Stealthy Edge Decoy Attacks Against Dynamic Channel Assignment in Wireless Networks

Ahmed H. Anwar[1], Janiece Kelly[2], George Atia[1], and Mina Guirguis[2]

[1]Department of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816
[2]Department of Computer Science, Texas State University, San Marcos, TX 78666
a.h.anwar@knights.ucf.edu, jek44@txstate.edu, george.atia@ucf.edu, msg@txstate.edu

***Abstract*—Dynamic channel assignment algorithms allow wireless nodes to select their communication channels based on the state of the network to reduce interference between the nodes and improve the overall network performance. They have been shown to outperform static channel assignment policies and thus are playing a critical role in Software Defined Networks (SDNs). In this paper, we examine the security of dynamic channel assignment algorithms against stealthy decoy attacks in which the attacker induces channel conflicts on a selective set of edges. When an edge is attacked, the victim nodes search for and switch to a different channel to use, possibly causing their neighbors to switch as well. As the effect of the attack propagates through the network, the performance of the network is severely degraded. The decisions of *which* edges to attack and *what* channels to use in creating conflicts are the solutions of various Markov Decision Processes (MDP) problems in which the attacker's goal is to maximize the number of conflicts in the network subject to an attack cost. We apply approximate policy iteration methods to identify suboptimal, albeit efficient, attack policies. Our results show that our exposed attack policy outperforms other attack policies and adapts to the cost of the attack.**

I. Introduction

Wireless communication systems face two major challenges. On one hand, there is a massive growth in the amount of traffic that is transmitted over wireless links due to the proliferation of wireless-enabled mobile devices along with the emergence of many new applications that are bandwidth-thirsty (e.g., streaming). On the other hand, the available bandwidth and frequency allocations are not growing as fast since they are tightly regulated by the FCC. This has prompted much research in areas such as Software Defined Networks [1] and femtocells [2] to optimize the usage of the frequency spectrum among devices. Through empowering wireless nodes to select appropriate frequency channels and adapt their choice (as well as their transmission ranges), the overall performance can be significantly improved.

To enable communication between multiple nodes in a network, neighboring nodes must use the same frequency channel to communicate. Due to the limited availability of wireless channels (e.g., only 11 channels are allowed in the US over the 802.11 specifications, [3]), nodes must reuse these channels over their interfaces. Ideally, the channel reuse policy should avoid assigning overlapping channels to different interfaces for a given node to minimize interference. In particular, there are two types of interference between channels:

1) Co-channel interference (CCI): CCI occurs if two radio nodes (interfaces) are within each other's interference radius and use the same frequency channel. This causes extra delay due to medium contention since the transmitting node must wait until the medium is idle before it can try to send its packets.
2) Adjacent channel interference (ACI): ACI occurs if two radio nodes (interfaces) are within each other's interference radius and use two adjacent frequency channels. This type of interference introduces bit errors as the signal of each node would be perceived as noise to the other adjacent node. Thus, channel separation should be used to prevent ACI.

Wireless interference (ACI and CCI) causes multiple retransmissions of packets with an even higher associated cost of retransmission timeouts that severely reduces the overall throughput. This has triggered a lot of work on smart channel assignment methods that can reduce the overall interference and improve the network performance. In these methods, the frequency channels are assigned to nodes in a non-overlapping non-interfering manner. Frequency channel assignment methods can be either static or dynamic. Static channel assignment (SCA) assigns the frequency channels to each radio node prior to network deployment. SCA suits networks with small variations and lower degree of mobility. In dynamic channel assignment (DCA), nodes dynamically select and adapt their selection over time based on the network state and sensed assignment of their neighbors. DCA requires nodes to be able to switch channels efficiently. Software Defined networks (SDNs) [1] are capable of handling such tasks at higher layers. SDNs can carry channel switching to be performed in software, hence, they are suitable for interference-aware dynamic channel assignments.

The dynamic nature of switching between channels in DCA methods opens a back door for stealthy attacks to be mounted. In this paper, we identify classes of stealthy attacks that exploit the self-resolving nature of such methods making the network perpetually switching over the channels used. In a typical 802.11 network, a node should perform an availability check before switching to a new frequency channel which may take up to 20 msec, [4]. Thus, we consider a stealthy attacker that can induce interference on a particular link in the network

978-1-5090-0073-9/15/$31.00 ©2015 IEEE

(by simply transmitting on the same or an adjacent channel) triggering the end nodes of this link to switch to another channel, hence, they cause their neighbors to potentially switch their channels as well. With the proper choice of the victim edge and which channel to induce, the attacker can exploit the topology of the network to cause a cascading switching behavior that propagates through the network causing inefficient operation and instability in the network. To identify optimal and suboptimal attack policies, the attacker solves Markov Decision Processes (MDP) problems in which the decision to attack a link is based on the expected damage inflicted (in terms of interference) and on a notion of attack cost.

The rest of the paper is organized as follows. Section II presents related work. Our system model and proposed policy are described in Section III. Results are presented in Section IV, and we conclude in Section V.

II. Related Work

Channel Assignment (CA) is an important problem in designing wireless networks. In [5], a CA algorithm in a WLAN was proposed based on minimizing the total interference between access points (APs). The objective is to maximize the signal-to-interference ratio at the user level. CA can be either static or dynamic. Static CA is generally considered as a graph coloring problem [6]. On the other hand, the authors in [3] designed a dynamic CA algorithm for IEEE 802.11 wireless networks that achieves minimum channel interference between all nodes resulting in higher throughput.

Despite a large body of literature on CA algorithms of multi-interface multi-channel (MIMC) wireless networks, the study of security of CA protocols continues to receive more attention, especially with the proliferation of cyber-physical systems. The existing literature focused primarily on node authentication [7], [8] and intrusion detection [9], [10] in wireless mesh networks. In [11], the authors investigated threats that can potentially break down the CA protocol of MIMC networks reducing the network overall throughput. Security vulnerabilities in the 802.11 standard were identified in [12]. It was shown that a denial of service (DOS) effect of up to one minute can be achieved with a single message by simply forging the channel switch information [12]. Radio interference jamming attacks on wireless networks were investigated in [13]. Moreover, several intelligent jamming attacks were studied and analyzed in [14].

In dynamic CA, each radio interface inherits channel switching capabilities, which is a source of threat as it can be exploited by a stealthy attacker. Since channel switching can be modeled as a discrete-time process, knowing the structure and the state of a certain network, an attacker can design and optimize an attack policy to inflict the maximum damage on the network. In this paper, we formulate the attack problem as an optimization problem from the attacker's standpoint to solve for an optimal attack policy. We identify suboptimal attack policies obtained through an approximate solution to a Markov Decision Process (MDP) problem [15], [16], [17].

III. System Model

We consider a network graph $G(\mathbb{V}, \mathbb{E})$, where $\mathbb{V} = \{v_i\}, i = 1, \ldots, |\mathbb{V}|$, is the set of vertices with cardinality $|\mathbb{V}|$, and $\mathbb{E} = \{e_{ij}\}, i, j \in \mathbb{V}$, is the set of edges. An edge e_{ij} represents a communication link between vertices i and j over a certain frequency channel $c_{e_{ij}} \in \mathbb{C}$. Herein, we consider $\mathbb{C} = \{1, 2, 3, ..., 11\}$, the set of usable frequency channels in the United States for the 2.4 GHz band in the 802.11n specifications [18].

We assume that each node has multi-radio interfaces. For successful decoding, a node should use non-interfering frequencies on each radio interface. Hence, the channel assignment problem can be considered as a graph coloring problem to obtain non-interfering edges. The goal is to assign colors, i.e, "frequency channels", to the edges of the graph so that no two adjacent edges have the same color. Generally, graph coloring is an NP hard problem, however, a greedy algorithm can be used to perform the channel assignment.

A. Channel Assignment

For every edge e_{ij}, we define a set $\mathbf{B}_{e_{ij}}$ that contains its neighboring edges. Neighboring edges are defined as edges connected to one of its end nodes. Hence,

$$\mathbf{B}_{e_{ij}} = \{e_{iu} | i, u \in \mathbb{V}\} \bigcup \{e_{vj} | v, j \in \mathbb{V}\}.$$

Whenever unambiguous, we henceforth use e without a suffix to simplify notation. We define a set $\mathbf{A}_e$ for every edge e as the set of available channels that may be assigned to that edge by defining the interference set $\mathbf{I}_m$ for each adjacent edge m,

$$\mathbf{A}_e = \mathbb{C} \setminus \bigcup_{m \in \mathbf{B}_e} \mathbf{I}_m, \; m, e \in \mathbb{E}.$$

The interference set for an edge e, is defined as $\mathbf{I}_e = \{\max(c_e - x, 1), \ldots, c_e, \ldots, \min(c_e + x, 11)\}$, where x is a channel separation constant. For example, if $x = 2$ and edge e is assigned channel 3, then none of its neighboring edges can be assigned channels $1, 2, 3, 4, 5$. Otherwise, we say that edge e is in conflict. After computing the above sets for each edge, we build a histogram that counts how many times a given channel $c \in \mathbb{C}$ appears in interference sets, $\mathbf{I}_e, \forall e \in \mathbb{E}$. This gives an indication of the most and least interfering channels.

Assume that a new channel is to be assigned to edge e based on the channel assignment of the remaining edges. If $\mathbf{A}_e$ is non-empty, i.e., $\mathbf{A}_e \neq \emptyset$, then e is assigned a channel c_e drawn randomly from the set $\mathbf{A}_e$. On the other hand, if $\mathbf{A}_e = \emptyset$, then e is assigned the *least interfering channel* according to the histogram as this channel will cause minimal interference in the network. We use a graph coloring greedy algorithm to assign a channel to each edge. This greedy algorithm is summarized in the table below.

We define the degree $\zeta_{e_{ij}}$ of an edge e_{ij} as the number of edges connected to its end nodes, i.e., $\zeta_{e_{ij}} = \delta_i + \delta_j - 2$, where δ_v is the node degree for node v.

Algorithm 1 Channel Assignment greedy Graph Coloring

```
procedure SORT:($\mathbb{E}$)                    ▷ according to edge degree
    for $\forall e \in \mathbb{E}$ do
        Find: $\mathbf{A}_e, \mathbf{B}_e$ and $\mathbf{I}_e$.
        Update: Channel Histogram.
        if $\mathbf{A}_e \neq \emptyset$ then
            Assign $e$ a channel $c_e = c, c \in \mathbf{A}_e$, drawn randomly.
        else
            Choose $c_e$ as the least interfering channel.
        end if
    end for
end procedure
```

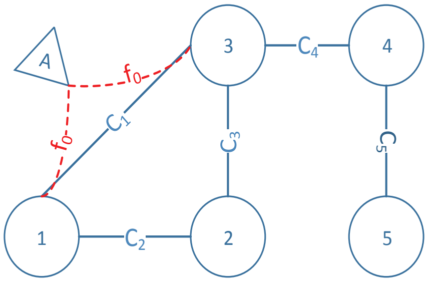

Fig. 1: Attacker is attacking the edges of the wireless network.

B. Stealthy Decoy Attacks

In order to assess the vulnerability of the network to attacks, we assume that a stealthy attacker attempts to degrade the network performance by attacking its communication links. The attacker approaches a selected edge and transmits a signal at a certain frequency that interferes with the edge's assigned channel and/or its neighboring edges. Hence, both of the edge end nodes will experience high interference at the corresponding radio interfaces. As illustrated in Fig.1, the attacker launches an attack by traveling to a victim (edge), e_{ij}, between its end nodes i and j, then transmits an interfering signal on channel f_0. Depending on the attack channel f_0, the attack can cause either CCI or ACI.

In order to select the victim edge and the attacking frequency, we assume the attacker is aware of the topology and the channels assigned to the different edges. Therefore, an attack action, u, consists of the selected victim edge e and the attacking frequency f_0. However, the action space is limited by the distance (hop counts) between the current location of the attacker at edge ℓ and the victim edge e. Note that the attacker may also choose not to attack at a given time instant to remain stealthy. To track the successive actions of the attacker, let b denote the number of successive no-attack actions. b is a limiting factor that determines the set of edges that are possible to be attacked. In particular,

- If $b = 0$, the victim edge e is one of the immediate neighbors of ℓ, i.e, $e \in \mathbf{B}_\ell$.
- If $b = 1$, the victim edge e is one of the neighbors of ℓ or the neighbors of neighbors of ℓ (i.e, 2 hops away). Generally, if $b > 0$, then the victim edges can be up to $b+1$ hops away.

To clarify, the attacker can reach further away victims after a sequence of no-attack actions, due to its power budget. In other words, we assume that the attacker can save power by choosing not to attack, hence, can attack far victims on future time steps. Considering the case where $b = 0$, after choosing its victim edge e, the attacker selects the attacking frequency f_0 such that f_0 may cause:

- CCI, if $f_0 = c_e$, i.e, jamming the channel of edge e.
- ACI, if $f_0 \in \bigcup \mathbf{I}_m \setminus \{c_e\},\ m \in \mathbf{B}_e \bigcup \{e\}$.

If f_0 causes both CCI and ACI, this simply means that the victim edge e was already causing ACI to its neighbors, i.e, it was already a conflict before it got attacked. The same explanation extends to the cases where $b > 0$ with the difference that the attacking set increases as explained earlier.

Launching an attack would cause conflicts due to the interference between the radio interfaces of the nodes, thereby forcing some nodes to switch their channels. Causing more conflicts leads to a more pronounced channel switching behavior[1]. Moreover, conflicts could cause more delay due to packet failure and retransmissions. Such conflicts are rewards from the attacker's viewpoint, but the cost is power transmitted by the attacker and higher risk of being caught. To maximize the total reward, the attacker aims to optimize its attacking policy. In this paper, we propose an attacking policy that maximizes the expected total reward for an attacker by formulating this problem as an MDP as explained next.

C. MDP Formulation

Let s_k denote the state of the network at time k. Specifically, $s_k = [\mathbf{c}_k, \ell_k, b_k]$, where $\mathbf{c}_k$ is a $1 \times |\mathbb{E}|$ vector representing the channels assigned to every edge in the network at time k, ℓ_k is the last attacked edge, and b_k denotes the number of successive no-attack actions at time k as defined earlier. Being in state s_k, the system evolves to the next state s_{k+1} with a transition probability $p(s_{k+1}|s_k, u_k)$ under action u_k taken by the attacker at time k, where $p(s_{k+1}|s_k, u_k)$ is uniform over all possible future states. The attacker gains a reward and incurs a cost for taking this action u_k. Let x_k^e be the number of conflicts at time k in the network with respect to edge $e \in \mathbb{E}$, and y_k the cost of action $u_k = [q, f_0]$. Hence,

$$x_k^e = |\{m | c_{e,k} \in \mathbf{I}_m, m \in \mathbf{B}_e\}|,\ e \in \mathbb{E}. \tag{1}$$

where $c_{e,k}$ is the channel assigned to link e at time k. In other words, x_k^e is the number of neighbors of e who have $c_{e,k}$ in their interference sets. The reward, $r(s_k, u_k, s_{k+1})$, is defined as the difference between the total number of conflicts and the cost of the attack,

$$r(s_k, u_k, s_{k+1}) = \sum_{e \in \mathbb{E}} x_{k+1}^e - y_k. \tag{2}$$

The Attack cost y_k is defined as,

$$y_k = \begin{cases} \infty & \text{if } \ d(\ell_k, q) > b_k + 1 \\ h \cdot d(\ell_k, q) + \zeta_q & \text{otherwise,} \end{cases} \tag{3}$$

[1] An attack may not cause the attacked edge to switch its channel since the system will select only one edge in conflict to resolve at a time.

where $d(\ell_k, q)$ is the distance between the current edge location ℓ_k and the victim edge q, and h is some constant capturing the cost per unit distance. The edge degree ζ_q is taken into consideration for cost computation since higher degree implies higher risk of being caught. Hence, the expected reward over all possible future states is

$$\bar{r}(s_k, u_k) = \sum_{s_{k+1}} p(s_{k+1}|s_k, u_k) \cdot r(s_k, u_k, s_{k+1}). \quad (4)$$

We consider a discounted cost MDP formulation, where the attacker aims to solve for the optimal policy that maximizes the total discounted expected reward [15]. In particular, let

$$J_\pi(s_0) = \sum_{k=0}^{\infty} \gamma^k \bar{r}(s_k, \mu_k(s_k)),$$

be the total discounted expected reward, where $\pi = \{\mu_0, \mu_1, \ldots, \}$ is a sequence of attack policies $\mu_k, k = 0, 1, \ldots$, where μ_k is a function that maps the state space to the set of allowable attack actions at time k. The constant $0 < \gamma < 1$ is a discount factor that accords less weight to future rewards and s_0 is the initial state. The attacker aims to solve the optimization problem

$$\max_\pi J_\pi(s_0). \quad (5)$$

D. Attack Policy

The optimal solution to the aforementioned discounted cost formulation is a linear time-invariant policy μ^* [15] defining the optimal action for the attacker to take at each state, and can be obtained as a solution to the Bellman equation,

$$J(s_k) = \max_{u_k \in \mathcal{U}(s_k)} \{\bar{r}(s_k, u_k) + \gamma \ \mathsf{E}[J(s_{k+1})]\} \quad (6)$$

where $\mathcal{U}(s_k)$ denotes the set of allowable controls given the current state s_k and $\mathsf{E}[.]$ denotes the expectation w.r.t. the future state. However, the exact solution is intractable as it requires the attacker to enumerate all possible states and choose an optimal action for every such state. Even if the maximum number of successive no-attack actions is restricted to N, the state space would be of size $|\mathbb{C}|^{|\mathbb{E}|} \times |\mathbb{E}| \times N$ which grows exponentially with the number of edges. Hence, it is computationally intractable to evaluate the expected state reward for all the reachable states and actions combinations.

Therefore, we develop an approach to obtain suboptimal attack policies based on Approximate Policy Iteration (API) [15] by judiciously extracting relevant state features and estimating the rewards for a set of representative states sampled from the state space. For every state s of these representative states, we define a vector of features, $\phi(s)$ to be weighted by a weighting vector $\mathbf{r}$ to form a value function. Let $\tilde{J}_\mathbf{r}(s)$ denote an approximate value for state s (approximation of J) as,

$$\tilde{J}_\mathbf{r}(s) = \sum_{j=1}^{M} r_j \phi_j(s) \quad (7)$$

where M is the total number of features, ϕ_j denotes the j-th feature, and $r_j, j = 1, \ldots, M$, its weight. The sub-optimal policy $\tilde{\mu}$ is obtained by iterating between policy evaluation and policy improvement steps. We start with some arbitrary initial policy. For policy evaluation, we use Monte Carlo simulations to run a large number of trajectories from every representative state. The average total reward of these trajectories is used to evaluate the feature weights vector $\mathbf{r}$ by solving a simple least squares problem. Then, to compute an improved action for state s, we solve

$$\tilde{\mu}(s) = \arg\max_u \sum_{s'} p(s'|s, u) \left[\bar{r}(s, u, s') + \gamma \ \tilde{J}_\mathbf{r}(s')\right] \quad (8)$$

where the optimal value function is replaced by the approximate parametric form $\tilde{J}_\mathbf{r}$.

E. Feature Selection

The state features were selected to capture important characteristics that affect the attacker's decision. We used a total of $M = 10$ features as described below.

- ϕ_1: Number of edges in conflict with one or more neighbors in the current state.
- ϕ_2: Ratio of maximum number of edges involved in the same conflict to the degree of the network graph.
- ϕ_3: Average number of edges involved in the same conflict.
- ϕ_4: Average number of channels unavailable to an edge.
- ϕ_5: Average conflict size of the highest degree edge(s).
- ϕ_6: Last edge attacked.
- ϕ_7: Steps since last attack.
- ϕ_8: Flag for whether attacker is at the most complex edge (MCE), i.e., the edge with highest degree.
- ϕ_9: Degree of largest neighbor of the last attacked edge.
- ϕ_{10}: Fraction of edges within hop distance.

We use these features to capture all relevant characteristics to each state.

IV. Experimental Results

In this section, we study the performance of the proposed API suboptimal policy in comparison to other attacking policies described below.

1) No-attack: We use this as a baseline to compare different attack policies. The attacker does nothing at every step. Hence, it is the lowest cost policy. In different representative states, a network may start with a number of conflicts that resolve over time, hence, there is a non-zero path reward even with the no-attack policy.
2) MCE: The attacker greedily attacks the most complex edge, i.e, the edge with the highest degree among all reachable victims. Hence, the attacker can disrupt the maximum number of edges with a single attack action. If the highest degree is not unique, the attacker will pick the closest one.
3) Random: In this policy, the attacker randomly chooses between attack or no-attack. The attacker selects any random edge within one hop distance if attacking and broadcasts a conflicting channel.

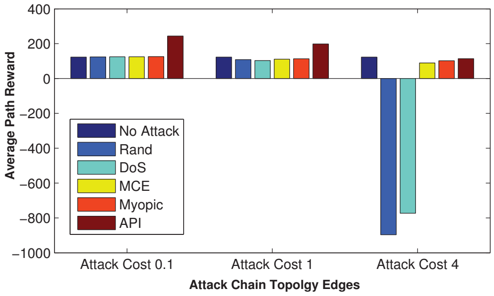

Fig. 2: Chain Network Topology-comparing different attack policies at different attack costs.

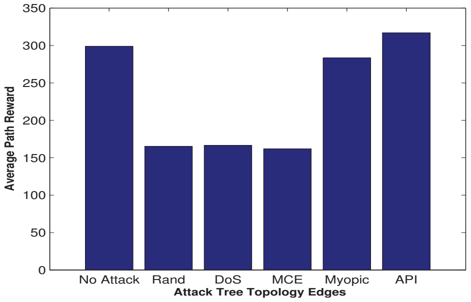

Fig. 3: Tree Network Topology-comparing different attack policies.

4) DoS: The attacker selects any edge and any random channel to broadcast at every step randomly. However in DoS attack, the attacker never chooses not to attack. Instead, attacks are launched at every step without considering the attack cost. Therefore, DoS incurs the maximum cost.
5) Myopic: The attacker selects a victim without considering any features of potential states, but only considering the immediate reward and cost.
6) API: The attacker uses the features of the state and the feature weights computed during policy iteration to intelligently select an attack that potentially leads to more rewarding states among all potential states.

We tested several network topologies. For a 6-node chain network topology, a range of attack cost values $(0.1, 1$ and $4)$ was considered over 30 steps in order to observe the attacker's behavior under different attack cost ranges. As shown in Fig. 2, the path reward for the no-attack policy is constant across all attack costs. We use the average path reward of the no-attack policy as a baseline to compare the performance of the different attack policies described above. When the attack cost is small, 0.1, the DoS attack policy outperforms other policies except for the proposed API attack policy. This is due to the low cost constant broadcasting of fake or jamming channels that forces constant switching in the network under the DoS policy. On the other hand, when we raise the attack cost to 4, API almost achieves the same average path reward of the no-attack policy due to the high cost of launching an attack. This shows that the proposed API policy adapts well to the cost of the attack to achieve a favorable cost-reward tradeoff.

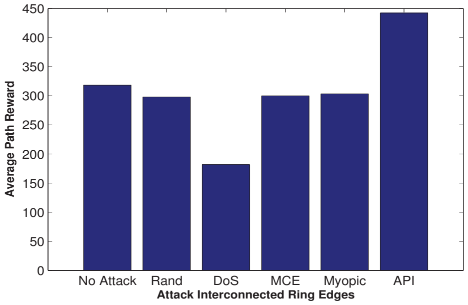

Fig. 4: Interconnected Ring Network Topology-comparing different attack policies.

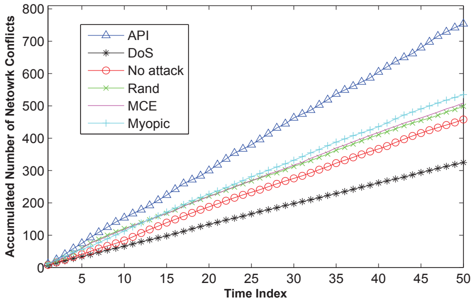

Fig. 5: Accumulated Damage by different attacking policies over time for an Interconnected Ring Network.

Fig. 3 shows the average path reward of the different attack policies with attack cost 1 for the tree topology illustrated in Fig. 8. It is clear that the proposed API policy outperforms all other attacking policies including the no-attack policy, which is generally desirable at higher values of the attack cost. For the 5-node interconnected ring topology illustrated in Fig. 9, the API policy significantly outperforms all the other attacking policies as demonstrated in Fig. 4. In Fig. 6, we plot the average path reward for the 6-node ring network topology of Fig. 7. It is clear that the average path reward of the API policy is almost equal to that of the Myopic policy when the attack cost is 1. This is due to the special structure of the ring network as all edges have the same degree. API outperforms the DoS policy and the Random policy. In general, the DoS policy is associated with the highest cost. On the other hand, the API policy adapts the control actions based on the values of the different states to strike a favorable tradeoff between cost and reward. Although the Myopic, DoS and API policies can create conflicts in the system, the created conflicts affect

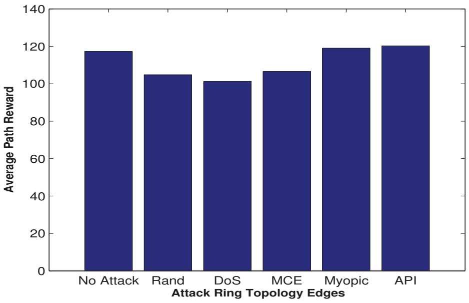

Fig. 6: Ring Network-comparing different attack policies.

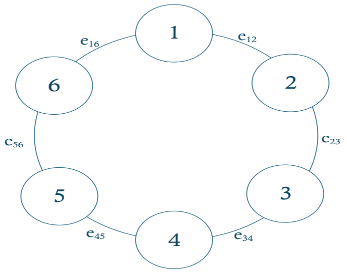

Fig. 7: 6-node Ring Network Topology.

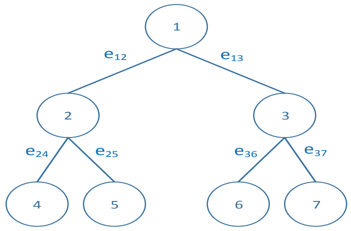

Fig. 8: 7-node Tree Network Topology.

the network unequally. Typically, DoS creates small conflicts in the system, which may not compound and the system could resolve them within few steps. The Myopic policy consistently creates larger conflicts than DoS. Even when the system is able to resolve these conflicts, the attacker can quickly create new ones. API creates large conflicts as well, but the system is typically unable to resolve them as much as it does for the myopic policy. This observation is demonstrated in Fig. 5 where the accumulated damage caused by the attacker to an Interconnected network is shown over time. The API policy is shown to cause the maximum damage over time.

V. Conclusion and Future Work

In this paper, we studied the susceptibility of dynamic channel assignment to stealthy edge decoy attacks. We formulated the problem through an MDP framework and identified suboptimal, yet efficient, attack policies through approximate dynamic programming. The attack policies obtained determine which edges to attack and at what frequency channels in order to cause conflicts that propagate through the network thereby degrading the overall performance. These policies were optimized to strike a favorable tradeoff between the damage inflicted by the attacker and the cost of the attack to ensure that the attacker remains stealthy. Our results show that the obtained API policy outperforms other attack policies such as Denial of Service and random attack, among other heuristic policies. The proposed API policy adapts to the cost of the attack, namely, with higher attack costs API is more selective. As Software Defined Networks become more mainstream, it is important to ensure that their adaptive behavior is resilient to exploits. This work exposes the vulnerability of these dynamic protocols and underscores the importance of developing proper defense mechanisms to ensure their correct behavior.

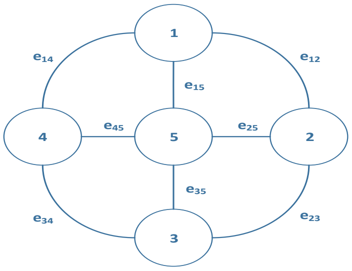

Fig. 9: 5-node Interconnected Network Topology.

References

[1] C. Monsanto, J. Reich, N. Foster, J. Rexford, D. Walker *et al.*, "Composing software defined networks," in *NSDI*, 2013, pp. 1–13.

[2] D. López-Pérez, A. Valcarce, G. De La Roche, and J. Zhang, "OFDMA femtocells: A roadmap on interference avoidance," *IEEE Communications Magazine*, vol. 47, no. 9, pp. 41–48, 2009.

[3] R. Akl and A. Arepally, "Dynamic channel assignment in ieee 802.11 networks," in *IEEE International Conference on Portable Information Devices (PORTABLE07)*. IEEE, 2007, pp. 1–5.

[4] D. Murray, M. Dixon, and T. Koziniec, "Scanning delays in 802.11 networks," in *The 2007 International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST'07)*. IEEE, 2007, pp. 255–260.

[5] M. Haidar, R. Ghimire, H. Al-Rizzo, R. Akl, and Y. Chan, "Channel assignment in an ieee 802.11 wlan based on signal-to-interference ratio," in *Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE, 2008, pp. 001 169–001 174.

[6] A. Mishra, S. Banerjee, and W. Arbaugh, "Weighted coloring based channel assignment for WLANs," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 9, no. 3, pp. 19–31, 2005.

[7] H. Zhu, X. Lin, R. Lu, P.-h. Ho, and X. Shen, "Slab: A secure localized authentication and billing scheme for wireless mesh networks," *IEEE Transactions on Wireless Communications*, vol. 7, no. 10, pp. 3858–3868, 2008.

[8] F. Martignon, S. Paris, and A. Capone, "Design and implementation of MobiSEC: A complete security architecture for wireless mesh networks," *Computer Networks*, vol. 53, no. 12, pp. 2192–2207, 2009.

[9] E. W. T. Ferreira, R. de Oliveira, G. A. Carrijo, and B. Bhargava, "Intrusion detection in wireless mesh networks using a hybrid approach," in *29th IEEE International Conference on Distributed Computing Systems Workshops, ICDCS Workshops*. IEEE, 2009, pp. 451–454.

[10] A. Morais and A. Cavalli, "A distributed and collaborative intrusion detection architecture for wireless mesh networks," *Mobile Networks and Applications*, vol. 19, no. 1, pp. 101–120, 2014.

[11] Q. Gu, M. Yu, W. Zang, and P. Liu, "Lightweight attacks against channel assignment protocols in MIMC wireless networks," in *IEEE International Conference on Communications (ICC)*. IEEE, 2011, pp. 1–6.

[12] B. Konings, F. Schaub, F. Kargl, and S. Dietzel, "Channel switch and quiet attack: New DoS attacks exploiting the 802.11 standard," in *IEEE 34th Conference on Local Computer Networks (LCN)*. IEEE, 2009, pp. 14–21.

[13] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The feasibility of launching and detecting jamming attacks in wireless networks," in *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2005, pp. 46–57.

[14] D. Thuente and M. Acharya, "Intelligent jamming in wireless networks with applications to 802.11 b and other networks," in *Proc. 25th IEEE Communications Society Military Communications Conference (MILCOM06), Washington, DC*, 2006, pp. 1–7.

[15] D. P. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific Belmont, MA, 1995, vol. 1, no. 2.

[16] M. Guirguis and G. Atia, "Stuck in traffic (SiT) attacks: A framework for identifying stealthy attacks that cause traffic congestion," in *IEEE 77th Vehicular Technology Conference (VTC Spring)*. IEEE, 2013, pp. 1–5.

[17] V. Nguyen, M. Guirguis, and G. Atia, "A unifying approach for the identification of application-driven stealthy attacks on mobile CPS," in *52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Sept 2014, pp. 1093–1101.

[18] Y. Xiao, "Ieee 802.11 n: enhancements for higher throughput in wireless lans," *IEEE Wireless Communications*, vol. 12, no. 6, pp. 82–91, 2005.