ECE 647 Final Project

Partial Ruleset Building for East-West Datacenter Intrusion Detection Systems

I certify that the work for this project was done entirely in this class, even if it is related to my own research.

Jason Lei

1 Introduction

Public cloud environments such as Amazon's AWS and Microsoft's Azure continue to grow in scale in both infrastructure and usage year over year. As many organizations and individuals move critical systems and sensitive data to these cloud environments, they have increasingly become the target of high-profile cybersecurity breaches, such as SolarWinds [1], MOVEit [4], and Midnight Blizzard [7]. Incidents such as these incur significant financial losses and leakage of private information, underscoring the importance of proactive security measures.

Network Intrusion Detection Systems (IDSs) serve as the first line of defense against malicious activity for many cloud environments. The vast majority of state-of-the-art IDSs place their focus on monitoring *north-south traffic*, communication between internal networks (the cloud) and external parties (users and potential attackers). In the process, these systems often overlook *east-west traffic*, communication occurring within the cloud infrastructure. Logically, attackers will typically come from outside the network, which is why this model has persisted as industry standard for so long. However, insufficient monitoring of this internal traffic exposes cloud environments to internal threats and lateral movement attacks [2, 5, 9].

The most common type of IDS currently deployed to cloud systems are rule-based, such as Suricata [3], Zeek [8], and Snort [6]. These IDSs inspect packet headers and payloads against massive sets of IDS rules. With network interfaces entering Tbps speeds, the costs of keeping these systems running easily exceeds thousands of dollars per year per instance.

IDSs only need to be deployed at the network edge to monitor all north-south traffic by its very nature. However, east-west traffic does not share this property. Datacenter infrastructure is typically decentralized by design, necessitating the deployment of IDSs to every node within the network for full coverage of east-west traffic. Modern cloud systems easily exceed hundreds or even thousands of instances, causing this cost to become prohibitively expensive. As a result, east-west traffic is often left poorly monitored, if at all.

2 Formulation

One way to reduce the cost of running these systems is to only check certain rules on a fraction of the network traffic. These rulesets are typically tens of thousands of rules long (or more), and are built collaboratively and incrementally over long periods of time, with each rule having varying levels of severity and likelihood of appearing. Consequently, this fractional traffic checking must be done carefully in order to minimize the cases where an attack goes unnoticed whereas they would have been detected had all traffic been checked against every rule.

With this in mind, let us consider a cloud network consisting of N nodes. Each node within the network $n \in N$ deploys an IDS using ruleset R, while checking each rule $r \in R$ on some percent of traffic entering or leaving the node using sampling rate s_{nr}. The cost to check rule r with $s_{nr} = 1$ (i.e. all traffic is checked) is $cost_r$, and the costs incurred by missing a rule match is represented by $penalty_r$. The likelihood that node n would detect at least one packet matching to rule r with $s_{nr} = 1$ is given by detection probability value p_{nr}.

We wish to minimize the total cost of deployment, while magnifying the importance of more severe rules, particularly those which are likely to appear. Logically, you can interpret a rule with a severity value of 2 as begin twice as worthwhile to include compared to a rule with a severity value of 1. Using this information, we can formulate the utility function as such:

$$U_n = \sum_{r \in R} \frac{cost_r s_{nr}}{penalty_r p_{nr}}, \quad \forall n \in N$$

We would also like to guarantee a maximum allowable false negative rate F, to prevent the system's performance from degrading below an acceptable level. With this information, we can formulate the problem as such:

$$\begin{aligned}
\text{minimize} \quad & \sum_{n \in N} \sum_{r \in R} \frac{cost_r s_{nr}}{penalty_r p_{nr}} \\
\text{subject to} \quad & \sum_{r \in R} p_{nr}(1 - s_{nr}) \leq F, \quad \forall n \in N \\
& 0 \leq s_{nr} \leq 1, \quad 0 \leq p_{nr} \leq 1, \quad \forall r \in R, \forall n \in N \\
& cost_r > 0, \quad penalty_r > 0, \quad \forall r \in R
\end{aligned}$$

This problem is linear and convex w.r.t s.

3 Analysis and Algorithm

Using the previously described convex optimization problem, we can form the Lagrangian:

$$\textcircled{1} \quad L(f,\lambda) = \sum_{n\in N}\sum_{r\in R}\frac{cost_r s_{nr}}{penalty_r p_{nr}} + \sum_{n\in N}\lambda_n\left[\sum_{r\in R} p_{nr}(1-s_{nr}) - F\right]$$

This can be rewritten as:

$$\textcircled{2} \quad L(f,\lambda) = \sum_{n\in N}\sum_{r\in R}\left[\frac{cost_r s_{nr}}{penalty_r p_{nr}} + \lambda_n p_{nr}(1-s_{nr})\right] - F\sum_{n\in N}\lambda_n$$

Therefore the dual function is:

$$\textcircled{3} \quad g(\lambda) = \begin{array}{ll} \underset{s}{\text{minimize}} & \displaystyle\sum_{n\in N}\sum_{r\in R}\left[s_{nr}\left(\frac{cost_r}{penalty_r p_{nr}} - \lambda_n p_{nr}\right) + \lambda_n p_{nr}\right] - F\sum_{n\in N}\lambda_n \\ \text{subject to} & 0 \le s_{nr} \le 1, \quad 0 \le p_{nr} \le 1, \quad \forall r \in R, \forall n \in N \\ & cost_r > 0, \quad penalty_r > 0, \quad \forall r \in R \end{array}$$

This function is linear with respect to s_{nr} therefore we know the solution lies at a boundary. With this knowledge, we can define that each node will set s_{nr} for each rule according to the following function:

$$\textcircled{4} \qquad s_{nr}(t) = \begin{cases} 1, & \text{if} \quad \lambda_n(t) \ge \frac{cost_r}{penalty_r p_{nr}^2} \\ 0, & \text{otherwise} \end{cases}$$

This informs us that each node will only check each given rule on *all traffic* it encounters, or on *none of the traffic*. Now, we can isolate the dual function only using terms related to λ_n, producing the dual problem:

$$\textcircled{5} \qquad \begin{array}{ll} \text{maximize} & \displaystyle\sum_{n\in N}\lambda_n\left[\sum_{r\in R}[p_{nr}(1-s_{nr}(t))] - F\right] \\ \text{subject to} & \lambda \ge 0 \end{array}$$

This dual problem is linear maximization problem, therefore its solution will also lie at a boundary. Using our definition in equation ④ we can determine the term $1 - s_{nr}(t)$ is strictly equal to 0 for rules included in node n's ruleset, and stricltly equal to 1 for rules not in n's ruleset. Therefore,

the full term $\sum_{r \in R} p_{nr}(1 - s_{nr}(t))$ can be interpreted as the sum of the detection probabilities of the rules which are checked against none of the incoming traffic. We can then update the λ_n values iteratively like so:

$$\textcircled{6} \qquad \lambda_n(t+1) = \lambda_n(t) + \gamma \left[\sum_{r \in R} [p_{nr}(1 - s_{nr}(t))] - F \right]$$

Logically, this means if the sum of detection probabilities of unchecked rules is below the maximum allowed threshold F, then λ_n will go decrease, attempting to enable more rules according to equation ④. However, if the threshold F is exceeded, then λ_n will increase, attempting to disable more rules from being checked.

The Lagrangian is strictly linear; therefore Slater's condition holds and there will be no duality gap.

4 Simulation Results

I evaluated this design using a software simulator with a network consisting of $N = 50$ nodes, a ruleset with $R = 100$ rules, and a maximum false positive rate of $F = 5\%$.

The cost values for each rule $cost_r$ were generated by sampling a Gaussian random distribution with $\mu = 100$ and $\sigma = 20$. Similarly, the penalty values for each rule $penalty_r$ were generated by sampling a Gaussian random distribution with $\mu = 1000$ and $\sigma = 100$. In a real-world setting, the majority of rules are rarely or never matched, whereas a few rules are significantly more likely to match. Consequently, the detection probabilities for each rule on each node p_{nr} were sampled from a Zipfian distribution using distribution parameter $a = 4$.

By directly implementing the algorithm described in Section 4, the following results were produced for the primal variable s_{nr} and dual variable λ_n in Figures 1 and 2:

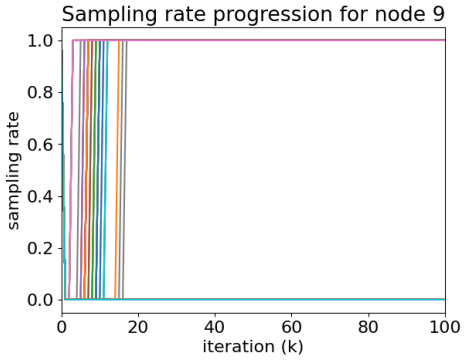

Figure 1: Sampling rate progression by rule for node 9

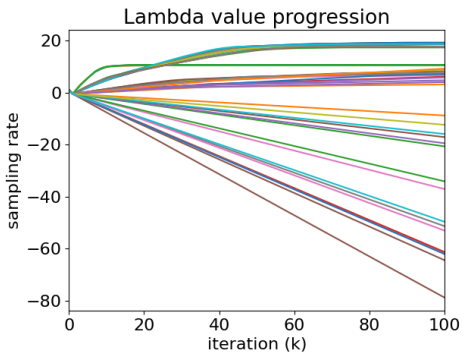

Figure 2: Lambda value progression by node

This resulted in a corresponding total false negative rate and cost shown in Figures 3 and 4:

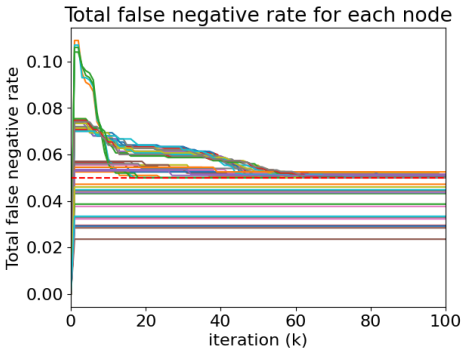

Figure 3: Total FN rate by node

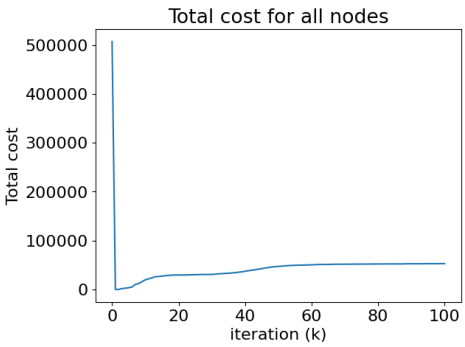

Figure 4: Total cost across all nodes

Ideally the total false negative rate would converge below the value $F = 5\%$, whereas Figure 5 shows most nodes converge to slightly above F, as shown by the dotted red line. To combat this, I imposed a penalty term of 0.25 on gradients greater than 0, i.e. when $\sum_{r \in R} p_{nr}(1 - s_{nr}(t)) > F$. Doing so brings the total false negative rate for each node below the permitted value, as shown in Figures 5 and 6:

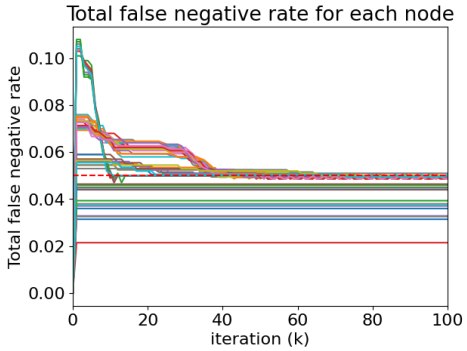

Figure 5: Total FN rate with penalty applied

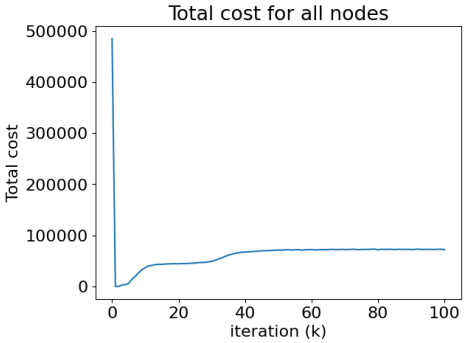

Figure 6: Total cost with penalty applied

Figure 6 shows the cost remains relatively the same as in Figure 4, while enforcing the performance guarantee of F as shown in Figure 5. The sampling rate and lambda values can be seen in Figures 7 and 8:

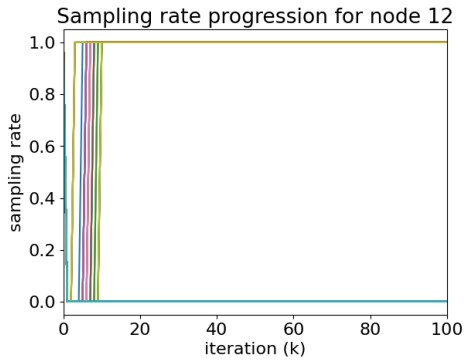

Figure 7: Sampling rate progression for node 12 with penalty applied

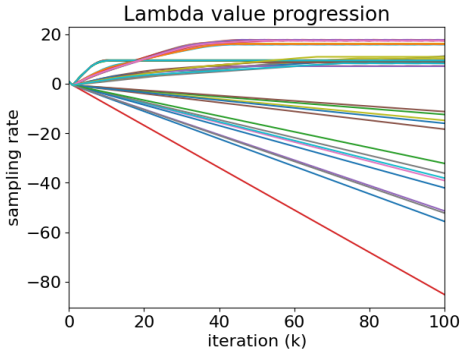

Figure 8: Lambda value progression with penalty applied

Figure 8 demonstrates the negative lambda values taking significantly longer to converge, as they linearly decrease throughout the course of the iterations shown. This is by design; after all, recall from Equation ⑥ that negative lambda values correspond to cases where the total false negative rate falls below F. It is not always possible to achieve a total false negative rate of exactly F, therefore the gradient will continue to be negative. However, if it is extremely close to the solution, then λ_n continues to decrease by a very small amount every iteration, which is exactly what we observe. However, in

the case where λ_n decreases enough to add an additional rule which would cause it to exceed F, the penalty term will quickly bring it back down. This allows us to ensure each node spends as much time below the threshold F as possible. This is demonstrated by the nodes corresponding to the trends highlighted by the red circle in Figures 9 and 10:

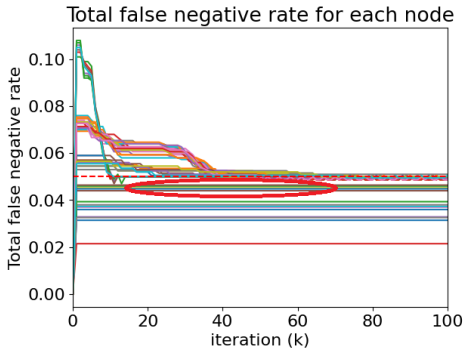

Figure 9: Many nodes approach the limit value F and then remain there for an extended period of time

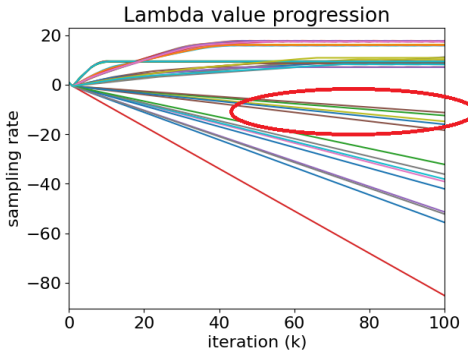

Figure 10: The corresponding lambda values decrease slowly

Additionally, for many nodes it is not necessary to include many rules to achieve a false negative rate below F. This is showcased by the grouping of flat total FN rates below the dotted red line for many nodes shown in Figure 5. These nodes are able to maintain a very low total FN rate at low cost without even needing to approach the limit F. However, this still produces a strong negative gradient value, thus allowing the lambda value to grow to a large negative value. This is demonstrated by the nodes corresponding to the trends highlighted by the red circles in Figures 11 and 12:

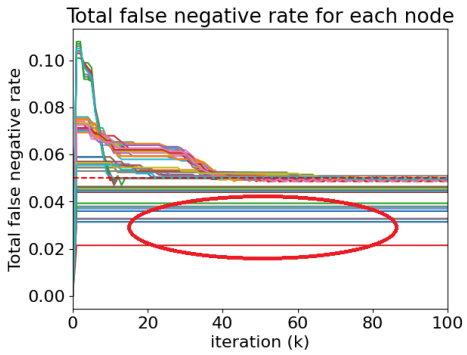

Figure 11: Nodes which can produce low FN at low cost converge quickly

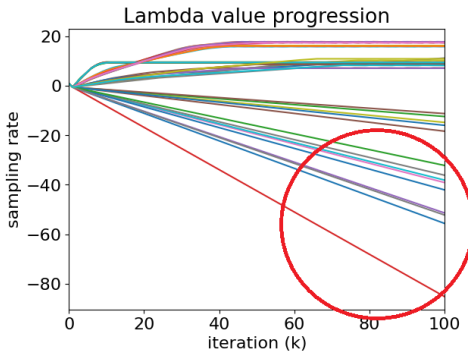

Figure 12: The corresponding lambda values decrease rapidly

Overall, the results agree with my expectation. The part I had not anticipated is that exact convergence of the primal and dual variables cannot be attained with the current setup, as it is very unlikely that a set of rules exists which sums to exactly F. As a consequence, the corresponding lambda values continue to follow a linear pattern even after the rulesets have converged. I believe this to be a consequence of Equation ④'s stepwise nature, which forces rules to only check all or no traffic on a node. It may be interesting to explore different utility functions or constraints that would produce convergence to a fixed point.

Thank you for the wonderful semester, and I hope you have a great summer!

5 Code

The code for this project can be found at `https://github.com/lei56/ece647_final`.

References

[1] 2020 United States federal government data breach. `https://en.wikipedia.org/wiki/2020_United_States_federal_government_data_breach`, Retrieved on 2023-04.

[2] Aviatrix Distributed Cloud Firewall. `https://aviatrix.com/distributed-cloud-firewall/`, Retrieved on 2023-09.

[3] Suricata. https://suricata.io/, Retrieved on 2023-09.

[4] Zero-day vulnerability in moveit transfer exploited for data theft. https://cloud.google.com/blog/topics/threat-intelligence/zero-day-moveit-data-theft, Retrieved on 2025-04.

[5] John Kindervag, Stephanie Balaouras, and Lindsey Coit. Build security into your network's DNA: The zero trust network architecture. *Forrester Research Inc*, 27, 2010.

[6] Vinod Kumar and Om Prakash Sangwan. Signature based intrusion detection system using SNORT. *International Journal of Computer Applications & Information Technology*, 1(3), 2012.

[7] MSRC. Microsoft actions following attack by nation state actor midnight blizzard. https://msrc.microsoft.com/blog/2024/01/microsoft-actions-following-attack-by-nation-state-actor-midnight-blizzard/, Retrieved on 2024-01.

[8] Vern Paxson. Bro: A system for detecting network intruders in real-time. *Comput. Netw.*, 31(23–24):2435–2463, dec 1999.

[9] Scott Rose, Oliver Borchert, Stu Mitchell, and Sean Connelly. Zero trust architecture. Technical report, National Institute of Standards and Technology, 2020.