Change-Point Detection in Enterprise Attack Surface for Network Hardening

Ghanshyam S. Bopche and Babu M. Mehtre

Abstract Applications of change-point detection typically originate from the perspective of enterprise network security and network monitoring. With the ever-increasing size and complexity of enterprise networks and application portfolios, network attack surface keeps changing. This change in an attack surface is detected by identifying increase or decrease in the number of vulnerabilities at network level. Vulnerabilities when exploited successfully, either provide an entry point to an adversary into the enterprise network or can be used as a milestone for staging multi-stage attacks. In this paper, we have proposed an approach for change-point detection in an enterprise network attack surface. In this approach, a sequence of static attack graphs are generated for dynamic (time varying) enterprise network, and successive graphs in a sequence are compared for their dissimilarity for change-point detection. We have presented a small case study to demonstrate the efficacy and applicability of the proposed approach in capturing a change in network attack surface. Initial results show that our approach is capable of capturing the newly introduced vulnerabilities into the network and is able to differentiate these vulnerabilities for efficient network hardening.

Keywords Attack surface · Attack graph · Change-point detection · Network security and protection · Security metric · Similarity measures

G.S. Bopche (✉) · B.M. Mehtre
Center for Information Assurance & Management (CIAM),
Institute for Development and Research in Banking Technology (IDRBT),
Castle Hills, Masab Tank 500057, Hyderabad, India
e-mail: ghanshyambopche.mca@gmail.com

B.M. Mehtre
e-mail: mehtre@gmail.com

G.S. Bopche
School of Computer and Information Sciences (SCIS),
University of Hyderabad, Gachibowli 500046, Hyderabad, India

A. Nagar et al. (eds.), *Proceedings of 3rd International Conference on Advanced Computing, Networking and Informatics*, Smart Innovation, Systems and Technologies 43, DOI 10.1007/978-81-322-2538-6_49

1 Introduction

With the constant evolution of TCP/IP protocols and hence enterprise networks, there may be new plausible security risks as new servers, workstations, or services are added to the network. These network assets are increasingly vulnerable and always a soft target of sophisticated cyber attacks from potential adversaries such as hackers, corporate competitors, disgruntled employees, government agencies, etc. In order to keep the security posture of an enterprise network up to date, enterprise networks need to be scanned and hardened regularly.

Present day vulnerability scanners such as, *Nmap*, *Nessus*, *Retina*, *GFI LanGuard*, etc. identify vulnerabilities in isolation, i.e., vulnerabilities local to a system only. For an enterprise network of reasonable size, vulnerability scanners can report a large number of vulnerabilities. From the defender's standpoint, an administrator has to identify the vulnerabilities, which really matters in securing enterprise network. In other words, administrator has to identify vulnerabilities that allow an adversary to enter the network or the vulnerabilities that can be used as a milestone for staging multistage attacks against the enterprise network.

An attack graph- a "multihost, multistage" attack detection technique derives all possible attack paths available to an adversary. In order to do this, a network model description such as vulnerability information, network configuration, and network reachability information is used. The cause-consequence relationship between the extant vulnerabilities is taken into account to draw multistage correlated attack scenarios (that are used by an adversary to get incremental access to enterprise critical resources). Understanding such relationship is vital for optimal placement of the security countermeasures and hence for efficient network hardening. Vulnerability scanners are not capable of finding such correlation among the vulnerabilities. Further, prioritization of vulnerabilities for efficient network hardening based on a scanning report alone is highly impossible and not a viable solution in terms of time, and effort.

In this paper, we have used an attack graph-based approach for the detection and prioritization of vulnerabilities, which really plays a key role during network compromise. We have taken a snapshot of a network at time t and generated an attack graph. It is obvious that, it shows the vulnerabilities and their dependency on each other, which may lead to network compromise when exploited successfully. The graph-assisted metrics proposed in literature, for example shortest path metric [1], the number of paths metric [2], mean of path lengths [3], and others [4, 5] can be used to identify attack paths/attack scenarios of special interest. Even though administrator is aware about these attack paths and causal vulnerabilities, she cannot patch/fix all because of the various constraints. The constraint may be countermeasure cost, limited security budget, unavailability of patch/workarounds, patch time, etc. Unpatched vulnerabilities will straightaway appear in the attack graph generated for the same network over time Δt called sampling interval. The administrator already knows external causes of these vulnerabilities. She has to worry about the new vulnerabilities that will be introduced into this attack

graph/enterprise network over time Δt. For an enterprise network of reasonable size, an attack graph is of enormous size. Searching newly introduced vulnerabilities manually in this graph is not feasible in real time. Again, identifying the most relevant vulnerabilities for efficient network hardening is also poses great difficulty/challenge. This challenge motivates our work. We used graph similarity-based approach to solve above said problem. We have compared the attack graphs generated at time t and $t + \Delta t$ for their similarity (may not be structural) in terms of common nodes and edges and by applying some heuristics we have identified the most relevant among the newly introduced vulnerabilities.

The remainder of this paper is organized as follows: Sect. 2 presents an attack graph model. Section 3 discusses our method of the change-point detection in terms of number of newly introduced vulnerabilities in a successive attack graph. A case study is presented in Sect. 4 to demonstrate the usefulness of our approach. Finally, Sect. 5 concludes the paper.

2 Preliminaries

An attack graph for the enterprise network is a directed graph representing prior knowledge about vulnerabilities, their dependencies, and network connectivity [6]. In this paper, we have generated labeled, goal-oriented attack graphs for the enterprise network. Exploits and security conditions are the two types of nodes in the attack graph. Here, exploit represents an adversary action on the network host in order to take advantage of extant vulnerability. Security conditions represent properties of system or network relevant for successful execution of an exploit. The existence of a host vulnerability, network reachability, and trust relationship between two hosts are the kind of security conditions required for successful exploitation of vulnerability on a remote host.

Exploits and conditions are connected by directed edges. No two exploits or two security conditions are directly connected. Directed edge from security condition to an exploit represent the *require relation*. It means, an exploit cannot be executed until all the security conditions have been satisfied. An edge from an exploit to a security condition indicates the *imply relation* [7]. It means, successful execution of an exploit will create few more conditions. Such newly created security conditions, i.e., post-conditions act as a pre-condition for other exploits. With the perception of an attack graph discussed above, Wang et al. [8] formally defined an attack graph as an exploit-dependency graph as follows:

Definition 1 (*Attack Graph Model*) Given a set of exploits e, a set of conditions c, a require relation $R_r \subseteq c \times e$, and an imply relation $R_i \subseteq e \times c$, an attack graph G is the directed graph $G(e \cup c, R_r \cup R_i)$, where $(e \cup c)$ is the vertex set and $(R_r \cup R_i)$ is the edge set [8].

As evident from the Definition 1, an attack graph G is a directed bipartite graph with two disjoint sets of vertices namely, exploit and condition. The edge set consist of two types of edges, namely, *require edge* and *imply edge*. As an important feature of an attack graph, require relation, i.e., R_r should be always conjunctive, whereas the imply relation, i.e., R_i should be always disjunctive [8]. The conjunctive nature of the conditions implies an exploit cannot be executed until all of its preconditions have been satisfied. An imply edge should identify those conditions which can be generated after the successful execution of an exploit. Introduction of a new vulnerability in an enterprise network is shown in an attack graph by the exploit, it's pre-conditions and post-condition.

3 Change-Point Detection

Change-point detection in the attack surface of an enterprise network can be determined by representing a given network, observed at time t and $t + \Delta t$, by attack graphs G and G', respectively. An attack surface represents, the set of ways an adversary can compromise an enterprise network [9]. Δt an arbitrary sampling interval depends on the time required for gathering vulnerability information, network configuration details and construction of an attack graph. It is an important parameter in security monitoring, defines how often an attack graph is constructed and governs the type of vulnerability (attack) can be detected. For a given window of time W and for a given enterprise network, attack graphs are generated (at discrete instants of time depending on Δt). This leads to the sequence of an attack graphs. Then the consecutive attack graphs in a sequence are analyzed for the detection of change in attack surface.

The algorithm presented in this paper works on labeled attack graphs. Let L_V and L_E denote the finite set of vertex and edge labels in an attack graph, respectively.

Definition 2 An attack graph G is a four-tuple $G = (V, E, \rho, \mu)$, where

- V is a finite set of vertices i.e., $V = e \cup c$.
- $E \subseteq V \times V$ is a set of Edges i.e., $E = R_r \cup R_i$
- $\rho : V \rightarrow L_V$ is a function assigning labels to the vertices
- $\mu : E \rightarrow L_E$ is a function assigning labels to the edges

Here, L_V and L_E represents the set of symbolic labels uniquely identifying each node and each edge, respectively in an attack graph G. Application of a minimum spanning tree algorithm (MST) on both G and G', gives the nodes and edges present in each graph. Once nodes and edges with respect to each input attack graph are identified, following three cases are considered:

1. $G' \backslash G$: nodes and edges unique to G' only, i.e., $G' - G$. These nodes represent newly introduced vulnerabilit(y)ies in the network.

2. $G \backslash G'$: nodes and edges unique to G only, i.e., $G - G'$. These nodes and edges belong to the vulnerabilities that are already patched.
3. $G \cap G'$: nodes and edges that are common to both G and G'. Common nodes and edges appeared because of the persistence of some vulnerabilities in an enterprise network both at time t and $t + \Delta t$. These vulnerabilities and their external causes (i.e., preconditions) are already known to the administrator but because of some constraints like limited security budget, unavailability of patches etc. those vulnerabilities are unpatched.

Case 1 is of special interest to an administrator since it gives information about the newly introduced vulnerabilities into the network. Our goal is to analyze these vulnerabilities for their use by an adversary in multistage attacks. Case 1 gives the node set N consisting of exploits, pre-conditions, post-conditions, i.e., $N = e \cup c$ and edge set M consisting of require relation and imply relation, i.e., $R_r \cup R_i$ with respect to the newly introduced vulnerabilities. The edge set M gives more information about the vulnerability dependency and hence is of special interest.

For each exploit there should be two or more pre-conditions need to be satisfied conjunctively. It means second node/vertex in two or three require relations (i.e., edges in set M), is common. This common vertex represents an exploit and the node preceding to it in those edges represent the pre-conditions required for its exploitation. Removal of any one of these pre-conditions can stop an exploit from executing. If the exploit is the first vertex in one or more imply relations (edges), then the second vertex of those edges represent the post-condition of an exploit. This post-condition may act as a precondition for other exploits. An approach of identifying and differentiating new vulnerabilities introduced in an enterprise network during the time period Δt is given in algorithm 1.

4 Case Study

In this section, a case study is presented to detect the vulnerability change in an attack surface of an enterprise network by means of dissimilarity between consecutive attack graphs generated for the same enterprise network at time t and $t + \Delta t$. From the context of an input attack graphs, it is shown that the obtained results provide unique security relevant information, which will enhance the security administrator's ability in hardening network security more efficiently.

A network similar to [10] has been considered as the test network. Topology of the test network is given in Fig. 1. There are 4 hosts in the network viz. $Host_0$ (H_0), $Host_1$ (H_1), $Host_2$ (H_2) and $Host_3$ (H_3). The description of each host is given below:

- H_0: a Web Server (Windows NT 4.0)
- H_1: a Windows Domain Server (Windows 2000 SP1)
- H_2: a Client (Windows XP Pro SP2)
- H_3: a Linux Server (Red Hat 7.0)

Algorithm 1 Change-point Detection in Attack surface

Input:
$G \rightarrow$ an attack graph for an enterprise network at time t
$G' \rightarrow$ an attack graph for the same network at time $t + \Delta t$
$\Delta t \rightarrow$ a sampling interval
Output:
$Exploit \subseteq (V' - V) \rightarrow$ is a set of new exploits/ vulnerabilities added to the G over the time interval Δt
$Initial \subseteq Exploit \rightarrow$ is a set of new exploits that can initiate new attack paths.
$Middle \subseteq Exploit \rightarrow$ is a set of new exploits that can be part of attack paths but does not initiate a new attack path
$PreCondn[i] \rightarrow$ pre-conditions of an exploit $i \in Exploit$

1: $\langle V, E \rangle \leftarrow$ MST (G) ▷ Apply minimum spanning tree algorithm (MST) on G to identify all the unique nodes and edges in G
2: $\langle V', E' \rangle \leftarrow$ MST (G') ▷ Apply MST on G' to identify all the unique nodes and edges in G'
3: $\langle (V' - V), (E' - E) \rangle \leftarrow G' \backslash G$ ▷ Compute $(G' - G)$. It gives vertices, i.e., $(V' - V)$ and edges, i.e., $(E' - E)$ belong to G' only
4: **if** $(G' \backslash G = \phi)$ **then**
5: Print: "no new vulnerability introduced into the enterprise network"
6: **else**
7: **for all** $v \in (V' - V)$ **do**
8: **if** $(v == j)$ for two or more $(i, j) \in (E' - E)$ **then**
9: $Exploit \leftarrow v$ ▷ v is an exploit
10: $PreCondn \leftarrow i$ ▷ i is the pre-condition for an exploit v
11: **end if**
12: **end for**
13: **for all** $e' \in Exploit$ **do**
14: **if** $(e' == i)$ for one or two $(i, j) \in (E' - E)$ **then**
15: $PostCondn \leftarrow j$ ▷ j is the post-condition of an exploit e'
16: **end if**
17: **end for**
18: **for all** $(i, j) \in (E' - E)$ **do**
19: **if** $i, j \in (Exploit \cup PostCondn)$ **then**
20: $E'' = (i, j)$
21: **end if**
22: **end for**
23: **for all** $i \in Exploit$ in G'' **do** ▷ G'' is constructed out of E''. G'' is a bi-partite graph and may have one or more connected components
24: find indegree $\delta(i)$
25: **if** $(\delta(i) == 0)$ **then**
26: $Initial \leftarrow i$ ▷ i is the first exploit in an attack sequence
27: **else**
28: $Middle \leftarrow i$ ▷ i is the middle exploit in an attack paths
29: **end if**
30: **end for**
31: **for all** $i \in Initial$ **do**
32: Find $PreCondn[i]$ ▷ Identify pre-conditions of an exploit, which can initiate a new attack sequence
33: **end for**
34: **for all** $i \in Middle$ **do**
35: Find $PreCondn[i]$ ▷ Identify pre-conditions of an exploit, which does not initiate a new attack sequence but can be part of other attack sequence
36: **end for**
37: **end if**

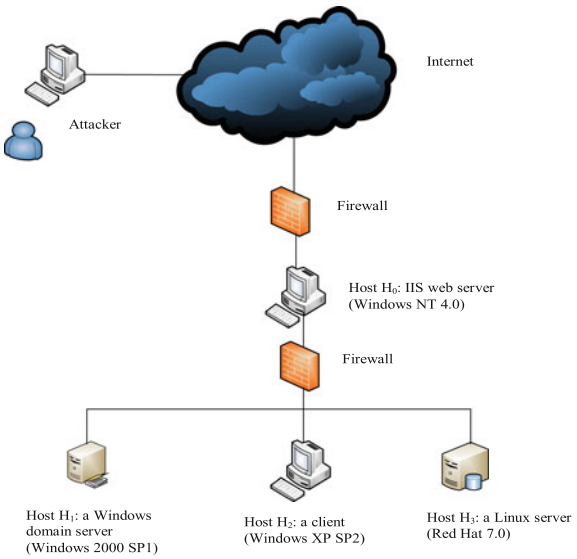

Fig. 1 Network configuration [10]

Here host H_3 is the target machine for an attacker and *MySQL* is the critical resource running over it. The attacker is an entity with malicious intent from outside the internal network. Here the attacker's intention is to gain root-level privileges on host H_3. The job of a firewalls is to separate internal network from the Internet and the connectivity-limiting firewall policies for the network configuration are given in Table 2. Tables 1 and 3 shows the system characteristics for the hosts available in the network at time t and $t + \Delta t$, respectively. Such kind of data is available in public vulnerability databases viz. *National Vulnerability Database* (*NVD*), *Bugtraq*, *Open Source Vulnerability Database* (*OSVDB*) etc. Here external firewall allows any external host to only access services running on host H_0. Connections to all other services/ports available on other hosts are blocked. Host's within the internal network are allowed to connect to only those ports specified by the

Table 1 System characteristics for network configuration at time t [10]

Host	Services	Ports	Vulnerabilities	CVE IDs
H_0	IIS web service	80	IIS buffer overflow	CVE-2010-2370
H_1	ssh	22	ssh buffer overflow	CVE-2002-1359
	rsh	514	rsh login	CVE-1999-0180
H_2	rsh	514	rsh login	CVE-1999-0180
H_3	LICQ	5190	LICQ-remote-to-user	CVE-2001-0439
	Squid proxy	80	squid-port-scan	CVE-2001-1030
	MySQL DB	3306	local-setuid-bof	CVE-2006-3368

Table 2 Policies for connectivity-limiting firewall

Host	Attacker	H_0	H_1	H_2	H_3
Attacker	Localhost	All	None	None	None
H_0	All	Localhost	All	All	Squid
					LICQ
H_1	All	IIS	Localhost	All	Squid
					LICQ
H_2	All	IIS	All	Localhost	Squid
					LICQ
H_3	All	IIS	All	All	Localhost

Table 3 System characteristics for network configuration at time $t + \Delta t$ [10]

Host	Services	Ports	Vulnerabilities	CVE IDs
H_0	IIS web service	80	IIS buffer overflow	CVE-2010-2370
	ftp	21	ftp buffer overflow	CVE-2009-3023
H_1	ftp	21	ftp rhost overwrite	CVE-2008-1396
	ssh	22	ssh buffer overflow	CVE-2002-1359
	rsh	514	rsh login	CVE-1999-0180
H_2	Netbios-ssn	139	Netbios-ssn nullsession	CVE-2003-0661
	rsh	514	rsh login	CVE-1999-0180
H_3	LICQ	5190	LICQ-remote-to-user	CVE-2001-0439
	Squid proxy	80	squid-port-scan	CVE-2001-1030
	MySQL DB	3306	local-setuid-bof	CVE-2006-3368

connectivity limiting firewall policies as shown in Table 2. In Table 2, *All* specifies that source host may connect to any port on a destination host in order to have access to the services running on those ports. *None* indicates that source host is prevented from having access to any port on the destination host [10].

An attack graph for the network configuration at time t and $t + \Delta t$ is shown in Fig. 2. These graphs for the same enterprise network at different instant of time (for the sampling interval Δt) are generated using model checking-based tool called SGPlan [11, 12]. Existing vulnerabilities in the test network are logically combined to generate different attack scenario, which in turn represents different attack paths. Attack graph is generated by collapsing several such attack paths for the same initial and goal condition. As shown in Fig. 2, nodes in both the attack graphs represent an exploit, required pre-conditions and implied post-conditions. Exploits are shown by a circle and named by alphabets.

As evident from the Fig. 2, number of attack paths for an attack graph at time t are only 2, whereas it is 16 for an attack graph at time $t + \Delta t$. It is because of an increase in the number of vulnerable services in the enterprise network within the sampling interval Δt. Vulnerabilities appeared in an attack graph (i.e., in an

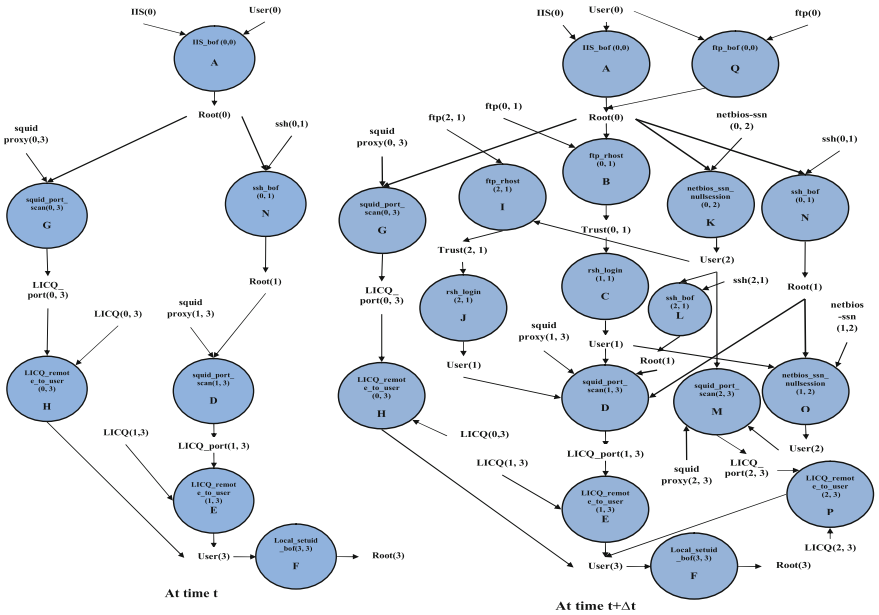

Fig. 2 Attack graph for the network configuration at time t and $t + \Delta t$

enterprise network) at time t are not patched and hence will appear straightaway in an attack graph at time $t + \Delta t$. The external causes for these unpatched vulnerabilities are already well known to the administrator. She has to worry about the new vulnerabilities introduced in the network.

Change point detection algorithm 1 proposed in Sect. 3 detects the new vulnerabilities appeared in attack graph at time $t + \Delta t$. It gives an exploit set $Exploit = \{B, C, I, J, K, L, M, O, P, Q\}$ and respective post-condition set $PostCondn$. From these two sets, we have derived a bipartite graph representing chains of exploits separated by post-conditions. Indegree, i.e., δ for each exploit in this graph is calculated. Exploit with indegree zero, i.e., $\delta = 0$, are responsible for initiating new attack paths. Remaining exploits with indegree value other than zero can be used by an attacker as a milestone for staging multi-stage attacks. From the defender's standpoint, the administrator must decide on which exploits to be focused on for efficient network hardening. Accordingly, pre-conditions must be identified and disabled using one or more security countermeasures to remove attack path or break an attack path in between in order to stop an adversary from reaching target machine. Algorithm 1 successfully identifies those preconditions and assist administrator in hardening network. In our case study, exploit Q provide a new way to an adversary in defeating network security and remaining exploits, i.e., B, C, I, J, K, L, M, O, P can be used by an adversary as a milestone. If

vulnerable *ftp* service on the host H_0 is patched or stopped whichever feasible, can remove 8 attack paths from the second attack graph. Remaining attack paths are removed or can be stopped in between by preventing the pre-conditions of the responsible exploits.

5 Conclusion and Future Work

In this paper, we have proposed a new approach for change-point detection in a vulnerability state of an enterprise network. An attack graph at fixed sampling interval is generated for an enterprise network and compared with the previous one for finding dissimilarity in terms of newly introduced vulnerabilities. Newly identified vulnerabilities are differentiated based on their use by an adversary in initiating new attack path or using it as a milestone for staging multistage attack. The external causes for newly introduced vulnerabilities are identified in terms of preconditions and patched for further network hardening. A case study is presented to show the usefulness of our approach for a small toy network. We found that our approach is capable of detecting the newly introduced vulnerabilities into the enterprise network. Our future work, includes applying the proposed change point detection algorithm for more sophisticated/complex enterprise network.

References

1. Phillips, C., Swiler, L.: A graph-based system for network vulnerability analysis. In: Proceedings of the 1998 Workshop on New Security Paradigms (NSPW '98), pp. 71–79. ACM, New York (1998)
2. Ortalo, R., Deswarte, Y., Kaaniche, M.: Experimenting with quantitative evaluation tools for monitoring operational security. IEEE Trans. Softw. Eng. **25**, 633–650 (1999)
3. Li, W., Vaughn, B.: Cluster security research involving the modeling of network exploitations using exploitation graphs. In: Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID '06), vol. 2, pp. 26–26. IEEE Computer Society, Washington. May 2006
4. Idika, N., Bhargava, B.: Extending attack graph-based security metrics and aggregating their application. IEEE Trans. Dependable Secur. Comput. **9**(1), 75–78 (2012)
5. Noel, S., Jajodia, S.: Metrics suite for network attack graph analytics. In: 9th Annual Cyber and Information Security Research Conference (CISR '14), pp. 5–8. ACM, Oak Ridge National Laboratory, Tennessee, New York, April 2014
6. Wang, L., Singhal, A., Jajodia, S.: Measuring the overall security of network configurations using attack graphs. In: 21st Annual IFIP WG 11.3 Working Conference on Data and Applications Security, vol. 4602, pp. 98–112. Springer-Verlag, Berlin, Heidelberg (2007)
7. Wang, L., Liu, A., Jajodia, S.: An efficient and unified approach to correlating, hypothesizing, and predicting intrusion alerts. In: Proceedings of the 10th European Conference on Research in Computer Security (ESORICS'05), pp. 247–266. Springer-Verlag, Berlin, Heidelberg (2005)
8. Wang, L., Noel, S., Jajodia, S.: Minimum-cost network hardening using attack graphs. J. Comput. Commun. **29**(18), 3812–3824 (2006)

9. Sun, K., Jajodia, S.: Protecting enterprise networks through attack surface expansion. In: Proceedings of the 2014 Workshop on Cyber Security Analytics, Intelligence and Automation (SafeConfig '14). pp. 29–32. ACM, New York (2014)
10. Ghosh, N., Ghosh, S.K.: An approach for security assessment of network configurations using attack graph. In: First International Conference on Networks and Communications 2009, (NETCOM'09), pp. 283–288, Dec 2009
11. SGPlan 5: http://wah.cse.cuhk.edu.hk/wah/programs/SGPlan/
12. Ghosh, N., Ghosh, S.K.: A planner-based approach to generate and analyze minimal attack graph. J. Appl. Intell. **36**(2), 369–390 (2012)