Intrusion Prevention through Optimal Stopping

Invited Talk @Alan Turing Institute London

Kim Hammar & Rolf Stadler

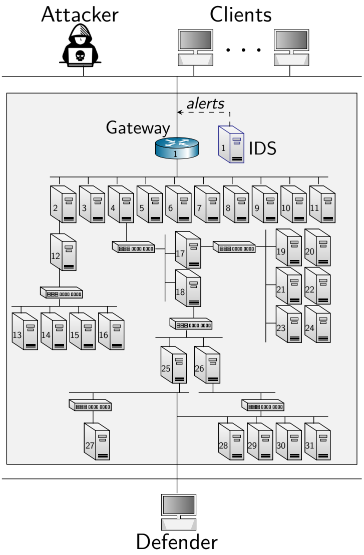
kimham@kth.se & *stadler@kth.se*

Division of Network and Systems Engineering
KTH Royal Institute of Technology

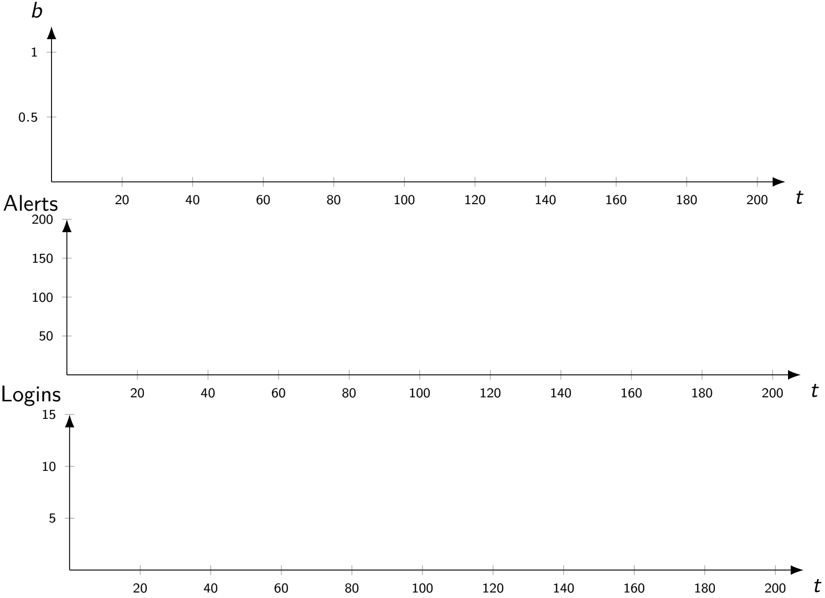
Mar 25, 2022

Use Case: Intrusion Prevention

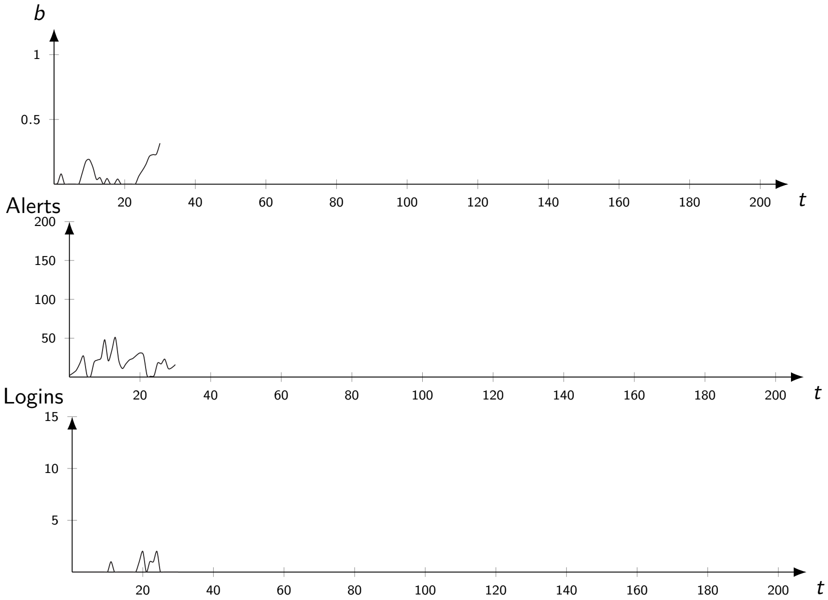
- A **Defender** owns an infrastructure
  - Consists of connected components
  - Components run network services
  - Defender defends the infrastructure by monitoring and active defense
  - Has partial observability
- An **Attacker** seeks to intrude on the infrastructure
  - Has a partial view of the infrastructure
  - Wants to compromise specific components
  - Attacks by reconnaissance, exploitation and pivoting



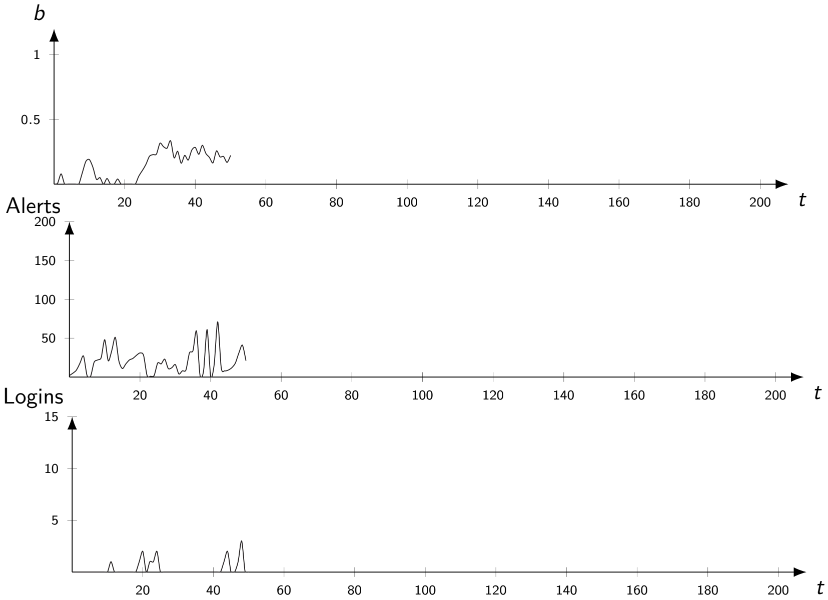
The Intrusion Prevention Problem



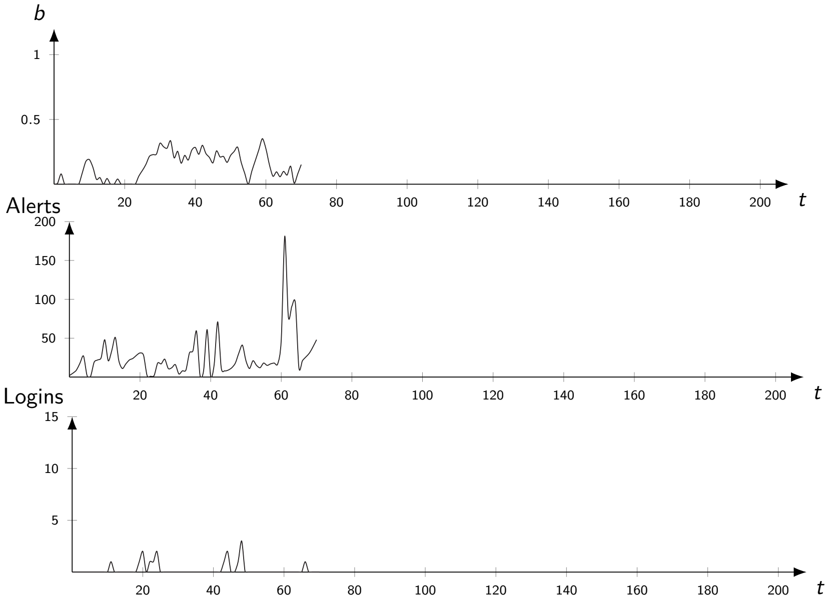
The Intrusion Prevention Problem



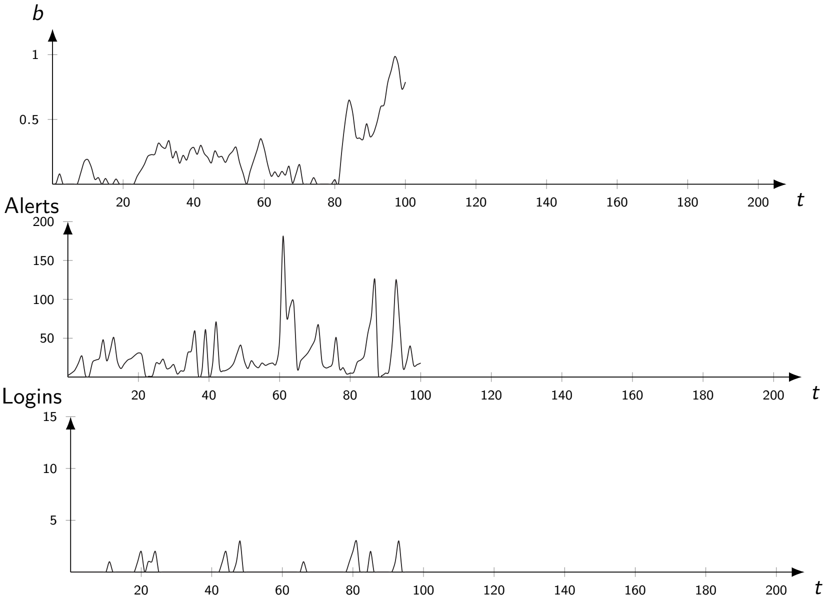
The Intrusion Prevention Problem



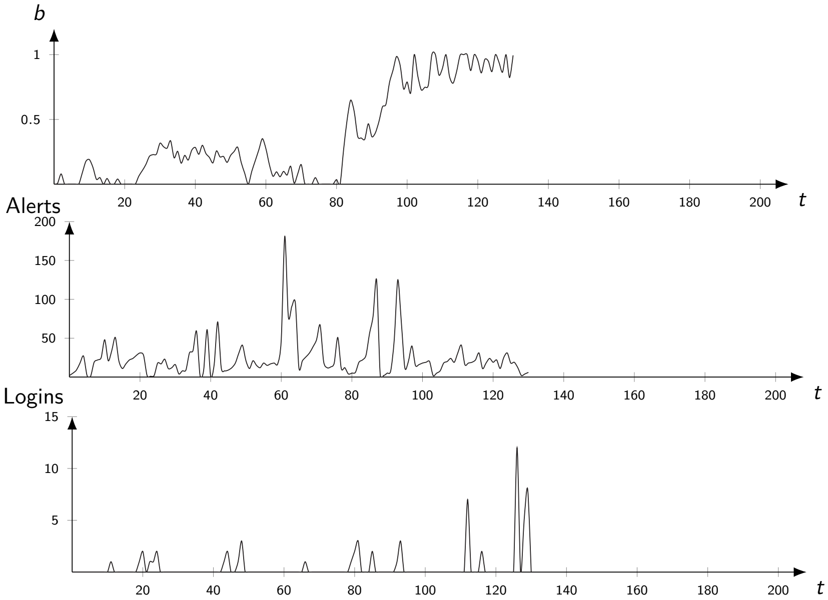
The Intrusion Prevention Problem



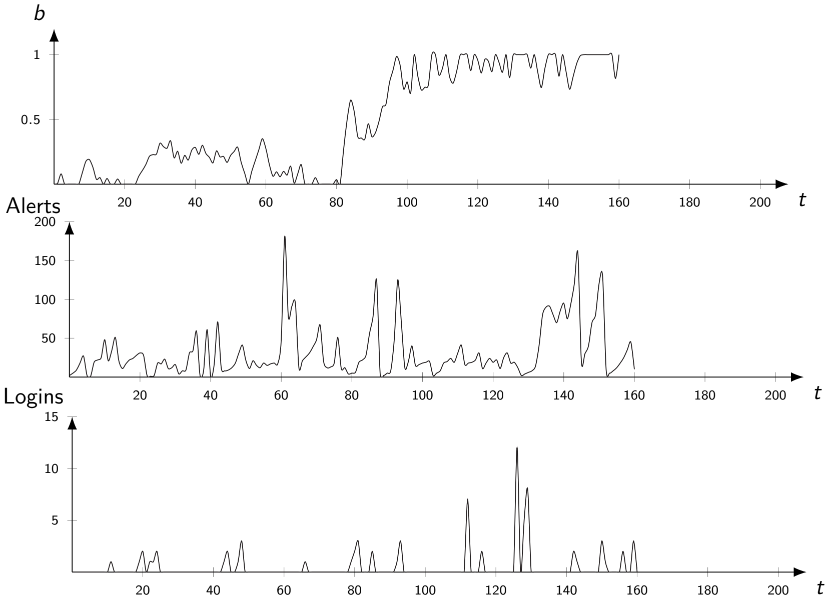
The Intrusion Prevention Problem



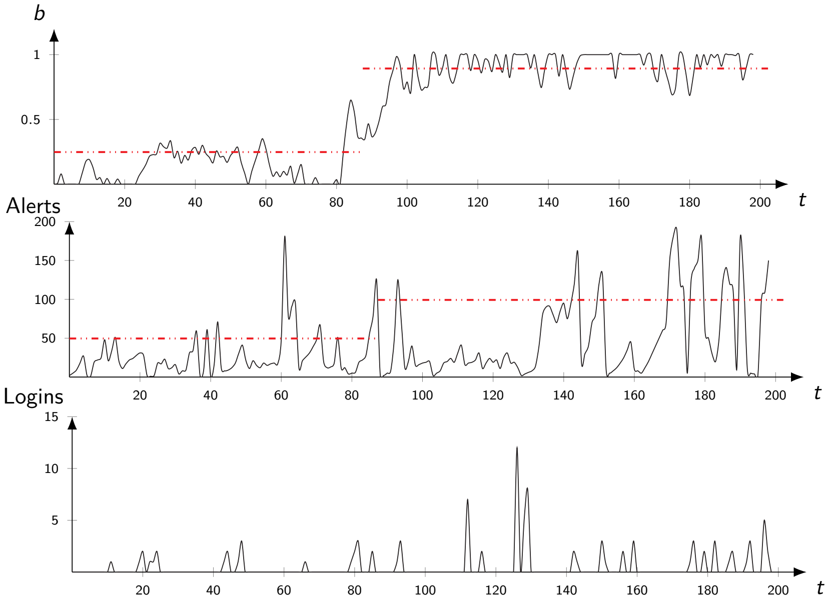
The Intrusion Prevention Problem



The Intrusion Prevention Problem

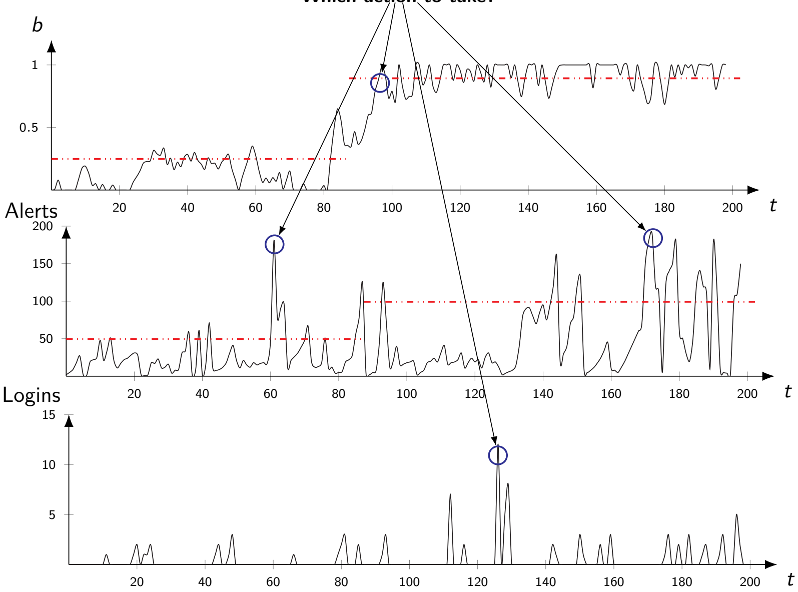


The Intrusion Prevention Problem

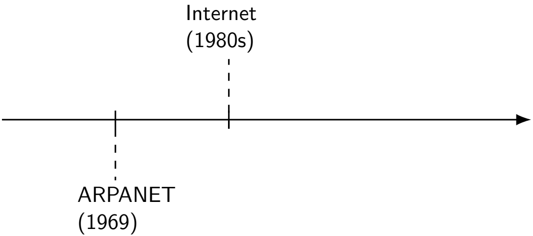


The Intrusion Prevention Problem

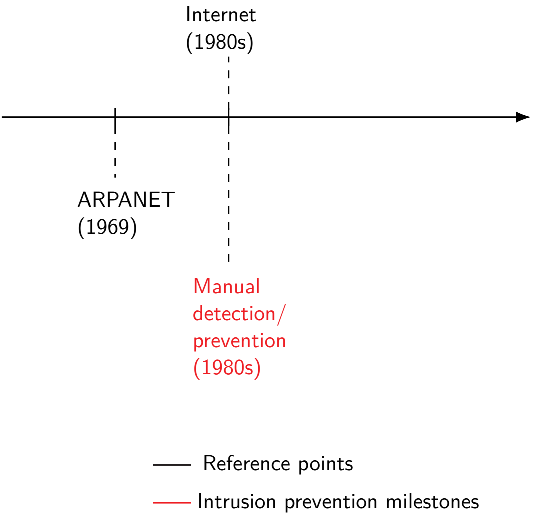
When to take a defensive action?
Which action to take?



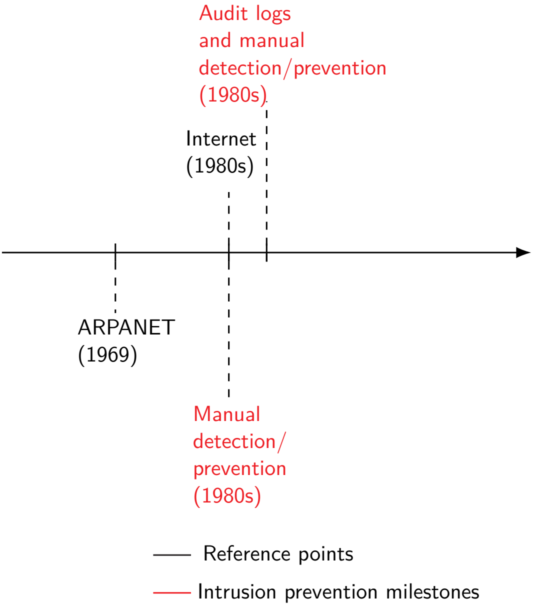
A Brief History of Intrusion Prevention



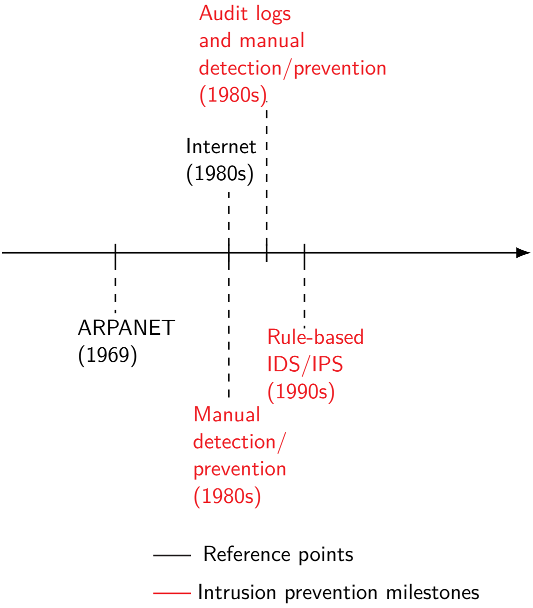
A Brief History of Intrusion Prevention



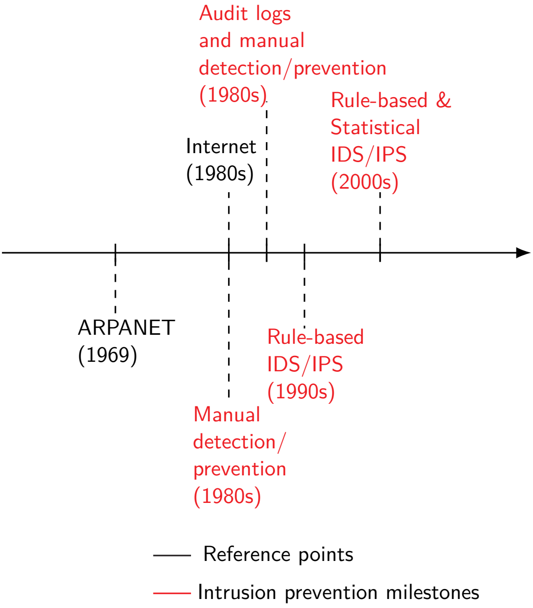
A Brief History of Intrusion Prevention



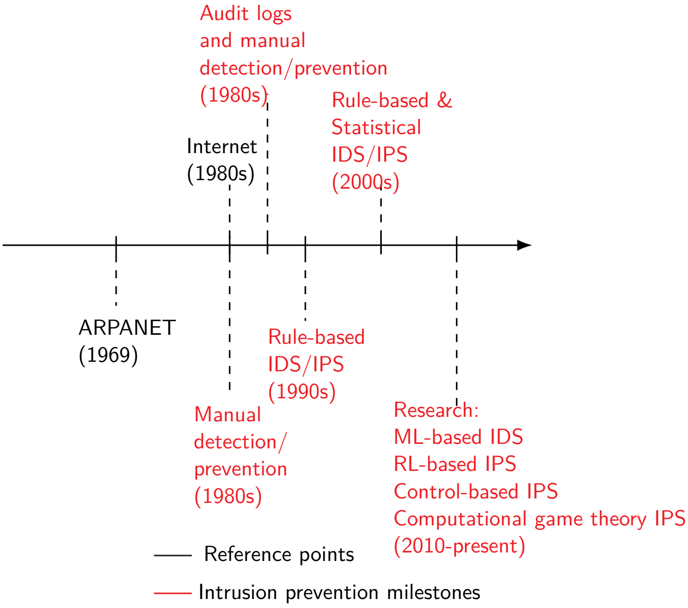
A Brief History of Intrusion Prevention



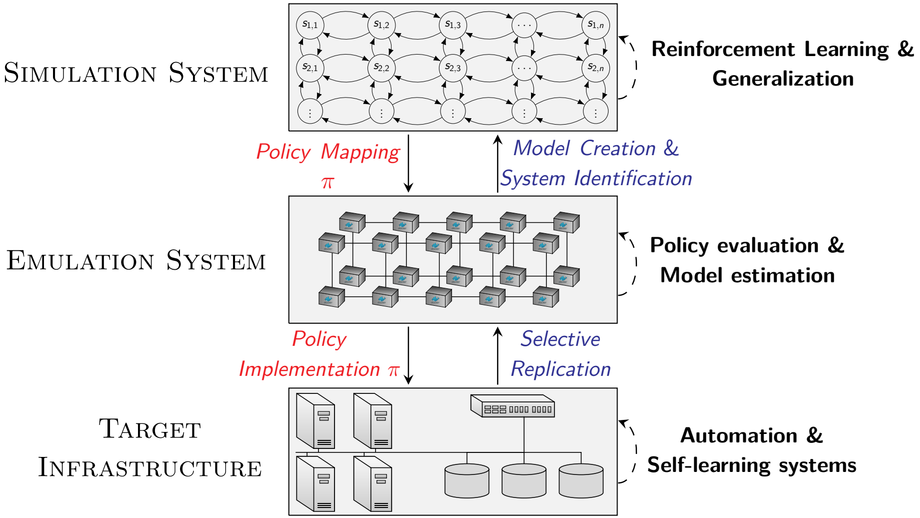
A Brief History of Intrusion Prevention



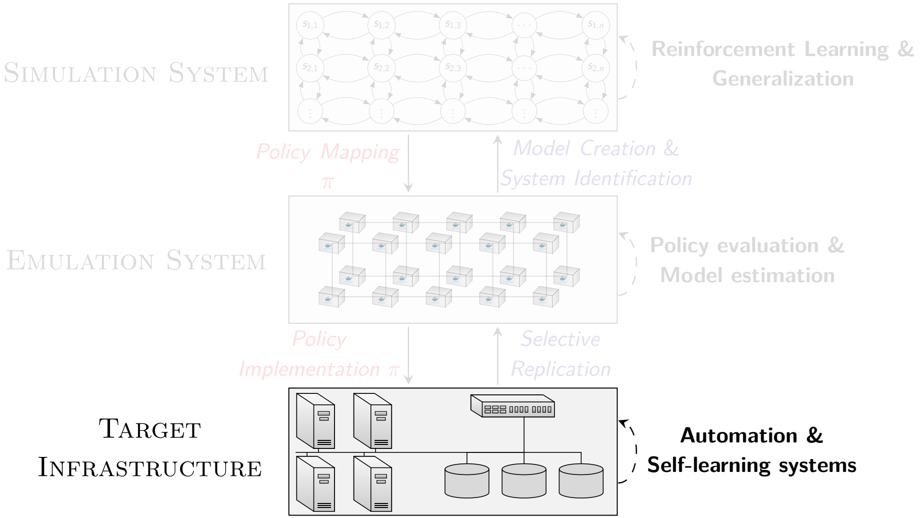
A Brief History of Intrusion Prevention



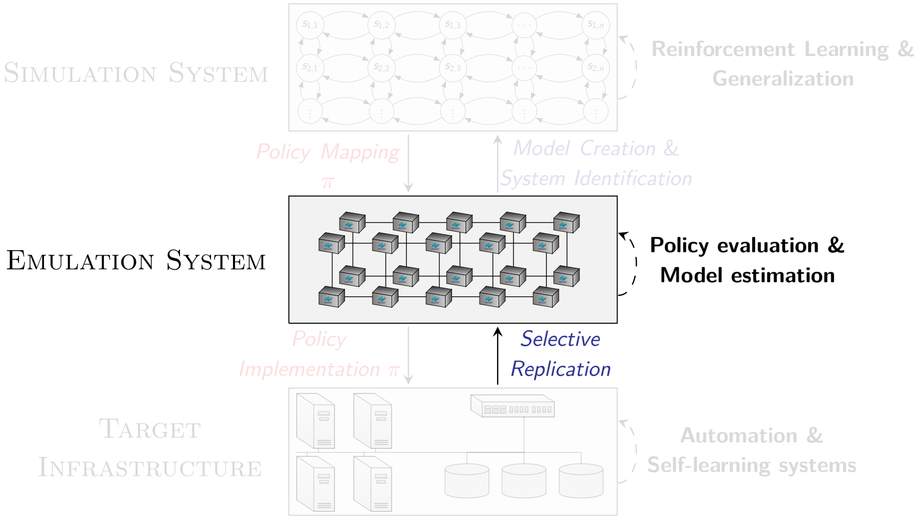
Our Method for Learning Effective Security Strategies



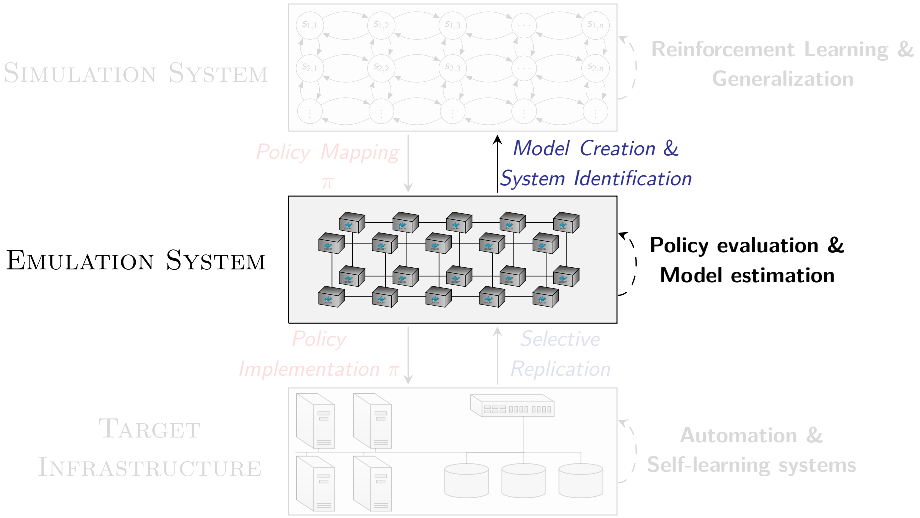
Our Method for Finding Effective Security Strategies



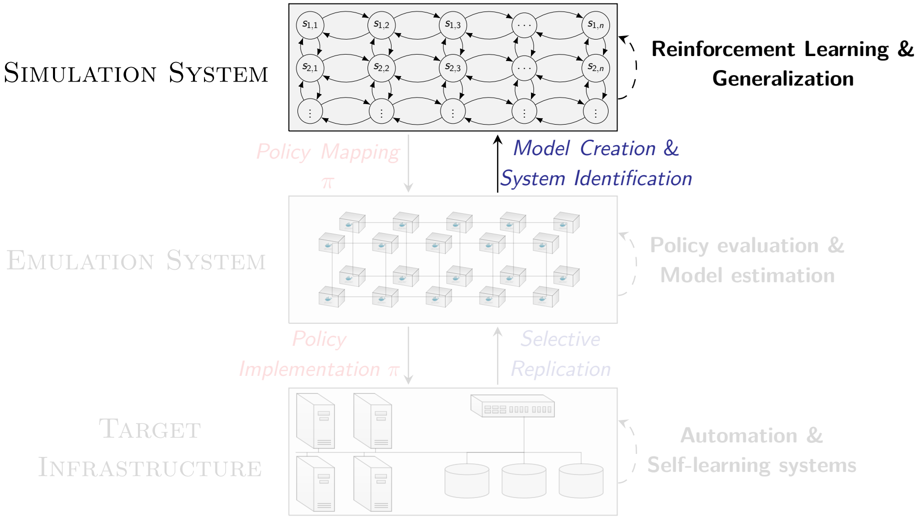
Our Method for Finding Effective Security Strategies



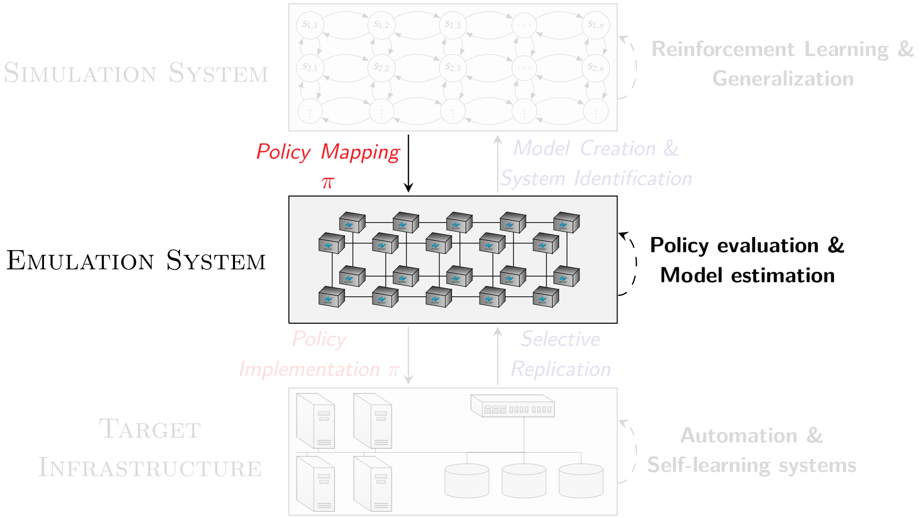
Our Method for Finding Effective Security Strategies



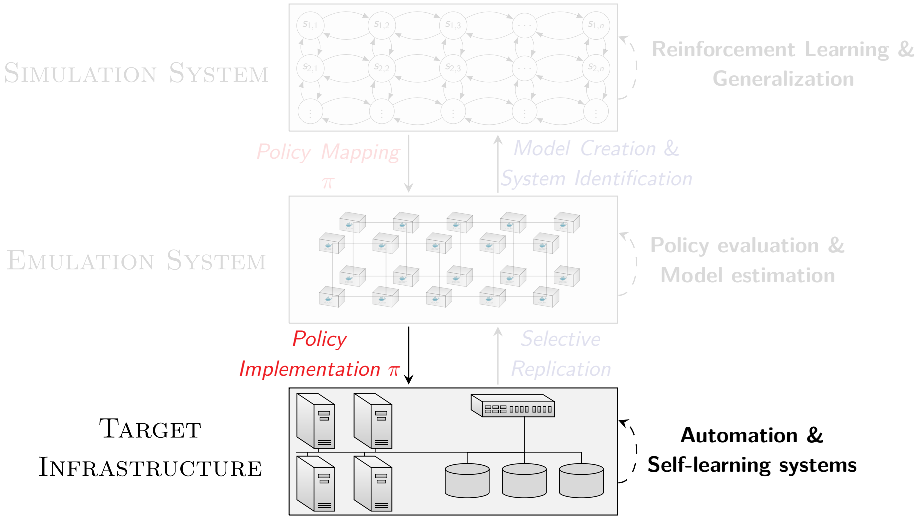
Our Method for Finding Effective Security Strategies



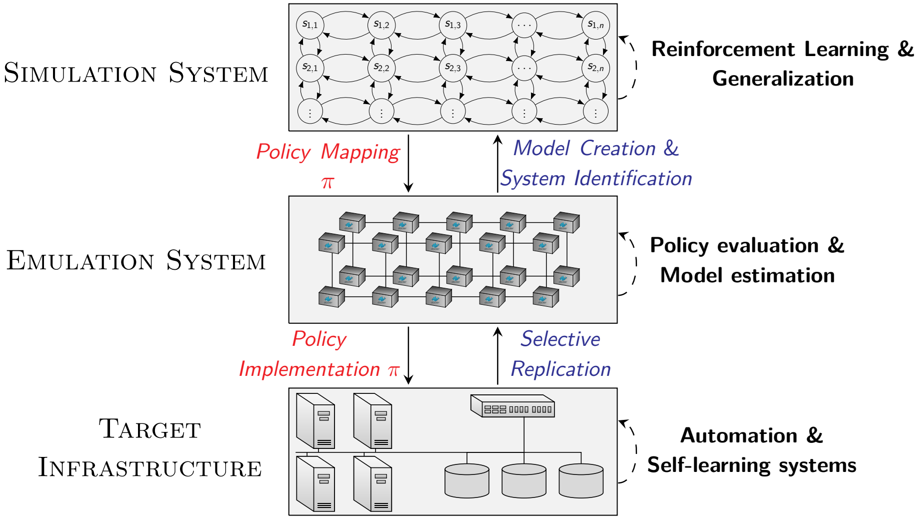
Our Method for Finding Effective Security Strategies



Our Method for Finding Effective Security Strategies



Our Method for Finding Effective Security Strategies



Outline

- **Use Case & Approach:**
  - Intrusion prevention
  - System identification
  - Reinforcement learning and optimal stopping
- **Formal Model of The Use Case**
  - Intrusion prevention as an optimal stopping problem
  - Partially observed Markov decision process
- **Structure of π^***
  - Existence of optimal multi-threshold policy π_l^*
  - Stopping sets $\mathscr{S}_l$ are connected and nested
- **Reinforcement learning method**
  - Learning threshold policies & the policy gradient
  - Emulated infrastructure
- **Results & Conclusion**
  - Numerical evaluation results & Demo
  - Conclusion & future work

Outline

- **Use Case & Approach:**
  - Intrusion prevention
  - System identification
  - Reinforcement learning and optimal stopping
- **Formal Model of The Use Case**
  - Intrusion prevention as an optimal stopping problem
  - Partially observed Markov decision process
- **Structure of π^***
  - Existence of optimal multi-threshold policy π_l^*
  - Stopping sets $\mathscr{S}_l$ are connected and nested
- **Reinforcement learning method**
  - Learning threshold policies & the policy gradient
  - Emulated infrastructure
- **Results & Conclusion**
  - Numerical evaluation results & Demo
  - Conclusion & future work

Outline

- **Use Case & Approach:**
  - Intrusion prevention
  - System identification
  - Reinforcement learning and optimal stopping
- **Formal Model of The Use Case**
  - Intrusion prevention as an optimal stopping problem
  - Partially observed Markov decision process
- **Structure of π^***
  - Existence of optimal multi-threshold policy π_l^*
  - Stopping sets $\mathscr{S}_l$ are connected and nested
- **Reinforcement learning method**
  - Learning threshold policies & the policy gradient
  - Emulated infrastructure
- **Results & Conclusion**
  - Numerical evaluation results & Demo
  - Conclusion & future work

Outline

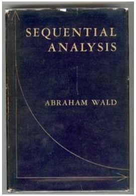
- **Use Case & Approach:**
  - Intrusion prevention
  - System identification
  - Reinforcement learning and optimal stopping
- **Formal Model of The Use Case**
  - Intrusion prevention as an optimal stopping problem
  - Partially observed Markov decision process
- **Structure of π^***
  - Existence of optimal multi-threshold policy π_l^*
  - Stopping sets $\mathscr{S}_l$ are connected and nested
- **Reinforcement learning method**
  - Learning threshold policies & the policy gradient
  - Emulated infrastructure
- **Results & Conclusion**
  - Numerical evaluation results & Demo
  - Conclusion & future work

Outline

- **Use Case & Approach:**
  - Intrusion prevention
  - System identification
  - Reinforcement learning and optimal stopping
- **Formal Model of The Use Case**
  - Intrusion prevention as an optimal stopping problem
  - Partially observed Markov decision process
- **Structure of π^***
  - Existence of optimal multi-threshold policy π_l^*
  - Stopping sets $\mathscr{S}_l$ are connected and nested
- **Reinforcement learning method**
  - Learning threshold policies & the policy gradient
  - Emulated infrastructure
- **Results & Conclusion**
  - Numerical evaluation results & Demo
  - Conclusion & future work

Background: Optimal Stopping

- **History**:
  - Studied in the 18th century to analyze a gambler's fortune
  - Formalized by Abraham Wald in 1947[1]
  - Since then it has been generalized and developed by (Chow[2], Shiryaev & Kolmogorov[3], Bather[4], Bertsekas[5], etc.)



[1] Abraham Wald. *Sequential Analysis*. Wiley and Sons, New York, 1947.

[2] Y. Chow, H. Robbins, and D. Siegmund. "Great expectations: The theory of optimal stopping". In: 1971.

[3] Albert N. Shirayev. *Optimal Stopping Rules*. Reprint of russian edition from 1969. Springer-Verlag Berlin, 2007.

[4] John Bather. *Decision Theory: An Introduction to Dynamic Programming and Sequential Decisions*. USA: John Wiley and Sons, Inc., 2000. ISBN: 0471976490.

[5] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. 3rd. Vol. I. Belmont, MA, USA: Athena Scientific, 2005.

Background: Optimal Stopping

- **The General Problem**:
  - A stochastic process $(s_t)_{t=1}^{T}$ is observed sequentially
  - Two options per t: (i) continue to observe; or (ii) stop
  - Find the *optimal stopping time* τ^*:

$$\tau^* = \arg\max_{\tau} \mathbb{E}_{\tau}\left[\sum_{t=1}^{\tau-1} \gamma^{t-1}\mathcal{R}^{C}_{s_t s_{t+1}} + \gamma^{\tau-1}\mathcal{R}^{S}_{s_\tau s_\tau}\right] \qquad (1)$$

where $\mathcal{R}^{S}_{ss'}$ & $\mathcal{R}^{C}_{ss'}$ are the stop/continue rewards

Background: Optimal Stopping

- **Applications & Use Cases**:
  - Hypothesis testing[6]
  - Change detection[7],
  - Selling decisions[8],
  - Queue management[9],
  - Industrial control[10],
  - Advertisement scheduling[11], etc.

---

[6] Abraham Wald. *Sequential Analysis*. Wiley and Sons, New York, 1947.

[7] Alexander G. Tartakovsky et al. "Detection of intrusions in information systems by sequential change-point methods". In: *Statistical Methodology* (2006). ISSN: 1572-3127. DOI: https://doi.org/10.1016/j.stamet.2005.05.003. URL: https://www.sciencedirect.com/science/article/pii/S1572312705000493.

[8] Jacques du Toit and Goran Peskir. "Selling a stock at the ultimate maximum". In: *The Annals of Applied Probability* 19.3 (2009). ISSN: 1050-5164. DOI: 10.1214/08-aap566. URL: http://dx.doi.org/10.1214/08-AAP566.

[9] Arghyadip Roy et al. "Online Reinforcement Learning of Optimal Threshold Policies for Markov Decision Processes". In: *CoRR* (2019). http://arxiv.org/abs/1912.10325. eprint: 1912.10325.

[10] Maben Rabi and Karl H. Johansson. "Event-Triggered Strategies for Industrial Control over Wireless Networks". In: *Proceedings of the 4th Annual International Conference on Wireless Internet*. WICON '08. Maui, Hawaii, USA, 2008. ISBN: 9789639799363.

[11] Vikram Krishnamurthy, Anup Aprem, and Sujay Bhatt. "Multiple stopping time POMDPs: Structural results & application in interactive advertising on social media". In: *Automatica* 95 (2018), pp. 385–398. ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2018.06.013. URL: https://www.sciencedirect.com/science/article/pii/S0005109818303054.

Background: Optimal Stopping

- **Applications & Use Cases**:
  - Hypothesis testing[12]
  - Change detection[13],
  - Selling decisions[14],
  - Queue management[15],
  - Industrial control[16],
  - Advertisement scheduling,
  - **Intrusion prevention**[17] etc.

---

[12] Abraham Wald. *Sequential Analysis*. Wiley and Sons, New York, 1947.

[13] Alexander G. Tartakovsky et al. "Detection of intrusions in information systems by sequential change-point methods". In: *Statistical Methodology* (2006). ISSN: 1572-3127. DOI: https://doi.org/10.1016/j.stamet.2005.05.003. URL: https://www.sciencedirect.com/science/article/pii/S1572312705000493.

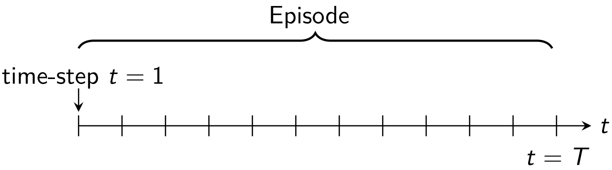
[14] Jacques du Toit and Goran Peskir. "Selling a stock at the ultimate maximum". In: *The Annals of Applied Probability* 19.3 (2009). ISSN: 1050-5164. DOI: 10.1214/08-aap566. URL: http://dx.doi.org/10.1214/08-AAP566.

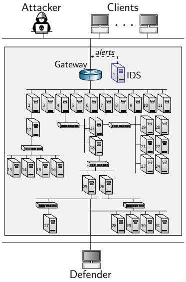
[15] Arghyadip Roy et al. "Online Reinforcement Learning of Optimal Threshold Policies for Markov Decision Processes". In: *CoRR* (2019). http://arxiv.org/abs/1912.10325. eprint: 1912.10325.

[16] Maben Rabi and Karl H. Johansson. "Event-Triggered Strategies for Industrial Control over Wireless Networks". In: *Proceedings of the 4th Annual International Conference on Wireless Internet*. WICON '08. Maui, Hawaii, USA, 2008. ISBN: 9789639799363.

[17] Kim Hammar and Rolf Stadler. "Intrusion Prevention through Optimal Stopping". In: (). 2021, https://arxiv.org/abs/2111.00289. arXiv: 2111.00289.

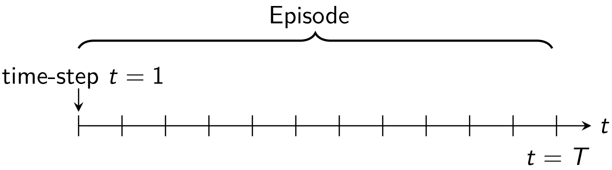
Formulating Intrusion Prevention as a Stopping Problem

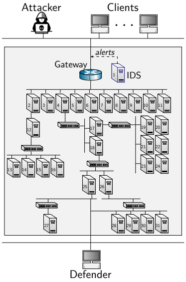




- The system evolves in discrete time-steps.
- Defender observes the infrastructure (IDS, log files, etc.).
- An intrusion occurs at an unknown time.
- The defender can make L stops.
- Each stop is associated with a defensive action
- The final stop shuts down the infrastructure.
- **Based on the observations, when is it optimal to stop?**

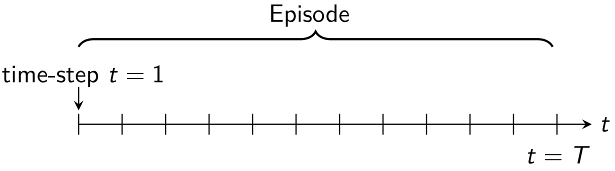
Formulating Intrusion Prevention as a Stopping Problem

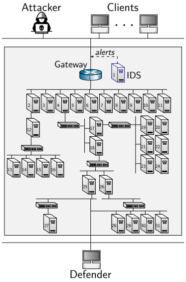




- The system evolves in discrete time-steps.
- Defender observes the infrastructure (IDS, log files, etc.).
- An intrusion occurs at an unknown time.
- The defender can make L stops.
- Each stop is associated with a defensive action
- The final stop shuts down the infrastructure.
- **Based on the observations, when is it optimal to stop?**

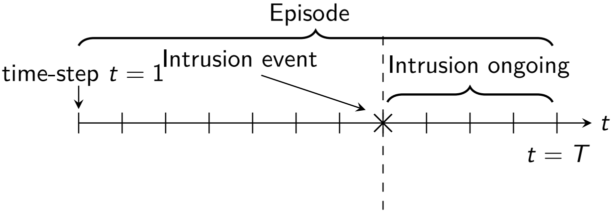
Formulating Intrusion Prevention as a Stopping Problem

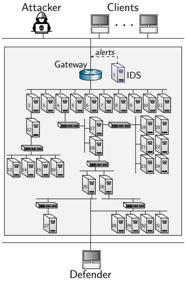




- The system evolves in discrete time-steps.
- Defender observes the infrastructure (IDS, log files, etc.).
- An intrusion occurs at an unknown time.
- The defender can make L stops.
- Each stop is associated with a defensive action
- The final stop shuts down the infrastructure.
- **Based on the observations, when is it optimal to stop?**

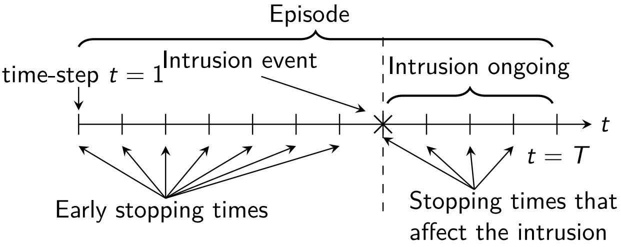
Formulating Intrusion Prevention as a Stopping Problem

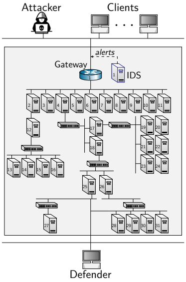




- The system evolves in discrete time-steps.
- Defender observes the infrastructure (IDS, log files, etc.).
- An intrusion occurs at an unknown time.
- The defender can make L stops.
- Each stop is associated with a defensive action
- The final stop shuts down the infrastructure.
- **Based on the observations, when is it optimal to stop?**

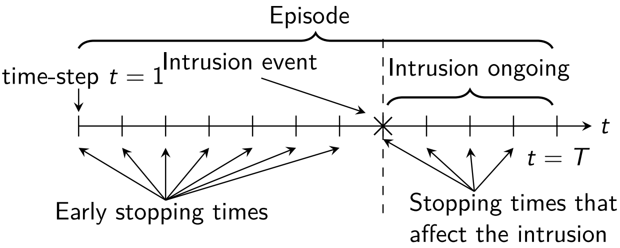
Formulating Intrusion Prevention as a Stopping Problem

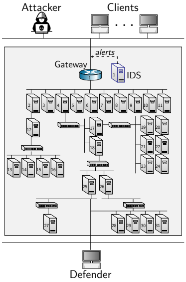




- The system evolves in discrete time-steps.
- Defender observes the infrastructure (IDS, log files, etc.).
- An intrusion occurs at an unknown time.
- The defender can make L stops.
- Each stop is associated with a defensive action
- The final stop shuts down the infrastructure.
- Based on the observations, when is it optimal to stop?

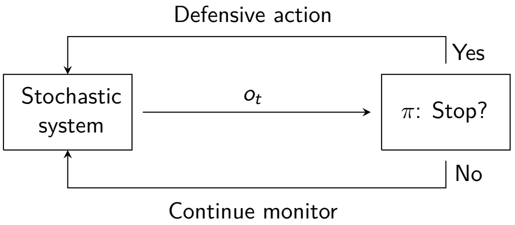
Formulating Intrusion Prevention as a Stopping Problem





- The system evolves in discrete time-steps.
- Defender observes the infrastructure (IDS, log files, etc.).
- An intrusion occurs at an unknown time.
- The defender can make L stops.
- Each stop is associated with a defensive action
- The final stop shuts down the infrastructure.
- **Based on the observations, when is it optimal to stop?**

Intrusion Prevention through Optimal Stopping



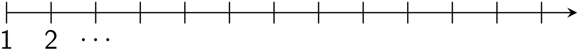
- The $L-l$th **stopping time** τ_l is:

$$\tau_l = \inf\{t : t > \tau_{l-1}, a_t = S\}, \qquad l \in 1, .., L, \ \tau_{L+1} = 0$$

- τ_l is a random variable from sample space Ω to $\mathbb{N}$, which is dependent on $h_{\tau_l} = \rho_1, a_1, o_1, \ldots, a_{\tau_l - 1}, o_{\tau_l}$ and independent of $a_{\tau_l}, o_{\tau_l + 1}, \ldots$

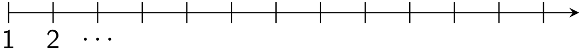
Intrusion Prevention through Optimal Stopping

We consider the class of stopping times
$\mathcal{T}_t = \{\tau_l \leq t | \tau_l > \tau_{l-1}\} \in \mathcal{F}_k$ ($\mathcal{F}_k$ =natural filtration on h_t).

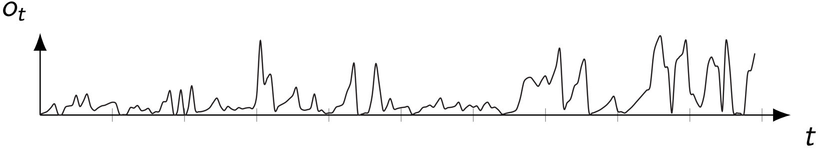


Intrusion Prevention through Optimal Stopping

We consider the class of stopping times
$\mathcal{T}_t = \{\tau_l \leq t | \tau_l > \tau_{l-1}\} \in \mathcal{F}_k$ ($\mathcal{F}_k$ =natural filtration on h_t).

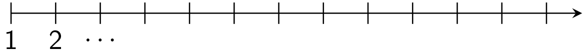


Given the observations:

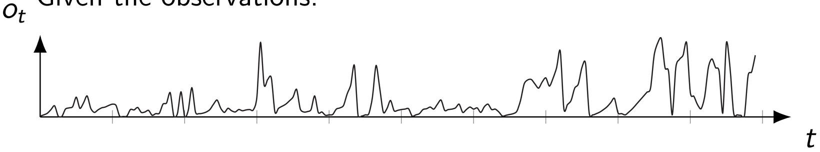


Intrusion Prevention through Optimal Stopping

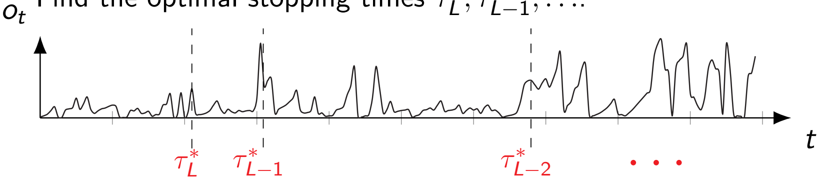
We consider the class of stopping times
$\mathcal{T}_t = \{\tau_l \leq t | \tau_l > \tau_{l+1}\} \in \mathcal{F}_k$ ($\mathcal{F}_k$ =natural filtration on h_t).



Given the observations:

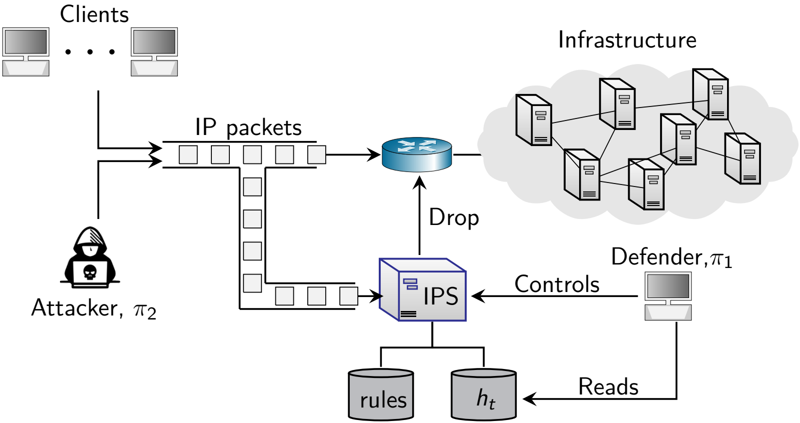


Find the optimal stopping times $\tau^*_L, \tau^*_{L-1}, \ldots$:



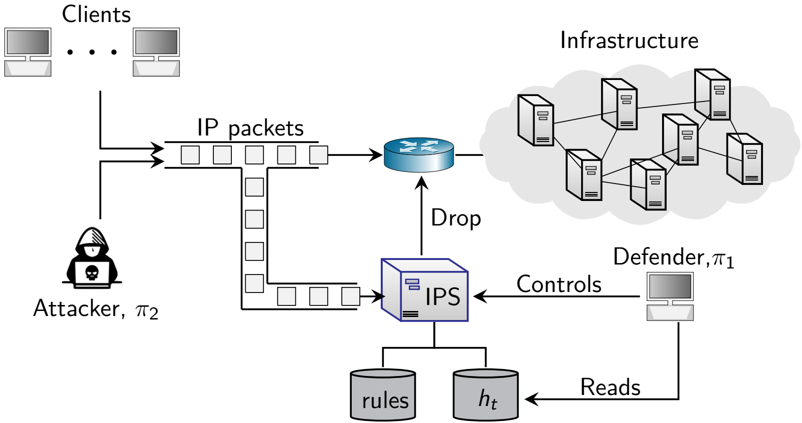
The Defender's Stop Actions

- Ingress traffic goes through deep packet inspection at gateway
- Gateway runs the Snort IDS/IPS and may drop packets
- At each stopping time, we update the IPS configuration



The Defender's Stop Actions

- Ingress traffic goes through deep packet inspection at gateway
- Gateway runs the Snort IDS/IPS and may drop packets
- At each stopping time, we update the IPS configuration
- **Objective:** find optimal π^* or Nash equilibrium



Approaches to Solving Optimal Stopping Problems

Two main approaches:

- The *Markovian approach*
  - Assume process is Markov
  - Utilize Markov deision theory
- The *martingale approach*
  - More general
  - No Markov assumption
  - Utilize martingale convergence theorems

The Markovian Approach to Optimal Stopping

- Model the problem as a MDP or POMDP
- A policy π^* that satisfies the Bellman-Wald equation is optimal:

$$\pi^*(s) = \arg\max_{\{S,C\}} \left[\underbrace{\mathbb{E}\left[\mathcal{R}_s^S\right]}_{\text{stop}}, \underbrace{\mathbb{E}\left[\mathcal{R}_s^C + \gamma V^*(s')\right]}_{\text{continue}} \right] \quad \forall s \in \mathcal{S}$$

- Solve by backward induction, dynamic programming, or reinforcement learning

- **Alternative optimality condition**:
  - Theorem: $V^*(s)$ is the minimal excessive function which majorizes $R_s^{\emptyset}$.
  - Assume all rewards are received upon stopping: $R_s^{\emptyset}$
  - $V^*(s)$ **majorizes** $R_s^{\emptyset}$ if $V^*(s) \geq R_s^{\emptyset}$ $\forall s \in \mathcal{S}$
  - $V^*(s)$ is **excessive** if $V^*(s) \geq \sum_{s'} \mathcal{P}_{s's}^C V^*(s')$ $\forall s \in \mathcal{S}$

The Markovian Approach to Optimal Stopping

- Model the problem as a MDP or POMDP
- A policy π^* that satisfies the Bellman-Wald equation is optimal:

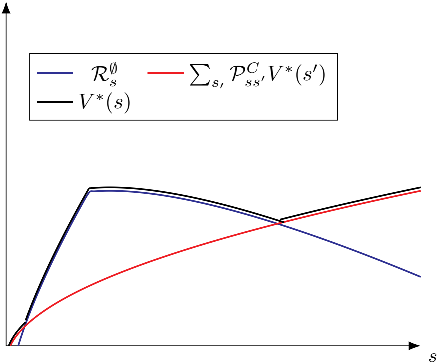
$$\pi^*(s) = \underset{\{S,C\}}{\arg\max} \left[\underbrace{\mathbb{E}\left[\mathcal{R}_s^S\right]}_{\text{stop}}, \underbrace{\mathbb{E}\left[\mathcal{R}_s^C + \gamma V^*(s')\right]}_{\text{continue}} \right] \quad \forall s \in \mathcal{S}$$

- Solve by backward induction, dynamic programming, or reinforcement learning

- **Alternative optimality condition**:
  - Theorem: $V^*(s)$ is the minimal excessive function which majorizes $R_s^{\emptyset}$.
  - Assume all rewards are received upon stopping: $R_s^{\emptyset}$
  - $V^*(s)$ **majorizes** $R_s^{\emptyset}$ if $V^*(s) \geq R_s^{\emptyset}$ $\forall s \in \mathcal{S}$
  - $V^*(s)$ is **excessive** if $V^*(s) \geq \sum_{s'} \mathcal{P}^C_{s's} V^*(s')$ $\forall s \in \mathcal{S}$

The Markovian Approach to Optimal Stopping

- **Alternative optimality condition**:
  - Theorem: $V^*(s)$ is the minimal excessive function which majorizes $R_s^{\emptyset}$.
  - Assume all rewards are received upon stopping: $R_s^{\emptyset}$
  - $V^*(s)$ **majorizes** $R_s^{\emptyset}$ if $V^*(s) \geq R_s^{\emptyset}\ \forall s \in \mathcal{S}$
  - $V^*(s)$ is **excessive** if $V^*(s) \geq \sum_{s'} \mathcal{P}_{s's}^{C} V^*(s')\ \forall s \in \mathcal{S}$



The Martingale Approach to Optimal Stopping

- Model the state process as an arbitrary stochastic process
- The reward of the optimal stopping time is given by the Snell envelope[18].
- Snell envelope: **smallest supermartingale that stochastically dominates the process**

[18] J. L. Snell. "Applications of martingale system theorems". In: *Transactions of the American Mathematical Society* 73 (1952), pp. 293–312.

The Martingale Approach to Optimal Stopping

- Model the state process as an arbitrary stochastic process
- The reward of the optimal stopping time is given by the

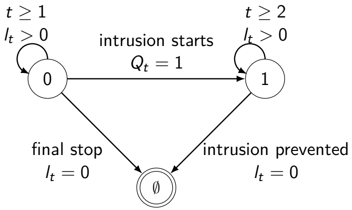
We follow the Markovian approach and model the problem as a POMDP

stochastically dominates the process

[19] J. L. Snell. "Applications of martingale system theorems". In: *Transactions of the American Mathematical Society* 73 (1952), pp. 293–312.

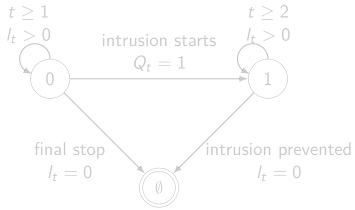
A Partially Observed MDP Model for the Defender

- **States:**
  - Intrusion state $s_t \in \{0,1\}$, terminal $\emptyset$.
- **Observations:**
  - Severe/Warning IDS Alerts $(\Delta x, \Delta y)$, Login attempts Δz, stops remaining $l_t \in \{1,..,L\}$, $f_{XYZ}(\Delta x, \Delta y, \Delta z | s_t)$
- **Actions:**
  - "Stop" (S) and "Continue" (C)
- **Rewards:**
  - Reward: security and service. Penalty: false alarms and intrusions
- **Transition probabilities:**
  - Bernoulli process $(Q_t)_{t=1}^{T} \sim Ber(p)$ defines intrusion start $I_t \sim Ge(p)$
- **Objective and Horizon:**
  - $\max \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T_\emptyset} r(s_t, a_t)\right]$, $T_\emptyset$



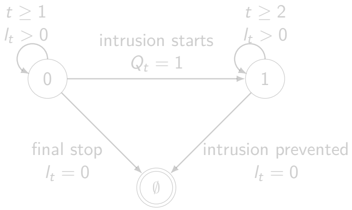
A Partially Observed MDP Model for the Defender

- **States:**
  - Intrusion state $s_t \in \{0, 1\}$, terminal $\emptyset$.
- **Observations:**
  - Severe/Warning IDS Alerts $(\Delta x, \Delta y)$, Login attempts Δz, stops remaining $l_t \in \{1, .., L\}$, $f_{XYZ}(\Delta x, \Delta y, \Delta z | s_t)$
- **Actions:**
  - "Stop" (S) and "Continue" (C)
- **Rewards:**
  - Reward: security and service. Penalty: false alarms and intrusions
- **Transition probabilities:**
  - Bernoulli process $(Q_t)_{t=1}^{T} \sim Ber(p)$ defines intrusion start $I_t \sim Ge(p)$
- **Objective and Horizon:**
  - $\max \mathbb{E}_{\pi_\theta} \left[\sum_{t=1}^{T_\emptyset} r(s_t, a_t) \right]$, $T_\emptyset$



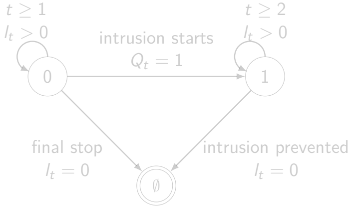
A Partially Observed MDP Model for the Defender

- **States:**
  - Intrusion state $s_t \in \{0, 1\}$, terminal $\emptyset$.
- **Observations:**
  - Severe/Warning IDS Alerts $(\Delta x, \Delta y)$, Login attempts Δz, stops remaining $l_t \in \{1, .., L\}$, $f_{XYZ}(\Delta x, \Delta y, \Delta z | s_t)$
- **Actions:**
  - "Stop" (S) and "Continue" (C)
- **Rewards:**
  - Reward: security and service. Penalty: false alarms and intrusions
- **Transition probabilities:**
  - Bernoulli process $(Q_t)_{t=1}^{T} \sim Ber(p)$ defines intrusion start $I_t \sim Ge(p)$
- **Objective and Horizon:**
  - $\max \mathbb{E}_{\pi_\theta} \left[\sum_{t=1}^{T_\emptyset} r(s_t, a_t) \right], \ T_\emptyset$



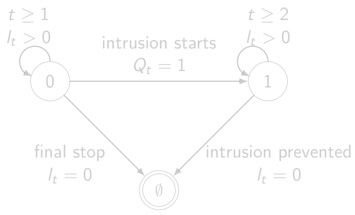
A Partially Observed MDP Model for the Defender

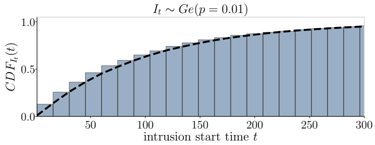
- **States:**
  - Intrusion state $s_t \in \{0, 1\}$, terminal $\emptyset$.
- **Observations:**
  - Severe/Warning IDS Alerts $(\Delta x, \Delta y)$, Login attempts Δz, stops remaining $l_t \in \{1, .., L\}$, $f_{XYZ}(\Delta x, \Delta y, \Delta z | s_t)$
- **Actions:**
  - "Stop" (S) and "Continue" (C)
- **Rewards:**
  - Reward: security and service. Penalty: false alarms and intrusions
- **Transition probabilities:**
  - Bernoulli process $(Q_t)_{t=1}^{T} \sim Ber(p)$ defines intrusion start $I_t \sim Ge(p)$
- **Objective and Horizon:**
  - $\max \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T_\emptyset} r(s_t, a_t)\right]$, $T_\emptyset$



A Partially Observed MDP Model for the Defender

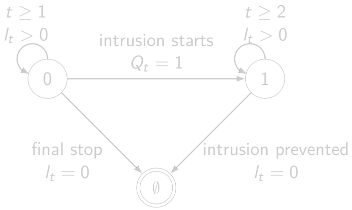
- **States:**
  - Intrusion state $s_t \in \{0,1\}$, terminal $\emptyset$.
- **Observations:**
  - Severe/Warning IDS Alerts $(\Delta x, \Delta y)$, Login attempts Δz, stops remaining $l_t \in \{1,..,L\}$, $f_{XYZ}(\Delta x, \Delta y, \Delta z | s_t)$
- **Actions:**
  - "Stop" (S) and "Continue" (C)
- **Rewards:**
  - Reward: security and service. Penalty: false alarms and intrusions
- **Transition probabilities:**
  - Bernoulli process $(Q_t)_{t=1}^{T} \sim Ber(p)$ defines intrusion start $I_t \sim Ge(p)$
- **Objective and Horizon:**
  - $\max \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T_\emptyset} r(s_t, a_t)\right]$, $T_\emptyset$





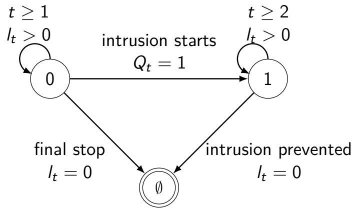
A Partially Observed MDP Model for the Defender

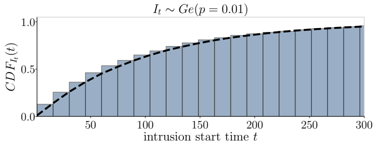
- **States:**
  - Intrusion state $s_t \in \{0, 1\}$, terminal $\emptyset$.
- **Observations:**
  - Severe/Warning IDS Alerts $(\Delta x, \Delta y)$, Login attempts Δz, stops remaining $l_t \in \{1, .., L\}$, $f_{XYZ}(\Delta x, \Delta y, \Delta z | s_t)$
- **Actions:**
  - "Stop" (S) and "Continue" (C)
- **Rewards:**
  - Reward: security and service. Penalty: false alarms and intrusions
- **Transition probabilities:**
  - Bernoulli process $(Q_t)_{t=1}^{T} \sim Ber(p)$ defines intrusion start $I_t \sim Ge(p)$
- **Objective and Horizon:**
  - $\max \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T_\emptyset} r(s_t, a_t)\right], \ T_\emptyset$



A Partially Observed MDP Model for the Defender

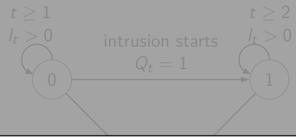
- **States:**
  - Intrusion state $s_t \in \{0, 1\}$, terminal $\emptyset$.
- **Observations:**
  - Severe/Warning IDS Alerts $(\Delta x, \Delta y)$, Login attempts Δz, stops remaining $l_t \in \{1, .., L\}$, $f_{XYZ}(\Delta x, \Delta y, \Delta z | s_t)$
- **Actions:**
  - "Stop" (S) and "Continue" (C)
- **Rewards:**
  - Reward: security and service. Penalty: false alarms and intrusions
- **Transition probabilities:**
  - Bernoulli process $(Q_t)_{t=1}^{T} \sim Ber(p)$ defines intrusion start $I_t \sim Ge(p)$
- **Objective and Horizon:**
  - $\max \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T_\emptyset} r(s_t, a_t)\right]$, $T_\emptyset$





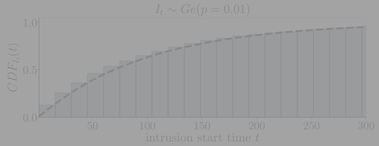
A Partially Observed MDP Model for the Defender

- **States:**
  - Intrusion state $s_t \in \{0, 1\}$, terminal $\emptyset$.
- **Observations:**
  - Severe/Warning IDS Alerts $(\Delta x, \Delta y)$, Login attempts Δz, stops remaining $l_t \in \{1, .., L\}$, $f_{XYZ}(\Delta x, \Delta y, \Delta z | s_t)$



We analyze the structure of π^* using POMDP & stopping theory

  - Reward: security and service. Penalty: false alarms and intrusions
- **Transition probabilities:**
  - Bernoulli process $(Q_t)_{t=1}^{T} \sim Ber(p)$ defines intrusion start $I_t \sim Ge(p)$

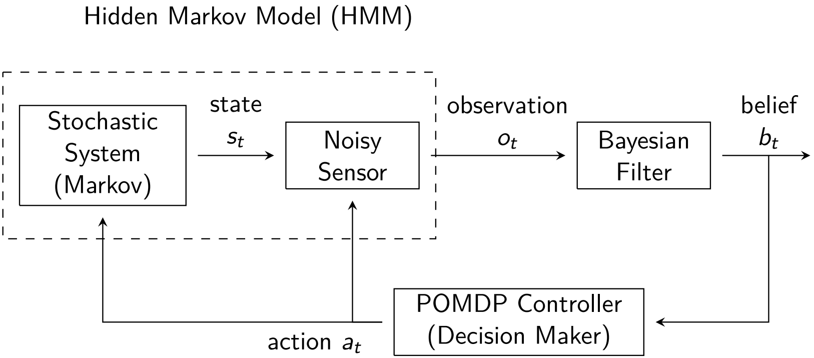


- **Objective and Horizon:**
  - $\max \mathbb{E}_{\pi_\theta} \left[\sum_{t=1}^{T_\emptyset} r(s_t, a_t) \right]$, $T_\emptyset$

Outline

- **Use Case & Approach:**
  - Intrusion prevention
  - System identification
  - Reinforcement learning and optimal stopping
- **Formal Model of The Use Case**
  - Intrusion prevention as an optimal stopping problem
  - Partially observed Markov decision process
- **Structure of π^***
  - Existence of optimal multi-threshold policy π_l^*
  - Stopping sets $\mathscr{S}_l$ are connected and nested
- **Reinforcement learning method**
  - Learning threshold policies & the policy gradient
  - Emulated infrastructure
- **Results & Conclusion**
  - Numerical evaluation results & Demo
  - Conclusion & future work

Background: POMDPs



- **POMDP:** $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}^{a_t}_{s_t, s_{t+1}}, \mathcal{R}^{a_t}_{s_t, s_{t+1}}, \gamma, \rho_1, T, \mathcal{O}, \mathcal{Z} \rangle$
- Controlled hidden Markov model, states $s_t \in \mathcal{S}$ are hidden
- Agent observes history $h_t = (\rho_1, a_1, o_1, \ldots, a_{t-1}, o_t) \in \mathcal{H}$

Background: POMDPs

- s_t is Markov: $\mathbb{P}[s_{t+1}|s_t] = \mathbb{P}[s_{t+1}|s_1, \ldots, s_t]$
- $\implies \pi^*(a_t|h_t) = \pi^*(a_t|\mathbb{P}[s_t|h_t]) = \pi^*(a_t|b_t)$
- **Optimality (Bellman) Eq:**

$$\pi^*(b) \in \arg\max_{a \in \mathcal{A}} \left[\sum_s b(s)\mathcal{R}_s^a + \gamma \sum_{o,s,s'} \mathcal{Z}(o, s', a) b(s) \mathcal{P}_{ss'}^a V^*(b_a^o) \right]$$

-

$$\begin{aligned}
\mathbb{P}[s_t|h_t] &= \mathbb{P}[s_t|o_t, a_{t-1}, h_{t-1}] \\
&= \frac{\mathbb{P}[o_t|s_t, a_{t-1}, h_{t-1}]\mathbb{P}[s_t|a_{t-1}, h_{t-1}]}{\mathbb{P}[o_t|a_{t-1}, h_{t-1}]} && \text{Bayes} \\
&= \frac{\mathcal{Z}(o_t, s_t, a_{t-1}) \sum_{s_{t-1}} \mathcal{P}_{s_{t-1}s_t}^{a_{t-1}} \mathbb{P}[s_{t-1}|h_{t-1}]}{\sum_{s'} \sum_s \mathcal{Z}(o_t, s', a_{t-1}) \mathbb{P}[s_{t-1}|h_{t-1}]} && \text{Markov}
\end{aligned}$$

- $\mathbb{P}[s_{t-1}|h_{t-1}]$ **with** a_t, o_t **is a sufficient statistic for** s_t
- $b_t \triangleq \mathbb{P}[s_{t-1}|h_{t-1}]$**: belief state at time** t
- b_t **computed recursively using the equation above**

Background: POMDPs

- s_t is Markov: $\mathbb{P}[s_{t+1}|s_t] = \mathbb{P}[s_{t+1}|s_1, \ldots, s_t]$
- $\implies \pi^*(a_t|h_t) = \pi^*(a_t|\mathbb{P}[s_t|h_t]) = \pi^*(a_t|b_t)$
- **Optimality (Bellman) Eq:**

$$\pi^*(b) \in \arg\max_{a\in\mathcal{A}} \left[\sum_s b(s)\mathcal{R}_s^a + \gamma \sum_{o,s,s'} \mathcal{Z}(o,s',a)b(s)\mathcal{P}_{ss'}^a V^*(b_a^o) \right]$$

-

$$\begin{aligned}
\mathbb{P}[s_t|h_t] &= \mathbb{P}[s_t|o_t, a_{t-1}, h_{t-1}] \\
&= \frac{\mathbb{P}[o_t|s_t, a_{t-1}, h_{t-1}]\mathbb{P}[s_t|a_{t-1}, h_{t-1}]}{\mathbb{P}[o_t|a_{t-1}, h_{t-1}]} && \text{Bayes} \\
&= \frac{\mathcal{Z}(o_t, s_t, a_{t-1}) \sum_{s_{t-1}} \mathcal{P}_{s_{t-1}s_t}^{a_{t-1}} \mathbb{P}[s_{t-1}|h_{t-1}]}{\sum_{s'} \sum_s \mathcal{Z}(o_t, s', a_{t-1}) \mathbb{P}[s_{t-1}|h_{t-1}]} && \text{Markov}
\end{aligned}$$

- $\mathbb{P}[s_{t-1}|h_{t-1}]$ **with** a_t, o_t **is a sufficient statistic for** s_t
- $b_t \triangleq \mathbb{P}[s_{t-1}|h_{t-1}]$**: belief state at time** t
- b_t **computed recursively using the equation above**

Background: POMDPs

- s_t is Markov: $\mathbb{P}\left[s_{t+1}|s_t\right] = \mathbb{P}\left[s_{t+1}|s_1, \ldots, s_t\right]$
- $\implies \pi^*(a_t|h_t) = \pi^*(a_t|\mathbb{P}[s_t|h_t]) = \pi^*(a_t|b_t)$
- **Optimality (Bellman) Eq:**

$$\pi^*(b) \in \arg\max_{a \in \mathcal{A}} \left[\sum_s b(s)\mathcal{R}_s^a + \gamma \sum_{o,s,s'} \mathcal{Z}(o, s', a) b(s) \mathcal{P}_{ss'}^a V^*(b_a^o) \right]$$

-

$$\begin{aligned}
\mathbb{P}[s_t|h_t] &= \mathbb{P}[s_t|o_t, a_{t-1}, h_{t-1}] \\
&= \frac{\mathbb{P}[o_t|s_t, a_{t-1}, h_{t-1}]\mathbb{P}[s_t|a_{t-1}, h_{t-1}]}{\mathbb{P}[o_t|a_{t-1}, h_{t-1}]} && \text{Bayes} \\
&= \frac{\mathcal{Z}(o_t, s_t, a_{t-1}) \sum_{s_{t-1}} \mathcal{P}_{s_{t-1}s_t}^{a_{t-1}} \mathbb{P}[s_{t-1}|h_{t-1}]}{\sum_{s'} \sum_s \mathcal{Z}(o_t, s', a_{t-1}) \mathbb{P}[s_{t-1}|h_{t-1}]} && \text{Markov}
\end{aligned}$$

- $\mathbb{P}[s_{t-1}|h_{t-1}]$ **with** a_t, o_t **is a sufficient statistic for** s_t
- $b_t \triangleq \mathbb{P}[s_{t-1}|h_{t-1}]$: **belief state at time** t
- b_t **computed recursively using the equation above**

Background: POMDPs

- s_t is Markov: $\mathbb{P}[s_{t+1}|s_t] = \mathbb{P}[s_{t+1}|s_1, \ldots, s_t]$
- $\implies \pi^*(a_t|h_t) = \pi^*(a_t|\mathbb{P}[s_t|h_t]) = \pi^*(a_t|b_t)$
- **Optimality (Bellman) Eq:**

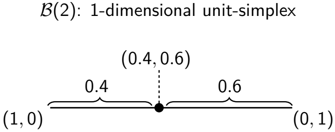
$$\pi^*(b) \in \arg\max_{a \in \mathcal{A}} \left[\sum_s b(s)\mathcal{R}_s^a + \gamma \sum_{o,s,s'} \mathcal{Z}(o, s', a) b(s) \mathcal{P}_{ss'}^a V^*(b_a^o) \right]$$

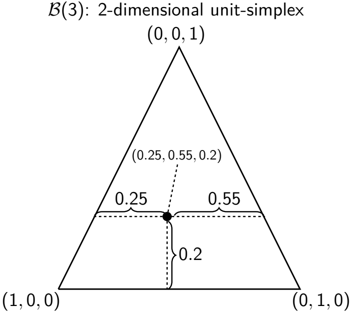
-

$$\begin{aligned}
\mathbb{P}[s_t|h_t] &= \mathbb{P}[s_t|o_t, a_{t-1}, h_{t-1}] \\
&= \frac{\mathbb{P}[o_t|s_t, a_{t-1}, h_{t-1}]\mathbb{P}[s_t|a_{t-1}, h_{t-1}]}{\mathbb{P}[o_t|a_{t-1}, h_{t-1}]} && \text{Bayes} \\
&= \frac{\mathcal{Z}(o_t, s_t, a_{t-1}) \sum_{s_{t-1}} \mathcal{P}_{s_{t-1}s_t}^{a_{t-1}} \mathbb{P}[s_{t-1}|h_{t-1}]}{\sum_{s'} \sum_s \mathcal{Z}(o_t, s', a_{t-1}) \mathbb{P}[s_{t-1}|h_{t-1}]} && \text{Markov}
\end{aligned}$$

- $\mathbb{P}[s_{t-1}|h_{t-1}]$ **with** a_t, o_t **is a sufficient statistic for** s_t
- $b_t \triangleq \mathbb{P}[s_{t-1}|h_{t-1}]$: **belief state at time** t
- b_t **computed recursively using the equation above**

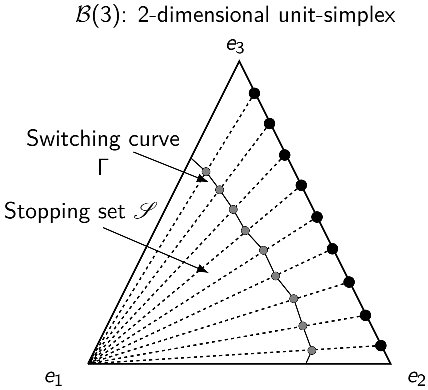
Background: POMDPs





- $b \in \mathcal{B}$, $\mathcal{B}$ is the unit $(|\mathcal{S}| - 1)$-simplex
- **To characterize π^*, partition $\mathcal{B}$ based on $\pi^*(a|b)$**
  - e.g. stopping set $\mathscr{S}$ and continuation set $\mathscr{C}$

Background: POMDPs



- $b \in \mathcal{B}$, $\mathcal{B}$ is the unit $(|\mathcal{S}| - 1)$-simplex
- **To characterize π^*, partition $\mathcal{B}$ based on $\pi^*(a|b)$**
  - e.g. stopping set $\mathscr{S}$ and continuation set $\mathscr{C}$

Structural Result: Optimal Multi-Threshold Policy

Theorem

Given the intrusion prevention POMDP, the following holds:

1. $\mathscr{S}_{l-1} \subseteq \mathscr{S}_l$ *for* $l = 2, \dots L$.
2. *If* $L = 1$, *there exists an optimal threshold* $\alpha^* \in [0, 1]$ *and an optimal policy of the form:*

$$\pi_L^*(b(1)) = S \iff b(1) \geq \alpha^* \tag{2}$$

3. *If* $L \geq 1$ *and* f_{XYZ} *is totally positive of order 2 (TP2), there exists* L *optimal thresholds* $\alpha_l^* \in [0, 1]$ *and an optimal policy of the form:*

$$\pi_l^*(b(1)) = S \iff b(1) \geq \alpha_l^*, \qquad l = 1, \dots, L \tag{3}$$

where α_l^* *is decreasing in* l.

Structural Result: Optimal Multi-Threshold Policy

Theorem

Given the intrusion prevention POMDP, the following holds:

1. $\mathscr{S}_{l-1} \subseteq \mathscr{S}_l$ *for* $l = 2, \ldots L$.
2. *If* $L = 1$, *there exists an optimal threshold* $\alpha^* \in [0, 1]$ *and an optimal policy of the form:*

$$\pi_L^*(b(1)) = S \iff b(1) \geq \alpha^* \tag{4}$$

3. *If* $L \geq 1$ *and* f_{XYZ} *is totally positive of order 2 (TP2), there exists* L *optimal thresholds* $\alpha_l^* \in [0, 1]$ *and an optimal policy of the form:*

$$\pi_l^*(b(1)) = S \iff b(1) \geq \alpha_l^*, \qquad l = 1, \ldots, L \tag{5}$$

where α_l^* *is decreasing in* l.

Structural Result: Optimal Multi-Threshold Policy

Theorem

Given the intrusion prevention POMDP, the following holds:

1. $\mathscr{S}_{l-1} \subseteq \mathscr{S}_l$ *for* $l = 2, \ldots L$.
2. *If* $L = 1$, *there exists an optimal threshold* $\alpha^* \in [0, 1]$ *and an optimal policy of the form:*

$$\pi_L^*(b(1)) = S \iff b(1) \geq \alpha^* \tag{6}$$

3. *If* $L \geq 1$ *and* f_{XYZ} *is totally positive of order 2 (TP2), there exists* L *optimal thresholds* $\alpha_l^* \in [0, 1]$ *and an optimal policy of the form:*

$$\pi_l^*(b(1)) = S \iff b(1) \geq \alpha_l^*, \qquad l = 1, \ldots, L \tag{7}$$

where α_l^* *is decreasing in* l.

Structural Result: Optimal Multi-Threshold Policy

Theorem

Given the intrusion prevention POMDP, the following holds:

1. $\mathscr{S}_{l-1} \subseteq \mathscr{S}_l$ *for* $l = 2, \ldots L$.
2. *If* $L = 1$, *there exists an optimal threshold* $\alpha^* \in [0, 1]$ *and an optimal policy of the form:*

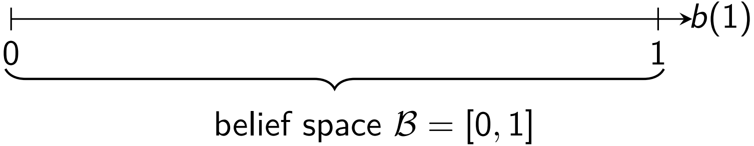
$$\pi_L^*(b(1)) = S \iff b(1) \geq \alpha^* \tag{8}$$

3. *If* $L \geq 1$ *and* f_{XYZ} *is totally positive of order 2 (TP2), there exists* L *optimal thresholds* $\alpha_l^* \in [0, 1]$ *and an optimal policy of the form:*

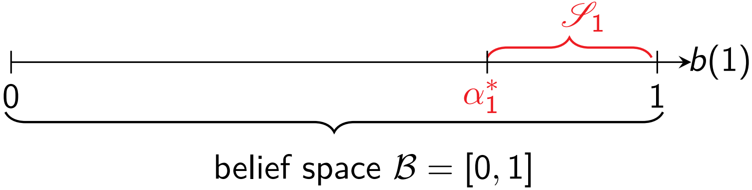
$$\pi_l^*(b(1)) = S \iff b(1) \geq \alpha_l^*, \qquad l = 1, \ldots, L \tag{9}$$

where α_l^* *is decreasing in* l.

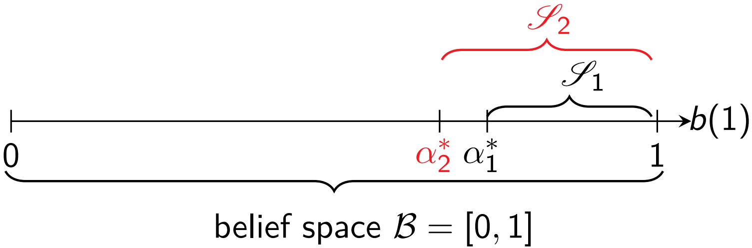
Structural Result: Optimal Multi-Threshold Policy



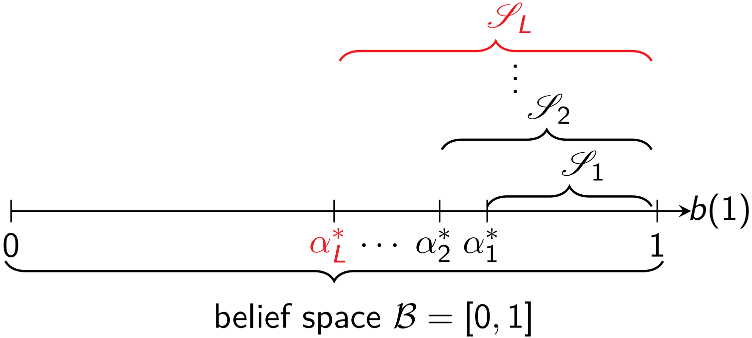
Structural Result: Optimal Multi-Threshold Policy



Structural Result: Optimal Multi-Threshold Policy



Structural Result: Optimal Multi-Threshold Policy



Outline

- **Use Case & Approach:**
  - Intrusion prevention
  - System identification
  - Reinforcement learning and optimal stopping
- **Formal Model of The Use Case**
  - Intrusion prevention as an optimal stopping problem
  - Partially observed Markov decision process
- **Structure of π^***
  - Existence of optimal multi-threshold policy π^*_l
  - Stopping sets $\mathscr{S}_l$ are connected and nested
- **Reinforcement learning method**
  - Learning threshold policies & the policy gradient
  - Emulated infrastructure
- **Results & Conclusion**
  - Numerical evaluation results & Demo
  - Conclusion & future work

Our Reinforcement Learning Algorithm for Learning Threshold Policies

- We **use the structural result** that an optimal threshold policy exist (Theorem 1) **to design an efficient reinforcement learning algorithm**.
- We seek to learn L thresholds: $\alpha_1^*, \alpha_2^*, \ldots, \alpha_L^*$
- We learn these thresholds iteratively through Robbins and Monro's stochastic approximation algorithm.[20]

[20] Herbert Robbins and Sutton Monro. "A Stochastic Approximation Method". In: *The Annals of Mathematical Statistics* 22.3 (1951), pp. 400 –407. DOI: 10.1214/aoms/1177729586. URL: https://doi.org/10.1214/aoms/1177729586.

Our Reinforcement Learning Algorithm for Learning Threshold Policies

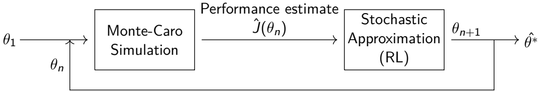
- We **use the structural result** that an optimal threshold policy exist (Theorem 1) **to design an efficient reinforcement learning algorithm**.
- We seek to learn L thresholds: $\alpha_1^*, \alpha_2^*, \ldots, \alpha_L^*$
- We learn these thresholds iteratively through Robbins and Monro's stochastic approximation algorithm.[21]

[21] Herbert Robbins and Sutton Monro. "A Stochastic Approximation Method". In: *The Annals of Mathematical Statistics* 22.3 (1951), pp. 400 –407. DOI: 10.1214/aoms/1177729586. URL: https://doi.org/10.1214/aoms/1177729586.

Our Reinforcement Learning Algorithm for Learning Threshold Policies

- We **use the structural result** that an optimal threshold policy exist (Theorem 1) **to design an efficient reinforcement learning algorithm**.

- We seek to learn L thresholds: $\alpha_1^*, \alpha_2^*, \ldots, \alpha_L^*$
- We learn these thresholds iteratively through Robbins and Monro's stochastic approximation algorithm.[22]



[22] Herbert Robbins and Sutton Monro. "A Stochastic Approximation Method". In: *The Annals of Mathematical Statistics* 22.3 (1951), pp. 400 –407. DOI: 10.1214/aoms/1177729586. URL: https://doi.org/10.1214/aoms/1177729586.

Our Reinforcement Learning Algorithm for Learning Threshold Policies

1. Parameterize the policy $\pi_{I,\theta^{(1)}}$ by $\theta \in \mathbb{R}^L$
2. The *policy gradient*

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_{I,\theta}}\left[\sum_{t=1}^{\infty} \nabla_\theta \log \pi_{I,\theta}(a_t|s_t) \sum_{\tau=t}^{\infty} r_t\right]$$

   exists as long as $\pi_{I,\theta}$ is differentiable.
3. A pure threshold policy is not differentiable.
4. To ensure differentiability and to constrain the thresholds to be in $[0, 1]$, we define $\pi_{\theta,I}$ to be a smooth stochastic policy that approximates a threshold policy:

$$\pi_{i,\theta}(S|b(1)) = \left(1 + \left(\frac{b(1)(1-\sigma(\theta_l))}{\sigma(\theta_l)(1-b(1))}\right)^{-20}\right)^{-1}$$

   where $\sigma(\cdot)$ is the sigmoid function and $\sigma(\theta_l)$ is the threshold.

Our Reinforcement Learning Algorithm for Learning Threshold Policies

1. Parameterize the policy $\pi_{I,\theta(1)}$ by $\theta \in \mathbb{R}^L$
2. The *policy gradient*

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_{I,\theta}}\left[\sum_{t=1}^{\infty} \nabla_\theta \log \pi_{I,\theta}(a_t|s_t) \sum_{\tau=t}^{\infty} r_t\right]$$

   exists as long as $\pi_{I,\theta}$ is differentiable.

3. A pure threshold policy is not differentiable.
4. To ensure differentiability and to constrain the thresholds to be in $[0, 1]$, we define $\pi_{\theta,I}$ to be a smooth stochastic policy that approximates a threshold policy:

$$\pi_{i,\theta}(S|b(1)) = \left(1 + \left(\frac{b(1)(1-\sigma(\theta_I))}{\sigma(\theta_I)(1-b(1))}\right)^{-20}\right)^{-1}$$

   where $\sigma(\cdot)$ is the sigmoid function and $\sigma(\theta_I)$ is the threshold.

Our Reinforcement Learning Algorithm for Learning Threshold Policies

1. Parameterize the policy $\pi_{I,\theta^{(1)}}$ by $\theta \in \mathbb{R}^L$
2. The *policy gradient*

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_{I,\theta}}\left[\sum_{t=1}^{\infty} \nabla_\theta \log \pi_{I,\theta}(a_t|s_t) \sum_{\tau=t}^{\infty} r_t\right]$$

   exists as long as $\pi_{I,\theta}$ is differentiable.

3. A pure threshold policy is not differentiable.
4. To ensure differentiability and to constrain the thresholds to be in $[0,1]$, we define $\pi_{\theta,I}$ to be a smooth stochastic policy that approximates a threshold policy:

$$\pi_{i,\theta}(S|b(1)) = \left(1 + \left(\frac{b(1)(1-\sigma(\theta_l))}{\sigma(\theta_l)(1-b(1))}\right)^{-20}\right)^{-1}$$

   where $\sigma(\cdot)$ is the sigmoid function and $\sigma(\theta_l)$ is the threshold.

Our Reinforcement Learning Algorithm for Learning Threshold Policies

1. Parameterize the policy $\pi_{I,\theta^{(1)}}$ by $\theta \in \mathbb{R}^L$
2. The *policy gradient*

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_{I,\theta}}\left[\sum_{t=1}^{\infty} \nabla_\theta \log \pi_{I,\theta}(a_t|s_t) \sum_{\tau=t}^{\infty} r_t\right]$$

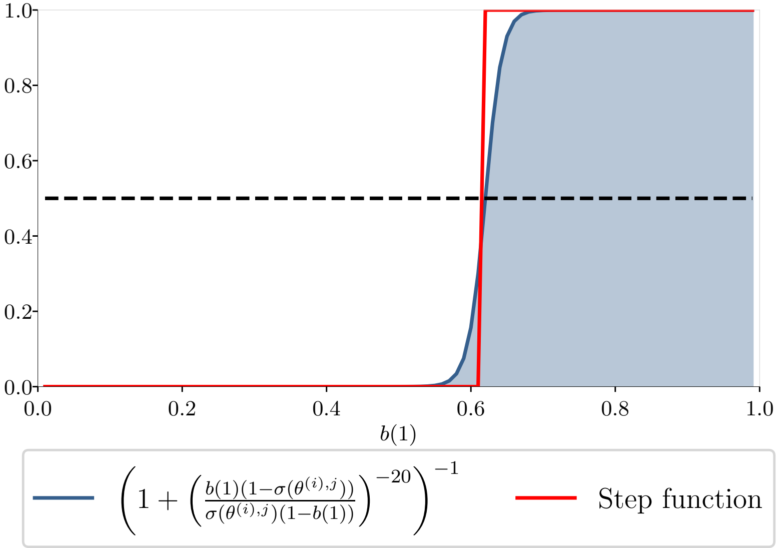
   exists as long as $\pi_{I,\theta}$ is differentiable.

3. A pure threshold policy is not differentiable.
4. To ensure differentiability and to constrain the thresholds to be in $[0,1]$, we define $\pi_{\theta,I}$ to be a smooth stochastic policy that approximates a threshold policy:

$$\pi_{i,\theta}(S|b(1)) = \left(1 + \left(\frac{b(1)(1-\sigma(\theta_I))}{\sigma(\theta_I)(1-b(1))}\right)^{-20}\right)^{-1}$$

   where $\sigma(\cdot)$ is the sigmoid function and $\sigma(\theta_I)$ is the threshold.

Smooth Threshold Policy



Our Reinforcement Learning Algorithm for Learning Threshold Policies

1. We learn the thresholds through simulation.
2. For each iteration $n \in \{1, 2, \ldots\}$, we perturb θ_n to obtain $\theta_n + c_n\Delta_n$ and $\theta_n - c_n\Delta_n$.
3. Then, we simulate two POMDP episodes
4. We then use the obtained episode outcomes $\hat{J}(\theta_n + c_n\Delta_n)$ and $\hat{J}(\theta_n - c_n\Delta_n)$ to estimate $\nabla_\theta J(\theta)$ using the Simultaneous Perturbation Stochastic Approximation (SPSA) gradient estimator[23]:

$$\left(\hat{\nabla}_{\theta_n} J(\theta_n)\right)_k = \frac{\hat{J}(\theta_n + c_n\Delta_n) - \hat{J}(\theta_n - c_n\Delta_n)}{2c_n(\Delta_n)_k}$$

5. Next, we use the estimated gradient and update the vector of thresholds through the stochastic approximation update:

$$\theta_{n+1} = \theta_n + a_n \hat{\nabla}_{\theta_n} J(\theta_n)$$

[23] James C. Spall. "Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation". In: *IEEE TRANSACTIONS ON AUTOMATIC CONTROL* 37.3 (1992), pp. 332–341.

Our Reinforcement Learning Algorithm for Learning Threshold Policies

1. We learn the thresholds through simulation.
2. For each iteration $n \in \{1, 2, \dots\}$, we perturb θ_n to obtain $\theta_n + c_n\Delta_n$ and $\theta_n - c_n\Delta_n$.
3. Then, we simulate two POMDP episodes
4. We then use the obtained episode outcomes $\hat{J}(\theta_n + c_n\Delta_n)$ and $\hat{J}(\theta_n - c_n\Delta_n)$ to estimate $\nabla_\theta J(\theta)$ using the Simultaneous Perturbation Stochastic Approximation (SPSA) gradient estimator[23]:

$$\left(\hat{\nabla}_{\theta_n} J(\theta_n)\right)_k = \frac{\hat{J}(\theta_n + c_n\Delta_n) - \hat{J}(\theta_n - c_n\Delta_n)}{2c_n(\Delta_n)_k}$$

5. Next, we use the estimated gradient and update the vector of thresholds through the stochastic approximation update:

$$\theta_{n+1} = \theta_n + a_n \hat{\nabla}_{\theta_n} J(\theta_n)$$

[23] James C. Spall. "Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation". In: *IEEE TRANSACTIONS ON AUTOMATIC CONTROL* 37.3 (1992), pp. 332–341.

Our Reinforcement Learning Algorithm for Learning Threshold Policies

1. We learn the thresholds through simulation.
2. For each iteration $n \in \{1, 2, \dots\}$, we perturb θ_n to obtain $\theta_n + c_n\Delta_n$ and $\theta_n - c_n\Delta_n$.
3. Then, we simulate two POMDP episodes
4. We then use the obtained episode outcomes $\hat{J}(\theta_n + c_n\Delta_n)$ and $\hat{J}(\theta_n - c_n\Delta_n)$ to estimate $\nabla_\theta J(\theta)$ using the Simultaneous Perturbation Stochastic Approximation (SPSA) gradient estimator[23]:

$$\left(\hat{\nabla}_{\theta_n} J(\theta_n)\right)_k = \frac{\hat{J}(\theta_n + c_n\Delta_n) - \hat{J}(\theta_n - c_n\Delta_n)}{2c_n(\Delta_n)_k}$$

5. Next, we use the estimated gradient and update the vector of thresholds through the stochastic approximation update:

$$\theta_{n+1} = \theta_n + a_n \hat{\nabla}_{\theta_n} J(\theta_n)$$

[23] James C. Spall. "Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation". In: *IEEE TRANSACTIONS ON AUTOMATIC CONTROL* 37.3 (1992), pp. 332–341.

Our Reinforcement Learning Algorithm for Learning Threshold Policies

1. We learn the thresholds through simulation.
2. For each iteration $n \in \{1, 2, \ldots\}$, we perturb θ_n to obtain $\theta_n + c_n\Delta_n$ and $\theta_n - c_n\Delta_n$.
3. Then, we simulate two POMDP episodes
4. We then use the obtained episode outcomes $\hat{J}(\theta_n + c_n\Delta_n)$ and $\hat{J}(\theta_n - c_n\Delta_n)$ to estimate $\nabla_\theta J(\theta)$ using the Simultaneous Perturbation Stochastic Approximation (SPSA) gradient estimator[23]:

$$\left(\hat{\nabla}_{\theta_n} J(\theta_n)\right)_k = \frac{\hat{J}(\theta_n + c_n\Delta_n) - \hat{J}(\theta_n - c_n\Delta_n)}{2c_n(\Delta_n)_k}$$

5. Next, we use the estimated gradient and update the vector of thresholds through the stochastic approximation update:

$$\theta_{n+1} = \theta_n + a_n \hat{\nabla}_{\theta_n} J(\theta_n)$$

[23] James C. Spall. "Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation". In: *IEEE TRANSACTIONS ON AUTOMATIC CONTROL* 37.3 (1992), pp. 332–341.

Our Reinforcement Learning Algorithm for Learning Threshold Policies

1. We learn the thresholds through simulation.
2. For each iteration $n \in \{1, 2, \dots\}$, we perturb θ_n to obtain $\theta_n + c_n\Delta_n$ and $\theta_n - c_n\Delta_n$.
3. Then, we simulate two POMDP episodes
4. We then use the obtained episode outcomes $\hat{J}(\theta_n + c_n\Delta_n)$ and $\hat{J}(\theta_n - c_n\Delta_n)$ to estimate $\nabla_\theta J(\theta)$ using the Simultaneous Perturbation Stochastic Approximation (SPSA) gradient estimator[23]:

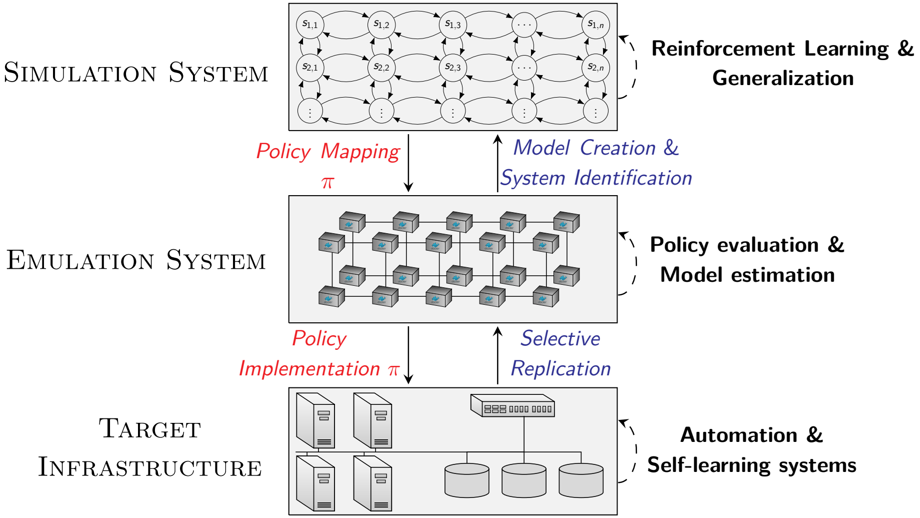
$$\left(\hat{\nabla}_{\theta_n} J(\theta_n)\right)_k = \frac{\hat{J}(\theta_n + c_n\Delta_n) - \hat{J}(\theta_n - c_n\Delta_n)}{2c_n(\Delta_n)_k}$$

5. Next, we use the estimated gradient and update the vector of thresholds through the stochastic approximation update:

$$\theta_{n+1} = \theta_n + a_n \hat{\nabla}_{\theta_n} J(\theta_n)$$

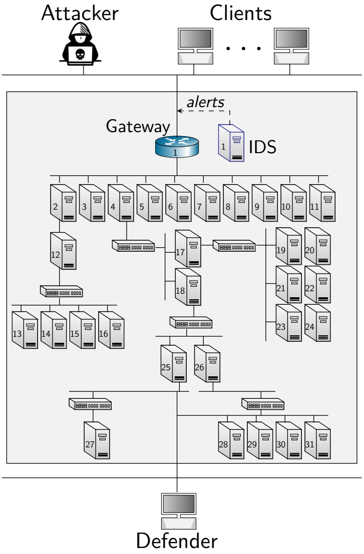
[23] James C. Spall. "Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation". In: *IEEE TRANSACTIONS ON AUTOMATIC CONTROL* 37.3 (1992), pp. 332–341.

To evaluate Policies Learned in Simulation we Run them in the Emulation



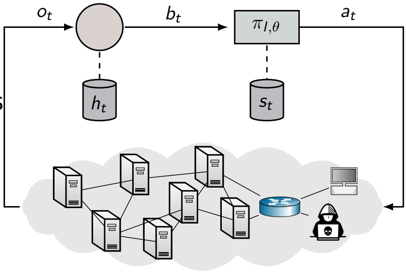
Emulating the Target Infrastructure

- Emulate **hosts** with docker containers
- Emulate **IDS and vulnerabilities** with software
- Network isolation and **traffic shaping** through NetEm in the Linux kernel
- Enforce **resource constraints** using cgroups.
- Emulate **client arrivals** with Poisson process
- **Internal connections** are full-duplex & loss-less with bit capacities of 1000 Mbit/s
- **External connections** are full-duplex with bit capacities of 100 Mbit/s & 0.1% packet loss in normal operation and random bursts of 1% packet loss



Running a POMDP Episode in the Emulation

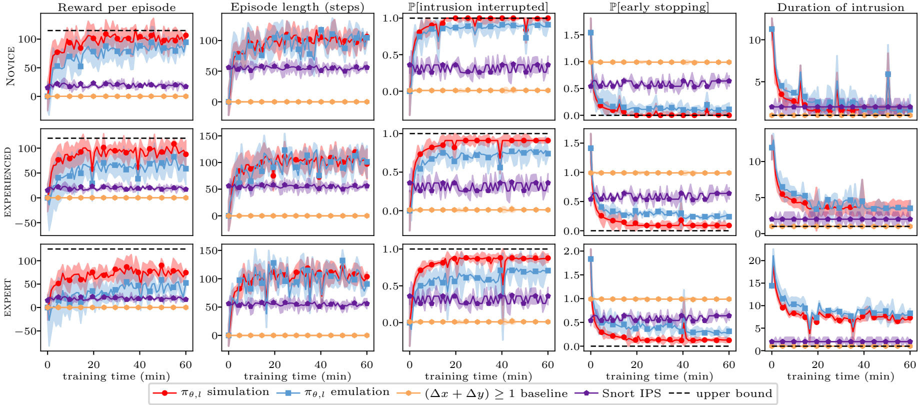
- A distributed system with synchronized clocks
- We run software sensors on all emulated hosts
- Sensors produce messages to a distributed queue (Kafka)
- A stream processor (Spark) consumes messages from the queue and computes statistics
- Actions are selected based on the computed statistics and the policies
- Actions are sent to the emulation using gRPC
- Actions are executed by running commands on the hosts



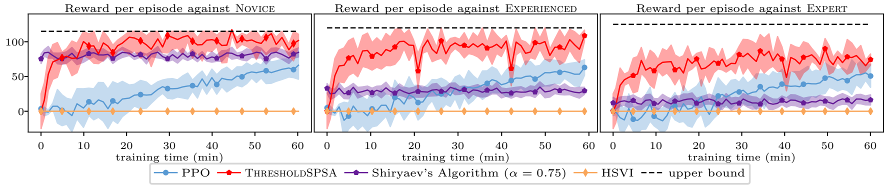
Outline

- **Use Case & Approach:**
  - Intrusion prevention
  - System identification
  - Reinforcement learning and optimal stopping
- **Formal Model of The Use Case**
  - Intrusion prevention as an optimal stopping problem
  - Partially observed Markov decision process
- **Structure of π^***
  - Existence of optimal multi-threshold policy π^*_l
  - Stopping sets $\mathscr{S}_l$ are connected and nested
- **Reinforcement learning method**
  - Learning threshold policies & the policy gradient
  - Emulated infrastructure
- **Results & Conclusion**
  - Numerical evaluation results & Demo
  - Conclusion & future work

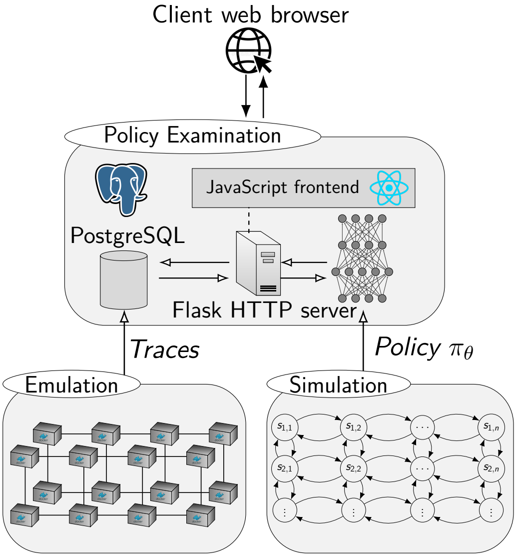
Evaluation Results



Evaluation Results



Demo - A System for Interactive Examination of Learned Security Policies



Architecture of the system for examining learned security policies.

Conclusions & Future Work

- **Conclusions:**
  - We develop a *method* to automatically learn security policies
    - (1) emulation system; (2) system identification; (3) simulation system; (4) reinforcement learning and (5) domain randomization and generalization.
  - We apply the method to an **intrusion prevention use case**
  - We formulate intrusion prevention as a multiple stopping problem
    - We present a POMDP model of the use case
    - We apply the stopping theory to establish structural results of the optimal policy
    - We design a reinforcement learning algorithm that outperforms state-of-the-art on our use case
    - We show numerical results in realistic emulation environment
- **Our research plans:**
  - Extending the model
    - Active attacker: Partially Observed Stochastic Game, Equilibrium analysis
    - Less restrictions on defender