2019 International Conference on Computing, Networking and Communications (ICNC): Communications and Information Security Symposium

What To Do First: Ranking The Mission Impact Graph for Effective Mission Assurance

Pranavi Appana, Xiaoyan Sun, and Yuan Cheng
Department of Computer Science
California State University, Sacramento, CA 95819, USA
Email: {pranaviappana, xiaoyan.sun, yuan.cheng}@csus.edu

***Abstract*—Network attacks continue to pose threats to missions in cyber space. To prevent critical missions from getting impacted or minimize the possibility of mission impact, active cyber defense is very important. Mission impact graph is a graphical model that enables mission impact assessment and shows how missions can be possibly impacted by cyber attacks. Although the mission impact graph provides valuable information, it is still very difficult for human analysts to comprehend due to its size and complexity. Especially when given limited resources, human analysts cannot easily decide which security measures to take first with respect to mission assurance. Therefore, this paper proposes to apply a ranking algorithm towards the mission impact graph so that the huge amount of information can be prioritized. The actionable conditions that can be managed by security admins are ranked with numeric values. The rank enables efficient utilization of limited resources and provides guidance for taking security countermeasures.**

I. Introduction

Mission assurance in cyber space is still a challenge considering the increasing number of cyber attacks. The threats posed by attacks are not only towards network assets, but also missions. A cyber attack can possibly steal confidential information of missions, change mission flows, or even make the mission abort. Therefore, active cyber defense and cyber resilience are the prerequisites for successful mission assurance. That is, to defend computer networks against various attacks and ensure mission success, security analysts need to answer questions such as: 1) which vulnerabilities or security holes could be possibly leveraged to compromise the missions? 2) what are the potential impacts towards missions? 3) With limited resources, which security measures should be taken for mission assurance? For example, which security holes should be given higher priority and patched first?

A number of research efforts have been made to answer such questions. One major direction of efforts is attack graph [1], [2]. Today's computer networks are usually big in size and complicated in structure. With the large number of hosts and limited resources, maintenence becomes a big issue and vulnerabilities become inevitable. As a result, attackers might conduct multi-host and multi-stage attacks in order to achieve the attack goal. Given the vulnerabilities inside a network, attack graphs are able to generate the potential attack paths before the attack actually happens. Each attack path is a sequence of attack steps. The attack graph provides useful information with respect to how the vulnerabilities are leveraged to compromise the target machine. However, it is constrained to only asset level and cannot show the impact towards missions.

The other direction of efforts is mission impact assessment [3]. Different forms of mission dependency graphs are proposed to enable effective mission impact assessment. Most of these graphs aim to establish the connections across different abstract layers, such as the asset layer, the service layer, and the mission layer, by identifying the dependency relations among these layers. With the connections, the mission dependency graphs are able to show how the missions can be impacted by the lower-level assets. Since the possibility of multi-step attacks and their consequences are usually not considered in these graphs, Sun et al. [4] propose to integrate attack graphs and mission dependency graphs into a new form of mission impact graph. The mission impact graph is able to show the impact of an attack towards a mission even if the attack is not targeted at a host that has direct dependency relationship with the mission.

While mission impact graph is an effective tool for mission impact assessment, the graph might contain too much information for human analysts to comprehend. As a result, even if being presented with the graph, it's still very difficult for security analysts to make right recommendations with regard to mission assurance. Therefore, human analysts need a tool to help filter the large amount of information into a priority list of the actionable conditions, such as vulnerabilities that can be patched, connections that can be cut, or services that can be disabled. With this list, human analysts can use the limited resources in an efficient way. Sawilla and Ou [5] proposed an algorithm named Asset Rank, which is based on the PageRank algorithm [6], to identify the most important hosts and vulnerabilities for attackers to reach the target victim machine. In this paper, we extend the Asset Rank algorithm and apply it towards mission impact graphs so that the actionable conditions can be prioritized. Each actionable condition will be given a number showing its relative importance towards impacting a specific mission. Therefore, the generated priority list can be used to guide making recommendations of appropriate countermeasures in mission assurance.

This paper is an adaption of the work presented in [7]. It is organized as follows. Section II introduces the background knowledge of attack graphs and mission impact graphs. Section III discusses the approach and implementation details. Section IV briefly describes the attack and mission scenarios of the experiment in this paper, and then presents and analyzes the experiment results. Section V concludes the entire paper.

978-1-5386-9223-3/19/$31.00 ©2019 IEEE

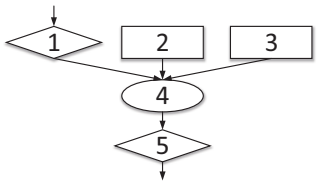

Figure 1: A part of a simplified attack graph

II. Background

A. Attack Graphs

Given the vulnerabilities in a network, an attack graph can provide the potential attack paths that an attacker might follow to reach an attack goal [1], [2]. Attack graphs are of two types, namely state enumeration attack graphs and dependency attack graphs. Any kind of attack graph consists of vertices and arcs. In the state enumeration attack graph, the entire network state is represented by a vertex and the state transitions that can be caused by an attack are represented by arcs. In a dependency attack graph, a system condition is represented by a vertex and the causality relations between vulnerabilities and exploitations are represented by arcs.

Traditional dependency attack graphs consist of two types of nodes, namely fact nodes and derivation nodes (also known as rule nodes, denoted with ellipse) [1], [2]. Fact nodes are then differentiated into primitive fact nodes (denoted with rectangles) and derived fact nodes (denoted with diamonds). In general, fact nodes are preconditions and derived fact nodes are post-conditions of derivation nodes [5].

As shown in Figure 1, if node 1 "the attacker has access to the server", node 2 "the server has a vulnerability that allows an attacker to gain access to a remote system and get root privileges" and node 3 "the server provides a service on a system" are all satisfied, then the rule in node 4 "remote exploit of a server program" will take effect and make node 5 "execution of arbitrary code on a server" become true. That is, an attacker is able to compromise the server.

B. Mission Impact Graphs

The purpose of the mission impact graph is to facilitate mission impact assessment. As proposed in [4], the mission impact graph unifies attack graphs, service dependency graphs, and mission dependency graphs into a new graphical model. It is a directed graph that also contains fact nodes and derivation nodes. Fact nodes represent a logical statement and derivation nodes represent an interaction rule applied for derivation. The edges represent causality relations among vertices or nodes. Derived fact node in a mission impact graph depends on one or more derivation nodes and derivation nodes in turn depend on fact nodes [4]. Similar as attack graphs, for the derivation node, fact nodes are preconditions and derived fact nodes will be post-conditions.

In mission impact graph implementation, interaction rules have been added in MulVAL to model causality relations between pre-conditions and post-conditions. For example, "attack on a server will impact the services provided by this server" could be represented in the form of an interaction rule using Datalog clauses as shown below [4]. This means, if a host provides a service, and attacker is able execute arbitrary code on the server, then service will be impacted by the attack.

```
interaction_rule(
   (serviceImpacted(Service, H, Perm):-
      hostProvideService(H, Service),
      execCode(H, Perm)),
   rule_desc('An compromised server will impact the dependent
service')).
```

III. Approach and Implementation

To identify the most important hosts and vulnerabilities for achieving specific attack goals, Sawilla and Ou [5] proposed an algorithm named AssetRank, which is based on the PageRank algorithm that ranks webpages according to their importance. In this paper, we extend the AssetRank algorithm and apply it to mission impact graphs to compute relative importance of graph vertices. Each graph vertex receives a numeric value as the rank to represent the importance of this vertex in terms of mission impact. The higher the rank of vertex, the more impact it may generate towards specific missions.

Google PageRank. The Google PageRank algorithm [6] calculates a webpages rank based on the number of web pages it is connected to. Pages will have higher ranks if they are linked to more pages. The importance of dependent webpages will also play a crucial role in deciding the rank of a webpage. Even if a webpage is pointed to by a few number of important pages, the webpage can have a higher rank than a webpage pointed to by a large number of unimportant pages. Google PageRank algorithm can be used in other forms of dependency graphs. Attack graphs and mission impact graph are both different forms of dependency graphs.

AssetRank. In this paper, we extend the AssetRank algorithm proposed in [5] for ranking vertices in mission impact graph. Based on the PageRank algorithm, the AssetRank algorithm is designed to rank the importance of vertices in attack graphs. Since vertices in an attack graph represent different network conditions or status, the rank of a vertex represents its importance in achieving a specific attack goal. The AssetRank algorithm is implemented in a way that suits the semantics of dependency attack graphs.

A graph G can be shown as $G = (V, A, f, g, h)$ [5]. V, A, f, g and h are set of vertices, arcs, mapping of weights to vertices, weights to arcs and vertices to their type, respectively. Out-neighborhood and in-neighborhood of v are:

$N^+(v) = \{w \in V: (v, w) \in A\}$ [5]

$N^-(v) = \{u \in V: (u, v) \in A\}$ [5]

$|N|$ is cardinality of a set. Sum of all vertex weights need to be 1. Sum of all arc weights need to be 1 if v is of type OR, and number of out-neighborhoods if v is an AND vertex. When v is a sink, sum should be zero.

$$\Sigma_{w \in N^+(v)} g(v, w) = \begin{cases} 1, if h(v) = OR \\ |N^+(v)|, if h(v) = AND \end{cases} \quad [5]$$

The logic behind asset ranking is that every vertex is given a numeric value which represents its level of importance. The vertex inherits some value from vertices depending on it, and

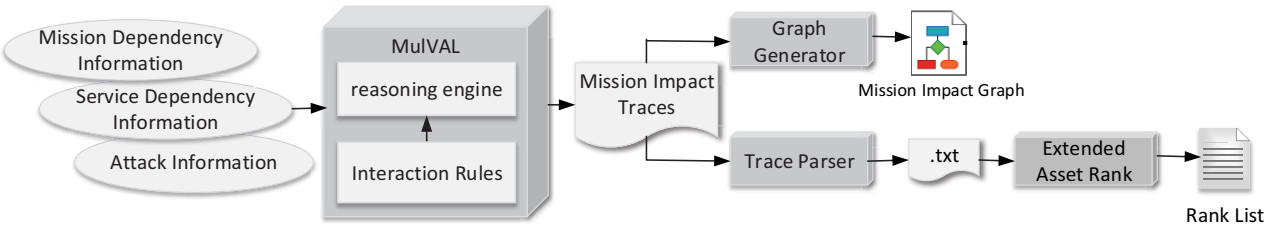

Figure 2: The Rank Generation of Mission Impact Graph

the remaining value is of itself. A vertex also receives value from its dependents. The numeric value is made of two parts, intrinsic value and the value that a vertex receives from its dependents.

The equation calculating rank of vertex v is:

$x_v = \delta \Sigma_{u \in N^-(v)} g(u,v) x_u + (1-\delta) f(v)$ [5]

The damping factor is represented by δ, which sets the ratio between rank value received from dependents and the intrinsic value to x_v.

All the vertex ranks x_v are gathered into a vector X. All the dependency relationships between the vertices are put into a weighed adjacency matrix represented as D, in a way such that $D_{vu} = g(u,v)$ [5]. Then Jacobi method of iteration is applied to the sequence and the resulting equations are shown as:

$X = \delta DX + (1-\delta) IV$ [5]

$X_t = \delta DX_{t-1} + (1-\delta) IV$ [5]

There is a difference in computing an AND vertex and an OR vertex. AssetRank algorithm is proposed to address this change [5], which makes it different from the Google PageRank algorithm where all pages are considered of the type OR. When a vertex is of the type AND, instead of splitting its value among dependents, the value will be replicated to all its dependents. The equation with inclusion of an AND type will be normalized at each iteration, so that the values of vertices are still summed to 1.

Therefore, the final equation for our computation becomes:

$X_{t'} = \delta DX_{t-1} + (1-\delta) IV$ [5]

$X_t = \frac{1}{||X_{t'}||} X_{t'}$ [5]

Implementation. Since the mission impact graph shares similar semantics as the attack graph, we can extend the AssetRank algorithm to rank the mission impact graph. The algorithm is extended in two aspects. First, the extended ranking algorithm is able to take multiple vertices as the mission impact goals. This means the rank of vertices can be generated by treating multiple missions as attackers' target missions. Second, the extended ranking algorithm can assign different weights to missions according to factors such as mission importance. Mission importance can affect the rank value of vertices. A high rank condition may get a lower rank if the mission related to the condition becomes less important.

Figure 2 shows the flow of ranking mission impact graph. The mission impact traces is first generated in MulVAL [2] by taking mission dependency information, service dependency information, and attack information as input. MulVAL is an attack graph generation tool and has been extended in [4] to enable automatic generation of mission impact graphs. With the trace file, security analysts can generate mission impact graph using a graph generator such as graphviz [8], or rank the vertices by applying the ranking algorithm.

The ranking takes two steps. Step 1, the trace file, which is usually a .dot file, will be parsed using the Perl script. The output will be a text file that consists of information like node, type and the dependencies. Step 2, the parsed .txt file will be sent as an input to the ranking algorithm, which is implemented as a Java program. The Java program will convert the dependencies among nodes into an adjacency matrix. Depending on out-neighbors and in-neighbors of each vertex as well as the vertex weight, we will compute ranks based on the asset ranking approach. The Java code will output results in the form of a numeric value for each node, which represent the importance of the node in our mission impact graph.

IV. Experiment

A. Attack and Mission Scenario

The mission impact scenario shown in Figure 3 is taken from the work by Sun et al. [4], which consists of three enterprise networks existing on the same cloud: a startup company A, a medical group B, and a vaccine supplier C. Vaccine supplier C only accepts client requests from trusted IPs. The main reason is that along with producing vaccines based on existing formulas, C and its collaborators are also developing a new type of vaccine whose formula is still highly confidential. Medical group B is a trusted client to vaccine supplier C. Startup company A's webserver and medical group B's database server are both virtual machines, and they exist on the same physical host. This co-residency relationship can be taken advantage of by the attackers.

There are two missions in this scenario, namely Bm1 and Cm1. Bm1 is the mission for medical group B, which is to provide medical services to all of its patients. Cm1 is the mission for vaccine supplier C, which is to supply vaccines to authorized medical groups and to develop the new vaccines together with its collaborators. Mission Bm1 includes tasks such as making patient appointments, accessing medical records, ordering shots or medicine, administering shots, and updating medical records, etc. Mission Cm1 performs tasks such as asking for login ID and password, authenticate the entered ID and password. If the user is the medical group, the

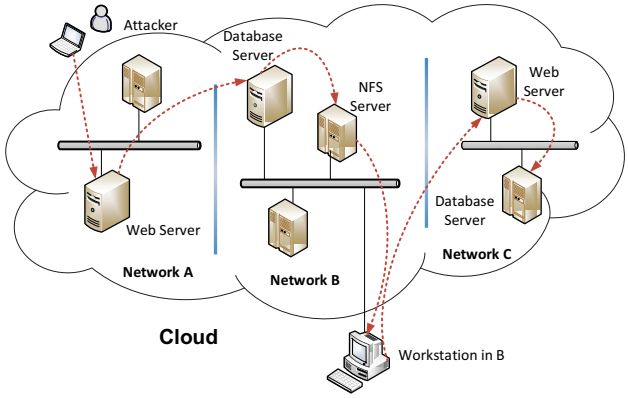

Figure 3: The Attack and Mission Scenario

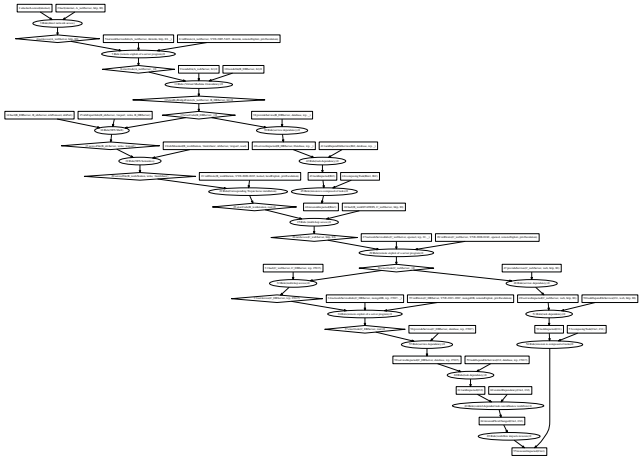

Figure 4: The Generate Mission Impact Graph [4]

user can order vaccines. If the user is a collaboration partner, the user is able to check and update vaccine information etc.

In the attack scenario [4], the attacker Mallory, who could be from a rival company, is trying to steal the confidential formula for the new vaccine from supplier C. To enter into the network, Mallory first attacks A's webserver by exploiting the vulnerability in tikiwiki 1.9.8. To take over B's database server, Mallory gets hold of the co-residency relationship based on a side channel attack in the cloud. Mallory then uploads a software named tool.deb, which is actually a Trojan horse, to a shared directory. The directory is shared by all hosts inside B's network. B's database server was not supposed to have write permissions to this shared directory (/exports), it was given write permission due to a configuration error of the NFS export table. Without knowing that the tool.deb file is a Trojan, a workstation user from B, downloads it from NFS server and installs it, which creates an unsolicited connection to Mallory. Since medical group B is a trusted client of vaccine supplier C, the workstation in B has access to C's webserver. Mallory manages to take over Cs webserver via a brute-force key guessing attack. To further get credential information from an employee login table, Mallory tries to compromise Vaccine supplier C's MongoDB database server. Mallory now logs in into vaccine supplier C's webserver as a collaborator and accesses project proprietary documents and collects formula related information.

To make this paper self-contained, we include the generated mission impact graph (Figure 4) from [4] into this paper. The graph shows how the attacks at the asset layer can generate impact towards missions. Mission impact graphs extend attack graphs in a way that prediction of attack paths directly enables prediction of potential mission impact. However, the information presented in this graph is very limited. By only referring to this graph, it's still very difficult for human analysts to make recommendations of appropriate countermeasures for mission assurance. Therefore, we apply the extended AssetRank algorithm towards Figure 4 and present the result in Section IV-B.

B. Result Analysis

The extended AssetRank algorithm has two merits. First, it is able to take multiple mission impact goals. For example, in the scenario discussed in Section 3, both mission Bm1 and Cm1 can be treated as the victim missions that attackers want to affect. Second, it can assign different weight to different missions based on specific factors. For example, if one mission is more important than another one, the actionable conditions that are related to the more important mission may get higher ranks in the final rank results.

We conducted two sets of experiments to demonstrate the effectiveness of our approach. In the first set, both mission Bm1 and mission Cm1 are of equal importance and get the same weight. In the second set, mission Cm1 is more important than Bm1. The importance ratio for Bm1 and Cm1 is 10:90. We applied the extended AssetRank algorithm to Figure 4 and computed the ranks of all the nodes in this graph. Although every node can get a rank number through the ranking algorithm, we only focus on the actionable conditions that security admins are able to make changes of, such as the configuration options, the vulnerabilities, etc.

Scenario with Equal Importance Ratio. Table I shows the rank values of actionable conditions given equal importance to mission Bm1 and Cm1. The "resideOn", "vulExists" and "hacl" predicates are a few of the categories that the system administrators are able to change. The highest ranked node is node 38, which means B's Database Server provides Database service. If the Database service is not enabled, then the service won't be affected. The countermeasure could be taking the database service down, which is usually not recommended. Since this action is not very affordable, security admin can then check the nodes with second highest ranks. Node 12, 9 and 10 are nodes that enable the virtual machine co-residency relationship between startup company A's webserver and medical group B's database server. This is consistent with common sense as the attackers can take advantage of virtual machine co-residency to impact mission Bm1 and Cm1. The countermeasures could be eliminating the co-residency relationship by moving one of the virtual machines. This is very feasible comparing to the countermeasure of taking database service down. From the results, we can observe that the algorithm can pick out the most important actionable conditions among all the other nodes related to the missions

Table I: Rank of Actionable Conditions with Equal Importance Ration of Missions

Vertex	Node name	Rank
38	provideService(B_DBServer,database,tcp,_)	0.020275
12	stealthyBridgeExists(A_webServer,B_DBServer,h1):0	0.017724
9	resideOn(A_webServer,h1):0	0.011643
10	resideOn(B_DBServer,h1):0	0.011643
47	provideService(C_webServer,web,http,80)	0.010138
27	networkServiceInfo(C_webServer,openssl,tcp,22,_)	0.008548
28	vulExists(C_webServer,'CVE-2008-0166',openssl,remoteExploit,privEscalation)	0.008548
5	networkServiceInfo(A_webServer,tikiwiki,http,80,_)	0.007649
6	vulExists(A_webServer,'CVE-2007-5423',tikiwiki,remoteExploit,privEscalation)	0.007649
56	provideService(C_DBServer,database,tcp,27017)	0.00666
24	hacl(B_workSTATION,C_webServer,http,80)	0.005615
2	hacl(internet,A_webServer,http,80)	0.005025
34	networkServiceInfo(C_DBServer,mongoDB,tcp,27017,_)	0.004375
35	vulExists(C_DBServer,'CVE-2013-1892',mongoDB,remoteExploit,privEscltn)	0.004375
21	vulExists(B_workStation,'CVE-2009-2692',kernel,localExploit,privEscalation)	0.003689
31	hacl(C_webServer,C_DBServer,tcp,27017)	0.002874
18	nfsMounted(B_workStation,'/mnt/share',nfsServer,'/export',read)	0.002423
14	hacl(B_DBServer,B_nfsServer,nfsProtocol,nfsPort)	0.001592
15	nfsExportInfo(B_nfsServer,'/export',write,B_DBServer)	0.001592

Table II: Rank of Actionable Conditions with 10:90 Importance Ration of Missions

Vertex	Node name	Rank
47	provideService(C_webServer,web,http,80)	0.018109
27	networkServiceInfo(C_webServer,openssl,tcp,22,_)	0.015215
28	vulExists(C_webServer,'CVE-2008-0166',openssl,remoteExploit,privEscalation)	0.015215
56	provideService(C_DBServer,database,tcp,27017)	0.011871
24	hacl(B_workSTATION,C_webServer,http,80)	0.009973
34	networkServiceInfo(C_DBServer,mongoDB,tcp,27017,_)	0.007781
35	vulExists(C_DBServer,'CVE-2013-1892',mongoDB,remoteExploit,privEscltn)	0.007781
21	vulExists(B_workStation,'CVE-2009-2692',kernel,localExploit,privEscalation)	0.006538
12	stealthyBridgeExists(A_webServer,B_DBServer,h1):0	0.005533
31	hacl(C_webServer,C_DBServer,tcp,27017)	0.005101
18	nfsMounted(B_workStation,'/mnt/share',nfsServer,'/export',read)	0.004285
38	provideService(B_DBServer,database,tcp,_)	0.004024
9	resideOn(A_webServer,h1):0	0.003627
10	resideOn(B_DBServer,h1):0	0.003627
14	hacl(B_DBServer,B_nfsServer,nfsProtocol,nfsPort)	0.002809
15	nfsExportInfo(B_nfsServer,'/export',write,B_DBServer)	0.002809
5	networkServiceInfo(A_webServer,tikiwiki,http,80,_)	0.002377
6	vulExists(A_webServer,'CVE-2007-5423',tikiwiki,remoteExploit,privEscalation)	0.002377
2	hacl(internet,A_webServer,http,80)	0.001558

Bm1 and Cm1.

Scenario with 10:90 Importance Ratio. Table II shows the rank values of actionable conditions given the importance ratio of 10% to Bm1 and 90% to Cm1. From the results, we can observe that when 90% of importance is assigned to mission Cm1, the algorithm suggests to either stop the web service and ssh service on C's webserver (node 47 and 27), or patch the vulnerability on this machine first. Once the vulnerability is patched, the mission impact graph is only be left with mission Bm1. Mission Cm1 will become safe. The results show that the algorithm is able to correctly identify the most important actionable conditions when given different mission importance. The actionable conditions that can impact more important missions stand out and get higher rank values.

V. Conclusion

Mission impact graphs provide great insights in identifying possible attack paths that might impact missions. However, due to its size and complexity, it might still be difficult for human analysts to fully understand the details and take appropriate counter measures to ensure mission success. Therefore, further analysis is needed to prioritize actionable conditions such as vulnerabilities existing in the network, configuration options, etc. The list of prioritized items can enable efficient utilization of limited resources and provide guidance to security practice. This paper extends the AssetRank algorithm and applies it to the mission impact graph. By providing a numeric value of importance (Rank), we are able to identify the most important actionable conditions that can impact specific missions. This will be very useful for users of enterprise networks to better understand the security risks and make recommendations of proper countermeasures.

References

[1] X. Ou, W. F. Boyer, and M. A. McQueen. A scalable approach to attack graph generation. ACM CCS, 2006.

[2] X. Ou, S. Govindavajhala, and A. W. Appel. MulVAL: A Logic-based Network Security Analyzer. USENIX security, 2005.

[3] Gabriel Jakobson. Mission Cyber Security Situation Assessment Using Impact Dependency Graphs.

[4] Xiaoyan Sun, Anoop Singhal, Peng Liu. Towards Actionable Mission Impact Assessment in the Context of Cloud Computing, 2017.

[5] Reginald Sawilla, Xinming Ou. Googling Attack Graphs, Defence Research and Development, Canada, September 2007.

[6] Page, Lawrence, Brin, Sergey, Motwani, Rajeev, and Winograd, Terry. The PageRank Citation Ranking: Bringing Order to the Web, Technical Report Stanford Digital Library Technologies Project, 1998.

[7] Pranavi Appana. Mission impact analysis in enterprise networks. http://csus-dspace.calstate.edu/handle/10211.3/198818

[8] GraphViz. http://www.graphviz.org/.

[9] Kyle Ingols, Lippmann, Richard, and Piwowarski, Keith. Practical Attack Graph Generation for Network Defense, 22nd Annual Computer Security Applications Conference, Florida, 2006.

[10] Emden R. Gansner Stephen C. North. An open graph visualization system and its applications to software engineering, SOFTWARE - PRACTICE AND EXPERIENCE, vol. 30, no. 11, pp. 1203-1233, 2000.