Self-Learning Systems for Cyber Security

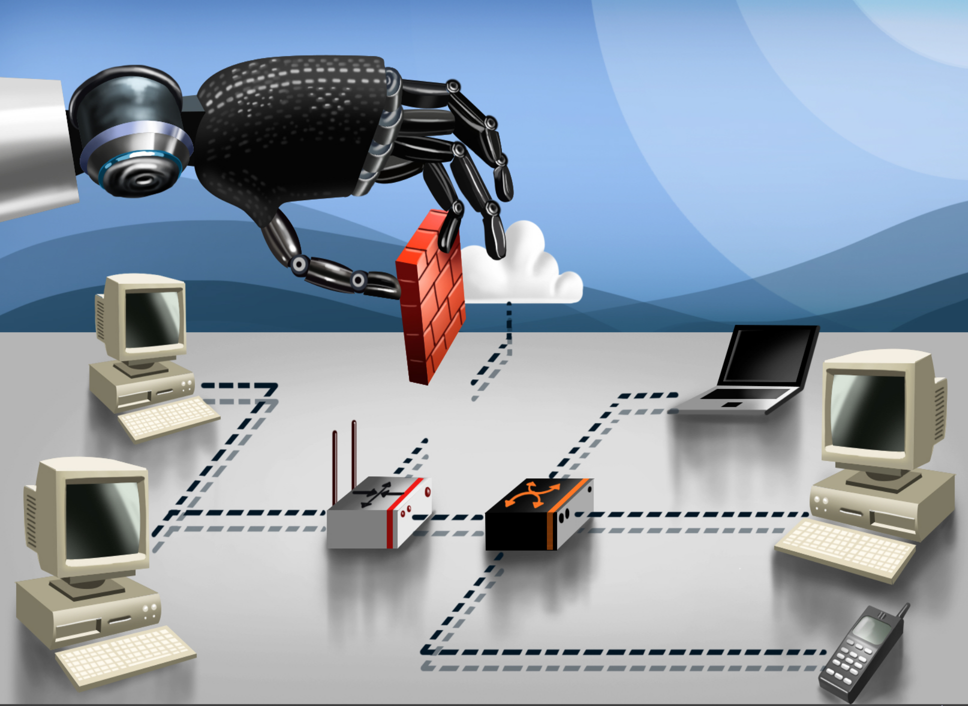
NSE Seminar

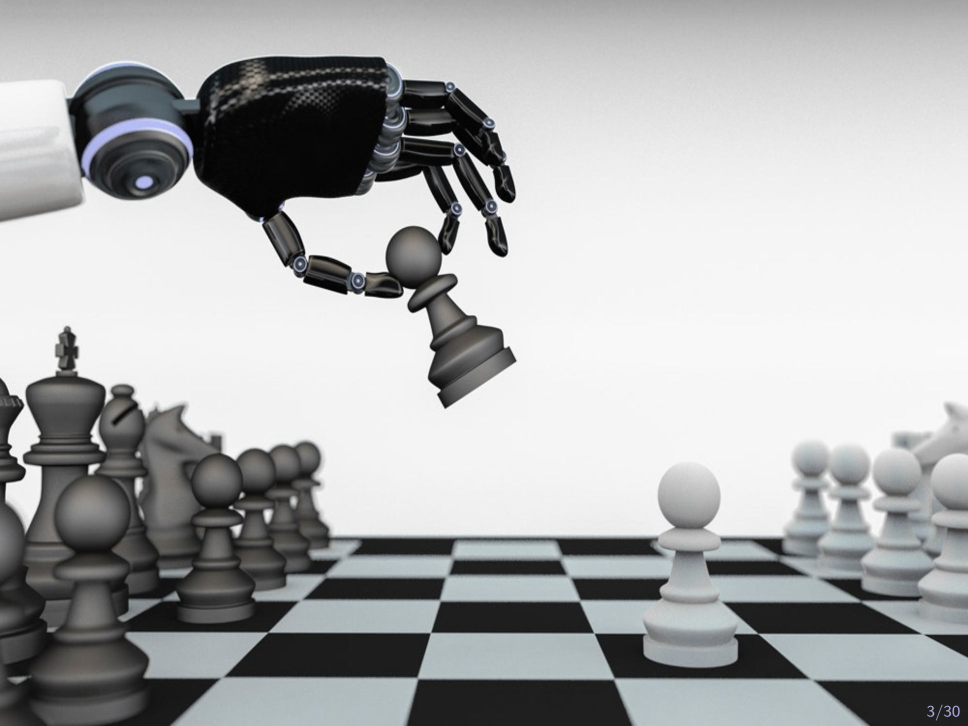
Kim Hammar & Rolf Stadler

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

Division of Network and Systems Engineering
KTH Royal Institute of Technology

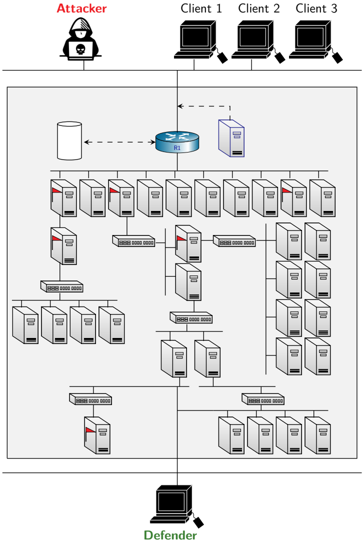
April 9, 2021





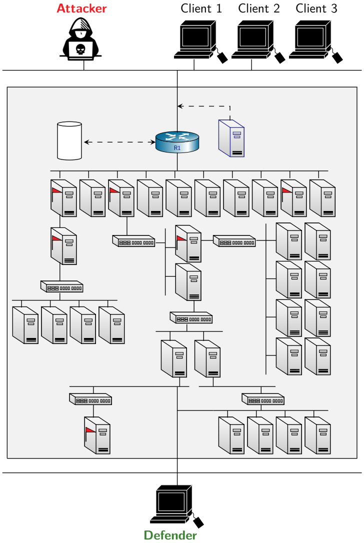
Challenges: Evolving and Automated Attacks

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



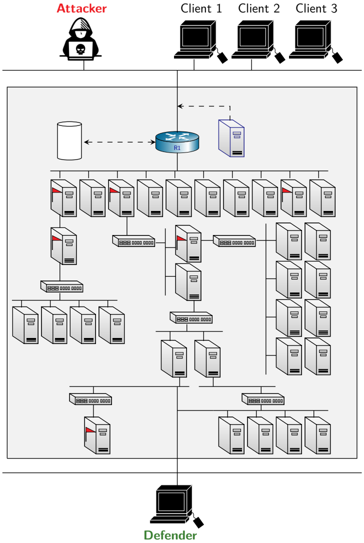
Goal: Automation and Learning

- **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], automated intrusion detection etc.
  - $\implies$ Need for *automation* and better security tooling
- **Mathematical Modeling**:
  - Game theory[3]
  - Markov decision theory
  - $\implies$ 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.

State of the Art

- **Game-Learning Programs**:
  - TD-Gammon, AlphaGo Zero[4], OpenAI Five etc.
  - $\implies$ Impressive empirical results of *RL and self-play*
- **Attack Simulations**:
  - Automated threat modeling[5], automated intrusion detection etc.
  - $\implies$ Need for *automation* and better security tooling
- **Mathematical Modeling**:
  - Game theory[6]
  - Markov decision theory
  - $\implies$ Many security operations involves *strategic decision making*

---

[4] 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.

[5] 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.

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

State of the Art

- **Game-Learning Programs**:
  - TD-Gammon, AlphaGo Zero[7], OpenAI Five etc.
  - $\implies$ Impressive empirical results of *RL and self-play*
- **Attack Simulations**:
  - Automated threat modeling[8], automated intrusion detection etc.
  - $\implies$ Need for *automation* and better security tooling
- **Mathematical Modeling**:
  - Game theory[9]
  - Markov decision theory
  - $\implies$ Many security operations involves *strategic decision making*

---

[7] 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.

[8] 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.

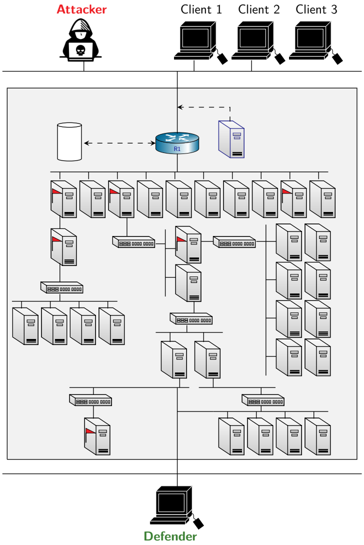
[9] Tansu Alpcan and Tamer Basar. *Network Security: A Decision and Game-Theoretic Approach*. 1st. USA: Cambridge University Press, 2010. ISBN: 0521119324.

Our Work

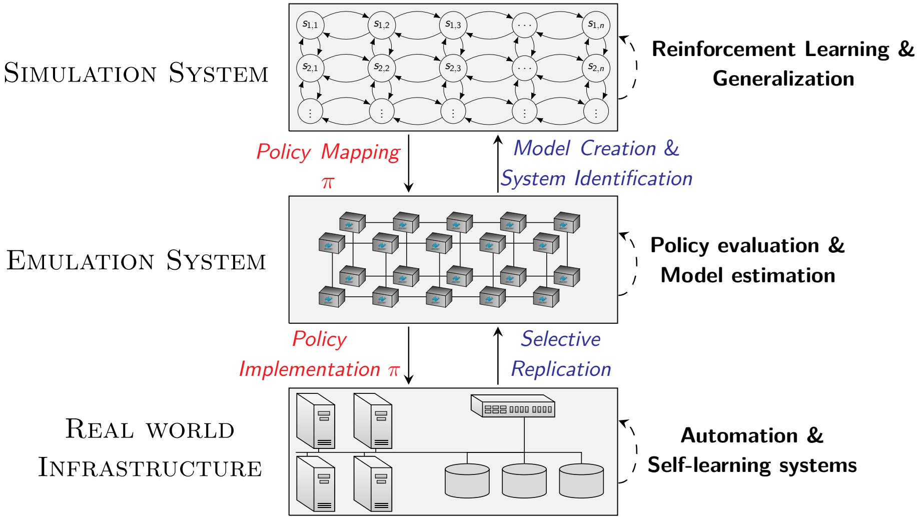
- **Use Case:** Intrusion Prevention
- **Our Method:**
  - Emulating computer infrastructures
  - System identification and model creation
  - Reinforcement learning and generalization
- **Results:**
  - Learning to Capture The Flag
  - Learning to Detect Network Intrusions
- **Conclusions and Future Work**

Use Case: Intrusion Prevention

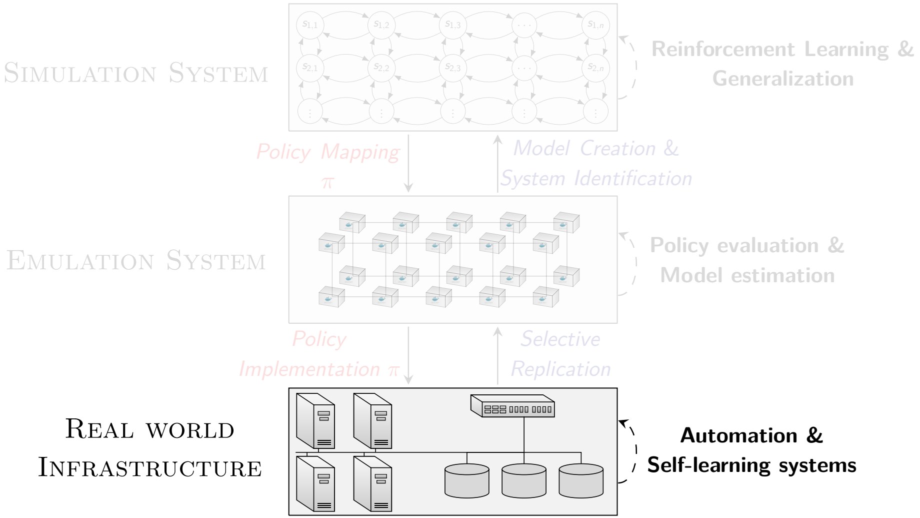
- A **Defender** owns an infrastructure
  - Consists of connected components
  - Components run network services
  - Defender defends the infrastructure by monitoring and patching
- An **Attacker** seeks to intrude on the infrastructure
  - Has a partial view of the infrastructure
  - Wants to compromise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



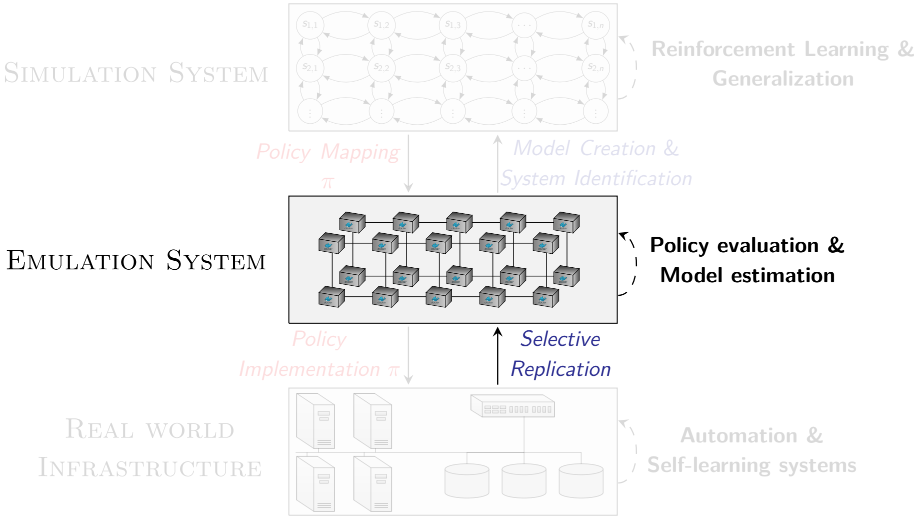
Our Method for Finding Effective Security Strategies



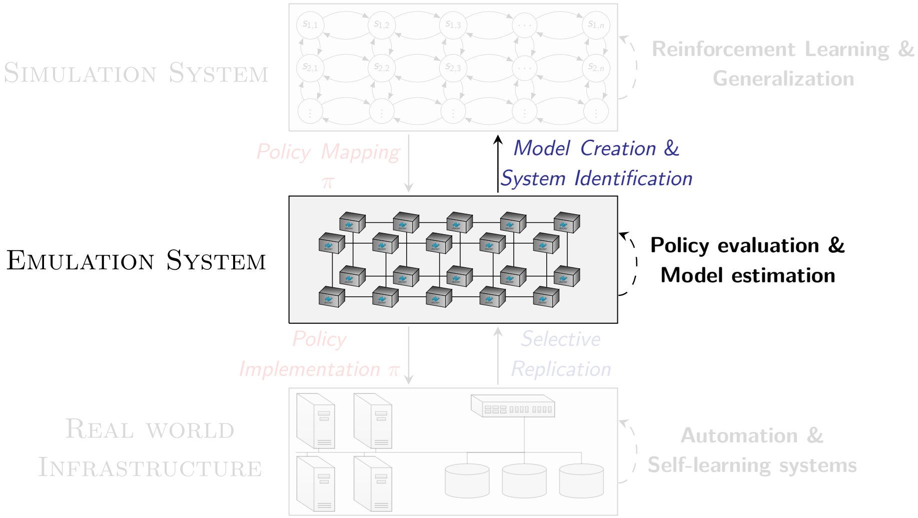
Our Method for Finding Effective Security Strategies



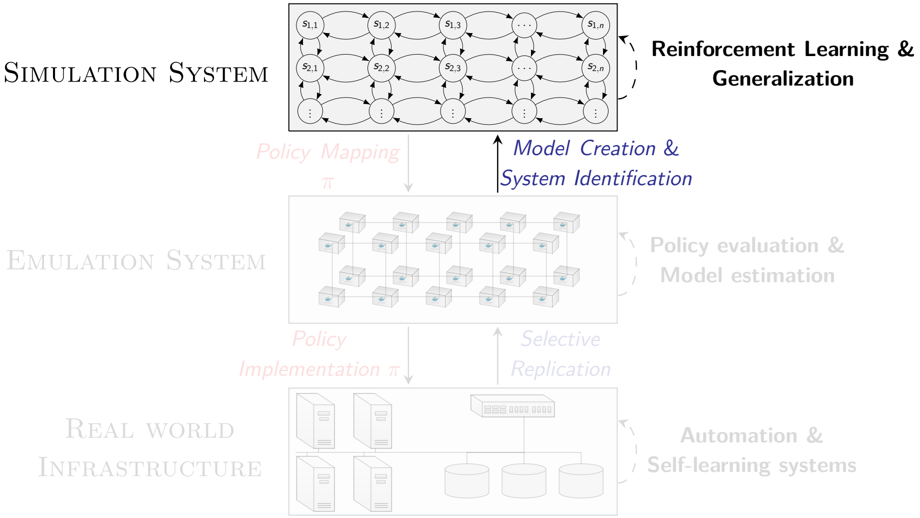
Our Method for Finding Effective Security Strategies



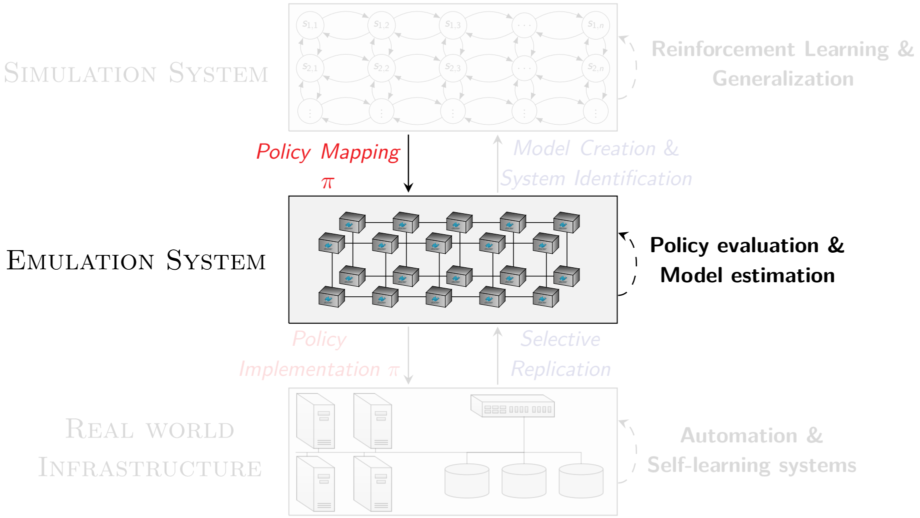
Our Method for Finding Effective Security Strategies



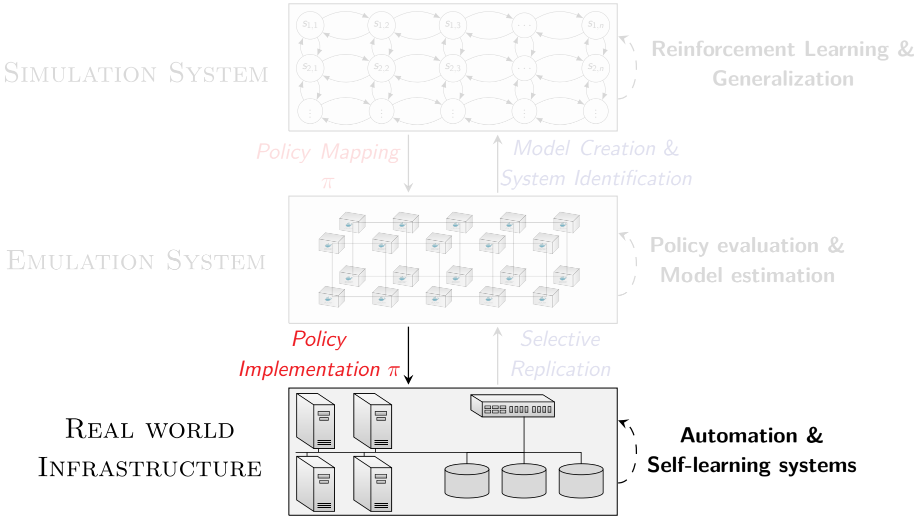
Our Method for Finding Effective Security Strategies



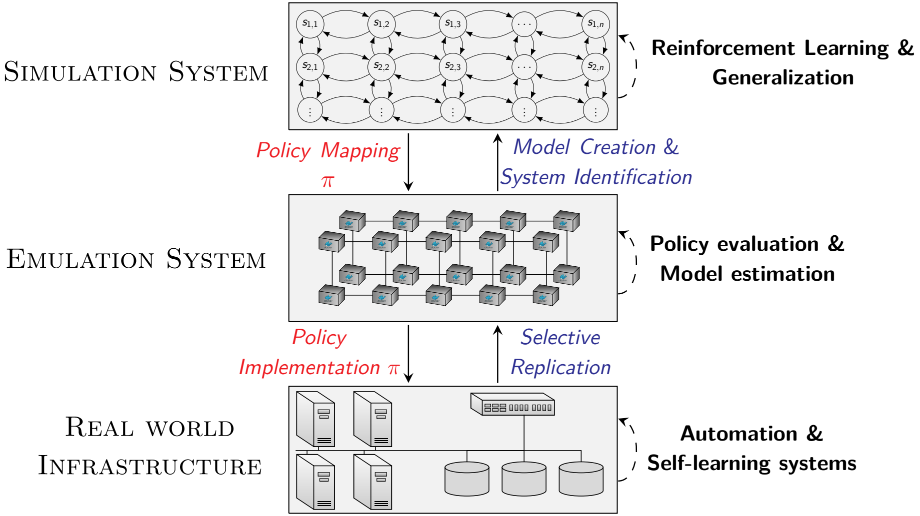
Our Method for Finding Effective Security Strategies



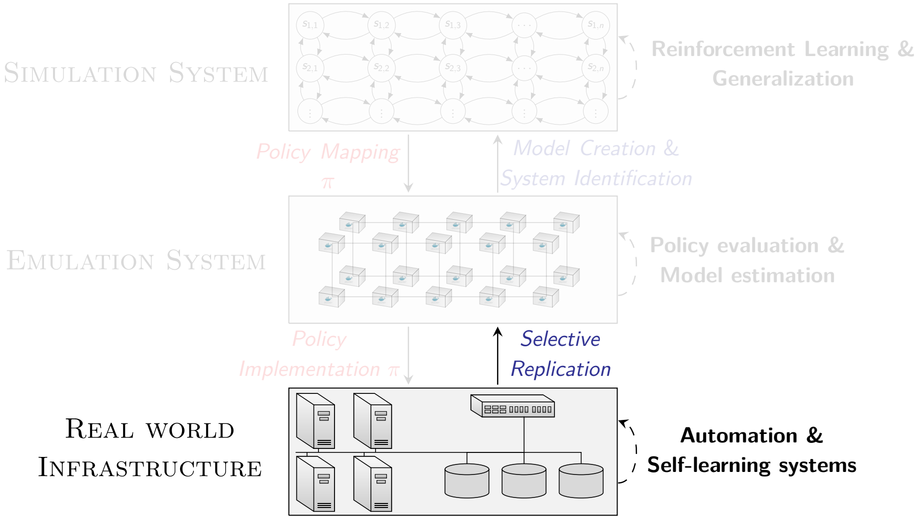
Our Method for Finding Effective Security Strategies



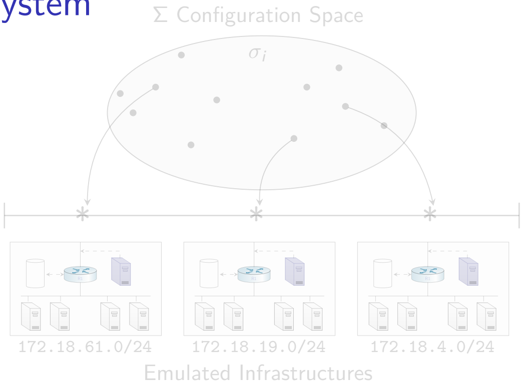
Our Method for Finding Effective Security Strategies



Our Method for Finding Effective Security Strategies



Emulation System

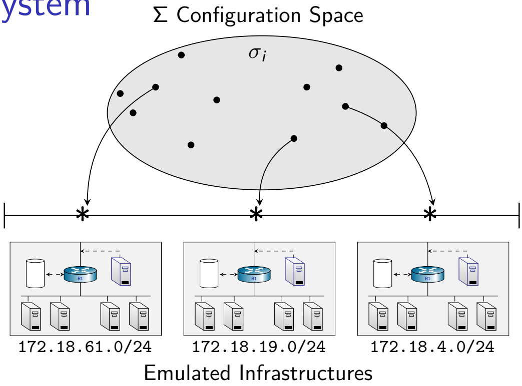


Emulation

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

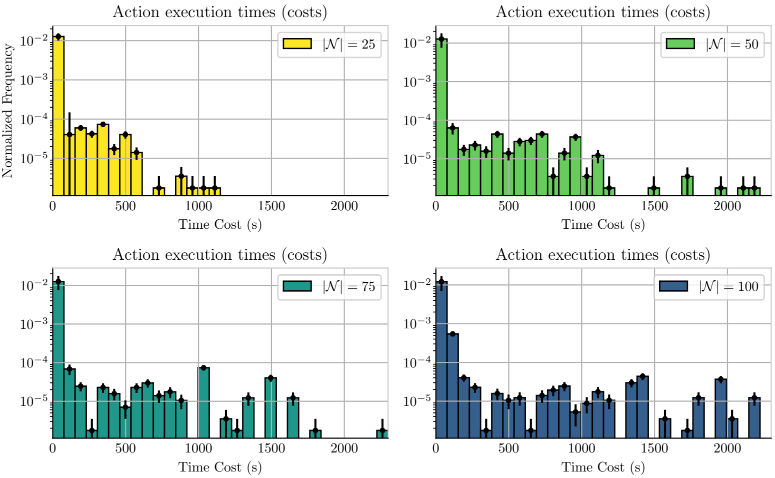


Emulation

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

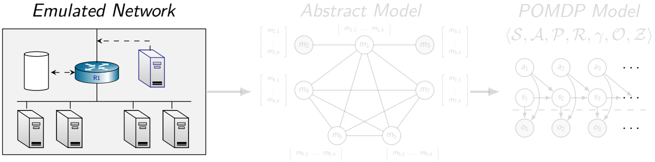
- The set of virtualized configurations define a *configuration space* $\Sigma = \langle \mathcal{A}, \mathcal{O}, \mathcal{S}, \mathcal{U}, \mathcal{T}, \mathcal{V} \rangle$.
- A specific emulation is based on a configuration $\sigma_i \in \Sigma$.

Emulation: Execution Times of Replicated Operations



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

From Emulation to Simulation: System Identification

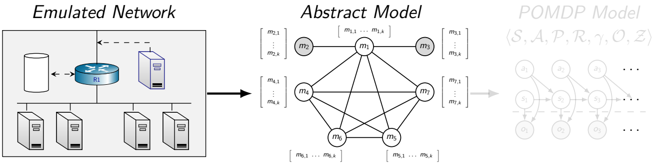


- **Abstract Model Based on Domain Knowledge**: Models the set of *controls*, the *objective function*, and the *features* of the emulated network.
  - Defines the static parts a POMDP model.

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

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

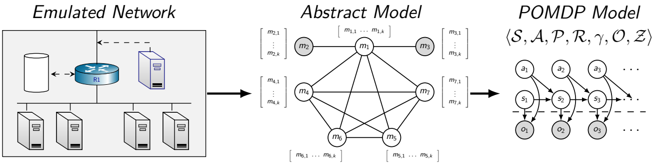
From Emulation to Simulation: System Identification



- **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.

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

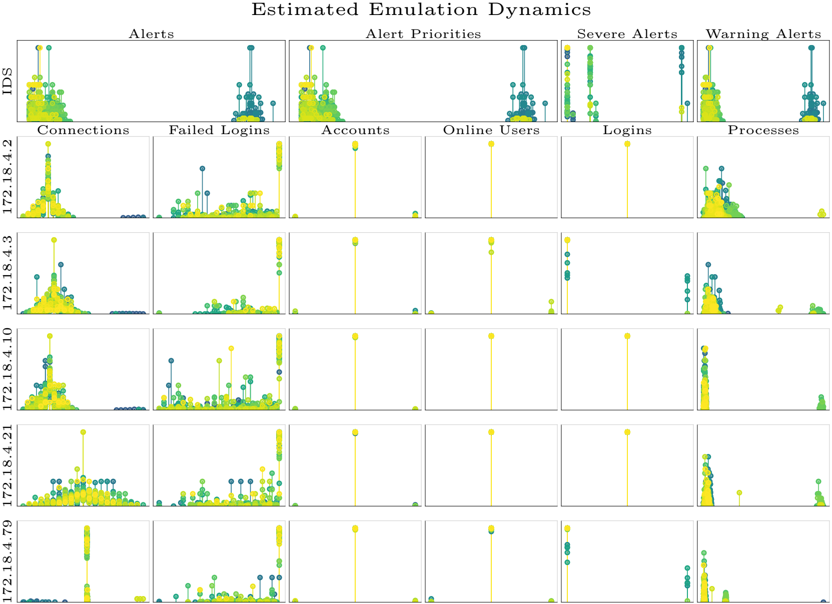
From Emulation to Simulation: System Identification



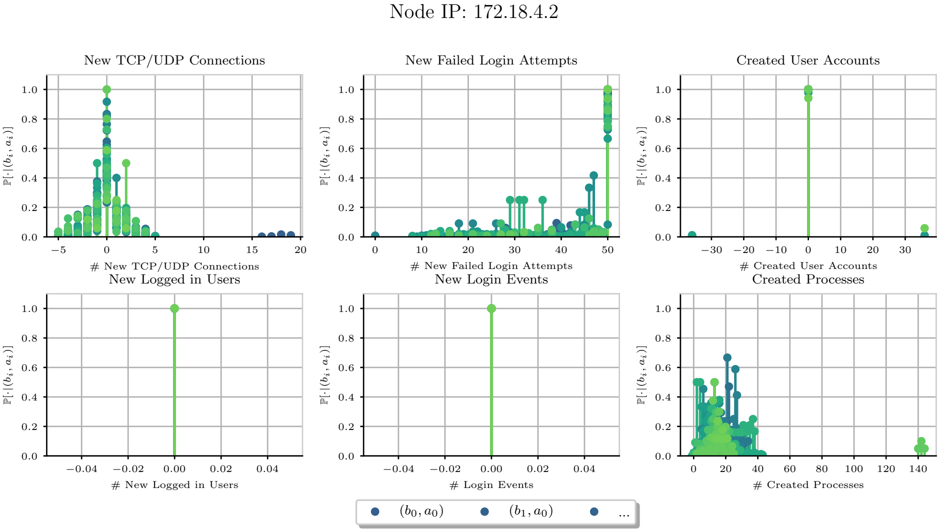
- **Abstract Model Based on Domain Knowledge**: Models the set of *controls*, the *objective function*, and the *features* of the emulated network.
  - Defines the static parts a POMDP model.
- **Dynamics Model ($\mathcal{P}$, $\mathcal{Z}$) Identified using System Identification**: Algorithm based on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

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

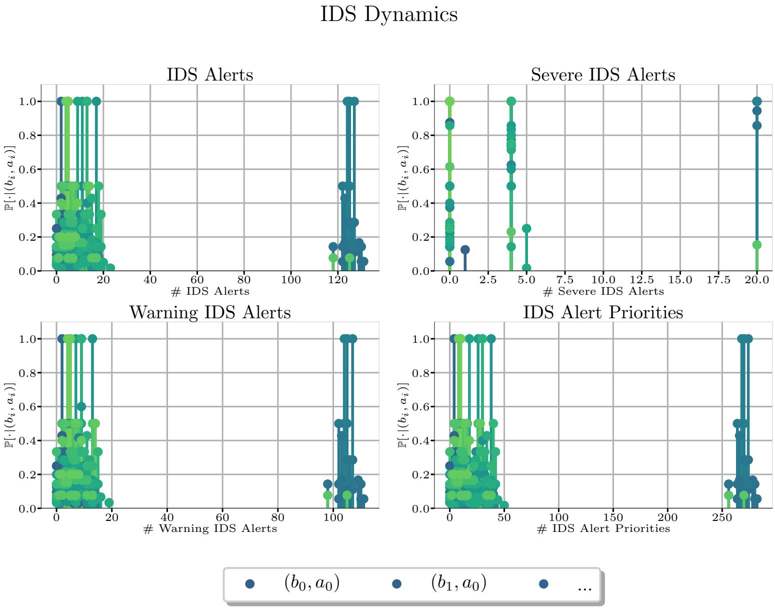
System Identification: Estimated Dynamics Model



System Identification: Estimated Dynamics Model



System Identification: Estimated Dynamics Model



Policy Optimization in the Simulation System using Reinforcement Learning

- **Goal**:
  - Approximate $\pi^* = \arg\max_\pi \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_{t+1}\right]$

- **Learning Algorithm**:
  - Represent π by π_θ
  - Define objective $J(\theta) = \mathbb{E}_{o\sim\rho^{\pi_\theta}, a\sim\pi_\theta}[R]$
  - Maximize $J(\theta)$ by stochastic gradient ascent with gradient $\nabla_\theta J(\theta) = \mathbb{E}_{o\sim\rho^{\pi_\theta}, a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|o) A^{\pi_\theta}(o, a)\right]$
- **Domain-Specific Challenges**:
  - Partial observability
  - Large state space $|\mathcal{S}| = (w+1)^{|\mathcal{N}|\cdot m\cdot(m+1)}$
  - Large action space $|\mathcal{A}| = |\mathcal{N}| \cdot (m+1)$
  - Non-stationary Environment due to presence of adversary
  - Generalization

Policy Optimization in the Simulation System using Reinforcement Learning

- **Goal**:
  - Approximate $\pi^* = \arg\max_\pi \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_{t+1}\right]$

- **Learning Algorithm**:
  - Represent π by π_θ
  - Define objective $J(\theta) = \mathbb{E}_{o \sim \rho^{\pi_\theta}, a \sim \pi_\theta}[R]$
  - Maximize $J(\theta)$ by stochastic gradient ascent with gradient
    $\nabla_\theta J(\theta) = \mathbb{E}_{o \sim \rho^{\pi_\theta}, a \sim \pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|o) A^{\pi_\theta}(o, a)\right]$
- **Domain-Specific Challenges**:
  - Partial observability
  - Large state space $|\mathcal{S}| = (w+1)^{|\mathcal{N}| \cdot m \cdot (m+1)}$
  - Large action space $|\mathcal{A}| = |\mathcal{N}| \cdot (m+1)$
  - Non-stationary Environment due to presence of adversary
  - Generalization

Policy Optimization in the Simulation System using Reinforcement Learning

- **Goal**:
  - Approximate $\pi^* = \arg\max_\pi \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_{t+1}\right]$

- **Learning Algorithm**:
  - Represent π by π_θ
  - Define objective $J(\theta) = \mathbb{E}_{o \sim \rho^{\pi_\theta}, a \sim \pi_\theta}[R]$
  - Maximize $J(\theta)$ by stochastic gradient ascent with gradient $\nabla_\theta J(\theta) = \mathbb{E}_{o \sim \rho^{\pi_\theta}, a \sim \pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|o) A^{\pi_\theta}(o, a)\right]$
- **Domain-Specific Challenges**:
  - Partial observability
  - Large state space $|\mathcal{S}| = (w+1)^{|\mathcal{N}| \cdot m \cdot (m+1)}$
  - Large action space $|\mathcal{A}| = |\mathcal{N}| \cdot (m+1)$
  - Non-stationary Environment due to presence of adversary
  - Generalization

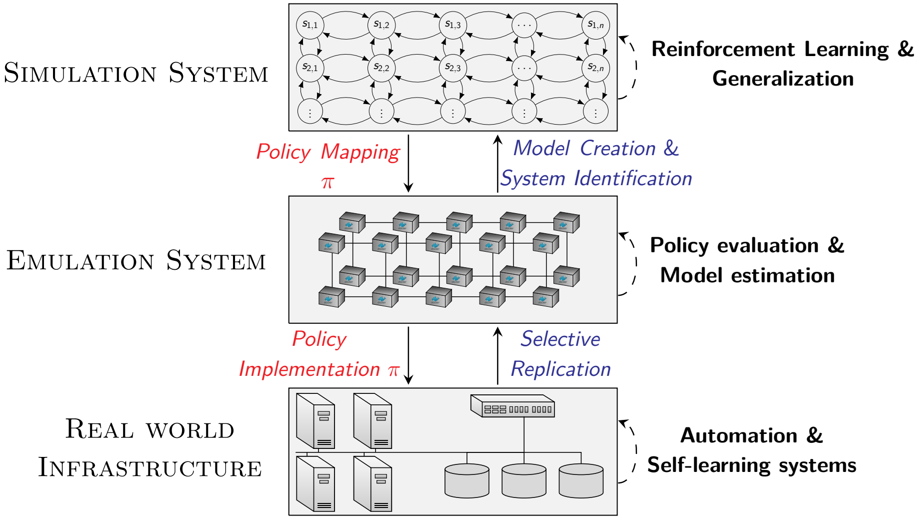
Policy Optimization in the Simulation System using Reinforcement Learning

- **Goal**:
  - Approximate $\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_{t+1}\right]$
- **Learning Algorithm**:
  - Represent π by π_θ
  - Define objective $J(\theta) = \mathbb{E}_{o\sim\rho^{\pi_\theta}, a\sim\pi_\theta}[R]$
  - Maximize $J(\theta)$ by stochastic gradient ascent with gradient $\nabla_\theta J(\theta) = \mathbb{E}_{o\sim\rho^{\pi_\theta}, a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|o) A^{\pi_\theta}(o,a)\right]$
- **Domain-Specific Challenges**:
  - Partial observability
  - Large state space $|\mathcal{S}| = (w+1)^{|\mathcal{N}|\cdot m\cdot(m+1)}$
  - Large action space $|\mathcal{A}| = |\mathcal{N}| \cdot (m+1)$
  - Non-stationary Environment due to presence of adversary
  - Generalization

- Finding Effective Security Strategies through Reinforcement Learning and Self-Play[a]

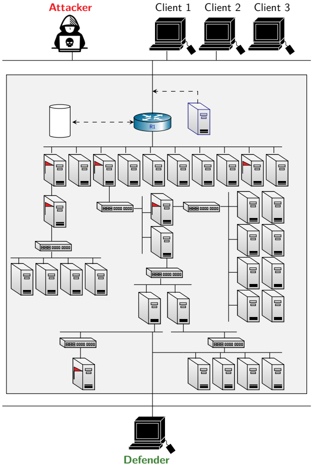
[a] Kim Hammar and Rolf Stadler. "Finding Effective Security Strategies through Reinforcement Learning and Self-Play". In: *International Conference on Network and Service Management (CNSM 2020) (CNSM 2020)*. Izmir, Turkey, Nov. 2020.

Our Method for Finding Effective Security Strategies



Learning Capture-the-Flag Strategies: Target Infrastructure

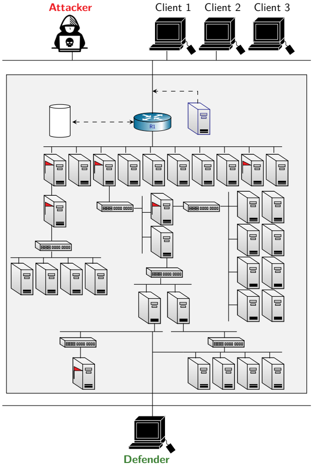
- **Topology**:
  - 32 Servers, 1 IDS (Snort), 3 Clients
- **Services**
  - 1 SNMP, 1 Cassandra, 2 Kafka, 8 HTTP, 1 DNS, 1 SMTP, 2 NTP, 5 IRC, 1 Teamspeak, 1 MongoDB, 1 Samba, 1 RethinkDB, 1 CockroachDB, 2 Postgres, 3 FTP, 15 SSH, 2 FTP
- **Vulnerabilities**
  - 2 CVE-2010-0426, 2 CVE-2010-0426, 1 CVE-2015-3306, 1 CVE-2015-5602, 1 CVE-2016-10033, 1 CVE-2017-7494, 1 CVE-2014-6271
  - 5 Brute-force vulnerabilities
- **Operating Systems**
  - 14 Ubuntu-20, 9 Ubuntu-14, 1 Debian 9:2, 2 Debian Wheezy, 5 Debian Jessie, 1 Kali
- **Traffic**
  - FTP, SSH, IRC, SNMP, HTTP, Telnet, IRC, Postgres, MongoDB, Samba
  - curl, ping, tracerotue, nmap..



Target infrastructure.

Learning Capture-the-Flag Strategies: System Model 1/3

- A hacker (pentester) has T time-periods to **collect flags** hidden in the infrastructure.
- The hacker is located at a dedicated starting position N_0 and can **connect to a gateway** that exposes public-facing services in the infrastructure.
- The hacker has a **pre-defined set (cardinality** ~ 200**) of network/shell commands available**.

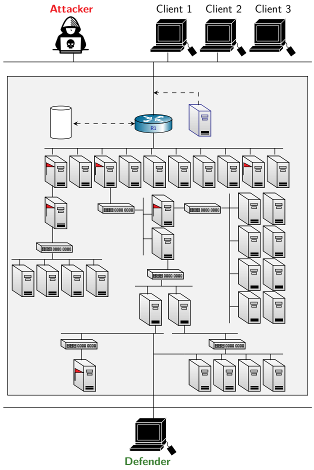


Target infrastructure.

Learning Capture-the-Flag Strategies: System Model 2/3

- By execution of commands, the hacker **collects information**
  - Open ports, failed/successful exploits, vulnerabilities, costs, OS, ...
- Sequences of commands can yield shell-access to nodes
  - Given shell access, the hacker can **search for flags**
- Associated with each command is a **cost** c (execution time) and **noise** n (IDS alerts).

- *The objective is to capture all flags with the minimal cost within the fixed time horizon T. What strategy achieves this end?*

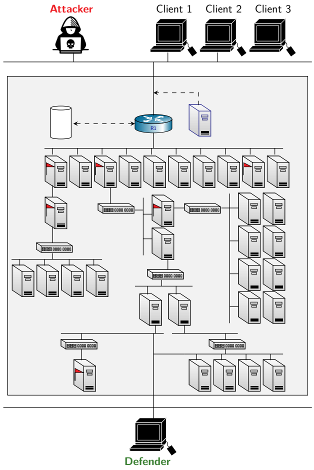


Target infrastructure.

Learning Capture-the-Flag Strategies: System Model 2/3

- By execution of commands, the hacker **collects information**
  - Open ports, failed/successful exploits, vulnerabilities, costs, OS, ...
- Sequences of commands can yield shell-access to nodes
  - Given shell access, the hacker can **search for flags**
- Associated with each command is a **cost** c (execution time) and **noise** n (IDS alerts).

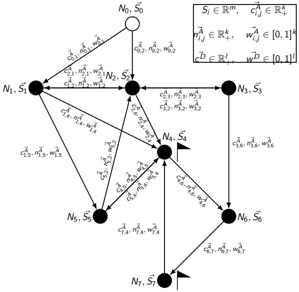
- *The objective is to capture all flags with the minimal cost within the fixed time horizon T. What strategy achieves this end?*



Target infrastructure.

Learning Capture-the-Flag Strategies: System Model 3/3

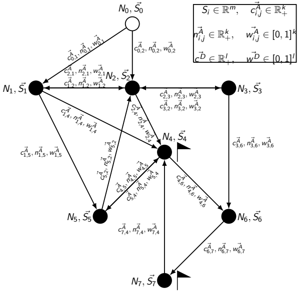
- **Contextual Stochastic CTF with Partial Information**
  - Model infrastructure as a graph $\mathcal{G} = \langle \mathcal{N}, \mathcal{E} \rangle$
  - There are k flags at nodes $\mathcal{C} \subseteq \mathcal{N}$
  - $N_i \in \mathcal{N}$ has a *node state s_i* of m attributes
  - Network state $s = \{s_A, s_i \mid i \in \mathcal{N}\} \in \mathbb{R}^{|\mathcal{N}| \times m + |\mathcal{N}|}$
  - Hacker observes $o^A \subset s$
  - Action space: $\mathcal{A} = \{a_1^A, \ldots, a_k^A\}$, a_i^A (commands)
  - $\forall (b, a) \in \mathcal{A} \times \mathcal{S}$, 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 $s' = F(s, a)$
  - ***Exact dynamics $(F, c^A, n^A, w^A, det(\cdot), \varphi(\cdot))$, are unknown to us!***



Graphical Model.

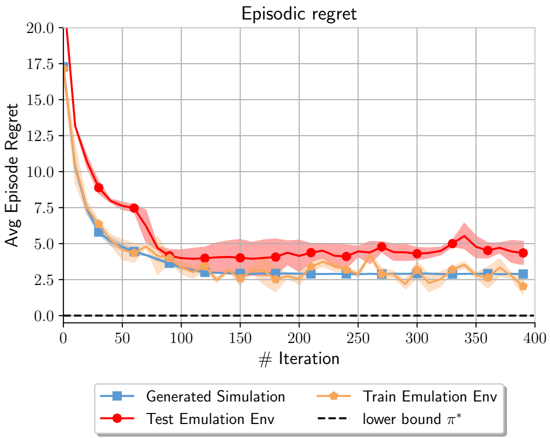
Learning Capture-the-Flag Strategies: System Model 3/3

- **Contextual Stochastic CTF with Partial Information**
  - 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 $s = \{s_A, s_i \mid i \in \mathcal{N}\} \in \mathbb{R}^{|\mathcal{N}| \times m + |\mathcal{N}|}$
  - Hacker observes $o^A \subset s$
  - Action space: $\mathcal{A} = \{a_1^A, \ldots, a_k^A\}$, a_i^A (commands)
  - $\forall (b, a) \in \mathcal{A} \times \mathcal{S}$, 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 $s' = F(s, a)$
  - ***Exact dynamics ($F, c^A, n^A, w^A, det(\cdot), \varphi(\cdot)$), are unknown to us!***

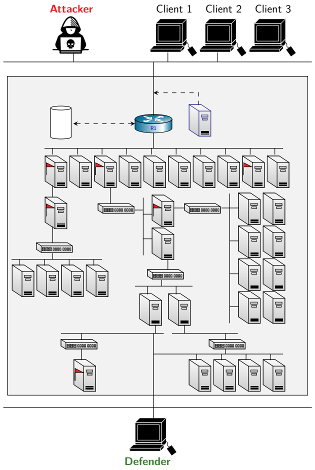


Graphical Model.

Learning Capture-the-Flag Strategies

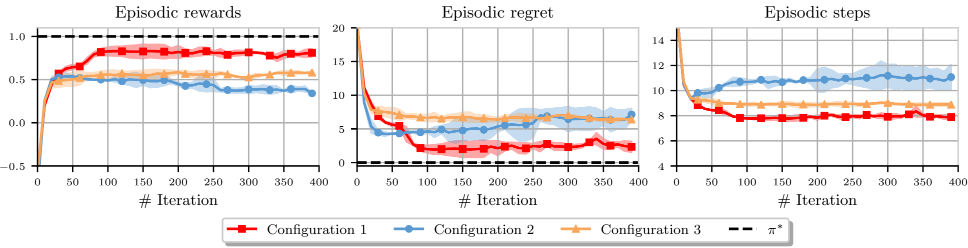


Learning curves (simulation and emulation) of our proposed method.

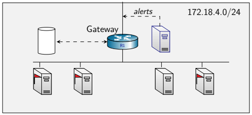


Evaluation infrastructure.

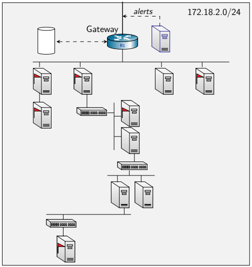
Learning Capture-the-Flag Strategies



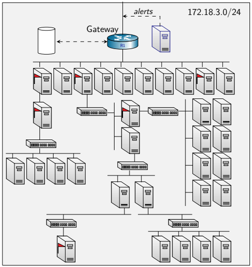
Configuration 1

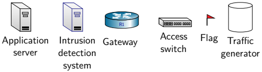


Configuration 2



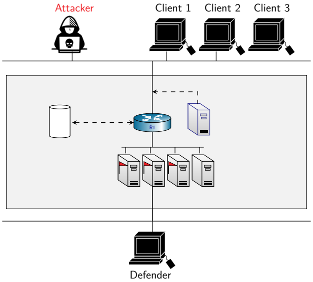
Configuration 3





Learning to Detect Network Intrusions: Target Infrastructure

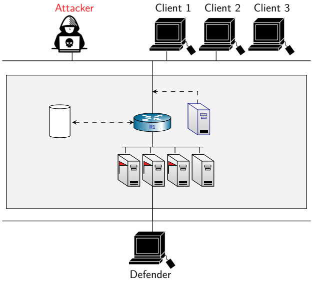
- **Topology**:
  - 6 Servers, 1 IDS (Snort), 3 Clients
- **Services**
  - 3 SSH, 2 HTTP, 1 DNS, 1 Telnet, 1 FTP, 1 MongoDB, 2 SMTP, 1 Tomcat, 1 Teamspeak3, 1 SNMP, 1 IRC, 1 Postgres, 1 NTP
- **Vulnerabilities**
  - 1 CVE-2010-0426, 3 Brute-force vulnerabilities
- **Operating Systems**
  - 4 Ubuntu-20, 1 Ubuntu-14, 1 Kali
- **Traffic**
  - FTP, SSH, IRC, SNMP, HTTP, Telnet, IRC, Postgres, MongoDB,
  - curl, ping, tracerotue, nmap..



Evaluation infrastructure.

Learning to Detect Network Intrusions: System Model (1/3)

- An admin should **manage the infrastructure** for T time-periods.
- The admin can **monitor the infrastructure** to get a belief about it's state b_t
- $b_1, \ldots, b_{T-1}$ can be assumed to be generated from some **unknown** distribution φ.
- If the admin suspects that the infrastructure is being intruded based on b_t, he can **suspend the suspicious user/traffic**.

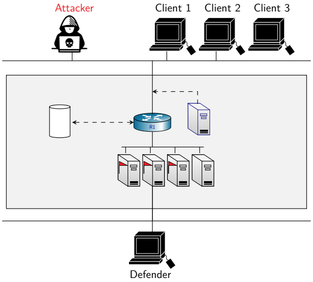


Target infrastructure.

Learning to Detect Network Intrusions: System Model (1/3)

- An admin should **manage the infrastructure** for T time-periods.
- The admin can **monitor the infrastructure** to get a belief about it's state b_t
- $b_1, \ldots, b_{T-1}$ can be assumed to be generated from some **unknown** distribution φ.

- If the admin suspects that the infrastructure is being intruded based on b_t, he can **suspend the suspicious user/traffic**.

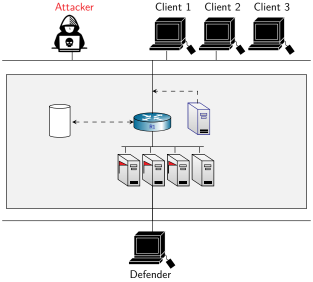


Target infrastructure.

Learning to Detect Network Intrusions: System Model (2/3)

- Suspending traffic from a **true intrusion** yields a reward r (salary bonus)
- Not suspending traffic of a **true intrusion**, incurs a cost c (admin is fired)
- Suspending traffic of a **false intrusion**, incurs a cost of o (breaking the SLA)

- *The objective is to to decide an optimal response for suspending network traffic. What strategy achieves this end?*

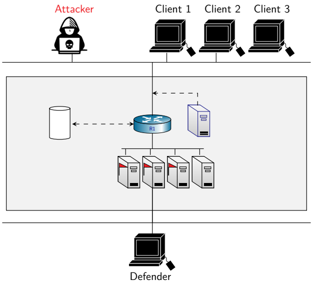


Target infrastructure.

Learning to Detect Network Intrusions: System Model (2/3)

- Suspending traffic from a **true intrusion** yields a reward r (salary bonus)
- Not suspending traffic of a **true intrusion**, incurs a cost c (admin is fired)
- Suspending traffic of a **false intrusion**, incurs a cost of o (breaking the SLA)

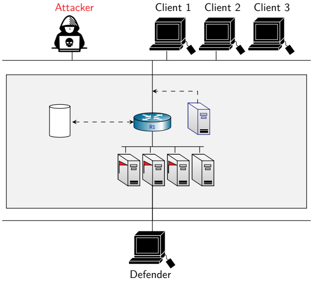
- *The objective is to to decide an optimal response for suspending network traffic. What strategy achieves this end?*



Target infrastructure.

Learning to Detect Network Intrusions: System Model (3/3)

- **Optimal Stopping Problem**
  - Action space $\mathcal{A} = \{\texttt{STOP}, \texttt{CONTINUE}\}$
- **Belief state space $\mathcal{B} \in \mathbb{R}^{8+10\cdot m}$**
  - A belief state $b \in \mathcal{B}$ contains relevant metrics to detect intrusions
  - Alerts from IDS, Entries in `/var/log/auth`, logged in users, TCP connections, processes, ...
- **Reward function $\mathcal{R}$**
  - $r(b_t, \texttt{STOP}, s_t) = \mathbb{1}_{intrusion}\frac{\beta}{t_i}$
  - β is a positive constant and t_i is the number of nodes compromised by the attacker
  - $\implies$ incentive to detect intrusion early.



Target infrastructure.

Structural Properties of the Optimal Policy

- Assumptions: Always an intrusion before T, $f(b_t)$: probability of intrusion given b_t, b_t and p are Markov, $f(b_t)$ is non-decreasing in t.
- Claim: Optimal policy is a **threshold based policy**
  - Necessary condition for optimality (Bellman):

$$u_t(b_t) = \sup_a \left[r_t(b_t, a) + \sum_{b' \in \mathcal{B}} p_t(b'|b_t, a) u_{t+1}(b', a) \right] \quad (1)$$

  - Thus I have that it is optimal to stop at state b_t iff

$$f(b_t) \cdot \frac{\beta}{t_i} \geq \sum_{b' \in \mathcal{B}} \varphi(b') u_{t+1}(b') \quad (2)$$

  - Stopping threshold α_t:

$$\alpha_t \triangleq \frac{t_i}{\beta} \sum_{b' \in \mathcal{B}} \varphi(b') u_{t+1}(b') \quad (3)$$

Structural Properties of the Optimal Policy

- Assumptions: Always an intrusion before T, $f(b_t)$: probability of intrusion given b_t, b_t and p are Markov, $f(b_t)$ is non-decreasing in t.
- Claim: Optimal policy is a **threshold based policy**
  - Necessary condition for optimality (Bellman):

$$u_t(b_t) = \sup_a \left[r_t(b_t, a) + \sum_{b' \in \mathcal{B}} p_t(b'|b_t, a) u_{t+1}(b', a) \right] \quad (4)$$

$$= \max \left[f(b_t) \cdot \frac{\beta}{t_i}, \quad \sum_{b' \in \mathcal{B}} \varphi(b') u_{t+1}(b') \right] \quad (5)$$

  - Thus I have that it is optimal to stop at state b_t iff

$$f(b_t) \cdot \frac{\beta}{t_i} \geq \sum_{b' \in \mathcal{B}} \varphi(b') u_{t+1}(b') \quad (6)$$

  - Stopping threshold α_t:

$$\alpha_t \triangleq \frac{t_i}{\beta} \sum_{b' \in \mathcal{B}} \varphi(b') u_{t+1}(b') \quad (7)$$

Structural Properties of the Optimal Policy

- Assumptions: Always an intrusion before T, $f(b_t)$: probability of intrusion given b_t, b_t and p are Markov, $f(b_t)$ is non-decreasing in t.
- Claim: Optimal policy is a **threshold based policy**
  - Necessary condition for optimality (Bellman):

$$u_t(b_t) = \sup_a \left[r_t(b_t, a) + \sum_{b' \in \mathcal{B}} p_t(b'|b_t, a) u_{t+1}(b', a) \right] \tag{8}$$

$$= \max \left[f(b_t) \cdot \frac{\beta}{t_i}, \quad \sum_{b' \in \mathcal{B}} \varphi(b') u_{t+1}(b') \right] \tag{9}$$

  - Thus I have that it is optimal to stop at state b_t iff

$$f(b_t) \cdot \frac{\beta}{t_i} \geq \sum_{b' \in \mathcal{B}} \varphi(b') u_{t+1}(b') \tag{10}$$

  - Stopping threshold α_t:

$$\alpha_t \triangleq \frac{t_i}{\beta} \sum_{b' \in \mathcal{B}} \varphi(b') u_{t+1}(b') \tag{11}$$

Structural Properties of the Optimal Policy

- Assumptions: Always an intrusion before T, $f(b_t)$: probability of intrusion given b_t, b_t and p are Markov, $f(b_t)$ is non-decreasing in t.
- Claim: Optimal policy is a **threshold based policy**
  - Necessary condition for optimality (Bellman):

$$u_t(b_t) = \sup_a \left[r_t(b_t, a) + \sum_{b' \in \mathcal{B}} p_t(b'|b_t, a) u_{t+1}(b', a) \right] \tag{12}$$

$$= \max \left[f(b_t) \cdot \frac{\beta}{t_i}, \quad \sum_{b' \in \mathcal{B}} \varphi(b') u_{t+1}(b') \right] \tag{13}$$

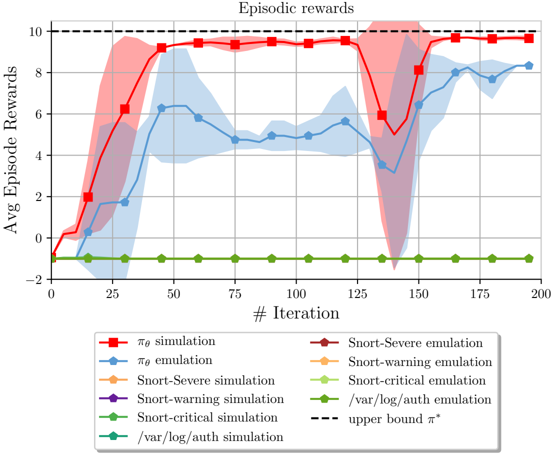
  - Thus I have that it is optimal to stop at state b_t iff

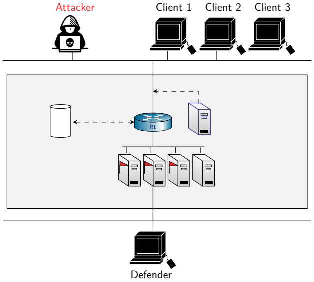
$$f(b_t) \cdot \frac{\beta}{t_i} \geq \sum_{b' \in \mathcal{B}} \varphi(b') u_{t+1}(b') \tag{14}$$

  - Stopping threshold α_t:

$$\alpha_t \triangleq \frac{t_i}{\beta} \sum_{b' \in \mathcal{B}} \varphi(b') u_{t+1}(b') \tag{15}$$

Learning to Detect Network Intrusions

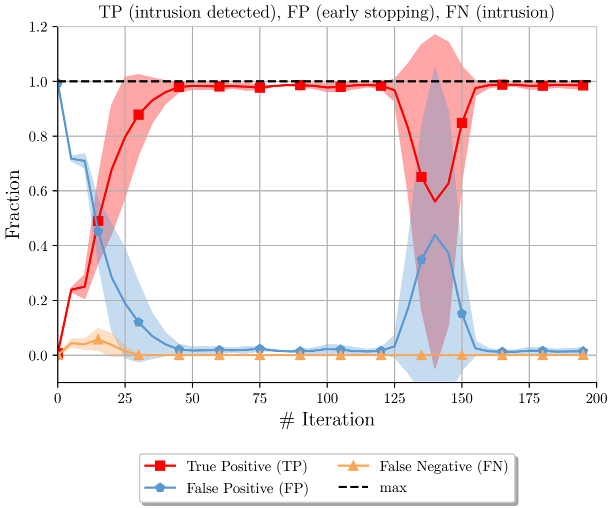




Evaluation infrastructure.

Learning curves (simulation and emulation) of our proposed method.

Learning to Detect Network Intrusions



Trade-off between detection and false positives

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
- **Our research plans:**
  - Improving the system identification algorithm & generalization
  - Evaluation on real world infrastructures