

Inter-Dependent Effect Of Vulnerabilities For Generation Of Effective Attack Path Score

Urvashi Garg¹ Sandeep Bansal¹ Deepak Prashar¹ Sanjeev Kumar¹

¹*Department of Computer Engineering,
Lovely Professional University, Jalandhar, India*
urvashi.garg.24@gmail.com, sandeep.15732@lpu.co.in, deepak.prashar@lpu.co.in,
sanjeev.18522@lpu.co.in

Abstract

Exploitation of vulnerabilities is often the choice of attackers for launching an attack that provides paths for gaining access of restricted resources. Usually a large number of possible attack sequences exist which can be used by attacker. Hence, rather than securing network by third party entity, a mechanism for identifying vulnerabilities access path using attack path score will trigger attack before happening. Common Vulnerability Scoring System (CVSS) provide severity level of existing vulnerability or efforts required to exploit vulnerability. CVSS assign score to each vulnerability ranging from 1 to 10. Attack sequences have multiple vulnerabilities, which has to be exploited to attack target machine. CVSS score is independent score of vulnerabilities, but interdependent effect of vulnerabilities is never addressed in CVSS. This paper proposes a technique to calculate score for attack path to network vulnerability severity level. Proposed technique considered inter-vulnerability effect by modifying CVSS score of vulnerability. With the increase of vulnerabilities, complexity in generation of attack graph increases. This paper also provides ease of development of attack graph with the help of Attack graph generator tool.

KEYWORDS ATTACK GRAPHS, VULNERABILITY, ATTACK MODEL, VULNERABILITY SCORE, ATTACK SEQUENCE

1. INTRODUCTION

Attack graph drawn a lot of attention of researcher due to its effectiveness in providing vulnerabilities exploiting information, graphically. Analysis of attack graph provides faster access to vulnerabilities available in network and the method

of how these vulnerabilities can be exploited. According to the study report conducted by Symantec, every year 4000-6000 vulnerabilities have been reported [2]. This signifies an increase in existence of vulnerabilities in network. Due to these many vulnerabilities, information provided by attack graph also has started overwhelming. In this paper, a technique to analyze network attacks graph and assess severity level of attack paths to prioritize critical vulnerabilities in network are addressed. Wyss et al [1] uses probabilistic Risk Analysis (PRA) technique for generation of attack graph. PRA will generate a network attack graph, but determination of attack path for a particular node is difficult. Also, it does not consider possibility of multiple attackers.

A large number of vulnerability scanners like nmap, nessus, nexpose, nikto [11] are available which can scan network and give details of host and vulnerabilities present in network like vulnerability ID, Host Operating system etc. Usually the information includes vulnerabilities exist on which machine of network. These scanners do not have any mechanism to show interdependency among vulnerabilities. Attack graph addresses this issue by giving information about interdependency among vulnerabilities. Attack graph shows, how vulnerabilities can be chained together to form attack. Thus attack graph is an important tool to find out vulnerable points in the network and help in making decision to patch high vulnerability points or provide another means to defeat attacker to reach his goal. The major extension in proposed method is that it can also generate separate attack paths for each host. Also, it can find multiple attack attempts for a particular host. Proposed method considers complete network for attack graph generation.

CVSS [12] scoring scheme is suitable score generation schema for providing score to each path, so as to get most vulnerable path. Cynthia et al [4] proposed a method based on shortest path algorithm to find most vulnerable path. At times, though path is shorter, launching exploits along that path are difficult than longer path. In such cases the algorithm for path score may not provide accuracy about attack path severity, as it uses weight criteria for determining cost of each edge. Sometimes attack path having only one edge with less CVSS score is very difficult to exploit as compared to attack path having more than one edge with high CVSS score. This paper addresses all these issues in providing efficient attack path score. Proposed method uses optimized path on the basis of score provided to each attack path. Although attack graph is useful technique, but for real-time networking environment, generated attack graph is quite large and makes it difficult for network administrator to analyze the graph. Attack path's score provides benchmarks to find which highly vulnerable attack path can be selected for securing. Network administrator can set a threshold for attack path's score and then consider only high score's attack path. This reduces domain of attack analysis, thus ease analyst's load.

Proposed attack graph generation technique requires pre-processing phase, which involve vulnerability scanning, managing vulnerability dataset and configuring firewall rules. Vulnerability scanning gives complete list of all vulnerabilities in each host (Refer Table 2), while datasets reveals vulnerabilities along with pre/post conditions. Firewall configuration file details about system accessibility through firewall. Also an algorithm is proposed that use CVSS metric to calculate scores for complete attack path through various hosts. This paper considers CVSS Base group

metrics for total attack path's score calculation. Base group's metrics of all vulnerabilities in attack path are compared and some technique is applied to get one among all the values in attack path. Base score formula is applied on these selected values and path score is determined.

2. PREVIOUS TECHNIQUES

A number of methods have already been developed for attack graph generation and vulnerability analysis:

CVSS: CVSS, was first published by US national infrastructure assurance council (NIAC), which was the first open vulnerability scoring system [12]. CVSS composed of three basic metrics: group base, temporal and environmental. Base score include metrics which does not change over time, temporal metric group measure properties of vulnerabilities that do change over time and Environmental metric group measures the properties of vulnerabilities related to user's real environment.

Topological: Sushil et al [10] proposed the idea of Topological vulnerability analysis approach for attack graph generation, which uses multi-step network penetration analysis. This technique provides some hardening measures of network using critical resources approach i.e. they will create some equations depending upon attack graph and then find out all critical resources of network that needs to be protected first, to prevent network attacks.

Probabilistic CVSS: Nirnay et al [3] proposed a technique that uses CVSS score for probability of exploits. CVSS base and temporal score calculates individual probability and then cumulative score, that is probability security metric. Attack resistance metric is also calculated, based on which network assessment is carried out. Attack resistance metric is reciprocal of probability security metric, which is a measure of efforts required in order to perform successful exploit.

Attack Template: Phillips et al [4] devised a new technique to generate attack graphs, which uses three types of inputs, attack template, configuration file and attacker profile. Attack template has conditions for exploiting vulnerabilities, i.e. operating system version, user access level. Network topology information is fed by configuration file while attacker profile includes attacker's capability. Nodes in attack graph represents attacker's state and edge represents state transition. Their approach also assigns some weight to edge known as success probability, time to succeed or cost of attack. Attack graph is generated backward from goal state to attacker state. Initially attack goal is specified and then algorithm checks for attack template to find matching attack state for transition. Each attack path's total cost is the sum of all edge's weight occurred in path. Finally low cost attack path is determined in order to provide defense mechanism. Success probability considers only one aspect of attack, either time to succeed or cost.

Model-Checker: Sheyner [5] proposed a model checking technique to generate attack graph. They used NuSMV model checker to generate attack graph. Attack graph analysis involves finding minimal critical set. Minimal critical set is a set of states, which is necessary to reach attack's goal state. Sheyner used probability to compute state transition likelihood. In their approach, some edges are assigned random probability and results in mixed attack graph known as probabilistic attack graph. Problem with their approach that only few of edges have probability score, some of them not assigned any score hence do not provide complete score for attack path.

MOEA: Pavan et al [6] suggested the idea of attacker defender policy. They have used memory based multi objective evolutionary algorithm (MOEA). A reward metrics is assigned to each player whether it is an attacker or defender depending upon who is the winner i.e. attacker if it attacks before defender patches the vulnerability and defender if it patches the vulnerability before attacker attacks the node. There is a cost associated with rule implementation at each node and time to be taken to implement rule on each node. In order to win, it is necessary to minimize total cost and time.

Probabilistic-Risk: Wyss et al [1] proposed a model that describes probabilistic risk analysis. In this method, initially find all the nodes that are directly reachable from attacker's node and these will be considered as child node. This child node is then further divided into sub nodes that will be directly reachable from child node and so on. This method uses probability technique to find most vulnerable attack path.

Vulnerabilities chaining form a base on which attacker launch attacks. However, above-mentioned techniques do not consider vulnerabilities interdependence effect (exploiting secondary vulnerability which is dependent of primary). Also, previous techniques do not provide any efficient scoring mechanism by which only most vulnerable attack paths can be considered first to minimize attack attempts. Proposed method resolves these issues by considering metrics of vulnerabilities involved in attack path.

3. PRE-REQUISITE OF ATTACK GRAPH GENERATION

This section explores pre-requisite for attack graph generation. Attack graph generation requires details of existing vulnerabilities in network, pre-conditions required to exploit vulnerability, post-conditions that is after exploiting vulnerability.

3.1. Vulnerability Scanning

First phase of attack graph generation is to find out vulnerabilities in existing network. Nessus vulnerability scanner has vast number of plugins for scanning vulnerabilities [13]. Scanning results are available in html, pdf, nessus and nessus version v1 format. Nessus version v1 is used in this paper. Nessus version v1 is in xml structure, where xml parser is used to extract vulnerability related information and save them in database for later use.

3.2. Vulnerability Dataset

Vulnerability dataset is vector containing 5 variables as shown below:

$\langle cve, pre-privilege, pre-resource, post-privilege, post-resource \rangle$.

Preconditions denote condition that must be satisfied to exploit the vulnerability and post conditions represents access level an attacker can gain, on successfully exploitation. Precondition information containing privilege and resource pre-requisition (for launching attack) is used in creation of attack graph. Post conditions specify privilege; an attacker will get on target machine plus resources. Below-mentioned Table 1 details the structure of dataset prepared in MySQL database, which describes preference information.

Table 1. Vulnerability Precondition Post-condition Dataset

CVE ID	Pre privilege	Pre resources	Post privilege	Post resources
2011-3650	Guest	Interact with exploit mechanism	Guest	Memory crash or DoS
1990-0520	Guest	NULL	Guest	Gain access to Shared files
2011-4517	Guest	Malformed JPEG	Root	Remote code execution
2012— 0450	Guest	Local Access	Guest	Invalid disclosure

- First column gives the Common Vulnerability Exposures (CVE) [9] of vulnerabilities. It is the unique identification provided for all vulnerabilities on the basis of publication date of vulnerability.
- Second column describes the privilege level required for attacker at the target system to exploit the vulnerability.
- Third column describes the resources or conditions that must be satisfied for exploitation of vulnerability. For example as shown in 4th row of Table 1, there should be local access to target system for exploitation of this vulnerability.
- Fourth column gives the privilege level that attacker can get on the target system after exploitation of vulnerability.
- Fifth column gives the output result or capability that an attacker gains after exploitation of vulnerability. For example 3rd row shows the capability to execute remote code on target machine.

3.3. Firewall Rule

Firewall configuration file have firewall rules which determine accessibility of machines outside of network. Firewall rules are specified in the following format:

```
<source_ip, source_port, dest_ip, dest_port, action>
```

Source ip and port specify incoming packet's ip and port number; dest_ip and dest_port specify ip address and port number for destination/target machine. Action specify action to be taken when packet come from source_ip and source_port for dest_ip and dest_port..

4. ATTACK GRAPH GENERATION TECHNIQUE

Nessus provides vulnerability scanning strength to Tool “Attack Graph Generator”. Proposed tool have nessus interface so that user can interact with nessus without using web browser. Nessus provides vulnerabilities scanning details in form of report, used by this tool. Parser will parse the report generated by Nessus. Existing vulnerability is matched against pre-condition. Attack graph generator fetch firewall rule from firewall configuration file and network vulnerabilities details from database and process all these information to generate attack graph.

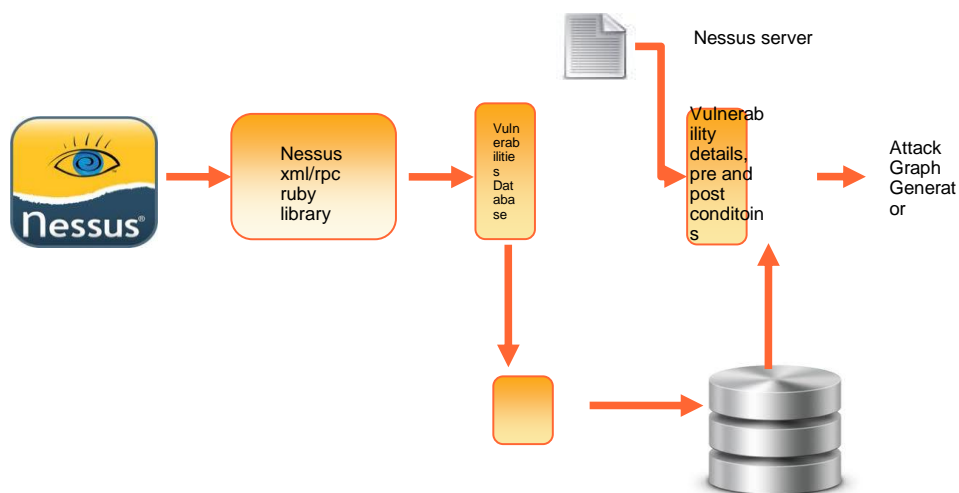


Figure 1 Attack graph Generator

1. **Nessus Report Parser:** In Nessus [13] report parser;nessus scan report is parsed and storesthese results into database. These results are consumed by attack graph generator tool for generating attack graph.
2. **Nessus Server Window Client:**An inbuilt capability to provide interaction with Nessus server [13] is provided. So as to start new scan or download report as per requirement. This window client uses a ruby library[14] that implements Nessus/XMLRPC protocol and provides communication between window client and Nessus server.

3. **Attack Rule Library:** Attack rule library contains pre and post conditions for vulnerabilities to exploit them. Pre/Post condition actually has only privilege level required before and gained after exploitation. Library also contains pre and post resources, preresource identify extra resource required for exploit and post-resource represents resources gained after successful exploit.
4. **Attack Graph Generator:** Forward chaining approach has been used for attack graph generation i.e given attacker capability and then find attack paths from attacker. It uses data stored by nessus report parser and attack rule library to generate attack graph.
5. **Firewall Rule:** firewall.conf file contains simple firewall rule to specify which host can be accessed from outside the network. It contains source ip, source port, destination ip, destination port and action.
6. **Attacker's Capability:** Attacker initial privilege level of entry, which is considered as guest.
7. **Attack Path Score:** CVSS (Complete Vulnerability Scoring Scheme)[12] score is used for attack path's score calculation using metrics value. These metrics are divided into three groups:
 - a) **Base metrics:** Characteristics of vulnerability that is constant over time and environment. For example access complexity, impact on confidentiality of target machine etc. Base score \$ can be computed from following equations:

$$\$ = \text{round to 1 decimal } (((0.6 * \alpha) + (0.4 * \beta) - 1.5) * f(\alpha))$$
 where $\alpha = 10.41 * (1 - (1 - \theta) * (1 - \psi) * (1 - \rho))$,
 $\beta = 20 * n * \beta * \square$,
 $f(\alpha) = 0$ if $\alpha = 0$; 1.176 otherwise
 where, α denote impact, β denote exploitability, θ denote confidentiality impact, ψ denote integrity impact, ρ denote availability impact, n denotes access vector, β denotes access complexity and \square denotes authentication. Base score is represented by \$.
 In these equations initial impact, which depends upon confidentiality, integrity and availability sub metrics, is estimated, and later the overall impact of the vulnerability is determined. Exploitability impact score depends upon Access vector, access complexity and authentication sub metrics. It describes the overall exploit capability of the vulnerability, where Impact function is the constant value. These three metrics are used for final calculation of Base Score.
 - b) **Temporal metrics:** These are optional metrics used because vulnerability may change over time.
 - c) **Environmental metrics:** It maintains the behaviour of metrics that are associated with environment. It is optional and can be skipped.

Attackers do seek assistance of intermediate nodes for launching attacks on target system, signifying intermediate nodes do contribute towards severity score. Proposed score calculation algorithm do consider metrics from all intermediate node. After the completion of attack graph, attack-path score is

compared to find out most severe (intense) attack path. In CVSS scoring, high scoring signifies high ease of exploitation, on the other hand, low score signify, huge complexity in exploitation of that particular vulnerability. Scoring provides means of prioritizing vulnerabilities in order of risk. Therefore to protect system from network attack, all these vulnerabilities need to be patched first. As new vulnerabilities are coming into existence daily, patching all vulnerabilities is not feasible. If critical vulnerabilities are patched, it reduces chances of successful attacks. Attack path's score provide basis on which highly vulnerable attack path can be distinguished. This score is also helpful in determining, how much vulnerable a system is.

4.1. ATTACK GRAPH GENERATION ALGORITHM

The process of attack graph generation is derived from 4 steps, each step is carried out with an algorithm given:

Algorithm 1 is used to initiate attack graph by taking Host IP address. It will add vulnerability list of that host to queue and call next step.

ALGORITHM 1: ATTACKGRAPHGENERATOR()

1. **Input:** IP address q of host
2. **Output:** Attack graph for a host.
3. for each vulnerability v on q do
 - Form a node n of $q, v, \text{privilege } \rho$
 - Add n to queue θ
4. end for
5. **GenerateSequence**(θ)

Algorithm 2 is used to generate attack path of given host. It checks the post-condition of each vulnerability of node fetched from queue and match it with precondition of each vulnerability of next node that can be exploited and if there is a match then it will add this node to list. This process continues till algorithm find node, which can be accessed from outside the network. If algorithm not able to find any such node then it call next step.

ALGORITHM 2: GENERATESEQUENCE(θ)

1. **Input:** θ : Queue of all nodes.
 H : Scanned Hosts list of network
 f : Flag Variable
2. **Output:** \mathcal{L} : List of nodes to be displayed
3. while $\theta \neq \square$ do
 - $\$ = \text{front}(\theta)$
 - $q = \$ \rightarrow \text{IP address}, \mathcal{P} = \$ \rightarrow \text{precondition}$
 - for $h \in H$ do
 - for each vulnerability v on h do

```

        if  $h$  can directly access  $q$  then
            if  $\mathcal{P} = v \rightarrow \text{postcondition}$  then
                Form a node  $n$  of  $q, v, \text{privilege } \rho$ 
                Add  $n$  to  $\mathcal{L}$ 
                Set flag  $\mathcal{f}$ 
            end if
        else
            if  $\mathcal{P} \neq v \rightarrow \text{postcondition}$  then
                Add  $n$  to  $\theta$ 
                Add  $n$  to  $\mathcal{L}$ 
            end if
        end if
        if  $v \langle \rangle \text{ set}$  then
            PostProcessing( $\$, \mathcal{L}$ )
        end if
    end for
end for
4. end while

```

Algorithm 3 takes each attack path and checks whether its leaf node is directly accessible from attacker node or not. If it is not directly accessible then it will remove that node and all above its ancestors for which they have single child, from the list.

ALGORITHM 3: POSTPROCESSING($\$, \mathcal{L}$)

1. **Input:** $\$$: Attack path sequence to be processed
 \mathcal{L} : List of all nodes.
 2. **Output:** list of nodes to be displayed.
 3. $n = \text{parent}(\$)$
 4. while $\$$ is the only child of n do
 - Remove $\$$ from \mathcal{L}
 - $\$ = n$
 - $n = \text{parent}(\$)$
 5. end while
-

4.2. SCORE CALCULATION ALGORITHM

Algorithm 4 compute score for each attack path. It will take base metrics values of vulnerabilities exploited at each node and compare it with another node on that path and take minimum values among them. Finally after using these values, it will apply Base score calculation formula and calculates score for that path.

ALGORITHM 4: COMPUTEATTACKPATHSCORE()

1. **Input:** ρ : All Attack paths of host.
 2. **Output:** Score of each attack path.
 3. Extract attack sequence $\$$ from ρ
 4. for each vulnerability v in $\$$ do
 - Fetch it's CVSS base metric value
 - if $v \neq$ root then
 - Compare every CVSS Base group's sub metric to previous one
 - Store minimum CVSS metric value
 - end if
 5. end for
 6. Calculate base score using minimum retained sub metrics values.
-

5. EXPERIMENTAL SETUP

Network of two hosts with one router is used for demonstration of attack graph generation, where each host has 2-3 vulnerabilities. These experiments are carried out on real-time machines, which are connected in LAN as shown in Figure 2.

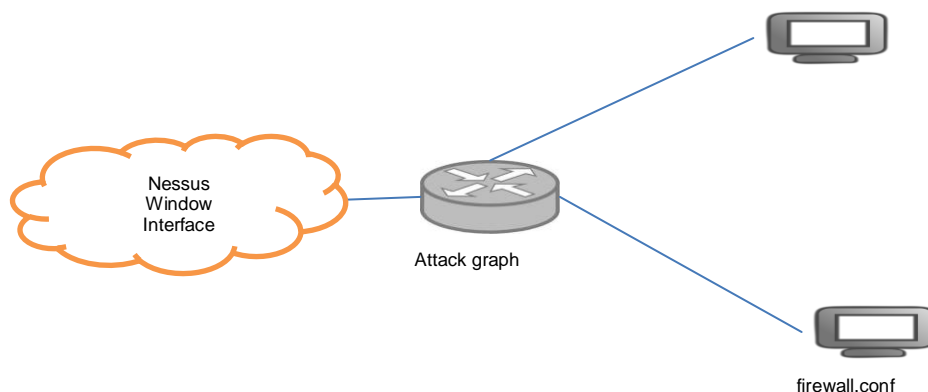


Figure 2: Experimental Setup

As shown in Table 2

CVE-2005-0044 [9] vulnerability allows attacker to execute arbitrary code. This vulnerability exists in Object Linking and Embedding (OLE) component; due to which attacker can gain root privilege after exploiting this vulnerability. Next vulnerability, CVE-2006-3747 [9] exists in apache server. This vulnerability causes Denial of Service attacks.

CVE-2011-4517 [9] vulnerability present in Jasper allows remote attacker to trigger a buffer overflow. Attacker can launch attacks for denial of service or privilege escalation.

CVE-2003-0449 [9] requires local access to exploit it. Local user can gain privilege by specifying path environment variable that will point to malicious library.

CVE-2006-0144 [9] vulnerability present in php pear 0.2.2, which allow remote attacker to execute arbitrary code.

CVE-1999-0511 shows that ip forwarding enabled on that machine. Fiel "firewall.conf" is configured to deny direct access to host 172.17.4.48. Now attacker has only access to 172.17.4.87 and 172.17.4.1.

Generated attack graph's node show host ip address, vulnerability id and user privilege gained by attacker. User levels are access privilege that an attacker can gain after exploiting vulnerability, also helpful in checking the preconditions level for next node to find chaining attack. At the end of each path a score for that path is determined, which measures the extent of vulnerability. This representation gives knowledge about both, severity level of attack path and vulnerability patching preference. Severity level of attack path determines patching preference i.e. which vulnerability should be patched first. If vulnerability appeared in attack path having higher severity level then it is assigned high patching preference.

Table 2. Attack Graph Nodes Description

Host IP address	Vulnerability	Operating System
172.17.4.87	CVE-2005-0044	Linux Kernel 2.6.38-13-generic, Ubuntu 11.04
172.17.4.87	CVE-2006-3747	Linux Kernel 2.6.38-13-generic, Ubuntu 11.04
172.17.4.87	CVE-2011-4517	Linux Kernel 2.6.38-13-generic, Ubuntu 11.04
172.17.4.48	CVE-2003-0449	Window 7 Ultimate
172.17.4.48	CVE-2006-0144	Window 7 Ultimate
172.17.4.1	CVE-2011-4517	Gateway
172.17.4.1	CVE-1999-0511	Gateway

6. RESULTS

Figure 3 depicts a subset of graph formed from above experiment, consisting of nodes and edges. Each node represents Host IP, CVE of exploited vulnerability and privilege level gained after successful exploitation. Each edge represents the next level in attack path and figures represent vulnerability score of each path. The path depicting 7.6 is the highest score and hence it is most vulnerable that can be easily exploited; also it can provide root level access on target Host. Finally vulnerability

score can decide that to attack target host, CVE-2006-3747 [9] is suitable to be exploited on intermediate host and after that CVE-2001-0144 [9] is exploited on target host. Hence these two are the vulnerabilities that need to be patched first to protect the system from riskier attack.

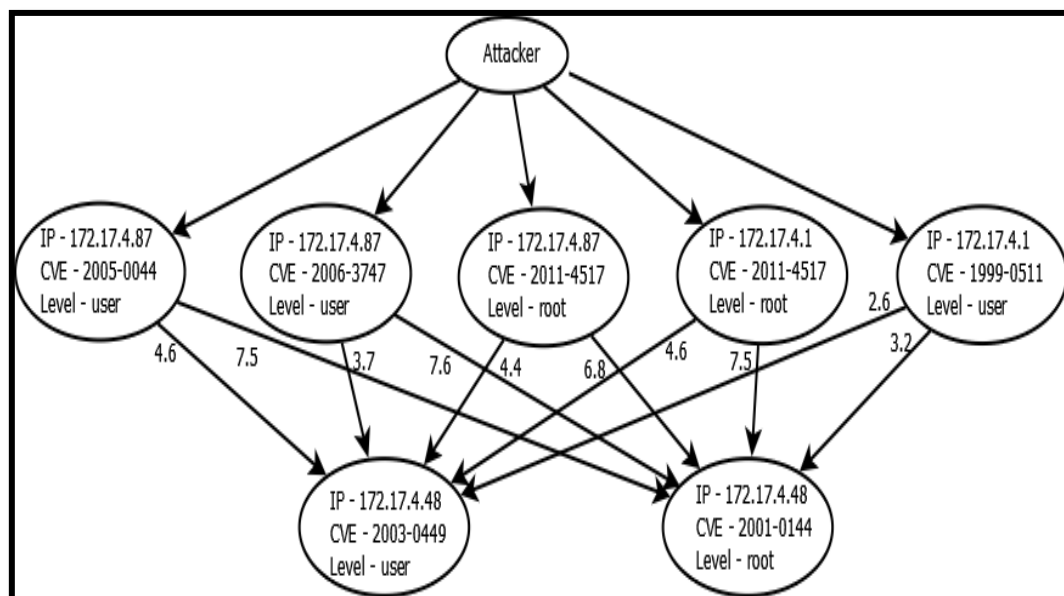


Figure 3. Attack Graph

7. CONCLUSIONS

The research presented here consists of development of a graphical tool to generate attack graph and provide score to each attack path for different attack models. User can easily choose optimal scored attack path and edges that are of interest. In addition to this, an administrator can focus on riskier paths for better security. It also records the undesirable activities that an attacker performed after entering in the network. It is a dynamic graph, which will work on current state of network. The most important and challenging task in this method is to create a dataset of all possible vulnerabilities. There are vast number of possible vulnerabilities reported and still new ones are continuing to be reported. Ideally formation of complete attack graph requires all feasible vulnerabilities that exist in network. Experimental dataset have vulnerabilities details, some of which are present in test network. The proposed method works effectively, if the attack graph is generated based on pre and post conditions validation of vulnerabilities. However, the method will cover all possible attack paths, once proper input is given to it. It also satisfies the needs of models that are used to create attack graphs and to calculate the vulnerability score for that graph. In this paper, an extension has been provided from simple network scanner to attack graph generation tool. It can scan whole network and generate vulnerability report and after that it can generate attack graph for each and every host separately and thereby entire network as well. It also provides a complete vulnerability score to each path,

which is helpful in analyzing the path. This research has provided a better scoring mechanism beyond CVSS scoring limits.

8. FUTURE WORK

Since our current state of work is restricted to only fewer vulnerabilities, hence its efficiency cannot be determined. As the vulnerability dataset scales, more accurate attack path to the target host will be generated. However, to get complete set of attack paths, dataset should have information of all vulnerabilities that are present in target node and intermediate nodes as well. It is very difficult to maintain the accuracy of this graph, due to the new vulnerabilities being added up on larger scale. This limitation is considered in future work so that dataset will automatically update.

REFERENCES

- [1] Wyss, Schriener, and T Gaylor, (1996), "Probabilistic Logic Modeling of Hybrid Network Architectures", *IEEE Conference on Local Computer Networks*, pages 404-413.
- [2] New vulnerabilities reporting "http://www.symantec.com/threatreport/topic.jsp?id=vulnerability_trends&aid=total_number_of_vulnerabilities"
- [3] Nirnay and S K ghosh, (2009), "An approach for Security Assessment of Network Configurations Using AttackGraph", *IEEE Conference on network and communication*, pages 283-288.
- [4] Cynthia Phillips and Laura Painton Swiler. Nspw '98, (1999), "Proceedings of the 1998", *Workshop on New Security Paradigms. ACM Trans. Program. Lang. Syst.*
- [5] Oleg Sheyner, Somesh Jha, and Richard Lippmann, (2002), "Automated Generation and Analysis of Attack graph", *IEEE Symposium on Security and Privacy*, pages 273-284.
- [6] Pavan Vejjandla, Dipankar Dasgupta, Aishwarya Kaushal, and Fernando Nino, (2010), "Evolving Gaming Strategies for Attacker-Defender in a Simulated Network Environment", *IEEE International Conference on Privacy, Security, Risk and Trust*, pages 889-896.
- [9] Cve Details: <http://www.cve.mitre.org/>, May 2012.
- [10] Sushil Jajodia and Steven Noel (2007), "Topological Vulnerability Analysis", *A Powerful New Approach For Network Attack Prevention, Detection and Response. World Scientific Press.*
- [11] https://www.owasp.org/index.php/Category:Vulnerability_Scanning_Tools
- [12] <http://www.first.org/cvss/cvss-guide.html>
- [13] Nessus: <http://www.nessus.org/>, May 2012.
- [14] <http://nessus-xmlrpc.rubyforge.org/> 14 nov 2013

