders cover the state-of-the-art designs:

R1 Port-drop drops all traffic on the ingress port for 30 s, as in Swileh and Zhang [3].

R2 Template installs one classic rule per class without verification. For floods and scans it "shuns" the observed source IP on all switches with $\tau = 300$ s, as signature-driven intrusion prevention does.

R2v V-Template passes a ladder of increasingly narrow templates through a pre-deployment verifier and a replay twin. It is a strong non-language-model responder.

R3 LLM synthesizes rules with a fine-tuned Qwen2.5-1.5B model [30, 31]. The controller only rejects rules that do not compile, which reproduces the retry loop of El Hachimi et al. [5].

R4 VeriMitigate is the same synthesizer behind a verifier. The verifier checks syntax and

class semantics, time-to-live, scope, conflicts [28], protected-service reachability including link-discovery side effects, and a blast-radius bound of 5% of legitimate (host, service) pairs [26]. It is followed by a 5-s traffic-replay twin and counterexample-guided repair.

R5 VeriMitigate+D and **R2v+D** are R4 and R2v with defenses D1–D4.

3.5 Defenses

The defenses wrap *any* responder $\mathcal{R}$ (Algorithm 2). Let $w(\kappa, t)$ be the number of consecutive 1-s windows up to t in which key κ was alerted.

D1 Provenance-aware scoping. During warm-up the controller learns a binding table $\beta : \text{IP} \to (\text{MAC}, s, i)$, as DHCP snooping or IP source guard would. An incident with ingress (s_E, i_E) is *spoof-suspected* if

$$\text{spoof}(E) = \exists\, x \in \text{ids}(E) :\ \beta(x) \text{ undefined } \vee\ \beta(x).(s,i) \neq (s_E, i_E). \tag{5}$$

If so, every candidate rule is rewritten as $m_r \leftarrow m_r \cup \{\texttt{in_port}{=}i_E\}$, $\sigma_r \leftarrow \{s_E\}$.

D2 Evidence thresholding. An incident enters the queue only if $w(\kappa, t) \geq 3$, or $w(\kappa, t) \geq 2$ when its confidence is at least 0.995.

D3 Cost-sensitive gating. A rule whose match names a server address (IP or MAC) requires $w(\kappa, t) \geq 5$, and its time-to-live is capped at 30 s.

D4 Hysteresis and rule budgets. D4 combines three limits:

- at most $B = 6$ installs per switch per 60 s;
- a minimum time-to-live $\tau_{\min} = 60$ s;
- a cooldown $T_c = 30$ s after a rule for key κ expires, during which κ cannot trigger again.

When the budget is exhausted, pending rules are aggregated into one verified ingress-port rate limit.

Proposition 1 (Confinement and fail-safety of D1)**.** *Let c' be the D1 rewrite of a rule set c for an incident with ingress (s_E, i_E) at an access port. Two properties hold:*

(i) Every rule in c' matches only packets that enter the network at (s_E, i_E). Hence, for a fixed forwarding state, the legitimate hosts whose traffic c' can drop at its source are contained in $\mathcal{H}(s_E, i_E)$.

(ii) For every header $\mathbf{h}$ and switch s, $\text{match}(r', \mathbf{h}, s) \Rightarrow \text{match}(r, \mathbf{h}, s)$, so c' never blocks a packet that c would not block.

Proof. For (ii), the rewrite only adds a conjunct to m_r and shrinks σ_r, so the rewritten rule's match set is a subset of the original. For (i), on s_E the conjunct $\texttt{in_port} = i_E$ restricts matches to frames received on port i_E, which at an access port are sent by the hosts in $\mathcal{H}(s_E, i_E)$. Traffic from other legitimate hosts, including the victim's own traffic under O1 and the protected service's traffic under O2, enters at other ports and is never matched. Traffic *towards* a victim is matched only if it enters at i_E. □

Algorithm 2 Hardened responder intake with defenses D1–D4

Input: incident (κ, E) at time t; base responder $\mathcal{R}$; binding table β; verifier V; budget state
Output: installed rules (possibly none)

if $w(\kappa,t) < 3$ **and not** $(w(\kappa,t) \geq 2$ **and** $E.\text{conf} \geq 0.995)$ **then return** ▷ D2 corroboration
end if
if κ in cooldown (expired $< T_c$ ago) **then return** ▷ D4 hysteresis
end if
$susp \leftarrow \text{spoof}(E)$ ▷ D1, Eq. (5); logged for audit
$c \leftarrow \mathcal{R}(E)$ ▷ template ladder or language-model loop, incl. its verifier
for $r \in c$ **do**
    if *susp* **then** $m_r \leftarrow m_r \cup \{\texttt{in_port}{=}E.i\}$; $\sigma_r \leftarrow \{E.s\}$
    end if ▷ D1 rewrite
    if m_r names a server address **then** ▷ D3 cost gating
        if $w(\kappa,t) < 5$ **then** drop r from c; **continue**
        end if
        $\tau_r \leftarrow \min(\tau_r, 30\,\text{s})$
    end if
    $\tau_r \leftarrow \max(\tau_r, \tau_{\min})$ ▷ D4 minimum time-to-live
end for
if $V(c).ok$ **is false then return**
end if ▷ re-verify after rewriting
if installs on $E.s$ in last $60\,\text{s} + |c| > B$ **then** ▷ D4 budget
    $c \leftarrow \{\text{verified ingress meter on } (E.s, E.i)\}$
end if
INSTALL(c); record $\text{inc}(r) \leftarrow (\kappa, E)$ for $r \in c$

Proposition 1 has two consequences that we test empirically.

- Under D1, O1 and O2 can only harm hosts that share the adversary's access port.
- A *stale* binding only causes a legitimate incident to be flagged as spoof-suspected, which by (ii) makes its rules narrower, never wider. Staleness should therefore degrade auditability, not safety.

Starvation. Let $\bar{t}_{\text{syn}}$ be the mean synthesis time per incident and λ_I the rate of incidents admitted to the single-server queue. The queue is unstable when the utilization satisfies

$$\varrho = \lambda_I \, \bar{t}_{\text{syn}} \geq 1. \tag{6}$$

Under O3 every rotated identity that is not covered by an installed rule raises a new incident on the adversary's key. For a template responder $\bar{t}_{\text{syn}} \approx 0$ and $\varrho \ll 1$. For a language-model responder on a consumer graphics processor $\bar{t}_{\text{syn}} \approx 10\,\text{s}$, so $\varrho \geq 1$ already at one admitted incident every ten seconds. D2 and D3 do not change λ_I for a sustained adversary. D4's cooldown and budgets lower it, but they do not shorten $\bar{t}_{\text{syn}}$.

4 Experimental setup

4.1 Dataset and detector

LAN-SDN-NIDS [29, 32] provides labelled flow records with 39 features, including OpenFlow control-plane features, on five topologies. The detector is XGBoost [33], trained leave-one-

topology-out: for a run on topology T it never saw T. Every emulated flow, benign or adversarial, is paired with a real held-out record of matching class and protocol, and the detector's decision on that record is used. Detector errors inside the loop are therefore those of a real classifier, including false positives on benign ARP traffic.

4.2 Testbed and real data-plane validation

Emulator. The closed-loop harness is a flow-level OpenFlow emulator with a Ryu-like controller. It models the security and forwarding tables with priorities, time-to-live values and meters; link capacities of 100 Mbit/s at the access and 1 Gbit/s in the core; and Packet-In (500/s) and flow-table (1,000 entries) limits. It runs six topologies, the five LAN-SDN-NIDS shapes and an IoT gateway, each with a shared legacy access segment of three hosts and six service hosts. Benign clients and IoT devices generate DNS, web, gateway, MQTT, SSH, FTP, CoAP and RTP flows. Runs last 120 s, and the adversary is active from 30 to 90 s (O4: 30 to 120 s). Synthesis latency is the measured single-request latency of the synthesizer on an NVIDIA GTX 1060.

Real data plane. We validated the emulator's data-plane predictions on Containernet [34] with Docker hosts, Open vSwitch 2.17 and a Ryu 4.34 controller running the same two-table pipeline. 180 responder rule sets, recorded under genuine attacks and under objectives O1, O2 and O4, were installed through Ryu's REST interface. Every legitimate (host, service) pair was then probed with real TCP/UDP traffic (Table 10).

4.3 Design, baselines and hypotheses

Factorial. The main design is full factorial: 7 responders $\times$ 4 objectives $\times$ 3 budgets $\times$ 6 topologies. It uses five seeds for non-language-model responders and three for the language-model responders, giving 2088 runs, plus a no-response reference. Paired tests always use matched (objective, budget, topology, seed) cells. Every run is deterministic given its cell, because seeds are process-independent and all charged latencies are measured offline or fixed.

Ablations and sensitivity.

- Single defenses added to R2v: all objectives and budgets, five seeds, 1,440 runs.
- Single defenses added to R4: medium budget, three seeds, 288 runs.
- Binding-staleness study: R2v+D with 10% and 30% of client bindings pointing to a wrong port.
- Defense cost on genuine attacks: 5 classes $\times$ 6 topologies $\times$ 5 seeds without an adversary.

Baselines and the state of the art. R1 and R3 re-implement the response policies of the two closest language-model SDN systems [3, 5], and R2 the classic shun-the-source template. Because none of these systems was evaluated against an adversary, we compare them under identical conditions in one harness rather than against published numbers.

Pre-registered hypotheses. The following were fixed before the factorial was run:

- **H1**: R1–R3 exhibit $\mathcal{A} > 1$ for at least one objective.
- **H2**: R4 reduces O2 to near zero but does not eliminate O1.

Table 3. Harm induced by the adversary, per responder and objective (means over 3 budgets × 6 topologies × 5 seeds; 3 seeds for R3–R5). Amp.: legitimate bytes lost to adversary-triggered responder rules per adversary byte; Down: legitimate-host downtime (s) per 1,000 adversary packets; Prot.: protected-service availability (%); Coll.: share of benign flows broken by any responder rule, including rules triggered by detector false positives (%).

	O1 Collateral induction				O2 Protected-service isolation			
Responder	Amp.	Down	Prot.	Coll.	Amp.	Down	Prot.	Coll.
R1 Port-drop	1.58	2.28	96.5	3.80	2.18	3.17	96.1	4.06
R2 Template	4.75	6.89	89.9	11.06	2.46	31.72	65.8	23.32
R2v V-Template	0.33	0.00	98.6	1.74	0.00	0.00	97.6	1.57
R2v+D V-Template+D	0.00	0.00	99.9	0.08	0.00	0.00	100.0	0.00
R3 LLM	0.47	0.49	99.3	0.75	0.00	0.00	99.2	0.86
R4 VeriMitigate	0.00	0.00	99.3	0.84	0.00	0.00	99.2	0.89
R5 VeriMitigate+D	0.00	0.00	99.9	0.12	0.00	0.00	100.0	0.00
	O3 Responder exhaustion				O4 Oscillation			
Responder	Amp.	Down	Prot.	Coll.	Amp.	Down	Prot.	Coll.
R1 Port-drop	3.59	5.21	91.5	8.89	10.13	14.05	96.2	4.11
R2 Template	0.00	0.00	98.5	1.66	21.74	28.22	89.9	10.84
R2v V-Template	0.01	0.15	98.3	1.88	1.65	0.02	98.8	1.55
R2v+D V-Template+D	0.01	0.09	99.7	0.46	0.00	0.01	100.0	0.03
R3 LLM	0.00	0.01	99.9	0.10	1.50	2.03	99.1	0.97
R4 VeriMitigate	0.00	0.01	100.0	0.06	0.00	0.00	99.1	0.94
R5 VeriMitigate+D	0.00	0.01	100.0	0.05	0.00	0.00	100.0	0.00

- **H3**: R5 reduces O1 to $\mathcal{A} < 0.01$ at a TTM cost below 2× under genuine attacks.

Statistics. We fit linear mixed-effects models of each metric, log-transformed where noted. Responder, objective and their interaction are fixed effects, with a random intercept per topology. Pairwise contrasts use paired Wilcoxon signed-rank tests [35] with Holm correction [36] and the matched-pairs rank-biserial correlation as effect size. Intervals are 95% bootstrap intervals [37].

5 Results and discussion

5.1 State-of-the-art responders can be turned against legitimate hosts

Responders that act on the identifiers in an alert without checking them were cheap to abuse (Table 3, Fig. 2).

- **R2 Template.** The classic template destroyed 4.75 bytes of legitimate traffic per adversary byte under O1 and 21.74 under oscillation (O4). Under O2 it disconnected the protected service itself: availability fell to 65.8%, and legitimate hosts lost 31.72 s of name resolution per thousand adversary packets.
- **R1 Port-drop** [3]. Port blocking cannot be pointed at a victim by spoofing, because it acts on the physical port. It still amplified the adversary's effort ($\mathcal{A} = 1.58$–10.13) whenever the adversary sat on a shared segment and took its neighbours down with it.

Harm per adversary byte was largest at the *lowest* budget. A quiet adversary that barely crosses the detection threshold obtains the same blocking rule as a loud one, so a responder that acts on threshold crossings charges nothing for being quiet.

Why. R2's shun rule matches `ip_src` $= v$ on *all* switches. By construction of O1 and O2, v is

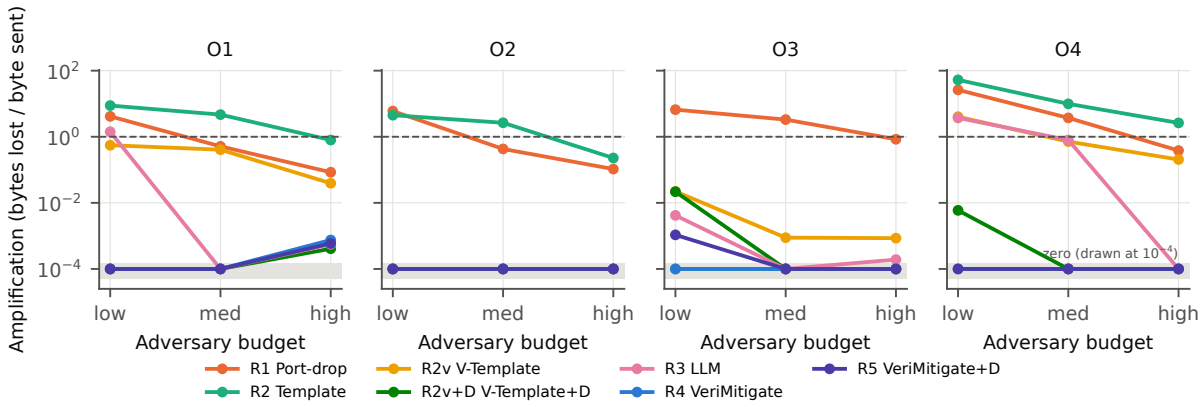

Figure 2. Amplification by adversary budget. Legitimate bytes lost to adversary-triggered responder rules per adversary byte, for each objective and responder (log scale). The dashed line marks break-even; zero values are drawn at 10^{-4}.

the victim's address, so every packet the victim sends is dropped network-wide. The adversary's cost is a few seconds of attack-class traffic, while the victim's loss lasts the rule's 300-s time-to-live. R1 avoids this mapping by acting on the port, but pays for the coarseness of the port on shared segments.

5.2 Verification removes service isolation but not collateral induction

Passing the same templates through a pre-deployment verifier (R2v) rejected the O2 rule that would have cut DNS or the broker. O2 amplification fell to 0.00, and availability remained at 97.6%. Collateral induction was reduced but not eliminated (O1: 0.33), and oscillation persisted (O4: 1.65). The reason is structural. Cutting a single victim changes only a few legitimate (host, service) pairs and fits inside any blast-radius budget that still allows an attacker to be blocked. Under oscillation, each burst re-triggers the same admissible rule.

5.3 Language-model responders are rarely fooled, but not never

The fine-tuned language-model responder (R3) had learned from verifier-approved examples, which bind identifiers to the ingress port. It was therefore abused in only 7 of 216 adversarial runs, against 144 for templates and 70 for port blocking (Table 4). Averages hide the tail. In those 7 runs the victim shared the adversary's legacy segment, and a drop on the spoofed address at the adversary's port cut it off ($\mathcal{A}$ up to 66.8). This is precisely the residual case of Proposition 1(i). With the verifier (R4), such candidates failed the reachability check or the replay and were replaced by a rate limit, and no objective produced harm in more than one run (largest $\mathcal{A} = 0.0$). The mixed-effects model does not separate R3 and R4 (Supplementary Table S3): the difference lies in worst cases, not averages.

5.4 Exhaustion starves slow responders

Rotating the spoofed address (O3) did little direct damage, because the addresses belonged to nobody, but it attacked the responder's capacity (Table 5). The two kinds of responder failed in

Table 4. Frequency and severity of successful abuse: number of adversarial runs (of 54 per objective; seeds 0–2 for all responders) in which adversary-triggered rules destroyed legitimate traffic (amplification $> 10^{-3}$), and the largest amplification observed over all objectives.

Responder	O1	O2	O3	O4	Worst amplification
R1 Port-drop	10	17	31	12	201.0
R2 Template	53	39	0	52	173.8
R2v V-Template	8	0	5	9	111.5
R2v+D V-Template+D	1	0	2	0	0.1
R3 LLM	2	0	3	2	66.8
R4 VeriMitigate	1	0	0	0	0.0
R5 VeriMitigate+D	1	0	1	0	0.0

Table 5. O3 responder exhaustion (means over budgets, topologies and seeds). The genuine DDoS starts 20 s after the adversary; its TTM is censored at 60 s.

Responder	Incidents	Rules inst.	Peak table	Loop p95 (s)	Genuine TTM (s)	Genuine resid. (%)
R1 Port-drop	8.1	8.1	4.1	0.0	1.1	5.6
R2 Template	59.9	59.9	59.9	0.0	1.1	3.1
R2v V-Template	9.0	5.8	5.8	3.8	3.2	9.5
R2v+D V-Template+D	4.2	4.0	4.0	2.8	4.2	10.9
R3 LLM	3.4	2.8	2.8	72.3	60.0	99.9
R4 VeriMitigate	2.1	2.1	2.1	70.1	60.0	100.0
R5 VeriMitigate+D	3.8	3.8	3.8	31.2	22.6	62.7

opposite ways.

- **Templates.** The template installed 59.9 rules per run, one per spoofed address, filling switch tables with useless entries. Its synthesis is instantaneous, so it still contained the genuine DDoS within 1.1 s.
- **Language-model responders.** They installed few rules, but every spoofed identity occupied the synthesizer for about ten seconds, the case $\varrho \geq 1$ of Eq. (6). The 95th-percentile loop latency rose to 70.1 s, and the genuine DDoS was *never* mitigated by R3 or R4.

Verification does nothing against starvation. Only D4 relieved it (R5: genuine TTM 22.6 s).

5.5 Four defenses close the gap at a small cost

With D1–D4, both hardened responders reduced harm to essentially zero under every objective. Amplification was at most 0.01, and O2 availability was 100.0%. The hardened responders also eliminated the harm caused by detector false positives (Supplementary Table S2).

Cost. On genuine attacks without an adversary (Table 6), the defenses added about one second to the median TTM, the extra alert windows required by D2. Residual attack traffic rose by a few points (V-Template 8.9% → 14.9%; VeriMitigate 31.0% → 34.1%). The share mitigated changed by less than one percentage point, and collateral fell to zero because isolated false positives no longer reached the rule table. The TTM ratio of R5 to R4 was 1.08.

Table 6. Defense cost under genuine attacks (genuine-attack grid: 5 classes × 6 topologies × 5 seeds, no adversary).

Responder	Mitig. (%)	TTM (s)	Residual (%)	Collateral (%)	FP incidents/run
R2v V-Template	96.7	3.1	8.9	1.34	0.43
R2v+D V-Template+D	96.0	4.1	14.9	0.00	0.00
R4 VeriMitigate	95.3	13.6	31.0	0.81	0.38
R5 VeriMitigate+D	94.7	14.7	34.1	0.00	0.00

5.6 Pre-registered hypotheses

- **H1 was supported.** Port blocking exceeded break-even under every objective, templates under every objective except exhaustion, and the unverified language-model responder under oscillation ($\mathcal{A} = 1.50$).
- **H2 was not supported as stated.** R4 eliminated O2 (availability 99.2%) and, contrary to expectation, O1 as well. Its weakness was starvation, which H2 did not anticipate.
- **H3 was supported**, with O1 amplification 0.00 and a TTM ratio of 1.08.

Pairwise tests are in Supplementary Table S4.

5.7 Discussion: design guidelines

1. **Bind every blocking rule to the physical ingress port** of the evidence, and treat unbound identifiers as unauthenticated (D1). By Proposition 1 this confines adversary-induced cuts to the adversary's own port, and it fails safe.
2. **Verify protected-service reachability before installation**, including control-plane side effects, and keep a blast-radius budget.
3. **Bound the work any single port can create for the responder** (D4), and key incidents by ingress rather than by claimed identity. Slow responders need this most.
4. **Require corroboration before blocking** (D2). It costs about one second against genuine attacks and removes false-positive collateral.
5. **Log the provenance decision** for audit.

6 Ablation and analysis

Which defense does what. Table 7 adds the defenses one at a time to the verified template ladder. Provenance-aware scoping alone removed collateral induction (O1: 0.000 instead of 0.333) and oscillation (O4: 0.004 instead of 1.652), as Proposition 1 predicts. Evidence thresholding, cost gating and budgets alone did not.

Defenses on the language-model responder. R4 was already robust to O1, O2 and O4, so the defenses differed only under exhaustion (Table 8). With no defense, or with D1, D2 or D3 alone, the genuine DDoS was never mitigated. D4 alone brought its median TTM to 21.6 s and its residual traffic to 55.1%. This is consistent with Eq. (6): only D4 lowers λ_I, and none of the defenses shortens $\bar{t}_{\text{syn}}$. The damage was halved, not removed.

Table 7. Which defense does what: single defenses added to the verified template ladder (all budgets, 6 topologies × 5 seeds). Mean amplification per objective; O3 columns give rules installed per run and the genuine DDoS's median TTM.

Variant	O1 amp.	O2 amp.	O3 amp.	O4 amp.	O3 rules	O3 genuine TTM (s)
R2v (no defense)	0.333	0.000	0.008	1.652	5.8	3.2
R2v + D1 provenance	0.000	0.000	0.008	0.004	5.8	3.2
R2v + D2 threshold	0.333	0.000	0.009	2.061	4.6	4.2
R2v + D3 cost gating	0.333	0.000	0.008	1.652	5.8	3.2
R2v + D4 hysteresis/budget	0.333	0.000	0.000	1.651	4.2	3.2
R2v + D1–D4	0.000	0.000	0.007	0.002	4.0	4.2

Table 8. Defense ablation on VeriMitigate (medium budget, 6 topologies × 3 seeds). Adversary-induced amplification was zero for every variant and objective at this budget; the columns show where the variants differ: protected-service availability under O2 and the fate of the concurrent genuine DDoS under exhaustion (O3; TTM censored at 60 s).

Variant	O2 prot. (%)	O3 genuine TTM (s)	O3 genuine residual (%)	O3 loop p95 (s)
R4 (no defense)	99.7	60.0	100.0	70.5
R4 + D1 provenance	99.7	60.0	100.0	71.5
R4 + D2 threshold	100.0	60.0	99.3	68.6
R4 + D3 cost gating	99.7	60.0	100.0	70.5
R4 + D4 hysteresis/budget	99.7	21.6	55.1	29.1
R5 = R4 + D1–D4	100.0	22.4	58.7	29.5

Dependence on the binding table. With 10% and 30% of client bindings pointing to a wrong port (Table 9), harm and efficacy did not change. As Proposition 1(ii) predicts, a stale binding only makes a rule narrower. What degrades is the audit trail: the precision of the spoof-suspected flag fell from 1.00 to 0.93 and 0.86.

Fidelity of the data-plane predictions. The real Containernet network agreed with the emulator on 99.41% of 14,292 reachability tests (Cohen's $\kappa = 0.953$; Table 10). It never cut a pair that the emulator predicted reachable. The only disagreements came from ARP caching and vanished with cold caches (100.00%).

7 Limitations and threats to validity

Internal validity. All responders face identical, deterministic scenarios with shared detection and queueing. Harm attribution (Algorithm 1) prevents detector false positives from being counted as adversary harm. The language-model responders use three seeds instead of five, which lowers power for rare events. We therefore report frequencies and worst cases (Table 4) alongside means.

Construct validity. Amplification weighs all legitimate bytes equally, and downtime counts only name resolution. Protected-service availability and collateral complement them. The budget levels and the O4 duty cycle are representative choices, not optima for the adversary.

External validity. Four factors limit generalization:

- *Fidelity.* The emulator is flow-level. Its reachability predictions were validated on a real data plane, but packet-level dynamics and absolute timings were not, so relative comparisons between responders are the claim.

Table 9. Sensitivity of provenance-aware scoping (R2v+D) to stale IP bindings (a fraction of clients whose binding points to a wrong port). Adversarial rows: medium budget, 6 topologies × 5 seeds; genuine-attack rows: genuine-attack grid without adversary.

Stale bindings	0%	10%	30%
O1 amplification	0.0000	0.0000	0.0000
O2 amplification	0.0000	0.0000	0.0000
O3 amplification	0.0000	0.0000	0.0000
O4 amplification	0.0000	0.0000	0.0000
Genuine: mitigated (%)	96.0	96.0	96.0
Genuine: residual (%)	14.9	14.9	14.9
Genuine: collateral (%)	0.00	0.00	0.00
Spoof-flag precision	1.00	0.93	0.86
Spoof-flag recall	1.00	1.00	1.00

Table 10. Validation on a real data plane: Containernet with Docker hosts, Open vSwitch 2.17 (kernel datapath) and a Ryu controller with the same two-table pipeline. Responder rule sets recorded in the emulator were installed through Ryu's REST interface, and every legitimate (host, service) pair was probed with real TCP/UDP traffic. Agreement: share of pairs whose reachability matched the emulator's prediction; "missed cuts": pairs the emulator predicted reachable but the real network cut; "cold": the same, with ARP caches flushed before probing.

Responder	Rule sets	Pairs	Agreement (%)	κ	Missed cuts	Agreement, cold (%)
Port-drop	30	2382	100.00	1.000	0	100.00
Template	30	2382	100.00	1.000	0	100.00
V-Template	30	2382	100.00	1.000	0	100.00
LLM (unverified)	30	2382	97.23	0.692	0	100.00
VeriMitigate	30	2382	99.24	0.841	0	100.00
V-Template+D	15	1191	100.00	1.000	0	100.00
VeriMitigate+D	15	1191	100.00	1.000	0	100.00
All	180	14292	99.41	0.953	0	100.00

- *Adversary.* The adversary controls one host and spoofs only the IPv4 source address. MAC forgery on shared segments, collusion across ports, and feature-adaptive traffic [14] are natural extensions.
- *Bindings.* D1 assumes bindings learned before the attack. An adversary present during warm-up could poison them, and the staleness study shows graceful degradation but not robustness to poisoning.
- *Model and hardware.* One small model ran on one consumer graphics processor. Faster hardware would raise the rate an adversary must sustain (Eq. (6)) but would not remove the vulnerability.

Conclusion validity. Hypotheses were fixed in advance. Tests are paired and Holm-corrected, effects come with rank-biserial correlations and bootstrap intervals, and non-supported hypotheses are reported as such.

8 Conclusion and future work

Automated mitigation makes the responder part of the attack surface. With one compromised host and a forged source address, an adversary can aim responders that act on alert identifiers at

legitimate hosts and protected services, and can starve slow responders so that they never reach a genuine attack. Pre-deployment verification removes service isolation. Provenance-aware scoping confines induced harm to the adversary's own port. Corroboration and rule budgets remove false-positive collateral and halve starvation. Together these measures cost about one second of time to mitigate, and we recommend them as defaults for any SDN responder that acts without a human in the loop.

Future work will pursue four directions:

1. adversaries that poison the binding table or adapt to the detector;
2. priority scheduling and speculative synthesis that remove the starvation channel;
3. validation of timing on hardware switches;
4. extension of the harm model to multi-controller and multi-tenant deployments.

Declarations

Data availability. LAN-SDN-NIDS is publicly available under CC BY 4.0 at `https://doi.org/10.21950/QMAXKP`. All per-run result logs and the language-model completion cache generated in this study are available in a public GitHub repository; the link is withheld from this anonymized manuscript for double-blind review and is provided on the title page. The cache reproduces every model output without a graphics processor.

Code availability. The evaluation harness, responders, defenses, experiment grids and the real data-plane validation harness (Containernet, Open vSwitch, Ryu) are available in the same repository under the MIT licence. Adversary generators are parameterized components of the emulated testbed only.

Competing interests. The authors declare no competing interests.

References

[1] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: Enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008. doi: 10.1145/1355734.1355746.

[2] Diego Kreutz, Fernando M. V. Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015. doi: 10.1109/jproc.2014.2371999.

[3] Mohammed N. Swileh and Shengli Zhang. Proactive DDoS detection and mitigation in decentralized software-defined networking via port-level monitoring and zero-training large language models, 2025. URL `https://arxiv.org/abs/2511.00460`. arXiv preprint.

[4] Tongze Wang, Xiaohui Xie, Lei Zhang, Chuyi Wang, Liang Zhang, and Yong Cui. Shieldgpt:

An llm-based framework for ddos mitigation. In *Proceedings of the 8th Asia-Pacific Workshop on Networking*, pages 108–114, 2024. doi: 10.1145/3663408.3663424.

[5] Anouar El Hachimi, Nicola Di Cicco, Memedhe Ibrahimi, Francesco Musumeci, and Massimo Tornatore. Flow-rule generation for sdn using llms with retry-based deployment validation. In *2025 21st International Conference on Network and Service Management (CNSM)*, pages 1–5, 2025. doi: 10.23919/cnsm67658.2025.11297487.

[6] Franziska Lichtblau, Florian Streibelt, Thorben Krüger, Philipp Richter, and Anja Feldmann. Detection, classification, and analysis of inter-domain traffic with spoofed source IP addresses. In *Proceedings of the 2017 Internet Measurement Conference*, pages 86–99, 2017. doi: 10.1145/3131365.3131367.

[7] Paul Ferguson and Daniel Senie. Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing. Best Current Practice RFC 2827 (BCP 38), IETF, May 2000. URL `https://www.rfc-editor.org/rfc/rfc2827`.

[8] T. Toth and C. Kruegel. Evaluating the impact of automated intrusion response mechanisms. In *18th Annual Computer Security Applications Conference, 2002. Proceedings.*, pages 301–310, 2002. doi: 10.1109/csac.2002.1176302.

[9] Wenke Lee, Wei Fan, Matthew Miller, Salvatore J. Stolfo, and Erez Zadok. Toward cost-sensitive modeling for intrusion detection and response. *Journal of Computer Security*, 10 (1-2):5–22, 2002. doi: 10.3233/jcs-2002-101-202.

[10] Giovanni Apruzzese, Mauro Andreolini, Luca Ferretti, Mirco Marchetti, and Michele Colajanni. Modeling realistic adversarial attacks against network intrusion detection systems. *Digital Threats: Research and Practice*, 3(3):1–19, 2022. doi: 10.1145/3469659.

[11] Ryan Sheatsley, Nicolas Papernot, Michael J. Weisman, Gunjan Verma, and Patrick McDaniel. Adversarial examples for network intrusion detection systems. *Journal of Computer Security*, 30(5):727–752, 2022. doi: 10.3233/jcs-210094.

[12] Fabio Pierazzi, Feargus Pendlebury, Jacopo Cortellazzi, and Lorenzo Cavallaro. Intriguing properties of adversarial ml attacks in the problem space. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1332–1349, 2020. doi: 10.1109/sp40000.2020.00073.

[13] James Aiken and Sandra Scott-Hayward. Investigating adversarial attacks against network intrusion detection systems in sdns. In *2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 1–7, 2019. doi: 10.1109/nfv-sdn47374.2019.9040101.

[14] Zhenpeng Li. Controllability-aware adversarial examples against LLM-based network traffic classifiers, 2026. URL `https://arxiv.org/abs/2607.07739`. arXiv preprint.

[15] Sungmin Hong, Lei Xu, Haopei Wang, and Guofei Gu. Poisoning network visibility in software-defined networks: New attacks and countermeasures. In *Proceedings 2015 Network and Distributed System Security Symposium*, 2015. doi: 10.14722/ndss.2015.23283.

[16] Mohan Dhawan, Rishabh Poddar, Kshiteej Mahajan, and Vijay Mann. Sphinx: Detecting security attacks in software-defined networks. In *Proceedings 2015 Network and Distributed System Security Symposium*, 2015. doi: 10.14722/ndss.2015.23064.

[17] Richard Skowyra, Lei Xu, Guofei Gu, Veer Dedhia, Thomas Hobson, Hamed Okhravi, and James Landry. Effective topology tampering attacks and defenses in software-defined networks. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 374–385, 2018. doi: 10.1109/dsn.2018.00047.

[18] Changhoon Yoon, Seungsoo Lee, Heedo Kang, Taejune Park, Seungwon Shin, Vinod Yegneswaran, Phillip Porras, and Guofei Gu. Flow wars: Systemizing the attack surface and defenses in software-defined networks. *IEEE/ACM Transactions on Networking*, 25(6):3514–3530, 2017. doi: 10.1109/tnet.2017.2748159.

[19] Seungwon Shin, Vinod Yegneswaran, Phillip Porras, and Guofei Gu. Avant-guard: Scalable and vigilant switch flow management in software-defined networks. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security - CCS '13*, pages 413–424, 2013. doi: 10.1145/2508859.2516684.

[20] Haopei Wang, Lei Xu, and Guofei Gu. Floodguard: A dos attack prevention extension in software-defined networks. In *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 239–250, 2015. doi: 10.1109/dsn.2015.27.

[21] Lei Xu, Jeff Huang, Sungmin Hong, Jialong Zhang, and Guofei Gu. Attacking the brain: Races in the SDN control plane. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 451–468, Vancouver, BC, 2017. USENIX Association. URL `https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/xu-lei`.

[22] Sandra Scott-Hayward, Sriram Natarajan, and Sakir Sezer. A survey of security in software defined networks. *IEEE Communications Surveys & Tutorials*, 18(1):623–654, 2016. doi: 10.1109/comst.2015.2453114.

[23] Stefan Axelsson. The base-rate fallacy and the difficulty of intrusion detection. *ACM Transactions on Information and System Security*, 3(3):186–205, 2000. doi: 10.1145/357830.357849.

[24] Min Suk Kang, Soo Bum Lee, and V. D. Gligor. The crossfire attack. In *2013 IEEE Symposium on Security and Privacy*, pages 127–141, 2013. doi: 10.1109/sp.2013.19.

[25] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, pages 79–90, 2023. doi: 10.1145/3605764.3623985.

[26] Peyman Kazemian, George Varghese, and Nick McKeown. Header space analysis: Static checking for networks. In *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pages 113–126, San Jose, CA, 2012. USENIX

Association. URL https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/kazemian.

[27] Ahmed Khurshid, Wenxuan Zhou, Matthew Caesar, and P. Brighten Godfrey. Veriflow: Verifying network-wide invariants in real time. *ACM SIGCOMM Computer Communication Review*, 42(4):467–472, 2012. doi: 10.1145/2377677.2377766.

[28] E. S. Al-Shaer and H. H. Hamed. Discovery of policy anomalies in distributed firewalls. In *IEEE INFOCOM 2004*, pages 2605–2616, 2004. doi: 10.1109/infcom.2004.1354680.

[29] Juliana Arévalo-Herrera, Jorge E. Camargo, José Ignacio Martínez Torre, Juan Marcos Ramírez, and Tatiana Zona-Ortiz. A multi-class sdn intrusion detection dataset with synchronized openflow control-plane telemetry. *Data*, 11(8):195, 2026. doi: 10.3390/data11080195.

[30] Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2.5 technical report, 2024. URL https://arxiv.org/abs/2412.15115. arXiv preprint.

[31] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

[32] Juliana Arevalo Herrera et al. LAN-SDN-NIDS: A multi-class dataset for OpenFlow attack detection. Dataset, e-cienciaDatos, 2026. URL https://doi.org/10.21950/QMAXKP.

[33] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016. doi: 10.1145/2939672.2939785.

[34] Manuel Peuster, Johannes Kampmeyer, and Holger Karl. Containernet 2.0: A rapid prototyping platform for hybrid service function chains. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 335–337, 2018. doi: 10.1109/netsoft.2018.8459905.

[35] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80, 1945. doi: 10.2307/3001968.

[36] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979. URL https://www.jstor.org/stable/4615733.

[37] B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1), 1979. doi: 10.1214/aos/1176344552.