A reinforcement learning approach for attack graph analysis

Yousefi, Mehdi; Mtetwa, Nhamo; Zhang, Yan; Tianfield, Huaglory

Published in:
2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)

DOI:
10.1109/TrustCom/BigDataSE.2018.00041

Publication date:
2018

Document Version
Author accepted manuscript

A Reinforcement Learning Approach for Attack Graph Analysis

Mehdi Yousefi, Nhamo Mtetwa, Yan Zhang, and Huaglory Tianfield
Department of Computer, Communications and Interactive Systems (CCIS)
Glasgow Caledonian University
Glasgow, United Kingdom
E-mail: {mehdi.yousefi, nhamoinesu.mtetwa, yan.zhang, h.tianfield}@gcu.ac.uk

Abstract— Attack graph approach is a common tool for the analysis of network security. However, analysis of attack graphs could be complicated and difficult depending on the attack graph size. This paper presents an approximate analysis approach for attack graphs based on Q-learning. First, we employ multi-host multi-stage vulnerability analysis (MulVAL) to generate an attack graph for a given network topology. Then we refine the attack graph and generate a simplified graph called a transition graph. Next, we use a Q-learning model to find possible attack routes that an attacker could use to compromise the security of the network. Finally, we evaluate the approach by applying it to a typical IT network scenario with specific services, network configurations, and vulnerabilities.

Keywords— cyber security; reinforcement learning; Q-learning; attack graph

I. INTRODUCTION

Computer networks are often connected to the Internet which is inherently an insecure network and therefore they will be a target for cyber-attacks. Moreover, due to weaknesses in technical design, configurations, and security policies of the network equipment and services, vulnerabilities are commonplace in computer networks. Therefore, an attacker can combine vulnerabilities in different ways and incrementally penetrate the network and compromise critical network resources. Proactive security measurements are vital to protecting the network against highly sophisticated malicious activities. Tracking vulnerabilities and finding the interconnectivity between them is one important measurement that can reduce the impact of an attack by showing the chain of vulnerabilities that lead to a successful attack. However, this process is a laborious, time-consuming, and complex task which requires a great deal of knowledge which is also prone to error [1].

The automation of vulnerability assessment is necessary in order to understand the overall security posture of a network. The attack graph technique is an important approach that simulates possible attack paths in a network [1, 2]. Quantitative analysis of attack graphs can reveal important information that can help security practitioners to defend their network proactively. Q-learning is a form of model-free reinforcement learning [3] can be used to analyze an attack graph. Here, an agent tries an action among all the possible actions in a particular state, and evaluates the results based on the reward it receives. The agent tries all the possible actions in all states repeatedly, and overall it learns which are best, and decides based on a long-term discounted reward [3]. If we consider an attacker as the agent, we can rank different paths in the attack graph based on attacker maximum reward (for instance, the reward could be the amount of damage an action causes to the network) using Q-learning.

In this paper, a security approach is proposed to find optimal action policy to defend against how an attacker might compromise the security of a given network. This optimal action policy could be used to improve the network security. First, we employ MulVAL [4] to find all the attack paths in the network. Second, we use an algorithm [5] to refine the attack graph to produce a transition graph which is used to model the environment for Q-learning. Furthermore, this framework makes it easy to understand and analyze the attack graph for monitoring purposes [5]. We then use CVSS to model the reward system for attacker's possible actions and apply Q-learning to find attacker's possible actions. The rest of the paper is organized as follows: in section II, we review some work in this field. In section III, the details of the framework are presented. In section IV, we give an example to evaluate the framework. Finally, in section V, we compare the work with similar work in the field and conclude the paper.

II. LITERATURE REVIEW

Cyber vulnerability analysis is "the process of identifying the vulnerabilities in a system and prioritizing them according to their severity". It assists in discovering weaknesses in a given system in the application of proper patches [6]. In [7-9], the authors provide a standard security evaluation baseline. However, it does not include quantitative measurements. To address this issue, CVSS [9] was developed by NIST as a standard composite scoring system model. This model is usable and understandable by security practitioners. The issue is that CVSS considers vulnerabilities as isolated entities. In case there are multiple vulnerabilities, CVSS has no foresight of exploitable possible interrelationships between the vulnerabilities in the system. Further work was done in [11-16], to address the aforementioned problem. The attack graph technique is a common technique for the evaluation of network security. It helps to automatically identify possible multi-stage attacks in enterprise networks. The focus of the authors in [16, 17-18] is to improve the attack graph generation and reduce the complexity of employed algorithms. The main advantage of their approach is the consideration of interrelationships between

vulnerabilities in the system. However, attack graphs are too large and complex to be interpreted by security practitioners. This problem has been addressed using various approaches [15, 20-21]. These attempts try to improve the visualization of an attack graph through abstraction [15], data reduction [19], and user interaction [21].

Attack graph techniques draw all the possible attack paths. However, it is necessary to integrate the attack graph with some sort of metric to measure the validity and possibility of each attack path. In other words, there is a need to establish security frameworks capable of measuring the security risks in enterprise networks objectively.

III. Security Framework using Attack Graph and Q-learning

The proposed framework includes several steps. The first step is to reconnoiter the network which includes gathering information about the current state of the network. This includes a list of known vulnerabilities, configured network services and access rules. This information is necessary to model the environment to apply Q-learning. The second step is generating the attack graph. It shows the interdependencies between vulnerabilities in the system based on network connectivity (all possible attack paths). We use MulVAL to generate the attack graph. MulVAL is an open source tool and the complexity of the attack graph generation algorithm grows between $O(n^2)$ and $O(n^3)$ [4]. Then we simplify the attack graph and generate a transition graph [5]. The final step is to use the transition graph to model the environment and the reward system for the agent (attacker) and apply Q-learning which returns the possible routes for the attacker to reach the goal. We use an example scenario to show the aforementioned steps and evaluate the proposed framework.

A. Q-learning Algorithm

Reinforcement learning techniques can be used for the quantitative security evaluation of large-scale enterprise networks. One such reinforcement learning technique is Q-learning [3]. This is a simple approach for agents to take actions optimally in controlled Markovian domains [22]. This model seeks to find an optimal action selection policy that produces maximal cumulative rewards using a trial-and-error approach [23]. Here, the problem model consists of an agent, states and the agent has a set of actions per state. The agent's movement between states is called an action. Taking an action by the agent in a specific state results in a numerical value reward. The agent's aim is to maximize its total reward. The quality of each action is decided based on the associated reward which is a feedback from the environment. Q-learning in its simplest form which is known as one-step Q-learning is defined by the following equations (1) [22].

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right] \quad (1)$$
$$Q : S \times A \rightarrow R$$

Here, S is a set of states and A is a set of actions per state. By performing one action $A_t \in A$, the agent moves from one state to another state. R_t is the reward observed with the current state S_t. Factor α is a learning rate parameter between 0 and 1, setting it to 0 means that the Q-values are never updated. Factor γ is called the discount factor, which is a number between 0 and 1. If γ is closer to 0, the agent tends to consider an immediate reward, but, if it is closer to 1, the agent is willing to delay the reward. In this case, Q is the learnt action-value function, independently of the policy being followed, approximates Q* as optimal action-value function which is defined by the following equations (2) [23].

$$Q_n(S_t, A_t) \rightarrow Q^*(S_t, A_t) \ \ as \ \ n \rightarrow \infty \quad (2)$$

Here the agent without prior knowledge explores the state until it reaches the goal. The exploration that starts from initial state and ends in goal state that is called an episode. Once the agent arrives at the goal state, the agent starts the next episode until the algorithm reaches the convergence. Fig. 1 shows the interaction between the agent and the environment in reinforcement learning [23].

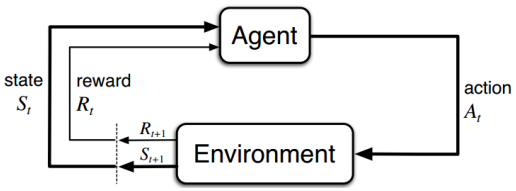

Fig. 1. Reinforcement learning flowchart

The pseudo code of Q-learning is shown in Algorithm Q-learning.

Algorithm Q-learning

```
\\ Input: State/Reward (Environment) Matrix (R),
         Discounted factor(γ),
         Learning rate (α)
\\Output: Optimal action selection policy (Matrix Q)
\\ States/Actions: State: S_t ϵ S/Action: A_t ϵ A
Procedure qlearning (R, learning rate)
    Initialize Q as 0s (arbitrarily for all state-action pairs in R)
    For each Exploration do
      Select a random initial state (S_t ϵ S)
      While current state != goal state do
        Select one action from all possible actions for the current
         State
        Take the action and observe the outcome state
         and reward
        Update Q: Q(S_t, A_t) ← Q(S_t, A_t) + α[R_{t+1} +
                                  γ max_a Q(S_{t+1}, a) − Q(S_t, A_t)
        Current state = next state
      End While
    End For
    Return Q
End Procedure
```

In the above algorithm, the calculation of the matrix Q is based on a single goal. However, in attack graphs, usually there is more than one goal. We alter the Q-learning algorithm to apply to attack graph scenarios. Details of the algorithm to analyze the attacker behavior are described in Algorithm Q-learning-Attack-Graph. We use sub-matrixes of R (Matrix r) to generate a Q matrix for each goal. In order to do that each time one goal is chosen and the remaining goals along with their

corresponding rows and columns will be removed from the matrix R (Algorithm Q-learning-Attack-Graph, line 5, 6) and the result will be the submatrix r as reward matrix for the Q-learning.

Algorithm Q-learning-Attack-Graph

1: \\ **Input:** State/Reward (Environment) Matrix (R) is a n×n and n is the number vertices in transition graph, Discounted factor(γ), Learning rate (α)
2: **Output:** Attacker's optimal action selection policy to compromise security of the network (Matrix Q)
3: \\ **States/Actions:** State: $S_t \in S$ Action: $A_t \in A$
4: **Procedure** Q-learning-Attack Graph (R, learning rate)
5: **For** all goals in the transition graph **do**
6: r = remove rows and columns of all goals except the current goal from matrix R
7: **Initialize** Q as 0s (arbitrarily for all state-action pairs in r)
8: \\ Q Calculation
9: **For** each Exploration **do**
10: Select a random initial state ($S_t \in S$)
11: **While** current state != goal state **do**
12: Select one action from all possible actions for the current State
13: Take the action and observe the outcome state and reward
14: Update Q : $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)$
15: Current state = next state
16: **End While**
17: **End For**
18: **Return Q**
19: **End For**
20: **End Procedure**

B. *Modelig the Environment*

In order to apply Q-learning, we need to model the environment. Here, the environment is modelled by an attack graph which consists of all the possible attack paths in a given network with specific configuration and the attacker is the agent which is located on the Internet. We use MulVAL [4] to generate the attack graph and use an approch proposed in [5] to simplify the attack graph (transition graph). The transition graph is used to model the simplified environment. In this case, an attacker can move between the nodes in any direction in the graph until the attacker arrives in one of the goal states, where the attacker will stay forever. We also need to introduce a kind of reward value to each edge of the graph. We use Common Vulnerability Scoring System (CVSS) to assign reward values based on vulnerabilities in the system.

IV. EXAMPLE TO EVALUATE THE FRAMEWORK

A. *Topology, System Configuration*

The network topology diagram is illustrated in Fig. 2. There are three distinct services including a webServer and a mailServer in the same subnet and a fileServer on another subnet. It has also two distinct subnets (subnet_1 and subnet_2 in Fig. 2). The client in subnet_1 is running Windows 2000 and client in subnet_2 is running Internet Explorer. There is also a Workstation, which is running Acrobat in the same subnet as the fileServer.

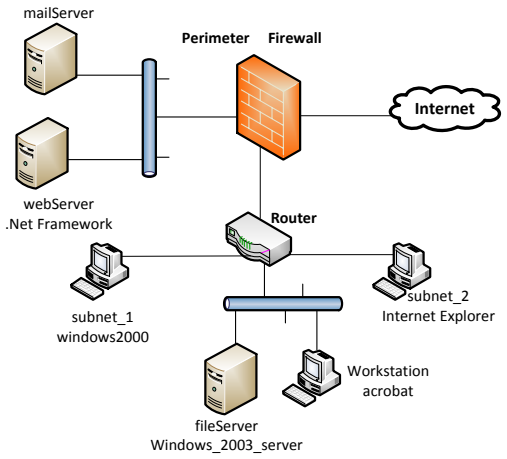

Fig. 2. Network Topology

There is a perimeter firewall. The network connectivity in this network topology is based on firewall rules described as follows: (i) The attacker is located on the Internet and has access to the webServer through the HTTP protocol and HTTP port, (ii) There is bidirectional connectivity between webServer and other machines in the network. (iii) The webServer and fileServer have access to each other through NFS protocol and NFS port, (iv) fileServer, subnet_1 and subnet_2 have access to the Internet through the HTTP protocol and the HTTP port, (v) The fileServer and Workstation have access to each other through the NFS protocol and the NFS port.

There are four distinct software vulnerabilities in the topology. National Vulnerability Database (NVD) assigns a unique identifier to each known vulnerability. These identifiers help security practitioner to obtain information about vulnerabilities. The webServer contains a vulnerability named 'CVE-2002-0392'. This is a weakness that allows remote attackers to cause a denial of service and possibly execute arbitrary code. The client in subnet_1 contains a vulnerability named 'CVE-2010-0483'. This weakness is related to vbscript.dll in VBScript 5.1,6,7, and 8 in Windows 2000 SP4, when Internet Explorer (IE) is used allows the attacker to execute arbitrary code. The client in subnet_2 contains a vulnerability named 'CVE-2010-0490'. This is a weakness in IE 6, 7 and 8 with the possibility that remote intruder can execute arbitrary code on the target machine. The fileServer contains a vulnerability named 'CVE-2010-0492' that is related to Windows 2003 SP2 with the possibility that attacker can bypass intended IPv4 source address restrictions. Table 1 summarizes the vulnerabilities in the system.

Table 1: Vulnerabilities in the Topology

List of Vulnerabilities	Vulnerabilities and Associated Machines	
	Machine	***Vulnerability ID (NVD)***
1	webServer	CVE-2002-0392
2	subnet_1	CVE-2010-0483
3	subnet_2	CVE-2010-0490
4	fileServer	CVE-2010-0812

B. *Generating Attack Graph and Transition Graph*

We use MulVAL [4] to generate the attack graph. Fig. 3 shows the attack graph along with a description of nodes in the graph.

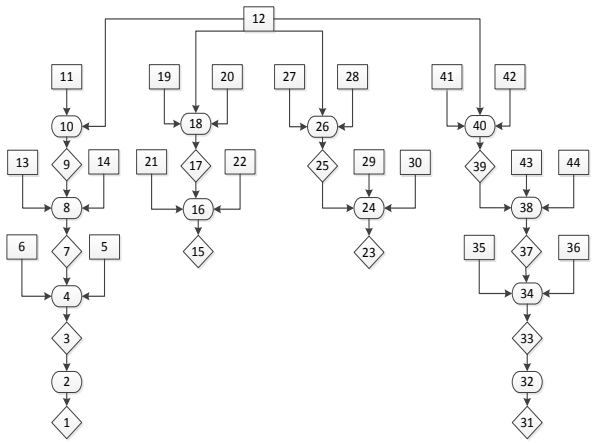

Fig. 3. Attack Graph with nodes' description

1,"execCode(mailServer,root)","OR",0
2,"RULE 4 (Trojan horse installation)","AND",0
3,"accessFile(mailServer,write,'/export')","OR",0
4,"RULE 17 (NFS shell)","AND",0
5,"hacl(webServer,mailServer,nfsProtocol,nfsPort)","LEAF",1
6,"nfsExportInfo(mailServer,'/export',write,webServer)","LEAF",1
7,"execCode(webServer,apache)","OR",0
8,"RULE 2 (remote exploit of a server program)","AND",0
9,"netAccess(webServer,tcp,80)","OR",0
10,"RULE 6 (direct network access)","AND",0
11,"hacl(internet,webServer,tcp,80)","LEAF",1
12,"attackerLocated(internet)","LEAF",1
13,"networkServiceInfo(webServer,httpd,tcp,80,apache)","LEAF",1
14,"vulExists(webServer,'CAN-2002-0392',httpd,remoteExploit,privEscalation)","LEAF",1
15,"execCode(subnet_1,user)","OR",0
16,"RULE 3 (remote exploit for a client program)","AND",0
17,"accessMaliciousInput(subnet_1,victim_1,windows_2000)","OR",0
18,"RULE 22 (Browsing a malicious website)","AND",0
19,"hacl(subnet_1,internet,httpProtocol,httpPort)","LEAF",1
20,"inCompetent(victim_1)","LEAF",1
21,"hasAccount(victim_1,subnet_1,user)","LEAF",1
22,"vulExists(subnet_1,'CVE-2010-0483',windows_2000,remoteClient,privEscalation)","LEAF",1
23,"execCode(subnet_2,user)","OR",0
24,"RULE 3 (remote exploit for a client program)","AND",0
25,"accessMaliciousInput(subnet_2,victim_2,ie)","OR",0
26,"RULE 22 (Browsing a malicious website)","AND",0
27,"hacl(subnet_2,internet,httpProtocol,httpPort)","LEAF",1
28,"inCompetent(victim_2)","LEAF",1
29,"hasAccount(victim_2,subnet_2,user)","LEAF",1
30,"vulExists(subnet_2,'CVE-2010-0490',ie,remoteClient,privEscalation)","LEAF",1
31,"execCode(workStation,root)","OR",0
32,"RULE 4 (Trojan horse installation)","AND",0
33,"accessFile(workStation,write,'/export')","OR",0
34,"RULE 17 (NFS shell)","AND",0
35,"hacl(fileServer,workStation,nfsProtocol,nfsPort)","LEAF",1
36,"nfsExportInfo(workStation,'/export',write,fileServer)","LEAF",1
37,"execCode(fileServer,user)","OR",0
38,"RULE 3 (remote exploit for a client program)","AND",0
39,"accessMaliciousInput(fileServer,victim_3,windows_2003_server)","OR",0
40,"RULE 22 (Browsing a malicious website)","AND",0
41,"hacl(fileServer,internet,httpProtocol,httpPort)","LEAF",1
42,"inCompetent(victim_3)","LEAF",1
43,"hasAccount(victim_3,fileServer,user)","LEAF",1
44,"vulExists(fileServer,'CVE-2010-0812',windows_2003_server,remoteClient,privEscalation)","LEAF",1

There are three different vertices in the attack graph. The square vertices (e.g. nodes 11, 22 and 44 in Fig. 3) are related to system configuration. For example, firewall rules that let the web server be accessible from the Internet or the buggy software on a machine. The diamond vertices (e.g. nodes 9, 25, and 37 in Fig. 3) represent potential privileges or access that an attacker can obtain in the system, e.g., code execution privilege on the web server. The elliptical vertices (e.g. nodes 10, 16, and 24 in Fig. 3) link preconditions to postconditions. As an example, it is necessary for an attacker to have access to a machine that has a vulnerability, to be able to exploit the vulnerability and obtain privileges.

We use the approach proposed in [5] to refine the attach graph and generate the transition graph. This graph draws all the possible attacker's movements between vulnerabilities in the network and it is simplified to be understood and interpreted easily. In previous work the researchers used this graph as a transition graph for a Markov model and in this work the researchers used it to generate a reward matrix for the Q-learning model. Fig. 4 shows the refined attack graph using the algorithm in [5].

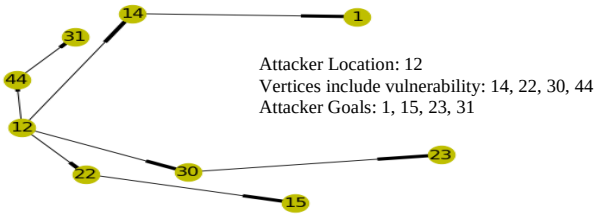

Fig. 4. Transition Graph

C. *Modeling the attack graph as Q-learning Environment*

The transition graph in Fig. 4 is used to model the environment. Here, the attacker is the agent which is located on the Internet. We represent the attacker's location, vulnerabilities, and attacker goals as vertices (states) and each edge represents the possibility of the attacker's movement and it is called an action. In this case, we suppose the attacker can move between the nodes in any direction in the transition graph until the attacker arrives at one of the goal states. Furthermore, we use Common Vulnerability Scoring System (CVSS) to assign reward values. We use the overall score which includes the exploitability rate and impact on the system. If the edge leads the attacker to a node with vulnerability (attacker exploits the vulnerability) the overall score related to that vulnerability will be considered as the reward value. If the edge of the graph leads to a goal state (red nodes) the reward will be considered as 100 (maximum possible score). Additional looped edges are added to the goal vertices with the reward of 100, it is because when attacker reaches the goal, the attacker will stay there forever. The reward model is shown in Fig. 5.

Suppose the attacker is on the Internet which is shown as node 12 (green node). The Attacker has the possibility to move to four different possible nodes including 14, 22, 30, and 44 that each arrow or edge (action) has got the appropriate reward (based on existing vulnerabilities) which is shown next to each edge.

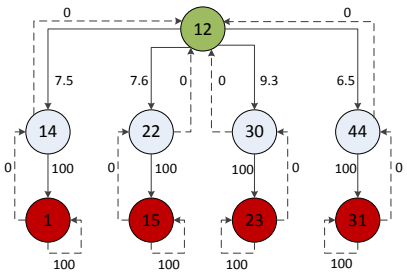

Fig. 5. Graph related modelling the environment

Suppose the attacker takes the action to go to node 44, now there are two options, either to move to node 31 which is a goal state and earn 100 points or go back to the starting location with the reward of zero. We put the state diagram along with the instant reward values (Fig. 5) into the matrix R. The matrix R for the diagram in Fig. 5 is shown in Table 2.

Table 2: Matrix R

R	12	14	22	30	44	1	15	23	31
12	-1	7.5	7.6	9.3	6.5	-1	-1	-1	-1
14	0	-1	-1	-1	-1	100	-1	-1	-1
22	0	-1	-1	-1	-1	-1	100	-1	-1
30	0	-1	-1	-1	-1	-1	-1	100	-1
44	0	-1	-1	-1	-1	-1	-1	-1	100
1	-1	0	-1	-1	-1	100	-1	-1	-1
15	-1	-1	0	-1	-1	-1	100	-1	-1
23	-1	-1	-1	0	-1	-1	-1	100	-1
31	-1	-1	-1	-1	0	-1	-1	-1	100

The rows in matrix R represent the states, and the columns represent the actions. For instance, R(12, 44) means the attacker goes from node 12 to node 44 and earns 6.5 points, and R(15, 31) means there is no direct edge between these two nodes which is shown by a reward of -1. Matrix R represents the modeled environment and reward system. Details of the algorithm to analyze the attacker behavior is described in Algorithm (Q-learning-Attack-Graph). First, we use the transition graph in Fig. 4 to model our environment then we feed the matrix R into our algorithm.

D. Result and Analysis

In this section, we apply the algorithm (Q-learning-Attack-Graph) on our model which is shown with matrix R. The algorithm each time extracts submatrix r from matrix R and generates matrix Q for the submatrix r. Fig. 6. shows the diagram after keeping one of the goals (node 1) and removing the other goals (nodes 15, 23, and 31) and corresponding edges. Using diagram in Fig. 6 the reward matrix (r) which is a submatrix of matrix R is calculated and is shown in Table 3.

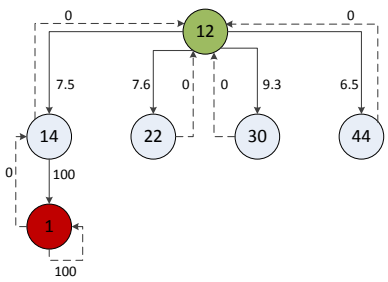

Fig. 6. Environment diagram (goal is node 1)

Table 3: Matrix r (goal state is 1)

r	12	14	22	30	44	1
12	-1	7.5	7.6	9.3	6.5	-1
14	0	-1	-1	-1	-1	100
22	0	-1	-1	-1	-1	-1
30	0	-1	-1	-1	-1	-1
44	0	-1	-1	-1	-1	-1
1	-1	0	-1	-1	-1	100

Fig. 7 shows the diagram related to matrix Q after convergence. Here the weights on the edges of the graph show the reward that will be earned by the attacker by moving between different nodes. Using matrix Q the attacker can reach the goal state (node 1) from each individual state in the diagram. For example, if the attacker is in state 12 (Internet), the attacker can use matrix Q to reach the goal state (node 1) as follow: 1) from state 12 going to state 14 produces maximum reward (325.99). 2) From state 14 going to state 1(goal state) produces maximum reward (499.99). Therefore, the sequence of 12, 14, and 1 (12 -> 14 -> 1) with the reward of 907.49 is the preferred action sequence as it produces the maximum reward.

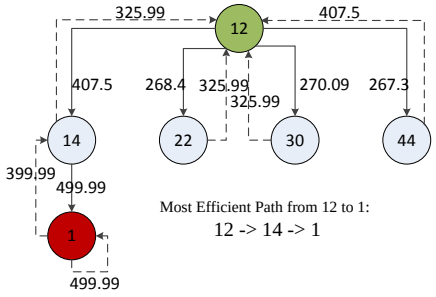

Fig. 7. Reward matrix diagram for Goal 1

The same approch applies for the remaining goals including goal 15, goal 23, and goal 31. The results of this Q-learning based approach in modelling the possible attack routes by the attacker to compromise the different goals in the system is summarised in Table 4.

Table 4: Attacker optimal action policies compromising goal states

Attacker Current State	Goal State	Optimal Path to Compromise the Goal State	Maximum Cumulative Rewards
12 (Internet)	**31**	12 -> 44 -> 31	906.48
12 (Internet)	**1**	12 -> 44 -> 31	907.49
12 (Internet)	**15**	14 -> 12 -> 44 -> 31	907.59
12 (Internet)	**23**	12 -> 30 -> 23	909.29

Using this information, we can observe the maximum damage that the attacker could cause and we can rank the

vulnerabilities in the system and tackle them based on their priority. Considering the attacker is on the Internet, if attacker compromise goal 23, the reward is 909.29, therefore, it is ranked first in terms of importance. Goal 15 and goal 1 are ranked as second and third with a reward of 907.5 and 907.49 respectively. Finally, goal 31 with the maximum reward of 906.48 is ranked as the least important goal.

Fig. 8 demonstrates the convergence of the model (matrix Q) when finding the best path, the attacker will take to compromise different goals in the system.

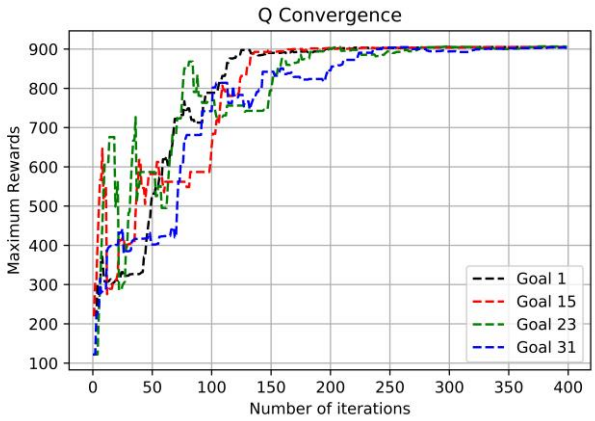

Fig. 8. Q Convergence for all Goals

Here, the x-axis shows the number of iterations and the y-axis shows the cumulative reward for each iteration. We ran the model for 1000 iterations (400 iterations are plotted) and it took around 350 iterations for the model to converge for all 4 scenarios. The maximum reward for the best path to reach the goals is slightly above 900.

V. CONCLUSION

In this work, we have proposed a novel way of assessing the security of a computer network based on the application of Q-learning to a refined attack graph. The main contribution of this paper is modeling the network topology for reinforcement learning and altering the Q-learning algorithm to be applicable to the vulnerability analysis of computer networks. First, we used a transition graph which is a simplified attack graph [5] to model the environment. Then we applied the Q-learning algorithm to find the possible attack routes which system administrators can use to formulate optimal action selection policies to patch the vulnerabilities. This is a valid way of analysing network security as we know what are the possible paths the attacker might take to compromise security and system administrators can take appropriate actions to mitigate the attacks accordingly.

REFERENCES

[1] Jajodia, S., & Noel, S. (2009). Topological vulnerability analysis: A powerful new approach for network attack prevention, detection, and response. In Algorithms, architectures and information systems security (pp. 285-305).

[2] Ou, X., & Singhal, A. (2012). Quantitative security risk assessment of enterprise networks. Springer.

[3] Watkins, C. J. C. H. (1989). Learning from delayed rewards(Doctoral dissertation, King's College, Cambridge).

[4] Ou, X., Govindavajhala, S., & Appel, A. W. (2005, July). MulVAL: A Logic-based Network Security Analyzer. In USENIX security.

[5] Yousefi, M., Mtetwa, N., Zhang, Y., & Tianfield, H. (2017, July). A novel approach for analysis of attack graph. In Intelligence and Security Informatics (ISI), 2017 IEEE International Conference on (pp. 7-12). IEEE.

[6] Kissel, R. (Ed.). (2011). Glossary of key information security terms. Diane Publishing.

[7] Ferraiolo, K. (2000). The Systems Security Engineering Capability Maturity Model.

[8] Stoneburner, G., Hayden, C., & Feringa, A. (2001). Engineering principles for information technology security (a baseline for achieving security). BOOZ-ALLEN AND HAMILTON INC MCLEAN VA.

[9] Grance, T., Hash, J., Stevens, M., O'Neal, K., & Bartol, N. (2003). SP 800-35. Guide to Information Technology Security Services.

[10] https://www.first.org/cvss. Web page accessed on May 15, 2015

[11] Dawkins, J., & Hale, J. (2004, April). A systematic approach to multi-stage network attack analysis. In Information Assurance Workshop, 2004. Proceedings. Second IEEE International (pp. 48-56). IEEE.

[12] Homer, J., Zhang, S., Ou, X., Schmidt, D., Du, Y., Rajagopalan, S. R., & Singhal, A. (2013). Aggregating vulnerability metrics in enterprise networks using attack graphs. Journal of Computer Security, 21(4), 561-597.

[13] Ingols, K., Lippmann, R., & Piwowarski, K. (2006, December). Practical attack graph generation for network defense. In Computer Security Applications Conference, 2006. ACSAC'06. 22nd Annual (pp. 121-130). IEEE.

[14] Jajodia, S., Noel, S., & O'Berry, B. (2005). Topological analysis of network attack vulnerability. In Managing Cyber Threats (pp. 247-266). Springer US.

[15] Lippmann, R. P., & Ingols, K. W. (2005). An annotated review of past papers on attack graphs (No. PR-IA-1). MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB.

[16] Sheyner, O., Haines, J., Jha, S., Lippmann, R., & Wing, J. M. (2002). Automated generation and analysis of attack graphs. In Security and privacy, 2002. Proceedings. 2002 IEEE Symposium on (pp. 273-284). IEEE.

[17] Ritchey, R. W., & Ammann, P. (2000). Using model checking to analyze network vulnerabilities. In Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on (pp. 156-165). IEEE.

[18] Lippmann, R., Ingols, K., Scott, C., Piwowarski, K., Kratkiewicz, K., Artz, M., & Cunningham, R. (2006, October). Validating and restoring defense in depth using attack graphs. In Military Communications Conference, 2006. MILCOM 2006. IEEE (pp. 1-10). IEEE.

[19] Homer, J., Varikuti, A., Ou, X., & McQueen, M. (2008). Improving attack graph visualization through data reduction and attack grouping. Visualization for computer security, 68-79.

[20] Noel, Steven, and Sushil Jajodia. "Managing attack graph complexity through visual hierarchical aggregation." Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security. ACM, 2004.

[21] Williams, L., Lippmann, R., & Ingols, K. (2008). An interactive attack graph cascade and reachability display. In VizSEC 2007 (pp. 221-236). Springer Berlin Heidelberg.

[22] Watkins, C. J., & Dayan, P. (1992). Q-learning. Machine learning, 8(3-4), 279-292.

[23] Sutton, R. S., & Barto, A. G. (1998). Reinforcement learning: An introduction (Vol. 1, No. 1). Cambridge: MIT press.