Scalable Learning of Intrusion Response through Recursive Decomposition

GameSec 2023, Avignon, France
Conference on Decision and Game Theory for Security

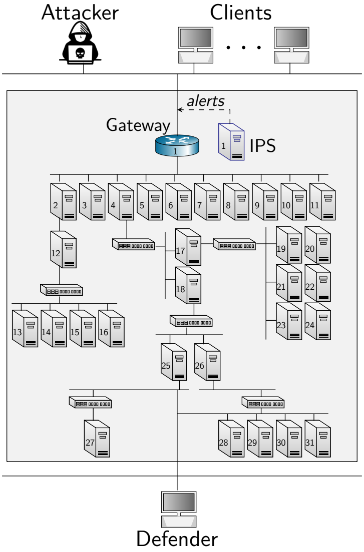
Kim Hammar & Rolf Stadler

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

Oct 18, 2023

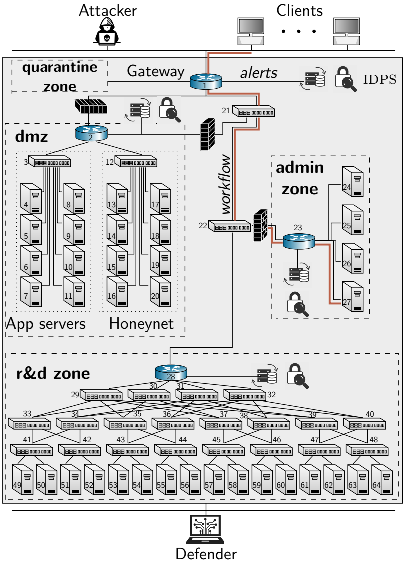
Use Case: Intrusion Response

- 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



System Model

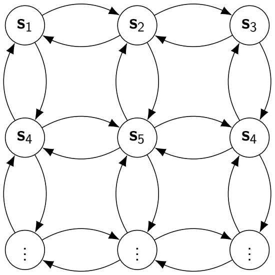
- $\mathcal{G} = \langle \{\text{gw}\} \cup \mathcal{V}, \mathcal{E} \rangle$: directed tree representing the virtual infrastructure
- $\mathcal{V}$: finite set of virtual components.
- $\mathcal{E}$: finite set of component dependencies.
- $\mathcal{Z}$: finite set of zones.



State Space

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

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



- 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

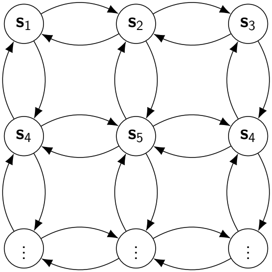
$$\boldsymbol{v}_{t,i} = (\underbrace{v_{t,i}^{(Z)}}_{\text{D}}, \underbrace{v_{t,i}^{(I)}, v_{t,i}^{(R)}}_{\text{A}})$$

- 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

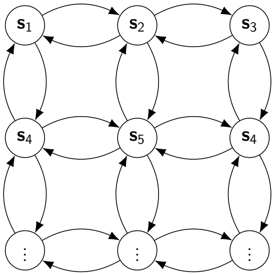
$$\mathbf{v}_{t,i} = (\underbrace{v_{t,i}^{(Z)}}_{\text{D}}, \underbrace{v_{t,i}^{(I)}, v_{t,i}^{(R)}}_{\text{A}})$$

- 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.

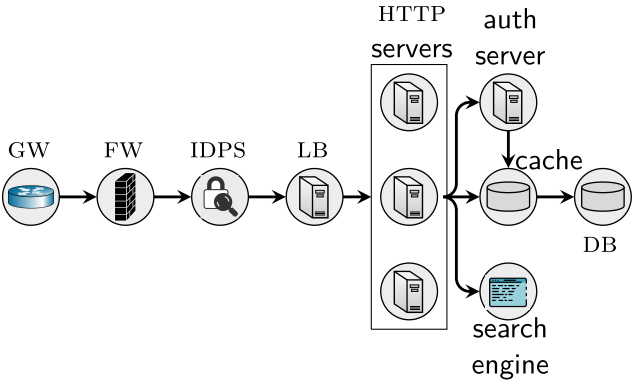


Workflows

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

Workflows

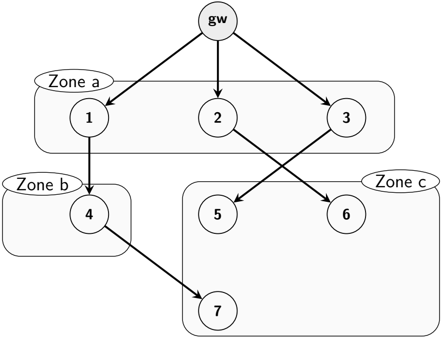
- Services are connected into **workflows** $\mathcal{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_{|\mathcal{W}|}\}$.



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.

Workflows

- 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 subtree $\mathcal{G}_{\mathbf{w}} = \langle \{\mathrm{gw}\} \cup \mathcal{V}_{\mathbf{w}}, \mathcal{E}_{\mathbf{w}} \rangle$ of $\mathcal{G}$
- $\mathcal{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_{|\mathcal{W}|}\}$ induces a partitioning

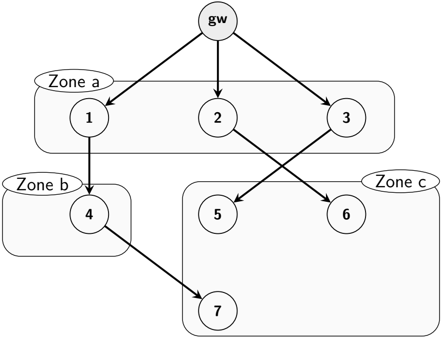


A workflow tree

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

Workflow

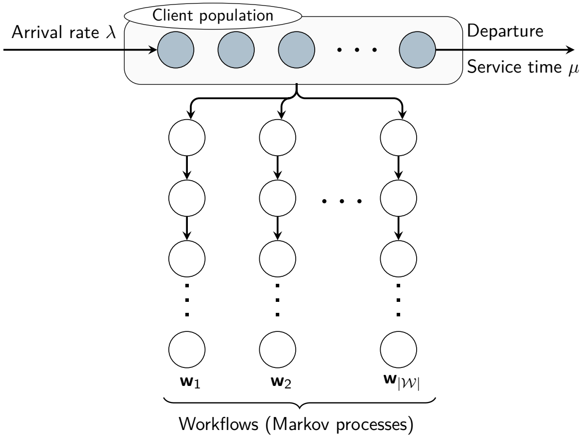
- Services are connected into **workflows** $\mathcal{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_{|\mathcal{W}|}\}$.
- Each $\mathbf{w} \in \mathcal{W}$ is realized as a subtree $\mathcal{G}_\mathbf{w} = \langle \{\mathrm{gw}\} \cup \mathcal{V}_\mathbf{w}, \mathcal{E}_\mathbf{w} \rangle$ of $\mathcal{G}$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



A workflow tree

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

Clients



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

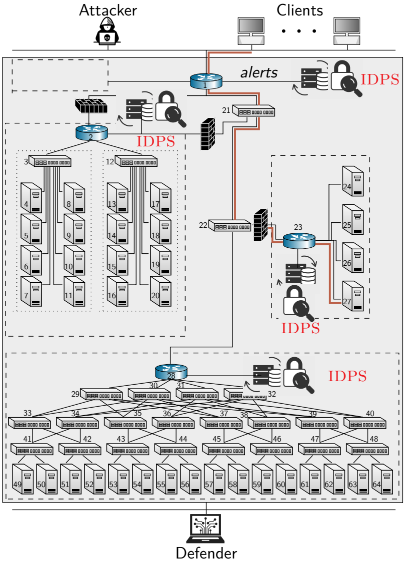
Observations

- 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



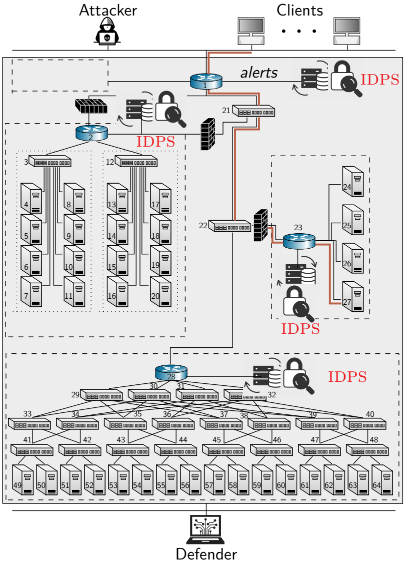
Observations

- IDPSs inspect network traffic and generate alert vectors:

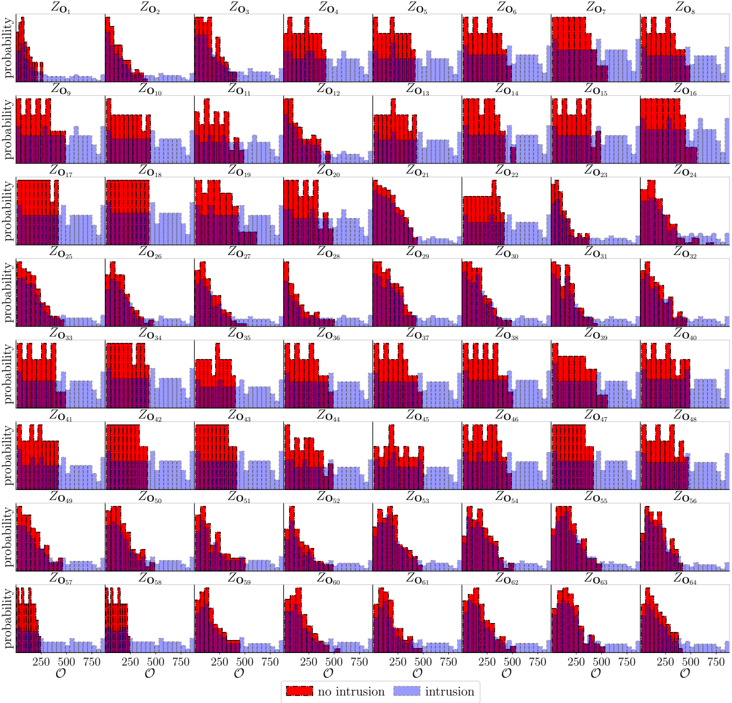
$$\mathbf{o}_t \triangleq \left(\mathbf{o}_{t,1}, \ldots, \mathbf{o}_{t,|\mathcal{V}|}\right) \in \mathbb{N}_0^{|\mathcal{V}|}$$

  $\mathbf{o}_{t,i}$ 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

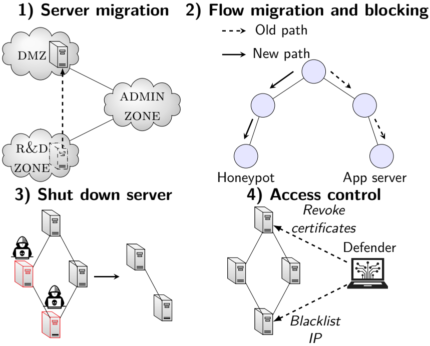


Distributions of # alerts weighted by priority $Z_{\mathbf{O}_i}(\mathbf{O}_i \mid \mathbf{S}_i^{(\mathrm{D})}, \mathbf{A}_i^{(\mathrm{A})})$ per node $i \in \mathcal{V}$



Defender

- 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 $\pi_{\mathrm{D}} \in \Pi_{\mathrm{D}} : \mathcal{H}_{\mathrm{D}} \rightarrow \Delta(\mathcal{A}_{\mathrm{D}})$, where

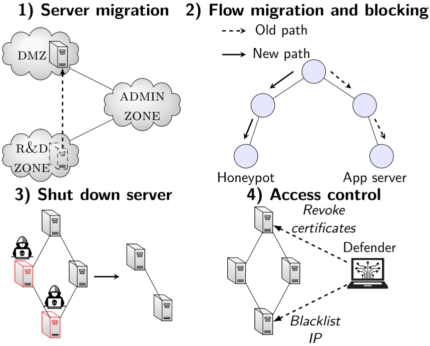
$$\mathbf{h}_t^{(\mathrm{D})} = (\mathbf{s}_1^{(\mathrm{D})}, \mathbf{a}_1^{(\mathrm{D})}, \mathbf{o}_1, \ldots, \mathbf{a}_{t-1}^{(\mathrm{D})}, \mathbf{s}_t^{(\mathrm{D})}, \mathbf{o}_t) \in \mathcal{H}_{\mathrm{D}}$$

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

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

Defender

- 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)



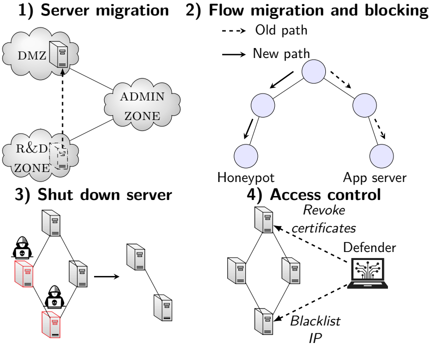
- A **defender strategy** is a function $\pi_{\mathrm{D}} \in \Pi_{\mathrm{D}} : \mathcal{H}_{\mathrm{D}} \rightarrow \Delta(\mathcal{A}_{\mathrm{D}})$, where

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

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

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

Defender



- 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 $\pi_{\mathrm{D}} \in \Pi_{\mathrm{D}} : \mathcal{H}_{\mathrm{D}} \rightarrow \Delta(\mathcal{A}_{\mathrm{D}})$, where

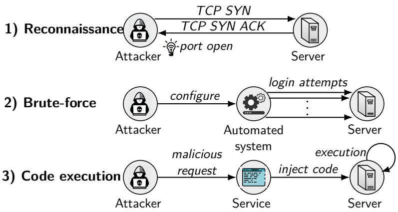
$$\mathbf{h}_t^{(\mathrm{D})} = (\mathbf{s}_1^{(\mathrm{D})}, \mathbf{a}_1^{(\mathrm{D})}, \mathbf{o}_1, \ldots, \mathbf{a}_{t-1}^{(\mathrm{D})}, \mathbf{s}_t^{(\mathrm{D})}, \mathbf{o}_t) \in \mathcal{H}_{\mathrm{D}}$$

- 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

- 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 $\pi_A \in \Pi_A : \mathcal{H}_A \to \Delta(\mathcal{A}_A)$, where $\mathcal{H}_A$ is the space of all possible attacker histories

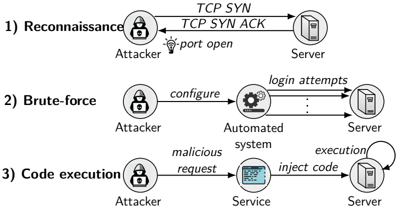
$$\mathbf{h}_t^{(A)} = (\mathbf{s}_1^{(A)}, \mathbf{a}_1^{(A)}, \mathbf{o}_1, \ldots, \mathbf{a}_{t-1}^{(A)}, \mathbf{s}_t^{(A)}, \mathbf{o}_t) \in \mathcal{H}_A$$

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

$$-J$$

Attacker

- 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

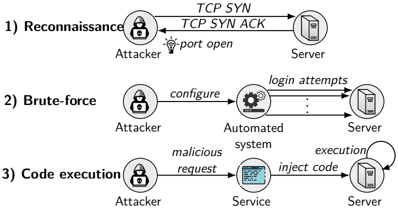
$$\mathbf{h}_t^{(\mathrm{A})} = (\mathbf{s}_1^{(\mathrm{A})}, \mathbf{a}_1^{(\mathrm{A})}, \mathbf{o}_1, \ldots, \mathbf{a}_{t-1}^{(\mathrm{A})}, \mathbf{s}_t^{(\mathrm{A})}, \mathbf{o}_t) \in \mathcal{H}_{\mathrm{A}}$$

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

$$-J$$

Attacker

- 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

$$\underset{\pi_{\mathrm{D}} \in \Pi_{\mathrm{D}}}{\text{maximize}} \ \underset{\pi_{\mathrm{A}} \in \Pi_{\mathrm{A}}}{\text{minimize}} \ \mathbb{E}_{(\pi_{\mathrm{D}}, \pi_{\mathrm{A}})}[J] \tag{1a}$$

$$\text{subject to } \mathbf{s}_{t+1}^{(\mathrm{D})} \sim f_{\mathrm{D}}(\cdot \mid \mathbf{A}_t^{(\mathrm{D})}, \mathbf{A}_t^{(\mathrm{D})}) \quad \forall t \tag{1b}$$

$$\mathbf{s}_{t+1}^{(\mathrm{A})} \sim f_{\mathrm{A}}(\cdot \mid \mathbf{S}_t^{(\mathrm{A})}, \mathbf{A}_t) \quad \forall t \tag{1c}$$

$$\mathbf{o}_{t+1} \sim Z(\cdot \mid \mathbf{S}_{t+1}^{(\mathrm{D})}, \mathbf{A}_t^{(\mathrm{A})}) \quad \forall t \tag{1d}$$

$$\mathbf{a}_t^{(\mathrm{A})} \sim \pi_{\mathrm{A}}(\cdot \mid \mathbf{H}_t^{(\mathrm{A})}), \ \mathbf{a}_t^{(\mathrm{A})} \in \mathcal{A}_{\mathrm{A}}(\mathbf{s}_t) \quad \forall t \tag{1e}$$

$$\mathbf{a}_t^{(\mathrm{D})} \sim \pi_{\mathrm{D}}(\cdot \mid \mathbf{H}_t^{(\mathrm{D})}), \ \mathbf{a}_t^{(\mathrm{D})} \in \mathcal{A}_{\mathrm{D}} \quad \forall t \tag{1f}$$

$\mathbb{E}_{(\pi_{\mathrm{D}}, \pi_{\mathrm{A}})}$ denotes the expectation of the random vectors $(\mathbf{S}_t, \mathbf{O}_t, \mathbf{A}_t)_{t \in \{1, \ldots, T\}}$ when following 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):

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

Existence of a Solution

Theorem

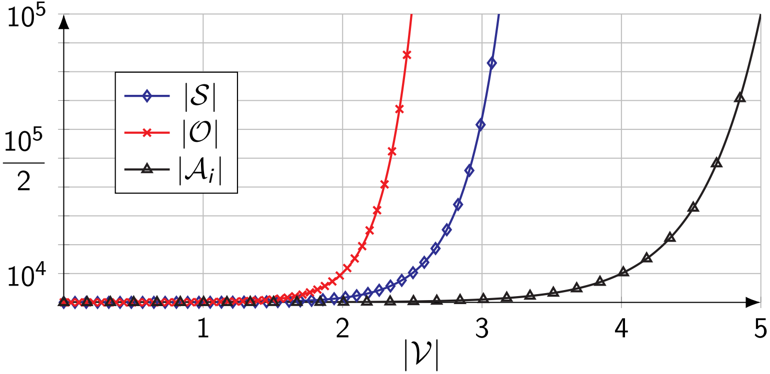
Given the PO-POSG Γ *(2), the following holds:*

(A) Γ *has a mixed Nash equilibrium and a value function* $V^* : \mathcal{B}_{\mathrm{D}} \times \mathcal{B}_{\mathrm{A}} \to \mathbb{R}$ *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 (*POMDP*s):* $\mathscr{M}^{(\mathrm{D})}$ *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

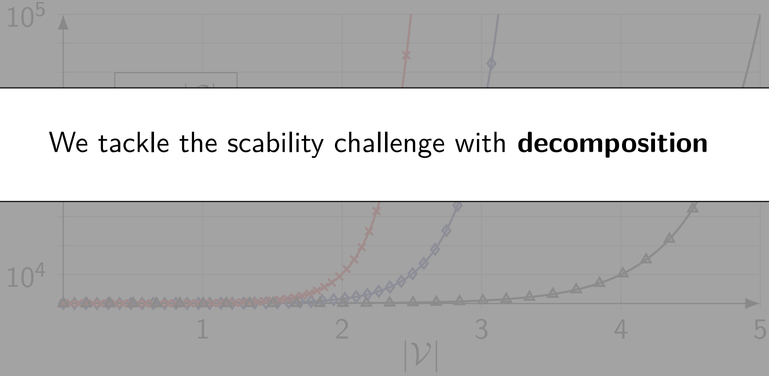
- While Γ has a value, computing it is intractable. The state, action, and observation spaces of the game **grow exponentially** with $|\mathcal{V}|$.



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

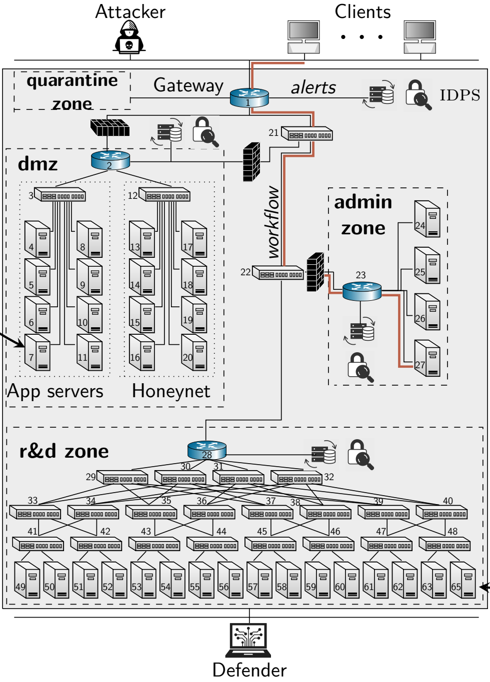


We tackle the scability challenge with **decomposition**

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

Intuitively..

The optimal action here...



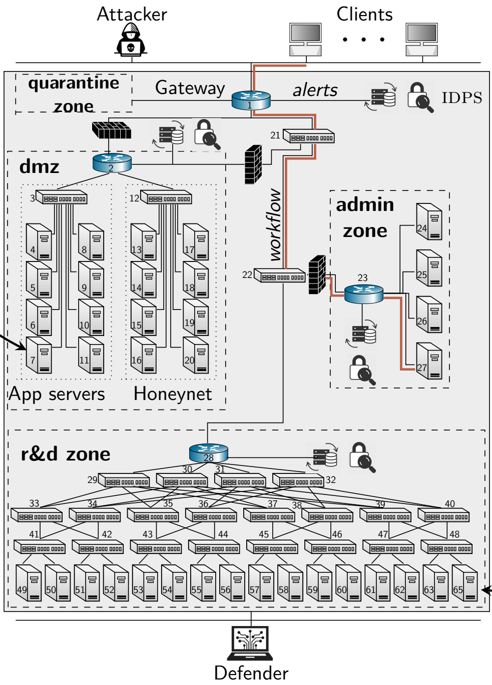
Does not directly depend on the state or action of a node down here

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

Our Approach: System Decomposition

To avoid explicitly enumerating the very large state, observation, and action spaces of Γ, 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

Our Approach: System Decomposition

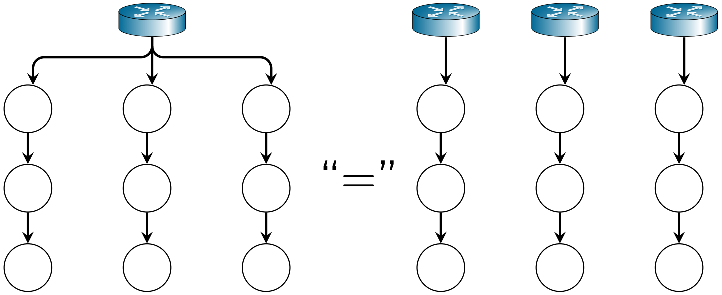
To avoid explicitly enumerating the very large state, observation, and action spaces of Γ, we exploit three structural properties.

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

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

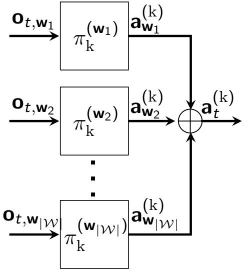
Additive Structure Across Workflows

Theorem (Additive structure across workflows)

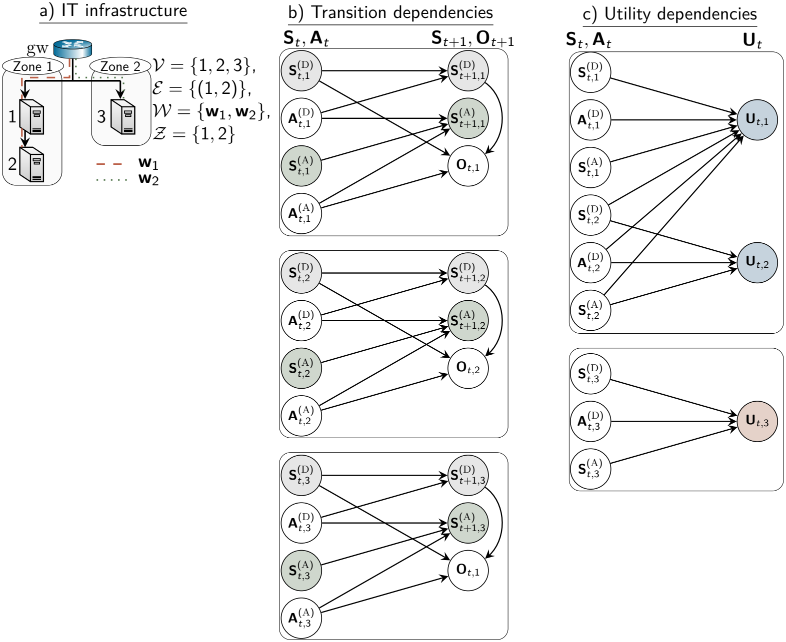
(A) All nodes $\mathcal{V}$ in the game Γ are transition independent.
(B) If there is no path between i and j in the topology graph $\mathcal{G}$, then i and j are utility independent.

Corollary

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



Additive Structure Across Workflows: Example



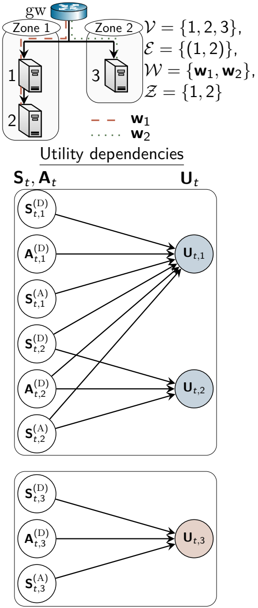
Our Approach: System Decomposition

To avoid explicitly enumerating the very large state, observation, and action spaces of Γ, 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

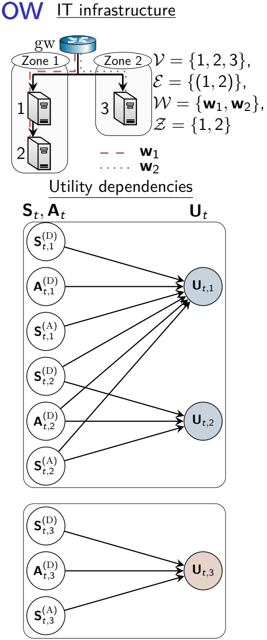
Optimal Substructure Within a Workflow

- 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.

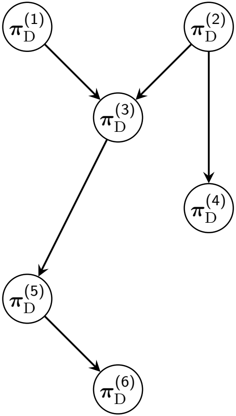


Optimal Substructure Within a Workflow

- 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.

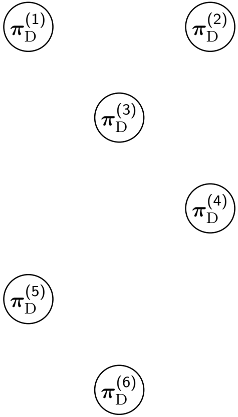


Algorithm for Combining Locally-Optimal Node Strategies into Optimal Workflow Strategies



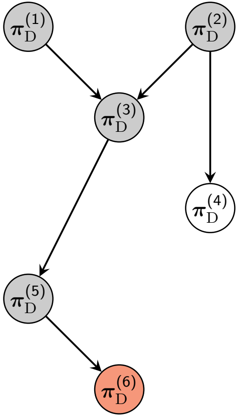
$(\pi_D^{(i)})_{i \in \mathcal{V}_\mathbf{w}}$: local strategies in the same workflow $\mathbf{w} \in \mathcal{W}$

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



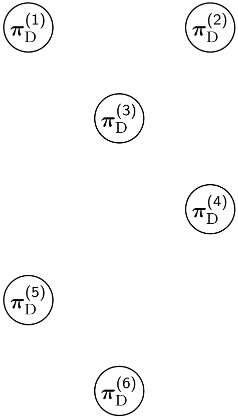
$(\boldsymbol{\pi}_D^{(i)})_{i \in \mathcal{V}_{\mathbf{w}}}$: local strategies in the same workflow $\mathbf{w} \in \mathcal{W}$

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



Can redefine the utility function for each node i
to take into account the utility impact on its ancestors.
e.g. utility of node 6 need to include utility impact for $1, 3, 5$.

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



Can prove that this utility transformation makes the nodes utility independent.

$\implies$ Optimal substructure.

Our Approach: System Decomposition

To avoid explicitly enumerating the very large state, observation, and action spaces of Γ, 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

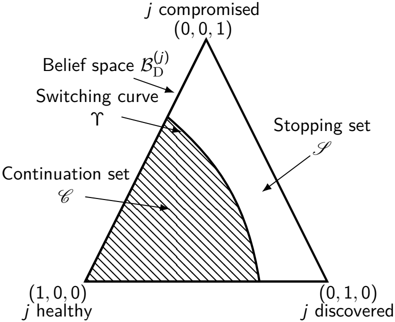
Threshold Properties of Local Defender Strategies.

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

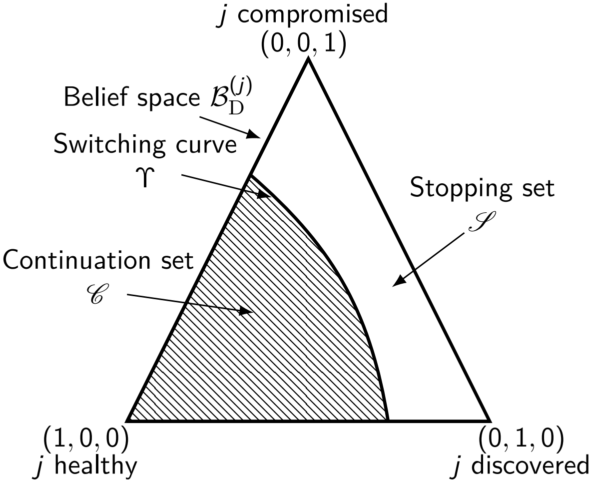
$$\max_{\pi_{\mathrm{D}}} \sum_{t=1}^{T} J = \max_{\pi_{\mathrm{D}}} \sum_{t=1}^{\tau_1} J_1 + \sum_{t=1}^{\tau_2} J_2 + \ldots \quad (2)$$

where $\tau_1, \tau_2, \ldots$ 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:



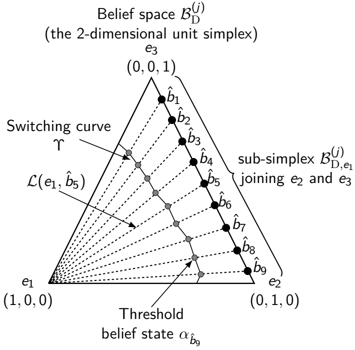
Threshold Properties of Local Defender Strategies.



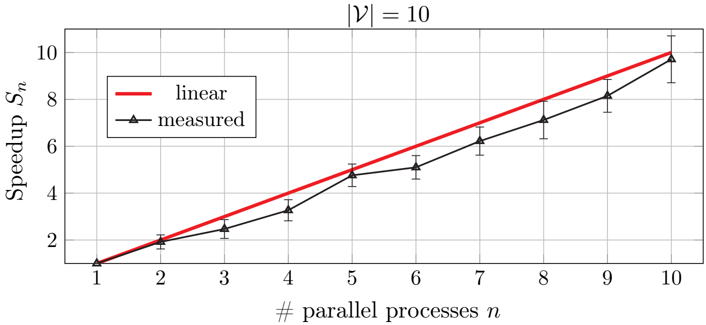
- A node can be in three attack states $s_t^{(\mathrm{A})}$: Healthy, Discovered, Compromised.
- The defender has a belief state $\mathbf{b}_t^{(\mathrm{D})}$

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 Υ.

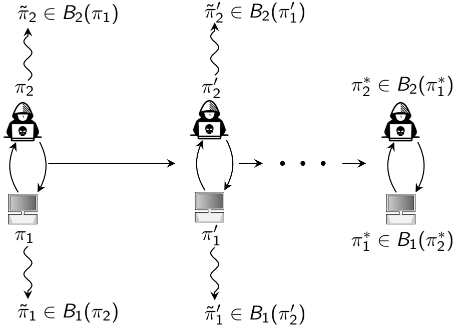


Scalable Learning through Decomposition



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.

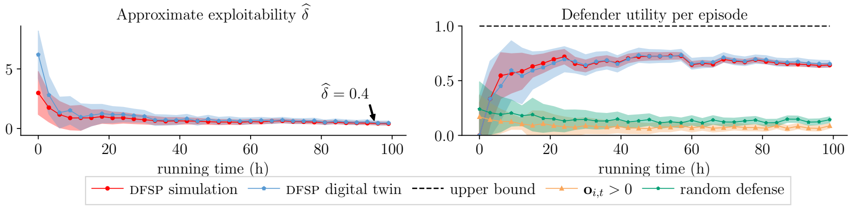
Decompositional Fictitious Play (DFSP)



Fictitious play: iterative averaging of best responses.

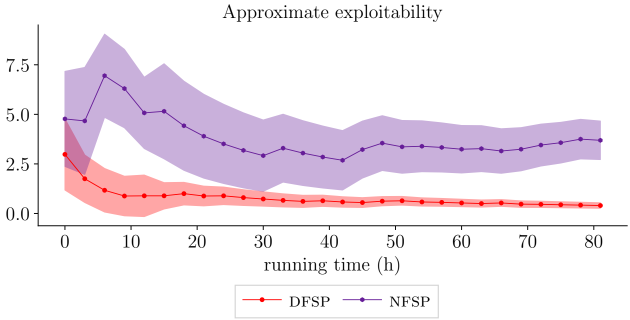
- Learn best response strategies iteratively through the parallel solving of subgames in the decomposition
- Average best responses to approximate the equilibrium

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.

Comparison with 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 study an **intrusion response use case**.
- We formulate the use case as a **POSG**
- We design a novel decompositional approach to approximate equilibria
- We show that the decomposition allows scalable approximation of equilibria.