Towards Reproducible Attack Scenarios With Automated Labeling

Jiayi Xie
Imperial College London
j.xie24@imperial.ac.uk

Abdullah Aldaihan
Imperial College London
a.aldaihan23@imperial.ac.uk

Sergio Maffeis
Imperial College London
sergio.maffeis@imperial.ac.uk

***Abstract*—Intrusion detection research is limited by a shortage of usable datasets that are shareable, precisely labeled, and not locked to a single telemetry footprint. We present RAS, a work-in-progress framework for Reproducible Attack Scenarios. RAS turns a scenario definition into a shareable package and uses container or VM instrumentation to execute behaviors, collect logs, and assign deterministic event-level labels from an artifact list. Packages declare hosts and topology, behavior scripts, collection points, and preprocessing hooks. We exercised RAS on a vulnerable web server scenario. With minimal edits we created multiple derivative scenarios that varied benign and malicious workloads, and we replayed the base scenario with additional IDS telemetry by enabling Suricata, without changing behaviors or labels. These early results indicate that RAS can generate labeled datasets with predictable characteristics while preserving reproducibility and extensibility.**

***Index Terms*—Intrusion Detection, Network Security, Computer Security**

I. Motivation & Background

Modern intrusion detection research aims to identify previously unseen attacks, yet progress is throttled by weaknesses in the datasets we use to develop and evaluate methods. We outline three issues and how they distort conclusions.

1) **Scarcity of usable datasets**. There is a general shortage of usable datasets for benchmarking intrusion detection systems [1], datasets that are large enough, diverse in behavior, and legally shareable. Many available corpora are too small for meaningful evaluation or cannot be redistributed due to licensing and privacy constraints.
2) **Coarse-grained and ambiguous ground truth**. Ground truth is, in some cases, supplied at a level that is too coarse to support fair evaluation. Researchers then infer ground truth that vary from work to work, leading to inconsistent metrics and limited comparability. For example, batch-level labels can make models look strong even when per-event or per-process accuracy is weak [2].
3) **Rigid telemetry**. Most datasets lock you into the signals captured at collection time, making it hard to add or swap sensors or change fidelity. For example, network-only sets exclude endpoint-centric methods; host-only sets exclude flow and DNS-aware methods. Researchers cannot easily attach Zeek alongside Sysmon, or simply eBPF probes.

To address these problems, we propose RAS, a work-in-progress framework to turn intrusion detection datasets into shareable scenario packages. The objectives are simple: (i) enable creation of datasets with minimal effort, (ii) automate labeling using node-level artifacts, and (iii) facilitate derivative datasets with additional telemetry.

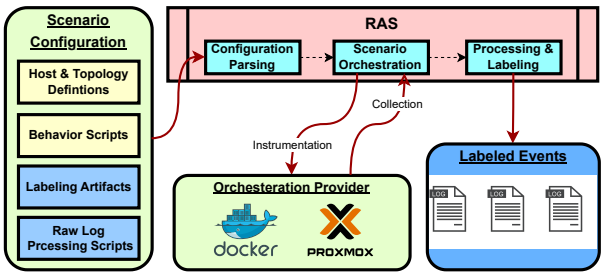

Fig. 1. RAS architecture overview.

II. Approach

RAS is an orchestration framework for the automated production of attack scenarios datasets (Figure 1). Given a scenario configuration, it parses the configuration and then instruments the scenario environment with VMs or containers, collects logs, and produces labeled events. The reusable unit is a scenario configuration bundle that can be easily shared and distributed.

Host and Topology Definitions. The bundle declares the hosts in the scenario and how they are connected. Each host specifies a base image (for example, a Docker Alpine image), collection points (a simple list of files or directories to pull if collection is needed), and any host-specific configuration, such as a listening port. In addition, the definitions may include a network collection point where `tcpdump` captures transit traffic. Finally, the scenario configuration defines when each host is spun up, the startup order, and how long it remains active.

Behavior Scripts. Behavior scripts define the host workload, whether malicious or benign. They are a cornerstone to creating derivative scenarios with alternative behavior to test specific hypotheses and observe how detection systems react. Scripts can specify, for example, how a benign host interacts with services, how a malicious user proceeds through initial access and exploitation, and what post-exploitation impact occurs.

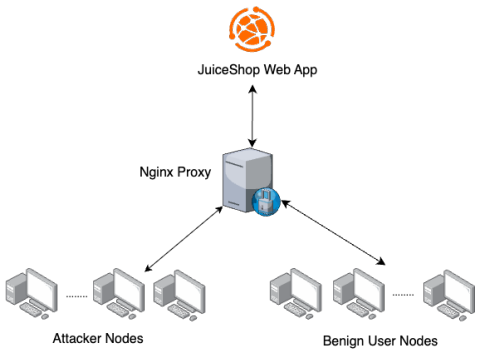

Fig. 2. Web-exploitation scenario with OWASP Juice Shop and Ngnix

TABLE I
SCENARIO CONFIGURATIONS AND OBSERVED ATTACK RATIOS FOR FOUR SAMPLES

Scenario	Benign Nodes	Attack Nodes	Duration	Dataset Size
`Balanced`	1	1	60min	221.9 MB
`Stress`	2	3	30min	111.1 MB
`Realistic`	3	1	120min	360 MB
`APT`	3	2	360min	1.03 GB

Labeling Artifacts. To automate labeling, the bundle includes an artifact list (for example, domains, IP addresses, executable names, file paths, hash prefixes, payload tags). The labeling rule is simple: if a log event contains one of the malicious artifacts, the event is labeled malicious. Additionally, artifacts can be scoped to a time window to bound labeling to the intended phase of the scenario.

Raw Log Processing Hooks. Preprocessing hooks are user-defined scripts that transform raw logs into a specific format before labeling them. For example, to join log sources together, or to transform network traffic dump (PCAP) into specific CSV schema. Hooks are invoked after the execution of a scenario terminates. Without them, RAS labels still performs default artifact-based labeling, but users can provide hooks to extend or replace this step with their own parsing or labeling methods.

III. Preliminary Results

Implementation. We implemented RAS as a command-line tool in Python, using Docker as the instrumentation backend. This early version provisions containers from a scenario bundle, executes workloads, pulls the declared collection points, and applies artifact-based labeling to produce event-level ground truth.

Base Scenario. We implemented a base scenario (Figure 2) consisting of a vulnerable web application (OWASP Juice Shop[1]) served behind an Nginx reverse proxy used to log all inbound HTTP requests. The environment includes a variable, configurable number of benign and malicious clients. Benign clients are built from Alpine Linux images and scripted to browse, add items to cart, place orders, and check out; interaction delays are randomized to avoid perfectly periodic traffic.

[1]https://owasp.org/www-project-juice-shop/

TABLE II
SURICATA ON/OFF COMPARISON (3-RUN AVERAGE).

Mode	TotalFlows	AttackFlows	BenignFlows	AttackRatio (Δ pp)
Suricata off	212.3	111.7	100.7	52.6% (+2.6)
Suricata on	142.3	68.7	73.7	48.3% (-1.7)

Malicious clients use Kali Linux base images to run exploitation steps. We define multiple attacker variants: some fail to exploit, others are partially successful, and one completes exploitation and establishes a C2 channel. Each variant inserts bounded random waits (up to 60 seconds) between steps to preserve behavior and to prevent detection systems from making spurious correlations. All hosts are connected with the target over a simple virtual bridge.

Experiment 1: Behavior Variants. We instantiated four variants that share the same environment but differ in client workloads and attacker paths. As shown in Table I, the dataset size scales with scenario complexity and duration, these changes are simple few lines edits, configuring which behavior script to run for each host and how many of them. Running the scenario is simple as well, a single command line invocation.

Experiment 2: Additional Telemetry. We replayed the base scenario with the addition of a monitoring host that runs Suricata. All traffic to JuiceShop was first forwarded to Suricata and then to JuiceShop, with no changes to behaviors. As summarized in Table II, total flows decreased on average from 212.3 to 142.3, but the attack share remained near the target (52.6% vs. 48.3% a shift of only a few percentage points). The decrease in flows occurred because of resources constraints, which resulted in some packets being dropped from the main bridge connecting benign and malicious hosts with Suricata.

IV. Future Work

We plan to expand RAS along two fronts: broader scenario coverage and a virtualization-based instrumentation layer. We plan to exercise the current implementation across a wider set of scenarios that features a variety of application and attack tactics. Our goal is to confirm that observed characteristics track configuration targets, and that labels remain stable across runs. In parallel, we will add a Proxmox backend alongside Docker to support full system virtualization. Proxmox will enable scenarios that require kernels-level visibility, including kernel-level attacks and advanced monitoring such as eBPF and audit frameworks.

References

[1] A. Kenyon, L. Deka, and D. Elizondo, "Are public intrusion datasets fit for purpose characterising the state of the art in intrusion event datasets," *Computers & Security*, vol. 99, p. 102022, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404820302959

[2] B. Jiang, T. Bilot, N. El Madhoun, K. Al Agha, A. Zouaoui, S. Iqbal, X. Han, and T. Pasquier, "Orthrus: Achieving high quality of attribution in provenance-based intrusion detection systems," in *Security Symposium (USENIX Sec'25). USENIX*, 2025.