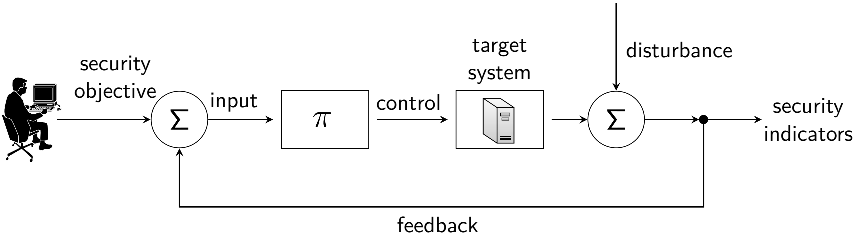
Self-Learning Intrusion Prevention Systems

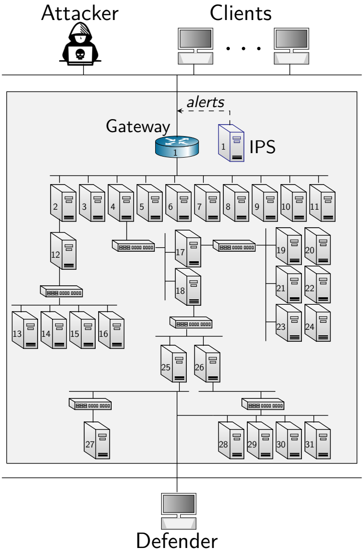
NSE Seminar 21/10 2022

Kim Hammar & Rolf Stadler



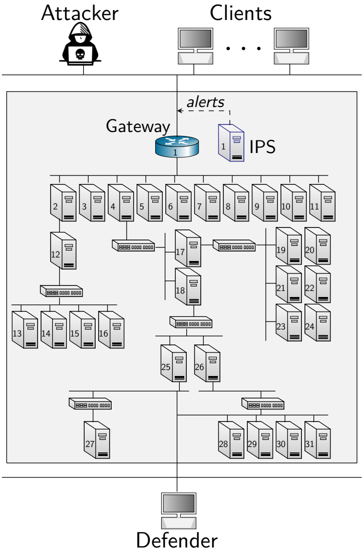
Use Case: Intrusion Prevention

- 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



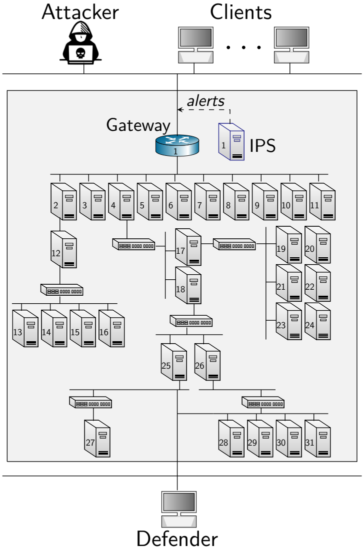
Challenges: Evolving and Automated Attacks

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



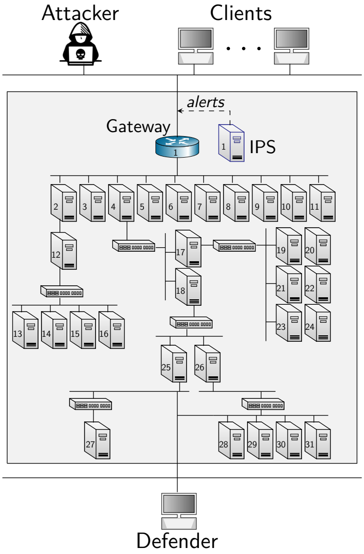
Goal: Automation and Learning

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

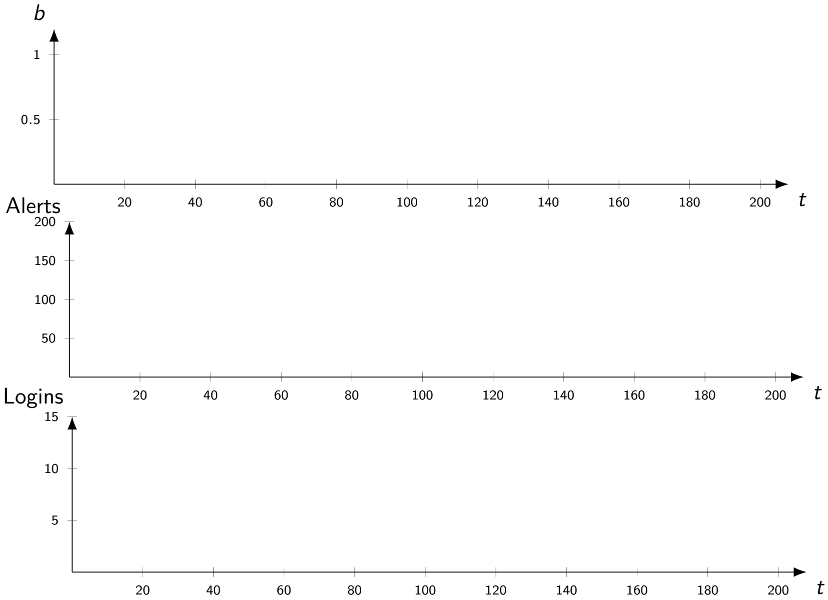


Approach: Self-Learning Security Systems

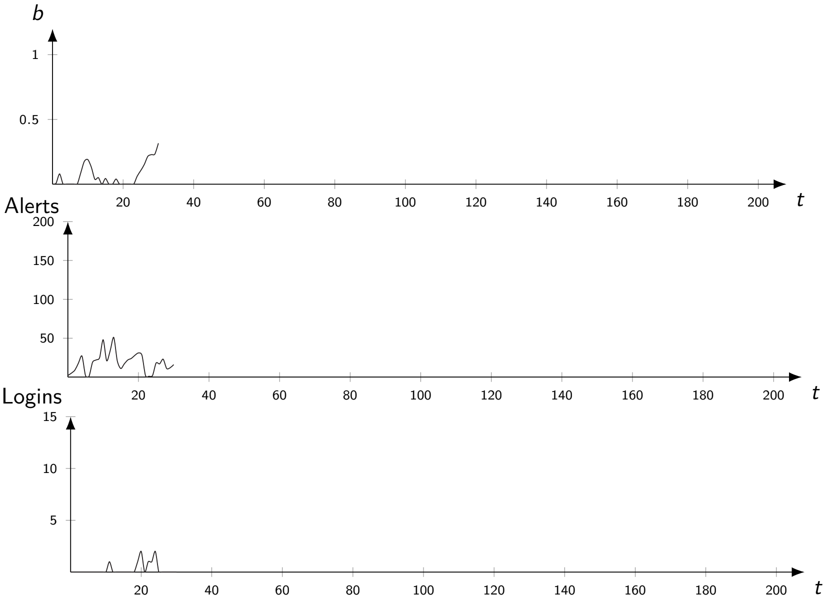
- **Challenges**
  - Evolving & automated attacks
  - Complex infrastructures
- **Our Goal**:
  - Automate security tasks
  - Adapt to changing attack methods
- **Our Approach: Self-Learning Systems**:
  - real-time telemetry
  - stream processing
  - theories from control/game/decision theory
  - computational methods (e.g. dynamic programming & reinforcement learning)
  - automated network management (SDN, NFV, etc.)



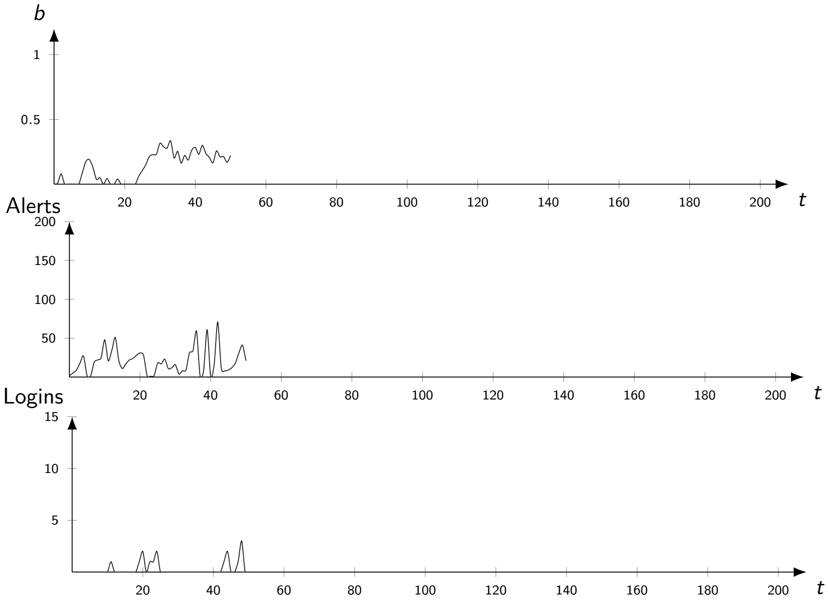
The Intrusion Prevention Problem



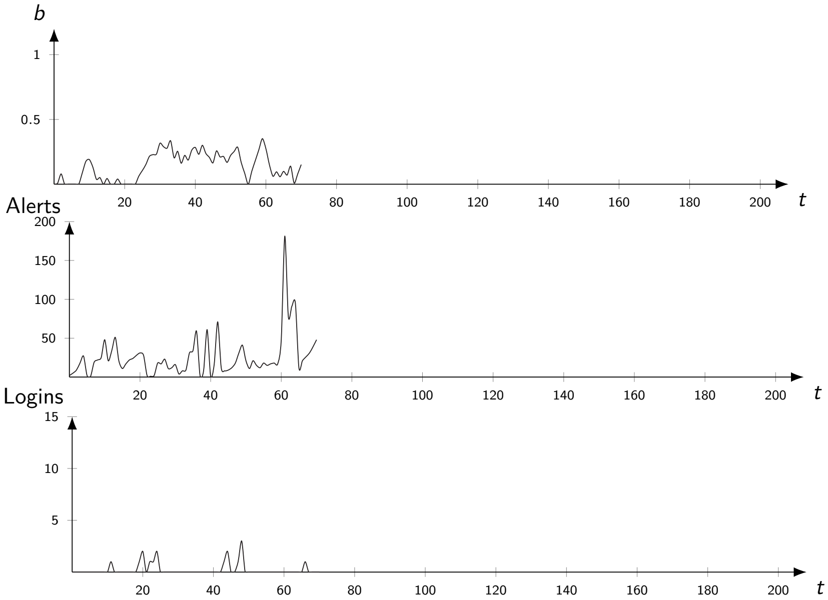
The Intrusion Prevention Problem



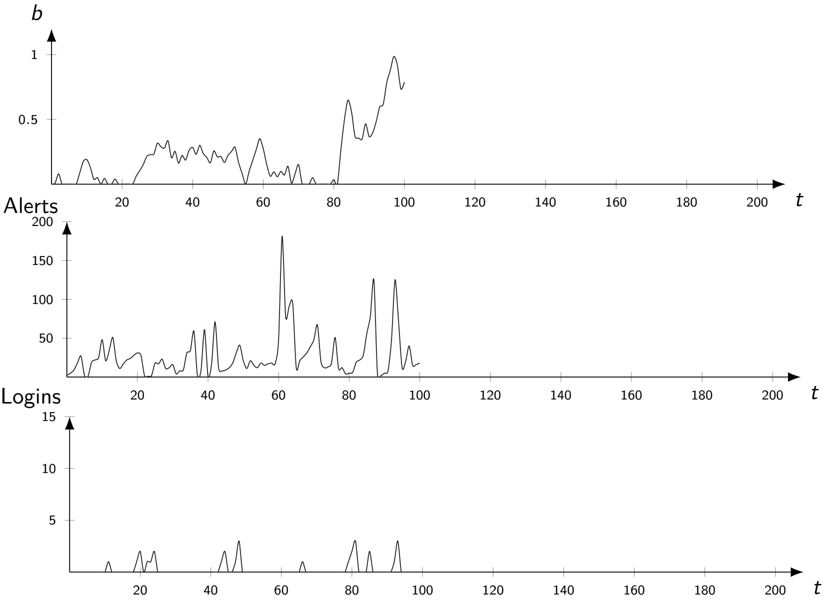
The Intrusion Prevention Problem



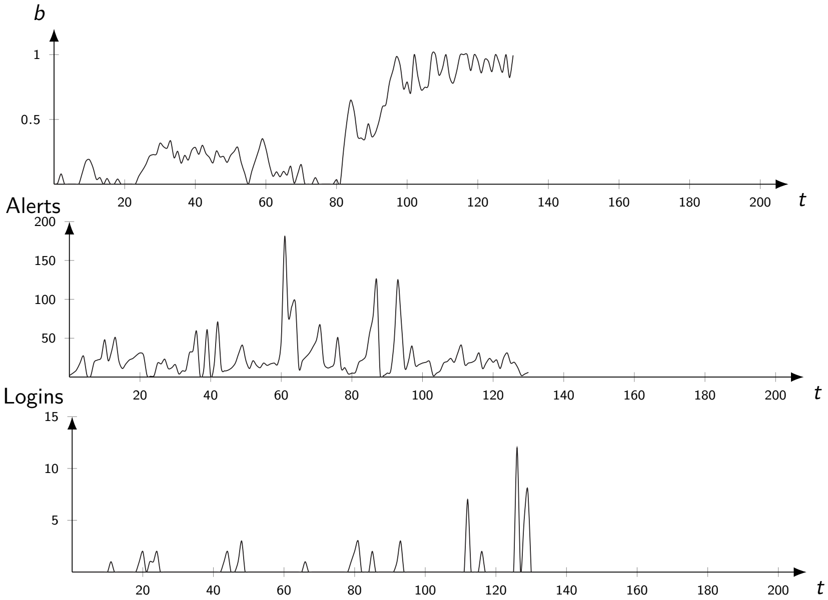
The Intrusion Prevention Problem



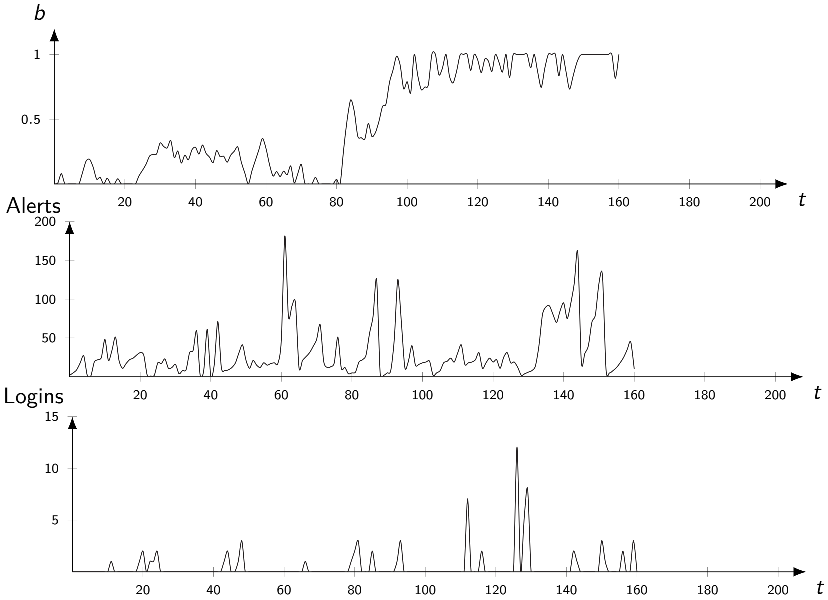
The Intrusion Prevention Problem



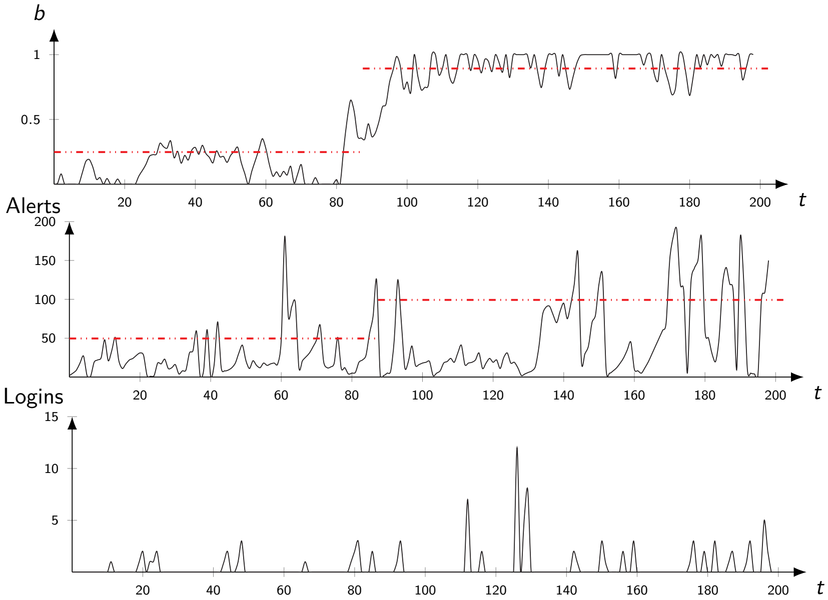
The Intrusion Prevention Problem



The Intrusion Prevention Problem

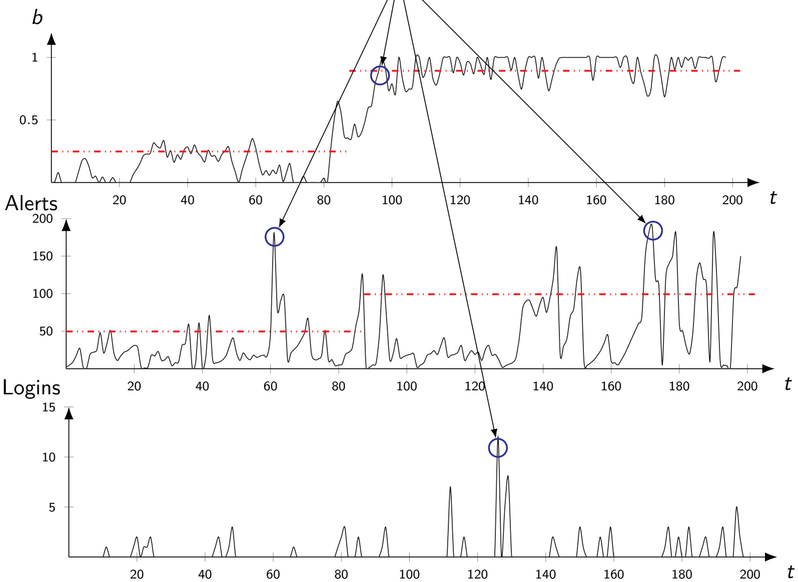


The Intrusion Prevention Problem

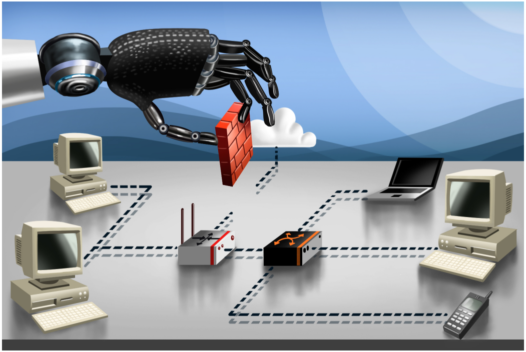


The Intrusion Prevention Problem

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



Self-learning Intrusion Prevention: Current Landscape



Levels of security automation

No automation.
Manual detection.
Manual prevention.
No alerts.
No automatic responses.
Lack of tools.

1980s

Operator assistance.
Manual detection.
Manual prevention.
Audit logs.
Security tools.

1990s

Partial automation.
System has automated functions
for detection/prevention
but requires manual
updating and configuration.
Intrusion detection systems.
Intrusion prevention systems.

2000s-Now

High automation.
System automatically
updates itself.
Automated attack detection.
Automated attack mitigation.

Research

Outline

- **Use Case & Motivation:**
  - Use case: Intrusion prevention
  - Self-learning security systems: current landscape
- **Our Approach**
  - Network emulation and digital twin
  - Stochastic game simulation and reinforcement learning
- **Summary of results so far**
  - Comparison with related work
  - Intrusion prevention through optimal multiple stopping
  - Dynkin games and learning in dynamic environments
  - System for policy validation
- **Outlook on future work**
  - Extend use case
  - Rollout-based methods
- **Conclusions**
  - Takeaways

Outline

- **Use Case & Motivation:**
  - Use case: Intrusion prevention
  - Self-learning security systems: current landscape
- **Our Approach**
  - Network emulation and digital twin
  - Stochastic game simulation and reinforcement learning
- **Summary of results so far**
  - Comparison with related work
  - Intrusion prevention through optimal multiple stopping
  - Dynkin games and learning in dynamic environments
  - System for policy validation
- **Outlook on future work**
  - Extend use case
  - Rollout-based methods
- **Conclusions**
  - Takeaways

Outline

- **Use Case & Motivation:**
  - Use case: Intrusion prevention
  - Self-learning security systems: current landscape
- **Our Approach**
  - Network emulation and digital twin
  - Stochastic game simulation and reinforcement learning
- **Summary of results so far**
  - Comparison with related work
  - Intrusion prevention through optimal multiple stopping
  - Dynkin games and learning in dynamic environments
  - System for policy validation
- **Outlook on future work**
  - Extend use case
  - Rollout-based methods
- **Conclusions**
  - Takeaways

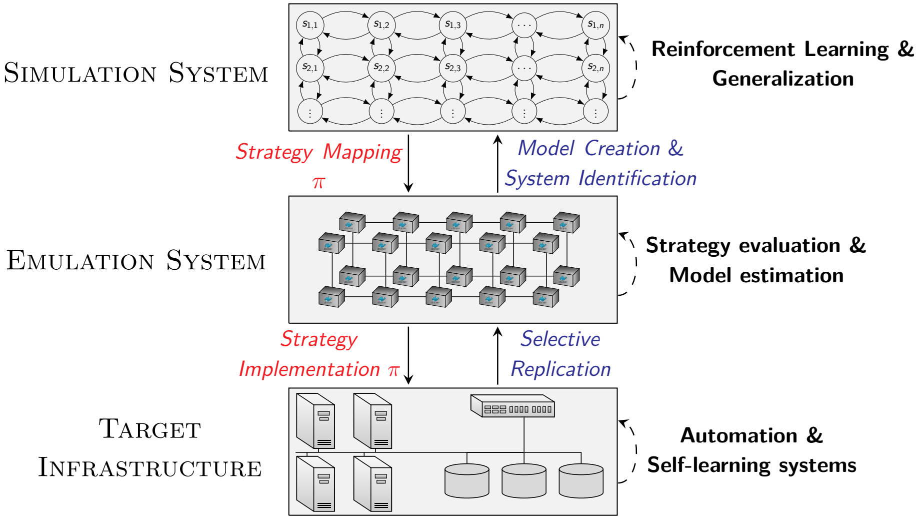
Outline

- **Use Case & Motivation:**
  - Use case: Intrusion prevention
  - Self-learning security systems: current landscape
- **Our Approach**
  - Network emulation and digital twin
  - Stochastic game simulation and reinforcement learning
- **Summary of results so far**
  - Comparison with related work
  - Intrusion prevention through optimal multiple stopping
  - Dynkin games and learning in dynamic environments
  - System for policy validation
- **Outlook on future work**
  - Extend use case
  - Rollout-based methods
- **Conclusions**
  - Takeaways

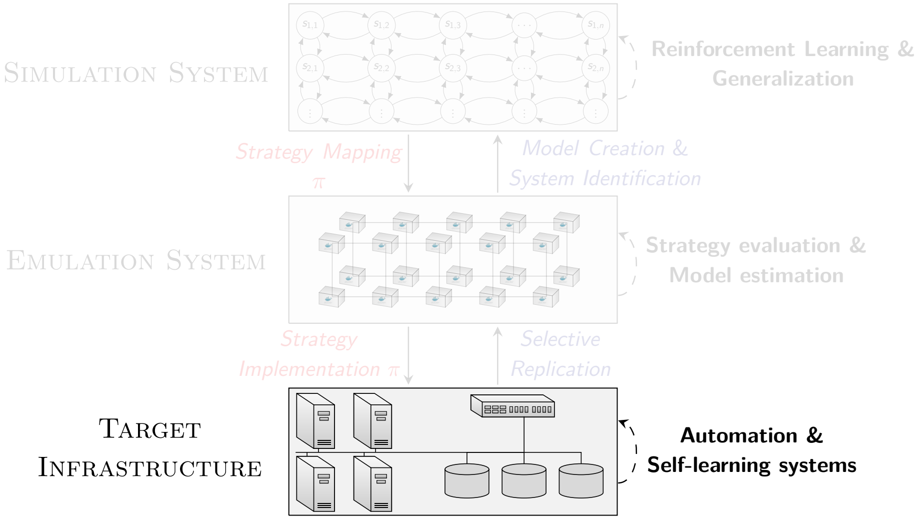
Outline

- **Use Case & Motivation:**
  - Use case: Intrusion prevention
  - Self-learning security systems: current landscape
- **Our Approach**
  - Network emulation and digital twin
  - Stochastic game simulation and reinforcement learning
- **Summary of results so far**
  - Comparison with related work
  - Intrusion prevention through optimal <u>multiple</u> stopping
  - Dynkin games and learning in dynamic environments
  - System for policy validation
- **Outlook on future work**
  - Extend use case
  - Rollout-based methods
- **Conclusions**
  - Takeaways

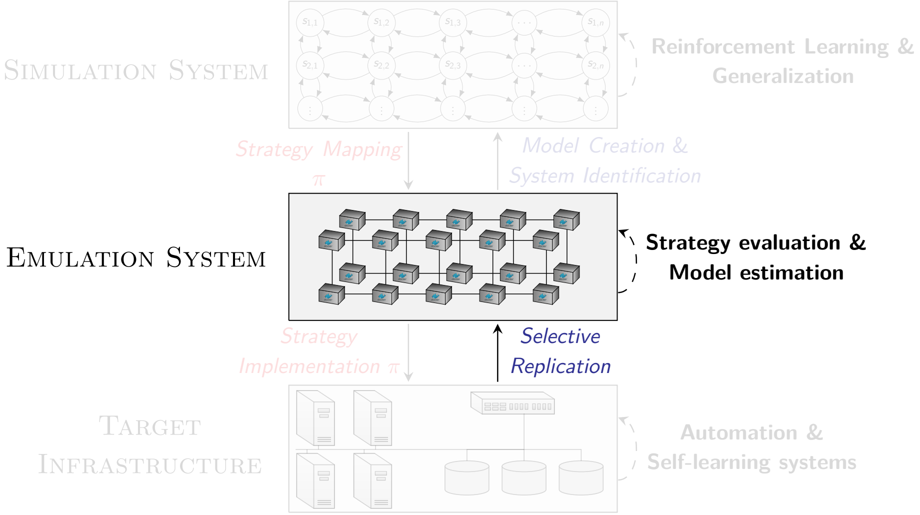
Our Approach for Automated Network Security



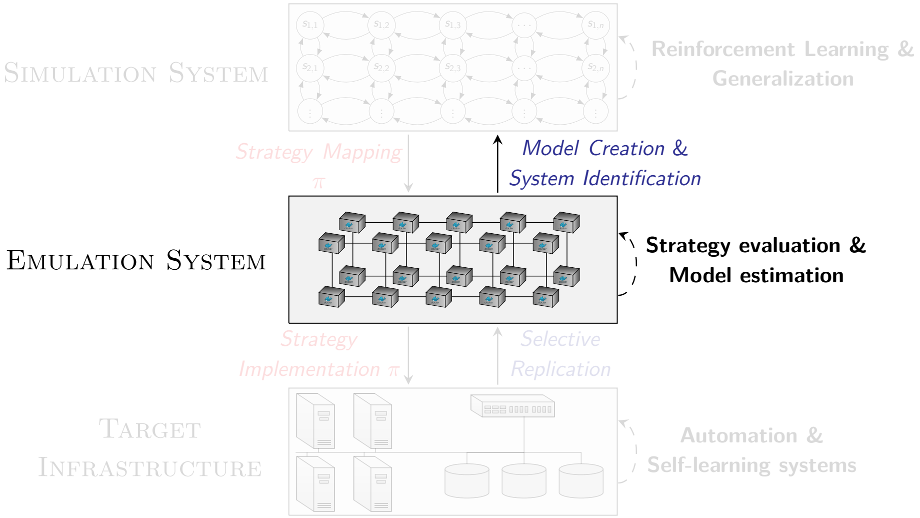
Our Approach for Automated Network Security



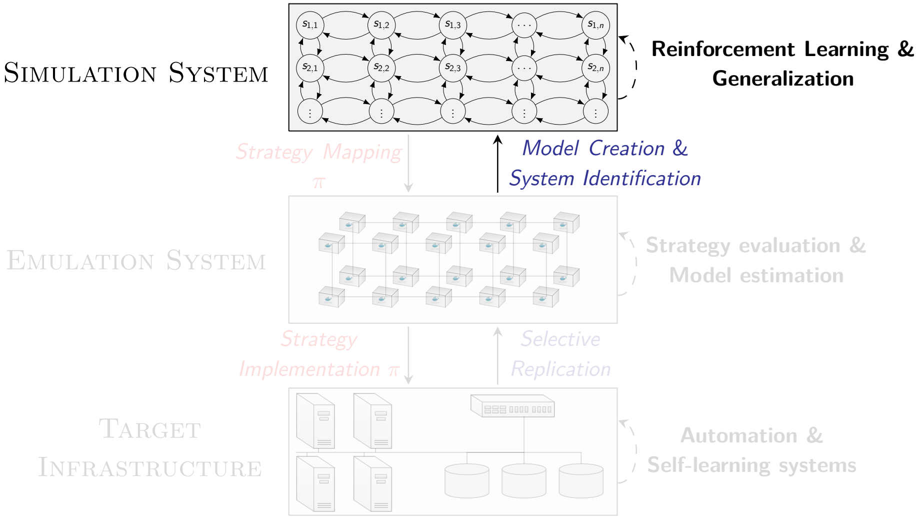
Our Approach for Automated Network Security



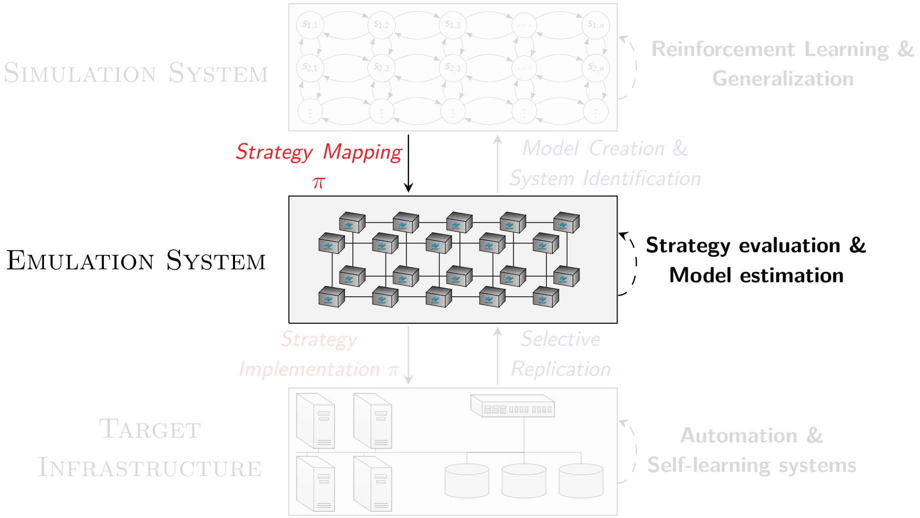
Our Approach for Automated Network Security



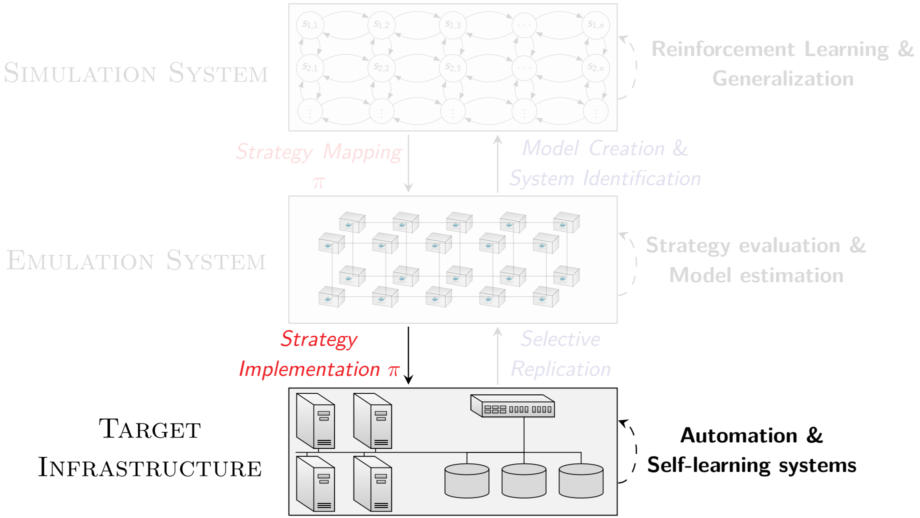
Our Approach for Automated Network Security



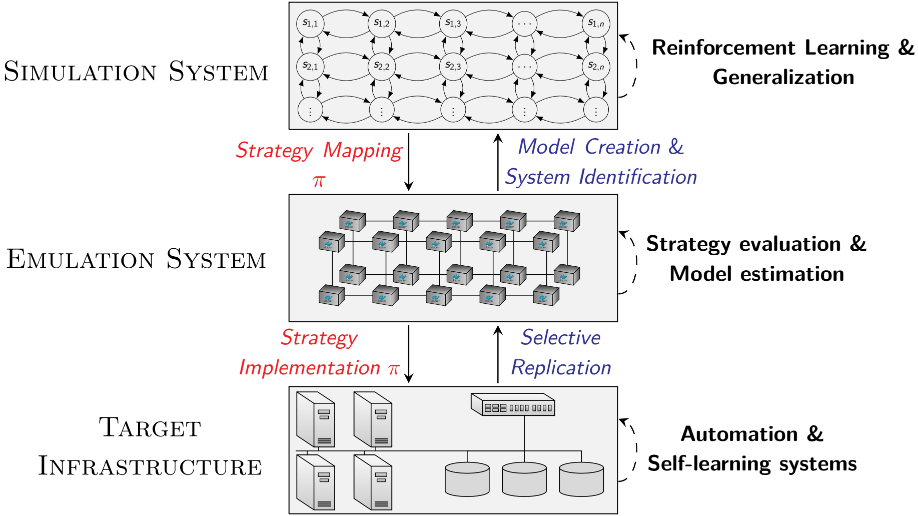
Our Approach for Automated Network Security



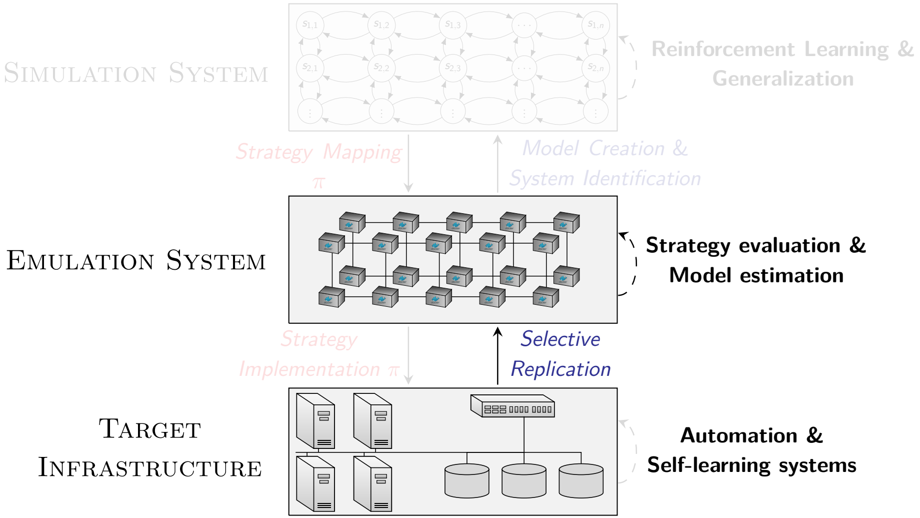
Our Approach for Automated Network Security



Our Approach for Automated Network Security

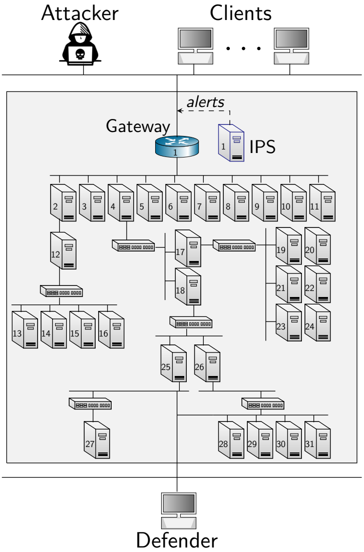


Creating a Digital Twin of the Target Infrastructure



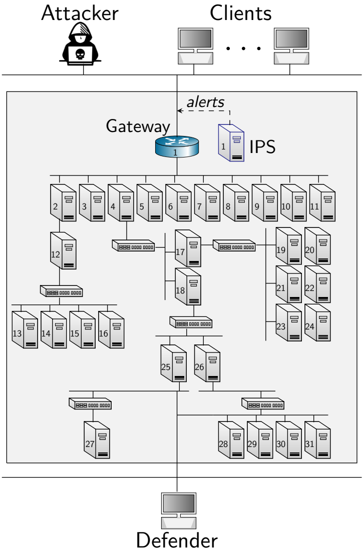
Creating a Digital Twin of the Target Infrastructure

- Emulate **hosts** with docker containers
- Emulate **IPS 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



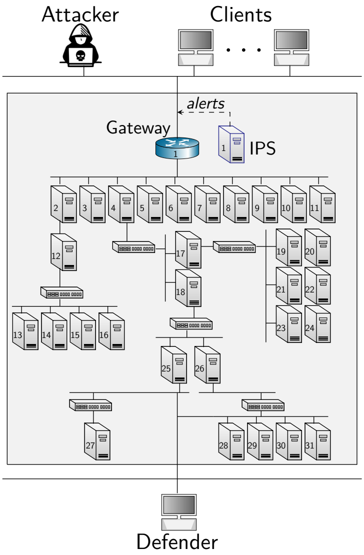
Creating a Digital Twin of the Target Infrastructure

- Emulate **hosts** with docker containers
- Emulate **IPS 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



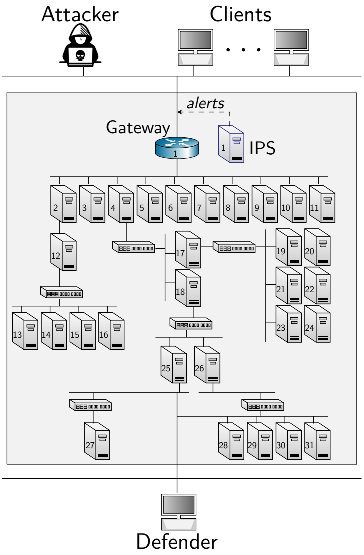
Creating a Digital Twin of the Target Infrastructure

- Emulate **hosts** with docker containers
- Emulate **IPS 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



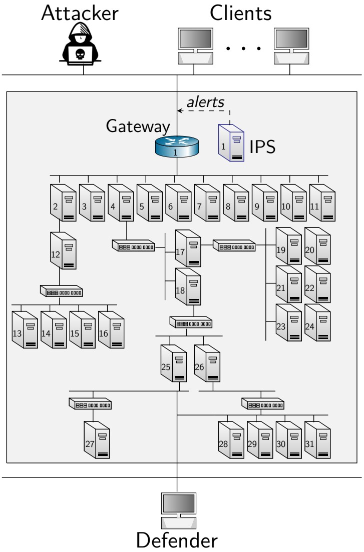
Creating a Digital Twin of the Target Infrastructure

- Emulate **hosts** with docker containers
- Emulate **IPS 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



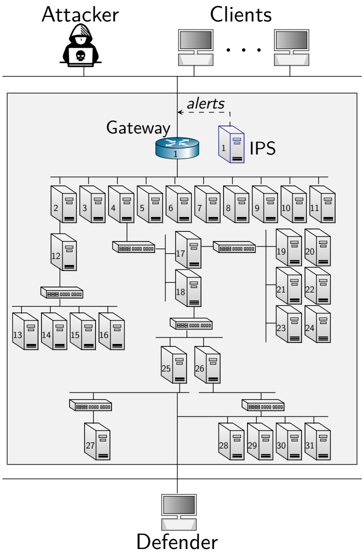
Creating a Digital Twin of the Target Infrastructure

- Emulate **hosts** with docker containers
- Emulate **IPS 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

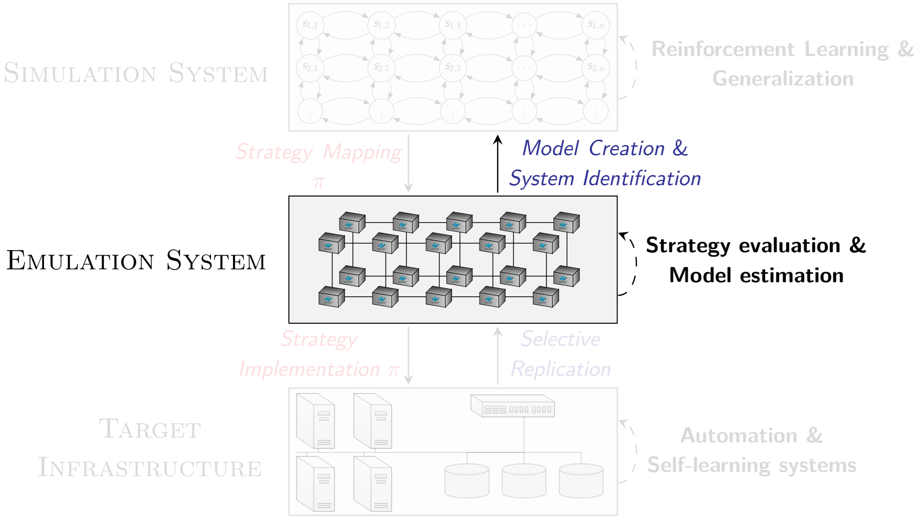


Creating a Digital Twin of the Target Infrastructure

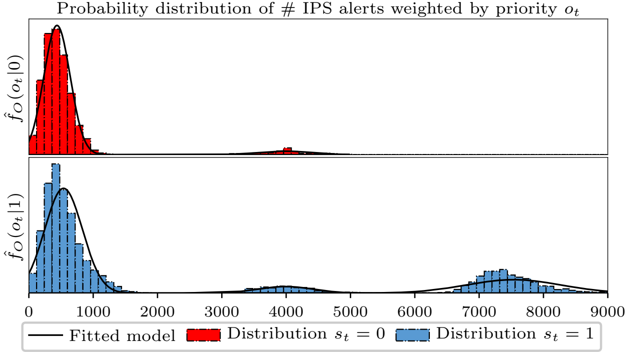
- Emulate **hosts** with docker containers
- Emulate **IPS and vulnerabilities** with software
- Network isolation and **traffic shaping** through NetEm in the Linux kernel
- Enforce **resource constraints** using cgroups.
- Emulate **client arrivals** with Poisson process
- **Internal connections** 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



Our Approach for Automated Network Security

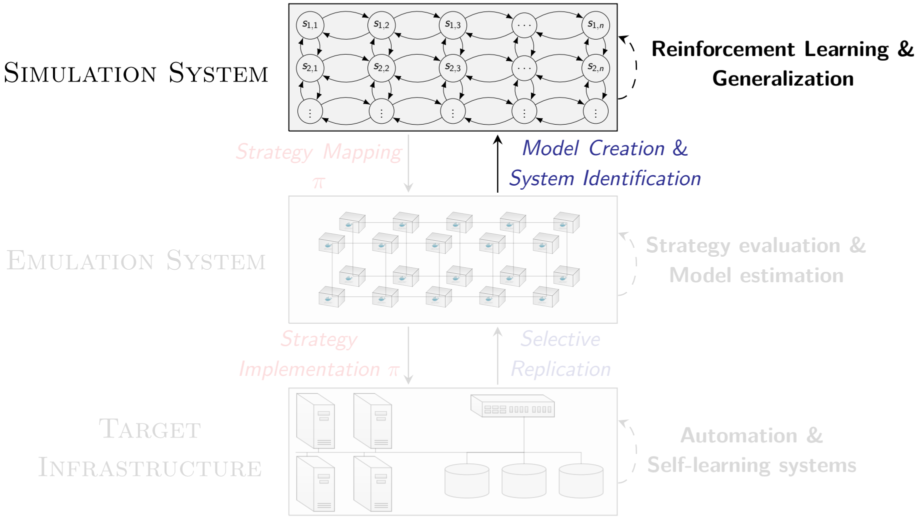


System Identification



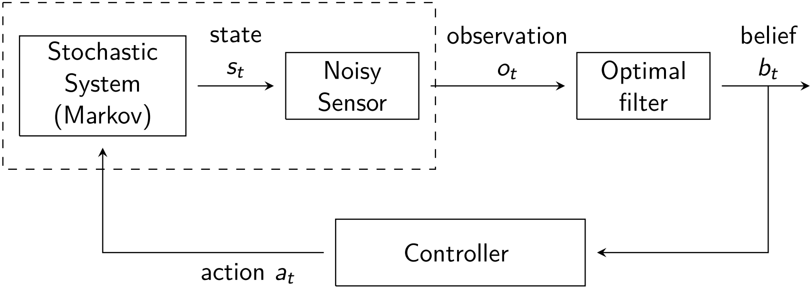
- The distribution f_O of defender observations (system metrics) is unknown.
- We fit a Gaussian mixture distribution $\hat{f}_O$ as an estimate of f_O in the target infrastructure.
- For each state s, we obtain the conditional distribution $\hat{f}_{O|s}$ through expectation-maximization.

Our Approach for Automated Network Security



The Simulation System

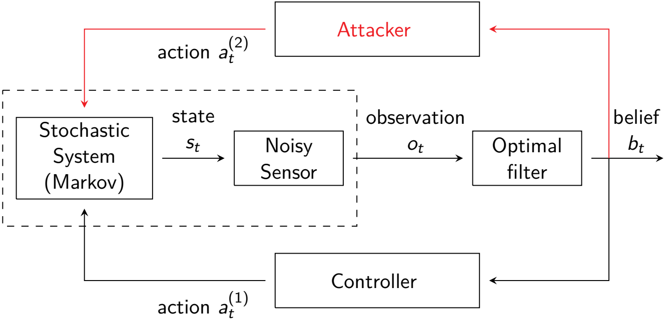
- We model the evolution of the system with a discrete-time dynamical system.
- We assume a Markovian system with stochastic dynamics and partial observability.



The Simulation System

- We model the evolution of the system with a discrete-time dynamical system.
- We assume a Markovian system with stochastic dynamics and partial observability.

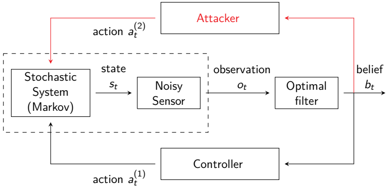
- A Partially Observed Markov Decision Process (POMDP)
  - If attacker is static.
- A Partially Observed Stochastic Game (POSG)
  - If attacker is dynamic.



The Simulation System

- We model the evolution of the system with a discrete-time dynamical system.
- We assume a Markovian system with stochastic dynamics and partial observability.

- A Partially Observed Markov Decision Process (POMDP)
  - If attacker is static.
- A Partially Observed Stochastic Game (POSG)
  - If attacker is dynamic.



- **Compute/learn control strategies**: Stochastic approximation (RL), dynamic programming, linear programming, etc.

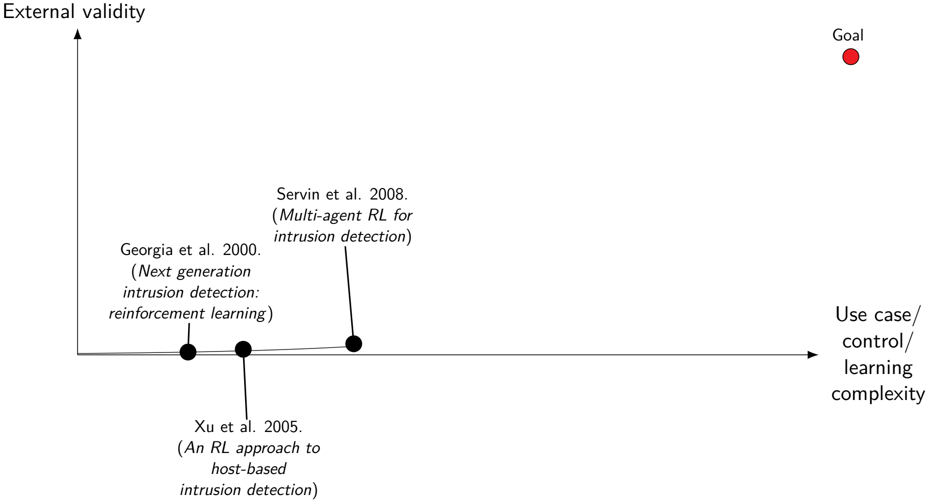
Outline

- **Use Case & Motivation:**
  - Use case: Intrusion prevention
  - Self-learning security systems: current landscape
- **Our Approach**
  - Network emulation and digital twin
  - Stochastic game simulation and reinforcement learning
- **Summary of results so far**
  - Comparison with related work
  - Intrusion prevention through optimal <u>multiple</u> stopping
  - Dynkin games and learning in dynamic environments
  - System for policy validation
- **Outlook on future work**
  - Extend use case
  - Rollout-based methods
- **Conclusions**
  - Takeaways

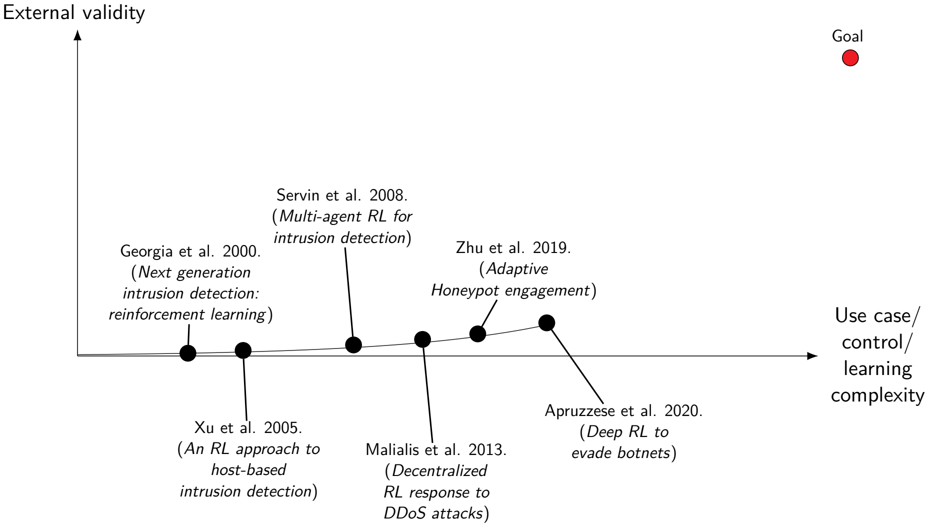
Outline

- **Use Case & Motivation:**
  - Use case: Intrusion prevention
  - Self-learning security systems: current landscape
- **Our Approach**
  - Network emulation and digital twin
  - Stochastic game simulation and reinforcement learning
- **Summary of results so far**
  - Comparison with related work
  - Intrusion prevention through optimal multiple stopping
  - Dynkin games and learning in dynamic environments
  - System for policy validation
- **Outlook on future work**
  - Extend use case
  - Rollout-based methods
- **Conclusions**
  - Takeaways

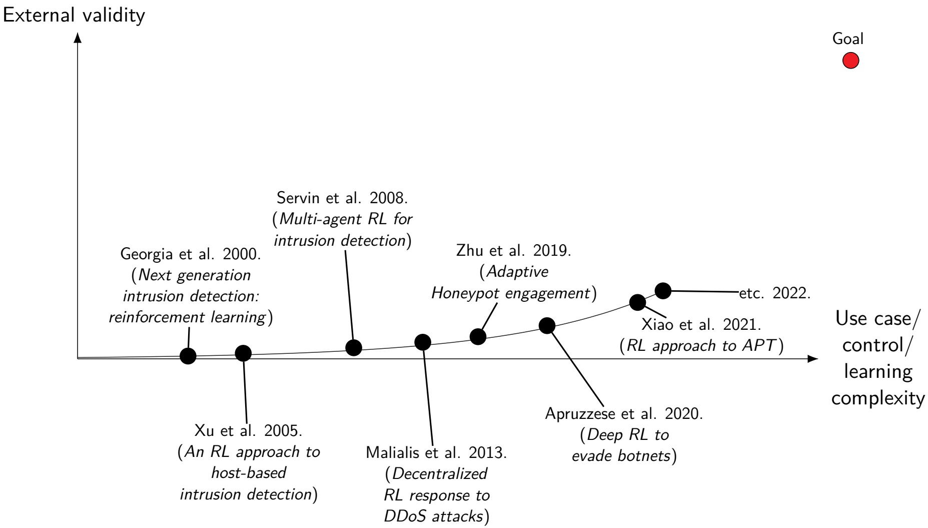
Related Work on Self-Learning Security Systems



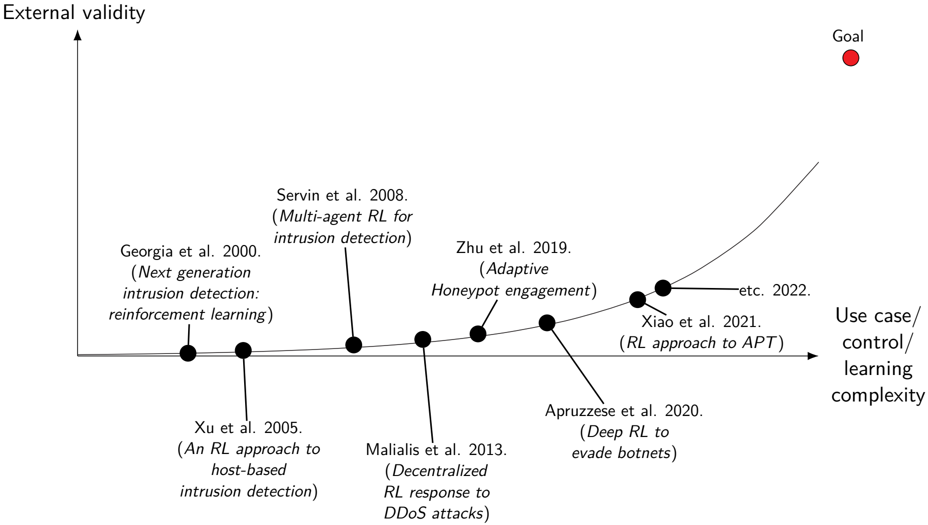
Related Work on Self-Learning Security Systems



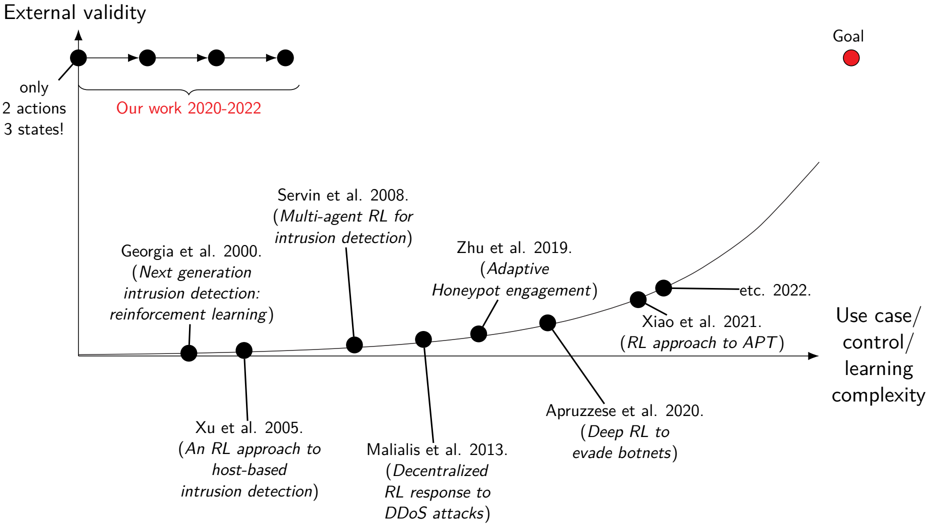
Related Work on Self-Learning Security Systems



Related Work on Self-Learning Security Systems



Related Work on Self-Learning Security Systems



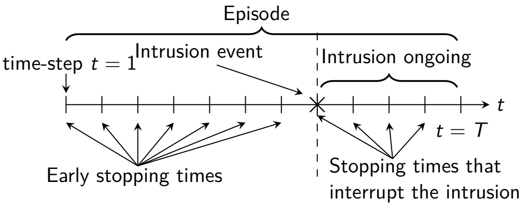
1: Intrusion Prevention through Optimal Stopping[1]

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

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

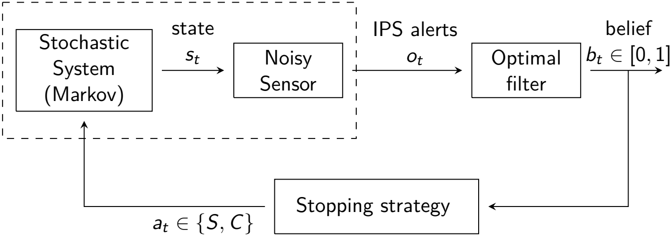
where $\mathcal{R}^{S}_{ss'}$ & $\mathcal{R}^{C}_{ss'}$ are the stop/continue rewards

- Stop action = Defensive action



[1] Kim Hammar and Rolf Stadler. "Learning Intrusion Prevention Policies through Optimal Stopping". In: *International Conference on Network and Service Management (CNSM 2021)*. http://dl.ifip.org/db/conf/cnsm/cnsm2021/1570732932.pdf. Izmir, Turkey, 2021.

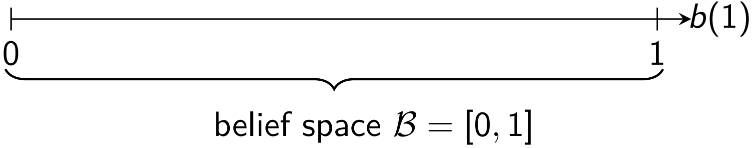
1: Intrusion Prevention through Optimal Stopping[2]



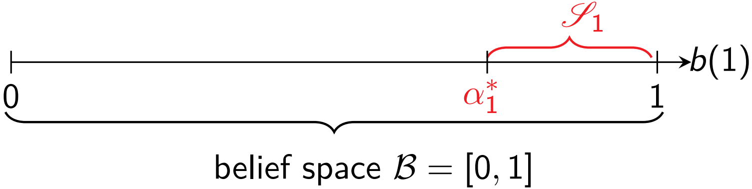
- **States:** Intrusion $s_t \in \{0, 1\}$, terminal $\emptyset$.
- **Observations:**
  - Number of IPS Alerts $o_t \in \mathcal{O}$
  - o_t is drawn from r.v. $O \sim f_O(\cdot|s_t)$.
  - Based on history h_t of observations, the defender can compute the belief $b_t(s_t) = \mathbb{P}[s_t|h_t]$.
- **Actions:** $\mathcal{A}_1 = \mathcal{A}_2 = \{S, C\}$
- **Rewards:** security and service.
- **Transition probabilities:** Follows from game dynamics.

[2] Kim Hammar and Rolf Stadler. "Learning Intrusion Prevention Policies through Optimal Stopping". In: *International Conference on Network and Service Management (CNSM 2021).* http://dl.ifip.org/db/conf/cnsm/cnsm2021/1570732932.pdf. Izmir, Turkey, 2021.

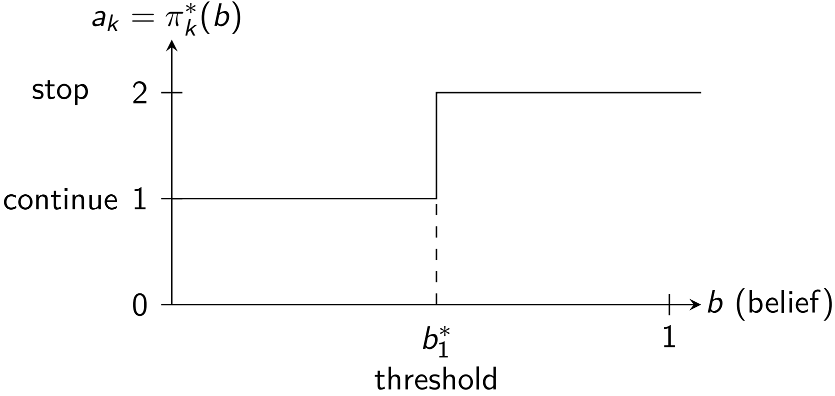
Convex Stopping set with Threshold $\alpha_1^* \in \mathcal{B}$



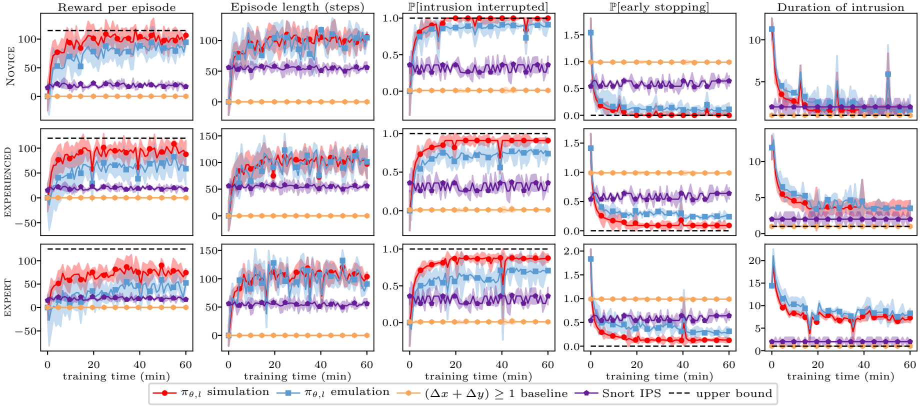
Convex Stopping set with Threshold $\alpha_1^* \in \mathcal{B}$



Bang-Bang Controller with Threshold $\alpha_1^* \in \mathcal{B}$



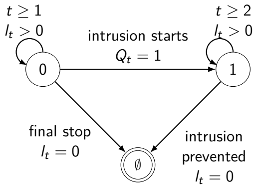
Learning Curves in Simulation and Emulation



2: Intrusion Prevention through Optimal Multiple Stopping[3]

- **Intrusion Prevention through Multiple Optimal Stopping:**
  - Maximize reward of stopping times $\tau_L, \tau_{L-1}, \ldots, \tau_1$:

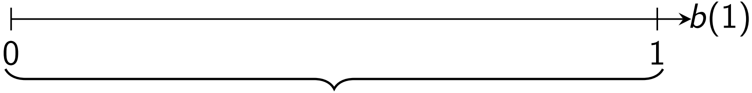
$$\pi_l^* \in \arg\max_{\pi_l} \mathbb{E}_{\pi_l}\Bigg[\sum_{t=1}^{\tau_L - 1} \gamma^{t-1}\mathcal{R}^C_{s_t,s_{t+1},L} + \gamma^{\tau_L-1}\mathcal{R}^S_{s_{\tau_L},s_{\tau_L+1},L} + \ldots + \sum_{t=\tau_2+1}^{\tau_1-1} \gamma^{t-1}\mathcal{R}^C_{s_t,s_{t+1},1} + \gamma^{\tau_1-1}\mathcal{R}^S_{s_{\tau_1},s_{\tau_1+1},1}\Bigg]$$



- Each stopping time = one defensive action

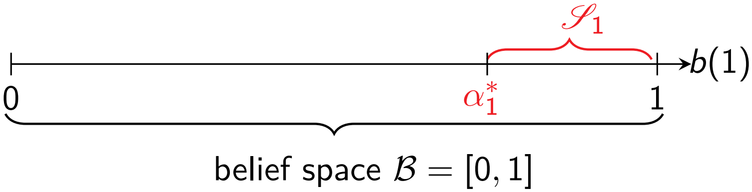
[3] Kim Hammar and Rolf Stadler. "Intrusion Prevention Through Optimal Stopping". In: *IEEE Transactions on Network and Service Management* 19.3 (2022), pp. 2333–2348. DOI: 10.1109/TNSM.2022.3176781.

Structural Result: Optimal Multi-Threshold Policy & Nested Stopping Sets

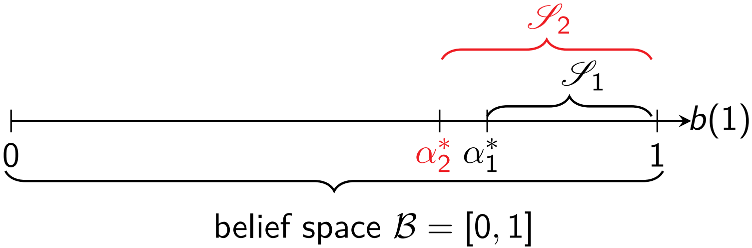


belief space $\mathcal{B} = [0, 1]$

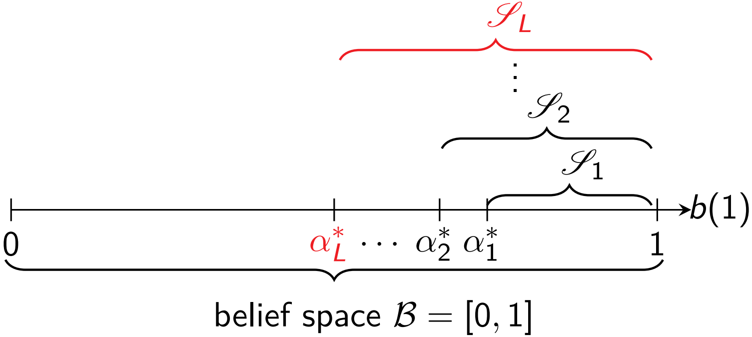
Structural Result: Optimal Multi-Threshold Policy & Nested Stopping Sets



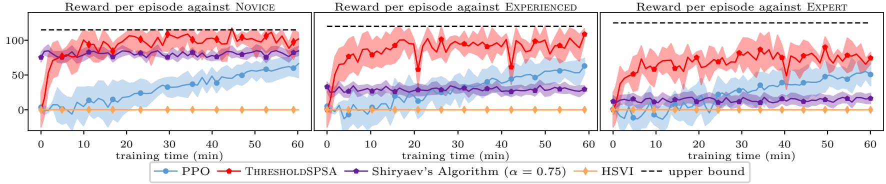
Structural Result: Optimal Multi-Threshold Policy & Nested Stopping Sets



Structural Result: Optimal Multi-Threshold Policy & Nested Stopping Sets



Comparison against State-of-the-art Algorithms



3: Intrusion Prevention through Optimal Multiple Stopping and Game-Play[4]

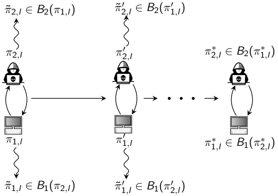
- **Optimal stopping (Dynkin) game**:
  - Dynamic attacker
  - Stop actions of the defender determine when to take defensive actions
  - Goal: find Nash Equilibrium (NE) strategies and game value

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

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

$$B_2(\pi_{1,l}) = \underset{\pi_{2,l}\in\Pi_2}{\arg\min}\, J_1(\pi_{1,l}, \pi_{2,l})$$

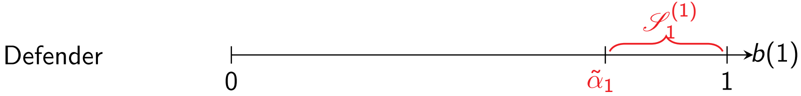
$$(\pi^*_{1,l}, \pi^*_{2,l}) \in B_1(\pi^*_{2,l}) \times B_2(\pi^*_{1,l}) \quad \text{NE}$$



[4] Kim Hammar and Rolf Stadler. "Learning Security Strategies through Game Play and Optimal Stopping". In: *Proceedings of the ML4Cyber workshop, ICML 2022, Baltimore, USA, July 17-23, 2022*. PMLR, 2022.

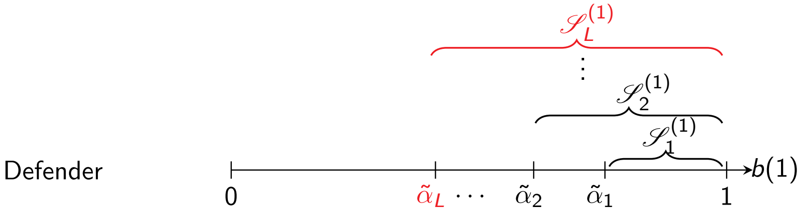
Structure of Best Response Strategies

Structure of Best Response Strategies

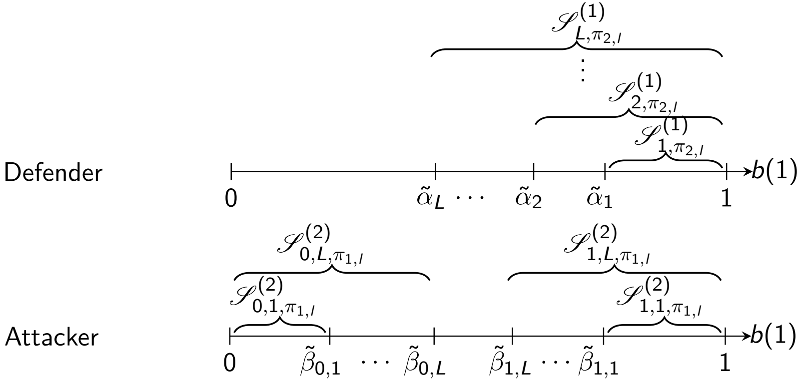


Structure of Best Response Strategies

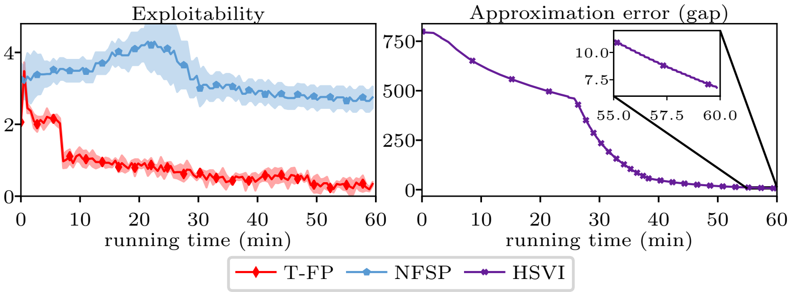
Structure of Best Response Strategies



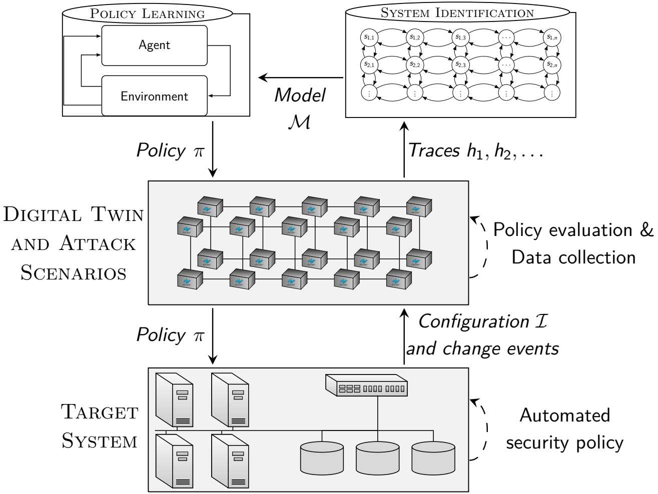
Structure of Best Response Strategies



Converge Rates and Comparison with State-of-the-art Algorithms



4: Learning in Dynamic IT Environments[5]



[5] Kim Hammar and Rolf Stadler. "An Online Framework for Adapting Security Policies in Dynamic IT Environments". In: *International Conference on Network and Service Management (CNSM 2022)*. Thessaloniki, Greece, 2022.

4: Learning in Dynamic IT Environments[6]

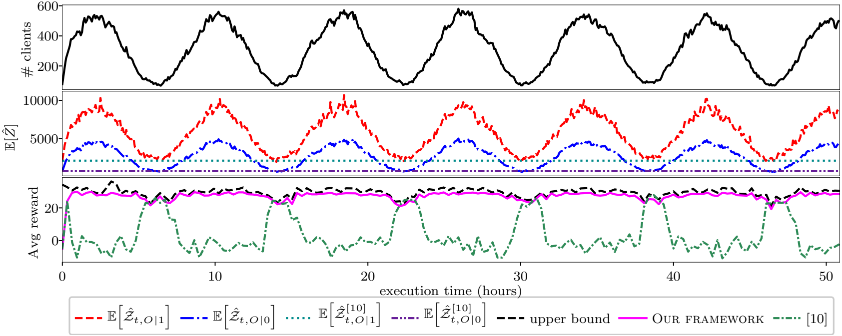
Algorithm 1: High-level execution of the framework

Input: *emulator*: method to create digital twin
φ: system identification algorithm
ϕ: policy learning algorithm

```
Algorithm (emulator, φ, ϕ)
    do in parallel
        DigitalTwin(emulator)
        SystemIdProcess(φ)
        LearningProcess(ϕ)
    end

Procedure DigitalTwin(emulator)
    Loop
        π ← ReceiveFromLearningProcess()
        h_t ← CollectTrace(π)
        SendToSystemIdProcess(h_t)
        UpdateDigitalTwin(emulator)
    EndLoop

Procedure SystemIdProcess(φ)
    Loop
        h_1, h_2, ... ← ReceiveFromDigitalTwin()
        M ← φ(h_1, h_2, ...)                 // estimate model
        SendToLearningProcess(M)
    EndLoop

Procedure LearningProcess(ϕ)
    Loop
        M ← ReceiveFromSystemIdProcess()
        π ← ϕ(M)                             // learn policy π
        SendToDigitalTwin(π)
    EndLoop
```

[6] Kim Hammar and Rolf Stadler. "An Online Framework for Adapting Security Policies in Dynamic IT Environments". In: *International Conference on Network and Service Management (CNSM 2022)*. Thessaloniki, Greece, 2022.

Learning in Dynamic IT Environments[7]



Results from running our framework for 50 hours in the digital twin/emulation.

[7] Kim Hammar and Rolf Stadler. "An Online Framework for Adapting Security Policies in Dynamic IT Environments". In: *International Conference on Network and Service Management (CNSM 2022)*. Thessaloniki, Greece, 2022.

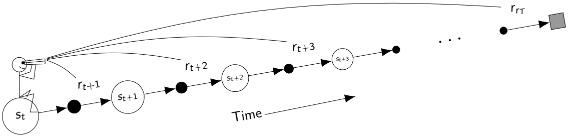
Outline

- **Use Case & Motivation:**
  - Use case: Intrusion prevention
  - Self-learning security systems: current landscape
- **Our Approach**
  - Network emulation and digital twin
  - Stochastic game simulation and reinforcement learning
- **Summary of results so far**
  - Comparison with related work
  - Intrusion prevention through optimal multiple stopping
  - Dynkin games and learning in dynamic environments
  - System for policy validation
- **Outlook on future work**
  - Extend use case
  - Rollout-based methods
- **Conclusions**
  - Takeaways

Outline

- **Use Case & Motivation:**
  - Use case: Intrusion prevention
  - Self-learning security systems: current landscape
- **Our Approach**
  - Network emulation and digital twin
  - Stochastic game simulation and reinforcement learning
- **Summary of results so far**
  - Comparison with related work
  - Intrusion prevention through optimal multiple stopping
  - Dynkin games and learning in dynamic environments
  - System for policy validation
- **Outlook on future work**
  - Extend use case
  - Rollout-based methods
- **Conclusions**
  - Takeaways

Current and Future Work



1. **Extend use case**
   - Additional defender actions
   - Utilize SDN controller and NFV-based defenses
   - Increase observation space and attacker model
   - More heterogeneous client population
2. **Extend solution framework**
   - Model-predictive control
   - Rollout-based techniques
   - Extend system identification algorithm
3. **Extend theoretical results**
   - Exploit symmetries and causal structure
   - Utilize theory to improve sample efficiency
   - Decompose solution framework hierarchically

Conclusions

- We develop a *method* to automatically learn security strategies.
- We apply the method to an **intrusion prevention use case**.
- We design a solution framework guided by the theory of optimal stopping.
- We present several theoretical results on the structure of the optimal solution.
- We show numerical results in a realistic emulation environment.