Cyber Attack Intent Recognition and Active Deception using Factored Interactive POMDPs

Aditya Shinde, Prashant Doshi
THINC Lab, Dept. of Computer Science
University of Georgia
Athens, GA
adityas,pdoshi@uga.edu

Omid Setayeshfar
Institute for Cyber Security and Privacy, Dept. of Computer Science
University of Georgia
Athens, GA
omid.s@uga.edu

ABSTRACT

This paper presents an intelligent and adaptive agent that employs deception to recognize a cyber adversary's intent on a honeypot host. Unlike previous approaches to cyber deception, which mainly focus on delaying or confusing the attackers, we focus on engaging with them to learn their intent. We model cyber deception as a sequential decision-making problem in a two-agent context. We introduce factored finitely-nested interactive POMDPs (I-POMDP$_\mathcal{X}$) and use this framework to model the problem with multiple attacker types. Our approach models cyber attacks on a single honeypot host across multiple phases from the attacker's initial entry to reaching its adversarial objective. The defending I-POMDP$_\mathcal{X}$-based agent uses decoys to engage with the attacker at multiple phases to form increasingly accurate predictions of the attacker's behavior and intent. The use of I-POMDPs also enables us to model the adversary's mental state and investigate how deception affects their beliefs. Our experiments in both simulation and with the agent deployed on a host system show that the I-POMDP$_\mathcal{X}$-based agent performs significantly better at intent recognition than commonly used deception strategies on honeypots. This emerging application of autonomous agents offers a new approach that contrasts with the traditional action-reaction dynamic that has defined interactions between cyber attackers and defenders for years.

KEYWORDS

Cyber deception; Emerging application; Honeypots; Multi-agent decision making

ACM Reference Format:
Aditya Shinde, Prashant Doshi and Omid Setayeshfar. 2021. Cyber Attack Intent Recognition and Active Deception using Factored Interactive POMDPs. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 9 pages.

1 INTRODUCTION

An important augmentation of conventional cyber defense utilizes deception-based cyber defense strategies [17]. These are typically based on the use of decoy sandbox systems called *honeypots* [22], which are instrumented with extensive monitoring capabilities. Currently, honeypots tend to be passive systems with the purpose of consuming the attacker's CPU cycles and time, and possibly logging the attacker's actions. However, the information inferred about the attackers' precise intent and capability is usually minimal.

On the other hand, honeypots equipped with fine-grained logging abilities offer an untapped opportunity to better understand attackers' intent and capabilities. We may achieve this by engaging and manipulating the attacker to perform actions that reveal his or her true intent. One way of accomplishing this is to employ active deception. Active strategies entail adaptive deception which seeks to influence the attackers' beliefs and manipulates the attackers into performing desired actions [13]. We investigate how multi-agent decision making can be used toward automating adaptive deception strategies to better understand the attacker.

We represent cyber deception on a single host as an interactive decision-making problem between a defender and an attacker. We introduce a factored variant of the well-known interactive partially observable Markov decision process [10], labeled as I-POMDP$_\mathcal{X}$, to computationally model the decision making of the defender while reasoning about the attacker's beliefs, capabilities, and preferences as both agents act and observe. I-POMDP$_\mathcal{X}$ exploits the factored structure of the problem, representing the dynamics and observation function using algebraic decision diagrams, and solving the model using a method that directly operates on these factored representations [1]. This brings a new level of tractability to an otherwise intractable framework, sufficient to adequately solve the cyber deception domain. I-POMDP$_\mathcal{X}$ explicitly models the beliefs of the attacker and the defender throughout the interaction. This allows for detailed inferences about how specific deceptive actions affect the attacker's subjective view of the system.

We evaluate the performance of I-POMDP$_\mathcal{X}$ in promoting active deception with multiple attacker types both in simulation and with the agent deployed on a system host. The attacker types are realistic implementing the techniques in MITRE's ATT&CK matrix [23] as attacker actions using Metasploit [16], which is a well-known tool for exploiting known vulnerabilities and penetration testing. Our results show that the I-POMDP-based agent learns the intent of the attacker much more accurately compared to baselines that do not engage the attacker or immediately deploy all decoys en masse. These baselines are similar to commonly-used deception strategies on existing honeypots.

2 BACKGROUND ON INTERACTIVE POMDPS

Interactive POMDPs (I-POMDPs) are a generalization of POMDPs to sequential decision-making in multi-agent environments [4, 10]. Formally, an I-POMDP for agent i in an environment with one other agent j is defined as,

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online.

Table 1: The state of the cyber deception domain is comprised of 11 variables resulting in a total of 4,608 states.

State Variable Name	Values	Description
`HOST_HAS_DATA`	`sensitive_data, critical_data, none`	Type of valuable data on the system
`S_DATA_DECOYS`	`yes, no`	Presence of sensitive data decoys
`C_DATA_DECOYS`	`yes, no`	Presence of critical data decoys
`PRIVS_DECEPTION`	`user, root, none`	Deceptive reporting of privileges
`DATA_ACCESS_PRIVS`	`user, root`	Privileges required to access or find data
`ATTACKER_PRIVS`	`user, root`	Attacker's highest privileges
`DATA_FOUND`	`yes, no`	Valuable data found by the attacker
`VULN_FOUND`	`yes, no`	Local privilege escalation (*PrivEsc*) discovered by attacker
`IMPACT_CAUSED`	`yes, no`	Attack successful
`ATTACKER_STATUS`	`active, inactive`	Presence of attacker on the host
`HOST_HAS_VULN`	`yes, no`	Presence of local *PrivEsc* vulnerability

$$\text{I-POMDP}_i = \langle IS_i, A, T_i, \Omega_i, O_i, R_i, OC_i \rangle$$

• IS_i denotes the interactive state space. This includes the physical states S as well as models of the other agent M_j, which may be intentional or subintentional [3]. *In this paper, we ascribe intentional models to the other agent as they model the other agent's beliefs, capabilities, and preferences as a rational agent.*
• $A = A_i \times A_j$ is the set of joint actions of both agents.
• T_i represents the transition function, T_i: $S \times A \times S \longrightarrow [0, 1]$. The transition function is defined over the physical states and excludes the other agent's models. This is a consequence of the model non-manipulability assumption, which states that an agent's actions do not directly influence the other agent's models.
• Ω_i is the set of agent i's observations.
• O_i is the observation function, O_i: $S \times A \times \Omega \longrightarrow [0, 1]$. The observation function is defined over the physical state space only as a consequence of the model non-observability assumption, which states that other's private model parameters may not be observed directly.
• R_i defines the reward function for agent i, R_i: $S_i \times A \longrightarrow \mathbb{R}$. The reward function for I-POMDPs usually assigns preferences over the physical states and actions only.
•OC_i is the subject agent's optimality criterion, which may be a finite horizon H or a discounted infinite horizon where the discount factor $\gamma \in (0, 1)$.

We limit our attention to a finitely nested I-POMDP, in which the interactive state space $IS_{i,l}$ at strategy level l is defined bottom up as,

$$\begin{aligned}
IS_{i,0} &= S, & \Theta_{j,0} &= \{\langle b_{j,0}, \hat{\theta}_j \rangle : b_{j,0} \in \Delta(IS_{j,0})\} \\
IS_{i,1} &= S \times \Theta_{j,0}, & \Theta_{j,1} &= \{\langle b_{j,1}, \hat{\theta}_j \rangle : b_{j,1} \in \Delta(IS_{j,1})\} \\
&\vdots \\
IS_{i,l} &= S \times \Theta_{j,l-1}, & \Theta_{j,l} &= \{\langle b_{j,l}, \hat{\theta}_j \rangle : b_{j,l} \in \Delta(IS_{j,l})\}.
\end{aligned}$$

Above, $\hat{\theta}_j$ denotes agent j's frame, defined as $\hat{\theta}_j = \langle A_j, \Omega_j, T_j, O_j, R_j, OC_j \rangle$. Here, OC_j represents j's optimality criterion and the other terms are as defined previously. $\Theta_{j,l}$ is the set of agent j's intentional models, each of which is defined as $\theta_{j,l} = \langle b_{j,l}, \hat{\theta}_j \rangle$. The interactive state space is typically restricted to a finite set of j's models, which are updated after every interaction to account for the belief update of agent j. The interactive state space for agent i at level l can be then defined as,

$$IS_{i,l} = S \times \text{Reach}(\Theta_{j,l-1}, H), \quad \Theta_{j,l} = \{\langle b_{j,l}, \hat{\theta}_j \rangle : b_{j,l} \in \Delta(IS_{j,l})\}.$$

Here, $\text{Reach}(\Theta_{j,l-1}, H)$ is the set of level $l-1$ models that j could have in H steps; $\text{Reach}(\Theta_{j,l-1}, 0) = \Theta_{j,l-1}$. We obtain $\text{Reach}(\Theta_{j,l-1}, H)$) by repeatedly updating j's beliefs in the models in $\Theta_{j,l-1}$.

The value function for I-POMDPs maps agent i's type $\Theta_{i,l} \longrightarrow \mathbb{R}$, and is piecewise linear and convex. Analogously to POMDPs, we may decompose the value function into its components:

$$V^t(b_{i,l}, \hat{\theta}_i) = \sum_{is \in IS_{i,l}} \alpha^t(is) \times b_{i,l}(is) \tag{1}$$

where

$$\alpha^t(is) = \max_{a_i \in A_i} \Bigg\{ \sum_{is} R_i(s, a_i, a_j) P(a_j|M_j) + \gamma \sum_{o_i' \in \Omega_i} \sum_{is'} \sum_{a_j} P(a_j|M_j)$$
$$\Bigg[T_i(s, a_i, a_j, s')\ O_i(s', a_i, a_j, o_i') \sum_{o_j' \in \Omega_j} O_j(s', a_i, a_j, o_j')$$
$$\times \tau_{\theta_j^t}(b_{j,l-1}, a_j, o_j', b_{j,l-1}') \Bigg] \alpha^{t+1}(is').$$

Here, $\tau_{\theta_j^t}(b_{j,l-1}, a_j, o_j', b_{j,l-1}')$ denotes the recursive belief update of j's beliefs in an intentional model. For small problem domains, we may compute the α-vectors exactly using dynamic programming as shown in Gmytrasiewicz and Doshi [10].

3 A NOVEL CYBER DECEPTION DOMAIN

Engaging and deceiving human attackers into intruding controlled systems and accessing obfuscated data offers a proactive approach to computer and information security. It wastes attacker resources and potentially misleads the attacker. Importantly, it offers an untapped opportunity to understand the attackers' beliefs, capabilities, and preferences and how they evolve by sifting the detailed activity logs. Identifying these mental and physical states not only informs

Table 2: The nine actions available to the attacker.

Action name	States affected	Description
FILE_RECON_SDATA	DATA_FOUND	Search for sensitive data for theft
FILE_RECON_CDATA	DATA_FOUND	Search for critical data for manipulation
VULN_RECON	VULN_FOUND	Search for local *PrivEsc* vulnerability
PRIV_ESC	ATTACKER_PRIVS	Exploit local *PrivEsc* vulnerability
CHECK_ROOT	none	Check availability of root privileges
START_EXFIL	IMPACT_CAUSED	Download data from target
PERSIST	IMPACT_CAUSED	Establish a permanent presence in the system
MANIPULATE_DATA	IMPACT_CAUSED	Manipulate stored data
EXIT	ATTACKER_STATUS	Terminate the attack

the defender about the attacker's intent, but also guides new ways of deceiving the attacker. In this section, we introduce the domain of cyber deception and subsequently discuss how it can be modeled in a factored I-POMDP in the next section.

The cyber deception domain models the interactions between the attacker and the defender on a single honeypot host system. In our domain, we capture specific aspects of a host which are likely to be affected by an attacker's actions. To this end, we use techniques listed in the MITRE ATT&CK matrix [23] as our reference. A state of the interaction is modeled using 11 state variables defining a total of **4,608 states**. Table 1 briefly summarizes the state space. The HOST_HAS_DATA variable represents the true type of valuable data on the system. We assume that a system cannot have two different types of valuable data simultaneously. This is a reasonable assumption because usually different hosts on enterprise networks possess different assets. We differentiate between sensitive_data and critical_data as distinct targets. Sensitive data, for example, includes private data of employees, high ranking officials, or any data that the attacker would profit from stealing. Also, in practical scenarios, honeypots never contain any real valuable data. Consequently, in the cyber deception domain, the value of HOST_HAS_DATA is typically none. However, the attacker is unaware that the host is a honeypot or in the presence of data decoys and hence forms a belief over this state variable. Thus, the HOST_HAS_DATA variable gives a subjective view of the attacker being deceived. The S_DATA_DECOYS and C_DATA_DECOYS state variables represent the presence of sensitive-data decoys and critical-data decoys. These decoys, identical to real data, are commonly used and make it challenging for the attacker to differentiate them from real data [24].

We include three different types of attackers; the *data exfil* attacker, *data manipulator*, and *persistent threat*. The *data exfil* attacker represents a threat that aims to steal valuable private data from the host. The *data manipulator* attacker represents a threat that seeks to manipulate data that is critical for the operation of a business or a physical target. Thus, the *data exfil* attacker targets sensitive_data in the system and the *data manipulator* attacker targets critical_data. The *persistent threat* attacker wants to establish a strong presence in the system at a high privilege level.

There are 5 observation variables for the attacker which make a total of **48 unique observations**. The attacker in the interaction can perform one of **9 actions** to gather information about the system, manipulate the system, or take action on objectives. Table 2 briefly summarizes the actions available to the attacker. The FILE_RECON_SDATA and FILE_RECON_CDATA actions cause the DATA_FOUND variable to transition to yes with high probabilities that depend on the type of data (finding critical data is more likely than sensitive data). The FILE_RECON_SDATA action is slightly worse at finding data than the FILE_RECON_CDATA. This reflects the fact that private sensitive information is slightly difficult to find because it is often stored in user directories in arbitrary locations. On the other hand, critical data, like service configuration or database files, are stored in well-known locations on the system. The attacker gets information about the DATA_FOUND transition through the DATA observation variable. It simulates the data discovery phase of an attack. VULN_RECON is another action that works similarly and causes the VULN_FOUND to transition to yes. This transition depicts the attacker looking for vulnerabilities to raise privileges. Depending on the type of the attacker, the START_EXFIL, MANIPULATE_DATA, or PERSIST actions can be performed to achieve the attacker's main objectives. As the attacker is unable to discern between decoy data and real data, hence, is unable to determine which variable influences the DATA_FOUND state transition during file discovery. The attacker, however, can distinguish between different types of valuable data. So, if the system contains data that is different from what the attacker expects, the attacker can observe this from the DISCREPANCY observation variable. As DATA and DISCREPANCY are separate observation variables, the attacker can observe a discrepancy even when data has been found. When this occurs, the attacker may start believing in the presence of decoys and develops a belief over the decoy data states as the host may not contain real data. This realistically models a situation in which the attacker encounters multiple decoys of different types and suspects deception.

The defender in the interaction starts with complete information about the system. The defender's actions mostly govern the deployment and removal of different types of decoys. These actions influence the S_DATA_DECOYS and C_DATA_DECOYS states. Additionally, the defender can influence the attacker's observations about his or her privileges through the PRIVS_DECEPTION state. The defender gets perfect observations whenever the attacker interacts with a decoy. Additionally, the defender gets stochastic observations about the attacker's actions through the LOG_INFERENCE observation variable, which represents low-level log analysis. The attacker is rewarded for exiting the system after causing an impact. For the *data exfil* and *data manipulator* attacker types, this is achieved by performing the START_EXFIL and MANIPULATE_DATA actions respectively. The *persistent threat* attacker is rewarded for getting root level persistence in the system.

4 FACTORED I-POMDP FOR MODELING CYBER DECEPTION

Factored POMDPs have been effective toward solving structured problems with large state and observation spaces [7, 18]. Motivated by this observation, we extend the finitely-nested I-POMDP reviewed in Section 2 to its factored representation, I-POMDP$_\mathcal{X}$. Formally, this extension is defined as:

$$\text{I-POMDP}_\mathcal{X} = \langle \mathcal{IS}_i, A, T_i, \mathcal{Y}_i, O_i, \mathcal{R}_i, OC_i \rangle$$

• $\mathcal{IS}_i$ is the factored interactive state space consisting of physical state factors $\mathcal{X}$ and agent j's models M_j. In a finitely-nested I-POMDP$_\mathcal{X}$ the set M_j is bounded similarly to finitely-nested I-POMDPs.
• Action set A is defined exactly as before in Section 2. We use algebraic decision diagrams (ADDs) [1] to represent the factors for agent i's transition, observation, and reward functions compactly.
• T_i defines the transition function represented using ADDs as $P^{a_i}(\mathcal{X}'|\mathcal{X}, A_j)$ for all $a_i \in A_i$.
• $\mathcal{Y}_i$ is the set of observation variables which make up the observation space.
• O_i is the observation function represented as ADDs, $P^{a_i}(\mathcal{Y}_i'|\mathcal{X}', A_j)$.
• $\mathcal{R}_i$ defines the reward function for agent i. The reward function is also represented as an ADD, $\mathcal{R}^{a_i}(\mathcal{X}, A_j)$.
• Optimality criterion OC_i is as defined previously in Section 2.

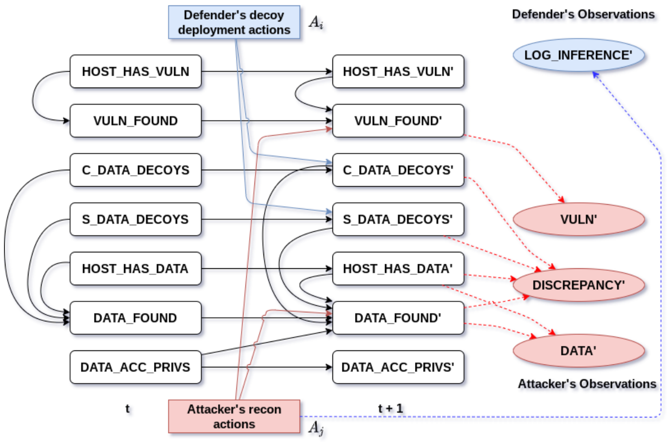
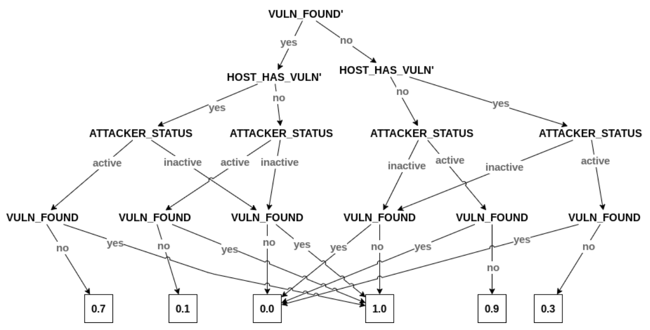
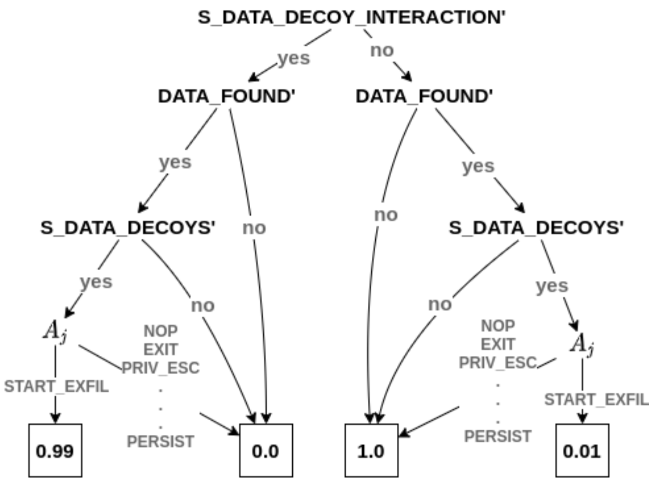


Figure 1: Dynamics compactly represented as a two time-slice DBN for select joint actions and observation variables in I-POMDP$_\mathcal{X}$.

We illustrate I-POMDP$_\mathcal{X}$ by modeling the cyber deception domain of Section 3 in the framework. Figure 1 shows the DBN for select state and observation variables given that the attacker engages in reconnaissance actions. The two slices in the DBN represent the sets of pre- and post-action state variables, $\mathcal{X} = \{X_1, ..., X_n\}$ and $\mathcal{X}' = \{X_1', ..., X_n'\}$ where X_n represents a single state variable. Similarly, $\mathcal{Y}_i' = \{Y_{i_1}', ..., Y_{i_n}'\}$ and $\mathcal{Y}_j' = \{Y_{j_1}', ..., Y_{j_n}'\}$ denote the sets of observation variables for agents i and j, respectively. The ADD $P^{a_i}(\mathcal{X}'|\mathcal{X}, A_j) = P^{a_i}(X_1'|X_2', \ldots, X_n', \mathcal{X}, A_j) \times P^{a_i}(X_2'|X_3', \ldots, X_n', \mathcal{X}, A_j) \times \ldots \times P^{a_i}(X_n'|\mathcal{X}, A_j)$ represents the complete transition function for action $A_i = a_i$. This is analogous to the *complete action diagram* defined by Hoey et al. [14] for MDPs. Similarly, the observation function is represented using the ADD, $P^{a_i}(\mathcal{Y}_i'|\mathcal{X}', A_j) = P^{a_i}(Y_{i_1}'|\mathcal{X}', A_j) \times P^{a_i}(Y_{i_2}'|\mathcal{X}', A_j) \times \ldots \times P^{a_i}(Y_{i_n}'|\mathcal{X}', A_j)$ which is analogous to the *complete observation diagram* [7]. We illustrate example ADDs that are a part of the transition and observation function for A_i = NOOP in Fig. 2. In total, the transition function is composed of **792 ADDs** whereas **216 ADDs** constitute the observation function.



(a) An ADD representing the transition function P^{NOP}(VULN_FOUND'$|\mathcal{X}, A_j$ = VULN_RECON).



(b) An ADD representing the observation function P^{NOP}(S_DATA_DECOY_INTR'$|\mathcal{X}', A_j$).

Figure 2: Example ADDs in the I-POMDP$_\mathcal{X}$ representation of the cyber deception domain.

Additionally, in an I-POMDP$_\mathcal{X}$, agent i also recursively updates the beliefs of agent j. The attacker types are modeled as frames in M_j. Let $M_j = \{m_{j_1} : \langle b_{j_1}, \hat{\theta}_{j_1} \rangle, ..., m_{j_n} : \langle b_{j_q}, \hat{\theta}_{j_r} \rangle\}$ be the set of all models in Reach$(\Theta_{j,l-1}, H)$. Because neither a_j nor o_j are directly accessible to agent i, they are represented as ADDs $P(A_j|M_j)$ and $P^{a_i}(\mathcal{Y}_j'|\mathcal{X}', A_j)$. The distribution over M_j' is then

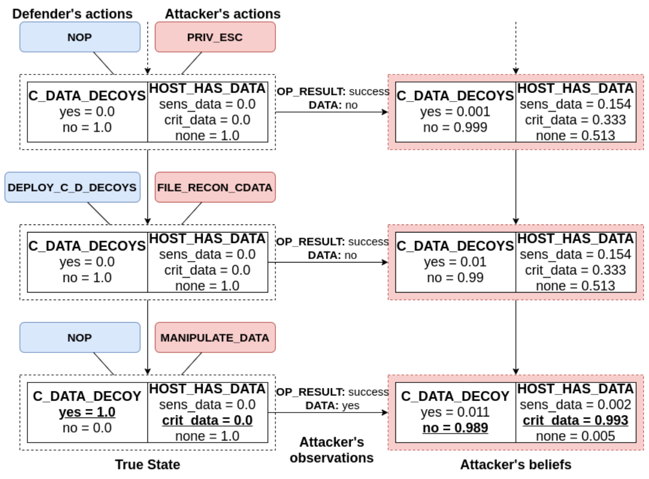


Figure 3: The true states of the system is shown on the left while the attacker's belief progression is shown on the right. The attacker starts with a low prior belief on the existence of decoys. If decoys are indistinguishable from real data, the attacker attributes his observation to the existence of real data even when the host has none.

$P^{a_i}(M'_j|M_j, \mathcal{Y}'_j, A_j, \mathcal{X}') = P^{a_i}(M'_j|M_j, A_j, \mathcal{Y}'_j) \times P^{a_j}(\mathcal{Y}'_j|\mathcal{X}', A_i)$. Using these factors, we can now define the distribution over $\mathcal{X}'$ and M'_j given action a_i and observation o_i as a single ADD using existential abstraction:

$$\begin{aligned} P^{a_i,o_i}(\mathcal{X}', M'_j|\mathcal{X}, M_j) &= \textstyle\sum_{A_j, \mathcal{Y}'_j} P^{a_i,o_i}(\mathcal{Y}'_j, M'_j, \mathcal{X}', A_j|M_j, \mathcal{X}) \\ &= \sum_{A_j} P(A_j|M_j)\, P^{a_i}(\mathcal{X}'|\mathcal{X}, A_j) \sum_{\mathcal{Y}'_j} P^{a_i}(\mathcal{Y}'_i|\mathcal{X}', A_j) \\ &\times P^{a_i}(M'_j|M_j, \mathcal{Y}'_j, \mathcal{X}', A_j). \end{aligned} \tag{2}$$

Here, the ADD $P^{a_i}(\mathcal{X}'|\mathcal{X}, A_j)$ compactly represents $T_i(s, a_i, a_j, s')$, $P^{a_i}(\mathcal{Y}'_i|\mathcal{X}', A_j)$ represents the probabilities $O_i(s, a_i, a_j, o'_i)$, $P(A_j|M_j)$ represents $P(a_j|\theta^t_j)$, and $P^{a_i}(M'_j|M_j, \mathcal{Y}'_j, \mathcal{X}', A_j)$ represents the recursive belief update transition $\tau_{\theta^t_j}(b_j, a_j, o'_j, b'_j) \times O_j(s, a_i, a_j, o'_j)$ of the original I-POMDP. Thus, the constructed ADD $P^{a_i,o_i}(\mathcal{X}', M'_j|\mathcal{X}, M_j)$ contains the transition probabilities for all interactive state variables given action a_i and observation o_i. The I-POMDP$_\mathcal{X}$ belief update can then be computed as:

$$b_i^{a_i,o_i}(\mathcal{X}', M'_j) = \sum_{\mathcal{X}, M_j} b(\mathcal{X}, M_j) \times P^{a_i,o_i}(\mathcal{X}', M'_j|\mathcal{X}, M_j) \tag{3}$$

where the ADD $P^{a_i,o_i}(\mathcal{X}', M'_j|\mathcal{X}, M_j)$ is obtained as in Eq. 2.

Symbolic Perseus [18] offers a relatively scalable *point-based* approximation technique that exploits the ADD structure of factored POMDPs. Toward generalizing this technique for I-POMDP$_\mathcal{X}$, we are aided by the existence of point-based value iteration for I-POMDPs [5], which approximates the complete belief space with a set of belief points for optimization. Subsequently, we may generalize the α-vectors and its backup from the latter to the factored representation of I-POMDP$_\mathcal{X}$:

$$\begin{aligned} &\Gamma^{a_i,*} \leftarrow \alpha^{a_i,*}(\mathcal{X}, M_j) = \sum_{A_j} R^{a_i}(\mathcal{X}, A_j) P(A_j|M_j) \\ &\Gamma^{a_i,o_i} \overset{\cup}{\leftarrow} \alpha^{a_i,o_i}(\mathcal{X}, M_j) = \gamma \sum_{\mathcal{X}', M'_j} P^{a_i,o_i}(\mathcal{X}', M'_j|\mathcal{X}, M_j)\alpha^{t+1}(\mathcal{X}', M'_j), \\ &\qquad\qquad \forall \alpha^{t+1} \in \mathcal{V}^{t+1} \\ &\Gamma^{a_i} \leftarrow \Gamma^{a_i,*} \oplus_{o_i} \underset{\Gamma^{a_i,o_i}}{\arg\max}(\alpha^{a_i,o_i} \cdot b_i), \\ &\mathcal{V}^t \leftarrow \underset{\alpha^t \in \bigcup_{a_i} \Gamma^{a_i}}{\arg\max} (\alpha^t \cdot b_i), \quad \forall b_i \in B_i. \end{aligned} \tag{4}$$

Here, $\mathcal{V}^{t+1}$ is the set of α-vectors from the next time step and b_i is a belief point from the set of considered beliefs B_i. A popular way of building B_i is to project an initial set of beliefs points forwards for H time steps using the belief update of Eq. 3.

5 EXPERIMENTS AND ANALYSIS

We modeled the cyber deception domain described in Section 3 from the perspective of a level-1 defender using I-POMDP$_\mathcal{X}$. We implemented the generalized Symbolic Perseus using the point-based updates of the α-vectors and the belief set projection as given in Section 4, to solve the I-POMDP$_\mathcal{X}$. The code for the I-POMDP$_\mathcal{X}$ solver and the supplementary material is available at https://github.com/dityas/Protos. The solver has several enhancements such as cached ADD computations and approximations for speed up.

5.1 Example Attacker-Defender Interactions

Figure 3 illustrates a scenario taken from an actual simulation run with the *data manipulator* attacker. Initially, the attacker has a non-zero belief over the existence of data. However, the true state of the system on the left shows that it does not actually contain any data. In the absence of the defender or any static data decoys, the attacker will eventually update his beliefs to accurately reflect the reality by performing the `FILE_RECON_CDATA` action and observing the result. To avoid this belief state, the defender deploys data decoys when the attacker acts. The attacker's inability to tell the difference between decoy data and real data and his prior belief about the absence of decoys leads him to attribute his observations to the existence of real data leading to the attacker being deceived.

Figure 4 shows a similar scenario, but with the *data exfil* attacker. This particular trace highlights the adaptive nature of the defender's policy. Initially, the defender believes the attacker to be of the *data manipulator* type. Hence, the defender deploys `critical_data` decoys. On receiving subsequent observations, it realizes this initial belief to be inconsistent with the observations. The defender takes corrective action by removing previously deployed decoys and deploying `sensitive_data` decoys instead.

5.2 Performance Evaluation

We evaluate the deception policy generated by I-POMDP$_\mathcal{X}$ in simulations and on an actual system consisting of a standalone attacker programmed via Metasploit [16] and a defender workstation. We simulate each attacker type on Metasploit using the optimal policy computed by the level-0 attacker POMDP. We show these policies

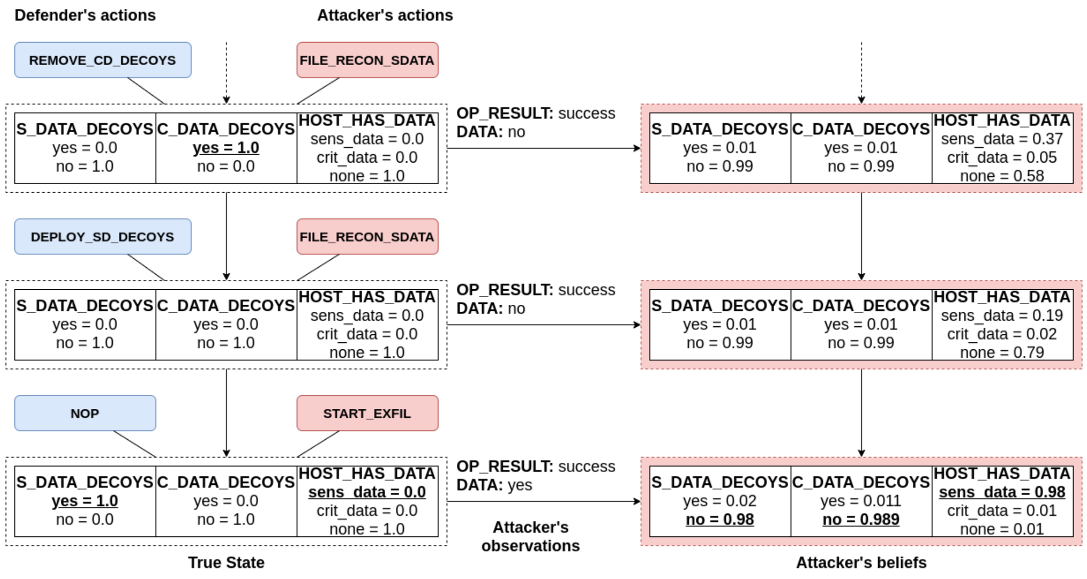

Defender's actions
Attacker's actions
REMOVE_CD_DECOYS
FILE_RECON_SDATA
S_DATA_DECOYS yes = 0.0 no = 1.0
C_DATA_DECOYS yes = 1.0 no = 0.0
HOST_HAS_DATA sens_data = 0.0 crit_data = 0.0 none = 1.0
OP_RESULT: success DATA: no
S_DATA_DECOYS yes = 0.01 no = 0.99
C_DATA_DECOYS yes = 0.01 no = 0.99
HOST_HAS_DATA sens_data = 0.37 crit_data = 0.05 none = 0.58
DEPLOY_SD_DECOYS
FILE_RECON_SDATA
S_DATA_DECOYS yes = 0.0 no = 1.0
C_DATA_DECOYS yes = 0.0 no = 1.0
HOST_HAS_DATA sens_data = 0.0 crit_data = 0.0 none = 1.0
OP_RESULT: success DATA: no
S_DATA_DECOYS yes = 0.01 no = 0.99
C_DATA_DECOYS yes = 0.01 no = 0.99
HOST_HAS_DATA sens_data = 0.19 crit_data = 0.02 none = 0.79
NOP
START_EXFIL
S_DATA_DECOYS yes = 1.0 no = 0.0
C_DATA_DECOYS yes = 0.0 no = 1.0
HOST_HAS_DATA sens_data = 0.0 crit_data = 0.0 none = 1.0
OP_RESULT: success DATA: yes
S_DATA_DECOYS yes = 0.02 no = 0.98
C_DATA_DECOYS yes = 0.011 no = 0.989
HOST_HAS_DATA sens_data = 0.98 crit_data = 0.01 none = 0.01
True State
Attacker's observations
Attacker's beliefs

Figure 4: The defender takes corrective actions if the attacker is incorrectly predicted in the initial steps. In this case, on observing file discovery actions, the defender deploys critical data decoys. Later, the defender forms a better belief over the attacker's frame from the observations and replaces the decoys before the attacker discovers the discrepancy.

for each type of attacker in the **supplementary material**. For the simulations, we randomly sample the frame and the starting privileges of the attacker to simulate a threat with unknown intentions and privileges. The defender begins knowing about the existence of decoys on the system. The attacker, on the other hand, does not have prior knowledge about any vulnerabilities or data on the system. The defender engages with the attacker by deploying decoys, facilitating deceptive observations, or adding known vulnerabilities to the system. In the simulations, the state transitions and observations for both agents are generated by sampling from the joint transition functions and individual observation functions.

5.2.1 Simulations. In the simulations, the state transitions and observations for both agents are generated by sampling from the joint transition functions and individual observation functions. We compare the I-POMDP$_\mathcal{X}$ policy against other baselines: one that does not engage and passively observes the attacker, and another which uses deception indiscriminately having deployed both sensitive and critical data decoys and all vulnerabilities in the honeypot at the beginning. We label the first baseline as NO-OP(no decoy) and the second baseline as NO-OP(all decoys). These baselines represent approaches commonly in use on current honeypots.

We perform the simulations for 30 trials with an attacker type randomly picked in each trial. During each trial, the defender begins not knowing the type of the attacker and believes that the state is that the attacker's privileges are not known. We set H in Reach($\Theta_{j,l-1}, H$) to 5. We start with 100 opponent models and set the discount factor $\gamma = 0.95$. As the interaction progresses, the beliefs of the other agent are updated recursively and the number of opponent models grows accordingly. The generalized Symbolic Perseus is then run on 200 projected belief points until convergence to obtain the policy, which prescribes the subsequent actions for the defender until the end of the trial. It converges in about 6 minutes with a mean time per backup of 37 secs on an Ubuntu 18 PC with Intel i7 and 64 GB RAM.

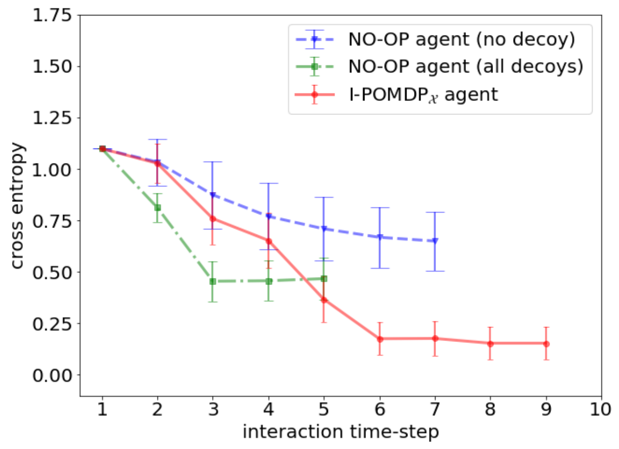


Figure 5: Cross entropy (KL divergence) of the beliefs of the I-POMDP$_\mathcal{X}$ agent and other baselines in simulations. Cross entropies near zero signify good intent recognition. When engaging a defender-unaware attacker, I-POMDP$_\mathcal{X}$-based defender outperforms existing honeypot strategies in engaging the attackers and recognizing their intent.

The NO-OP(no decoy) and NO-OP(all decoy) yielded a mean (± std dev.) of **4.42 ± 0.78** and **2.93 ± 1.11** steps of engagement with the attacker, respectively. The longest engagement among these

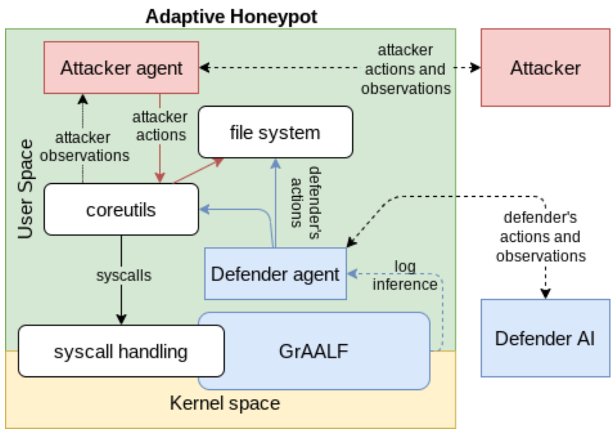


Figure 6: System architecture of the testbed used to deploy the agents. The defender manipulates the system through decoys and commonly used *coreutils* binaries to give deviant observations.

consisted of 7 and 6 steps, respectively. With NO-OP(no decoy), the attacker spends time searching for data and attempting to escalate his privileges but without much success, finally exiting the system. With NO-OP(all decoys), the attacker either quickly exploits the vulnerabilities or encounters the data decoys but quickly exits often due to the encountered data not being as expected. However, the I-POMDP$_\mathcal{X}$ agent engaged with the attacker for a mean duration of **5.81 ± 1.7** with the longest interaction happening for 9 steps. It leverages the information gained by the first few observations to avoid using decoys that the attacker would find suspicious. For example, the defender first manipulates the attacker's observations about her own privileges. This increases the defender's chances of observing file enumeration or vulnerability discovery activity, forming a belief over the frames. Subsequently, the defender baits the attacker using decoys and observes the interaction to solidify his belief. This minimizes the risk of the attacker encountering unexpected decoys or noticing discrepancies.

These simulations are predicated on the level-1 defender believing that none of the level-0 attacker types are aware of the deception, which is the typical case. However, if the defender's strategy level is 2 and it believes that the attacker believes that there is a small chance at 0.1 of decoys being used, we observed that the attacker often quickly exited the system as one would expect.

Do the extended engagements facilitated by the I-POMDP$_\mathcal{X}$ agent help in intent recognition? Figure 5 shows the cross-entropy between the defender's belief of the attacker's frame and the attacker's true type, as it varies across the steps of the interaction. The defender's I-POMDP$_\mathcal{X}$ policy eventually yields the lowest cross-entropy values compared to the baselines, often reaching zero in 6 steps. We show the cross-entropy for more steps because the attacker remains in the system performing a few more actions. The sharp decrease in cross-entropy in the first three steps is because the attacker's decoy interactions (if the attacker is of type *data exfil* or *manipulator*) are perfectly observed by the defender (some other interactions generate noisy observations). Multiple consecutive data reconnaissance steps filter out the persistence attacker type, and the final step of either exfiltrating the data or manipulating it allows the defender to distinguish between the remaining two attacker types. But, for the NO-OP(no decoy) with no deception, the only source of information about the attacker is his general actions, which is noisy. Hence, such a defender is unable to form accurate beliefs before the attacker leaves the interaction. For the NO-OP(all decoy) agent that indiscriminately uses deception, observations from decoy interactions are perfect, but the risk of the attacker encountering contradicting decoys and suspecting deception is also high leading to early exits.

5.2.2 *Host deployment.*

In our next phase of experimentation, we evaluated the real-world feasibility of deploying an operational I-POMDP$_\mathcal{X}$ on a host system and testing its efficacy. The testbed consists of 3 separate hosts: the *attacker*, the *adaptive honeypot* and the *defender*. Figure 6 shows the overall architecture of our testbed. The *attacker* system runs a Kali Linux distribution which is well known for the variety of offensive and defensive cybersecurity tools that are preinstalled on it. The *adaptive honeypot* on which the interaction takes place runs a Metasploitable 3 Linux distribution. This distribution has a wide range of builtin vulnerabilities and is commonly used to simulate victim workstations in cyber attack simulations. The *adaptive honeypot* also contains an attacker agent that executes the attacks and communicates with the *attacker*. The *attacker agent* implements the actions given by the attacker's optimal plan located on the attacker host using realistic techniques commonly used by real attackers. *We implement real exploits to facilitate privilege escalation on the host.* The *adaptive honeypot* also has a defender agent that implements the defender's actions and gets observations.

The defender AI located on the *defender* workstation solves the I-POMDP$_\mathcal{X}$ and computes the optimal action. For implementing the observation function, the I-POMDP$_\mathcal{X}$ agent monitors and analyzes the system logs to get information about the attacker's actions (i.e., observations). To enable this, we use GrAALF [20], a graphical framework for processing and querying system call logs. GrAALF analyzes system call logs in real-time and provides the stochastic `LOG_INFERENCE` observation variable values (pertaining to file and vulnerability searches) as well as the perfectly observed `DATA_DECOY_INTERACTION` variable values to the defender.

Our results in Fig. 7 show the adaptive deception strategy employed by the I-POMDP$_\mathcal{X}$ agent is better at engaging adversaries on a honeypot as compared to the deception strategies that are commonly used. While the cross entropy does not reach zero due to the challenge of accurately inferring the attacker's actions from the logs (leading to noisier observations), it gets close to zero, which indicates accurate intent recognition.

In addition to these experiments, we also evaluate our I-POMDP$_\mathcal{X}$ based defender against human attackers. We provide more details on these experiments in the supplementary material.

6 RELATED WORK

Agent methods are beginning to be explored for use in cyber deception. An area of significant recent interest has been game-theoretic

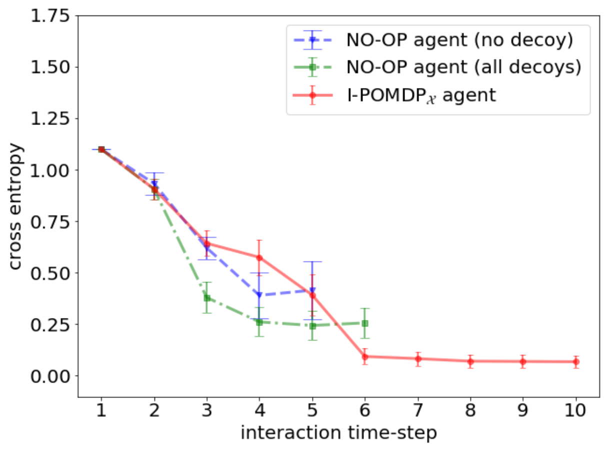


Figure 7: On the actual host deployment, the I-POMDP$_\mathcal{X}$-based agent uses implemented deception techniques and audit log analysis for observations, to engage with the attacker for longer duration than other agents and form more informative beliefs.

multi-agent modeling of cyber deception, which contrasts with the decision-theoretic modeling adopted in this paper.

Schlenker et al. [19] introduced cyber deception games based on Stackelberg games [21]. These model deception during the network reconnaissance phase when the attacker is deceived into intruding a honeypot. Another similar approach [6] allocates honeypots in a network using a Stackelberg game. The game uses attack graphs to model the attacker and creates an optimal honeypot allocation strategy to lure attackers. Jajodia et al. [12] develop probabilistic logic to model deception during network scanning. While these efforts focus on static deployment of deception strategies at the network level, we seek active deception at the host level – once the attacker has entered the honeypot. Further, we model individual phases of the attack in greater detail. This leads to realistic deception techniques at each phase. Also, as opposed to the above approaches, our framework performs opponent modeling explicitly.

At the host level, Carroll et al. [2] models deception as a signaling game while Horak et al. [11] creates a model for active deception using partially observable stochastic games. However, both of these take a high-level view modeling defender actions rather abstractly. In contrast, our defender actions are realistic and can be implemented on honeypots as demonstrated in Section 5. Ferguson-Walter et al. [8] model possible differences between the attacker's and defender's perceptions toward the interaction by modeling cyber deception as a hypergame [15]. Hypergames model different views of the game being played from the perspective of the players. While this approach, similar to ours, represents the attacker's perspective of the game, we explicitly model the adversary using a subjective decision-theoretic approach and do not solve for equilibrium.

7 CONCLUDING REMARKS

Our approach of utilizing automated decision making for deception to recognize attacker intent is a novel application of autonomous agents and, specifically decision making in cyber security. It elevates the extant security methods from anomaly and threat detection to intent recognition. We introduced a factored variant of the well-known I-POMDP framework, which exploits the environment structure and utilized it to model the new cyber deception domain. Our experiments revealed that the I-POMDP$_\mathcal{X}$-based agent succeeds in engaging various types of attackers for a longer duration than current honeypot strategies, which facilities intent recognition. Importantly, the agent is practical on a real system with logging capabilities paving the way for its deployment in actual honeypots.

On a broader scale, the I-POMDP$_\mathcal{X}$ framework that we introduce makes I-POMDPs tractable to be applied to larger problems. This has a multitude of applications such as in negotiations, studying human behavior, cognition, and security. Another area that we hope to motivate through our research is a model of deception in human interactions. Modeling other agents explicitly will help understand how deceptive or real information influences an individual's beliefs. This research can eventually motivate further research in areas such as counter deception and deception resilience in agents.

At an application level, our work aims to motivate the use of autonomous agents and decision making to create informed cyber defense strategies. It provides a new perspective different from the traditional action-reaction dynamic that has defined interactions between cyber attackers and defenders for years. The recent release of MITRE's SHIELD matrix [9], a knowledge base on active defense, signifies a change in the approach towards cyber defense. Coincidentally, our I-POMDP$_\mathcal{X}$-based policy is tactically grounded in the SHIELD matrix. Specifically, the I-POMDP$_\mathcal{X}$-based policy uses deception to *detect* the adversary through decoy interactions, *facilitate* the adversary's progress to observe their actions, *legitimize* the deception through use of selective decoys, and finally *test* the adversary to determine their intent. The I-POMDP$_\mathcal{X}$-based defender strategically employs these tactics during different phases of the interaction. In addition, our framework models the opponent's mental states and preferences. This will aid security teams in understanding threats at a deeper level. The framework will potentially motivate the development of adaptive and intelligent deceptive solutions that can study and predict attackers at a deeper level. Understanding attackers' mental models, inherent biases, and preferences will go a long way in forming flexible cyber defense strategies that can adapt to different threats.

While Section 5.2.2 demonstrated that the deployment of the active deception agent is feasible, we are currently engaged in assessing the viability of such a deployment. Toward this, we intend to move the host with the deployment to outside our institutional firewall, further enhance its logging capabilities, while safeguarding it against common bot attacks.

ACKNOWLEDGMENTS

This research was supported, in part, by a grant from the Army Research Office under grant number W911NF-18-1-0288. We acknowledge feedback from the seminar participants in Dr. David Leibowitz's group at UNSW, Australia.

REFERENCES

[1] R Iris Bahar, Erica A Frohm, Charles M Gaona, Gary D Hachtel, Enrico Macii, Abelardo Pardo, and Fabio Somenzi. 1997. Algebric decision diagrams and their applications. *Formal methods in system design* 10, 2-3 (1997), 171–206.

[2] Thomas E. Carroll and Daniel Grosu. 2011. A game theoretic investigation of deception in network security. *Security and Communication Networks* 4, 10 (2011), 1162–1172. https://doi.org/10.1002/sec.242

[3] Daniel Dennett. 1986. Intentional systems. Brainstorms.

[4] Prashant Doshi. 2012. Decision Making in Complex Multiagent Settings: A Tale of Two Frameworks. *AI Magazine* 33, 4 (2012), 82–95.

[5] Prashant Doshi and Dennis Perez. 2008. Generalized Point Based Value Iteration for Interactive POMDPs.. In *AAAI*. AAAI Press, 63–68.

[6] Karel Durkota, Viliam Lisỳ, Branislav Bošanskỳ, and Christopher Kiekintveld. 2015. Approximate solutions for attack graph games with imperfect information. In *International Conference on Decision and Game Theory for Security*. Springer, Springer, 228–249.

[7] Zhengzhu Feng and Eric A Hansen. 2014. Approximate planning for factored POMDPs. In *Sixth European Conference on Planning*. AAAI, 99–106.

[8] Kimberly Ferguson-Walter, Sunny Fugate, Justin Mauger, and Maxine Major. 2019. Game theory for adaptive defensive cyber deception. In *Proceedings of the 6th Annual Symposium on Hot Topics in the Science of Security*. ACM, Association for Computing Machinery, New York, NY, USA, 4.

[9] Christina Fowler, Mike Goffin, Bill Hill, Richard Lamourine, and Andrew Sovern. 2020. *An Introduction to MITRE Shield*. Technical Report. MITRE Corp.

[10] Piotr J Gmytrasiewicz and Prashant Doshi. 2005. A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research* 24 (2005), 49–79.

[11] Karel Horák, Quanyan Zhu, and Branislav Bošanskỳ. 2017. Manipulating adversary's belief: A dynamic game approach to deception by design for proactive network security. In *International Conference on Decision and Game Theory for Security*. Springer, Springer, 273–294.

[12] Sushil Jajodia, Noseong Park, Fabio Pierazzi, Andrea Pugliese, Edoardo Serra, Gerardo I Simari, and VS Subrahmanian. 2017. A probabilistic logic of cyber deception. *IEEE Transactions on Information Forensics and Security* 12, 11 (2017), 2532–2544.

[13] Sushil Jajodia, VS Subrahmanian, Vipin Swarup, and Cliff Wang. 2016. *Cyber deception*. Springer.

[14] AH Jesse Hoey, Robert St Aubin, and Craig Boutilier. 1999. SPUDD: stochastic planning using decision diagrams. In *Proceedings of Uncertainty in Artificial Intelligence (UAI). Stockholm, Sweden*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 279–288.

[15] Nicholas S Kovach, Alan S Gibson, and Gary B Lamont. 2015. Hypergame theory: a model for conflict, misperception, and deception. *Game Theory* 2015 (2015), e570639.

[16] David Maynor. 2011. *Metasploit toolkit for penetration testing, exploit development, and vulnerability research*. Elsevier.

[17] Pingree. 2018. Emerging Technology Analysis : Deception Techniques and Technologies Create Security Technology Business Opportunities. *Trapx security* (2018), 1–18.

[18] Pascal Poupart. 2005. *Exploiting structure to efficiently solve large scale partially observable Markov decision processes*. Ph.D. Dissertation. University of Toronto.

[19] Aaron Schlenker, Omkar Thakoor, Haifeng Xu, Long Tran-Thanh, Fei Fang, Phebe Vayanos, Milind Tambe, and Yevgeniy Vorobeychik. 2018. Deceiving cyber adversaries: A game theoretic approach. *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS* 2 (2018), 892–900.

[20] Omid Setayeshfar, Christian Adkins, Matthew Jones, Kyu Hyung Lee, and Prashant Doshi. 2019. GrAALF: Supporting Graphical Analysis of Audit Logs for Forensics. *arXiv e-prints*, Article arXiv:1909.00902 (Sept. 2019), 11 pages. arXiv:1909.00902 [cs.CR]

[21] M. Simaan and Jr. J.B. Cruz. 1973. On the Stackelberg Strategy in Nonzero-Sum Games. *Journal of Optimization Theory and Applications* 11, 5 (1973), 533–555.

[22] Lance Spitzner. 2003. The honeynet project: Trapping the hackers. *IEEE Security & Privacy* 1, 2 (2003), 15–23.

[23] Blake E Strom, Andy Applebaum, Doug P Miller, Kathryn C Nickels, Adam G Pennington, and Cody B Thomas. 2018. *Mitre Att&ck: Design and Philosophy*. Technical Report. MITRE Corp.

[24] Jim Yuill, Mike Zappe, Dorothy Denning, and Fred Feer. 2004. Honeyfiles:Deceptive Files for Intrusion Deception. In *Fifth Annual IEEE SMC Workshop on Information Assurance*. 116–122.