RL-BAGS: A Tool for Smart Grid Risk Assessment

Yatin Wadhawan
Department of Computer Science
University of Southern California (USC)
Los Angeles, USA
ywadhawa@usc.edu

Dr. Clifford Neuman
Department of Computer Science
Information Science Institute, USC
Los Angeles, USA
bcn@isi.edu

***Abstract*—The security of critical infrastructure such as Smart Grid is of significant concern because cyber-physical attacks are becoming a frequent occurrence. Cybercriminals compromise cyberinfrastructure to control physical processes maliciously. It is the system administrator's goal to find vulnerabilities in the smart grid functions and patch them before they are compromised. Unfortunately, limited resources and a large attack surface make it difficult to decide which function to protect in a particular system state. In this research paper, we tackle the problem of resource allocation in the smart grid system by proposing a tool, Reinforcement Learning-Bayesian Attack Graph for Smart Grid System (RL-BAGS), which provides functionality to the system engineers to compute optimal policies on regular intervals about whether to SCAN or PATCH a particular function of the smart grid system. RL-BAGS considers functions and network architecture of the system to generate a Bayesian Network, which represents the state of the system. RL-BAGS implements two reinforcement learning algorithms, Q-Learning and SARSA learning, on the generated Bayesian Network to learn optimal policies. RL-BAGS assists system administrators performing in-depth studies of one of the functions of the smart grid system advising effective actions to scan or patch a system component.**

Keywords—Smart Grid; Cyber-Physical Systems; Reinforcement Learning; Cyber Security; Risk Assessment; Power Systems; Vulnerability Assessment

Nomenclature

S1: Quality of smart meter and electric vehicle reads
S2: Quality of smart sync head
S3: Billing system performance
S4: Performance of outage management system
S5: Quality of data captured by vendor
S6: Meter data management
S7: Performance of electricity energy control center

I. Introduction

Cyber-Physical Systems (CPS) are a new generation of systems where physical processes are controlled and monitored from the cyber domain through advanced information and communication technologies with humans in the loop [1]. Deep integration between the cyber and physical systems has imposed extraordinary challenges on the security of critical infrastructures such as Smart Grid Systems (SGS). While adding to and improving the functionalities of SGS such as Demand Response, Advance Metering Infrastructure etc., it also contributes to failures and brings new cyber-physical threats [2]. A Stuxnet-style attack on US SGS could cost $1 trillion to the US [3]. Advanced persistent attacks on the Ukraine Power Grid [4], [5] demonstrate that cyber attacks on such critical systems will be frequent. Therefore, it is important to identify the vulnerable functions in the system and protect them by patching before they are compromised.

Researchers have focused on analyzing the impact of cyber attacks on components of the SGS [6]-[8]. Resilience evaluation is an important task that explains the behavior of the system in the presence of attacks. It is important to determine how a system administrator (SA) should make effective decisions so that attacks do not propagate to other parts of the system. This is difficult because the state space of a system increases exponentially (x^y) with the increase in a number of system functions (y) and status of functions (x). The challenge for the SA is to decide which system to scan to determine whether it is vulnerable and which system to patch. Scanning and patching system functions in critical infrastructures are complex and time-consuming tasks. To perform either of these actions, system engineers must place the function in standby mode, and it is not possible to stop the functionality of various functions simultaneously. Furthermore, an SA knows the internal network but not the position where an attack might have occurred until the symptoms of the attack are observed. The problem of performing efficient vulnerability assessment and patching a system component/function has largely been ignored in the current literature.

In this paper, we propose the **R**einforcement **L**earning for **B**ayesian **A**ttack **G**raph for **S**mart Grid System (**RL-BAGS**) tool to provide functionality to compute optimal policies on regular intervals whether to scan or patch a function of the SGS. This tool is an extension of the BAGS tool [10]. RL-BAGS takes functions, network architecture and vulnerability report (if available) of the SGS to generate a Bayesian Network (BN). BN represents the state of the system where smart grid functions are in different states (such as VULNERABLE, HACKED, etc.). RL-BAGS implements two Reinforcement Learning (RL) algorithms, *Q-Learning* and *SARSA learning*, on the generated BN to learn optimal policies. RL-BAGS assists SA performing in-depth studies of one of the functions of the SGS (as in BAGS) advising effective actions to scan or patch a system component. Smart grid engineers should incorporate this tool into their system to make optimal decisions on regular intervals.

RL-BAGS not only assists in analyzing system resilience but supports actions to maintain resilience. Administrators can devise appropriate secure strategies and deploy resources effectively and efficiently using RL algorithms. The structure of this paper is as follows: Section II describes the related work. Section III discusses the motivation of the work. Section IV provides System Description. Section V describes RL algorithms. Section VI explains Experimental Setup. Section VII provides Simulation Analysis. Section VIII and IX discusses the Conclusion and Future Work respectively.

II. Related Work

Traditionally, researchers have focused on modeling and analyzing the impact of Cyber-Physical Threats (CPT) on functions and components of the SGS [7]-[12]. Suzhi Bi and Ying Jun [7] (2016) introduced a graph-based framework for analyzing the security of SGS state estimation. They represented the power network in the form of an undirected graph where vertices and edges represent buses and transmission lines respectively, and phasor of the bus is used to represent the state of the power system. Then, authors describe the state estimation problem over the power system graph and explained how graph algorithms, such as Maximum flow, S-T cut, Tree pruning etc., are used to perform security analysis.

Parisa *et al*. [12] (2016) demonstrated the interdependence between the power grid and communication network by performing vulnerability analysis in the dynamic power grid. The authors evaluated the impact of cyber attacks on the system in the form of line and links removal. Lu *et al*. [8] presented a risk assessment approach for power grid considering the reliability of the information system. They evaluated the influence of the information system on the security of the power grid and reliability model is established. Liu *et al*. [11] (2017) presented a framework for analyzing the resilience of the power grid with integrated microgrids in the presence of extreme conditions. The authors described the impact of extreme events on the transmission, distribution, and microgrid. The authors evaluated the impact using resilience indices such as Index *K* (measures the expected number of lines on outage), index loss of load probability (load not being supplied), and index to measure the difficulty level of grid recovery.

Wadhawan and Neuman [6] (2017) presented a roadmap to evaluate the resilience of SGS through contingency simulations in the Power World simulator considering attacks on multiple functions of the SGS. These approaches demonstrate how to model the system and evaluate its resilience in the presence of cyber-physical attacks, but they do not discuss how to provide security. Providing security to such critical systems is a complex task. For providing security, SA should understand how the system is behaving, which functions are vulnerable and when to patch those functions. Since SGS implements various functions, it is hard and costly for SA to decide what action to take in a particular state.

With a motive to make optimal decisions in different system states, the concept of RL [13], [14] has frequently been used in securing CPS [15], [16]. Yan *et al*. [15] (2016) analyzed the vulnerability of the transmission grid in the presence of sequential topology attacks using the Q-learning framework. The authors identify the critical attack sequences by considering the behavior of the physical power system. Richard *et al*. [16] (2017) described a Markov game on a network with incomplete information between defender and attacker of a network which is a part of the internet. The game is formulated as an adversarial sequential decision making problem played between two agents using Q-learning, Neural Network, and Monte Carlo learning. Glavic *et al*. [9] described the efforts in the field of electric power system decision and control using RL. The number of researchers has worked on various problems such as Transient angle instability, voltage control, etc. but none of them have worked on the security of the power system from a cyber threat point of view. The problem of making efficient decisions for security assessment is largely been ignored in the current literature [9].

III. Motivating Domain

While many organizations face the challenge of cyber-physical threats, we highlight some of the challenges faced by companies providing security solutions to critical infrastructures. Most companies provide solutions to implement Security Operations Center to manage risks and provide better security for the smart grid infrastructure. The problems faced by engineers are: 1) how to analyze the system status and information flow in real time, 2) discover and analyze the vulnerabilities associated with the system functions, and 3) decide when to patch the discovered vulnerabilities.

To address these challenges, in [10], we proposed the BAGS tool to quantify the SGS resilience in the presence of cyber-physical attacks in real time. BAGS considers functions, network architecture, applications and a vulnerability report as input and generates three Bayesian Networks (BN): Functional Bayesian Network, Network Bayesian Network, and Vulnerability Bayesian Network. BAGS enables SA to analyze how a failure of a network component controlling a particular power grid functionality propagates from the cyber to the physical domain and quantify the probability of its compromise using CVSS scores [17].

In BAGS, we assume SA has already performed a vulnerability assessment of different system functions and we use a vulnerability report to compute the probability of compromise of a particular component. The problem is: it is not possible to perform the vulnerability assessment of all the functions simultaneously. It is costly because it affects the normal working of the function and in such critical systems, it is not advisable to place functions on standby for testing. Furthermore, to patch vulnerable components takes time and for that duration, components might not be functional. When SA has multiple vulnerable components, it is hard to make a choice which component to patch first. There are two actions corresponding to each component: 1) SCAN-X to find out whether component X is vulnerable, patched or hacked and 2) PATCH-X to patch the vulnerabilities of system X. SA has to choose which system to scan or patch in a given system state. To compute the optimal policy for a time interval, we use model-free RL algorithms. The optimal policy will provide an action for each state that maximizes the discounted future rewards for the SA.

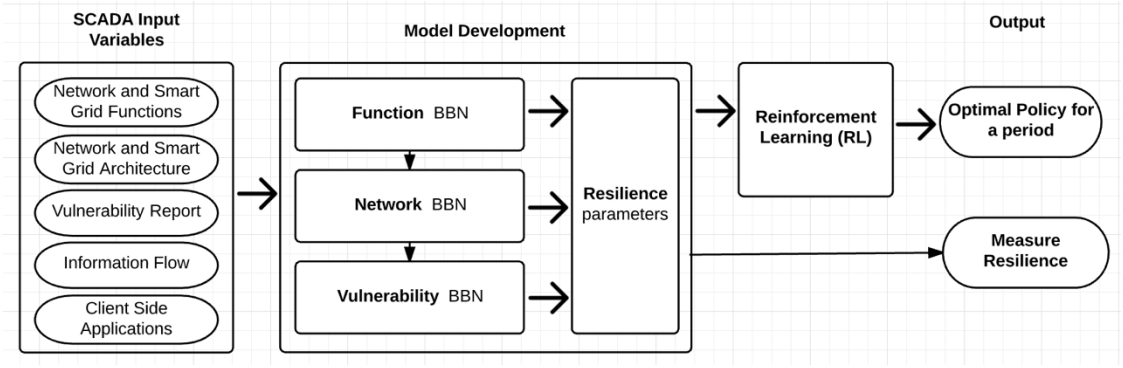

Fig. 1. RL-BAGS tool design.

IV. System Description

In this section we describe the tool design for RL-BAGS. We demonstrate how to model the smart grid system in the form of Markov Decision Process (MDP). Finally, we describe what actions an SA (learning agent) can perform, the different states of the system, and how rewards are calculated corresponding to each system function.

A. RL-BAGS

In RL-BAGS (see Fig. 1), we provide functionality to compute optimal policy for the SGS using RL algorithms represented by BAGS [10]. The optimal policy represents the set of actions that SA should take when the system is in a particular state to maximize the discounted future rewards. The remaining functionality of the tool is same as described in BAGS [10]. It provides a way to evaluate the resilience of the system in real time and map system resources to resilience parameters.

B. System Modeling

AGENT: The system has only one agent who performs actions on the system: defender (SA). Although there could be multiple defenders trying to protect the system, in this work we consider only one agent that abstracts all other defenders. The main goal of the defender is to patch all the functions of the system. The defender has no idea of the state of the system. He should scan the nodes to know their status and perform patching. He is allowed to perform one action on a single node at a time due to cost constraints because it is impossible to keep functions in standby mode in a critical infrastructure. We have not considered attacker in this work. Instead, we specify the function states as HACKED to understand the behavior of defender when functions are compromised.

ACTIONS: The agent has two actions at disposal: SCAN-X and PATCH-X. X denotes the symbol of the function considered by the SA. In total, we have 7 functions in the graph. Therefore, SA has 14 actions. *A* denotes the action set:

{SCAN-S1, PATCH-S1, SCAN-S2, PATCH-S2, SCAN-S3, PATCH-S3, SCAN-S4, PATCH-S4, SCAN-S5, PATCH-S5, SCAN-S6, PATCH-S6, SCAN-S7, PATCH-S7}.

Whenever an action is performed by the SA, a reward is observed which depends on the functional importance described in the next subsection.

SYSTEM: The Functional Bayesian Network (FBN) built in the BAGS [10] represents the state of the system. The node represents a function of the SGS such as Smart Meters (S1), Outage Management (S4), etc. The edges represent the flow of information from a function to another. Each function in the FBN consists of network components and each network component has some vulnerabilities associated with it. In order to discover the set of vulnerabilities, SA must perform the vulnerability assessment. On the basis of vulnerabilities discovered, BAGS computes the probability of compromise for each function using CVSS score [17]. We use this probability to compute importance of each function. Initially, we assume SA has no idea of the status of all the functions and he must learn about the state by performing actions. A function can stay in any one of the following states:

{UNKNOWN, VULNERABLE, PATCHED, HACKED}

The initial state (s_i) of the system is when all the functions' status is *UNKNOWN*. SA does not know the status of any function. The terminal states (s_t) of the system are as follows:

TERMINAL STATE 1: functions' status are PATCHED

TERMINAL STATE 2: function S7 is HACKED.

In the first terminal state, SA will receive 500 reward when he is able to PATCH all the systems. If some systems are HACKED, SA cannot control their functionality and therefore, he should remove those system instances out of the network for scanning. The status of system will not change once they are hacked until SA patches it.

In the second terminal state, if the attacker is able to compromise S7, he can control the power dispatch and SA has failed in protecting the system. SA will receive reward of -500. Since we have 7 (y) functions in the graph and each function can be of any 4 (x) states, the state space will consist of (x^y) 4^7=16384 states. An example of a state:

s: {S1: VULNERABLE, S2: VULNERABLE, S3: VULNERABLE, S4: VULNERABLE, S5: VULNERABLE, S6: VULNERABLE, S7: VULNERABLE}.

C. Function Significance

The significance of each function of the SGS is precomputed in the system and it changes with the change in one of the following factors:

1. *Asset value (AV):* The value of the asset to the SGS.

2. *Rate of Occurrence (RO):* The frequency at which attacks happen on a specific function.
3. *Risk Exposure (RE):* The amount of loss to the smart grid if a particular function is compromised.
4. *Probability of Compromise (POC):* The average exploitability score that is calculated using CVSS score (see [10], [17]) based on the vulnerabilities.
5. *Influence of Function (IOF):* The influence of a function on the smart grid.
6. *Cost to Patch (CTP):* Each function consists of network components. To patch those network components, there is an inherent cost involved.

The rewards of a function *f* is computed as:

$$\text{Importance}_f = \text{AV}\times\text{RO}\times\text{RE}\times\text{POC}\times\text{IOF} - \text{CTP} \quad (1)$$

The probability of compromise parameter is frequently changed because the vulnerabilities are dynamic. Initially, this value is not known because SA does not know whether functions are vulnerable or not. SA scans a function to discover its status. In the simulation, SA performs an action randomly on a state and rewards are observed. SA must know the status of the function before taking PATCH action. The function importance should be calculated either through data using statistical measures or on the basis of SA's experience.

D. State Transition and Rewards

Table 1. represents how the status of the function changes and what rewards SA will receive for performing an action. If SA performs SCAN function and the previous status of the function is HACKED, the new status will remain HACKED. SA receives -200 rewards since it is a bad move. Similarly, if the previous state is VULNERABLE and next state is also VULNERABLE, it is a bad move and SA receives -200 rewards. If the previous state is UNKNOWN, with random distribution we decide whether it is PATCHED, VULNERABLE or HACKED. If it is PATCHED, SA receives 0 rewards because it was not worth scanning it and if it turns out to be VULNERABLE or HACKED, SA receives function importance rewards. Now SA can perform PATCH action on the VULNERABLE function. If SA performs PATCH function and the previous status is HACKED or VULNERABLE, the next status will be PATCHED. It is possible that a node which is in the PATCHED state can be found in VULNERABLE or HACKED state.

V. Reinforcement Learning

Reinforcement learning is an area of machine learning that explains how agents should act in an environment to maximize their cumulative rewards. RL algorithms are modeled as MDP [14]. The agent interacts with the environment in discrete time steps by performing an action (see Fig. 2). At each time step, it receives a reward (an observation) and the environment moves to a new state where another action is chosen. The algorithms follow the same routine until a terminal state is reached or algorithm converges. In this section, we describe two model-free temporal difference RL algorithms implemented for our problem.

TABLE 1: FUNCTION STATE TRANSITION AND REWARD FUNCTION

Action	Previous State	Next State	Rewards
SCAN	HACKED	HACKED	-200
	VULNERABLE	VULNERABLE	-200
	UNKNOWN	VULNERABLE	importance
	UNKNOWN	HACKED	importance
	UNKNOWN	PATCHED	0
	PATCHED	VULNERABLE	importance
	PATCHED	HACKED	importance
	PATCHED	PATCHED	-200
PATCH	HACKED	PATCHED	importance
	VULNERABLE	PATCHED	importance
	PATCHED	PATCHED	-200
	UNKNOWN	UNKNOWN	-200
-	ANY	TERMINAL STATE 1	500
-	ANY	TERMINAL STATE 2	-500

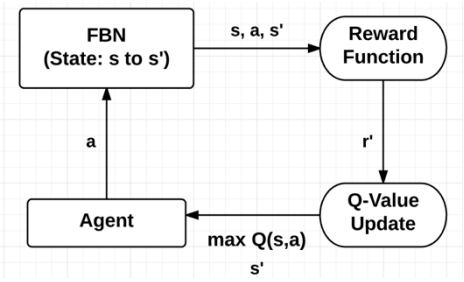

Fig. 2. Q-Learning and SARSA learning Iteration. Agent (SA) performs an action on (s) system state represented by FBN. The system moves to a state (*s*'). The values *s*, *a*, *s*' are given to the reward function that computes reward *r*' and then to Q-value update function. Finally, agent observes *s*' and maximum value of $Q(s,a)$.

A. Q-Learning

Q-Learning is a model-free RL algorithm [13]. It is proven that it converges to an optimal policy for finite number of states, and actions for a single agent. The model-free means we do not need state transition functions to move from one state to another. In this algorithm, the action-value function, *Q,* directly approximates to optimal action-value function, *Q**. *Q* value is assigned to each state-action pair. It is an off-policy temporal difference algorithm. It does not depend on the policy followed by the agent. The agent chooses an action in the environment based on the ε-Greedy Policy. *Q*-learning could be sensitive to local optima problems. It forces the agent to take some actions at random (non-optimal) with probability ε (exploration) and optimal action (exploitation) with probability $(1-\varepsilon)$. This policy allows the agent to explore all the states possible and eventually it will tune the Q-values by choosing the best actions that maximizes the discounted rewards. After performing an action, the system moves to a new state and agent receives reward. The reward is used to update the $Q_t(s, a)$ values corresponding to each state and action. The total rewards Q is computed according to (2) and update rule for Q-learning is (3):

$$Q = \sum_{t}^{n} \gamma^{t-1} r_t(s_t, a_t) \quad (2)$$

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\left[r' + \gamma \max_b Q(s',b) - Q_t(s,a)\right] \quad (3)$$

where $\alpha \in [0,1]$ is the learning parameter, and $\gamma \in [0,1]$ is the discount factor to prefer future rewards. The above rule updates the Q value for the last state (s) and action (a) pair with respect to the observed outcome state (s') and reward (r'). The optimal actions corresponding to a particular state is determined by:

$$a_t = \arg\max_a Q(s_t, a) \quad (4)$$

B. SARSA (λ) Learning

SARSA stands for State-Action-Reward-State-Action. It is an on-policy model-free temporal difference algorithm. In contrast to Q-learning, it does not find the best action possible that maximize the rewards rather it continues to choose next action using the same policy. In SARSA learning, the action is taken on a specific state and reward is received, the system moves to next state and using the same policy another action is chosen. The update rule for SARSA learning is:

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\left[r' + \gamma Q(s',a') - Q_t(s,a)e_t(s,a)\right] \quad (5)$$

$$e_t(s,a) = \begin{cases} \gamma\lambda e_{t-1}(s,a) + 1 & (\text{if } s = s_t \text{ and } a = a_t) \\ \gamma\lambda e_{t-1}(s,a) & (\text{otherwise}) \end{cases} \quad (6)$$

The idea of using eligibility trace is to apply temporal difference prediction to state-action pairs instead of states. λ refers to the use of eligibility trace. When λ=0, there is a one-step backup return as in Q-learning. When λ=1, there is one episode backup return as in Monte Carlo. In SARSA learning model, λ=0.4. e_t (s, a) in (5) denotes the trace of state-action pair.

VI. Experiment

A. Assumptions

In our simulation, we made the following assumptions:

1. In SGS, the availability of the data is a measure problem. Therefore, for computing the rewards of the functions, we logically assign the values to the variables (AV, RO, POC) in a way that important function value is more.
2. In real time, POC values are adjusted when SA SCANS a function because without scanning he would not know the set of vulnerabilities and hence POC value. In the simulation, we calculate the POC values for all in advance with their function importance because there is a limitation of the database to compute function importance.
3. SA must compute optimal policies after some regular interval because the probability of compromise of the system changes over a period of time. It is because either vulnerability of the system change or SA has patched the network components. We have not considered the change in the probability of compromise.
4. SGS is a complex system that contains many network components. Including all network components in the simulation will increase the state space exponentially and is impossible to track. Therefore, we grouped the network components under the function they support for. We use the FBN [10] as a state of the system.
5. In the simulation, we are computing the optimal policy for the defender (SA). In order to simulate an attacker, we introduce HACKED status for functions randomly during simulation so that to learn the behavior of the SA in the presence of HACKED states.
6. We have not considered the time it takes to patch or scan a system. We assume it is to be the same for all functions.

B. Experiment Setup

We use the BURLAP [18] library to implement Q-learning and SARSA learning algorithm. Our main task was to model the problem in the form of MDP without state transition probability function and implement both learning algorithms. We implemented the StateWorld class to represent the model of the system. In StateWorld, the function takes state and action as function parameters and returns the next state. We implemented the Reward and Terminal state interface that forms the part of the StateWorld. The Q-learning algorithm runs 100,000 episodes and SARSA learning algorithm runs 300,000 episodes to explore the state space and learn optimal policy. In each episode, the agent learns from the initial state until it reaches one of the terminal states (described in section IV-B) and updates the Q-values within each episode. At the end of the simulation, the Q-values converge to an optimal value and we can compute optimal policy using Eq. (4).

C. Agent Learnning Procedure

Each episode starts from the s_i state. The defender randomly selects a function to SCAN (SCAN-SX) and discover whether it is VULNERABLE or HACKED. The system moves to the next state s_{i+1} where the function SX is in the discovered state. And defender receives some reward for performing that action in state s_i and moves to state s_{i+1} according to Table 1. The reward received by the defender depends on the importance of the function on which action is performed, what action performed and its discovered status. If the s_{i+1} state is a terminal state, the episode ends. We store the action sequence, state sequence, and rewards of each episode to plot the learning graph.

An example of how defender behaves and learns. Suppose the initial state of the system is described in Fig. 3 (left side). Initially, the status of S2 is UNKNOWN. SA takes action SCAN-S2 and discovers that S2 is in the HACKED state. The system moves to the state described in Fig. 3 (right side) and SA receives reward importance of S2. SA updates the Q values of the state-action pairs using the reward received according to (3) and (4) equations in Q-learning and SARSA learning algorithms respectively. Now, SA takes PATCH-S2 action and the system moves to state Fig. 4 (right side) where S2 is patched. SA receives a much higher reward for patching S2 (depends on the importance of the node) and updates its Q values. Similarly, the simulation continues until the final state of the system is reached. The episode ends when either of the TERMINAL STATE 1 or TERMINAL STATE 2 (section IV-B) have been reached and SA receives a reward (Table 1).

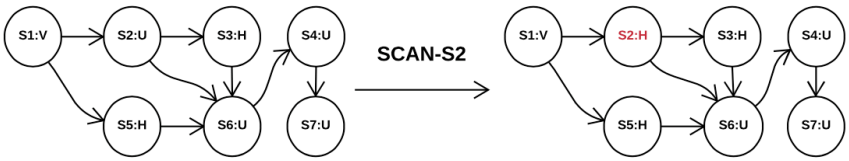

Fig. 3. Each node represents a function and status (V: VULNERABLE, H: HACKED, U: UNKNOWN, P: PATCHED). SA performs action SCAN-S2 and discovers that it is Hacked, the system moves to next state where S2 is Hacked and SA receives reward: function importance.

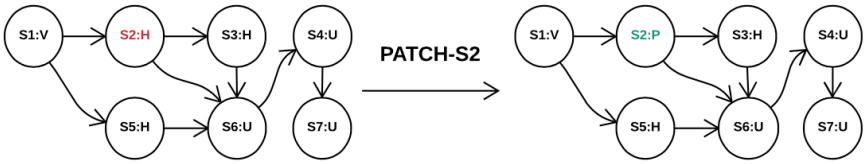

Fig. 4. SD chooses action PATCH-S2 . The status of the function S2 changes to P and system moves to next state described in right side figure. SA receives reward in terms of the importance of S2.

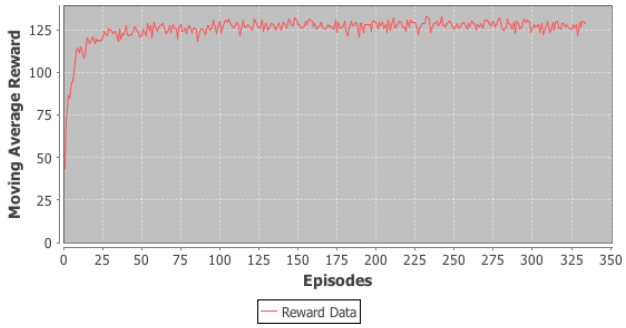

Fig. 5. Q-Learning: Plot of the moving average of the 300 average rewards per episode for 100,000 trials with ε-Greedy policy for exploration and exploitation ε=0.2, learning rate α=0.2 and discount factor γ=0.2.

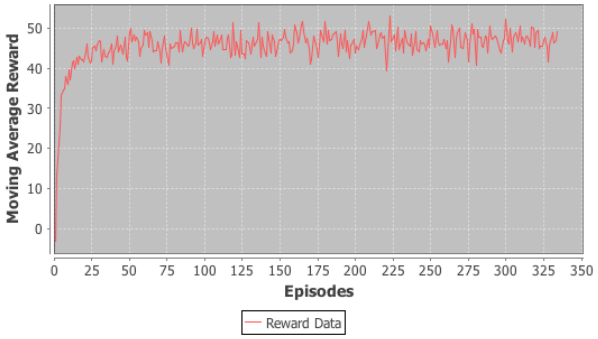

Fig. 6. Q-Learning: Plot of the moving average of the 300 average rewards per episode for 100,000 trials with ε-Greedy policy for exploration and exploitation ε=0.6, learning rate α=0.2 and discount factor γ=0.2.

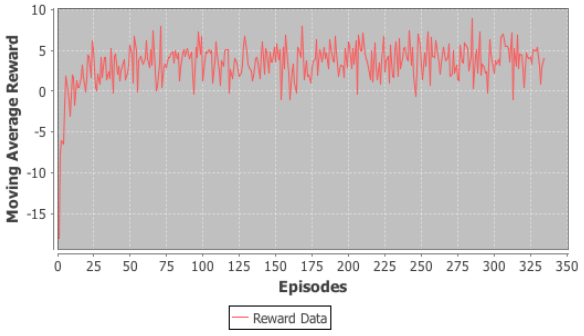

Fig. 7. Q-Learning: Plot of the moving average of the 300 average rewards per episode for 100,000 trials with ε-Greedy policy for exploration and exploitation ε=0.8, learning rate α=0.2 and discount factor γ=0.2.

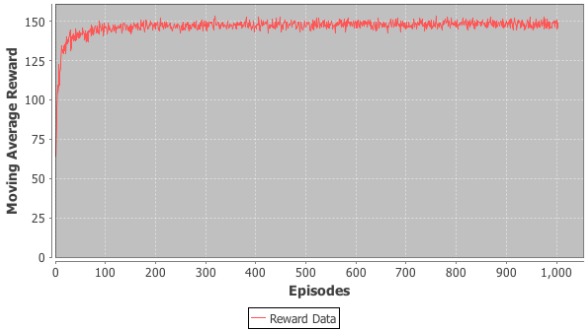

Fig. 8. SARSA-Learning: Plot of the moving average of the 300 average rewards per episode for 300,000 trials, learning rate α=0.2, lamda λ=0.4 and discount factor γ=0.2.

VII. Simulation Analysis

The results can be seen in Figs. 5, 6, 7 and 8, which show the average reward per episode against the number of episodes we run the Q-learning (see Figs. 5, 6, 7) and SARSA (see Fig. 8) learning algorithms. We calculate the moving average reward per 300 episodes over: 1) 100,000 episodes for Q-learning for ε value 0.2 (Fig. 5), 0.6 (Fig. 6) and 0.8 (Fig. 7), and 2) 300,000 episodes for SARSA learning. If we set ε value to any other constant between [0,1], still the algorithm will converge to the optimal policy. The exploration (with probability ε) diminishes over time and policy becomes greedy and thus optimal. The only difference is that agent takes random action with high probability even if the optimal policy is learned. In Fig. 5, the agent takes optimal action with probability 0.8 (and 0.2 for random actions), therefore we see stable average rewards after some episodes are completed. In contrast to Fig. 5, in Fig. 7, the agent takes random action when an optional policy has been computed.

One can easily make the difference between the Q-learning (off-policy) and SARSA (see Fig. 8) learning (on-policy) approach. The average reward per episode in SARSA learning is more than Q-learning. This shows that SARSA learning is useful when we want to optimize the reward for SA that is exploring the state's space of the system (not exploiting the best possible action). SARSA follows the current exploration policy which may or may not be greedy. And it is possible that SARSA will find different policies than Q-learning (in the case

when following exploration strategy leads to huge penalties). For example, if a function status in our model is VULNERABLE and optimal action may not be to patch this function for now. But the function is in VULNERABLE state and attacker can hack that function, SARSA learning will discover this and PATCH that function.

On the other hand, Q-learning will PATCH that function if it is the optimal action in the current state (return maximum reward for the state and action pair). This is the reason why SARSA learning coverages and reaches terminal state quickly and may have a different optimal policy as compared to Q-learning. Q-Learning may take a long time to find an optimal policy as compared to SARSA learning and it always computes the optimal policy at a particular state [13]. SARSA learning algorithm should be used by the agent who wants to explore different strategies quickly and optimize the reward value at the same time. Once the *Q-values* converges, we calculate the optimal policy using Eq. 4. It returns the action corresponding to a state that maximizes the *Q-value*. This will be stored within the tool after computation so that SA should be able to access which action to take in a particular state. Consider state:

s: {*S1: UNKNOWN, S2: UNKNOWN, S3: PATCHED, S4: VULNERABLE, S5: PATCHED, S6: UNKNOWN, S7: VULNERABLE}*.

Using Q-Learning results (from the simulation of Fig. 7), the best action is S7-PATCH (Electricity Control Center) with Q-value for this state is 282.2781. Since S7 is vulnerable, it is beneficial to protect S7 first otherwise SA will receive -500 rewards if it gets compromised. It is also possible that algorithm learns to patch some different function because that may lead to compromising the S7 function. It depends on the importance of the node how algorithm learns the optimal policy. If we give importance to nodes that are present in the starting of the graph, the agent will learn a policy that SCAN and PATCH those functions first even if functions present in the end are vulnerable. It is recommended for the SA to choose function importance values carefully to learn effective optimal policy. SA should implement this algorithm after every regular interval so that to incorporate changes in the system such as the change in vulnerability, the importance of the functions, frequency of attacks occurred on the system, etc. Furthermore, SA can learn the optimal policy for given number of timesteps. Since there are more than 16000 states possible theoretically, but in reality, in a given period, the system will be in few states. It is efficient to learn the policy for the given number of timesteps.

VIII. Conclusion

In this paper, we extend BAGS tool to RL-BAGS with the motive to compute optimal resource allocation policy for the smart grid security in the presence of cyber attacks. We implement two RL algorithms over BAGS, Q-Learning and SARSA learning, on the generated BN to learn optimal policies. The results showed that it is possible to learn the policy using model-free RL algorithms. The most important parameter is function importance which must be computed carefully. We discussed that Q-learning provides an optimal policy by exploiting the best possible action rather following the current exploration policy in case of SARSA learning. SARSA learning should be used by the attacker to explore all the states quickly and optimally.

We can learn the optimal policy for the attacker in a similar way as of defender. The attacker has to try all the options to decide which one was optimal because he does not have complete knowledge about the system (partially observable). Unlike defender, he does not know the system functionality until he scans the system. Since defender is the SA, he knows the functionality and initial importance of the system (without considering the effect of vulnerabilities). SA should continuously learn the optimal policy for the change in the vulnerabilities because it will change the importance of the function. We can also build a recommendation system where SA will provide their recommendations to change function importance manually. RL-BAGS assists SA performing in-depth studies of one of the functions of the SGS and advising effective actions to scan or patch functions. Furthermore, smart grid engineers should incorporate this tool into their system to make optimal decisions at regular intervals. RL-BAGS not only assists in analyzing system resilience but supports actions to maintain resilience.

IX. Future Work

We plan to extend this problem to 1) partially observable environment for the attacker to compromise the system and, 2) two-player stochastic game between defender and attacker and analyze competitive interaction between them.

MODELING TWO PLAYER GAME: The system description remains same as described in section IV. Now we have two agents [19] who take actions in a system state and receive separate rewards. The actions of the SA remains same. The actions of the attacker are SCAN-X and HACK-X. The attacker can scan only those functions (systems) that are directly accessible from the compromised function.

Initially, we assume the attacker has compromised a function such as smart meters, vendor servers, etc. that is easily reachable from the internet. The motive of the attacker is to reach Electricity control center so that to control power dispatch. The motive of the defender is to prevent attacker reaching the goal state by patching the functions. There are two TERMINAL states: 1) if defender and attacker are on the same node, the defender catches the attacker, and 2) when the attacker has compromised the goal function. At each state, both the players perform an action and system moves to a new state defined by the state transition function. Now, the motive is to learn what actions agents should take at each state to maximize their discounted future rewards and find game equilibria at the same time. Therefore, we will implement MultiAgentQLearning using CorrelatedQ as a backup operator to learn Correlated Equilibrium (CE) [20]. Correlated Equilibrium provides a joint strategy for the agents so that there is no incentive for an agent to deviate from their behavior. The details of about how we modeled the two-player for computing CE and results of experiments will be discussed as a part of our future work.

ACKNOWLEDGMENT

This work was conducted with partial funding by Northrop Grumman Information Systems through the Cyber Security Research Consortium.

REFERENCES

[1] R. Baheti and H. Gill, "Cyber-physical systems." *The Impact of Control Technology*, vol. 12, pp. 161-166, 2011.

[2] C. Neuman and K. Tan, "Mediating cyber and physical threat propagation in secure smart grid architectures," In *Proc. of IEEE Int. Conf. on Smart Grid Communications*, 2011, pp. 238-243.

[3] Stuxnet style attack on US Smart Grid could cost governemnt $1 trillion. [Online]. Available: https://www.scmagazineuk.com/stuxnet-style-attack-on-us-smart-gridcould-cost-government-1-trillion/article/535452/

[4] Ukraine's Power outage was a cyber attack: Ukrenergo, 2017. [Online]. Available: http://www.reuters.com/article/us-ukraine-cyber-attack-energyidUSKBN1521BA

[5] Analysis of the Cyber Attack on the Ukrainian Power Grid, March 2016. [Online]. Available: http://www.nerc.com/pa/CI/ESISAC/Documents/EISAC_SANS_Ukraine_DUC_18Mar2016.pdf

[6] Y. Wadhawan, C. Neuman, and A. AlMajali, "Analyzing cyber-physical attacks on smart grid systems," In *Proc. of IEEE Workshop on Modeling and Simulation of Cyber-Physical Energy Systems*, 2017, pp. 1-6.

[7] S. Bi and Y. J. A. Zhang, "Graph-based Cyber Security Analysis of State Estimation in Smart Power Grid," *IEEE Communications Magazine*, 15 Feb, 2017. Volume:PP, Issue:99. DOI: 10.1109/MCOM.2017.1600210CM.

[8] D. Lu, Y. Liu, & Y. Zeng, "Risk assessment of power grid considering the reliability of the information system," In *Proc. of IEEE Int. Conf. on Smart Grid Communications*, 2016, pp. 723-728.

[9] M. Glavic, R. Fonteneau, and D. Ernst, "Reinforcement Learning for Electric Power System Decision and Control: Past Considerations and Perspectives," In *Proc. of The 20th World Congress of the Int. Federation of Automatic Control*, 2017, pp. 1-10.

[10] Y. Wadhawan and C. Neuman. "BAGS: A Tool to Quantify Smart Grid Resilience," in *Proc. of Int. Workshop on Cyber-Physical Systems (IWCPS)*, 2017, DOI: 10.15439/2017F77.

[11] X. Liu, M. Shahidehpour, Z. Li, *et al*., "Microgrids for enhancing the power grid resilience in extreme conditions," *IEEE Trans. on Smart Grid*, vol. 8, no. 2 pp. 589-597, 2017.

[12] P. Akaber, B. Moussa, M. Debbabi, and C. Assi, "Cascading link failure analysis in interdependent networks for maximal outages in smart grid," In *Proc. of 2016 IEEE Int. Conf. on Smart Grid Communications*, 2016, pp. 429-434.

[13] C. Watkins and P. Dayan. "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279-292, 1992.

[14] R. Bellman, "A Markovian decision process," *Journal of Mathematics and Mechanics*, pp. 679-684, 1957.

[15] J. Yan, H He, X Zhong, and Y Tang. "Q-learning-based vulnerability analysis of smart grid against sequential topology attacks," *IEEE Trans. on Information Forensics and Security*, vol. 12, no. 1, pp. 200-210, 2017.

[16] R. Elderman, L. J. J. Pater, A. S. Thie, *et al.*, "Adversarial Reinforcement Learning in a Cyber Security Simulation," In Proc. of *ICAART*, 2017, pp. 559-566.

[17] CVSS Score. [Online]. Available: https://www.first.org/cvss/specification- document

[18] Burlap Reinforcement Learning. [Online]. Available: http://burlap.cs.brown.edu/

[19] G. Neto. "From single-agent to multi-agent reinforcement learning: Foundational concepts and methods." Learning theory course. 2005.

[20] A. Greenwald, K. Hall, and R. Serrano, "Correlated Q-learning," *ICML*, vol. 3, pp. 242-249, 2003.