

Reliable Decision Rules for AI-Assisted Web Security

Sanda Bujačić Babić

Joint work with Tajana Ban Kirigin

Faculty of Mathematics, University of Rijeka, Croatia

This work is supported by the Croatian Science Foundation under project number HRZZ IP-2024-05-3882 and by the European Union–NextGenerationEU under project number uniri-iz-25-74-MMUI.

uniri



Table of Contents

Introduction

Statistical Model

Likelihood-Ratio Function

AI-Assisted Calibration

HAProxy Implementation

Future Work

The Basic Problem

Every web application produces a continuous stream of activity:

- ▶ logins,
- ▶ requests,
- ▶ errors,
- ▶ retries,
- ▶ redirects.

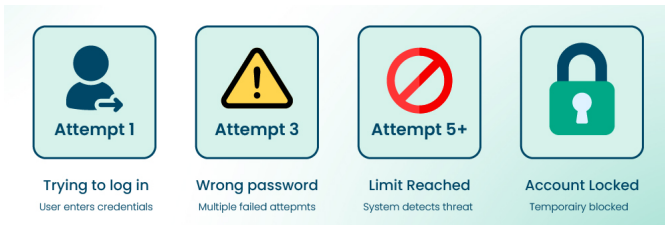
Most of this activity is legitimate, but some of it may be caused by an attack.

The challenge is to decide when the observed activity becomes suspicious enough to act.

When Should We Raise an Alert?

- ▶ Security-relevant events, such as **failed login attempts**, occur both during legitimate use and during attacks.
- ▶ Our goal is to try to understand how observed security events can be translated into **reliable alert decisions**, especially since the frequency of such events can vary substantially even during legitimate activity.

When does an observation provide enough evidence to raise an alert?



When Should We Raise an Alert?

Consider a detector monitoring failed login attempts. For illustration, suppose the detector produces the following attack scores:

Failed logins	"Attack score"	Decision
2	0.25	No alert
5	0.62	?
8	0.79	?
10	0.84	Alert

If the alert threshold is set at 0.8, only the last observation triggers an alert.

But why 0.8?

Changing the threshold changes both **what** is marked as suspicious and **how often** we raise false alarms.

From Observations to Decisions

Two Security Concerns

- ▶ How should observations be translated into alert decisions?
 - ▶ Should larger counts always be interpreted as stronger evidence of an attack?
-
- ▶ A more sensitive rule may detect more attacks, but at the cost of more **false positives**.
 - ▶ A more conservative rule may reduce false alarms, but at the cost of more missed attacks (**false negatives**).

Scenario I: Vertical Brute-Force

Concentrated Attack on a single account

- ▶ **Attack:** compromise a high-privilege account.
- ▶ **Method:** submit many password guesses against the same username in a short time interval.

Time	User	Source IP	Request	Result	Failures
10:01:43	admin	185.72.14.9	POST /login	Failed	1
10:01:46	admin	185.72.14.9	POST /login	Failed	2
10:01:49	admin	185.72.14.9	POST /login	Failed	3
10:01:53	admin	185.72.14.9	POST /login	Failed	4
10:01:56	admin	185.72.14.9	POST /login	Failed	5
10:02:02	admin	185.72.14.9	POST /login	Failed	6
10:02:07	admin	185.72.14.9	POST /login	Failed	7

A simple per-account threshold can detect this concentrated attack quickly.

Scenario II: Password Spraying

Distributed Attack

- ▶ **Attack:** compromise any valid account.
- ▶ **Method:** test a small number of common passwords across many different accounts.

Time	User	Source IP	Result	Failures/account
10:01:02	ana	185.72.14.9	Failed	1
10:01:05	ivan	185.72.14.9	Failed	1
10:01:08	marko	185.72.14.9	Failed	1
10:01:11	petra	185.72.14.9	Failed	1
10:01:14	luka	185.72.14.9	Failed	1
		⋮		
10:05:00	user10000	185.72.14.9	Failed	1

Scenario II: Password Spraying

A Different Attack Pattern

The attack is distributed across many accounts. Each individual account may register only one failed login, even though the system observes large-scale coordinated activity.

10,000 failed attempts globally but 1 per account

Limitation

A per-account rule may miss an attack that distributes authentication attempts across many different accounts.

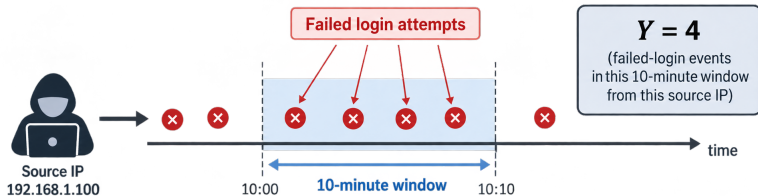
Decision therefore depends not only on the threshold, but also on what we count.

What Are We Counting?

Define the Observation

For the mathematical model, we consider a fixed observation window and let

Y is number of selected failed-login events in a fixed 10-minute window from a single source IP.



From HTTP Responses to Security Events

Authentication Example

- ▶ **Failed login:** the application may re-display the login page **HTTP 200 OK**.
- ▶ **Successful login:** the application may redirect the user to an authenticated area **HTTP 302 Found**.

Application-Specific Event Definition

A failed authentication event may therefore be identified by a rule such as

$$\text{POST} \wedge \text{/login} \wedge \text{HTTP 200},$$

provided that this pattern corresponds to a failed login in the application under consideration.

What Are We Counting?

In our case:

- ▶ events may occur **independently**,
- ▶ events may occur at an approximately **constant rate** within the observation window,
- ▶ we are interested in the **number of events** occurring during a fixed period of time.

Model the Count

A natural starting point for count data is the Poisson model:

$$Y \sim \text{Pois}(\lambda), \quad P(Y = n) = e^{-\lambda} \frac{\lambda^n}{n!}, \quad n = 0, 1, 2, \dots$$

where $\lambda > 0$ represents the expected number of events in the observation window.

Modeling the Heterogeneous Background ($\mathbb{P}_0$)

A single Poisson rate assumes one typical level of legitimate activity. Instead, we allow the system to operate in **two legitimate regimes**.

Let Λ denote the **latent legitimate rate**:

$$\Lambda = \begin{cases} a, & \text{with probability } w, \\ b, & \text{with probability } 1 - w, \end{cases} \quad 0 < a < b, \quad 0 < w < 1.$$

Here,

- ▶ a is the expected failed-login count in the **quiet** legitimate regime,
- ▶ b is the expected failed-login count in the **busy** legitimate regime,
- ▶ w is the probability of the quiet regime.

Modeling the Heterogeneous Background ($\mathbb{P}_0$)

Conditional on the latent regime,

$$Y \mid \Lambda \sim \text{Pois}(\Lambda).$$

Modeling Assumption

The two regimes form a minimal interpretable approximation. The parameters a, b, w are environment-specific and must be estimated from historical benign data.

Modeling an Attack

- ▶ Under legitimate activity, the failed-login count is generated by either the quiet or the busy regime.
- ▶ We model an attack as an **additional source of failed login attempts**, independent of the current legitimate regime:

$$Z \sim \text{Pois}(\theta), \quad \theta > 0.$$

Additive Attack Model

Under attack, the total observed count is the sum of the legitimate background count and Z . Since independent Poisson counts add θ (attack intensity),

$$a \longrightarrow a + \theta, \quad b \longrightarrow b + \theta.$$

The attack increases the event rate by the same amount in both legitimate regimes.

Formalizing the Hypotheses (H_0 vs. H_1)

Null Hypothesis (H_0): Legitimate Operation

The observed count follows the two-regime legitimate background model:

$$p_0(n) := \mathbb{P}_0(Y = n) = w \frac{e^{-a} a^n}{n!} + (1 - w) \frac{e^{-b} b^n}{n!}.$$

Alternative Hypothesis (H_1): Under Attack

The attack adds an independent Poisson component of intensity θ , shifting both legitimate regimes:

$$p_1(n) := \mathbb{P}_1(Y = n) = w \frac{e^{-(a+\theta)} (a + \theta)^n}{n!} + (1 - w) \frac{e^{-(b+\theta)} (b + \theta)^n}{n!}.$$

Which Observation Provides Stronger Evidence?

Suppose we observe

$$Y = n.$$

The same value n may occur under both hypotheses:

$$\mathbb{P}_0(Y = n) > 0, \quad \mathbb{P}_1(Y = n) > 0.$$

Therefore, the relevant question is not simply

“Is n large?”

but rather

How should we quantify the relative evidence for H_1 versus H_0 provided by the observation n ?

Likelihood Ratio as Statistical Evidence

For an observed count $Y = n$, we compare its probability under the two hypotheses through the likelihood ratio

$$L_n = \frac{we^{-(a+\theta)}(a+\theta)^n + (1-w)e^{-(b+\theta)}(b+\theta)^n}{we^{-a}a^n + (1-w)e^{-b}b^n} = \frac{p_1(n)}{p_0(n)} = \frac{d\mathbb{P}_1}{d\mathbb{P}_0}(n).$$

A larger value of L_n means stronger evidence in favour of H_1 relative to H_0 .

Key Question

Does stronger evidence necessarily correspond to a larger count? In other words, must

$$L_{n+1} \geq L_n$$

hold for every n ?

One-Valley Structure of the Likelihood Ratio

Define

$$D = \{n \in \mathbb{N}_0 : L_{n+1} < L_n\}.$$

Theorem

For every admissible choice of a, b, w, θ , the set D is either empty or is a finite set of consecutive integers.

Equivalently, the likelihood-ratio sequence has at most one decreasing interval.

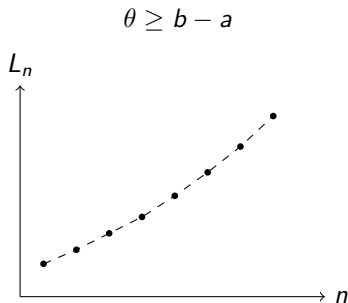
When Can Nonmonotonicity Occur?

It can be shown that

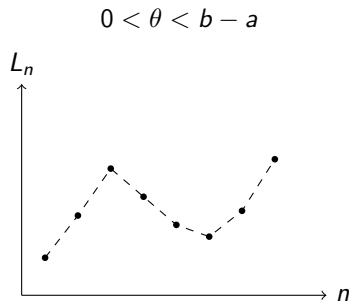
$$\theta \geq b - a \implies L_n \text{ is increasing for every } n \in \mathbb{N}.$$

Hence, **nonmonotonicity** is possible only when

$$0 < \theta < b - a.$$



monotonic



at most one valley

Neyman–Pearson Rejection Region

For the simple hypotheses H_0 and H_1 , the Neyman–Pearson lemma says that a most powerful level- α test rejects H_0 at observations with the largest likelihood-ratio values.

Thus, for a suitable threshold c_α , define the **rejection region**

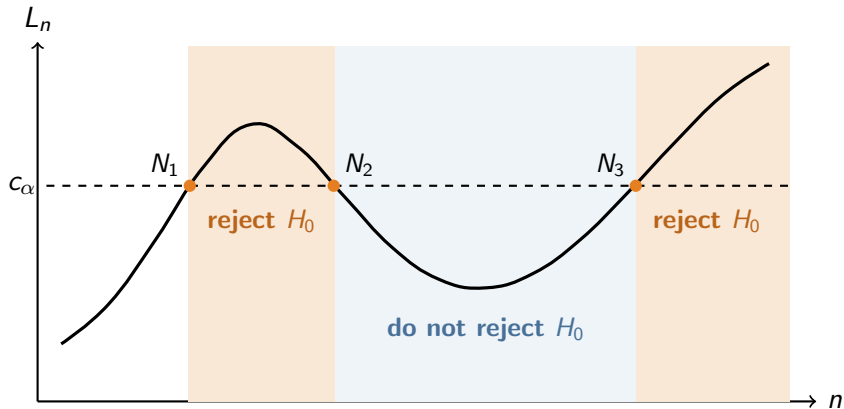
$$\mathcal{R}_\alpha = \{n \in \mathbb{N}_0 : L_n > c_\alpha\}.$$

If L_n were increasing in n , then $\mathcal{R}_\alpha$ would simply be an upper tail (single-threshold rule).

Our one-valley result shows that, in the nonmonotonic case, the rejection region may instead have two components:

$$\mathcal{R}_\alpha = [N_1, N_2]_{\mathbb{Z}} \cup [N_3, \infty)_{\mathbb{Z}}.$$

Rejection Regions



The test thresholds the likelihood ratio L_n , not the count n .

But Where Do the Model Parameters Come From?

In a real system, the values of

$$a, \quad b, \quad w$$

are **not known in advance**.

They must be learned from large amounts of historical traffic, and legitimate traffic patterns may change over time.

We do not want to learn the model while an attack is happening.

Can AI help us learn the environment before the decision has to be made?

Where Can AI Assist?

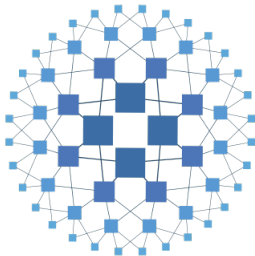
Instead of estimating the background model during an attack, we can learn it in advance from historical legitimate traffic. AI/ML can help us:

- ▶ identify different operating regimes in legitimate traffic,
- ▶ estimate the characteristic event rates in those regimes,
- ▶ estimate how frequently each regime occurs,
- ▶ periodically recalibrate the model as legitimate traffic changes.

In our case: AI learns and calibrates the evidence model from data. Statistical decision theory determines when that evidence is sufficient for an alert.

Learn the background offline; make the decision online.

HAProxy provides the observation and enforcement layer; the statistical model provides the decision rule.



HAProxy

Testing the Approach with HAProxy

HAProxy is a high-performance open-source **load balancer and reverse proxy** for large-scale TCP/HTTP applications.

It operates directly in the request path:



- ▶ HAProxy is deployed in large-scale infrastructures including **Booking.com**, **LinkedIn**, **Microsoft**, **GitHub**, **Vimeo**, **PayPay**, **Adyen**, and **DigitalOcean**.

Testing the Approach with HAProxy

- ▶ Public HAProxy benchmarks have demonstrated throughput exceeding

2 000 000 HTTP requests per second

on a single 64-core AWS Graviton2 instance.

- ▶ We contacted **HAProxy Technologies** and tested our statistical detection approach using traffic data provided through this collaboration.

This provides a realistic high-throughput setting in which to evaluate whether mathematically controlled detection rules can be used in practice.

From HTTP Traffic to a Statistical Decision

Once the count is available, the statistical layer evaluates

$$Y = n \longrightarrow L_n = \frac{p_1(n)}{p_0(n)} \longrightarrow \text{alert / no alert.}$$

Offline / online separation

The model parameters and the corresponding likelihood-ratio scores can be estimated and precomputed offline. Online, HAProxy only has to count, look up the precomputed score, and apply the resulting action.

Likelihood-Ratio Scores in the HAProxy Demo

For the demonstration configuration,

$$a = 0.5, \quad b = 5, \quad w = 0.8, \quad \theta = 0.5.$$

Python precomputes the integer score

$$z(n) = \text{round} \left(100 \log \frac{P_1(n)}{P_0(n)} \right),$$

and stores it in a lookup table such as `llr.map`.

Likelihood-Ratio Scores in the HAProxy Demo

The demonstration maintains the cumulative score

$$S_n = \max\{0, S_{n-1} + z(n)\}.$$

Failure	n	$z(n)$	S_n	HAProxy result
	1	+18	18	Processed
	2	+72	90	Processed
	3	+63	153	Processed
	4	+18	171	Processed
	5	+4	175	Processed
	6	+8	183	Processed
	7	+17	200	Threshold reached
	8	+26	226	Hypothetical

These are illustrative parameters used by the HAProxy demonstration; they are different from the ML estimates presented later.

Why Use a Likelihood-Ratio Lookup Table?

Why is this useful?

- ▶ HAProxy does not need to evaluate probabilities or likelihood ratios online; it only retrieves a precomputed score.
- ▶ The table preserves the likelihood-ratio property: a larger count n does not necessarily provide stronger evidence for an attack.

The mathematical likelihood-ratio rule is reduced to a simple online lookup.

Synthetic Authentication Dataset

We generated synthetic authentication traffic representing a single web site observed over 90 days.

Dataset

- ▶ 10-minute observation windows,
- ▶ 12,960 windows in total,
- ▶ 11,985 benign windows,
- ▶ 975 attack windows,
- ▶ 79 attack episodes.

Observed information

- ▶ login attempts and failures,
- ▶ successful logins,
- ▶ unique users and source IPs,
- ▶ response time,
- ▶ time and attack labels.

The attack windows include credential stuffing, vertical brute force, and password spraying.

The traffic was generated first: no values of a, b, w, θ were built into the dataset.

What Is Observed and What Is Latent?

The training windows are labeled as

benign or attack.

However, the benign windows are **not** labeled as

quiet or busy.

For a benign window we observe the failed-login count Y_i , but its background regime is unknown:

Y_i is observed, $\Lambda_i \in \{a, b\}$ is latent.

The quiet/busy structure must therefore be inferred from the data.

Learning the Background Model

Using only the **benign training windows**, we fit the model

$$Y \mid H_0 \sim w \text{Pois}(a) + (1 - w) \text{Pois}(b).$$

Since the quiet/busy regime of each window is unknown, we use the **Expectation–Maximization (EM) algorithm**.

Starting from initial values of a, b, w , EM alternates between:

1. **E-step** estimating, for each observation Y_i , the posterior probability that it belongs to the quiet or busy regime;
2. **M-step** updating a, b, w by maximizing the corresponding expected log-likelihood.

$$\text{benign counts} \xrightarrow{\text{EM}} \hat{a}, \hat{b}, \hat{w}$$

Estimating the Attack Intensity

The background parameters $\hat{a}, \hat{b}, \hat{w}$ were estimated using the benign training windows.

We now use the **labeled attack windows**. With the background parameters fixed, the attack model is

$$Y_i \mid H_1 \sim \hat{w} \text{Pois}(\hat{a} + \theta) + (1 - \hat{w}) \text{Pois}(\hat{b} + \theta).$$

The common additive attack intensity is estimated by maximum likelihood:

$$\hat{\theta} = \arg \max_{\theta \geq 0} \sum_{i=1}^m \log \left[\hat{w} p_{\hat{a}+\theta}(Y_i) + (1 - \hat{w}) p_{\hat{b}+\theta}(Y_i) \right],$$

where

$$p_{\lambda}(n) = e^{-\lambda} \frac{\lambda^n}{n!}.$$

Two Different Statistical Regimes

L_n is a Monotonic Function

Estimated parameters:

$$\hat{a} = 0.478, \quad \hat{b} = 2.002,$$

$$\hat{w} = 0.8, \quad \hat{\theta} = 11.233.$$

Hence

$$\hat{\theta} > \hat{b} - \hat{a}.$$

Therefore L_n is increasing for every $n \in \mathbb{N}$.

NP rule = single-threshold rule

up to boundary randomization.

L_n is a Nonmonotonic Function

Estimated parameters:

$$\hat{a} = 0.5, \quad \hat{b} = 5,$$

$$\hat{w} = 0.5, \quad \hat{\theta} = 0.5.$$

Hence

$$0 < \hat{\theta} < \hat{b} - \hat{a}.$$

Here L_n is nonmonotonic, $L_2 > L_8$.

NP rule $\neq$ single-threshold rule

and the rejection region may have two components.

Same False-Alarm Level, Different Power

Decision rule	False-alarm probability	Attack detection probability
Single-threshold rule	0.100	0.134661
Neyman–Pearson	0.100	0.155580

The absolute gain in detection probability is

$$0.155580 - 0.134661 = \boxed{0.020918},$$

while the relative increase is approximately

$$\boxed{15.5\%}.$$

The gain comes from using the likelihood-ratio ordering, not from allowing more false alarms.

Future Work: Empirically Motivated Extensions

More Generalized Background Models

Replace the two-component Poisson mixture by a finite K -component mixture or a negative-binomial mixture.

More Flexible Attack Models

Allow the attack to change both component intensities and weights:

$$p_1(n) = w_1 \text{Pois}_{a+\theta_a}(n) + (1 - w_1) \text{Pois}_{b+\theta_b}(n).$$

Temporal Calibration

Estimate parameters from repeated or sequential monitoring.

Future Work: Beyond a Single Count

The present model is based on a single count $Y \in \mathbb{N}_0$.

For distributed attack patterns, a natural extension is a **multivariate observation**

$$\mathbf{Y} = (Y_1, \dots, Y_d) \in \mathbb{N}_0^d,$$

where the components may describe, for example,

Y_1 = number of failed login attempts,

Y_2 = number of distinct targeted accounts,

Y_3 = number of distinct source IPs.

The likelihood ratio then becomes

$$L(\mathbf{y}) = \frac{p_1(\mathbf{y})}{p_0(\mathbf{y})},$$

and the Neyman–Pearson rejection region has the form

$$\mathcal{R}_\alpha = \left\{ \mathbf{y} \in \mathbb{N}_0^d : L(\mathbf{y}) \geq c_\alpha \right\}.$$

References

-  Basseville, M., & Nikiforov, I. V. (1993). *Detection of Abrupt Changes: Theory and Application*. Prentice Hall.
-  HaProxy Technologies <https://www.haproxy.com/>
-  Lehmann, E. L., & Romano, J. P. (2005). *Testing Statistical Hypotheses*. Springer.
-  McLachlan, G. J., & Peel, D. (2000). *Finite Mixture Models*. John Wiley & Sons.
-  Sommer, R., & Paxson, V. (2010). *Outside the Closed World: On Using Machine Learning for Network Intrusion Detection*. IEEE S&P.
-  Tartakovsky, A., Nikiforov, I., & Basseville, M. (2014). *Sequential Analysis: Hypothesis Testing and Changepoint Detection*. CRC Press.
-  Wang, G., et al. (2017). *Anatomy of a Credential Stuffing Attack*. USENIX Security.

Thank you for your attention!