Appendix to Paper #335

ACM Reference Format:
. 2021. Appendix to Paper #335. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 5 pages.

APPENDIX

1 ATTACKER POLICIES

We model the attackers using optimal policies of their level-0 POMDPs. For our problem, we define three distinct types of attackers which are modeled as separate frames in the I-POMDP. Below we discuss the optimal policies for each type.

1.1 The *data exfil* attacker frame

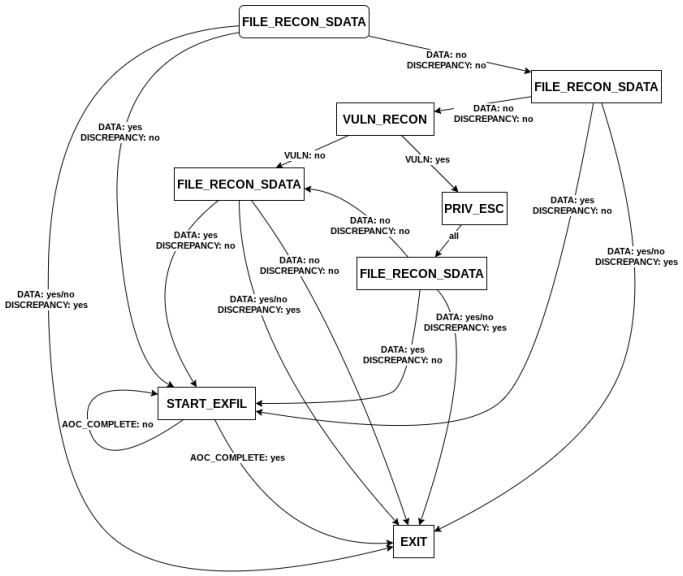

Figure 1: Optimal policy for *data exfil* type attacker

The *data exfil* type attacker is rewarded for stealing `sensitive_data` on the host. We model this type based on threats that steal private data and other sensitive data from systems, as shown in Fig. 1. The attacker starts with no knowledge of the existence of data on the system. We see that the optimal policy recommends the `FILE_RECON_SDATA` action which simulates sensitive data discovery on computers. On failing to find data after the first few attempts, the attacker attempts to escalate privileges and search again. If the attacker encounters unexpected types of decoys, the attacker leaves since there is no reward for stealing data that is not sensitive. Also, the observation of discrepancies when data is found will inform the attacker about the possibility of deception. This is because the system only contains a single type of data. On being alerted to the possibility of being deceived, the attacker leaves the system.

1.2 The *data manipulator* attacker frame

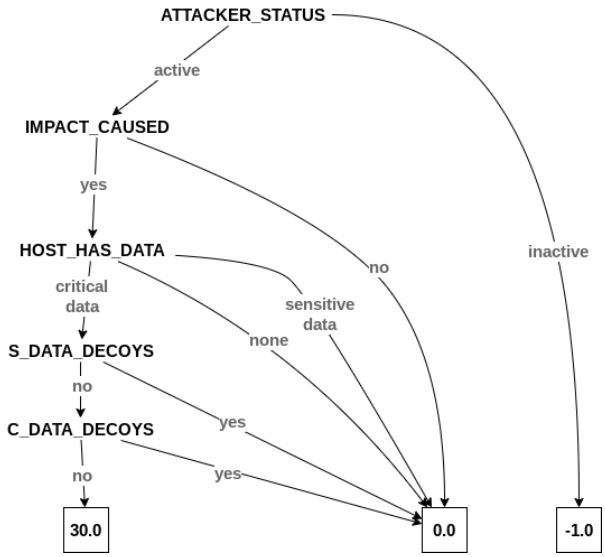

Figure 2: A reward ADD for $\mathcal{R}^{\texttt{EXIT}}(\mathcal{X})$ for the *data manipulator* attacker. The attacker is rewarded for performing the `EXIT` action if the `IMPACT_CAUSED` state is `yes` and `critical_data` is found.

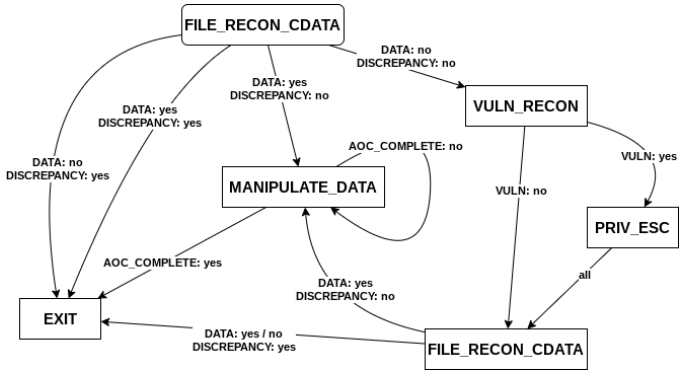

Figure 3: Optimal policy for *data manipulator* type attacker

The *data manipulator* type attacker is rewarded for manipulating `critical_data` on the host. An example ADD constituting its reward function is shown in Fig. 2. This attacker type is modeled after attackers that intrude systems to manipulate data that is critical for a business operation. Similar to the *data exfil* type, the attacker

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.

starts with no information about the data, as shown in Fig. 3. The optimal policy for this attacker type recommends FILE_RECON_CDATA action in the initial steps. Because critical data like service configurations or databases are usually stored in well-known locations, the FILE_RECON_CDATA is modeled to find critical_data quickly as compared to sensitive data. In the subsequent interaction steps, the attacker escalates privileges to continue the search if data is not found in the initial steps. Like the *data exfil* attacker, the *data manipulator* also leaves the system on observing discrepancies, suspecting deception, or on failure to find data.

1.3 The *persistent threat* attacker frame

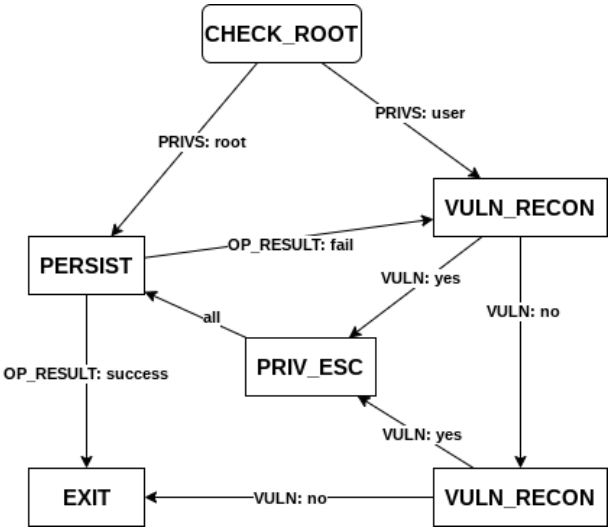

Figure 4: Optimal policy for *persistent threat* type attacker

The *persistent threat* type attacker aims to establish root level persistence on the host. Such attacks are common. Attackers establish a strong presence in an organization's network and stay dormant for an extended duration. For this attacker type, the policy consists of vulnerability discovery actions in the initial steps, as shown in Fig. 4. The attacker escalates privileges by performing the PRIV_ESC action on finding vulnerabilities. Once the attacker has the required privileges, the PERSIST action is performed to complete the objective.

While all three attacker policies may seem significantly different from their actions, the defender's observations of these actions are noisy. The errors in observation come from the noisy nature of real-time log analysis. For example, the VULN_RECON action models vulnerability discovery on a host. This action involves looking through the local file system for any vulnerable scripts, enumerating system information, listing services, among others. Thus a VULN_RECON can be mistaken for a FILE_RECON_CDATA or a FILE_RECON_SDATA in real-time log analysis. Similarly, it is difficult to tell the difference between the FILE_RECON_CDATA and FILE_RECON_SDATA from logs alone. Hence, without baiting the attacker into performing further actions, it is challenging to infer the intent of the attacker from the first few actions.

2 EVALUATION AGAINST HUMAN ATTACKERS

We also evaluate our I-POMDP$_\chi$ based defender against human attackers. For this study, we recruit graduate students enrolled in the cybersecurity course in our university to act as attackers. We assign a randomly sampled attacker frame to each subject. Based on the frame, we give the subjects an attack objective – to steal a particular file for the *data exfil* frame, to manipulate a configuration file for the *data manipulator* frame, and to establish persistence for the *persistence* frame.

The subjects start with user-level privileges. Every subject performs two engagements – one against the I-POMDP$_\chi$-based defender, and another against the NO-OP (no decoys) baseline. The subjects are unaware of the existence of the defender agent during both interactions. To maintain synchronization between the attacker and defender actions, we prevent the attacker from taking action before the defender agent has executed its action for a particular turn. We log the beliefs of the I-POMDP$_\chi$-based defender over the attacker's frame throughout the interaction.

Figure 5 shows an interaction trace between the I-POMDP$_\chi$-based agent and a human subject acting as a *data manipulator* type attacker. The boxes on the far left show the defender's beliefs over the attacker's frame. The defender starts with a uniform belief over all the frames. As the interaction progresses, the defender enables the attacker to escalate privileges. Consequently, the attacker exploits a vulnerability in the system and gets root privileges. Once the attacker has root privileges, the defender baits the attacker by deploying a critical_data decoy file. The attacker encounters the decoy file and manipulates it to complete his objective. This action triggers a C_DATA_DECOY_INTERACTION observation that assigns a high probability to the attacker type being *data manipulator*. Thus, the defender can recognize the attacker type by enabling the attacker's actions and then deploying appropriate decoys to determine intent.

Figure 6 shows an interaction trace between the I-POMDP$_\chi$-based agent and a human subject acting as a *data exfil* type attacker. In this particular scenario, the defender deploys a vulnerability for the attacker to exploit. However, the attacker does not attempt privilege escalation. Instead, the attacker performs file discovery actions repeatedly to find the sensitive_data file. The defender associates these actions with the *data exfil* and the *data manipulator* types, and consequently assigns a low probability to the *persistence* type. Since the defender is unable to differentiate between the *data exfil* type and the *data manipulator* type from the previous observations alone, the defender deploys a critical_data decoy file. The attacker does not encounter this decoy since he is looking for the sensitive_data file in user directories. Because the attacker does not interact with the deployed critical_data decoy, the defender attributes a high probability to the *data exfil* attacker type. Thus, the defender uses the attacker's models and his preference to not interact with the critical_data decoy to form an accurate belief over the attacker's type.

Figure 7 shows an interaction trace between the I-POMDP$_\chi$-based agent and a human subject acting as a *persistence* type attacker. Similar to the previous scenarios, the defender enables privilege escalation by deploying a vulnerability. However, in this case,

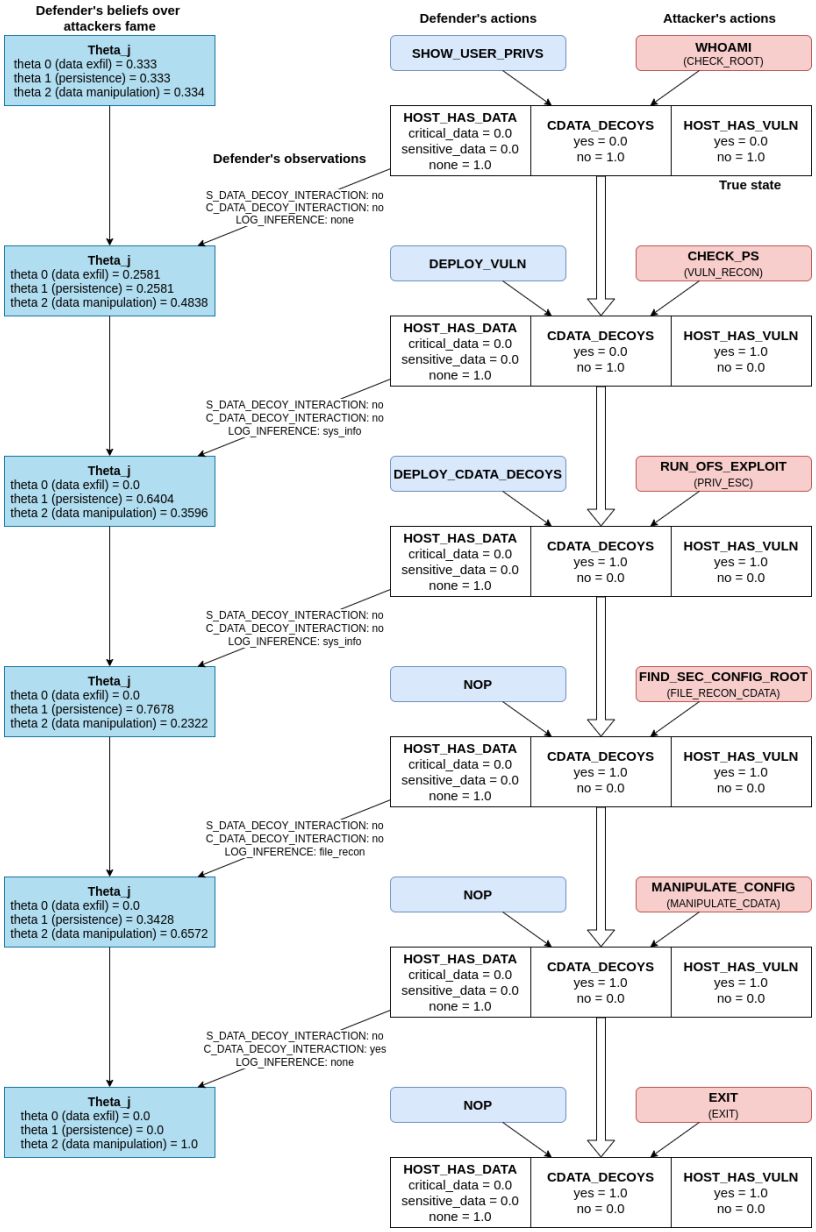

Figure 5: Interaction trace for I-POMDP$_\chi$-based defender against a human subject acting as *data manipulator* type attacker

the attacker initially performs file discovery actions. In the defender's models of the attacker, only the *data exfil* and the *data manipulator* type attackers perform these actions repeatedly. Hence the defender assigns higher probabilities to these frames. In the later stages of the attack, the attacker performs actions that are more relevant to the *persistence* frame. However, in the defender's model of the *persistence* type attacker, these actions are performed in the initial stages of the attack. Hence, during the later stages, the defender attributes the observations from these actions to noise. An attacker executing an optimal policy is likely to perform these actions in the initial stages of the attack. In realistic scenarios, it is highly unlikely that a *persistence* type attacker will perform file discovery actions repeatedly. In this particular case, the subject performed actions that were relevant to other frames and not optimal for achieving the given objective. Consequently, the defender formed incorrect beliefs over the attacker's frame.

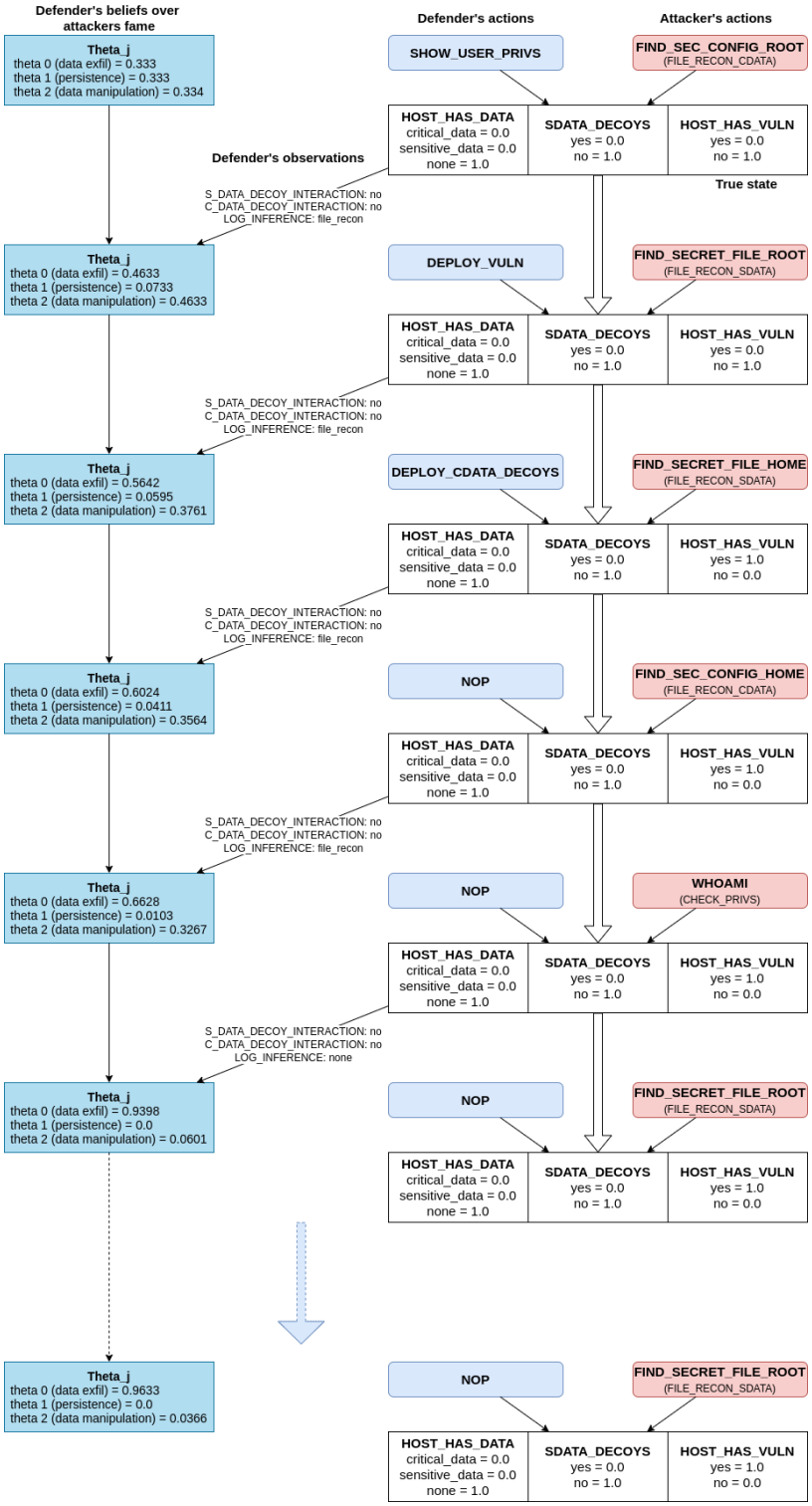

Figure 6: Interaction trace for I-POMDP$_\chi$-based defender against a human subject acting as *data exfil* type attacker

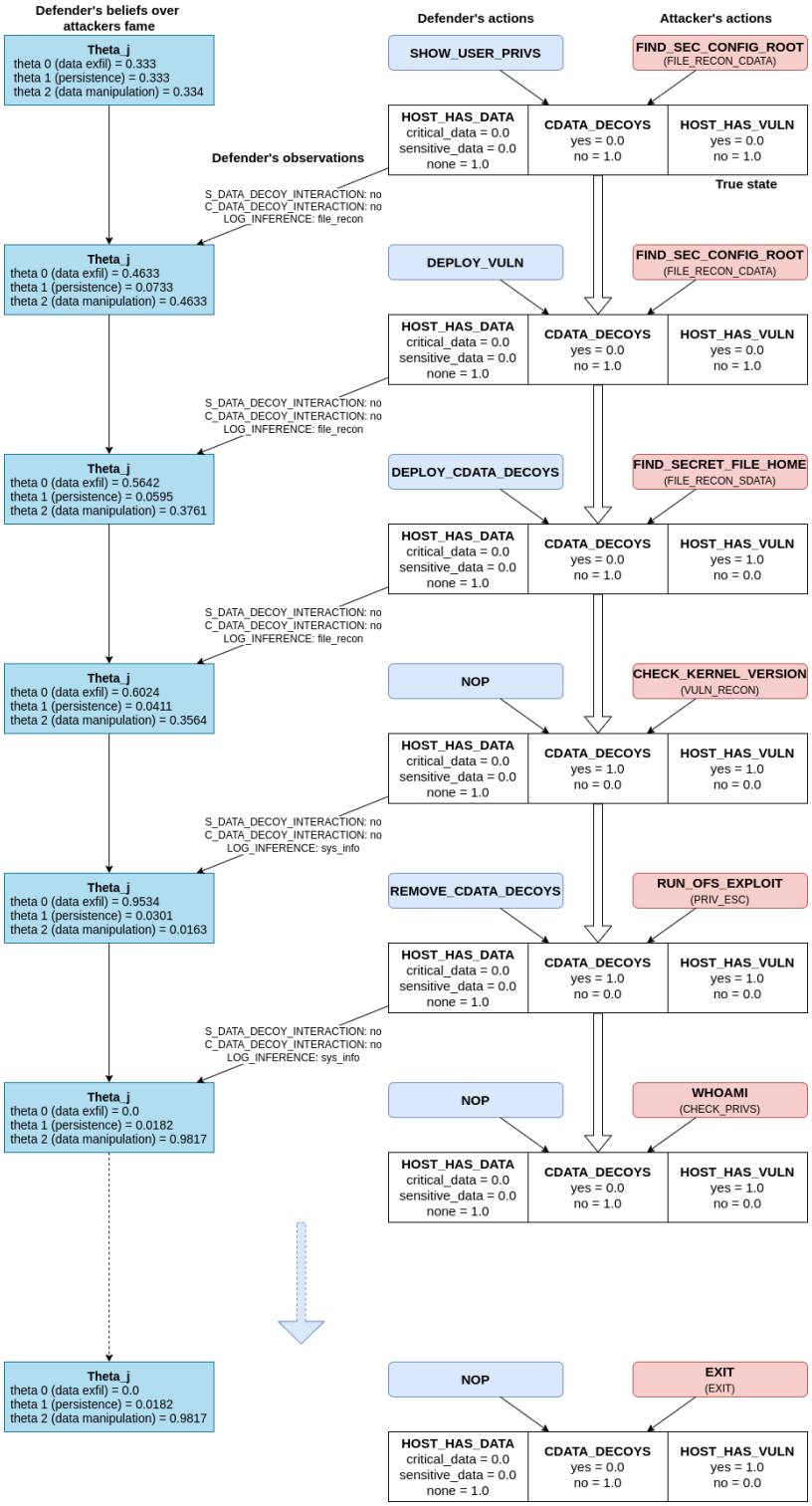

Figure 7: Interaction trace for I-POMDP$_\chi$-based defender against a human subject acting as *persistence* type attacker