DREVAN: Deep Reinforcement Learning-based Vulnerability-Aware Network Adaptations for Resilient Networks

Qisheng Zhang, Jin-Hee Cho
Department of Computer Science, Virginia Tech
Falls Church, VA, USA
{qishengz19, jicho}@vt.edu

Terrence J. Moore, Frederica Free Nelson
US Army Research Laboratory
Adelphi, MD, USA
{terrence.j.moore, frederica.f.nelson}.civ@army.mil

***Abstract*—In this work, we proposed a vulnerability-aware network adaptation framework that can generate a robust network topology against epidemic attacks by leveraging deep reinforcement learning (DRL) to build a resilient network. We call our proposed framework DREVAN, representing Deep REinforcement Learning-based Vulnerability-Aware Network Adaptations. The goal of the proposed DREVAN is to minimize security vulnerability caused by epidemic attacks exploiting the software monoculture for node compromise while maximizing network connectivity in terms of the size of the giant component. To be specific, the DREVAN aims to autonomously identify a pair of network adaptation budgets for adding or removing edges (i.e., how many edges to add or remove) by leveraging DRL. In this work, we tackle the inherent challenge of using DRL in reducing the learning curve of a DRL agent by proposing two algorithms. First, we proposed a vulnerability ranking algorithm of edges and nodes, namely VREN, for the DRL agent to select which edges to add or remove based on the lowest expected vulnerability between two nodes or the highest vulnerability of edges, respectively. Second, we also developed a Fractal-based Solution Search algorithm (FSS) to effectively direct the DRL agent towards the effective samples to visit and quickly identify and converge to an optimal solution (i.e., budget sizes of edge additions and removals). Via our extensive comparative performance analysis of the six different schemes, we demonstrated the outperformance of our proposed DREVAN-based schemes over counterpart and baseline schemes.**

***Index Terms*—Network vulnerability, network resilience, network adaptation, epidemic attacks, deep reinforcement learning.**

I. Introduction

Network resilience research has been studied in order to maximize network connectivity even under the presence of node failures or compromise in the network science domain [25]. However, network scientists mainly focused on maximizing the size of the giant component to maintain network resilience, whereas security vulnerability exposed in the network due to epidemic attacks has not been sufficiently addressed. Further, in the network security domain, researchers have proposed the concept of diversity as a security design to ensure software polyculture with the aim of preventing attackers from taking huge advantages of using the same software [30]. When nodes use the same software, this allows attackers to exploit the known vulnerability of a single software in a network with the so-called software monoculture [27]. Although software shuffling based on graph coloring algorithms has been used to solve a software assignment problem [27], their focus is mainly on minimizing security vulnerability. However, software shuffling has been proven less effective in reducing security vulnerability in [29], which showed the clear outperformance of edge shuffling in maximizing system security and network connectivity over software shuffling, which does not change network topology. Further, although a vulnerability-aware diversity metric has been proposed to determine which edges to shuffle [29], identifying an optimal budget of edge adaptations to generate a resilient network topology has not been studied by minimizing security vulnerability and maximizing network connectivity in the presence of epidemic attacks.

In this work, we aim to develop a vulnerability-aware network adaptation framework that can generate a robust network topology against epidemic attacks by leveraging deep reinforcement learning (DRL). We call our proposed framework DREVAN, representing Deep REinforcement Learning-based Vulnerability-Aware Network Adaptations.

We make the following **key contributions** in this work:

- Although DRL has been used to generate an optimal robust network topology against adversarial attacks, this is the first that considered both edge additions and removals to maximize network resilience by achieving the conflicting goals of maximizing system security and network connectivity.
- We presented two algorithms to support the DRL agent to efficiently identify an optimal adaptation budget strategy to meet the two system goals. They include: (1) the Vulnerability Ranking algorithm of Edges and Nodes, namely VREN, and using the vulnerability levels to select candidates for edge adaptations; and (2) the Fractal-based Solution Search algorithm, namely FSS, to substantially reduce the steps to identify an optimal adaptation budget of edge additions and removals.
- Via simulation experiments, we conducted extensive comparative performance analysis based on six network topology adaptation schemes, including DRL algorithms, three Deep Q-learning algorithms (i.e., DQN, Double DQN, and Dueling

DQN) and three existing counterparts (i.e. Greedy, Random, and Optimal). We found that DRL-based network topology adaptations particularly outperform with regard to minimizing system security vulnerability.

II. Related Work

In this section, we discuss an overview of existing approaches of generating secure network topologies in cyber-physical systems, in terms of diversity-based network shuffling and DRL used to produce secure network topologies for cyber-physical systems (CPSs).

The generation of diversity-based secure network topology has been studied to produce a resilient network against various types of attacks. The diversity concept has been applied to components of network settings [23], such as hardware or software components. This diversity has been shown to achieve enhanced system security and survivability [10]. Hence, it becomes more critical to generating high quality diversity, which naturally leads to the question of how to measure diversity. The quality of network diversity has been measured in terms of how multiple variants of software have been deployed. The network diversity has been measured based on the diversity of resources, minimum attack effort to compromise a host, and the average effort to compromise a host [28], or path diversity [21] to maximize flow reliability. However, these prior metrics do not show a clear relation between the defined diversity and its effectiveness in enhancing network security. The deployment of diversity techniques has also significantly impacted their effectiveness by: using a software allocation algorithm to measure diversity leveraging Shannon entropy [23]; solving a software assignment problem for maximizing software diversity using a genetic algorithm [3]; or quantifying the software diversity in each node to determine edge adaptations [29].

Other studies have focused on diversity deployment approaches in networks without developing any novel metrics. These approaches typically involve adapting graph coloring [13] to solve software assignment problems, including an approach to defend against sensor worms [27], distributed algorithms to improve network tolerance [20], and a centrality-based approach to solve a multi-coloring problem [12]. Network topology shuffling has been studied under the paradigm of moving target defense (MTD). Recent MTD network shuffling approaches redirect a certain number of edges to make the network topology more robust against worm attacks [11]. In addition, MTD network shuffling has handled reconnaissance attacks by changing virtual network topologies [1]. However, determining an optimal number of edge adaptations to obtain an optimal topology considering both security vulnerability and network connectivity has not been studied.

Recently graph embedding-based topology adaptation algorithms using deep reinforcement learning (DRL) have been proposed to develop adversarial attacks in graph neural networks (GNN). Dai et al. [5] used structure2vec (S2V) to obtain a node embedding, which was used by the DRL agent to learn two independent Q-functions in deep Q-learning (DQN). However, their work only considered removing a single edge to perform adversarial attacks on GNN models. Similarly, Darvariu et al. [6] adapted a network topology by adding edges from an empty network based on an S2V variant to obtain a node embedding and used a customized metric to evaluate network robustness. Chai et al. [4] proposed a DRL-based network topology shuffling MTD framework to deal with Distributed Denial-of-Service (DDoS) attacks by shuffling connections between users and servers in a given CPS. Zhang et al. [31] proposed a mechanism to periodically redirect routes in which DQN agents determine a mutated route to thwart DoS and targeted attacks to routes. Unlike these works, our approach considers both the additions and removals in the edge rewiring processes and leveraged DRL to identify an optimal number of edge adaptations which will support mitigation of attacks while the attacker expends their resources with little to no success.

Even though the above works leveraged DQN to identify an optimal network topology robust against adversarial attacks, due to the inherent nature of the problem complexity with a large solution space, it is highly challenging for a DRL agent to achieve fast convergence or even find the optimal solution. We propose a novel, break-through idea to reduce this solution space. In addition, our network adaptation algorithm allows the DRL agent to intelligently converge to an optimal (or close-to-optimal) solution without suffering much from any lack of convergence in the reward, which has been a common and challenging issue in DRL research.

III. Problem Statement

In this work, we aim to identify a network topology robust against epidemic attacks by adding or removing edges between nodes to minimize security vulnerabilities introduced by compromised nodes (see our attack model in Section IV-C) while maximizing network connectivity even under epidemic attacks within the limits of budget B to adjust edges. We leverage the state-of-the-art algorithms of DRL, particularly a series of Deep Q-learning network (DQN) (i.e., regular DQN, Double DQN, and Dueling DQN), and let a DRL agent identify a pair of budgets for adding edges, b_A, and removing edges, b_R, given the constrained budget $0 \leq b_A + b_R \leq B$.

Given an original network topology G, the DRL agent aims to generate the pair of edge adaptations (i.e., (b_A, b_R)) and generate an adapted network G' by identifying (b_A, b_R), respectively. Since it is an NP-hard problem to identify an optimal robust network topology against epidemic attacks by considering all possible edges to remove or add, we use two heuristic algorithms to substantially reduce the solution space while maintaining high solution optimality: (1) *Vulnerability Ranking of Edges and Nodes* (VREN) is designed to generate vulnerability scores of existing edges and nodes based on how often an edge can be used by attackers or how often a node appears on attack paths. When the DRL agent identifies an optimal pair of budgets (b_A, b_R), the system simply removes the b_R most vulnerable edges from G and adds the b_A least vulnerable edges to the original network G, resulting in an adapted network, G'. We provided the detail of the VREN in

Section V-A; and (2) *Fractal-based Solution Search* (FSS) is presented to significantly reduce the complexity of identifying an optimal pair of the two edge adaptation budgets, (b_A, b_R), as the complexity increases depending on how many pairs of (b_A, b_R) the DRL agent tries to identify the optimal solution. FSS is designed to significantly reduce this complexity and introduce a fast convergence of the solution identified by the DRL agent, as detailed in Section V-B.

To state this formally, the DRL agent aims to achieve the following objective via the proposed VREN and FSS:

$$\arg\max_{b_A, b_R} f(G') - f(G), \quad s.t. \quad 0 \leq b_A + b_R \leq B, \tag{1}$$

where B is an upper bound of the total budgets that can be used to make edge adaptations, $f(G)$ returns the difference between the two key performance metrics, an expected size of the giant component, $\mathcal{S}_{\mathcal{G}}$, to ensure network connectivity and an expected fraction of compromised nodes, $\mathcal{F}_C$, i.e., $f(G) = \mathcal{S}_{\mathcal{G}}(G) - \mathcal{F}_C(G)$. We described how to compute these two metrics in Section VI. Since these two system goals are conflicting to each other (i.e., higher $\mathcal{S}_{\mathcal{G}}$ tends to increase $\mathcal{F}_C$ whereas lower $\mathcal{F}_C$ decreases $\mathcal{S}_{\mathcal{G}}$), it is a non-trivial task for the DRL agent to identify an optimal (b_A, b_R) that can meet both goals. This implies that the optimal solutions identified by the DRL do not necessarily achieve the best performance in both metrics. We observe this phenomenon and discuss their underlying reasons in Section VII.

IV. System Model

In this section, we give the description of our system model including the network model, node model, and attack model along with the assumptions made in this work.

A. Network Model

In this work, we assume a system with a centralized controller that can run the proposed DRL-based network topology adaptation framework, DREVAN. Example network environments include a software-defined network (SDN) with a single SDN controller [15], an Internet-of-Things with an edge server or central cloud server, or a hierarchical wireless sensor or ad-hoc network with a centralized coordinator [14]. Although our work considered a single DRL agent to make a centralized decision for reconfiguring a robust network topology, it can be easily extended to consider multiple DRL agents for a large-scale network. We leave this for our future research as outlined in Section VIII.

We assume that each node's connectivity with other nodes and its software package information is known to the centralized controller so the DRL can make edge adaptation decisions. We represent a network as an undirected graph, $G = (V, E)$, where V is a set of nodes and E is a set of edges existing in the network G. A network topology can be changed by network dynamics introduced by node failures, node compromise by epidemic attacks, or network adaptations made by a chosen network adaptation scheme. We keep track of the network topology using an adjacency matrix $\mathbf{A}$ where the entry $a_{ij} = 1$ if there exists an edge between nodes i and j and $a_{ij} = 0$ if there is no edge between them.

We leverage DRL to generate a network topology that can achieve the goals of maximum network connectivity and minimum security vulnerability. A DRL agent will run on the centralized coordinator to make decisions of identifying an optimal budget for adding and removing edges. Per network topology adaptation cycle, the DRL agent will adapt a given network topology to achieve the two conflicting goals. We provided the detail of the proposed DREVAN using DRL-based network adaptations in Section V.

We assume that there exists a network-based intrusion detection system (NIDS), which is capable of detecting compromised nodes. If a node is compromised by an epidemic attacker and detected by the NIDS with the probability γ^*, it can be interpreted as the removal probability in the Susceptible-Infected-Removed (SIR) epidemic model. If the compromised node is detected by the NIDS, we assume the system runs a rekeying process (i.e., changing secret keys) and isolates the detected, compromised nodes. In the network topology, the detected nodes by the NIDS, including both false positives and negatives, have their all edges disconnected from the network. Note that our work does not focus on developing an IDS, which is out of the scope of this work. Hence, for simplicity, the false negative and positive probabilities of the NIDS are modeled as $\mathrm{P_{fn}} = 1 - \gamma^*$ and $\mathrm{P_{fp}}$ in this work.

B. Node Model

Each node i's attributes are characterized by: (1) $na_i = 1$ or $na_i = 0$ refers to node i being active or dead, respectively; (2) $nc_i = 1$ or $nc_i = 0$ indicates a node being compromised or not, respectively; (3) s_i is the software package installed in node i and l is the number of software packages available in the system in which s_i is in the range of $[1, l]$ as an integer; and (4) sv_i denotes node i's software vulnerability, which refers to the Common Vulnerabilities and Exposures (CVE) [8] score based on the Common Vulnerability Scoring System (CVSS) [18]. In the CVSS, the mean vulnerability of a node for multiple vulnerabilities is estimated as a real number in $[0, 10]$, which is normalized in $[0, 1]$ to represent sv_i as a real number in this work.

C. Attack Model

This work considers the following attack behaviors to investigate the performance of the proposed DREVAN:

- *Epidemic Attacks*: Attacker j can spread malware and infect its adjacent nodes based on the infection probability, β_{ji}, given by [9]:

$$\beta_{ji} = \begin{cases} 1 & \text{if } \sigma_j(s_i) > 0; \\ sv_i & \text{otherwise,} \end{cases} \tag{2}$$

  where $\sigma_j \in \mathbb{B}^l$ defines a boolean vector on l software packages and indicates whether each software package's vulnerability is learned by the attacker j. The β_{ji} returns 1 when attacker j has compromised nodes with the same s_i in the

Algorithm 1 Vulnerability Ranking of Edges and Nodes (VREN)

$G \leftarrow$ An original network
$N \leftarrow$ Total number of nodes in a network
$b_A \leftarrow$ Addition budget
$b_R \leftarrow$ Removal budget
$n_a \leftarrow$ Total number of attack simulations to identify edge and node vulnerabilities based on their appearance on attack paths
$k \leftarrow$ Upper hop bound for edge addition
$G' =$ **VREN**(G, b_A, b_R)
$V_V \leftarrow$ Node vulnerability levels, initialized as a zero vector
$V_E \leftarrow$ Edge vulnerability levels, initialized as a zero vector
1.A: Edges Removals
for $t \leftarrow 1$ to n_a **do**
    $V_V^*, V_E^* =$ EpidemicAttacks(G) ▷ Single attack simulation
    $V_E = V_E + V_E^*$
end for
$R_E \leftarrow$ rank E in descending order with key V_E
$G^* \leftarrow$ remove top b_R edges from G according to R_E
for $t \leftarrow 1$ to n_a **do**
    $V_V', V_E' =$ EpidemicAttacks(G^*) ▷ Single attack simulation
    $V_V = V_V + V_V'$
end for
1.B: Edges Additions
$R_V \leftarrow$ rank V in ascending order with key V_V
$gc \leftarrow$ the giant component of G^*
$R_V^{gc}, R_V^{ngc} \leftarrow$ order gc and the complement ngc based on R_V
$G' = G^*$ ▷ All the following adaptations are on G'
for i in R_V^{ngc} **do**
    for j in R_V^{gc} **do**
        $G' \leftarrow$ Connect nodes i, j and break if $dist(i, j) \in [2, k]$ ▷ $dist$ considers the length of a shortest path between nodes i and j and k is the maximum number of hops allowed
    end for
end for
$b_A^{temp} \leftarrow \|R_V^{ngc}\|$
for $h \leftarrow 2$ to N **do**
    $R_V^h \leftarrow$ top h nodes in R_V
    for i in R_V^h **do**
        $R_V^{i-1} \leftarrow$ top $i-1$ nodes in R_V
        for j in R_V^{i-1} **do**
            break if $b_A^{temp} \geq b_A$
            if $dist(i, j) \leq k$ **then**
                $G' \leftarrow$ Connect nodes i, j if $dist(i, j) \in [2, k]$
                $b_A^{temp} = b_A^{temp} + 1$
            end if
        end for
    end for
end for
return G'

past (i.e., $\sigma_j(s_i) > 0$) and can exploit the vulnerabilities of the software package to compromise other nodes; otherwise, it returns sv_i. As described in Section IV-A, we use the SIR model [19] to consider the behavior of epidemic attacks. That is, an outside attacker can compromise its adjacent nodes without access rights to their settings or files by spreading malwares or viruses, such as botnets [16].

- *State Manipulation Attacks*: Compromised but undetected inside attackers can disrupt the process of the DRL agent's learning and decision process. The DRL agent can learn and make decisions based on the feedback received for a previous action taken, which is considered as a reward to the environment, and aims to reach a desirable system state (i.e., minimum security vulnerability and maximum network connectivity). Hence, it is critical for the DRL agent to keep track of correct system states. In order to validate the resilience of the proposed DREVAN, we considered a state manipulation attack (SMA) with P_s, which is the probability for a system state to be manipulated by the attacker. To counter this type of attack, the proposed DREVAN also uses a detector to detect the SMA with the detection probability, D_r. We evaluate the impact of the SMA by varying P_s, given D_r. Note that developing a detector for the SMA is beyond the scope of our work. It will be considered in our future work. In the DREVAN, if the SMA is correctly detected, the DRL agent will simply use a previous state instead of using a compromised state by the SMA. Otherwise, the DRL agent will use the compromised state to identify an optimal strategy of the edge adaptation budget pair.

V. Proposed DREVAN Framework

In this section, we describe the detail of our proposed DREVAN which uses the proposed VREN and FSS.

A. Vulnerability Ranking of Edges and Nodes (VREN)

The proposed *VREN* can perform network topology adaptations with any given budget pair (b_A, b_R), where b_A and b_R refer to edge addition and edge removal budgets, respectively. We define the node (or vertex) vulnerability levels V_V and edge vulnerability levels V_E in an underlying graph $G = (V, E)$ of a network. We define the vulnerability level of each edge by the number of times used by attackers to compromise other nodes. Similarly, we define the vulnerability level of each node by the number of times it can be compromised by attackers as it appears on attack paths. Then, we calculate V_V and V_E by running n_a attack simulation runs on G. To this end, `EpidemicAttacks`(G) is implemented and returns the vulnerability values of nodes and edges in graph G based on the frequency of them appearing on attack paths. Lastly, we rank edges and nodes respectively based on their vulnerability levels. We denote the rank of edges and nodes as R_E and R_V, respectively, where R_E is sorted in descending order and R_V is sorted in ascending order. The R_E and R_V converge as n_a increases. Hence, choosing an appropriate n_a is critical. The expected fraction of compromised nodes ($\mathcal{F}_C$) in the network can be estimated by:

$$\begin{aligned}\mathbb{E}(\mathcal{F}_C) &= \sum_{i=1}^{N} P_i = \sum_{I \subseteq V} P_I \sum_{i=1}^{N} P_i^I = \sum_{i=1}^{N} \sum_{I \subseteq V} P_I P_i^I \\ &= \sum_{i=1}^{N} \lim_{n_a \to \infty} (V_V)_i + \|I\| = \|I\| + \lim_{n_a \to \infty} \sum_{i=1}^{N} V_V^i \\ &= \|I\| + \lim_{n_a \to \infty} \sum_{1 \leq i < j \leq N} V_E^{ij}, \end{aligned} \tag{3}$$

where N is the total number of nodes in a network, I is the set of initial attackers, $\|\cdot\|$ returns the set cardinality, P_i is the

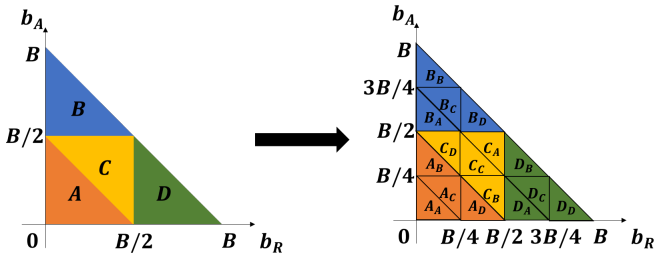


Fig. 1. Generation of self-similar fractals to reduce solution search space in edge addition and removal budgets, (b_A, b_R).

probability of node i being compromised, P_I is the probability of I as the set of initial attackers, P_i^I is the conditional probability of i being compromised if I is the set of initial attackers, V_V^i is the vulnerability of node i and V_E^{ij} is the vulnerability of an edge between nodes i and j.

Now we describe the network topology adaptations procedures as follows. Given $G = (V, E)$ and (b_A, b_R), we first calculate R_E of G and remove the top b_R edges according to R_E. This process results in an intermediate adapted graph, G^*, as in Algorithm 1. Then, we add a minimum number of edges to restore the connectivity of G^* and denote the adapted graph by G'. Next, we calculate R_V of G' and add at most b_A edges according to R_V. Unlike the removal process, the addition process between disconnected nodes needs to meet a distance constraint k. We provided the detail of the proposed VREN algorithm in Algorithm 1.

We consider Random Adaptation (RA) as a baseline alternative adaptation if an adaptation algorithm does not use VREN. Given fixed budget pair (b_A, b_R), RA would first remove b_R edges at random and then add b_A edges at random.

B. Fractal-based Solution Search (FSS)

In this section, we discuss the *Fractal-based Solution Search* algorithm, namely *FSS*. Recall that in Eq. (1), the objective function is defined based on the differences of the rewards from the original network G and the adapted network G'. We rewrite the objective function Eq. (1) by:

$$\arg\max_{b_A, b_R} f(G') - f(G) \quad \text{where} \quad G' = \text{VREN}(G, b_A, b_R) \quad (4)$$
$$s.t. \quad 0 \le b_A + b_R \le B,$$

where $f : G \mapsto \mathcal{S}_G(G) - \mathcal{F}_C(G)$, b_A is the addition budget, b_R is the removal budget, VREN follows Algorithm 1, and B is the upper bound of the total adaptation budget.

Since Eq. (4) above is an optimization problem defined within the triangle area $\{(b_A, b_R) \mid 0 \le b_A + b_R \le B\}$, we consider the fractal structure of this area in order to develop an efficient search space for the DRL agent to quickly identify an optimal solution. To be specific, we divide the triangle as described in Fig. 1. Since it is a self-similar fractal, we can obtain four self-similar parts for each subdivision, namely A, B, C, and D. This allows us to evaluate these divisions by using the centroid to represent each division and the nearest integer pair (b_A, b_R) from the centroid for evaluating Eq. (4).

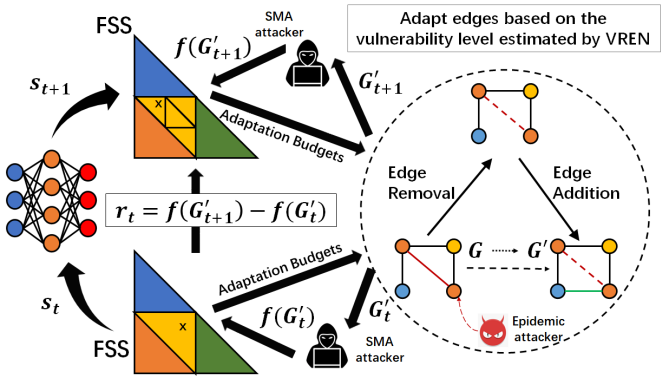


Fig. 2. The overall architecture of the proposed DREVAN: The color of each node refers to a different software package installed in it.

For each subdivision, division C always has the invariant centroid. Considering b_A and b_R as integers, it is sufficient to iterate the subdivision until the standard triangle (with leg length less than 1) is obtained.

To clearly show the enhanced efficiency in DRL by this FSS, we conducted comparative performance analysis of DRL schemes with FSS and conventional linear search (LS) as a baseline model in Section VII-A. LS starts searching an optimal budget from the origin in the triangular area and keeps increasing one budget unit per step until finding an optimal budget (b_A, b_R).

C. DRL-based Budget Adaptation

Here we describe how the DRL agent makes decisions to add or remove edges when it is given the total budget of edge adaptations following the learning stage of the DRL agent. For this paper to be self-contained, we describe the basic steps of DRL agents to learn and make decisions as follows.

For the DRL agent to learn a policy following the principle of RL, we use the mathematical framework of RL following a Markov Decision Process (MDP), consisting of (1) a set of states $\mathcal{S} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_t, \ldots \mathbf{s}_T\}$ with a distribution of starting states $\text{p}(\mathbf{s}_1)$ where $\mathbf{s}_t$ refers to a set of states s_t at time t; (2) a set of actions, $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_t, \ldots \mathbf{a}_T\}$ where $\mathbf{a}_t$ is a set of actions a_t available at time t; (3) a transition probability, $\mathcal{T}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$, from state $\mathbf{s}_t$ to state $\mathbf{s}_{t+1}$ via $\mathbf{a}_t$; (4) an immediate reward function, $\mathcal{R}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$; (5) a discount factor $\gamma \in [0, 1]$ where lower γ counts immediate rewards more; and (6) a policy π mapping from states to a probability distribution of actions, $\pi : \mathcal{S} \to (\mathcal{A} = a \mid \mathcal{S})$. Given the MDP has an episode with length T, the sequence of states, actions, and rewards in an episode are the components of a trajectory of the policy. The accumulated reward based on the trajectory of the policy results in $R = \sum_{t=0}^{T-1} \gamma^t r_{t+1}$ [22]. Therefore, RL aims to identify an optimal policy π^* that can generate the maximum expected reward from all states [2]: $\pi^* = \arg\max_{\pi} \mathbb{E}[R \mid \pi]$.

Now we describe how the key components of the MDP, including states, actions, and rewards, are formulated to identify

an optimal setting of budget constraints. We describe these under our proposed FSS and the baseline LS as follows:

- **States**: There will be in total $\lceil \log_2 B \rceil$ steps if we use FSS and do the subdivision for each step, where B is the total budget. In each step t of RL, state s_t is represented by:

$$s_t = (b_A^t, b_R^t, G_t'), \tag{5}$$

  where b_A^t and b_R^t refer to the addition budget and removal budget identified through the division evaluation at time t, respectively, and G_t' is an adapted network at time t. Under LS, B steps will be taken to guarantee the availability of optimal trajectories in each episode.
- **Actions**: Under FSS, the action $\mathbf{a}_t$ at step t is determined by the division ID for a next subdivision, and given by:

$$\mathbf{a}_t = \{A, B, C, D\}, \quad 1 \le t \le \lceil \log_2 B \rceil.$$

  On the other hand, under LS, $\mathbf{a}_t$ is given by:

$$\mathbf{a}_t = \{\texttt{stop}, \texttt{add}, \texttt{remove}\}, \quad 1 \le t \le B,$$

  where, given the total budget for edge adaptations B, `stop` refers to terminating the episode at a current step t, and `add` and `remove` increment b_A and b_R by 1, respectively.
- **Rewards**: The same reward function is used for both FSS and LS. We define the reward function at step t, denoted by $\mathcal{R}(s_t, a_t, s_{t+1})$, based on the difference between the evaluation functions of two consecutive states by:

$$\mathcal{R}(s_t, a_t, s_{t+1}) = f(G_{t+1}') - f(G_t'), \tag{6}$$

  where $f : G \mapsto \mathcal{S}_G(G) - \mathcal{F}_C(G)$ is defined in Eq. (1).

In Fig. 2, we summarized the overall process of the proposed DREVAN described in this section.

VI. Experimental Setup

We first deploy a given network and adapt the network once based on our proposed network adaptation algorithm. DRL agents would adapt the network by addressing the optimization problem in Eq. (1). After then, epidemic attacks are applied to the network, which is evaluated in terms of system security and network connectivity. We assume attackers can perform *state manipulation attacks* (SMAs) in Section IV-C, with probability P_s. We assume DRL agents can detect such an attack with detection rate D_r. If the attack is detected, DRL agents would use the system state of the previous step; otherwise, it uses the falsified state. For the attack simulations to identify the expected vulnerabilities of nodes and edges (see Algorithm 1), we used $n_a = 500$ for all our experiments. The original network is initialized by an Erdös–Rényi (ER) model [7], representing a random graph, with the total number of nodes $N = 200$ and its connection probability, $p = 0.05$. All demonstrated results are based on the averages collected from $n_r = 200$ simulation runs. We evaluate all three DRL-based schemes (i.e., DQN, Double DQN, and Dueling DQN), after $n_e = 1,000$ training episodes. We summarized the key design parameters, their meanings, and default values used in Table I.

We used the following **metrics** for our experiments:

- **Converged reward** ($\mathcal{CR}$): This is the converged reward identified by a DRL agent aiming to solve the optimization problem defined in Eq. (1).
- **Size of the giant component** ($\mathcal{S}_G$): This captures the extent of network connectivity composed of non-compromised, active nodes in a network, and measured by $\mathcal{S}_G = \frac{N_G}{N}$, where N is the total number of nodes in the network and N_G is the number of nodes in the giant component. Higher $\mathcal{S}_G$ is more desirable.
- **Fraction of compromised nodes** ($\mathcal{F}_C$): This measures the fraction of the number of compromised nodes (both detected and undetected) over the total number of nodes N in the network due to epidemic attacks. Lower $\mathcal{F}_C$ is more desirable.

To emphasize the necessity of each component (VREN/FSS) in our model, we consider different variants of DQN algorithms and baselines for **comparative performance analysis**:

- **DQN with DREVAN** [17]: This utilizes neural networks parameterized by $\boldsymbol{\theta}$ to represent an action-value function (i.e., Q function) and assumes that the agent can fully observe the environment. This uses VREN and FSS.
- **Double DQN with DREVAN** [24]: This has two networks, namely an online network and a target network. By updating parameters only based on the periodically updated target network, the agent could perform more stably than DQN. This uses VREN and FSS.
- **Dueling DQN with DREVAN** [26]: This also has both an online network and a target network and utilizes the dueling network architecture decoupling value and advantage. This is known for its outperformance over DQN and DDQN in Atari games. This uses VREN and FSS.
- **DQN with VREN**: This is the same as DQN, but it does not use FSS and instead uses linear search (LS).
- **DQN with FSS**: This is the same as DQN, but it does not use VREN and instead uses random adaptation (RA).
- **DQN**: This is a baseline model that does not use DREVAN (i.e., VREN and FSS). That is, this uses LS and RA.
- **Greedy**: This uses FSS to make greedy choices by looking one-step ahead at each step by choosing the subdivision returning a maximum reward.
- **Random**: This is considered as a baseline model where an agent first selects a removal budget b_R at random and then selects an addition budget b_A at random.
- **Optimal**: This identifies adaptation budgets corresponding to a maximum converged reward provided by VREN.

VII. Numerical Results & Analysis

A. Learning Performance and Complexity of DREVAN

In this section, we discuss the learning performance and complexities of our proposed schemes. Since we observed that Double DQN and Dueling DQN perform similar to DQN, we focus on analyzing the algorithmic complexity of DQN with DREVAN, DQN with FSS, DQN with VREN, and DQN and the speed of reaching an optimal solution in this section. As shown in Table II, DQN with DREVAN and DQN with FSS have less complexity than DQN with VREN and DQN since

TABLE I
KEY DESIGN PARAMETERS, MEANINGS, AND DEFAULT VALUES

Param.	Meaning	Value
n_a	Attack simulation times	500
n_r	Number of simulation runs	200
n_e	Training episodes of DRL-based schemes	1000
N	Total number of nodes in a network	200
k	Upper hop bound for edge addition	3
γ^*	Intrusion detection probability	0.9
P_{fn}, P_{fp}	False negative or positive probability	0.1, 0.05
$\mathbf{x}$	Degree of software vulnerability	0.5
p	Connection probability between pairs of nodes in an ER network	0.05
l	Number of software packages available	5
P_a	Fraction of initial attackers in a network	0.3
B	Upper bound of the total adaptation budget	500
P_s	Probability of state manipulation attacks	0.3
D_r	Detection rate of state manipulation attacks	0.99

TABLE II
ASYMPTOTIC ANALYSIS OF THE COMPARED SCHEMES

Scheme	Complexity
DQN with DREVAN	$O(n_e \times \lceil \log_2 B \rceil \times T_{\text{train}} \times n_a)$
DQN with FSS	$O(n_e \times \lceil \log_2 B \rceil \times T_{\text{train}} \times n_a)$
DQN with VREN	$O(n_e \times B \times T_{\text{train}} \times n_a)$
DQN	$O(n_e \times B \times T_{\text{train}} \times n_a)$
Greedy	$O(\lceil \log_2 B \rceil \times n_a)$
Random	$O(n_a)$
Optimal	$O(B^2 \times n_a)$

the proposed fractal environment could significantly reduce the number of trajectories per training episode. Here, n_e is the training episode, T_{train} is the training time per episode, B is the upper bound of total adaptation budget, and n_a is the attack simulation times. Since Greedy, Random and Optimal are rule-based algorithms that do not have the training phase, their complexities are only based on B and n_a. As shown in Table II, Optimal has higher complexity than Greedy while Random is the most efficient algorithm among all. Note that the assumption in Optimal is that an agent knows all the rewards in advance, thus it will not be realistic in practice to have all the rewards available as prior knowledge.

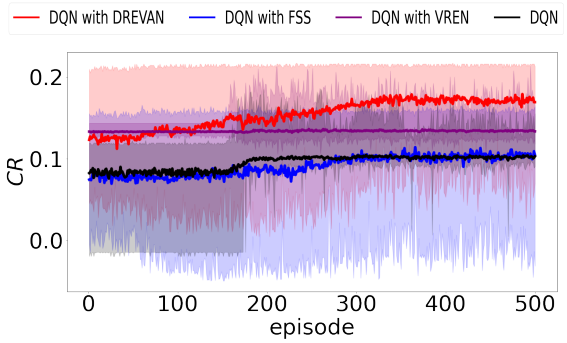


Fig. 3. Converged reward with respect to training episodes.

Now we discuss how quickly each DRL algorithm (i.e., DQN with DREVAN, DQN with FSS, DQN with VREN, or DQN) identifies a best solution achievable. Fig. 3 shows the learning curve of the four DQN-based schemes with respect to training episodes under the default settings in Table I. We observed that DQN with DREVAN performs the best among all. DQN with FSS can only learn a sub-optimal policy since the underlying RA performs much worse than VREN. DQN with VREN and DQN cannot learn well with LS. This clearly demonstrates that FSS provides a critical environment for a DRL agent to efficiently identify the best solution achievable.

B. Sensitivity Analysis

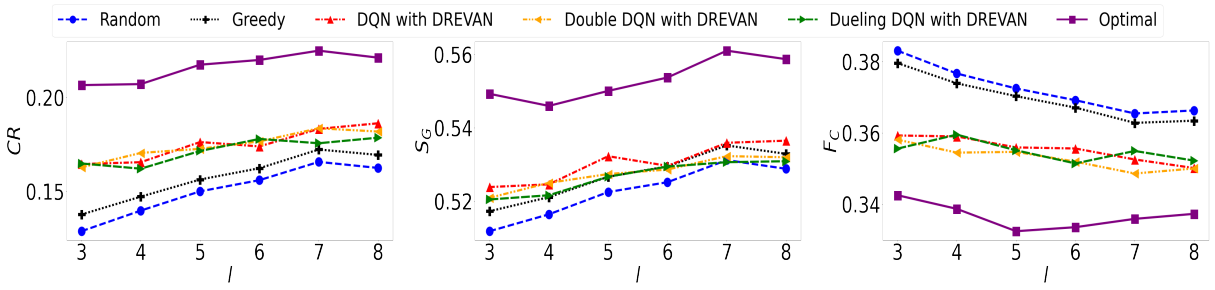
In this section, we conducted in-depth comparative performance analysis of three types of DQN (i.e., DQN, Double DQN, and Dueling with DREVAN) and three baseline and existing counterparts (i.e., Greedy, Random, and Optimal) under varying a wide range of the number of available software packages l, the upper bound of the total adaptation budget B, probability of state manipulation attacks P_s, and detection rate of state manipulation attacks D_r. Since we already proved that DQN with DREVAN outperforms in Section VII-A, we did not include non-DREVAN-based DQN algorithms for comparative performance analysis in this section. Rather, to investigate further performance analysis of DREVAN-based DQN algorithms, we include two additional, advanced DQN algorithms, Double DQN and Dueling DQN.

1) **Effect of Varying the Number of Software Packages:** Fig. 4 shows the effect of varying the number of software packages available (l) on the performance of the six schemes with respect to the three metrics under the ER network model. We observed that increasing l increases converged reward ($\mathcal{CR}$) and size of the giant component ($\mathcal{S}_G$) while decreasing the fraction of compromised nodes ($\mathcal{F}_C$). Based on the concept of N-version programming, the number of software packages (l) here refers to the number of versions being implemented for the same piece of software. Hence, as l increases, the level of software diversity increases, resulting in a decrease of the percentage of nodes being compromised due to attacks, an increase of the network connectivity because less nodes are being compromised. The overall performance order in terms of all three metrics is summarized as: Optimal $\geq$ DQN $\approx$ Double DQN $\approx$ Dueling DQN $\geq$ Greedy $\geq$ Random.

2) **Effect of Varying the Amount of Adaptation Budget:** Fig. 5 shows the effect of varying the upper bound of the total adaptation budget (B) on the performance of the six schemes in terms of the three metrics under the ER network model. We observed that higher B increases $\mathcal{CR}$ while decreasing $\mathcal{F}_C$. Notice that higher B does not necessarily improve $\mathcal{S}_G$ since all schemes adapt the network based on $\mathcal{CR}$ where edge adaptations are based on the ranks of expected vulnerabilities. The overall performance order with respect to $\mathcal{CR}$, $\mathcal{F}_C$ and $\mathcal{S}_G$ is summarized as: Optimal $\geq$ DQN with DREVAN $\approx$ Double DQN with DREVAN $\approx$ Dueling DQN with DREVAN $\geq$ Greedy $\geq$ Random.

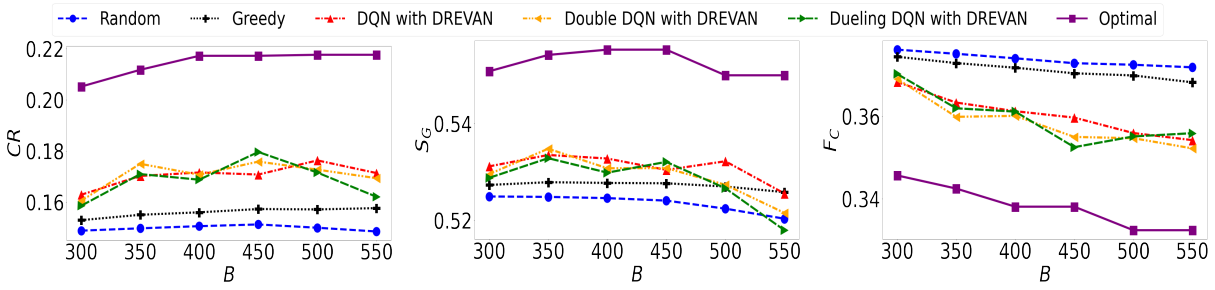
Dueling DQN with DREVAN is more sensitive to B than DQN with DREVAN and Double DQN with DREVAN as it performs significantly worse under a higher budget B. This is due to the dueling structure used in Dueling DQN, which is in general much harder to be trained than the other two DQN algorithms with a larger episode length for higher B.

3) **Effect of Varying the Strength of State Manipulation Attacks:** Fig. 6 shows the effect of varying the probability of SMAs (P_s) on the performance of the six schemes in terms



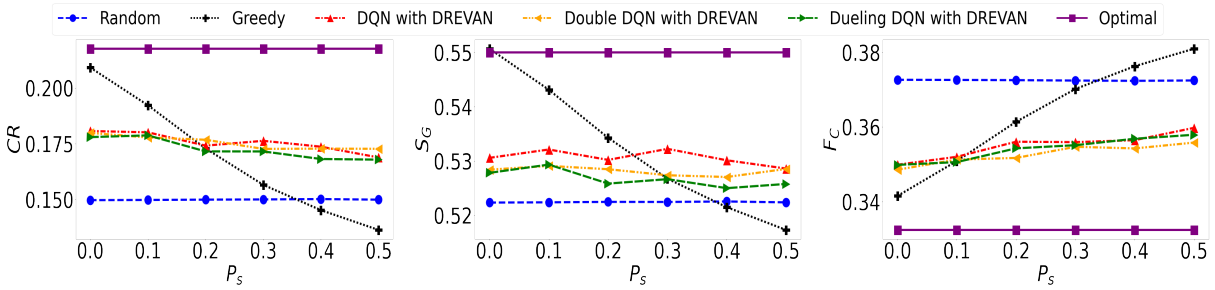
(a) Converged reward ($C\mathcal{R}$) (b) Size of the giant component ($\mathcal{S}_G$) (c) Fraction of compromised nodes ($\mathcal{F}_C$)

Fig. 4. Effect of varying the number of software packages available (l) under an ER network.



(a) Converged reward ($C\mathcal{R}$) (b) Size of the giant component ($\mathcal{S}_G$) (c) Fraction of compromised nodes ($\mathcal{F}_C$)

Fig. 5. Effect of varying the upper bound of the total adaptation budget (B) under an ER network.



(a) Converged reward ($C\mathcal{R}$) (b) Size of the giant component ($\mathcal{S}_G$) (c) Fraction of compromised nodes ($\mathcal{F}_C$)

Fig. 6. Effect of varying probability of state manipulation attacks (P_s) under an ER network.

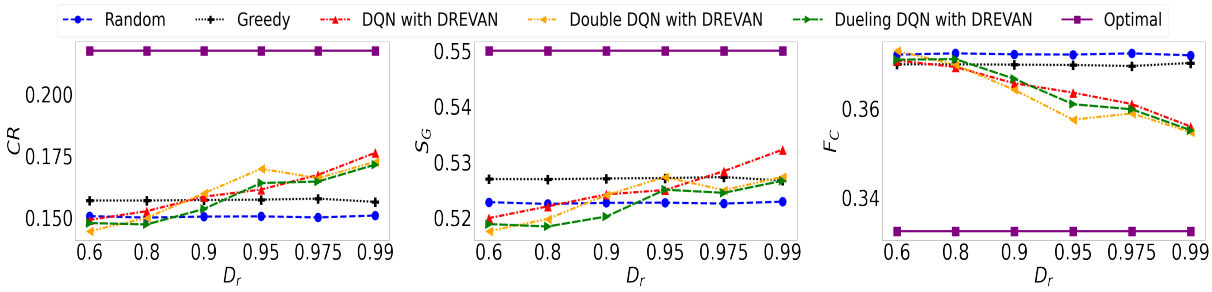
of the three metrics under the ER network model. We observe that higher P_s brings lower $C\mathcal{R}$ and $\mathcal{S}_G$ while introducing more $\mathcal{F}_C$. Since SMAs can be only performed when rewards are used, their severity only affects the performance of the Greedy and DRL-based schemes while no effect is introduced to Random and Optimal. Greedy is more sensitive to P_s than DRL-based schemes as it performs better than three DRL algorithms with smaller P_s but significantly worse than them with higher P_s. Again, this is due to the nature of Greedy, which looks ahead at each step for the reward of a next action.

4) **Effect of Varying the Detection Rate of State Manipulation Attacks (SMAs):** Fig. 7 shows the effect of varying detection rate of SMAs (D_r) on the performance of the six schemes in terms of the three metrics under the ER network model. We observed that higher D_r increases $C\mathcal{R}$ and $\mathcal{S}_G$ while lowering $\mathcal{F}_C$. Since the detection mechanism is only implemented under DRL-based schemes, the other three schemes have no sensitivity against D_r. The results show the outperformance of DRL-based schemes only with higher D_r. We can observe that overall DQN with DREVAN performs slightly better than Double DQN with DREVAN and Dueling DQN with DREVAN.

VIII. Conclusion

From this study, we obtained the following **key findings**:

- Our proposed DREVAN uses a fractal-based environment (FSS) that can significantly reduce the training complexity of our DRL algorithms. This consequently improves the speed of convergence to an optimal solution with a higher converged reward.
- Our proposed DREVAN uses a vulnerability-aware ranking algorithm (i.e., VREN) to strategically adapt edges for the network to be reconfigured effectively and efficiently.
- Our DREVAN using three different types of Deep Q-learning algorithms (i.e., DQN with DREVAN, Double DQN with DREVAN, and Dueling DQN with DREVAN) showed higher



(a) Converged reward ($\mathcal{CR}$) (b) Size of the giant component ($\mathcal{S}_G$) (c) Fraction of compromised nodes ($\mathcal{F}_C$)

Fig. 7. Effect of varying detection rate of state manipulation attacks (D_r) under an ER network.

performance than the counterpart and baseline schemes particularly in minimizing system vulnerability while maintaining comparable or better network connectivity under a wide range of key design parameter values.

ACKNOWLEDGEMENT

This work is partly supported by the Army Research Office under Grant Contract Number W91NF-20-2-0140. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

REFERENCES

[1] S. Achleitner, T. L. Porta, P. McDaniel, S. Sugrim, S. V. Krishnamurthy, and R. Chadha, "Deceiving network reconnaissance using SDN-based virtual topologies," *IEEE Transactions on Network and Service Management*, vol. 14, pp. 1098–1112, Dec. 2017.

[2] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[3] D. Borbor, L. Wang, S. Jajodia, and A. Singhal, "Optimizing the network diversity to improve the resilience of networks against unknown attacks," *Computer Communications*, 2019.

[4] X. Chai, Y. Wang, C. Yan, Y. Zhao, W. Chen, and X. Wang, "DQ-MOTAG: Deep reinforcement learning-based moving target defense against DDoS attacks," in *2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC)*. IEEE, 2020, pp. 375–379.

[5] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song, "Adversarial attack on graph structured data," *Proceedings of Machine Learning Research (PMLR)*, vol. 80, pp. 1115–1124, 2018.

[6] V.-A. Darvariu, S. Hailes, and M. Musolesi, "Improving the robustness of graphs through reinforcement learning and graph neural networks," *arXiv preprint arXiv:2001.11279*, 2020.

[7] P. Erdös and A. Rényi, "On the evolution of random graphs," in *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 1960, pp. 17–61.

[8] "Common Vulnerabilities and Exposures (CVE)," Forum of Incident Response and Security Teams. [Online]. Available: https://cve.mitre.org/

[9] K. J. Hole, "Diversity reduces the impact of malware," *IEEE Security Privacy*, vol. 13, no. 3, pp. 48–54, May 2015.

[10] J. B. Hong and D. S. Kim, "Assessing the effectiveness of moving target defenses using security models," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 2, pp. 163–177, Mar. 2016.

[11] J. B. Hong, S. Yoon, H. Lim, and D. S. Kim, "Optimal network reconfiguration for software defined networks using shuffle-based online mtd," in *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2017, pp. 234–243.

[12] C. Huang, S. Zhu, Q. Guan, and Y. He, "A software assignment algorithm for minimizing worm damage in networked systems," *Journal of Information Security and Applications*, vol. 35, pp. 55–67, 2017.

[13] T. R. Jensen and B. Toft, *Graph coloring problems*. John Wiley & Sons, 2011, vol. 39.

[14] T. Kaur and J. Baek, "A strategic deployment and cluster-header selection for wireless sensor networks," *IEEE Transactions on Consumer Electronics*, vol. 55, no. 4, pp. 1890–1897, 2009.

[15] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[16] S. Mavoungou, G. Kaddoum, M. Taha, and G. Matar, "Survey on threats and attacks on mobile networks," *IEEE Access*, vol. 4, pp. 4543–4572, 2016.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[18] "Common Vulnerability Scoring System (CVSS)," National Vulnerability Database. [Online]. Available: https://www.first.org/cvss/

[19] M. Newman, *Networks: An Introduction*. Oxford Univ. Press, 2010.

[20] A. J. O'Donnell and H. Sethu, "On achieving software diversity for improved network security using distributed coloring algorithms," in *Proceedings of the 11th ACM Conference on Computer and Communications Security*. ACM, 2004, pp. 121–131.

[21] J. P. Rohrer, A. Jabbar, and J. P. Sterbenz, "Path diversification for future internet end-to-end resilience and survivability," *Telecommunication Systems*, vol. 56, no. 1, pp. 49–67, 2014.

[22] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[23] O. Temizkan, S. Park, and C. Saydam, "Software diversity for improved network security: optimal distribution of software-based shared vulnerabilities," *Info. Systems Research*, vol. 28, no. 4, pp. 828–849, 2017.

[24] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[25] Z. Wan, Y. Mahajan, B. W. Kang, T. J. Moore, and J.-H. Cho, "A survey on centrality metrics and their implications in network resilience," 2020.

[26] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *Int'l Conf. Machine Learning*. PMLR, 2016, pp. 1995–2003.

[27] Y. Yang, S. Zhu, and G. Cao, "Improving sensor network immunity under worm attacks: A software diversity approach," *Ad Hoc Networks*, vol. 47, no. Supplement C, pp. 26–40, 2016.

[28] M. Zhang, L. Wang, S. Jajodia, A. Singhal, and M. Albanese, "Network diversity: a security metric for evaluating the resilience of networks against zero-day attacks," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 5, pp. 1071–1086, 2016.

[29] Q. Zhang, J.-H. Cho, T. J. Moore, and R. Chen, "Vulnerability-aware resilient networks: Software diversity-based network adaptation," *IEEE Transactions on Network and Service Management*, 2020, early access.

[30] Q. Zhang, A. Z. Mohammed, Z. Wan, J. Cho, and T. J. Moore, "Diversity-by-design for dependable and secure cyber-physical systems: A survey," *CoRR*, vol. abs/2007.08688, 2020. [Online]. Available: https://arxiv.org/abs/2007.08688

[31] T. Zhang, C. Xu, B. Zhang, X. Kuang, Y. Wang, S. Yang, and G.-M. Muntean, "Dq-rm: Deep reinforcement learning-based route mutation scheme for multimedia services," in *2020 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE, 2020, pp. 291–296.