

QATAR UNIVERSITY

COLLEGE OF ENGINEERING

A THREAT-SPECIFIC RISK EVALUATION TOOL FOR CLOUD ENVIRONMENTS

BY

ALAA SAYED AHMED HUSSEIN

A Project Submitted to
the Faculty of the College of
Engineering
in Partial Fulfillment
of the Requirements
for the Degree of
Masters of Science in Computing

June 2018

© 2018 Alaa Sayed Ahmed Hussein. All Rights Reserved.

COMMITTEE PAGE

The members of the Committee approve the Project of Alaa Sayed Ahmed
Hussein defended on 23/05/2018.

Khaled Khan
Thesis/Dissertation Supervisor

Qutaiba Malluhi
Committee Member

Jihad Jaam
Committee Member

Abdullatif Shikfa
Committee Member

ABSTRACT

HUSSEIN, ALAA, S., Masters : June : 2018, Masters of Science in Computing

Title: A Threat-Specific Risk Evaluation Tool for Cloud Environments

Supervisor of Project: Khaled M. Khan.

With the spread of using cloud computing; both as organizations and individual, it has become a target for attackers. The cloud environment has several weaknesses that pose threats to its users' assets. To assess any type of attacks, security administrators must regularly apply threat modelling techniques and run risk evaluation on cloud infrastructures. This allows them to identify risky assets and identify appropriate security controls to mitigate the risks. One of the key challenges with current risk evaluation approaches is that they do not distinguish the risks posed by different threats. The computation of the risk value compounds all threats.

In this project, we propose a threat-specific risk evaluation tool for security administrators. The tool allows security administrators to model topologies of their organization's networks. Then, using specific formulas, the tool will calculate the risk values for the entire system and for each component of the system with respect to specific threats based on Microsoft's STRIDE threat categorization. The key features of the tool are demonstrated through its application to cloud deployment example.

DEDICATION

To my parents, sisters and best friends.

ACKNOWLEDGMENTS

- وَأَخِرُ دَعْوَاهُمْ أَنْ الْحَمْدُ لِلَّهِ رَبِّ الْعَالَمِينَ -

First, I want to thank my parents, for their prayers and infinite love and support, it is what got me to this point. I am also grateful for my lovely sisters and best friends, I would not have made it without their support and advices.

Second, I would like to express my great gratitude and appreciation to my supervisor Dr. Khaled Khan and Dr. Armstrong Nhlabatsi for their unlimited efforts in assisting me in this project

TABLE OF CONTENTS

DEDICATION	iv
ACKNOWLEDGMENTS	v
List of Tables	viii
List of Figures	ix
Chapter 1: Introduction	1
Motivation.....	2
Problem Definition.....	2
Project Significance	4
High-Level Description of Solution.....	5
Organization of The Report	6
Chapter 2: Background	7
What is Threat Modelling?	7
The STRIDE Threat Model	7
An Example	9
Literature Review.....	9
Chapter 3: Requirement Analysis	15
Threat-Specific Risk Evaluation Approach	15
Risk Calculation.....	17
Functional Requirements	19
Use Case Diagram and Use Cases Specifications.....	20
Domain Model	22
CHAPTER 4: PROPOSED SOLUTION	23
System Architecture.....	23
Class Diagram.....	26
Design Sequence Diagrams	28
CHAPTER 5: IMPLEMENTATION	33
Tools and Frameworks.....	33
Design Pattern.....	33
Methodology	34
User Interface Design	41
CHAPTER 6: EVALUATION	50
Usability Study	50
Experiment Analysis.....	51

Experiment Setup.....	51
Experiments Results.....	52
CHAPTER 7: DISCUSSION.....	55
Limitations	56
CHAPTER 8: CONCLUSION	57
REFERENCES	58
APPENDIX A: DRAW NODE USE CASE SPECIFICATION	61
APPENDIX B: DRAW EDGE USE CASE SPECIFICATIONS.....	63
APPENDIX C: EVALUATE TOPOLOGY USE CASE SPECIFICATIONS.....	64
APPENDIX D: OPEN TOPOLOGY USE CASE SPECIFICATION.....	65
APPENDIX E: SHOW ATTACK PATHS USE CASE SPECIFICATIONS	66
APPENDIX F: USABILITY QUESTIONNAIRE	67

LIST OF TABLES

Table 1 Threat-specific risk calculations symbols and their definition.	17
Table 2 Mapping to STRIDE.	25

LIST OF FIGURES

Figure 1. Screenshot of Microsoft Threat Modelling Tool.....	12
Figure 2. Screenshot of analysis report of Microsoft Threat Modelling Tool.....	13
Figure 3. System use case diagram.	20
Figure 4. Domain model of the system.	22
Figure 5. System architecture for risk evaluation tool.....	24
Figure 6. Design class diagram.....	28
Figure 7. Draw Node use case DSD.	29
Figure 8. Connect Nodes use case DSD.	29
Figure 9. Open Topology use case DSD.....	30
Figure 10. Evaluate Topology use case DSD.	31
Figure 11. Show Attack Path use case DSD.....	32
Figure 12. MVC design pattern.	34
Figure 13. Initial screen of the system.....	41
Figure 14. Close-up of buttons panel.....	42
Figure 15. System Assessment Panel.....	43
Figure 16. Node Risk Evaluation Panel.....	43
Figure 17. Node Risk Severity Table.....	44
Figure 18. Attacker Paths Table.....	44
Figure 19: screenshot of drawing node with form.....	45
Figure 20. Close-up shot of VM properties form.	46
Figure 21. Screenshot of filled form.....	47

Figure 22. Screenshot of drawn topology.	48
Figure 23. Opening existing topology screenshot.....	49
Figure 24. Usability scores for the study questions.	51
Figure 25. Topologies of set A.	52
Figure 26. Topologies of set B.....	53
Figure 27. Topologies of set C.....	53
Figure 28. Topologies of set D.	54
Figure 29. Time Vs. number of nodes.	54

CHAPTER 1: INTRODUCTION

Cloud computing is a model where storing, managing and processing data takes place on remote servers, rather than on local servers. The exact definition is given by National Institute of Standards and Technology (NIST) [1] as “Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.” It is being more used nowadays for its various advantages, including sharing resources, elasticity, on-demand access, and scalability [2]. Sharing resources means having multiple virtual machines (VM) installed on a single host. In other words, different users’ applications are using different VMs running on the same hardware [3].

Resource sharing may impose some threats that represent potential violations of the system’s security [4]. To overcome these threats, security administrators usually perform risk analysis to determine which component is more vulnerable, or to what extent a component is at risk. The existing tools provide a coarse-grained analysis of risk that gives an overall risk evaluation. They usually either calculate the probability of a successful attack, or the risk imposed at a specific component.

This project aims to create a threat-specific security risk analysis tool that provides security administrators (SA) with accurate and relevant security risk assessment values for the different threats that are considered in the STRIDE threat model, which are *Spoofing*, *Tampering*, *Repudiation*, *Information Disclosure*, *Denial of Service*, and *Elevation of Privileges*. Chapter 2 will explain the STRIDE threat model in detail. The tool developed

for this project will allow security administrators to model their organizations' network as a topology of virtual machines, connections, and attackers. Depending on the characteristics of each virtual machine (VM) such as vulnerabilities, assets, and threats; the tool will evaluate the security risk for the whole system and for each VM in the network. More details about the computation of the risk will be explained later.

Motivation

The definition of cloud computing proposed by NIST carries several advantages of cloud computing. First, cloud computing is ubiquitous; it is convenient because mainly the user only has to host their application on the cloud, without worrying about how it is deployed, as technical support is handled by the cloud service provider. Moreover, cloud computing follows a pay-as-you-go model, namely only paying for exactly what you use, which makes it even more accessible to start-ups and individuals [5].

Despite its numerous benefits, cloud computing has several shortcomings. Some of the potential problems that could occur in the cloud environments include infrastructure failure, difficulty of identifying the source of a problem, data transfer bottlenecks, and more [5]. The multi-tenant model that enables serving multiple customers using the same resources may expose the cloud environments to several security issues. These potential issues in cloud security highly affect data confidentiality, integrity, and availability. The threats range from unauthorized access to data corruption, and service unavailability [5].

Problem Definition

Due to its various advantages, cloud computing is increasingly becoming an attractive option for organisations as well as individual users to store data, use hardware devices, and access to a wide choice of application systems. Because of its popularity, it

has also become a target for attackers. The cloud computing platform has various potential security weaknesses that attackers could exploit, such as unauthorised data access, denial of service, data corruption, etc. [5]. These weaknesses require users of cloud services to have tools for performing risk/threat analysis and assessment as a prerequisite for using cloud services.

One of the most effective ways to analyse threats is threat modelling [4]. It is an approach used for analysing an application security - by identifying possible threats to the assets such as sensitive data [6]. Once the probability and impact of threats has been measured through a risk evaluation, appropriate security measures can be taken to mitigate against the risk [6]. However, the existing tools for risk analysis mainly compute the total/compound risk for the system, without detailing, in a fine-grained manner, the risk for each possible threat. This makes it difficult for security administrators to decide about what mitigation strategies to take against each threat.

There are several threat models, such as STRIDE [7] and CIA [8]. CIA categorises threats according to their violation of the main security goals, which are Confidentiality, Integrity, and Availability. The STRIDE threat model, that we are using in this project, is a threat classification model developed by Microsoft. It classifies threats into six different categories, depending on the attackers' goals, namely: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege [7].

To demonstrate the issue, a hospital system would make a good example. This is because patients' medical records must not be disclosed to unauthorized entities (confidentiality), and they should not be tampered by unauthorized entities (integrity). In a hospital system, leaking patients' data or modifying them without authorization would do

huge damage to the patients, and to the reputation of the hospital.

With existing tools, it is not possible to tell the extent of the risk of a cloud deployment with respect to specific security threats. Different organisations may care about different threats depending on the nature of their business. For example, the hospital may care more about confidentiality of medical records but for a banking institution the integrity of financial transaction can be a high priority. Risk evaluation with respect to specific threats would enable the two organisations address the threats that matter the most for their respective businesses.

Project Significance

The development of a tool to support threat-specific risk evaluation is important in the security risk assessment field for the following reasons:

- Security risk analysis is essential and a valuable exercise for every organization in order to know in advance what risk they face for their data and applications while using the Cloud services due to vulnerabilities in their network infrastructure.
- It is also important for system administrators to understand which specific node (e.g., VM) has which threat(s) and the degree of severity of the threat(s). Knowing these types of risk help them decide more accurately the course of action that must be taken to protect and mitigate against specific threats.
- As stated earlier, the STRIDE provides a framework for classifying threats. However, currently there is a limited number of tools that take advantage of this threat classification to evaluate risk with respect to each type of threat in the

STRIDE. Knowing the risk with respect to each threat has the potential to help security administrators (SA) have a better understanding of their organizational compound and individual security risk on their assets. This, in turn, will help a security administrator to identify which threat is most critical, know how to address it appropriately, and design more effective mitigation strategies.

- Lastly, it will help the management of the organization allocate appropriate resources to mitigate against security threats that are most relevant according to their business objectives, security requirements, and the specific threats they face.

High-Level Description of Solution

The suggested solution to the problem defined earlier is a risk evaluation tool for security administrators to create graphical models of their networks (nodes and edges), and then evaluate the systems' risk in terms of numerical values. The expected outputs of this tool are:

- A set of values that include the risk of the entire system,
- The magnitude of risk for each virtual machine, and
- The degree of risk with respect to each threat category of the STRIDE.

The idea is to represent a network infrastructure of an organization by drawing a set of virtual machines, one or more attackers, and a set of edges connecting the virtual machines to the attackers. It should be possible for a security administrator to set detailed properties of each virtual machine such as its name, function, type of operating system, and the suitable impact weight for each of the STRIDE components.

In addition to the basic information about a virtual machine, more importantly, it should be possible to capture information about the vulnerabilities it has. Each vulnerability can be characterized by its impact and exploitability. Exploitability is the probability that the vulnerability can be exploited successfully by an attacker to compromise the virtual machine (component), and impact is a measure of the damage that can be suffered by the organization if the vulnerability is exploited successfully.

Upon the completion of the system topology drawing, the evaluation takes place by calculating the risk for each VM and for the whole system based on the vulnerabilities it has. The computation should also show the risk for each threat category (STRIDE) for each VM and for the entire system.

Organization of The Report

This report is structured as follows: chapter 2 discusses the background of the problem and the existing solutions. Chapter 3 provides the requirement analysis of the system. Chapter 4 presents the detailed design of the proposed solution and chapter 5 discuss the implementation details. In chapters 6 an evaluation of the tool on network topologies is presented. Chapter 7 presents a discussion of some of the design rationale and implementation alternatives. Finally, in Chapter 8, we conclude our project with a summary of pointers for further work.

CHAPTER 2: BACKGROUND

This chapter gives background information about threat modelling process and specifically the STRIDE threat model. It also covers some of the related works.

What is Threat Modelling?

As mentioned in the introduction, threat modelling is one of the many ways that help identifying and addressing security *threats*. Generally, threat modelling consists of three main stages. First, identifying the assets of the system and their vulnerabilities. Second, analyzing the possible threats to the system. Lastly, defining any countermeasures that can be used to prevent the threats analyzed [9].

The STRIDE Threat Model

The threat model used for our proposed tool is the STRIDE classification model, developed by Microsoft [7]. This model categorizes threats based on the motivation of the attacker. The acronym is formed by taking the first letter of each of the following categories:

- ***Spoofing Identity***: where a person or a program masquerades to be another legitimate person or program. An example would be a malicious person illegally accessing a legitimate user's credentials (i.e. username and password) and using them as if they were the genuine user.
- ***Tampering with Data***: that is malicious modifications of data. For example, modifying persistent data in the database by unauthorized entities such as the amount paid in a payment is modified to a higher value without the knowledge of the payee.

- ***Repudiation or non-repudiation***: is when a user denies performing an action and other parties cannot prove otherwise. In such cases, the attacker performs an illegal act but it is not traceable, so other parties will not be able to prove that the attacker did the illegal act. For example, a client paid a bill of a merchant using her credit card, but the merchant denied later that he received the payment.
- ***Information Disclosure***: this type of threats involves revealing information that should not be revealed, whether revealing publicly (to everyone), or revealing to a person that should not have access to that information. For example, a legitimate user for a system reading a file that they should not have privilege to read, or an intruder reading some data in transit between two computers.
- ***Denial of Service***: these attacks result in service failures to valid users. For example, a web server being temporarily unavailable due to the attacker flooding the it with false requests such that it is too busy to respond to genuine requests for service.
- ***Elevation of Privilege***: where users with limited privileges gain higher privileges that they should not otherwise have. Such privileges enable the attacker to compromise the entire system hugely. For example, if an attacker gains root access to a host he may delete, modify, or migrate virtual machine hosted there – thus compromising the security of the cloud deployment.

An Example

To further illustrate the STRIDE threats, let us refer to the hospital example again. A hospital system is a very sensitive system, and must always comply with the security objectives of the hospital. Patients' records must be confidential and accessible by only authorized users. Data integrity must also be fulfilled to ensure that any modification is authorized such as changes to prescriptions or ailments suffered by a patient. Lastly, the availability of data is critical for such system, because in some cases, a patient's life may depend on the availability of their medical history in a critical situation. So, breaching any of these three security goals does not only harm the patients, but can destroy the hospital's reputation as well.

For such security-aware critical systems, a security risk analysis tool is essential in order to assess the security posture of the cloud deployment with respect to each of the threats described above. Our tool would provide the needed analysis to support such risk assessment activities. Since the patients' medical records should maintain confidentiality, integrity, and availability, it is logical for the security administrator using our tool to give more weight to the threats affecting those three security objectives. Following the STRIDE model, this hospital system may give the information disclosure, tampering, and denial of service components high impact weights than the rest of the STRIDE elements to correspond to the confidentiality, integrity and availability respectively.

Literature Review

Security risk assessment has been thoroughly studied in the literature, and several metrics have been developed. However, a limited number of frameworks have been implemented into risk assessment tools. We will discuss three well-known tools in this

section.

Cyber Security Modelling Language (CySeMoL) [10] is a tool created to evaluate the vulnerabilities of enterprise system architectures. It is designed to cover several attacks, including but not limited to, flooding attacks, software exploits, and social engineering attacks. It is built based on a Probabilistic Relational Model (PRM). The PRM specifies how to create Bayesian network from a class diagram-like model, similar to the one produced by UML. The classes in a PRM consist of two parts; attributes and reference slots which are organized into templates. A security-risk analysis template defines abstract classes, attributes, reference slots and conceptual-attribute parents. This template has the classes: Asset, Owner, Threat, ThreatAgent, AttackStep, and Countermeasure. The countermeasures are: Contingency Countermeasure, Preventive Countermeasure, Detective Countermeasure, Reactive Countermeasure, Accountability Countermeasure.

Constructing a PRM according to this template, along with assigning the PRM's conditional probabilities allows using the PRM to perform an analysis that is used to calculate reachability values for different attack paths, which is used by CySeMoL. Assessing security risk with CySeMoL involves creating a Bayesian network for each identified attack path – paving way to the calculation of the success probabilities of attack for each attack step. The key limitation of CySeMoL is that it only focuses on calculating the probability of an attack. Our tool complements CySeMoL as it is aimed at evaluating risk from the perspective of specific threats, assuming that the probability of attacks is already known.

Another risk assessment tool is defined in [11]. This tool provides a quantitative risk assessment for each VM, physical hosts, and SLAs by using objective data to

determine the probability of events and associated risks. The risk assessment process has 6 stages, namely: Risk inventory, Vulnerability identification, Threat Identification, Data monitoring, Event Analysis, Quantitative risk analysis. One limitation of this tool is that although threats are identified, it does not provide a comprehensive risk assessment mechanism detailing risk with respect to those threats.

Microsoft has developed a threat modelling tool called Microsoft Security Threat Modelling Tool [12], which is the core of their Security Development Lifecycle. It allows the software designers to identify possible security issues early, and mitigate them properly. *Figure 1* below shows a screenshot of the Microsoft Threat Modelling Tool. In addition to the drawing canvas, this tool has the “Stencils” panel, where the different system components are placed, such as database, web application, connection, browser, and several other components. It also contains different types of connections like “REQUEST” and “RESPONSE”.

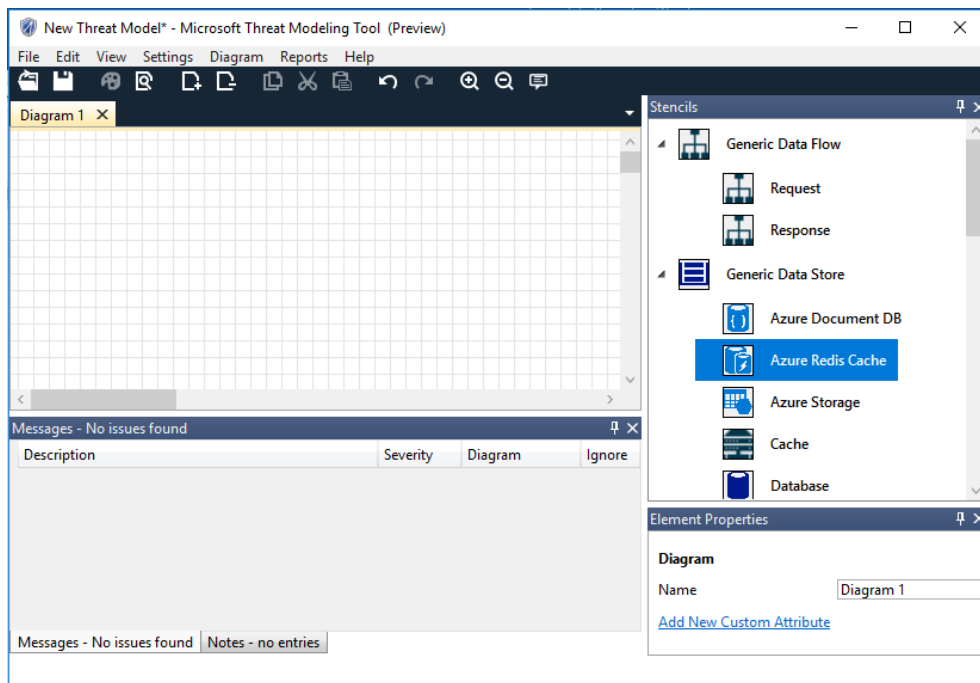


Figure 1. Screenshot of Microsoft Threat Modelling Tool

After drawing the diagram, the analysis is done by asking to tool to generate an analysis report. As Figure 2 illustrates, the report contains different pieces of information about the potential threats. It gives details about the threat such as description, category, possible mitigation strategy.

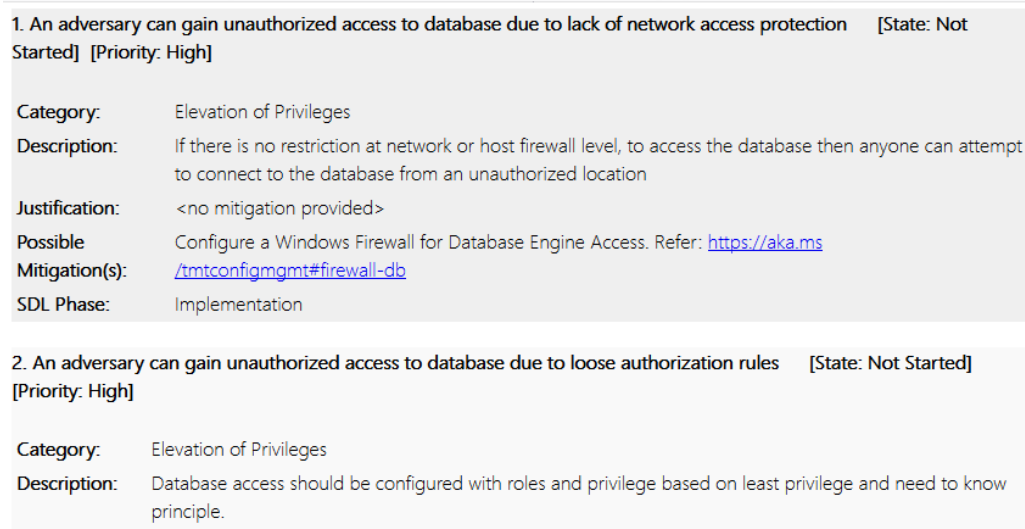


Figure 2. Screenshot of analysis report of Microsoft Threat Modelling Tool

There are several differences between the Microsoft Threat Modelling Tool and our proposed tool. First, Microsoft Threat Modelling Tool gives a qualitative analysis to the different components, rather than a quantitative analysis. It labels the components threats severity as High and Low, but these two values have a wide range of severity that is not captured using the qualitative analysis. Moreover, when evaluating the component risk, Microsoft Threat Modelling Tool uses default impacts weights depending on the component type, rather than the impacts weights defined by the organizations' security objectives. Lastly, Microsoft Threat Modelling Tool requires much deeper understanding of the system components and their connections. It is very detailed in terms of diagrams and it takes the components to the lowest level. On the other hand, our tool is more generic in terms of components. It does not have numerous different types of components and connections. This makes it easier to use by the software designers, as it does not require

deep understanding of how smaller components build big components or how the components are linked using different types of connections.

In summary, the existing security risk analysis tools mainly provide either a probability of a successful attack, like CySeMoL, or a specific risk value for the virtual machine or the physical host, such as the tool in [11]. On the other hand, our security risk analysis tool provides a more accurate risk value tailored to specific threats of the component and security requirements of the organization. The key benefit of threat-based risk analysis is that it will make it possible to tailor mitigation mechanism to specific threats. Such customized design of mitigation strategies is not feasible with the current existing risk evaluation tools.

CHAPTER 3: REQUIREMENT ANALYSIS

The proposed tool is built to calculate the risks based on a specific approach defined in [13]. This chapter first explains in detail what these formulas are and how they are defined. Then, the functional requirements of the tool are described. It also includes the use case diagram and the use cases specifications. Lastly, it shows the domain model of the system. This project mainly implements the risk evaluation tool and is not responsible for the shortcomings of the approach.

Threat-Specific Risk Evaluation Approach

The risk assessment tool is developed based on the formulas defined in [13]. In [13], Nhlabatsi et. al propose a threat-specific risk evaluation approach for cloud computing, which can evaluate the risk associated with each threat category in the STRIDE. After discovering a vulnerability in one of the cloud components (VMs), the security administrator determines two things. First, the type of threats that could be posed if the vulnerability is successfully exploited. Second, the impact of each of the determined threats on the assets of the client. Assigning the impact weighting of each threat type depends on the importance of the component. To use the hospital example one more time, the web servers that run the hospital system should have more weight for denial of service threats, since successful attacks on them would make the whole system unavailable. However, the hospital database must have more weight for information disclosure threats, as if it is compromised, the patients' confidentiality would be violated.

This approach has several equations containing different variables. *Table 1* defines all the symbols and functions that are used in the risk computations. Each vulnerability (v_j) has

an impact value (α_{v_j}) measuring the damage to an asset in case of successfully exploiting the vulnerability and an exploitability value (β_{v_j}) measuring the probability of successfully exploiting that vulnerability. The impact and exploitability of a node having several vulnerabilities are calculated using equation (1) and equation (2), respectively.

$$\lambda(n_i) = \sum_{j=1}^{|V(n_i)|} \alpha_{v_j}, v_j \in V(n_i), V(n_i) \subseteq \mathbb{V} \quad (1)$$

$$\mu(n_i) = 1 - \prod_{j=1}^{|V(n_i)|} \{1 - \beta_{v_j}\}, v_j \in V(n_i), V(n_i) \subseteq \mathbb{V} \quad (2)$$

In equation (1), $\lambda(n_i)$ represents the total impact of node n_i , which is a summation of all its vulnerabilities' impacts. While in equation (2), $\mu(n_i)$ represents the exploitability of node n_i which is the combined probability of failing to exploit all the vulnerabilities. Equation (3) below defines $\Gamma(n_i)$ as the total threats of component n_i which are derived from the threats posed by the component's vulnerabilities. This set of threats is formulated by taking each vulnerability from the set of vulnerabilities of component n_i , $V(n_i)$, and then taking the threats posed by each vulnerability, $\Gamma(v_i)$.

$$\Gamma(n_i) = \left| \begin{array}{c} V(n_i) \\ j = 1 \\ \hline \gamma_{v_j} \\ t = 1 \end{array} \right|_{t=1} \kappa \leftarrow t, \forall t \in \gamma_{v_j}, \forall v_j \in V(n_i) \quad (3)$$

Risk Calculation

Component Risk Calculation:

Equation (4) shows the total threat risk of component n_i , which is a product of the overall impact $\lambda(n_i)$, combined probability $\mu(n_i)$, and the sum of all impacts of threats imposed by the vulnerabilities in the component n_i .

$$Risk(n_i) = \lambda(n_i) * \mu(n_i) * \sum_{t=1}^{|\Gamma(n_i)|} \omega(n_i, t) \forall t \text{ in } \Gamma(n_i) \mid \Gamma(n_i) \in 2^\Phi \quad (4)$$

Threat Risk Calculation:

$$Risk(n_i, \kappa) = \lambda(n_i) * \mu(n_i) * \sum_{t=1}^{|\kappa|} \omega(n_i, t), \forall t \in \kappa \mid \kappa \subseteq \Gamma(n_i) \Rightarrow \kappa \in 2^\Phi \quad (5)$$

The component threat risk is calculated in a similar manner. Instead of taking the whole set of threats $\Gamma(n_i)$ in component n_i , only the selected threats κ that are imposed by the vulnerabilities in component n_i are considered.

State Risk Calculation:

Lastly, to compute the total risk of the system, the risk values ($Risk(n_i)$) of all system's components are summed up, as shown in equation (6).

$$Risk(\Omega) = \sum_{i=1}^{|\mathbb{N}|} Risk(n_i), \forall n_i \in \mathbb{N} \quad (6)$$

Table 1

Threat-specific risk calculations symbols and their definition.

Symbol	Definition
\mathbb{N}	The set of all components/nodes in the cloud system, $\mathbb{N} = \{n_1, n_2, n_3, \dots, n_i\}$
\mathbb{E}	Set of all connections between components in the cloud system, $\mathbb{E} \subseteq \{\mathbb{N} \times \mathbb{N}\}$
Φ	A set of STRIDE threats, $\Phi = \{S, T, R, I, D, E\}$
t	A single threat in the STRIDE threats, $t \in \Phi$
2^Φ	The power set of Φ , $2^\Phi = \left\{ \{\}, \{S\}, \{T\}, \{R\}, \{I\}, \{D\}, \{E\}, \{S, T\}, \{S, R\}, \dots, \{S, T, R, I, D, E\} \right\}$
Ω	A cloud system consisting of a network of nodes and connections, $\Omega = \{\mathbb{N}, \mathbb{E}\}$
n_i	A component in a cloud system/sub-system, $n_i \in \mathbb{N}$
\mathbb{V}	The set of all vulnerabilities in the cloud system.
v_j	A j^{th} vulnerability in a component of a cloud system, such that $v_j \in \mathbb{V}$
α_{v_j}	Impact of a vulnerability v_j
β_{v_j}	Probability of successful exploitation of vulnerability v_j
γ_{v_j}	Set of all threats that can exploit vulnerability v_j , where $\gamma_{v_j} \in 2^\Phi$
κ	A set of selected threats for the risk evaluation given $\kappa \in 2^\Phi$
$\lambda(n_i)$	Impact on component n_i - calculated from the set of vulnerabilities in the component.
$\mu(n_i)$	Probability of a successful attack on component n_i - calculated from the set of vulnerabilities.
$V(n_i)$	Set of vulnerabilities in component n_i , where $V(n_i) \subseteq \mathbb{V}$
$\Gamma(n_i)$	Set of threats in component n_i , where $\Gamma(n_i) \in 2^\Phi$
$\omega(n_i, t)$	The impact weighting of component n_i (as set by the security administrator) from the perspective of one of the STRIDE threats, $t \in \Phi$. The impact weighting depends of user security requirements.
$Risk(\Omega)$	Computes the risk of a cloud system considering all threats.
$Risk(\Omega, \kappa)$	Computes the risk of a cloud system considering a subset of threats κ .
$Risk(n_i)$	Computes the risk of a cloud component considering all threats.
$Risk(n_i, \kappa)$	Computes the risk of a component in a cloud system considering a subset of threats κ .

Functional Requirements

The main functional requirement of the proposed system is to evaluate risk. However, to accurately calculate the different values of VM risk, threat risk and state risk, a few requirements must be fulfilled first. This section covers in depth the system functional requirements.

The first requirement is to *create network topologies*. Each topology is created when the security administrator *draws nodes* (VMs and attackers) and *connects VMs* either to each other or to an attacker. This network topology developed by the system administrator represents the system of the organization.

Upon drawing a VM, the security administrator will be required to enter the information of that VM, such as its name, functionality (e.g. host, firewall...etc.), the operating system it runs and its version. The tool focuses on the operating system of the virtual machine, but can be extended to include the list of applications running on that virtual machine. The operating system and its version will provide us with a genuine list of vulnerabilities existing on the specified operating system with the selected version. The vulnerabilities are retrieved from the remote *National Vulnerability Database (NVD)* [14], which is a huge repository for all the security-related software flaws from the year 1999 up till 2018. The system administrator will get data directly from NVD.

Retrieving the list of vulnerabilities for each VM is crucial to calculate its impact value and its exploitability (i.e. probability of attack) according to formulas (1) and (2).

After creating the network topology, the system will then evaluate the risk upon security administrator's request, using the equations (4), (5), and (6) to calculate VM risk, threat risk, and state risk, respectively.

Use Case Diagram and Use Cases Specifications

Figure 3 below shows the use case diagram of our security risk analysis system. There are four main use cases: *Draw Node*, *Connect Nodes*, *Evaluate Topology*, and *Show Attack Paths*. The remaining use cases are *Retrieve Vulnerabilities*, *Make Topology*, and *Open Topology*. An external system is also shown that represents the National Vulnerability Database, that is by the Retrieve Vulnerability use case to download the vulnerabilities.

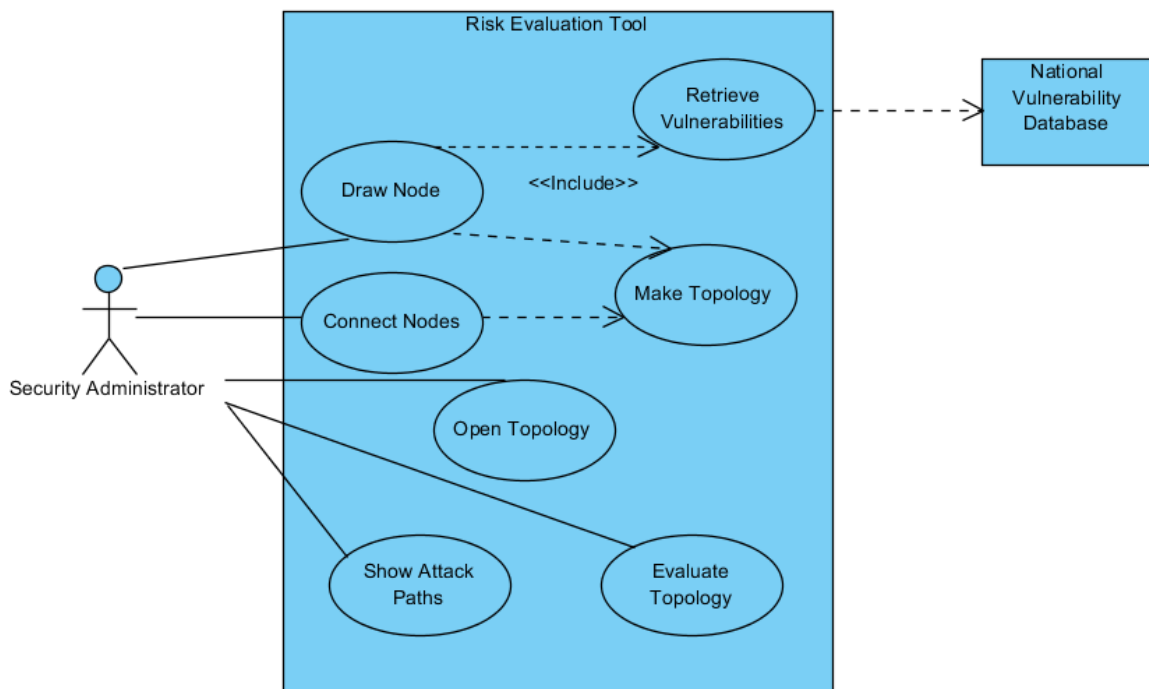


Figure 3. System use case diagram.

The following section will elaborate on the specifications of each use case.

- 1. *Draw Node*:** as the name suggests, it allows the security administrator to draw a VM on a particular point on the screen, and then enter the information of this VM, such as its name, functionality, operating system, and operating system version. From the last two pieces of information, operating system and operating system version, this use case retrieves the list of vulnerabilities from NVD that this version has. Upon retrieving the VM vulnerabilities, the impact score and exploitability score are automatically calculated based on equation 1 and equation 2 respectively. A node can also represent an attacker, where all the information will be null, except for the ID and name.
- 2. *Connect Nodes*:** allows the security administrator to connect two VMs, or a VM to an attacker. This use case along with Draw Node use case combined make the *Make Topology* use case.
- 3. *Open Topology*:** is used to retrieve a topology that has already been drawn and saved.
- 4. *Evaluate Topology*:** calculates the system risk, threat risk, and node risk and reports them to the security administrator.
- 5. *Show Attack Paths*:** prints the paths for a particular target VM from the attacker to the VM.

The complete use case specifications can be found in Appendices A to E.

Domain Model

After defining the main use cases, the main entity classes that this system has are shown in *Figure 4*, the domain model. This model shows the required classes which are Node, Edge, Vulnerability, Path, and RiskReport. It also shows the cardinality between the classes. For example, one node may have zero or many incoming edges; where zero incoming edge means it is *not a destination node* from other nodes. A node may also have zero or many outgoing edges, where zero outgoing edges means it is not connected *as a source node* to other nodes. Moreover, a node can have from 1 to several vulnerabilities. A node can also have zero to many attack paths. Finally, a RiskReport is generated from the information gained from all the other classes.

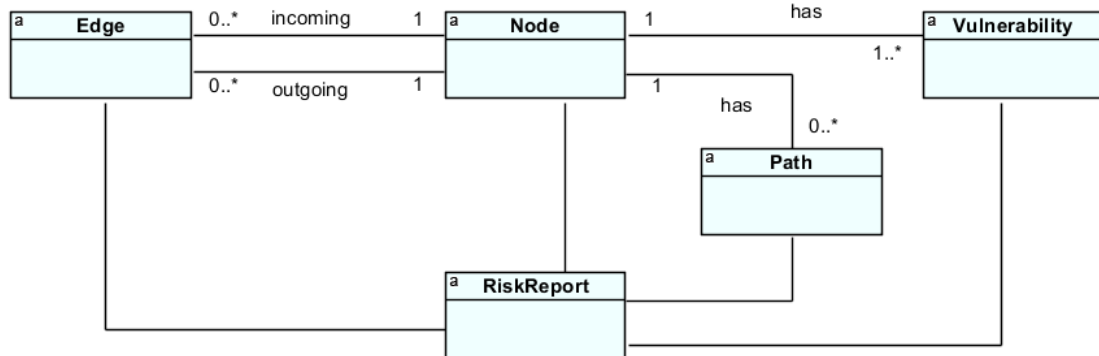


Figure 4. Domain model of the system.

CHAPTER 4: PROPOSED SOLUTION

After explaining the system requirements, this chapter will talk about the proposed tool in depth. It will illustrate the system architecture, design class diagram and system sequence diagrams.

The proposed system is a risk assessment tool, that will be used by security administrator to help them in evaluating their organization's security risk. As explained earlier, the tool provides a clear canvas for the security administrator to create their network topology by drawing collection of virtual machines and the possible attackers, then connecting them via edges. The security administrator will enter the properties of each VM, such as its name, function, operating system, and the suitable impact weight for each of the STRIDE components. As mentioned in Chapter 3, the impact and exploitability of each VM is calculated based on its list of vulnerabilities. When the security administrator completes the topology drawing, the tool will calculate the risk for each VM and the risk of each of its threats (STRIDE), the state risk of the system, and the risk for each threat.

System Architecture

The architecture of the proposed tool is demonstrated in *Figure 5* below. As explained in the previous chapter, a database of vulnerabilities is needed in order to perform the computations. Using the bash script provided in [15], 17 XML files are downloaded, for the years 2002 to 2018. Each file contains information about the vulnerabilities retrieved from the National Vulnerability Database, using CVSS v2.0.

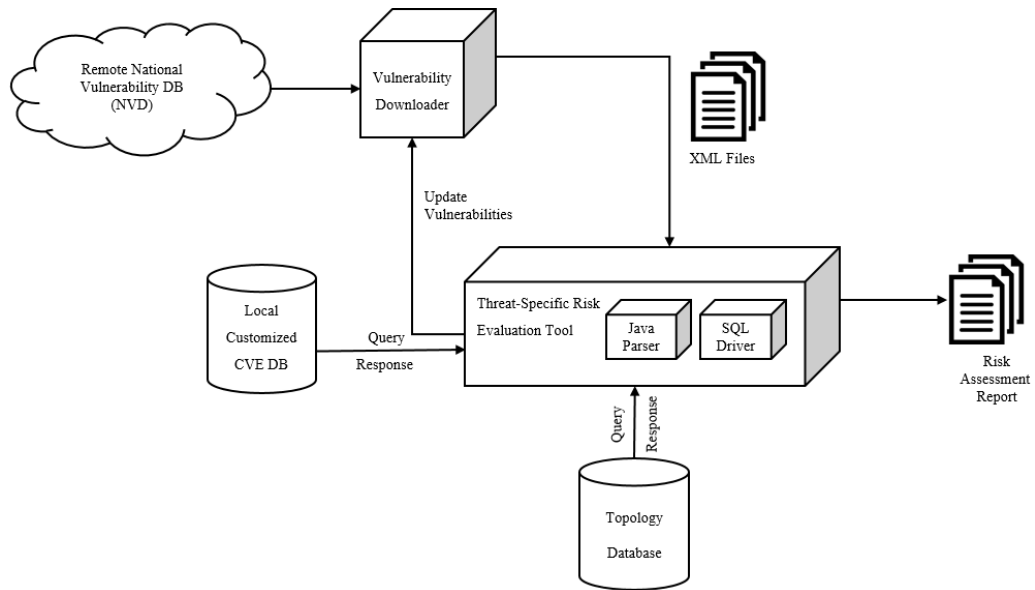


Figure 5. System architecture for risk evaluation tool.

For the sake of this project, only the 2018 file is considered, but the tool can be modified to include all the vulnerabilities files. Since the XML file contains all the information about each vulnerability, it is very large. This makes accessing the desired pieces of information a bit slower. Therefore, a Java parser called “DocumentBuilderFactory” is used to extract the needed information from the XML file, which are *Vulnerability ID*, *Impact Score*, *Exploitability Score*, and *Impact Vector*. However, the Impact Vector stored does not follow the STRIDE model. It is stored as:

CVSS_vector="(AV:N/AC:L/Au:N/C:P/I:P/A:P)"

AV: Attack Vector

AC: Attack Complexity

Au: Authentication

C: Confidentiality

I: Integrity

A: Availability

‘N’ means None, ‘L’ means Low, ‘P’ means Partial, ‘H’ means High. We are mainly interested in the latter four; authentication, confidentiality, integrity, and availability. To match the Risk Evaluation Formulas (defined in Chapter 3), the categorizing model used by NVD must be mapped to STRIDE model, this is illustrated in *Table 2* below.

Table 2

Mapping to STRIDE.

Security Objectives	Security Threat
Authentication	Spoofing
Integrity	Tampering
Confidentiality	Information Disclosure
Availability	Denial of Service

Then, a Java SQL Driver creates a local database, having a table for all the vulnerabilities consisting of six columns: *Vulnerability ID*, *Impact Score*, *Exploitability Score*, *Operating System*, *OS Version*, and *STRIDE Elements*. Another database is linked to the tool where the previously created network topologies are stored.

Class Diagram

As outlined in the Domain Model (Chapter 3), the main entities of the risk evaluation tool are *Node*, *Path*, *Edge*, *Vulnerability*, and *Risk Report*. This section will elaborate on each entity and what is captured by the objects of these classes. The *Node* class represents a virtual machine or an attacker, having these attributes:

- ID: a unique identifier for each node.
- X: an integer value representing the x-coordinate of the center of the node.
- Y: an integer value representing the y-coordinate of the center of the node.
- Radius: an integer value to represent the size of the node on the screen.
- Name: the name provided by the security administrator when entering the VM properties.
- Type: the type of the node, either VM or attacker. The following attributes are set to null if the type is 'attacker'.
- Function: the role of the VM, (e.g. firewall, host...etc.).
- Operating system; the operating system that is running on the VM.
- OS version: the operating system version.
- Impact: the impact of the VM after considering its list of vulnerabilities.
- Probability: the exploitability score of the VM after considering its list of vulnerabilities.
- STRIDE: a list of 6 doubles, where each value represents the impact weighting of each threat of the STRIDE.
- Risk: the component risk, that is calculated using formula (4).

The class *Edge* represents the connection between the nodes (i.e. VM to VM or Attacker to VM), and it has the attributes: Source Node and Destination Node.

Each node (if it is a VM) has a set of vulnerabilities, that are represented in the class *Vulnerability*. This class has the attributes:

- ID: the vulnerability CVE ID.
- Impact: the impact score of the vulnerability (retrieved from local CVE database).
- Probability: the exploitability score of the vulnerability (retrieved from local CVE database).
- Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privileges: Boolean values that represent whether the vulnerability poses the specified threat or not.

Lastly, the class *Risk Report* captures the risk report for the evaluation process. It has the attributes State Risk and STRIDE Risks, where the system's risk and each threat risk are calculated. The class diagram of the system can be found in *Figure 6* below.

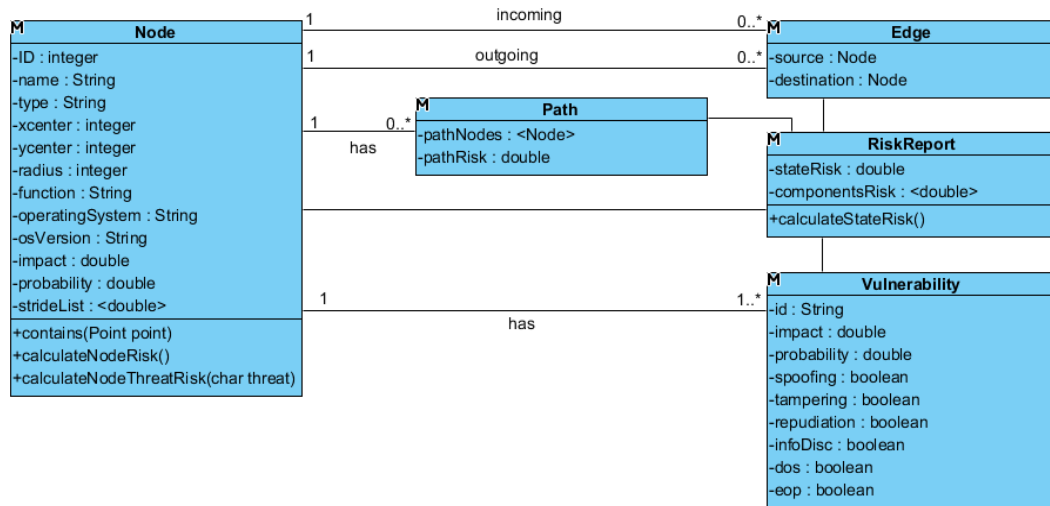


Figure 6. Design class diagram.

Design Sequence Diagrams

It is important to use design sequence diagrams (DSD) to show the interaction between the security administrator and the different components of the system. This section will illustrate the DSDs for the main use cases:

Figure 7 below shows the DSD of Draw Node use case. The user interacts with the system solely through the user interface. The interface then interacts with the controller, that handles everything else. It gives the order for object creation and initializing.

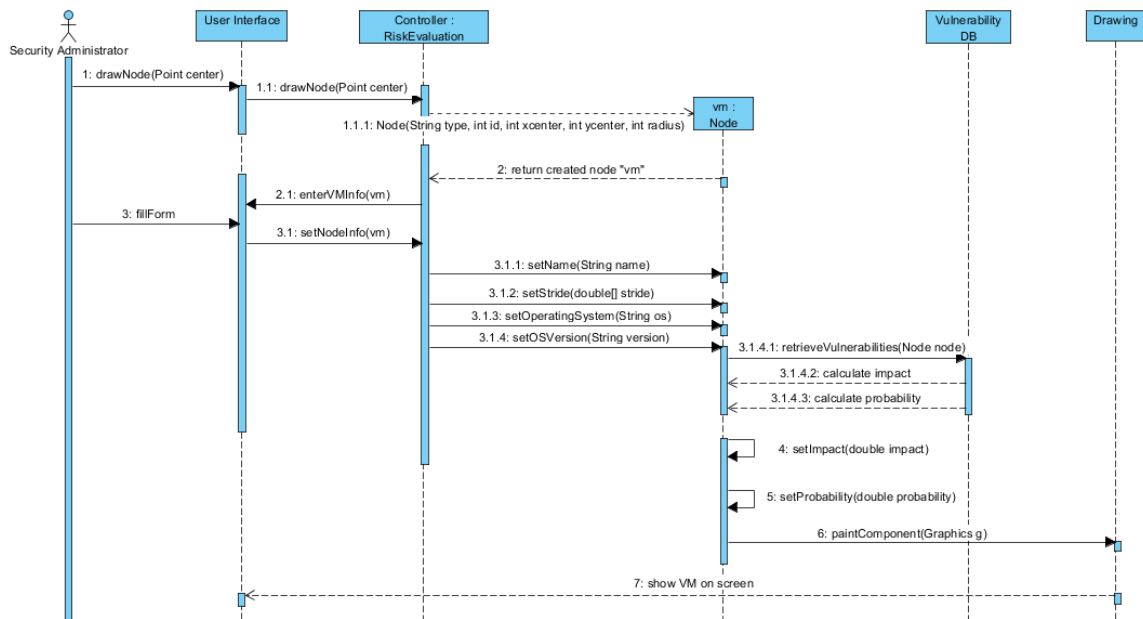


Figure 7. Draw Node use case DSD.

After drawing nodes, the user should connect them. *Figure 8* shows the DSD of Connect Nodes use case. The user first clicks on the nodes to be connected, the controller will then create an Edge, and will order the Drawing class to draw a line between the two nodes.

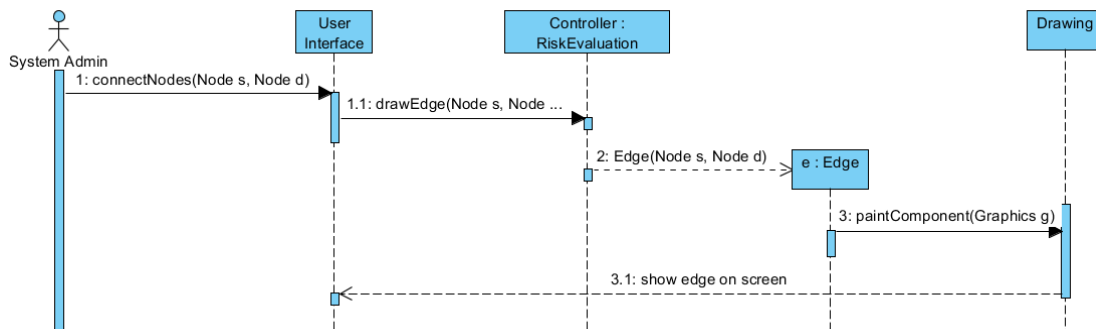


Figure 8. Connect Nodes use case DSD.

Figure 9 illustrates the DSD of Open Topology use case. The user selects an existing topology file, and the controller will then read the file, extracts the nodes information and the edges information. Lastly, it orders the Drawing to paint all the components.

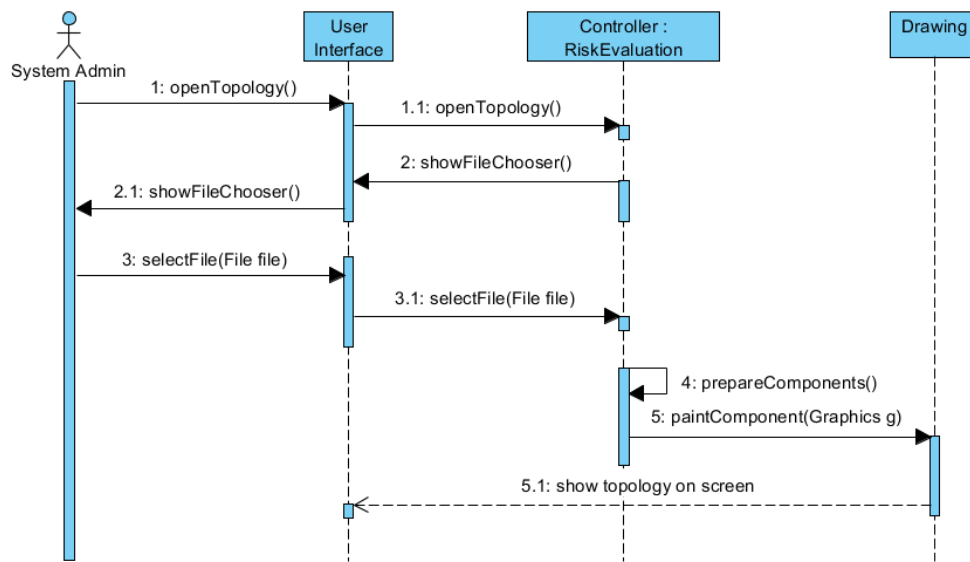


Figure 9. Open Topology use case DSD.

When the topology is completed, the evaluation begins once the user clicks the “Evaluate” button. The DSD of this use case is shown in Figure 10. The controller will gather all the nodes’ risk values, and their threat risk values (STRIDE). It will then calculate the threat risk values for all STRIDE elements, and the state risk of the system. All the calculated values will be shown to the user.

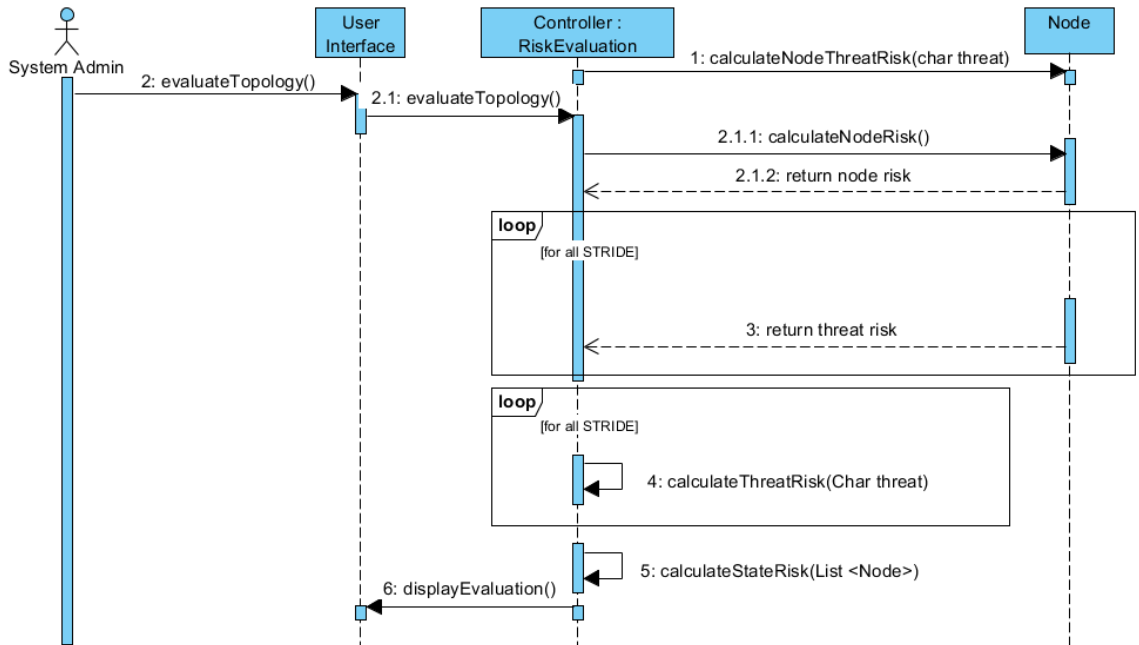


Figure 10. Evaluate Topology use case DSD.

Figure 11 below shows the DSD of the “Show Attack Paths” use case. When drawing the topology is complete, each node automatically stores its attack paths as an attribute. When the user click on one virtual machine to view its information, the controller gets all the paths for the selected virtual machine, along with each path’s risk value. Then, the paths and the risk values will be shown to the user.

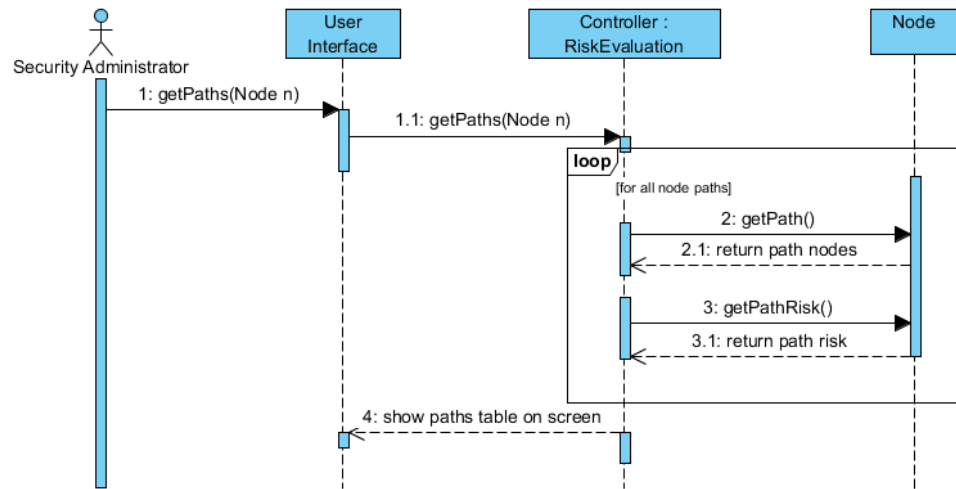


Figure 11. Show Attack Path use case DSD.

After analyzing the system and its requirements, the implementation details are explained in the next chapter.

CHAPTER 5: IMPLEMENTATION

This chapter will cover the main implementation aspects, such as tools used, pattern used, and screenshots of the user interface.

Tools and Frameworks

The IDE used to develop the Risk Analysis Tool is NetBeans IDE 8.1. with Java Derby Database to store the needed tables. Moreover, a VM running Ubuntu was used to run the bash script files that downloads the vulnerabilities information and stores them as XML files.

The user interface of the tool is designed using mainly Java Swing Toolkit [16]. It is an Application Programming Interface (API) that implements a set of components for designing graphical user interfaces (GUIs). The toolkit enhances Java applications with rich graphics interactivity and functionality. Another important component that was used is the “DocumentBuilder”. It defines an API that help programmers obtain a Document Object Model (DOM) instance from an XML document. The method *parser(File f)* of this class was used to obtain a Document object of the vulnerabilities XML file.

Design Pattern

The implementation of this project somewhat follows the Model-View-Controller design pattern. It was chosen because it is known as one of the most appropriate design patterns for desktop applications with graphical user interface. In the Risk Analysis tool, the *model* is represented by the entity classes, the security administrator interacts with the system via its GUI, which is the *view*. The *controller* will then use the security administrator input and perform the required operations that will result in changes in the

model. However, as a result of using Swing, it is difficult to split the view and controller, as they require tight coupling [17]. Hence, the implementation is based on MVC, but the view and controller components are collapsed into one entity. *Figure 12* shows the structure of the MVC pattern used. The “View/Controller” component represents the combined View and Controller. The user interacts with the *View*, which then executes the *Controller* events. The controller then updates the *Model*.

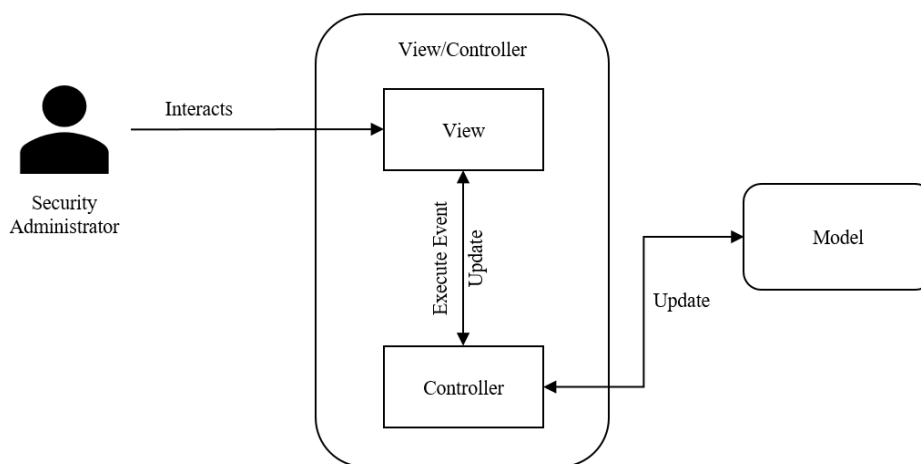


Figure 12. MVC design pattern.

Methodology

As explained in Chapter 3, the tool computes each node’s impact score, exploitability score and risk values based on specific equations. The calculations of a node’s impact score and exploitability score depend on the list of vulnerabilities of that node. Algorithm 1 explains how the vulnerabilities are extracted from the XML file. The first step in this algorithm is to parse the XML file to a Document object. Each vulnerability

in the XML file is called “*entry*”. After parsing, the tool loops through each entry to get its information. The extracted information from each vulnerability are the vulnerability’s name, impact score, exploitability score, CVSS vector, affected operating systems, and all the versions of the affected operating systems. As mentioned in Chapter 3, the CVSS vector is mapped to STRIDE. Each vulnerability is then inserted to the local Vulnerabilities Database.

Algorithm 1 Extracting and Inserting Vulnerabilities to Database

extractVulnerabilities(xmlFile, insertStatement)

Input: An XML File containing vulnerabilities details and an SQL *PreparedStatement* to insert the vulnerability in the database

Output:

```
01: document = parse(XML File)
02: for each entry in document do
03:   name = entry.getName()
04:   impactScore = entry.getImpactScore()
05:   exploitabilityScore = entry.getExploitabilityScore()
06:   CVSS_vector = entry.getCVSS_vector()
07:   STRIDE = mapToSTRIDE(CVSS_vector)
08:   operatingSystems = entry.getOperatingSystems()
09:   for each operatingSystem in operatingSystems do
10:     OSVersions = operatingSystem.getVersions()
11:     for each version in OSVersions do
12:       insertStatement(name, impactScore, exploitabilityScore, STRIDE,
        operatingSystem, version)
13:     end for
14:   end for
15: end for
```

When the security administrator draws a virtual machine in the drawing canvas, a form collecting the information of the virtual machine is displayed. Algorithm 2 represents adding a virtual machine, where the virtual machine is created by retrieving the information entered in the form. The list of vulnerabilities is then retrieved depending on the operating system and its version, and used to calculate the impact and probability. After creating the virtual machine, it is added to the list that holds all the system's virtual machines.

Algorithm 2 Creating a Virtual Machine

drawVM(point)

Input: A Point object that contains the x and y coordinates of the virtual machine to be drawn

Output:

01: showVMinfoForm()

02: user fills form with name, operatingSystem, OSVersion, and STRIDE impact weights

03: RetrievedVulnerabilities = retrieveVulnerabilities(operatingSystem, OSVersion)

04: vm.vulnerabilities = RetrievedVulnerabilities

05: impact = calculateVMImpact()

06: probability = calculateVMProbability()

07: vm = VM(point, name, operatingSystem, OSVersion, STRIDE, impact, probability)

08: vmList.add(vm)

After retrieving the vulnerabilities of the virtual machine, the impact and probability of the virtual machine are calculated according to Equations (1), and (2), respectively. Algorithms 3 and 4 show the details of calculating these two variables.

Algorithm 3 Calculate Impact

calculateImpact(vm)

Input: A Node object

Output:

```
01: impact = 0;
02: for each vulnerability in vm.getVulnerabilities() do
03:   impact += vulnerability.getImpact()
04: end for
05: vm.setImpact(impact)
06: return impact
```

Algorithm 4 Calculate Probability

calculateProbability (vm)

Input: A Node object

Output:

```
01: product = 1;
02: for each vulnerability in vm.getVulnerabilities() do
03:   product *= (1 - vulnerability.getProbability())
04: end for
05: vm.setProbability(1-product)
06: return probability
```

Creating an attacker is simply creating a node that has only an ID and a name, other values are set to null.

After drawing the virtual machines and the attackers, the security administrator must connect them to create the final topology. Algorithm 5 shows in detail how the connection is formed. The source and destination nodes are captured from the mouse clicks. An Edge object, *connection*, is then created with the retrieved nodes as source and destination. To

link the nodes to each other, the connection is added to the source's outgoing edges list, and to the destinations' incoming edges list.

Algorithm 5 Connecting Two Nodes

connectNodes(source, destination)

Input: Two Node object representing the source and the destination of the Edge

Output:

```
01: source = getNodeFromClick()
02: destination = getNodeFromClick()
03: connection = Edge(source, destination)
04: source.getOutgoingEdges().add(connection);
05: destination.getIncomingEdges().add(connection);
06: connectionsList.add(connection)
```

After completing the drawing of the topology, the security administrator starts the evaluation. The evaluation takes place in four steps. The first step is calculating the risk of each threat of each node. This is further explained in Algorithm 6. The input of this algorithm is the node and threat (out of the STRIDE threats) to be evaluated. An iterator loops through all the vulnerabilities of the input virtual machine and verifies if each of the vulnerabilities might cause the threat defined as input. If the vulnerability does cause the threat, then the threat risk value will be calculated by multiplying the threat weight (entered by user), the virtual machine impact, and the virtual machine probability.

Algorithm 6 Calculate Node Threat Risk

calculateNodeThreatRisk(vm, threat)

Input: The Node to be evaluated and a Character specifying which threat to calculate its risk

Output: Threat risk value

```
01: for each vulnerability in vm do
02:   if vulnerability.STRIDE contains threat then
03:     nodeThreatRisk = threat weight × VM Probability × VM Impact
04:   end if
05: end for
06: return nodeThreatRisk
```

The output of Algorithm 6, *nodeThreatRisk*, is used in calculating the total risk of the virtual machine. Algorithm 7 gives the steps to how it is done. The total risk of the virtual machine is calculated by adding the risk values of all its threats.

Algorithm 7 Calculate Node Total Risk

calculateNodeRisk(vm)

Input: The Node to be evaluated

Output: Node risk value

```
01: nodeRisk = 0;
02: for each threat in STRIDE do
03:   nodeRisk += calculateNodeThreatRisk(vm, threat)
04: end for
05: return nodeRisk
```

Similar to Algorithm 7, Algorithm 8 uses the output of Algorithm 6, *nodeThreatRisk*, in calculating a threat risk. Algorithm 8 takes as input a character specifying with threat risk to calculate (out of the STRIDE Threats). The output is the summation of the specified threat risk of all the nodes.

Algorithm 8 Calculate Threat Risk

calculateThreatRisk(threat)

Input: A character specifying with threat risk to calculate

Output: Threat risk value

01: threatRisk = 0;

02: nodes; // list of all virtual machines of the system

03: **for each** *node* in *nodes* **do**

04: threatRisk += *calculateNodeThreatRisk(node, threat)*

05: **end for**

06: **return** threatRisk

Lastly, the system state risk defined as the summation of risk values of all of the virtual machines of the system. This is illustrated in Algorithm 9.

Algorithm 9 Calculate State Risk

calculateStateRisk(nodes)

Input: A List of Nodes representing all the virtual machines of the system.

Output: State risk value

01: stateRisk = 0;

02: **for each** *node* in *nodes* **do**

03: stateRisk += *calculateNodeRisk(node)*

04: **end for**

05: **return** stateRisk

User Interface Design

This section includes screenshots of all system components that shows how each use case work. *Figure 13* below shows the first screen that appears when running the tool. The white area is the drawing canvas where the security administrator draws the virtual machines, attackers, and edges.

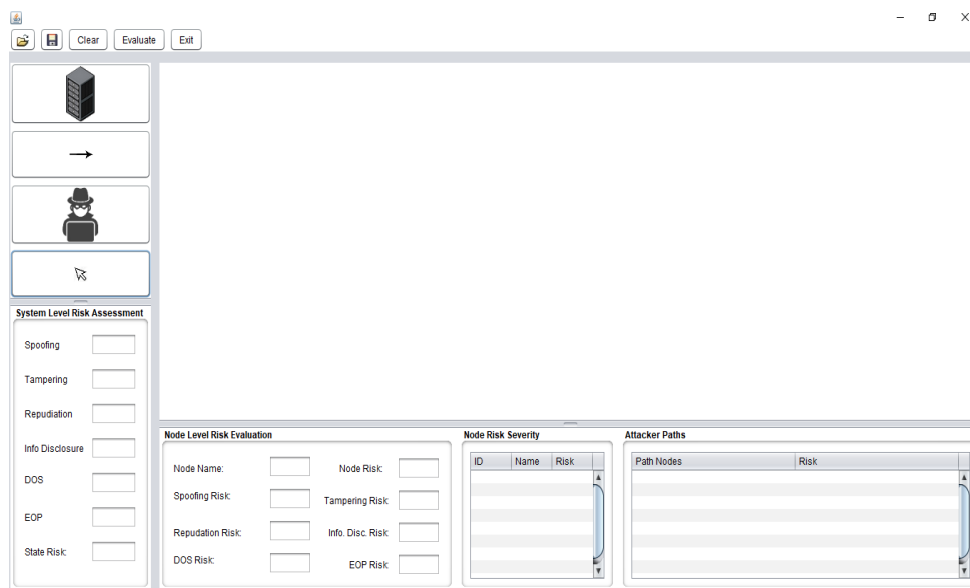


Figure 13. Initial screen of the system.

Figure 14 shows the buttons panel. The first button is used to draw a virtual machine. When the user clicks this button and clicks on any point on the white canvas, a virtual machine will be drawn and a form will appear on the screen asking to fill the VM's properties. After drawing at least two nodes (two virtual machines or one virtual machine and one attacker) the second button is used to connect the nodes. The user will have to click on the source node first, and then click on the destination node. The third button is used to draw an

attacker. Similar to the first button, the user clicks this button and then click any point on the canvas. This will create an “attacker” node, with an ID and a name only, and all other information are null values. Lastly, the cursor button is used to put the mouse in the “cursor” state, so the user can click on a node and get its information.

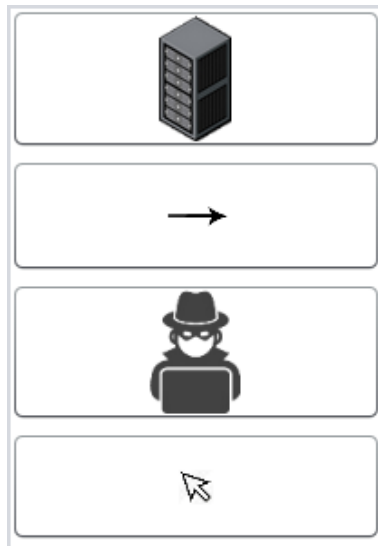


Figure 14. Close-up of buttons panel.

The System Level Risk Assessment panel in *Figure 15* shows different text fields for the different STRIDE components. When the topology is evaluated, the threat risk values and the state risk value will appear in the corresponding text fields which are not editable.

System Level Risk Assessment

Spoofing	<input type="text"/>
Repudiation	<input type="text"/>
Tampering	<input type="text"/>
Info Disclosure	<input type="text"/>
DOS	<input type="text"/>
EOP	<input type="text"/>
State Risk:	<input type="text"/>

Figure 15. System Assessment Panel.

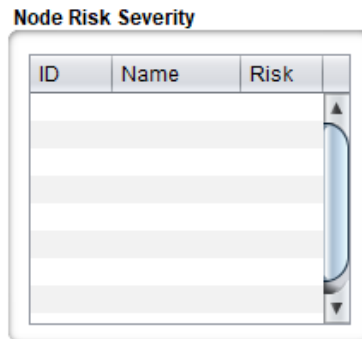
The next panel as shown in *Figure 16* is the Node Level Risk Evaluation panel. It has text fields for the node information, Node Name, Node Risk, Spoofing Risk, Tampering Risk, Repudiation Risk, Information Disclosure Risk, Denial of Service Risk, and Elevation of Privileges Risk. Similarly, these text field are not editable. After the evaluation is completed, and when the user clicks on a virtual machine, the text fields will have the mentioned values.

Node Level Risk Evaluation

Node Name:	<input type="text"/>	Node Risk:	<input type="text"/>
Spoofing Risk:	<input type="text"/>	Tampering Risk:	<input type="text"/>
Repudiation Risk:	<input type="text"/>	Info. Disc. Risk:	<input type="text"/>
DOS Risk:	<input type="text"/>	EOP Risk:	<input type="text"/>

Figure 16. Node Risk Evaluation Panel.

Figure 17 shows Node Risk Severity. This panel holds a table of all the nodes and their risk values, sorted descending such that the riskiest virtual machine is at the top of the list.

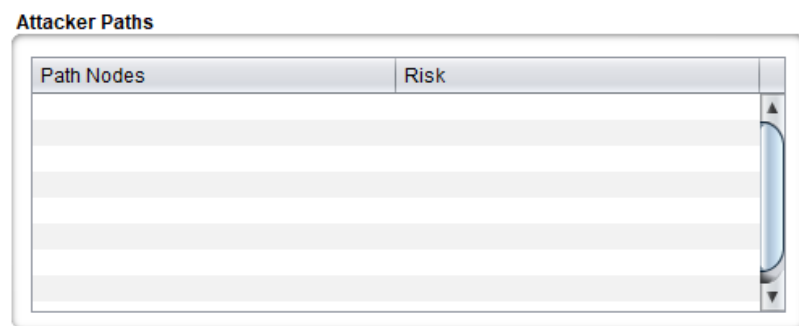


The image shows a window titled "Node Risk Severity" containing a table with three columns: "ID", "Name", and "Risk". The table has five empty rows below the header. A vertical scrollbar is visible on the right side of the table.

ID	Name	Risk

Figure 17. Node Risk Severity Table.

The last panel shown in Figure 18 is the Attacker Paths panel. In this panel, when the user clicks on a virtual machine. The paths leading the attacker to this virtual machine are shown in this table. Each path has a risk value, that is the summation of the risk values of the nodes constructing the path.



The image shows a window titled "Attacker Paths" containing a table with two columns: "Path Nodes" and "Risk". The table has five empty rows below the header. A vertical scrollbar is visible on the right side of the table.

Path Nodes	Risk

Figure 18. Attacker Paths Table.

When the user clicks on the screen to draw a virtual machine, a form will appear to be filled with the virtual machine properties, as shown in *Figure 19*. A clearer shot of the form is shown in *Figure 20*.

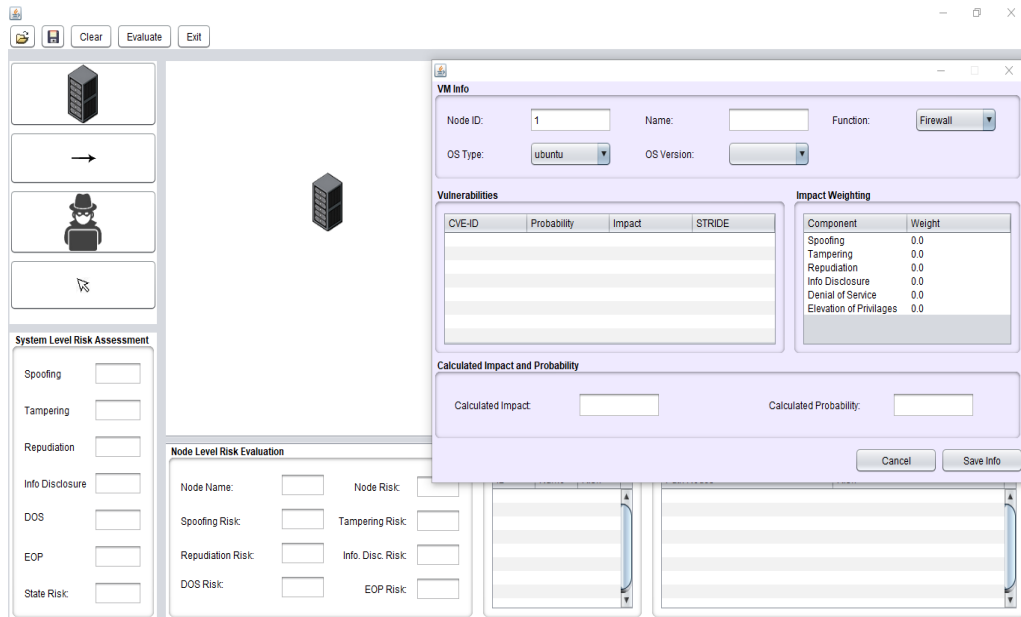


Figure 19: screenshot of drawing node with form

Figure 20 shows the VM information form, it consists of five panels. The first panel is for the VM basic information, which include ID (auto-generated), Name, Function (firewall, host...etc.) Operating System Type, OS Version. The Vulnerabilities panel has a table of the vulnerabilities retrieved from the database, depending on the operating system and its version. In the Impact Weighting Panel, the security administrator must enter the STRIDE values that their organization prefers. The summation of these values must be exactly 1. The last panel is the Calculated Impact and Probability panel. This panel shows the

calculated impact of the virtual machine and the calculated probability, according to its list of vulnerabilities. The “Cancel” button will cancel the node drawing operation, and the “Save Info” button will store the virtual machine details.

The screenshot shows a window titled "VM Info" with the following sections:

- VM Info:** Node ID: 1, Name: (empty), Function: Firewall, OS Type: ubuntu, OS Version: (empty).
- Vulnerabilities:** A table with columns CVE-ID, Probability, Impact, and STRIDE. It contains five empty rows.
- Impact Weighting:** A table with columns Component and Weight. It lists Spoofing, Tampering, Repudiation, Info Disclosure, Denial of Service, and Elevation of Privileges, all with a weight of 0.0.
- Calculated Impact and Probability:** Calculated Impact: (empty), Calculated Probability: (empty).
- Buttons:** Cancel and Save Info.

Figure 20. Close-up shot of VM properties form.

The security administrator should then enter the information of the virtual machine. When the operating system and OS version are selected, the list of vulnerabilities will be updated depending on the selected values, as shown in *Figure 21*. The impact and probability will be calculated upon the selection of operating system and its version, because they are calculated based on the retrieved vulnerabilities.

VM Info

Node ID: Name: Function:

OS Type: OS Version:

Vulnerabilities

CVE-ID	Probability	Impact	STRIDE
CVE-2018-0742	6.4	0.39	TID
CVE-2018-0744	6.4	0.34	TID
CVE-2018-0746	2.9	0.34	I
CVE-2018-0747	2.9	0.34	I
CVE-2018-0748	6.4	0.39	TID
CVE-2018-0749	6.4	0.39	TID
CVE-2018-0751	4.9	0.39	TI
CVE-2018-0752	6.4	0.39	TID

Impact Weighting

Component	Weight
Spoofing	0.0
Tampering	0.2
Repudiation	0.0
Info Disclosure	0.3
Denial of Service	0.5
Elevation of Privileges	0.0

Calculated Impact and Probability

Calculated Impact: Calculated Probability:

Figure 21. Screenshot of filled form.

Figure 22 shows 3 virtual machines which are a Firewall, Database, and a Web Server and an attacker. After clicking the “Evaluation” button, the text fields of the System Level Risk Assessment panel will be filled with the corresponding values. When the user clicks on one of the virtual machines, its information is shown in the appropriate fields in the Node Level Risk Assessment panel. The Attack Path table is also updated to show the different paths from the attacker to the clicked node. It also shows the Risk value of the path, which is the summation of the paths’ nodes’ risk values.

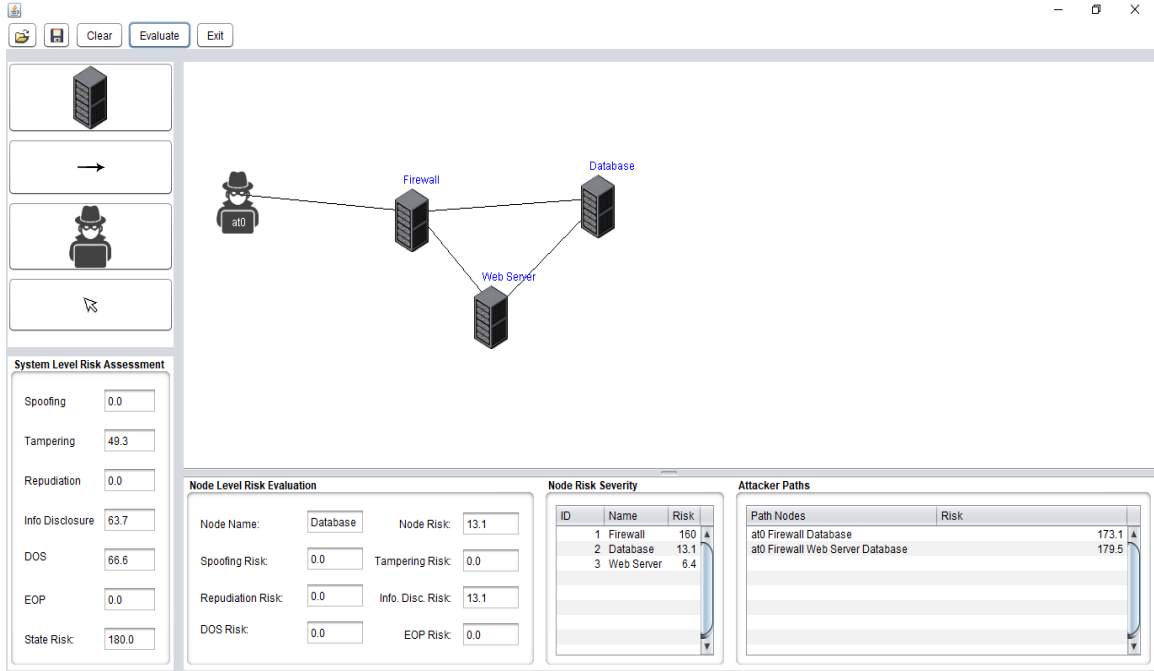


Figure 22. Screenshot of drawn topology.

The tool can also allow the user to open an existing topology. Figure 23 shows a screenshot of opening an existing topology.

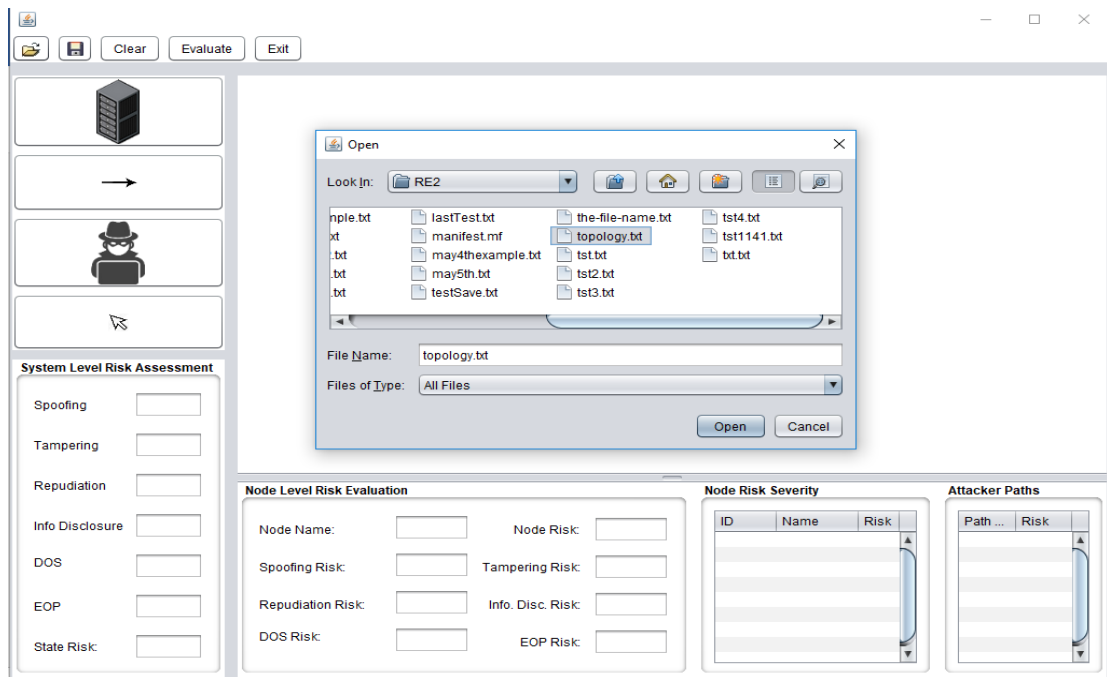


Figure 23. Opening existing topology screenshot.

The next chapter presents the evaluation of our risk evaluation tool.

CHAPTER 6: EVALUATION

This chapter focuses on the evaluation of two quality attributes, usability and scalability. The chapter includes a usability study and an experimental analysis.

Usability Study

To evaluate the tool's usability, three cyber security researchers were asked to test tool. Due to the time limit, we were unable to reach actual security administrators to evaluate the tool, hence our choice of security researchers. The evaluators were asked to draw a topology with specific features. The Usability Questionnaire can be found in Appendix F. The questionnaire studied five aspects; the difficulty of creating a new virtual machine the difficulty of connecting two nodes, the difficulty of evaluating the network topology, the organization of the user interface, and how beneficial the tool is to security administrators. The rationale behind the first three questions come from the point that the proposed tool takes as input the network topology. So, it is of a great importance to ensure that drawing the topology nodes and connecting them is easy. In addition, the user interface is similarly significant, because this is the only method of interaction. Lastly, the evaluators were asked about their opinions of how beneficial the tool is for security administrators, as the security administrators are the main users of the tool. *Figure 24* illustrates the usability scores for the study questions. The usability score was calculated by taking the average scores for each question.

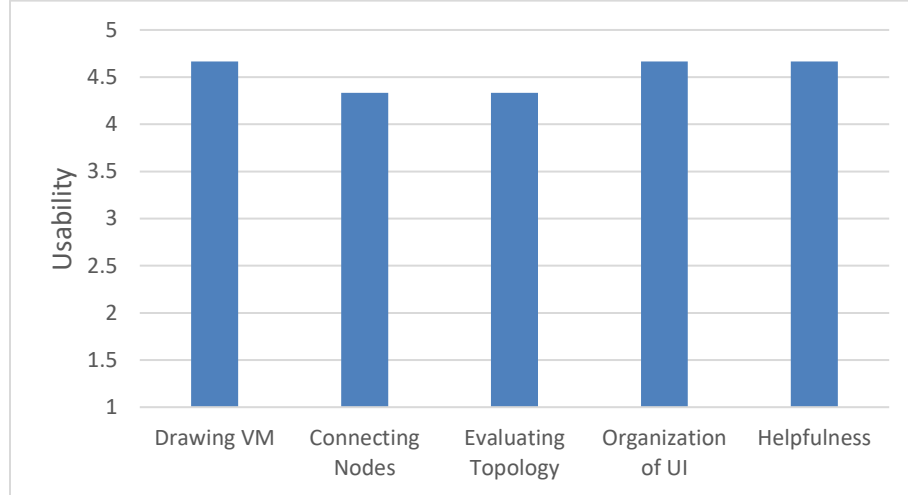


Figure 24. Usability scores for the study questions.

The simplicity of Drawing a VM is of a score of 4.667 out of 5. This shows that creating a new virtual machine and setting its properties is straightforward. Connecting Nodes and Evaluating Topology both scored 4.333 on the scale. This is mainly because the edges connecting the nodes do not have arrows. The evaluators agreed that the lack of directional arrows causes confusion. User Interface Organization of the tool has a score of 4.667 out of 5. This indicates that the tool is highly user friendly. Lastly, the Helpfulness of the tool scored 4.667, which shows a real need for such a tool in the field.

Experiment Analysis

Experiment Setup

The experiment analysis was conducted on an HP laptop running Windows 10. The machine has an Intel(R) Core i7-6500U processor of speed 2.60 GHz, and an 8GB RAM. The experiment was done by plotting the number of virtual machines in the topology against the time to evaluate the topology. The next section presents the result in detail.

Experiments Results

To perform this experiment, four sets of topologies were evaluated; A, B, C, D. The sets consist of 3, 6, 12, and 20 virtual machines, respectively. To obtain more accurate results, three topologies were evaluated for each set. The figures below illustrate the topologies used for the experiment. *Figure 25.a, b, and c* show three 3-node topologies connected with 3, 4 and 5 edges, respectively.

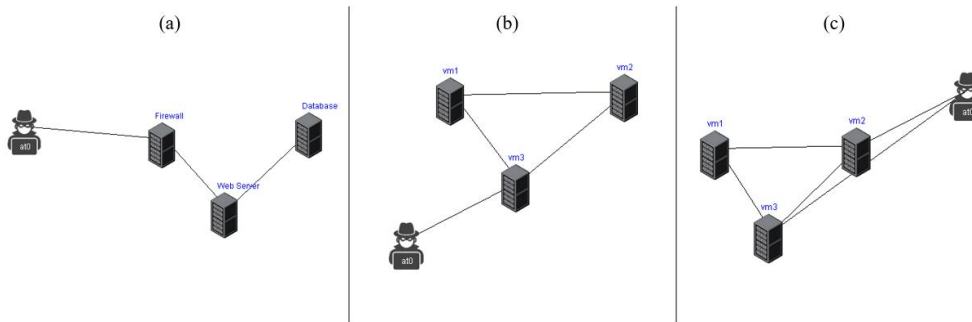


Figure 25. Topologies of set A.

Figure 26.a, b, and c show three 6-node topologies connected with 9, 12 and 15 edges, respectively.

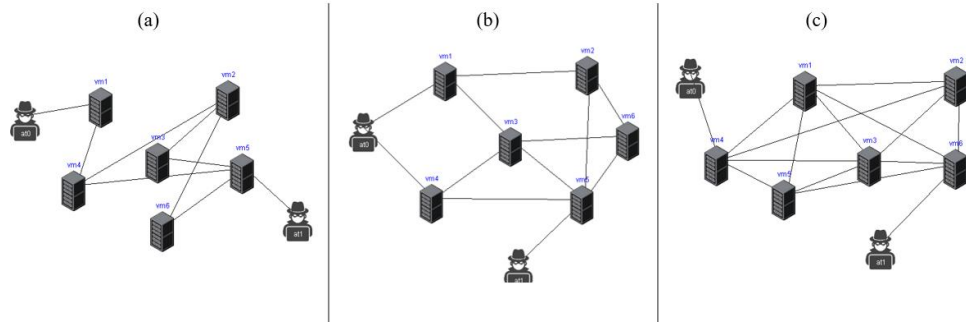


Figure 26. Topologies of set B.

Figure 27.a, b, and c show three 12-node topologies connected with 18, 21 and 25 edges, respectively.

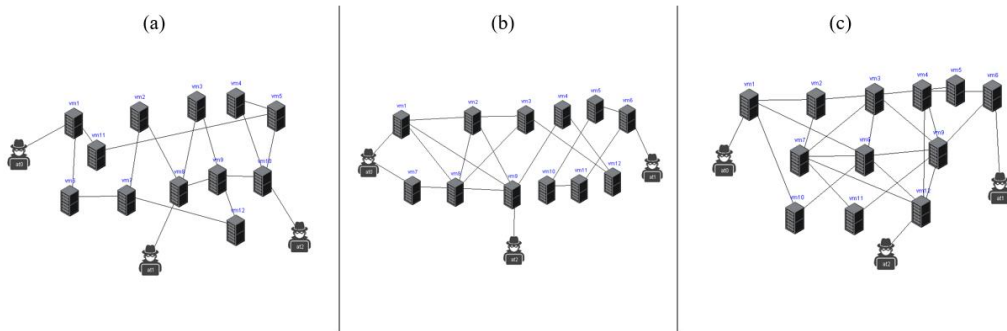


Figure 27. Topologies of set C.

Figures 28.a, b, and c show three 20-node topologies connected with 30, 35 and 40 edges, respectively.

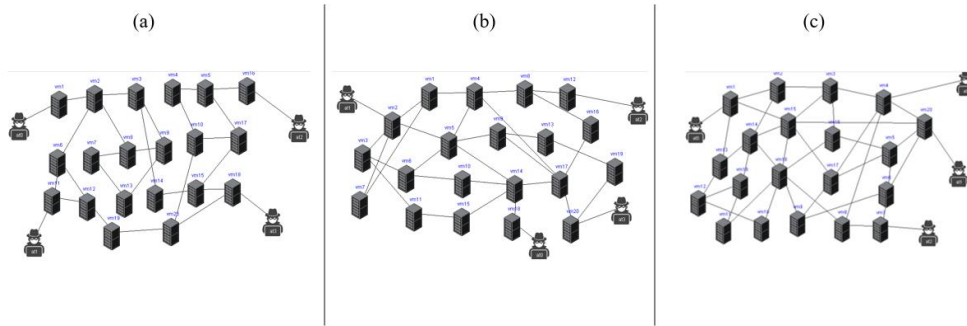


Figure 28. Topologies of set D.

The average evaluation time of each set was calculated and then used in plotting *Figure 29*. The figure illustrates the evaluation time against the number of nodes in the topology.

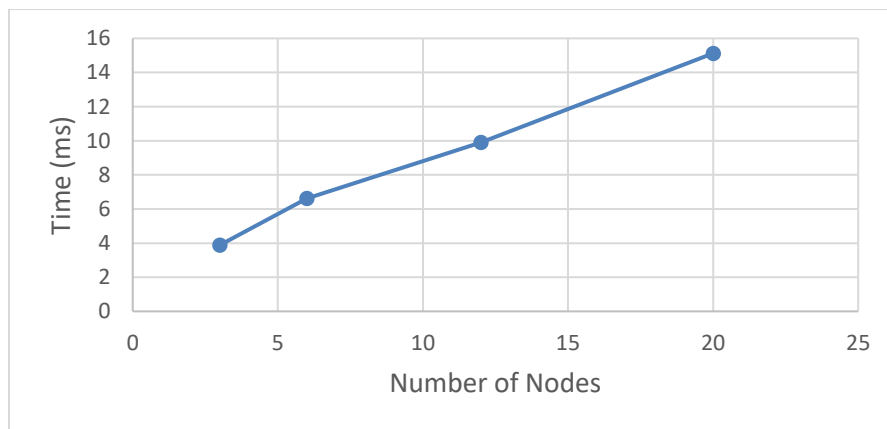


Figure 29. Time Vs. number of nodes.

The evaluation time of the tool is of a linear curve; it increases as the number of nodes in the topology increases. This growth is rational because the tool will take more time processing each node to calculate its risk values.

CHAPTER 7: DISCUSSION

This chapter explores the rationale for the choice of programming language and the method of the vulnerability extractor. Moreover, the chapter discusses the limitations of this version of the risk evaluation tool.

In this project, we propose a risk analysis tool that uses the STRIDE threat model in categorizing the different types of threats. The tool will help security administrators understand the risk of the specific threats posed on their assets. As a result, security administrator will be able to accurately predict and identify the VMs causing higher risk to the organization, and then address it properly by choosing the appropriate mitigation strategy.

The programming language used to implement our risk analysis tool is Java, and it is chosen for several reasons. First, Java is significantly more powerful than many other languages in terms of portability, it is known to be cross-platform. Another justification for using Java is its libraries. Languages such as JavaScript and Python are similarly powerful, but for our tool where the focus is on the graphical user interface, Java is much richer with libraries for that purpose that makes it more suitable.

Another critical decision of our tool was how to extract vulnerabilities. One option was retrieving the vulnerabilities from the National Vulnerability Database using the tool provided in [18], which is a tool used to perform local searches for known vulnerabilities. However, the database created from this tool lacks important attributes that we are interested in. The vulnerabilities table created has attributes such as *Vulnerability ID*, *CVSS*, *Impact Vector*, but it lacks significant attributes that our calculations are based on,

like *Impact Score* and *Exploitability Score*.

Hence, we decided upon the vulnerabilities downloader provided in [15] because it extracts all the vulnerability's details in an XML file. Then, the Java SQL Driver takes the attributes of interest and creates a local database for all the vulnerabilities.

Limitations

Due to the time constraint, this version of the risk evaluation tool has some limitations. The first limitation of this tool is that it does not print the attack paths properly in case of multiple attackers. Another limitation is the lack of directional arrows between the nodes when drawing the topology. This results in confusion and possibility of misunderstanding the topology. Moreover, the drawn items (i.e. nodes and edges) are not movable, once the user clicks on a point to draw a node, it is fixed there. The feature of moving the drawn items would make the tool more usable. Lastly, this version of the tool does not retrieve the vulnerabilities in real-time. To get updated list of vulnerabilities, we will have to run the Vulnerability Downloader to download the XML files of the most recent vulnerabilities. A real-time vulnerability scanner that runs automatically when starting the tool would be much more convenient for this tool. Hence integrating its risk evaluation features with real-time vulnerability scanner could enrich the tool and the practicality of deploying it in a real-world risk assessment environment.

CHAPTER 8: CONCLUSION

In the scope of this project, a threat-specific risk evaluation tool was developed. The developed tool varies from the existing tools in several aspects. First, it gives a numeric representation of the evaluated risk for the entire system, and for each virtual machine. This will allow the security administrator to easily identify the assets at risk. Second, the tool's evaluation provides the security administrator with the degree of severity for each of the STRIDE threat, for the system as a whole, and for each virtual machine. The usability study results reveal that the proposed tool is highly user friendly, and there is a need for such a tool. In addition, the experimental analysis of the tool gives satisfactory results for the scalability.

The current risk evaluation tool can be improved in several ways. First, addressing the limitations relevant to the user interface will greatly enhance the usability of the tool. Second, the implementation of a real-time vulnerability scanner will provide the tool with up-to-date vulnerabilities database. As a result, this will deliver more accurate risk values to the security administrators for their networks. Moreover, the tool can be improved by giving the security administrator the option of evaluating a subsystem within the organization's network topology. Selecting the nodes of the subsystem from a drawn topology will save the security administrators' time and efforts of drawing a different topology for each subsystem.

REFERENCES

1. Mell, P. M., & Grance, T. (2011). The NIST Definition of Cloud Computing. doi:10.6028/nist.sp.800-145
2. Curran, K. (2013). *Pervasive and ubiquitous technology innovations for ambient intelligence environments*. Hershey, PA: Information Science Reference.
3. Gariba, Z. P., & Poll, J. A. (2017). Security Failure Trends of Cloud Computing. *2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC)*. doi:10.1109/cic.2017.00041
4. Saripalli, P., & Walters, B. (2010). QUIRC: A Quantitative Impact and Risk Assessment Framework for Cloud Security. *2010 IEEE 3rd International Conference on Cloud Computing*. doi:10.1109/cloud.2010.22
5. Efozia, N. F., Ariwa, E., Asogwa, D. C., Awonusi, O., & Anigbogu, S. O. (2017). A review of threats and vulnerabilities to cloud computing existence. *2017 Seventh International Conference on Innovative Computing Technology (INTECH)*. doi:10.1109/intech.2017.8102448
6. Application Threat Modelling. (n.d.). Retrieved January 21, 2018, from https://www.owasp.org/index.php/Application_Threat_Modeling
7. The STRIDE Threat Model. (n.d.). Retrieved February 16, 2018, from [https://msdn.microsoft.com/en-us/library/ee823878\(v=cs.20\).aspx](https://msdn.microsoft.com/en-us/library/ee823878(v=cs.20).aspx)
8. (Cybersecurity), H. G. (n.d.). STRIDE, CIA and the Modern Adversary. Retrieved January 21, 2018, from

<https://blogs.msdn.microsoft.com/heinrichg/2016/06/07/stride-cia-and-the-modern-adversary/>

9. Kazim, M., & Evans, D. (2016). Threat Modelling for Services in Cloud. 2016 IEEE Symposium on Service-Oriented System Engineering (SOSE).
doi:10.1109/sose.2016.55
10. Sommestad, T., Ekstedt, M., & Holm, H. (2013). The Cyber Security Modeling Language: A Tool for Assessing the Vulnerability of Enterprise System Architectures. IEEE Systems Journal, 7(3), 363-373.
doi:10.1109/jsyst.2012.2221853
11. Djemame, K., Armstrong, D., Guitart, J., & Macias, M. (2016). A Risk Assessment Framework for Cloud Computing. IEEE Transactions on Cloud Computing, 4(3), 265-278. doi:10.1109/tcc.2014.2344653
12. Santos R. (2017). Microsoft Threat Modeling Tool - Azure. Retrieved from <https://docs.microsoft.com/en-us/azure/security/azure-security-threat-modeling-tool>
13. Nhlabatsi, A., Hong, J., Kim, D., Fernandez, R., Hussein, A., Fetais, N., Khan, K., (2018). Threat-specific Security Risk Evaluation in the Cloud. Submitted to IEEE Transactions on Cloud Computing.
14. NIST - National Vulnerability Database. (n.d.). Retrieved from <https://nvd.nist.gov/>
15. Galsterer, C. NVD Download Script. Retrieved from <https://gist.github.com/christiangalsterer/5f55389b9c50c74c31b9>

16. Swing. (n.d.). Retrieved from

<https://docs.oracle.com/javase/8/docs/technotes/guides/swing/>

17. Fowler, A. (n.d.). A Swing Architecture Overview. Retrieved from

<http://www.oracle.com/technetwork/java/architecture-142923.html>

18. Dulaunoy, A., et. al. CVE-Search. Retrieved from

<https://gist.github.com/christiangelsterer/5f55389b9c50c74c31b9>

APPENDIX A: DRAW NODE USE CASE SPECIFICATION

Name	Draw Node
Actor	Security Administrator
Description	Allows the security administrator to draw a virtual machine or an attacker on a particular point on the drawing canvas, and enter the node information in case it is a virtual machine
Pre-conditions	None
Post-conditions	The security administrator successfully creates a node (VM/attacker)
Basic flow	<p>When the security administrator clicks on the draw node button and then clicks on the canvas, a node icon will appear on the clicked point of the canvas. If the node is a VM, a form will pop up for the security administrator to fill the information of the virtual machine which are:</p> <ol style="list-style-type: none"> 1. Name 2. Functionality 3. Operating System 4. OS Version <p>With the changes of the <i>Operating System</i> and <i>OS Version</i> values, the table of vulnerabilities will change dynamically. When the security administrator clicks the button “Save Info”, the node information will be updated with the entered values.</p>

	<p>If the node is an attacker, no form will appear and the node information will be stored as null values, except for the attacker ID and attacker name.</p>
--	--

APPENDIX B: DRAW EDGE USE CASE SPECIFICATIONS

Name	Draw Edge
Actor	Security Administrator
Description	This use case links two virtual machines to each other, or links an attacker to a virtual machine.
Pre-conditions	At least two virtual machines or one virtual machine and an attacker are drawn.
Post-conditions	Two virtual machines or one virtual machine and an attacker are successfully connected with an edge.
Basic Flow	after the security administrator draws at least two virtual machines or one virtual machine and an attacker, the button “Draw Edge” is clicked to link the two nodes together. The security administrator should click on the source node first, and then click on the destination node.

APPENDIX C: EVALUATE TOPOLOGY USE CASE SPECIFICATIONS

Name	Evaluate Topology
Actor	Security Administrator
Description	This use case performs all the risk calculations.
Pre-conditions	A system's topology exists.
Post-conditions	The risk of the system is evaluated and the different risk values are shown to the security administrator.
Basic Flow	When the construction of the topology is completed, and the security administrator clicks on "Evaluate" button, the calculations to compute the system state risk, threat risks (STRIDE), each node risk, and each node threat risks (STRIDE).

APPENDIX D: OPEN TOPOLOGY USE CASE SPECIFICATION

Name	Open Topology
Actor	Security Administrator
Description	This use case allows the user to open an existing topology.
Pre-conditions	A system's topology exists.
Post-conditions	The existing topology is
Basic Flow	After constructing and evaluating a topology, the user clicks on the desired VM to view its attack paths. A table on the tool window will be updated with the paths to the selected VM, along with the risk value of the path.

APPENDIX E: SHOW ATTACK PATHS USE CASE SPECIFICATIONS

Name	Show Attack Paths
Actor	Security Administrator
Description	This use case prints to the user the attack paths of the selected virtual machine.
Pre-conditions	A system's topology exists.
Post-conditions	The attack paths of a virtual machine are shown to the user, with their risk values.
Basic Flow	After constructing and evaluating a topology, the user clicks on the desired VM to view its attack paths. A table on the tool window will be updated with the paths to the selected VM, along with the risk value of the path.


APPENDIX F: USABILITY QUESTIONNAIRE

Thank you for testing of my Threat-Specific Risk Evaluation Tool – a tool developed as part of my Masters project. The main objective of the tool is supporting Security Administrators in evaluating risk with respect to specific threats in network-based information systems in their organizations. Your feedback will help me in evaluating and improving the usability of the tool.

Using the Tool

The tool provides a clear canvas for you to create a network topology by drawing collection of virtual machines and the possible attackers, then connecting them via edges. To calculate the risk of a virtual machine, we first need to calculate its impact and exploitability values. These values are calculated from the virtual machine's vulnerabilities list. The three key steps in using the tool are addition of nodes, reachability edges, and risk evaluation. These are explained in detail below.

1. Adding Virtual Machine Nodes

To add a virtual machine, click on the virtual machine button. 

When you draw a virtual machine, a form will pop up asking you to provide the virtual machine details, which are:

1. Name
2. Function
3. Operating System

4. OS Version

VM Info

Node ID: Name: Function:

OS Type: OS Version:

Vulnerabilities

CVE-ID	Probability	Impact	STRIDE

Impact Weighting

Component	Weight
Spoofing	0.0
Tampering	0.0
Repudiation	0.0
Info Disclosure	0.0
Denial of Service	0.0
Elevation of Privileges	0.0

Calculated Impact and Probability

Calculated Impact: Calculated Probability:

When you select the operating system and its version:

OS Type: OS Version:

the vulnerabilities of the selected operating system will be shown:

Vulnerabilities

CVE-ID	Probability	Impact	STRIDE
CVE-2018-0742	6.4	0.39	TID
CVE-2018-0744	6.4	0.34	TID
CVE-2018-0746	2.9	0.34	I
CVE-2018-0747	2.9	0.34	I
CVE-2018-0748	6.4	0.39	TID
CVE-2018-0749	6.4	0.39	TID
CVE-2018-0751	4.9	0.39	TI
CVE-2018-0752	6.4	0.39	TID

and the impact and the probability of the virtual machine are calculated accordingly:

Calculated Impact and Probability

Calculated Impact:	<input type="text" value="147.0"/>	Calculated Probability:	<input type="text" value="1.0"/>
--------------------	------------------------------------	-------------------------	----------------------------------

Next, you'll have to enter the STRIDE Impact Weights.

Impact Weighting

Component	Weight
Spoofing	0.0
Tampering	0.0
Repudiation	0.0
Info Disclosure	0.0
Denial of Service	0.0
Elevation of Privileges	0.0

Give more weight to the components you are concerned about¹. For example, if you value the *Confidentiality* of your system more than you evaluate the *Availability*, give more weight to *Information Disclosure*. Or if you run an online newspaper agency, and you may care more about the *Integrity* of news articles than their disclosure and in that case, you give more weighting to *Tampering*, and so on. Make sure that the STRIDE Weighting values add up to 1.

¹ The weighting depends on security requirements.

Impact Weighting	
Component	Weight
Spoofing	0.1
Tampering	0.3
Repudiation	0.0
Info Disclosure	0.2
Denial of Service	0.4
Elevation of Privileges	0.0

After entering the desired impact weighting, pressing the “Down Arrow ↓” key in your keyboard to exit editing mode.

After filling the form, save the virtual machine information by clicking the “Save Info” button:



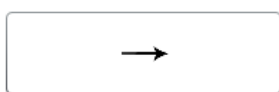
When you finish drawing your virtual machines, you can add an “Attacker” by pressing the “Attacker” button:



Then, click on where you want to place it in the canvas.

2. Adding Reachability Edges

After placing all the virtual machines and the attacker on the canvas, click on the “Arrow” button to connect nodes:




To create a connection, click the source node first, then the destination node. A solid line

will appear indicating that a connection has been established between the nodes. The connection indicates that the destination node is reachable from the source node.

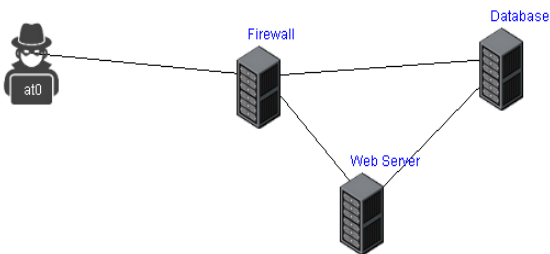
3. Risk Evaluation

After you complete drawing the topology, you can start the evaluation by pressing “Evaluate” button at the top of the screen:



All the risk values of the system will be calculated and displayed. To view the risk details for each virtual machine, click the “Cursor  ” button and select the node you want to view.

To evaluate the tool, create the following topology with the provided data. Then, answer the evaluation questions.



Firewall Node Information	
Name	Firewall
Function	Firewall

OS Type	Ubuntu
OS Version	16.04
Spoofing	0.0
Tampering	0.1
Repudiation	0.3
Information Disclosure	0.0
Denial of Service	0.4
Elevation of Privileges	0.2

Database Node Information	
Name	Database
Function	Host
OS Type	Windows_10
OS Version	1511
Spoofing	0.1
Tampering	0.4
Repudiation	0.0
Information Disclosure	0.2
Denial of Service	0.1
Elevation of Privileges	0.2

Web Server Node Information

Name	Web Server
Function	Host
OS Type	Windows_8.1
OS Version	“”
Spoofing	0.15
Tampering	0.1
Repudiation	0.05
Information Disclosure	0.2
Denial of Service	0.4
Elevation of Privileges	0.1

Evaluation Questions:

Please answer the following questions based on your experience of using the tool. On a scale from 1 to 5:

1. How would you describe the difficulty of creating a new virtual machine? [1 to 5 (1: “very difficult”, 5: “very easy”)]
2. How would you describe the difficulty of connecting two nodes? [1 to 5 (1: “very difficult”, 5: “very easy”)]
3. How would you describe the difficulty of evaluating the network? [1 to 5 (1: “very difficult”, 5: “very easy”)]
4. How did would you describe the organization of the user interface? [1 to 5 (1: “Poorly Organized”, 5: “Well Organized”)]

5. How beneficial would be this tool be to security administrators? [1 to 5 (1: “not useful difficult”, 5: “very useful”)]