Intrusion Prevention through Optimal Stopping and Self-Play

NSE Seminar

Kim Hammar & Rolf Stadler

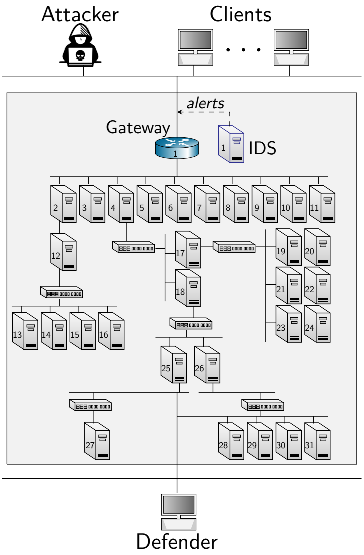
kimham@kth.se & *stadler@kth.se*

Division of Network and Systems Engineering
KTH Royal Institute of Technology

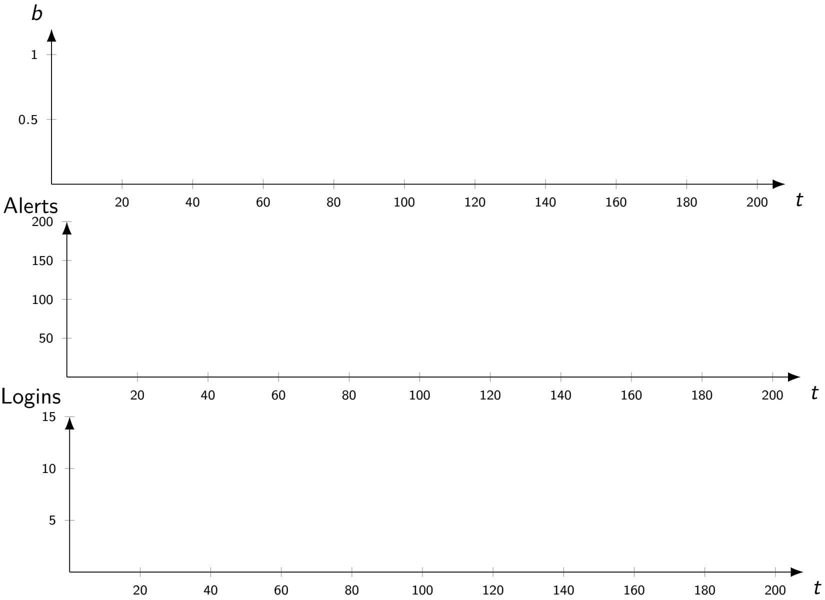
Mar 18, 2022

Use Case: Intrusion Prevention

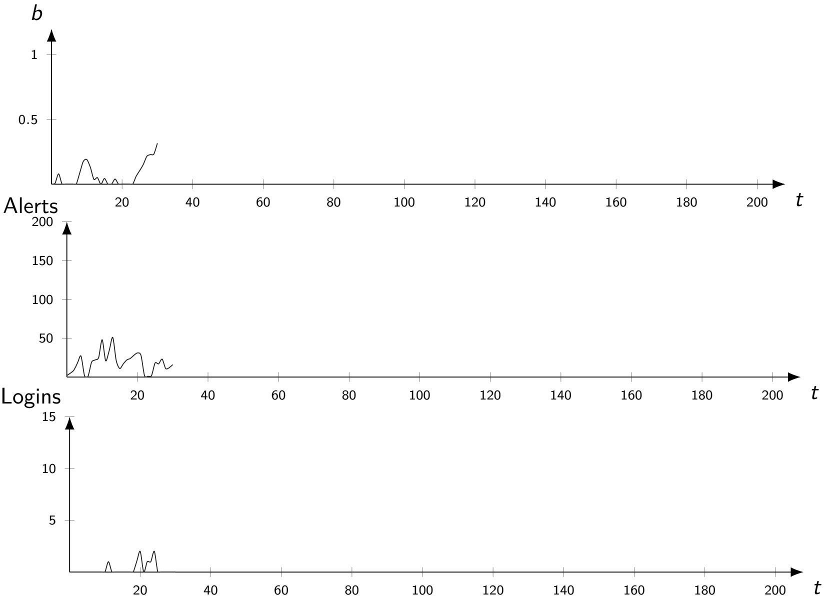
- A **Defender** owns an infrastructure
  - Consists of connected components
  - Components run network services
  - Defender defends the infrastructure by monitoring and active defense
  - Has partial observability
- An **Attacker** seeks to intrude on the infrastructure
  - Has a partial view of the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



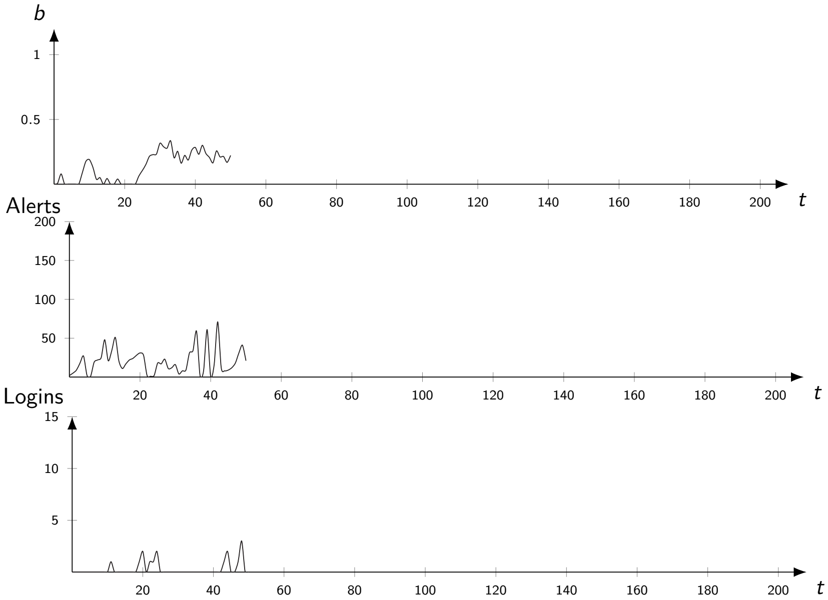
The Intrusion Prevention Problem



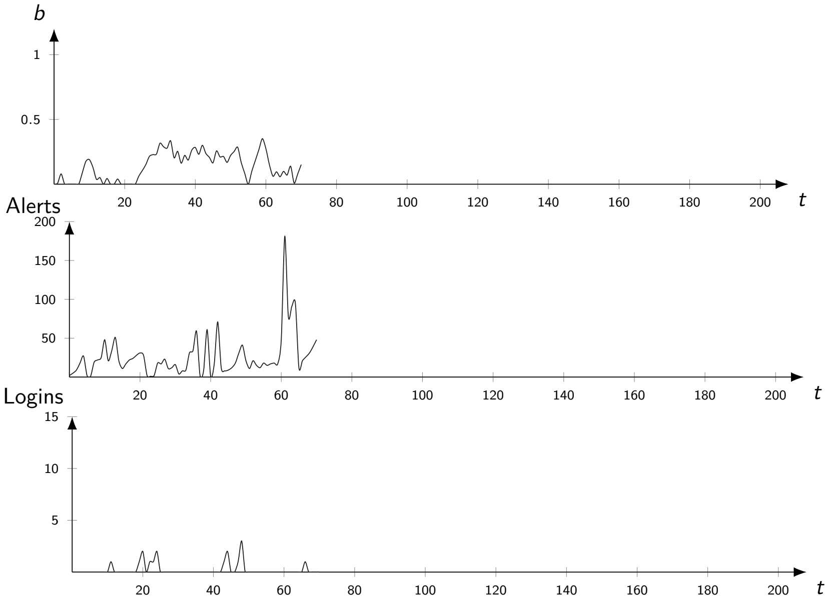
The Intrusion Prevention Problem



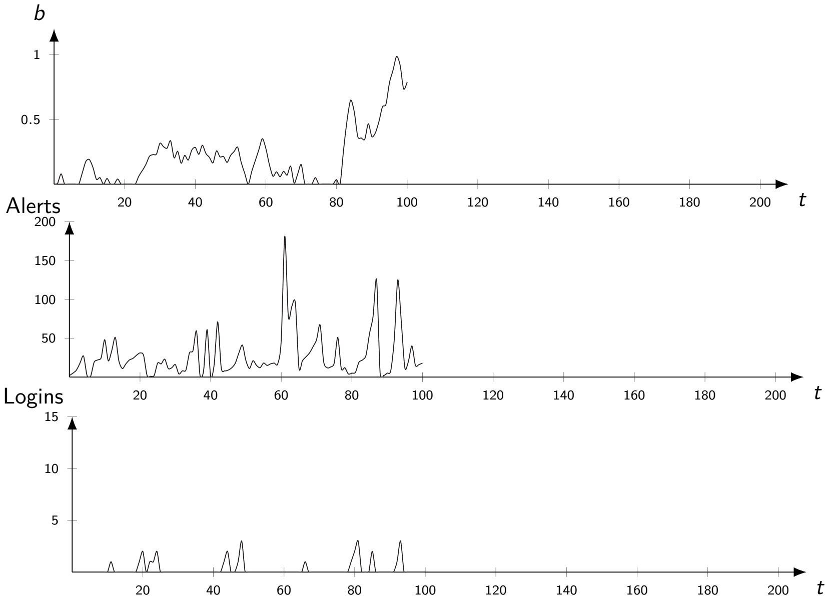
The Intrusion Prevention Problem



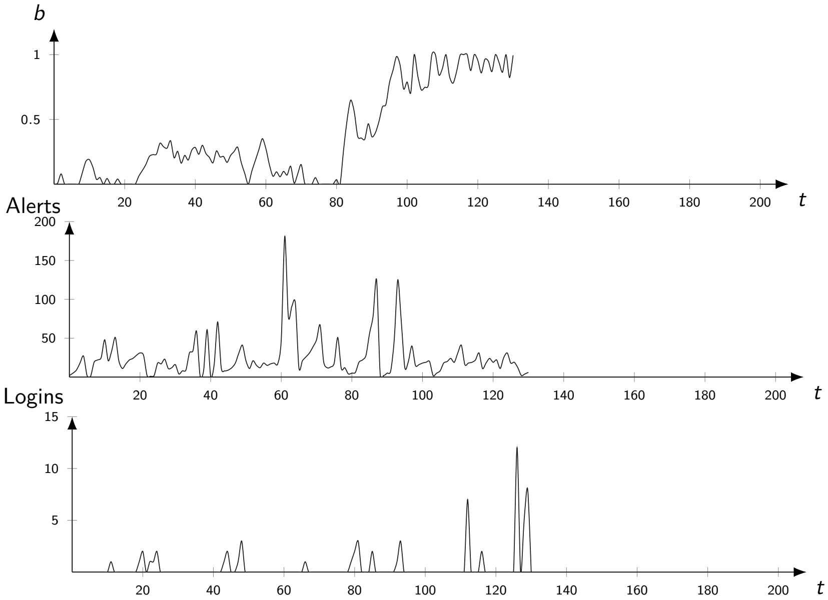
The Intrusion Prevention Problem



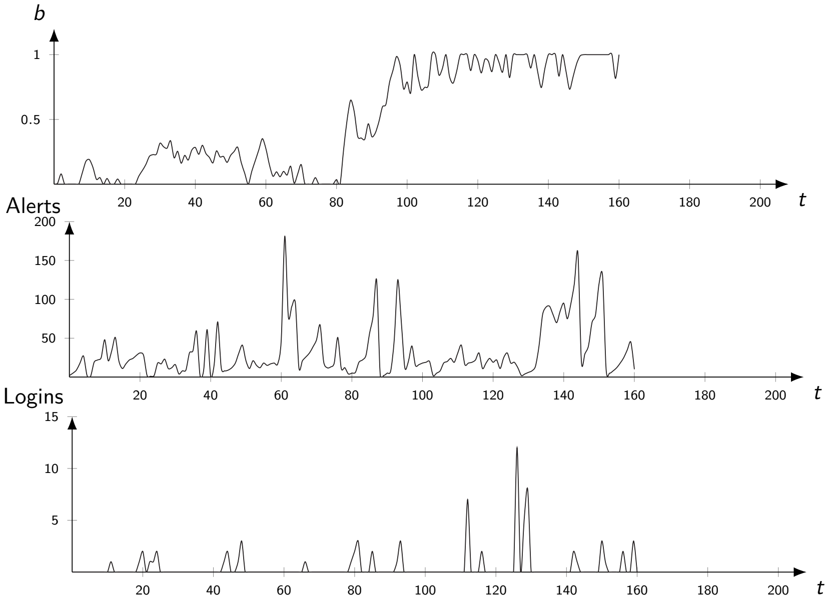
The Intrusion Prevention Problem



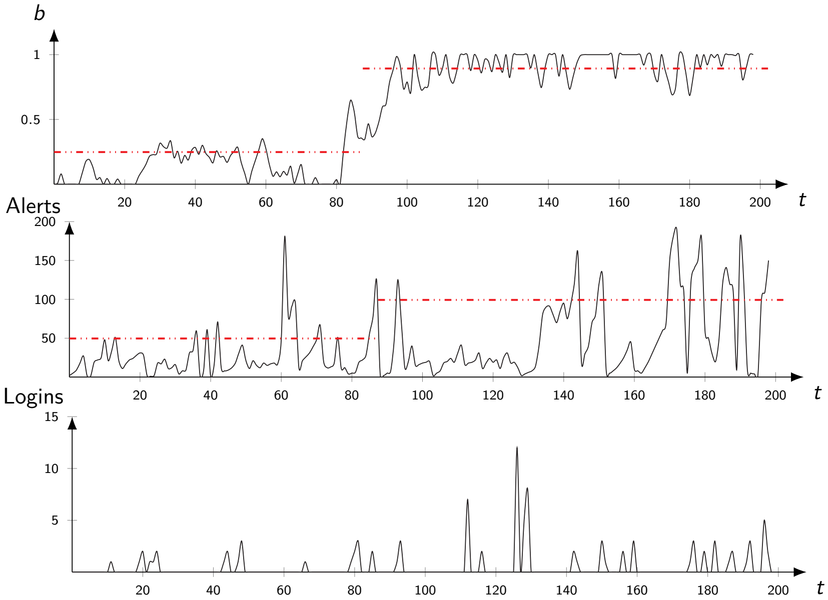
The Intrusion Prevention Problem



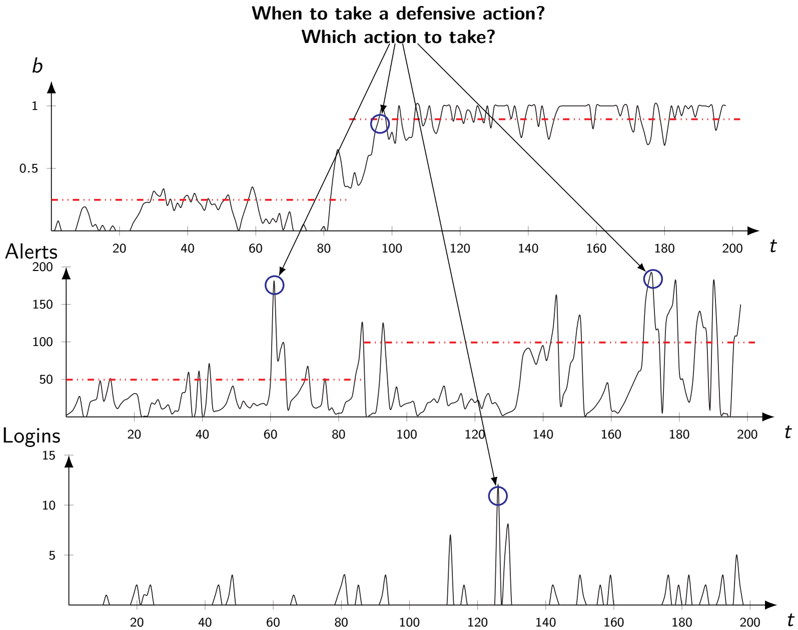
The Intrusion Prevention Problem



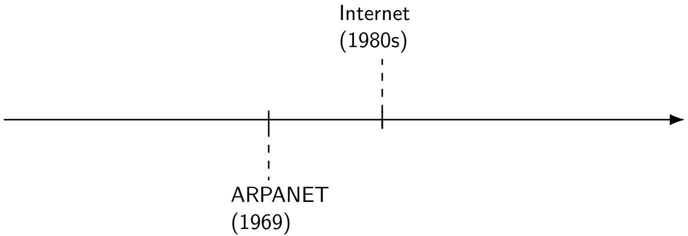
The Intrusion Prevention Problem



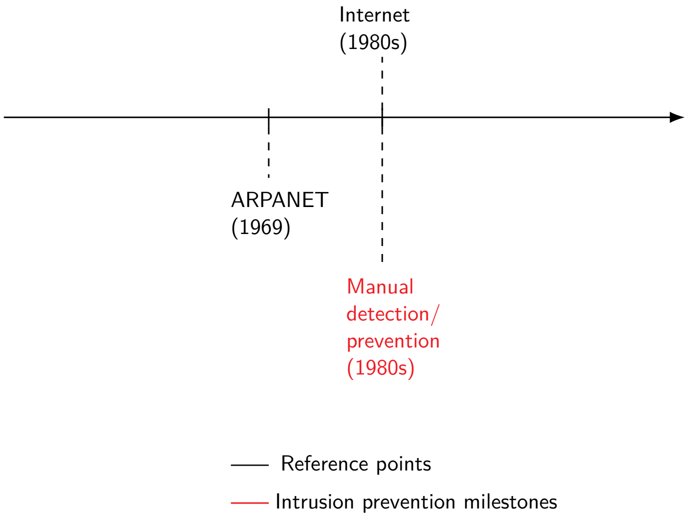
The Intrusion Prevention Problem



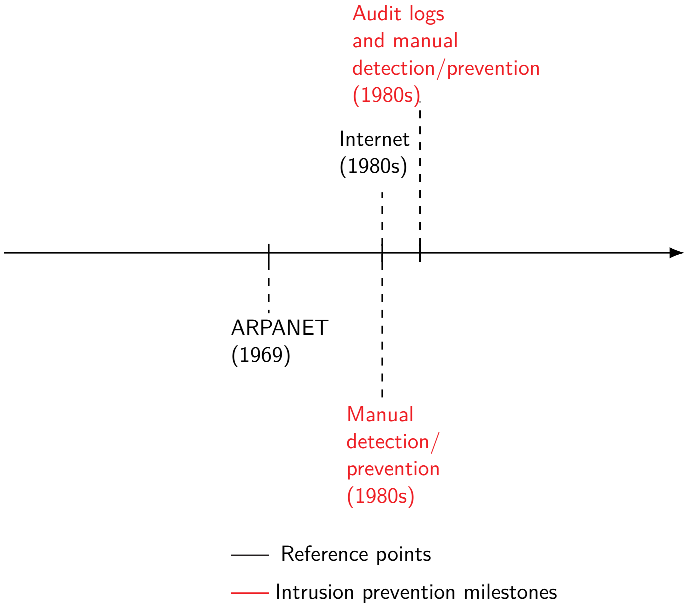
A Brief History of Intrusion Prevention



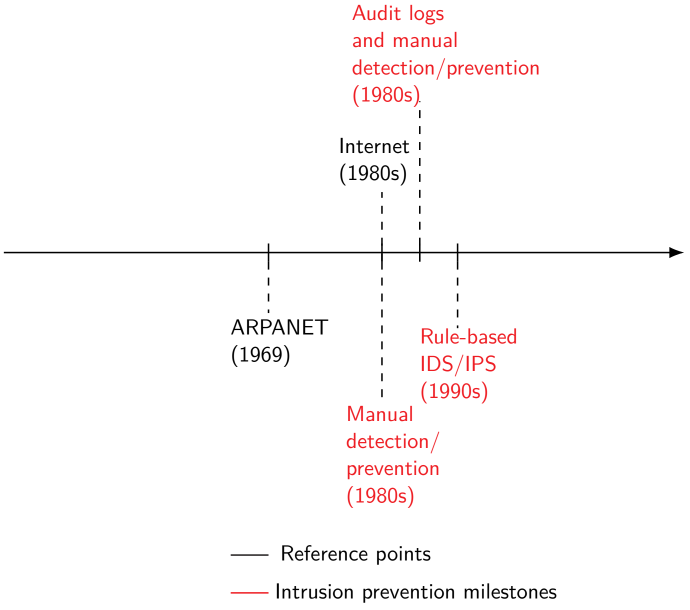
A Brief History of Intrusion Prevention



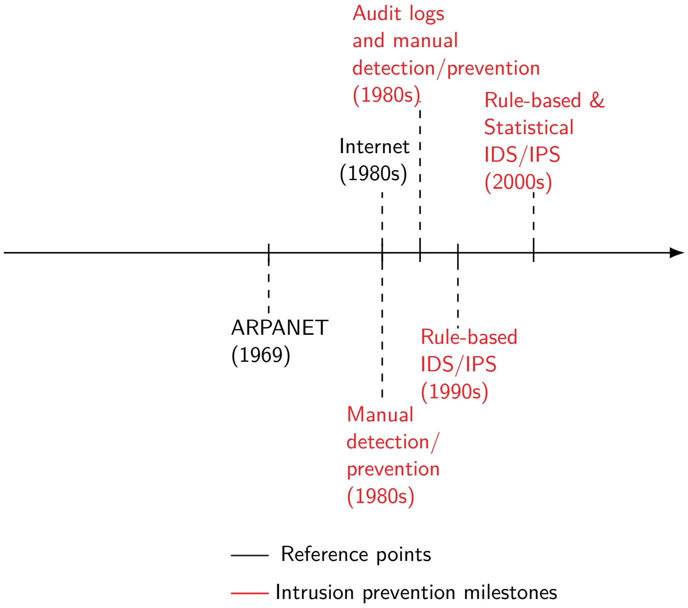
A Brief History of Intrusion Prevention



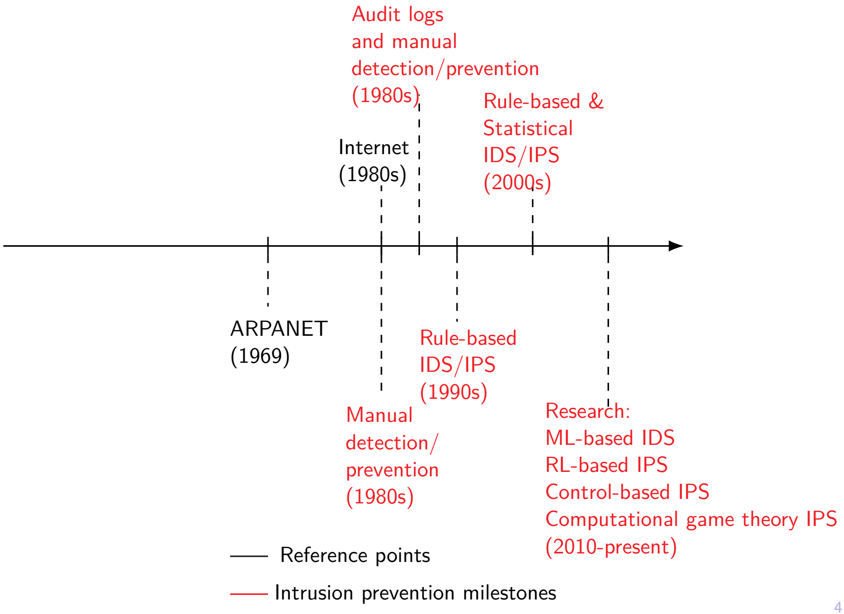
A Brief History of Intrusion Prevention



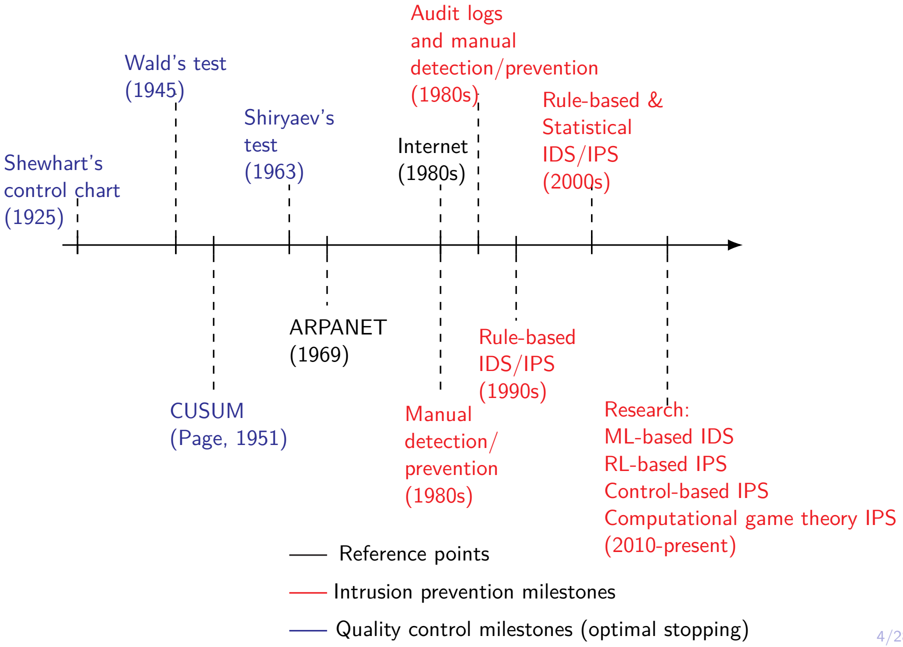
A Brief History of Intrusion Prevention



A Brief History of Intrusion Prevention

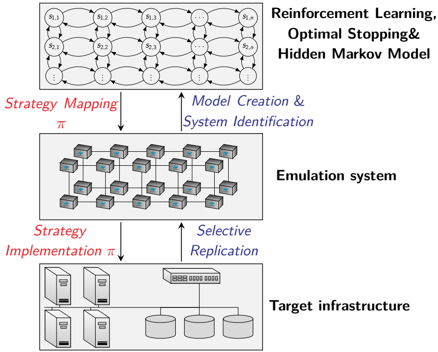


A Brief History of Intrusion Prevention



Our Approach

- **Formal model**:
  - Controlled Hidden Markov Model
  - Defender has partial observability
  - A game if attacker is active
- **Data collection**:
  - Emulated infrastructure
- **Finding defender strategies**:
  - Self-play reinforcement learning
  - Optimal stopping



Outline

- **Use Case & Approach:**
  - Intrusion prevention
  - Reinforcement learning and optimal stopping
- **Formal Model of The Use Case**
  - Intrusion prevention as an optimal stopping problem
  - Partially observed stochastic game
- **Game Analysis and Structure of $(\tilde{\pi}_1, \tilde{\pi}_2)$**
  - Structural result: multi-threshold best responses
  - Stopping sets $\mathscr{S}_l^{(1)}$ are connected and nested
- **Our Method**
  - Emulation of the target infrastructure,
  - System identification
  - Our reinforcement learning algorithm
- **Results & Conclusion**
  - Numerical evaluation results & Demo
  - Conclusion & future work

Outline

- **Use Case & Approach:**
  - Intrusion prevention
  - Reinforcement learning and optimal stopping
- **Formal Model of The Use Case**
  - Intrusion prevention as an optimal stopping problem
  - Partially observed stochastic game
- **Game Analysis and Structure of $(\tilde{\pi}_1, \tilde{\pi}_2)$**
  - Structural result: multi-threshold best responses
  - Stopping sets $\mathscr{S}_l^{(1)}$ are connected and nested
- **Our Method**
  - Emulation of the target infrastructure
  - System identification
  - Our reinforcement learning algorithm
- **Results & Conclusion**
  - Numerical evaluation results & Demo
  - Conclusion & future work

Outline

- **Use Case & Approach:**
  - Intrusion prevention
  - Reinforcement learning and optimal stopping
- **Formal Model of The Use Case**
  - Intrusion prevention as an optimal stopping problem
  - Partially observed stochastic game
- **Game Analysis and Structure of $(\tilde{\pi}_1, \tilde{\pi}_2)$**
  - Structural result: multi-threshold best responses
  - Stopping sets $\mathscr{S}_l^{(1)}$ are connected and nested
- **Our Method**
  - Emulation of the target infrastructure
  - System identification
  - Our reinforcement learning algorithm
- **Results & Conclusion**
  - Numerical evaluation results & Demo
  - Conclusion & future work

Outline

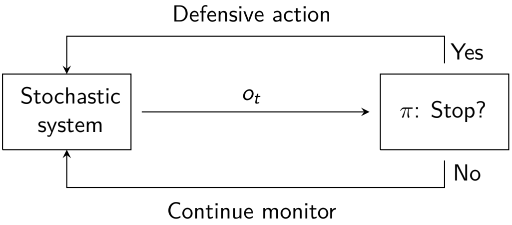
- **Use Case & Approach:**
  - Intrusion prevention
  - Reinforcement learning and optimal stopping
- **Formal Model of The Use Case**
  - Intrusion prevention as an optimal stopping problem
  - Partially observed stochastic game
- **Game Analysis and Structure of $(\tilde{\pi}_1, \tilde{\pi}_2)$**
  - Structural result: multi-threshold best responses
  - Stopping sets $\mathscr{S}_l^{(1)}$ are connected and nested
- **Our Method**
  - Emulation of the target infrastructure
  - System identification
  - Our reinforcement learning algorithm
- **Results & Conclusion**
  - Numerical evaluation results & Demo
  - Conclusion & future work

Outline

- **Use Case & Approach:**
  - Intrusion prevention
  - Reinforcement learning and optimal stopping
- **Formal Model of The Use Case**
  - Intrusion prevention as an optimal stopping problem
  - Partially observed stochastic game
- **Game Analysis and Structure of $(\tilde{\pi}_1, \tilde{\pi}_2)$**
  - Structural result: multi-threshold best responses
  - Stopping sets $\mathscr{S}_l^{(1)}$ are connected and nested
- **Our Method**
  - Emulation of the target infrastructure
  - System identification
  - Our reinforcement learning algorithm
- **Results & Conclusion**
  - Numerical evaluation results & Demo
  - Conclusion & future work

Intrusion Prevention through Optimal Stopping

- The system evolves in discrete time-steps.
- A stop action = a defensive action. (e.g. reconfigure IPS)



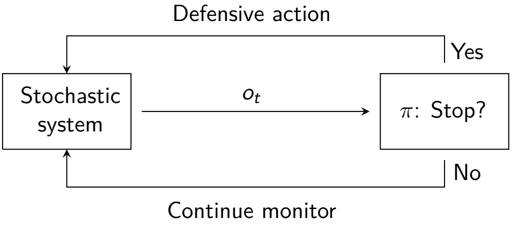
- The $L-l$th **stopping time** τ_l is:

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

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

Intrusion Prevention through Optimal Stopping

- The system evolves in discrete time-steps.
- A stop action = a defensive action. (e.g. reconfigure IPS)



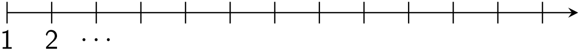
- The $L-l$th **stopping time** τ_l is:

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

- 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 $a_{\tau_l}, o_{\tau_l + 1}, \ldots$

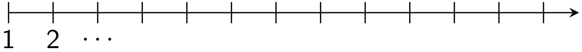
Intrusion Prevention through Optimal Stopping

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).

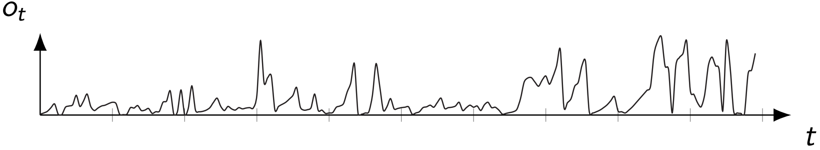


Intrusion Prevention through Optimal Stopping

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).

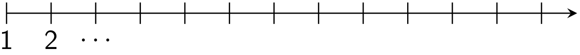


Given the observations:

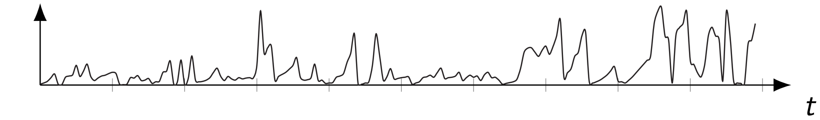


Intrusion Prevention through Optimal Stopping

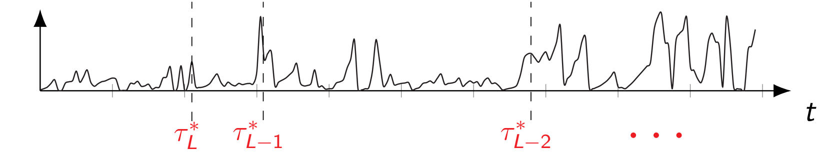
We consider the class of stopping times $\mathcal{T}_t = \{\tau_l \leq t | \tau_l > \tau_{l+1}\} \in \mathcal{F}_k$ ($\mathcal{F}_k$ =natural filtration on h_t).



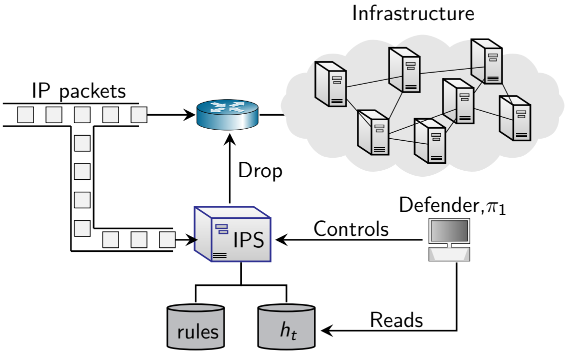
Given the observations:



Find the optimal stopping times $\tau_L^*, \tau_{L-1}^*, \ldots$:



The Defender's Stop Actions



1. Ingress traffic goes through deep packet inspection at gateway
2. Gateway runs the Snort IDS/IPS and may drop packets
3. The defender controls the IPS configuration
4. At each stopping time, we update the IPS configuration

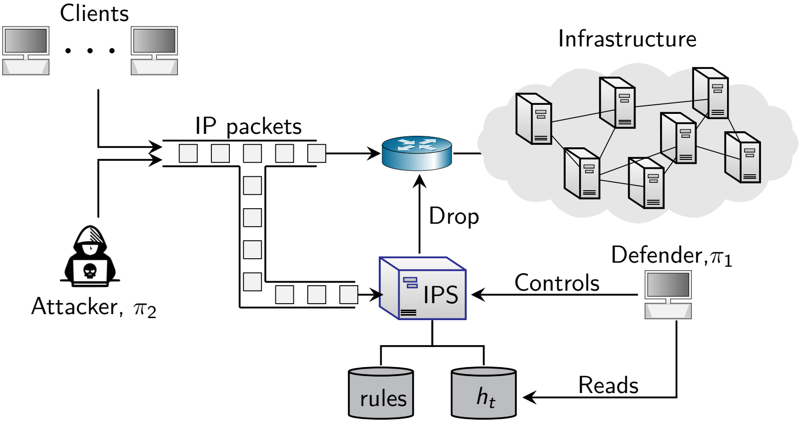
The Defender's Stop Actions

l	*Alarm class c*	*Action*
34	Attempted administrator privilege gain	DROP
33	Attempted user privilege gain	DROP
32	Inappropriate content was detected	DROP
31	Potential corporate privacy violation	DROP
30	Executable code was detected	DROP
29	Successful administrator privilege gain	DROP
28	Successful user privilege gain	DROP
27	A network trojan was detected	DROP
26	Unsuccessful user privilege gain	DROP
25	Web application attack	DROP
24	Attempted denial of service	DROP
23	Attempted information leak	DROP
22	Potentially Bad Traffic	DROP
21	Attempt to login by a default username and password	DROP
20	Detection of a denial of service attack	DROP
$\vdots$	$\vdots$	$\vdots$

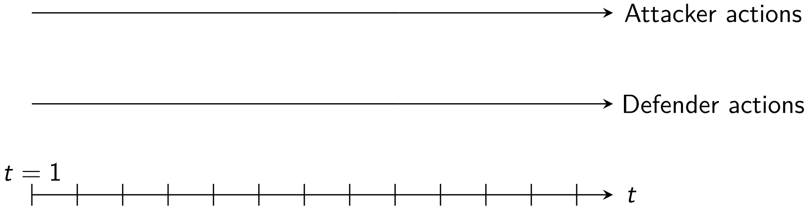
Table 1: Defender stop actions in the emulation; l denotes the number of stops remaining.

Optimal Stopping Game

- **We assume a strategic attacker**
  - Attacker knows which actions generate alarms
  - Attacker tries to be stealthy
  - Attacker may try to achieve denial of service

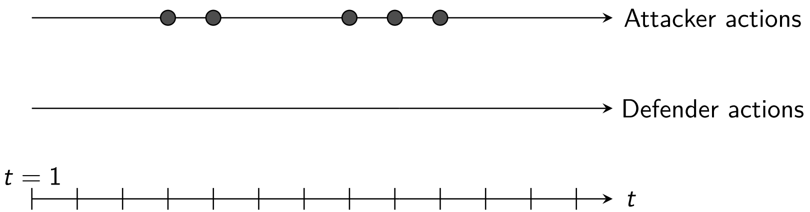


Optimal Stopping Game



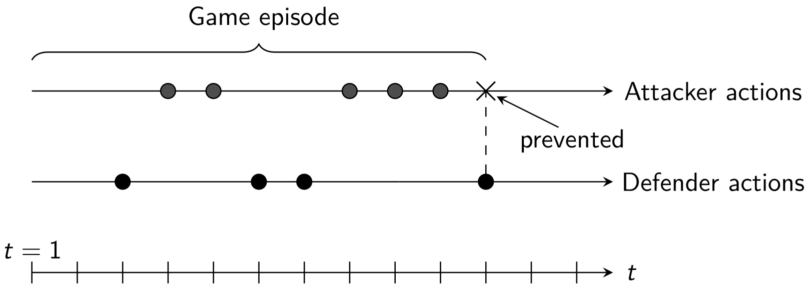
- The attacker's stopping times $\tau_{2,1}, \tau_{2,2}, \ldots$ determine the times to start/stop the intrusion
  - During the inrusion, the attacker follows a fixed intrusion strategy
- The defender's stopping times $\tau_{1,L}, \tau_{1,L-1}, \ldots$ determine the times to update the IPS configuration

Optimal Stopping Game



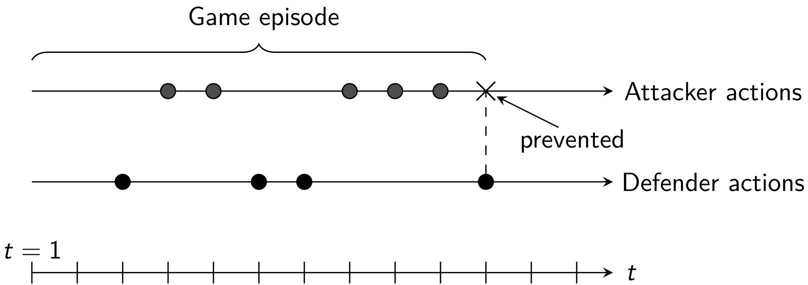
- The attacker's stopping times $\tau_{2,1}, \tau_{2,2}, \ldots$ determine the times to start/stop the intrusion
  - During the intrusion, the attacker follows a fixed intrusion strategy
- The defender's stopping times $\tau_{1,L}, \tau_{1,L-1}, \ldots$ determine the times to update the IPS configuration

Optimal Stopping Game



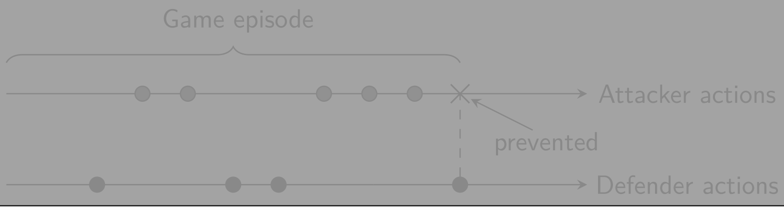
- The attacker's stopping times $\tau_{2,1}, \tau_{2,2}, \ldots$ determine the times to start/stop the intrusion
  - During the intrusion, the attacker follows a fixed intrusion strategy
- The defender's stopping times $\tau_{1,L}, \tau_{1,L-1}, \ldots$ determine the times to update the IPS configuration

Optimal Stopping Game



- The attacker's stopping times $\tau_{2,1}, \tau_{2,2}, \ldots$ determine the times to start/stop the intrusion
  - During the intrusion, the attacker follows a fixed intrusion strategy
- The defender's stopping times $\tau_{1,1}, \tau_{1,2}, \ldots$ determine the times to update the IPS configuration
- We seek a **Nash equilibrium** (π_1^*, π_2^*), from which we can extract the optimal defender strategy π_1^* against a worst-case attacker.

Optimal Stopping Game

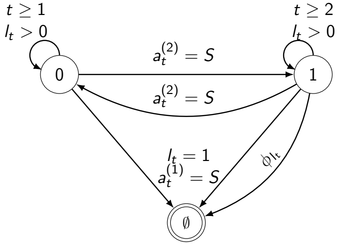


We model this game as a **zero-sum** partially observed stochastic game

- The attacker's stopping times $\tau_{2,1}, \tau_{2,2}, \ldots$ determine the times to start/stop the intrusion
  - During the intrusion, the attacker follows a fixed intrusion strategy
- The defender's stopping times $\tau_{1,1}, \tau_{1,2}, \ldots$ determine the times to update the IPS configuration
- We seek a **Nash equilibrium** (π_1^*, π_2^*), from which we can extract the optimal defender strategy π_1^* against the

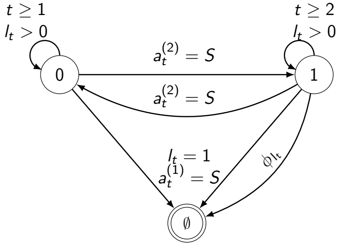
Partially Observed Stochastic Game

- **Players:** $\mathcal{N} = \{1, 2\}$ (Defender=1)
- **States:** Intrusion $s_t \in \{0, 1\}$, terminal $\emptyset$.
- **Observations:**
  - IDS Alerts $\Delta x_{1,t}, \Delta x_{2,t}, \ldots, \Delta x_{M,t}$, defender stops remaining $l_t \in \{1, .., L\}$, $f_X(\Delta x_1, \ldots, \Delta x_M | s_t)$
- **Actions:**
  - $\mathcal{A}_1 = \mathcal{A}_2 = \{S, C\}$
- **Rewards:**
  - Defender reward: security and service.
  - Attacker reward: negative of defender reward.
- **Transition probabilities:**
  - Follows from game dynamics.
- **Horizon:**
  - ∞



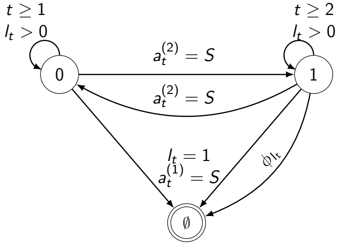
Partially Observed Stochastic Game

- **Players:** $\mathcal{N} = \{1, 2\}$ (Defender=1)
- **States:** Intrusion $s_t \in \{0, 1\}$, terminal $\emptyset$.
- **Observations:**
  - IDS Alerts $\Delta x_{1,t}, \Delta x_{2,t}, \ldots, \Delta x_{M,t}$, defender stops remaining $l_t \in \{1, .., L\}$, $f_X(\Delta x_1, \ldots, \Delta x_M | s_t)$
- **Actions:**
  - $\mathcal{A}_1 = \mathcal{A}_2 = \{S, C\}$
- **Rewards:**
  - Defender reward: security and service.
  - Attacker reward: negative of defender reward.
- **Transition probabilities:**
  - Follows from game dynamics.
- **Horizon:**
  - ∞



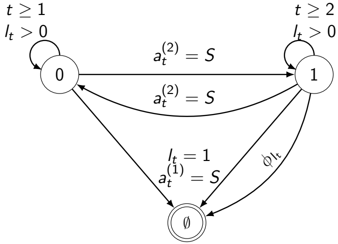
Partially Observed Stochastic Game

- **Players:** $\mathcal{N} = \{1, 2\}$ (Defender=1)
- **States:** Intrusion $s_t \in \{0, 1\}$, terminal $\emptyset$.
- **Observations:**
  - IDS Alerts $\Delta x_{1,t}, \Delta x_{2,t}, \ldots, \Delta x_{M,t}$, defender stops remaining $l_t \in \{1, .., L\}$, $f_X(\Delta x_1, \ldots, \Delta x_M | s_t)$
- **Actions:**
  - $\mathcal{A}_1 = \mathcal{A}_2 = \{S, C\}$
- **Rewards:**
  - Defender reward: security and service.
  - Attacker reward: negative of defender reward.
- **Transition probabilities:**
  - Follows from game dynamics.
- **Horizon:**
  - ∞



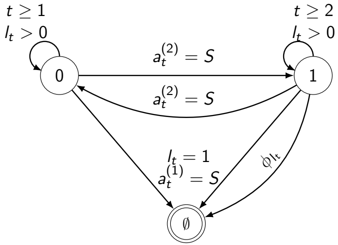
Partially Observed Stochastic Game

- **Players:** $\mathcal{N} = \{1, 2\}$ (Defender=1)
- **States:** Intrusion $s_t \in \{0, 1\}$, terminal $\emptyset$.
- **Observations:**
  - IDS Alerts $\Delta x_{1,t}, \Delta x_{2,t}, \ldots, \Delta x_{M,t}$, defender stops remaining $l_t \in \{1, .., L\}$, $f_X(\Delta x_1, \ldots, \Delta x_M | s_t)$
- **Actions:**
  - $\mathcal{A}_1 = \mathcal{A}_2 = \{S, C\}$
- **Rewards:**
  - Defender reward: security and service.
  - Attacker reward: negative of defender reward.
- **Transition probabilities:**
  - Follows from game dynamics.
- **Horizon:**
  - ∞



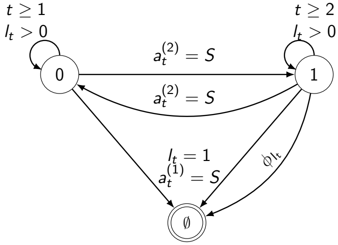
Partially Observed Stochastic Game

- **Players:** $\mathcal{N} = \{1, 2\}$ (Defender=1)
- **States:** Intrusion $s_t \in \{0, 1\}$, terminal $\emptyset$.
- **Observations:**
  - IDS Alerts $\Delta x_{1,t}, \Delta x_{2,t}, \ldots, \Delta x_{M,t}$, defender stops remaining $l_t \in \{1, .., L\}$, $f_X(\Delta x_1, \ldots, \Delta x_M | s_t)$
- **Actions:**
  - $\mathcal{A}_1 = \mathcal{A}_2 = \{S, C\}$
- **Rewards:**
  - Defender reward: security and service.
  - Attacker reward: negative of defender reward.
- **Transition probabilities:**
  - Follows from game dynamics.
- **Horizon:**
  - ∞



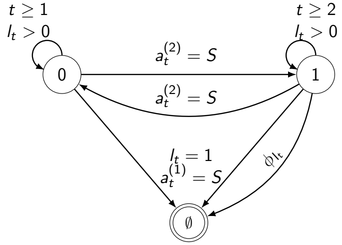
Partially Observed Stochastic Game

- **Players:** $\mathcal{N} = \{1, 2\}$ (Defender=1)
- **States:** Intrusion $s_t \in \{0, 1\}$, terminal $\emptyset$.
- **Observations:**
  - IDS Alerts $\Delta x_{1,t}, \Delta x_{2,t}, \ldots, \Delta x_{M,t}$, defender stops remaining $l_t \in \{1, .., L\}$, $f_X(\Delta x_1, \ldots, \Delta x_M | s_t)$
- **Actions:**
  - $\mathcal{A}_1 = \mathcal{A}_2 = \{S, C\}$
- **Rewards:**
  - Defender reward: security and service.
  - Attacker reward: negative of defender reward.
- **Transition probabilities:**
  - Follows from game dynamics.
- **Horizon:**
  - ∞



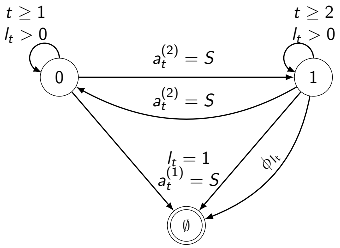
Partially Observed Stochastic Game

- **Players:** $\mathcal{N} = \{1, 2\}$ (Defender=1)
- **States:** Intrusion $s_t \in \{0, 1\}$, terminal $\emptyset$.
- **Observations:**
  - IDS Alerts $\Delta x_{1,t}, \Delta x_{2,t}, \ldots, \Delta x_{M,t}$, defender stops remaining $l_t \in \{1, .., L\}$, $f_X(\Delta x_1, \ldots, \Delta x_M | s_t)$
- **Actions:**
  - $\mathcal{A}_1 = \mathcal{A}_2 = \{S, C\}$
- **Rewards:**
  - Defender reward: security and service.
  - Attacker reward: negative of defender reward.
- **Transition probabilities:**
  - Follows from game dynamics.
- **Horizon:**
  - ∞



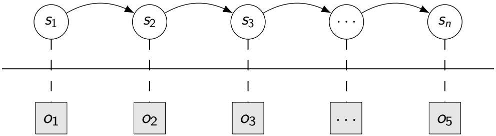
Partially Observed Stochastic Game

- **Players:** $\mathcal{N} = \{1, 2\}$ (Defender=1)
- **States:** Intrusion $s_t \in \{0, 1\}$, terminal $\emptyset$.
- **Observations:**
  - IDS Alerts $\Delta x_{1,t}, \Delta x_{2,t}, \ldots, \Delta x_{M,t}$, defender stops remaining $l_t \in \{1, .., L\}$, $f_X(\Delta x_1, \ldots, \Delta x_M | s_t)$
- **Actions:**
  - $\mathcal{A}_1 = \mathcal{A}_2 = \{S, C\}$
- **Rewards:**
  - Defender reward: security and service.
  - Attacker reward: negative of defender reward.
- **Transition probabilities:**
  - Follows from game dynamics.
- **Horizon:**
  - ∞

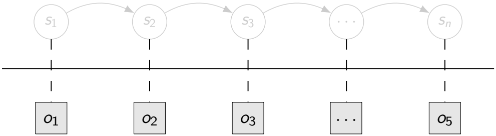


One-Sided Partial Observability

- We assume that the **attacker has perfect information**. Only the **defender has partial information**.
- The attacker's view:



- The defender's view:



One-Sided Partial Observability

- We assume that the **attacker has perfect information**. Only the **defender has partial information**.
- The attacker's view:

- The defender's view:

- Makes it tractable to compute the defender's belief $b^{(1)}_{t,\pi_2}(s_t) = \mathbb{P}[s_t | h_t, \pi_2]$ (avoid nested beliefs)

Game Analysis

- Defender strategy is of the form: $\pi_{1,l} : \mathcal{B} \rightarrow \Delta(\mathcal{A}_1)$
- Attacker strategy is of the form: $\pi_{2,l} : \mathcal{S} \times \mathcal{B} \rightarrow \Delta(\mathcal{A}_2)$

- **Defender and attacker objectives**:

$$J_1(\pi_{1,l}, \pi_{2,l}) = \mathbb{E}_{(\pi_{1,l}, \pi_{2,l})} \left[\sum_{t=1}^{\infty} \gamma^{t-1} \mathcal{R}_l(s_t, \boldsymbol{a}_t) \right]$$

$$J_2(\pi_{1,l}, \pi_{2,l}) = -J_1$$

- **Best response correspondences**:

$$B_1(\pi_{2,l}) = \arg\max_{\pi_{1,l} \in \Pi_1} J_1(\pi_{1,l}, \pi_{2,l})$$

$$B_2(\pi_{1,l}) = \arg\max_{\pi_{2,l} \in \Pi_2} J_2(\pi_{1,l}, \pi_{2,l})$$

- **Nash equilibrium** $(\pi^*_{1,l}, \pi^*_{2,l})$:

$$\pi^*_{1,l} \in B_1(\pi^*_{2,l}) \text{ and } \pi^*_{2,l} \in B_2(\pi^*_{1,l}) \tag{1}$$

Game Analysis

- Defender strategy is of the form: $\pi_{1,l} : \mathcal{B} \rightarrow \Delta(\mathcal{A}_1)$
- Attacker strategy is of the form: $\pi_{2,l} : \mathcal{S} \times \mathcal{B} \rightarrow \Delta(\mathcal{A}_2)$

- **Defender and attacker objectives**:

$$J_1(\pi_{1,l}, \pi_{2,l}) = \mathbb{E}_{(\pi_{1,l}, \pi_{2,l})}\left[\sum_{t=1}^{\infty} \gamma^{t-1} \mathcal{R}_l(s_t, \boldsymbol{a}_t)\right]$$

$$J_2(\pi_{1,l}, \pi_{2,l}) = -J_1$$

- **Best response correspondences**:

$$B_1(\pi_{2,l}) = \arg\max_{\pi_{1,l} \in \Pi_1} J_1(\pi_{1,l}, \pi_{2,l})$$

$$B_2(\pi_{1,l}) = \arg\max_{\pi_{2,l} \in \Pi_2} J_2(\pi_{1,l}, \pi_{2,l})$$

- **Nash equilibrium** $(\pi_{1,l}^*, \pi_{2,l}^*)$:

$$\pi_{1,l}^* \in B_1(\pi_{2,l}^*) \text{ and } \pi_{2,l}^* \in B_2(\pi_{1,l}^*) \tag{2}$$

Game Analysis

- Defender strategy is of the form: $\pi_{1,l} : \mathcal{B} \rightarrow \Delta(\mathcal{A}_1)$
- Attacker strategy is of the form: $\pi_{2,l} : \mathcal{S} \times \mathcal{B} \rightarrow \Delta(\mathcal{A}_2)$

- **Defender and attacker objectives**:

$$J_1(\pi_{1,l}, \pi_{2,l}) = \mathbb{E}_{(\pi_{1,l}, \pi_{2,l})}\left[\sum_{t=1}^{\infty} \gamma^{t-1} \mathcal{R}_l(s_t, \boldsymbol{a}_t)\right]$$
$$J_2(\pi_{1,l}, \pi_{2,l}) = -J_1$$

- **Best response correspondences**:

$$B_1(\pi_{2,l}) = \arg\max_{\pi_{1,l} \in \Pi_1} J_1(\pi_{1,l}, \pi_{2,l})$$
$$B_2(\pi_{1,l}) = \arg\max_{\pi_{2,l} \in \Pi_2} J_2(\pi_{1,l}, \pi_{2,l})$$

- **Nash equilibrium** $(\pi^*_{1,l}, \pi^*_{2,l})$:

$$\pi^*_{1,l} \in B_1(\pi^*_{2,l}) \text{ and } \pi^*_{2,l} \in B_2(\pi^*_{1,l}) \tag{3}$$

Game Analysis

- Defender strategy is of the form: $\pi_{1,l} : \mathcal{B} \rightarrow \Delta(\mathcal{A}_1)$
- Attacker strategy is of the form: $\pi_{2,l} : \mathcal{S} \times \mathcal{B} \rightarrow \Delta(\mathcal{A}_2)$

- **Defender and attacker objectives**:

$$J_1(\pi_{1,l}, \pi_{2,l}) = \mathbb{E}_{(\pi_{1,l}, \pi_{2,l})}\left[\sum_{t=1}^{\infty} \gamma^{t-1} \mathcal{R}_l(s_t, \boldsymbol{a}_t)\right]$$

$$J_2(\pi_{1,l}, \pi_{2,l}) = -J_1$$

- **Best response correspondences**:

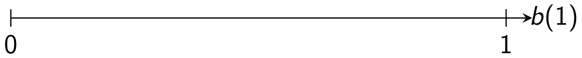
$$B_1(\pi_{2,l}) = \arg\max_{\pi_{1,l} \in \Pi_1} J_1(\pi_{1,l}, \pi_{2,l})$$

$$B_2(\pi_{1,l}) = \arg\max_{\pi_{2,l} \in \Pi_2} J_2(\pi_{1,l}, \pi_{2,l})$$

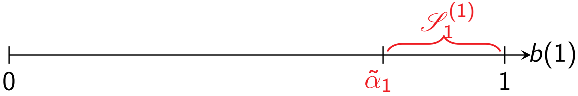
- **Nash equilibrium** $(\pi^*_{1,l}, \pi^*_{2,l})$:

$$\pi^*_{1,l} \in B_1(\pi^*_{2,l}) \text{ and } \pi^*_{2,l} \in B_2(\pi^*_{1,l})$$

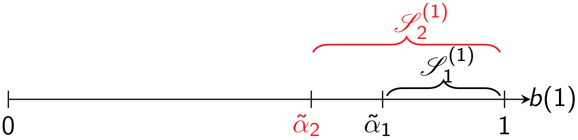
Structure of Best Response Strategies



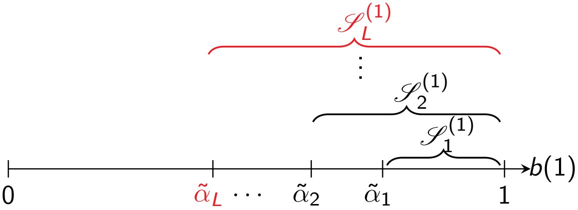
Structure of Best Response Strategies



Structure of Best Response Strategies

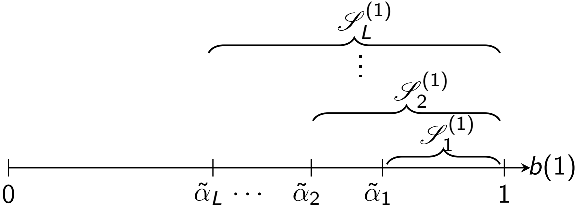


Structure of Best Response Strategies

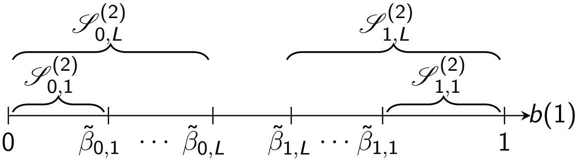


Structure of Best Response Strategies

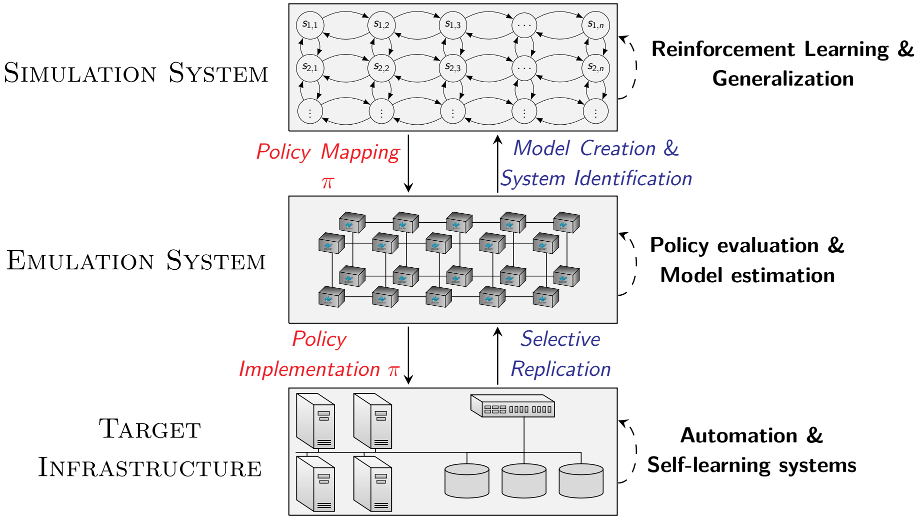
Defender



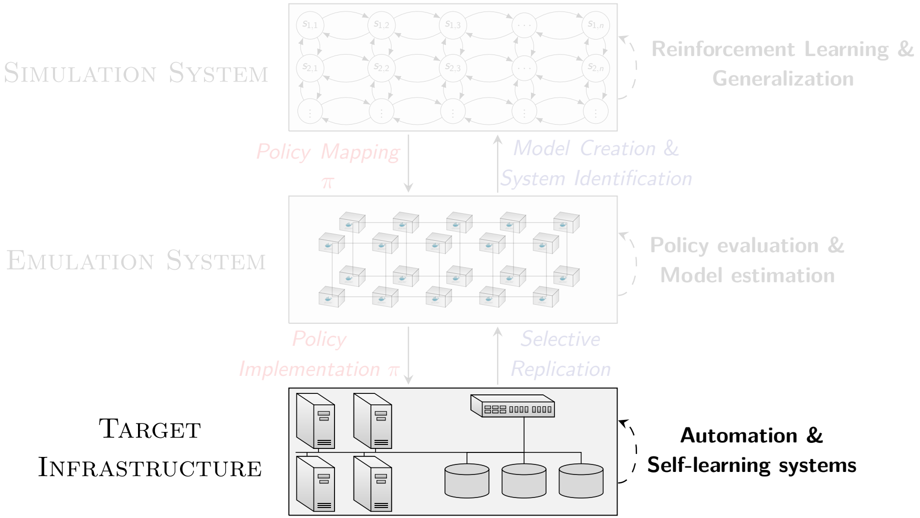
Attacker



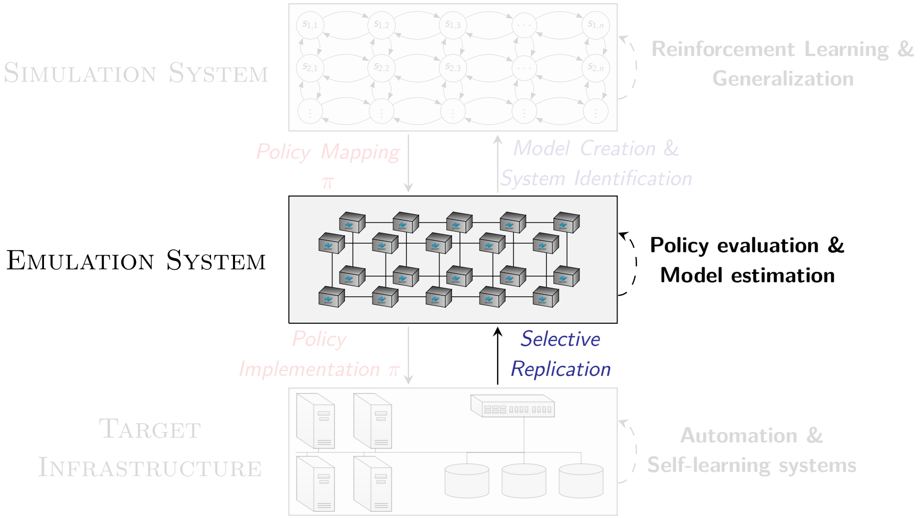
Our Method for Learning Effective Security Strategies



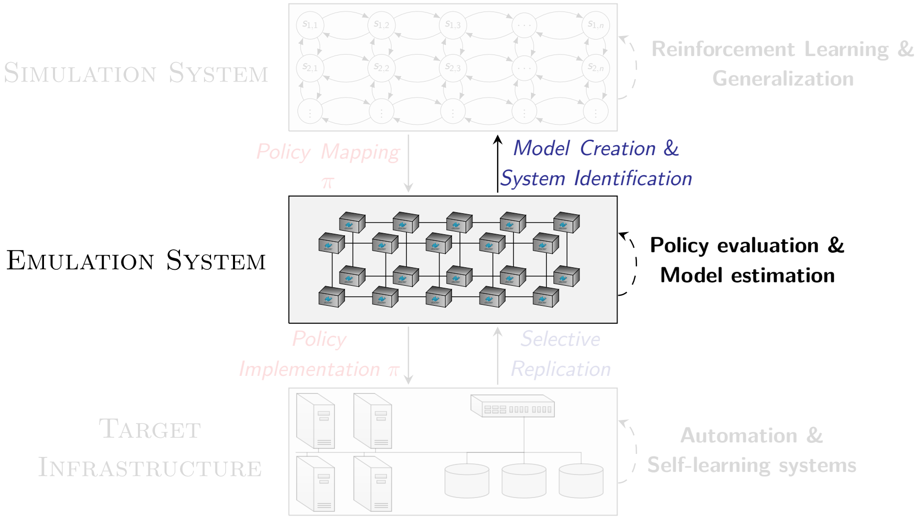
Our Method for Finding Effective Security Strategies



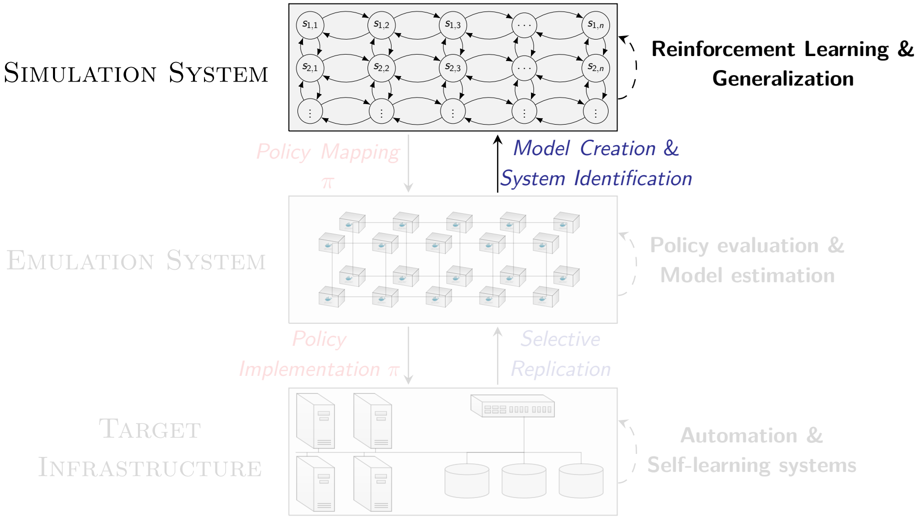
Our Method for Finding Effective Security Strategies



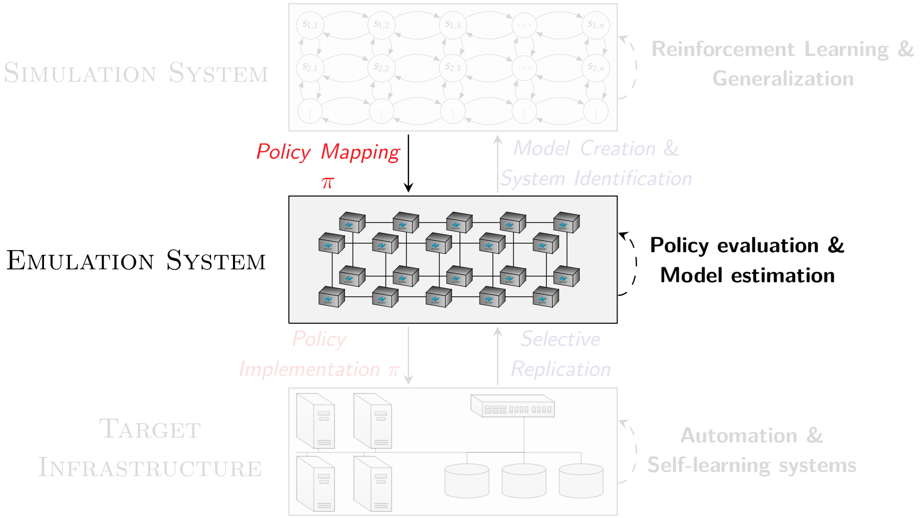
Our Method for Finding Effective Security Strategies



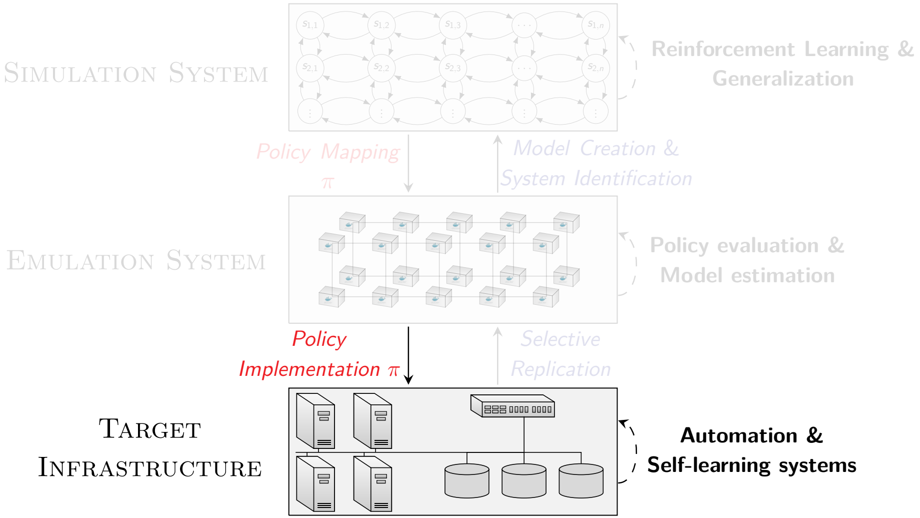
Our Method for Finding Effective Security Strategies



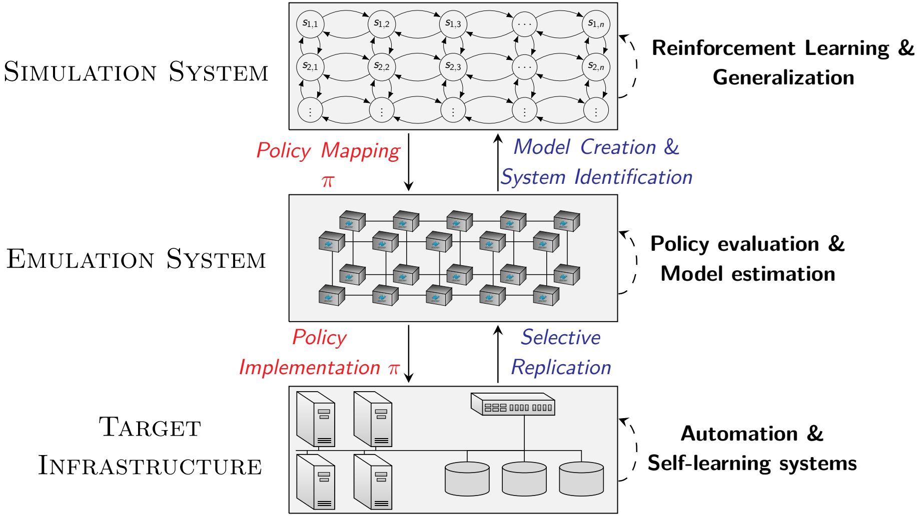
Our Method for Finding Effective Security Strategies



Our Method for Finding Effective Security Strategies

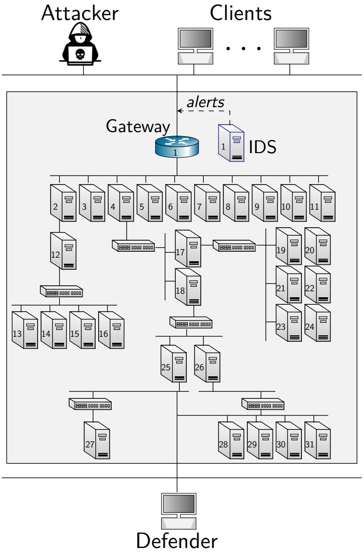


Our Method for Finding Effective Security Strategies



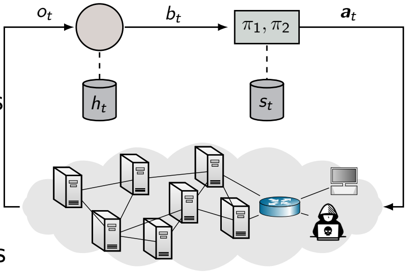
Emulating the Target Infrastructure

- 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



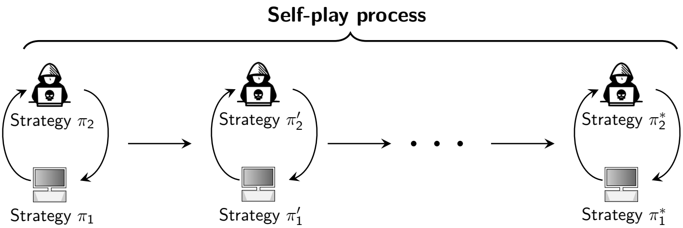
Running a Game Episode in the Emulation

- 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 strategies
- Actions are sent to the emulation using gRPC
- Actions are executed by running commands on the hosts



Our Reinforcement Learning Approach

- We learn a Nash equilibrium $(\pi^*_{1,l,\theta^{(1)}}, \pi^*_{2,l,\theta^{(2)}})$ through **fictitious self-play.**
- In each iteration:
  1. Learn a best response strategy of the defender by solving a POMDP $\tilde{\pi}_{1,l,\theta^{(1)}} \in B_1(\pi_{2,l,\theta^{(2)}})$.
  2. Learn a best response strategy of the attacker by solving an MDP $\tilde{\pi}_{2,l,\theta^{(2)}} \in B_2(\pi_{1,l,\theta^{(1)}})$.
  3. Store the best response strategies in two buffers Θ_1, Θ_2
  4. Update strategies to be the average of the stored strategies



Fictitious Self-Play

Input

Γ^p: the POSG

Output

$(\pi^*_{1,\theta}, \pi^*_{2,\theta})$: an approximate Nash equilibrium

```
procedure ApproximateFP
    θ^(1) ~ N_L(−1, 1),    θ^(2) ~ N_2L(−1, 1)
    Θ^(1) ← {θ^(1)},    Θ^(2) ← {θ^(2)}, δ̂ ← ∞
    while δ̂ ≥ δ do
        θ^(1) ← ThresholdBR(Γ^p, π_{2,l,θ}, N, a, c, λ, A, ε)
        θ^(2) ← ThresholdBR(Γ^p, π_{1,l,θ}, N, a, c, λ, A, ε)
        Θ^(1) ← Θ^(2) ∪ θ^(1),    Θ_2 ← Θ^(2) ∪ θ^(2)
        π_{1,l,θ} ← MixtureDistribution(Θ^(1))
        π_{2,l,θ} ← MixtureDistribution(Θ^(2))
        δ̂ = Exploitability(π_{1,l,θ}, π_{2,l,θ})
    end while
    return (π_{1,l,θ}, π_{2,l,θ})
end procedure
```

Fictitious Self-Play

Input

Γ^p: the POSG

Output

$(\pi^*_{1,\theta}, \pi^*_{2,\theta})$: an approximate Nash equilibrium

procedure $\textsc{ApproximateFP}$

$\theta^{(1)} \sim \mathcal{N}_L(-1,1), \quad \theta^{(2)} \sim \mathcal{N}_{2L}(-1,1)$

$\Theta^{(1)} \leftarrow \{\theta^{(1)}\}, \quad \Theta^{(2)} \leftarrow \{\theta^{(2)}\}, \hat{\delta} \leftarrow \infty$

while $\hat{\delta} \geq \delta$ **do**

$\theta^{(1)} \leftarrow \textsc{ThresholdBR}(\Gamma^p, \pi_{2,l,\theta}, N, a, c, \lambda, A, \epsilon)$

$\theta^{(2)} \leftarrow \textsc{ThresholdBR}(\Gamma^p, \pi_{1,l,\theta}, N, a, c, \lambda, A, \epsilon)$

$\Theta^{(1)} \leftarrow \Theta^{(2)} \cup \theta^{(1)}, \quad \Theta_2 \leftarrow \Theta^{(2)} \cup \theta^{(2)}$

$\pi_{1,l,\theta} \leftarrow \textsc{MixtureDistribution}(\Theta^{(1)})$

$\pi_{2,l,\theta} \leftarrow \textsc{MixtureDistribution}(\Theta^{(2)})$

$\hat{\delta} = \textsc{Exploitability}(\pi_{1,l,\theta}, \pi_{2,l,\theta})$

end while

return $(\pi_{1,l,\theta}, \pi_{2,l,\theta})$

end procedure

Fictitious Self-Play

Input

Γ^p: the POSG

Output

$(\pi^*_{1,\theta}, \pi^*_{2,\theta})$: an approximate Nash equilibrium

procedure $\textsc{ApproximateFP}$
  $\theta^{(1)} \sim \mathcal{N}_L(-1,1), \quad \theta^{(2)} \sim \mathcal{N}_{2L}(-1,1)$
  $\Theta^{(1)} \leftarrow \{\theta^{(1)}\}, \quad \Theta^{(2)} \leftarrow \{\theta^{(2)}\}, \hat{\delta} \leftarrow \infty$
  while $\hat{\delta} \geq \delta$ **do**
    $\theta^{(1)} \leftarrow \textsc{ThresholdBR}(\Gamma^p, \pi_{2,l,\theta}, N, a, c, \lambda, A, \epsilon)$
    $\theta^{(2)} \leftarrow \textsc{ThresholdBR}(\Gamma^p, \pi_{1,l,\theta}, N, a, c, \lambda, A, \epsilon)$
    $\Theta^{(1)} \leftarrow \Theta^{(2)} \cup \theta^{(1)}, \quad \Theta_2 \leftarrow \Theta^{(2)} \cup \theta^{(2)}$
    $\pi_{1,l,\theta} \leftarrow \textsc{MixtureDistribution}(\Theta^{(1)})$
    $\pi_{2,l,\theta} \leftarrow \textsc{MixtureDistribution}(\Theta^{(2)})$
    $\hat{\delta} = \textsc{Exploitability}(\pi_{1,l,\theta}, \pi_{2,l,\theta})$
  end while
  return $(\pi_{1,l,\theta}, \pi_{2,l,\theta})$
end procedure

Fictitious Self-Play

Input

Γ^p: the POSG

Output

$(\pi^*_{1,\theta}, \pi^*_{2,\theta})$: an approximate Nash equilibrium

procedure $\textsc{ApproximateFP}$
    $\theta^{(1)} \sim \mathcal{N}_L(-1,1), \quad \theta^{(2)} \sim \mathcal{N}_{2L}(-1,1)$
    $\Theta^{(1)} \leftarrow \{\theta^{(1)}\}, \quad \Theta^{(2)} \leftarrow \{\theta^{(2)}\}, \hat{\delta} \leftarrow \infty$
    while $\hat{\delta} \geq \delta$ **do**
        $\theta^{(1)} \leftarrow \textsc{ThresholdBR}(\Gamma^p, \pi_{2,l,\theta}, N, a, c, \lambda, A, \epsilon)$
        $\theta^{(2)} \leftarrow \textsc{ThresholdBR}(\Gamma^p, \pi_{1,l,\theta}, N, a, c, \lambda, A, \epsilon)$
        $\Theta^{(1)} \leftarrow \Theta^{(2)} \cup \theta^{(1)}, \quad \Theta_2 \leftarrow \Theta^{(2)} \cup \theta^{(2)}$
        $\pi_{1,l,\theta} \leftarrow \textsc{MixtureDistribution}(\Theta^{(1)})$
        $\pi_{2,l,\theta} \leftarrow \textsc{MixtureDistribution}(\Theta^{(2)})$
        $\hat{\delta} = \textsc{Exploitability}(\pi_{1,l,\theta}, \pi_{2,l,\theta})$
    end while
    return $(\pi_{1,l,\theta}, \pi_{2,l,\theta})$
end procedure

Fictitious Self-Play

Input

Γ^p: the POSG

Output

$(\pi^*_{1,\theta}, \pi^*_{2,\theta})$: an approximate Nash equilibrium

procedure $\textsc{ApproximateFP}$

  $\theta^{(1)} \sim \mathcal{N}_L(-1,1), \quad \theta^{(2)} \sim \mathcal{N}_{2L}(-1,1)$

  $\Theta^{(1)} \leftarrow \{\theta^{(1)}\}, \quad \Theta^{(2)} \leftarrow \{\theta^{(2)}\}, \hat{\delta} \leftarrow \infty$

  while $\hat{\delta} \geq \delta$ **do**

    $\theta^{(1)} \leftarrow \textsc{ThresholdBR}(\Gamma^p, \pi_{2,l,\theta}, N, a, c, \lambda, A, \epsilon)$

    $\theta^{(2)} \leftarrow \textsc{ThresholdBR}(\Gamma^p, \pi_{1,l,\theta}, N, a, c, \lambda, A, \epsilon)$

    $\Theta^{(1)} \leftarrow \Theta^{(2)} \cup \theta^{(1)}, \quad \Theta_2 \leftarrow \Theta^{(2)} \cup \theta^{(2)}$

    $\pi_{1,l,\theta} \leftarrow \textsc{MixtureDistribution}(\Theta^{(1)})$

    $\pi_{2,l,\theta} \leftarrow \textsc{MixtureDistribution}(\Theta^{(2)})$

    $\hat{\delta} = \textsc{Exploitability}(\pi_{1,l,\theta}, \pi_{2,l,\theta})$

  end while

  return $(\pi_{1,l,\theta}, \pi_{2,l,\theta})$

end procedure

Our Reinforcement Learning Algorithm for Learning Best-Response Threshold Strategies

- We **use the structural result** that threshold best response strategies exist (Theorem 1) **to design an efficient reinforcement learning algorithm** to learn best response strategies.

- We seek to learn:
  - L thresholds of the defender, $\tilde{\alpha}_1, \geq \tilde{\alpha}_2, \ldots, \geq \tilde{\alpha}_L \in [0,1]$
  - $2L$ thresholds of the attacker, $\tilde{\beta}_{0,1}, \tilde{\beta}_{1,1}, \ldots, \tilde{\beta}_{0,L}, \tilde{\beta}_{1,L} \in [0,1]$

- We learn these thresholds iteratively through Robbins and Monro's stochastic approximation algorithm.[1]

[1] 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 Best-Response Threshold Strategies

- We **use the structural result** that threshold best response strategies exist (Theorem 1) **to design an efficient reinforcement learning algorithm** to learn best response strategies.

- We seek to learn:
  - L thresholds of the defender, $\tilde{\alpha}_1, \geq \tilde{\alpha}_2, \ldots, \geq \tilde{\alpha}_L \in [0, 1]$
  - $2L$ thresholds of the attacker, $\tilde{\beta}_1, \tilde{\beta}_2, \ldots, \tilde{\beta}_L \in [0, 1]$

- We learn these thresholds iteratively through Robbins and Monro's stochastic approximation algorithm.[2]

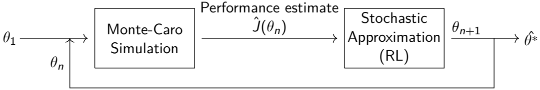
[2] Herbert Robbins and Sutton Monro. "A Stochastic Approximation Method". In: *The Annals of Mathematical Statistics* 22.3 (1951), pp. 400 –407. DOI: 10.1214/aoms/1177729586. URL: https://doi.org/10.1214/aoms/1177729586.

Our Reinforcement Learning Algorithm for Learning Best-Response Threshold Strategies

- We **use the structural result** that threshold best response strategies exist (Theorem 1) **to design an efficient reinforcement learning algorithm** to learn best response strategies.

- We seek to learn:
  - L thresholds of the defender, $\tilde{\alpha}_1, \geq \tilde{\alpha}_2, \ldots, \geq \tilde{\alpha}_L \in [0, 1]$
  - $2L$ thresholds of the attacker, $\tilde{\beta}_1, \tilde{\beta}_2, \ldots, \tilde{\beta}_L \in [0, 1]$

- We learn these thresholds iteratively through Robbins and Monro's stochastic approximation algorithm.[3]



[3] 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

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

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

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

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

   where $\sigma(\cdot)$ is the sigmoid function and $\sigma(\theta^{(i),j})$ is the threshold.

Our Reinforcement Learning Algorithm

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

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

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

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

   where $\sigma(\cdot)$ is the sigmoid function and $\sigma(\theta^{(i),j})$ is the threshold.

Our Reinforcement Learning Algorithm

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

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

exists as long as $\pi_{i,I,\theta^{(i)}}$ is differentiable.

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

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

where $\sigma(\cdot)$ is the sigmoid function and $\sigma(\theta^{(i),j})$ is the threshold.

Our Reinforcement Learning Algorithm

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

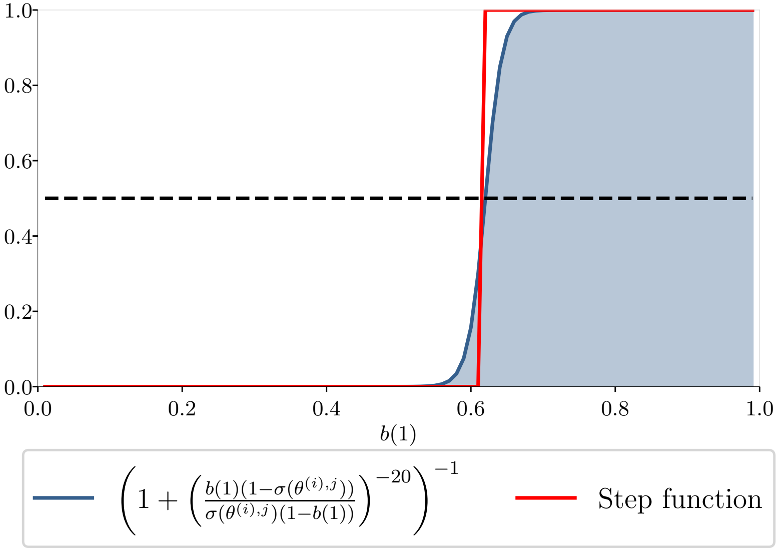
$$\nabla_{\theta^{(i)}} J(\theta^{(i)}) = \mathbb{E}_{\pi_{i,l,\theta^{(i)}}}\left[\sum_{t=1}^{\infty} \nabla_{\theta^{(i)}} \log \pi_{i,l,\theta^{(i)}}(a_t^{(i)}|s_t) \sum_{\tau=t}^{\infty} r_t\right]$$

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

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

   where $\sigma(\cdot)$ is the sigmoid function and $\sigma(\theta^{(i),j})$ is the threshold.

Smooth Threshold



Our Reinforcement Learning Algorithm

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

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

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

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

[4] 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

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

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

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

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

[4] 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

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

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

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

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

[4] 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

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

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

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

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

[4] 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

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

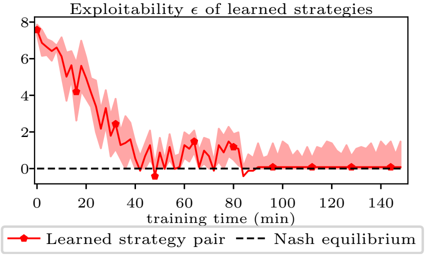
$$\left(\hat{\nabla}_{\theta_n^{(i)}} J_i(\theta_n^{(i)})\right)_k = \frac{\hat{J}_i(\theta_n^{(i)} + c_n\Delta_n) - \hat{J}_i(\theta_n^{(i)} - c_n\Delta_n)}{2c_n(\Delta_n)_k}$$

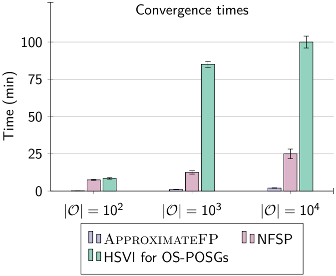
5. Next, we use the estimated gradient and update the vector of thresholds through the stochastic approximation update:

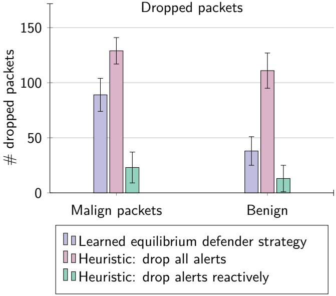
$$\theta_{n+1}^{(i)} = \theta_n^{(i)} + a_n \hat{\nabla}_{\theta_n^{(i)}} J_i(\theta_n^{(i)})$$

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

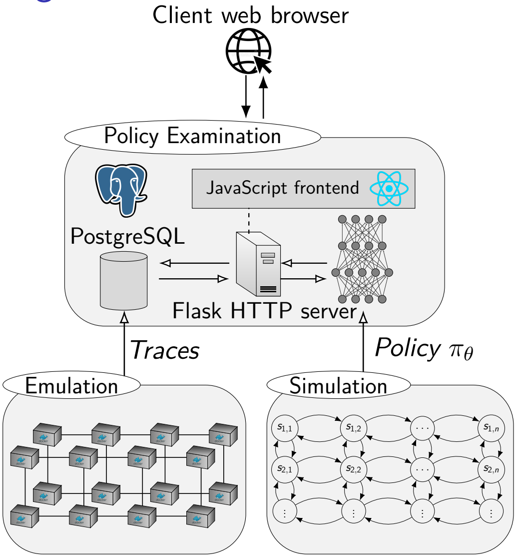
Simulation Results (Emulation Results TBD..)







Demo - A System for Interactive Examination of Learned Security Strategies



Architecture of the system for examining learned security strategies.

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 Partially Observed Stochastic Game of the use case
    - We present a POMDP model of the defender's problem
    - We present a MDP model of the attacker's problem
    - We apply the stopping theory to establish structural results of the best response strategies
- **Our research plans:**
  - Run experiments in the emulation system
  - Make learned strategy available as plugin to the Snort IDS
  - Extend the model
    - Less restrictions on defender

    Scaling up the emulation system:
    - Non-static infrastructures