Computer Communications 174 (2021) 61–74

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

An ensemble approach for optimization of penetration layout in wide area networks

Urvashi Garg *, Geeta Sikka, Lalit K. Awasthi

Dr B R Ambedkar National Institute of Technology, Jalandhar, Punjab, 144011, India

ARTICLE INFO

Keywords:
Security
Vulnerability analysis
Attack graph optimization
Attack graph ensemble
Penetration scenario

ABSTRACT

The complexity of IoT based networks and an exponential increase in new vulnerabilities has increased the demand for security assessment strategies manifold. Attack graphs or Penetration layouts play a paramount role to harden and analyze such complex networks. As the size of the network grows, administrators may find it difficult to comprehend penetration layout. In this article, we present a methodology to bridge the gap between large networks and penetration layouts leading to a strategy that automatically generates, optimizes, and improves visualization of penetration layout in large networks. More specifically, we take the network model as input to the designed simulator which analyzes the network and generates the penetration layout. Additionally, we have designed an algorithm to optimize the size of the penetration layout at various levels. This will also improve the visualization of the graph. We designed a simulator that uses a real-time network blueprint to visualize and analyze the effect and performance of the proposed approach. The results show that there is a lossless reduction in the size of penetration layout by 99.95% for the example real-time network.

1. Introduction

As the network complexity increases, security is emerging as a major challenge for administrators these days [1]. This leads to the increasing requirement of security planning and risk assessment techniques. Theoretically, patching of vulnerability once it is detected seems to be a promising solution to major security problems. However, this is a never-ending process. Additionally, it is not enough to consider an individual vulnerability scenario. As the complexity of the network grows, various layers of connectivity may lead to more risky scenarios. Here, we briefly introduce the problem of identifying these vulnerabilities and the generation & optimization of a penetration layout that can be used to analyze these vulnerabilities.

1.1. Chained vulnerabilities and penetration layout

To consider the broad network scenario it is not sufficient to consider only isolated vulnerabilities associated with individual systems. Hence, chained vulnerability exploitation must be considered. Computing systems are connected in a network that may introduce various security holes. Also, vulnerabilities can be exploited in a sequence forming a chain to achieve the final target. To meet the requirements of connectivity issues, security specialists come up with attack trees in 1999 which were named as attack graphs [2]. The complete process of attack graph generation can be visualized from Fig. 1 which explains that vulnerabilities of systems are scanned through vulnerability scanners. In the next step, this vulnerability data along with connectivity policy is used to generate an attack graph which is also known as penetration layout. Penetration layout is a combination of various penetration scenarios. Every penetration scenario is a series of exploits on different machines that lead them to an undesirable state. Penetration layout plays a vital role in ensuring security in the network including analysis and defense e.g.

- It can be used to detect the number of ways in which an attacker can enter into a network. Additionally, the riskier vulnerability can be analyzed from a penetration scenario.
- We can predict the set of vulnerabilities or actions that should be tackled to prevent the attacker from reaching the target i.e. we may patch more risky vulnerabilities first.

1.2. Scope of paper

In this article, a simulator is designed to generate a possible penetration layout that is tested on a real-time network. Additionally, a methodology is proposed to optimize the size of the layout to increase readability and scalability. This will be a promising solution to major security problems in larger networks. However, the major focus of this article is on the generation and optimization of penetration layout. We have not considered the analysis of penetration scenarios and

* Correspondence to: Department of CSE, Dr B R Ambedkar National Institute of Technology, Jalandhar, Punjab, 144011, India.
E-mail addresses: urvashi.garg.24@gmail.com, urvashi@nitj.ac.in (U. Garg).

https://doi.org/10.1016/j.comcom.2021.04.009
Received 14 October 2020; Received in revised form 4 April 2021; Accepted 9 April 2021
Available online 16 April 2021

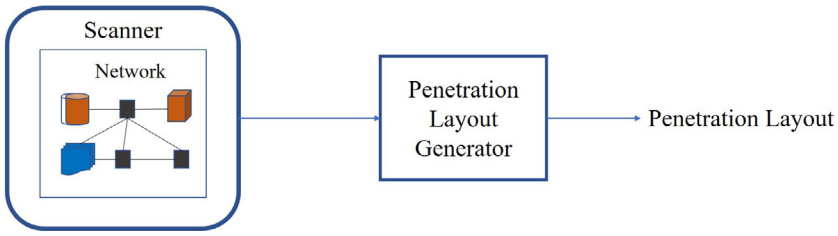

Fig. 1. Generation of penetration layout of a network.

vulnerabilities, although the designed tool is performing the analysis part also. Vulnerability analysis and prioritization are themselves a broader topic that needs a separate discussion and is out of the scope of this paper.

1.3. Our contribution

While reviewing the articles on attack graph generation, optimization and analysis, we have found that there are a few contributions to optimization and visualization of attack graphs. To this end, we have found the research questions for this field. In this article, we have focused on these research questions i.e. optimization problem, visualization, and scalability. To the best of our knowledge, this is the novel work in the field of penetration layout optimization and visualization improvement. The major and novel contributions of this paper are:

- Design a methodology to automatically generate a penetration scenario in real-time network scenario.
- Develop an algorithm to optimize the size of penetration layout at various levels to make it scalable.
- Development of an algorithm to improve the visualization of penetration layout that makes it easier to analyze and prioritize penetration scenario and layout.
- Design a simulator to generate, optimize and analyze a penetration scenario and layouts.

1.4. Organization of paper

The rest of this paper is organized as follows. Section 2 discusses the background on attack graphs generation, optimization, and analysis work. Section 3 elaborates the attack model and penetration layout. Layout components and penetration scenarios are also discussed in this section. In Section 4, we have presented the blueprint of an example real-time network taken for validation and analysis of the proposed methodology. The penetration layout optimization algorithm is also presented in this section. Section 5 portrays the basic architecture and functionalities of a designed simulator. Section 6 presented the results and outcomes of the proposed scheme. Discussion on results is also presented in this section. Results of the proposed methodology on a broader network are also discussed in this section. Finally, Section 7 concludes this article and also highlights the future scope in this field.

2. Related work

Attack graphs or penetration layouts have been an emerging topic for researchers since 1990. Several contributions have been published since then describing the generation and representation types of attack graphs. The proposed scheme is a foundation of three research areas i.e. attack graph generation, scalability & optimization, and attack graph risk analysis.

2.1. Attack graph generation

The attack graph concept started with the privilege graph in 1996 discussed by Dacier where nodes were represented by the privilege attained by the user after exploiting any vulnerability [3]. In 1999, the attack tree concept was introduced by Bruce Schneier [2] to show the attacker's activity at the tree node. The attack tree consists of all the activities which were the combined effect of multiple activities performed by the attacker represented using Boolean AND and OR nodes. In the next year, scenario graphs were discussed by Jha [4] which can portray the counter-example of a given property in the form of attack graphs, i.e. it represents all the paths nullifying the given security property. In the year 2013 goal plan graph [5] was proposed which was able to predict the next moves of an attacker. Some authors have mentioned the simplest approach to generate attack graphs, i.e. Attack State Transition Graph (ASTG) [6]. In these types of graphs, each node represents the state of the attacker and the edge represents the attackers move in succession to the previous step. These types of graphs were easy to design and develop but difficult to visualize and understand. Another similar type of graph is Host Level Attack Graph (HLAG) [7]. In these types of graphs, nodes portrayed the information about hosts exploited in sequence to attain the target host. HLAG has fewer nodes than ASTG. However, it requires the maintenance of large precondition postcondition data set. To overcome the problem of the size of an attack graph, exploit dependency graph [8] and Bayesian attack graph [9] comes into the picture. A bayesian attack graph was used to represents the next possible exploit depending upon the probability of execution of the previous exploit. The exploit dependency graph consists of exploits as nodes, and each edge corresponds to the next exploit in succession with the previous one. This graph was generated for individual host separately. However, it is possible to expand these graphs to generate a network attack graph [10]. The network attack graph represents all possible exploitation paths within a network. Although it is a useful tool to analyze the complete network scenario, it may have scalability issues and redundant path problems.

2.2. Attack graph optimization and scalability

As the network complexity rises, the size and complexity of the attack graph also increase [11]. To mitigate such problems, Farmad [12] worked on the Bayesian attack graph which significantly reduces the state explosion problem [13]. In this graph, the conditional probability of the next vulnerability (node) to be exploited is calculated using parent node probability. However, this scheme fails to consider attacker's capability and vulnerability complexity. Kaynar uses a parallel approach for generating distributed attack graphs to reduce running time [14]. They have used distributed parallel agents to generate graph parallelly and virtual shared memory for storing agent-specific data. Although, both the schemes were good, the visualization of the graph is not improved much. Further, they have not addressed the optimization problem of graphs at all. Some authors have worked on subnet wise optimization of attack graph [15–17]. Although they have proposed a good strategy in this field, their scheme has some disadvantages as discussed in Table 6.

2.3. Attack graph risk analysis

Nowadays, vulnerability analysis in an attack graph is becoming a substantial challenge for researchers. To overcome this challenge, numerous schemes have been discussed to analyze and prioritize vulnerabilities. In 2005, FIRST (Forum of Incident Response and Security Teams) have proposed CVSS v1 to rank individual vulnerabilities [18]. After that various authors have worked upon scoring schemes by either modifying CVSS base score or proposing a rank analysis approach [19–25]. Wing [26] has proposed a survivability model for network i.e. ability of a system to maintain its state despite the presence of unusual events. The sequence of these events is known as a scenario graph which was further discussed and used by other researchers also [27–31]. Idika and Bhargava have modified attack path length metrics to calculate the normalized mean, median, mode, and standard deviation of attack path length [32].

One of the best solutions to avoid attack paths were provided by Chao [33]. He had discussed an algorithm to find a minimal critical set of initial conditions which once patched can avoid a complete attack graph. In 2012, Wang has proposed an algorithm to patch the most critical vulnerabilities first and leave a minimal set of vulnerabilities for attackers [34]. Some hybrid techniques were also proposed by researchers to combine two or more schemes such that their advantages can be combined to get better results [35,36]. One of the analysis methodologies was a risk score based algorithm integrated with a mathematical function [37]. This scheme was based on CVSS (Common Vulnerability Scoring System) [18] scoring scheme. The proposed scheme was an extension of CVSS v2 and v3. This methodology has considered all three types of characteristics of vulnerability [18]. Additionally, it was proved that the proposed methodology produces better results as compared to other schemes. In other analysis work, time to compromise a vulnerability model is proposed to compare and analyze various vulnerabilities and systems [38]. The systems were analyzed and compared for their risk factors calculated according to the designed methodology. Finally, the proposed scheme can be used to find the riskier system that needs the attention of administrators.

Size and complexity of the attack graph is the major issue in most of the contributions. Therefore, in this paper, we have proposed a technique to increase the understandability and scalability of attack graphs. The proposed technique will help in getting only a relevant portion of the graph leaving behind the irrelevant one with the help of a designed simulator.

3. Attack model and penetration layout

Penetration scenario represents the possible sequence of actions performed by an attacker to achieve a goal i.e. service disruption on the target system; gaining root privilege on the system, etc. Penetration layout is the graphical representation of a combination of different penetration scenarios. In this section, attack models are discussed which are employed with penetration layout components to create a penetration layout.

3.1. Attack model

The attack model is the formal representation of a network attack situation. According to this article, an attack situation can be represented as the privileges or actions performed by an attacker on the target system. Therefore, the attack model is represented by the ways of defining an attacker's actions.

Definition 1. An attack model is a set of three elements $AM = (A, \Theta, A_0)$, where A represents the states, Θ represents the reachability from one state to other and A_0 represents the root state. The state A is again a combination of four elements $AM = (N, v, \mathbb{S}_I, \mathbb{D}_I)$, where N represents the state number, v is the vulnerability CVE Id [39], $\mathbb{S}_I$ is the source system id from where the attack is generated and $\mathbb{D}_I$ is the target system id on which the attack is performed.

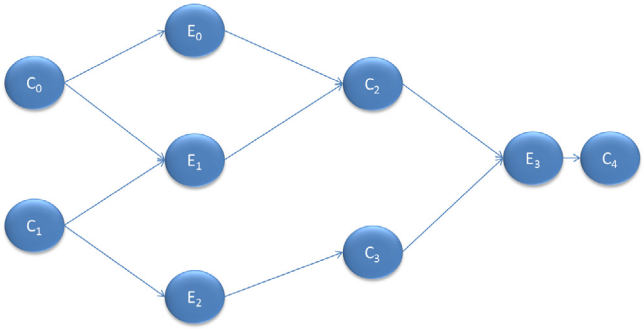

Fig. 2. Exploit dependency graph [8].

Intruder starts with a root system A_0 which is directly reachable from outside the network. Intruder then performs further actions to achieve some unsafe states on different systems that are directly reachable from the root system. This process goes on to achieve an unsafe state on the target system. Here, the unsafe state represents the undesirable privileges attained by the attacker on a system. Penetration layout is the graphical representation of intruder actions in sequence in this process.

3.2. Penetration layout components

Some of the major components used in this work are given below.

3.2.1. Nodes

Nodes are the systems running services, performing actions, and storage hub of some confidential data. Every move of an intruder is targeting some host. The major purpose of an attacker is to convert a host into an unsafe state. A node in a penetration layout is represented by a finite automaton $AM = (N, v, \mathbb{S}_I, \mathbb{D}_I)$, where

- N is the node number or identifier to uniquely identify a node.
- v is the CVE id [39] of the vulnerability exploited on that particular system or node. This may lead to service disruption, gaining of root privilege, or escalation of existing access privilege, etc.
- Attacker at source system S_1 wants access to destination system S_2 and thence will exploit any vulnerability on S_2. $\mathbb{S}_I$ represents the source system.
- $\mathbb{D}_I$ represents the destination system.

3.2.2. Edges

An edge connects two nodes in the penetration layout. It also represents the connectivity relation between the two systems.

Definition 2. Every edge is a finite automaton $E = (\mathbb{P}, \mathbb{S})$, where every incoming edge is represented as privilege $\mathbb{P}$ required to exploit a vulnerability on target system $\mathbb{S}$ and every outgoing edge represents the privilege $\mathbb{P}$ attained after exploiting the vulnerability on system $\mathbb{S}$.

3.3. Penetration layout

Numerous types of attack graph generation schemes and types of attack graphs have been proposed till date [6,7,12,40–42]. There is a common problem among all these techniques, i.e. the complexity of the graph is very high. However, the exploit dependency graph is less complex as compared to other types of graphs in terms of the number of nodes and edges [8]. We have used an exploit dependency graph as a base model to portray network behavior with further reduced size. Before discussing the proposed methodology to generate the penetration layout, we will discuss the sample exploit dependency graph scheme.

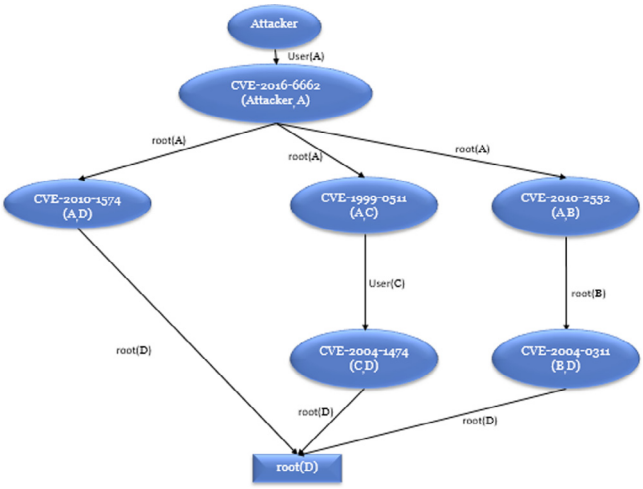

Fig. 3. Proposed penetration layout scheme.

3.3.1. Sample penetration layout

Fig. 2 shows the sample exploit dependency graph proposed by Ghosh and Bhattacharya [2]. In this figure, there were nine nodes and ten edges, and every node is provided with some label over it, i.e. C_0 to C_4 and E_0 to E_3. Here C represents condition and E represents exploit. Every condition node before exploit node portrays the information that what preconditions were required to exploit a vulnerability on a system. Every condition node after the exploit node portrays the postcondition achieved after exploiting the vulnerability of a system which may act as a precondition for another system. However, these types of graphs were less complex than other types of graphs but, further size reduction is possible in this graph.

3.3.2. Proposed penetration layout generation scheme

In the proposed scheme, we have modified the exploit dependency graph in such a way that every node portrays some additional information, e.g. node number, vulnerability CVE id, its source system id which initiated the attack, and destination system id on which the vulnerability has been exploited. Additionally, every edge also portrays some information, i.e. every incoming edge provides the details about the precondition of the exploit and every outgoing edge portrays the information about the postcondition of exploit. This process will further reduces the number of nodes as a condition node is not required anymore. Additionally, the proposed scheme significantly improved the readability of layout, i.e. every node and edge label itself explains everything. Fig. 3[1] represents the small portion of the complete layout generated for network architecture as shown in Fig. 4.

4. The network blueprint

Fig. 4 represents the real-time network used for the proposed scheme. It consists of three target systems with details provided in Table 1. This network consists of one more system which is directly reachable from the attacker's system. An attacker will use this system to reach other target systems. The whole network is separated through a firewall from the Internet.

[1] In this layout, the size of the penetration scenario is less, i.e. maximum four edges, but the actual layout consists of a lot many nodes and edges. Therefore, in the actual scenario, we have to hide the details of nodes (shown only when the cursor is over that node), and nodes size is reduced such as to cover all nodes in a single screen.

Table 1
Target systems details.

System IP address	Operating system
10.10.54.101	Windows 2008
10.10.54.201	Windows 2000
10.10.54.254	Linux Ubuntu 16.04 LTS

Table 2
Number of vulnerabilities in local network hosts.

Host IP address	Number of vulnerabilities in the system
10.10.54.101	22
10.10.54.188	165
10.10.54.201	1
10.10.54.254	18

4.1. Systems and vulnerabilities

The network systems consist of various vulnerabilities depending on the services running on them as shown in Table 2. The information about the systems connectivity policy including privileges is provided in Table 3. In this table, the first value is 1 if two systems are connected by a link, and the second value is 1 if the destination system is accessible via port access otherwise these values are zero. It can be analyzed from Table 3 that system S_1 can be accessed from the attacker's system via a physical link as well as a port. The network was scanned through the Nessus scanner [43] and it was observed that there were 206 vulnerabilities in four hosts (The details are given in the Appendix). The detailed vulnerability database along with precondition & postcondition dataset and connectivity relation of the network is used to generate the penetration layout.

4.2. Penetration layout

While looking at Table 3, we can find that intruder has some access to only one local host with IP address 10.10.54.188. However, other internal hosts having LAN connectivity could access each other. The attacker host first exploits the directly reachable system and then starts progressing towards other internal hosts in a chained scenario to form a penetration scenario. The combined penetration scenarios are known as penetration layout.

We could analyze a case of generating a penetration scenario e.g. there is a vulnerability with ID CVE-2016-6662 [39] which is present on the host with IP address 10.10.54.188 (S_1). This shall provide the leverage to execute any code with root privilege on the target system. Once this vulnerability is exploited on this host, the intruder targets the other hosts inside the network. An intruder can target a host with IP address 10.10.54.101 (S_2) by exploiting the vulnerability ID CVE-2008-4250 [39] which further allows the intruder to execute arbitrary code on the target system. This is possible because of connectivity relation between S_1 and S_2. Finally a penetration scenario is generated from intruder to S_1 to S_2. This process goes on to find all possible penetration scenarios which then combined to form penetration layout as shown in Fig. 5.[2]

4.3. Penetration layout optimization

Penetration layout size increases exponentially as the number of hosts or vulnerabilities increases in a network. Therefore, it becomes difficult to analyze it to get the riskier paths. We have proposed an ensemble method to aggregate layout nodes in such a way that without

[2] In this scenario, there were 37 700 attack paths generated for four hosts and 206 vulnerabilities as presented in Tables 1 and 2. Therefore, we have only displayed node number in this layout and hide other details which are provided in a separate file.

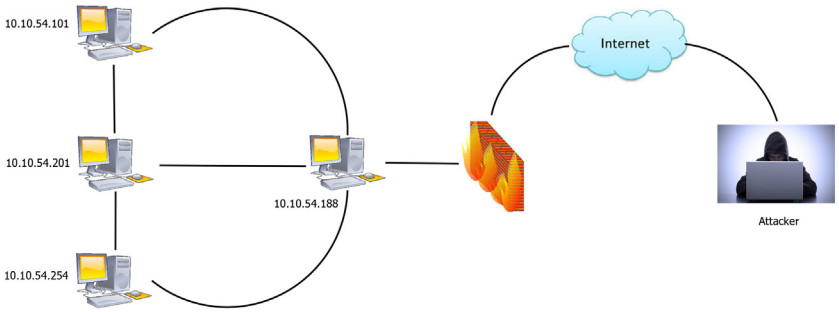

Fig. 4. Network architecture.

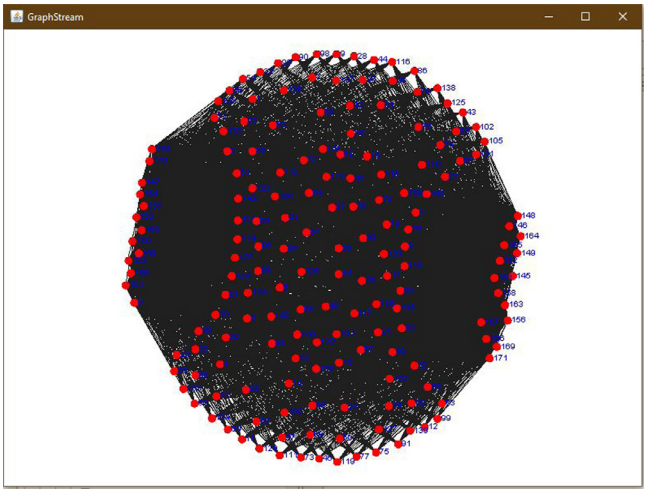

Fig. 5. Initial penetration layout.

Table 3
Firewall connectivity policies.

Source/destination Attacker	Attacker Atk(1,1)	S_1 Atk(1,1)	S_2 Atk(0,0)	S_3 Atk(0,0)	S_4 Atk(0,0)
10.10.54.188 (S_1)	Atk(1,0)	Atk(1,1)	Atk(1,1)	Atk(1,1)	Atk(1,1)
10.10.54.101 (S_2)	Atk(0,0)	Atk(1,0)	Atk(1,1)	Atk(0,0)	Atk(0,1)
10.10.54.201 (S_3)	Atk(0,0)	Atk(1,0)	Atk(0,0)	Atk(1,1)	Atk(0,1)
10.10.54.254 (S_4)	Atk(0,0)	Atk(1,1)	Atk(0,0)	Atk(0,0)	Atk(0,0)

any loss of information, the readability of the layout is improvised. We have also designed a tool to generate a penetration layout at various levels of optimizations. Algorithm 1 describes the complete methodology for penetration layout optimization.

4.3.1. Penetration layout with a reduced set of vulnerabilities

It is generally difficult to write an exploit code for a newly detected vulnerability and it takes approximately 5.8 days to write an exploit code for exploiting a new vulnerability [44]. Attackers are predominately interested in exploiting the vulnerabilities with available exploit code so that they can get the network access in less time. Although this step can skip zero-day vulnerabilities that need to be addressed separately but, most of the time attackers are not interested in writing fresh code for a new vulnerability. Therefore, this step is induced in the proposed algorithm. In this step, we have taken only those vulnerabilities whose exploit code [45] is available. The rule for this step of optimization also specifies that if a vulnerability is not present in the exploit database then remove it from the vulnerability database created for penetration layout generation.

Conditions Level 1 optimization
Vulnerability condition
$\mathbb{V} \notin$ ExploitDB
Effects Database
Vulnerability effect
Database($\mathbb{V}$) $\nvdash \mathbb{V}$

Algorithm 1 Penetration_Layout_Optimizer()

Input $\mathbb{P}$: Precondition and post condition dataset.
$\mathbb{V}$: Vulnerability dataset extracted from scan report of network.
$\mathbb{H}$: List of hosts extracted from scan report of network.
$\mathbb{E}$: Exploit code availability dataset .
$\mathbb{C}$: Connectivity policy.
Output $\mathbb{G}_i$: Penetration layout at four level of optimization process.

$\mathbb{G}_0$: Generate initial layout using $\mathbb{V}$ and $\mathbb{C}$
Copy $\mathbb{G}_0$ to $\mathbb{G}_1$
for $\forall$ nodes $\eta \in \mathbb{G}_0$ **do**
  if $\mathbb{V}(\eta) \notin \mathbb{E}$
  remove η and corresponding edges from $\mathbb{G}_1$
Copy $\mathbb{G}_1$ to $\mathbb{G}_2$
for $\forall$ edges $\varepsilon(\eta_1, \eta_2) \in \mathbb{G}_1$ **do**
  if $\mathbb{P}(\mathbb{V}(\eta_2)) \nsubseteq \mathbb{P}(\mathbb{V}(\eta_1))$
  remove η_2 and edge $\varepsilon(\eta_1, \eta_2)$ from $\mathbb{G}_2$
Copy $\mathbb{G}_2$ to $\mathbb{G}_3$
for $\forall$ edges $\varepsilon(\eta_1, \eta_2) \in \mathbb{G}_2$ **do**
  if $\mathbb{H}(\eta_1) = \mathbb{H}(\eta_2)$
  remove η_2 and edge $\varepsilon(\eta_1, \eta_2)$ from $\mathbb{G}_3$ and update η_1.
Copy $\mathbb{G}_3$ to $\mathbb{G}_4$
for $\forall$ edges $\varepsilon(\eta_3, \eta_1) \in \mathbb{G}_3$ AND $\varepsilon(\eta_3, \eta_2) \in \mathbb{G}_3$ **do**
  if $\mathbb{H}(\eta_1) = \mathbb{H}(\eta_2)$
  remove η_1 and edge $\varepsilon(\eta_3, \eta_1)$ from $\mathbb{G}_4$ and update η_2.

4.3.2. Penetration layout with precondition and postcondition component

Since it is not so easy to get access to a password protected system in one go. Some conditions must be met to exploit a vulnerability and get into the system. These conditions are known as preconditions, e.g. to exploit CVE-2016-6662 vulnerability, an attacker must have at least user-level privilege on the target system [39]. Once a vulnerability is exploited, it may provide some privilege to the attacker e.g. CVE-2016-6662 provides root access to an attacker on the target system after exploitation. Hence, it is necessary to consider precondition and postcondition of every vulnerability as this is the only deciding factor that whether a vulnerability will be exploited or not. Therefore, in this phase, we have added precondition and postcondition of every vulnerability. The rule for this level specifies that if precondition of $\mathbb{V}_2$ is not the subset of postcondition of $\mathbb{V}_1$ then remove an edge from node $\mathbb{N}_1(\mathbb{V}_1)$ to $\mathbb{N}_2(\mathbb{V}_2)$ in the edge database and node $\mathbb{N}_2(\mathbb{V}_2)$ from node database. Additionally, include the node information of $\mathbb{N}_2(\mathbb{V}_2)$ to $\mathbb{N}_1(\mathbb{V}_1)$.

Conditions Level 2 optimization
Vulnerability condition
pre_condition($\mathbb{V}_2$) $\nsubseteq$ post_condition($\mathbb{V}_1$)

Effects Penetration layout

Edge effect

Database($\mathbb{E}$) $\nvdash$ e($\mathbb{N}_1(\mathbb{V}_1)$, $\mathbb{N}_2(\mathbb{V}_2)$)

Node effect

Database($\mathbb{N}$) $\nvdash$ $\mathbb{N}_2(\mathbb{V}_2)$

4.3.3. Penetration layout ensemble

Penetration scenario consists of vulnerabilities exploited consecutively on a same host. On a system, it may happen that there may be different vulnerabilities giving different privileges when exploited. If there exists two such vulnerabilities on same host such that one of them giving less privileges than other, in that case these two can be ensemble into one giving higher privileges. As specified in its rule file that if there is an edge e($\mathbb{N}_1(\mathbb{V}_1)$, $\mathbb{N}_2(\mathbb{V}_2)$) such that $\mathbb{N}_1(\mathbb{V}_1)$ and $\mathbb{N}_2(\mathbb{V}_2)$ exists on same host($\mathbb{H}$) then we should remove edge e($\mathbb{N}_1(\mathbb{V}_1)$, $\mathbb{N}_2(\mathbb{V}_2)$) from edge database and node $\mathbb{N}_2(\mathbb{V}_2)$ from node database and store this vulnerability information in node $\mathbb{N}_1$ itself.

Conditions Level 3 optimization

Edge and node condition

e($\mathbb{N}_1(\mathbb{V}_1)$, $\mathbb{N}_2(\mathbb{V}_2)$) $\Rightarrow$ $\mathbb{N}_1(\mathbb{V}_1) \in$ Host($\mathbb{H}$) and $\mathbb{N}_2(\mathbb{V}_2) \in$ Host($\mathbb{H}$)

Effects Penetration layout

Edge effect

Database($\mathbb{E}$) $\nvdash$ e($\mathbb{N}_1(\mathbb{V}_1)$, $\mathbb{N}_2(\mathbb{V}_2)$)

Node effect

Database($\mathbb{N}$) $\nvdash$ $\mathbb{N}_2(\mathbb{V}_2)$ and update $\mathbb{N}_1(\mathbb{V}_1)$

4.3.4. Final penetration layout

In most applications, penetration layout optimization until the fourth phase is sufficient to increase the readability of penetration layout. However, when the number of hosts increases then the fifth phase is required to reduce the size of the layout further. In some cases, the postcondition of a vulnerability on one system may act as a precondition for multiple vulnerabilities of the same host. In such cases, sibling nodes may have a similar host with different vulnerabilities. Therefore, these nodes can be ensemble. As the rule of this level also specifies that if precondition of $\mathbb{V}_1$ and $\mathbb{V}_2$ matches with postcondition of $\mathbb{V}_3$ and $\mathbb{N}_1(\mathbb{V}_1)$ & $\mathbb{N}_2(\mathbb{V}_2)$ belongs to same host $\mathbb{H}$ then remove edge e($\mathbb{N}_3(\mathbb{V}_3)$, $\mathbb{N}_1(\mathbb{V}_1)$) and node $\mathbb{N}_1(\mathbb{V}_1)$ and update the information of node $\mathbb{N}_1(\mathbb{V}_1)$ to $\mathbb{N}_2(\mathbb{V}_2)$. This process coupled with the previous two processes is the major contribution of the proposed work which has led to the optimization of penetration layout as well as visualization of the layout is also improved.

Conditions Level 4 optimization

Vulnerability condition

precondition($\mathbb{V}_1$) $\subseteq$ postcondition($\mathbb{V}_3$) and
precondition($\mathbb{V}_2$) $\subseteq$ postcondition($\mathbb{V}_3$)

Node condition

$\mathbb{N}_1(\mathbb{V}_1) \in$ Host($\mathbb{H}$) and $\mathbb{N}_2(\mathbb{V}_2) \in$ Host($\mathbb{H}$)

Effects Penetration layout

Edge effect

Database($\mathbb{E}$) $\nvdash$ e($\mathbb{N}_3(\mathbb{V}_3)$, $\mathbb{N}_1(\mathbb{V}_1)$)

Node effect

Database($\mathbb{N}$) $\nvdash$ $\mathbb{N}_1(\mathbb{V}_1)$ and update $\mathbb{N}_2(\mathbb{V}_2)$

5. Penetration layout simulator

We have designed a simulator for generation, analysis, visualization, and optimization of penetration layout using the attack model discussed in Section 3. In this section, we discuss the basic architecture of the simulator that includes the databases and other relevant information required for the working of a simulator. Additionally, the implementation details and functionality of this simulator is also discussed in this section.

5.1. Simulator architecture

Fig. 6 shows the basic architecture of the designed simulator. There are five parts to this model. The first part scans the network which takes the network configuration file as input and generates a scan report as output. This report along with the attack model act as an input for the next part i.e. Attack model analyzer which further analyzes and generates the network model. This information is fed to the next part i.e. penetration layout generator. This part also takes the connectivity policy of the network and generates the GUI interface for penetration layout.

In the next part, exploit database information and precondition & postcondition database is fed to the optimizer. This part generates a penetration layout at four levels of the optimization process. The output of this part acts as input to the next part i.e. analyzer part. This part analyzes and prioritizes individual vulnerabilities as well as penetration scenarios employing three different models. But this part is beyond the scope of this article.

5.2. Simulator implementation details

It took around 5.5 months to develop the proposed simulator. We have generated 119 863 attack paths in 3.8 s on an 8 GB machine with a 2.3 GHz processing speed. It takes around 4 s for each level of optimization and 0.9 s for analysis of penetration layout using different strategies. The hardware requirement is standard system with processor and memory unit with minimum 500 MB disk and 128 MB RAM capacity. We also need at least 266 MHz pentium 2 processor for the same. For software requirement, we have used JAVA development kit and JAVA runtime environment on frontend and MYsql at the backend. The proposed simulator can be run on different machines with different operating systems without any additional requirement.

5.3. Simulator functionalities

The simulator performs various actions and has numerous benefits and functionalities e.g.

- Real-time scanning of a network employing Nessus scanner [43] in the background.
- XML parsing of scan report and extraction of useful information e.g. host IP address, vulnerability CVE ID [39], CVSS score [18] etc.
- Creation of vulnerability dataset, layout nodes, and edges dataset.
- Penetration layout generation and visualization using the GUI interface.
- Penetration layout optimization at four different levels and visualization using different GUI interfaces for all levels.
- Penetration layout analysis and prioritization using a designed formula for the generation of vulnerability and penetration scenario risk value.
- Penetration layout analysis and prioritization using a designed formula for the generation of vulnerability and penetration scenario cost of exploitation.
- Penetration layout analysis and prioritization using a designed formula for the generation of the time of exploiting vulnerability and penetration scenario.

6. Results and discussion

The proposed approach is applied to the real-time network discussed in Section 4 as well as a large network. The results are discussed in this section.

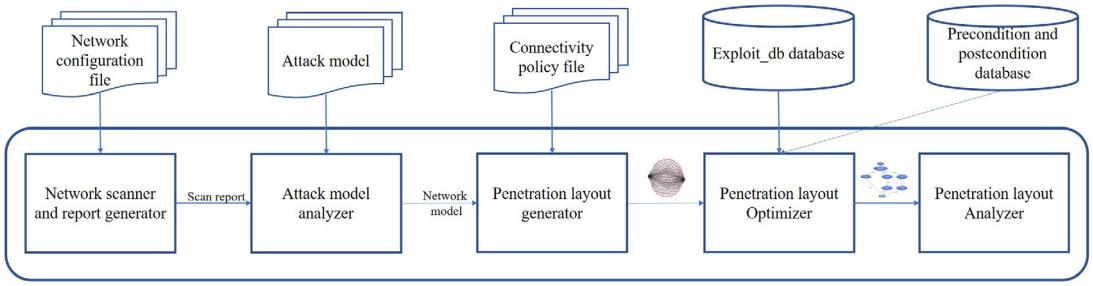

Fig. 6. Simulator architecture.

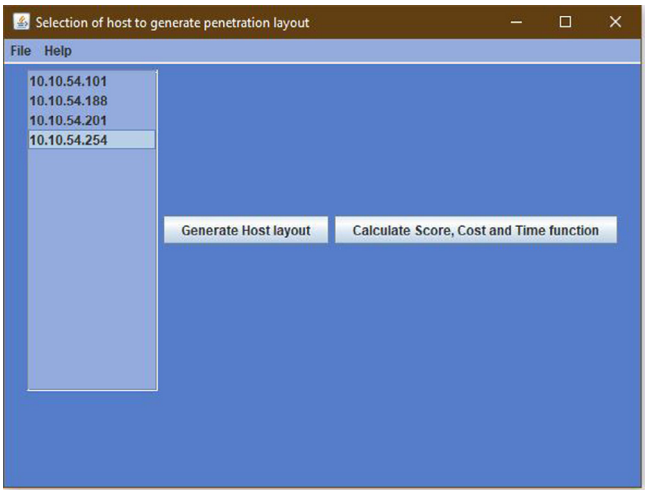

Fig. 7. Initial simulator setup.

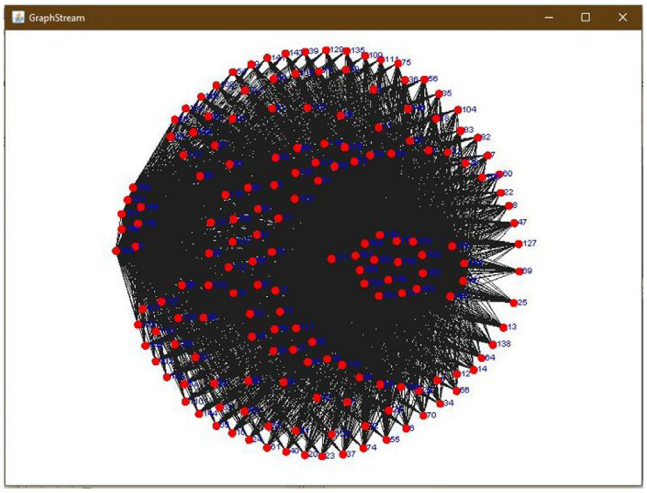

Fig. 8. Penetration scenario with reduced set of vulnerabilities.

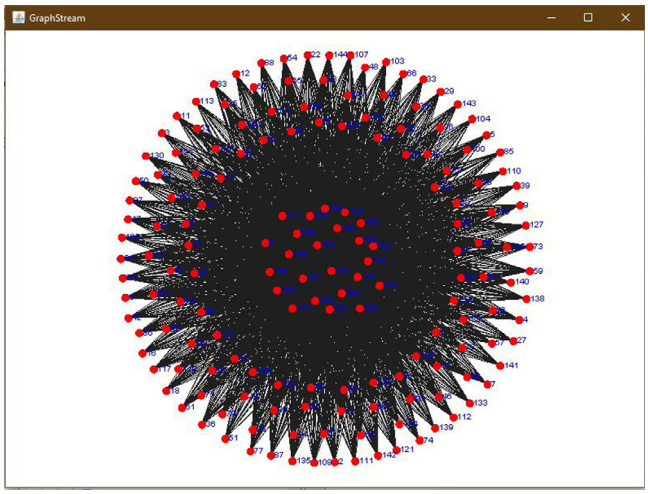

Fig. 9. Penetration scenario with precondition and post condition component.

Table 4
Final penetration layout details.

Node number	Host IP address
0	Attacker node
1	10.10.54.188
2	10.10.54.101
3	10.10.54.201
4	10.10.54.254
Paths in the layout	
0 → 1 → 2 → 4	
0 → 1 → 3 → 4	
0 → 1 → 4	
0 → 2 → 4	
0 → 3 → 4	
0 → 4	

Table 5
Comparison of optimization at different levels.

Level of optimization	Number of nodes	Number of paths	Optimization (%)
0	178	37 700	0
1	154	15 660	58.5
2	154	10 590	71.9
3	15	144	99.7
4	5	6	99.95

6.1. Experiment and results on a smaller network

Taking the real-time network discussed in Section 4 and considering some conditions, we experimented to verify and validate the proposed approach. Fig. 7 shows the initial GUI used to generate and analyze the penetration layout. It consists of an option to select the network for scanning and selection of reports which will further extract and display host IP addresses.

In a network, practically not every internal host over LAN may have access to each other. Any organization must limit the access permission of internal LAN connected hosts to ensure the security of the network which is the basis of the first level optimization process of penetration layout. In the first phase, we have limited access permission for internal hosts according to a specific organization as shown in Table 3. Additionally, in this step, we have taken only those vulnerabilities whose exploit code [45] is available. Once again the layout is generated for four hosts and 206 vulnerabilities as shown in Fig. 8. This layout consists of 15 660 attack paths. In this step also, it is very difficult to analyze and predict any attack path. So, we apply the third phase of the methodology.

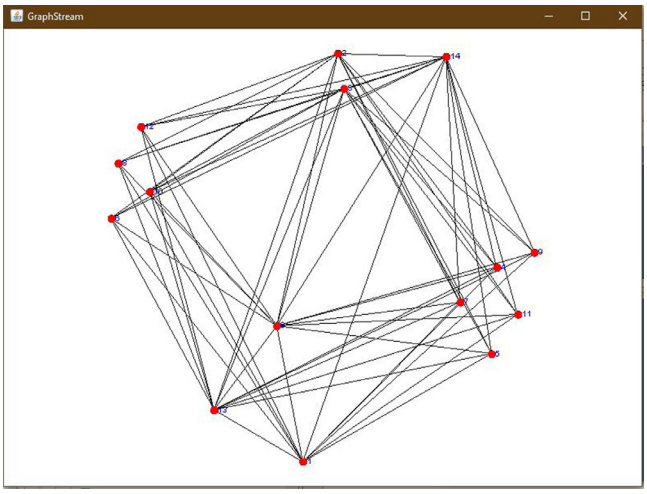

Fig. 10. Penetration scenario ensemble.

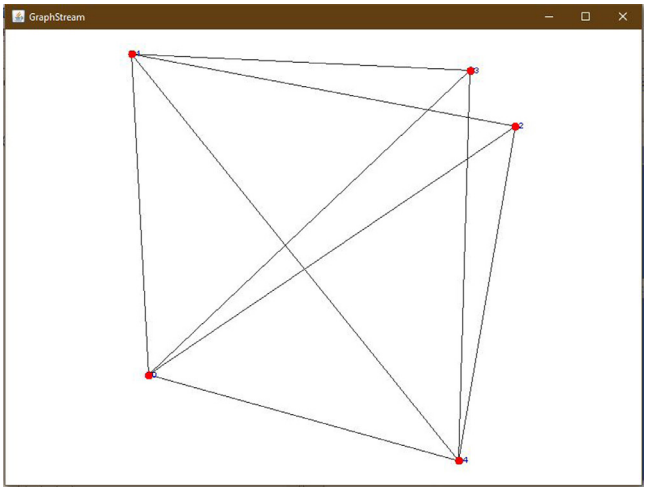

Fig. 11. Final Penetration scenario.

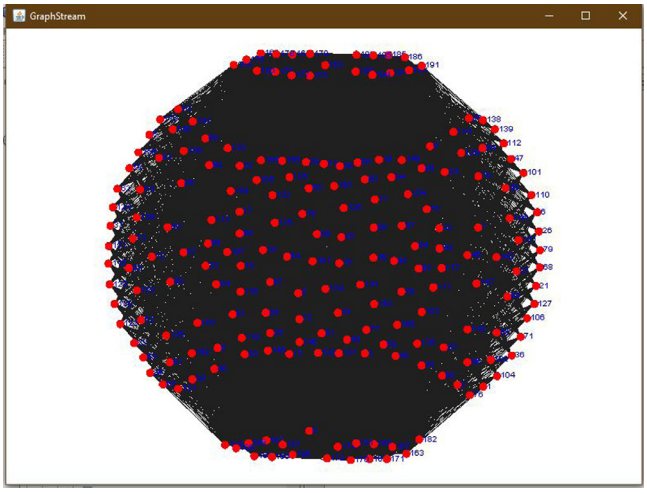

Fig. 12. Penetration scenario with reduced set of vulnerabilities for broader network.

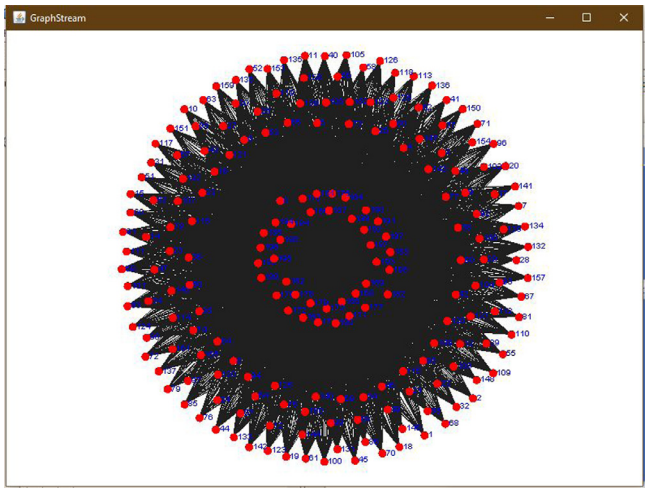

Fig. 13. Penetration scenario with precondition and post condition component for broader network.

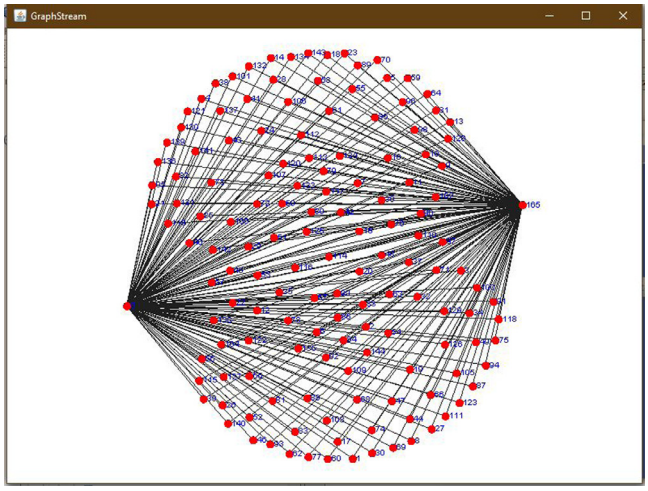

Fig. 14. Penetration scenario ensemble.

It is necessary to consider the precondition and postcondition of every vulnerability as this is the only deciding factor that whether a vulnerability will be exploited or not. Therefore, in the next phase, we have added precondition and postcondition of every vulnerability. In this way, penetration layout is generated as shown in Fig. 9. This layout consists of 10 590 paths and this is the first penetration layout that consists of only feasible nodes ignoring all irrelevant nodes.

In Fig. 9, many paths were having the same host exploited consecutively with different vulnerabilities. So we can ensemble these nodes to further reduce the size of the layout as shown in Fig. 10. This penetration layout consists of 144 paths only. It is an important step of the ensemble process because in this step we can still analyze and prioritize attack paths.

In the last step, we have finally ensemble all sibling nodes having the same host as shown in Fig. 11. This layout consists of only six paths. Details of these layout nodes are given in Table 4. Table 5 also specifies that the proposed methodology produced significant results as 99.95% optimization is there for a particular scenario.

6.2. Experiment and results on a broader network

This part is used to represent the scalability of the proposed approach. We have taken the broader network and verified & validated the proposed approach.

Table 6
Comparison among proposed methodology and existing techniques.

Author & Year	Title	Features/Advantages	Limitations
Noel et al. 2004 [15]	Managing attack graph complexity through visual hierarchical aggregation	• Initially, the exploit dependency graph is used. • Aggregation was performed based on connectedness. • Machines and exploits are aggregated if they form a connected subgraph. • At the last level, protection domains are aggregated. • This strategy shows subnet wise aggregation also.	• Extra condition node is present in the graph along with the exploit node. • This methodology may cause loss of information. Additionally, this process is not possible automatically, hence it is a manual process that needs prior knowledge. • Protection domains may or may not have disconnected components.
Williams et al. 2007 [17]	An interactive attack graph cascade and reachability display	• Multiple prerequisite graph generated by NetSPA tool was used for this work. • Three nodes were used for single vulnerability exploitation; prerequisite node, vulnerability instance, and state node. • Initially, they have removed prerequisite node and vulnerability instance node to optimize graph size. • In the next step, they have grouped state nodes based on reachability.	• Precondition and postcondition of vulnerability was not considered. • Size of the graph is still more than other schemes.
Homer et al. 2008 [16]	Improving attack graph visualization through data reduction and attack grouping	• MulVAL tool was used for the attack graph generation. • Initially, they have employed the methodology provided by Noel et al. Noel and Jajodia [15] and applied some additional techniques to further optimize the size of the graph. • In the next step, they have trimmed the edges and nodes between inter subnets when the transition between two hosts is useless. • At intra subnet level, they have trimmed the path from one node to another if both nodes belong to the same subnet and the second one is not a goal or the second one is not used for getting access to other subnets. • This scheme is also based on subnet wise aggregation.	• This methodology does not consider vulnerability exploited on hosts. • Manual interference is required at some level.
The proposed methodology	Improved penetration layout representation for security assessment in broader networks: An ensemble approach	• Modified exploit dependency graph is used such that condition node and exploit node were aggregated to reduce the size of the layout. • Multilevel optimization is performed. the same host were ensemble. • A simulator is designed to implement and automate the process. • This proposed methodology provides a significant improvement in layout visualization as compared to other schemes. • The proposed scheme can be implemented on broader networks also and hence penetration layout scaling is improved. • There is no loss of information and the complete process is automatic which does not require administrator knowledge or interference.	• The algorithm can skip zero-day vulnerabilities which need to be addressed separately. • We have gathered the precondition and postcondition dataset information manually as there a no direct database available for this information. So, automatic updation of precondition and postcondition dataset is required.

Table 7
Details of large network.

Parameter name	Parameter value
Number of hosts	146
Number of vulnerabilities	307
Number of critical vulnerabilities	41
Number of high risk vulnerabilities	37
Number of medium risk vulnerabilities	139
Number of low risk vulnerabilities	90

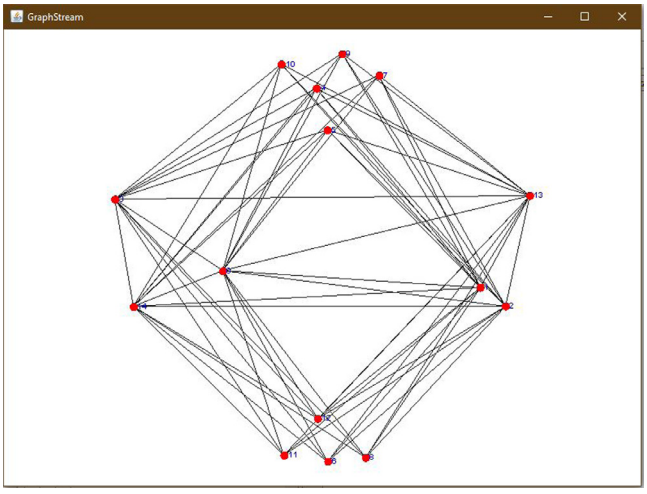

Fig. 15. Final Penetration scenario for broader network.

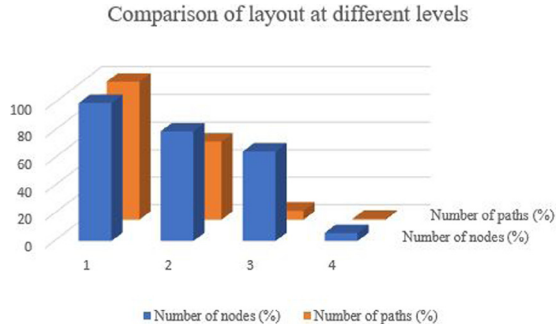

Fig. 16. Comparison of number of nodes and paths at different level of optimization.

6.2.1. The network prototype

The proposed technique was further tested on a large network consisting of 146 hosts and various vulnerabilities from critical risk levels to low-risk levels. The details are provided in Table 7. Initially, the network had 307 vulnerabilities on 146 hosts. In the present scenario, an 8GB RAM and i5 processor system were unable to draw such a large layout. Therefore, we have started from the first level optimization process by taking only vulnerabilities present in the exploit database, and we have also limited the access permission of systems within the network. Finally, we have applied the optimization steps at each level of the ensemble process.

6.2.2. Results

Since the network is large and it requires a lot of processing speed and memory for the generation of a larger layout. Therefore, we have directly applied the first level of optimization and produced the results as shown in Fig. 12. The initial layout in Fig. 12 consists of 254 nodes

Table 8
Details of vulnerabilities present in internal network hosts.

Host IP address	CVE ID	Description	CVSS score	Risk level	Protocol	Port
10.10.54.101	CVE-1999-0505	Microsoft windows SMB guest account local user access	5	Medium	TCP	445
	CVE-1999-0519 CVE-1999-0520 CVE-2002-1117	Microsoft windows SMB null session authentication				
	CVE-2006-1314 CVE-2006-1315	MS06-035: Vulnerability in server could allow remote code execution	7.5	High		
	CVE-1999-0519 CVE-1999-0520	Microsoft windows SMB shares unprivileged access				
	CVE-2005-1206	MS05-027: Vulnerability in SMB could allow remote code execution (896422) (uncredentialed check)	10	Critical		
	CVE-2006-3439	MS06-040: Vulnerability in server service could allow remote code execution (921883) (uncredentialed check)				
	CVE-2008-4250	MS08-067: Microsoft windows server service crafted RPC request handling remote code execution (958644) (uncredentialed check)				
	CVE-2008-4834 CVE-2008-4835 CVE-2008-4114	MS09-001: Microsoft windows SMB vulnerabilities remote code execution (958687) (uncredentialed check)				
	CVE-2010-0020 CVE-2010-0021 CVE-2010-0022 CVE-2010-0231	MS10-012: Vulnerabilities in SMB could allow remote code execution (971468) (uncredentialed check)				
	CVE-2010-2550 CVE-2010-2551 CVE-2010-2552	MS10-054: Vulnerabilities in SMB server could allow remote code execution (982214) (remote check)				
10.10.54.201	CVE-1999-0511	IP forwarding enabled	5.8	Medium	TCP	0
10.10.54.254	CVE-1999-0511	IP forwarding enabled	5.8	Medium	TCP	0
	CVE-1999-0517	SNMP agent default community names (public)	7.5	High	UDP	161
	CVE-1999-0186 CVE-1999-0254 CVE-1999-0472 CVE-1999-0516 CVE-1999-0517 CVE-1999-0792 CVE-2000-0147 CVE-2001-0380 CVE-2001-0514 CVE-2001-1210 CVE-2002-0109 CVE-2002-0478 CVE-2002-1229	SNMP agent default community names	10	Critical		

(*continued on next page*)

and paths come out to be 64 490. At each level of the ensemble process, layout size is reduced (see Figs. 13–15) and the final layout in Fig. 15 consists of 15 nodes and 504 paths. Hence we can deduce that the proposed methodology works well on broader networks also makes it scalable. The comparison among several nodes and paths among the various levels of optimization is shown in Fig. 16 also.

6.3. Discussion

The designed methodology is used to optimize the size of the penetration layout. There are various conclusions gathered from the proposed approach which includes some benefits along with some limitations:

6.3.1. Advantages

The proposed approach have following benefits:

- We have designed a real-time simulator that automatically scans and analyzes the network.
- There are four levels of optimization which increase the scalability and readability of graph.
- This is a lossless optimization process. Therefore, there is no loss of information while optimizing the size of network. Hence, network can be easily analyzed with increased readability.
- Administrator can use any level layout for visualization and analysis purposes.
- Analysis performed by the simulator helps detect the riskier vulnerabilities and riskier scenarios that need to be patched to avoid complete penetration layout. This will prevent loss of services as minimal vulnerabilities are needed to be fixed to avoid complete penetration scenario.

6.3.2. Limitations

The proposed approach have following limitations:

- The proposed algorithm has a limitation that it can skip zero-day vulnerabilities which need to be addressed separately.
- The second limitation of the proposed work is that we have gathered the precondition and postcondition dataset information manually as there a no direct database available for this information.

Table 8 (*continued*).

Host IP address	CVE ID	Description	CVSS score	Risk level	Protocol	Port
10.10.54.254	CVE-2004-0311 CVE-2004-1474 CVE-2010-1574	SNMP agent default community names	10	Critical	UDP	161
10.10.54.188	CVE-2015-0245	Ubuntu 12.04 LTS/14.04 LTS/16.04 LTS/16.10 : dbus vulnerabilities (USN-3116-1)	1.9	Low	TCP	0
	CVE-2016-2183 CVE-2016-6329	SSL 64-bit block size cipher suites supported (SWEET32)	2.6	Low	TCP	9390/443/9391
	CVE-2016-5584 CVE-2016-7440	Ubuntu 12.04 LTS/14.04 LTS/16.04 LTS/16.10: mysql-5.5/5.7 vuln.	3.5	Low		
	CVE-2016-1586 CVE-2016-5181 CVE-2016-5182 CVE-2016-5185 CVE-2016-5186 CVE-2016-5187 CVE-2016-5188 CVE-2016-5189 CVE-2016-5192 CVE-2016-5194	Ubuntu 14.04 LTS/16.04 LTS/16.10 : oxide-qt vulnerabilities (USN-3113-1)	4.3	Medium		
	CVE-2016-7042	Ubuntu 16.04 LTS : linux vulnerability (USN-3128-1)	4.9	Medium		
	CVE-2016-5195	Ubuntu 16.04 LTS : linux vulnerability (USN-3106-1) (Dirty COW)	7.2			
	CVE-2016-5250 CVE-2016-5257 CVE-2016-5270 CVE-2016-5272 CVE-2016-5274 CVE-2016-5276 CVE-2016-5277 CVE-2016-5278 CVE-2016-5280 CVE-2016-5281 CVE-2016-5284	Ubuntu 12.04 LTS/14.04 LTS/16.04 LTS/16.10 : thunderbird vulnerabilities (USN-3112-1)			TCP	0
	CVE-2016-6911 CVE-2016-7568 CVE-2016-8670	Ubuntu 12.04 LTS/14.04 LTS/16.04 LTS/16.10 : libgd2 vulnerabilities (USN-3117-1)				
	CVE-2016-7141 CVE-2016-7167 CVE-2016-8615 CVE-2016-8616 CVE-2016-8617 CVE-2016-8618 CVE-2016-8619 CVE-2016-8620 CVE-2016-8621 CVE-2016-8622 CVE-2016-8623 CVE-2016-8624	Ubuntu 12.04 LTS/14.04 LTS/16.04 LTS/16.10 : curl vulnerabilities (USN-3123-1)	7.5	High		
	CVE-2014-8354 CVE-2014-8355 CVE-2014-8562 CVE-2014-8716	Ubuntu 12.04 LTS/14.04 LTS/16.04 LTS/16.10 : imagemagick vulnerabilities (USN-3131-1)				

(*continued on next page*)

- We need a stronger processor for large number of hosts and vulnerabilities as the processing is increased as network size increases.

6.4. Comparative study of the proposed methodology and existing closely related approaches

Table 6 represents the comparative study of the proposed methodology and already existing techniques.

7. Conclusion and future work

In this paper, we have proposed, designed, and implemented an optimized penetration layout generation scheme. We have used an ensemble approach to further optimize the size of the penetration layout. Additionally, the proposed scheme is beneficial in improving the visualization of penetration layouts. We have also designed a simulator to sustenance the proposed approach. This simulator uses a GUI interface for the visualization of results. This is integrated with the Nessus scanner to fully automate the process.

We have created a database of 3000 vulnerabilities preconditions and postconditions. A real-time network with four hosts and 206 vulnerabilities has been used to test and implement the proposed approach which allows us to create an initial penetration layout with 37 700 paths and 178 nodes. The major contribution of the proposed approach is to enhance the scalability and readability of the layout. We have proposed four-levels of the optimization process for penetration layouts. Every level of the proposed methodology has significantly reduced the size of the layout. In the case of four hosts and 206 vulnerabilities with a penetration layout of 37 700 paths and 178 nodes, our approach has

Table 8 (*continued*).

Host IP address	CVE ID	Description	CVSS score	Risk level	Protocol	Port
	CVE-2014-9805 CVE-2014-9806					
10.10.54.188	CVE-2014-9807 CVE-2014-9808 CVE-2014-9809 CVE-2014-9810 CVE-2014-9811 CVE-2014-9812 CVE-2014-9813 CVE-2014-9814 CVE-2014-9815 CVE-2014-9816 CVE-2014-9817 CVE-2014-9818 CVE-2014-9819 CVE-2014-9820 CVE-2014-9821 CVE-2014-9822 CVE-2014-9823 CVE-2014-9826 CVE-2014-9828 CVE-2014-9829 CVE-2014-9830 CVE-2014-9831 CVE-2014-9833 CVE-2014-9834 CVE-2014-9835 CVE-2014-9836 CVE-2014-9837 CVE-2014-9838 CVE-2014-9839 CVE-2014-9840 CVE-2014-9841 CVE-2014-9843 CVE-2014-9844 CVE-2014-9845 CVE-2014-9846 CVE-2014-9847 CVE-2014-9848 CVE-2014-9849 CVE-2014-9850 CVE-2014-9851 CVE-2014-9853 CVE-2014-9854 CVE-2014-9907 CVE-2015-8894 CVE-2015-8895 CVE-2015-8896 CVE-2015-8897 CVE-2015-8898 CVE-2015-8900 CVE-2015-8901 CVE-2015-8902 CVE-2015-8903	Ubuntu 12.04 LTS/14.04 LTS/16.04 LTS/16.10 : imagemagick vulnerabilities (USN-3131-1)	7.5	High	TCP	0

(*continued on next page*)

optimized it to 6 paths and 5 nodes, which is 99.95% reduction, and the utility of penetration layout is also not lost as it carries information about all vulnerabilities. There are few limitations of the proposed approach which need to be addressed in the future e.g. automatic updation of precondition and postcondition dataset and incorporation of zero-day vulnerabilities.

CRediT authorship contribution statement

Urvashi Garg: Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing - original draft, Writing - review & editing. **Geeta Sikka:** Conception and design of study, Writing - original draft, Writing - review & editing. **Lalit K. Awasthi:** Conception and design of study, Writing - original draft, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

All authors approved the version of the manuscript to be published.

Appendix. Example network vulnerabilities details

See Table 8.

Table 8 (*continued*).

Host IP address	CVE ID	Description	CVSS score	Risk level	Protocol	Port
10.10.54.188	CVE-2015-8957 CVE-2015-8958 CVE-2015-8959 CVE-2016-4562 CVE-2016-4563 CVE-2016-4564 CVE-2016-5010 CVE-2016-5687 CVE-2016-5688 CVE-2016-5689 CVE-2016-5690 CVE-2016-5691 CVE-2016-5841 CVE-2016-5842 CVE-2016-6491 CVE-2016-6823 CVE-2016-7101 CVE-2016-7513 CVE-2016-7514 CVE-2016-7515 CVE-2016-7516 CVE-2016-7517 CVE-2016-7518 CVE-2016-7519 CVE-2016-7520 CVE-2016-7521 CVE-2016-7522 CVE-2016-7523 CVE-2016-7524 CVE-2016-7525 CVE-2016-7526 CVE-2016-7527 CVE-2016-7528 CVE-2016-7529 CVE-2016-7530 CVE-2016-7531 CVE-2016-7532 CVE-2016-7533 CVE-2016-7534 CVE-2016-7535 CVE-2016-7536 CVE-2016-7537 CVE-2016-7538 CVE-2016-7539 CVE-2016-7540	Ubuntu 12.04 LTS/14.04 LTS/16.04 LTS/16.10 : imagemagick vulnerabilities (USN-3131-1)	7.5	High	TCP	0
	CVE-2016-6130 CVE-2016-6480 CVE-2016-6828 CVE-2016-7039	Ubuntu 16.04 LTS : linux vulnerabilities (USN-3099-1)	7.8			
	CVE-2016-0718 CVE-2016-1000028 CVE-2016-1000029	Tenable Nessus 6.x < 6.8 multiple vulnerabilities	9	High	TCP	8834
	CVE-2016-5287 CVE-2016-5288	Ubuntu 12.04 LTS/14.04 LTS/16.04 LTS/16.10: firefox vuln(USN-3111-1)	9.3	High	TCP	0
10.10.54.188	CVE-2016-5542 CVE-2016-5554 CVE-2016-5573 CVE-2016-5582 CVE-2016-5597	Ubuntu 16.04 LTS/16.10 : openjdk-8 vulnerabilities (USN-3121-1)	9.3	High	TCP	0
	CVE-2016-6662	Ubuntu 12.04 LTS/14.04 LTS/16.04 LTS : mysql-5.5, mysql-5.7 vulnerability (USN-3078-1)	10	Critical		

References

[1] A. Zimba, H. Chen, Z. Wang, M. Chishimba, Modeling and detection of the multi-stages of advanced persistent threats attacks based on semi-supervised learning and complex networks characteristics, Future Gener. Comput. Syst. 106 (2020) 501–517.

[2] B. Schneier, Attack trees, Dr. Dobb's J. (1999).

[3] M. Dacier, Y. Deswarte, M. Kaaniche, Models and tools for quantitative assessment of operational security, in: 12th International Information Security Conference (IFIP/SEC'96), 1996, pp. 177–186, http://dx.doi.org/10.1007/978-1-5041-2919-0_15, URL: http://link.springer.com/10.1007/978-1-5041-2919-0_15.

[4] S. Jha, J. Wing, R. Linger, T. Longstaff, Survivability analysis of network specifications, in: Proceedings of the 2002 International Conference on Dependable Systems and Networks, 2000, pp. 613–622, http://dx.doi.org/10.1109/ICDSN.2000.857597.

[5] Y. Liu, W.X. Gu, An effective recognition method for network attack, Optik 124 (21) (2013) 4823–4826, http://dx.doi.org/10.1016/j.ijleo.2013.02.036.

[6] H.Y. Lv, R.M. Wang, Network real-time threat awareness and analysis based on attack state transition, 1–6.

[7] Z. Liu, S. Li, J. He, D. Xie, Z. Deng, Complex network security analysis based on attack graph model, in: 2012 Second International Conference on Instrumentation, Measurement, Computer, Communication and Control, 2012, pp. 183–186, http://dx.doi.org/10.1109/IMCCC.2012.50.

[8] S. Ghosh, P. Bhattacharya, Analytical framework for measuring network security using exploit dependency graph, IET Inf. Secur. 6 (4) (2012) 264–270, http://dx.doi.org/10.1049/iet-ifs.2011.0103, URL: http://digital-library.theiet.org/content/journals/10.1049/iet-ifs.2011.0103.
[9] N. Poolsappasit, R. Dewri, I. Ray, Dynamic security risk management using Bayesian attack graphs, IEEE Trans. Dependable Secure Comput. 9 (1) (2012) 61–74, http://dx.doi.org/10.1109/TDSC.2011.34.
[10] J.B. Hong, D.S. Kim, T. Takaoka, Scalable attack representation model using logic reduction techniques, in: 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 2013, pp. 404–411, http://dx.doi.org/10.1109/TrustCom.2013.51.
[11] A. Zimba, H. Chen, Z. Wang, Bayesian network based weighted APT attack paths modeling in cloud computing, Future Gener. Comput. Syst. 96 (2019) 525–537.
[12] M. Khosravi-Farmad, R. Rezaee, A.G. Bafghi, Considering temporal and environmental characteristics of vulnerabilities in network security risk assessment, in: 2014 11th International ISC Conference on Information Security and Cryptology, ISCISC 2014, 2014, pp. 186–191, http://dx.doi.org/10.1109/ISCISC.2014.6994045.
[13] M. Kot, The state explosion problem, 2003, pp. 1–6, URL: www.cs.vsb.cz/kot/down/Texts/StateSpace.pdf.
[14] K. Kaynar, F. Sivrikaya, Distributed attack graph generation, IEEE Trans. Dependable Secure Comput. (99) (2015) http://dx.doi.org/10.1109/TDSC.2015.2423682.
[15] S. Noel, S. Jajodia, Managing attack graph complexity through visual hierarchical aggregation, in: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security, ACM, 2004, pp. 109–118.
[16] J. Homer, A. Varikuti, X. Ou, M.A. McQueen, Improving attack graph visualization through data reduction and attack grouping, in: Visualization for Computer Security, Springer, 2008, pp. 68–79.
[17] L. Williams, R. Lippmann, K. Ingols, An interactive attack graph cascade and reachability display, in: VizSEC 2007, Springer, 2008, pp. 221–236.
[18] FIRST, Common vulnerability scoring system, 1989, http://www.first.org/cvss.
[19] P. Mell, K. Scarfone, S. Romanosky, The Common Vulnerability Scoring System (CVSS) and Its Applicability to Federal Agency Systems, NIST Interagency Report 7435, 2007.
[20] N. Ghosh, S.K. Ghosh, An approach for security assessment of network configurations using attack graph, in: 1st International Conference on Networks and Communications, NetCoM 2009, 2009, pp. 283–288, http://dx.doi.org/10.1109/NetCoM.2009.83.
[21] Q. Liu, Y. Zhang, VRSS: A new system for rating and scoring vulnerabilities, Comput. Commun. 34 (3) (2011) 264–273, http://dx.doi.org/10.1016/j.comcom.2010.04.006.
[22] L. Allodi, F. Massacci, A preliminary analysis of vulnerability scores for attacks in wild, in: Proceedings of the 2012 ACM Workshop on Building Analysis Datasets and Gathering Experience Returns for Security - BADGERS '12, 2012, p. 17, http://dx.doi.org/10.1145/2382416.2382427.
[23] P. Li, X. Qiu, NodeRank: An algorithm to assess state enumeration attack graphs, in: 2012 International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM 2012, 2012, pp. 0–4, http://dx.doi.org/10.1109/WiCOM.2012.6478585.
[24] I. Kotenko, E. Doynikova, A. Chechulin, Security metrics based on attack graphs for the olympic games scenario, in: Proceedings - 2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP 2014, 2014, pp. 561–568, http://dx.doi.org/10.1109/PDP.2014.113.
[25] A.A. Younis, Y.K. Malaiya, Comparing and evaluating CVSS base metrics and microsoft rating system, in: 2015 IEEE International Conference on Software Quality, Reliability and Security, 2015, pp. 252–261, http://dx.doi.org/10.1109/QRS.2015.44.
[26] S. Jha, J.M. Wing, Survivability analysis of networked systems, 2001, pp. 307–317.
[27] T. Hughes, O. Sheyner, Attack scenario graphs for computer network threat analysis and prediction, Complexity 9 (2003) 15–18.
[28] J.M. Wing, Scenario graphs applied to security (summary paper), Scenario (2003) 1–6.
[29] O.M. Sheyner, Scenario graphs and attack graphs, Architecture (October) (2004) 3–4.
[30] V. Mehta, C. Bartzis, H. Zhu, E. Clarke, J. Wing, Ranking attack graphs, in: 9th International Symposium on Recent Advances in Intrusion Detection (RAID'06), Vol. 4219, 2006, pp. 127–144, http://dx.doi.org/10.1007/11856214_7, URL: http://www.springerlink.com/index/ML6MQ1653W672477.pdf.
[31] J.M. Wing, Scenario graphs applied to network security, Inf. Assur. (2008) 247–277, http://dx.doi.org/10.1016/B978-012373566-9.50011-2.
[32] N. Idika, B. Bhargava, Extending attack graph-based security metrics and aggregating their application, IEEE Trans. Dependable Secure Comput. 9 (1) (2012) 75–85, http://dx.doi.org/10.1109/TDSC.2010.61.
[33] Z. Chao, W. Huiqiang, G. Fangfang, Z. Mo, Z. Yushu, A heuristic method of attack graph analysis for network security hardening, in: 2014 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, 2014, pp. 43–47, http://dx.doi.org/10.1109/CyberC.2014.18.
[34] C. Wang, N. Du, H. Yang, Generation and analysis of attack graphs, Procedia Eng. 29 (2012) 4053–4057, http://dx.doi.org/10.1016/j.proeng.2012.01.618.
[35] S. Noel, S. Jajodia, Metrics suite for network attack graph analytics, 10 (2014) 2–5. http://dx.doi.org/10.1145/2602087.2602117. URL: http://csis.gmu.edu/noel/pubs/2014{_}CISRC.pdf.
[36] F. Zhao, H. Huang, H. Jin, Q. Zhang, A hybrid ranking approach to estimate vulnerability for dynamic attacks, Comput. Math. Appl. 62 (12) (2011) 4308–4321, http://dx.doi.org/10.1016/j.camwa.2011.09.031.
[37] U. Garg, G. Sikka, L.K. Awasthi, Empirical analysis of attack graphs for mitigating critical paths and vulnerabilities, Comput. Secur. 77 (2018) 349–359.
[38] U. Garg, G. Sikka, L.K. Awasthi, Empirical risk assessment of attack graphs using time to compromise framework, Int. J. Inf. Comput. Secur. (IJICS) (2019).
[39] US-CERT, Common vulnerabilities and exposures, 1999, URL: http://www.cve.mitre.org/.
[40] M. Keramati, A. Akbari, An attack graph based metric for security evaluation of computer networks, in: 2012 6th International Symposium on Telecommunications, IST 2012, 2012, pp. 1094–1098, http://dx.doi.org/10.1109/ISTEL.2012.6483149.
[41] S. Fayyad, C. Meinel, Attack scenario prediction methodology, in: Proceedings of the 2013 10th International Conference on Information Technology: New Generations, ITNG 2013, 2013, pp. 53–59, http://dx.doi.org/10.1109/ITNG.2013.16.
[42] M.A. Alhomidi, M.J. Reed, Attack graphs representations, in: 2012 4th Computer Science and Electronic Engineering Conference, CEEC 2012 - Conference Proceedings, 2012, pp. 83–88, http://dx.doi.org/10.1109/CEEC.2012.6375383.
[43] A. Yoran, Nessus vulnerability scanner, 2002, URL: https://www.tenable.com/products/nessus-vulnerability-scanner.
[44] M.A. McQueen, W.F. Boyer, M.A. Flynn, G.A. Beitel, Time-to-compromise model for cyber risk reduction estimation, Qual. Prot. (2005) 49–64, http://dx.doi.org/10.1007/978-0-387-36584-8_5, URL: http://link.springer.com/10.1007/978-0-387-36584-8{_}5.
[45] Stroke, Exploit database, 2004, URL: https://www.exploit-db.com/.

Urvashi Garg is Assistant Professor with Department of CSE in NIT Jalandhar. She received her BTech degree in CSE from the Kurukshetra University, Haryana, India, in 2010. After completing her Bachelor's degree, she had completed MTech in CSE from NIT Jaipur, Rajasthan in 2012. She has completed her PhD in CSE from NIT Jalandhar, Punjab, India, in 2019. Her current research interests include cyber security, parallel computing, information security and machine learning.

Geeta Sikka is an Associate Professor in the department of CSE in NIT Jalandhar. She has completed her MTech in CSE from Punjab Agriculture University in 2004 and PhD in CSE from NIT Jalandhar in 2013. She has more than 20 years of experience in teaching. She started working in NIT Jalandhar in 1996 as an Assistant Professor. She is the member of various societies and activities in NIT Jalandhar. She has attended various STC and workshop as session chair. She has organized various courses and workshops in the department of Computer Science and Engineering at NIT Jalandhar.

Lalit K. Awasthi is the Director at NIT Jalandhar. He has completed his M Tech in CSE from IIT Delhi in 1993 and PhD in CSE from IIT Roorkee in 2002. He has been an active faculty member at CSED, NIT Hamirpur and have contributed in various positions for 25 years. He had worked as Head of CSE Department and Dean (Students and Alumni) for quite good number of years. He is responsible for overall planning and execution of all functions related to establishment of a new Government Engineering College, Atal Bihari Vajpayee Govt. Institute of Engineering and Technology, Pragtinagar, India.