A Framework for Secure Structural Adaptation
Abstract

A (self-) adaptive system is a system that can dynamically adapt its behavior or structure during execution to "adapt" to changes to its environment or the system itself. From a security standpoint, there has been some research pertaining to (self-) adaptive systems in general but not enough care has been shown towards the adaptation itself. Security of systems can be reasoned about using threat models to discover security issues in the system. Essentially that entails abstracting away details not relevant to the security of the system in order to focus on the important aspects related to security. Threat models often enable us to reason about the security of a system quantitatively using security metrics. The structural adaptation process of a (self-) adaptive system occurs based on a reconfiguration plan, a set of steps to follow from the initial state (configuration) to the final state. Usually, the reconfiguration plan consists of multiple strategies for the structural adaptation process and each strategy consists of several steps with each step representing a specific configuration of the (self-) adaptive system. Different reconfiguration strategies have different security levels as each strategy consists of a different sequence configuration with different security levels. To the best of our knowledge, there exist no approaches which aim to guide the reconfiguration process in order to select the most secure available reconfiguration strategy, and the explicit security of the issues associated with the structural reconfiguration process itself has not been studied. In this work, based on an in-depth literature survey, we aim to propose several metrics to measure the security of configurations, reconfiguration strategies and reconfiguration plans based on graph-based threat models. Additionally, we have implemented a prototype to demonstrate our approach and automate the process. Finally, we have evaluated our approach based on a case study of our making. The preliminary results tend to expose certain security issues during the structural adaptation process and exhibit the effectiveness of our proposed metrics.

Preface

The domain of this work is not a domain that a student can be taught during undergraduate studies and is very broad including a variety of different disciplines of computer science. In the beginning, I did not have the required background knowledge in this particular domain. As a consequence, I was keenly interested to examine these new areas of knowledge instead of studying subjects that I am already familiar with. Thus, engaging myself with this project has been a true learning experience.

Firstly, I have to express my profound gratitude to my supervisor, Dr. Narges Khakpour, for proposing this topic and her recommendation through the process of researching and authoring the thesis. I am also grateful to Charilaos Skandylas for all his advice, discussions and technical support. A, a quick thank you to my friend Prasannjeet Singh for his constructive criticism and also for proofreading the manuscript. Finally, I would like to thank Mauro Caporuscio, my examiner, for his advice and suggestions on how to improve the quality of the thesis. I also appreciate his commendation for the idea and the outcomes of the thesis.

This research could not have been written without the generous assistance of my friends and family who always encouraged and supported me throughout my authorial journey. To all of you, I extend my deep appreciation.
Contents

1 Introduction ________________________________________________ 5
  1.1. Background______________________________________________ 5
  1.2. Motivation_____________________________________________ 6
  1.3. Problem Statement and Research Questions__________________ 8
  1.4. Method ________________________________________________ 9
  1.5. Contributions and Limitations______________________________ 10
  1.6. Target groups ___________________________________________ 10
  1.7. Report Structure __________________________________________ 10

2 Background ________________________________________________ 11
  2.1. Self-Adaptive Systems____________________________________ 11
  2.2. Threat Modeling and Analysis_______________________________ 14
  2.3. Network Security Metrics (NSMs)____________________________ 26

3. Method_____________________________________________________ 34
  3.1. Scientific Approach_______________________________________ 34
  3.2. Method Description_______________________________________ 34
  3.3. Reliability and Validity____________________________________ 57

4. Implementation ______________________________________________ 59

5. Evaluation __________________________________________________ 65
  5.1. Case Study_______________________________________________ 65
  5.2. Results_________________________________________________ 70
  5.3. Analysis________________________________________________ 72
  5.4. Discussion_______________________________________________ 74

6. Conclusions and Future Work___________________________________ 76

7. References __________________________________________________ 77

Appendix 1 ____________________________________________________ 84
1 Introduction

Systems face an increasing amount of vulnerabilities to attacks as digitalization of most systems occurs. Almost every aspect of our day to day life is now being managed by an information system connected with other information systems, as systems hardly exist in isolation anymore. Consequently, it has never been truer that the security of a system is dependent on the security of its weakest link. Organizations must ensure they can measure and compare the security levels of their systems using qualitative as well as quantitative models and formalizations to protect themselves in the face of security threats.

On the other hand, the growth of enterprise system architecture, the demand for automating all aspects of the interconnected cyber environment, and the need for any system to become (self-) adaptive have made systems more complicated than they have been in the past. The high degree of complexity and automation of an enterprise system requires a high degree of security against an attacker. The attack techniques employed by cybercriminals have grown very rapidly. In addition, the increasing use of user-owned machines in a system and increasing demand for complex functionality and adaptation has left the system vulnerable. Also, the complexity has left a multitude of new entry points for attackers to gain entry into sensitive systems.

Over the last two decades, the science and technologies of evaluating and analyzing the vulnerability of systems have significantly been investigated. The investigations produced various methodologies and tools for vulnerability analysis. Furthermore, improving the security of systems has become vital and inevitable.

1.1. Background

A (Self-) Adaptive System is a system that can dynamically adjust its behavior and/or structure (at run-time) to acclimatize to changes to the system itself or its environment. In principle, there are two foremost adaptation techniques. That is Structural and Behavioral Adaptation. Behavioral adaptation allows for altering the computational entities of the system while structural adaptation allows for changing the system architecture itself. These changes occur based on a plan which the (self-) adaptive system generates to guide the whole process of adaptations, which is called Reconfiguration Plan.

Generally, the Reconfiguration Plan is the collection of decisions, together with the reasonable choices that the developers or administrators must make. It can also be defined as a predefined plan that is capable of supervising the reconfiguration process from the start of execution until its termination. In addition, a mechanism is needed to supervise this process which is called adaptation manager. The adaptation manager is responsible for planning to select a new state of the system during the adaptation process.

Security is a system attribute that indicates the strength of the system to defend itself from outside attacks, which may be accidental or deliberate. These external attacks are reasonable because most general-purpose computers are now networked and are therefore approachable by outsiders. To name a few, some of the attacks might be: the installation of Trojan horses and viruses, unauthorized modification of a system or its data or unauthorized use of system services. An attack can be
defined as an exploitation of a system’s vulnerability. Usually, an attack is from outside the system and is a deliberate effort to induce some damage. Vulnerability is a property of a computer-based system which indicates the weakness that may be exploited by an attacker to cause damage or infliction. For identifying and evaluating the security level of any type of systems, all existing vulnerabilities and their interaction-effect and the combinations of exploits have to be taken into consideration. For this purpose, a security modeling technique might be used.

Security modeling involves using models to uncover the security problems of a system. Models aid by removing unnecessary detail and focusing on the bigger picture. Illustration of these methods can be done in a graphical or a textual way. The Attack Graph is an example of the graphical security modeling.

An attack graph is a brief illustration of all paths for a system that end in a state(s) where an attacker has successfully reached his goal(s). Each path in the attack graph points from the attacker’s location to an undesirable position, such as one depicting an attacker obtaining administrator access of a mail server. Furthermore, for measuring the security level of a system, a security metric has to be utilized.

Security metrics based on models enables us to evaluate the overall resilience of the computer-based system against attackers qualitatively and quantitatively. In general, security metrics are utilized to support decision-making concerning security-related properties of a system or process.

This research sheds new light on security management for system design and improving security for the (self-) adaptive system during its adaptation processes. To understand and contribute to this purpose, we should be familiar with the broad scientific domains described in short in the next chapter.

1.2. Motivation

It is commonplace for people to worry about the security implications of their products (software and hardware). The vendors often are responsive to those who report a bug or a flaw and they directly fix this flaw in order to deliver secure products. These efforts by vendors are businesslike and competitive in nature. Secure products, however, in isolation cannot provide a secure world because security is a process, not a product. Products can give some protection, whereas the only way to provide a secure world is to put processes in place that identify and improve the inherent vulnerability in the products.

Due to the intricate associations of its components amongst each other, the system isn’t just an assembly of its parts. It has some properties known as ‘emergent properties’ (Checkland, 1981), which belong to the system en-masse. These properties do not belong to a particular component of the system. They may not be visible when the components are segregated. Nonetheless, they are evident when everything is integrated. There might be some elementary properties such as weight that can be gauged by determining individual component’s weight and accumulating them together. However, this is not the case in general, as they materialize from compound interrelationships between subsystems. In other words, a direct deduction of system properties from the characteristics of individual components may not be possible. Assuming, a system with a given structure (i.e. specific components interconnections) and a given emergent properties claim to be
changed as soon as the structure of the given system is changed, in that case, the emergent properties directly change. The emergent properties according to [1] include Volume, Reliability, Security, Repairability, and Usability. The focal point of this study will be 'Security' amongst those listed above.

Structural Reconfiguration is one of the two adaptive system's functionalities. The process of Structural Reconfiguration also goes through many steps (i.e. structures) until reaching the target structure. Therefore, due to unconstrained (freedom, with the perspective of security) reconfiguration process, many insecure schemas can be created.

However, there exist numerous works for measuring security level of a static and dynamic network. As a dynamic network that is more related to our work, many kinds of research have already been done and most of them use the HARMs approach that has been described in detail in chapter 2. For instance [2] has proposed and developed a model named Temporal-Hierarchical Attack Representation Model (T-HARM). However, one of the primary purposes of their work is to capture a network change and re-measure the security level of the dynamic network, but this network-change only refers to patch a vulnerable software on hosts. They did not consider a structural change of the dynamic network. Furthermore, [3] introduced an approach to capture the expansion of vulnerabilities in a network by using a Bayesian Network based model. That work is a theoretical work and they suggest it to be used for analyzing the dynamic security characteristics of a network. Again, that work also does not mention anything about structural-change. To the best of our knowledge, no previous work directly mentions or addresses the security level of a (self-) adaptive system during its structural reconfiguration.

**Challenges in realizing secure adaptive systems:** Now, a reasonable question can be asked; “What is the security property (vulnerability) of the system in each step (structure) while the structural reconfiguration is executing?” To the best of our knowledge, there is no research done on this question. Thus, the answer can be simple; “in each step, the security is unknown and not handled.” The next question is; “What is the solution for this not handled and unknown security?” To the best of our knowledge, nobody has investigated such an approach to identify the security level and vulnerability of a system during this process. Hence, this is our first motivation to tackle this issue and improving the security of a system with structural reconfiguration functionality.

The next challenge and motivation are solving the issue that the (self-) adaptive system faces during a structural reconfiguration due to having alternative and different structures with a different security levels which can be selected along the process. Also, to the best of our knowledge, there are no approaches that guide the structural reconfiguration process in a secure fashion for a (self-) adaptive system.

As third and last motivation, we desire secure reconfiguration planning to be a part of the system design field to raise awareness among system engineers and system designers about the serious security issues.
1.3. Problem Statement and Research Questions

It is clear that the growth of the system's architecture and its complexity and the demand of adaptability and multi-functionality, drive a system to change (reconfigure) itself continuously. This change can be managed by a system administrator or the system itself at any moment during the system's lifetime, this means that the change does not necessarily occur at runtime. As mentioned above, this process of changing may go through several structures until it reaches the target structure and there are several alternative structures to be selected in each step. Obviously, each component (software) that constructs the system has some vulnerability against attackers, and the vulnerability of a component influences the vulnerability of its neighbors. As a result, the whole system has more vulnerable points against attackers. Thus, this validates the fact that the security of the weakest point is a deciding factor in calculating the security of the system. Therefore, organizations must ensure they can measure and manage the security level in any possible situations, during structural reconfiguration process in this case.

In addition, the characteristics and nature of vulnerabilities make the process of security management harder; a vulnerability can be exploited in the presence of or independently of other vulnerabilities and the links between vulnerabilities facilitate these dependencies. Therefore, the security level during structural reconfiguration is constantly changing along the process.

For a given reconfiguration plan (i.e. a plan to change a system) within a current system state and knowing the vulnerabilities of each component, from the first step of structural reconfiguration process, the system may go through unknown and unhandled situations from the security perspective. Hence, before restructuring executes, the security level of the system after and during the reconfiguration process is not known and it requires a security risk analysis. It is likely to create a vulnerable structure (while less vulnerable structures might exist) against attackers and attackers easily can exploit this vulnerable point to attack the system. Furthermore, the adaptation manager can select a structure among several possible ones in each step to finally reach the final configuration, which raises this question "Which adaptations?". Now, the question is "Which structure is the most secure structure to be selected in each step along the reconfiguration process?" This problem has to be addressed by techniques to guide the adaptation manager to ensure the security of the process.

Our research question can be broken down into the following sub-questions:

RQ1. How can a (self-) adaptive system manage its security in order to keep the system at the highest possible security level during the structural reconfiguration process, i.e. which reconfiguration strategies should be used to reconfigure a (self-) adaptive system during the reconfiguration process to guarantee the highest possible security level?

RQ2. Which threat modeling technique can be used for security modeling of a (self-) adaptive system during and after its structural adaptations?
RQ3. How the security level of a reconfiguration plan as a whole and also each configuration can be measured in order to select the most secure reconfiguration plan of the structural adaptation?

RQ4. How to implement a tool to compute the security level of a reconfiguration plan automatically?

1.4. Method

Amongst numerous methodologies used for writing a thesis, we have adopted a systematic approach. To answer the research questions, RQ1-RQ4, we have proposed a framework that evaluates and identifies all the combinations of existing vulnerabilities in a (self-) adaptive system that are critical to the overall security level of the system during its structural adaptation process. By this evaluation, the framework can specify the most secure structural reconfiguration. The framework uses the Attack Graphs for the security modeling. Through evaluating the security level of all possible reconfiguration strategies for a given reconfiguration plan, the framework is able to evaluate the security level of a reconfiguration plan as a whole. Lastly, we have implemented a prototype tool in Java to verify the framework to compute the security level of a reconfiguration plan automatically.

Our method consists of the following steps:

1. We performed an in-depth literature review of all related areas including (Self-) Adaptive Systems, Information Security, Security and Threat Modeling Techniques, Attack Model and Attack Graphs Generation, Network Security Metrics, etc. that are described in chapter 2. Furthermore, to simulate our contributed approach and test our case study, we chose MulVAL as the threat modeling generator. MulVAL is an open source tool for system security analysis. This literature review consisted of reading scientific articles, books, web resources, review previous course materials, manual of tools, etc.

2. Based on the above literature review, we chose attack graphs as the security modeling technique to model the security of an adaptive system during its reconfiguration phases.

3. Given the initial structure and the reconfiguration plan, a technique was determined to generate all possible configurations of the (self-) adaptive system; thereupon we determined a method to generate all possible reconfiguration strategies that the system will go through from the initial configuration to the goal configuration of the reconfiguration plan. The reconfiguration strategies are described as a list of sequences of configurations. Each configuration is described as a specific file to be fed to MulVAL attack graphs generator.

4. We have selected several security metrics (Shortest path metric, Number of Paths metric, Mean of Path Lengths metric, Attack graph probabilistic metric) related to our defined security modeling that is able to capture and measure the security level of every goal state of an attacker in a reconfiguration phase. Furthermore, we proposed a few new security metrics to measure the security level of the reconfiguration paths and the
reconfiguration plan as a whole. Following which, we calculated the security level of each reconfiguration path to identify the most secure path in the adaptive system.

5. Lastly, we have performed a case study wherein we designed a tangible test case to bring the research problem to reality and followed the approach proposed in the thesis to find the most secure path in the said adaptive system.

1.5. Contributions and Limitations
   a. Contributions
   The main contribution of this work is proposing, implementing and testing a framework to analyze and measure the security of (self-) adaptive systems under structural adaptations. The framework can be used by a (self-) adaptive system designer to analyze a (self-) adaptive system security under structural adaptations and design proper reconfiguration plans to reconfigure a system in a more secure way.

   Furthermore, the result of the test case can be considered as a contribution because it exposes these security issues and raises awareness among system engineers and system designers about these pressing security issues.

   b. Limitations
   One of the limitations of this work is that we did not have a real system to implement and test our proposed framework. The required input was provided manually to the framework.

1.6. Target groups
Since this thesis primarily discusses the security measurement for adaptive systems and we have performed deep literature reviews on security and threat modeling, therefore it should be helpful for the security risk management community. Additionally, this thesis could also be advantageous for anyone in the IT Security Industry and the designer of (self-) adaptive systems.

1.7. Report Structure
This report is structured as follows: Chapter 2 gives a detailed description of the scientific background of our research and introduces the background of the domain broadly for the reader. Chapter 3 provides a detailed description of our proposed frameworks describing their reliability and validity. Chapter 4 shortly describes the implementation of the framework. In chapter 5, we apply our framework on a case study in the domain of (self-) adaptive system and analyze and discuss the results. Lastly, conclusion and suggestion for future works are found in Chapter 6.
2 Background

In this chapter, fundamental concepts and technologies will be demonstrated in detail.

2.1. Self-Adaptive Systems

A self-adaptive system is a system that can dynamically adapt its behavior and/or its structure at run-time according to the changes made to the dynamic environment and the system itself.

From the security perspective, the reason behind the adaptation can be different which is not important in this work. However, the process undertaken by the system while the adaptation is important.

The widely-used conceptual design for Autonomic Computing (also known as AC) is the MAPE-K concept which was first introduced by IBM in their white paper [4] that stands for Monitoring-Analysis-Planning-Execution and common shared Knowledge. Thereafter the concept of MAPE-k has been discussed in the context of self-adaptive systems [5]. Within the MAPE-K, a self-adaptive system and its environment are continuously Monitored and Analyzed. In case an adaptation is necessary, adaptation steps are Planned if there is a reconfiguration, behavioral and/or structural reconfiguration will happen. Moreover, the K represents the standard knowledge distributed among the monitor, analyze, plan and execute functions which includes data such as system topology information, metrics, historical logs, policies and etc. K is monitored by M part while it might be updated by E part. Finally, the self-adaptive system executes the computed plan within the managed system and might K be updated.

In general, Behavioral adaptation and Structural adaptation are the two primary dynamic adaptation techniques. Behavioral adaptation emphasizes on altering the behavior of the computational entities, whereas Structural adaptation intends to adapt the behavior by modifying the system’s architecture [6]. These two primary adaptation’s processes must be accomplished in such a manner that the system retains its required functionality and possibly at the highest security level. To perform this, the system needs information to analyze the environment and determine the appropriate adaptation. Usually, the system can monitor the required information for later use in order to decide which actions should be executed for achieving adaptivity [7], [8]. For instance, one of these actions can be a selection of a new configuration of the self-adaptive system. This selection executes by a method which should be perceptive and insightful into many aspects simultaneously. To name a few, for performance, reliability, security, etc.

2.1.1 Architectural Modeling

An architectural modeling is a broad aspect area describes the high-level structure of a system and a model will be created by using composition and decomposition of system components. It can be expressed by utilizing architectural styles which are well-known and commonly understood.

Furthermore, an architectural model is an artifact that catches all or some of the design decisions. It is the reification and documentation of these design decisions [9].
Abstraction is the primary objective of Architectural Modeling. Abstraction according to (ISO/IEC 2010) is an object demonstration that concentrates on the information associated with a distinct purpose and neglects the rest of the information. In other words, abstraction enables concentration on fundamental characteristics in order to hide unimportant details.

Intuitively, a system that will be modeled will have too many details including structural size, structural complexity, behaviorally complexity, emergent properties, etc. Consequently, Architectural Modeling concentrates on a few important characteristics to be computationally and intellectually tractable. Also, Architectural Modeling techniques by using different forms of abstractions solve this complexity, e.g. by focusing only on the security issue.

Two crucial key concepts should be applied in modeling different objects of abstraction, which are View and Viewpoint. IEEE 1471 has standardized architectural modeling and defined those key concepts as follows [10]:

- **View** is a description of an entire system from the perspective of a relevant set of concerns.
- **Viewpoint** is a specification of the conventions required for assembling and handling a view. In other words, a viewpoint is a pattern/s or template/s to generate particular views by discovering the goals and audience for a view and determine methods for model creation and analysis.

The primary architectural concepts involved in the modeling abstraction according to [11] includes the following:

- **Components**: That represent subsets of functionalities and data, presented via explicitly specified interfaces.
- **Connectors**: That represent the way of interaction among components.
- **Interfaces**: That represent the interaction points between components and connectors and outside world.
- **Configurations**: That represent distinct relationships among components and connectors within a system.
- **Rationales**: That represent the documentations of architectural decisions.

There are different approaches to architectural modeling, and graph-based methods are one of the common techniques used to model components and their interconnections.

Lastly, we can conclude that the architectural modeling enables us to analyze and reason about a system architecture at an abstract level.

### 2.1.2 Dynamic Reconfiguration

In general, dynamic reconfiguration (DR) is a dynamic process that allows for someone to change the system configuration without deactivating the system or the major affected node.

Commonly, according to the moment at which DR is specified, it can be classified into two categories: either programmed or ad-hoc. While programmed
reconfiguration is specified at the design time, ad-hoc reconfiguration is unpredictable at that time and can happen at runtime [12]. Therefore, programmed DR is relevant to the concept of (Self-) Adaptive System.

In this work, we are focusing on structural reconfiguration for (self-) adaptive systems in order to answer the research questions, hence the following section highlights structural reconfiguration.

Structural Reconfiguration involves changing the set of component instances and their interconnections at runtime. In other words, Structural Reconfiguration involves transforming one configuration into another configuration, i.e. transforming a system from one specific structure to other structure [13].

In addition, the reconfiguration process involves adding components, deleting components, controlling stability and balance with the environment by reacting to its changes. A system that claims to achieve benefits of structural reconfiguration needs to be reconfigurable at run-time.

Structural reconfiguration became the primary process for modern systems such as adaptive systems, component models, component-based-system and autonomic component systems. For instance, in autonomic component systems, each component is treated as an autonomic element that can continuously and dynamically reconfigure itself according to such specific functions [14], [15].

In literature, reconfiguration or dynamic reconfiguration is not a new issue and many methods have previously been proposed for obtaining structural reconfiguration process; Each with its own rules. For instance, Metamodeling [16], graph transformation [17] [15], reconfiguration patterns [18], etc. For example, graph transformation rule can be applied for generating a new reconfiguration architecture and its rule consists of a left child and right child. The left child gives the system configuration in which a rule can be applied, and the right child shows the result [15].

Moreover, the dynamic reconfiguration is considered as an essential criterion to ensure an acceptable availability and quality of service, for some crucial and highly available systems [16]. Furthermore, dynamic reconfiguration means the ability to reconfigure a system at run-time and some new, sometimes unpredictable conditions should be taken into consideration without completely stopping it.

Also, in [16] the concept of auto-reconfiguration is used and it is described as the ability to take dynamic reconfiguration decisions without the intervention of an external actor (usually a human administrator).

2.1.2.1 Reconfiguration plan
In general, the reconfiguration plan is the set of decisions, together with the possible choices, the developers or administrators must make. It can also be defined as a predefined plan that is capable of guiding the reconfiguration process from the start of execution until termination. Probably, a single reconfiguration plan includes a number of sequential steps (processes), and in each step, probably (not always) there are more than one available choice to process; therefore, a decision must be made to select a right choice among other choices for specific purpose/s. However,
a reconfiguration plan describes all possible steps while only some of them are valid. Besides, a Secure Reconfiguration Plan is different from an ordinary reconfiguration plan. And at the end of this work, we would like to define a Secure Reconfiguration Plan according to the research questions and the expected contributions.

Also, the query of how to write reconfiguration plans and how to manage, control and maintain the influence has been left to the system administrator and require administrators to write reconfiguration plans manually [19] and it is still a challenging area for investigation, perhaps because of the problem-diversity for each case.

Reconfiguration plans are used for many purposes in different literature. One research [20] proposed a reconfiguration plan that aims to determine a series of radial operative topologies for the medium voltage (MV) network that reduces the entire operative costs (energy). This process goes through three phases: firstly, defining sets of search-spaces for the solution; Secondly, verifying some constraints for the operation and finally, applying techniques to determine the plan. The aim of defining this plan is reducing energy consumption.

Additionally, another research [21] defines a reconfiguration plan as the way one configuration rearranges into another using certain sets of reconfiguration actions. In other words, it tells us what connections are to be added and deleted in transferring the system from an arbitrary structure to another structure without a single deviation of system's principle. The aim of defining this plan is minimizing the number of reconfiguration steps (least number of reconfiguration steps) during the reconfiguration process. Thus, those steps are costly in terms of time and energy consumption.

In conclusion, the most proper summarization of defining reconfiguration plan in the context of the above examples is according to [6] that defines a reconfiguration plan as a process that "describes different strategies (sequences of actions) to reconfigure a system structure to reach a target structure".

2.2. Threat Modeling and Analysis
Managing and analyzing the security of a system is not a straightforward process to handle, we need a technique such as Security Modeling Technique or Threat Modeling to illustrate a general view of information security.

In literature, threat modeling uses models to discover security problems. Using a model means removing a lot of details to provide a look at a bigger picture, rather than describing every aspect that requires experience in information security for understanding it. It is only used in determining issues and problems against an attacker in a system that has not built yet [22], [23].

In the domain of security modeling, the sooner you find problems, the easier is it to fix them. Threat modeling is all about finding problems (i.e. threats and defense) in the system, hence it should be addressed early in the design process or development cycle. Generally, there are three strategies for security modeling: focusing on assets, focusing on attackers, and focusing on software. These can be interpreted by text-
based or graphical-based approaches, for instance, attack tree and STRIDE [22] approaches, respectively.

In addition, another useful variant for security modeling is Graphical Security Models (GSM) to show and analyze security situations that explore the vulnerabilities of enterprise systems. The reasons behind using GSMs are its user-friendly interface, intuitive, visual features with formal semantics, and algorithms that allow qualitative and quantitative analyses [24]. In addition, the GSM technique can be divided into two parts; Attack modeling techniques and defense modeling techniques. Attack models concentrate on instructions that are triggered by an attacker and compute the vulnerabilities of the system, while defense models are centers on detection, reaction, responses, and prevention [24].

Both attack and defense modeling techniques have various methods and tools that in turn, have their own properties and behaviors. Vilhelm Verendel [25] reviewed more than 100 methods for security metrichation [26]. Also, Barbara Kordy, Ludovic Piètre-Cambacédès, Patrick Schweitzer [24] describe more than 30 methods for attack and defense modeling each has specific features and aims.

2.2.1. Modeling Languages

Security modeling languages define the different views and the different ways of demonstrating threat modeling that generally includes three main concepts: asset-related, risk-related and risk-treatment related [27]. Asset-related concepts define what assets are essential to protect, and what criteria ensure the security of the assets. An asset is anything that is valuable for the system and is required for delivering its objectives. Risk-related concepts represent how the risk itself is described. A risk is the compound of a threat with one or more vulnerabilities leading to exploit one or more of the assets. Risk treatment-related concepts present what requirements, decisions, and controls should be established and executed in order to decrease possible risks.

There are various types of security modeling languages and making a comprehensive classification is not an easy task. For instance, BPMN [28], Secure Tropos [29], Misuse Cases [30], Mal-Activity Diagrams [31] are four different security modeling languages that manually require human’s interaction in the process of designing security model for a system. On the other hand, there are various modeling languages that are programmable and are able to generate security models automatically. For instance [24] describes various types of modeling language; Attack Tree, Attack Graph, Hierarchical Attack Representation Models, etc.

2.2.1.1. Attack Trees (ATs)

“Attack Trees provide a formal, methodical way of describing the security of systems, based on varying attacks. Basically, you represent attacks against a system in a tree structure, with the goal as the root node and different ways of achieving that goal as leaf nodes” [32]. It can be used to find threats, or organize threats found with other building blocks, or merging both.

Understanding this approach gives us a fundamental cognition for almost all attack-defense approaches that exist today, recently there have been investigated more than 30 approaches for analyzing attack and defense scenarios and most of them
extend the original model in one or more dimensions [24]. Therefore, we would like to consider the basic approach of ATs in detail here.

This technique creates a tree as a model and the root of the tree represents the attackers’ foremost target. Successors of the root represent the attackers’ sub-targets. Leaves of the tree depict the minimum required action for attackers, that corresponding the atomic components, or they are the starting point for the attackers. The root successors can be shown either disjunctively (OR) or conjunctively (AND). Disjunctive lines represent a different alternative for the attackers to reach the goal, while conjunctive lines represent simultaneous action needed to reach the goal. As shown in fig 2.1 and fig 2.2. These two examples are simple ones and probably incomplete because there are many other ways that an attacker can opt for achieving the goals, therefore establishing an AT is subjective and not straightforward.

Figure 2.1. An example of a threat tree taken from [24]: Gaining administrator privileges on a UNIX system.

Figure 2.2. An example of a threat tree with values assigned to each node. Taken from [33].

Once the tree is complete, it is time to assign values to various leaf nodes (presumably this assignment (values) will be the result of painstaking research in each node), then make calculate (bottom up) the nodes followed by calculating the security of the goal.

After the above process, OR and AND nodes should be focused upon; The value of an OR node is possible when at least one of its children exists, and impossible if none of the children are present. Furthermore, AND node is possible only if all children exist, else impossible. Then the possible attack path is shown as dotted lines from a leaf to the goal (root) as shown in Figure 2.2.
The above mentioned "values" can be presented in both boolean and continuous form. For instance, to name a few, some of the possible boolean values are: specialized equipment required versus no specialized equipment, easy versus difficult, intrusive versus nonintrusive, expensive versus inexpensive, legal versus illegal., etc. Similarly, continuous values can be like in Figure 2.2, where the cost in dollar is assigned to the leaves followed by bottom-up calculation with respect to OR and AND constraints.

2.2.1.2. Attack Graphs (AGs)
The demand to automate the process of evaluating system's vulnerabilities against attackers has increased given the continuous growth of systems in general. When the demand is evaluating the security of a system, it is never sufficient to consider only the appearance or non-appearance of isolated vulnerabilities. Since there are many non-appearance factors that affect the strength of these vulnerabilities, such as relations and dependencies between vulnerabilities. To overcome this predicament, an attack graph approach with its characteristics is a sufficient technique to demonstrate and analyze the security level of a target system.

The concept of attack graph was first introduced by Phillips and Swiler [34]. An attack graph is a collection of all scenarios that demonstrate how attackers can compromise dependability and security of any type of a system and through a brief illustration of all paths within a system that end with a state where an attacker has successfully reached its goal, meanwhile offering meaningful capabilities to analyze the security in the system.

To clarify and understand the concept of an attack graph, first, we need to know what the requirement is to construct an attack graph. The first step is to gather information about the system’s connectivity (i.e. system structures), software installed on each host, running services, policy, and host access list. The second step is to identify and collect information about existing vulnerabilities on the whole given system. Although both steps can be reasoned as one single step, however, since the techniques used in gathering the information is different, they are broken into two distinct steps.

As a case in point, the first step can be determined by using firewall rule and tools like nmap [35] or Wireshark [36]. And the following step can be discovered by searching online vulnerability repositories (CVE) or using vulnerability scanners such as Nessus [37], Retina [38].

Next, we need to know how an attack graph is presented. For representation and analyzing system security, several varieties of attack graphs have been introduced, to name a few: “exploit dependency attack graph” [13], “multiple-prerequisite attack graph” [25], graph-based approach attack graph [39], logic-based approach attack graph [41] etc. Two of the frequently used types, viz. Logic-Based and Graph-Based are discussed below:

A. Graph-based approach Attack Graph
Generally, the graph-based approach attack graph is presented by two main elements: conditions and exploits. While exploit indicates a vulnerability and hosts that are affected by the vulnerability, conditions are system attributes can either be a precondition or postcondition of an exploit. precondition shows the execution
requirements for an exploit and postconditions are outcome consequences from the realization of an exploit. System attributes contain information such as access rights (privilege levels) of machines and a combined relationship between machines. With mutual dependence between these vulnerabilities, post-conditions that are an outcome from an execution of an exploit may function as pre-conditions for subsequent exploits.

It should be noted that all preconditions for an exploit have logical AND relation (conjunctive relation), and all postconditions for an exploit have logical OR relation (disjunctive relation).

Commonly, an attack graph can be seen as a directed graph that shows all attack paths that take the system from the original (secure) state to one or more target (compromised) states. This presentation of attack paths can be categorized into three categories based on how nodes and edges are described: condition-oriented, exploit-oriented, or condition-exploit-oriented [39]. Condition-oriented presentation describes nodes as conditions and edges as exploits. In contrast, exploit-oriented presentation delineates nodes as exploits and edges as conditions. Finally, in case of the frequently used condition-exploit-oriented presentation, an edge may connect an exploit to a condition or vice-versa, i.e., it can represent nodes as both conditions and exploits.

Figure 2.3 shows an example of the graph-based approach attack graph presented as condition-exploit oriented. It is a directed attack graph which was generated from a networked-based system that consisting of two hosts with a fixed security policy that a user on host 0 (user(0)) is not allowed to obtain root privileges on host 1 (user(1)) unless the network is in a compromised state. There are two sets of vertices exploits and conditions, while exploits are shown inside ellipses, conditions are displayed along the arrows. For example, rsh(0, 1) depicts an exploit (remote shell login) from user(0) to user(1), and the condition trust(0, 1) means a trust relation is installed from user(0) to user(1). Directed arrows travelling from an exploit to a condition suggests that a condition is required to be satisfied for executing that exploit. Similarly, when the arrows travel in the opposite direction, it means that if an attacker can execute the exploit, it can satisfy the condition.

Figure 2.3. An example of Attack Graph. Taken from [40].
Additionally, three attack paths can be identified in Figure 2.3. The attack path on the leftmost side is a sequence of exploits and conditions: \{user(0), ftp_rhosts(0,2), trust(0,2), rsh(0,2), user(2), local_bof(2,2), root(2)\}. An attacker, first starts to exploit the ftp vulnerability on host 2 to add more of trusted hosts on this machine. Next, they try to establish a trust relationship. If successful, the attacker can execute shell command on host 2 without having a password. Finally, exploiting a local buffer overflow on host 2 increases the attacker’s privilege to be the root of the target host (2).

**B. Logic-based approach Attack Graphs**

A logic-based attack graph operates to encode the logical relationship between causes and effects among configuration settings and possible attacker privileges. It explains (why an attack can happen), rather (how an attack happens) as in some other attack-graphs approaches [41].

Logic-Based attack graphs are presented by three types of vertices as shown in figures 2.4 and 2.5: diamond, rectangle and elliptic vertices. Diamond (privileges) vertexes describe privileges an attacker could achieve through exploiting the vulnerabilities in the system. Ellipse (exploit) vertexes describe an attack step that can point to a privilege. Rectangle vertices describe the facts about system configuration, including existing vulnerabilities in the system, firewall configuration, open port and etc.

Figure 2.4 shows a simplified attack graph generated by MulVAL where the numbers are the id for each node. For example, Node 18 is an attacker’s location – Internet in this case, and node 1 is the attacker’s sole goal which is a workstation to gain root privilege.

![An example of simplified Logic-based Attack Graph generated by MulVAL.](image)

Figure 2.4. An example of simplified Logic-based Attack Graph generated by MulVAL.
The attack graph is expressed as a Boolean formula where Ellipse is considered as an AND expression requiring its children-nodes to be true for the exploit to work. As an illustration, for the exploit described in node 14 to work, all the three nodes, viz. 15, 19 and 20 should be true.

The diamonds represent the logical OR expression that requires one of its children to be true for obtaining privileges. For example, an attacker needs either the exploit described as node 23 or node 6 to be true for obtaining the privilege node 5.

The rectangles are the configuration of the system. For example, node 17 is a network configuration, that allows connection between the system and internet through port 80 and tcp protocol.

Generally, the logic-based approach is a well-understood field, and due to the semantics of logic, this field is well-developed in computer science. Knowledge about attacks and system configuration is formulated in logic.

Another advantage of logic-based approaches is the clarified specifications of causality relationships between attacker’s potential privileges and pieces of information of system configuration.

Logical attack graphs are very useful as a simple depth-first-search traversal identifies and calculates all possible attack paths. This property is beneficial for some types of security metrics such as security path metric described in [39].

2.2.1.3. Hierarchical Attack Representation Models (HARMs)
Existing attack models which described above (Attack Graph and Attack Tree) suffer from the state-space explosion (scalability) and dynamic change problems. Firstly, the scalability problem for attack graphs happens because of calculation of full attack paths, i.e., full attack graphs generation, that has exponential complexity. Researches have been done on attack graphs that show the state-space explosion problem [42], [43], [44], and to solve this problem they consider the subset of attack graphs, i.e., whole attack paths. On the other hand, according to [32] there is no issue with state explosion in ATs [32], [45], [46], [47], [48], but the main issue is

![Figure 2.5. A sample of Logic-based Attack Graph generated by MulVAL.](image-url)
about there is not a generation-technique to construct ATs straight from the network system specifications. Moreover, according to [47] the constructions of ATs are a hand-operated task that will be done by a security analyzer (i.e., Red team, security expert) in their organization.

Secondly, the dynamic adjustment problem happens when there are changes in the network system, such as network configuration and vulnerabilities. Those modifications in the network change the attack model accordingly. Both scalability and dynamic adjustment difficulties are essential factors to construct attack models especially for dynamic network and adaptive system because they sometimes make the reasonable use of attack models infeasible.

**A simple definition of HARMs:** To overcome or mitigate the AGs and ATs above mentioned issues, Hierarchical Attack Representation Model (HARMs) has been proposed [2], [49], [50], [51] and represented as two vertical layers. Generally, the upper layer (upper level) captures and represents the network information (e.g. Topological Reachability, Network Routing Rules), and the lower layer (lower level) captures and represents the vulnerability information of individual hosts in the network separately. Usually, this vulnerability information refers to CVSS base score. The CVSS is a public and open framework for assigning quantitative values (scores) to software vulnerabilities according to their severity with a decimal number scaled in the range between 0.0 and 10.0.

Recently some research for analyzing the security of dynamic networks is developed based on HARMs. For instance, [2] proposes a security model based on HARMs to analyze the security of dynamic networks and investigate the effects on existing security metrics when the network changes.

The model defines a set of temporal HARMs (T-HARM) to capture changes in the network at different times, for each time of network-change a snapshot is defined as a separate HARM for this network configuration. For example, if the network changes twelve times, twelve snapshots of HARMs are generated. However, the changes that are captured by T-HARM include only the appearances of a new vulnerability and patching of vulnerabilities. The model cannot capture other changes, e.g., firewall rules, hosts reachability rules, network topologies.

**HARM construction:** T-HARM for a dynamic network is defined as a sequence of HARM snapshots. Each snapshot (HARM) is defined as two layers as demonstrated in Figure 2.6.

The upper layer represents the host’s reachability information and has been called as a dynamic Attack Graph that is defined as a directed graph and depicts as: \( AG = (H, E) \), where \( H \) is a finite set of hosts in the network and \( E \subseteq H \times H \) is a set of edges which connects the hosts.

The lower layer is responsible for capturing vulnerability information and is represented as a dynamic Attack Tree and defined as 5-tuple and depicted as: \( AT = (A, B, c, g, root) \), where \( A \) is a set of components which are the leaves of at. \( B \) is a set of gates which are the inner hosts of at which Requires \( A \cap B = \emptyset \) and root \( \in A \cup B \). \( c \) is a function \( c: B \rightarrow P(A \cup B) \) represents the children of each inner host in at (assuming there are no cycles). Also, \( g \) is a function \( g: B \rightarrow \{\text{AND, OR}\} \) describes
the type of each gate that means the vulnerabilities of a host are joined using a logical AND and OR gates.

**Security Metrics and Measuring Security based on constructed HARM:**

Now that the construction of HARM is completed as described above, the model is prepared to be utilized for measuring severity for individual vulnerabilities existing in each host of the network. Before applying any network security metrics, the vulnerability of each host must be calculated individually. Intuitively, the only information presented by HARM comes from both layers, while the upper layer provides the reachability information of the network and the lower layer provides the vulnerabilities of the hosts. To calculate vulnerability for each host in the network; from the upper layer, the values of three attributes of Network Centrality Measures (NCMs) (degree, closeness, and betweenness) are calculated for each host. The details of computing NCMs are described in [53]. And the average of these three values is computed and denoted as NSv (v means vulnerability).

![Figure 2.6. Demonstrate the two layers of T-HARM with their relation. Taken from [2].](image)

The lower layer represents a number of ATs. Each AT has the corresponding host in the upper layer as shown in Figure 2.6. Related to CVSS base score, assumptions are made and assign probability, impact and cost values to each vulnerability in order to perform security analysis. The security metric for vulnerability v denoted as VSv. It is normalized and used the CVSS base score as an only metric.

Now, by using equation 2.1 from [52] which computes combined importance measures values of NSv and VSv for each existing vulnerability and denoted as CVv.

\[
CVv = aNSv + (1 - a)VSv \tag{2.1}
\]

where \(0 \leq a \leq 1\) is a weight value and according to [51] \(a = 0.5\).

The result of this calculation is scoring each existing vulnerability with CVv value.
Furthermore, the following security metrics related graph can be used in the model: Risk on attack path, Return on attack path, Cost on attack path, Standard deviation of attack path lengths, Probability of attack success on path, Normalized mean of attack path lengths, Mean of attack path lengths, Mode of attack path lengths, Number of attack paths and Shortest attack path.

Furthermore, [2] described and analyzed the effect of vulnerability-change on the value of the aforementioned security metrics by two scenarios. The first scenario describes the first snapshot (HARM) as: Each host in the upper layer has only one corresponding vulnerability in the lower layer and then by applying the described approach, the CVv value for each vulnerability is listed followed by execution of aforementioned security metrics. Then the sequence of snapshots (HARMs) is generated, and on each snapshot, one vulnerability (Highest CVv value) will be patched and the metrics will be calculated again.

In summary of analyzing the results, the obvious point was that all the considered security metrics presented some level of changes in their values, especially the Number of Attack Path was most sensitive to the changes (patching vulnerability), but the shortest attack path metric remained same.

The only difference between the first and second scenario was assigning two vulnerabilities for each corresponding host instead of one from the first snapshot. And for the sequence of the snapshots, only one of the two vulnerabilities patches. The results of the second scenario showed that the Path Metrics did not change in the sequence of the snapshots. Because of one corresponding vulnerability on each host has been remained (not patched), the attack paths have not been changed.

In conclusion, recently, many different approaches for analyzing dynamic network as attack scenarios have been proposed. Most of them propose HARMs as a modeling approach, for example [49], [51], [52]. Furthermore, all of them extend the original model of HARMs as described above. This extensiveness is in one or several dimensions, for instance, defining HARMs as two layers, upper layer capture the network hosts reachability and handled as an attack graph, the lower layer captures the network vulnerabilities and handled as an attack tree. Moreover, the CVSS base score is used as the initial value of vulnerabilities.

2.2.2. Attack Model and Attack Graphs Generation
An attack model can generally be described as a formal representation of all security-related attributes of the attacker, the underlying system and the circumscribed defender in the network.

One part of modeling the global appearance of system security is constructing attack graphs. Basically, an attack graph is a subgraph of Attack Model which includes all possible attack paths, where the attacker lastly succeeds in obtaining his goal.

Attack graph production by hand, however, is error-prone, tedious, and impractical for a network with more than a hundred nodes, and when the purpose of generating attack graph is such as the purpose of this work which is calculating and analyzing the combined security level of a number of networks. Therefore, automated
techniques for generating and analyzing attack graphs have been in high demand in both academic and business domain.

A number of algorithms for automatic graph generation have already been published, for instance [53], [54]. Moreover, a number of tools have been developed for security modeling; One such class of developed tools is based on attack graphs. Examples include MulVAL [55], NetSPA [56], CySeMoL [57], TVA [58] and Sheyners attack graph-tool [59]. These tools have the same view in determining which system security-related attributes are important in order to be involved in security evaluation and producing attack paths. However, in literature for example [55], [60], [61], [62] shows some differences between them in terms of: Monotonic or non-monotonic, Single path or all paths, backward or forward chaining, probabilistic or deterministic models, visualization variants, input formats and Logic-based or graph-based approaches.

2.2.2.1. MulVAL Attack Graphs Generator

MulVAL stands for Multi-host, Multi-stage Vulnerability Analysis Language. MulVAL is an open source project that was started in Princeton University. The Logic-based Network Security Analyzer utilizes Datalog as its modeling language. Network information is encoded as Datalog, that includes existing vulnerabilities, configurations of each component and the system as a whole and all other security-related data. The MulVAL reasoning engine is composed of a combination of Datalog commands that captures the system software's behavior and the interaction between various components in the given network. Datalog is found on first-order logic and therefore it needs to be valid and flawless. To ensure that the facts are only computed once, Datalogs inference engine utilizes XSB [63] tabling mechanism. XSB was established in Stoney Brook and is one of the keystones behind the reason why MulVAL attack graph generation complexity level is polynomial concerning to the system size. Also, another advantage of using XSB is that the order of rules does not affect the result of the execution, because the tabling mechanism gives comprehensive declarative-style logic programing.

The essential part of knowing how the tool works and specifications of the tool are by understanding its input data. Inputs to the tool according to [55] are categorized into six categories as listed down;

1. **Network configuration**: This input describes the firewalls and router configurations as abstract host access-control lists (HACL). Basically, hacl shows which machine to machine connections are possible. For example, HACL entry that allows httpProtocol traffic to flow from webServer to port httpPort on workStation:

   hacl(webServer, workStation, httpProtocol, httpPort)

2. **Advisories**: It describes the existing vulnerabilities on hosts. To do that MulVAL uses Open Vulnerability Assessment Language [oval] (OVAL) that has been developed for formalizing appearance of vulnerabilities on hosts in a network. An OVAL uses such formalized vulnerability definitions and scans the network for vulnerable running software on each host. Then the result from the scan will be converted into Datalog clauses as below:

Namely, based on CVE the recognized vulnerable software is identified, and provided information about the vulnerability's impact on the network.

3. **Host configuration**: Identifying all active software and services with their configurations. OVAL scanner retrieves configuration parameters on each host. For example, port number, protocols, privileges, etc. Then the result will be converted into Datalog clauses as below:

   networkService (webServer, httpd, TCP, 80, apache).

4. **Principals**: Describes the user for a host.
   hasAccount(victim_7, 'webServer', user).
   hasAccount(sysAdmin, 'dataHistorian', root).

5. **Interaction**: Usually, an attacker to reach his goal needs to go through multistage attack path, the semantics of the vulnerability and configuration of the system software and running services define an opponent's alternatives in each stage. These are encoded as Horn clauses (i.e., Prolog). Basically, every rule in Prolog is a horn clause, where the first statement is a conclusion that is enabled by the remaining statements. For example,

   execCode(Attacker, webServer, Priv) :-
   vulProperty('CAN-2002-0392', remoteExploit, privEscalation).
   networkServiceInfo(webServer, httpd, tcp, 80, apache).
   netAccess(Attacker, webServer, tcp, 80).

If httpd running on the webServer with CVE defined id and IDSB defined the impact of the vulnerability as privilege escalation that is a remotely exploitable vulnerability.

The buggy software is running on apache and listening on tcp and port 80, and the attacker is able to communicate with the webServer through the tcp and port 80. Then the attacker can run arbitrary code on the webServer under Priv.

6. **Policy**: This type of input describes the access privileges. For example:

   allow (systemAdmin, write, dataHistorian).

That means the system admin is able to perform "write" on the database.

The MulVAL framework is shown in Figure 2.7. An OVAL scanner scans each host, the result of the scan as described above with all other mentioned input data are loaded into an XSB environment. Based on the inputs and especially the interaction rules, an attack graph will be generated. Figure 2.5 shows the detailed attack graph generated by MulVAL.
2.3. Network Security Metrics (NSMs)

NSMs allow assessing the overall resilience of networked systems against attacks quantitatively. For that purpose, such metrics are of great value to the progress of security-related decision-making of organizations.

Because the security metrics field has an extremely wide scope, we only highlight some important aspects in this work. First, we describe metrics and measurements in order to distinguish metrics from measurements. Second, specify the desired properties of security metrics. Third, we describe the CVSS (Common Vulnerability Scoring System) that is a vulnerability impact quantification framework used by various NSMs. Fourth and last, we describe the NSMs related to attack graph (AG).

2.3.1. Metrics and measurements

A measurement numerically calculates only a single parameter of the objective of measurement which is not able to include a value (facilitate decision making) by itself. On the other hand, a metric is obtained from more than one measurement to illustrate an important correlation that can assist a decision [64]. In other words, a measurement is a perceivable value (associated with a given property or attribute) that can be obtained by using any proper technique which converts this value into data. While a metric is constituted by using a collection of measurements along with a set of predefined rules that let the translation of the collected data values. Basically, measurements are raw data, while metrics are generated from the analysis of those data. It is similar to the process of data mining when knowledge is generated from a set of information by using a set of techniques and then this knowledge will be used by a decision maker.

According to [65] metrics can be classified into three categories: direct metrics, indirect metrics, and indicators. For obtaining their respective measures, these three kinds of metrics use generalized measurement procedures of various methods. A direct metric utilizes a measurement method as described before. An indirect metric uses a measurement function that holds upon other direct or indirect metrics. In other words, an indirect metric measures through their established relations to the base metric. Lastly, an indicator applies an analysis model based on decision guidelines to achieve a measure that satisfies an information requirement.
For instance, the well-known software quality metric Line Of Code (LOC) is a direct metric for measuring the quality of a single class in a software. If the quality of the whole system is required to be measured based on LOC, an indirect metric can be defined which uses the outputs of the LOC metric by a function such as summation, minimum, etc. of all class’ LOC values.

In general, security metrics are used by security-related decision-maker to improve a process, system, or organization.

In addition, security metrics can be used to compare different security techniques or to indicate the degree to which security requirements of a system are achieved. Also, NSMs are used to improve the security level of a system systematically and automatically, and to predict this security level at a later point in time [39].

2.3.2. Desired Properties of Security Metrics
To design a security metric, a number of properties must take into consideration [39]:

- **Granularity**: Metrics should capture differences of all system states regarding to the variations among the respective attribute values. A system state with different security level must be regarded as having a different security level. Granularity property presents a metric with this ability.

- **Availability**: regardless of complexity, metrics should regularly be possible to measure the security level, preferably easily and efficiently.

- **Cost effectiveness**: The advantages of using a metric must justify its cost. Therefore, the computation process should be cost-effective.

- **Localization**: Metrics should have a scale with a specified type, range, and their respective interpretation. This property is required for the accurate application of the metric.

- **Validation**: This is the most scientific property. It states that a metric should be based on a hypothesis that relates the calculated or expected value to the real value of the security attribute being examined. Consequently, it is necessary to validate a metric to check whether the value of the calculated metric represents the associated security attribute of the analyzed system. Particularly, validation for model-based security metrics is necessary, since it is not derived from actual system behavior, it is derived from system abstractions.

2.3.3. Common Vulnerability Scoring System (CVSS)
CVSS is a public and open framework for communicating the characteristics and severity of security vulnerabilities and provides a standardized approach for assigning quantitative values (scores) to software vulnerabilities according to their severity [39] [66] [67]. A CVSS score is a decimal number scaled in the range between 0.0 and 10.0. This software vulnerability score is calculated from a set of qualitative values that are defined by security experts. Each qualitative value of a given attribute has a predefined corresponding quantitative value. Then the qualitative values are converted into quantitative values and combined by applying
complex formulas in order to provide the final score. Complete details about the computation of the final score and its formulas can be found in [67]. In addition, the final score does not include effects of any interaction with other vulnerabilities. In other words, each vulnerability is scored independently.

CVSS consists of three groups of metrics: the base group, the temporal group, and the environmental group. We are focusing on the base group in this work because the two other groups are derived from the base group by scoring the Temporal and Environmental metrics. “However, scoring the Temporal and Environmental metrics is not required” [66]. The base group and their respective sub-metrics are shortly described as follows [67]:

**Base group**: Represents the native vulnerability characteristics that do not change over time. It is constant with a fixed value and does not depend on the network environment. This group consists of six sub-metrics, and the last three sub-metrics can be categorized under the impact sub-group:

- **Access Vector**: Indicates how the vulnerability is exploited, its accepted values are Network, Adjacent Network, Local. It describes if a vulnerability can be exploited locally, from an adjacent network or wide network. The vulnerability access vector values are assigned as follows:
  - Requires local access: 0.395
  - Adjacent network accessible: 0.646
  - Network accessible: 1.0

- **Access Complexity**: It is the complexity measurement of the attack required to exploit a vulnerability. The assigned values are High, Medium, and Low. The values of Access Complexity are inversely relational. As follows:
  - High: 0.35
  - Medium: 0.61
  - Low: 0.71

- **Authentication**: To exploit a vulnerability, an attacker must authenticate to the system one or several times. This metric measures these number of times. Assigned values are Multiple, Single, and None. As follows:
  - Attacker needs no authentication: 0.704
  - Attacker needs single instance of authentication: 0.56
  - Attacker needs multiple instances of authentication: 0.45

- **Impact sub-metric**: This sub-metric measures the impact on Integrity, Availability, and Confidentiality, of a successfully exploited vulnerability. The accepted values for these three sub-metrics are:
  - None: 0.0
  - Partial: 0.275
  - Complete: 0.660

The equations and algorithms for calculating the base score are described below.
\[
BaseScore = roundToOneDecimal \left( \left( (0.6 \times Impact) \\
\quad + (0.4 \times Exploitability) - 1.5 \right) \times f(\text{Impact}) \right) \\
\]

\[Impact = 10.41 \times \left(1 - (1 - ConfImpact) \times (1 - IntegImpact) \right.
\quad \times (1 - AvailImpact) \right) \]

\[Exploitability = 20 \times AccessVector \times AccessComplexity \times Authentication \]

\[f(\text{Impact}) = 0 \text{ if } \text{Impact} = 0 \text{ else } \text{Impact} = 1.176 \]

The CVSS scoring system has been globally recognized and has been used by several vulnerability databases. For instance, CVSS is used by the US National Vulnerability Database (NVD) [68] to quantify the severity of reported vulnerabilities. Also, CVSS is used by the CVE system [69]. CVE stands for Common Vulnerabilities and Exposures, MITRE [70] has copyrighted the CVE and established in 1999. MITRE is a nonprofit organization that works research and development centers sponsored by the United States government and assist them with scientific research and analysis, development and acquisition and systems engineering and integration. CVE is a list of information that presents descriptions for publicly disclosed cybersecurity vulnerabilities and exposures. This list includes, for instance, CVSS vulnerability score, CVE Entry and so forth. CVE Entries are composed of an identification number, a description, and at least one public reference (publicly known cybersecurity vulnerabilities.). The identification includes “CVE” as a prefix and the published year of the vulnerability or exposure and ends with a sequence number, for example (CVE-1999-0065). These identifications are used by cybersecurity product/service vendors and researchers as a regular method for recognizing vulnerabilities and for cross-referencing with other repositories which accept CVE IDs.

There are several versions of CVSS, and the third version is the most current one [66]. However, the CVSS explained above refers to the second version of CVSS [67], which is the most used version by NSMs that are based on CVSS.

2.3.4. **NSMs Based on Attack Graphs**

As described before an attack graph is a model that represents all possible ways (paths) which an attacker takes to achieve its goal. Generally, an attack graph can be viewed as a directed graph that represents all attack paths that an attacker can take to reach its goal. The NSMs based on attack graphs are broken into two classes; Path Metrics and Non-Path Metrics.

2.3.4.1. **Path Metrics**

These metrics focus on the attack-path characteristics, such as the size or number of attack paths. Followings are some example of path metrics:

**A. Shortest path metric (SP):** According to this metric, a network security level will compare to the length of the smallest attack path
to the desired goal state (that denotes a critical network asset). The inspiration for this metric is that a network as a whole is not more secure than its smallest attack path. The formula of this metric is:

\[ SP = \min(L(P_1), L(P_2), L(P_3), \ldots, L(P_n)) \]  \hspace{1cm} (2.6)

where \( L \) is a function that calculates the length of the attack path \( (P_n) \). As a result, the longer the shortest path is, the more secure the network is [71].

B. **Number of Paths metric (NP):** This metric describes the number of different ways an attacker can compromise a given network asset. Simply, NP counts the total number of possible attack paths that exist in an attack graph between the attacker’s start point and the attacker’s goal [72]. A larger number of NP means less security. In other words, intuitively, if there are more separated ways to attack, it indicates that the attackers have more chances to compromise a network successfully.

C. **Mean of Path Lengths metric (MPL):** This metric calculates the average number of steps per all attack paths that exist for a given attackers’ goal in a network. In other words, MPL equals the total steps of all graphs divided by NP [73].

D. **Normalized Mean of Path Lengths (NMPL):** This metric represents the ratio between MPL and NP, as follow:

\[ \text{NMPL} = \frac{\text{MPL}}{\text{NP}} \]  \hspace{1cm} (2.7)

This metric solves the shortcoming of MPL, which the number of attack-path has not considered [72].

Furthermore, the authors of [72] also propose three other metrics which are driven from previous metrics that are: Mode of Path Lengths (MoPL), Standard Deviation of Path Lengths (SDPL), and Median of Path Lengths (MePL). Furthermore, the seven aforementioned path metrics can be classified into two groups:

- SP, NP, and NMPL are Decision Metrics.
- MPL, SDPL, MoPL, and MePL are Assistive Metrics.

The Decision Metrics are used to determine which out of many networks has the highest security level, while the Assistive Metrics can only be used in combination with Decision Metrics to improve determining the most secure network. This occurs only in a case that the application of decision metrics could not draw any conclusion. For instance, if two different networks have the same values of SP and NP, while the actual attack steps in these two networks are different. In this case, the assistive metrics can be used to improve the security assessments.
2.3.4.2. Non-Path Metrics
The non-path metrics produce measurements based on more general information about the attack graph, such as the number of hosts that are able to be compromised instead of depending on attack paths.

A. **Weakest Adversary metric**: This metric represents the security strength of a network using the least amount of work an adversary requires to reach its goal. The metric regards the least work spent by an adversary as the strength of the set of initial conditions (AGs arc) that enable the compromise of a system. In other words, if comparing two networks, the most secure will be the one that has a stronger set of initial conditions. For example, two attack graphs for two networks with different sets of initial conditions X and Y, if X is a definite subset of Y, network X is regarded weaker than the network Y. In the case, if neither of the sets can be a proper subset of the other, the security administrator can define different alternatives for comparison, e.g. the cardinality of the sets can be defined, then the set with larger size of initial conditions (AGs arc) can be regarded as more secure [74].

B. **Network Compromise Percentage (NCP)**: Represents the percentage of hosts in which an attacker can achieve any available privilege level. The metric computes security level of a network by dividing the number of compromisable hosts by the total number of hosts [75].

Intuitively, a higher NCP value means less secure network. One advantage of this metric is that it considers all hosts in a network to have the same impact on the network security level, instead of focusing on a specific host (attack goal) as most other attack graph-based security metrics do. Additionally, it has one shortcoming that the number of vulnerabilities on hosts is not considered. Therefore, if a vulnerability is removed from a host that initially had two vulnerabilities, the NCP value remains same, even though this removal has made the network more secure [39].

C. **Cumulative Score Metric**: This metric is based on independent attack paths in which the author, Noel et al. [76], supposes that the variant exploits may have different chances of being executed. It uses the exploit dependency graph in order to quantify system security by propagating exploit probability scores from initial exploits (i.e., attackers starting point) to the goal exploit (i.e., attackers’ goal).

In this strategy, each exploit has an exclusive score along with a cumulative score. The exclusive score of each exploit is provided as input to the model and represent the conditional probability of the exploit occurring when all its preconditions are already satisfied. In contrast, a cumulative score is a probability value which is computing by accumulating scores of directly preceding exploits, in agreement with the type of relationship that exists between them, the type of relations can be conjunctive or disjunctive.

When a single exploit A only precedes an exploit B, it means that B can only happen if A also happens (i.e., conjunctive relationship). In that case, the cumulative score of B is presented by the product of the exclusive scores of A and B. Then, supposing that B also has a conjunctive relationship with an
exploit C (with B preceding C), the product of the cumulative score of B and the exclusive score of C is presenting the cumulative score of C.

Besides, in a case which an exploit F can only be completed if either exploit D or E happen, it means that F has a disjunctive relationship with D and E. In such case, the cumulative score of F is presented by the product of the exclusive probability of F, and the exclusive probability of D or E happening, i.e.,

$$P(F) \times P(D \text{ or } E) = P(F) \times \left( P(D) + P(E) - P(D) \times P(E) \right)$$  \hspace{1cm} (2.8)

Finally, when the cumulative scores for all exploits of the graph have been measured, the cumulative score of the attackers' goal, based on the attack scenario indicates the security level of the whole system.

D. **Attack graph probabilistic (AGP) metric**: This metric is provided by Wang et al. [77]. It is recognized as an extended version of the cumulative score metric [39]. The condition-exploit-oriented attack graph is used to measure AGP. Consequently, both conditions and exploits will have exclusive and cumulative scores. This implies that the disjunctive relation will take place between a postcondition and the exploits that lead to it. Besides, a conjunctive relation will happen between an exploit and its preconditions.

The basic definition of AGP can be described as follows: Each condition c and exploit e with two probabilities is associated, viz, P(e) and P(c) for the cumulative score, and p(e) and p(c) for the individual score.

Where the cumulative score P(e) and P(c) includes the overall probability which an adversary can exploit e successfully or satisfy the condition c. While the individual score p(e) and p(c) are the fundamental probability for executing an exploit e, given that in the given attack graph, all conditions required to execute e are satisfied.

The individual scores have been obtained by reforming vulnerability scores presented by the CVSS base score and temporal score to probabilities.

On the other hand, the cumulative score considers the causal relations between exploits and conditions. These causal relationships are shown in two different ways: (i) An exploit to be executed; their preceding multiple conditions is required if there is a conjunction causal relationship between conditions. (ii) A disjunction exists between multiple exploits that satisfy the same condition. Then these two cases are defined similarly to the probability of the intersection and union of random events. Which is:

(i) If the two conditions c₁ and c₂ are required for executing e, then

$$P(e) = P(c_1) \times P(c_2) \times p(e)$$  \hspace{1cm} (2.9)

(ii) If e₁ or/and e₂ is required to satisfy a condition c, then

$$P(c) = p(c)(P(e_1) + P(e_2) - P(e_1) \times P(e_2))$$  \hspace{1cm} (2.10)
The complete definitions of formalizing Equation 2.9 and 2.10 can be found in [77].

Figure 2.8 uses an example to further expound the above two rules. The cumulative scores of exploits are described in the figure in the form of plaintext alongside corresponding exploits. The cumulative score can be calculated as follows. However, calculating the cumulative score of only two exploits have been demonstrated below.

(i) \( P(rsh(0, 1)) = P(trust(0, 1) \times p(rsh(0, 1)) = 0.8 \times 0.9 = 0.72 \)

(ii) \( P(user(1)) = P(rsh(0, 1)) + P(sshd\ bof(0, 1)) - P(rsh(0, 1)) \times P(sshd\ bof(0, 1)) = 0.72 + 0.1 - 0.72 \times 0.1 = 0.748 \)

Figure 2.8. Example of computing cumulative score. Taken from [77].

In addition, according to [39] the significant contribution of AGP is the method the authors propose to handle cycles in the attack graph, while the cumulative score metric does not provide any information about how to handle cycles in the attack graph.

It is worth remarking that the scope of this report does not include describing all the NSMs based attack graphs, such as Probabilistic Metrics including: PageRank-Based Metric, Metrics Based on Independent Attack Paths and Metric Based on D-Separation. And Bayesian Network-Based Metrics including: Metric Based on OR-Decomposition, CVSS-Based Metrics and Metric Based on Subgraphs. In addition, there exist specific metrics to quantify the resilience of networks against zero-day attacks such as d2-Diversity and d3-Diversity.
3. Method

3.1. Scientific Approach
In the literature, for measuring and analyzing the security level of a system, both quantitative and qualitative methods have been introduced. However only quantitative approaches are adopted in this work, as they are usually more suitable for the comparison of different methods.

3.2. Method Description
This section describes our proposed framework that aims to answer the research questions to generate the relatively most secure reconfiguration plan acted as a strategy to guide the reconfiguration processes for the (self-) adaptive system securely. This section is broken down into two main sections; First, a General Framework is proposed and briefly described. The aim is to comprehensively illustrate each step of the framework and to discuss the applicability of the framework; each step can be developed and improved separately by utilizing any proper technologies and approaches. Second, the Instantiated Framework is described in detail in which each step has been specified by utilizing a specific approach which our implementation is based on it.

3.2.1. General Framework
The framework is divided into five different steps, each step uses the processed data that provided by its previous steps. The framework is demonstrated in Figures 3.1 and 3.2.

![Figure 3.1. Architecture of General Framework for secure reconfiguration plan.](image-url)
Step 1: Data collection

The first step is responsible for collecting and preprocessing two different types of information in order to provide the input for the analysis. First, The System Information; that represents the initial system configuration including system reachability and connectivity, existing security measures in the system such as policies, procedures, and rules placed in firewalls, routers/switches, existing vulnerabilities and other system components. This process of information-collection can be accomplished by using such discovering tools nmap [35] or Wireshark [36] which can be used to gather information of system’s components such as OS versions, open ports, and current running services but do not identify the vulnerabilities that are correlated with them. Hence, scanning tools such as Nessus [37] or Retina [38] can be utilized, which are designed to scan and report all vulnerability(s) in the network [40]. In addition, this step faces one difficulty in which possibly scanning tools cannot capture some required information such as information about hosts (softwires) that are not yet included in the initial configuration; hence, these pieces of information have to be taken into consideration, for instance, they can be prepared by the system administrator.
Lastly, the collected information must be processed and filtered in such a way to provide the next step’s requirements.

Second, Reconfiguration Plan’s information which acts as a guideline that describes a set of atomic reconfiguration actions to achieve the target configuration from an initial configuration. The specified language for describing a reconfiguration plan according to [6] can be as follow:

\[
\begin{align*}
    a & := a; a' \mid a \mid a + a' \mid \emptyset : a \mid \alpha \\
    \alpha & := add(c) \mid del(c) \mid con(c, c') \mid dis(c, c')
\end{align*}
\]

where the \( \alpha \) is a primitive reconfiguration action. The actions include del(c), con(c, b), dis(c, b) and add(b) which mean removing the component c, connecting c to b, disconnecting c from b and adding component b, respectively. These actions can occur simultaneously in parallel and it can be depicted as (||), for example, con(c, b) || dis(c, d) that mean in one event host c connecting to host b and simultaneously disconnecting c from d. Furthermore, the symbols (;), (+) and (\( \emptyset : a \)) where \( \emptyset \) is a condition defined on the system structure) depict that a reconfiguration plan can be sequential composition, an internal non-deterministic choice and a conditional choice, respectively. The non-deterministic choice operator describes a source of selection points, that collectively establish the domain of controllability [6].

Adequate information about how formally and systematically a reconfiguration plan can be described can be found in [6]. The given (self-) adaptive system should provide these pieces of information.

**Step 2: Configurations and Reconfiguration Paths Generation**

The second step consists of two main activities. First, generate a set of all possible configurations for the given (self-) adaptive system based on the given reconfiguration plan and the system initial configuration.

Furthermore, the syntax of reconfiguration plan (RCP) is specified by using the mathematical symbols (brackets, braces and parentheses), ; and + (where ; depicts that a reconfiguration plan can be sequential composition like logical AND, and + operates as logical OR) and the primitive reconfiguration actions as mentioned before. Example 3.1 illustrates the syntax of specified RCP, different configurations generated and all possible reconfiguration paths generated.

Example 3.1: Figure 3.3 shows the initial configuration (IC) and the target configuration (TC) of a system which consists of five softwires (hosts) (W, F, V, M and S). The initial configuration does not include the connection (data communication) between M and W, and the connection between F and V, and S does not include in the initial configuration. For this system a reconfiguration plan (RCP) is defined to perform this reconfiguration.

The reconfiguration plan can be defined as follow:

\[
RCP = \{ [con(F, V) ; con(M, W) ; con(S, M)] + [con(S, M) ; con(F, V) ; con(M, W)] \}
\]
In this plan, the reconfiguration processes go through one of these two sequences of actions:

- First, the connection between F and V, then connection between M and W, and finally connection between S and M are established.
- First, the connection between S and M, then connection between F and V, and finally connection between M and W are established.

![Figure 3.3. The initial structure and the target structure](image)

![Figure 3.4. The initial configuration, target configuration, all configurations and reconfiguration paths.](image)

From the above reconfiguration plan as shown in Figure 3.4 four different new network configurations \((C_n)\) will be created as follows:

1. \(C_1\): Represents initial configuration with connection between F and V.
2. \(C_3\): Represent initial configuration with connection between F and V and connection between W and M.
3. \(C_2\): Represent initial configuration with connection between S and M.
4. \(C_4\): Represent initial configuration with connection between S and M and connection between F and V.

Second, by using the generated configurations \((C)\) and reconfiguration plan which guide the reconfiguration processes to generate different reconfiguration paths, a set of all possible reconfiguration paths (sequences) should be created. Using the above example, the set of all possible reconfiguration paths can be like: \{\(IC-C_1-C_2-TC\), \(IC-C_3-C_4-TC\)\}. That means two strategies have been generated for the given adaptation process.
**Step 3: Configurations Security Model Generation**

This step is responsible for producing a security model for each produced configuration from step two. Using a security modeling language described in Section 2.2.

Security model generation is not a straightforward process, based on the security modeling language a model generator should be developed, i.e., the development should satisfy the desired security model properties. Several tools have been developed, a number of them have been mentioned in Section 2.2.2, that can be used to generate the security model in this step.

**Step 4: Configurations Security Assessment**

A security level for each configuration is computed in this step. The generated security model from the previous step cannot evaluate the security level of a configuration alone quantitatively. Therefore, an NSM related to the adopted security model should be used for computing security level of the configuration. The property and scope of the adopted NSM should take into consideration, because, for example, some NSM describe security level of a system against only zero-day-attack, other only identify the weakest point of the system, while our foremost intention of this security level calculation is for comparison purpose, therefore the selected NSM should comprehensively describe the security level of the system configuration.

Furthermore, generally, almost NSMs compute the security level of a system according to attacker’s goal(s), while in most cases, more than one attacker’s goals exist in a system. Therefore, an indirect NSM (NSM') with adopted NSM as a base metric should be designed to calculate the overall system security level. In other words, once the NSM of all of the attacker’s goals has been computed, an NSM' based on the adopted NSM should be designed and it should abstract and aggregate the adopted NSM values of all attacker’s goals, then the security level of the whole system will correspond to the developed NSM'. We call the developed NSM' as CNSM where NSM is the name of the base metric and starting C indicates a configuration as a whole. For instance, Number of Paths (NP) metric through computing the total number of paths that an attacker can exert to reach its goal, shows the security level of a system, but in cases that there is more than one attacker's goal in the system, an indirect metric such as CNP must be developed to show the overall security level of the system as a summation of all NPs. For instance, such CNP as below can be designed:

\[
CNP = \sum_{i=1}^{n} NP_i
\]  

(3.1)

where \( n \) is a total number of attacker’s goal in the system, \( NP \) is the total number of attack path for \( i^{th} \) attacker’s goal.

Besides, there are NSMs that directly indicates the overall security level of a system, in such case, an indirect metric does not require. For instance, Weakest Adversary metric and Network Compromise Percentage (NCP) as described in Section 2.3.4.
Step 5: Reconfiguration Paths Security Evaluation

This step describes the last step of the framework. As shown in Figures 3.1 and 3.2 the framework uses the security level values for each configuration which was obtained from the previous step and also employs the set of reconfiguration paths from step two. The primary objective of this step is to calculate the security level for each reconfiguration path individually and then sort them based on this calculation from the most secure reconfiguration path to less secure in order to provide the most secure reconfiguration plan.

It is worth remarking that we do not calculate the security level for the last configuration in a path, but we calculate it for the reconfiguration path as a whole which describes the combined security level for all configuration in the path.

Given the configurations metric (CNSM) and reconfiguration paths from step four and step two, respectively, we proceed to design indirect metrics for calculating the security level of a reconfiguration path as a whole. The designed indirect metric is called PNSM metric, where NSM is the name of the metric from step four and the starting P indicates reconfiguration path. As it is clear, the CNSMs calculated in step four shows configuration security level, but a PNSM indicates the security level of a reconfiguration path as a whole.

Additionally, while we have PNSM, now easily an indirect metric, defined for a reconfiguration plan, based on PNSM can be designed to compute the security level for the reconfiguration plan as a whole that consists of all the reconfiguration paths. We call this metric as RNSM where NSM is the name of the base metric that measures security level of the reconfiguration paths, and the starting R indicates reconfiguration plan.

Furthermore, we can summarize the description of the configuration metrics, reconfiguration paths metrics and reconfiguration plan metrics by the following functions, respectively:

\[ CNSM = f(NSM, C) \]  
(3.2)

where \( f \) is a function to calculate the metric CNSM for a configuration \( C \) by using an NSM.

\[ PNSM = g(CNSM, P) \]  
(3.3)

where \( g \) is a function to calculate the metric PNSM for a reconfiguration path \( P \) by using a metric CNSM introduced in Eq. (3.2).

\[ RNSM = h(PNSM, L) \]  
(3.4)

where \( h \) is a function to calculate the metric RNSM for a reconfiguration plan \( L \) by using a metric PNSM introduced in Eq. (3.3).

In conclusion, the General Framework describes the required input data and processes for assessing the security risks of reconfiguration strategies of a reconfiguration plan without any technical detail. Using technologies are left to developers, as it can be observed that each step can be seen as a business-logic of the framework; hence, each step can utilize different approaches, technologies, and tools. Utilizing different approaches in any steps can directly effect on one or more dimensions of the framework’s quality.
3.2.2. Instantiated Framework

In this section, we instantiate the general framework introduced in Section 3.2.1 as shown in Figures 3.5 and 3.6.

Figure 3.5. Shows the Instantiated Framework for secure reconfiguration plan.

Figure 3.6. High-level Architecture of the Instantiated Framework.
**Step 1: Data collection via manual means**
In the first step of our framework data collection is carried out by manually extracting and specifying the required information for the steps that follow. We gather security-related information regarding the (self-) adaptive system: components, links, and services, as well as the configuration, existing vulnerabilities and deployment. Then the information is transformed in a format that is suitable to be used by the model generation tool used henceforth (MulVAL). Finally, the step terminates by creating a reconfiguration plan (RCP) based on the initial configuration (IC) of the system.

**Step 2: Configurations and Reconfiguration Paths Generation**
Given IC and RCP from the previous step, we will proceed to generate all possible configurations and all possible reconfiguration paths. We have broken down the processes:

1. Specify the RCP.
2. Converting RCP into an RCP Tree.
3. Traverse the RCP Tree to retrieve all possible paths.
4. Generating all possible configurations.

Let us discuss each process below:

1. **RCP:** We can expect the input RCP to be syntactically and semantically in a format described as follows:

   **Syntax definition of RCP:** Refers to the form (structure) and grammar of the specified format of RCP. By using Context-free Grammars, the accepted syntax of a reconfiguration plan can be described by the grammar $G = (T, N, P, S)$ where

   - $T = \{ \text{con}(\text{hostID}, \text{hostID}), \text{dis}(\text{hostID}, \text{hostID}), \text{del}(\text{hostID}, \text{hostID}), \text{add}(\text{hostID}), ; , + , ( , ) \}$ Tokens in our RCP,
   - $N = \{ E \}$ is a set of non-terminal,
   - $P = N \rightarrow ( N \cup T )^*$ is a set of productions,
   - $S = E$ is the start symbol.

   and the set of productions, $P$, are defined as

   1. $E \rightarrow \text{con}(\text{hostID}, \text{hostID}) | \text{dis}(\text{hostID}, \text{hostID}) | \text{del}(\text{hostID}) | \text{add}(\text{hostID})$
   2. $E \rightarrow ( E )$
   3. $E \rightarrow E ; E$
   4. $E \rightarrow E + E$

   **Semantics of RCP:** There are four variables in the desirable RCP: con(\text{hostID}, \text{hostID}), dis(\text{hostID}, \text{hostID}), add(\text{hostID}) and del(\text{hostID}) that indicates the type of events which described in Section 3.2.1. The next variables are braces used to indicate aggregation (i.e. grouping) and creating
groups or clarifying the order that operations are to be done in the reconfiguration processes.

Lastly, the symbol ; indicates a sequential process, like the logical AND. Likewise, the symbol + imitates the logical OR operation.
A desirable RCP format can be as follow:

\[
((\text{con}(H1,H8) + \text{con}(H3,H4)) ; \text{con}(subnet,H3))
\]

As we can observe this function consists of various sub-function in the format of \(f(X,Y)\), where X and Y are hosts, and \(f\) indicates the type of the primitive reconfiguration action.
In further sections, we will denote all unique primitive reconfiguration actions with the symbols A, B, C, etc. Therefore, the above reconfiguration plan can be rewritten as:

\[
[A + B] ; C
\]

2. **Converting RCP into an RCP Tree:** We will now create an RCP Tree based on the specified reconfiguration plan. We define the RCP Tree \(T\) as:
\[
T = \{N, Q\}, \text{ where } N \text{ is all the non-terminal nodes and } N \subseteq \{; , + \}, \text{ and } Q \text{ is all the terminal nodes, where } Q \subseteq \{R\}, \text{ and } R \text{ is the set of all the primitive reconfiguration action in the specified reconfiguration plan.}
\]
For the above reconfiguration plan, the RCP Tree, \(T\) will be: \(T = \{N, C\}, \text{ where } N \subseteq \{; , + \}, \text{ and } Q \subseteq \{A, B, C\}.\) The corresponding RCP Tree for this reconfiguration plan is shown in Figure 3.7.

![Figure 3.7. Demonstrates the RCP Tree with the locations of “;” and “+” operators, and primitive reconfiguration action.](image)

Algorithm 3.1 is the algorithm to convert a reconfiguration plan into an RCP Tree:
Algorithm 3.1: Generate an RCP Tree

**INPUT:** A reconfiguration plan. (RCP)

**OUTPUT:** The RCP Tree (RCP_Tree)

// Pre-Define Functions:
// next() : Go to the next item in the list
// prev() : Go to the previous item in the list
// insert(X) : Insert item X in the list before the current item
// list.last() : Access the last item in the list
// list.remove(X,Y,...) : Remove items X,Y,etc. from the list
// list.prefix() = Convert current format to the prefix format
// list.length() = Returns the number of items in the list

list = RCP
list = list.prefix()
op = { ; , + }

while list ≠ ϕ do
    c = list.last()
    while c ∉ op do
        c = c.prev()
    end
    x = c;
    c = c.next()
    y = c
    c = c.next()
    z = c

    if list.length() > 3 then
        c = c.next()
    end

    subtree.node = x;
    subtree.leftChild = y;
    subtree.rightChild = z;
    list.remove(x,y,z);

    if list = ∅ then
        RCP_Tree = subtree;
        return RCP_Tree;
    end

    c.insert(subtree);
end

In the above algorithm, we first convert the reconfiguration plan into the prefix format and then sequentially access each element from the tail until we reach the first operator. We then pop this operator and the proceeding
two items out of the list and then create a subtree with the operator as a node and the two items respectively as left and right children. This subtree will then be inserted back into the list at the same place from where these items were popped out. This process continues until there is just one item left in the list. This item will be the final RCP Tree.

3. **Retrieval all possible paths:** We will now recursively traverse the generated RCP Tree for a reconfiguration to retrieve all the possible reconfiguration paths. The traversal will be depth-first post-order according to the following algorithm:

### Algorithm 3.2: Retrieve all reconfiguration paths.

**INPUT:** An RCP Tree  
**OUTPUT:** All possible reconfiguration paths.

Algorithm find\_paths (tree)  
<table>
<thead>
<tr>
<th>if (tree not empty)</th>
</tr>
</thead>
<tbody>
<tr>
<td>if (left sub-tree is a leaf) then</td>
</tr>
<tr>
<td>S1 = left leaf</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>S1 = find_paths (left sub-tree)</td>
</tr>
<tr>
<td>end-if</td>
</tr>
<tr>
<td>if (right sub-tree is a leaf) then</td>
</tr>
<tr>
<td>S2 = right leaf</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>S2 = find_paths (right sub-tree)</td>
</tr>
<tr>
<td>end-if</td>
</tr>
<tr>
<td>if (root node is +) then</td>
</tr>
<tr>
<td>return {S1 ∪ S2}</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>for each X ∈ S1</td>
</tr>
<tr>
<td>for each Y ∈ S2</td>
</tr>
<tr>
<td>result = {XY ∪ result}</td>
</tr>
<tr>
<td>end-for</td>
</tr>
<tr>
<td>end-for</td>
</tr>
<tr>
<td>return result</td>
</tr>
<tr>
<td>end-if</td>
</tr>
</tbody>
</table>

4. **Generating all possible configurations:** Now that we have all the reconfiguration paths from the reconfiguration plan, we can identify all the unique configurations. And generate the required file for each configuration which can satisfy the format which is accepted in the next step, which is MulVAL in this case.
Step 3: Configurations Security Model Generation

This step is responsible for generating a security model for each configuration created in the previous step. It is worth mentioning that choosing a proper security modeling technique is the most affecting factor of the quality of the framework. Therefore, after an in-depth literature survey and studying a different security modeling language as a several of them described in Section (2.2.1), we decided to choose Attack Graphs as the framework's security modeling technique. After determined attack graphs as our modeling language, we need a generator to generate attack graphs automatically.

The main reasons for selecting attack graph as our modeling approach is the fact that an attack graph can encode the causal relationship among vulnerabilities, it can capture the dependencies and independences between vulnerabilities. In other words, an attack graph describes a system as a set of vulnerabilities and their interdependencies. Also, it presents whether given important resources in a system can be compromised throughout multi-step attacks i.e. it shows all attack paths with attack steps that an adversary can take to compromise a specific resource.

There exists a large and growing literature on methods and tools of attack graphs generation as shortly described in Section 2.2.2 — those in which the effect of some causal variables like open-source or commercialized, availability, the ability of automatic generation, and so forth, have affected our decision to choose a tool.

Finally, we decided to use MulVAL as the framework's attack graphs generator. Furthermore, MulVAL generates the attack graphs in different formats, including (pdf, image, XML, etc.), these formats can be used for visualizations and demonstrating security-related issues in a system. Also, the NSMs related to attack graph automatically can be achieved by the generated attack graphs.

Step 4: Configurations Security Assessment

In this step, we compute the overall security level of each configuration. Given attack graph for each configuration, we will proceed to use some NSMs defined over attack graph to calculate the overall security level of each configuration. After an in-depth literature survey on NSMs defined for attack graphs as a few of them has been shortly reported in (See Section 2.3.4). We decided to use the following security metrics and denote them as:

- Attack graph probabilistic metric as AGP
- Number of attack paths as NP
- Mean of attack path lengths as MPL
- Shortest attack path as SP

The security grade of each configuration will be measured based on the mentioned metrics separately, i.e. each configuration will have four different security level indicators.

In addition, this step generates detailed information about attack paths; for each attack path in an attack graph, we store the attack steps by the name of the nodes along the path. Let us call it as Attack Path Detail (APD). For instance, the sample attack graph shown in Figure 2.5 one attack path with five attack steps is as follow:
1. `attackerLocated(internet)`
2. `RULE 22 (Browsing a malicious website)`
3. `accessMaliciousInput(commServer,victim_1,windows_2000)`
4. `RULE 3 (remote exploit for a client program)`
5. `execCode(commServer,user)`

Then, we give a unique ID to each attack step ($S_i$) and then a path ($P_i$) is defined as the sequence of attack step IDs that exist along the path. For instance, given the above attack path, we call the path as: $S_1 S_2 S_3 S_4 S_5$.

Observe that the description of a node in the MulVAL generated attack graphs consists of three segments: node ID, node type and node AGP value. For instance, the fourth attack step in the above attack path originally looks as shown below. This attack path is also shown in Figure 2.5.

**22: RULE 3 (remote exploit for a client program):** 0.512

where 22 is the node ID, RULE 3 (remote exploit for a client program) is node type and 0.512 is the AGP value for this node.

In the following section, we introduce a few metrics based on the base NSM metrics defined for attack graphs as mentioned above. The new metrics are CAGP, CNP, CMPL, and CSP.

### CAGP

In [77], the AGP metric is defined for an attacker’s goal and not for the attack graph as a whole, therefore we need to design a new metric based on the original AGP introduced in [77] to measure the security of a configuration. Let us call the new indirect metric as CAGP (Configuration AGP). Since AGP is a probabilistic metric and AGP of different attacker's goals are independent, and the purpose of this new metric is for comprehensively showing the overall security level of a configuration which then will be used for comparisons, therefore, we design the CAGP as the average of AGP value of all attacker's goals, as below:

$$\text{CAGP} = \frac{1}{n} \sum_{i=1}^{n} AGP_i$$

(3.5)

where $n$ is the total number of attacker’s goals and $AGP_i$ indicates the AGP of the $i^{th}$ attack goal.

### Path Metrics (PMs)

NSMs defined based on the characteristics of attack paths are regarded as path metrics (PM). This kind of metric is focused on information such as size and/or the number of attack paths. Therefore, we measure the security level of configurations based on NP, MPL, and SP together. Furthermore, the scope of all three metrics is attacker’s goal, in other words, these metrics measure security level of a system based on one attacker’s goal, but we need a metric that indicates the security level of the whole system. For this reason, we need to design indirect metrics CNP (Configuration NP), CMPL (Configuration MPL), and CSP (Configuration SP) with the following properties:
• With NP, MPL, and SP as the basis metrics, respectively.
• They are defined for configurations.
• Base and indirect metric are defined on the same information i.e., (size, number, mean, summation, min, max) of attack paths.
• With the nearest possibility of comprehensively measuring the overall security level of a system.

**CNP**
With respect to the above properties, the CNP can be described as follow:

\[
CNP = \sum_{i=1}^{n} NP_i
\]  
(3.6)

where \( n \) is a number of attacker’s goal in the network, NP is the total number of attack path for \( i^{th} \) attacker’s goal. The metric is defined as a summation of all attack paths that exist in a configuration to follow the same mathematical function as NP which is the sum of all attack paths for a given attackers’ goal.

**CMPL**
With respect to the above properties, the CMPL can be described as follow:

\[
CMPL = \frac{1}{n} \sum_{i=1}^{n} MPL_i
\]  
(3.7)

where \( n \) is a number of attacker’s goal in the system, MPL\(_i\) is the mean attack path length for \( i^{th} \) attacker’s goal. The metric is defined as a mean of MPLs for a given configuration to follow the same mathematical function as MPL which is the mean of all attack paths length for a given attackers’ goal.

**CSP**
With respect to the above properties, the CSP can described as follow:

\[
CSP = \min (SP_1, SP_2, ..., SP_n)
\]  
(3.8)

where \( n \) is a number of attacker’s goal in the system, and SP\(_i\) is the shortest path length for \( i^{th} \) attacker’s goal. The metric is defined as a min of SPs for a given configuration to follow the same mathematical function as SP which is the smallest attack path length for a given attackers’ goal.

**Step 5: Reconfiguration Paths Security Evaluation**
The outputs of this step are the main contribution of the framework, which identifies the security level values of all available reconfiguration strategies (paths) for a given reconfiguration plan. Given the reconfiguration paths along with CAGP, CNP, CMPL, CSP obtained in the previous step, we will proceed to develop indirect metrics for reconfiguration
First, let us name the new reconfiguration path-level indirect metrics that are driven from the configuration-level metrics that include PAGP (Path AGP), PNP (Path NP), PMPL (Path MPL) and PSP (Path SP).

### i. Path Attack Graph Probabilistic (PAGP) metric

Since CAGP provides us with the probabilities of the security level of a single configuration that indicates the probabilities of attack success of each configuration, we have adopted a probabilistic approach to calculating the combined security level of a path.

Let us assume that we have a total of three configurations in our path. Let’s name them A, B and C in that order. Observe that, PAGP can be extrapolated for any number of configurations in a path.

Before we calculate PAGP, we introduce a few conventions. Let X and Y be two arbitrary events. Then,

1. $\bar{X}$ denotes all the events in the event space without X.
2. $P(X)$ denotes the probability that the event occurs.
3. $P(\bar{X})$ denotes the inverse probability of event X, i.e. the probability that event X does not occur. Note that $P(\bar{X}) = 1-P(X)$ and X and $\bar{X}$ are mutually exclusive.
4. $P(X \cap Y)$ denotes the probability that both X and Y occur.
5. $P(X \cup Y)$ denotes the probability that any one of the X or Y occurs.

Although the probability that a particular configuration in a path is successfully attacked depends upon its previous configurations (which will be proved later), however, contrary to the popular belief, they are mutually exclusive (i.e. they are independent and occurrence of one cannot affect the occurrence of the other) of each other. This can be easily proved by reductio ad impossibilem (“Proof of a proposition which involves demonstrating that its negation entails a contradiction; since a contradiction cannot be true, whatever entails it cannot be true.” [78]) as follows:

We consider $P(A)$, $P(B)$ and $P(C)$ as the probability that an attack is successfully carried out in A, B and C, respectively, when they are in a path. Let us also assume that these three events are not mutually exclusive (in other words, they are mutually implicated). The Venn diagram in Figure 3.8 illustrates these events:

![Venn Diagram](image)

Figure. 3.8 Venn diagram shows mutually implicated between three events.
Since they are mutually implicated, we have many intersected events between them. For these events to be mutually implicated, the following conditions must hold true:

1. \( P(A \cap B) > 0 \)
2. \( P(B \cap C) > 0 \)
3. \( P(C \cap A) > 0 \), and
4. \( P(A \cap B \cap C) > 0 \)

Let us examine them:
The first condition says that there is the slightest possibility (since we counter argue it is greater than 0) that both A and B are successfully attacked. However, since A, B and C occur sequentially, and if any one of the configurations is successfully attacked, considering any further configurations is redundant. Our sole objective is to calculate the probability that an attack occurs in a reconfiguration path. Therefore, if an attack happens in any intermediate configuration, it is evident that the probability of attack-success for that particular path is one. As a result, whatever happens in the succeeding configurations in the path has absolutely no value. Moreover, if a particular configuration is successfully attacked, it clearly implies that previous configurations were not attacked, or they (attackers) were unsuccessful. We can apply the same argument for the first three conditions above. Now that these three conditions are false, the fourth condition is falsified automatically. Therefore, we can confidently assume that all the above four conditions are false. Hence, we have contradicted our initial assumption that the three events are mutually implicated. It proves that all the three events are mutually exclusive.

Moreover, we know that mutually exclusive events cannot happen at the same time [79], and since events A, B and C occur sequentially, i.e. event B occurs after A finishes and event C occur after B finishes clearly implies that these events do not occur at the same time. Furthermore, events \( \overline{A} \), \( \overline{B} \) and \( \overline{C} \) occur in a different time frame, therefore they are mutually exclusive as well. We can summarize the above discussion as follows:

Events A, \( \overline{A} \), B, \( \overline{B} \), C and \( \overline{C} \) are all mutually exclusive from each other. Rule (3.1)

Now, based on our assumed scenario, let us design another Venn diagram which suits our situation.

![Venn diagram](image)

Figure 3.9. Venn diagram shows the dependency between configurations, individually and when they are in a path.
In the above Venn diagram Fig. 3.9, A, B and C are the events that an attack is successful at configuration A, B and C respectively, when they are independent, i.e. they are not along any path. The rectangle delineates our sample space (Ω) that includes all the events for a path, which are:

1. Successful attack at A (Shape A)
2. Successful attack at B (Shape Bi)
3. Successful attack at C (Shape Ci)
4. Not attack at all (Shaded (yellow) Region)

We can observe that there is nothing common between A, B and C which was proved above. Moreover, parts of B (Bo) and C (Co) are outside the sample space, which means that the probability that these configurations are successfully attacked decreases as soon as they become the part of a path. We can prove it below:

We know that P(A), P(B) and P(C) are the probabilities that an attack is carried out successfully at A, B and C respectively when they are not in any path. Let’s consider the following sample scenario:

1. \[ P(A) = 0.2 \]
2. \[ P(B) = 0.4, \text{and} \]
3. \[ P(C) = 0.3 \]
4. Let us also assume there are a total of 10 possible attacks a1, a2, a3, ..., a10.
5. a1 and a3 are successful in attacking A
6. a1, a4, a7 and a9 are successful in attacking B
7. a5, a7 and a10 are successful in attacking C

We have assumed 2 successful attacks at A, 4 at B and 3 at C based on their assumed probabilities. Now, if A, B and C become part of a path in that order, although B is vulnerable by attack a1, there is no way B can be attacked by a1 in the path, as a1 will successfully attack A, and the B will not exist anymore because the path is already attacked. This argument is valid for event C as well. However, if there are no common attacks between any events, Bo, Co and Ao will be \( \emptyset \), or a null set, i.e. that particular event will be completely inside the sample space (Ω). Further, as we know there are no preceding configurations with A, we can assert that Ao is a null set, or \( \emptyset \), i.e. shape A is completely inside the sample space. However, for consistency in the illustration, we have shown part of A to be outside the sample space, with Ao = \( \emptyset \).

Now that we have established A, B and C are mutually exclusive, total probability that a path is successfully attacked is the probability of all successful attacks happening in the sample space, which is:

\[
P(A_i \cup B_i \cup C_i) \tag{3.9}
\]

However, since A, B and C are mutually exclusive, any subsets of them will also be mutually exclusive. Moreover, we know that for two events X and Y:

\[
P(X \cup Y) = P(X) + P(Y) \tag{80} \text{ if } X \text{ and } Y \text{ are mutually exclusive.} \]  

Rule (3.2)
Therefore, based on Rule 3.1 and Rule 3.2, Equation 3.9 can be re-written as:

\[ P(A_i \cup B_i \cup C_i) = P(A_i) + P(B_i) + P(C_i) \] [79]

Let us calculate \( P(A_i) \):

Based on Figure 3.9, we can say that

\[ P(A_i) = P(\Omega \cap A) \]

where \( \Omega \) is the sample space. This can also be illustrated as Figure 3.10:

![Figure 3.10](image)

Figure 3.10. Common part between \( \Omega \) and A.

where \( A_i \) is the shaded region, which illustrates the event that A was successfully attacked when inside a path. However, the above equation cannot be solved due to insufficient data. Nevertheless, \( A_i \) can also be represented as Figure 3.11:

![Figure 3.11](image)

Figure 3.11. Event \( \sigma \) (Shaded in purple). Event \( A_i \) can also be represented as the intersection between event A and event \( \sigma \).

This is because the intersection between event \( \sigma \) and event A is still \( A_i \), where event A is the event that a successful attack can happen at A (when it is not in the path).

Event \( \sigma \): As can be seen from Figure 3.11, event \( \sigma \) describes all the events in the sample space, excluding events B and C. In other words, \( \sigma \) describes all the events when there is no attack on B (\( \bar{B} \)) and there is no attack at C (\( \bar{C} \)), mathematically:

\[ \sigma = \bar{B} \text{ AND } \bar{C} \] (3.10)
Therefore,
\[ \sigma = P(\overline{B} \text{ AND } \overline{C}) \]  \hspace{1cm} (3.11)

We know that
\[ P(X \text{ AND } Y) = P(X) \times P(Y) \iff X \cap Y = \emptyset \] \hspace{1cm} Rule (3.3)

So, using Rule 3.3 and Rule 3.1, we can rewrite Equation 3.11 as:
\[ \sigma = (\overline{B}) \times (\overline{C}) \] \hspace{1cm} (3.12)

Now, describing event \( A_i \) based on Figure 3.11:
\[ A_i = A \cap \sigma \]
Therefore,
\[ P(A_i) = P(A \cap \sigma) \] \hspace{1cm} (3.13)

We know that for two events X and Y:
\[ P(X \cap Y) = P(X) \times P(Y) \] \hspace{1cm} Rule (3.4)

Therefore, Equation 3.13, based on Rule 3.1 and Rule 3.4 can be re-written as:
\[ P(A_i) = P(A) \times P(\sigma) \] \hspace{1cm} (3.14)

Using Equation 3.12,
\[ P(A_i) = P(A) \times P(\overline{B}) \times P(\overline{C}) \] \hspace{1cm} (3.15)

This, can also be applied for events \( B_i \) and \( C_i \), thus getting the following equations:
\[ P(B_i) = P(B) \times P(\overline{A}) \times P(\overline{C}) \text{ for } B_i \] \hspace{1cm} (3.16)
\[ P(C_i) = P(C) \times P(\overline{A}) \times P(\overline{B}) \text{ for } C_i \] \hspace{1cm} (3.17)

However, it should be noted that since these events occur in a sequence, for a particular event (configuration in a path), we should only consider the events that occur previously, as future events have not yet occurred, therefore, they have no effect on the current calculation.

Above hypothesis will affect all the equations where we have considered future events as well. Therefore, rewriting Equations 3.15, 3.16 and 3.17 again:
\[ P(A_i) = P(A) \] \hspace{1cm} (3.15')
\[ P(B_i) = P(B) \times P(\overline{A}) \] \hspace{1cm} (3.16')
\[ P(C_i) = P(C) \times P(\overline{A}) \times P(\overline{B}) \] \hspace{1cm} (3.17')

It is worth noting that Equation 3.17 had no change, as it was the last event, and it was affected by all the previous events.

Finally, PAGP will be:
PAGP = P(A_i) + P(B_i) + P(C_i)

Therefore,
\[ PAGP = P(A) + [P(B) \times P(\bar{A})] + [P(C) \times P(\bar{A}) \times P(\bar{B})] \]

Note that this can also be generalized for any number of events happening in sequence:

Let us assume we have \( n \) events (C) \( C_1, C_2, C_3, \ldots C_{n-1}, C_n \). All of them occurring sequentially, then PAGP, according to our calculations will be:

\[ PAGP = P(C_1) + [P(C_1) \times P(C_2)] + [P(C_1) \times P(C_2) \times P(C_3)] + \ldots + [P(C_1) \times P(C_2) \times P(C_3) \times \ldots \times P(C_{n-1}) \times P(C_n)] \]

\[ PAGP = P(C_1) + \sum_{i=2}^{n} \left[ \left( \prod_{j=1}^{i-1} P(C_j) \right) \times P(C_i) \right] \] (3.18)

where \( n \) is a number of configurations in the given reconfiguration path, \( P(C_1) \) denote the initial configuration.

**ii. PNP**

While the base NP *separately* counts the total number of attack paths of each attacker’s goal that exist in the attack graph of a given system, however, CNP counts the total number of attack paths for all attacker’s goal that exist in the attack graph of a given system. Nonetheless, for not deviating the base theory of computation of NP and CNP, and to apply the same method, we define PNP as the total number of attack paths that exist in a reconfiguration path of a given reconfiguration plan.

In other words, the design of PNP has to follow the same computational theory as its base metrics (NP and CNP); otherwise, it will be a different security metric, while this metric evaluates the security level of a reconfiguration path by describing and counting the total number of ways an adversary can compromise it. Formally, this metric is provided by:

\[ PNP = \sum_{i=1}^{n} CNP_i \] (3.19)

where \( n \) is a number of configurations in the given reconfiguration path, and \( CNP_i \) is the total number of attack path for \( i^{th} \) configuration.

**iii. PMPL**

Applying the same theory of designing PNP, we design PMPL based on the same computational theory as MPL and CMPL, we formally define the PMPL as follow:

\[ PMPL = \frac{1}{n} \sum_{i=1}^{n} CMPL_i \] (3.20)
where \( n \) is a number of configurations in the given reconfiguration path, and CMPL\(_i\) is the mean attack path length for \( i^{th} \) configuration.

\textbf{iv. PSP}

Applying the same theory of designing PNP, we design PSP based on the same computational theory as SP and CSP, we formally define the PSP as follow:

\[
PSP = \min (CSP_1, CSP_2, ..., CSP_n)
\]

(3.21)

where \( n \) is a number of configurations in the given reconfiguration path, and SP\(_i\) is the shortest path length for \( i^{th} \) configuration.

\textbf{v. DPNP}

By using APD from the previous step which includes the detailed information about the NP metric, we will proceed to develop an indirect metric with NP as a basis metric to measure the security level of a reconfiguration path. Before we start designing this metric, two points should be taken into consideration:

A. Definition of NP: "This metric represents the number of distinct ways an adversary can compromise a given network asset." [39]. Here we focus on the word "distinct".

B. Designing NP and CNP satisfy the principle of the NP definition above, but when we designed PNP, we did not satisfy the principle. In other words, there is a high likelihood of counting the same attack-path in different configurations in the same reconfiguration path.

As per the two observations described above, we design an indirect metric which counts the distinct number of attack paths in the whole reconfiguration path by using the APDs. Let us call it DPNP, where the starting D indicates distinct.

Consider a scenario: a reconfiguration path that consists of three configurations (A-B-C), each with CNP = 3, and APD\(_A\) = \((e_1, e_2, e_3)\), APD\(_B\) = \((e_1, e_2, e_4)\) and APD\(_C\) = \((e_1, e_7, e_8)\), where \( e_1, e_2, ... e_n \) indicate different attack paths.

\[
PNP = \text{Sum} (\text{CNP}_A, \text{CNP}_B, \text{CNP}_C)
\]

= 9

\[
DPNP = \text{Count-Distinct} (\text{APD}_A, \text{APD}_B, \text{APD}_C)
\]

= \text{Count-Distinct} \((e_1, e_2, e_3, e_1, e_2, e_4, e_1, e_7, e_8)\)

= 6

where Count-Distinct is a function which counts the number of distinct attack-path inside the set.

It is worth mentioning that the common attack paths in a reconfiguration path are found and defined based on the exactly the full sequence of the attack paths i.e. attack path from attacker's start point to attacker's goal. Note that the overlaps in the attack paths, if any, between configurations in a reconfiguration path, is not
considered. The process of defining an attack path has been described in Section 3.2.1 step 4.

**Security Assessment of a reconfiguration plan:** Now that we have calculated the security level for a reconfiguration path by designing five different metrics described in step five, we can also calculate the security level for the reconfiguration plan as a whole that consists of all the paths. In this work, the most interesting and concentrating metric is PAGP, therefore for this purpose, we only design a metric based on PAGP. Let us name it as RAGP, where the starting R indicates a Reconfiguration Plan as its scope.

Example 3.2: A reconfiguration plan has three distinct reconfiguration paths. Let name the PAGP value for these three paths as PAGP_1, PAGP_2 and PAGP_3. To calculate the RAGP, we have to assume that the system is equally likely to follow any configuration path. Based on this assumption, we can say that,

\[
\text{RAGP} = \frac{1}{3}(\text{RAGP}_1) + \frac{1}{3}\text{RAGP}_2 + \frac{1}{3}(\text{RAGP}_3)
\]

This can also be generalized as follow:

\[
\text{RAGP} = \sum_{k=1}^{P} \frac{1}{P} \ast (\text{PAGP})_k
\]

where \(P\) is the number of reconfiguration paths in the given reconfiguration plan, and PAGP_\(k\) denotes PAGP value of \(k^{th}\) reconfiguration path.

### 3.2.3. Computational Complexity of Instantiated Framework

In this section, we discuss the computational complexity of our instantiated framework as specified in section 3.2. However, because of the time limitation, and due to the broadness of the thesis, the computational complexity of our method could not be formally provided, however, we can still make some observations and expositions briefly as follows.

The main time-consuming tasks of the framework are attack graph generations and PNSMs computations for each configuration. The computational complexity of attack graph generation is quadratic time \(O(n^2 \log(n))\) [63, 81], where \(n\) is the numbers of vulnerable softwares (components) in the targeted system. Table 3.1 reviews and compares the various methods for attack graph generation including the one we adopted and their computational complexity. We can observe that our chosen approach is nearly the best amongst the others with respect to the time complexity.

<table>
<thead>
<tr>
<th>Papers</th>
<th>Methods</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>[82, 83]</td>
<td>Full attack graph</td>
<td>O(n!)</td>
</tr>
<tr>
<td>[60]</td>
<td>Attribute attack graph</td>
<td>O(n^6)</td>
</tr>
<tr>
<td>[84]</td>
<td>Host-based attack graph</td>
<td>O(n^3)</td>
</tr>
</tbody>
</table>
Logical attack graph $O(n^2 \log(n))$

Predictive multi-prerequisite attack graph between $O(n \log(n))$, $O(n^2 \log(n))$

Table 3.1. Computational complexity comparison. Taken from [40].

The next main time-consuming task is the computations of PNSMs. As mentioned before, logical attack graphs are very useful as a simple depth-first-search traversal identifies and calculates all possible attack paths. Thus, the computational complexity of this task is $O(V + E)$ where $V$ is vertice and $E$ is the edge in attack graphs. We get this complexity by virtue of the fact that we are visiting each node only once.

**The framework's applicability for the (self-) adaptive system:** Regarding the concept of the MAPE-k loop, the key requirement for the (self-) adaptive systems are cost and timeliness as the system has to monitor (M) and analyze (A) system changes at run-time and take appropriate actions for planning (P) and executing (E) the adaptation at a reasonable cost and in a timely manner. In this thought, if the required adaptation is due to security threat detection, the system might be compromised and resulting in a failure because of the time consumption in producing the most secure strategy for the given plan. In this case of adaptation, we can make two suggestions:

1. Train the system for updating its base knowledge (k) to include the security measurements (all PNSMs) of the calculated configurations. As a result, the total computations of attack graph generations and PNSMs computations will be replaced by a simple lookup table mechanism or another simple approach.

2. Making use of the concept of 4Ws model [85]. According to the concept, the systems can change either their **structure** and/or their **behavior** through adaptability. The characteristics of the changes are defined by four dimensions as follows:

   1. Why there is the need to change?
   2. What does (not) change?
   3. When does the change happen?
   4. What/Who manages the change?

The first dimension (*why*) describes the need for the change. This change is always performed to meet the system’s requirements. For example, to improve performance, reliability, security etc. From this point of view, the (self-) adaptive system should consider the reasons behind the need for change, as an exemplification, if the need is solving security issues, performing the secure reconfiguration plan (our approach) should be analyzed (i.e. executing secure reconfiguration plan or not) or directly rejected.

The second dimension (*What*) describes what part of the system is influenced by the change i.e. components can get in and out, new connectors between components can be established and eliminated. We do not consider this dimension for applicability of our approach as this dimension refers to the
architectural models and generates one of the inputs (reconfiguration plan) into our framework.

The *when* dimension takes into account the moment during the system's life cycle in which the change happens, without considering whether the change occurs statically (offline) or dynamically (at run-time). This dimension does not affect the applicability of our approach as the approach is applicable during both situations (static and dynamic execution environments).

*What/Who* describes the mechanisms to accomplish the change. It can be either manually by a configuration manager or by the system itself. The framework is applicable to both cases as it can be managed by the configuration manager and it can automatically generate secure reconfiguration plan by the system itself.

### 3.3 Reliability and Validity

**Reliability** means how well others are able to get the same result as we got if they replicate our work. First of all, our primary work is proposing a framework generates the most secure strategies for a given reconfiguration plan, and the framework consists of five steps, each step has its responsibilities which basically get some specified input and generate some specified output based on specified approaches, if the framework will be replicated by others, if any step deviates these specifications it cannot be counted as our proposed instantiated framework. Therefore, we can claim that the reliability of Instantiated Framework is provided. However, the reliability of proposed General Framework is not given because of the generality of each step, especially, the step three, four and five which different threat modeling approaches and NSMs can be used and different reconfiguration path metrics can be designed which will sequel different results.

**Validity** is related to whether the results produced from the data are valid, for this purpose there exist various views on Validity and the most important ones are: Construct Validity, Internal Validity, and External Validity.

**Construct Validity** is concerned with whether the researcher leaves room for diverse definitions of theoretical constructs observed in the literature, by the user of the study. To bypass such threats to Validity, we utilize the most widely-used and well-known terms in this domain of research and perform satisfactory explanation on the terms we use.

**Internal Validity** is involved with the level of the precision by which the results represent reality, and the results reflect the collected data. To strengthen the internal validity of our work, we provide the instantiated framework with specified approaches to be used. The most effective factor which affects the internal validity is designing the reconfiguration path security metrics in step 5, because of the domain subjectivity (designing indirect metric or create new metric) it is very difficult to be completely objective about the result of the framework. In addition, the accuracy of data collection about the initial configuration of a system affects the result-accuracy of the framework.

The last factors that influence the internal validity are the way which we model the reconfiguration path may affect the internal validity of our defined framework.
because we model each configuration in a reconfiguration path in sequential manner i.e. not one single model for the whole reconfiguration path. Therefore, we could not consider the effects of the attacker’s effort in vulnerating a configuration in a reconfiguration path to the other configuration. And there may exist unknown vulnerabilities which could not be considered because it was out of our knowledge while modeling.

**External Validity** is describing whether it is safe to make generalizations based on the outcomes of the research at hand. Since we had to limit the scope of our work; to select one security modeling language and a number of NSMs while there exist many other security modeling languages and NSMs, it is difficult to generalize our results. However, the readers have to assess how well the outcomes of this work transfers to other cases. Hence, to improve External Validity, we provide as much detailed information as possible about the domain of this work. In this way, the readers may make more accurate observations of whether our findings are applicable and transferable to other cases.
4. Implementation

We have implemented a prototype tool in Java to support our proposed instantiated framework. The tool has a main component called controller to manage data transformation between all five steps of the framework. The following briefly describes the implementation of each step and the controller:

Figure 4.1. Shows the implementation of Instantiated Framework of Secure Reconfiguration Plan.

**Controller:**

Firstly, we focus on describing the implementation of the controller. The controller carries out a set of operations in a particular order by passing a parameter and using the returned data in subsequent operations.

As processes are shown in Figure 4.1, each process of the controller can be described as follows:

1. In the first step, the controller receives (i) the directory path (CD) of the generated file (IC input Mulval) that represent the initial configuration, and (ii) the reconfiguration plan as a string input via a simple User Interface.

2. Next, in the second step, the two values (i) & (ii) are passed as a parameter to the second step, which generates (iii) all possible reconfiguration paths and returns this value as a list of string to the controller. At the same time, back in the second step, unique files are generated for all possible configurations that exist in the reconfiguration paths by using the IC file. These files are named in a given pattern that signifies the changes in this
particular configuration, simultaneously, they are saved in the same directory as the initial configuration (i).

3. Further, in step three, we design a for loop, which iterates for each file that was generated in the CD (i). This loop firstly generates the attack graph via the MulVAL attack graph generator, following which, it calls the NSM and CNSM metrics generator which returns a (iv) list of metrics to the controller. The list includes: CAGP, CNP, CMPL, CSP. Simultaneously, it also generates APD and stores it in a specific directory, let’s call it (v) Attack Path Detail Directory (APDD). Following our previous conventions, we name the APD with the same name as the current configuration. It should be noted that the two steps described above are executed for each of the files (Configurations (InputMulVAL)), as it runs inside the for loop.

4. Now that the controller has all the metric values for each configuration (iv), and the reconfiguration paths (iii), in the fourth step, these values are passed as a parameter to the fifth step to calculate the (vi) security level of all the reconfiguration paths based on each metrics separately. The reconfiguration paths are now sorted based on each metric mentioned above, from the most secure to the least. These sorted reconfiguration paths, along with their corresponding security values are then returned to the controller. It is worth mentioning that the reconfiguration path security generator utilizes the APD files from (v) to generate DPNP metric for each reconfiguration path.

Besides, while we have the PAGP values (vi) for all reconfiguration paths, easily we implemented the defined formula to generate RAGP and the final value is returned to the controller.

Figure 4.2. Class diagram of Instantiated Framework.
The refined class diagram is presented in Figure 4.2 showing all the classes that are required for functioning all five steps of the instantiated framework. Apart from the Controller which has been described before, each class(es) exhibits its own step and can be described as follows:

**Step 1:**
Because we do not have a (self-) adaptive system in place to apply our approach as mentioned before as our limitation, this step has been created manually as follow:

First, we collect necessary security-related information of a (self-) adaptive system as a whole. As a whole means including even such as devices, hosts, subnets which are not yet connected to the current configuration, it will be connected in some future time of the system's lifecycle.

These pieces of information consist of the data which are described in Section 2.2.2.1 point 2 to 6 and in the same format. And we store them in a text file, let us call it InputMulVAL. Second, the current system configuration (connection between hosts) is adding to the InputMulVAL in such format that described in Section 2.2.2.1 point 1. Now InputMulVAL represents the initial configuration (IC) of the system.

Lastly, we create a reconfiguration plan (RCP) for the initial configuration of the system. The RCP stores as a line of string, for example as below:

\[ RCP = [\text{con}(H1,H8) \text{ OR con}(H3,H4)] \text{ AND con}(\text{subnet},H3) \]

The class (InitialConfigurationAndRCPGenerator) representing this step.

**Step 2:**
This step has been carried through by implementing the class (ConfigurationsAndReconfigurationPathGenerator). The first method gets the reconfiguration plan and directory path of the created initial configuration file as a parameter (which was fabricated in step one) and generates a unique file for all possible configurations in the same directory. The second method gets a reconfiguration plan as a parameter and returns a list of all possible reconfiguration paths.

**Step 3:**
The class (AttackGraphGenerator) has been implemented by the tool to generate the attack graphs:

The method in the class takes the absolute directory path of the configuration’s files, one at a time, and the file that represents a particular configuration is read by MulVAL to generate the attack graph. The generated attack graph is in different formats such as pdf, image, text, XML.

**Step 4:**
We have implemented two classes to complete the requirements of this step. The succeeding steps in the class (ConfigurationSecurityLevelCalculatorUsingAGP) should be processed, as illustrated in the class diagram, which is for scoring a configuration by AGP.
The first method generates AGP value for all nodes (including attacker's goal(s)) of the attack graph. This process needs a file called (probAssess) which contains a quantitative risk assessment algorithm based on [77]. It combines the CVSS metrics and the attack graph to measure a probabilistic risk metrics for the given configuration. The output of this method is a file let us call it AG-AGP which contains AGP value for each node of the attack graph. The second method takes the AG-AGP file and extracts the AGP value for all attacker’s goal. The output is a list of attacker’s goals with its associated AGP value. The third method takes a list of AGP values for the attacker’s goals and returns the CAGP value for the given configuration.

The second class is (ConfigurationSecurityLevelCalculatorUsingPM) includes all required methods for computing adopted PMs. Attack graphs generated from step three by MulVAL is in a different format such as text, image, XML. This attack graph in such format is not useful for computing PMs, therefore as a first step we generate a graph data structure by using the attack graph in XML format. Once we have the graph, simply by using a depth-first-search approach, we generate all attack path information such as the number of attack paths, the length of each attack path. The first method (generateNP) takes the graph as a parameter and return a list of attack path number for each attacker’s goal. Then this list passes to the second method (generateCNP) to utilize the designed function for measuring CNP and return the CNP value of the current graph which represents the current configuration. The methods for generating the (MPL, CMPL) and (SP, PSP) function in the same manner.

The seventh method generates APD for each configuration and names the APD file with the same name as the current configuration name. This information will be used by the next step for further computation.

**Step 5:**
The class (SecureReconfigurationPlanGenerator) represent the functions of this step. For the class, we have formal equations for PAGP, PNP, PMPL and PSP which we apply in the first, second, third and fourth method, respectively. Also, we have the list of reconfiguration paths which have been provided from step two. As it is shown in the diagram, we have four methods for the four aforementioned metrics, each method takes the corresponding configuration metrics value with the reconfiguration paths. The results are four lists of reconfiguration paths with corresponding aforementioned security metrics, each list is sorted from the most secure reconfiguration path to less secure.

Lastly, using DPNP formula by the last method in the class, we can compute the security level of a reconfiguration path based on the DPNP by passing the APD’s absolute directory path which has been called as APDD.

To demonstrate the activity for our instantiated framework clearly, we have taken the help of a UML Activity Diagram, as we know, an activity diagram is a graphical representation of an executed set of procedural system activities. This diagram can be referred from the Figure 5.3. Our procedure begins with simultaneously executing two processes: Getting the reconfiguration plan and getting the initial configuration.
Getting the reconfiguration plan: A string that is provided by the given (self-) adaptive system that acts as a rule to guide the whole reconfiguration processes.

Getting the initial reconfiguration: In this process, we obtain the initial configuration of the (self-) adaptive system, which will then undergo several reconfigurations based on the reconfiguration plan that we obtained above.
After the above two processes have been executed, two other independent processes executes simultaneously: Generate all reconfiguration paths and generate all possible configurations.

*Generate all reconfiguration paths:* To generate all possible strategies that a given (self-) adaptive system can undergo. The *reconfiguration plan* is used for this process.

*Generate all possible configurations:* Here we generate all possible configurations of the adaptive system in accordance with the format which is accepted by MulVAL. Note that these configurations are based on the reconfiguration plan, and not appertaining to all possible permutations of the base events that change the system.

After all possible configurations are generated, the next step is to *Generate the attack graph* using MulVAL.

Further, two parallel sequences of activities are simultaneously executed: *Generate graph data structure* and *Generate AGP Values.* Generating Graph Data Structure is followed by *Generating CPMs* and *Generating APD.* These two processes execute simultaneously by using the *Generated Graph Data Structure.* At the same time, *generating AGP values* is followed by *extracting AGP values for attackers’ goals,* with the subsequent process being *generating the CAGP.*

In the above process sequence, after the three activities: *Generating CPMs, Generating APD* and *Generating CAGP* have successfully executed, the framework checks whether there are more configurations to evaluated using the above process. If any, all these configurations undergo the process of *Generating the attack graph* and its succeeding process one at a time. After all the configurations have gone through this process, the *reconfiguration path metrics are generated,* which also uses the *generated reconfiguration paths,* which was executed in the very first step. All the above activities have been described comprehensively in this section.
5. Evaluation

In this section, we demonstrate the utility of our instantiated framework and show its effectiveness through a case study, followed by analyzing the results from the case study. Thereafter, we will discuss our findings and answer our research questions.

5.1. Case Study

Our case study can mainly be categorized into two parts: (i) an initial configuration of a network. (ii) Two scenarios with different reconfiguration plan. It is worth mentioning that although the case study is a networked-based system, but the framework is generic and accepts any type of systems.

1. Initial Configuration IC: Our testbed (as shown in Fig. 5.1) consists of 11 hosts amongst which two of them are not yet connected to the network and attacker’s location is outside of the network who connects through the Internet.

![Figure 5.1. Initial configuration (IC) of the case study.](image)

In this scenario we assumed that all hosts (VPN Server, Citrix Server, CommServer, DataHistorian, Subnet_1, Workstation, File Server, Web Server, Mail Server) lie in the domain of the attacker’s interest, i.e. they are attacker’s goal, barring the two firewalls.

The connections between hosts are clearly shown in Figure 5.1 via the blue arrows. Furthermore, we have assigned at least one vulnerability to each host. For instance, the Mail Server has one vulnerability which has been identified and described based on CVE as below:

**CVE-ID:** CVE-2010-0490

**CVE-Description:** This vulnerability enables remote attackers to execute arbitrary code through vectors associated with the CTimeAction object,
destruction of markup, and the TIME2 behavior, commanding to memory corruption, as known as "HTML Object Memory Corruption Vulnerability". It is related to the use-after-free vulnerability in mstime.dll in Microsoft Internet Explorer 8.

Appendix-1 completely defines the configuration of the test cast, which includes the details of the connections between each host and all the existing vulnerabilities within them. This description is in a format that the attack graph generator accepts.

2. Reconfiguration Plan (RCP): Two scenarios according to the two RCPs are created to transfer the given network shown in Figure 5.1 from the current configuration to another configuration as shown in Figure 5.2. This reconfiguration process goes through four different events. The order of these event executions is based on the rule of the RCPs’ scenarios. These four events as shown in Figure 5.2 can be described as below:

   I. The connection between We and Wo is removed.
   II. The connection from We to Ma and Ma to Fl is established simultaneously.
   III. The connection between Ma and Vp is established.
   IV. The connection from Su to Da and F2 is established simultaneously.

For simplicity, we call these events as E1 (Event 1), E2 (Event 2), E3 (Event 3) and E4 (Event 4) respectively. Then we define two RCPs as two different scenarios as below:

![Figure 5.2. Target configuration of the case study.](image-url)
**Scenario I.** Let us call the first RCP as RCP-1. The rule for the RCP-1 can be described as below:

\[
\text{RCP-1} = \\
\{ E_3; ((E_4; E_1; E_2) + (E_2; E_4; E_1)) \} + \\
\{ E_2; E_3; E_1 \} + \\
\{ E_1; [ (E_3; (E_2; E_4) + (E_4; E_2)) + (E_2; E_3; E_4) ] \} + \\
\{ E_4; [ (E_3; E_2; E_1) + (E_2; (E_3; E_1) + (E_1; E_3)) ] \}
\]

For a better human cognition, Figure 5.3 represents a graph of the RCP-1 where IC is the initial configuration and is depicted as the starting node of the graph. All possible configurations which are created by the RCP-1 events are labeled and shown in the second column of Table 5.1.

<table>
<thead>
<tr>
<th>Events (E₁, E₂, E₃ and E₄)</th>
<th>Configurations</th>
</tr>
</thead>
<tbody>
<tr>
<td>E₁</td>
<td>C1-1</td>
</tr>
<tr>
<td>E₂</td>
<td>C1-2</td>
</tr>
<tr>
<td>E₃</td>
<td>C1-3</td>
</tr>
<tr>
<td>E₄</td>
<td>C1-4</td>
</tr>
<tr>
<td>E₃E₄</td>
<td>C1-5</td>
</tr>
<tr>
<td>E₂E₃</td>
<td>C1-6</td>
</tr>
<tr>
<td>E₁E₃</td>
<td>C1-7</td>
</tr>
<tr>
<td>E₁E₂</td>
<td>C1-8</td>
</tr>
<tr>
<td>E₂E₄</td>
<td>C1-9</td>
</tr>
<tr>
<td>E₁E₂E₄</td>
<td>C1-10</td>
</tr>
<tr>
<td>E₂E₃E₄</td>
<td>C1-11</td>
</tr>
<tr>
<td>E₁E₂E₃</td>
<td>C1-12</td>
</tr>
<tr>
<td>E₁E₂E₄</td>
<td>C1-13</td>
</tr>
<tr>
<td>E₁E₂E₃E₄</td>
<td>C1-14</td>
</tr>
</tbody>
</table>

Table 5.1. Configurations according to RCP-1 events.

Figure 5.3. shows all possible reconfiguration paths along with all possible configurations. Table 5.2 that follows the figure lists all possible reconfiguration paths from RCP-1, nine in this case. Each of the reconfiguration paths are uniquely labeled for simplicity. This can be seen in the second column of the table.

![Figure 5.3. Describes the RCP-1.](image-url)
Table 5.2. All possible reconfiguration paths for RCP-1.

Table 5.2 demonstrates that the case study according to the RCP-1 has nine different strategies which can be employed to transfer the IC to final configuration (C1-14). Let us explain the first reconfiguration path (IC → C1-3 → C1-5 → C1-10 → C1-14); the first state of the path is IC followed by the execution of the event E3 to generate the second configuration (C1-3). Subsequently, the event E4 executes and generate next configuration (C1-5) and so on until the final configuration (C1-14) is generated.

Scenario II. Let us call the second RCP as RCP-2. The rule for the RCP-2 can be described as below:

RCP-2 = \{E_4 ; E_1 ; ((E_3 ; E_2) + (E_2 ; E_3))\} + \\
\{E_3 ; E_1 ; E_2 ; E_4\} + \\
\{E_2 ; ((E_1 ; E_4 ; E_3) + (E_4 ; E_3 ; E_1))\}

In a similar manner as described in Scenario I, the initial configuration IC transfers to the final configuration. All possible configurations which are created by the RCP-2 events are labeled and shown in the second column of Table 5.3. However, RCP-2 generates different reconfiguration paths and some different configurations as shown in Figure 5.4 and Table 5.4.

<table>
<thead>
<tr>
<th>Reconfiguration paths</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC → C1-3 → C1-5 → C1-10 → C1-14</td>
<td>P1-1</td>
</tr>
<tr>
<td>IC → C1-3 → C1-6 → C1-11 → C1-14</td>
<td>P1-2</td>
</tr>
<tr>
<td>IC → C1-2 → C1-6 → C1-11 → C1-14</td>
<td>P1-3</td>
</tr>
<tr>
<td>IC → C1-1 → C1-7 → C1-12 → C1-14</td>
<td>P1-4</td>
</tr>
<tr>
<td>IC → C1-1 → C1-7 → C1-10 → C1-14</td>
<td>P1-5</td>
</tr>
<tr>
<td>IC → C1-1 → C1-8 → C1-12 → C1-14</td>
<td>P1-6</td>
</tr>
<tr>
<td>IC → C1-4 → C1-5 → C1-11 → C1-14</td>
<td>P1-7</td>
</tr>
<tr>
<td>IC → C1-4 → C1-9 → C1-11 → C1-14</td>
<td>P1-8</td>
</tr>
<tr>
<td>IC → C1-4 → C1-9 → C1-13 → C1-14</td>
<td>P1-9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Events (E1, E2, E3 and E4)</th>
<th>Configurations</th>
</tr>
</thead>
<tbody>
<tr>
<td>E2</td>
<td>C2-1</td>
</tr>
<tr>
<td>E3</td>
<td>C2-2</td>
</tr>
<tr>
<td>E4</td>
<td>C2-3</td>
</tr>
<tr>
<td>E1E4</td>
<td>C2-4</td>
</tr>
<tr>
<td>E1E3</td>
<td>C2-5</td>
</tr>
<tr>
<td>E1E2</td>
<td>C2-6</td>
</tr>
<tr>
<td>E2E4</td>
<td>C2-7</td>
</tr>
<tr>
<td>E1E3E4</td>
<td>C2-8</td>
</tr>
<tr>
<td>E1E2E4</td>
<td>C2-9</td>
</tr>
<tr>
<td>E1E2E3</td>
<td>C2-10</td>
</tr>
<tr>
<td>E2E3E4</td>
<td>C2-11</td>
</tr>
<tr>
<td>E1E2E3E4</td>
<td>C2-12</td>
</tr>
</tbody>
</table>

Table 5.3. Configurations according to RCP-1 events.
Table 5.4. All possible reconfiguration paths for RCP-2.

<table>
<thead>
<tr>
<th>Reconfiguration paths</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC → C2-3 → C2-4 → C2-8 → C2-12</td>
<td>P2-1</td>
</tr>
<tr>
<td>IC → C2-3 → C2-4 → C2-9 → C2-12</td>
<td>P2-2</td>
</tr>
<tr>
<td>IC → C2-2 → C2-5 → C2-10 → C2-12</td>
<td>P2-3</td>
</tr>
<tr>
<td>IC → C2-1 → C2-6 → C2-9 → C2-12</td>
<td>P2-4</td>
</tr>
<tr>
<td>IC → C2-1 → C2-7 → C2-11 → C2-12</td>
<td>P2-5</td>
</tr>
</tbody>
</table>

Figure 5.5 shows a part of the attack graph generated for one of the configurations. This figure is exhibited partially because of its vastness, and consequently, that makes it impossible to display the whole figure while keeping it comprehensible at the same time.

Figure 5.5. A partial attack graph of a configuration.
5.2. Results

First, to shed some light on the results, in Table 5.5, we list the CNSMs for all the existing configurations (unique nodes in both Figures 5.3 and 5.4) which were obtained from step four of the framework for both scenarios.

<table>
<thead>
<tr>
<th>Configurations</th>
<th>CAGP</th>
<th>CNP</th>
<th>CMPL</th>
<th>CSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC</td>
<td>0.75998</td>
<td>571</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>C1-3, C2-2</td>
<td>0.72106</td>
<td>572</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>C1-2, C2-1</td>
<td>0.74666</td>
<td>578</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>C1-1</td>
<td>0.70879</td>
<td>169</td>
<td>19</td>
<td>5</td>
</tr>
<tr>
<td>C1-4, C2-3</td>
<td>0.79141</td>
<td>15170</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>C1-6</td>
<td>0.75576</td>
<td>4381</td>
<td>26</td>
<td>5</td>
</tr>
<tr>
<td>C1-7, C2-5</td>
<td>0.67554</td>
<td>170</td>
<td>19</td>
<td>5</td>
</tr>
<tr>
<td>C1-5</td>
<td>0.75323</td>
<td>15171</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>C1-8, C2-6</td>
<td>0.70114</td>
<td>176</td>
<td>18</td>
<td>5</td>
</tr>
<tr>
<td>C1-9, C2-7</td>
<td>0.77166</td>
<td>15177</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>C2-4</td>
<td>0.74590</td>
<td>14768</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>C1-12, C2-10</td>
<td>0.71024</td>
<td>1165</td>
<td>26</td>
<td>5</td>
</tr>
<tr>
<td>C1-11, C2-11</td>
<td>0.78221</td>
<td>18980</td>
<td>21</td>
<td>5</td>
</tr>
<tr>
<td>C1-10, C2-8</td>
<td>0.71227</td>
<td>14769</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>C1-13, C2-9</td>
<td>0.73070</td>
<td>14775</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>C1-14, C2-12</td>
<td>0.74125</td>
<td>15764</td>
<td>20</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 5.5. CNSMs for all existing configurations generated by RCP-1 and RCP-2.

5.2.1. Scenario I

The result of the reconfiguration path metrics for this scenario is shown in Table 5.6. Each reconfiguration path has five security metrics.

A total of five security metrics have been measured for each of the reconfiguration path from the first scenario. These results were tabulated and are shown in Table 5.6. Each column of the table (except the first) represents a particular security metrics, as shown in the first row. Likewise, each row contains the security metrics value for the corresponding reconfiguration paths, the names of which are listed in the first column.

<table>
<thead>
<tr>
<th>RCP-1</th>
<th>PNP</th>
<th>PMPL</th>
<th>PSP</th>
<th>DPNP</th>
<th>PAGP</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1-1</td>
<td>46847</td>
<td>16</td>
<td>5</td>
<td>11618</td>
<td>0.9987699877346</td>
</tr>
<tr>
<td>P1-2</td>
<td>40268</td>
<td>17.4</td>
<td>5</td>
<td>14432</td>
<td>0.9990785016448</td>
</tr>
<tr>
<td>P1-3</td>
<td>40274</td>
<td>17.4</td>
<td>5</td>
<td>14432</td>
<td>0.9991630717389</td>
</tr>
<tr>
<td>P1-4</td>
<td>17839</td>
<td>16.8</td>
<td>5</td>
<td>11618</td>
<td>0.9982997543156</td>
</tr>
<tr>
<td>P1-5</td>
<td>31443</td>
<td>16.8</td>
<td>5</td>
<td>11618</td>
<td>0.9983152826057</td>
</tr>
<tr>
<td>P1-6</td>
<td>17845</td>
<td>16.6</td>
<td>5</td>
<td>11618</td>
<td>0.998439077628</td>
</tr>
<tr>
<td>P1-7</td>
<td>65656</td>
<td>16.2</td>
<td>5</td>
<td>14432</td>
<td>0.9993037973279</td>
</tr>
</tbody>
</table>
As we have used various security metrics to calculate the network security of a particular path, it is evident that different metrics may give us different results on which path is the most secure. Therefore, we have computed the most secure path based on each adopted security metric by sorting them based on the particular metric values, where the first item denotes the most secure reconfiguration path. These results are tabulated in Table 5.7. Each column represents the sorted reconfiguration paths based on a particular metric, the names of which can be seen in the first row.

Table 5.6. Reconfiguration paths of RCP-1 with security metrics.

Table 5.7. Sorted Reconfiguration paths of RCP-1 from the most secure path(s) to the least secure path(s).

The last result from the instantiated framework is the RAGP value for a reconfiguration plan, which, in this case is RCP-1. This value describes the security level of RCP-1 as a whole.

RAGP = 0.998879987319413

5.2.2. Scenario II

In this Scenario, we repeat the same procedure, as done in Scenario I to get the network security value of each configuration path based on five different security metrics (Table 5.8). Likewise, we further sort the configuration paths based on each metric. The results are shown in Table 5.9, and lastly, the RAGP value is calculated for RCP-2.

Table 5.8. Reconfiguration paths of RCP-2 with security metrics.
<table>
<thead>
<tr>
<th>PAGP</th>
<th>PNP</th>
<th>PMPL</th>
<th>DNP</th>
<th>PSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2-3</td>
<td>P2-3</td>
<td>P2-4</td>
<td>P2-1, P2-2, P2-3, P2-4</td>
<td>P2-1, P2-2, P2-3, P2-4</td>
</tr>
<tr>
<td>P2-4</td>
<td>P2-4</td>
<td>P2-1</td>
<td>P2-2, P2-3, P2-4</td>
<td>P2-1, P2-2, P2-3, P2-4</td>
</tr>
<tr>
<td>P2-1</td>
<td>P2-5</td>
<td>P2-1</td>
<td>P2-2, P2-3, P2-4</td>
<td>P2-1, P2-2, P2-3, P2-4</td>
</tr>
<tr>
<td>P2-2</td>
<td>P2-1</td>
<td>P2-5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2-5</td>
<td>P2-2</td>
<td>P2-3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.9. Sorted Reconfiguration paths of RCP-2 from the most secure path(s) to the least secure path(s).

The final RAGP value for this Scenario is:

RAGP= 0.998897831857609

5.3. Analysis

After carefully inspecting all the metrics and their results, we make the following observations. It is worth mentioning that the results are based solely on two experiments, however, there is a possibility that increasing the number of experiments may improve the generalized result.

Let us first look at PAGP and PNP. In Scenario I, we notice that the most secure reconfiguration path in both the case is P1-4. Likewise, the three least secured reconfiguration paths are also the same according to both the metrics. Moreover, we can observe that the second and third positions are interchanged but it should be noted that they still occupy the second and third positions in both the metrics. Additionally, in Scenario II, we notice that the two most secure reconfiguration paths are the same according to both the metrics. Thus, we can say that irrespective of the fact that both the metrics are different and they compute different absolute values in terms of their respective units because they employ distinct algorithms for the computation of the security of reconfiguration path, the results obtained regarding the most or least secured path is the same.

Although it is hard to generalize our results, however, if we observe them, we notice that the paths P1-9, P1-8 and P1-7 have been found to be the least secure based on both the metrics. Delving further, we see that the very first event to construct the first configuration in these three paths is E4. This event connects a subnet with two vulnerabilities to the network and throughout its journey from the start of the reconfiguration until its target configuration, the network is connected to the subnet. At the same time, in a different path, i.e. P1-4, E4 is the concluding event to be executed, whereas first event in this path is E1 which removes the link between the workstation and the webserver which has one vulnerability each. Intuitively, removing the connection between vulnerabilities decreases attackers opportunity to exploit the system while adding vulnerability to the system increases the opportunity. Therefore, it can be inferred that for a path to be more secure, adding links between vulnerable components should take place in the very end, whereas removing links between these components should happen at the very beginning of the reconfiguration process. Thence, we can say that the results are reasonable and reflective. In addition, if a reconfiguration plan has multiple alternatives of events for adding links (data communications) between vulnerabilities (vulnerable
software) to select the next configuration, conclusively finding out the most secure alternative is not possible, as we don’t know how this vulnerability affects the security level of the configuration. It could be said likewise, in case of having multiple alternatives of events for removing links (data communications) between vulnerabilities (vulnerabile software) to select the next configuration.

Now, let us observe PMPL and PNP. While both of them are categorized as an attack path metric from their base metrics MPL and NP, respectively, we can still see that the results obtained by them are entirely different. This varied result in both the metrics is agreeable and expected, as it distinguishes the inherent properties of both metrics. While the base MPL takes into consideration the attack steps involved in an attack path, but the base NP doesn't. However, taking the attack steps into consideration, as it is done in base MPL is not preferred, because it may not capture certain changes in the security level of the configurations, given that MPL is based on the arithmetic mean. In other words, two configurations with different security levels may still be assigned with the same MPL value[39].

Next, we look at the PSP metric. It remains constant throughout all the reconfiguration paths in both the scenarios. This is because the shortest distance from the attacker to the target host remains the same all the time (this is also stated in [2] wherein the security of the dynamic network is evaluated). Therefore, we can conclude that PSP might not useful for our purposes.

Finally, we observe DPNP. As it can be seen in both scenarios, it categorizes all the reconfiguration paths into two and only two numerical values, which are the security metric values for the reconfiguration paths. This property of DPNP of categorizing all the reconfiguration paths into two distinct numerical values and the fact that it cannot generate unique security values for each path, makes this metric not suitable for our security decisions.

Let us discuss this behavior of DPNP that causes this categorization. DPNP only counts distinct attack paths along a reconfiguration path and it does not take into consideration if the configuration with high amount of attack path occurred only once or multiple times during the reconfiguration, i.e. the numerical output generated by DPNP if a specific group of attack paths exists only once, or twice or more in the reconfiguration path will be one and the same making all these paths equally secure or insecure according to DPNP. This phenomena demonstrated by the metric was unexpected while designing.

By comparing the RAGP values of both the Scenarios, we can determine the most secure reconfiguration plan. The plan with a lesser value of RAGP is more secure. In our case it is RCP-1.

Referring to our previous argument that we cannot generalize our finding based only on these two experiments, let us still make some observation. We can notice that the generated number of reconfiguration paths for RCP-1 is more than RCP-2. There may be a common perception that more way to reconfigure makes a reconfiguration plan more secure, however that is not true for this case, unless we add a condition: More way to reconfigure may make a reconfiguration plan more secure, if the likelihood of generating more secure reconfiguration paths increase along with the increasing number of the paths.
5.4. Discussion
We found that our designed security metrics differently evaluate the security level of reconfiguration plans. According to the results that be obtained in Scenario I and II, we observed that the metrics PAGP, PMPL and PNP generate unique values for almost all the reconfiguration plans whereas it was observed that the metrics SSP and DNP generates identical values for almost all reconfiguration plans.

Based on this observation we feel that a fine line can be drawn between these two types of metrics, and therefore, it was decided to categorize the aforementioned metrics into two different groups, let us call them group X and group Y. We define these groups as follows:

Group X: Metric that generates unique values for all the reconfiguration plans, notwithstanding the fact that any, but not more than two of the reconfiguration plans may have identical values.

Group Y: Metrics that generates only one or two values for all possible reconfiguration plans.

<table>
<thead>
<tr>
<th>Group X</th>
<th>Group Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGP</td>
<td>SSP</td>
</tr>
<tr>
<td>PMPL</td>
<td>DNP</td>
</tr>
<tr>
<td>PNP</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.8. Categorizing the metrics based on thier response to system configuration changes.

From the results in Table 5.7 and Table 5.9, Table 5.8 is created. They show that the group X security metrics can be used to evaluate the security level of the reconfiguration paths and generate the most secure reconfiguration plan. Since the group Y security metrics evaluate the entire reconfiguration paths as a single or a couple of security level, therefore this group might not fit for the framework's purposes.

Next, we look at the group X metrics, PMPL and PNP are relying on the characteristics of attack paths and focused on information such as number or size of attack paths. Thus, one deficiency of these metrics is that they do not cover the effort an attacker takes to execute an attack. In other words, the difficulty degree of exploiting a given vulnerability is not considered. Conversely, PAGP which is driven from AGP as a base metric, by using CVSS metric, covered this shortcoming.

All the conclusions made by us in this report where based on the limited analysis and discussion done by us in this report. Therefore, to further strengthen and generalize the results additional experiments should be performed which will be reserved for future work.
In conclusion, we can argue that our research questions have been answered by our proposed Instantiated Framework and the results from the test case, as follows:

1. By utilizing SP, NP, MPL, and AGP the security level of each configuration in the reconfiguration paths for a given reconfiguration plan have been measured followed by designing the indirect metrics (PSP, PNP, PMPL, DNP, and PAGP), the security level of each reconfiguration paths have also been measured. These processes have answered RQ1.

2. The selection of Attack Graphs as the framework's modeling technique answered the RQ2.

3. By designing RAGP for measuring security level of a reconfiguration plan as a whole, the RQ3 has been answered.

4. The developed prototype tool, which is described in chapter 4 (Implementation) that automatically measures the security level of a reconfiguration plan, answered the RQ4.

However, to the best of our knowledge, and based on our literature review, since there aren't any research works or findings to evaluate the security level of a reconfiguration plan, we are unable to discuss any similarity or dissimilarity with any previous research. Nevertheless, let us discuss this approach with one framework described earlier i.e. HARM. It is an approach for evaluating the security of a dynamic network which differs from our approach as described below.

- HARM needs two types of security model to be used, whereas the Current Approach needs only one security model, which is the Attack Graph. Note that using only one model decreases the effect of the inaccuracy of modeling approaches. Nonetheless, multi-level modeling makes modeling more modular and readable.

- HARM works only when the network change is adding or deleting a vulnerable software. It cannot handle structural changes, whereas the Current Approach is capable of handling both cases. Hence, we can say that the current approach has better availability than HARM.

- HARM considers only vulnerabilities existing on a host and host’s reachability. It does not consider dependency and interaction between vulnerabilities among the hosts, but in case of the Current Approach, the attack graph considers the dependencies, if any, between vulnerabilities even if they are not directly connected. This property makes the current approach more accurate.

- In case of HARM, evaluation involves incompetent assumptions of always declaring the relation between vulnerabilities in the attack tree to be logical OR (Section 2.2.1.3), and thus, it digresses from its theoretical definition. However, there are no assumptions in the Current Approach.
6. Conclusions and Future Work

In this thesis, we have developed a framework for vulnerability analysis of (self-) adaptive system and evaluating the security of different reconfiguration strategies of a reconfiguration plan. Moreover, based on our in-depth literature survey, we chose attack graphs as our security modeling technique and proposed several metrics to measure the security of configurations, reconfiguration strategies and reconfiguration plans based on this modeling language. We implemented a prototype tool to support our framework that (i) automatically generates configurations and the reconfiguration strategies of a given reconfiguration plan, (ii) uses MulVAL to generate an attack model for each configuration automatically, and (iii) measures the security levels of configurations, reconfiguration strategies and reconfiguration plans based on the metrics proposed in this report that is used to rank security of reconfiguration strategies and reconfiguration plans.

Based on the proposed Instantiated Framework, we have analyzed the test case to demonstrate the effectiveness of the framework. Through our analysis and discussion, we have also identified the most sensitive and proper indirect NSMs for the security evaluation of the reconfiguration path.

We expect that this work establishes a foundation on which a significant series of research could be based. The following are some suggestions for follow-up studies.

HARM security modeling approach as described in Section 2.2.1.3 can be employed in step three instead of attack graphs and adopt some NSMs related to HARM in step four followed by designing a new or indirect security metrics for measuring security level of the reconfiguration-path in step five.

We could also have designed a test case that would have been included in the reconfiguration plan by adding and/or deleting vulnerable software(s) on a component(s) and then evaluating the result. However, we did not consider this particular case, nevertheless, we expect that the framework will be able to generate a secure strategy in the same fashion as our case.

Other NSMs related to attack graphs can be utilized in step four followed by designing indirect metrics or new metrics with reconfiguration-path as their scope for measuring security level of reconfiguration-paths and then evaluating the results of metrics according to their responsiveness to system structural changes.

The security level of a reconfiguration path can be calculated by mapping between attack graphs’ nodes within two adjoined configurations in the same reconfiguration-path in order to identify the changes part as of the attack graphs of these two adjoined configurations.

Our last suggestion is an extension of this work so that might be a machine learning algorithm or a statistical approach can be utilized to aggregate all five metrics values of reconfiguration paths to find the similarity between their security evaluations of the reconfiguration paths in order to the generated secure strategy will be evaluated based on all the metrics.
7. References


[19] Zhikun Zhao and Wei Li. Dynamic Reconfiguration Planning with Influence Control. 6th IEEE/ACIS International Conference on Computer and Information


[52] F. Cadini, E. Zio, and C. Petrescu, Using centrality measures to rank the importance of the components of a complex network infrastructure, in Critical Information Infrastructure Security, ser. Lecture Notes in Computer Science, R.


Appendix 1

Initial configuration (IC) of the case study, Figure 5.1.

attackerLocated(internet).
attackGoal(execCode(workStation, _)).
attackGoal(execCode(fileServer, _)).
attackGoal(execCode(vpnServer, _)).
attackGoal(execCode(dataHistorian, _)).
attackGoal(execCode(webServer, _)).
attackGoal(execCode(commServer, _)).
attackGoal(execCode(citrixServer, _)).
%attackGoal(execCode(subnet_1, _)).
attackGoal(execCode(mailServer, _)).

/*Network connections*/
hacl(internet, webServer, tcp, 80).
hacl(fileServer, workStation, nfsProtocol, nfsPort).

hacl(webServer, internet, _, _).

/*Adding subnet_1 to the network by uncommenting the four lines below*/
%hacl(subnet_1, dataHistorian, httpProtocol, httpPort).

/*Adding mail server to the network by uncommenting the four lines below*/
%hacl(webServer, mailServer, nfsProtocol, nfsPort).
%hacl(mailServer, internet, tcp, 80).
%hacl(vpnServer, mailServer, nfsProtocol, nfsPort).
%hacl(mailServer, vpnServer, nfsProtocol, nfsPort).

/*/ configuration information of workStation */
remote_client_vul_exists(workStation,acrobat).
vulExists(workStation,'CVE-2010-0491',acrobat).
vulProperty('CVE-2010-0491',remoteExploit,privEscalation).
vulProperty('CVE-2010-0491',remoteClient,privEscalation).
inCompetent(victim_4).
hasAccount(victim_4, 'workStation', user).
nfsExportInfo(workStation, '/export', write, fileServer).

/* configuration information of mailServer */
networkServiceInfo(mailServer, mountd, rpc, 100005, root).
nfsExportInfo(mailServer, '/export', read, vpnServer).
nfsExportInfo(mailServer, '/export', read, webServer).
nfsExportInfo(mailServer, '/export', write, vpnServer).
vulExists(citrixServer,'CVE-2010-0490',ie).
vulProperty('CVE-2010-0490',remoteExploit,privEscalation).
vulProperty('CVE-2010-0490',remoteClient,privEscalation).
inCompetent(victim_2).
hasAccount(victim_2, citrixServer, user).

/* configuration information of commServer */
vulExists(commServer,'CVE-2010-0483',windows_2000).
vulProperty('CVE-2010-0483',remoteExploit,privEscalation).
vulProperty('CVE-2010-0483',remoteClient,privEscalation).
inCompetent(victim_1).
hasAccount(victim_1, commServer, user).

/* configuration information of vpnServer */
vulExists(vpnServer,'CVE-2010-0492',openvpn).
vulProperty('CVE-2010-0492',remoteExploit,privEscalation).
vulProperty('CVE-2010-0492',remoteClient,privEscalation).
inCompetent(victim_5).
hasAccount(victim_5, 'vpnServer', user).

/* configuration information of webServer */
vulProperty('CAN-2002-0392', remoteExploit, privEscalation).
networkServiceInfo(webServer, httpd, tcp , 80 , apache).
isWebServer(webServer).
inCompetent(victim_7).
hasAccount(victim_7, 'webServer', user).

/* configuration information of dataHistorian */
vulExists(dataHistorian,'CVE-2010-0494',mountd).
vulProperty('CVE-2010-0494',remoteExploit,privEscalation).
vulProperty('CVE-2010-0494',remoteClient,privEscalation).
networkServiceInfo(webServer, httpd, tcp , 80 , apache).

/* configuration information of fileServer */
vulExists(fileServer,'CVE-2010-0812',windows_2003_server).
vulProperty('CVE-2010-0812',remoteExploit,privEscalation).
vulProperty('CVE-2010-0812',remoteClient,privEscalation).
inCompetent(victim_3).
hasAccount(victim_3, 'fileServer', user).

/* configuration information of subnet_1 */
vulExists(subnet_1,'CVE-2010-0483',windows_2000).
vulProperty('CVE-2010-0483',remoteExploit,privEscalation).
vulProperty('CVE-2010-0483',remoteClient,privEscalation).
inCompetent(victim_1_1).
hasAccount(victim_1_1, 'subnet_1', user).
vulExists(subnet_1,'N1-2010-0490',ie).
vulProperty('N1-2010-0490',remoteExploit,privEscalation).
vulProperty('N1-2010-0490',remoteClient,privEscalation).
inCompetent(victim_1_2).
hasAccount(victim_1_2, 'subnet_1', user).

/*cvss metrics*/
cvss('CVE-2010-0491',l).
cvss('CVE-2010-0494',m).
cvss('CVE-2010-0812',m).
cvss('CVE-2010-0483',h).
cvss('CVE-2010-0492',m).
cvss('CAN-2002-0392',h).
cvss('CVE-2009-2503',m).
cvss('CVE-2010-0490',m).
cvss('CVE-2010-0493',m).

/*Client side applications*/
isClient(office_frontpage).
isClient(itunes).
isClient(acrobat).
isClient(excel).
isClient(compatibility_pack_word_excel_powerpoint).
isClient(windows_2000).
isClient(mountd).
isClient(quicktime).
isClient(adobe_air).
isClient(ie).
isClient(flash_player).
isClient(office_word).
isClient(windows_2003_server).
isClient(visual_basic).
isClient(openvpn).
isClient(httpd).
isClient(firefox).
isClient('.net_framework').