Self-Learning Systems for Cyber Security

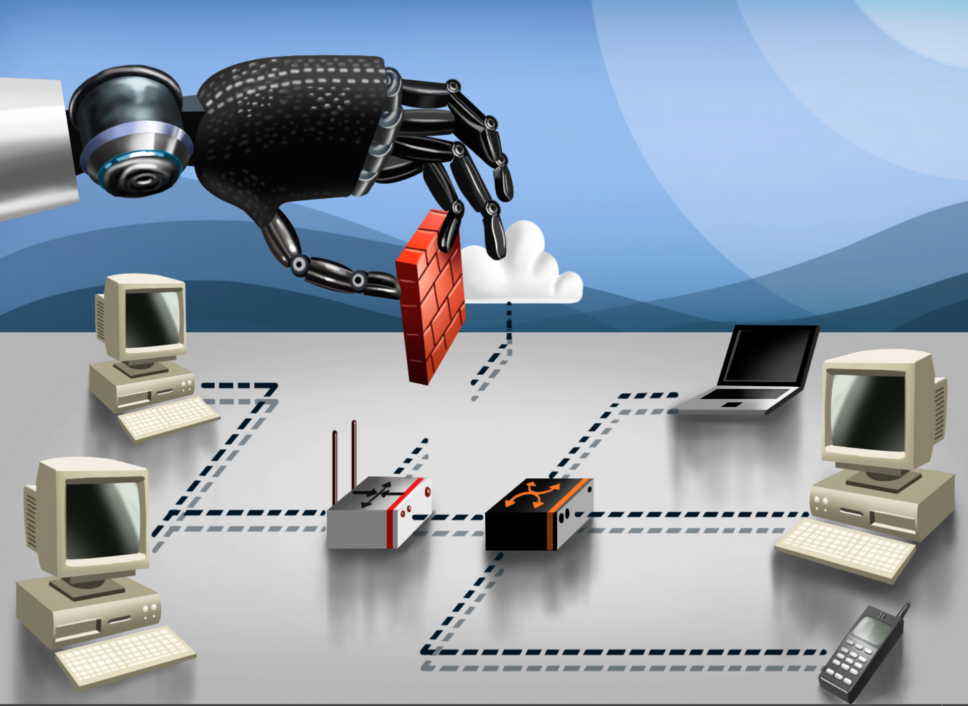
Ledningsregementet Enköping

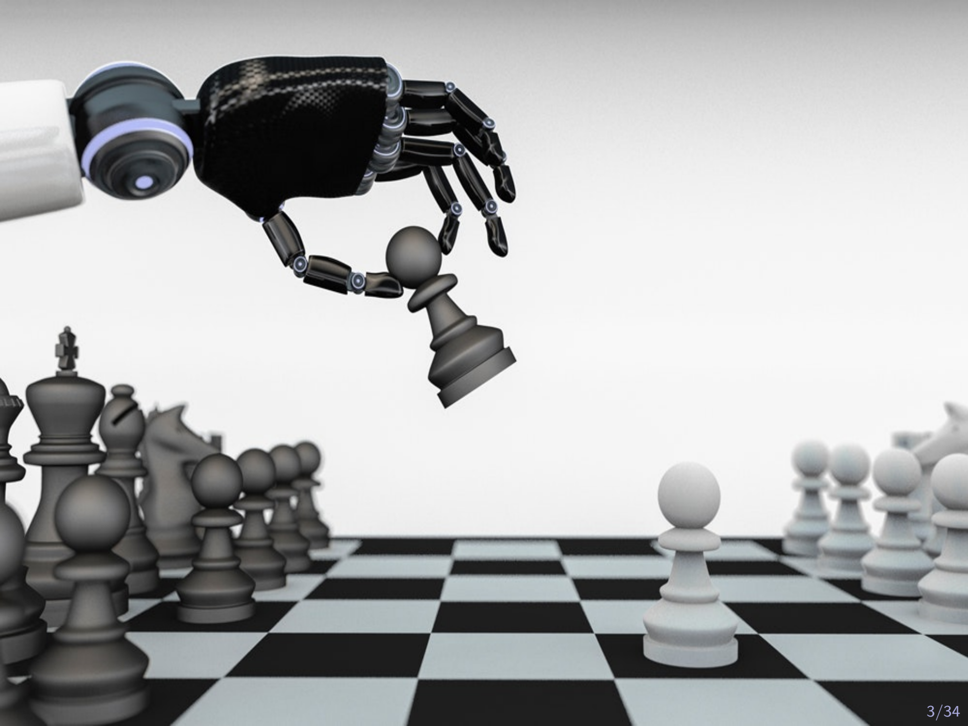
Kim Hammar & Rolf Stadler

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

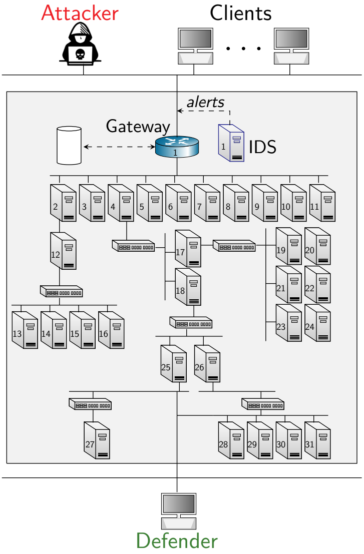
Division of Network and Systems Engineering
KTH Royal Institute of Technology

August 18, 2021





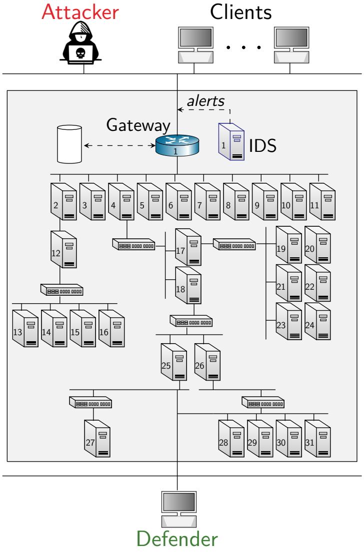
Challenges: Evolving and Automated Attacks



- **Challenges**:
  - Evolving & automated attacks
  - Complex infrastructures

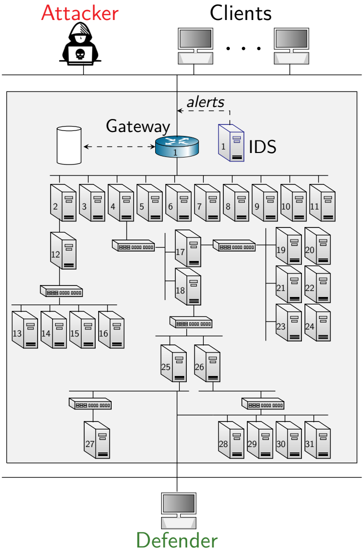
Goal: Automation and Learning

- **Challenges**
  - Evolving & automated attacks
  - Complex infrastructures
- **Our Goal**:
  - Automate security tasks
  - Adapt to changing attack methods



Approach: Game Model & Reinforcement Learning

- **Challenges**:
  - Evolving & automated attacks
  - Complex infrastructures
- **Our Goal**:
  - Automate security tasks
  - Adapt to changing attack methods
- **Our Approach**:
  - Model network attack and defense as *games*.
  - Use *reinforcement learning* to learn policies.
  - Incorporate learned policies in *self-learning systems*.



State of the Art

- **Game-Learning Programs**:
  - TD-Gammon, AlphaGo Zero[1], OpenAI Five etc.
  - $\implies$ Impressive empirical results of *RL and self-play*
- **Attack Simulations**:
  - Automated threat modeling[2] and intrusion detection etc.
  - $\implies$ Need for *automation* and better security tooling
- **Mathematical Modeling**:
  - Game theory[3]
  - Markov decision theory, dynamic programming[4]
  - $\implies$ Many security operations involves *strategic decision making*

---

[1] David Silver et al. "Mastering the game of Go without human knowledge". In: *Nature* 550 (Oct. 2017), pp. 354–. URL: http://dx.doi.org/10.1038/nature24270.

[2] Pontus Johnson, Robert Lagerström, and Mathias Ekstedt. "A Meta Language for Threat Modeling and Attack Simulations". In: *Proceedings of the 13th International Conference on Availability, Reliability and Security*. ARES 2018. Hamburg, Germany: Association for Computing Machinery, 2018. ISBN: 9781450364485. DOI: 10.1145/3230833.3232799. URL: https://doi.org/10.1145/3230833.3232799.

[3] Tansu Alpcan and Tamer Basar. *Network Security: A Decision and Game-Theoretic Approach*. 1st. USA: Cambridge University Press, 2010. ISBN: 0521119324.

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

State of the Art

- **Game-Learning Programs**:
  - TD-Gammon, AlphaGo Zero[5], OpenAI Five etc.
  - $\implies$ Impressive empirical results of *RL and self-play*
- **Attack Simulations**:
  - Automated threat modeling[6] and intrusion detection etc.
  - $\implies$ Need for *automation* and better security tooling
- **Mathematical Modeling**:
  - Game theory[7]
  - Markov decision theory, dynamic programming[8]
  - $\implies$ Many security operations involves *strategic decision making*

---

[5] David Silver et al. "Mastering the game of Go without human knowledge". In: *Nature* 550 (Oct. 2017), pp. 354–. URL: http://dx.doi.org/10.1038/nature24270.

[6] Pontus Johnson, Robert Lagerström, and Mathias Ekstedt. "A Meta Language for Threat Modeling and Attack Simulations". In: *Proceedings of the 13th International Conference on Availability, Reliability and Security*. ARES 2018. Hamburg, Germany: Association for Computing Machinery, 2018. ISBN: 9781450364485. DOI: 10.1145/3230833.3232799. URL: https://doi.org/10.1145/3230833.3232799.

[7] Tansu Alpcan and Tamer Basar. *Network Security: A Decision and Game-Theoretic Approach*. 1st. USA: Cambridge University Press, 2010. ISBN: 0521119324.

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

State of the Art

- **Game-Learning Programs**:
  - TD-Gammon, AlphaGo Zero[9], OpenAI Five etc.
  - $\implies$ Impressive empirical results of *RL and self-play*
- **Attack Simulations**:
  - Automated threat modeling[10] and intrusion detection etc.
  - $\implies$ Need for *automation* and better security tooling
- **Mathematical Modeling**:
  - Game theory[11]
  - Markov decision theory, dynamic programming[12]
  - $\implies$ Many security operations involves *strategic decision making*

---

[9] David Silver et al. "Mastering the game of Go without human knowledge". In: *Nature* 550 (Oct. 2017), pp. 354–. URL: http://dx.doi.org/10.1038/nature24270.

[10] Pontus Johnson, Robert Lagerström, and Mathias Ekstedt. "A Meta Language for Threat Modeling and Attack Simulations". In: *Proceedings of the 13th International Conference on Availability, Reliability and Security*. ARES 2018. Hamburg, Germany: Association for Computing Machinery, 2018. ISBN: 9781450364485. DOI: 10.1145/3230833.3232799. URL: https://doi.org/10.1145/3230833.3232799.

[11] Tansu Alpcan and Tamer Basar. *Network Security: A Decision and Game-Theoretic Approach*. 1st. USA: Cambridge University Press, 2010. ISBN: 0521119324.

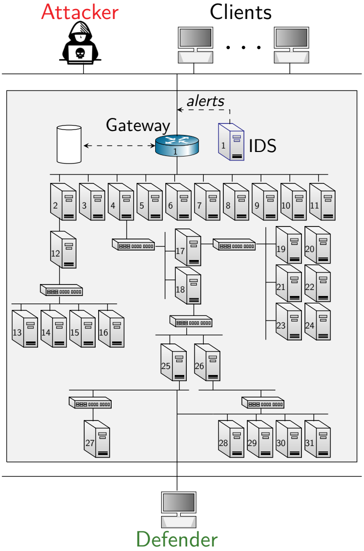
[12] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. 3rd. Vol. I. Belmont, MA, USA: Athena Scientific, 2005.

Our Work

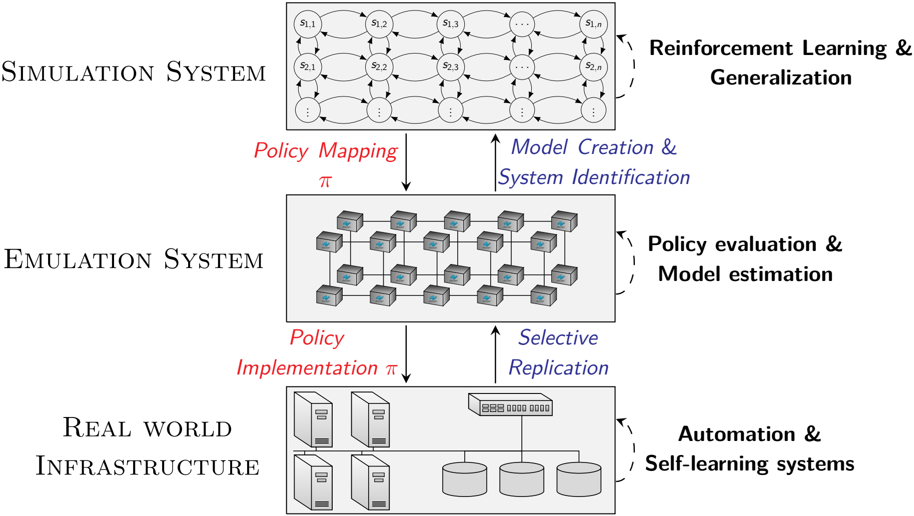
- **Use Case:** Intrusion Prevention
- **Our Method:**
  - Emulating computer infrastructures
  - System identification and model creation
  - Reinforcement learning and generalization
- **Results:**
  - Learning to Capture The Flag
  - Learning to Prevent Attacks (Optimal Stopping)
- **Conclusions and Future Work**

Use Case: Intrusion Prevention

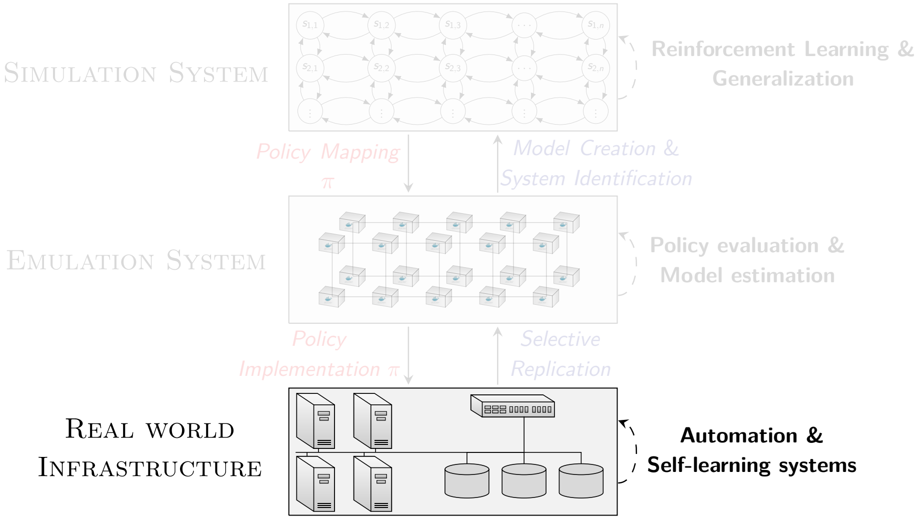
- A **Defender** owns an infrastructure
  - Consists of connected components
  - Components run network services
  - Defender defends the infrastructure by monitoring and active defense
- An **Attacker** seeks to intrude on the infrastructure
  - Has a partial view of the infrastructure
  - Wants to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



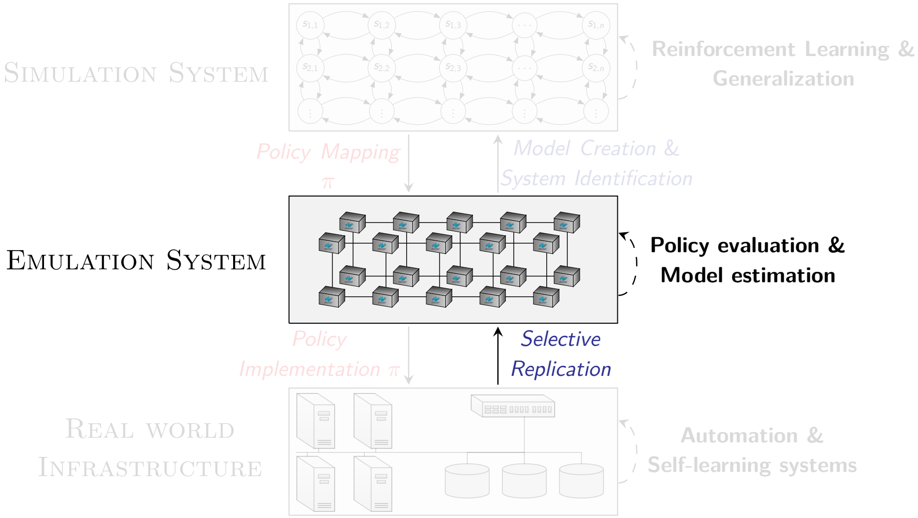
Our Method for Finding Effective Security Strategies



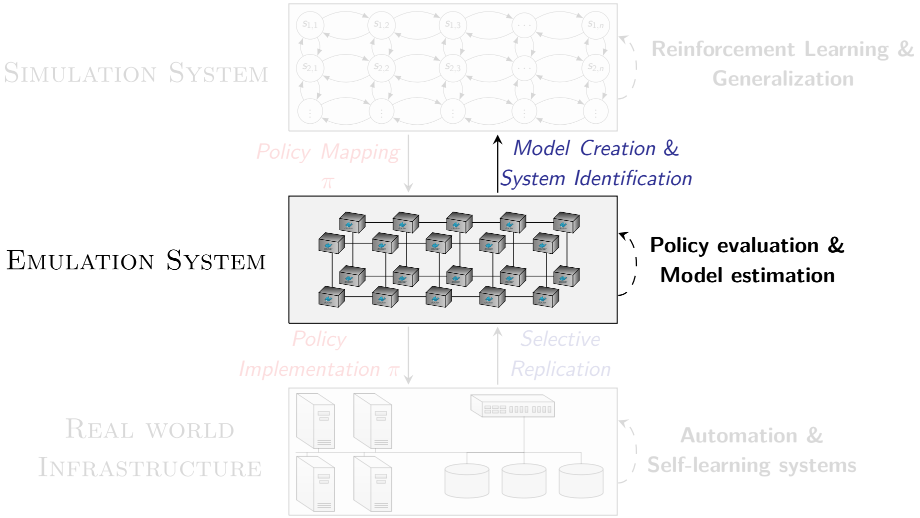
Our Method for Finding Effective Security Strategies



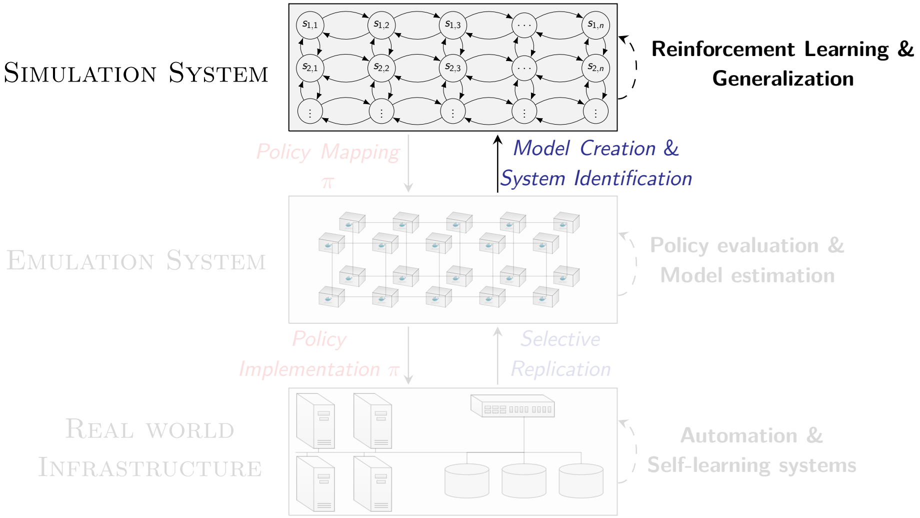
Our Method for Finding Effective Security Strategies



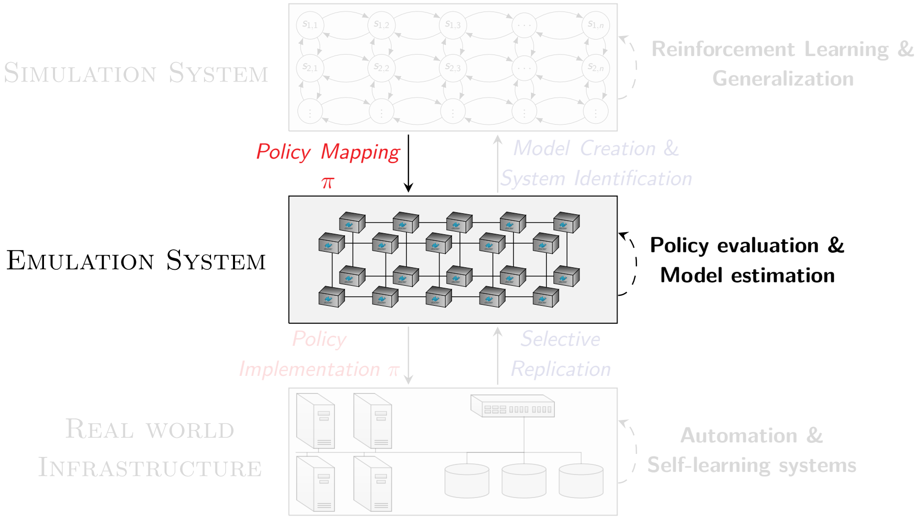
Our Method for Finding Effective Security Strategies



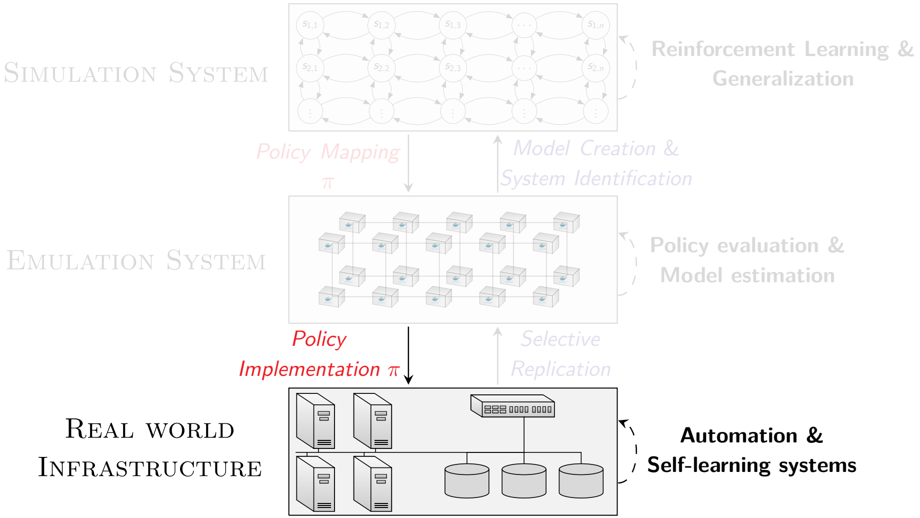
Our Method for Finding Effective Security Strategies



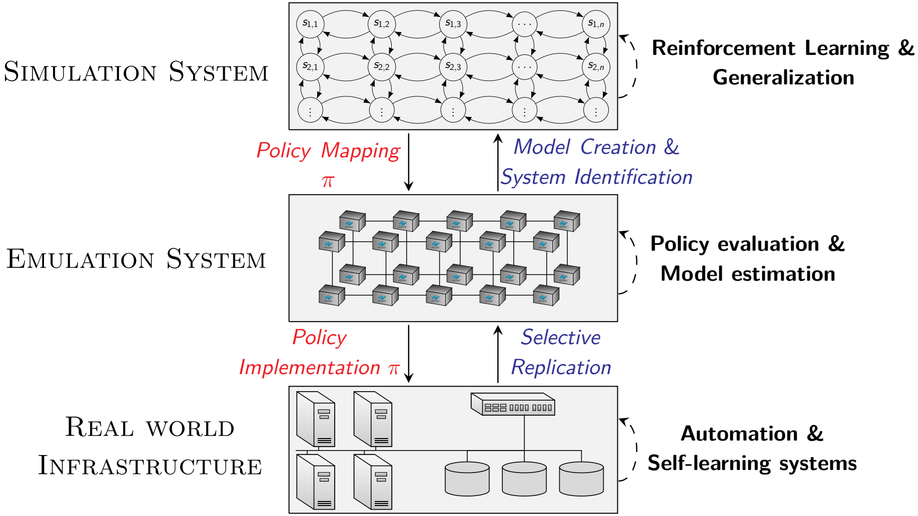
Our Method for Finding Effective Security Strategies



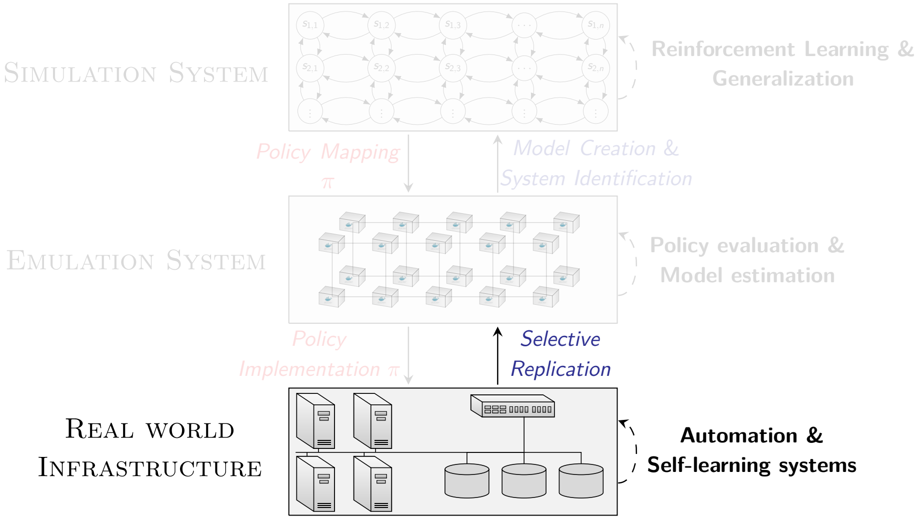
Our Method for Finding Effective Security Strategies



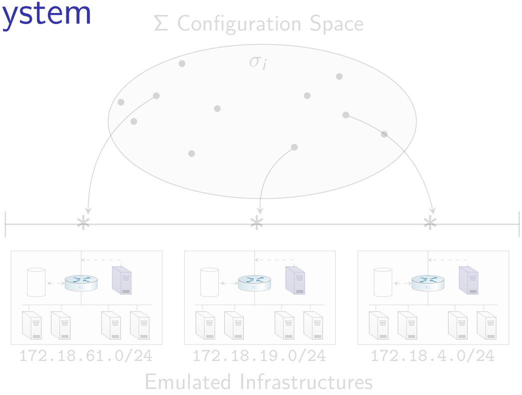
Our Method for Finding Effective Security Strategies



Our Method for Finding Effective Security Strategies



Emulation System

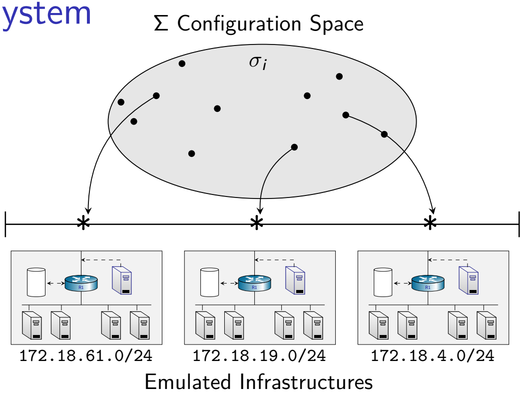


Emulation

A cluster of machines that runs a virtualized infrastructure which replicates important functionality of target systems.

- The set of virtualized configurations define a *configuration space* $\Sigma = \langle \mathcal{A}, \mathcal{O}, \mathcal{S}, \mathcal{U}, \mathcal{T}, \mathcal{V} \rangle$.
- A specific emulation is based on a configuration $\sigma_i \in \Sigma$.

Emulation System

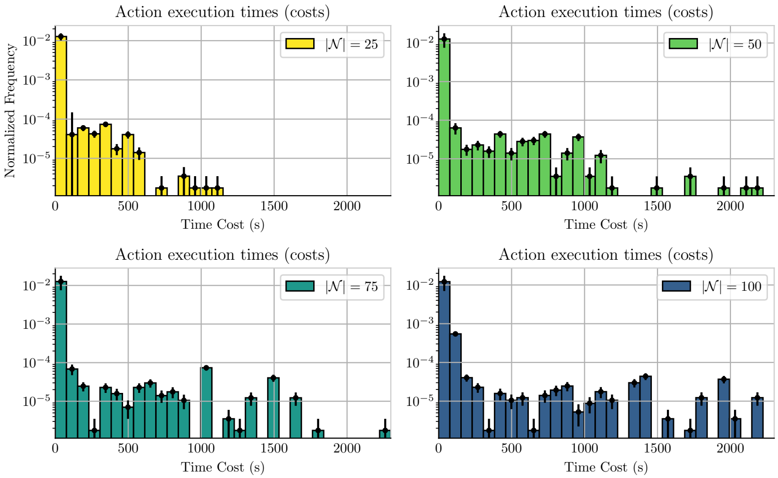


Emulation

A cluster of machines that runs a virtualized infrastructure which replicates important functionality of target systems.

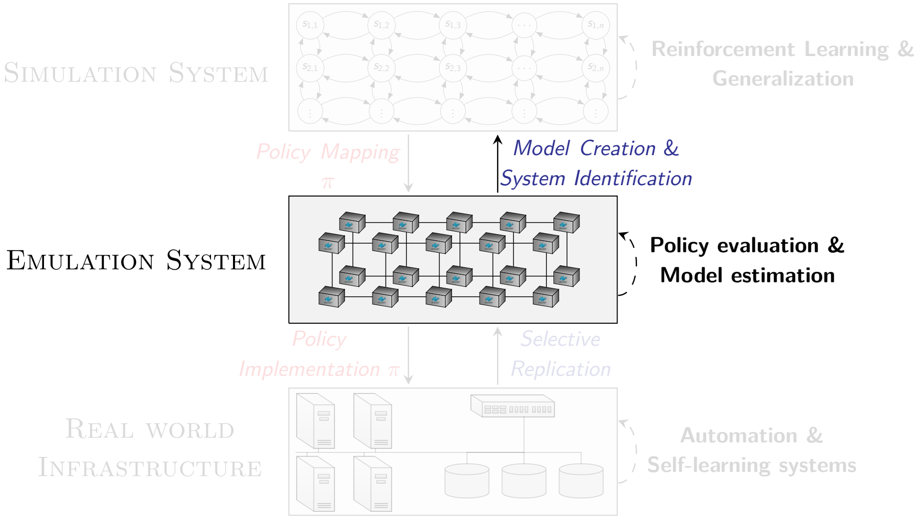
- The set of virtualized configurations define a *configuration space* $\Sigma = \langle \mathcal{A}, \mathcal{O}, \mathcal{S}, \mathcal{U}, \mathcal{T}, \mathcal{V} \rangle$.
- A specific emulation is based on a configuration $\sigma_i \in \Sigma$.

Emulation: Execution Times of Replicated Operations

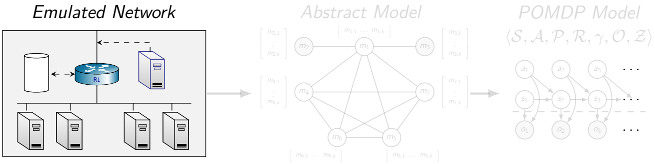


- **Fundamental issue**: Computational methods for policy learning typically require samples on the order of $100k - 10M$.
- $\Longrightarrow$ Infeasible to optimize in the emulation system

Our Method for Finding Effective Security Strategies



From Emulation to Simulation: System Identification

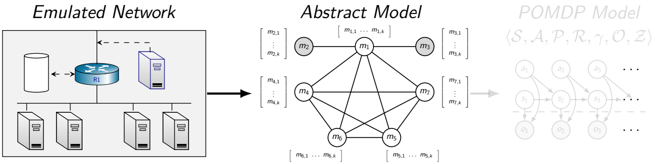


- **Abstract Model Based on Domain Knowledge**: Models the set of *controls*, the *objective function*, and the *features* of the emulated network.
  - Defines the static parts a POMDP model.

- **Dynamics Model ($\mathcal{P}$, $\mathcal{Z}$) Identified using System Identification**: Algorithm based on random walks and maximum-likelihood estimation.

$$\mathcal{M}(b'|b,a) \triangleq \frac{n(b,a,b')}{\sum_{j'} n(s,a,j')}$$

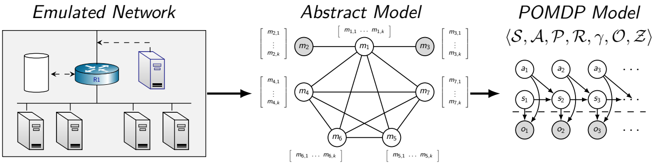
From Emulation to Simulation: System Identification



- **Abstract Model Based on Domain Knowledge**: Models the set of *controls*, the *objective function*, and the *features* of the emulated network.
  - Defines the static parts a POMDP model.
- **Dynamics Model ($\mathcal{P}$, $\mathcal{Z}$) Identified using System Identification**: Algorithm based on random walks and maximum-likelihood estimation.

$$\mathcal{M}(b'|b, a) \triangleq \frac{n(b, a, b')}{\sum_{j'} n(s, a, j')}$$

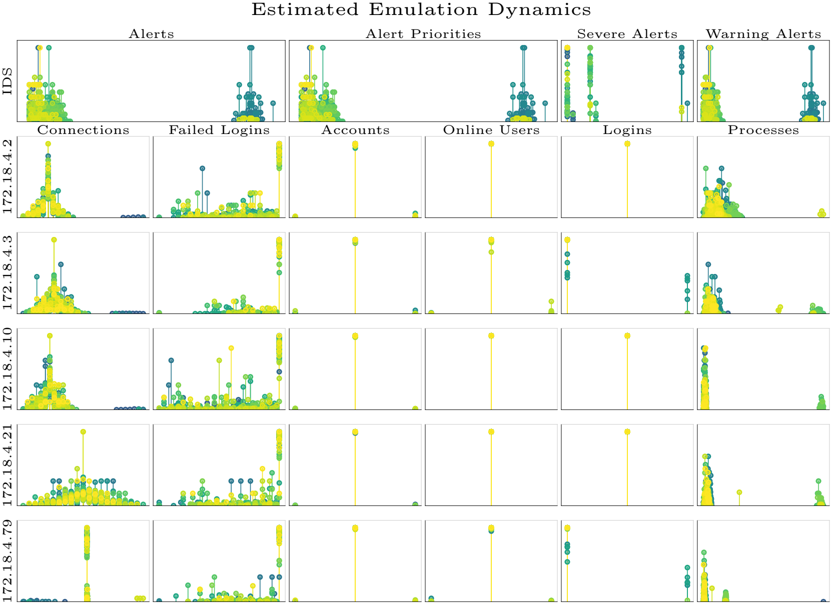
From Emulation to Simulation: System Identification



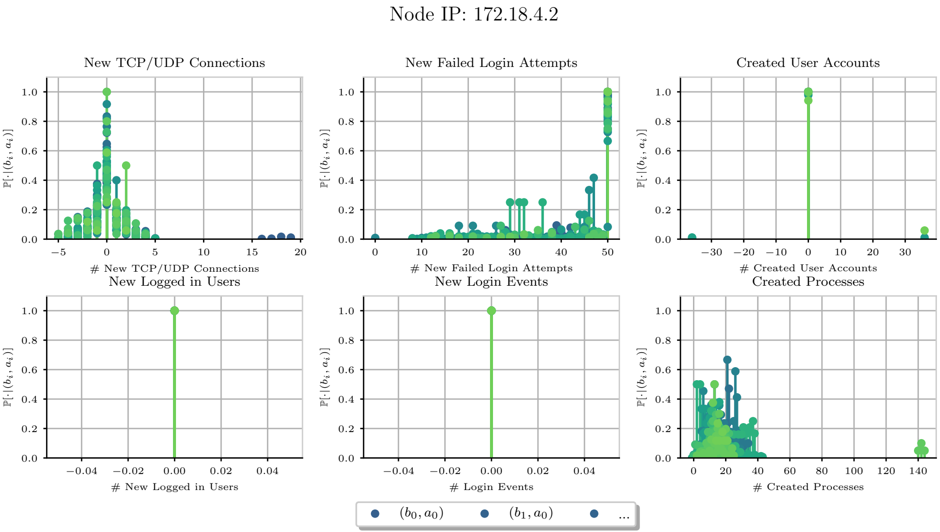
- **Abstract Model Based on Domain Knowledge**: Models the set of *controls*, the *objective function*, and the *features* of the emulated network.
  - Defines the static parts a POMDP model.
- **Dynamics Model ($\mathcal{P}$, $\mathcal{Z}$) Identified using System Identification**: Algorithm based on random walks and maximum-likelihood estimation.

$$\mathcal{M}(b'|b,a) \triangleq \frac{n(b,a,b')}{\sum_{j'} n(s,a,j')}$$

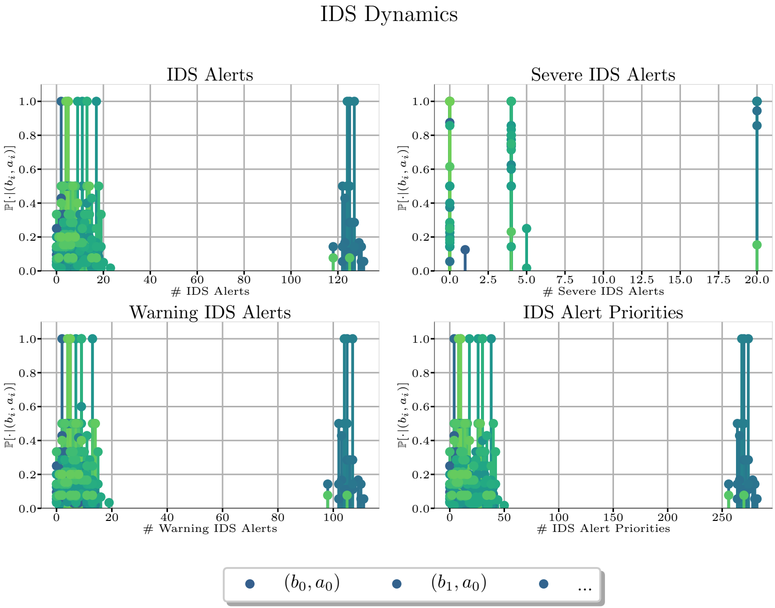
System Identification: Estimated Dynamics Model



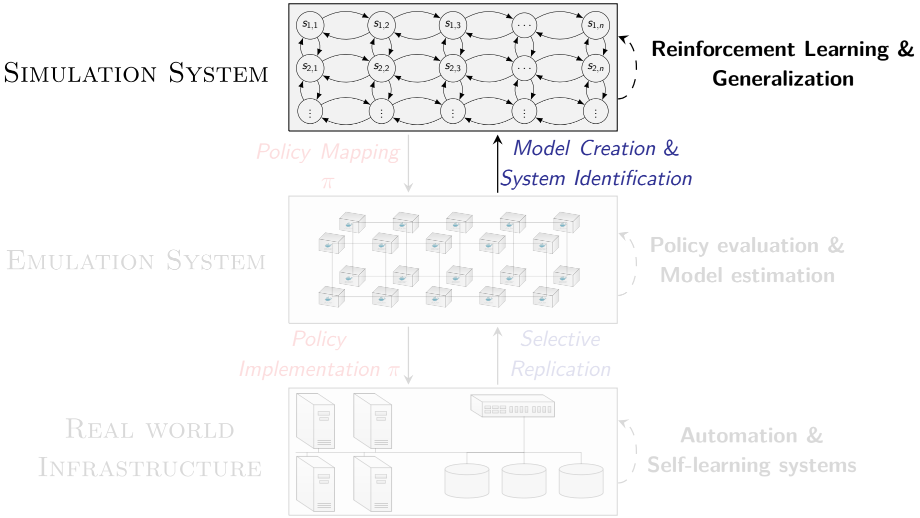
System Identification: Estimated Dynamics Model



System Identification: Estimated Dynamics Model



Our Method for Finding Effective Security Strategies



Policy Optimization in the Simulation System using Reinforcement Learning

- **Goal**:
  - Approximate $\pi^* = \arg\max_\pi \mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1} r_{t+1}\right]$

- **Learning Algorithm**:
  - Represent π by π_θ
  - Define objective $J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T} \gamma^{t-1} r(s_t, a_t)\right]$
  - Maximize $J(\theta)$ by stochastic gradient ascent

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\underbrace{\nabla_\theta \log \pi_\theta(a|s)}_{\text{actor}} \underbrace{A^{\pi_\theta}(s,a)}_{\text{critic}}\right]$$

- **Domain-Specific Challenges**:
  - Partial observability
  - Large state space
  - Large action space
  - Non-stationary Environment due to attacker
  - Generalization

Policy Optimization in the Simulation System using Reinforcement Learning

- **Goal**:
  - Approximate $\pi^* = \arg\max_\pi \mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1} r_{t+1}\right]$

- **Learning Algorithm**:
  - Represent π by π_θ
  - Define objective $J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T} \gamma^{t-1} r(s_t, a_t)\right]$
  - Maximize $J(\theta)$ by stochastic gradient ascent

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\underbrace{\nabla_\theta \log \pi_\theta(a|s)}_{\text{actor}} \underbrace{A^{\pi_\theta}(s, a)}_{\text{critic}}\right]$$

- **Domain-Specific Challenges**:
  - Partial observability
  - Large state space
  - Large action space
  - Non-stationary Environment due to attacker
  - Generalization

Policy Optimization in the Simulation System using Reinforcement Learning

- **Goal**:
  - Approximate $\pi^* = \arg\max_\pi \mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1} r_{t+1}\right]$

- **Learning Algorithm**:
  - Represent π by π_θ
  - Define objective $J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T} \gamma^{t-1} r(s_t, a_t)\right]$
  - Maximize $J(\theta)$ by stochastic gradient ascent

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\underbrace{\nabla_\theta \log \pi_\theta(a|s)}_{\text{actor}} \underbrace{A^{\pi_\theta}(s, a)}_{\text{critic}}\right]$$

- **Domain-Specific Challenges**:
  - Partial observability
  - Large state space
  - Large action space
  - Non-stationary Environment due to attacker
  - Generalization

Policy Optimization in the Simulation System using Reinforcement Learning

- **Goal**:
  - Approximate $\pi^* = \arg\max_\pi \mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1} r_{t+1}\right]$

- **Learning Algorithm**:
  - Represent π by π_θ
  - Define objective $J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T} \gamma^{t-1} r(s_t, a_t)\right]$
  - Maximize $J(\theta)$ by stochastic gradient ascent

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\underbrace{\nabla_\theta \log \pi_\theta(a|s)}_{\text{actor}} \underbrace{A^{\pi_\theta}(s, a)}_{\text{critic}}\right]$$

- **Domain-Specific Challenges**:
  - Partial observability
  - Large state space
  - Large action space
  - Non-stationary Environment due to attacker
  - Generalization

Policy Optimization in the Simulation System using Reinforcement Learning

- **Goal**:
  - Approximate $\pi^* = \arg\max_\pi \mathbb{E}[\sum_{t=1}^{T} \gamma^{t-1} r_{t+1}]$
- **Learning Algorithm**:
  - Represent π by π_θ
  - Define objective $J(\theta) = \mathbb{E}_{\pi_\theta}[\sum_{t=1}^{T} \gamma^{t-1} r(s_t, a_t)]$
  - Maximize $J(\theta)$ by stochastic gradient ascent
    $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) A^{\pi_\theta}(s, a)]$
- **Domain-Specific Challenges**:
  - Partial observability
  - Large state space
  - Large action space
  - Non-stationary Environment due to attacker
  - Generalization

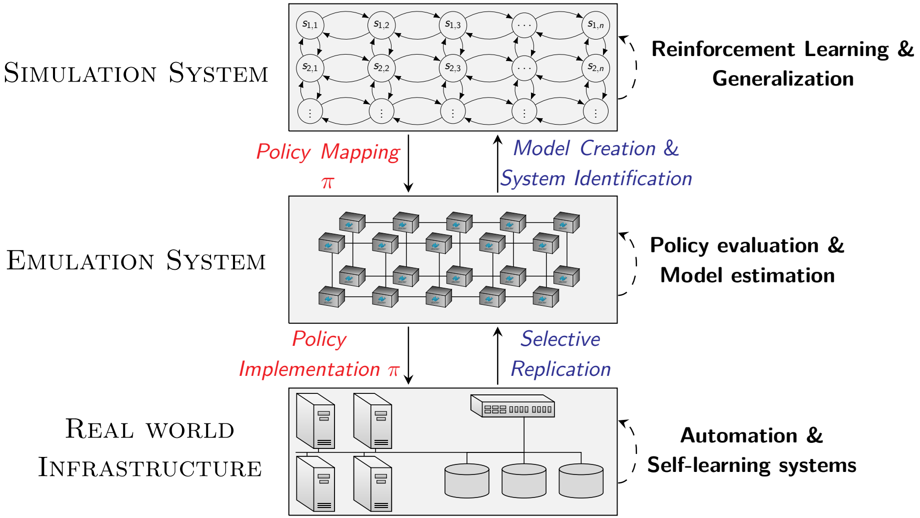
- *Finding Effective Security Strategies through Reinforcement Learning and Self-Play*[a]
- *Learning Intrusion Prevention Policies through Optimal Stopping*[b]

---

[a] Kim Hammar and Rolf Stadler. "Finding Effective Security Strategies through Reinforcement Learning and Self-Play". In: *International Conference on Network and Service Management (CNSM)*. Izmir, Turkey, Nov. 2020.

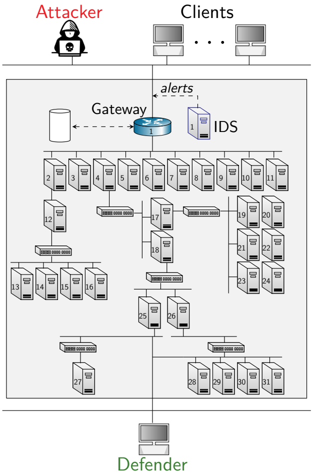
[b] Kim Hammar and Rolf Stadler. *Learning Intrusion Prevention Policies through Optimal Stopping*. 2021. arXiv: 2106.07160 [cs.AI].

Our Method for Finding Effective Security Strategies



The Target Infrastructure

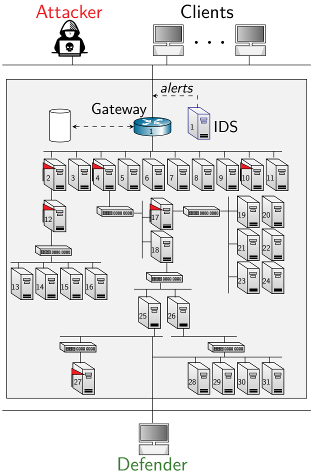
- **Topology**:
  - 30 Application Servers, 1 Gateway/IDS (Snort), 3 Clients, 1 Attacker, 1 Defender
- **Services**
  - 31 SSH, 8 HTTP, 1 DNS, 1 Telnet, 2 FTP, 1 MongoDB, 2 SMTP, 2 Teamspeak 3, 22 SNMP, 12 IRC, 1 Elasticsearch, 12 NTP, 1 Samba, 19 PostgreSQL
- **RCE Vulnerabilities**
  - 1 CVE-2010-0426, 1 CVE-2014-6271, 1 SQL Injection, 1 CVE-2015-3306, 1 CVE-2016-10033, 1 CVE-2015-5602, 1 CVE-2015-1427, 1 CVE-2017-7494
  - 5 Brute-force vulnerabilities
- **Operating Systems**
  - 23 Ubuntu-20, 1 Debian 9:2, 1 Debian Wheezy, 6 Debian Jessie, 1 Kali
- **Traffic**
  - Client 1: HTTP, SSH, SNMP, ICMP
  - Client 2: IRC, PostgreSQL, SNMP
  - Client 3: FTP, DNS, Telnet



Target infrastructure.

The Attacker Model: Capture the Flag (CTF)

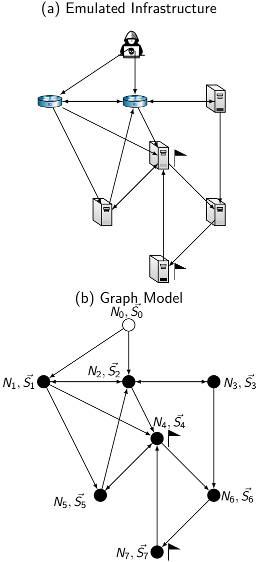
- The attacker has T time-steps to collect flags, with no prior knowledge
- It can **connect to a gateway** that exposes public-facing services in the infrastructure.
- It has a **pre-defined set (cardinality** ~ 200**) of network/shell commands available**, each command has a cost
- To collect flags, it has to interleave reconnaissance and exploits.
- Objective: collect all flags with minimum cost



Target infrastructure.

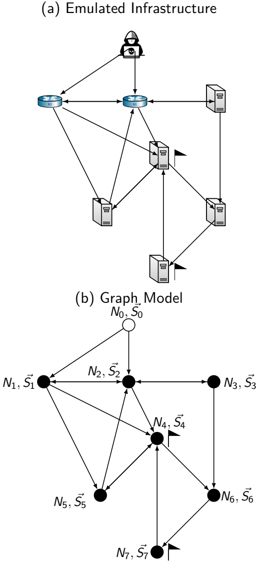
The Formal Attacker Model: A Partially Observed MDP

- Model infrastructure as a graph $\mathcal{G} = \langle \mathcal{N}, \mathcal{E} \rangle$
- There are k flags at nodes $\mathcal{C} \subseteq \mathcal{N}$
- $N_i \in \mathcal{N}$ has a *node state* s_i of m attributes
- Network state
  $s = \{s_A, s_i \mid i \in \mathcal{N}\} \in \mathbb{R}^{|\mathcal{N}| \times m + |\mathcal{N}|}$
- Attacker observes $o^A \subset s$ (results of commands)
- Action space: $\mathcal{A} = \{a_1^A, \ldots, a_k^A\}$, a_i^A (commands)
- $\forall (s, a) \in \mathcal{A} \times \mathcal{S}$, there is a probability $\vec{w}_{i,j}^{A,(x)}$ of failure & a probability of detection $\varphi(det(s_i) \cdot n_{i,j}^{A,(x)})$
- State transitions $s \to s'$ are decided by a discrete dynamical system $s' = F(s, a)$

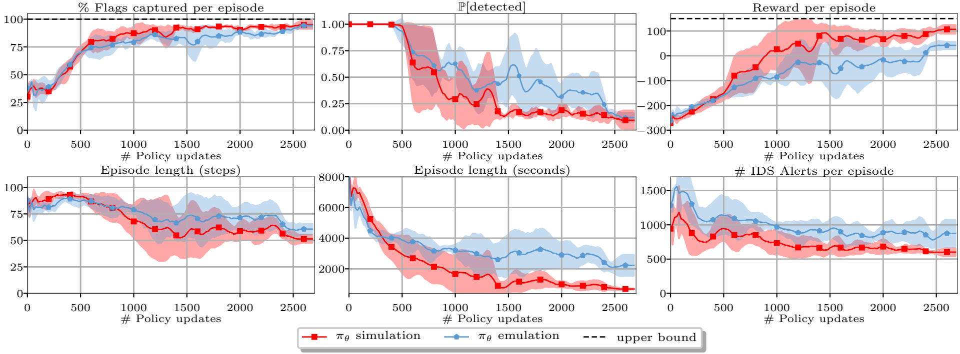


The Formal Attacker Model: A Partially Observed MDP

- Model infrastructure as a graph $\mathcal{G} = \langle \mathcal{N}, \mathcal{E} \rangle$
- There are k flags at nodes $\mathcal{C} \subseteq \mathcal{N}$
- $N_i \in \mathcal{N}$ has a *node state* s_i of m attributes
- Network state $s = \{s_A, s_i \mid i \in \mathcal{N}\} \in \mathbb{R}^{|\mathcal{N}| \times m + |\mathcal{N}|}$
- Attacker observes $o^A \subset s$ (results of commands)
- Action space: $\mathcal{A} = \{a_1^A, \dots, a_k^A\}$, a_i^A (commands)
- $\forall (s, a) \in \mathcal{A} \times \mathcal{S}$, there is a probability $\vec{w}_{i,j}^{A,(x)}$ of failure & a probability of detection $\varphi(det(s_i) \cdot n_{i,j}^{A,(x)})$
- State transitions $s \to s'$ are decided by a discrete dynamical system $s' = F(s, a)$

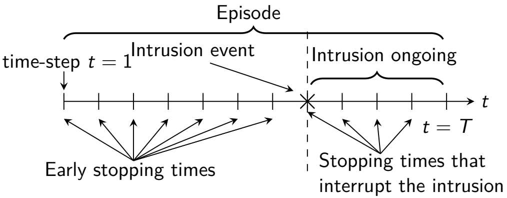


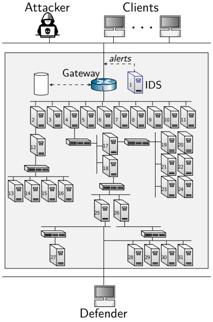
Learning to Capture the Flags: Training Attacker Policies



Learning curves (training performance in simulation and evaluation performance in the emulation) of our proposed method.

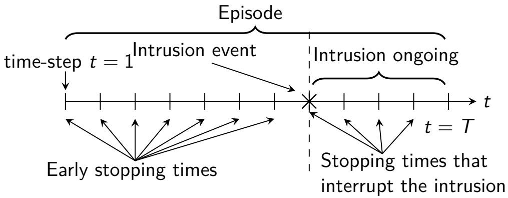
Learning Security Policies through Optimal Stopping

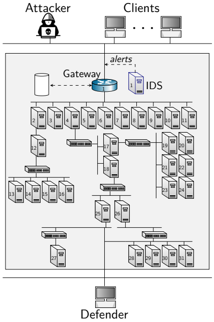




- **Intrusion Prevention as Optimal Stopping Problem**:
  - Defender observes the infrastructure (IDS, log files, etc.).
  - An intrusion occurs at an unknown time.
  - The defender can "stop" the intrusion.
  - *Stopping shuts down the service provided by the infrastructure.*
  - $\implies$ trade-off two objectives: service and security
  - **Based on the observations, when is it optimal to stop?**

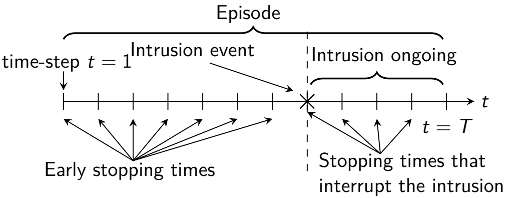
Learning Security Policies through Optimal Stopping

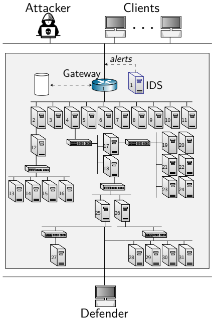




- **Intrusion Prevention as Optimal Stopping Problem**:
  - Defender observes the infrastructure (IDS, log files, etc.).
  - An intrusion occurs at an unknown time.
  - The defender can "stop" the intrusion.
  - *Stopping shuts down the service provided by the infrastructure.*
  - $\implies$ trade-off two objectives: service and security
  - **Based on the observations, when is it optimal to stop?**

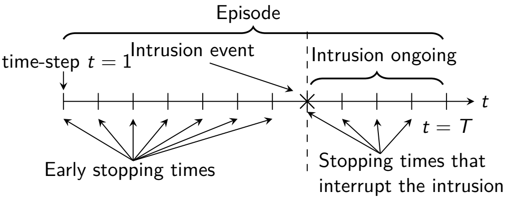
Learning Security Policies through Optimal Stopping

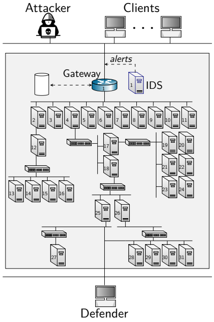




- **Intrusion Prevention as Optimal Stopping Problem**:
  - Defender observes the infrastructure (IDS, log files, etc.).
  - An intrusion occurs at an unknown time.
  - The defender can "stop" the intrusion.
  - *Stopping shuts down the service provided by the infrastructure.*
  - $\implies$ trade-off two objectives: service and security
  - **Based on the observations, when is it optimal to stop?**

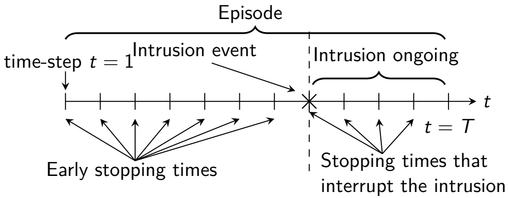
Learning Security Policies through Optimal Stopping

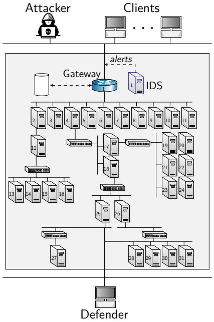




- **Intrusion Prevention as Optimal Stopping Problem**:
  - Defender observes the infrastructure (IDS, log files, etc.).
  - An intrusion occurs at an unknown time.
  - The defender can "stop" the intrusion.
  - *Stopping shuts down the service provided by the infrastructure.*
  - $\implies$ trade-off two objectives: service and security
  - **Based on the observations, when is it optimal to stop?**

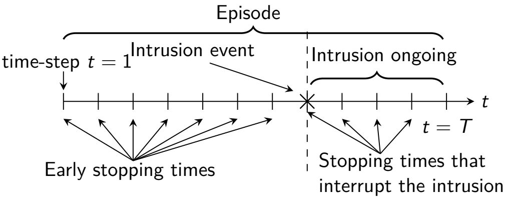
Learning Security Policies through Optimal Stopping

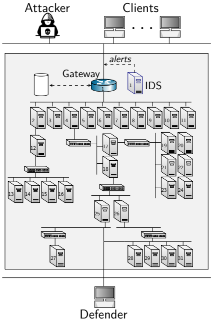




- **Intrusion Prevention as Optimal Stopping Problem**:
  - Defender observes the infrastructure (IDS, log files, etc.).
  - An intrusion occurs at an unknown time.
  - The defender can "stop" the intrusion.
  - *Stopping shuts down the service provided by the infrastructure.*
  - $\Longrightarrow$ trade-off two objectives: service and security
  - **Based on the observations, when is it optimal to stop?**

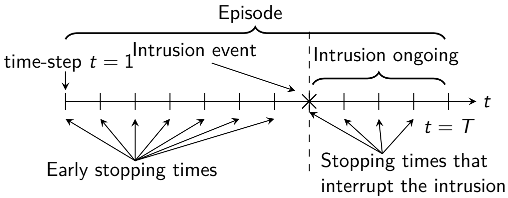
Learning Security Policies through Optimal Stopping

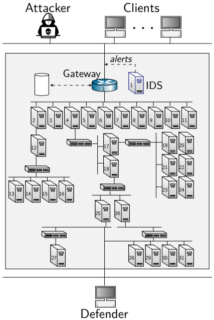




- **Intrusion Prevention as Optimal Stopping Problem**:
  - Defender observes the infrastructure (IDS, log files, etc.).
  - An intrusion occurs at an unknown time.
  - The defender can "stop" the intrusion.
  - *Stopping shuts down the service provided by the infrastructure.*
  - $\Longrightarrow$ trade-off two objectives: service and security
  - **Based on the observations, when is it optimal to stop?**

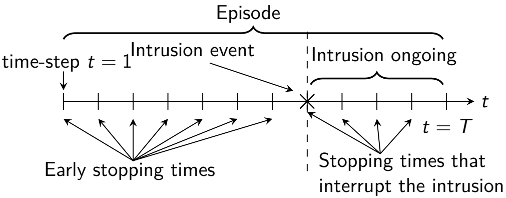
Learning Security Policies through Optimal Stopping

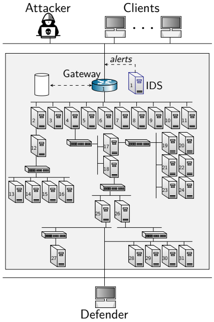




- **Intrusion Prevention as Optimal Stopping Problem**:
  - Defender observes the infrastructure (IDS, log files, etc.).
  - An intrusion occurs at an unknown time.
  - The defender can "stop" the intrusion.
  - *Stopping shuts down the service provided by the infrastructure.*
  - $\Longrightarrow$ trade-off two objectives: service and security
  - **Based on the observations, when is it optimal to stop?**

Learning Security Policies through Optimal Stopping

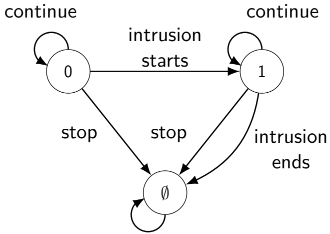




- **Intrusion Prevention as Optimal Stopping Problem**:
  - Defender observes the infrastructure (IDS, log files, etc.).
  - An intrusion occurs at an unknown time.
  - The defender can "stop" the intrusion.
  - *Stopping shuts down the service provided by the infrastructure*.
  - $\implies$ trade-off two objectives: service and security
  - **Based on the observations, when is it optimal to stop?**

A Partially Observed MDP Model for the Defender

- **States:**
  - Intrusion state $i_t \in \{0, 1\}$, terminal state $\emptyset$.

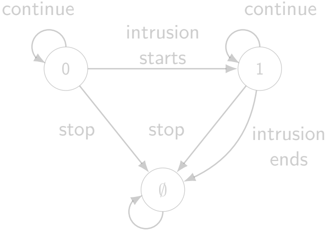


- **Observations:**
  - Severe/Warning IDS Alerts $(\Delta x, \Delta y)$, Login attempts Δz. $f_{XYZ}(\Delta x, \Delta y, \Delta z | i_t, I_t, t)$
- **Actions:**
  - "Stop" (S) and "Continue" (C)
- **Rewards:**
  - Reward: security and service. Penalty: false alarms and intrusions

- **Transition probabilities:**
  - Bernoulli process $(Q_t)_{t=1}^{T} \sim Ber(p)$ defines intrusion start $I_t \sim Ge(p)$
- **Objective and Horizon:**
  - $\max \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T_\emptyset} r(s_t, a_t)\right]$, $T_\emptyset$

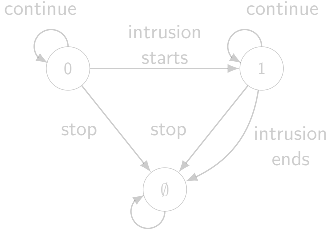
A Partially Observed MDP Model for the Defender

- **States:**
  - Intrusion state $i_t \in \{0, 1\}$, terminal state $\emptyset$.
- **Observations:**
  - Severe/Warning IDS Alerts $(\Delta x, \Delta y)$, Login attempts Δz. $f_{XYZ}(\Delta x, \Delta y, \Delta z | i_t, I_t, t)$
- **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:**
  - $\max \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T_\emptyset} r(s_t, a_t)\right]$, $T_\emptyset$



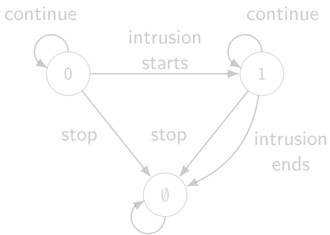
A Partially Observed MDP Model for the Defender

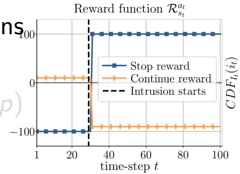
- **States:**
  - Intrusion state $i_t \in \{0, 1\}$, terminal state $\emptyset$.
- **Observations:**
  - Severe/Warning IDS Alerts $(\Delta x, \Delta y)$, Login attempts Δz. $f_{XYZ}(\Delta x, \Delta y, \Delta z | i_t, I_t, t)$
- **Actions:**
  - "Stop" (S) and "Continue" (C)
- **Rewards:**
  - Reward: 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:**
  - $\max \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T_\emptyset} r(s_t, a_t)\right]$, $T_\emptyset$

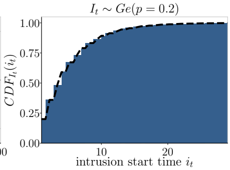


A Partially Observed MDP Model for the Defender

- **States:**
  - Intrusion state $i_t \in \{0, 1\}$, terminal state $\emptyset$.
- **Observations:**
  - Severe/Warning IDS Alerts $(\Delta x, \Delta y)$, Login attempts Δz. $f_{XYZ}(\Delta x, \Delta y, \Delta z | i_t, I_t, t)$
- **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:**
  - $\max \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T_\emptyset} r(s_t, a_t)\right]$, $T_\emptyset$

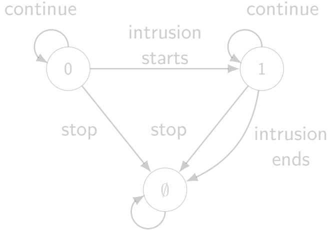


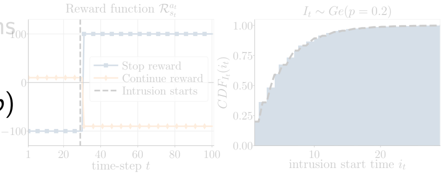




A Partially Observed MDP Model for the Defender

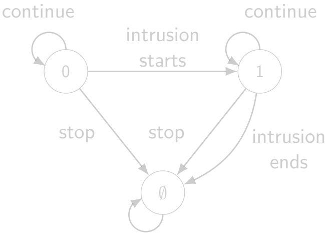
- **States:**
  - Intrusion state $i_t \in \{0,1\}$, terminal state $\emptyset$.
- **Observations:**
  - Severe/Warning IDS Alerts $(\Delta x, \Delta y)$, Login attempts Δz. $f_{XYZ}(\Delta x, \Delta y, \Delta z | i_t, I_t, t)$
- **Actions:**
  - "Stop" (S) and "Continue" (C)
- **Rewards:**
  - Reward: 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:**
  - $\max \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T_\emptyset} r(s_t, a_t)\right]$, $T_\emptyset$





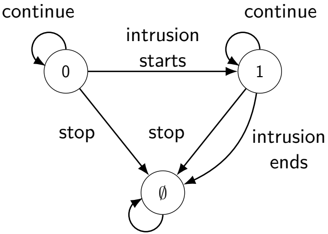
A Partially Observed MDP Model for the Defender

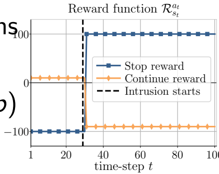
- **States:**
  - Intrusion state $i_t \in \{0, 1\}$, terminal state $\emptyset$.
- **Observations:**
  - Severe/Warning IDS Alerts $(\Delta x, \Delta y)$, Login attempts Δz. $f_{XYZ}(\Delta x, \Delta y, \Delta z | i_t, I_t, t)$
- **Actions:**
  - "Stop" (S) and "Continue" (C)
- **Rewards:**
  - Reward: 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:**
  - $\max \mathbb{E}_{\pi_\theta} \left[\sum_{t=1}^{T_\emptyset} r(s_t, a_t) \right], \ T_\emptyset$

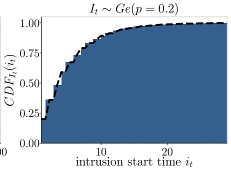


A Partially Observed MDP Model for the Defender

- **States:**
  - Intrusion state $i_t \in \{0, 1\}$, terminal state $\emptyset$.
- **Observations:**
  - Severe/Warning IDS Alerts $(\Delta x, \Delta y)$, Login attempts Δz. $f_{XYZ}(\Delta x, \Delta y, \Delta z | i_t, I_t, t)$
- **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:**
  - $\max \mathbb{E}_{\pi_\theta} \left[\sum_{t=1}^{T_\emptyset} r(s_t, a_t) \right]$, $T_\emptyset$







Threshold Property of the Optimal Defender Policy (1/4)

Theorem

The optimal policy π^* is a threshold policy of the form:

$$\pi^*(b(1)) = \begin{cases} S \ (stop) & \text{if } b(1) \geq \alpha^* \\ C \ (continue) & \text{otherwise} \end{cases}$$

where α^* is a unique threshold and $b(1) = \mathbb{P}[s_t = 1 | a_1, o_1, \ldots, a_{t-1}, o_t]$.

- To see this, consider the **optimality condition** (Bellman eq):

$$\pi^*(b(1)) = \arg\max_{a \in \mathcal{A}} \left[r(b(1), a) + \sum_{o \in \mathcal{O}} \mathbb{P}[o | b(1), a] V^*(b_o^a(1)) \right]$$

Threshold Property of the Optimal Defender Policy (1/4)

Theorem

The optimal policy π^* is a threshold policy of the form:

$$\pi^*(b(1)) = \begin{cases} S \text{ (stop)} & \text{if } b(1) \geq \alpha^* \\ C \text{ (continue)} & \text{otherwise} \end{cases}$$

where α^* is a unique threshold and
$b(1) = \mathbb{P}[s_t = 1 | a_1, o_1, \ldots, a_{t-1}, o_t]$.

- To see this, consider the **optimality condition** (Bellman eq):

$$\pi^*\big(b(1)\big) = \arg\max_{a \in \mathcal{A}} \left[r\big(b(1), a\big) + \sum_{o \in \mathcal{O}} \mathbb{P}[o | b(1), a] V^*\big(b_o^a(1)\big) \right]$$

Threshold Property of the Optimal Defender Policy (2/4)

- We use $\mathcal{A} = \{S, C\}$ and derive:

$$\pi^*(b(1)) = \operatorname*{argmax}_{a \in \mathcal{A}} \Bigg[\underbrace{r\big(b(1), S\big)}_{\omega}, \underbrace{r\big(b(1), C\big) + \sum_{o \in \mathcal{O}} \mathbb{P}[o|b(1), C] V^*\big(b_o^C(1)\big)}_{\epsilon} \Bigg]$$

- ω is the expected reward for stopping and ϵ is the expected cumulative reward for continuing
- Expanding the expressions and rearranging terms, we derive that **it is optimal to stop iff**:

$$b(1) \geq \underbrace{\frac{110 + \sum_{o \in \mathcal{O}} V^*\left(b_o^C(1)\right)\left(p\mathcal{Z}(o,1,C) + (1-p)\mathcal{Z}(o,0,C)\right)}{300 + \sum_{o \in \mathcal{O}} V^*\left(b_o^C(1)\right)\left(p\mathcal{Z}(o,1,C) + (1-p)\mathcal{Z}(o,0,C) - \mathcal{Z}(o,1,C)\right)}}_{\text{Threshold: } \alpha_{b(1)}}$$

Threshold Property of the Optimal Defender Policy (2/4)

- We use $\mathcal{A} = \{S, C\}$ and derive:

$$\pi^*(b(1)) = \operatorname*{argmax}_{a \in \mathcal{A}} \Big[\underbrace{r\big(b(1), S\big)}_{\omega}, \underbrace{r\big(b(1), C\big) + \sum_{o \in \mathcal{O}} \mathbb{P}[o|b(1), C] V^*\big(b_o^C(1)\big)}_{\epsilon} \Big]$$

- ω is the expected reward for stopping and ϵ is the expected cumulative reward for continuing
- Expanding the expressions and rearranging terms, we derive that **it is optimal to stop iff**:

$$b(1) \geq \underbrace{\frac{110 + \sum_{o\in\mathcal{O}} V^*\Big(b_o^C(1)\Big)\Big(p\mathcal{Z}(o,1,C) + (1-p)\mathcal{Z}(o,0,C)\Big)}{300 + \sum\limits_{o\in\mathcal{O}} V^*\Big(b_o^C(1)\Big)\Big(p\mathcal{Z}(o,1,C) + (1-p)\mathcal{Z}(o,0,C) - \mathcal{Z}(o,1,C)\Big)}}_{\text{Threshold: } \alpha_{b(1)}}$$

Threshold Property of the Optimal Defender Policy (3/4)

- Thus π^* is determined by the **scalar thresholds** $\alpha_{b(1)}$.
  - it is optimal to stop if $b(1) \geq \alpha_{b(1)}$
- The *stopping set* is:

$$\mathscr{S} = \left\{ b(1) \in [0,1] : b(1) \geq \alpha_{b(1)} \right\}$$

- Since $V^*(b)$ **is piecewise linear and convex**[13]
- When $b(1) = 1$ it is optimal to take the stop action S:

$$\pi^*(1) = \arg\max \left[100, -90 + \sum_{o \in \mathcal{O}} \mathcal{Z}(o,1,C) V^*\left(b_o^C(1)\right) \right] = S$$

- This means that $\beta^* = 1$

[13] Edward J. Sondik. "The Optimal Control of Partially Observable Markov Processes Over the Infinite Horizon: Discounted Costs". In: *Operations Research* 26.2 (1978), pp. 282–304. ISSN: 0030364X, 15265463. URL: http://www.jstor.org/stable/169635.

Threshold Property of the Optimal Policy (3/4)

- Thus π^* is determined by the **scalar thresholds** $\alpha_{b(1)}$.
  - it is optimal to stop if $b(1) \geq \alpha_{b(1)}$
- The *stopping set* is:

$$\mathscr{S} = \Big\{ b(1) \in [0,1] : b(1) \geq \alpha_{b(1)} \Big\}$$

- Since $V^*(b)$ **is piecewise linear and convex**[14]
- When $b(1) = 1$ it is optimal to take the stop action S:

$$\pi^*(1) = \arg\max \left[100, -90 + \sum_{o \in \mathcal{O}} \mathcal{Z}(o,1,C) V^*\big(b_o^C(1)\big) \right] = S$$

- This means that $\beta^* = 1$

[14] Edward J. Sondik. "The Optimal Control of Partially Observable Markov Processes Over the Infinite Horizon: Discounted Costs". In: *Operations Research* 26.2 (1978), pp. 282–304. ISSN: 0030364X, 15265463. URL: http://www.jstor.org/stable/169635.

Threshold Property of the Optimal Defender Policy (3/4)

- Thus π^* is determined by the **scalar thresholds** $\alpha_{b(1)}$.
  - it is optimal to stop if $b(1) \geq \alpha_{b(1)}$
- The *stopping set* is:

$$\mathscr{S} = \left\{ b(1) \in [0,1] : b(1) \geq \alpha_{b(1)} \right\}$$

- Since $V^*(b)$ **is piecewise linear and convex**[15]
- When $b(1) = 1$ it is optimal to take the stop action S:

$$\pi^*(1) = \arg\max \left[100, -90 + \sum_{o \in \mathcal{O}} \mathcal{Z}(o, 1, C) V^*\left(b_o^C(1)\right) \right] = S$$

- This means that $\beta^* = 1$

[15] Edward J. Sondik. "The Optimal Control of Partially Observable Markov Processes Over the Infinite Horizon: Discounted Costs". In: *Operations Research* 26.2 (1978), pp. 282–304. ISSN: 0030364X, 15265463. URL: http://www.jstor.org/stable/169635.

Threshold Property of the Optimal Defender Policy (3/4)

- Thus π^* is determined by the **scalar thresholds** $\alpha_{b(1)}$.
  - it is optimal to stop if $b(1) \geq \alpha_{b(1)}$
- The *stopping set* is:

$$\mathscr{S} = \left\{ b(1) \in [0,1] : b(1) \geq \alpha_{b(1)} \right\}$$

- Since $V^*(b)$ **is piecewise linear and convex**[16], $\mathscr{S}$ is also convex[17] and has the form $[\alpha^*, \beta^*]$ where $0 \leq \alpha^* \leq \beta^* \leq 1$.
- When $b(1) = 1$ it is optimal to take the stop action S:

$$\pi^*(1) = \arg\max \left[100, -90 + \sum_{o \in \mathcal{O}} \mathcal{Z}(o, 1, C) V^*\left(b_o^C(1)\right) \right] = S$$

- This means that $\beta^* = 1$

[16] Edward J. Sondik. "The Optimal Control of Partially Observable Markov Processes Over the Infinite Horizon: Discounted Costs". In: *Operations Research* 26.2 (1978), pp. 282–304. ISSN: 0030364X, 15265463. URL: http://www.jstor.org/stable/169635.

[17] Vikram Krishnamurthy. *Partially Observed Markov Decision Processes: From Filtering to Controlled Sensing*. Cambridge University Press, 2016. DOI: 10.1017/CBO9781316471104.

Threshold Property of the Optimal Defender Policy (3/4)

- Thus π^* is determined by the **scalar thresholds** $\alpha_{b(1)}$.
  - it is optimal to stop if $b(1) \geq \alpha_{b(1)}$
- The *stopping set* is:

$$\mathscr{S} = \left\{ b(1) \in [0,1] : b(1) \geq \alpha_{b(1)} \right\}$$

- Since $V^*(b)$ **is piecewise linear and convex**[18], $\mathscr{S}$ is also convex[19] and has the form $[\alpha^*, \beta^*]$ where $0 \leq \alpha^* \leq \beta^* \leq 1$.
- When $b(1) = 1$ it is optimal to take the stop action S:

$$\pi^*(1) = \arg\max \left[100, -90 + \sum_{o \in \mathcal{O}} \mathcal{Z}(o, 1, C) V^*\left(b_o^C(1)\right) \right] = S$$

- This means that $\beta^* = 1$

[18] Edward J. Sondik. "The Optimal Control of Partially Observable Markov Processes Over the Infinite Horizon: Discounted Costs". In: *Operations Research* 26.2 (1978), pp. 282–304. ISSN: 0030364X, 15265463. URL: http://www.jstor.org/stable/169635.

[19] Vikram Krishnamurthy. *Partially Observed Markov Decision Processes: From Filtering to Controlled Sensing*. Cambridge University Press, 2016. DOI: 10.1017/CBO9781316471104.

Threshold Property of the Optimal Defender Policy (4/4)

- As the stopping set is $\mathscr{S} = [\alpha^*, 1]$ and $b(1) \in [0, 1]$
- We have that it is optimal to stop if $b(1) \geq \alpha^*$
- Hence, **Theorem 1** follows:

$$\pi^*(b(1)) = \begin{cases} S \text{ (stop)} & \text{if } b(1) \geq \alpha^* \\ C \text{ (continue)} & \text{otherwise} \end{cases}$$

Threshold Property of the Optimal Defender Policy (4/4)

- As the stopping set is $\mathscr{S} = [\alpha^*, 1]$ and $b(1) \in [0, 1]$
- We have that it is optimal to stop if $b(1) \geq \alpha^*$

- Hence, **Theorem 1** follows:

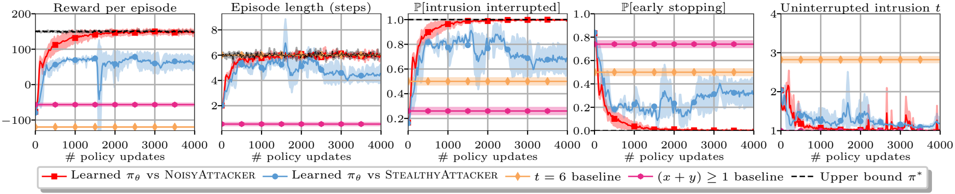
$$\pi^*(b(1)) = \begin{cases} S \text{ (stop)} & \text{if } b(1) \geq \alpha^* \\ C \text{ (continue)} & \text{otherwise} \end{cases}$$

Static Attackers to Emulate Intrusions

Time-steps t	*Actions*
1–$I_t \sim Ge(0.2)$	(Intrusion has not started)
$I_t + 1$–$I_t + 7$	RECON, brute-force attacks (SSH,Telnet,FTP) on N_2, N_4, N_{10}, login(N_2, N_4, N_{10}), backdoor(N_2, N_4, N_{10}), RECON
$I_t + 8$–$I_t + 11$	CVE-2014-6271 on N_{17}, SSH brute-force attack on N_{12}, login (N_{17}, N_{12}), backdoor(N_{17}, N_{12})
$I_t + 12$–$X + 16$	CVE-2010-0426 exploit on N_{12}, RECON SQL-Injection on N_{18}, login(N_{18}), backdoor(N_{18})
$I_t + 17$–$I_t + 22$	RECON, CVE-2015-1427 on N_{25}, login(N_{25}) RECON, CVE-2017-7494 exploit on N_{27}, login(N_{27})

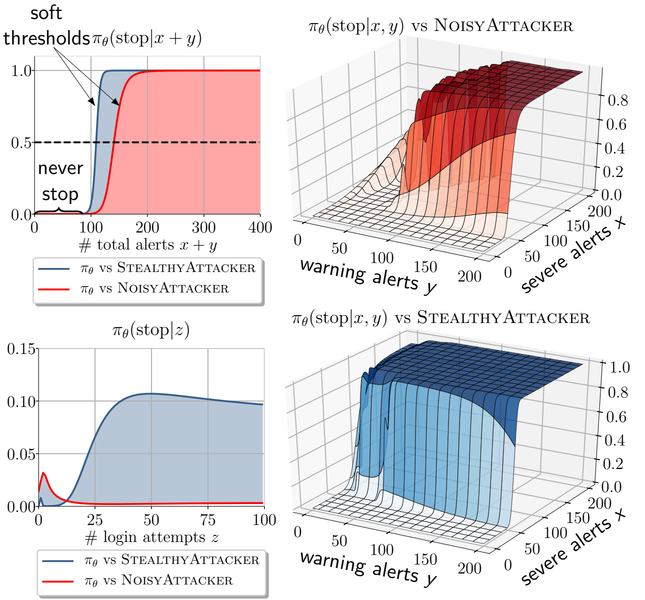
Table 1: Attacker actions to emulate an intrusion.

Learning Security Policies through Optimal Stopping

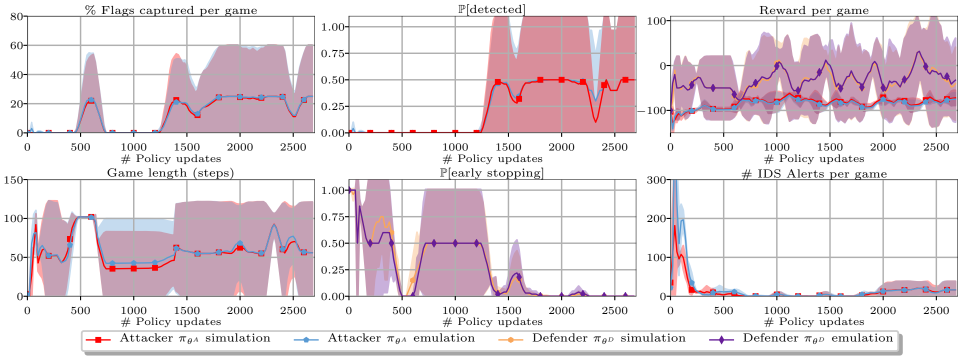


Learning curves of training defender policies against static attackers.

Threshold Properties of the Learned Policies



Open Challenge: Self-Play between Attacker and Defender



Learning curves of training the the attacker and the defender simultaneously in self-play.

Conclusions & Future Work

- **Conclusions:**
  - We develop a *method* to find effective strategies for intrusion prevention
    - (1) emulation system; (2) system identification; (3) simulation system; (4) reinforcement learning and (5) domain randomization and generalization.
  - We show that self-learning can be successfully applied to network infrastructures.
    - Self-play reinforcement learning in Markov security game
  - *Key challenges*: stable convergence, sample efficiency, complexity of emulations, large state and action spaces, theoretical understanding of optimal policies
- **Our research plans:**
  - Extending the theoretical model
    - Relaxing simplifying assumptions (e.g. multiple defender actions)
  - Evaluation on real world infrastructures

References

- *Finding Effective Security Strategies through Reinforcement Learning and Self-Play*[20]
  - **Preprint open access**: https://arxiv.org/abs/2009.08120
- *Learning Intrusion Prevention Policies through Optimal Stopping*[21]
  - **Preprint open access**: https://arxiv.org/pdf/2106.07160.pdf

[20] Kim Hammar and Rolf Stadler. "Finding Effective Security Strategies through Reinforcement Learning and Self-Play". In: *International Conference on Network and Service Management (CNSM)*. Izmir, Turkey, Nov. 2020.

[21] Kim Hammar and Rolf Stadler. *Learning Intrusion Prevention Policies through Optimal Stopping*. 2021. arXiv: 2106.07160 [cs.AI].