Recommender systems meeting security: From product recommendation to cyber-attack prediction

Nikolaos Polatidis[1], Elias Pimenidis[2], Michalis Pavlidis[1], Haralambos Mouratidis[1]

[1]School of computing Engineering and Mathematics, University of Brighton, BN2 4GJ, Brighton, United Kingdom
{N.Polatidis, M.Pavlidis, H.Mouratidis} @Brighton.ac.uk
[2]Department of Computer Science and Creative Technologies, University of the West of England, BS16 1QY, Bristol, United Kingdom
Elias.Pimenidis@uwe.ac.uk

Abstract. Modern information society depends on reliable functionality of information systems infrastructure, while at the same time the number of cyber-attacks has been increasing over the years and damages have been caused. Furthermore, graphs can be used to show paths than can be exploited by attackers to intrude into systems and gain unauthorized access through vulnerability exploitation. This paper presents a method that builds attack graphs using data supplied from the maritime supply chain infrastructure. The method delivers all possible paths that can be exploited to gain access. Then, a recommendation system is utilized to make predictions about future attack steps within the network. We show that recommender systems can be used in cyber defense by predicting attacks. The goal of this paper is to identify attack paths and show how a recommendation method can be used to classify future cyber-attacks. The proposed method has been experimentally evaluated and it is shown that it is both practical and effective.

Keywords: Recommender systems, Cyber security, Attack graph, Exploit, Vulnerability, Attack prediction, Classification

1 Introduction

Recommender systems are decision support systems available on the web to assist users in the selection of item or service selection in online domains. In doing so recommender systems assist users in overcoming the information overload problem [1, 2]. Collaborative filtering (CF) is the most widely used method for providing personalized recommendations. In CF systems, a database of user submitted ratings is used and the generated recommendations are generated on how much a user will like an unrated item based on previous common rated items. Thus, the recommendation process is based on assumptions about previous rating agreements and if these agreements will be maintained in the future. In addition, the ratings are used to create an n x m matrix with user ids, item ids and ratings, with an example of such a matrix shown in table 1. This database has four users and four items with values from 1 to 5.

The matrix is used as input when a user is requesting recommendations and for a recommendation to be generated the degree of similarity between the user who makes the request and the other users' needs to be predicted using a similarity function such as the Pearson Correlation Similarity (PCC) [3]. At the next step a user neighborhood which consists of users having the highest degree of similarity is created with the requester. Finally, a prediction is generated after computing the average values of the nearest neighborhood ratings about an item, resulting in a recommendation list of items with the highest predicted rating values.

Table 1. An Example of a Ratings Matrix

	Item 1	**Item 2**	**Item 3**	**Item 4**
User 1	1	2	5	-
User 2	4	5	4	1
User 3	-	-	3	2
User 4	1	1	2	5

Even though, recommender systems have been used for product or service recommendation, in the current era where cyber-attacks have been increasing we show how they can assist in the prediction of future attacks.

1.1 Problem definition and contributions

Cyber-attack prevention methods are based on graph analysis to identify attack paths or use previous attacker knowledge in combination with intrusion alerts to provide defense actions in real time. A gap is identified in attack prediction and mitigation which can be solved with the use of suitable recommendation technologies. We have made the following contributions:

1. We identify all attack paths in a graph according to constraints.
2. We use the attack paths in combination with common vulnerability data to build a recommender system that predicts future attacks.

1.2 Paper structure

In section 2 relevant background work is analyzed. In section 3 the proposed method is explained. Section 4 presents the experimental evaluation and section 5 contains the conclusions and future work parts.

2 Background

2.1 Collaborative filtering

As explained above a database of ratings and a similarity function such as PCC are the two essential parts of the CF recommendation process. Except for the classical recommendation method, PCC, another similar method found in the literature is weighted PCC (WPCC) which extends PCC by setting a statically defined threshold of common rated items. However, since the definitions of PCC and WPCC numerous approaches have been proposed with the aim of improving the recommendations. TasteMiner is a method that efficiently mines rating for learning partial users tastes to restrict the neighborhood size, thus reducing complexity and improving the accuracy of the recommendations [4]. Another CF approach that aims to improve the accuracy of the recommendations is entropy based can be found in the literature. In this approach an entropy driven similarity used to calculate the difference between ratings and a Manhattan distance model is then used to address the fat tail problem [5]. One more similarity measure for improving the accuracy of CF has been proposed with the name PIP. This measurement is based on Proximity, Impact and Popularity (PIP). Initially the proximity factor is applied to calculate the absolute difference between two ratings, then the impact factor is applied to show how strongly an item is preferred and finally the popularity factor is applied to how common the user ratings are. These three factors are then combined to calculate a final value [6]. HU-FCF is a hybrid fuzzy CF method for improved recommendations [7]. In this method, CF is extended with a fuzzy similarity that is calculated on user demographic data. A CF recommendation method based on singularities has been proposed [8]. In this method, the traditional similarities can be improved if contextual information from the entire user body are used to calculate singularities. Thus, the larger the singularity between users then the impact of it in the similarity is larger. Additionally, the use or power law augments to similarity values can be found in the literature with the name PLUS [9]. PLUS, is a method applied to user similarities to adjust their value using a power function and achieves a tradeoff between accuracy and diversity of the recommendations. Yet another approach for improved recommendations is the use of Pareto dominance [10]. Pareto dominance is used initially as a pre-filtering service were the less promising users are eliminated from the user neighborhood. Then, the rest are used in a typical CF recommendation process. An additional recommendation approach includes the breakup of the user neighborhood in multiples levels [11]. This can be done either using a static approach or a dynamic one [11, 12]. In both approaches the user similarities are adjusted either in a positive or a negative way based on the number of co-rated items and the PCC values and are assigned to one of multiple levels based on the final computed value. Thus, the predictions are made using the new user neighborhood and the recommendations are improved. An additional method that can be used to improve the quality of the recommendations is natural noise removal [13]. Items and users are characterized based on their profiles and a defined strategy is used to eliminate natural noise, thus receiving more accurate recommendations. Also, other traditional approaches exist that can be used to improve CF and include the use of

content-boosted CF or the utilization of sparsity measures [14, 15]. COUSIN is a recommendation model that improves both the accuracy and the diversity of the recommendations by using a regression model that affectively removes weak user relationships [16]. There is also an approach in the literature called Trinity that uses historical data and tags to provide personalized recommendations based on a three-layered object-user tag network [17]. In addition to the methods mentioned already the use of user-item subgroups has been proposed as a way of providing improved recommendation systems [18].

2.2 Attack graph generation and analysis

Cyber-attack prevention technologies typically use attack graph generation and analysis methods to identify all possible paths that attackers can exploit to gain unauthorized access to a system [19]. There are numerous methods available for attack graph generation and analysis. In [20] the authors use a general graph model, which is based on the JIGSAW specification language. Sample attack scenarios are created using different methods such as substitution, distribution and looping. In [21] the authors developed an intrusion correlator for intrusion alerts, which produces correlation graphs as output. Then, they use these graphs to create attack strategy graphs. The authors in [22] utilize modeling based approach that is used to perform an analysis of the security of the network. This is done using model checking tools and a model is presented that describes the vulnerability to attack of the network. In [23] the authors developed a tool called NuSMV. This is a model checking tool that implements an algorithm for automatic generation of attack graphs. A logic-based approach is proposed in [24]. In this approach, the authors use logic rules to compute the attack graph and use logic deduction to reach the final facts from the initial facts. Although, this approach suffers from performance issues as the state grows. In [25] a Breadth-first search solution is used by the authors to build the attack graph.

A layered solution is proposed where the bottom layer contains attacker privileges and the upper layer contains the privileges computer after each step of the algorithm. Once again, as the size of the graph grows there are performance issues. In [26] the authors propose an algorithm that only creates a graph containing the worst case scenarios. This approach performs better in terms of performance, but it cannot guarantee that all relevant paths will be returned. In [27] the authors try to reduce complexity by introducing the concept of group reachability. This method uses a breadth first method and uses prerequisite graphs that express reachability conditions among network hosts. The authors in [28] develop further the prerequisite graphs by adding information about client-side attacks, firewalls and intrusion detection. In [29] the authors use a distributed attack graph generation algorithm based on a multi-agent system, a virtual shared memory abstraction and hyper-graph partitioning to improve the overall performance of the system. The method is based on depth first search and it is shown that the performance is improved with the use of agents after a specific graph size. In [30] the authors use a bidirectional search method to generate the attack graph. They also apply a restriction about the depth of the search, which limits the algorithm from identifying less possible attacks. In [31] an approach that is based on artificial intelli-

gence with the name Planner is applied to generate the attack graph. Customized algorithms are used to generate attack paths in polynomial time.

In [32] the authors propose a graph-based approach to analyze vulnerabilities, that can analyze risk to a specific asset and examine possible consequence of an attack. In [33] the use of a probabilistic model is proposed. This model measures risk security, computes risk probability and considers dynamic network features. A somewhat different approach is proposed by the authors in [34]. The use of dynamic generation algorithm is proposed, that returns the top K paths. Furthermore, it is not required to calculate the full attack graph to return the top attack paths. NetSPA is a network security planning architecture that very efficiently generates the worst case attack graphs [35]. To do this the system uses information from software types and versions, intrusion detection systems, network connectivity and firewalls. In [36] the use Bayesian attack graph generation for dynamic security risk management.

In [37] the authors developed a MulVAL, a logic-based network security analyzer. This is a vulnerability analysis tool that models the interaction of software bugs along with network configurations. The data about the software bugs are provided by a bug-reporting community, while all the other relevant information is enclosed within the system. In addition to MulVAL, TVA is another tool for generating attack graphs [19, 38]. TVA is based on topological analysis of network attack vulnerability and the idea is to exploit dependency graph to represent preconditions and postconditions and then exploit. At the next step, a search algorithm finds attack paths that exploit multiple vulnerabilities.

3 Proposed method

Our proposed method takes elements from both collaborative filtering recommender systems and attack path discovery methods to identify attacks paths and predict attacks. Initially, we use an attack path discovery method that has unique characteristics, such as the attacker location, the attacker capability and which the entry and target points are. The, attack path discovery method returns all non-circular attack paths that exist between assets that belong to the specified characteristics. At the next step, we use the attack paths along with a recommender system to predict future attacks and to classify them.

3.1 Attack path discovery

Attackers can use a set of basic privileges that can satisfy some initial input requirements to gain unauthorized access to a system. Attack graphs show every possible path that an attacker can use to gain further privileges [19, 39]. In general, various vulnerabilities, such as software vulnerabilities or inappropriate configuration settings, exist in information systems and can be exploited by attackers to gain access. An infrastructure it typically comprised of numerous nodes that can be exploited to intrude into the network. In addition, the number of vulnerabilities that exist on the network and the reachability conditions that occur are the factors that determine the

size of the attack graph. In, addition as the graph becomes larger, the possibility of more exploitation options for an attacker increases. To build the attack graph we use direct conditions and utilize information from open sources. Initially, the weaknesses defined in the Common Weakness Enumeration (CWE) [40] are used and at the second step, Information from the Common Vulnerabilities and Exposures (CVE) [41] database are used. A model is introduced where an attacker can gain access to information system sources and move in a directed path. Moreover, a set of preconditions are specified, which include the length of the path, the location and capability of the attacker. The pseudocode of the attack path discovery is shown in algorithm 1.

Algorithm 1: Attack path discovery

Input: Asset graph (G), attacker location, attacker capability
Output: Graph, affected assets, attack paths

```
#We create two empty lists to hold attack paths and assets
   attackpaths = [] affectedassets = []
#We return all paths from source to target
   for e in parameters entry points
      If attacker location < required level of attacker location OR attacker capability <
      required attacker capability
      return empty graph
         else
         get single source shortest path length
         set propagation length for entry point e
            for target point t
#Create a list with all non-circular paths from entry e to target t
   get all paths in the graph G from entry e to target t that are up to the pre-specified
   path length
   for the size of paths found
      add paths to attackpaths [] list, add affected assets to affectedassets [] list
#Return the graph, the affected assets and the attack paths found as a direct input to
#the attack visualization algorithm
return Graph, affected assets, attack paths
```

3.2 Attack prediction

To recommend attack predictions we use a parameterized version of multi-level collaborative filtering method described in [11], although other methods could be applied according the scenario and the available data. This method applies collaborative filtering and then rearranges the order of the k nearest neighbors according to the similarity value and the number of co-rated items. We use characteristics from the above-mentioned method to classify attacks. To do that we initially apply classical collaborative filtering using PCC defined in equation 1. In PCC Sim (a, b) is the similarity of users a and b, $r_{a,p}$ is the rating of user a for product p, $r_{b,p}$ is the rating of user b for

product p and $\bar{r}a$, $\bar{r}b$ represent user's average ratings. P is the set of all products. At the next step, we check the similarity values returned by equation 1 and the number of co-rated vulnerabilities. Depending on the similarity value returned and the common vulnerabilities, we classify these attacks from very high to very low. Finally, we check if there are any attack paths between the assets before the classification process is finished. A detailed explanation of the steps can be found in algorithm 2 which provides the pseudocode of the attack prediction recommender system.

$$Sim\ {}^{PCC}_{a,b} = \frac{\sum p \in P(ra, p - \bar{r}a)(rb, p - \bar{r}b)}{\sqrt{\sum p \in P(ra, p - \bar{r}a)^2}\sqrt{\sum p \in P(rb, p - \bar{r}b)^2}} \quad (1)$$

Algorithm 2: Attack prediction

Input: attack paths, affected assets, vulnerabilities
Output: predicted attacks

```
#Vulnerabilities refers to common vulnerabilities between assets
load vulnerabilities
apply equation 1 using vulnerabilities as input
   get similarity values
   #If there are common vulnerabilities, then typically these receive the same score
   #between assets, thus, resulting in absolute similarities
   #Then we rearrange the order of the similarity by adding the number of co-rated
   #items as a constraint
   #classification refers to predicted attack classification, which is from very high to
   #very low
then #n is the number of co-rated items and x1, x2, x3 and x4 are fixed integers
   if n>=x1 then classification == very high
      else if n<x1 && n>=x2 then classification == high
         else if n<x2 && n>=x3 then classification == Medium
            else if n<x3 && n>=x4 then classification == Low
   else classification == very low
then
   get attack paths
      if attack path exists
         set classification == very high
      else if attack path does not exist && classification == very high then classifica-
      tion == high
      else classification == classification
Return predicted attacks
```

4 Experimental evaluation

The experiments took place in a simulated environment using a Pentium i7 2.8 GHz with 12 gigabytes of RAM, running windows 10. The data used were supplied by the maritime supply chain IT infrastructure and more particular from the port of Valencia and the experiments were conducted within a cyber-security maritime supply chain risk management system. The dataset contains 26 hardware and software assets, numerous vulnerabilities, with some of them being common within the assets. Initially, we evaluate the attack path discovery method in terms of performance, the results of which are shown in table 2. Then, we present a case study that shows how to predict attacks utilizing the data from the maritime supply chain IT infrastructure.

Table 2. Performance evaluation results

No. of test	Attacker capability	Propagation length	Running time (sec)
1	Low	3	<1
2	Low	4	<1
3	Low	5	<1
4	Medium	3	<1
5	Medium	4	<1
6	Medium	5	1
7	High	3	<1
8	High	4	1
9	Hugh	5	1.2

4.1 Case study: The maritime supply chain IT infrastructure

The maritime supply chain infrastructure it typically comprised of numerous assets that can be exploited to gain access and reach specific assets by popping from one to another. For the case study, we have used a snippet of data derived from the Valencia port IT infrastructure. In table 3 the data used show the common vulnerabilities between assets and their respective score. Assets 1, 2 and 3 are hardware assets, while the description column represents the vulnerable software asset that is installed on the respective hardware asset. Furthermore, the assets and attacks paths between them are a vital part of risk assessment. The following non-circular attack paths are present in the system:

1. Asset1 → Asset2
2. Asset2 → Asset3
3. Asset2 → Asset1

However, it should be noted that attack paths might vary according to the specific settings used, such as the propagation length, attacker location, capability, entry and target points.

Table 3. Common vulnerabilities

Assets	Description	CVE 2015-1769	CVE 2015-2423	CVE 2015-2433	CVE 2015-2485
Asset 1 (Desktop PC)	Windows 10 Installed on Desktop PC	10	2.9	2.9	10
Asset 2 (Laptop 1)	Windows 10 Installed on Laptop 1	10	2.9	2.9	10
Asset 3 (Laptop 2)	Windows 10 Installed on Laptop 2	10	2.9	2.9	-

Then the administrator executed algorithm 2 to predict very high and high classification attacks. Moreover, for the case study we have assigned the minimum number of co-rated items to be 3 for very high classification and 2 for high classification. Thus, algorithm 2 classified:

1. Asset1 → Asset2 as very high
2. Asset2 → Asset1 as very high
3. Asset1 → Asset3 as high
4. Asset3 → Asset1 as high
5. Asset2 → Asset3 as high
6. Asset3 → Asset2 as high

At the next step, the method checked for attack path relations between the assets and rearranged the classifications. Thus, the administrator received the following final predictions:

1. Asset1 → Asset2 as very high
2. Asset2 → Asset1 as very high
3. Asset2 → Asset3 as very high
4. Asset1 → Asset3 as high
5. Asset3 → Asset1 as high
6. Asset3 → Asset2 as high

4.2 Discussion

Cyber-attack prediction systems are important in risk management to provide mitigation solutions. To do that the identification of possible attack scenarios and providing defensive solutions for assets protection are the two most important parts. Furthermore, it is important for this to take place within a reasonable amount of time. It is shown that within a small amount of time the attack path discovery method delivers the non-circular attack paths between assets. Furthermore, at the next stage a classifi-

cation list is created that provides a prediction list of attack movement between assets. For example, the likelihood that an attacker who gained access to asset 1 to explore the possibility of gaining access to asset 4 is higher when compared to gaining access to either asset 2 or asset 3. However, the possibility of common vulnerabilities receiving different scores in different assets should be further exploited since this will result in different classification scales.

5 Conclusions and future work

Various online services use recommender systems for product or service recommendation. However, the use of such systems in the cyber-defense domain has not been explored. In this paper, we proposed a collaborative filtering based recommender system that uses common vulnerabilities between assets, identifies attack paths and combines the information to recommend future attacks. Although, the method is practical, it could become more effective if certain aspects are extended. Thus, in the future, we aim to investigate the following directions:

Path length recommendation. We aim to apply recommendation techniques to dynamically identify the length of the path that should be searched, thus making the attack path discovery process faster.

Attack recommendation. A part of our research will concentrate on more intelligent approaches for cyber-attack predictions based on advanced methods.

Defense recommendation. Another research direction will focus on defense strategy recommendation.

Prediction Validation. We aim to validate the attack predictions using real data from real world scenarios along with expert consultation.

Acknowledgement. This work has received funding from The European Union's Horizon 2020 research and innovation program under grant agreement No 653212.

References

1. Lu, J., Wu, D., Mao, M., Wang, W., Zhang, G.: Recommender system application developments: A survey. Decis. Support Syst. 74, 12–32 (2015).
2. Polatidis, N., Georgiadis, C.K.: Recommender Systems: The Importance of Personalization on E-business Environments. Int. J. E-entrepreneursh. Innov. 4, 32–46 (2013).
3. Su, X., Khoshgoftaar, T.M.: A Survey of Collaborative Filtering Techniques. Adv. Artif. Intell. 2009, 1–19 (2009).
4. Shams, B., Haratizadeh, S.: TasteMiner: Mining partial tastes for neighbor-based collaborative filtering. J. Intell. Inf. Syst. 48, 165–189 (2017).
5. Wang, W., Zhang, G., Lu, J.: Collaborative filtering with entropy-driven user similarity in recommender systems. In: International Journal of Intelligent Systems. pp. 854–870 (2015).

6. Liu, H., Hu, Z., Mian, A., Tian, H., Zhu, X.: A new user similarity model to improve the accuracy of collaborative filtering. Knowledge-Based Syst. 56, 156–166 (2014).
7. Son, L.H.: HU-FCF: A hybrid user-based fuzzy collaborative filtering method in Recommender Systems. Expert Syst. Appl. 41, 6861–6870 (2014).
8. Bobadilla, J., Ortega, F., Hernando, A.: A collaborative filtering similarity measure based on singularities. Inf. Process. Manag. 48, 204–217 (2012).
9. Gan, M., Jiang, R.: Improving accuracy and diversity of personalized recommendation through power law adjustments of user similarities. Decis. Support Syst. 55, 811–821 (2013).
10. Ortega, F., Sánchez, J.L., Bobadilla, J., Gutiérrez, A.: Improving collaborative filtering-based recommender systems results using Pareto dominance. Inf. Sci. (Ny). 239, 50–61 (2013).
11. Polatidis, N., Georgiadis, C.K.: A multi-level collaborative filtering method that improves recommendations. Expert Syst. Appl. 48, 100–110 (2016).
12. Polatidis, N., Georgiadis, C.K.: A dynamic multi-level collaborative filtering method for improved recommendations. Comput. Stand. Interfaces. 51, 14–21 (2017).
13. Toledo, R.Y., Mota, Y.C., Martínez, L.: Correcting noisy ratings in collaborative recommender systems. Knowledge-Based Syst. 76, 96–108 (2015).
14. Melville, P., Mooney, R.J., Nagarajan, R.: Content-boosted collaborative filtering for improved recommendations. “Proceedings 18th Natl. Conf. Artif. Intell. (AAAI).” 187–192 (2002).
15. Anand, D., Bharadwaj, K.K.: Utilizing various sparsity measures for enhancing accuracy of collaborative recommender systems based on local and global similarities. Expert Syst. Appl. 38, 5101–5109 (2011).
16. Gan, M.: COUSIN: A network-based regression model for personalized recommendations. Decis. Support Syst. 82, 58–68 (2016).
17. Gan, M.-X., Sun, L., Jiang, R.: Trinity: Walking on a User-Object-Tag Heterogeneous Network for Personalised Recommendations. J. Comput. Sci. Technol. 31, 577–594 (2016).
18. Xu, B., Bu, J., Chen, C., Cai, D.: An exploration of improving collaborative recommender systems via user-item subgroups. Proc. 21st Int. Conf. World Wide Web - WWW ’12. 21 (2012).
19. Ou, X., Singhal, A.: Attack Graph Techniques. Quant. Secur. Risk Assess. Enterp. Networks. 13–23 (2011).
20. Templeton, S.J., Levitt, K.: A requires/provides model for computer attacks. Proc. 2000 Work. New Secur. Paradig. - NSPW ’00. 31–38 (2000).
21. Ning, P., Xu, D.: Learning attack strategies from intrusion alerts. In: Proceedings of the 10th ACM conference on Computer and communication security - CCS ’03. p. 200 (2003).
22. Ritchey, R.W., Ammann, P.: Using model checking to analyze network vulnerabilities. Secur. Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symp. 156–165 (2000).
23. Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.M.: Automated generation and analysis of attack graphs. In: Proceedings - IEEE Symposium on Security and Privacy. pp. 273–284 (2002).
24. Xinming Ou, Wayne F. Boyer, M.A.M.: A Scalable Approach to Attack Graph

Generation. In: 13th ACM conference on Computer and communications security. pp. 336–345 (2006).
25. Ammann, P., Wijesekera, D., Kaushik, S.: Scalable, graph-based network vulnerability analysis. Proc. 9th ACM Conf. Comput. Commun. Secur. - CCS '02. 217 (2002).
26. Ammann, P., Pamula, J., Ritchey, R., Street, J.: A host-based approach to network attack chaining analysis. In: Proceedings - Annual Computer Security Applications Conference, ACSAC. pp. 72–81 (2005).
27. Ingols, K., Lippmann, R., Piwowarski, K.: Practical attack graph generation for network defense. In: Proceedings - Annual Computer Security Applications Conference, ACSAC. pp. 121–130 (2006).
28. Ingols, K., Chu, M., Lippmann, R., Webster, S., Boyer, S.: Modeling modern network attacks and countermeasures using attack graphs. In: Proceedings - Annual Computer Security Applications Conference, ACSAC. pp. 117–126 (2009).
29. Kaynar, K., Sivrikaya, F.: Distributed Attack Graph Generation. IEEE Trans. Dependable Secur. Comput. 13, 519–532 (2016).
30. Xie, A., Zhang, L., Hu, J., Chen, Z.: A probability-based approach to attack graphs generation. In: 2nd International Symposium on Electronic Commerce and Security, ISECS 2009. pp. 343–347 (2009).
31. Ghosh, N., Ghosh, S.K.: A planner-based approach to generate and analyze minimal attack graph. Appl. Intell. 36, 369–390 (2012).
32. Phillips, C., Swiler, L.P.: A Graph-based System for Network-vulnerability Analysis. Proc. 1998 Work. New Secur. Paradig. 71–79 (1998).
33. Almohri, H.M.J., Watson, L.T., Yao, D., Ou, X.: Security optimization of dynamic networks with probabilistic graph modeling and linear programming. IEEE Trans. Dependable Secur. Comput. 13, 474–487 (2016).
34. Kun Bi, Dezhi Han, and J.W.: K Maximum Probability Attack Paths Dynamic Generation Algorithm. Comput. Sci. Inf. Syst. 13, 677–689 (2016).
35. Artz, M.L.: NetSPA : A Network Security Planning Architecture. Netw. Secur. 1–97 (2002).
36. Poolsappasit, N., Dewri, R., Ray, I.: Dynamic Security Risk Management Using Bayesian Attack Graphs. IEEE Trans. Dependable Secur. Comput. 9, 61–74 (2012).
37. Ou, X., Govindavajhala, S., Appel, A.W.: MulVAL: a logic-based network security analyzer. In: Proceedings of the 14th conference on USENIX Security Symposium - Volume 14. pp. 8–8 (2005).
38. Jajodia, S., Noel, S., O'Berry, B.: Topological analysis of network attack vulnerability. Manag. Cyber Threat. 247–266 (2005).
39. Barik, M.S., Mazumdar, C.: A Graph Data Model for Attack Graph Generation and Analysis. In: Communications in Computer and Information Science. pp. 239–250 (2014).
40. Common Weakness Enumerationm, CWE, Retrieved from http://cwe.mitre.org/. 20 April 2017
41. Common Vulnerabilities and Exposures, CVE, Retrieved from https://cve.mitre.org/. 20 April 2017