General Sum Markov Games for Strategic Detection of Advanced Persistent Threats using Moving Target Defense in Cloud Networks

Sailik Sengupta,
Subbarao Kambhampati

Ankur Chowdhary,
Dijiang Huang

Yochan AI Lab

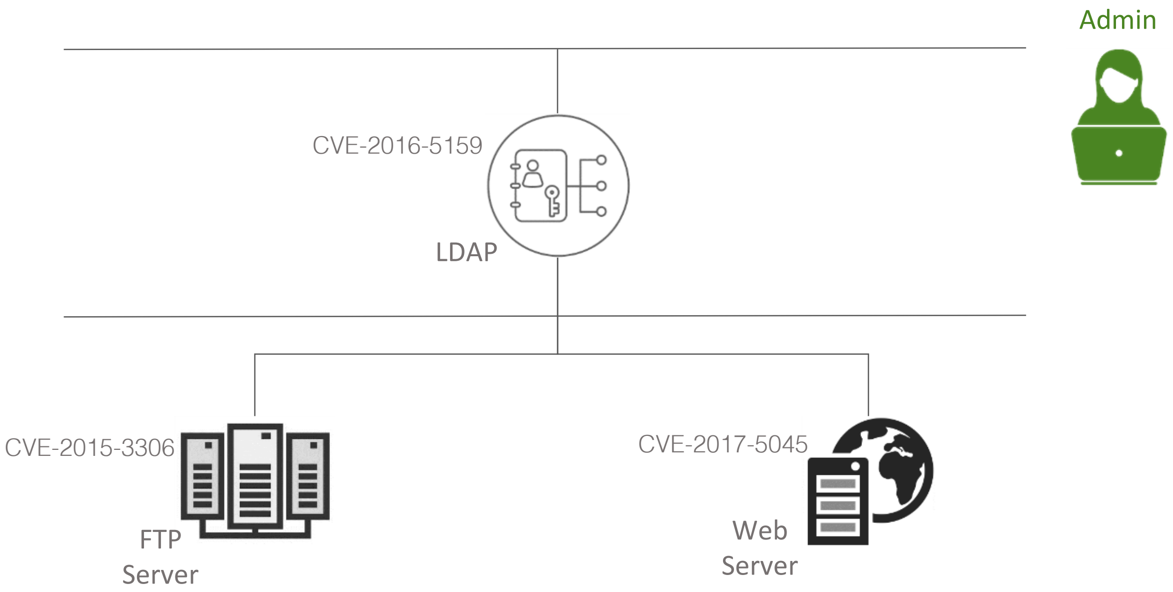
SNAC

Secure Networking and
Computing Lab

Conference on Decision and Game Theory for Security, 2019

General Sum Markov Games for Strategic Detection of Advanced Persistent Threats using Moving Target Defense in Cloud Networks

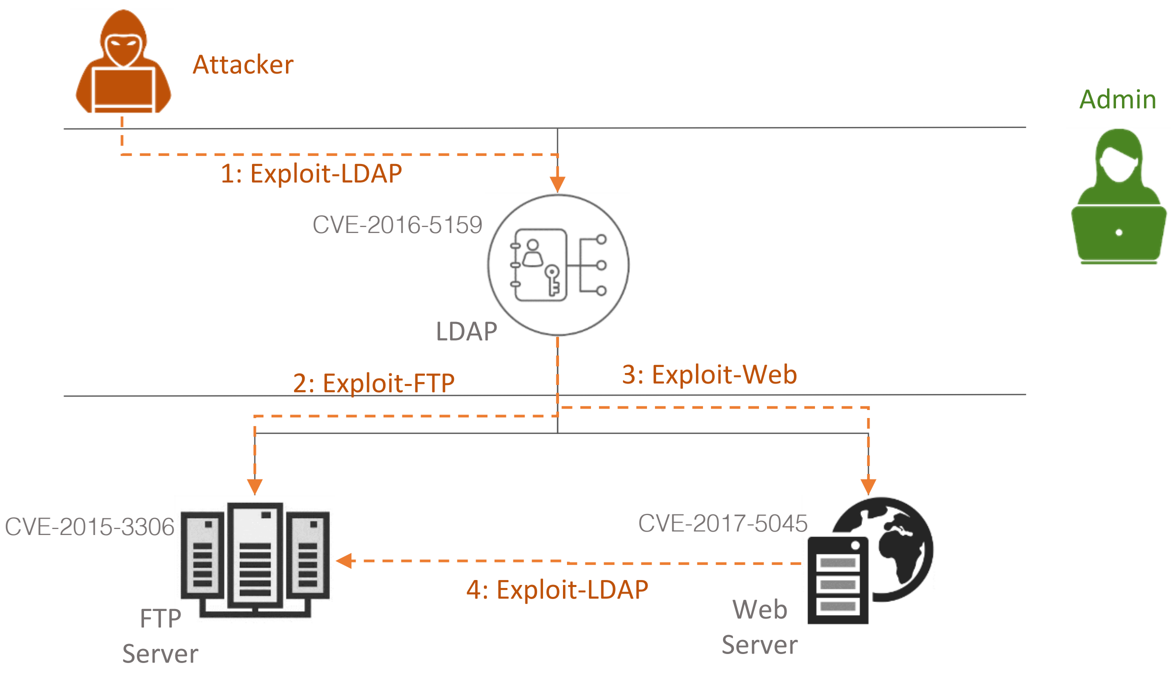
- Cloud service providers provide computing and network resources to third parties for business.
- Attackers seek to attack such systems leading to a loss of Confidentiality, Availability and/or Integrity.
- Defenders can choose to monitor attacks on these systems using intrusion detection systems.



ARIZONA STATE UNIVERSITY

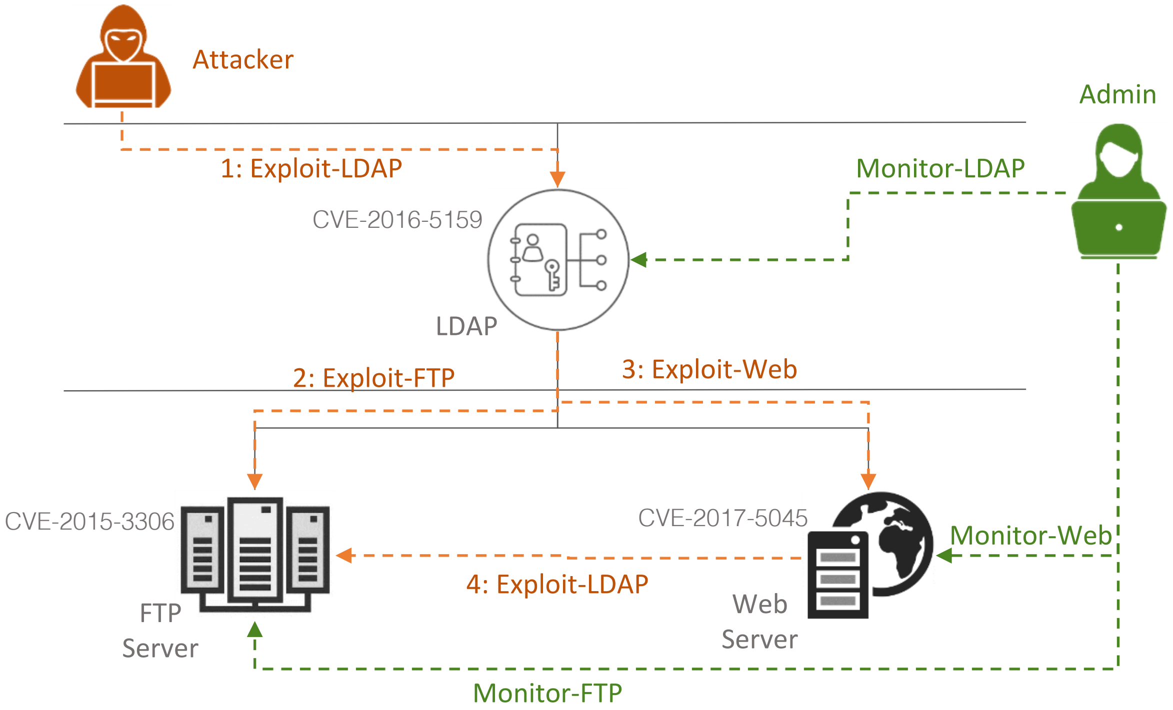
General Sum Markov Games for Strategic Detection of Advanced Persistent **Threats** using Moving Target Defense **in Cloud Networks**

- Cloud service providers provide computing and network resources to third parties for business.
- Attackers seek to attack such systems leading to a loss of Confidentiality, Availability and/or Integrity.
- Defenders can choose to monitor attacks on these systems using intrusion detection systems.



General Sum Markov Games for Strategic Detection of Advanced Persistent Threats using Moving Target Defense in Cloud Networks

- Cloud service providers provide computing and network resources to third parties for business.
- Attackers seek to attack such systems leading to a loss of Confidentiality, Availability and/or Integrity.
- Defenders can choose to monitor attacks on these systems using intrusion detection systems.



Detection of Threats in Cloud Networks

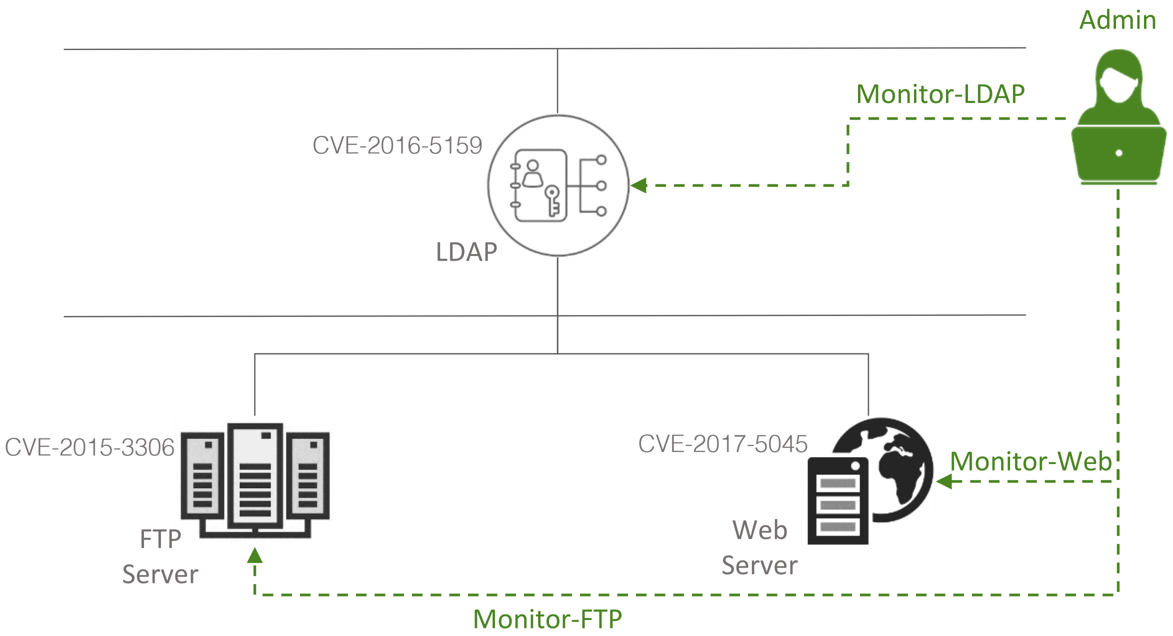
- Place all possible Network and Host-Based Intrusion Detection Systems.

☺ Every known attack can be detected.

☹ Network Performance and Computing Resources are used up for security leading to lower Quality of Service (QoS) for actual customers.

☺ Place a sub-set of them.

☹ Deterministic placement is bad!



Venkatesan, S et al. 2016, Sengupta et. al. 2018

Moving Target Defense
Security

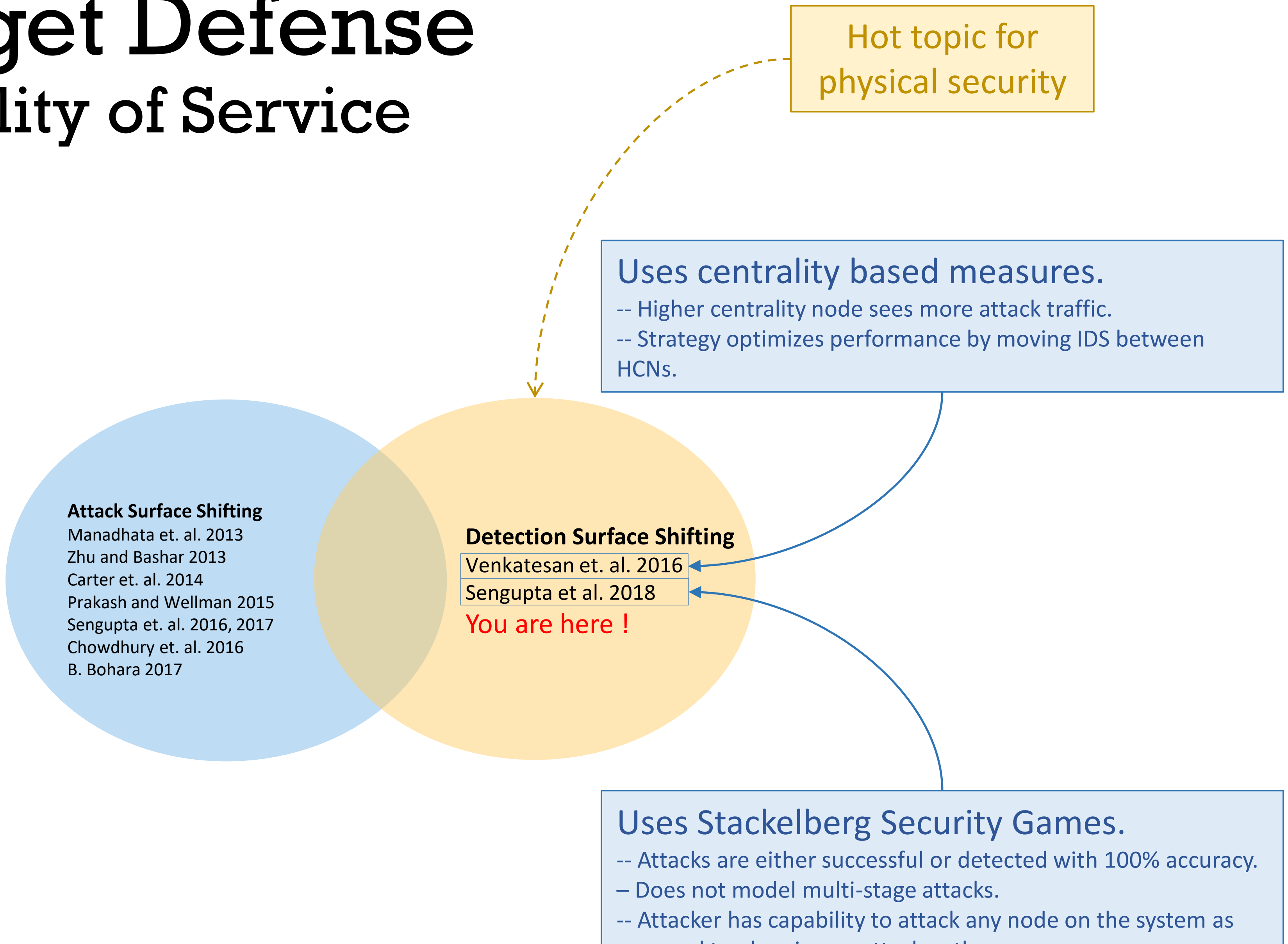
Shift the attack surface so that an attacker's attack, designed based on reconnaissance, is no longer valid at attack time.

Attack Surface Shifting
Manadhata et. al. 2013
Zhu and Bashar 2013
Carter et. al. 2014
Prakash and Wellman 2015
Sengupta et. al. 2016, 2017
Chowdhury et. al. 2016
B. Bohara 2017

Moving Target Defense
Security vs. Quality of Service

Shift the attack surface so that an attacker's attack, designed based on reconnaissance, is no longer valid at attack time.

Shift the detection surface to maximize security with limited number of resources. Helps improve QoS metrics.



Agenda

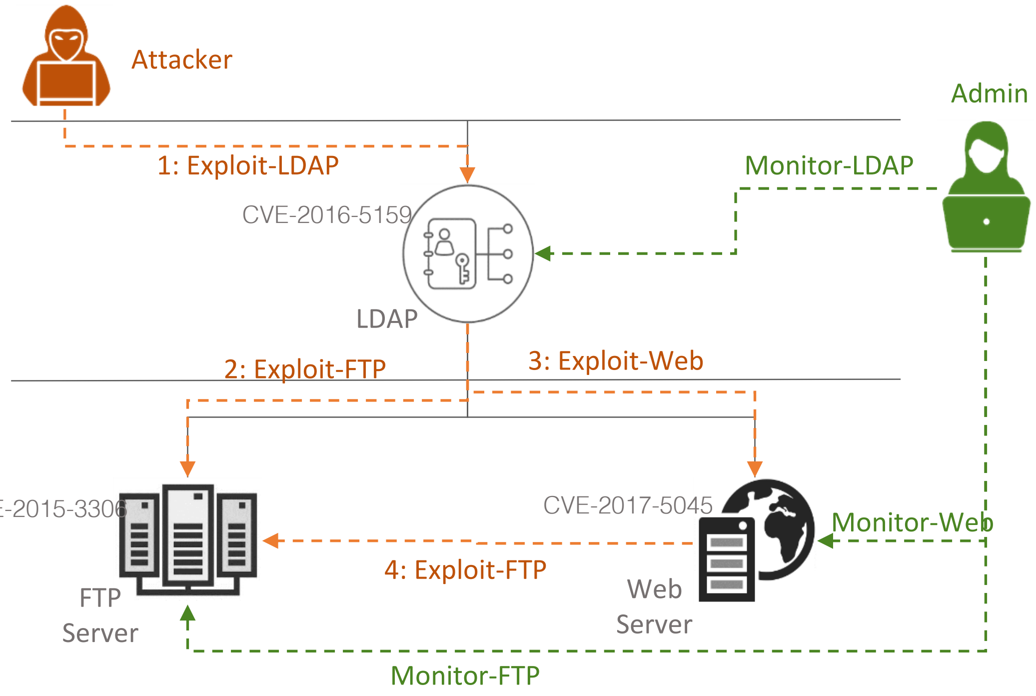
- Formulating the problem as a General Sum Markov Game
  - Attack Graphs
  - Common Vulnerabilities and Exploits (CVEs)
  - Common Vulnerability Scoring System (CVSS)
  - MiniNET simulations
- Placement Strategies
  - Stackelberg Equilibria in Markov Games
  - Anytime solutions with Dynamic Programming
- Experimental results
  - Simulation
  - Emulation

General Sum Markov Games

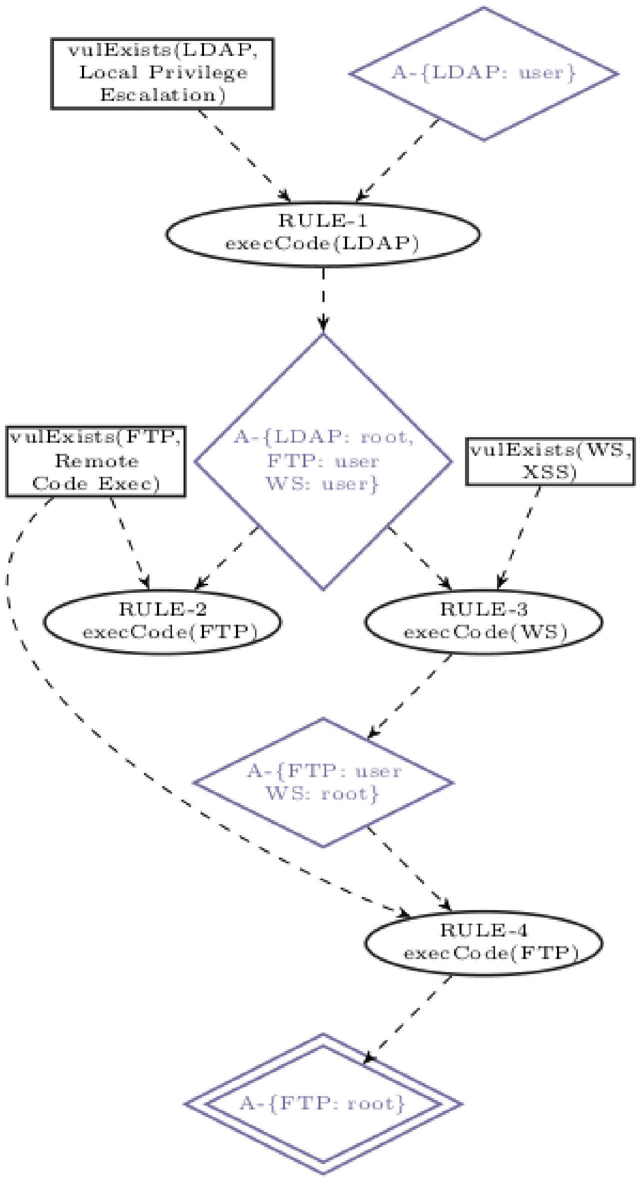
Markov Game (Shapley 1953) for two players P_1 and P_2 can be defined by the tuple $(S, A_1, A_2, \tau, R, \gamma)$ where,

- $S = \{s_1, s_2, s_3, \ldots, s_k\}$ are finite states of the game,
- $A_1 = \{a_1^1, a_1^2, \ldots, a_1^m\}$ represents the possible finite action sets for P_1,
- $A_2 = \{a_2^1, a_2^2, \ldots, a_2^n\}$ are finite action sets for P_2,
- $\tau(s, a_1, a_2, s')$ is the probability of reaching a state $s' \in S$ for state s if P_1 and P_2 take actions a_1 and a_2 respectively,
- $R^i(s, a_1, a_2)$ is the reward obtained by P_i if in state s, P_i and P_{-i} take the actions a_1 and a_2 respectively, and
- $\gamma \mapsto [0, 1)$ is discount factor for future discount rewards.

Attack Graphs



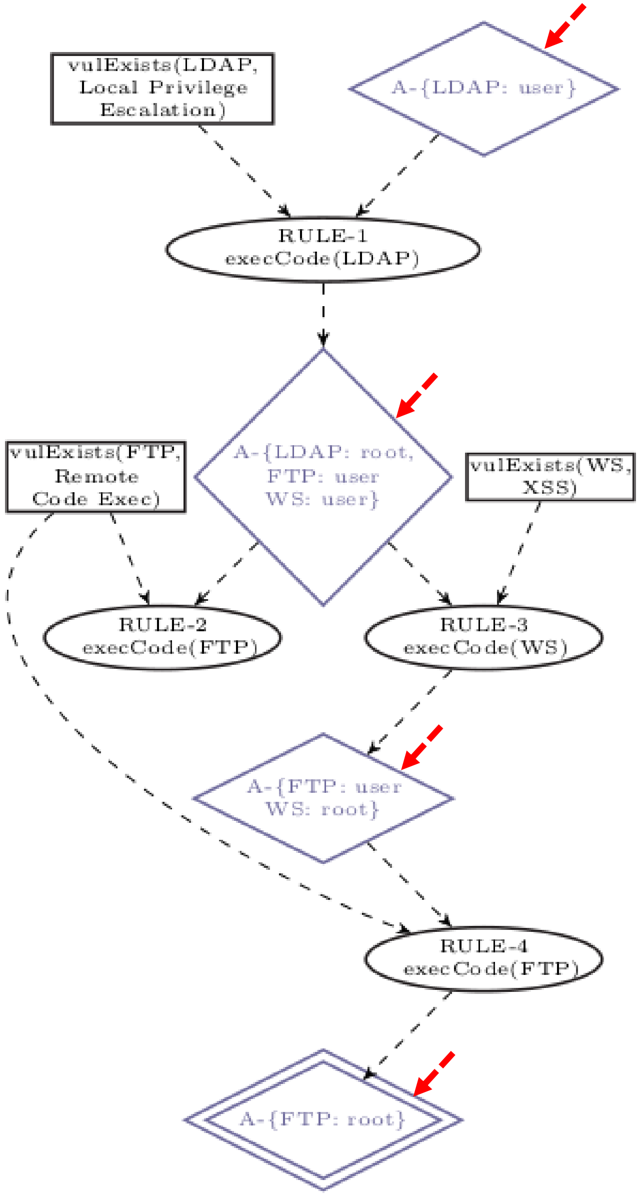
Sample Network scenario



Corresponding Attack Graph

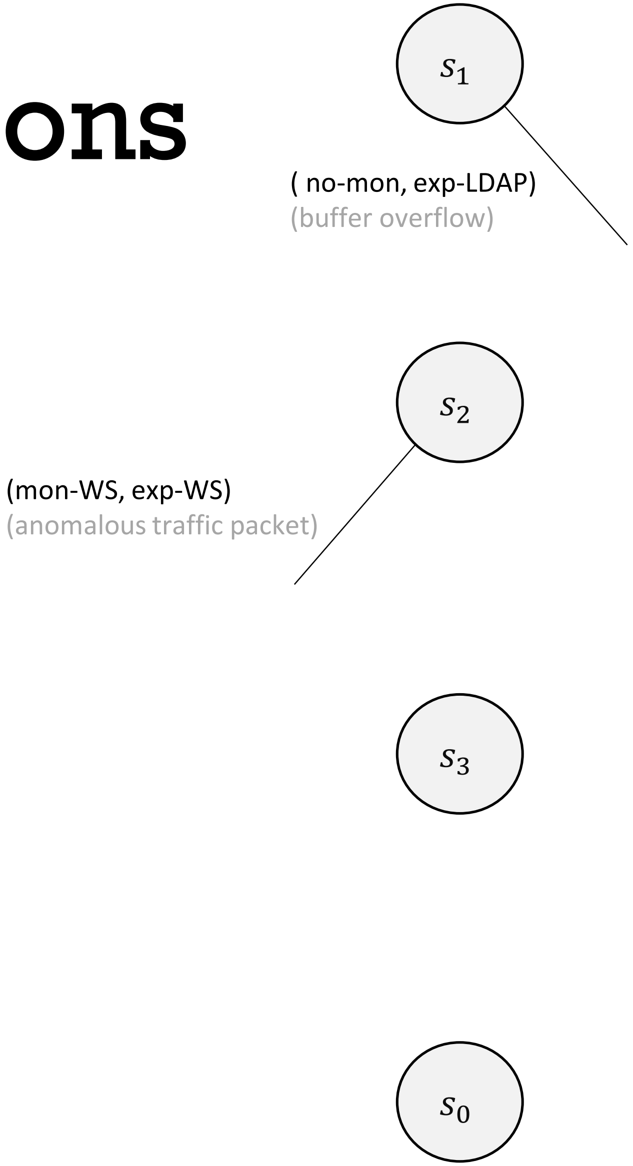
States

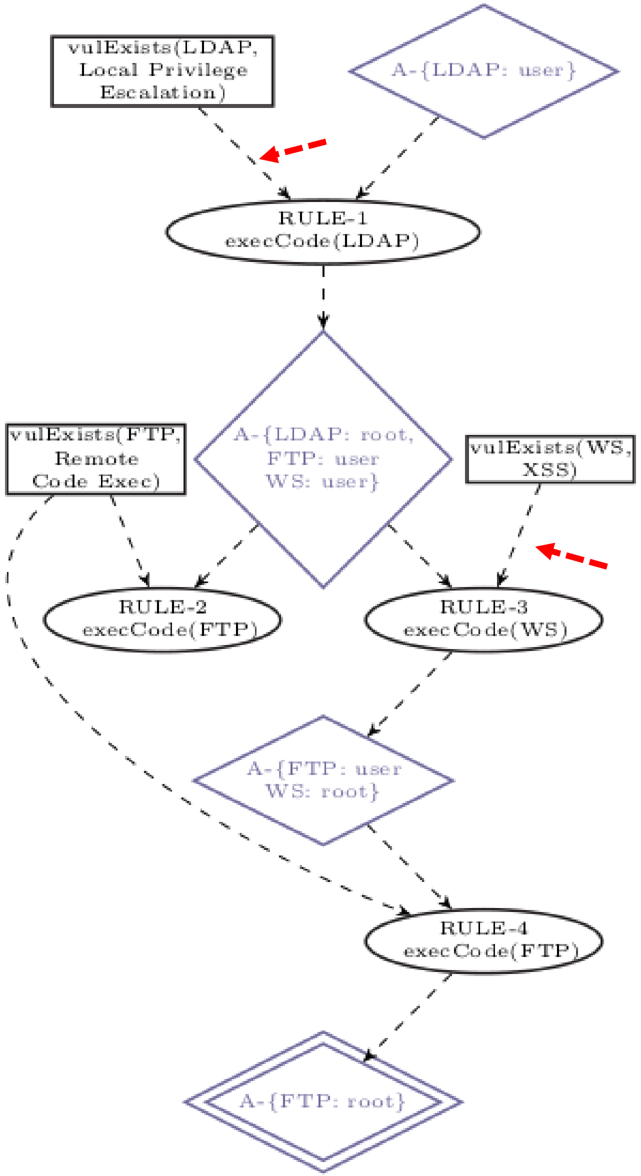
s_1
s_2
s_3
s_0



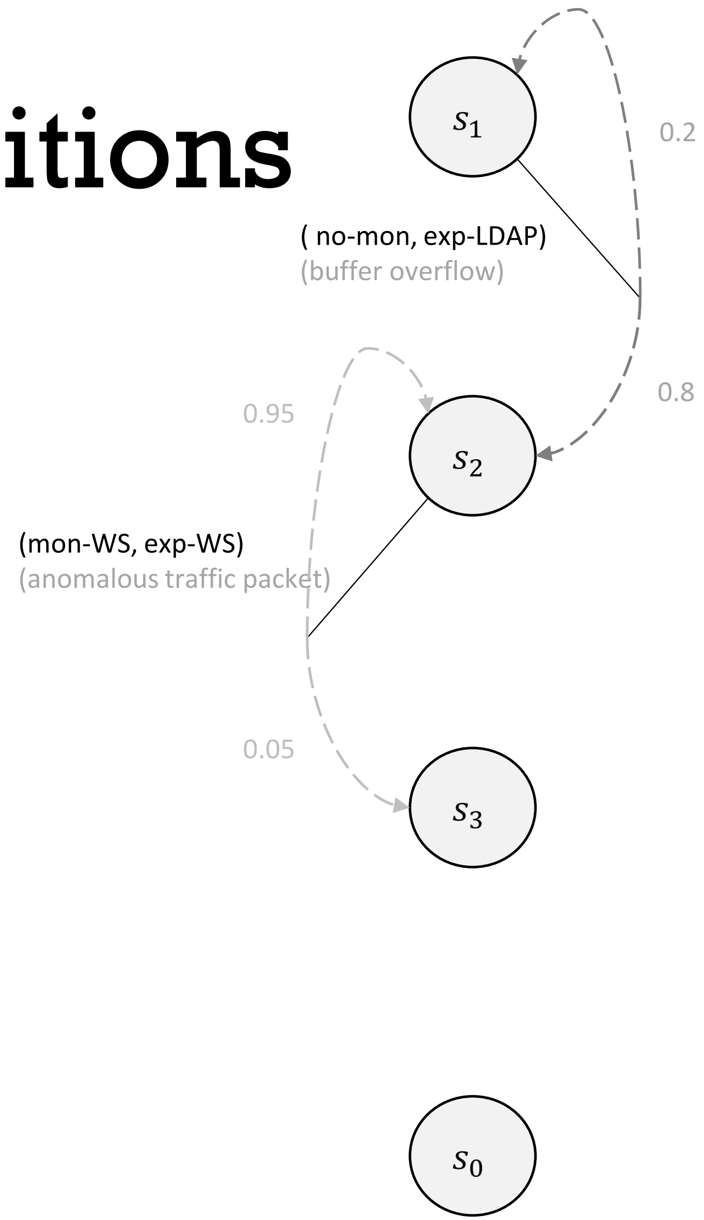
vulExists(LDAP, Local Privilege Escalation)
A-{LDAP: user}
RULE-1 execCode(LDAP)
vulExists(FTP, Remote Code Exec)
A-{LDAP: root, FTP: user WS: user}
vulExists(WS, XSS)
RULE-2 execCode(FTP)
RULE-3 execCode(WS)
A-{FTP: user WS: root}
RULE-4 execCode(FTP)
A-{FTP: root}

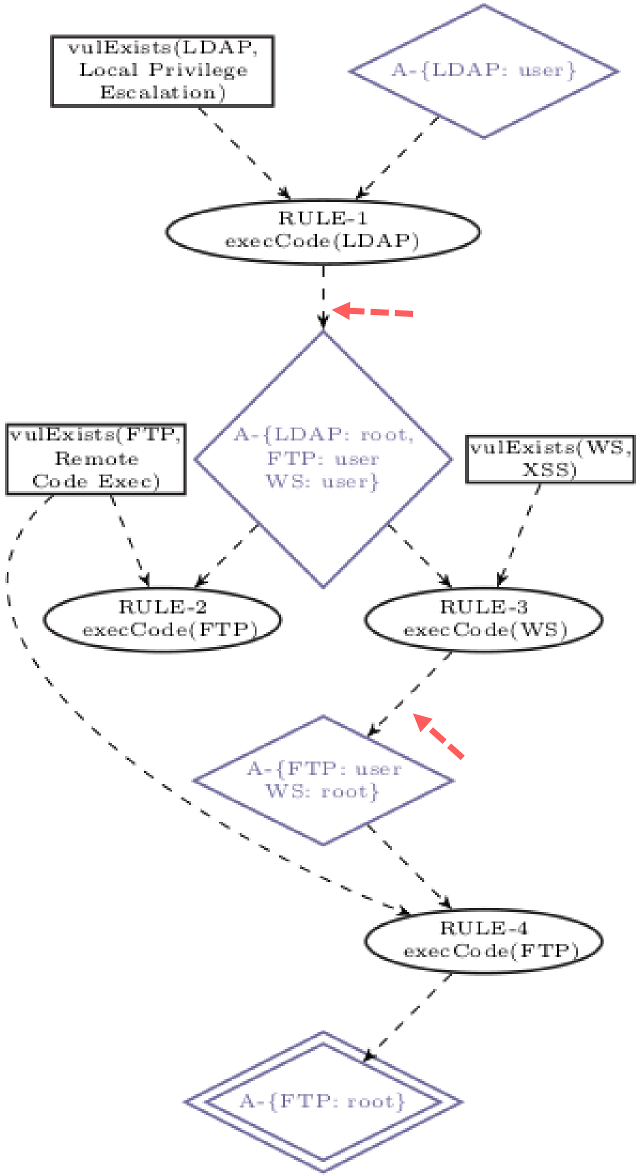
Actions


s_1
(no-mon, exp-LDAP)
(buffer overflow)
s_2
(mon-WS, exp-WS)
(anomalous traffic packet)
s_3
s_0

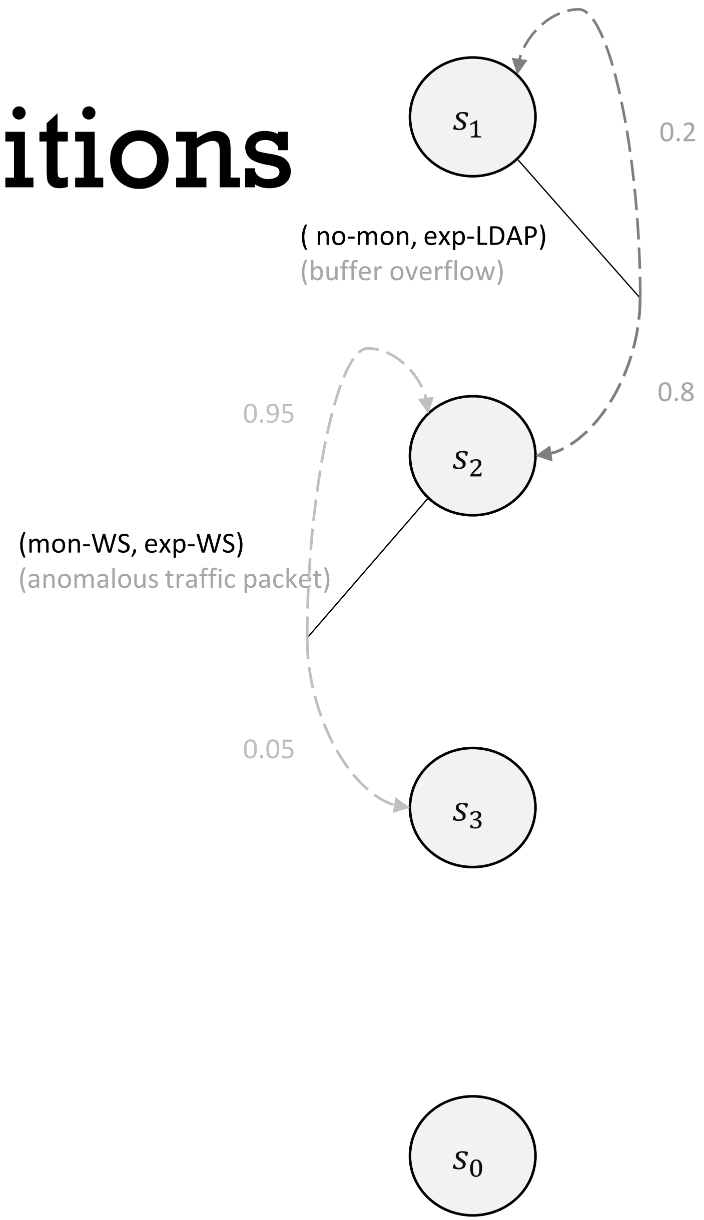

vulExists(LDAP, Local Privilege Escalation)
A-{LDAP: user}
RULE-1 execCode(LDAP)
vulExists(FTP, Remote Code Exec)
A-{LDAP: root, FTP: user WS: user}
vulExists(WS, XSS)
RULE-2 execCode(FTP)
RULE-3 execCode(WS)
A-{FTP: user WS: root}
RULE-4 execCode(FTP)
A-{FTP: root}

Transitions





Transitions



Exploitability score of a Common Vulnerability and Exposures (CVE)

Assumption is based on the fact that a random attacker is more likely to succeed if the attack is easy to exploit.

Chung et. al. 2013 shows how Exploitability Scores can be used in attack graphs for calculating the probability of an attacker being able to successfully exploit an attack.

Accuracies of ML-based monitoring systems can be used to calculate the probability of when an attack succeeds even when a monitoring system is deployed.

Reward Modeling

Impact Score of a Common Vulnerability and Exposures (CVE)

Assumptions that the reward structure results in a zero-sum game is an unreasonable one because an attacker does not care about defenders performance metrics or QoS to legitimate users.

How to find a value for the effect on QoS given that a monitoring system is deployed?

- *Venkateshan et. al. 2016 and Sengupta et. al. 2018 uses centrality measure of the nodes as a heuristic to estimate this value.*
- *We run MiniNET simulations– flood the network with traffic and run resource exhaustive processes with and without the IDS deployed. Measure the reduction in bandwidth or spike in cpu usage.*

	no-mon	mon-FTP
no-act	$0, 0$	$0, -2$
exp-FTP	$10, -10$	$-8, 6$

s_3

Agenda

- Formulating the problem as a General Sum Markov Game
  - Attack Graphs
  - Common Vulnerabilities and Exploits (CVEs)
  - Common Vulnerability Scoring System (CVSS)
  - MiniNET simulations
- Placement Strategies
  - Anytime solutions with Dynamic Programming
  - NE and SSE in Markov Games
- Experimental results
  - Simulation
  - Emulation

Optimal IDS placement policy

Mix max computation when game is zero sum.

Algorithm 1

procedure GIVEN $(S, M, E, \tau, U^D, U^A, \gamma^D = \gamma^A = \gamma)$,
OUTPUT$(V^i(s), \pi^i(s) \ \forall \ i \in \{A, D\})$
  $V(s) = 0 \ \forall \ s$
  loop: $i == k$ break;
  // Update Q-values
  Update $Q^D(s, m, e)$ and $Q^A(s, m, e)$ $\forall \ s \in S, m \in M(s), e \in E(s)$
    using U^D, U^A and $V(s)$.
  // Do value and policy computation
  Calculate $V^i(s)$ and $\pi^i(s)$ for $i \in \{A, D\}$ using the values $Q^i(s, m, e)$
  $i \leftarrow i + 1$
  goto *loop*.
end procedure

Optimal IDS placement policy

In General-sum Games, the notion of Nash and Stackelberg Equilibria may differ.

Algorithm 1

procedure GIVEN $(S, M, E, \tau, U^D, U^A, \gamma^D = \gamma^A = \gamma)$,
OUTPUT$(V^i(s), \pi^i(s) \; \forall \; i \in \{A, D\})$
    $V(s) = 0 \; \forall \; s$
    loop: $i == k$ break;
    // Update Q-values
    Update $Q^D(s, m, e)$ and $Q^A(s, m, e)$ $\; \forall \; s \in S, m \in M(s), e \in E(s)$
        using U^D, U^A and $V(s)$.
    // Do value and policy computation
    Calculate $V^i(s)$ and $\pi^i(s)$ for $i \in \{A, D\}$ using the values $Q^i(s, m, e)$
    $i \leftarrow i + 1$
    goto *loop*.
end procedure

Weaker threat model
Attacker has no idea about defender's placement strategy → NE

Stronger threat model
Attacker has knowledge about defender's placement strategy → SSE

Optimal IDS placement policy

In General-sum Games, the notion of Nash and Stackelberg Equilibria may differ.

Algorithm 1

```
procedure GIVEN (S, M, E, τ, U^D, U^A, γ^D = γ^A = γ),
OUTPUT(V^i(s), π^i(s) ∀ i ∈ {A, D})
    V(s) = 0 ∀ s
    loop: i == k break;
    // Update Q-values
    Update Q^D(s, m, e) and Q^A(s, m, e)  ∀ s ∈ S, m ∈ M(s), e ∈ E(s)
        using U^D, U^A and V(s).
    // Do value and policy computation
    Calculate V^i(s) and π^i(s) for i ∈ {A, D} using the values Q^i(s, m, e)
    i ← i + 1
    goto loop.
end procedure
```

Let us consider a set of IDS systems that the defender can choose to deploy. If every subset of this set can also be covered by the defender, the Set of Subsets Are Sets (SSAS) property holds. Korzhyk *et. al. 2011*

Lemma 1. If in each state of the Markov Game, SSAS holds, $SSE \subset NE$

Weaker threat model

Attacker has no idea about defender's placement strategy → NE

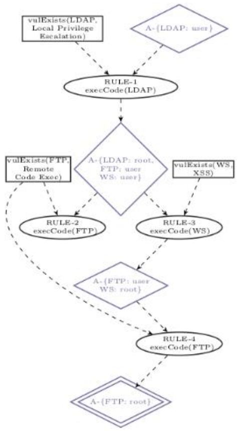
Stronger threat model

Attacker has knowledge about defender's placement strategy → SSE

Agenda

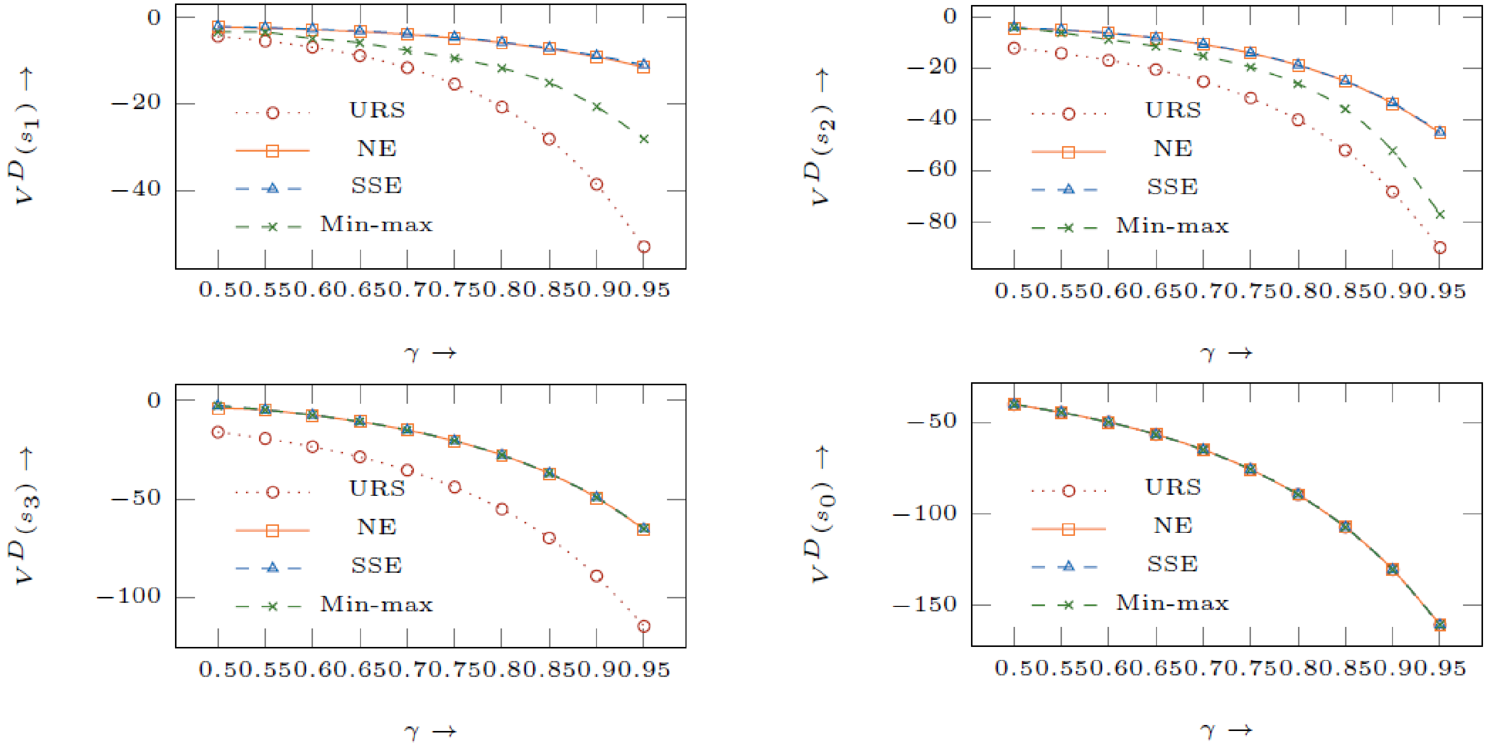
- Formulating the problem as a General Sum Markov Game
    - Attack Graphs
    - Common Vulnerabilities and Exploits (CVEs)
    - Common Vulnerability Scoring System (CVSS)
    - MiniNET simulations
- Placement Strategies
    - Anytime solutions with Dynamic Programming
    - NE and SSE in Markov Games
- Experimental results
    - Simulation
    - Emulation

Experimental Results



At the start of each time period t,

- ➢URS – Uniformly select an IDS system out of n-monitoring actions at random.
- ➢Min-max – Defender's reward is negative of the attacker's reward. Zero-sum Markov Game strategy.

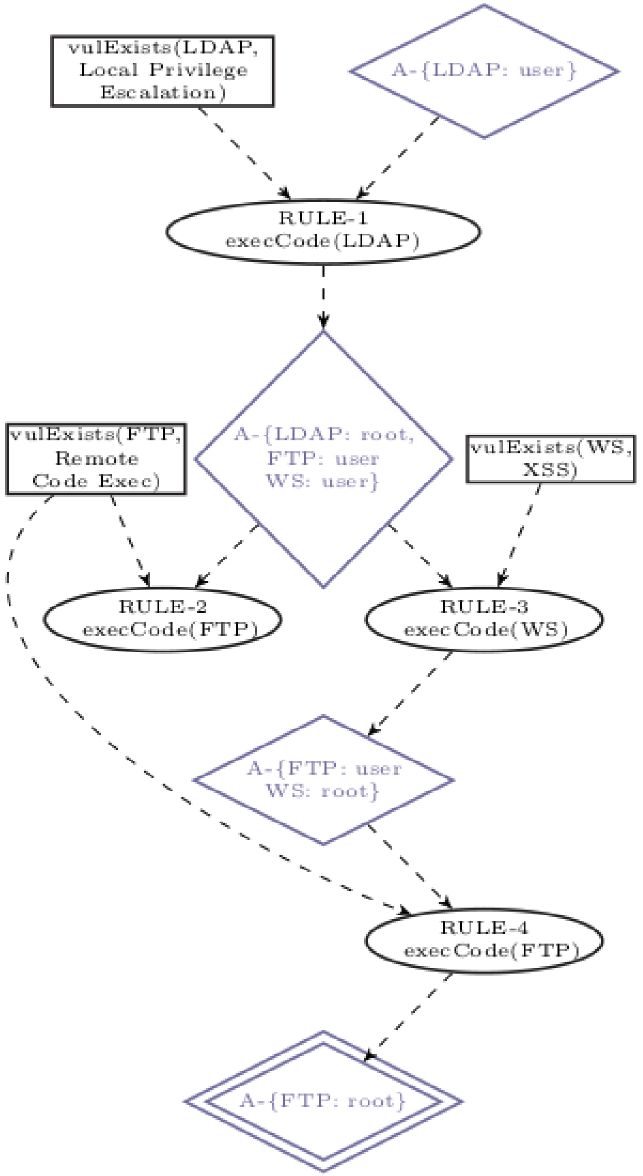


Experimental Results

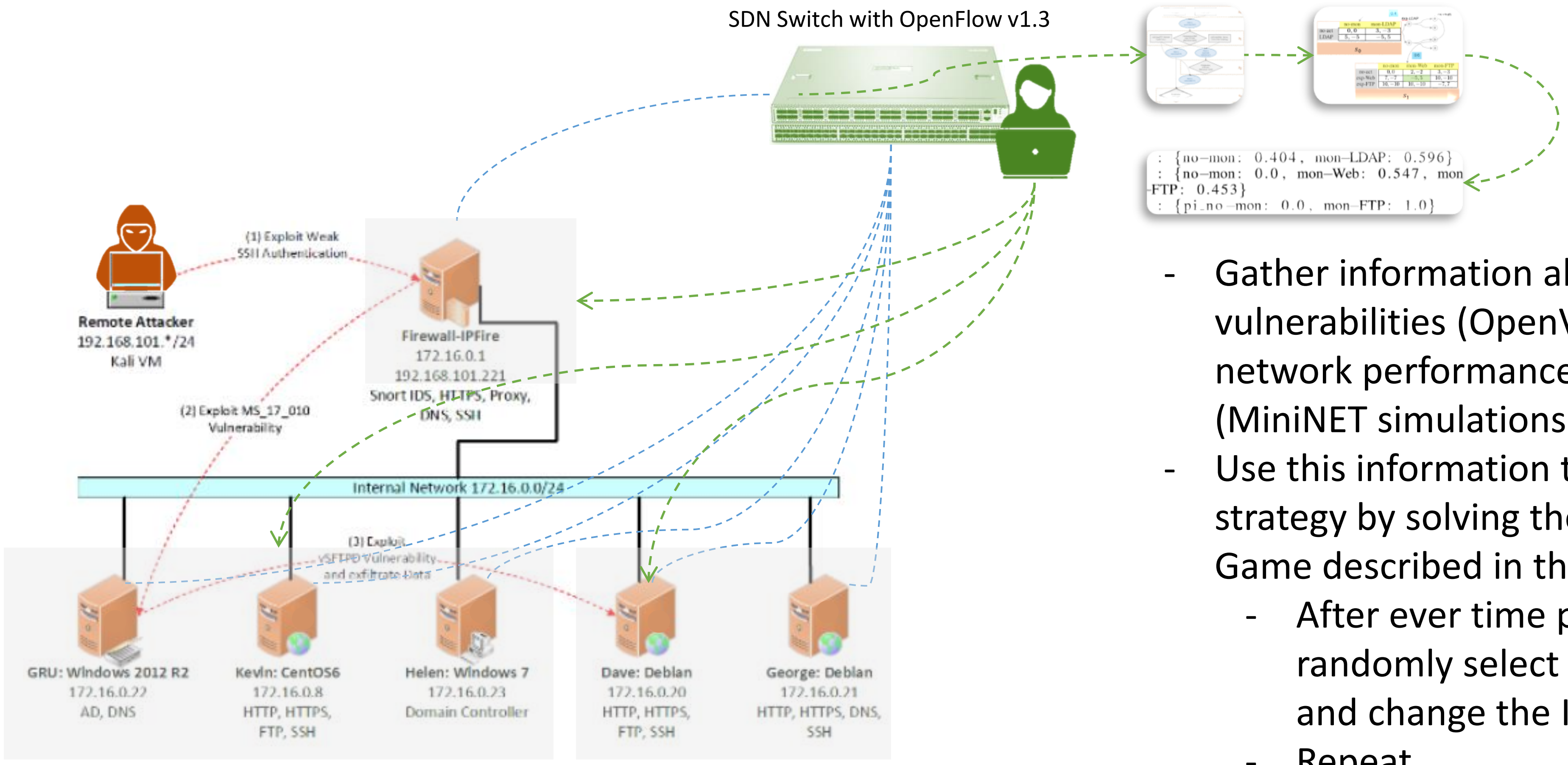
- For states further away from the goal, don't need to monitor at times to enhance performance QoS.

$$\pi_{\text{MG-SSE}}(s_1) : \{\text{no–mon}: 0.097, \text{mon–LDAP}: 0.903\}$$
$$\pi_{\text{MG-SSE}}(s_2) : \{\text{no–mon}: 0.0, \text{mon–Web}: 0.539, \text{mon–FTP}: 0.461\}$$
$$\pi_{\text{MG-SSE}}(s_3) : \{\text{pi\_no–mon}: 0.0, \text{mon–FTP}: 1.0\}$$

- For states closer to the goal, not monitoring is not an option. Security becomes more important that performance.



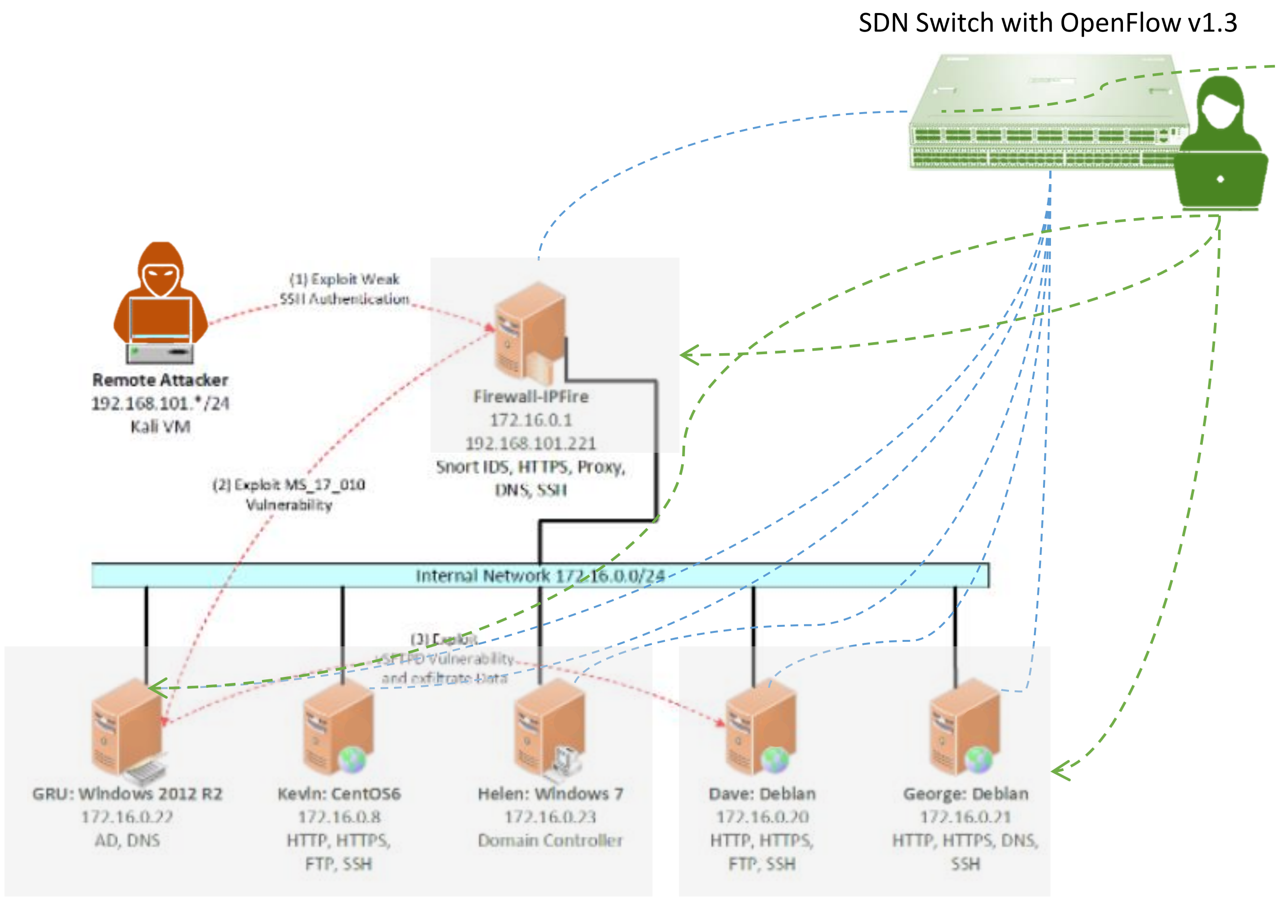
Implementation in ThothLab

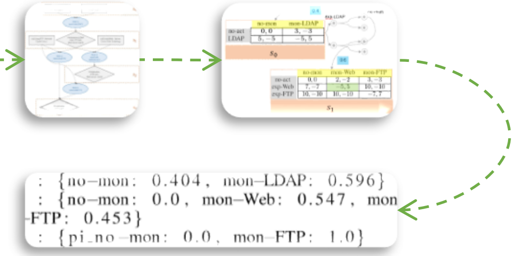


Western Region Cybersecurity Defense Competition (WRCCDC)

- Gather information about new vulnerabilities (OpenVAS) and average network performance over a time period T (MiniNET simulations).
- Use this information to precompute a strategy by solving the formulated Markov Game described in this work.
  - After ever time period $t << T$, randomly select switching strategy and change the IDS deployment.
  - Repeat.

Implementation in ThothLab

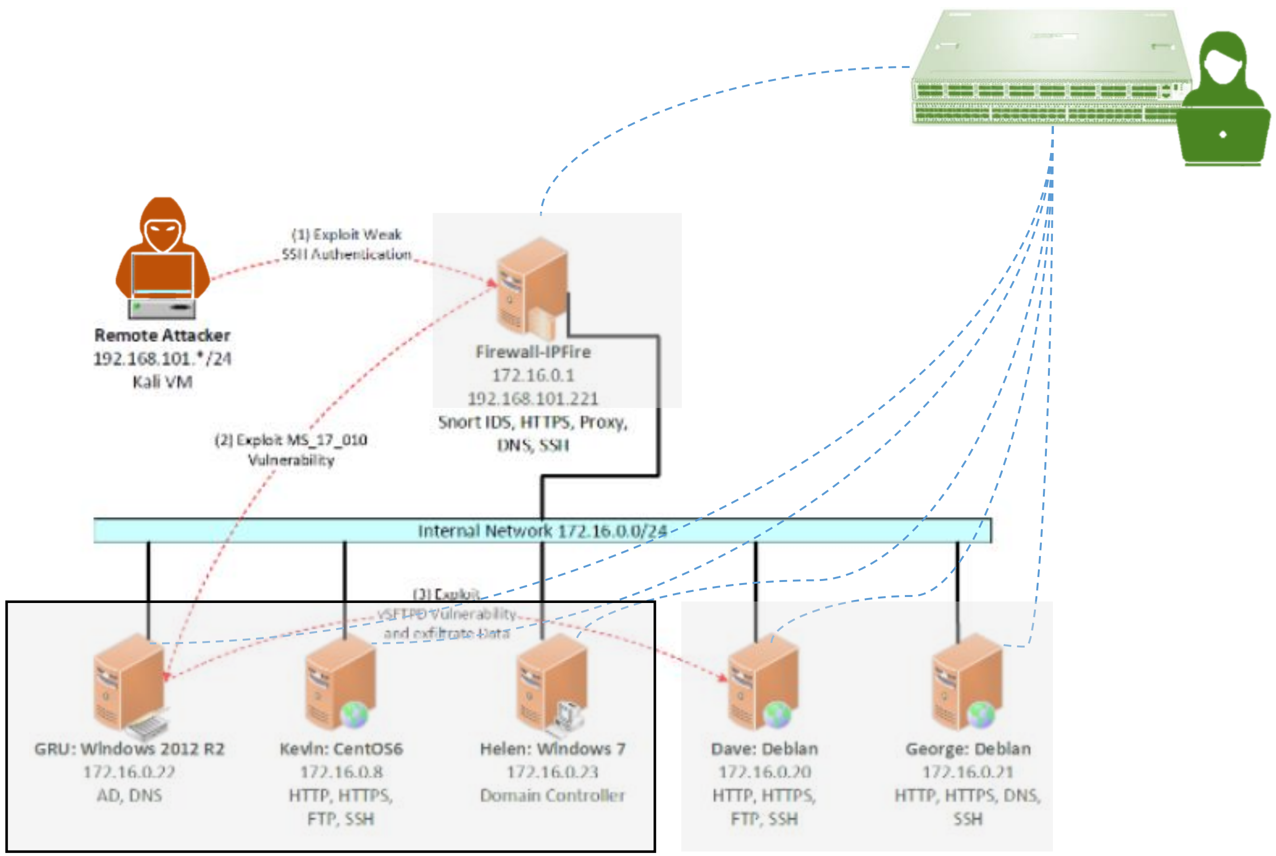




- Gather information about new vulnerabilities (OpenVAS) and average network performance over a time period *T* (MiniNET simulations).
- Use this information to precompute a strategy by solving the formulated Markov Game described in this work.
  - After ever time period *t* << *T*, randomly select switching strategy and change the IDS deployment.
  - Repeat.

ASU Arizona State University

Implementation in ThothLab



Western Region Cybersecurity Defense Competition (WRCCDC)

5 vulnerabilities (1H, 2M, 2L)
2 IDSs can be placed

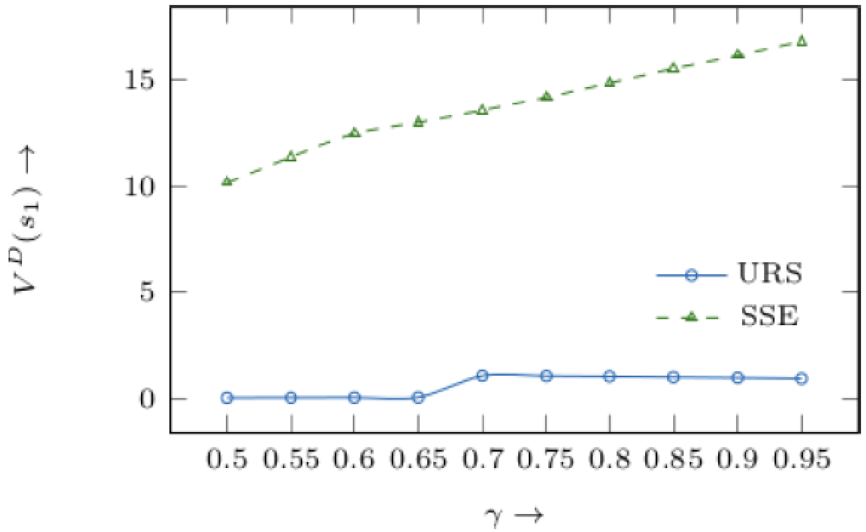
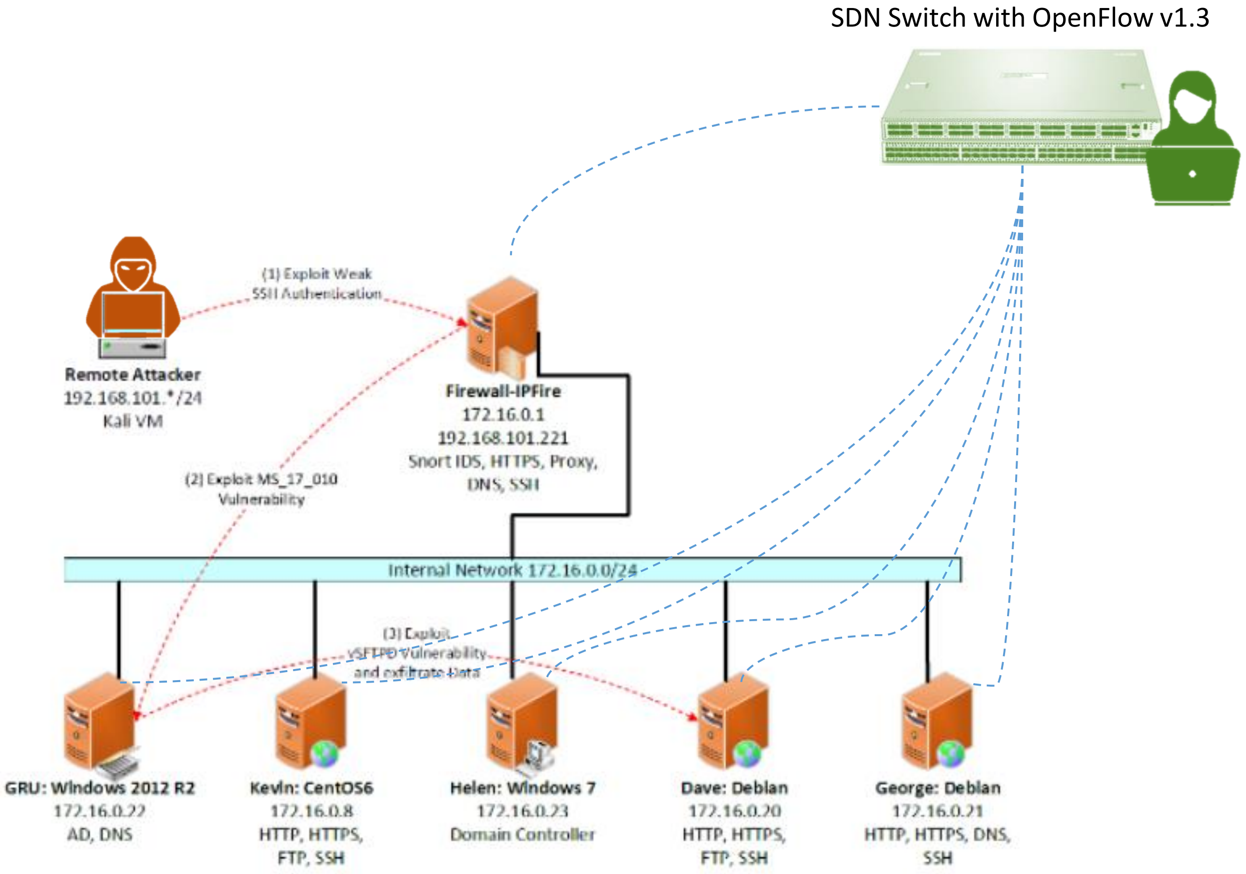


Fig. 6. Defender's value for the state s_1 as discount factor increases from 0.5 to 1.

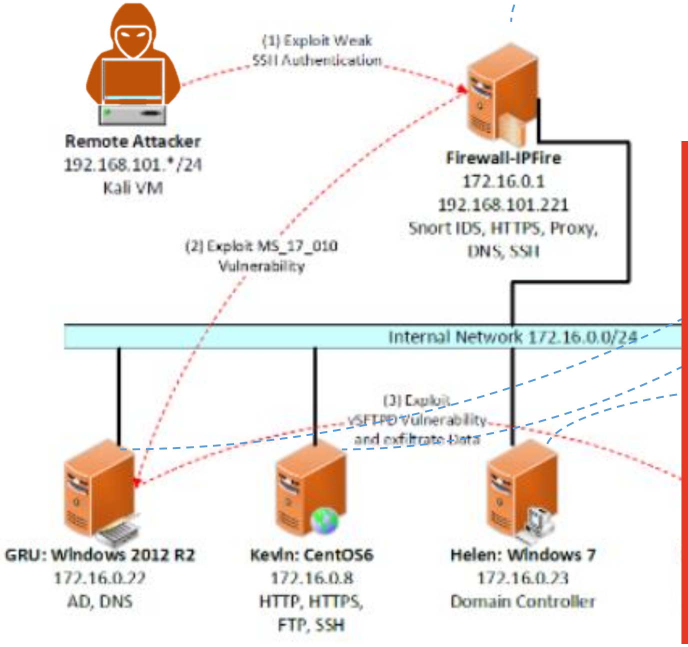
Conclusion & Future Work



- We formulated the placement of IDS systems in the cloud as a General Sum Markov Game. We found strategies for efficient detection surface shifting which allows the defender to trade-off between Security and Quality of Service. We showed its effectiveness on simulated data and emulation environments.

- We hope to relax a set of assumptions we made in this work in the future—
    - Game states are visible to both the players?
    - What happens when this is simulated in a real-world cloud network? How to obtain real-world attack data?
    - How does the incomplete knowledge of existing attacks and irrationality of attackers affect the quality of solution?
    - How does one reason about the zero day attacks – incomplete knowledge of the defender about the attacks?

Conclusion & Future Work



- We formulated the placement of IDS systems in the cloud as a General Sum Markov Game. We found strategies for efficient detection surface shifting which allows the defender to trade-off between Security and Quality of Service. We showed its effectiveness on simulated data and emulation environments.

- We hope to relax a set of assumptions we made in this work in the future—
  - Game states are visible to both the players?
  - What happens when this is simulated in a real-world cloud network? How to obtain real-world attack data?
  - How does the incomplete knowledge of existing attacks and irrationality of attackers affect the quality of solution?
  - How does one reason about the zero day attacks – incomplete knowledge of the defender about the attacks?