Learning Intrusion Prevention Policies Through Optimal Stopping[1]

CNSM 2021 | International Conference on Network and Service Management

Kim Hammar & Rolf Stadler

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

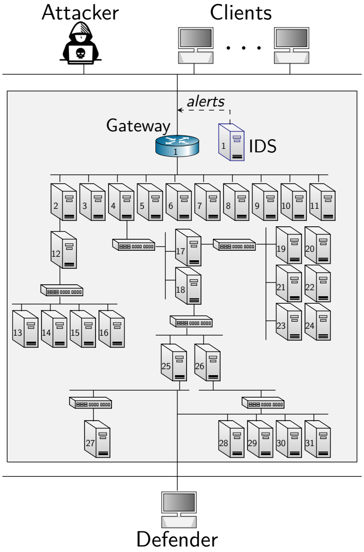
Division of Network and Systems Engineering
KTH Royal Institute of Technology

Oct 28, 2021

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

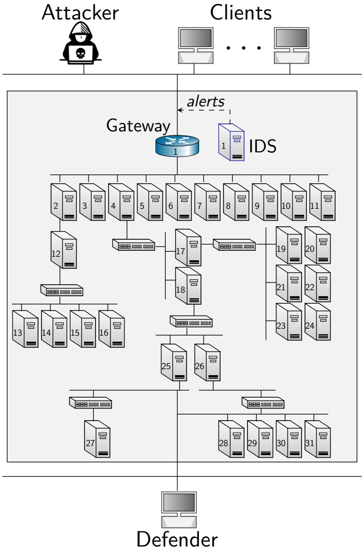
Challenges: Evolving and Automated Attacks

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



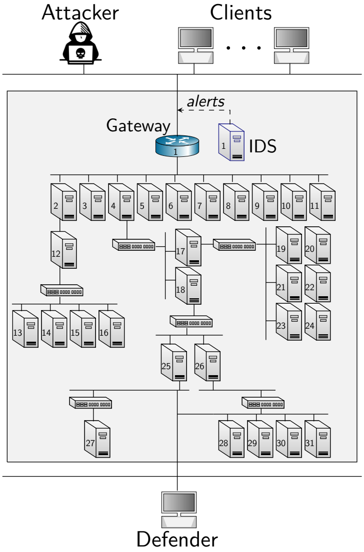
Goal: Automation and Learning

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



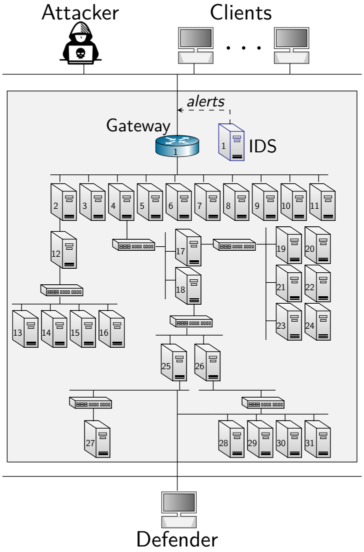
Approach: Game Model & Reinforcement Learning

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



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



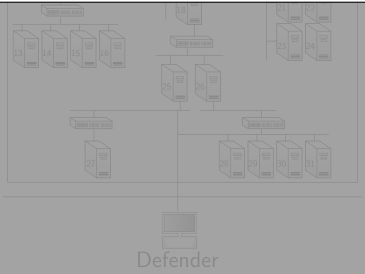
Use Case: Intrusion Prevention

- A **Defender** owns an infrastructure
  - Consists of connected components
  - Components run network services
  - Defender defends the infrastructure

We formulate this use case as an **Optimal Stopping** problem

infrastructure

  - Has a partial view of the infrastructure
  - Wants to compromise specific components
  - Attacks by reconnaissance, exploitation and pivoting



Background on Optimal Stopping Problems

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

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

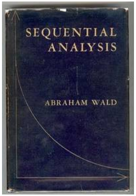
  where $\mathcal{R}^{S}_{ss'}$ & $\mathcal{R}^{C}_{ss'}$ are the stop/continue rewards

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

- **Applications & Use Cases**:
  - Change detection, machine replacement, hypothesis testing, gambling, selling decisions, queue management, advertisement scheduling, the secretary problem, etc.

Background on Optimal Stopping Problems

- **The General Problem**:
  - A Markov process $(s_t)_{t=1}^{T}$ is observed sequentially
  - Two options per t: (i) continue to observe; or (ii) stop
- **History**:
  - Studied in the 18th century to analyze a gambler's fortune
  - Formalized by Abraham Wald in 1947[2]
  - Since then it has been generalized and developed by (Chow[3], Shiryaev & Kolmogorov[4], Bather[5], Bertsekas[6], etc.)



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

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

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

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

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

Background on Optimal Stopping Problems

- **The General Problem**:
  - A Markov process $(s_t)_{t=1}^{T}$ is observed sequentially
  - Two options per t: (i) continue to observe; or (ii) stop
- **History**:
  - Studied in the 18th century to analyze a gambler's fortune
  - Formalized by Abraham Wald in 1947
  - Since then it has been generalized and developed by (Chow, Shiryaev & Kolmogorov, Bather, Bertsekas, etc.)
- **Applications & Use Cases**:
  - Change detection[7], selling decisions[8], queue management[9], advertisement scheduling[10], etc.

[7] Alexander G. Tartakovsky et al. "Detection of intrusions in information systems by sequential change-point methods". In: *Statistical Methodology* 3.3 (2006). ISSN: 1572-3127. DOI: https://doi.org/10.1016/j.stamet.2005.05.003. URL: https://www.sciencedirect.com/science/article/pii/S1572312705000493.

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

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

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

Background on Optimal Stopping Problems

- **The General Problem**:
  - A Markov process $(s_t)_{t=1}^{T}$ is observed sequentially
  - Two options per t: (i) continue to observe; or (ii) stop
- **History**:
  - Studied in the 18th century to analyze a gambler's fortune
  - Formalized by Abraham Wald in 1947
  - Since then it has been generalized and developed by (Chow, Shiryaev & Kolmogorov, Bather, Bertsekas, etc.)
- **Applications & Use Cases**:
  - Change detection[11], selling decisions[12], queue management[13], advertisement scheduling[14], **intrusion prevention**[15] etc.

---

[11] Alexander G. Tartakovsky et al. "Detection of intrusions in information systems by sequential change-point methods". In: *Statistical Methodology* 3.3 (2006). ISSN: 1572-3127. DOI: https://doi.org/10.1016/j.stamet.2005.05.003. URL: https://www.sciencedirect.com/science/article/pii/S1572312705000493.

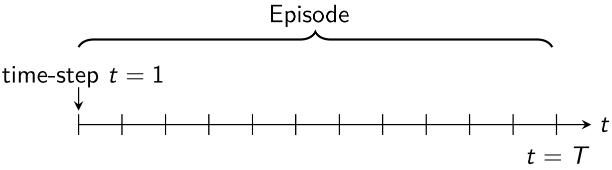
[12] Jacques du Toit and Goran Peskir. "Selling a stock at the ultimate maximum". In: *The Annals of Applied Probability* 19.3 (2009). ISSN: 1050-5164. DOI: 10.1214/08-aap566. URL: http://dx.doi.org/10.1214/08-AAP566.

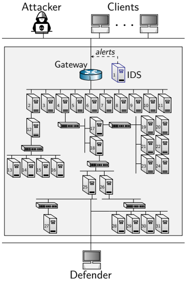
[13] Arghyadip Roy et al. "Online Reinforcement Learning of Optimal Threshold Policies for Markov Decision Processes". In: *CoRR* (2019). http://arxiv.org/abs/1912.10325. eprint: 1912.10325.

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

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

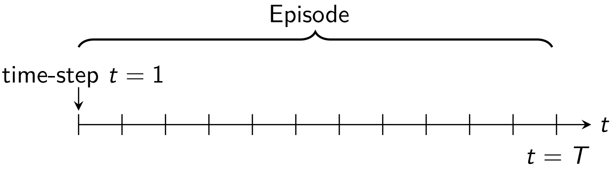
Formulating Intrusion Prevention as a Stopping Problem

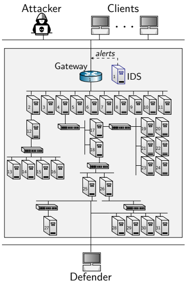




- **Intrusion Prevention as Optimal Stopping Problem**:
  - The system evolves in discrete time-steps.
  - Defender observes the infrastructure (IDS, log files, etc.).
  - An intrusion occurs at an unknown time.
  - The defender can make L stops.
  - Each stop is associated with a defensive action
 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

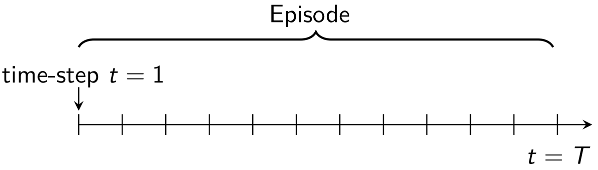
Formulating Intrusion Prevention as a Stopping Problem

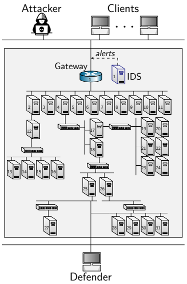




- **Intrusion Prevention as Optimal Stopping Problem**:
  - The system evolves in discrete time-steps.
  - Defender observes the infrastructure (IDS, log files, etc.).
  - An intrusion occurs at an unknown time.
  - The defender can make L stops.
  - Each stop is associated with a defensive action
 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

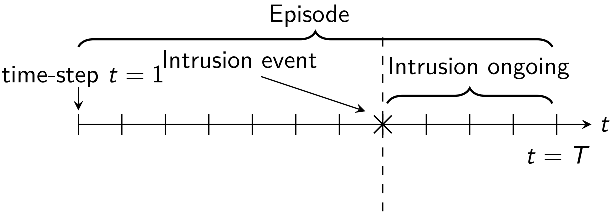
Formulating Intrusion Prevention as a Stopping Problem

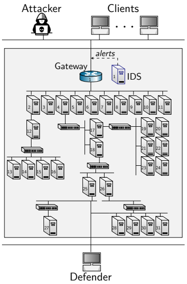




- **Intrusion Prevention as Optimal Stopping Problem**:
  - The system evolves in discrete time-steps.
  - Defender observes the infrastructure (IDS, log files, etc.).
  - An intrusion occurs at an unknown time.
  - The defender can make L stops.
  - Each stop is associated with a defensive action
 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

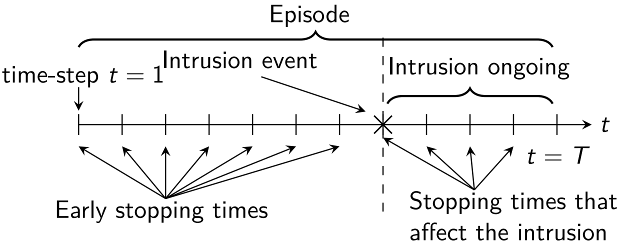
Formulating Intrusion Prevention as a Stopping Problem

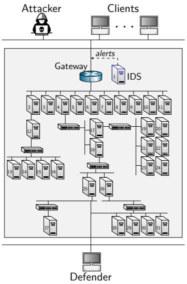




- **Intrusion Prevention as Optimal Stopping Problem**:
  - The system evolves in discrete time-steps.
  - Defender observes the infrastructure (IDS, log files, etc.).
  - An intrusion occurs at an unknown time.
  - The defender can make L stops.
  - Each stop is associated with a defensive action
 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

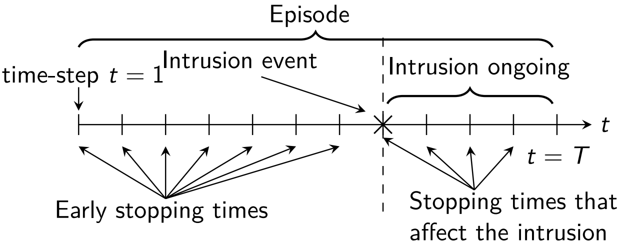
Formulating Intrusion Prevention as a Stopping Problem

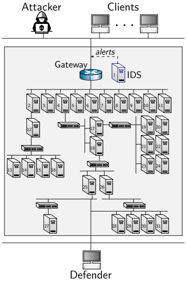




- **Intrusion Prevention as Optimal Stopping Problem**:
  - The system evolves in discrete time-steps.
  - Defender observes the infrastructure (IDS, log files, etc.).
  - An intrusion occurs at an unknown time.
  - The defender can make L stops.
  - Each stop is associated with a defensive action
 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

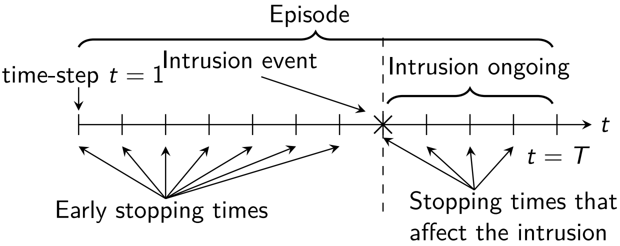
Formulating Intrusion Prevention as a Stopping Problem

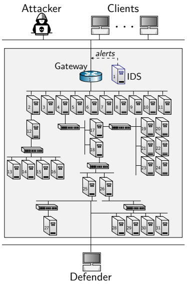




- **Intrusion Prevention as Optimal Stopping Problem**:
  - The system evolves in discrete time-steps.
  - Defender observes the infrastructure (IDS, log files, etc.).
  - An intrusion occurs at an unknown time.
  - The defender can make L stops.
  - Each stop is associated with a defensive action
 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

Formulating Intrusion Prevention as a Stopping Problem

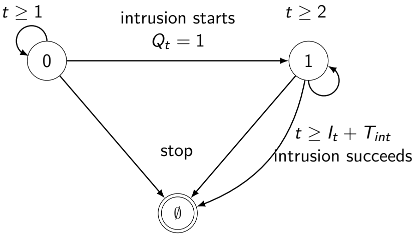




- **Intrusion Prevention as Optimal Stopping Problem**:
  - The system evolves in discrete time-steps.
  - Defender observes the infrastructure (IDS, log files, etc.).
  - An intrusion occurs at an unknown time.
  - The defender can make L stops.
  - Each stop is associated with a defensive action
 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

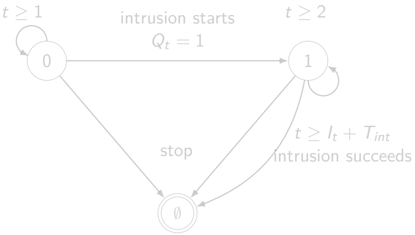
A Partially Observed MDP Model for the Defender

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



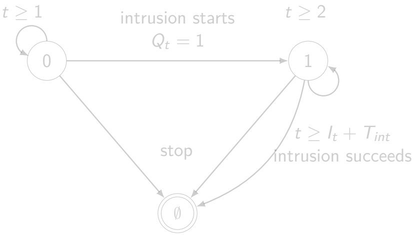
A Partially Observed MDP Model for the Defender

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



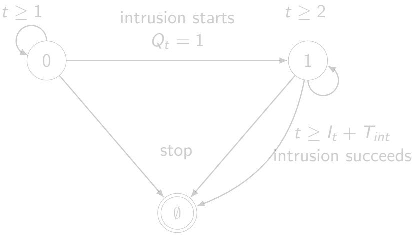
A Partially Observed MDP Model for the Defender

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



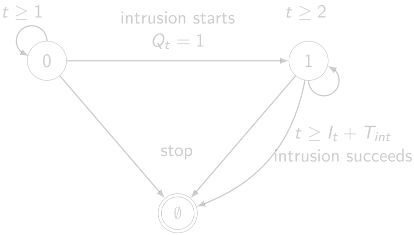
A Partially Observed MDP Model for the Defender

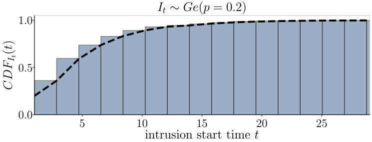
- **States:**
  - Intrusion state $s_t \in \{0, 1\}$, terminal $\emptyset$.
- **Observations:**
  - Severe/Warning IDS Alerts $(\Delta x, \Delta y)$, Login attempts Δz $f_{XYZ}(\Delta x, \Delta y, \Delta z | s_t, I_t, t)$
- **Actions:**
  - "Stop" (S) and "Continue" (C)
- **Rewards:**
  - Reward: security 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

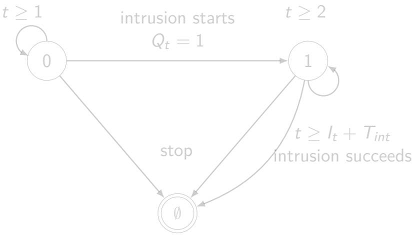
- **States:**
  - Intrusion state $s_t \in \{0, 1\}$, terminal $\emptyset$.
- **Observations:**
  - Severe/Warning IDS Alerts $(\Delta x, \Delta y)$, Login attempts Δz $f_{XYZ}(\Delta x, \Delta y, \Delta z | s_t, I_t, t)$
- **Actions:**
  - "Stop" (S) and "Continue" (C)
- **Rewards:**
  - Reward: security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





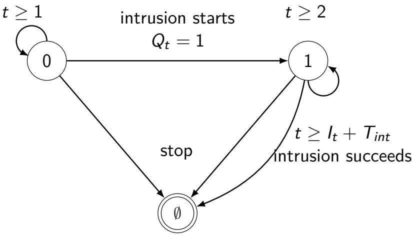
A Partially Observed MDP Model for the Defender

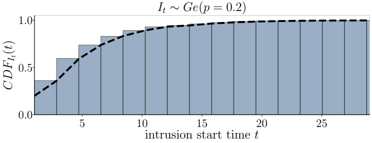
- **States:**
  - Intrusion state $s_t \in \{0, 1\}$, terminal $\emptyset$.
- **Observations:**
  - Severe/Warning IDS Alerts $(\Delta x, \Delta y)$, Login attempts Δz $f_{XYZ}(\Delta x, \Delta y, \Delta z | s_t, I_t, t)$
- **Actions:**
  - "Stop" (S) and "Continue" (C)
- **Rewards:**
  - Reward: security 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

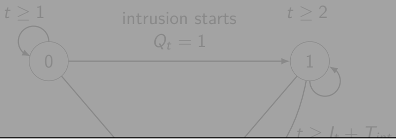
- **States:**
  - Intrusion state $s_t \in \{0, 1\}$, terminal $\emptyset$.
- **Observations:**
  - Severe/Warning IDS Alerts $(\Delta x, \Delta y)$, Login attempts Δz
    $f_{XYZ}(\Delta x, \Delta y, \Delta z | s_t, I_t, t)$
- **Actions:**
  - "Stop" (S) and "Continue" (C)
- **Rewards:**
  - Reward: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





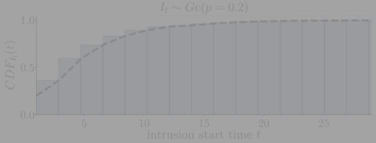
A Partially Observed MDP Model for the Defender

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



We analyze the optimal policy using optimal stopping theory

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

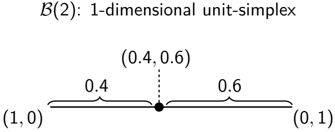


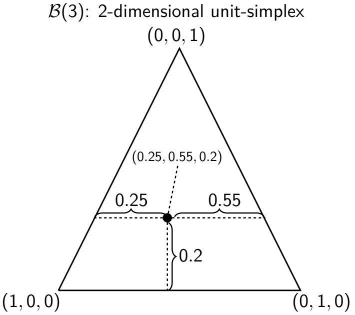
Belief States and Stopping Set

- **Belief States:**
  - The belief state $b_t \in \mathcal{B}$ is defined as $b_t(s_t) = \mathbb{P}[s_t|h_t]$
  - b_t is a sufficient statistic of s_t based on $h_t = (\rho_1, a_1, o_1, \ldots, a_{t-1}, o_t) \in \mathcal{H}$
  - $\mathcal{B}$ is the unit $(|\mathcal{S}| - 1)$-simplex
- **Characterizing the Optimal Policy π^*:**
  - To characterize the optimal policy π^* we partition $\mathcal{B}$ based on optimal actions.
  - $s_t \in \{0, 1\}$. b_t has two components: $b_t(0) = \mathbb{P}[s_t = 0|h_t]$ and $b_t(1) = \mathbb{P}[s_t = 1|h_t]$
  - Since $b_t(0) + b_t(1) = 1$, b_t is completely characterized by $b_t(1)$, $(b_t(0) = 1 - b_t(1))$
  - Hence, $\mathcal{B}$ is the unit interval $[0, 1]$
  - Stopping set $\mathscr{S} = \{b(1) \in [0, 1] : \pi^*\big(b(1)\big) = S\}$
  - Continue set $\mathscr{C} = \{b(1) \in [0, 1] : \pi^*\big(b(1)\big) = C\}$

Belief States and Stopping Set

- **Belief States:**
  - The belief state $b_t \in \mathcal{B}$ is defined as $b_t(s_t) = \mathbb{P}[s_t | h_t]$
  - b_t is a sufficient statistic of s_t based on $h_t = (\rho_1, a_1, o_1, \ldots, a_{t-1}, o_t) \in \mathcal{H}$
  - $\mathcal{B}$ is the unit $(|\mathcal{S}| - 1)$-simplex





Belief States and Stopping Set

- **Belief States:**
  - The belief state $b_t \in \mathcal{B}$ is defined as $b_t(s_t) = \mathbb{P}[s_t | h_t]$
  - b_t is a sufficient statistic of s_t based on $h_t = (\rho_1, a_1, o_1, \ldots, a_{t-1}, o_t) \in \mathcal{H}$
  - $\mathcal{B}$ is the unit $(|\mathcal{S}| - 1)$-simplex

- **Characterizing the Optimal Policy π^*:**
  - To characterize the optimal policy π^* we partition $\mathcal{B}$ based on optimal actions.
  - $s_t \in \{0, 1\}$. b_t has two components: $b_t(0) = \mathbb{P}[s_t = 0 | h_t]$ and $b_t(1) = \mathbb{P}[s_t = 1 | h_t]$
  - Since $b_t(0) + b_t(1) = 1$, b_t is completely characterized by $b_t(1)$, ($b_t(0) = 1 - b_t(1)$)
  - Hence, $\mathcal{B}$ is the unit interval $[0, 1]$
  - Stopping set $\mathscr{S} = \{b(1) \in [0, 1] : \pi^*\big(b(1)\big) = S\}$
  - Continue set $\mathscr{C} = \{b(1) \in [0, 1] : \pi^*\big(b(1)\big) = C\}$

Belief States and Stopping Set

- **Belief States:**
  - The belief state $b_t \in \mathcal{B}$ is defined as $b_t(s_t) = \mathbb{P}[s_t|h_t]$
  - b_t is a sufficient statistic of s_t based on $h_t = (\rho_1, a_1, o_1, \ldots, a_{t-1}, o_t) \in \mathcal{H}$
  - $\mathcal{B}$ is the unit $(|\mathcal{S}| - 1)$-simplex

- **Characterizing the Optimal Policy π^*:**
  - To characterize the optimal policy π^* we partition $\mathcal{B}$ based on optimal actions.
  - $s_t \in \{0, 1\}$. b_t has two components: $b_t(0) = \mathbb{P}[s_t = 0|h_t]$ and $b_t(1) = \mathbb{P}[s_t = 1|h_t]$
  - Since $b_t(0) + b_t(1) = 1$, b_t is completely characterized by $b_t(1)$, $(b_t(0) = 1 - b_t(1))$
  - Hence, $\mathcal{B}$ is the unit interval $[0, 1]$
  - Stopping set $\mathscr{S} = \{b(1) \in [0, 1] : \pi^*\big(b(1)\big) = S\}$
  - Continue set $\mathscr{C} = \{b(1) \in [0, 1] : \pi^*\big(b(1)\big) = C\}$

Belief States and Stopping Set

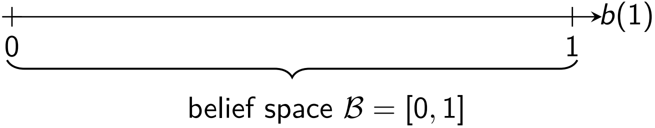
- **Belief States:**
  - The belief state $b_t \in \mathcal{B}$ is defined as $b_t(s_t) = \mathbb{P}[s_t | h_t]$
  - b_t is a sufficient statistic of s_t based on $h_t = (\rho_1, a_1, o_1, \ldots, a_{t-1}, o_t) \in \mathcal{H}$
  - $\mathcal{B}$ is the unit $(|\mathcal{S}| - 1)$-simplex

- **Characterizing the Optimal Policy π^*:**
  - To characterize the optimal policy π^* we partition $\mathcal{B}$ based on optimal actions.
  - $s_t \in \{0, 1\}$. b_t has two components: $b_t(0) = \mathbb{P}[s_t = 0 | h_t]$ and $b_t(1) = \mathbb{P}[s_t = 1 | h_t]$
  - Since $b_t(0) + b_t(1) = 1$, b_t is completely characterized by $b_t(1)$, $(b(0) = 1 - b(1))$
  - Hence, $\mathcal{B}$ is the unit interval $[0, 1]$
  - Stopping set $\mathscr{S} = \{b(1) \in [0, 1] : \pi^*(b(1)) = S\}$
  - Continue set $\mathscr{C} = \{b(1) \in [0, 1] : \pi^*(b(1)) = C\}$

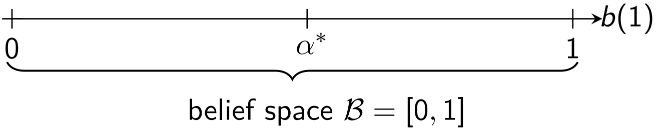
Belief States and Stopping Set

- **Belief States:**
  - The belief state $b_t \in \mathcal{B}$ is defined as $b_t(s_t) = \mathbb{P}[s_t | h_t]$
  - b_t is a sufficient statistic of s_t based on $h_t = (\rho_1, a_1, o_1, \ldots, a_{t-1}, o_t) \in \mathcal{H}$
  - $\mathcal{B}$ is the unit $(|\mathcal{S}| - 1)$-simplex
- **Characterizing the Optimal Policy π^*:**
  - To characterize the optimal policy π^* we partition $\mathcal{B}$ based on optimal actions.
  - $s_t \in \{0, 1\}$. b_t has two components: $b_t(0) = \mathbb{P}[s_t = 0 | h_t]$ and $b_t(1) = \mathbb{P}[s_t = 1 | h_t]$
  - Since $b_t(0) + b_t(1) = 1$, b_t is completely characterized by $b_t(1)$, $(b_t(0) = 1 - b_t(1))$
  - Hence, $\mathcal{B}$ is the unit interval $[0, 1]$
  - Stopping set $\mathscr{S} = \{b(1) \in [0, 1] : \pi^*\big(b(1)\big) = S\}$
  - Continue set $\mathscr{C} = \{b(1) \in [0, 1] : \pi^*\big(b(1)\big) = C\}$

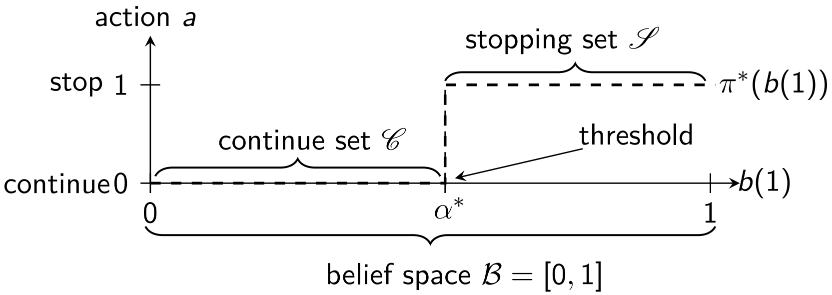
Threshold Properties of the Optimal Defender Policy



Threshold Properties of the Optimal Defender Policy

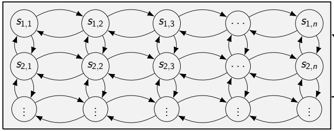


Threshold Properties of the Optimal Defender Policy



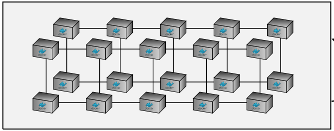
Our Method for Finding Effective Security Strategies

SIMULATION SYSTEM



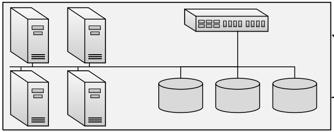
Reinforcement Learning & POMDP Model

EMULATION SYSTEM



Model estimation

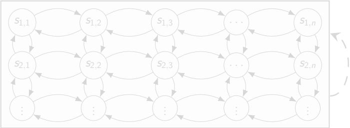
REAL WORLD INFRASTRUCTURE



Automation & Self-learning systems

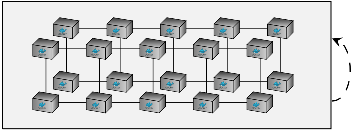
Our Method for Finding Effective Security Strategies

Simulation System



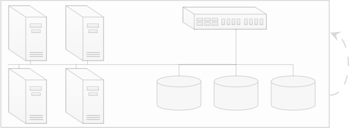
Reinforcement Learning & POMDP Model

Emulation System



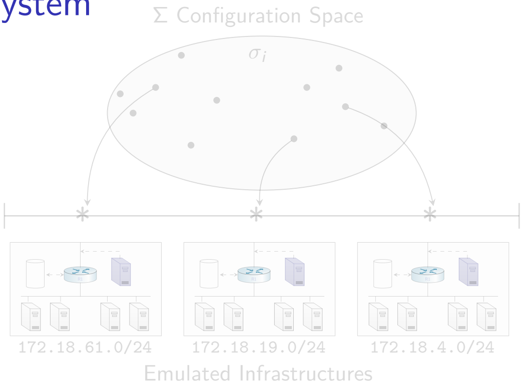
Model estimation

Real world Infrastructure



Automation & Self-learning systems

Emulation System

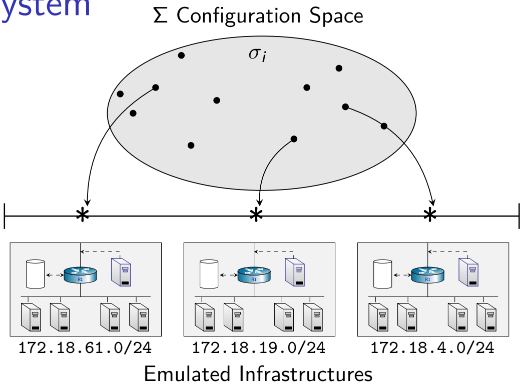


Emulation

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

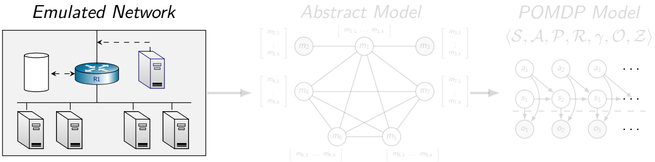


Emulation

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 $\sigma_i \in \Sigma$.

From Emulation to Simulation: System Identification

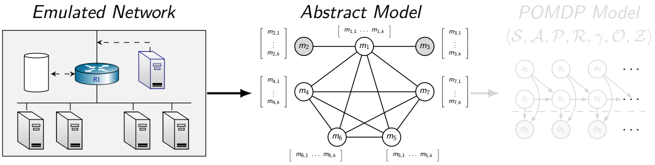


- **Abstract Model Based on Domain Knowledge**: Models the set of *controls*, the *objective function*, and the *features* of the emulated network.
  - Defines the static parts a POMDP model.

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

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

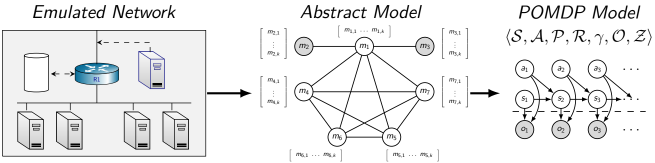
From Emulation to Simulation: System Identification



- **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

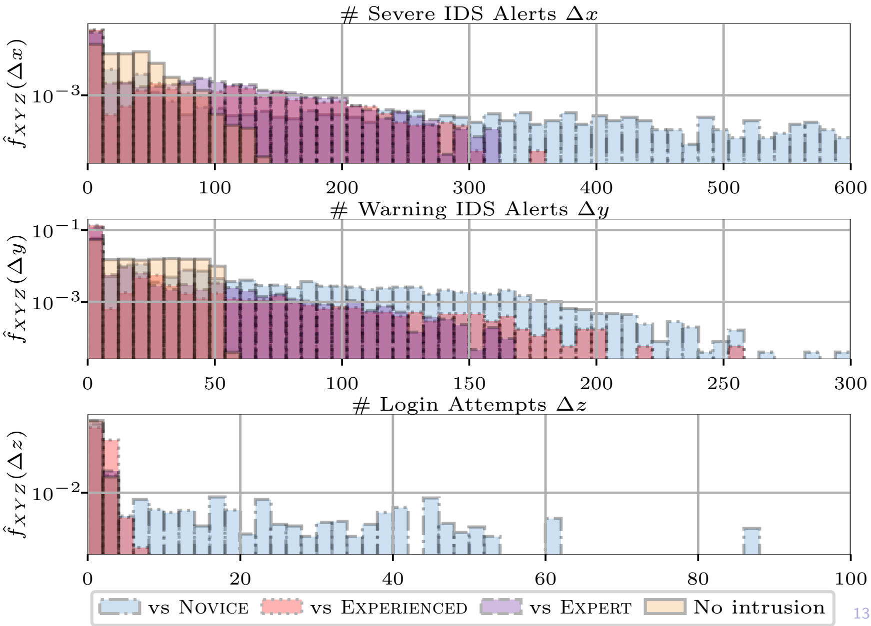


- **Abstract Model Based on Domain Knowledge**: Models the set of *controls*, the *objective function*, and the *features* of the emulated network.
  - Defines the static parts a POMDP model.

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

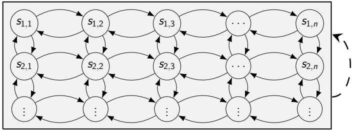
$$\mathcal{M}(b'|b,a) \triangleq \frac{n(b,a,b')}{\sum_{j'} n(s,a,j')}$$

System Identification: Estimated Empirical Distributions



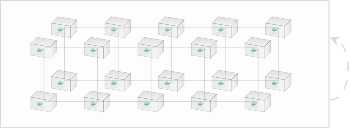
Our Method for Finding Effective Security Strategies

SIMULATION SYSTEM



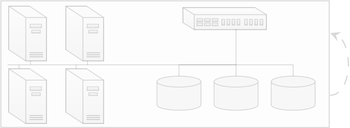
Reinforcement Learning & POMDP Model

EMULATION SYSTEM



Model estimation

REAL WORLD INFRASTRUCTURE



Automation & Self-learning systems

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

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

- **Method**:
  1. Simulate a series of POMDP episodes
  2. Use episode outcomes and trajectories to estimate $\nabla_\theta J(\theta)$
  3. Update policy π_θ with stochastic gradient ascent
  4. Continue until convergence

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

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

- **Method**:
  1. Simulate a series of POMDP episodes
  2. Use episode outcomes and trajectories to estimate $\nabla_\theta J(\theta)$
  3. Update policy π_θ with stochastic gradient ascent
  4. Continue until convergence

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

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

- **Method**:
  1. Simulate a series of POMDP episodes
  2. Use episode outcomes and trajectories to estimate $\nabla_\theta J(\theta)$
  3. Update policy π_θ with stochastic gradient ascent
  4. Continue until convergence

Policy Optimization in the Simulation System using Reinforcement Learning

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

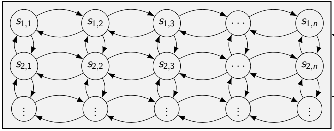
- **Learning Algorithm**:
  - Represent π by π_θ
  - Define objective $J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T} \gamma^{t-1} r(s_t, a_t)\right]$
  - Maximize $J(\theta)$ by stochastic gradient ascent

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

- **Method**:
  1. Simulate a series of POMDP episodes
  2. Use episode outcomes and trajectories to estimate $\nabla_\theta J(\theta)$
  3. Update policy π_θ with stochastic gradient ascent
  4. Continue until convergence

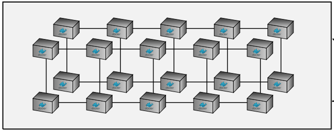
Our Method for Finding Effective Security Strategies

Simulation System



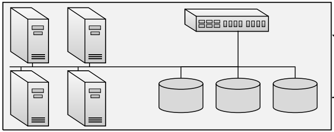
Reinforcement Learning & POMDP Model

Emulation System



Model estimation

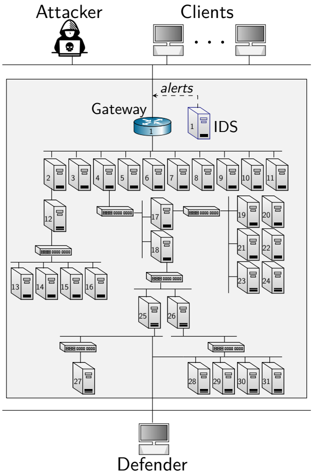
Real world Infrastructure



Automation & Self-learning systems

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



Target infrastructure.

Emulating the Client Population

Client	*Functions*	*Application servers*
1	HTTP, SSH, SNMP, ICMP	N_2, N_3, N_{10}, N_{12}
2	IRC, PostgreSQL, SNMP	$N_{31}, N_{13}, N_{14}, N_{15}, N_{16}$
3	FTP, DNS, Telnet	N_{10}, N_{22}, N_4

Table 1: Emulated client population; each client interacts with application servers using a set of functions at short intervals.

Emulating the Defender's Actions

Action	*Command in the Emulation*
Stop	`iptables -A INPUT -i eth0 -j DROP`

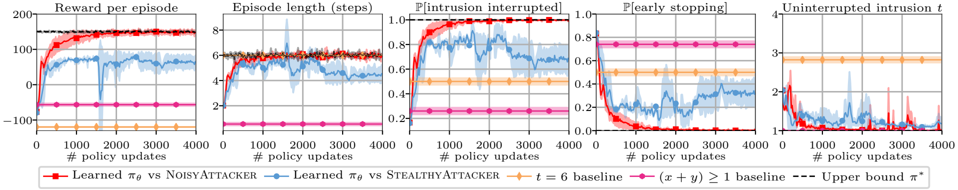
Table 2: Command used to implement the defender's stop action.

Emulating the Attacker's Actions

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})

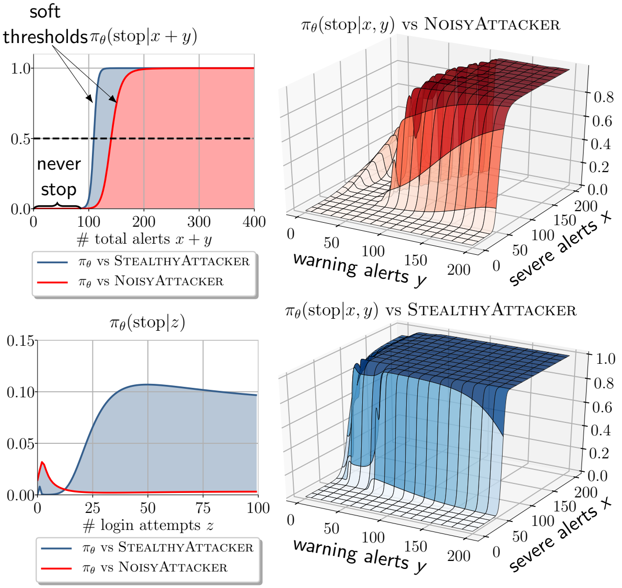
Table 3: Attacker actions to emulate an intrusion.

Learning Intrusion Prevention Policies through Optimal Stopping



Learning curves of training defender policies against static attackers.

Threshold Properties of the Learned Policies



Conclusions & Future Work

- **Conclusions:**
  - We develop a *method* to find learn intrusion prevention policies
    - (1) emulation system; (2) system identification; (3) simulation system; (4) reinforcement learning and (5) domain randomization and generalization.
  - We formulate intrusion prevention as a optimal stopping problem
    - We present a POMDP model of the use case
    - We apply the stopping theory to establish structural results of the optimal policy
- **Our research plans:**
  - Extending the theoretical model
    - Relaxing simplifying assumptions (e.g. more dynamic defender actions)
    - Active attacker
    - Multiple stops
  - Evaluation on real world infrastructures