



University of Piraeus
School of Information and Communication Technologies
Department of Digital Systems

Postgraduate Program of Studies
MSc Digital Systems Security

Use and analysis of Attack Graphs for decision making in a
Cybersecurity Defense Plan

Supervisor Professor: Christos Xenakis

Giannis Kalderemidis

jkalderemidis@gmail.com

mte1910

Piraeus
29/12/2020

Acknowledgments

I would like to thank the Supervising Professor and Director of the MSc Christos Xenakis and my colleague Aristeidis Farao for their guidance and continuous support throughout the elaboration of this thesis. I also am grateful to Professors Stefanos Gritzalis and Konstantinos Labrinoudakis for their notes that helped largely in its development, as well as individually all the professors of the MSc that with their contribution each helped to the acquisition of my knowledge during my studies.

Abstract

Modern information systems tend to encounter threats that evolve and become more sophisticated than someone can keep up to by just knowing their existing vulnerabilities. Attack graphs are the visual display of such potential courses of action an adversary may take in order to lead to a system compromise and this document presents how one may generate a graph, draw results from it, and create a defense based on these conclusions. Then, driven by international standards, we delve through the process of risk management and with the restriction of a network owner's budget, allocate the existing resources to the best of the business or organization by using various methods including knapsack problem and game theory.

Περίληψη

Τα σύγχρονα συστήματα πληροφοριών τείνουν να αντιμετωπίζουν απειλές που εξελίσσονται και γίνονται πιο περίπλοκες και εκλεπτυσμένες από ό, τι κάποιος μπορεί να ανταπεξέλθει γνωρίζοντας απλώς τις ευπάθειές τους. Οι γράφοι επιθέσεων είναι η οπτική απεικόνιση πιθανών πορειών δράσης που μπορεί να λάβει ένας επιτιθέμενος προκειμένου να οδηγηθεί σε έκθεση του συστήματος σε κίνδυνο, και αυτό το έγγραφο παρουσιάζει πώς μπορεί ένας διαχειριστής συστήματος να δημιουργήσει έναν γράφο, να αντλήσει αποτελέσματα από αυτόν, και να δημιουργήσει μια άμυνα βάσει των συμπερασμάτων που εξήχθησαν προς αποφυγή τέτοιων καταστάσεων. Στη συνέχεια, με γνώμονα διεθνή πρότυπα, εξετάζουμε τη διαδικασία διαχείρισης κινδύνων και με τον περιορισμό του προϋπολογισμού του κατόχου δικτύου, διαθέτουμε τους υπάρχοντες πόρους στο καλύτερο δυνατό τρόπο προς όφελος της επιχείρησης κάνοντας χρήση ποικίλων μεθόδων όπως το πρόβλημα του σακιδίου και τη θεωρία παιγνίων.

Table of Content

Chapter 1. Attack Graphs	6
1.1 Motivation and Contribution	6
1.2 Introduction to attack graphs	7
1.2.1 Risk Management Procedure	8
1.2.2 How attack graphs are used in cybersecurity	10
1.3 Attack graph generation approaches	10
1.3.1 Enumeration Based Approach	11
1.3.2 TVA (Topological Vulnerability Analysis) approach	12
1.3.3 NetSPA (Network Security Planning Architecture) approach	13
1.3.4 Logic Programming Based approach	14
1.4 Attack graph approaches Comparison	14
Chapter 2. Creating an attack graph	19
2.1 MulVAL Installation	19
2.1.1 XSB installation	19
2.1.2 Graphviz tool installation	20
2.1.3 MySQL installation	20
2.1.4 MulVAL installation	20
2.1.5 Bison & Flex installation	21
2.2 Using MulVAL	21
2.2.1 Setting up the network	21
2.2.2 Datalog expressions	22
2.2.3 Generating the graph	25
Chapter 3. Defense Strategy	28
3.1 Graph Analysis	28
3.2 Risk Estimation	31
3.2.1 Node Rating/ Risk Estimation Formula	31
3.2.2 Measuring the values	32
3.2.3 Attack simulations	33
3.3 Choosing the most competent defenses	34
3.3.1 Cybersecurity Controls	34

3.3.2 0-1 Knapsack Problem	35
3.3.3 Defense approaches	36
3.3.4 Random Based Defense Approach	37
3.3.5 Game Theory Based Defense Approach	38
3.3.6 Results	41
3.3.7 Comparison	44
3.3.8 Other methods	45
Chapter 4. Conclusions	47
4.1 Analysis	47
References	49
Appendix A	53
Troubleshooting	53
javac: command not found	53
Bison, flex no available candidates	56
Appendix B	59
Python Scripts	59

List of Tables

Table 1 - Advantages and Disadvantages of each approach	14
Table 2 - Proposed usage of graph generation approaches	18
Table 3- Attack Scenarios Results	43
Table 4 - Defense methods Advantages and Disadvantages	44

List of Figures

Figure 1 - Risk Management process	8
Figure 2 - Attack graph toolkit architecture	17
Figure 3 - Path after environmental variable configuration	20
Figure 4 - MulVAL architecture	22
Figure 5- Network Setup	23
Figure 6 - Datalog system setup	23
Figure 7 - Interaction Rules	24
Figure 8 - run.P file	26
Figure 9 - Generated attack graph	26
Figure 10 - Square Nodes	28
Figure 11 - Circular and Diamond Nodes	29
Figure 12 - Direct network access Interaction Rule	29
Figure 13 - First step of the attack: Remote exploit of server program	29
Figure 14 - Remote exploitation Interaction Rule	30
Figure 15 - Simplified Attack Graph	30
Figure 16 - Return on investment table	35
Figure 17 - Simplified Graph with control options	37
Figure 18 - Random attacker results vs Random attacker Controls	38
Figure 19 - Game Theory attacker vs Random attacker Controls	40
Figure 20 - New possibilities of attack occurrence (Po) (Game Theory attacker)	41
Figure 21 - Random and Game Theory attacker vs Unguarded System	42
Figure 22 - Random attacker and Game theory attacker vs Random based defense	43
Figure 23 - Random and Game theory vs Game theory based defense	44

Chapter 1. Attack Graphs

Building a network defense is a process that most times requires more than knowing and patching its vulnerabilities. Attack graphs are a great way to provide insight that can contribute to this process but since organizations and enterprises face challenges like restricted budget or limited resources, they are called to make decisions that sometimes include the retention of certain risks. This document tries to give a guideline on how a network manager can manage the given resources and distribute them properly based on conclusions extracted by the risk management process and the examination and analysis of the generated attack graphs.

1.1 Motivation and Contribution

A big concern of many network owners is how someone can use a limited amount of budget and resources to protect his or her system. There are a plethora of tools that specify the vulnerabilities in a network, but these tools usually do not take into consideration other crucial factors like their interdependence with other assets, conditions and requirements that may potentially affect the system state during an incident, or even human behavior inside or outside of the network which makes it more difficult to assess their criticality and how urgent is the acquisition of a countermeasure. Factors like the complexity of the attacks that involve many assets and preconditions, system states that change dynamically during an incident, and even non-directly, non-computable system threats via vulnerability assessment tools (like insider threat and social engineer attacks) are problems we are called to confront with the use of attack graphs and their analysis.

Also, after the evaluation of the vulnerabilities, the network owners are called to create a cybersecurity plan that has to be implemented in accordance with the previously mentioned limitations, but also the size of the network, business or organization since each one of them require different treatment options. Decision-making in this process is crucial especially when the risk owner is called to retain some of the existing vulnerabilities.

With that in mind, we try to implement a universal method of evaluation that would help the systems defense generation regardless of these parameters. More specifically, things that we delve into in this research include:

- The examination of the risk management process with the use of attack graphs.
- Different graph generation methods and approaches comparison.
- Presentation of a logic graph-generating tool and its programming language.
- A way to analyze and extract information from attack graphs.
- Evaluation and prioritization of existing system vulnerabilities.
- Generation of a cybersecurity plan based on results drawn by attack graphs.

In chapter 1, in particular, we provide some insight on how attack graphs can be used in cybersecurity, what automatic graph-generation techniques exist and how they differ, and we present a short guide on which approach should an administrator use depending on the scale of the network. Then, chapter 2 demonstrates the process of generation of an attack graph on a simple network, in chapter 3 we create different defense approaches based on a proposed formula (which can be also implemented on larger networks) that we next compare in a scenario of 6000 attacks in a simulated environment that matches our network's state. Finally, chapter 4 concludes with the accumulated results of the master's thesis and their analysis.

1.2 Introduction to attack graphs

As most enterprises and organizations continuously tend to virtualize their assets, their defense becomes more and more crucial and challenging at the same time. While vulnerability scanning, intrusion detection systems, and vulnerability assessment tools provide enough defensive mechanisms for the hosts of a network, sometimes an attacker may launch a multistage attack that bypasses the said defenses.

Mitigating these attacks requires a comprehensive, end-to-end approach that includes creating and maintaining security policies based on the security needs of the system in consideration. The first step in establishing the security needs of an organization is to identify likely threats and perform a risk analysis, the results of which are used to determine the security hardware and software implementations, the mitigation policies, and the network design [25]. Attack graphs can provide part of this

information by representing the system state, vulnerabilities, and potential exploits an attacker may take advantage of to compromise critical data [3].

1.2.1 Risk Management Procedure

A major part of a cybersecurity control plan is risk analysis and risk management. Although, prior to the graph generation, the network administrator should determine the assets, the risks, the threats, and the risk owners involved, and then proceed to further actions. According to [29], “Risk management process is a systematic application of management policies, procedures, and practices to the activities of communicating, consulting, establishing the context, and identifying, analyzing, evaluating, treating, monitoring and reviewing risk”. ISO/IEC 27005 [28] provides guidelines for the implementation of information security based on a risk management approach. Figure 1 displays the procedure of risk management according to the international standard.

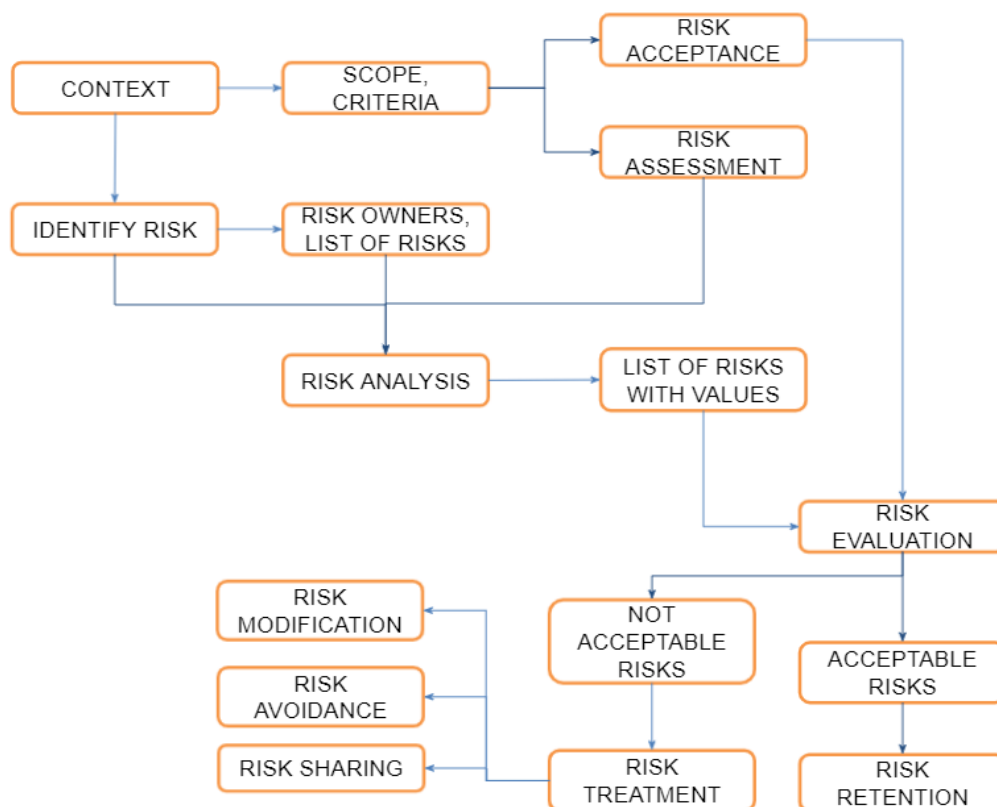


Figure 1 - Risk Management process

Beginning from the context establishment, the organization has to specify the standards that will set the wanted results after the procedures are made. Then, risk identification is a process that includes the source or sources of the risk, the potential area of impacts, and causes or consequences that may occur. In terms of a network defense strategy, it is mainly defined by potential system vulnerabilities or other factors that may lead to a system compromise. Techniques that may extract these system susceptibilities by testing and evaluating the system security are [25] :

Penetration Testing is an attack simulation, authorized by the system owners, that is used to evaluate its security. Types of pen tests vary (white box, black box) and the ones performing them usually use the same tools an attacker might do. Some tools used in penetration testing are Metasploit Framework, Aircrack-ng, Nikto, John the Ripper, etc.

Virus Detection and anti-virus software are used during the risk identification process to prevent, detect, identify, and remove malware on a system.

Network Scanning is the process of recognition of available network services, filtering mechanisms, and operating systems in use, in order to manage, maintain and secure the system using data found by the scanner. It also includes software that listen to open TCP ports displaying the available resources in a network. Nmap/Zenmap, SuperScan, and Port Scanner are three tools that can give this kind of information and be used to strengthen network security.

Vulnerability Scanning is a sub-branch of network scanning that detects and classifies system deficiencies within it. The vulnerability report outputs, in particular, will later be used as inputs for the generation of the attack graphs as they map most potential weaknesses an attacker may exploit during an incident. *Open Vulnerability Assessment Language (OVAL)* is also used by many compatible scanners and specifies the discovered vulnerabilities in XML that can be converted for the graph generator to use them [18] (for example Datalog clauses in MulVAL which is a Prolog based language explained in chapter 2). Tools that are mainly used for vulnerability scanning are, Nessus, OpenVAS, BurpSuite, etc.

After defining the risks, they need to be evaluated and then specify whether they can be retained, modified, or treated. Our work focuses mostly on the process of Risk analysis and the evaluation and management of the existing network threats.

1.2.2 How attack graphs are used in cybersecurity

For the sake of network security improvement, an administrator needs to constantly evaluate the system and preserve a safe state which can be challenging as many matters can incommode this task. Some of those matters are the increasing number of vulnerabilities (more than one hundred each week [39]) that make it difficult to track for each separate host and operation system and the complexity of the cyber attacks that use multi-host and multi-step techniques which cannot be detected via current attack detection methods (like modern Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS)) [3].

After the designation of the assets and the defense mechanisms owned by the network in consideration, attack graphs are used to describe each course of action an attacker may take, the system state in each step, and requirements that need to be met to reach a final goal state that usually indicates the success of the attack. Attack graphs consist of nodes and edges whose interpretation and representation differ depending on their types. Usually, the nodes of an attack graph are the system states and the edges refer to their transition created by various pre and post conditions[6].

Automatic security calculations like attack graphs can provide additional insight concerning those matters by accurately computing possible courses of action (CoA) that someone might take to compromise a network asset and give information that cannot be obtained from other network tools. In the next section, we analyze the main graph generation approaches and how they differ in their results and complexity depending on the inputs and the approach.

1.3 Attack graph generation approaches

Creating an attack graph from scratch can be pretty chaotic, given the assets that have to be protected in a network and the increasing number of attacks and vulnerabilities threatening more than one of the said components, the complexity increases exponentially as a network widens. To avoid the hand-made creation of such an attack graph, four major ways of automated generation have been presented [3].

- **Enumeration Based approach** nodes depict the state of the system during an attack including the entities involved in the attack, user privileges, and the outcome of the attack till this point.

- **TVA (topological vulnerability analysis) approach** focuses on the system's vulnerabilities instead of taking into consideration every possible threat. Then, by analyzing them, it defines the courses of actions the attacker can take to compromise the wanted network assets.
- **NetSPA Approach** identifies the most critical attack paths by analyzing the network topology. It is a multiple-prerequisite graph that includes state, precondition, and vulnerability nodes and gives the administrator the ability to visualize and repair the most threatening elements in the network.
- **Logic Programming Based Approach (MulVAL)** is a logic-based approach whose nodes represent logical statements and require that the cause of an attacker's potential privileges be expressible as a propositional formula in terms of network configuration parameters. [26]

1.3.1 Enumeration Based Approach

The first approaches that were used to create automatically-generated attack graphs are the state enumeration-based ones. The state enumeration graphs were introduced in 1998 where nodes represented the state of the system (information on hosts, access levels, consequences, etc) at a given time during an attack and edges show how the actions of the attacker can affect this state, based on other parameters like the success possibilities of the attack or the time needed for it to be successful [2]. Inputs used for the automated generation of the graph are:

- **Attack Templates:** Attack templates include the steps and requirements that are needed to be met for an attack to be successful. The defender has to take into consideration most known attacks that threaten the system but also involve scenarios that may not be popular but can be used to achieve a goal for the attacker. This creates many smaller graphs that depict how each one of them can either threaten or enable an unwanted state for the defender
- **Configuration File:** A configuration file consists of all information that describes the network. Hardware (routers, hosts, servers), network topology, group policies, and other aspects that may be of use during an attack for both the attacker and the defender.

- **Attacker Profile:** As each entity involved in an attack has different goals, needs, and assets that can be applied, the attacker profile describes those characteristics.

After the data input, the algorithm that generates the attack graph uses them to get into an initial state. It then compares the data and conditions from the attack templates and combines them with the system and network configurations. Finally, it repeats this process for each attacker profile and generates the graph.

As stated earlier though, the overabundance of existing threats and the complexity of the modern networks make it really difficult to apply this approach provided that the authors only used small-scale examples for the implementation of the algorithm. To improve that, Ritchey and Ammann proposed the use of Symbolic Model Checking (SMV) to simplify the model.

Model-checking is a method that specifies whether a finite-state model of a system meets the wanted requirements. SMV, in particular, takes the network configurations as input and performs a symbolic model checking of CTL(computational tree logic) formulae in order to verify whether the given properties are satisfied.

By using the model checker, the only input that's modified is the configuration file where a new parameter is added that defines a desirable state of our system. Moreover, the file contains the vulnerabilities and the connections of the hosts included, and the state of the attacker. Then, the model checker will provide any state-altering circumstances and their requirements, firstly creating the paths and then the attack graph.

1.3.2 TVA (Topological Vulnerability Analysis) approach

A major disadvantage of the enumeration-based approach is the exponential augment of the graphs in larger-scale networks. In [27] a new parameter is taken into consideration that is the variety of the attacker's behavior and the correlation of the targets involved in the attack (sometimes a denial of a service implies the inability of access to another). A graph generated in such a way lowers the complexity from exponential to polynomial and is known as exploit dependency graph [2]. Although, an attack path created this way can appear more than once in the graph and there can be paths that are not dependent on other pre-existing states. The nodes in this model

consist of exploit and security condition ones and the edges define the pre and post conditions in this state of the attack.

This approach differs from the previous techniques in a way that instead of examining each threat to a given system, it just delves into the existing vulnerabilities which can be targeted to affect criticality for the network assets. The advantage it brings is the lowered computational power needed for its generation but on the downside, it depends on the knowledge of the attacks that can result in a system compromise, to a level that they can be expressed in pre and post conditions so that they can be used as inputs to the graph generation algorithm.

1.3.3 NetSPA (Network Security Planning Architecture) approach

A new approach that has a linear increment depending on the range of the network, is the NetSPA approach [17]. The model's generation is based on the open ports that exist in a network's hosts. Network traffic rules, group policies, connections with other hosts, and existing vulnerabilities are the main inputs for this graph generation approach. After the data evaluation, the graph is created which consists of three types of nodes.

- **State Nodes:** This type of node represents the access level of the attacker on a host. This can be user, root, DoS, or other effects related to the confidentiality or integrity of system files.
- **Prerequisite Nodes:** As stated by the name, the preconditions of one or more attacks are expressed in these nodes. They also contain information on the reachability of the hosts and their groups.
- **Vulnerability Instance Nodes:** Vulnerabilities in this approach include not only insufficient protection against certain types of known threats but every possible way that an attacker can gain access to the system (software flaws, server misconfigurations, trust relationships, etc).

Also, since graph evaluation can prove to be a challenging task, the authors of [17] have proposed two ways to reduce its complexity. These are the automatic graph simplifications and recommendation generation. The first focuses on the reduction of the graph range by dropping similar nodes while the other tries to extract crucial information from the graph and present it to the users. By using vulnerability scanning tools like Nessus, the attack graph is generated according to rules stored in the tool's

database, and NetSPA determines whether a vulnerability is critical in a potential system compromise attempt. NetSPA runs offline using the given parameters, in order to avoid the leak of information to the attacker.

1.3.4 Logic Programming Based approach

The last category of graph generation tools is the Multihost, Multistage Vulnerability Analysis (MulVAL) [18] and we will use it to create an attack graph and analyze it in the next chapter.

Logical attack graphs use Datalog tuples to interpret the system configuration and contrary to its predecessors, it doesn't include the whole network state. After describing the system properties (vulnerabilities, network topology, etc), the reasoning engine XSB creates all possible scenarios that could lead to a specified state.

Each node in the graph is a logical statement and this statement reflects just an aspect of the whole network. The edges define the causality relations between network configurations and the attacker's capability to benefit from them. In short, whereas till now the graphs represented "how the attack can happen", logical attack graphs show "why the attack can happen". Although, a downside according to [18], is that attack conditions that cannot be expressed in propositional formulas cannot be captured by logical attack graphs. More on logical attack graphs, MulVAL, and their architecture will be presented in the second and third chapters.

1.4 Attack graph approaches Comparison

This part compares the discussed graph generation approaches and proposes where each one of them can be used most accurately. Although, before comparing the approaches, we should first recap the advantages and disadvantages each one of them offers. Starting with the enumeration-based approach, the advantages limit to its seniority compared to the other methods. It is slower than the other methods and has exponential complexity for larger networks making it incompetent for most cases. TVA partially fixes the problem of complexity and speed but its focus on the system vulnerabilities requires the network owners to have knowledge over all known attacks that can lead to the exploitation of each asset. This means that system owners should express the steps of each attack to pre and post conditions for the graph to be generated which is extremely difficult (especially in larger networks and

organizations) and needs constant maintenance and update to keep up with new upcoming threats. NetSPA solves the complexity problem for up to 50000 hosts and has a feature that evaluates the most vulnerable paths. Although, since the connectivity between hosts is given manually as an input, there is a chance that certain correlations may be ignored which can lead to partially false results (and nodes), and also graphs that include many prerequisite nodes can be confusing in larger networks since they may create loops difficult to interpret. Like TVA, NetSPA is difficult to obtain since they both are academic projects and were developed at George Mason University and MIT respectively, meaning they are not open to the public. Finally, MulVAL creates a fully detailed representation of the attack and its steps by demonstrating the interdependencies between system configurations, potential threats, and the vulnerabilities of the system. Its complexity can be at worst exponential but a downside is that the attack conditions of an attack have to be expressed in propositional formulas in order for the logic engine to capture them. The table below gives a synopsis of the discussed advantages and disadvantages.

Table 1 - Advantages and disadvantages of Graph Generation approaches

Graph generation approach	Advantages	Disadvantages
<i>Enumeration based</i>	<ul style="list-style-type: none"> • First approach in automatic graph generation 	<ul style="list-style-type: none"> • Other methodologies excel in many aspects
<i>TVA</i>	<ul style="list-style-type: none"> • Much faster to implement on a system • Less computational complexity (algorithm of Ammann $O(N^6)$ and later $O(N^3)$) 	<ul style="list-style-type: none"> • Depends on the knowledge of the known attacks that can achieve the wanted result • Nodes and edges can appear more than once in the graph • Difficult to obtain

<p><i>NetSPA</i></p>	<ul style="list-style-type: none"> ● Generation methods scale roughly as $O(N \log N)$, it can perform successfully for up to 50000 hosts ● Evaluates critical attack paths 	<ul style="list-style-type: none"> ● Reachability of hosts can be challenging to compute ● Multiple-prerequisite graphs have many loops making it difficult to interpret ● Difficult to obtain
<p><i>MulVAL</i></p>	<ul style="list-style-type: none"> ● Clearly cites how different network configurations affect the potential privileges of an attacker before, during, and after the attack, as well as the interdependencies between the vulnerabilities. ● Since the size of a MulVAL-generated graph is polynomial in the size of the network, the graph size can be at worst exponential to it ● Free and available to the public 	<ul style="list-style-type: none"> ● Worst-case running time: $O(N^2) \sim O(N^3)$ complexity for up to 1000 hosts with up to 100 vulnerabilities ● Attack conditions that cannot be expressed in propositional formulas cannot be captured by logical attack graphs.

The tools selected for the comparison are all free to use but NetSPA and TVA are difficult to obtain as explained earlier. Also, since enumeration-based attack graphs are considered outdated, for their replacement we added the Attack Graph Toolkit which creates state attack graphs and is the closest to enumeration-based approach architecture [22]. Attack graph Toolkit (state diagram) is obtainable via download on [40].

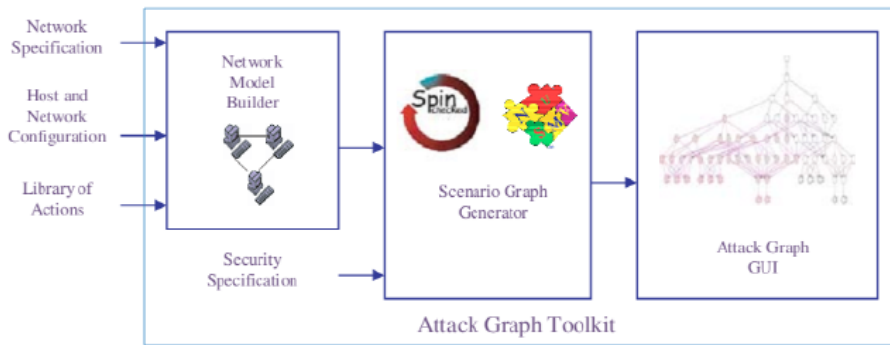


Figure 2- Attack Graph Toolkit Architecture

[22] also makes a comparison of academic graph generating projects (MulVAL, NetSPA, TVA, and Attack Graph Toolkit) with other commercial tools but for the ones we take into consideration, small networks (like a small office) with minimal security requirements and assets involved, all graph generation approaches seem to work efficiently but while more network factors appear, the exponential scalability of Attack Graph Toolkit deprives its competence to cope with the increased numbers (can only be used in Small Offices and business that have only a few resources that require protection). Finally, TVA, NetSPA, and MulVAL scale the best within large organizations. Specifically, organizations that choose the TVA approach, require the network administrator to have perfect system and vulnerability knowledge in order to set the right conditions and results of potential exploits (continuous penetration tests and vulnerability assessment may help in this task). Although, since the generated attack graph may prove challenging to read and interpret, NetSPA is best suited for networks that surpass 1000 hosts with the requirement that the interdependencies are expressed correctly. Lastly, MulVAL is overall acceptable on every occasion while contrariwise to the other tools is easily available to the public.

Table 2 - Proposed usage of graph generation approaches

Graph Generation Approach	Open-source	Scalability	Small Office / Home Office	Small Business	Large Enterprise/Organization
<i>Attack Graph Toolkit</i>	Yes	Exponential	✓	-Depends on the number of assets (fewer assets mean better implementation)	-Can't scale well
<i>TVA</i>	No	Polynomial: $O(N^3)$	-Depends on admin knowledge of attacks that can exploit the existing vulnerabilities	-Depends on admin knowledge of attacks that can exploit the existing vulnerabilities	-Depends on admin knowledge -Graph difficult to read
<i>NetSPA</i>	No	$O(N \log N)$	✓	✓	✓
<i>MulVAL</i>	Yes	Polynomial: $O(N^2) \sim O(N^3)$	✓	✓	✓

Chapter 2. Creating an attack graph

After comparing the most known methods of automatic graph generation, we will now generate an attack graph with the use of the MulVAL tool. We chose this approach because it fits better with most types of networks and clearly shows how different network configurations can affect the attacker's actions with its state nodes and the calculated interdependencies. This chapter consists of an installation guide of the logic engine and the graph generator, a part where we present and solve problems that may come up upon installation, a quick introduction to the PROLOG language that is used in MulVAL, and finally the generation of the graph which will later be analyzed. A file with the virtual machine which has the MulVAL installed will also be available for the readers.

2.1 MulVAL Installation

For the installation of the tool, we used Ubuntu and created a guide including the steps, software, and configurations needed in order for it to be functional. MulVAL uses a logic engine, graphics software, and a database management system whose information and issues encountered during the installation will be provided below.

2.1.1 XSB installation

XSB is a Logic Programming and Deductive Database system for Unix and Windows using Prolog and developed at Stony Brook University. The XSB Prolog System supports standard Prolog functionality and has been augmented by a powerful technique known as tabling, which greatly increases its applicability. In short, tabling (also called memoization or lemmatization) has the characteristic that each procedure is called only once, "remembers" everything that it returns and if it's ever called again, it uses the previous computations to satisfy the request.

After downloading the file from [36], we have to unzip it, navigate the path, and execute configure with the following command.

```
./configure
```

At this point, we have to set up the path to the XSB folder. To do this simply use:

```
export PATH=(XSB path)/bin:$PATH
```

Then to compile execute **./makexsb**

A problem you may encounter is the *javac: Command Not Found* whose solution is described in the troubleshooting section (*troubleshooting: javac not found*).

2.1.2 Graphviz tool installation

Graphviz is an open-source graph visualization software. It is mostly used for structural information representations as graphs or diagrams and it is the main way of MulVAL to depict the graph after the logic engine produces it. For its installation use

```
sudo apt-get install graphviz graphviz-doc
```

2.1.3 MySQL installation

MySQL is an open-source database management system, commonly installed as part of the popular LAMP (Linux, Apache, MySQL, PHP/Python/Perl) stack. It uses a relational database and SQL (Structured Query Language) to manage its data and is used for storing by MulVAL to import files from an enterprise network.

For the installation run the following orders:

```
sudo apt update  
sudo apt install mysql-server  
sudo mysql_secure_installation
```

If you use Kali Linux and don't want to use MySQL due to its support reduction, alternatives to MySQL are SQLite, ArangoDB, MariaDB, and PostgreSQL.

2.1.4 MulVAL installation

To install MulVAL, first, decompress the downloaded file in the same path with XSB, and then we need to set the environmental variable MULVALROOT point to the following folders. These are the commands needed:

```
export MULVALROOT=/home/tools/mulval  
export PATH=$MULVALROOT/bin:$MULVALROOT/utils:$PATH
```

The path should look like the figure below in order for it to be compiled successfully

```
root@john:~# echo $PATH  
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin  
root@john:~# export PATH=/home/tools/XSB/bin:$PATH  
root@john:~# export MULVALROOT=/home/tools/mulval  
root@john:~# export PATH=$MULVALROOT/bin:$MULVALROOT/utils:$PATH  
root@john:~# echo $PATH  
/home/tools/mulval/bin:/home/tools/mulval/utils:/home/tools/XSB/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin  
root@john:~# █
```

Figure 3- Path after configuration

For its compilation, we need to access the installation folder and use the command **make**

If everything is set up correctly, MulVAL will be successfully installed. Solutions to possible problems that may come up during the installation will be provided in Appendix A.

2.1.5 Bison & Flex installation

Flex and Bison are the modern equivalents Of Lex and Yacc. More specifically, Bison (or GNU Bison) is a parser generator that reads sequences of tokens and determines whether the sequence in consideration complies with the syntax specified by the grammar. Flex on the other hand is a lexical analyzer that provides the tokens to Bison after it converts the inputs.

To install simply use:

sudo apt install bison flex

Again, problems that may appear in the installation are resolved in Appendix A.

(troubleshooting: no available candidates)

2.2 Using MulVAL

Now that we are ready to use the MulVAL tool, we need to know the inputs and parameters needed for the graph generation. Network vulnerabilities, subnets, access control lists, and other information related to the system setting are required as input to the logic engine which will produce the logical graph. For their expression, Datalog tuples are used and we will now review the terminologies and the processes that lead to it.

2.2.1 Setting up the network

As stated, MulVAL uses rules made in Datalog to imprint the network topology and the connections of its components. As shown in the figure, the logic engine receives three inputs. The first is the analysis database which provides the software vulnerabilities. The other is the interaction rules that cite the correlation relationships of the hosts and how an attacker can use them to achieve a wanted result and lastly the security policy that establishes the wanted system property for the administrator.

Then, the logic engine processes the data and builds the logical graph. The figure below depicts this architecture.

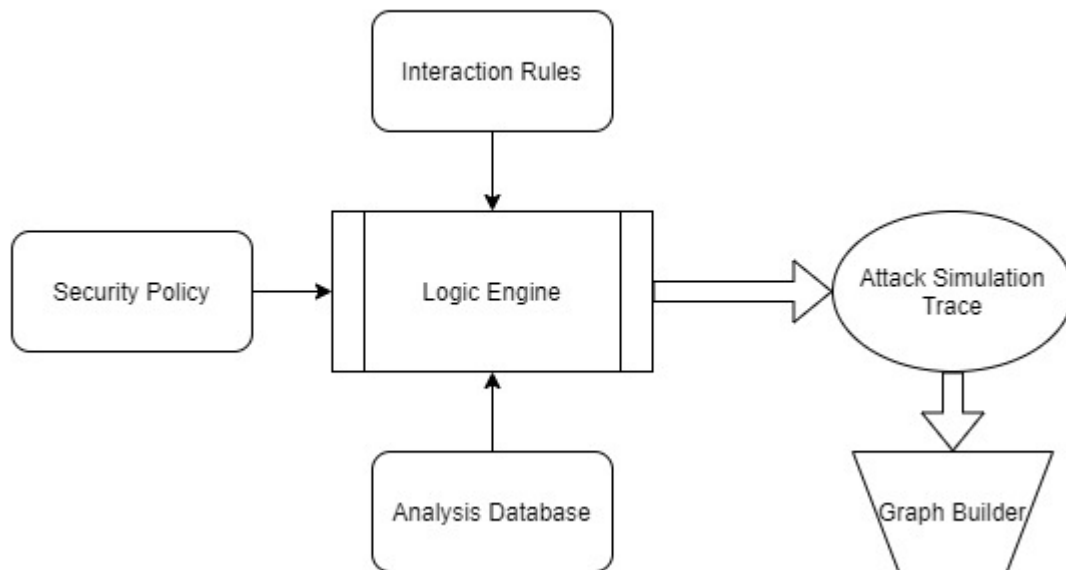


Figure 4- MulVAL architecture

So, what follows in order to create the graph, is the specification of the said inputs. For our example, we used the three-host premade files that are included in the VM (running_rules.P, input.P, run.P).

2.2.2 Datalog expressions

For our use case, the figure below represents the network that is taken into the examination. It is also included in the VM and it comes with the installation of MulVAL (folder named 3_hosts).

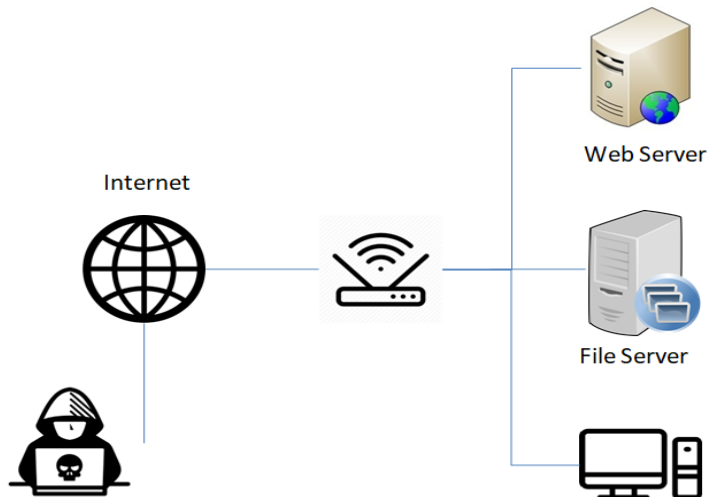


Figure 5 - Network Setup

To establish the wanted parameters, we use the file input.P that is presented in the figure and contains all information the logic engine needs in order to initialize the process.

```

running_rules.P × input.P × environment.P × metric.P × run.P ×
1 attackerLocated(internet).
2 attackGoal(execCode(workStation,_)).
3
4 hacl(internet, webServer, tcp, 80).
5 hacl(webServer, _, _, _).
6 hacl(fileServer, _, _, _).
7 hacl(workStation, _, _, _).
8 hacl(H,H,_,_).
9
10 /* configuration information of fileServer */
11 networkServiceInfo(fileServer, mountd, rpc, 100005, root).
12 nfsExportInfo(fileServer, '/export', _anyAccess, workStation).
13 nfsExportInfo(fileServer, '/export', _anyAccess, webServer).
14 vulExists(fileServer, vulID, mountd).
15 vulProperty(vulID, remoteExploit, privEscalation).
16 localFileProtection(fileServer, root, _, _).
17
18 /* configuration information of webServer */
19 vulExists(webServer, 'CAN-2002-0392', httpd).
20 vulProperty('CAN-2002-0392', remoteExploit, privEscalation).
21 networkServiceInfo(webServer, httpd, tcp, 80, apache).
22
23 /* configuration information of workstation */
24 nfsMounted(workStation, '/usr/local/share', fileServer, '/export', read).

```

Figure 6 - Datalog system setup

For a better understanding of the input file, a quick explanation is presented below.

Line 1-8: Information about the attacker and his/her goal. The goal is to execute code to the workstation and his only access is via webserver at TCP port 80

Line 11-16: Information about the file server

Line 19-24: Information about the webserver and workstation

To express an entity like the attacker in Datalog, we define the location and the goal of the said individual (or other information we may need) for the logic engine to conduct the potential courses of action it may take. In our use case, the attacker is

located on the internet (*attackerLocated(internet)*) and his goal is to execute a code in the workstation host (*execCode(Workstation,-)*).

Right below the goal of the attacker, *hacl* is the host access control list that specifies all accesses between the hosts and thus how the attacker can gain access to the wanted assets. Specifically, the syntax is *hacl(Source Destination, Protocol, Destination_Port)*, so what we get is that the attacker can only access the web server via TCP port 80 since he is located on the internet.

Now that we defined the system threat, we need to set the configurations of the hosts. As explained in the first chapter, the input comes from the vulnerability assessment process done either by dedicated software or by other network tools. Again in our example, what is crucial is the *vulExists(Host, ID, Program)* command that specifies the vulnerabilities with their ID in a program on the host, *vulProperty(ID, ExploitRange, Exploit Sequence)* that shows the consequences of the ID vulnerability and the *exportInfo(Server, Path, Access, Client)* [18].

Since we created the security policy and a network state for our hosts, the last component needed for the logic engine is the interaction rules.

```

70
71
72 /*****
73 /****      Interaction Rules      *****/
74 /*****
75
76 /***** Section execCode *****/
77 interaction_rule(
78   (execCode(H, Perm) :-
79     hasAccount(P, H, Perm)),
80   rule_desc('Insider threat', 1)).
81 */
82
83 interaction_rule(
84   (execCode(Host, Perm) :-
85     principalCompromised(Victim),
86     hasAccount(Victim, Host, Perm),
87     canAccessHost(Host)),
88   rule_desc('When a principal is compromised any machine he has an
89     account on will also be compromised',
90     0.5)).
91
92 interaction_rule(
93   (execCode(Host, root) :-
94     execCode(Host, _Perm2),
95     vulExists(Host, _, Software, localExploit, privEscalation)),

```

Figure 7 - Interaction Rules

In these interaction rules, amongst others we set the following conditions:

Line 77-80: *Anyone who has an account is considered an insider threat*

Line 84-88: *When a principal is compromised, and the user has an account, there is a 50% chance that the hosts with this account are also compromised.*

The figure displays the first two Datalog tuples that express two interaction rules concerning the binding information of users and principals that may be exploited. The syntax of the tuples is *hasAccount(Principal, Host, Account)* where principal expresses the property of the entity (employee, admin, attacker) and the account is the name of the user (chris, john, root).

This concludes the initialization part of the network and the interaction rules, and with all the inputs in place, the logic engine is now ready to generate the graph.

2.2.3 Generating the graph

Before building the graph, the MULVALROOT environment variable has to be set to the folder on which it will be built.

```
export MULVALROOT=/path/to/mulval  
export CXX=g++  
export CC=gcc  
cd /home/tools/mulval  
make
```

When the installation is complete, access the created folder and type the following commands to create the attack graph

```
cd /home/tools/mulval/utills  
chmod u+x graph_gen.sh  
cd /home/tools/mulval/testcases/3host  
graph_gen.sh input.P -v -p
```

```

Open  ~ / Desktop / tools / mulval / testcases / 3host  Save  Files  running_rules.P x  input.P x  environment.P x  metric.P x  run.P x
1 | :- [ '/home/john/Desktop/tools/mulval/lib/libmulval' ].
2 | :- [ '/home/john/Desktop/tools/mulval/src/analyzer/translate' ].
3 | :- [ '/home/john/Desktop/tools/mulval/src/analyzer/attack_trace' ].
4 | :- [ '/home/john/Desktop/tools/mulval/src/analyzer/auxiliary' ].
5
6 | :-dynamic meta/1.
7
8 | -load_dyn('running_rules.P').
9
10 | -load_dyn('input.P').
11
12 | :-assert(traceMode(completeTrace2)).
13
14 | :-load_dyn('/home/john/Desktop/tools/mulval/src/analyzer/advances_notrim.P').
15
16 | :-assert(cvss(_, none)).
17
18 | :-mulval_run.
19

```

Figure 8 - run.P file

As displayed in the figure above, run.P calls *running_rules* and *input* that were presented earlier, resulting in the creation of the graph (Figure below displays the generated graph).

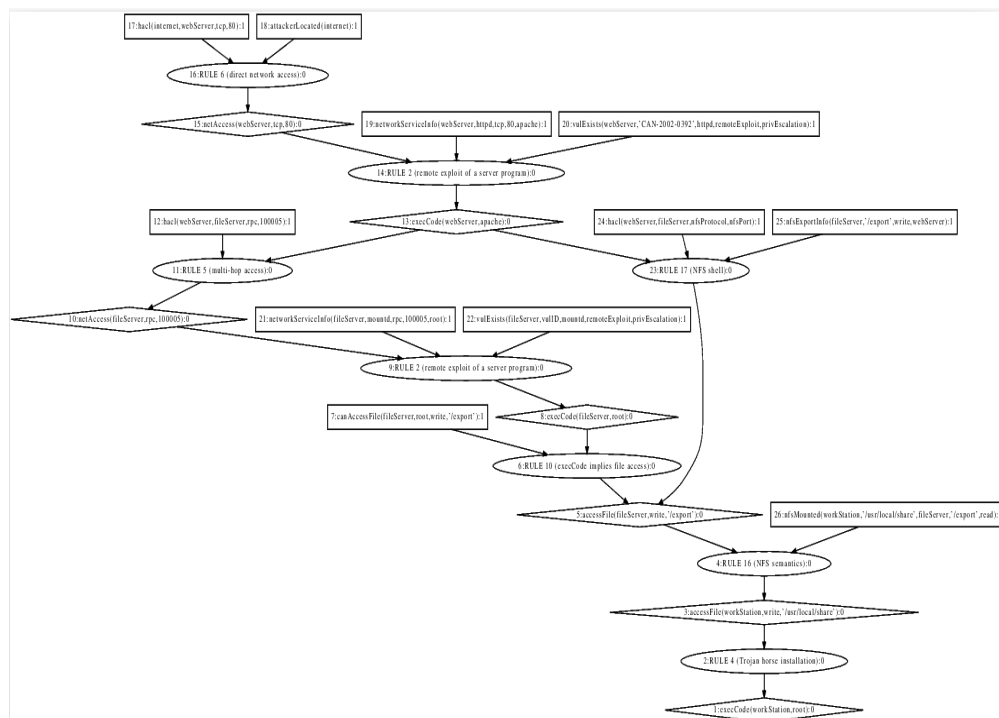


Figure 9 - Generated attack graph

This sums up the process of attack graph generation using MulVAL. Information about its analysis and the findings that we may extract from it will be presented in the next chapter.

Chapter 3. Defense Strategy

After generating the attack graph, we will now delve into its analysis and how we can make important defensive decisions and strategies by doing this. The first part of this chapter explains how we read the graph and draw data from it that can help in the next steps. We will then examine the vulnerabilities and the countermeasures that have to be taken in order to patch them and lastly we will evaluate them and choose the best line of defense depending on our budget and other internal and external parameters. Lastly, we will compare two potential defense strategies and their results after a number of system compromise attempts.

3.1 Graph Analysis

The logic attack graph we created depicts how an attacker can take advantage of the system's existing vulnerabilities and achieve his objectives. What differs though from other types of attack graphs is the type of nodes it has and what each one of them represents.

As shown, there are 3 types of nodes in our graph. The first schema presented is the rectangle that shows the current state of the system either it is an antivirus protecting a certain host or the existence of a threat. In our example, the first two nodes are displayed on the figure below:



Figure 10 - Square Nodes

We explained earlier how we represent the network status in Datalog. More specifically, *hacl(internet, webserver, tcp, 80)* represents a Host Access Control List with the parameters given (source, destination, protocol, destination port) and *attackerLocated (internet)* is pretty much self-descriptive as the “located” indicates an entity that has been reported and the parenthesis contains the location of the entity. The next type of node (figure 11) we meet is the circular one that describes the precondition and the postcondition of an attack, which is connected with the diamonds that portray a potential advantage that the attacker could gain.

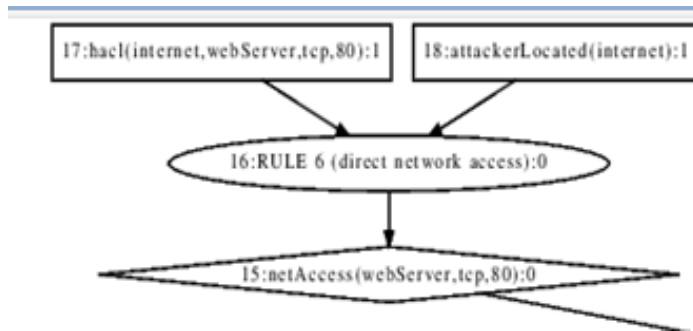


Figure 11 - Circular and Diamond Nodes

As described in the RULES section (rule 6 in this example), there is an interaction rule that gives an attacker access to the webserver through port 80. For better understanding, this rule is provided below.

```

interaction_rule(
  (netAccess(H, Protocol, Port) :-
    attackerLocated(Zone),
    hacl(Zone, H, Protocol, Port)),
  rule_desc('direct network access',
    1.0)).
  
```

Figure 12 - Direct network access Interaction Rule

Now, since the attacker has access to our network, we move to the next postcondition node. The result (postcondition) of the first step is the exploitation of a program of the webserver. As stated in the rectangle node (19), the webserver with the information provided in the parenthesis can be accessed by the attacker.

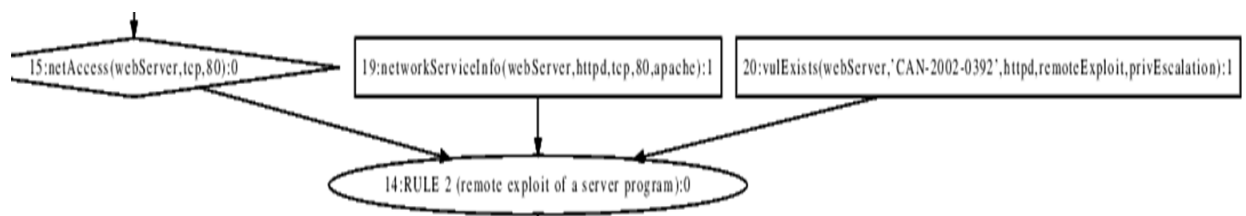


Figure 13 - First step of the attack (remote exploit of server program)

Combined with the vulnerability expressed in the rectangle node (20) CAN-2002-0392 can result in the remote exploitation of a server program (figure above displays the process). This interaction requires the existence of a software vulnerability in order for it to be successful as specified in the following interaction rule.

```

interaction_rule(
(execCode(H, Perm) :-
    vulExists(H, _, Software, remoteExploit, privEscalation),
    networkServiceInfo(H, Software, Protocol, Port, Perm),
    netAccess(H, Protocol, Port)),
rule_desc('remote exploit of a server program',
1.0)).

```

Figure 14 - Remote exploitation Interaction Rule

The same process of analysis is repeated till the attack graph is completely examined. So, the logic engine took the variables, compared them with the rules set by the administrator, and then generated the nodes where the requirements were met. With that in mind, the graph generation is completed with the attacker installing a Trojan horse and becoming the root. If we break down the graph, we can see that there are two courses of action that someone may take in order to have the wanted result. The simplification of the graph is presented in the figure below:

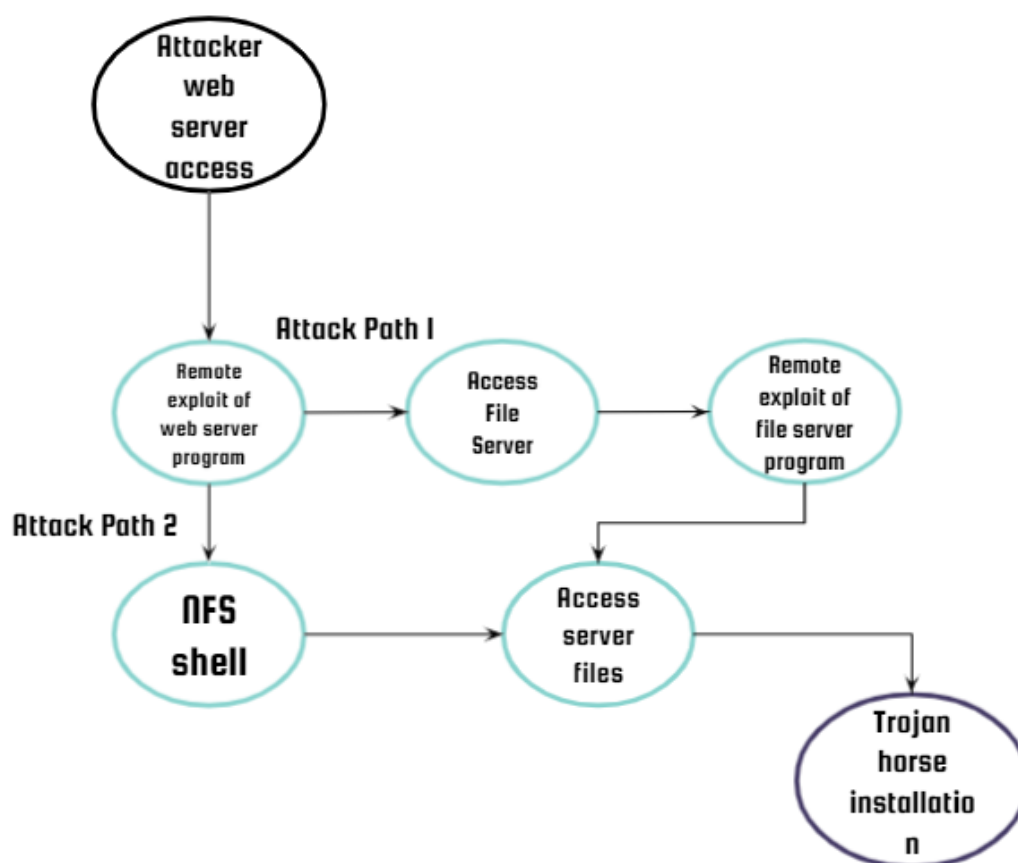


Figure 15 - Simplified graph

The first part of the graph indicated in green is the common course of actions that leads to two attack paths. Both lead to the access of the attacker to the file server files and the Trojan Horse installation and each node represents a system vulnerability that

can be partially protected or prevented by implementing certain countermeasures. This concludes the attack graph analysis and gives the administrator some insight in order to create the said defenses.

3.2 Risk Estimation

In the first chapter, we mentioned the risk management procedure and how we make use of attack graphs in order to identify them. The process of risk analysis, though, consists of two sub-processes [28]. Risk identification is the first part where all potential vulnerabilities are presented which is followed by risk estimation which assigns values to their likelihood and consequences. In this section, we present a formula to make such measurements and give values to the vulnerabilities (or nodes in attack graph representations), and provide an attack simulation environment using these values for demonstrating different scenarios.

3.2.1 Node Rating/ Risk Estimation Formula

As mentioned, attack graphs consist of nodes and edges that depending on the graph generation approach give a different interpretation to the system state at a given time. In their majority, these states are vulnerable points that are used by the attackers as “stepping stones” in order to achieve their goals and can be altered with the scope of improving the system security. After defining these vulnerabilities, what a network administrator has to do is prioritize the defense mechanisms depending on the needs of the network. In this part, we present a way to evaluate and rate the nodes and potential controls that will later constitute an effective network defense.

According to [24], the risk to which an information system is exposed is a function of the values of its assets (**A**), the nature and level of its vulnerabilities (**Rs**), the nature and possibility of threat occurrence (**Ro**), and the nature and level of impact of the consequences that may appear if these threats occur (**L**). This gives us

$$R = f(A, Rs, Ro, L)$$

We can express this in terms of cybersecurity with [5] and [10] that both use a formula to express the total loss of a potential compromise: $L = Rs \times Ro \times S \times A$ where **Rs** is the risk of success of an attack to this vulnerability, **Ro** is the risk of a threat occurrence, **A** is the total of the assets involved and **S** is the total level of

security that protects it. Although, since there may be no protection in a given node or vulnerability, S is calculated by $S = 1 - e$, where E expresses the efficiency of the implemented controls which finally gives us the formula of:

$$L = R_s \times R_o \times (1 - e) \times A$$

In the following chapters, we will provide information on how each one of these values is calculated and how they differ depending on the defense approach an administrator might take.

3.2.2 Measuring the values

This section provides the process of the evaluation of each parameter of the formula, and how we obtain these values.

Risk of success (Rs):

The risk of success is the probability that an attacker may succeed and affect the system state in any way. The state nodes denote these probabilities as well as the requirements that need to be met for them to succeed. [9] in particular, gives new approaches in calculating the success probabilities in MulVAL by taking into consideration the Common Vulnerability Scoring System (CVSS) to improve the accuracy of probability of each vulnerability. So attack graphs not only give us the most accurate information concerning the risk of success by calculating in the graph generation engine, but also show us the requirements and the system state at each time.

Risk of Occurrence (Ro):

One thing the defender needs to take into consideration is the probability of the attacker exploiting a vulnerability. Since each vulnerability works differently in favor of the attacker depending on his goal, the system administrator has to evaluate them according to a selected defense strategy plan. The risk of occurrence differs according to the potential gain of the attacker, the vulnerability type, the complexity of the attack, the existing defenses, and the impact of a successful exploitation but as explained in the next section, there are tools and standards that consider these factors and help the system owners calculate the final probability.

Security (S=I-E):

As security in the formula, we define the existing controls of a certain vulnerability. Security includes everything that contributes to the mitigation of the total damage done during or after an attack. Since not all controls have the same impact though, we measure S by its efficacy (1-E) where $0 \leq E < 1$ (because of the zero-day attacks). Efficacy and security values will be set arbitrarily in our examples.

Asset:

As a definition, an asset is “an item of property owned by a person or company, regarded as having value and available to meet debts, commitments, or legacies”. The formula considers A to be the overall value of the assets that the attacker has access to after the compromise of a vulnerability. Attack graphs help us calculate the interdependencies between the vulnerabilities and the total of the affected assets via attack paths that demonstrate the possible course of actions an attacker may take during an incident. Asset values are given arbitrarily in the examples.

3.2.3 Attack simulations

For a better understanding of how the implementation of certain controls alters the results of the successful attack attempts to the system into consideration, we created scripts in Python that represent the scenario where the attacker tries to exploit the said vulnerabilities. The IDE used for the demonstration was PyCharm and it ran on AMD Ryzen 5 1400 Quad-Core Processor at 3.2GHz with 8GB RAM.

The outputs show the results of the 2 attack paths, generated from our graph, initially on *a totally unguarded system* after ~3000 attacks were made in each one separately and how many attempts amongst them were successful.

The first indications show the results of attack path 1. Out of 3000 attacks, 1897 succeeded in exploiting the first vulnerability (remote exploit of web server program), 960 out of these took advantage of the second (access server file), 359 got past the remote exploit of the file server program, 127 accessed the file server and finally, 66 out of 3000 attacks were successfully completed the objective (2.2% of the total attacks). The same goes for the second path that had a total of 560 successful attempts out of 3000 attacks (18,66%) giving us an outcome of 626/6000 (10,43%) system violations.

Also, we created scripts that show the preference of the attackers' Courses of Action in multiple paths, knapsack problem solver, and how the chosen controls affect the success ratio after their implementation.

3.3 Choosing the most competent defenses

Perceiving the threats and managing their controls is crucial but under the restriction of the organization's budget, the administrator has to distribute his limited resources to get the maximum value out of them. Risk evaluation is the process of comparing the results of risk analysis with risk criteria to determine whether the risk is acceptable depending on its magnitude[28]. After examining the graph and assigning values to the risks, we can make a comparison between each system's weak spots and decide whether it would be beneficial for the organization to invest in corrective actions or not. Although, depending on the defense model, calls made by the system owners may differ so we will examine how administrators react based on these models presenting two potential strategies.

3.3.1 Cybersecurity Controls

Cybersecurity control describes a mechanism, action, or procedure whose goal is to reduce the risk and minimize the consequences created by a specific system vulnerability. Inside an Information Security Management System (ISMS), countermeasure categories usually include the physical security of an asset, procedural and technical changes, and personnel improvement, education, replacement, or removal [24]. The stages of vulnerability assessment and analysis (which includes the deduction that was created by attack graphs) demonstrate possible system susceptibilities that require the application of such controls. [15] presents a process of how the control candidates are chosen after the generation of an attack graph, and then it's recreated with the updated values of risks on its nodes.

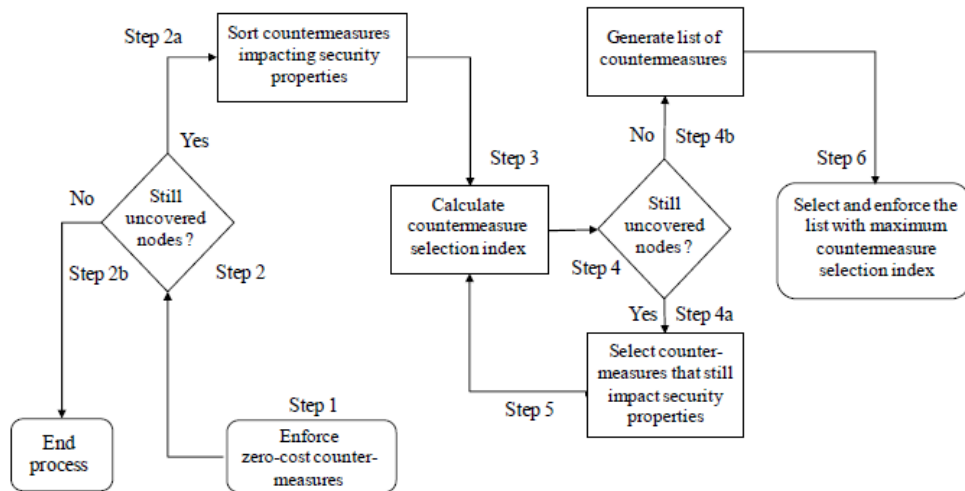


Figure 16 - Return on investment table [15]

As shown in the figure, the zero-cost countermeasures have to be enforced first, and then a complete list with the controls that impact security properties is generated in order for the organization to pick amongst them.

A list of countermeasures that may apply to certain vulnerabilities is provided within [34] for non-software controls and [35] for software, introducing how an organization may react in most cases.

3.3.2 0-1 Knapsack Problem

Optimal budget allocation is a problem explored vastly in [16] where it is posed as a problem whose solution is reminiscent of the classical knapsack problem. The knapsack problem is a problem in combinatorial optimization where, given a set of items that all have a weight and a value, we have to figure out the combination of items that will give us the maximum value within a total weight limit. In terms of network defense though, the algorithm differs depending on the resource interaction, and if the resources are spent evenly or not amongst the targets. In our examples, we use the multiple targets-shared resources pattern.

The variables of the knapsack problem in our model consist of two variables:

Value (Weight): The value of a vulnerability (node) is described by the node rating formula mentioned in the previous section. The potential loss in case of a system compromise because of the said vulnerability is proportional to its value meaning that it needs to be secured. So we consider $L = Rs \times Ro \times S \times A$ as the value of a vulnerability.

Cost: After controls and vulnerabilities are rated, they have to be purchased or put into action. Either way (since they are not zero-cost controls), there will be an economic impact on the organization during their enforcement. In our case study, we call this control cost. Control cost can be categorized into two types:

- *Direct Cost* is a one-time investment made by the network owners that is required for the control to be acquired or done.
- *Indirect Cost* is potential financial damage that may be created due to the implementation of certain controls. Some types of indirect costs are reduced system performance and re-training costs due to system changes [4]. Also, node removal and indirect damage that may be created due to interdependencies are discussed in [12].

Finally, one control may fall into both categories since it may require both a direct cost for its purchase and then an indirect cost for its implementation which can lead to service downtime or system maintenance. In this case, the total cost of the control can be expressed by $C(total)=C(d)+C(i)$. An example that has both direct and indirect costs is the installation of software that first needs to be bought (direct) and slows the system performance while active (indirect).

In the following sections, we examine how the Knapsack Problem is implemented depending on various defense scenarios an organization may take.

3.3.3 Defense approaches

We earlier discussed the formula that can be used for the evaluation of vulnerabilities in a system. To set up the parameters for the knapsack problem in order to distribute the organization's budget accordingly, we will use it in our generated graph. So let L be the potential loss from a potential node compromise. We then have $L = Rs \times Ro \times (1 - e) \times A$ where A is considered 1 (since all nodes affect the same asset) and the nodes have no security meaning $E=0$ until the implementation of a countermeasure. In the figure below we provide some potential control candidates in the nodes, they can be used on.

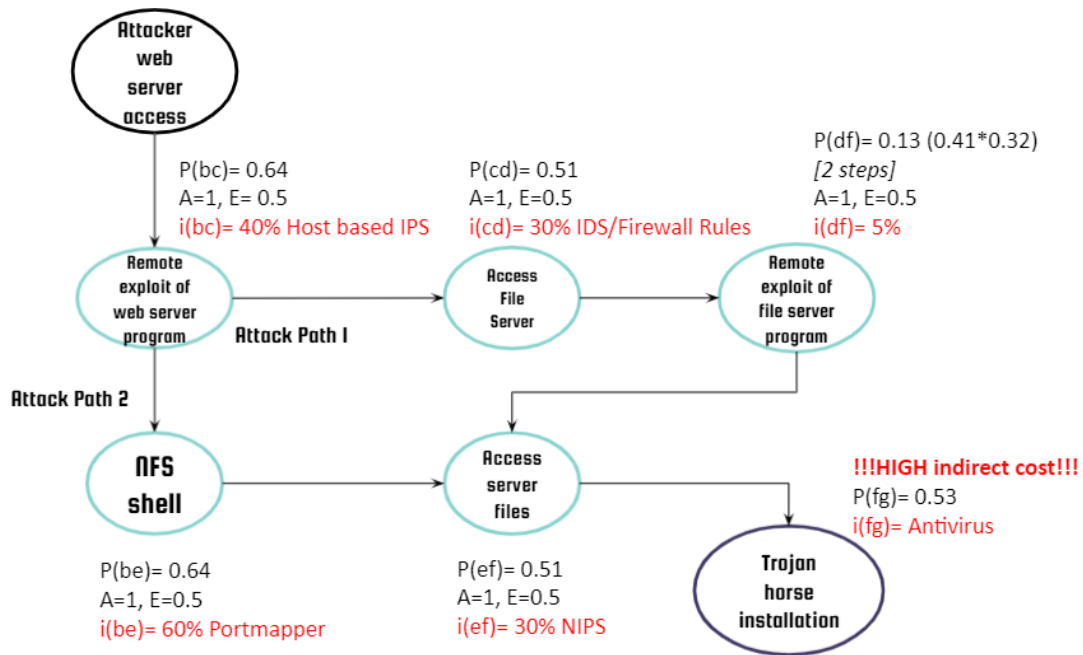


Figure 17 - Simplified Graph with Control Options

Each node has the values of P that give the success probability in the node compromise, the efficacy factor E that we set the arbitrary value of 0.5 for each control meaning that if the countermeasure is selected, security (S) will change to 0.5, and then i expresses the total investment that the control needs in compared with the total budget provided. We also consider that zero-cost controls were implemented and this is the latest-generated attack graph. As it is impossible to secure all nodes due to this restriction, the system administrator is called to make the call and prioritize them. We will use two examples of defense approaches one may take where the first considers that the attacker has no strategy (*random attacker*) and the second calculates the possible course of action the attacker may take based on a game-theoretic model (*game theory attacker*).

3.3.4 Random Based Defense Approach

Our first approach considers that the attacker has no strategy on how he or she will proceed and thus will attack randomly each path. So let N be the total courses of action an attacker may take, each one has a $1/N$ probability of getting attacked. Since there are two possible attack paths in our example, this gives us the $R(o)$ probability at $\sim 50\%$ (~ 3000 out of 6000 attacks will occur on each attack path).

With this in mind, by using as input the values and the cost of the nodes, the nodes that give the maximum value within the budget are *B*, *C*, *D*, and *F*. The newly generated graph after the implementation of the chosen countermeasures (nodes are multiplied with the new value of the security which is $1-0.5=0.5$) with the results of 6000 attacks is provided in the figure.

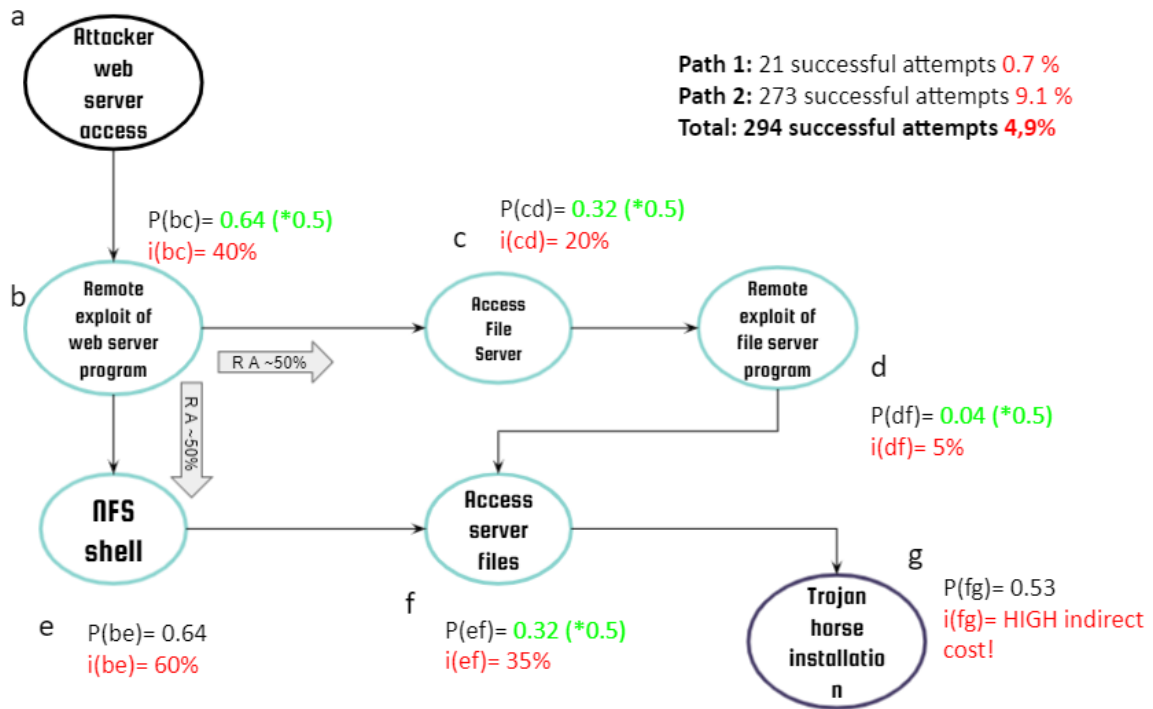


Figure 18 - Random attacker results after 6000 attacks vs Random Attacker Controls

The previous successful attempts on an unguarded system after 6000 attacks were 626 (10,43%) and with this defense approach, only 294 (4,9%) system compromises were reported. Although this is not a realistic scenario since many factors dictate the attacker’s choices in these situations as explained in the previous chapters and the next defense approach examines these aspects.

3.3.5 Game Theory Based Defense Approach

Many pieces of research were made that relate game theory with cybersecurity decision-making ([4], [5], [19], and [23] are some of them) and in general they present ways to predict the attacker’s behavior based on each model. By doing these predictions we can assert the probability of occurrence ($P(o)$) of an attack to a specific vulnerability. Specifically, [19] determines the probabilities of occurrence by calculating the attacker’s payoff considering the vulnerability’s CVSS score.

Common Vulnerability Security Score (CVSS) [3][19] is a standard that depicts *how a vulnerability can be exploited* (attack vector, attack complexity, privileges required, user interaction) as well as *how its exploitation impacts the organization* (confidentiality impact, integrity impact, availability impact).

Although, this may not always be representative since for some organizations, vulnerabilities that impact confidentiality may attract more attacks regardless of their difficulty level. This is why CVSS is composed of three metric groups.

Base: The base CVSS metric is the one mentioned above and it represents “the intrinsic characteristics of a vulnerability that are constant over time and across user environments”. It consists of exploitability metrics and impact metrics.

Temporal: The Temporal metric group “reflects the characteristics of a vulnerability that may change over time but not across user environments”. For example, the presence of an exploit tool accessible by anyone would increase the CVSS score, while the creation of an official patch would decrease it.

Environmental: representing “characteristics of a vulnerability that are relevant and unique to a particular user's environment”. This is where existing security controls that may mitigate part of the damage of a successful attack and requirements for confidentiality, integrity, and availability imposed by the system owners take place and change the evaluation of the vulnerability.

Back in our example, if we set the CVSS scores 1 and 10 for the two paths, the attacker will attack accordingly 9% of the time the first and 91% times the second node. Then, with the chosen defenses we get the following results:

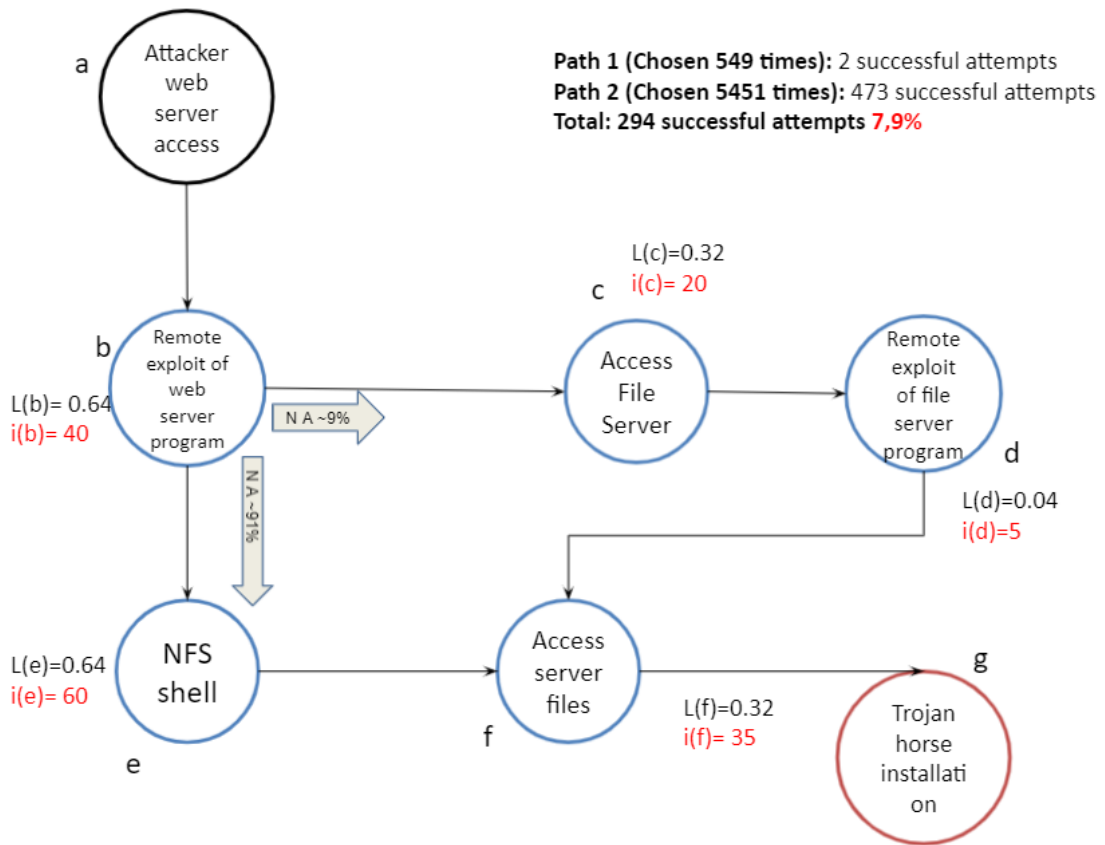


Figure 19 - Game Theory attacker vs Random attacker Controls

This time, the attacker has chosen to attack the first path 549 times and the second 5451 times. Also, even with the controls, we observe that a total of 475 (7.9%) attacks were successful which is slightly lower than the totally uncontrolled system results. By knowing the probabilities of attack occurrence in each node, we recalculate the node rating with the new values:

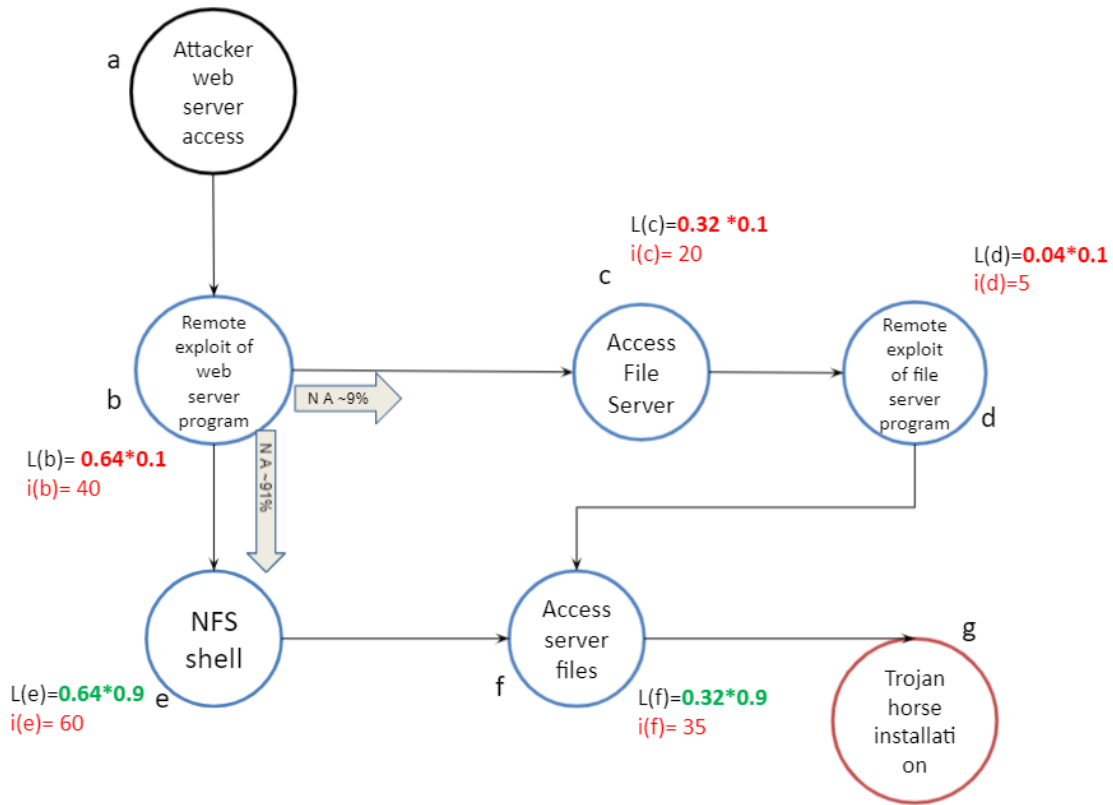


Figure 20 - New possibilities of attack occurrence (Po) chosen by Game theory attacker

With knapsack values changed, the nodes that are selected for the maximum efficiency within budget are now *e*, *f*, and *d*. The results of the newly generated defense, that uses controls on different nodes are presented and compared in the next section.

3.3.6 Results

In this section, we compare the results of both defense approaches (random attacker based and game theory attacker based) after 6000 attacks. The results involve the following scenarios:

- random and game theory attacker attack an unguarded system
- random and game theory attacker attack a system with a random attacker based defense
- and random and game theory attacker attack a system with a game theory attacker based defense

Random & Game theory VS Unguarded system

Starting with the total success of the attack and its success percentage as shown in section 3.2, what follows is the number of attacks that occurred on each path. The figures below depict these results.

Attacker Type	Total Success	Percentage	Path 1 Selected	Path 2 Selected
Random Attacker	594	9.9 %	3043	2957
Game Theory Attacker	938	15.633333333333333 %	554	5446

Attacker Type	Path	Step	Success Count	
Random Attacker	Attack Path 1	result of the attack path 1	65 successful attempts	
		step 1 success	1919	
		step 2 success	969	
		step 3 success	379	
	Attack Path 2	result of the attack path 2	529 successful attempts	
		step 1 success	1933	
		step 2 success	979	
		step 3 success	122	
	Game Theory Attacker	Attack Path 1	result of the attack path 1	11 successful attempts
			step 1 success	353
			step 2 success	191
			step 3 success	82
Attack Path 2		result of the attack path 2	927 successful attempts	
		step 1 success	3478	
		step 2 success	1765	
		step 3 success	29	

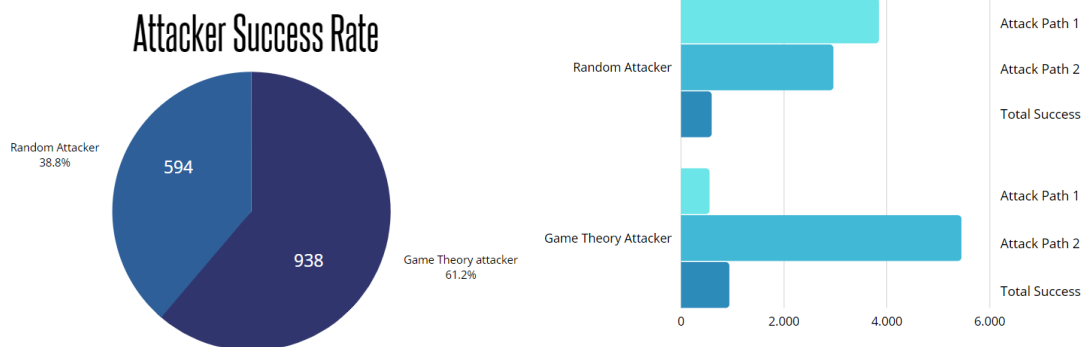


Figure 21 - Random and Game Theory attacker vs Unguarded System

Figure 21a - Results of Random attacker vs Unguarded System

Figure 21b - Results of Game Theory attacker vs Unguarded System

Figure 21c, d - Graphical Representation of the results

More specifically, the random attacker chose the first path 3043 times of which 65 were successful and the second 2957 times with 529 successful. On the other hand, the game theory attacker only used the first path 554 times and succeeded in 11

compromises, and the second attack was attacked 5446 times and the successful attempts were 927.

With the same methodology, we present the results of random-based and game theory-based defense below.

Random & Game theory VS Random Attacker Controls

This scenario (Figure 22) involves 6000 attacks to a system that implements controls by calculating $P(o)=0.5$ (random attacker probability of occurrence). As demonstrated earlier, the nodes selected are b, c, d, and f.

```

Total success: 272 Percentage: 4.533333333333333 %
Path 1 selected: 2955 times
Path 2 selected: 3045 times
Analysis:
***Attack Path 1***
result of the attack path 1: 10 successsful attempts
step 1 success: 993
step 2 success: 252
step 3 success: 53
step 4 success: 17
***Attack Path 2***
result of the attack path 2: 262 successsful attempts
step 1 success: 1932
step 2 success: 486

Total success: 494 Percentage: 8.233333333333333 %
Path 1 selected: 535 times
Path 2 selected: 5465 times
Analysis:
***Attack Path 1***
result of the attack path 1: 1 successsful attempts
step 1 success: 163
step 2 success: 39
step 3 success: 10
step 4 success: 2
***Attack Path 2***
result of the attack path 2: 493 successsful attempts
step 1 success: 3517
step 2 success: 884
    
```

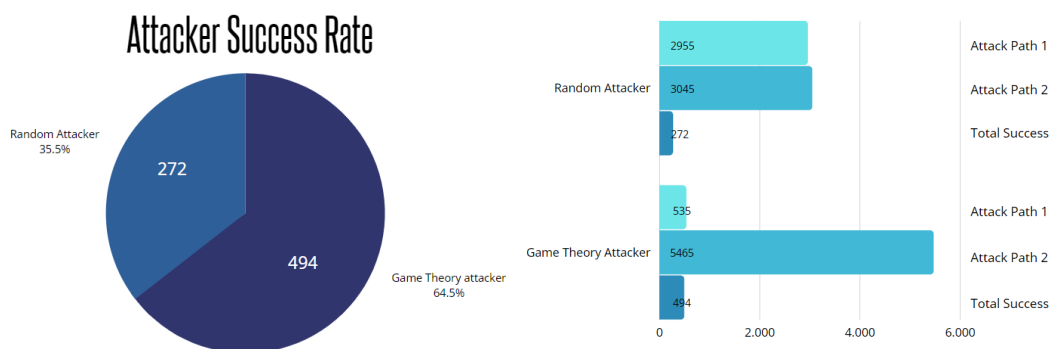


Figure 22 - Random attacker and Game theory attacker vs Random based defense

Figure 22a - Results of Random attacker vs Random based defense

Figure 22b - Results of Game Theory attacker vs Random based defense

Figure 22c, d - Graphical Representation of the results

Random & Game theory VS Game Theory Controls

Figure 23 demonstrates 6000 attacks to a system that implements controls by calculating the expected utility of the attacker with game theory ($P(o)=0.9$ on the second path and $P(o)=0.1$ on the first). The vulnerability nodes that we chose to defend are d, e, and f.

```

Total success: 171 Percentage: 2.85 %
Path 1 selected: 3056 times
Path 2 selected: 2944 times
Analysis:
***Attack Path 1***
result of the attack path 1: 35 successfull attempts
step 1 success: 1926
step 2 success: 1002
step 3 success: 210
step 4 success: 75
***Attack Path 2***
result of the attack path 2: 136 successfull attempts
step 1 success: 936
step 2 success: 255

Total success: 221 Percentage: 3.683333333333333 %
Path 1 selected: 582 times
Path 2 selected: 5418 times
Analysis:
***Attack Path 1***
result of the attack path 1: 8 successfull attempts
step 1 success: 360
step 2 success: 202
step 3 success: 40
step 4 success: 13
***Attack Path 2***
result of the attack path 2: 213 successfull attempts
step 1 success: 1742
step 2 success: 419
    
```

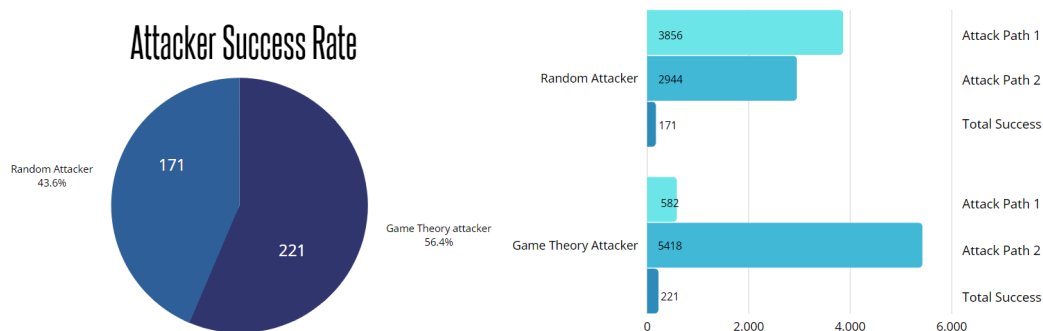


Figure 23 - Random and Game theory vs Game theory-based defense

Figure 23a - Results of Random attacker vs Game theory-based defense

Figure 23b - Results of Game Theory attacker vs Game theory-based defense

Figure 23c, d - Graphical Representation of the results

3.3.7 Comparison

The first thing we notice in these scenarios is that the Game Theory attacker always has more success than the random one. Even in the scenario with the defense created based on the random attacker, the game theory attacker achieved more system compromises than the other. Also, even when both attacker and defender know the

game in an adversary situation, ~90% of the times game theory attacker targets the weakest system path meaning that since the success percentage is always higher than the random attacker’s approach, the control implementation should prioritize its defense. The table below summarizes the results from all scenarios.

Table 3 - Attack Simulation Results

	Random Attacker	Game Theory Attacker
No defenses	Path 1 attempts: 3043 Path 2 attempts: 2957 Success: 9.9%	Path 1 attempts: 554 Path 2 attempts: 5446 Success: 15.63%
Random attacker based controls	Path 1 attempts: 2955 Path 2 attempts: 3045 Success: 4.5%	Path 1 attempts: 535 Path 2 attempts: 5465 Success: 8.23%
Game Theory attacker based Controls	Path 1 attempts: 3056 Path 2 attempts: 2944 Success: 2.85%	Path 1 attempts: 582 Path 2 attempts: 5418 Success: 3.63%

3.3.8 Other methods

The application of heuristic algorithm approaches for network hardening is also mentioned in [12] and [20]. More specifically, authors in [12] calculate the return on investment of each node and their interdependencies and then use the *Best-First search (BestFS)* algorithm to select the nodes whose closure is the most profitable. Respectively, [20] develops an *Ant Colony Optimization (ACO)* algorithm where ants populate the most significant system vulnerabilities and then the edges are ranked depending on the pheromone left by the ants. What differs in comparison with the game theory approach is that while these methods create fast and good solutions on complex systems but may give different results with each repetition [21], the latter provides a systematic quantitative approach for deciding the best strategy in most competitive scenarios. Table 4 summarizes some advantages and disadvantages posed by each method.

Table 4 - Defense methods Advantages and Disadvantages

	Advantages	Disadvantages
Game Theory	<ul style="list-style-type: none"> • Most researched method in combination with attack graphs • Gives mostly accurate predictions of an attacker's behavior 	<ul style="list-style-type: none"> • The increasing number and complexity of attacks sometimes lowers the accuracy • Simply provides rules of logic (not winning strategy) based on certain behaviors.
Ant Colony Optimization/ Hidden Markov Model/ BestFS	<ul style="list-style-type: none"> • Fast discovery of good solutions • Scale well on large networks 	<ul style="list-style-type: none"> • Increased Difficulty in their theoretical analysis • Probability Distribution changes by iteration

Chapter 4. Conclusions

In this document we presented, analyzed, and compared the advantages and disadvantages of the main graph generation techniques and provided a guideline that helps the system owners to evaluate their outputs. We also examined one graph generation tool (MulVAL) and created a cybersecurity plan based on its generated attack paths. Moreover, the simulation of the attacks helped in the demonstration of how different defense approaches affect the results in most cases and that the same amount of resources can provide a dissimilar outcome.

Risk management is a procedure that requires a different approach depending on the system owner or the assets that need to be protected. Attack graphs are a great way to depict potential system vulnerabilities that previous assessment steps didn't and combining them with defense evaluation approaches like Game-theory or Ant Colony Optimization simplifies this process and improves the efficiency of the existing resources to a better end.

4.1 Analysis

Attack graphs are a great way to have a visual representation of how a network is threatened by an attacker and provides the ability to analyze vulnerabilities individually, as the step by step depiction of the system state during an incident and the correlation of existing threats may prove crucial in decision-making before, during and after its occurrence.

Also, especially MulVAL is an excellent tool that calculates logic parameters like the human factor and insider threat, something that other evaluation methods don't. Of course, this can prove to be bad since it can lead to over-complicated attack graphs if too many conditions are set.

Since we used the tools on a small example, the speed was not an issue but work and research [9][18][22] done in larger networks indicate that their use can be applicable to a wide variety of systems.

Lastly, implementation (mainly with the free tools we used) is something that has to be improved since the program installation and the inputs given by the user are pretty complex. This can prove dangerous as wrong inputs give false results especially in approaches like TVA where the preconditions and postconditions are set by the administrator. With that in mind, network administrators have to constantly check and update the needed components in order for the attack graph to be effective.

References

- [1] Liu, C., Singhal, A., & Wijesekera, D. (2012, December). Mapping evidence graphs to attack graphs. In 2012 IEEE International Workshop on Information Forensics and Security (WIFS) (pp. 121-126). IEEE.
- [2] Barik, M. S., Sengupta, A., & Mazumdar, C. (2016). Attack Graph Generation and Analysis Techniques. *Defense Science Journal*, 66(6).
- [3] Singhal, A., & Ou, X. (2011). Security Risk Analysis of Enterprise Networks Using Probabilistic Attack Graphs. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Interagency or Internal Report (IR) 7788.
- [4] Panaousis, E., Fielder, A., Malacaria, P., Hankin, C., & Smeraldi, F. (2014, November). Cybersecurity games and investments: A decision support approach. In *International Conference on Decision and Game Theory for Security* (pp. 266-286). Springer, Cham.
- [5] Panda, S., Panaousis, E., Loukas, G., & Laoudias, C. (2020). Optimizing investments in cyber hygiene for protecting healthcare users. In *From Lambda Calculus to Cybersecurity Through Program Analysis* (pp. 268-291). Springer, Cham.
- [6] Haque, S., Keffeler, M., & Atkison, T. (2017). An evolutionary approach of attack graphs and attack trees: A survey of attack modeling. In *Proceedings of the International Conference on Security and Management (SAM)* (pp. 224-229). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).
- [7] Ou, X., Boyer, W. F., & McQueen, M. A. (2006, October). A scalable approach to attack graph generation. In *Proceedings of the 13th ACM conference on Computer and communications security* (pp. 336-345).
- [8] Sheyner, O., Haines, J., Jha, S., Lippmann, R., & Wing, J. M. (2002, May). Automated generation and analysis of attack graphs. In *Proceedings 2002 IEEE Symposium on Security and Privacy* (pp. 273-284). IEEE.
- [9] Sembiring, J., Ramadhan, M., Gondokaryono, Y. S., & Arman, A. A. (2015). Network security risk analysis using improved MulVAL Bayesian attack graphs. *International Journal on Electrical Engineering and Informatics*, 7(4), 735.
- [10] Khouzani, M. H. R., Liu, Z., & Malacaria, P. (2019). Scalable min-max multi-objective cyber-security optimisation over probabilistic attack graphs. *European Journal of Operational Research*, 278(3), 894-903.

- [11] Noel, S., Jajodia, S., Wang, L., & Singhal, A. (2010). Measuring security risk of networks using attack graphs. *International Journal of Next-Generation Computing*, 1(1), 135-147.
- [12] Sawilla, R., & Skillicorn, D. (2012, November). Partial cuts in attack graphs for cost-effective network defense. In *2012 IEEE Conference on Technologies for Homeland Security (HST)* (pp. 291-297). IEEE.
- [13] Wang, L., Noel, S., & Jajodia, S. (2006). Minimum-cost network hardening using attack graphs. *Computer Communications*, 29(18), 3812-3824.
- [14] Noel, S., Jajodia, S., O'Berry, B., & Jacobs, M. (2003, December). Efficient minimum-cost network hardening via exploit dependency graphs. In *19th Annual Computer Security Applications Conference, 2003. Proceedings.* (pp. 86-95). IEEE.
- [15] Gonzalez-Granadillo, G., Doynikova, E., Kotenko, I., & Garcia-Alfaro, J. (2017, October). Attack graph-based countermeasure selection using a stateful return on investment metric. In *International Symposium on Foundations and Practice of Security* (pp. 293-302). Springer, Cham.
- [16] Smeraldi, F., & Malacaria, P. (2014, May). How to spend it: optimal investment for cyber security. In *Proceedings of the 1st International Workshop on Agents and CyberSecurity* (pp. 1-4).
- [17] Ingols, K., Lippmann, R., & Piwowarski, K. (2006, December). Practical attack graph generation for network defense. In *2006 22nd Annual Computer Security Applications Conference (ACSAC'06)* (pp. 121-130). IEEE.
- [18] Ou, X., Govindavajhala, S., & Appel, A. W. (2005, August). MulVAL: A Logic-based Network Security Analyzer. In *USENIX security symposium* (Vol. 8, pp. 113-128).
- [19] Mishra, P. K., & Tyagi, G. Game Theory based Attack Graph Analysis for Cyber War Strategy.
- [20] Wang, S., Zhang, Z., & Kadobayashi, Y. (2013). Exploring attack graph for cost-benefit security hardening: A probabilistic approach. *Computers & Security*, 32, 158-169.
- [21] Selvi, V., & Umarani, R. (2010). Comparative analysis of ant colony and particle swarm optimization techniques. *International Journal of Computer Applications*, 5(4), 1-6.
- [22] Yi, S., Peng, Y., Xiong, Q., Wang, T., Dai, Z., Gao, H., ... & Xu, L. (2013, October). Overview on attack graph generation and visualization technology. In *2013*

International Conference on Anti-Counterfeiting, Security, and Identification (ASID) (pp. 1-6). IEEE.

[23] Hota, A. R., Clements, A. A., Bagchi, S., & Sundaram, S. (2018). A game-theoretic framework for securing interdependent assets in networks. In Game Theory for Security and Risk Management (pp. 157-184). Birkhäuser, Cham.

[24] Information Systems Risk Analysis and Management, Kostas Labrinoudakis

[25] Cisco Certificate Networking Associate security course

[26] <http://people.cs.ksu.edu/~xou/mulval/> accessed on 8/5/2021

[27] Ammann, P.; Wijesekera, D. & Kaushik, S. Scalable, graph-based network vulnerability analysis. In Proceedings of the 9th ACM Conference on Computer and Communications Security, 2002. pp. 217-224.

[28] ISO/IEC 27005: 2018 risk management standard

[29] ISO Guide 73:2009

[30] Installation and configuration of Mulval attack graph tool environment under Kali <https://www.programmingsought.com/article/37794643490/> accessed on 8/5/2021

[31] MulVAL: A logic-based, data-driven enterprise security analyzer

<http://people.cs.ksu.edu/~xou/argus/software/mulval/readme.html>

accessed on 8/5/2021

[32] SMV - Symbolic Model Checking

https://link.springer.com/chapter/10.1007/978-3-662-04558-9_12

accessed on 8/5/2021

[33] Training support package on the ad-hock course on specific topics of Cyber Security of Critical Infrastructures, VOLODYMYR DAHL EAST UKRAINIAN NATIONAL UNIVERSITY

[34] CIS Controls Version 7.1 Implementation Groups 1.2

[35] DHSES_OCT_CIRT_CIS_Controls_Tool_Mapping_v1.1.1

[36] XSB Downloads <http://xsb.sourceforge.net/downloads/downloads.html> accessed on 8/5/2021

[37] Common Vulnerability Scoring System version 3.1: Specification Document

<https://www.first.org/cvss/specification-document> accessed on 8/5/2021

[38] Bimatrix Game Solver https://cgi.csc.liv.ac.uk/~rahul/bimatrix_solver/ accessed on 8/5/2021

[39] CVE Details <https://www.cvedetails.com/browse-by-date.php> accessed on 8/5/2021

[40] Scenario and Attack Graphs <http://www.cs.cmu.edu/~scenariograph/#software> accessed on 8/5/2021

Appendix A

Troubleshooting

javac: command not found

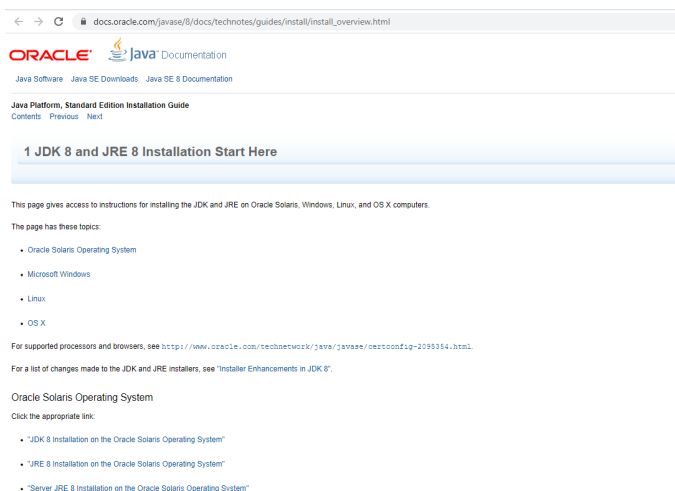
The first error we troubleshoot is javac: command not found. This happens because what we have installed is the Java Runtime Environment (JRE) only, which does not provide javac. For javac, we have to install the OpenJDK Development Environment

```
root@john:~/Desktop/mulval# export MULVALROOT=/root/Desktop/mulval/
root@john:~/Desktop/mulval# make
(cd src/adapter; make; make install)
make[1]: Entering directory '/root/Desktop/mulval/src/adapter'
javac GetCVEID.java -cp :/root/Desktop/mulval/lib/dom4j-1.6.1.jar:/root/Desktop
make[1]: javac: Command not found
make[1]: *** [Makefile:9: GetCVEID.class] Error 127
make[1]: Leaving directory '/root/Desktop/mulval/src/adapter'
make[1]: Entering directory '/root/Desktop/mulval/src/adapter'
javac GetCVEID.java -cp :/root/Desktop/mulval/lib/dom4j-1.6.1.jar:/root/Desktop
make[1]: javac: Command not found
make[1]: *** [Makefile:9: GetCVEID.class] Error 127
make[1]: Leaving directory '/root/Desktop/mulval/src/adapter'
make: *** [Makefile:6: adapter] Error 2
```

Part 1- Install JDK

<https://stackoverflow.com/questions/48609449/mulval-installation-problems>

The JRE is the Java Runtime Environment which is a package with everything necessary to run a compiled Java program while the JDK is the Java Development Kit, a full-featured SDK for Java. To install it, we firstly download it



JDK Download

Product / File Description	File Size	Download
Linux ARM32 Hard Float AIB	75.4 MB	jdk-8u261-linux-arm32-vfp-hflt.tar.gz
Linux ARM64 Hard Float AIB	70.3 MB	jdk-8u261-linux-arm64-vfp-hflt.tar.gz
Linux i686 RPM Package	12102 MB	jdk-8u261-linux-i586.rpm
Linux i686 Compressed Archive	106.81 MB	jdk-8u261-linux-i586.tar.gz
Linux x64 RPM Package	12113 MB	jdk-8u261-linux-x64.rpm
Linux x64 Compressed Archive	106.48 MB	jdk-8u261-linux-x64.tar.gz
macOS x64	20594 MB	jdk-8u261-macosx-x64.dmg
Solaris SPARC 64-bit (DVR4 package)	107.77 MB	jdk-8u261-solaris-sparcv9r2
Solaris SPARC 64-bit	88.72 MB	jdk-8u261-solaris-sparcv9r1
Solaris x64 (DVR4 package)	104.23 MB	jdk-8u261-solaris-x64r2

Account Creation Oracle (to download kit)

Η διαδικασία ολοκληρώθηκε επιτυχώς.
Ο λογαριασμός σας είναι έτοιμος για χρήση.

Ενημερώστε το λογαριασμό σας στην **Oracle** ανά πάσα στιγμή, από τους συνδέσμους στο πάνω μέρος των σελίδων της **Oracle.com**.

[Συνέχεια](#)

Βοήθεια λογαριασμού | Συνδρομές | Επικοινωνία συνδρομητή | Όροι χρήσης και προστασία προσωπικών δεδομένων | Προσέλευση cookies

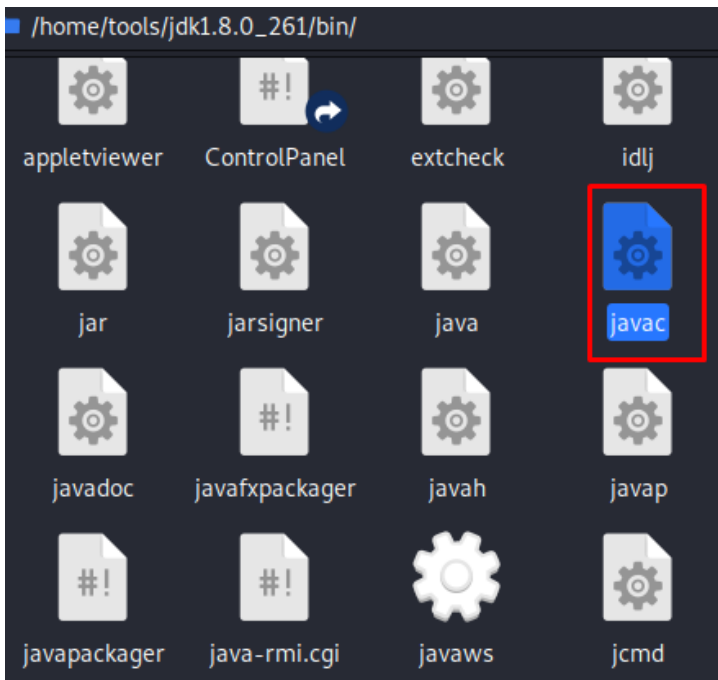
<https://community.linuxmint.com/tutorial/view/1372>

After the JDK installation we confirm that it is installed with the command `java -version`

```
root@john:~/Downloads# java -version
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
java version "1.8.0_261"
Java(TM) SE Runtime Environment (build 1.8.0_261-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.261-b12, mixed mode)
```

<https://stackoverflow.com/questions/35649140/make-bin-javac-command-not-found>

Next, we have to locate `javac` and set the `JAVA_HOME` path to it.




```
root@john:/home/tools/mulval# export JAVA_HOME=/home/tools/jdk1.8.0_261
root@john:/home/tools/mulval# make

root@john:/home/tools/mulval# export PATH=$PATH:$JAVA_HOME/bin
root@john:/home/tools/mulval# make
(cd src/adapter; make; make install)
make[1]: Entering directory '/home/tools/mulval/src/adapter'
javac GetCVEID.java -cp :/home/tools/mulval/lib/dom4j-1.6.1.jar:/home/tools/mulval/lib/jaxen-1.1.1.jar:/home/tools/mulval/lib/mysql-connector-java-5.1.8-bin.jar
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
javac GetTplQry.java -cp :/home/tools/mulval/lib/dom4j-1.6.1.jar:/home/tools/mulval/lib/jaxen-1.1.1.jar:/home/tools/mulval/lib/mysql-connector-java-5.1.8-bin.jar
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
```

https://vitux.com/how-to-setup-java_home-path-in-ubuntu/

The path should now look like this. If you try to compile MulVAL again the javac error should not appear.

```
root@john:/home/tools/mulval# echo $PATH
/home/tools/mulval/bin:/home/tools/mulval/utls:/home/tools/XSB/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/home/tools/jdk1.8.0_261/bin
```

SOLVED

Bison, flex no available candidates

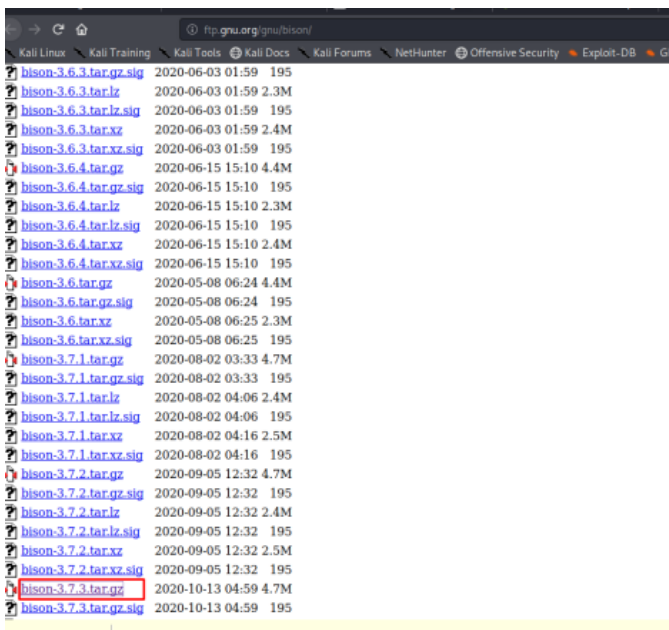
```
root@john:~# sudo apt-get install bison flex
Reading package lists... Done
Building dependency tree
Reading state information... Done
Package flex is not available, but is referred to by another package.
This may mean that the package is missing, has been obsoleted, or
is only available from another source

Package bison is not available, but is referred to by another package.
This may mean that the package is missing, has been obsoleted, or
is only available from another source

E: Package 'bison' has no installation candidate
E: Package 'flex' has no installation candidate
```

If apt-get install bison flex doesn't work there are two ways to install them. You can download and install them manually or by editing the etc/apt/sources.list file.

Manual Download and install

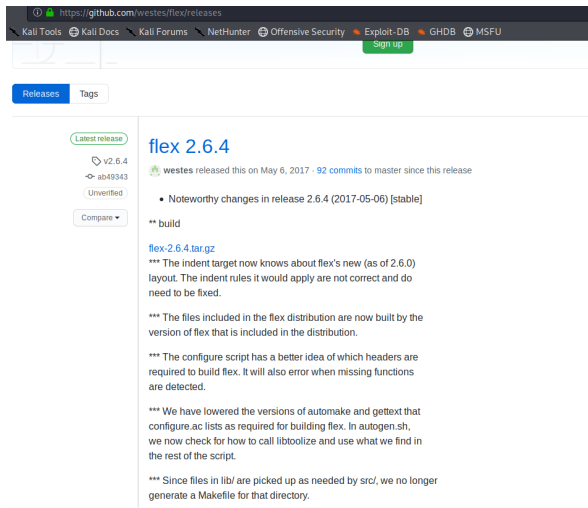


The screenshot shows a web browser window with the URL <http://gnu.org/gnu/bison/>. The page displays a list of bison tarballs for various versions (3.6.3, 3.6.4, 3.7.1, 3.7.2, 3.7.3) and formats (tar.gz, tar.xz, sig). The file `bison-3.7.3.tar.gz` is highlighted with a red box.

Manual Downloads

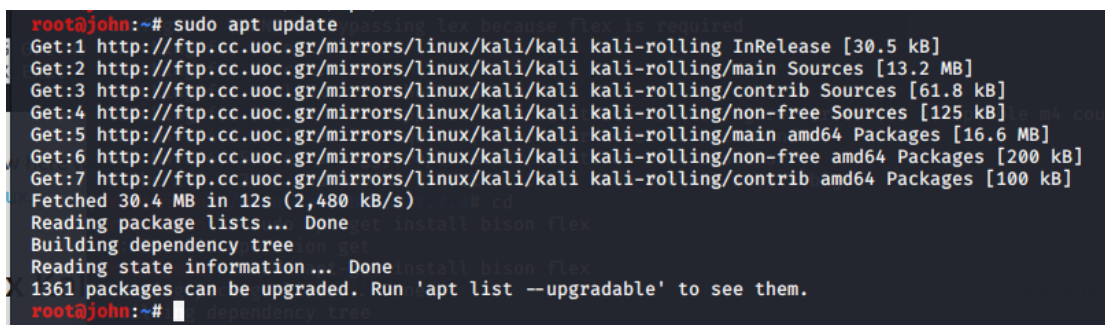
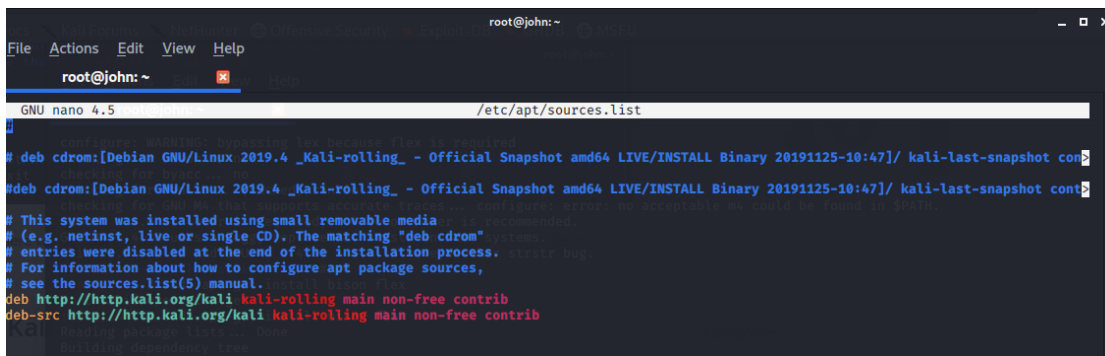
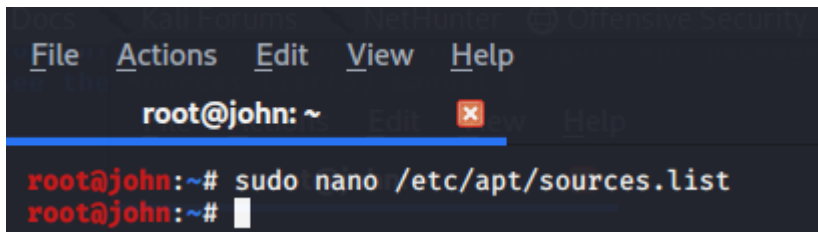
```
root@john:~/Downloads# tar xzf bison-3.7.3.tar.gz
root@john:~/Downloads# tar xzf flex-2.6.4.tar.gz
root@john:~/Downloads# cd bison-3.7.3/
root@john:~/Downloads/bison-3.7.3# ls
ABOUT-NLS  build-aux  Changelog-1998  configure.ac  doc  gnulib-po  lib  Makefile.am  PACKAGING  README-alpha  tests
aclocal.m4  cfg.mk     Changelog-2012  COPYING      etc  GNUMakefile  m4  Makefile.in  po  runtime-po  THANKS
AUTHORS     Changelog  configure       data         examples  INSTALL  maint.mk  NEWS  README  src  TODO
root@john:~/Downloads/bison-3.7.3# ./configure
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a race-free mkdir -p... /usr/bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking whether make supports nested variables... (cached) yes
checking whether make supports the include directive... yes (GNU style)
checking for gcc... gcc
```

Manual install



SOLVED

Alternate way



SOLVED

Appendix B

Python Scripts

Python Scripts that were used involve:

- **Attack Simulation** Attacks 6000 times. The user inputs how many attacks occur on each path.
- **Attack Simulation (weighted)** User defines the percentage of weight on each path and with that input, the script randomly attacks 6000 times.
- **Knapsack problem solver** Takes as parameters the data of Loss ($L(i)$) for each node and returns which nodes have to be protected
- **Brute force investments** The user manually inputs the investment percentage and the efficacy on each node.
- **BFS, BestFS** Heuristic Algorithms that return the results of Breadth First Search and Best First Search
- **Attack Path creator** The user inputs the connections of each node. Returns all possible attack paths.