# Learning Optimal Intrusion Responses for IT Infrastructures via Decomposition

## Visit to Princeton University

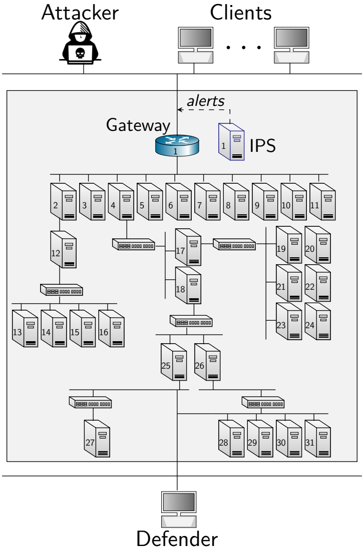
Kim Hammar

*kimham@kth.se*
Division of Network and Systems Engineering
KTH Royal Institute of Technology

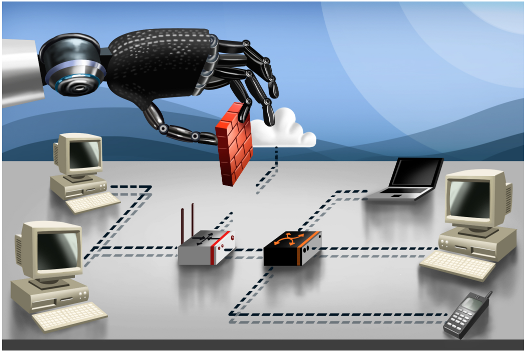
May 17, 2023

# Use Case: Intrusion Response

- 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



# Automated Intrusion Response: 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.

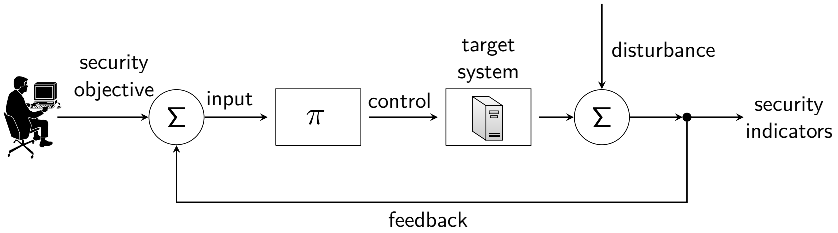
2000s-Now



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

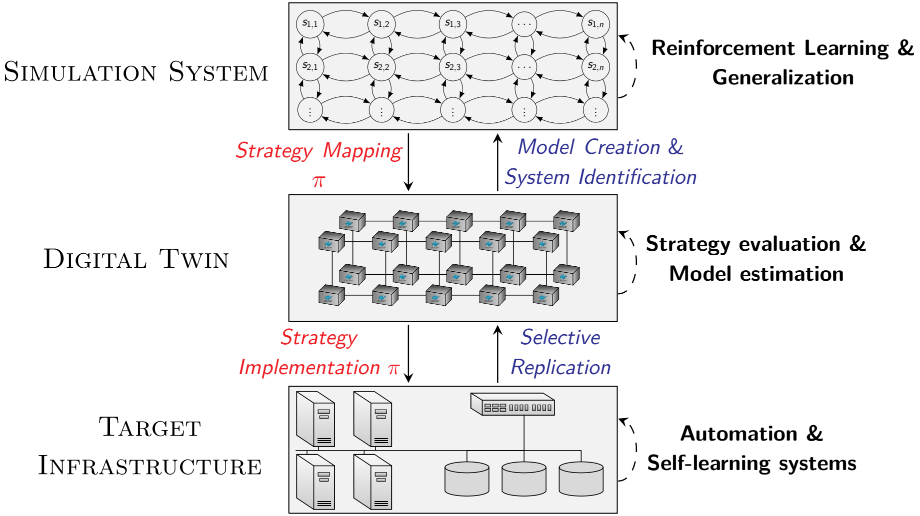
Research

*Can we use decision theory and learning-based methods to automatically find effective security strategies?*[1]

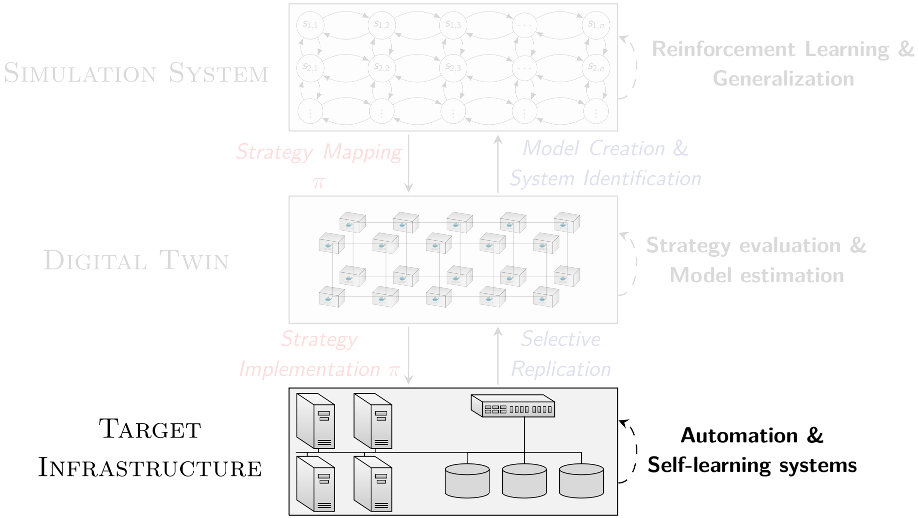



[1] Kim Hammar and Rolf Stadler. "Finding Effective Security Strategies through Reinforcement Learning and Self-Play". In: *International Conference on Network and Service Management (CNSM 2020)*. Izmir, Turkey, 2020, Kim Hammar and Rolf Stadler. "Learning Intrusion Prevention Policies through Optimal Stopping". In: *International Conference on Network and Service Management (CNSM 2021)*. Izmir, Turkey, 2021, 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, Kim Hammar and Rolf Stadler. *Learning Near-Optimal Intrusion Responses Against Dynamic Attackers*. 2023. DOI: 10.48550/ARXIV.2301.06085. URL: https://arxiv.org/abs/2301.06085.

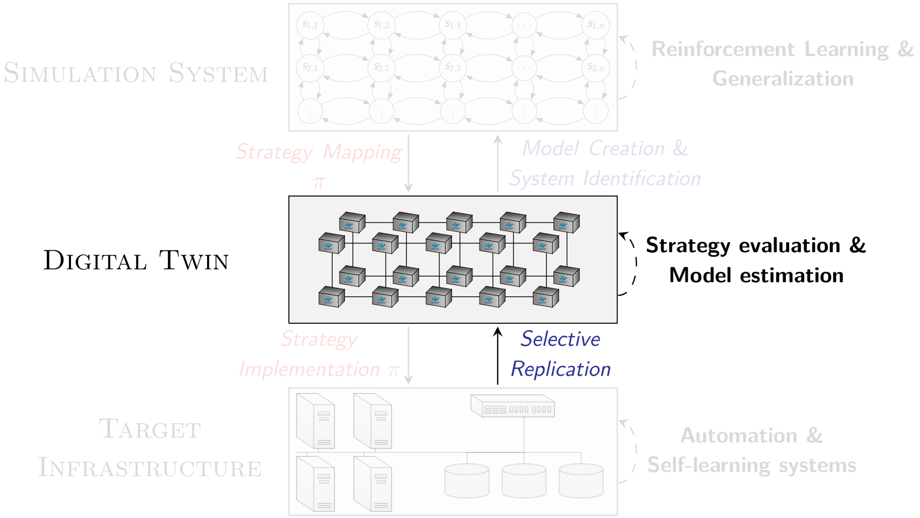
# Our Framework for Automated Network Security



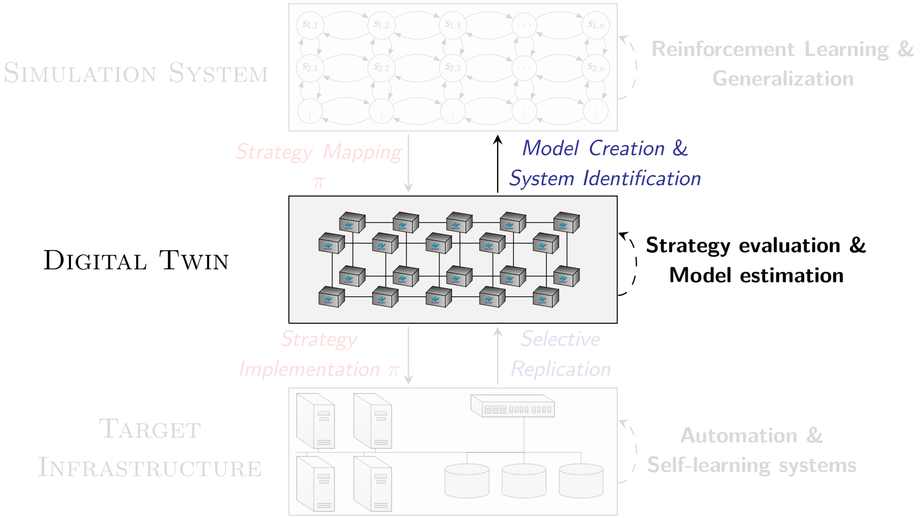
# Our Framework for Automated Network Security



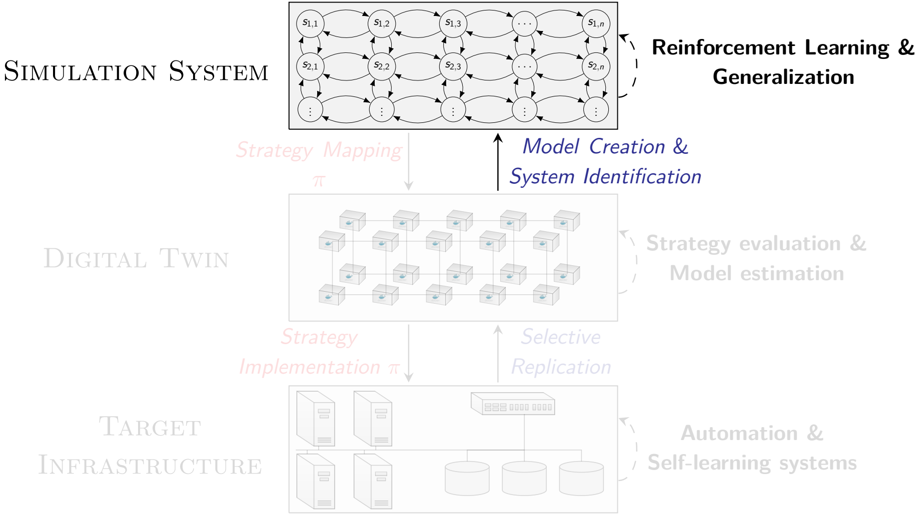
# Our Framework for Automated Network Security



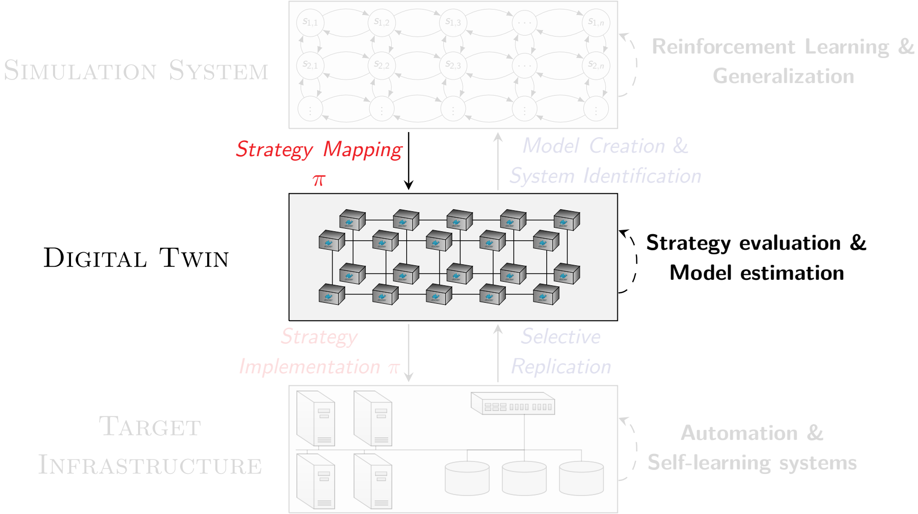
# Our Framework for Automated Network Security



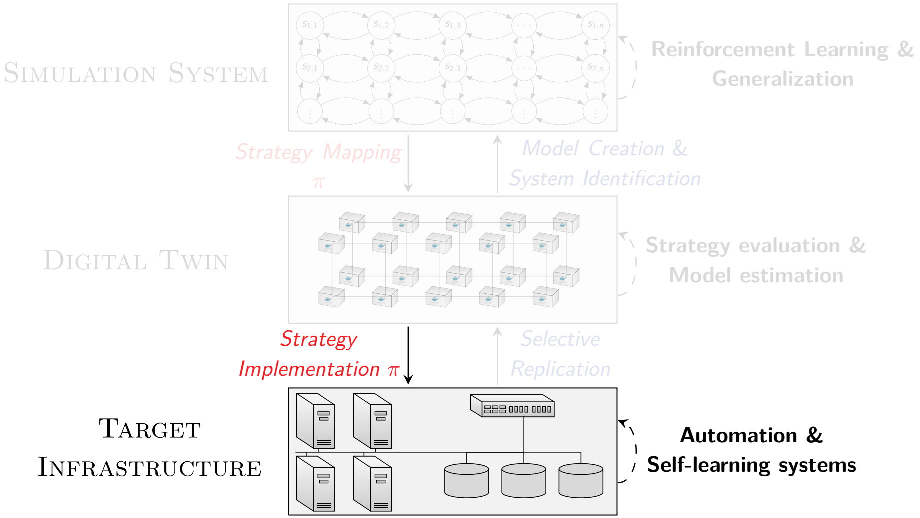
# Our Framework for Automated Network Security



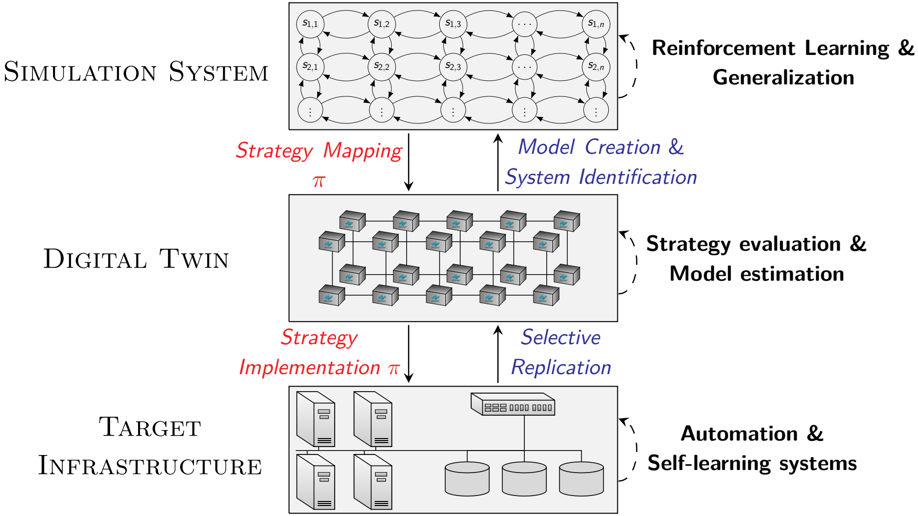
# Our Framework for Automated Network Security



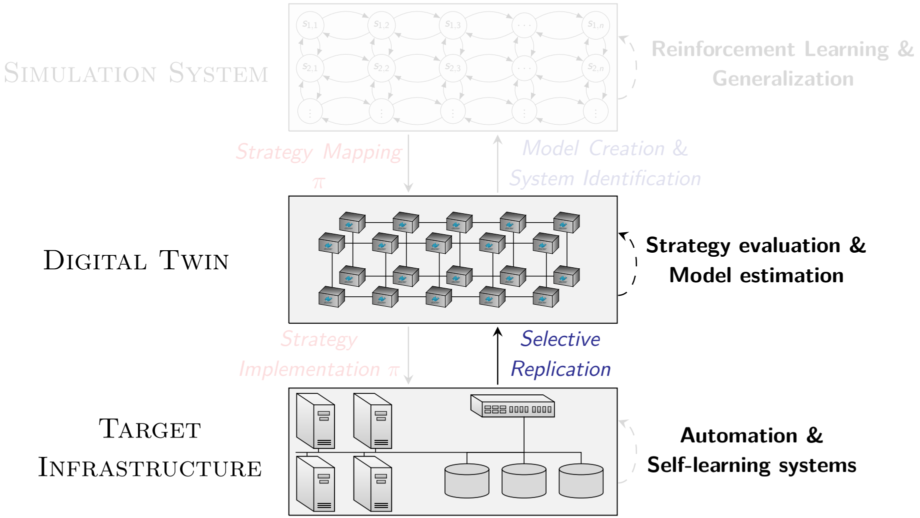
# Our Framework for Automated Network Security



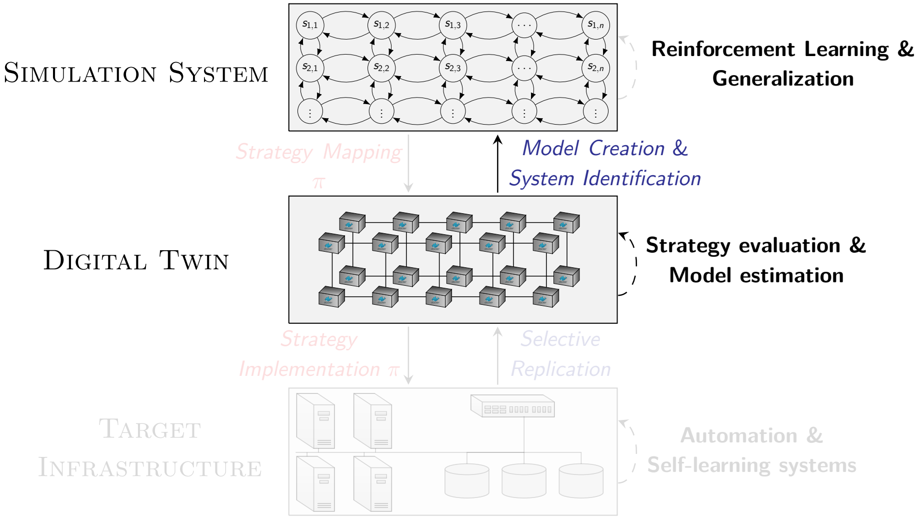
# Our Framework for Automated Network Security



# Creating a Digital Twin of the Target Infrastructure

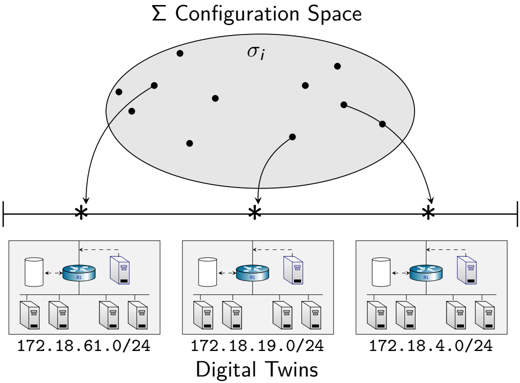


# Theoretical Analysis and Learning of Defender Strategies



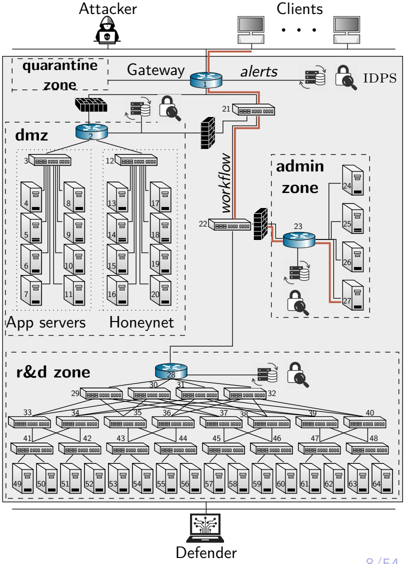
# Creating a Digital Twin of the Target Infrastructure

- An infrastructure is defined by its configuration.
- Set of configurations supported by our framework can be seen as a **configuration space**
- The configuration space defines the class of infrastructures for which we can create digital twins.



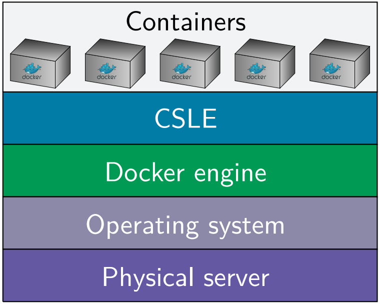
# The Target Infrastructure

- 64 **nodes.** 24 OVS switches, 3 gateways. 6 honeypots. 8 application servers. 4 administration servers. 15 compute servers.
- Topology shown to the right
- 11 vulnerabilities (CVE-2010-0426, CVE-2015-3306, CVE-2015-5602, etc.)
- 4 zones: DMZ, R&D ZONE, ADMIN ZONE, QUARANTINE ZONE
- 9 workflows
- Management: 1 SDN controller, 1 Kafka server, 1 elastic server.

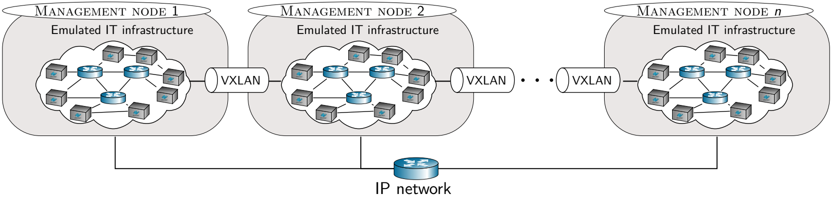


# Emulating Physical Components

- We emulate physical components with **Docker containers**
- Focus on linux-based systems
- The containers include everything needed to emulate the host: a runtime system, code, system tools, system libraries, and configurations.
- Examples of containers: IDPS container, client container, attacker container, CVE-2015-1427 container, Open vSwitch containers etc.



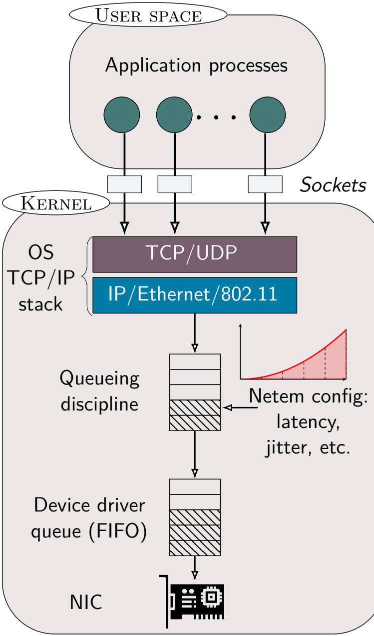
# Emulating Network Connectivity




- We emulate network connectivity on the same host using **network namespaces**.
- Connectivity across physical hosts is achieved using **VXLAN tunnels** with Docker swarm.

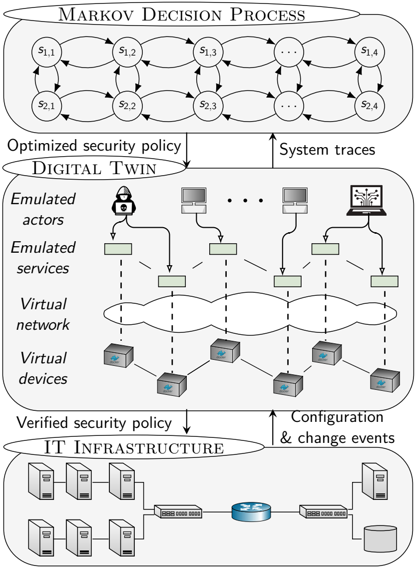
# Emulating Network Conditions

- We do traffic shaping using NetEm in the Linux kernel
- Emulate **internal connections** are full-duplex & loss-less with bit capacities of 1000 Mbit/s
- Emulate **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

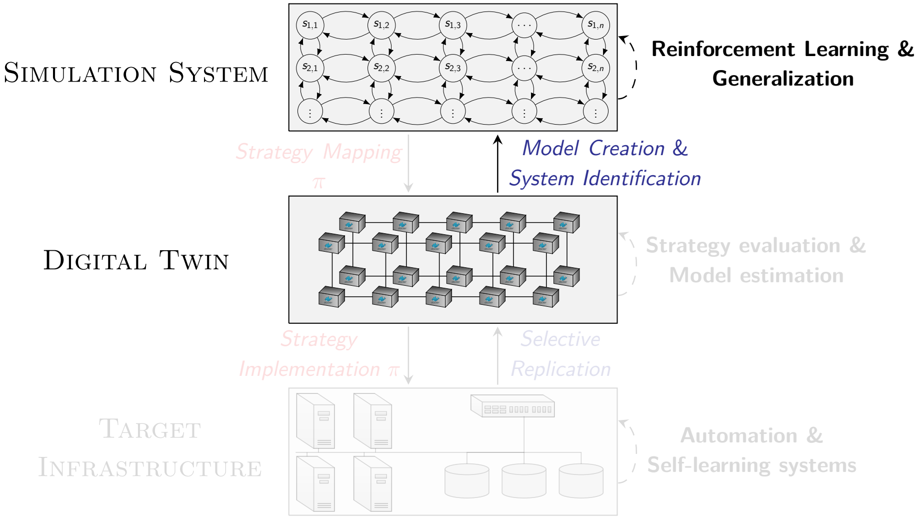


# Emulating Actors

- We emulate **client arrivals with Poisson processes**
- We emulate client interactions with load generators
- Attackers are emulated by automated programs that select actions from a pre-defined set
- Defender actions are emulated through a **custom gRPC API**.



# System Identification



# Outline

- **Use Case & Digital Twin**
  - Use case: intrusion response
  - Digital twin for data collection & evaluation
- **System Model**
  - Discrete-time Markovian dynamical system
  - Partially observed stochastic game
- **System Decomposition**
  - Additive subgames on the workflow-level
  - Optimal substructure on component-level
- **Learning Near-Optimal Intrusion Responses**
  - Scalable learning through decomposition
  - Digital twin for system identification & evaluation
  - Efficient equilibrium approximation
- **Conclusions & Future Work**

# Outline

- **Use Case & Digital Twin**
  - Use case: intrusion response
  - Digital twin for data collection & evaluation
- **System Model**
  - Discrete-time Markovian dynamical system
  - Partially observed stochastic game
- **System Decomposition**
  - Additive subgames on the workflow-level
  - Optimal substructure on component-level
- **Learning Near-Optimal Intrusion Responses**
  - Scalable learning through decomposition
  - Digital twin for system identification & evaluation
  - Efficient equilibrium approximation
- **Conclusions & Future Work**

# Outline

- **Use Case & Digital Twin**
  - Use case: intrusion response
  - Digital twin for data collection & evaluation
- **System Model**
  - Discrete-time Markovian dynamical system
  - Partially observed stochastic game
- **System Decomposition**
  - Additive subgames on the workflow-level
  - Optimal substructure on component-level
- **Learning Near-Optimal Intrusion Responses**
  - Scalable learning through decomposition
  - Digital twin for system identification & evaluation
  - Efficient equilibrium approximation
- **Conclusions & Future Work**

# Outline

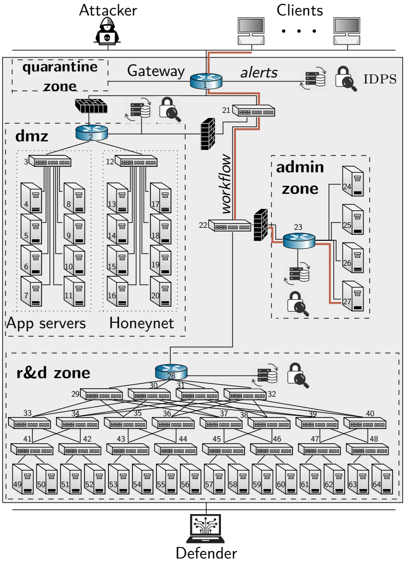
- **Use Case & Digital Twin**
  - Use case: intrusion response
  - Digital twin for data collection & evaluation
- **System Model**
  - Discrete-time Markovian dynamical system
  - Partially observed stochastic game
- **System Decomposition**
  - Additive subgames on the workflow-level
  - Optimal substructure on component-level
- **Learning Near-Optimal Intrusion Responses**
  - Scalable learning through decomposition
  - Digital twin for system identification & evaluation
  - Efficient equilibrium approximation
- **Conclusions & Future Work**

# Outline

- **Use Case & Digital Twin**
  - Use case: intrusion response
  - Digital twin for data collection & evaluation
- **System Model**
  - Discrete-time Markovian dynamical system
  - Partially observed stochastic game
- **System Decomposition**
  - Additive subgames on the workflow-level
  - Optimal substructure on component-level
- **Learning Near-Optimal Intrusion Responses**
  - Scalable learning through decomposition
  - Digital twin for system identification & evaluation
  - Efficient equilibrium approximation
- **Conclusions & Future Work**

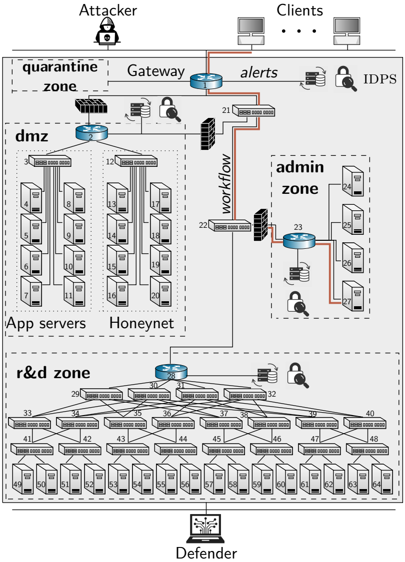
# System Model

- $\mathcal{G} = \langle \{\mathrm{gw}\} \cup \mathcal{V}, \mathcal{E} \rangle$: directed graph representing the virtual infrastructure
- $\mathcal{V}$: finite set of virtual components.
- $\mathcal{E}$: finite set of component dependencies.
- $\mathcal{Z}$: finite set of zones.



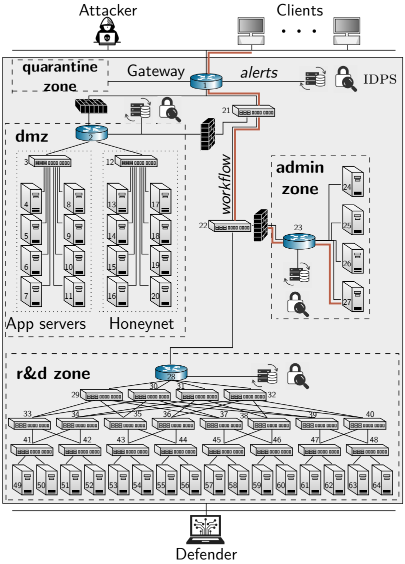
# System Model

- $\mathcal{G} = \langle \{\text{gw}\} \cup \mathcal{V}, \mathcal{E} \rangle$: directed graph representing the virtual infrastructure
- $\mathcal{V}$: finite set of virtual components.
- $\mathcal{E}$: finite set of component dependencies.
- $\mathcal{Z}$: finite set of zones.



# System Model

- $\mathcal{G} = \langle \{\text{gw}\} \cup \mathcal{V}, \mathcal{E} \rangle$: directed graph representing the virtual infrastructure
- $\mathcal{V}$: finite set of virtual components.
- $\mathcal{E}$: finite set of component dependencies.
- $\mathcal{Z}$: finite set of zones.



# State Model

- Each $i \in \mathcal{V}$ has a state

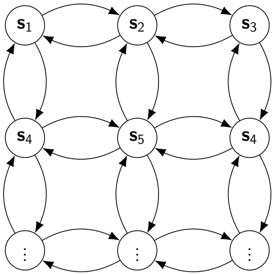
$$\mathbf{v}_{t,i} = (\underbrace{v_{t,i}^{(Z)}}_{\text{D}}, \underbrace{v_{t,i}^{(I)}, v_{t,i}^{(R)}}_{\text{A}})$$

- System state $\mathbf{s}_t = (\mathbf{v}_{t,i})_{i \in \mathcal{V}} \sim \mathbf{S}_t$.

- Markovian time-homogeneous dynamics:

$$\mathbf{s}_{t+1} \sim f(\cdot \mid \mathbf{S}_t, \mathbf{A}_t)$$

$\mathbf{A}_t = (\mathbf{A}_t^{(A)}, \mathbf{A}_t^{(D)})$ are the actions.



# State Model

- Each $i \in \mathcal{V}$ has a state

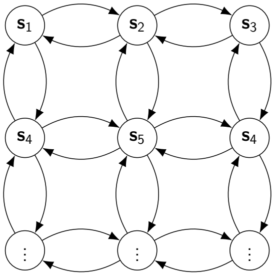
$$\mathbf{v}_{t,i} = (\underbrace{v_{t,i}^{(Z)}}_{\text{D}}, \underbrace{v_{t,i}^{(I)}, v_{t,i}^{(R)}}_{\text{A}})$$

- System state $\mathbf{s}_t = (\mathbf{v}_{t,i})_{i \in \mathcal{V}} \sim \mathbf{S}_t$.

- Markovian time-homogeneous dynamics:

$$\mathbf{s}_{t+1} \sim f(\cdot \mid \mathbf{S}_t, \mathbf{A}_t)$$

$\mathbf{A}_t = (\mathbf{A}_t^{(A)}, \mathbf{A}_t^{(D)})$ are the actions.



# State Model

- Each $i \in \mathcal{V}$ has a state

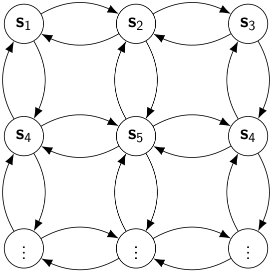
$$\mathbf{v}_{t,i} = (\underbrace{v_{t,i}^{(Z)}}_{\text{D}}, \underbrace{v_{t,i}^{(I)}, v_{t,i}^{(R)}}_{\text{A}})$$

- System state $\mathbf{s}_t = (\mathbf{v}_{t,i})_{i \in \mathcal{V}} \sim \mathbf{S}_t$.

- Markovian time-homogeneous dynamics:

$$\mathbf{s}_{t+1} \sim f(\cdot \mid \mathbf{S}_t, \mathbf{A}_t)$$

$\mathbf{A}_t = (\mathbf{A}_t^{(A)}, \mathbf{A}_t^{(D)})$ are the actions.

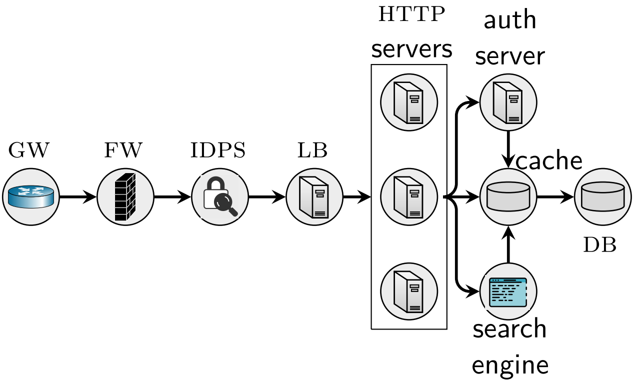


# Workflow Model

- Services are connected into **workflows** $\mathcal{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_{|\mathcal{W}|}\}$.

# Workflow Model

- Services are connected into **workflows** $\mathcal{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_{|\mathcal{W}|}\}$.

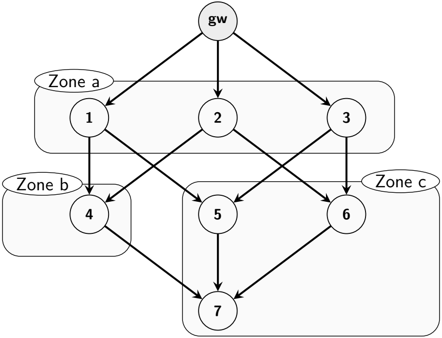



Dependency graph of an example workflow representing a web application; GW, FW, IDPS, LB, and DB are acronyms for gateway, firewall, intrusion detection and prevention system, load balancer, and database, respectively.

# Workflow Model

- Services are connected into **workflows** $\mathcal{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_{|\mathcal{W}|}\}$.
- Each $\mathbf{w} \in \mathcal{W}$ is realized as a directed acyclic subgraph (DAG) $\mathcal{G}_\mathbf{w} = \langle \{\text{gw}\} \cup \mathcal{V}_\mathbf{w}, \mathcal{E}_\mathbf{w} \rangle$ of $\mathcal{G}$
- $\mathcal{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_{|\mathcal{W}|}\}$ induces a partitioning

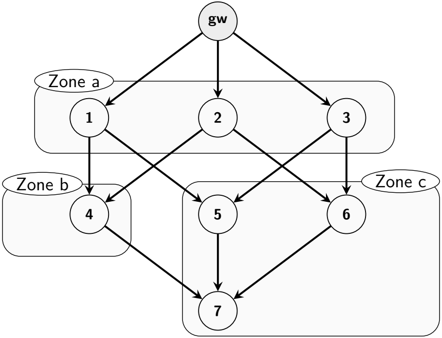
$$\mathcal{V} = \bigcup_{\mathbf{w}_i \in \mathcal{W}} \mathcal{V}_{\mathbf{w}_i} \text{ such that } i \neq j \implies \mathcal{V}_{\mathbf{w}_i} \cap \mathcal{V}_{\mathbf{w}_j} = \emptyset$$




A workflow DAG

# Workflow Model

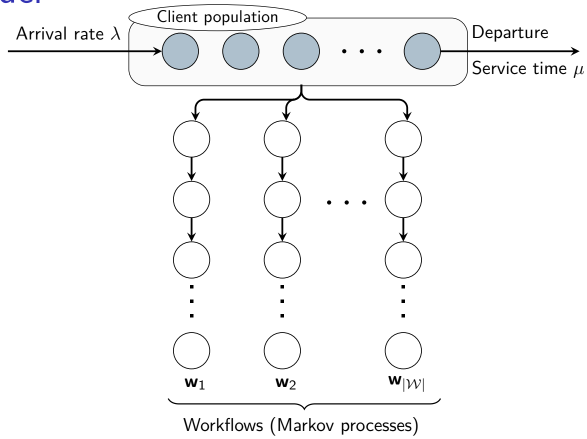
- Services are connected into **workflows** $\mathcal{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_{|\mathcal{W}|}\}$.
- Each $\mathbf{w} \in \mathcal{W}$ is realized as a directed acyclic subgraph (DAG) $\mathcal{G}_{\mathbf{w}} = \langle \{\mathrm{gw}\} \cup \mathcal{V}_{\mathbf{w}}, \mathcal{E}_{\mathbf{w}} \rangle$ of $\mathcal{G}$
- $\mathcal{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_{|\mathcal{W}|}\}$ induces a partitioning




A workflow DAG

$$\mathcal{V} = \bigcup_{\mathbf{w}_i \in \mathcal{W}} \mathcal{V}_{\mathbf{w}_i} \text{ such that } i \neq j \implies \mathcal{V}_{\mathbf{w}_i} \cap \mathcal{V}_{\mathbf{w}_j} = \emptyset$$

# Client Model




- Homogeneous client population
- Clients arrive according to $Po(\lambda)$, Service times $Exp(\frac{1}{\mu})$
- Workflow selection: uniform
- Workflow interaction: Markov process

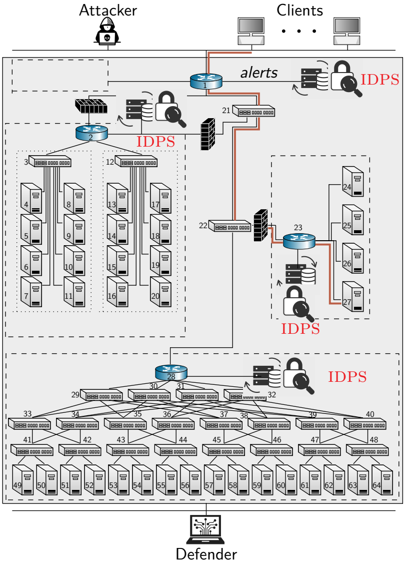
# Observation Model

- IDPSs inspect network traffic and generate alert vectors:

$$\mathbf{o}_t \triangleq \left(\mathbf{o}_{t,1}, \ldots, \mathbf{o}_{t,|\mathcal{V}|}\right) \in \mathbb{N}_0^{|\mathcal{V}|}$$

$\mathbf{o}_{t,i}$ is the number of alerts related to node $i \in \mathcal{V}$ at time-step $t$.

- $\mathbf{o}_t = \left(\mathbf{o}_{t,1}, \ldots, \mathbf{o}_{t,|\mathcal{V}|}\right)$ is a realization of the random vector $\mathbf{O}_t$ with joint distribution $Z$



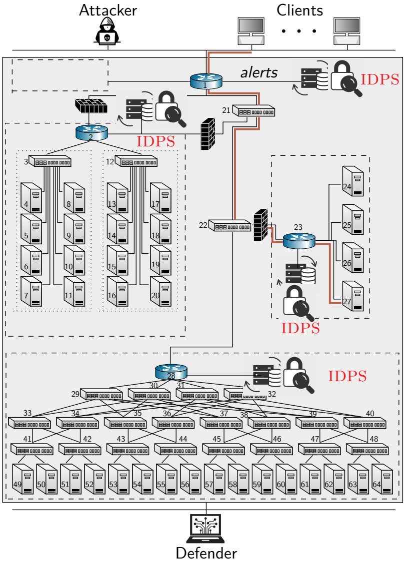
# Observation Model

- IDPSs inspect network traffic and generate alert vectors:

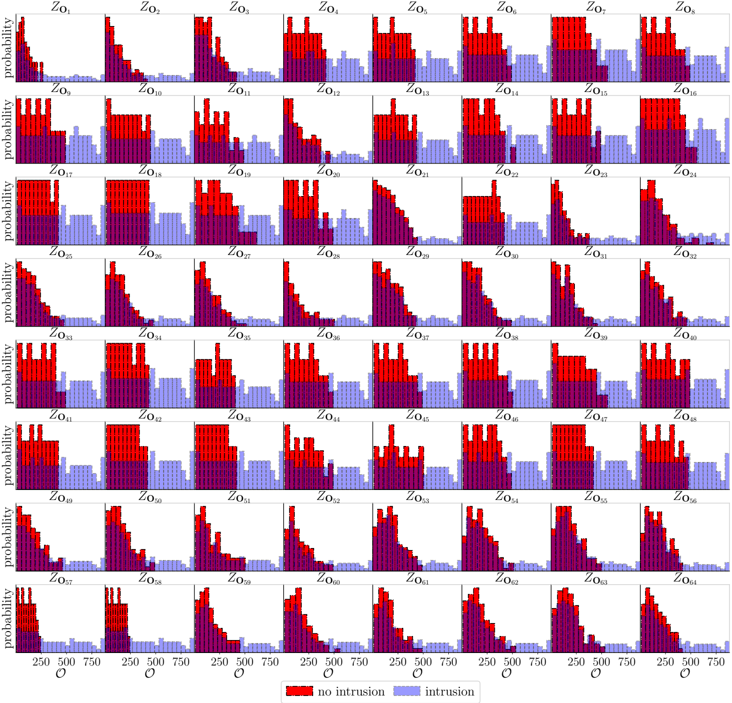
$$\mathbf{o}_t \triangleq \left(\mathbf{o}_{t,1}, \ldots, \mathbf{o}_{t,|\mathcal{V}|}\right) \in \mathbb{N}_0^{|\mathcal{V}|}$$

  $\mathbf{o}_{t,i}$ is the number of alerts related to node $i \in \mathcal{V}$ at time-step $t$.

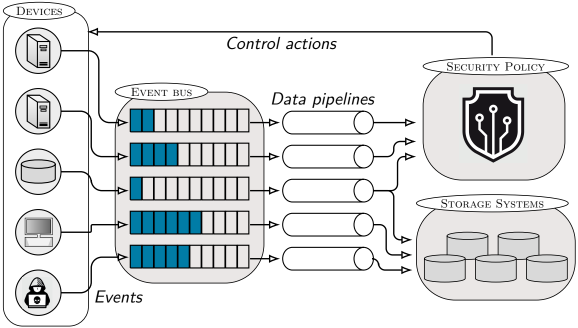
- $\mathbf{o}_t = \left(\mathbf{o}_{t,1}, \ldots, \mathbf{o}_{t,|\mathcal{V}|}\right)$ is a realization of the random vector $\mathbf{O}_t$ with joint distribution $Z$



Distributions of # alerts weighted by priority $Z_{\mathbf{O}_i}(\mathbf{O}_i \mid \mathbf{S}_i^{(\mathrm{D})}, \mathbf{A}_i^{(\mathrm{A})})$ per node $i \in \mathcal{V}$



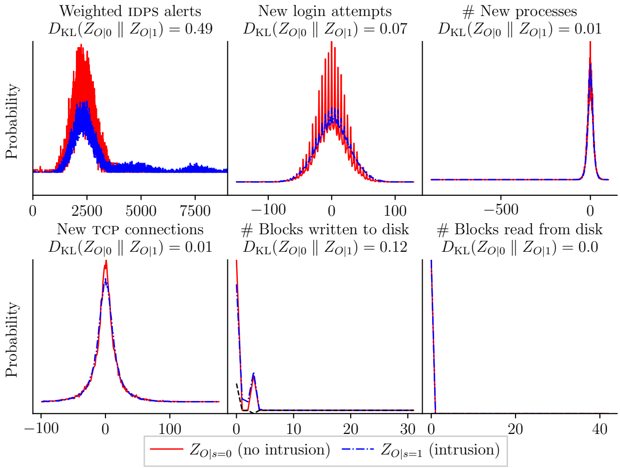
# Monitoring and Telemetry




- Emulated devices run monitoring agents that **periodically push metrics to a Kafka event bus**.
- The data in the event bus is consumed by data pipelines that process the data and write to storage systems.
- The processed data is used by an automated security policy to decide on control actions to execute in the digital twin.

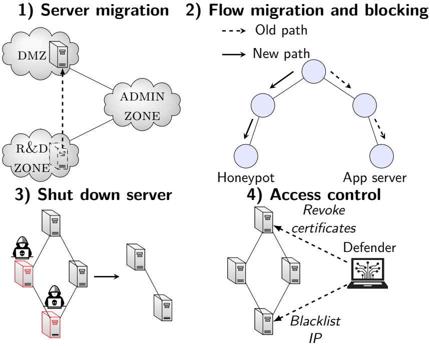
# Feature Selection

- Our framework collects 100s of metrics every time-step.
- We focus on the **IDPS alert metric** as it provides the most information for detecting the type of attacks we consider.



# Defender Model

- Defender action: $\mathbf{a}_t^{(D)} \in \{0,1,2,3,4\}^{|\mathcal{V}|}$
- 0 means do nothing. $1-4$ correspond to defensive actions (see fig)




- A **defender strategy** is a function $\pi_{\mathrm{D}} \in \Pi_{\mathrm{D}} : \mathcal{H}_{\mathrm{D}} \to \Delta(\mathcal{A}_{\mathrm{D}})$, where

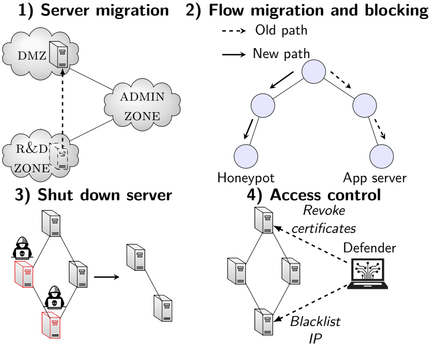
$$\mathbf{h}_t^{(\mathrm{D})} = (\mathbf{s}_1^{(\mathrm{D})}, \mathbf{a}_1^{(\mathrm{D})}, \mathbf{o}_1, \ldots, \mathbf{a}_{t-1}^{(\mathrm{D})}, \mathbf{s}_t^{(\mathrm{D})}, \mathbf{o}_t) \in \mathcal{H}_{\mathrm{D}}$$

- Objective: ($i$) maintain workflows; and ($ii$) stop a possible intrusion:

$$J \triangleq \sum_{t=1}^{T} \gamma^{t-1} \left( \underbrace{\eta \sum_{i=1}^{|\mathcal{W}|} u_{\mathrm{W}}(\mathbf{w}_i, \mathbf{s}_t)}_{\text{workflows utility}} - \underbrace{(1-\eta) \sum_{j=1}^{|\mathcal{V}|} c_{\mathrm{I}}(\mathbf{s}_{t,j}, \mathbf{a}_{t,j})}_{\text{intrusion and defense costs}} \right)$$

# Defender Model

- Defender action:
  $\mathbf{a}_t^{(D)} \in \{0,1,2,3,4\}^{|\mathcal{V}|}$
- 0 means do nothing. $1-4$ correspond to defensive actions (see fig)




- A **defender strategy** is a function $\pi_{\mathrm{D}} \in \Pi_{\mathrm{D}} : \mathcal{H}_{\mathrm{D}} \to \Delta(\mathcal{A}_{\mathrm{D}})$, where

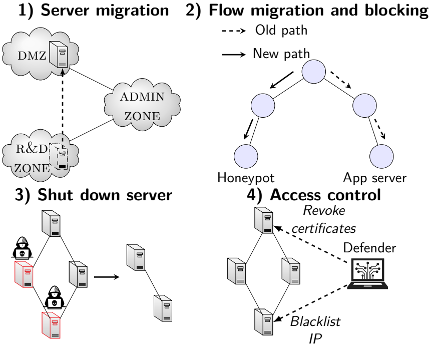
$$\mathbf{h}_t^{(\mathrm{D})} = (\mathbf{s}_1^{(\mathrm{D})}, \mathbf{a}_1^{(\mathrm{D})}, \mathbf{o}_1, \ldots, \mathbf{a}_{t-1}^{(\mathrm{D})}, \mathbf{s}_t^{(\mathrm{D})}, \mathbf{o}_t) \in \mathcal{H}_{\mathrm{D}}$$

- Objective: $(i)$ maintain workflows; and $(ii)$ stop a possible intrusion:

$$J \triangleq \sum_{t=1}^{T} \gamma^{t-1} \left( \underbrace{\eta \sum_{i=1}^{|\mathcal{W}|} u_{\mathrm{W}}(\mathbf{w}_i, \mathbf{s}_t)}_{\text{workflows utility}} - \underbrace{(1-\eta) \sum_{j=1}^{|\mathcal{V}|} c_{\mathrm{I}}(\mathbf{s}_{t,j}, \mathbf{a}_{t,j})}_{\text{intrusion and defense costs}} \right)$$

# Defender Model

- Defender action:
  $\mathbf{a}_t^{(D)} \in \{0,1,2,3,4\}^{|\mathcal{V}|}$
- 0 means do nothing. $1-4$ correspond to defensive actions (see fig)




- A **defender strategy** is a function $\pi_{\mathrm{D}} \in \Pi_{\mathrm{D}} : \mathcal{H}_{\mathrm{D}} \to \Delta(\mathcal{A}_{\mathrm{D}})$, where

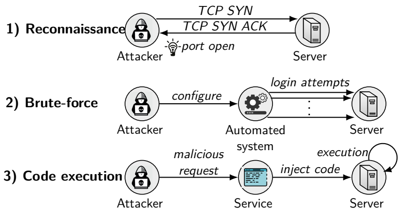
$$\mathbf{h}_t^{(\mathrm{D})} = (\mathbf{s}_1^{(\mathrm{D})}, \mathbf{a}_1^{(\mathrm{D})}, \mathbf{o}_1, \ldots, \mathbf{a}_{t-1}^{(\mathrm{D})}, \mathbf{s}_t^{(\mathrm{D})}, \mathbf{o}_t) \in \mathcal{H}_{\mathrm{D}}$$

- Objective: $(i)$ maintain workflows; and $(ii)$ stop a possible intrusion:

$$J \triangleq \sum_{t=1}^{T} \gamma^{t-1} \Bigg( \underbrace{\eta \sum_{i=1}^{|\mathcal{W}|} u_{\mathrm{W}}(\mathbf{w}_i, \mathbf{s}_t)}_{\text{workflows utility}} - \underbrace{(1-\eta) \sum_{j=1}^{|\mathcal{V}|} c_{\mathrm{I}}(\mathbf{s}_{t,j}, \mathbf{a}_{t,j})}_{\text{intrusion and defense costs}} \Bigg)$$

# Attacker Model

- Attacker action: $\mathbf{a}_t^{(A)} \in \{0,1,2,3\}^{|\mathcal{V}|}$
- 0 means do nothing. $1-3$ correspond to attacks (see fig)




- An **attacker strategy** is a function $\pi_{\mathrm{A}} \in \Pi_{\mathrm{A}} : \mathcal{H}_{\mathrm{A}} \rightarrow \Delta(\mathcal{A}_{\mathrm{A}})$, where $\mathcal{H}_{\mathrm{A}}$ is the space of all possible attacker histories

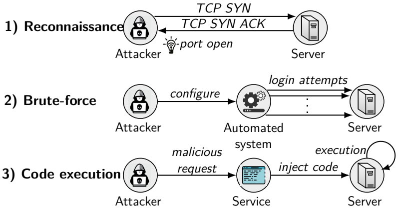
$$\mathbf{h}_t^{(\mathrm{A})} = (\mathbf{s}_1^{(\mathrm{A})}, \mathbf{a}_1^{(\mathrm{A})}, \mathbf{o}_1, \ldots, \mathbf{a}_{t-1}^{(\mathrm{A})}, \mathbf{s}_t^{(\mathrm{A})}, \mathbf{o}_t) \in \mathcal{H}_{\mathrm{A}}$$

- Objective: (*i*) disrupt workflows; and (*ii*) **compromise nodes**:

$$-J$$

# Attacker Model

- Attacker action: $\mathbf{a}_t^{(A)} \in \{0,1,2,3\}^{|\mathcal{V}|}$
- 0 means do nothing. $1-3$ correspond to attacks (see fig)




- An **attacker strategy** is a function $\pi_{\mathrm{A}} \in \Pi_{\mathrm{A}} : \mathcal{H}_{\mathrm{A}} \to \Delta(\mathcal{A}_{\mathrm{A}})$, where $\mathcal{H}_{\mathrm{A}}$ is the space of all possible attacker histories

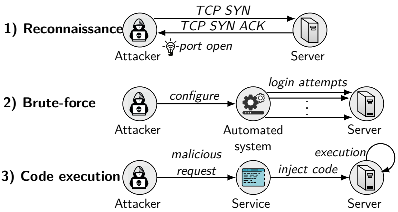
$$\mathbf{h}_t^{(\mathrm{A})} = (\mathbf{s}_1^{(\mathrm{A})}, \mathbf{a}_1^{(\mathrm{A})}, \mathbf{o}_1, \ldots, \mathbf{a}_{t-1}^{(\mathrm{A})}, \mathbf{s}_t^{(\mathrm{A})}, \mathbf{o}_t) \in \mathcal{H}_{\mathrm{A}}$$

- Objective: $(i)$ disrupt workflows; and $(ii)$ **compromise nodes**:

$$-J$$

# Attacker Model

- Attacker action: $\mathbf{a}_t^{(A)} \in \{0,1,2,3\}^{|\mathcal{V}|}$
- 0 means do nothing. $1-3$ correspond to attacks (see fig)




- An **attacker strategy** is a function $\pi_{\mathrm{A}} \in \Pi_{\mathrm{A}} : \mathcal{H}_{\mathrm{A}} \to \Delta(\mathcal{A}_{\mathrm{A}})$, where $\mathcal{H}_{\mathrm{A}}$ is the space of all possible attacker histories

$$\mathbf{h}_t^{(\mathrm{A})} = (\mathbf{s}_1^{(\mathrm{A})}, \mathbf{a}_1^{(\mathrm{A})}, \mathbf{o}_1, \ldots, \mathbf{a}_{t-1}^{(\mathrm{A})}, \mathbf{s}_t^{(\mathrm{A})}, \mathbf{o}_t) \in \mathcal{H}_{\mathrm{A}}$$

- Objective: $(i)$ disrupt workflows; and $(ii)$ **compromise nodes**:

$$-J$$

# The Intrusion Response Problem

$$
\begin{aligned}
\underset{\pi_{\mathrm{D}} \in \Pi_{\mathrm{D}}}{\text{maximize}} \ \underset{\pi_{\mathrm{A}} \in \Pi_{\mathrm{A}}}{\text{minimize}} \ & \mathbb{E}_{(\pi_{\mathrm{D}}, \pi_{\mathrm{A}})}\left[J\right] && && \text{(1a)} \\
\text{subject to } & \mathbf{s}^{(\mathrm{D})}_{t+1} \sim f_{\mathrm{D}}\big(\cdot \mid \mathbf{A}^{(\mathrm{D})}_t, \mathbf{A}^{(\mathrm{D})}_t\big) && \forall t && \text{(1b)} \\
& \mathbf{s}^{(\mathrm{A})}_{t+1} \sim f_{\mathrm{A}}\big(\cdot \mid \mathbf{S}^{(\mathrm{A})}_t, \mathbf{A}_t\big) && \forall t && \text{(1c)} \\
& \mathbf{o}_{t+1} \sim Z\big(\cdot \mid \mathbf{S}^{(\mathrm{D})}_{t+1}, \mathbf{A}^{(\mathrm{A})}_t\big) && \forall t && \text{(1d)} \\
& \mathbf{a}^{(\mathrm{A})}_t \sim \pi_{\mathrm{A}}\big(\cdot \mid \mathbf{H}^{(\mathrm{A})}_t\big), \ \mathbf{a}^{(\mathrm{A})}_t \in \mathcal{A}_{\mathrm{A}}(\mathbf{s}_t) && \forall t && \text{(1e)} \\
& \mathbf{a}^{(\mathrm{D})}_t \sim \pi_{\mathrm{D}}\big(\cdot \mid \mathbf{H}^{(\mathrm{D})}_t\big), \ \mathbf{a}^{(\mathrm{D})}_t \in \mathcal{A}_{\mathrm{D}} && \forall t && \text{(1f)}
\end{aligned}
$$

where $\mathbb{E}_{(\pi_{\mathrm{D}}, \pi_{\mathrm{A}})}$ denotes the expectation of the random vectors $(\mathbf{S}_t, \mathbf{O}_t, \mathbf{A}_t)_{t \in \{1,\ldots,T\}}$ under the strategy profile $(\pi_{\mathrm{D}}, \pi_{\mathrm{A}})$.

(1) can be formulated as a zero-sum Partially Observed Stochastic Game with Public Observations (a PO-POSG):

$$
\Gamma = \langle \mathcal{N}, (\mathcal{S}_i)_{i \in \mathcal{N}}, (\mathcal{A}_i)_{i \in \mathcal{N}}, (f_i)_{i \in \mathcal{N}}, u, \gamma, (\mathbf{b}^{(i)}_1)_{i \in \mathcal{N}}, \mathcal{O}, Z \rangle
$$

# Existence of a Solution

**Theorem**

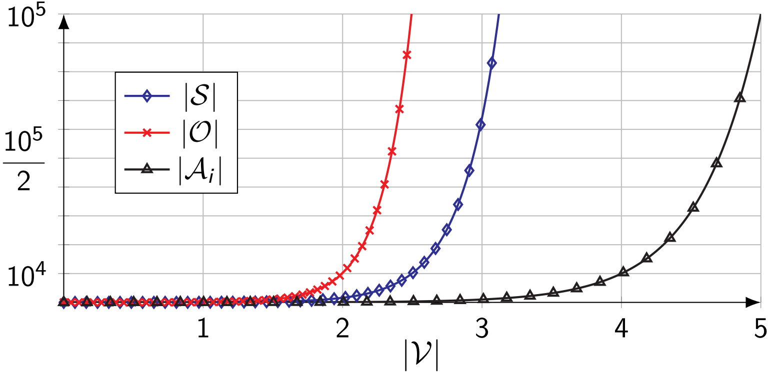
*Given the* PO-POSG $\Gamma$ *(2), the following holds:*

(A) $\Gamma$ *has a mixed Nash equilibrium and a value function* $V^* : \mathcal{B}_{\mathrm{D}} \times \mathcal{B}_{\mathrm{A}} \rightarrow \mathbb{R}$ *that maps each possible initial pair of belief states* $(\mathbf{b}_1^{(\mathrm{D})}, \mathbf{b}_1^{\mathrm{A}})$ *to the expected utility of the defender in the equilibrium.*

(B) *For each strategy pair* $(\pi_{\mathrm{A}}, \pi_{\mathrm{D}}) \in \Pi_{\mathrm{A}} \times \Pi_{\mathrm{D}}$*, the best response sets* $B_{\mathrm{D}}(\pi_{\mathrm{A}})$ *and* $B_{\mathrm{A}}(\pi_{\mathrm{D}})$ *are non-empty and correspond to optimal strategies in two Partially Observed Markov Decision Processes (*POMDPs*):* $\mathscr{M}^{(\mathrm{D})}$ *and* $\mathscr{M}^{(\mathrm{A})}$*. Further, a pair of pure best response strategies* $(\tilde{\pi}_{\mathrm{D}}, \tilde{\pi}_{\mathrm{A}}) \in B_{\mathrm{D}}(\pi_{\mathrm{A}}) \times B_{\mathrm{A}}(\pi_{\mathrm{D}})$ *and a pair of value functions* $(V^*_{\mathrm{D},\pi_{\mathrm{A}}}, V^*_{\mathrm{A},\pi_{\mathrm{D}}})$ *exist.*

# The Curse of Dimensionality

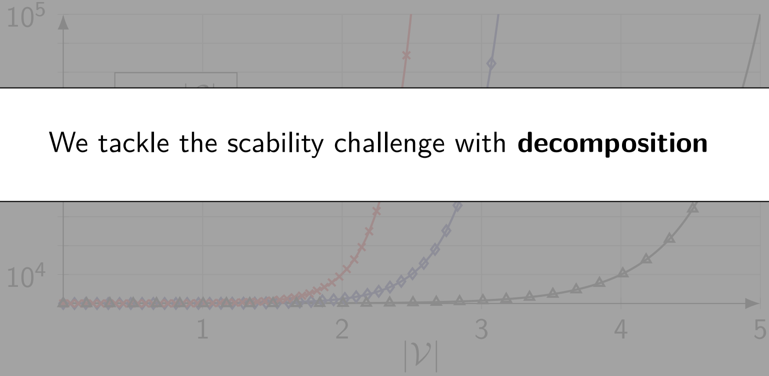
- While (1) has a solution (i.e the game $\Gamma$ has a value (Thm 1)), computing it is intractable since the state, action, and observation spaces of the game **grow exponentially** with $|\mathcal{V}|$.




Growth of $|\mathcal{S}|$, $|\mathcal{O}|$, and $|\mathcal{A}_i|$ in function of the number of nodes $|\mathcal{V}|$

# The Curse of Dimensionality

- While (1) has a solution (i.e the game $\Gamma$ has a value (Thm 1)), computing it is intractable since the state, action, and observation spaces of the game **grow exponentially** with $|\mathcal{V}|$.

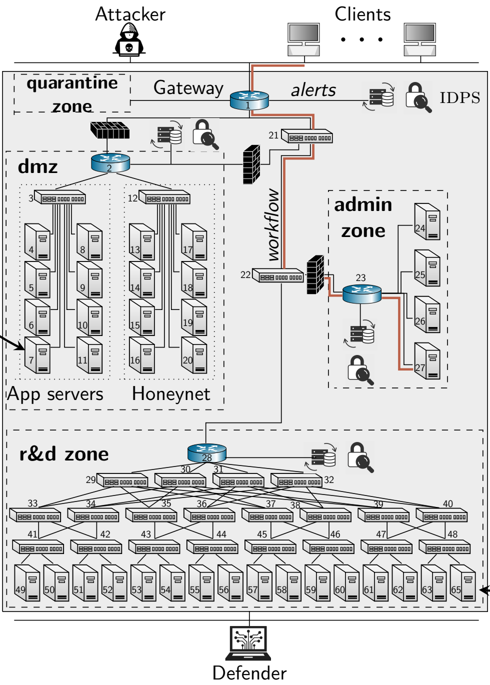



We tackle the scability challenge with **decomposition**

Growth of $|\mathcal{S}|$, $|\mathcal{O}|$, and $|\mathcal{A}_i|$ in function of the number of nodes $|\mathcal{V}|$

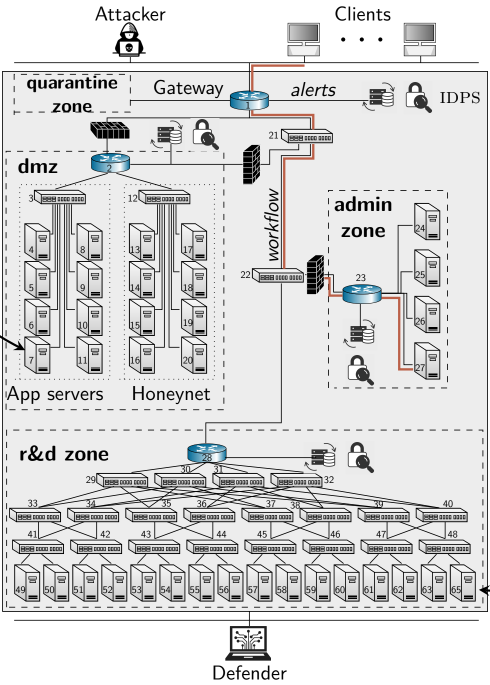
# Intuitively..

The optimal action here...




Does not directly depend on the state or action of a node down here

# Intuitively..




The optimal action here...

But they are not completely independent either.

How can we exploit this structure?

Does not directly depend on the state or action of a node down here

# System Decomposition

To avoid explicitly enumerating the very large state, observation, and action spaces of $\Gamma$, we exploit three structural properties.

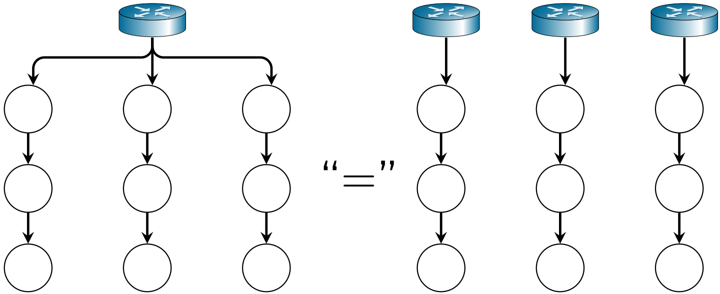
1. **Additive structure across workflows.**
   - The game decomposes into additive subgames on the workflow-level, which means that the strategy for each subgame can be optimized independently
2. **Optimal substructure within a workflow.**
   - The subgame for each workflow decomposes into subgames on the node-level that satisfy the *optimal substructure* property
3. **Threshold properties of local defender strategies.**
   - The optimal node-level strategies for the defender exhibit threshold structures, which means that they can be estimated efficiently

# System Decomposition

To avoid explicitly enumerating the very large state, observation, and action spaces of $\Gamma$, we exploit three structural properties.

1. **Additive structure across workflows.**
   - The game decomposes into additive subgames on the workflow-level, which means that the strategy for each subgame can be optimized independently
2. **Optimal substructure within a workflow.**
   - The subgame for each workflow decomposes into subgames on the node-level that satisfy the *optimal substructure* property
3. **Threshold properties of local defender strategies.**
   - The optimal node-level strategies for the defender exhibit threshold structures, which means that they can be estimated efficiently

# Additive Structure Across Workflows (Intuition)




- If there is no path between $i$ and $j$ in $\mathcal{G}$, then $i$ and $j$ are **independent** in the following sense:
  - Compromising $i$ has no affect on the state of $j$.
  - Compromising $i$ does not make it harder or easier to compromise $j$.
  - Compromising $i$ does not affect the service provided by $j$.
  - Defending $i$ does not affect the state of $j$.
  - Defending $i$ does not affect the service provided by $j$.

# Additive Structure Across Workflows

## Definition (Transition independence)

A set of nodes $\mathcal{Q}$ are transition independent iff the transition probabilities factorize as

$$f(\mathbf{S}_{t+1} \mid \mathbf{S}_t, \mathbf{A}_t) = \prod_{i \in \mathcal{Q}} f(\mathbf{S}_{t+1,i} \mid \mathbf{S}_{t,i}, \mathbf{A}_{t,i})$$

## Definition (Utility independence)

A set of nodes $\mathcal{Q}$ are utility independent iff there exists functions $u_1, \ldots, u_{|\mathcal{Q}|}$ such that the utility function $u$ decomposes as

$$u(\mathbf{S}_t, \mathbf{A}_t) = f(u_1(\mathbf{S}_{t,1}, \mathbf{A}_{t,1}), \ldots, u_1(\mathbf{S}_{t,|\mathcal{Q}|}, \mathbf{A}_{t,\mathcal{Q}}))$$

and

$$u_i \leq u_i' \iff f(u_1, \ldots, u_i, \ldots, u_{|\mathcal{Q}|}) \leq f(u_1, \ldots, u_i', \ldots, u_{|\mathcal{Q}|})$$

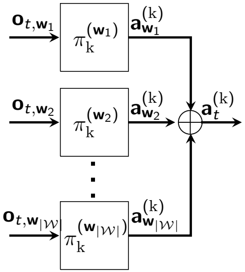
# Additive Structure Across Workflows

## Theorem (Additive structure across workflows)

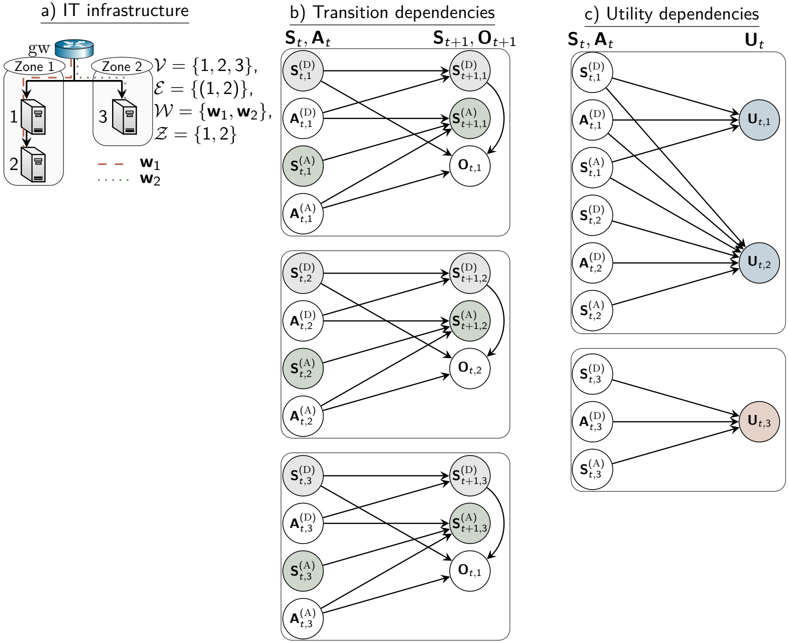
*(A) All nodes $\mathcal{V}$ in the game $\Gamma$ are transition independent.*
*(B) If there is no path between $i$ and $j$ in the topology graph $\mathcal{G}$, then $i$ and $j$ are utility independent.*

## Corollary

*$\Gamma$ decomposes into $|\mathcal{W}|$ additive subproblems that can be solved independently and in parallel.*



# Additive Structure Across Workflows: Minimal Example



# System Decomposition

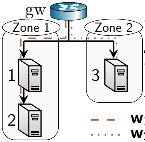
To avoid explicitly enumerating the very large state, observation, and action spaces of $\Gamma$, we exploit three structural properties.

1. **Additive structure across workflows.**
   - The game decomposes into additive subgames on the workflow-level, which means that the strategy for each subgame can be optimized independently
2. **Optimal substructure within a workflow.**
   - The subgame for each workflow decomposes into subgames on the node-level that satisfy the *optimal substructure* property
3. **Threshold properties of local defender strategies.**
   - The optimal node-level strategies for the defender exhibit threshold structures, which means that they can be estimated efficiently

# Optimal Substructure Within a Workflow

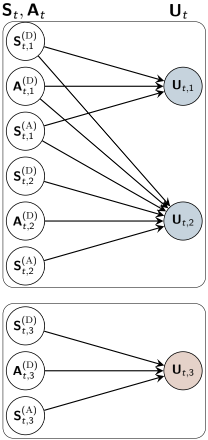
- Nodes in the same workflow are utility dependent.
- $\implies$ Locally-optimal strategies for each node **can not** simply be added together to obtain an optimal strategy for the workflow.
- However, the locally-optimal strategies satisfy the optimal substructure property.
- $\implies$ there exists an algorithm for constructing an optimal workflow strategy from locally-optimal strategies for each node.

IT infrastructure




$\mathcal{V} = \{1,2,3\}$,
$\mathcal{E} = \{(1,2)\}$,
$\mathcal{W} = \{\mathbf{w}_1, \mathbf{w}_2\}$,
$\mathcal{Z} = \{1,2\}$

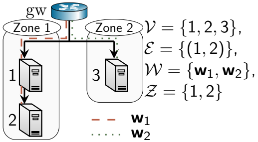
Utility dependencies



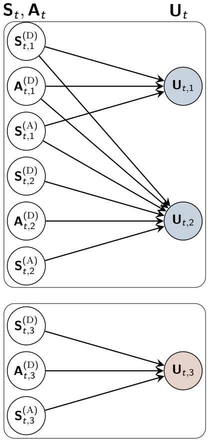
# Optimal substructure within a workflow

- Nodes in the same workflow are utility dependent.
- $\implies$ Locally-optimal strategies for each node **can not** simply be added together to obtain an optimal strategy for the workflow.
- However, the locally-optimal strategies satisfy the optimal substructure property.
- $\implies$ there exists an algorithm for constructing an optimal workflow strategy from locally-optimal strategies for each node.

IT infrastructure




Utility dependencies



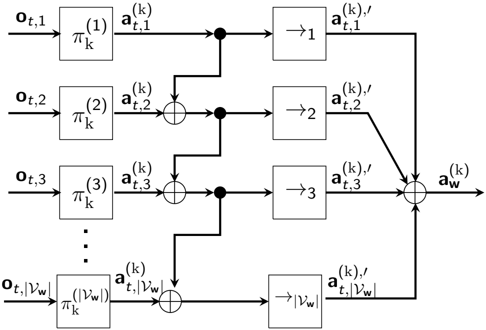
# Algorithm for Combining Locally-Optimal Node Strategies into Optimal Workflow Strategies

**Algorithm 1:** Algorithm for combining local strategies

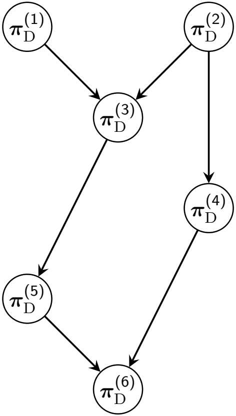
**Input:** $\Gamma$: the game,
$\boldsymbol{\pi}_k$: a vector with local strategies

**Output:** $(\pi_D, \pi_A)$: global game strategies

**Algorithm** $\textsc{composite-strategy}(\Gamma, \boldsymbol{\pi}_k)$
- **for** *player* $k \in \mathcal{N}$ **do**
  - $\pi_k \leftarrow \lambda\ (\mathbf{s}_t^{(k)}, \mathbf{b}_t^{(k)})$
    - $\mathbf{a}_t^{(k)} = ()$
    - **for** *workflow* $\mathbf{w} \in \mathcal{W}$ **do**
      - **for** *node* $i \in \textsc{topological-sort}(\mathcal{V}_{\mathbf{w}})$ **do**
        - $\mathbf{a}_t^{(k,i)} \leftarrow \boldsymbol{\pi}_k^{(i)}(\mathbf{s}_t^{(k)}, \mathbf{b}_t^{(k)})$
        - **if** $\mathrm{gw} \not\rightarrow_{\mathbf{a}_t^{(k)}} i$ **then**
          - $\mathbf{a}_t^{(k,i)} \leftarrow \bot$
        - **end**
        - $\mathbf{a}_t^{(k)} = \mathbf{a}_t^{(k)} \oplus \mathbf{a}_t^{(k,i)}$
      - **end**
    - **end**
    - **return** $\mathbf{a}_t^{(k)}$
- **end**
- **return** $(\pi_D, \pi_A)$



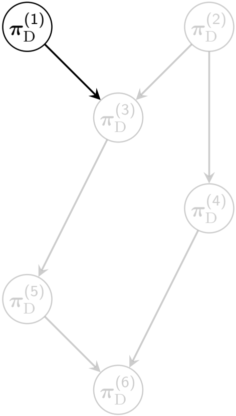
# Algorithm for Combining Locally-Optimal Node Strategies into Optimal Workflow Strategies




$(\boldsymbol{\pi}_D^{(i)})_{i \in \mathcal{V}_{\mathbf{w}}}$: local strategies in the same workflow $\mathbf{w} \in \mathcal{W}$

# Algorithm for Combining Locally-Optimal Node Strategies into Optimal Workflow Strategies

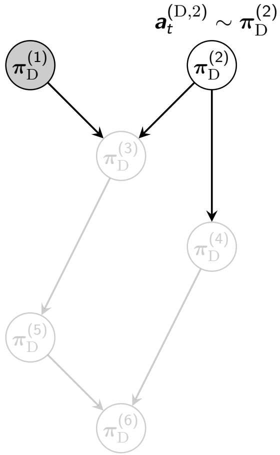
$\boldsymbol{a}_t^{(\mathrm{D},1)} \sim \boldsymbol{\pi}_{\mathrm{D}}^{(1)}$




Workflow action:
$(\boldsymbol{a}_t^{(\mathrm{D},1)})$

Step 1; select action for node 1 according to its local strategy

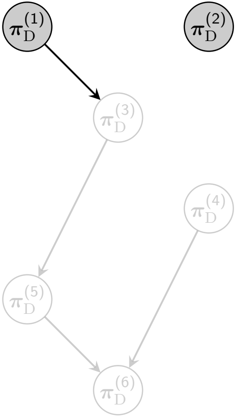
# Algorithm for Combining Locally-Optimal Node Strategies into Optimal Workflow Strategies




Workflow action:
$(\boldsymbol{a}_t^{(\mathrm{D},1)}, \boldsymbol{a}_t^{(\mathrm{D},2)})$

Step 2; update the topology based on the previous local action;
select action $a = 0$ for unreachable nodes;
move to the next node in the topological ordering (i.e. 2);
select the action for the next node according to its local strategy.

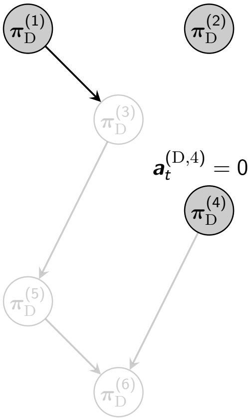
# Algorithm for Combining Locally-Optimal Node Strategies into Optimal Workflow Strategies




Workflow action:
$(\boldsymbol{a}_t^{(\mathrm{D},1)}, \boldsymbol{a}_t^{(\mathrm{D},2)})$

Step 3; update the topology based on the previous local action;
select action $a = 0$ for unreachable nodes;
move to the next node in the topological ordering (i.e. 3);
select the action for the next node according to its local strategy.

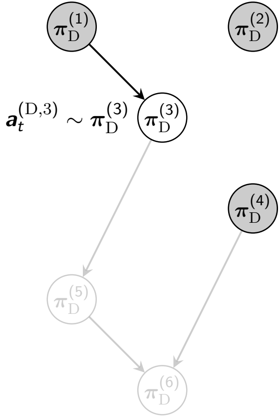
# Algorithm for Combining Locally-Optimal Node Strategies into Optimal Workflow Strategies




Workflow action:
$(\boldsymbol{a}_t^{(\mathrm{D},1)}, \boldsymbol{a}_t^{(\mathrm{D},2)}, \cdot, 0)$

Step 3; update the topology based on the previous local action;
select action $a = 0$ for unreachable nodes;
move to the next node in the topological ordering (i.e. 3);
select the action for the next node according to its local strategy.

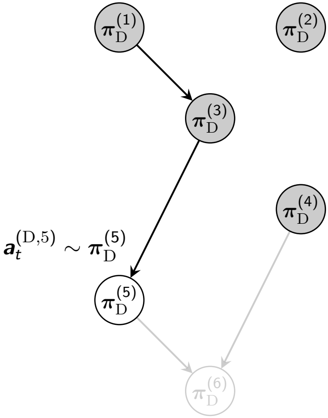
# Algorithm for Combining Locally-Optimal Node Strategies into Optimal Workflow Strategies




Workflow action:
$(\boldsymbol{a}_t^{(\mathrm{D},1)}, \boldsymbol{a}_t^{(\mathrm{D},2)}, \boldsymbol{a}_t^{(\mathrm{D},3)}, 0)$

Step 3; update the topology based on the previous local action;
select action $a = 0$ for unreachable nodes;
move to the next node in the topological ordering (i.e. 3);
select the action for the next node according to its local strategy.

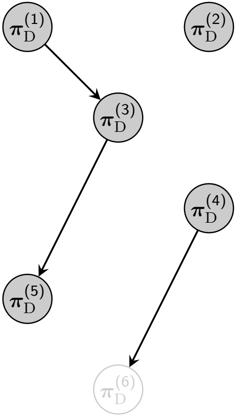
# Algorithm for Combining Locally-Optimal Node Strategies into Optimal Workflow Strategies




Workflow action:
$(a_t^{(\mathrm{D},1)}, a_t^{(\mathrm{D},2)}, a_t^{(\mathrm{D},3)}, 0, a_t^{(\mathrm{D},5)})$

Step 4; update the topology based on the previous local action;
select action $a = 0$ for unreachable nodes;
move to the next node in the topological ordering (i.e. 5);
select the action for the next node according to its local strategy.

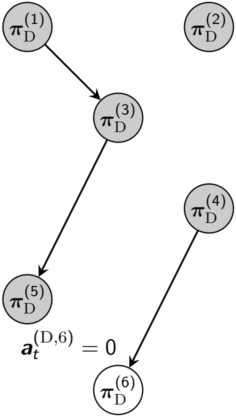
# Algorithm for Combining Locally-Optimal Node Strategies into Optimal Workflow Strategies




Workflow action:
$(\boldsymbol{a}_t^{(\mathrm{D},1)}, \boldsymbol{a}_t^{(\mathrm{D},2)}, \boldsymbol{a}_t^{(\mathrm{D},3)}, 0, \boldsymbol{a}_t^{(\mathrm{D},5)})$

Step 5; update the topology based on the previous local action;
select action $a = 0$ for unreachable nodes;

# Algorithm for Combining Locally-Optimal Node Strategies into Optimal Workflow Strategies

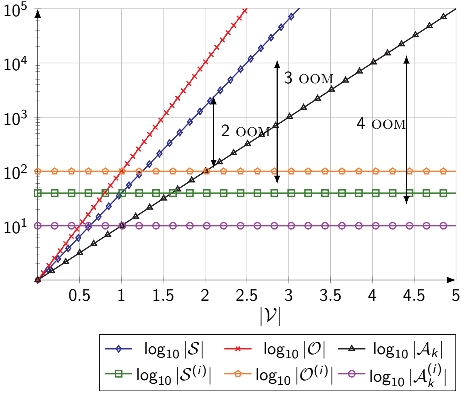



Workflow action:
$(\boldsymbol{a}_t^{(D,1)}, \boldsymbol{a}_t^{(D,2)}, \boldsymbol{a}_t^{(D,3)}, 0, \boldsymbol{a}_t^{(D,5)}, 0)$

Step 5; update the topology based on the previous local action;
select action $a = 0$ for unreachable nodes;

# Computational Benefits of Decomposition

- $\therefore$ we can obtain an optimal (best response) strategy for the full game $\Gamma$ by combining the solutions to $\mathcal{V}$ simpler subproblems that can be solved **in parallel** and have significantly smaller state, observation, and action spaces.



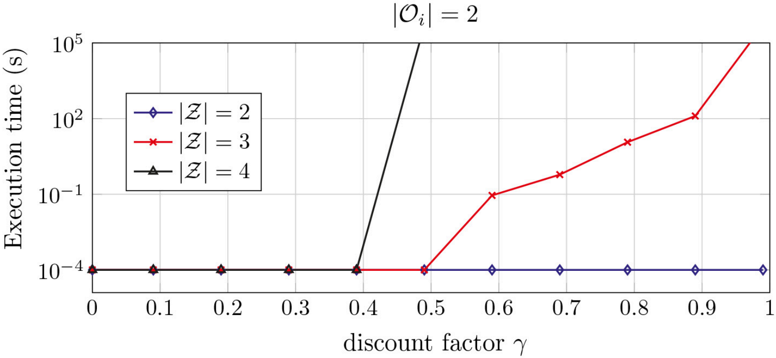

Space complexity comparison between the full game and the decomposed game.

# System Decomposition

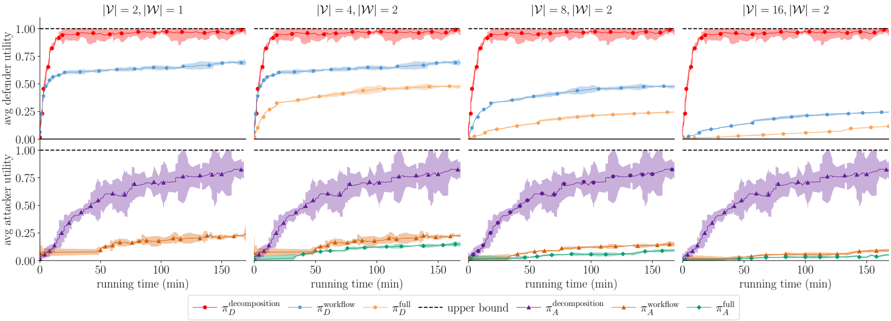
To avoid explicitly enumerating the very large state, observation, and action spaces of $\Gamma$, we exploit three structural properties.

1. **Additive structure across workflows.**
   - The game decomposes into additive subgames on the workflow-level, which means that the strategy for each subgame can be optimized independently
2. **Optimal substructure within a workflow.**
   - The subgame for each workflow decomposes into subgames on the node-level that satisfy the *optimal substructure* property
3. **Threshold properties of local defender strategies.**
   - The optimal node-level strategies for the defender exhibit threshold structures, which means that they can be estimated efficiently

# Can we Solve the Local Problems with Dynamic Programming?

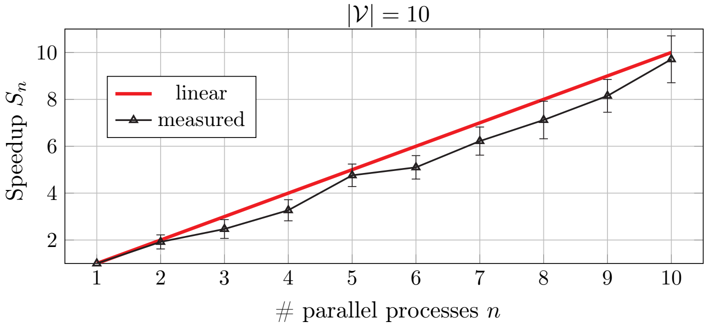


# Scalable learning through decomposition (Simulation)




Learning curves obtained during training of PPO to find best response strategies against randomized opponents; red, purple, blue and brown curves relate to decomposed strategies; the orange and green curves relate to the non-decomposed strategies.

# Scalable learning through decomposition (Simulation)



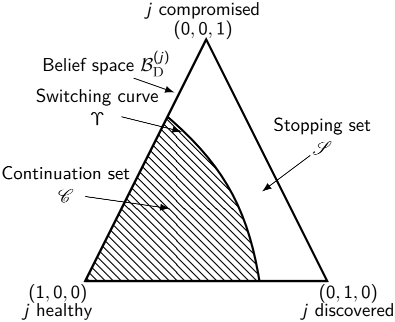

Speedup of completion time when computing best response strategies for the decomposed game with $|\mathcal{V}| = 10$ nodes and different number of parallel processes; the subproblems in the decomposition are split evenly across the processes; let $T_n$ denote the completion time when using $n$ processes, the speedup is then calculated as $S_n = \frac{T_1}{T_n}$; the error bars indicate standard deviations from 3 measurements.

# Threshold Properties of Local Defender Strategies.

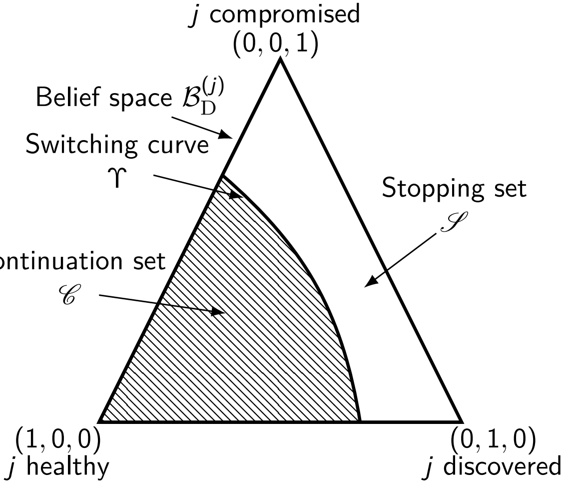
- The local problem of the defender can be decomposed in the temporal domain as

$$\max_{\pi_{\mathrm{D}}} \sum_{t=1}^{T} J = \max_{\pi_{\mathrm{D}}} \sum_{t=1}^{\tau_1} J_1 + \sum_{t=1}^{\tau_2} J_2 + \dots \tag{2}$$

  where $\tau_1, \tau_2, \dots$ are stopping times.
- $\implies$ (1) selection of defensive actions is simplified; and (2) the optimal stopping times are given by a threshold strategy that can be estimated efficiently:



# Threshold Properties of Local Defender Strategies.



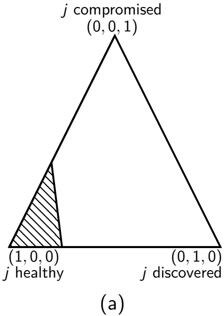

- A node can be in three attack states $s_t^{(\mathrm{A})}$: Healthy, Discovered, Compromised.
- The defender has a belief state $\mathbf{b}_t^{(\mathrm{D})}$

# Threshold Properties of Local Defender Strategies.

We estimate the optimal switching curves using a linear approximation

$$\pi_{\mathrm{D}}(\mathbf{b}^{(\mathrm{D})}) = \begin{cases} \text{Stop} & \text{if } \begin{bmatrix} 0 & 1 & \boldsymbol{\theta}^T \end{bmatrix} \begin{bmatrix} \mathbf{b}^{(\mathrm{D})} \\ -1 \end{bmatrix} < 0 \\ \text{Continue} & \text{otherwise} \end{cases} \tag{3}$$

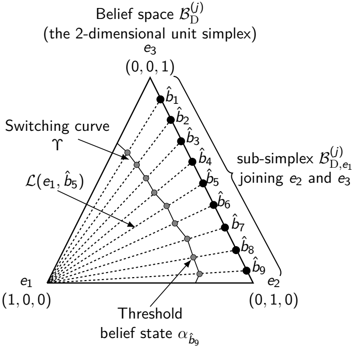
$$\text{subject to } \boldsymbol{\theta} \in \mathbb{R}^2,\ \boldsymbol{\theta}_2 > 0 \text{ and } \boldsymbol{\theta}_1 \geq 1 \tag{4}$$



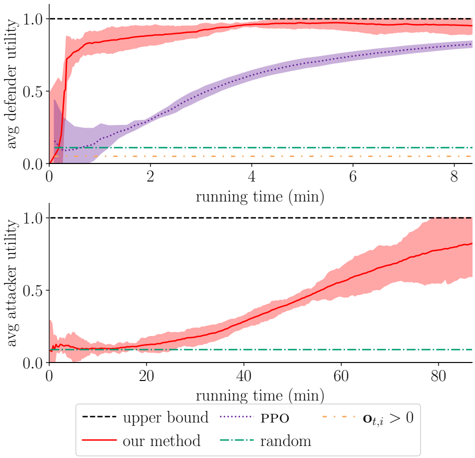

Examples of learned linear switching curves.

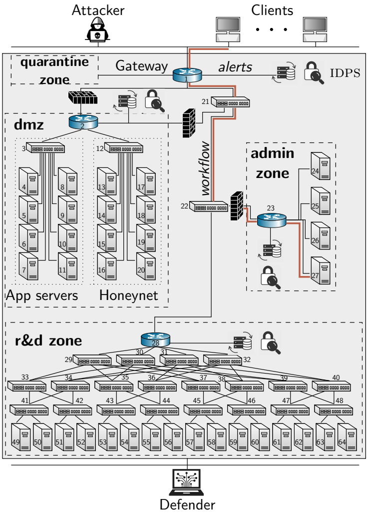
# Proof Sketch (Threshold Properties)

- Let $\mathcal{L}(e_1, \hat{b})$ denote the line segment that starts at the belief state $e_1 = (1, 0, 0)$ and ends at $\hat{b}$, where $\hat{b}$ is in the sub-simplex that joins $e_2$ and $e_3$.
- All beliefs on $\mathcal{L}(e_1, \hat{b})$ are totally ordered according to the Monotone Likelihood Ratio (MLR) order. $\implies$ a threshold belief state $\alpha_{\hat{b}} \in \mathcal{L}(e_1, \hat{b})$ exists where the optimal strategy switches from $C$ to $S$.
- Since the entire belief space can be covered by the union of lines $\mathcal{L}(e_1, \hat{b})$, the threshold belief states $\alpha_{\hat{b}_1}, \alpha_{\hat{b}_2}, \ldots$ yield a switching curve $\Upsilon$.

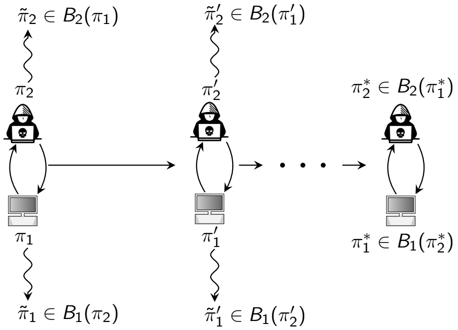


# Learning Best Responses for the Target Infrastructure (Simulation)



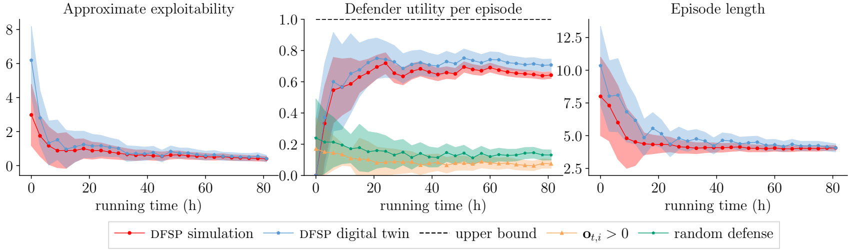
# Decompositional Fictitious Play (DFSP) to Approximate an Equilibrium




Fictitious play: iterative averaging of best responses.

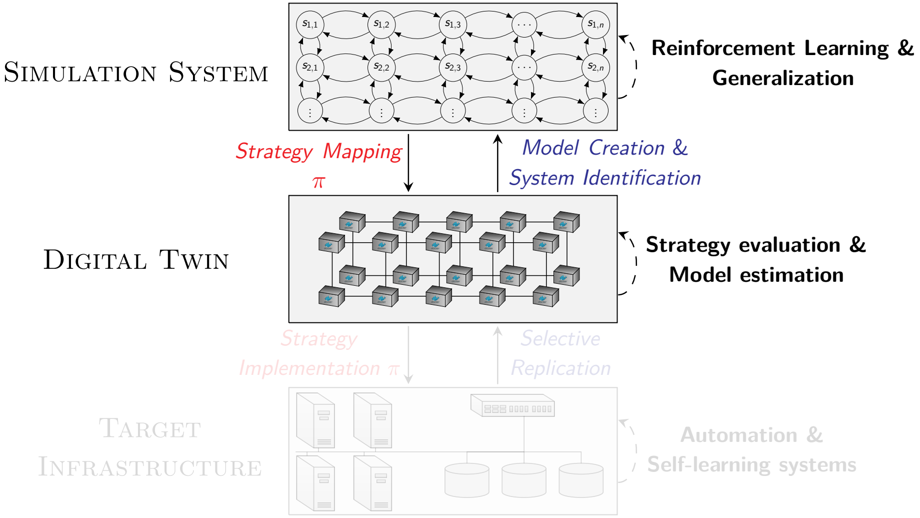
- Learn best response strategies iteratively through the parallel solving of subgames in the decomposition
- Average best responses to approximate the equilibrium

# Learning Equilibrium Strategies

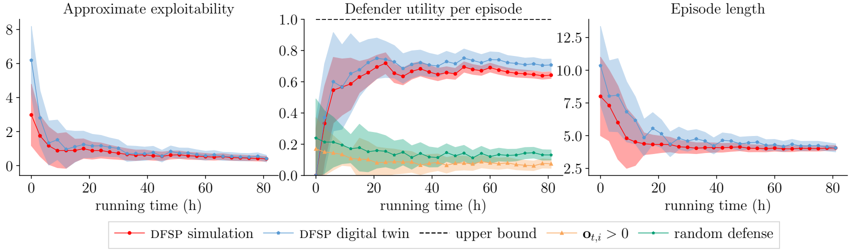



Learning curves obtained during training of DFSP to find optimal (equilibrium) strategies in the intrusion response game; red and blue curves relate to DFSP; black, orange and green curves relate to baselines.

# Evaluation in the Digital Twin

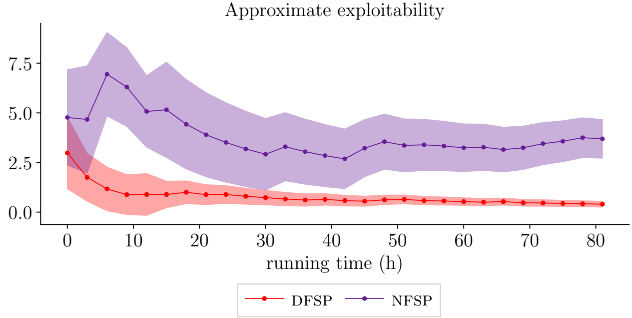


# Learning Equilibrium Strategies




Learning curves obtained during training of DFSP to find optimal (equilibrium) strategies in the intrusion response game; red and blue curves relate to DFSP; black, orange and green curves relate to baselines.

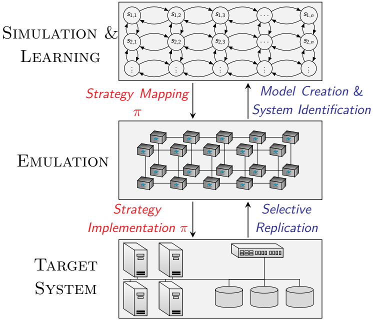
# Learning Equilibrium Strategies (Comparison against NFSP)



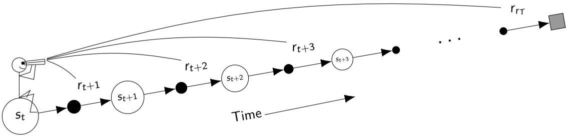

Learning curves obtained during training of DFSP and NFSP to find optimal (equilibrium) strategies in the intrusion response game; the red curve relate to DFSP and the purple curve relate to NFSP; all curves show simulation results.

# Conclusions

- We develop a *framework* to automatically learn security strategies.
- We apply the method to an **intrusion response use case**.
- We design a novel decompositional approach to find near-optimal intrusion responses for large-scale IT infrastructures.
- We show that the decomposition reduces both the computational complexity of finding effective strategies, and the sample complexity of learning a system model by several orders of magnitude.



# Current and Future Work




1. **Extend use case**
   - Heterogeneous client population
   - Extensive threat model of the attacker
2. **Extend solution framework**
   - Model-predictive control
   - Rollout-based techniques
   - Extend system identification algorithm
3. **Extend theoretical results**
   - Exploit symmetries and causal structure