ICC 2022 - IEEE International Conference on Communications | 978-1-5386-8347-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICC45855.2022.9838858

Fast Detection of Advanced Persistent Threats for Smart Grids: A Deep Reinforcement Learning Approach

Shi Yu
Dept. of Information & Communication Engineering, Xiamen University. Email: yushi@stu.xmu.edu.cn

***Abstract*—Data management systems in smart grids have to address advanced persistent threats (APTs), where malware injection methods are performed by the attacker to launch stealthy attacks and thus steal more data for illegal advantages. In this paper, we present a hierarchical deep reinforcement learning based APT detection scheme for smart grids, which enables the control center of the data management system to choose the APT detection policy to reduce the detection delay and improve the data protection level without knowing the attack model. Based on the state that consists of the size of the gathered power usage data, the priority level of the data, and the detection history, this scheme develops a two-level hierarchical structure to compress the high-dimensional action space and designs four deep dueling networks to accelerate the optimization speed with less over-estimation. Detection performance bound is provided and simulation results show that the proposed scheme improves both the data protection level and the utility of the control center with less detection delay.**

***Index Terms*—Smart grids, advanced persistent threat, reinforcement learning**

I. Introduction

Data management systems in smart grids gather power usage data from data concentrators (DCs) to monitor the system state and deliver the maintenance and control commands [1] against advanced persistent threats (APTs). APT attackers perform the malware injection methods, such as Stuxnet [2], to compromise the target component such as the DCs, with the aim of stealing the power usage data from smart grids without being caught [3]. For example, Black Energy in year 2016 stole the login information from the power system and used it to modify the control center commands that resulted in 700,000 users' power outages.

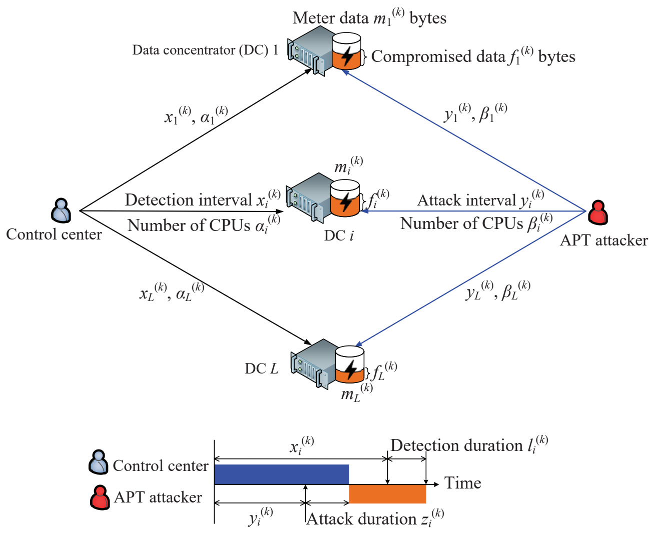
The detection interval and the central processing units (CPUs) allocation affect the APT detection performance [2], [4], [5]. For example, the two-player game-based APT detection scheme RSP in [4] chooses the detection interval to reach Nash equilibrium of the game with generating the attack-detection interval pairs randomly and thus increase the network controlling time. However, RSP suffers from a low data protection level against the smart APT attacker, because the attack intervals are rarely known by practical defenders.

This work is supported by the National Natural Science Foundation of China under Grant No. U21A20444 and 61971366.

The reinforcement learning (RL) based APT detection schemes in [6]–[8] determine the detection interval without relying on the attack interval information. For example, hotbooting policy hill climbing-based APT detection scheme HP as presented in [8] that chooses the detection interval to rapidly detect APTs with the sufficient number of CPUs improves the data protection level, but suffers a high detection delay if the APT attacker launches heavy attacks with more CPUs.

In this paper, we propose a hierarchical deep RL-based APT detection (HDRAD) for the control center of the data management system in smart grids to determine the detection policy consisting of the number of the allocated CPUs, the detection interval assignment, and the CPU allocation. Specifically, this scheme designs a hierarchical structure consisting of the top-level network and the low-level network to compress the high-dimensional action space, reduce the optimization complexity, and thus improve the detection performance. Instead of using deep Q-networks with the convolutional layers, the top-level network applies two dueling networks with one input layer and three fully-connected (FC) layers to output the Q-values of the number of allocated CPUs with the current state. With the chosen number of allocated CPUs and the state as the input, the low-level network including two dueling networks with the same structure as the top-level network outputs the Q-values of the detection interval assignment and the CPU allocation.

We investigate the Nash equilibrium of the APT detection game and the existence condition to provide the detection performance bound. Simulation results based on a three-DC system show that HDRAD improves the detection performance compared with the benchmarks RSP in [4] and HP in [8].



Meter data $m_1^{(k)}$ bytes
Data concentrator (DC) 1
Compromised data $f_1^{(k)}$ bytes
$x_1^{(k)}, \alpha_1^{(k)}$
$y_1^{(k)}, \beta_1^{(k)}$
$m_i^{(k)}$
Detection interval $x_i^{(k)}$
Number of CPUs $\alpha_i^{(k)}$
$f_i^{(k)}$
Attack interval $y_i^{(k)}$
Number of CPUs $\beta_i^{(k)}$
Control center
DC i
APT attacker
$x_L^{(k)}, \alpha_L^{(k)}$
$y_L^{(k)}, \beta_L^{(k)}$
DC L
$f_L^{(k)}$
$m_L^{(k)}$
$x_i^{(k)}$
Detection duration $l_i^{(k)}$
Control center
Time
APT attacker
$y_i^{(k)}$
Attack duration $z_i^{(k)}$

Fig. 1. The control center scans the data and detects DC i after $x_i^{(k)}$ time interval with $\alpha_i^{(k)}$ out of $b^{(k)}$ CPUs, while the APT attacker uses $\beta_i^{(k)}$ CPUs to compromise the data at DC i every $y_i^{(k)} \in [0, T]$ time interval.

II. System Model

In a data management system in smart grids as shown in Fig. 1, a control center manages and coordinates detection tasks assigned to L DCs against APT attacks. At time slot k, DC i with a distributed relational database is assumed to gather and store $m_i^{(k)}$ bytes power usage data from multiple smart meters for calculating the electricity consumption statistics and monthly billing information. The priority level of the data stored at DC i is denoted by $p_i^{(k)} \in \{1, 2, ..., P\}$, where P indicates the highest priority. For example, the power usage data from governments are more critical than the consumers with a higher $p_i^{(k)}$.

The control center applies string matching in [9] to scan the data and detects DC i after $x_i^{(k)} \in \{nT/N | 0 \leq n \leq N\}$ time interval since the previous detection was finished with $\alpha_i^{(k)} \in [K, Z]$ out of $b^{(k)}$ CPUs, where T is the maximum interval, K is the minimal number of the CPUs to detect the compromised data in time according to [10] and Z is the maximal available number of the CPUs. The control center analyzes the system log [11] and matches the data strings to complete the APT detection within the duration denoted by $l_i^{(k)}$. The control center recovers the $f_i^{(k)}$-byte compromised data with the largest normalized residual method in [12] and restores DC i.

The APT attacker uses $\beta_i^{(k)} \in [M, \mathcal{Z}]$ CPUs to perform malware injection, e.g., Stuxnet [13], every $y_i^{(k)} \in [0, T]$ time interval to compromise the data at DC i based on the greedy algorithm in [3] with the goal of maximizing the amount of the stolen data in each time slot. The attacker knows the network topology and the amount of the data of each DC.

III. Hierarchical Deep RL based APT Detection

We present a hierarchical deep RL-based APT detection (HDRAD) scheme to decrease the detection delay and improve the data protection level and the response speed for the smart grids with a large number of DCs. The two-level hierarchical structure consists of four dueling networks to choose the detection policy. Specifically, the top-level evaluated dueling network (EDN) chooses the number of the allocated CPUs based on the state, while the low-level EDN chooses the detection interval assignment and the CPU allocation to L DCs based on the top-level EDN output and the state. In addition, the two target dueling networks (TDNs) are designed to update the weights of the two EDNs. Unlike the hierarchical DQN in [14], the dueling network architecture replaces the convolutional layers with FC layers as shown in Fig. 2 to extract the detection features and thus accelerate the policy learning speed.

At time slot k, the state $\boldsymbol{s}^{(k)}$ consists of the size of the gathered power usage data $m_i^{(k)}$, the priority level of the data stored at DC i $p_i^{(k)}$, the size of the previous compromised data $f_i^{(k-1)}$ with $1 \leq i \leq L$, and the previous detection delay $D^{(k-1)}$, i.e.,

$$\boldsymbol{s}^{(k)} = \left[\left[m_i^{(k)}, p_i^{(k)}, f_i^{(k-1)} \right]_{1 \leq i \leq L}, D^{(k-1)} \right]. \quad (1)$$

The state is input to the top-level EDN that consists of an input layer (with $3L+1$ units), three FC layers and an aggregating module. The first FC layer in the top-level EDN has g_1 units that uses the matrix multiplication to extract the state features. The second FC layer with $2g_2$ units branches off into a value stream with weights $\boldsymbol{\omega}_1$ and an advantage stream with weights $\boldsymbol{\omega}_2$, both of which have g_2 units. The leaky rectified linear unit is chosen as the activation function. Similarly, the final FC layer with $Z-K+2$ units outputs the state value $\boldsymbol{V}(\boldsymbol{s}^{(k)}; \boldsymbol{\omega}_1)$ in the value stream, and the advantage stream outputs the estimated advantages $\boldsymbol{A}(\boldsymbol{s}^{(k)}, \cdot; \boldsymbol{\omega}_2)$ of the $Z-K+1$ feasible number of the allocated CPUs. The aggregating module estimates the Q-values given by

$$\begin{aligned} \boldsymbol{Q}_1\left(\boldsymbol{s}^{(k)}, \cdot; \boldsymbol{\omega}\right) &= \boldsymbol{V}\left(\boldsymbol{s}^{(k)}; \boldsymbol{\omega}_1\right) + \boldsymbol{A}\left(\boldsymbol{s}^{(k)}, \cdot; \boldsymbol{\omega}_2\right) \\ &- \frac{1}{Z-K+1} \sum_{b' \in [K,Z]} \boldsymbol{A}\left(\boldsymbol{s}^{(k)}, b'; \boldsymbol{\omega}_2\right), \end{aligned} \quad (2)$$

which is the basis to choose the number of allocated CPUs b via the ϵ-greedy algorithm to balance the exploration and exploitation.

The low-level EDN with weights $\boldsymbol{\sigma}$ inputs both b and $\boldsymbol{s}^{(k)}$, and has the similar structure with the top-level EDN. The three FC layers have d_1, $2d_2$ and $N^L Z^L + 1$ units, respectively. Based on the value stream with weights $\boldsymbol{\sigma}_1$ and the advantage stream with weights $\boldsymbol{\sigma}_2$, the low-level EDN

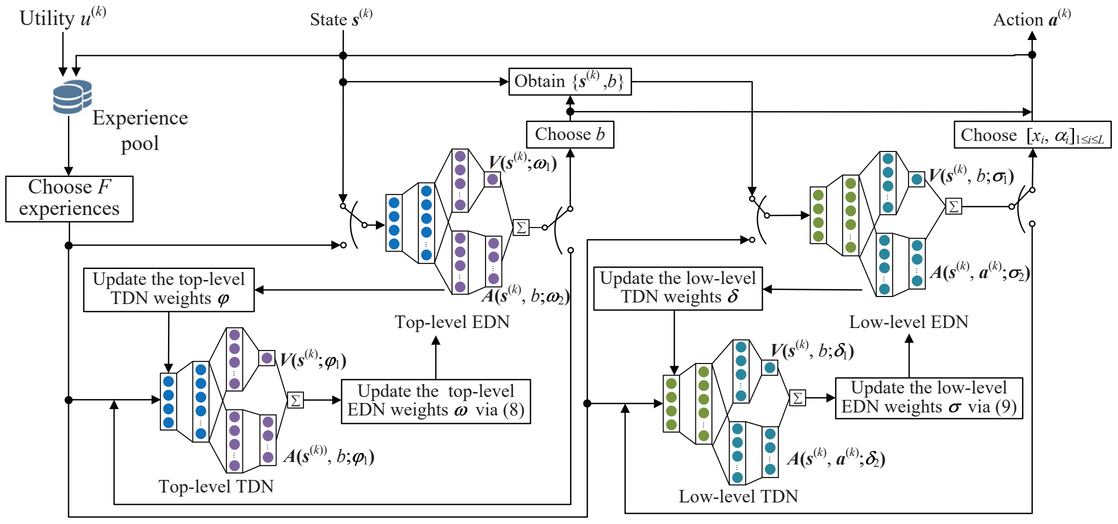

Utility $u^{(k)}$
State $s^{(k)}$
Action $a^{(k)}$
Experience pool
Obtain $\{s^{(k)},b\}$
Choose b
Choose $[x_i, \alpha_i]_{1\le i\le L}$
Choose F experiences
$V(s^{(k)};\omega_1)$
$A(s^{(k)}, b;\omega_2)$
Top-level EDN
Update the top-level TDN weights φ
$V(s^{(k)};\varphi_1)$
Update the top-level EDN weights ω via (8)
$A(s^{(k)}), b;\varphi_1)$
Top-level TDN
$V(s^{(k)}, b;\sigma_1)$
$A(s^{(k)}, a^{(k)};\sigma_2)$
Low-level EDN
Update the low-level TDN weights δ
$V(s^{(k)}, b;\delta_1)$
Update the low-level EDN weights σ via (9)
$A(s^{(k)}, a^{(k)};\delta_2)$
Low-level TDN

Fig. 2. Illustration of the hierarchical deep RL-based APT detection scheme.

outputs the estimated Q-values as

$$\boldsymbol{Q}_2(\boldsymbol{s}^{(k)},\cdot;\boldsymbol{\sigma}) = \boldsymbol{V}(\boldsymbol{s}^{(k)},b;\boldsymbol{\sigma}_1) + \boldsymbol{A}(\boldsymbol{s}^{(k)},b,\cdot;\boldsymbol{\sigma}_2) - \frac{1}{|\boldsymbol{\Delta}|}\sum_{\boldsymbol{\chi}'\in[K,Z]} \boldsymbol{A}\left(\boldsymbol{s}^{(k)},b,\boldsymbol{\chi}';\boldsymbol{\omega}_2\right), \tag{3}$$

which is the basis to choose the detection intervals and the allocated CPUs allocated to L DCs, i.e., $\boldsymbol{\chi} = [x_i, \alpha_i]_{1\le i\le L}$ via the ϵ-greedy algorithm from the feasible set $\boldsymbol{\Delta}$ given by

$$\boldsymbol{\Delta} = \left\{ [x_i,\alpha_i]_{1\le i\le L} \middle| 0\le x_i\le T; K\le \alpha_i\le Z; \sum_{i=1}^{L}\alpha_i \le b \right\}. \tag{4}$$

The detection policy $\boldsymbol{a}^{(k)}$ consists of the number of allocated CPUs b, the detection interval assignment x_i, and the CPU allocation α_i with $1\le i\le L$, i.e.,

$$\boldsymbol{a}^{(k)} = \left[b, [x_i,\alpha_i]_{1\le i\le L}\right] \in \boldsymbol{\Lambda}, \tag{5}$$

where $\boldsymbol{\Lambda}$ is the action set that contains $(N+1)^L(Z-K+1)^{L+1}$ feasible policies. The control center detects the L DCs according to the chosen policy $\boldsymbol{a}^{(k)}$.

The utility $u^{(k)}$ is calculated as

$$u^{(k)} = \sum_{i=1}^{L}\left(\mu e^{-f_i} - x_i - l_i\right) + c_1\sum_{i=1}^{L} x_i - c_2\sum_{i=1}^{L} f_i p_i - c_3 b, \tag{6}$$

where c_1, c_2 and c_3 represent the importance of the saving energy consumption of scanning, the compromised data, and the CPU scheduling overhead in the APT detection process.

The detection experiences $\{\boldsymbol{s}^{(k)},\boldsymbol{a}^{(k)},u^{(k)}\}$ are stored in the experience pool $\mathcal{D}$, from which F experiences are randomly chosen to build a minibatch $[e^{(\mathbf{n}_j)}]_{1\le j\le F}$, where $\mathbf{n}_j \sim \mathrm{U}(1,k)$. As shown in Algorithm 1, the minibatch and two additional TDNs are used to update the weights of the two EDNs: The weights of the top-level EDN denoted by $\boldsymbol{\omega}$ are updated via stochastic gradient descent (SGD) as

$$\boldsymbol{\omega} \leftarrow \arg\min_{\boldsymbol{\omega}'} \frac{1}{F}\sum_{j=1}^{F}\Bigg(u^{(\mathbf{n}_j)} - \boldsymbol{Q}_1\left(\boldsymbol{s}^{(\mathbf{n}_j)}, b^{(\mathbf{n}_j)};\boldsymbol{\omega}'\right) + \gamma\boldsymbol{Q}_1\left(\boldsymbol{s}^{(\mathbf{n}_j+1)}, \arg\max_{b'\in[K,Z]}\boldsymbol{Q}_1\left(\boldsymbol{s}^{(\mathbf{n}_j+1)}, b';\boldsymbol{\omega}'\right);\boldsymbol{\varphi}\right)\Bigg)^2, \tag{7}$$

where $\gamma\in[0,1]$ is a discount factor and the weights of the low-level EDN denoted by $\boldsymbol{\sigma}$ are updated via SGD as (8).

The top-level TDN with weights $\boldsymbol{\varphi}$ estimates the target Q-value of the number of allocated CPUs b that maximizes the estimated Q-value. The top-level TDN copies the the top-level EDN weights $\boldsymbol{\omega}$ every C steps to update its weights $\boldsymbol{\varphi}$ and keeps fixed on other steps. Similarly, the low-level TDN with weights $\boldsymbol{\delta}$ copies the the low-level EDN weights $\boldsymbol{\sigma}$ every C steps.

IV. Performance Evaluation

In the APT detection game, the control center chooses the number of CPUs $b\in[K,Z]$, the detection interval $x_i\in[0,T]$ and the CPU allocation $\alpha_i\in[K,Z]$ with $1\le i\le L$ to maximize the utility given (6), while the APT attacker determines the number of CPUs $\beta_i\in[M,\mathcal{Z}]$ and the attack interval $y_i\in[0,T]$to maximize its utility u_{A} given by

$$u_{\mathrm{A}} = -u + c_4\sum_{i=1}^{L} y_i - c_5\sum_{i=1}^{L}\beta_i. \tag{9}$$

$$\boldsymbol{\sigma} \leftarrow \arg\min_{\boldsymbol{\sigma}'} \frac{1}{F} \sum_{j=1}^{F} \left(u^{(\mathbf{n}_j)} - \boldsymbol{Q}_2 \left(\boldsymbol{s}^{(\mathbf{n}_j)}, \boldsymbol{a}^{(\mathbf{n}_j)}; \boldsymbol{\sigma}' \right) + \gamma \boldsymbol{Q}_2 \left(\boldsymbol{s}^{(\mathbf{n}_j+1)}, \arg\max_{\boldsymbol{\chi} \in \boldsymbol{\Delta}} \boldsymbol{Q}_2 \left(\boldsymbol{s}^{(\mathbf{n}_j+1)}, b^{(\mathbf{n}_j+1)}, \boldsymbol{\chi}; \boldsymbol{\sigma}' \right); \boldsymbol{\delta} \right) \right)^2 \tag{8}$$

Algorithm 1 Hierarchical Deep RL based APT detection scheme (HDRAD)

```
1: Initialize $L$, $C$, $F$, $\mu$, $[f_i]_{1\le i\le L} = \mathbf{0}$, $\boldsymbol{\omega}$, $\boldsymbol{\sigma}$, $\boldsymbol{\varphi}$, $\boldsymbol{\delta}$ and $\mathcal{D} = \varnothing$
2: for $k = 1, 2, \cdots$ do
3:    Obtain $[m_i]_{1\le i\le L}$ from $L$ DCs
4:    Evaluate $[p_i]_{1\le i\le L}$ and $[l_i]_{1\le i\le L}$
5:    Obtain $\boldsymbol{s}^{(k)}$ via (1)
6:    Input $\boldsymbol{s}^{(k)}$ to the top-level EDN
7:    Obtain the top-level EDN outputs $\boldsymbol{Q}_1(\boldsymbol{s}^{(k)}, b; \boldsymbol{\omega})$
8:    Choose $b$ with $\epsilon$-greedy
9:    Input $\{\boldsymbol{s}^{(k)}, b\}$ to the low-level EDN
10:   Obtain the low-level EDN outputs $\boldsymbol{Q}_2(\boldsymbol{s}^{(k)}, \boldsymbol{a}^{(k)}; \boldsymbol{\sigma})$
11:   Choose $[x_i, \alpha_i]_{1\le i\le L}$ with $\epsilon$-greedy
12:   Scan the data in DC $i$ within $x_i$ time interval
13:   Detect DC $i$ with $\alpha_i$ out of $b$ CPUs
14:   Recover $f_i$-byte compromised data in DC $i$
15:   Calculate $u^{(k)}$ via (6)
16:   Store $\{\boldsymbol{s}^{(k)}, \boldsymbol{a}^{(k)}, u^{(k)}\}$ in the experience pool $\mathcal{D}$
17:   Sample $F$ experiences from $\mathcal{D}$
18:   for $j = 1, 2, \cdots, F$ do
19:      $e^{(\mathbf{n}_j)} = \mathcal{D}(\mathbf{n}_j)$
20:   end for
21:   Update $\boldsymbol{\omega}$ via (7)
22:   Update $\boldsymbol{\sigma}$ via (8)
23:   if $k \mod C = 0$ then
24:      Update $\boldsymbol{\varphi}$ with $\boldsymbol{\omega}$
25:      Update $\boldsymbol{\delta}$ with $\boldsymbol{\sigma}$
26:   end if
27: end for
```

Similar to [15], the detection delay D is modeled by

$$D = \sum_{i=1}^{L} \left(x_i + \frac{X}{\alpha_i} - y_i - \frac{Y}{\beta_i} \right), \tag{10}$$

where X and Y denote the total required CPU time for the serial processing to complete a detection and an APT attack, respectively. According to [16], the data protection level B is given by

$$B = \frac{1}{L} \sum_{i=1}^{L} \min \left\{ \frac{y_i + Y\beta_i^{-1}}{x_i + X\alpha_i^{-1}}, 1 \right\}. \tag{11}$$

The size of compromised data that depends on the data protection level is modeled as $f_i = Bm_i$ according to [17].

By (6), (10), and (11), the utility function is given by

$$\begin{aligned} u =& c_1 \sum_{i=1}^{L} x_i + \frac{c_2}{L} \sum_{i=1}^{L} m_i p_i \min \left\{ \frac{y_i + Y\beta_i^{-1}}{x_i + X\alpha_i^{-1}}, 1 \right\} \\ & - c_3 b - \sum_{i=1}^{L} \left(x_i + \frac{X}{\alpha_i} - y_i - \frac{Y}{\beta_i} \right). \end{aligned} \tag{12}$$

Theorem 1. *The performance bound of the proposed hierarchical deep RL-based APT detection algorithm with $L = 1$ is given by*

$$u \le c_1 T + c_2 m_1 p_1 + c_3 K - \frac{X}{K} \tag{13}$$

$$D \le \frac{X}{K} - \frac{Y}{M} \tag{14}$$

$$B \le 1 \tag{15}$$

if

$$Y \le \min \left\{ \mathcal{Z}T + \frac{\mathcal{Z}X}{K}, c_5 M^2 \right\} \tag{16}$$

$$\min \{c_1, c_4\} \ge 1 \tag{17}$$

$$X \le c_3 K^2 \tag{18}$$

Proof. For given $L = 1$, we have $b = \alpha_1$. By (12), if (16) holds, we have

$$\min \left\{ \frac{y_1 + Y\beta_1^{-1}}{x_1 + X\alpha_1^{-1}}, 1 \right\} = 1. \tag{19}$$

By (12) and (19), thus we have

$$\begin{aligned} u([x_1, \alpha_1], [y_1, \beta_1]) =& (c_1 - 1)x_1 + c_2 m_1 p_1 - c_3 \alpha_1 \\ & - \frac{X}{\alpha_1} + y_1 + \frac{Y}{\beta_1}. \end{aligned} \tag{20}$$

Let

$$q_1(x_1) = (c_1 - 1)x_1 + T + c_2 m_1 p_1, \tag{21}$$

and $\forall x_1 \in [0, T]$, if (17) holds, we have

$$\frac{dq_1(x_1)}{dx_1} = c_1 - 1 \ge 0. \tag{22}$$

Thus, $q_1(x_1)$ is maximized at $x_1^* = T$. Let

$$q_2(\alpha_1) = \frac{Y}{M} - c_3 \alpha_1 - \frac{X}{\alpha_1}, \tag{23}$$

and $\forall \alpha_1 \in [K, Z]$, if (18) holds, we have

$$q_2(K) = -c_3 K - \frac{X}{K} \geq -c_3\alpha_1 - \frac{X}{\alpha_1} = q_2(\alpha_1). \tag{24}$$

Thus, $q_2(\alpha_1)$ is maximized at $\alpha_1^* = K$. By (20), (21), and (23), we have

$$u\left([x_1, \alpha_1], [T, M]\right) = q_1(x_1) + q_2(\alpha_1). \tag{25}$$

As x_1 is independent with α_1, $\forall x_1 \in [0, T]$ and $\forall \alpha_1 \in [K, Z]$, we have

$$\begin{aligned} & u\left([T, K], [T, M]\right) \\ & = c_1 T - c_3 K - \frac{X}{K} + \frac{Y}{M} + c_2 m_1 p_1 && (26) \\ & \geq (c_1 - 1)x_1 - c_3\alpha_1 - \frac{Y}{\alpha_1} + T + \frac{X}{M} + c_2 m_1 p_1 && (27) \\ & = u\left([x_1, \alpha_1)], [T, M]\right). && (28) \end{aligned}$$

Let

$$q_3(y_1) = (c_4 - 1)y_1 + (1 - c_1)T - c_2 m_1 p_1, \tag{29}$$

and $\forall y_1 \in [0, T]$, if (17) holds, we have

$$\frac{dq_3(y_1)}{dy_1} = c_4 - 1 \geq 0. \tag{30}$$

Thus, $q_3(y_1)$ is maximized at $y_1^* = T$. Let

$$q_4(\beta_1) = c_3 K + \frac{X}{K} - c_5\beta_1 - \frac{Y}{\beta_1}. \tag{31}$$

and $\forall \beta_1 \in [M, \mathcal{Z}]$, if (16) holds, we have

$$q_4(M) = c_3 K + \frac{X}{K} - c_5 M - \frac{Y}{M} \geq c_3 K + \frac{X}{K} - c_5\beta_1 - \frac{Y}{\beta_1} = q_4(\beta_1). \tag{32}$$

Thus, $q_4(\beta_1)$ is maximized at $\beta_1^* = M$. By (9), (29) and (31), we have

$$u_{\text{A}}\left([T, K], [y_1, \beta_1]\right) = q_3(y_1) + q_4(\beta_1). \tag{33}$$

As y_1 is independent with β_1, $\forall y_1 \in [0, T]$ and $\forall \beta_1 \in [M, \mathcal{Z}]$, we have

$$\begin{aligned} & u_{\text{A}}\left([T, K], [T, M]\right) \\ & =(c_4 - c_1)T - c_5 M - \frac{Y}{M} + c_3 K + \frac{X}{K} - c_2 m_1 p_1 && (34) \\ & \geq (c_4 - 1)y_1 - c_5\beta_1 - \frac{Y}{\beta_1} + (1 - c_1)T \\ & \quad + c_3 K + \frac{X}{K} - c_2 m_1 p_1 && (35) \\ & = u_{\text{A}}\left([T, K], [y_1, \beta_1]\right). && (36) \end{aligned}$$

Thus, by (26), (28), (34), and (36), we have a Nash equilibrium given by $([T, K], [T, M])$ and the performance bound of the HDRAD is given by (13) - (15). □

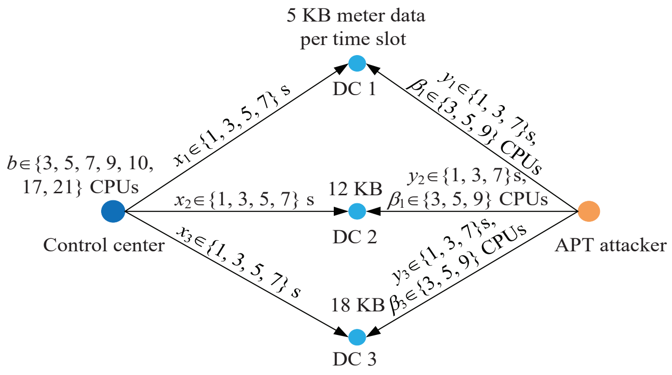

5 KB meter data
per time slot
DC 1
$x_1 \in \{1, 3, 5, 7\}$ s
$y_1 \in \{1, 3, 7\}$s,
$\beta_1 \in \{3, 5, 9\}$ CPUs
$b \in \{3, 5, 7, 9, 10,$
$17, 21\}$ CPUs
$x_2 \in \{1, 3, 5, 7\}$ s
12 KB
$y_2 \in \{1, 3, 7\}$s,
$\beta_1 \in \{3, 5, 9\}$ CPUs
Control center
DC 2
APT attacker
$x_3 \in \{1, 3, 5, 7\}$ s
$y_3 \in \{1, 3, 7\}$s,
$\beta_3 \in \{3, 5, 9\}$ CPUs
18 KB
DC 3

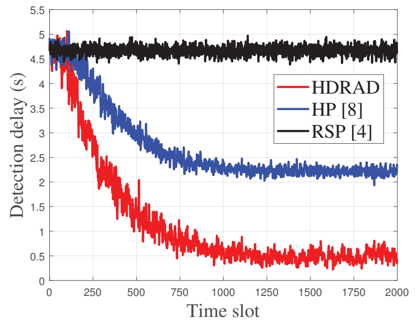
Fig. 3. Simulation topology of the data management system with three DCs in the smart grid.

Remark 1: The control center uses the longest APT detection interval under high energy consumption given by (17). As shown in (18), if the total required CPU time to complete a detection X is less than the bound that relies on the number of CPUs meeting the basic security requirement K, the control center completes the APT detection with the minimum number of CPUs.

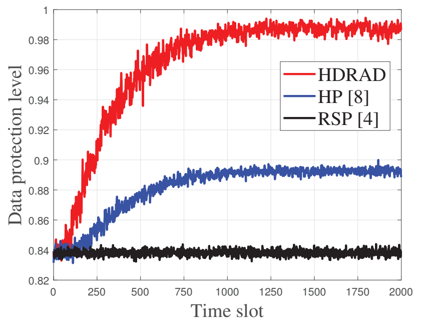
V. Simulation Results

Simulations were performed for a data management system as shown in Fig. 3 to evaluate the performance of the HDRAD scheme. Three DCs gather $\{5, 12, 18\}$ kilobytes (KB) power usage data from smart meters in each 10-second time slot [18]. The priority level of the data stored in the three DCs is chosen from $\{1, 2, 3, 4, 5, 6\}$. The control center chooses the detection interval from $\{1, 3, 5, 7\}$ seconds and allocates $\{3, 5, 7, 9, 10, 17, 21\}$ CPUs to detect APT attacks. The APT attacker applies the greedy algorithm in [3] to choose the attack interval from $\{1, 3, 7\}$ seconds and allocates $\{3, 5, 9\}$ CPUs in each DC.

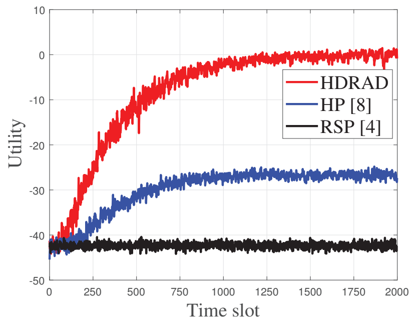
As illustrated in Fig. 4, HDRAD achieves the optimal detection policy, and reduces 15.4% detection delay from 4.7 s to 0.3 s, and improves 15.4% data protection level from 84.1% to 99.4%, and increases 48.3% utility from -41.2 to 0.9 after 1250 time slots. This scheme outperforms RSP in [4] by decreasing 94.1% detection delay and improving 18.9% data protection level and 49.7% utility of the control center. By designing the hierarchical structure to accelerate the exploration speed, HDRAD also exceeds HP in [8] by reducing 87.2% detection delay, improving 11.3% data protection level, and increasing 30.4% utility of the control center.



(a) Detection delay



(b) Data protection level



(c) Utility

Fig. 4. Performance of the APT detection schemes in the three-DC data management system as shown in Fig. 3.

VI. Conclusion

In this paper, we have proposed a hierarchical deep RL based APT detection scheme for the control center in smart grids to choose the number of the allocated CPUs, the detection interval assignment, and the CPU allocation. The proposed scheme designs a hierarchical structure to optimize the detection performance without requiring the knowledge of the attack model. In the simulation based on a three-DC system, the HDRAD outperforms both the benchmarks RSP in [4] and HP in [8]. For example, the HDRAD decreases 87.2% detection delay, improves 11.3% data protection level, and increases 30.4% utility of the control center after 1250 time slots compared with HP.

References

[1] H. Lin, A. Slagell, Z. T. Kalbarczyk, P. W. Sauer, and R. K. Iyer, "Runtime semantic security analysis to detect and mitigate control-related attacks in power grids," *IEEE Trans. Smart Grid*, vol. 9, no. 1, pp. 163–178, Jan. 2016.

[2] J. Tian, R. Tan, X. Guan, Z. Xu, and T. Liu, "Moving target defense approach to detecting Stuxnet-like attacks," *IEEE Trans. Smart Grid*, vol. 11, no. 1, pp. 291–300, Jun. 2019.

[3] M. Van Dijk, A. Juels, A. Oprea, and R. L. Rivest, "Flipit: The game of "stealthy takeover"," *J. Cryptol.*, vol. 26, no. 4, pp. 655–713, Oct. 2013.

[4] L. Yang, P. Li, Y. Zhang, X. Yang, Y. Xiang, and W. Zhou, "Effective repair strategy against advanced persistent threat: A differential game approach," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 7, pp. 1713–1728, Jul. 2019.

[5] Y. Chen, J. Hong, and C. Liu, "Modeling of intrusion and defense for assessment of cyber security at power substations," *IEEE Trans. Smart Grid*, vol. 9, no. 4, pp. 2541–2552, Sept. 2018.

[6] R. Zhang and Q. Zhu, "A game-theoretic cyber insurance framework for incentive-compatible cyber risk management of internet of things," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 2026–2041, Nov. 2019.

[7] L. Greige and P. Chin, "Reinforcement learning in FlipIt," *arXiv preprint arXiv:2002.12909*, 2020.

[8] L. Xiao, D. Xu, N. B. Mandayam, and H. V. Poor, "Attacker-centric view of a detection game against advanced persistent threats," *IEEE Trans. Mobile Comput.*, vol. 17, no. 11, pp. 2512–2523, Nov. 2018.

[9] T. Garfinkel, M. Rosenblum, *et al.*, "A virtual machine introspection based architecture for intrusion detection," in *Ndss*, vol. 3, pp. 191–206, Citeseer, 2003.

[10] S. Feng, Z. Xiong, D. Niyato, and P. Wang, "Dynamic resource management to defend against advanced persistent threats in fog computing: A game theoretic approach," *IEEE Trans. Cloud Comput.*, Jan. 2019.

[11] P. He, J. Zhu, S. He, J. Li, and M. R. Lyu, "Towards automated log parsing for large-scale log data analysis," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 6, pp. 931–944, Oct. 2017.

[12] Z. Qin, Q. Li, and G. Hsieh, "Defending against cooperative attacks in cooperative spectrum sensing," *IEEE Trans. Wireless Commun.*, vol. 12, no. 6, pp. 2680–2687, Jun. 2013.

[13] T. M. Chen, "Stuxnet, the real start of cyber warfare?[editor's note]," *IEEE Netw.*, vol. 24, no. 6, pp. 2–3, Nov. 2010.

[14] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Proc. Conf. Adv. Neural Inf. Process. Syst. (NIPS)*, pp. 3675–3683, Barcelona, Spain, Dec. 2016.

[15] A. Ghafouri, W. Abbas, A. Laszka, Y. Vorobeychik, and X. Koutsoukos, "Optimal thresholds for anomaly-based intrusion detection in dynamical environments," in *Proc. Int. Conf. Decision Game Theory Secur. (GameSec)*, pp. 415–434, New York, USA, Nov. 2016.

[16] M. Min, L. Xiao, C. Xie, M. Hajimirsadeghi, and N. B. Mandayam, "Defense against advanced persistent threats in dynamic cloud storage: A Colonel Blotto game approach," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4250–4261, Dec. 2018.

[17] P. Hu, H. Li, H. Fu, D. Cansever, and P. Mohapatra, "Dynamic defense strategy against advanced persistent threat with insiders," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, pp. 747–755, Hong Kong, China, Apr. 2015.

[18] J. Jiang and Y. Qian, "Distributed communication architecture for smart grid applications," *IEEE Commun. Mag.,*, vol. 54, no. 12, pp. 60–67, Dec. 2016.