Scalable Attack Modelling in Support of Security Information and Event Management

Keiran Dennie

Supervisor: Dr. Andrew Hutchison
Co-Supervisor: Dr. Anne Kayem

Thesis presented for the Degree of Master of Science
in the Department of Computer Science
University of Cape Town

April 30, 2014

Plagiarism Declaration

I know the meaning of Plagiarism and declare that all of the work in the document, save for that which is properly acknowledged, is my own. I also certify that the above statement applies to the implementation of the project.

Keiran Dennie, Bachelor of Science in Computer Science
University of Cape Town, Cape Town
April 30, 2014

Abstract

While assessing security on single devices can be performed using vulnerability assessment tools, modelling of more intricate attacks, which incorporate multiple steps on different machines, requires more advanced techniques. Attack graphs are a promising technique, however they face a number of challenges.

An attack graph is an abstract description of what attacks are possible against a specific network. Nodes in an attack graph represent the state of a network at a point in time while arcs between nodes indicate the transformation of a network from one state to another, via the exploit of a vulnerability. Using attack graphs allows system and network configuration information to be correlated and analysed to indicate imminent threats. This approach is limited by several serious issues including the state-space explosion, due to the exponential nature of the problem, and the difficulty in visualising an exhaustive graph of all potential attacks. Furthermore, the lack of availability of information regarding exploits, in a standardised format, makes it difficult to model atomic attacks in terms of exploit requirements and effects.

This thesis has as its objective to address these issues and to present a proof of concept solution. It describes a proof of concept implementation of an automated attack graph based tool, to assist in evaluation of network security, assessing whether a sequence of actions could lead to an attacker gaining access to critical network resources. Key objectives are the investigation of attacks that can be modelled, discovery of attack paths, development of techniques to strengthen networks based on attack paths, and testing scalability for larger networks. The proof of concept framework, Network Vulnerability Analyser (NVA), sources vulnerability information from National Vulnerability Database (NVD), a comprehensive, publicly available vulnerability database, transforming it into atomic exploit actions. NVA combines these with a topological network model, using an automated planner to identify potential attacks on network devices. Automated planning is an area of Artificial Intelli-

gence (AI) which focuses on the computational deliberation process of action sequences, by measuring their expected outcomes and this technique is applied to support discovery of a best possible solution to an attack graph that is created. Through the use of heuristics developed for this study, unpromising regions of an attack graph are avoided. Effectively, this prevents the state-space explosion problem associated with modelling large scale networks, only enumerating critical paths rather than an exhaustive graph. SGPlan5 was selected as the most suitable automated planner for this study and was integrated into the system, employing network and exploit models to construct critical attack paths. A critical attack path indicates the most likely attack vector to be used in compromising a targeted device. Critical attack paths are identified by SGPlan5 by using a heuristic to search through the state-space the attack which yields the highest aggregated severity score. CVSS severity scores were selected as a means of guiding state-space exploration since they are currently the only publicly available metric which can measure the impact of an exploited vulnerability. Two analysis techniques have been implemented to further support the user in making an informed decision as to how to prevent identified attacks.

Evaluation of NVA was broken down into a demonstration of its effectiveness in two case studies, and analysis of its scalability potential. Results demonstrate that NVA can successfully enumerate the expected critical attack paths and also this information to establish a solution to identified attacks. Additionally, performance and scalability testing illustrate NVA's success in application to realistically sized larger networks.

Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor Dr. Andrew Hutchison for his continuous support, patience, enthusiasm, and valuable input. I would also like to thank my co-supervisor Dr. Anne Kayem for guidance in writing this thesis. This work has also been partially supported by Project MASSIF FP7-257475, a collaborative project co-funded under the European Commission's FP7 ICT Work Programme.

Contents

List of Figures

LIST OF FIGURES

LIST OF FIGURES

Chapter 1

Introduction

In this chapter we introduce the context and motivation for the investigation into use of attack graphs for enhanced security. It also serves to provide a problem statement, highlight objectives and outline the structure of this thesis.

1.1 Motivation

Security has become a necessity in today's world, with people becoming more aware of its importance as malicious viruses, attempted intrusions and attacks become a frequent occurrence. Government and industry rely heavily on information technology, making them vulnerable targets to hackers and viruses. The repercussions of a well-orchestrated attack can be devastating to governments or enterprises and their clients. Moreover, sophisticated attacks on industrial systems can have crippling effects on cities.

One of the most pernicious types of attack is where the execution of a virus 'pivots' from machine to machine. In this context, *pivoting* refers to a shift in an attacker's position by relocating to a compromised machine. A compromised machine can acts as a new source of attack on other machines in a network. A virus known as Downadup or Conficker has been one of the most prolific viruses in the last few years, infecting a large number of computers worldwide. According to security provider F-Secure, an estimated 9 million computers were

infected in the period of June to December 2009.[1] Downadup spreads by exploiting a remote code execution vulnerability on many unpatched versions of Microsoft Windows. It has the ability to scan a network for vulnerable hosts as well as copy itself to removable drives. While Downadup remains on a removable drive it attempts to infect other computers when the is drive inserted, leveraging the autorun function, which is enabled on Microsoft Windows by default.[2] The effects of such an aggressive virus can be crippling for organizations because it can rapidly spread throughout its entire infrastructure. Consequently, there is a requirement for more advanced techniques in network protection.

Red October, a disturbing attack discovered in January 2013, targeted governmental organizations. Confidential documents regarding nuclear research, oil companies, geopolitical intelligence and computer access credentials were stolen in this attack. Moreover, Red October's code had the ability to undelete and steal files from removal disks plugged into any infected computers.[3] The malware was found to spread to prospective victims via e-mail, containing a Microsoft Word or Excel document attachment which contains the malicious payload. In addition to the aforementioned attack vector, it was also discovered that attackers infiltrated victim networks via a Java exploit known as Rhino. By e-mailing a link to a malicious site to victims, a Java payload is injected into victims with outdated versions of Java.[4]

Attacks typically do not involve just one machine, but rather target a number of systems, which may be through firewalls and several layers of security. These types of attacks are known as advanced persistent threats (APT). They are complex multi-step attacks, which involve the exploitation of a number of vulnerabilities on different machines. By installing backdoors on machines across a network, it allows an attacker to maintain access without discovery.[5] The overall result of such an attack allows an attacker to reach a specific goal, while circumventing security measures.

For example, we consider an example of an attack on an organization's customer database. In the wrong hands, this kind of information could have massively detrimental effects on

[1] http://www.f-secure.com/weblog/archives/00001584.html
[2] http://www.symantec.com/security_response/writeup.jsp?docid=2008-112203-2408-99
[3] http://www.bbc.co.uk/news/technology-21013087
[4] https://www.securelist.com/en/analysis/204792265/Red_October_Detailed_Malware_Description_1_First_Stage_of_Attack
[5] http://searchsecurity.techtarget.com/definition/advanced-persistent-threat-APT

the organization. An instance of this occurred in 2011, where hackers compromised Sony PlayStation's user database, costing Sony more than $170 million and took the service offline for two weeks.[6] In Figure 1.1, the topology of an example organization's network is illustrated, revealing how an attacker can interact with it.

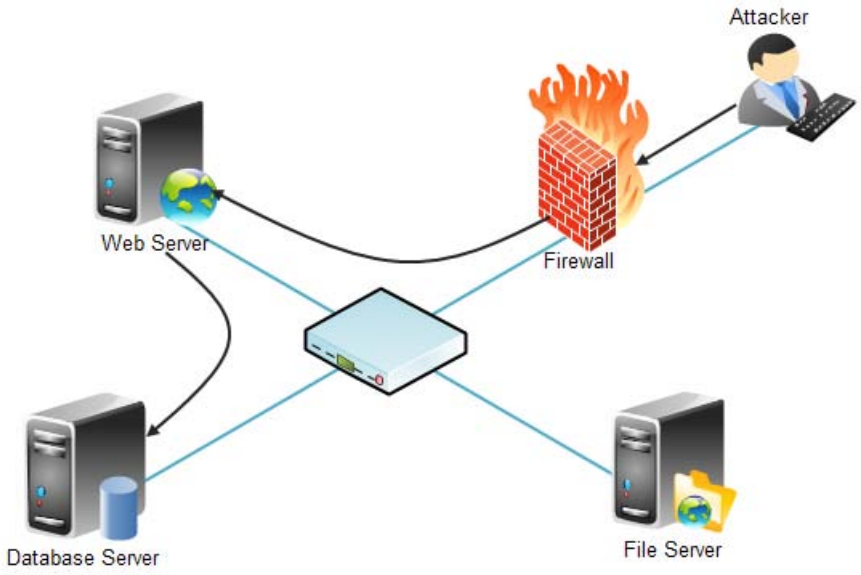


Figure 1.1: *An attacker can compromise a network by pivoting from machine to machine until he reaches his target.*

The network features a single firewall, web server, file server and a customer database. The public only has access to the organization's website, hosted on the web server. While the attacker's goal is to retrieve client information from the customer database, he does not have direct access to it due to restricted ingress connections by the firewall. So how could an attacker manipulate his current access to further gain access to his target? By exploiting a vulnerability in the organization's web server, the attacker pivots his position to the web server, gaining vision of the internal network topology and insight into other opportunities. Next, scanning the internal network, he finds that he now has direct access to the customer database and the file server. Reconnaissance on these machines also reveals that the file server has an exploitable vulnerability, and that the database server in fact has two exploitable vulnerabilities. The first database vulnerability is remotely exploitable and provides the attacker with user level privileges, while the second vulnerability is locally exploitable and yields administrator privileges. By exploiting these two vulnerabilities sequentially, the attacker not only gains access to the database server but also has the highest

[6] http://smallbusiness.chron.com/effects-computer-hacking-organization-17975.html

system privilege level. The result of this is that he can steal customer information from the database as well as place a backdoor on the system to maintain his access to it. It is important to note that the path of attack that the attacker chose was not the only path of attack, considering that he could have also opted to compromise the database server via the file server. At each step of an attack, the attacker can make a variety of decisions, each of which may or may not lead to the attacker achieving his goal.

Attack graphs provide a description of attacks by linking atomic attack steps to form a complex multi-step attack, similar to the situation described above. They illustrate each of the potential decisions an attacker can make at each step of an attack, based on information about the systems under attack. Nodes represent the configuration or state of a network at a point in time, while arcs between nodes represent exploits which have an effect on machines in a network, effectively transitioning it to a new state. Figure 1.2 shows an attack graph based on Figure 1.1, exhaustively enumerating each possible attack that an externally based attacker can perform on the network. Two pieces of information are required to construct attack graphs: information about a network and vulnerability information. Using this information, an attack graph can be constructed which enumerates every possible attack action. The advantage of this is that it condenses a large amount of security information into a visualization, which is far more comprehendable for security administrators. Moreover, analysis of attack graphs can offer insight into weak or reachable points in a network, making it simpler to identify the best points in a network to secure, given that it is not always feasible to patch every vulnerable host. With such large volumes of security information it would be highly beneficial to have automated tools which can compute security metrics and find potential security weaknesses.

In large enterprise networks, a vast number of events are generated by devices every second. Analysis of event data can potentially prevent malicious users from gaining access to systems in the network, however the overwhelming amount of data makes manual analysis highly impractical. Security Information and Event Management (SIEM) systems were developed for this sole task, aggregating event data from network devices and correlating it to help discover suspicious host activity. By responding to alerts in real-time, a security administrator can take appropriate countermeasures to impede an attack in progress. During a SIEM life-cycle multiple risk assessment phases are performed, to gauge the overall security level of network infrastructure and determine its weakness against attack. One must consider the implica-

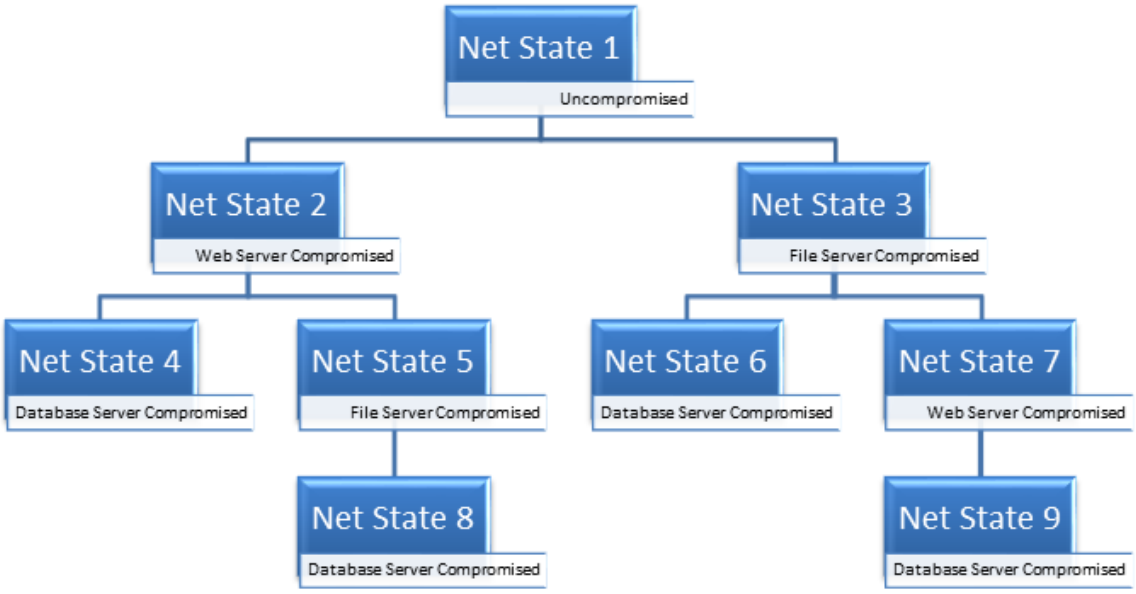


Figure 1.2: *An attack graph showing each step involved in a complex multi-step attack. The attacker pivots from machine to machine to reach his final goal.*

tions of updating network infrastructure, via the addition or removal of devices. Updating network infrastructure may result in the introduction of new vulnerabilities and in turn, new opportunities for an attacker to take advantage of. For this reason it is crucial to perform risk assessments on a regular basis, especially after the modification of network infrastructure. This thesis considers an approach whereby SIEM could be augmented with automated attack modelling, to enhance threat modelling capabilities with rich network configuration details.

1.2 Problem Statement

While attack graph construction is not a new concept, its use has been largely restricted in real-life applications due to certain computational limitations. Previous research on attack graph construction, elaborated on in Section 2.2.2, focused on generating exhaustive attack graphs, which enumerate every possible outcome of an attack action at each stage of attack. Exhaustive searches of attack graphs become exceedingly difficult to manage as networks grow in complexity. This issue is commonly known as the combinatorial problem, and occurs in many applied mathematics and theoretical computer science optimization problems.

Visually, it also becomes challenging to illustrate such large attack graphs, which is counterproductive considering that they are meant to be simple and understandable for security administrators.

1.3 Research Objectives

Taking into account the aforementioned problem statement, the following objectives will be addressed. The research investigation aims are:

- Investigation of the type of attacks that can be modelled effectively using an attack planning approach. Attack data will be sourced from publicly available vulnerability databases. The types of information available from these databases and the structure of entries will play a deciding factor in this aim.
- Discovery of whether critical paths can be enumerated, which lead to the compromise of vulnerable network resources. A proof of concept framework will be designed, implemented and tested against networks where expected critical paths have already been calculated.
- Identification of how a vulnerable network's security can best be strengthened, based on projected models. Investigation into attack graph analysis techniques will establish approaches most suitable for application to network security planning problems.
- Demonstration of how well the proposed solution can scale for large networks. By generating and evaluating networks of increasing complexity and size, success will be determined by the framework's ability to consistently discover solutions without crashing due resource depletion.

1.4 Structure of this Thesis

Chapter 1 serves to introduce this thesis and outline some of its key focuses. The motivation for this thesis was noted and followed by the problem statement. Based on these two sections, the research objectives of this work are listed and elaborated on.
Chapter 2 presents a background and related works which have influenced decisions made in this thesis. It first overviews SIEM and highlights its role in enhancing network security.

Then it discusses various attack graph approaches, outlining their benefits and shortcomings. Lastly, it looks at various automated planning models and describes several state-space searching algorithms.

Chapter 3 proposes a design for a proof of concept framework and details the system architecture. The chapter is broken down into three phases: Information Aggregation, Attack Graph Computation and Visualization.

Chapter 4 provides a description of the implementation of the proof of concept framework and includes the inner workings of the most important procedures. These include network generation, vulnerability extraction, network configuration extraction, model transformation and analysis.

Chapter 5 subjects the implementation to evaluation, using two different testing techniques. The first evaluates its effectiveness in identifying critical attack paths in two case studies. Furthermore, it guides the user through its usage with a walkthrough that demonstrates basic functionality. The second evaluation techniques gauges the framework's performance and scalability against increasingly large and complex generated networks.

Chapter 6 concludes this thesis, briefly revisiting and summarizing some of the key topics covered. It examines the effectiveness of this thesis, by comparing the evaluation results with the initial research objectives. Then it reveals opportunities which could be incorporated into future works on this topic.

Chapter 2

Background

In this chapter we review various topics pertaining to network security and attack graph construction. The concept of Security Information and Event Management (SIEM) is introduced, revealing its relevance in attack graph construction and how it can be enhanced. Furthermore, various attack graph construction techniques are reviewed, discussing some of the benefits and shortcomings of each. Lastly, we provide an overview of the automated planning field and how this approach can benefit attack graph construction.

2.1 Security Information and Event Management

2.1.1 Overview

In modern computer networks large amounts of security log data are generated from a variety of hardware devices and software applications by the second. These logs files include all sorts of information, ranging from events triggered by systems and devices to user activities within a network infrastructure. An organization's security status can be investigated by performing forensic operations on captured log data, revealing potential flaws or attacks against its network. Analysis of log files offers a wide variety of information, such as user level information (authentication, file access or internet activity), system and device level data (file read, write or delete, network traffic information and session status) and network security activities (identifying virus or attack signatures)[1].

A means of centralised log management and analysis on these vast amounts of data is offered by Security Information and Event Management (SIEM) systems. The concept of SIEM systems can be attributed to Gartner analysts Amrit Williams and Mark Nicolett[2] and is a combination of two disparate but complementary technologies, namely Security Information Management (SIM) and Security Event Management (SEM). The amalgamation of these two technologies offers several useful services to security administrators in charge of keeping a network secure, as is illustrated in Figure 2.1.

SIM is primarily concerned with a historical analysis of the network system and application logs, and is generally used for internal network threat management as well as regulatory compliance auditing. Several critical functions of SIM solutions include log storage and archival, data analysis and reporting[3].

Conversely, the SEM solution is focused on the real-time aspects of network security, providing monitoring and event correlation and processing. The events being analysed are alerts generated from devices such as firewalls, Intrusion Detection Systems (IDS) that react to suspicious host activity, based on several pre-constructed patterns[3].

2.1.2 Typical SIEM Roles

SIEMs aim to provide event archiving, reporting, real-time analysis and alert generation using events generated from various devices and applications within a network, aiding organizations in responding to attacks and managing data[4]. Log data is collected and aggregated from various sources, such as desktop computers, servers, routers, application and Syslog devices. This is achieved using edge-side agents installed on devices, filtering out inconsequential data and normalizing it into formats which can later be fed into core-side correlation engines for analysis[5]. At the most basic level, SIEMs often offer rule-based or statistical correlation engines to identify any relationships between events. Correlation of event data may offer insight into attacks against the network, producing crucial alerts so that administrators can take suitable countermeasures.

Although the term “SIEM” was only coined relatively recently (2005), the market has ma-

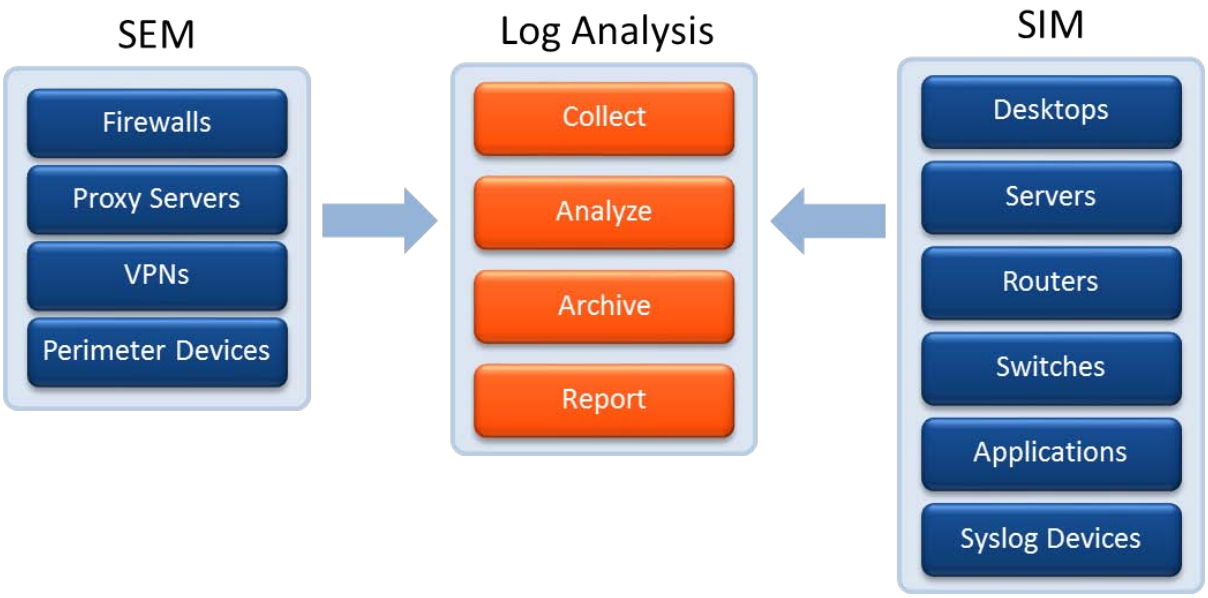


Figure 2.1: *A collection of functions offered by SIM and SEM, which are included in the operations of a SIEM system[1].*

tured and has become competitive quite rapidly, Multiple vendors now offering products featuring typical SIEM requirements[6]. Three primary use cases form the core of a typical SIEM's characteristics, these include the following:

- Management of log files and compliance reporting
- Real-time monitoring of user and application activity, data access and incident management
- A framework which offers administrators an array of suitable threat management options, such as policy countermeasure enforcement or decision support

2.1.3 Generic SIEM Architecture

The architecture of a SIEM is layered and consists a data collection layer, data integration layer and an analysis layer, as shown in Figure 2.2. The data collection layer is responsible for aggregation of log data that is stored in a repository, and made available to applications that handle analysis and interpretation. After the log data has been collected, the data integration layer manages data description and transport to locations required to do further processing. Finally, the analysis layer provides functionality for performing various

data analysis operations. The separation of these three layers removes the need for data collection, processing and integration for each analysis engine. This makes the development and integration of third party plugins and analysis engines much easier, by decoupling log data collection and normalization from analysis. Third party plugins can use pre-collected and normalized log data rather than repeating this process, which can be computationally expensive.

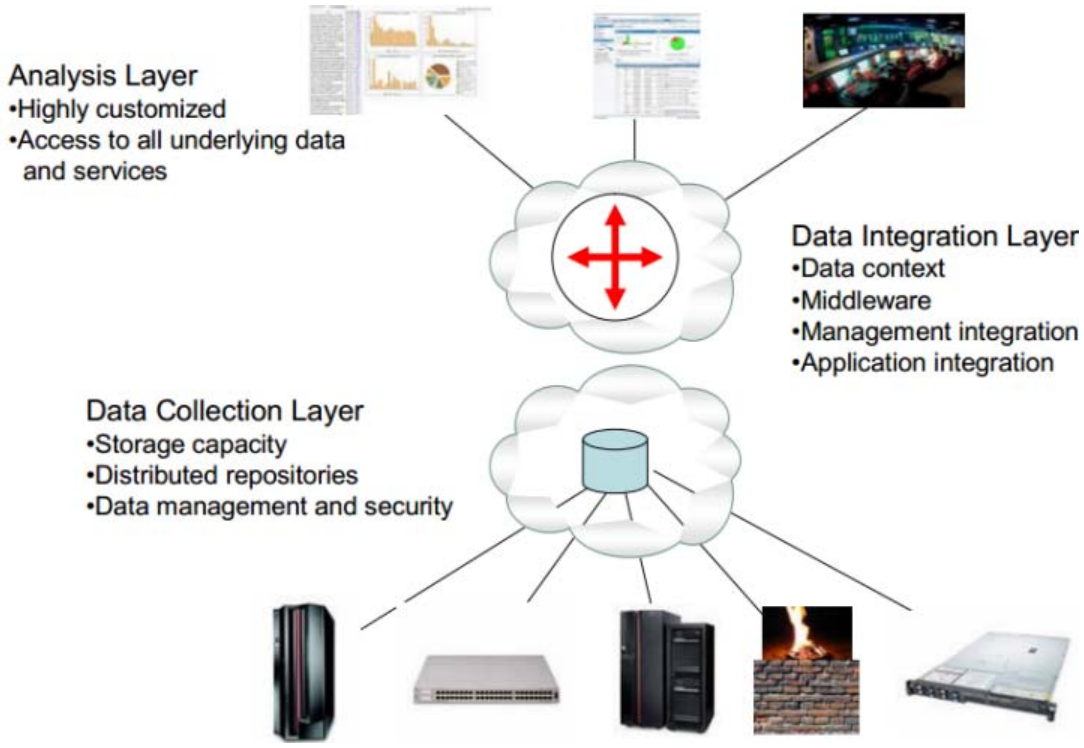


Figure 2.2: *A depiction of the general struction of a SIEM's architecture, including the various layers required for its operation[7].*

2.1.4 Further Development Opportunities

Although SIEMs have matured to a certain extent, many chief information security officers feel that SIEMs are failing to offer more sophisticated capabilities they would like to add[8]. Organizations face an increasing number of threats on a daily basis and malefactors are becoming more sophisticated in their attack strategies, resulting in increased difficulty in

threat detection. This issue prompts the advancement of SIEM technology and the development of of new approaches to aid in identifying threats posed against a network. One such approach is the generation and analysis of threat attack graphs, which can be used in SIEMs. Integration of the proof of concept framework in this research with SIEM is not within the scope of this thesis, however there is the potential for integration in future work.

2.2 Analysis of Network Security Using Attack Graphs

2.2.1 Overview

Attack graphs offer a graph-based approach to analysing network vulnerabilities, by showing how individual attacks can be composed to create larger attack paths. They symbolise a collection of potential scenarios whereby a malicious user attacks a network. Such capabilities can answer questions such as "Can an external malefactor find a path that will allow him to read sensitive corporate information?" or "If a network is under attack, what should we fix first?" These answers allow a network administrator to prioritize changes that need to made to the network in order to secure it. A scenario is represented by a sequence of attack steps by a malefactor, often resulting in a particular goal, such as gaining administrative access to a host device, access to a database or the disruption of service offered. The flexible nature of this technique allows for the analysis of attacks originating from within the network or externally, examining the risks posed against a particular device or the network as a whole.

An attack graph consists of nodes, each of which represent the state of a network as a whole, during the execution of an attack by a malefactor on one of the devices within the network. Edges connecting nodes symbolise state transitions of a network, corresponding with an atomic action executed by a malefactor. Numerous different types of approaches to attack graphs and trees have been developed, including a variety security analysis techniques.

2.2.2 Related Work

R. Ortalo et al.[9] consider an approach whereby attack graphs are generated using topological vulnerability analysis techniques, in order to enumerate possible sequences of exploits. Vulnerabilities in computer systems are represented using privilege graphs. These privilege

graphs contain nodes symbolising sets of privileges owned by users or sets of users, while edges represent vulnerabilities. An edge is drawn from a node X to a node Y should there be some method granting a user owning X privileges to obtain those of node Y. The system proposed assigned weights to edges corresponding to the effort required for an attacker to exploit a particular vulnerability. At the time, such detailed intrusion data was not available, prompting the usage of general parameters obtained from security experts. However, today this data is available using security metric data stored in the Common Vulnerability Scoring System (CVSS)[10] format, from the National Vulnerability Database (NVD)[1].

L. Swiler et al.[12] implemented a tool which constructs attack graphs by matching information for attack requirements to network configuration information. Attack paths with the highest probability of success can be identified by assigning weights to edges, based on estimates of the time to succeed in exploiting a vulnerability and the amount of effort to do so. The Naor and Brutlag's algorithm is used to computer ε-optimal paths and the number of ε-optimal paths an edge participates in. Nodes and edges of the attack graph occurring most frequently in the set of ε-optimal paths are defined as critical paths or paths with the highest probability of success. This technique thus far has only been suitable for smaller networks and only generates graphs forward from an initial node, rather than backward from a goal node.

R. Ritchey and P. Ammann[13] proposed the usage of a model checker to perform network vulnerability analysis. Vulnerabilities are encoded into a state machine description suitable to be processed by a model checker. The authors asserted that an attacker cannot acquire a certain privilege level on a particular host. The model checker is then able to either assure the correctness of this assertion or provide a counterexample with the detail of each level of attack.

O. Sheyner et al.[14] use algorithms which generate attack graphs based on a symbolic model checking approach. The algorithm attempts to produce a counterexample to demonstrate an attacker's path. An analysis technique called *Minimization Analysis* looks at identifying the most cost-effective method for a network administrator to prevent an attack from occurring.

[1]NVD is a repository of vulnerability management data, including security checklists, security related software flaws, misconfigurations, product names and impact metrics[11].

The essence of this analysis algorithm looks at determining a minimal set of atomic attack actions which must be prevented to guarantee that an intruder cannot reach his goal.

R. Rieke[15] provides a tool for vulnerability assessment, which computes attack paths and verifies certain security properties. Elements modelled include computer networks, vulnerabilities, exploits and attacker strategies, in order to automatically compute all possible attack paths. Future research objectives suggest the investigation of techniques to manage the state space explosion issue [2], by refining concepts of abstraction of system specifications.

R. Lippmann and K. Ingols[16] developed a tool called NetSPA, which is able to automatically construct and analyse attack graphs, aiming to discover firewall configuration defects and critical host vulnerabilities. Attack graphs require large amounts of source information, such as network configurations, host product information and vulnerability information. NetSPA receives most of its data by importing it from common sources, including vulnerability scanners, firewalls and online vulnerability databases.

I. Kotenko and A. Chechulin[17] designed a tool named AMSEC, which models attacks and evaluates security metrics for SIEM systems. AMSEC features an internal security repository and uses open security databases such as NVD to source vulnerability data. Attack trees are generated by incorporating information about service dependencies, zero-day vulnerabilities, predictions of possible attack actions, security metrics, and attack and response impacts. The tool functions in two modes, called design mode and exploitation mode. Design mode produces a set of attack trees based on network configuration information an security policies. Exploitation mode adjusts attack trees generated during the design mode and re-generates attack trees, by taking into account predicted attacker capabilities, to produce countermeasures. AMSEC addresses the state-space explosion problem by combining the use of attack graphs with service dependency graphs. The former provides insight into the evolution of an attack on a network, while the latter can track the impact of an attack as it advances. Moreover, the set of actions that an attacker can perform are determined by attacker models, which define an attacker's skill level. This limits the amount of exploit actions that can be performed at each step of attack, helping to prevent state-space explosion.

[2] A serious problem in model checking practices is the state space explosion issue. The number of states in a model grows exponentially according to the number of program variables in it. Exploration of the entire state space can become impractical when there are a vast number of states.

C. Sarraute et al.[18] consider an approach which targets real world scenarios. Attack models are represented in the Planning Domain Definition Language (PDDL)[19] language [3], allow the problem to be approached using a planner. The planner is integrated into a penetration testing suite, allowing for automated generation of attack paths for penetration testing scenarios and furthermore providing validation of attacks by executing corresponding exploits against actual target networks. The state space explosion problem caused by scalability issues is associated with most attack graphing techniques mentioned above, since they make use of model checking approaches. It has been noted that a major limitation of previous studies is that most attack graph algorithms have only been able to generate attack graphs on small networks with fewer than 20 hosts. Building a complete attack graph of medium or large-sized networks proves to be infeasible.

2.3 Automated Planning

2.3.1 Overview

Automated Planning is an area of Artificial Intelligence (AI) which focuses on the computational deliberation process of action sequences by measuring their expected outcomes. This deliberation process attempts to discover a best possible solution with respect to a stated goal, often requiring optimization in multidimensional space. A motivation for automated planning, in a more practical sense, comes with the consideration of designing information processing tools which provide resource planning.

2.3.2 Fields of Planning Application

Since planning domains consist of various types of actions, a number of fields or forms of planning applications exist. Some of the most common of these fields include path and motion planning, perception planning and information gathering, navigation planning and

[3]PDDL was developed as a standardized language used in benchmarking Artificial Intelligence planners. The common formalism allows planners to be compared, while attempting to solve planning problems.

manipulation planning[20, p. 3].

Path and motion planning relates to the design an construction of a path from an initial point to a goal point, taking into account the configurations of the system responsible for locomotion and its environment; for example, a vehicle or robotic arm.

Perception planning involves plans concerned with sensing actions. In scenarios requiring information gathering from numerous sources, it can aide the system in deciding where to look for it, which sensors are required to do so and how to use them. This field of planning is prevalent in robotics, where a controller may be responsible for obtaining states related with an environment or actions for motion control and planning[21].

Navigation planning is a combination of both motion and perception planning, used in reaching targeted spatial goals or area exploration. This technique aids robots in localizing themselves within an environment, mapping areas and generation or waypoints within these areas.

Manipulation planning deals with the handling or manipulation of objects, taking into consideration information which can be translated into mechanical actions. This information includes forces, touch, vision, auditory and other sensory information. Plans of this nature can be utilized in scenarios such as a production line in a factory, where robotic arms may be responsible for picking up objects and inserting them into assemblies for example.

2.3.3 Conceptual Planning Model

A plan can be described as a structure which aims to achieve certain objectives by using certain appropriate actions[20, p. 7]. It is an enumeration of a path through a state-transition graph, describing an ordered sequential list of actions to be performed. These objectives can be defined in several ways, each of which offer benefits suited to solving problems of specific domain types. Ways in which objectives can be described are listed as follows:

- An objective can be defined as a goal state or set of goal states, which are valid for any sequence of state transitions ending in one of the goal states

- A sequence of state transitions can be forced to adhere to certain conditions in order to achieve goal states
- Utility functions can provide conditions to state sequences, such as penalties or rewards. A goal may aim to satisfy various conditions required, while attempting to optimize these utility functions over the sequence of valid states
- Further tasks to be performed can be listed as objectives, providing a defined set of actions to complete

A state-transition system describes all possible evolutionary outcomes of a system. It can be described using the 4-tuple $\Sigma = (S, A, E, \gamma)$, where S is a finite set of states $\{s_1, s_2, ..., s_k\}$, A is a finite set of actions $\{a_1, a_2, ..., a_k\}$ and E is a finite set of events $\{e_1, e_2, ..., e_k\}$, $\forall\ k \in \mathbb{N}$. The last symbol of the 4-tuple is γ, a state-transition function with $S \times (A \bigcup E) \rightarrow 2^S$. An action $a \in A$ is valid in s if $\gamma(s, a) \neq \varnothing$. For all valid $a \in A$, the system transitions from state s to s', using $s' = \gamma(s, a)$[22].

The planning model, component interaction and plan execution flow can be visualized using Figure 2.3. The **planner** component is fed the description of the state-transition system Σ, an initial state s_0 and a set of objectives or goals S_g. It is responsible for the generation of a plan, according to its input. The **controller** receives a plan from the planner as well as a current state s from the state-transition system, from an observation function, as input. There exists an observation function, which contains partial knowledge of the current state, $\eta(s)$ such that η: $S \rightarrow O$, where $O = \{o_1, o_2, ...\}$, which provides an observation o to the controller. The **controller** yields an action a, which is dispatched to the state-transition system. Finally, the **state-transition system** evolves the system, using the state-transition function γ, based on event input and actions received from the controller component[20, p. 8].

The concept of states and state-transitions can be illustrated by the domain of a dock worker crane and transport robot, shown in Figure 2.4. This example is very simple and provides a very clear demonstration of how actions on objects in a domain result in disparate states. In this domain, the crane is capable of performing actions such as moving putting a container down(put), taking a container(take), loading a container onto the transport robot(load) or unloading a container from the transport robot(unload). On the other hand, the transport

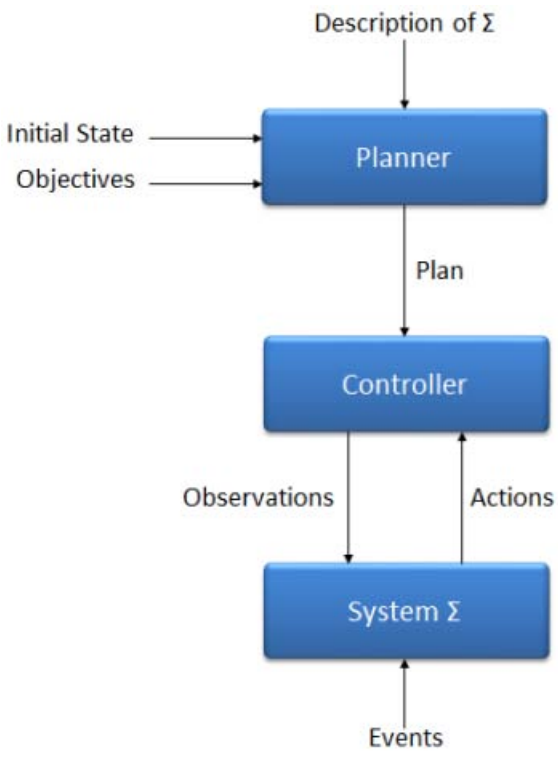


Figure 2.3: *Interaction between components involved in plan execution.*

robot is able to move to and from two separate locations(loc1 and loc2). The set of actions $A = \{\text{put, take, load, unload, move1, move2}\}$. The system can be in a number of configurations or states, which are defined as $S = \{s_0, s_1, s_2, s_3, s_4, s_5\}$. The blue arrows between each state represent the state-transition function γ indicating the transition from one state to another, based on the valid execution of an action a.

The above scenario can be likened to a cyber security problem where a network of devices is modelled instead of a dock with a crane and transport robot. This model could include various types of devices which run software applications and have connectivity relations. Configurations of these devices could be used to determine paths of compromised hosts in an attack.

Several types of planning techniques exist, including *Temporal Planning*, *Probabilistic Planning* and *Classical Planning*. *Temporal Planning* is used to examine scenarios where the timing of actions play a role in solving problems. This may be due to deadlines, resources

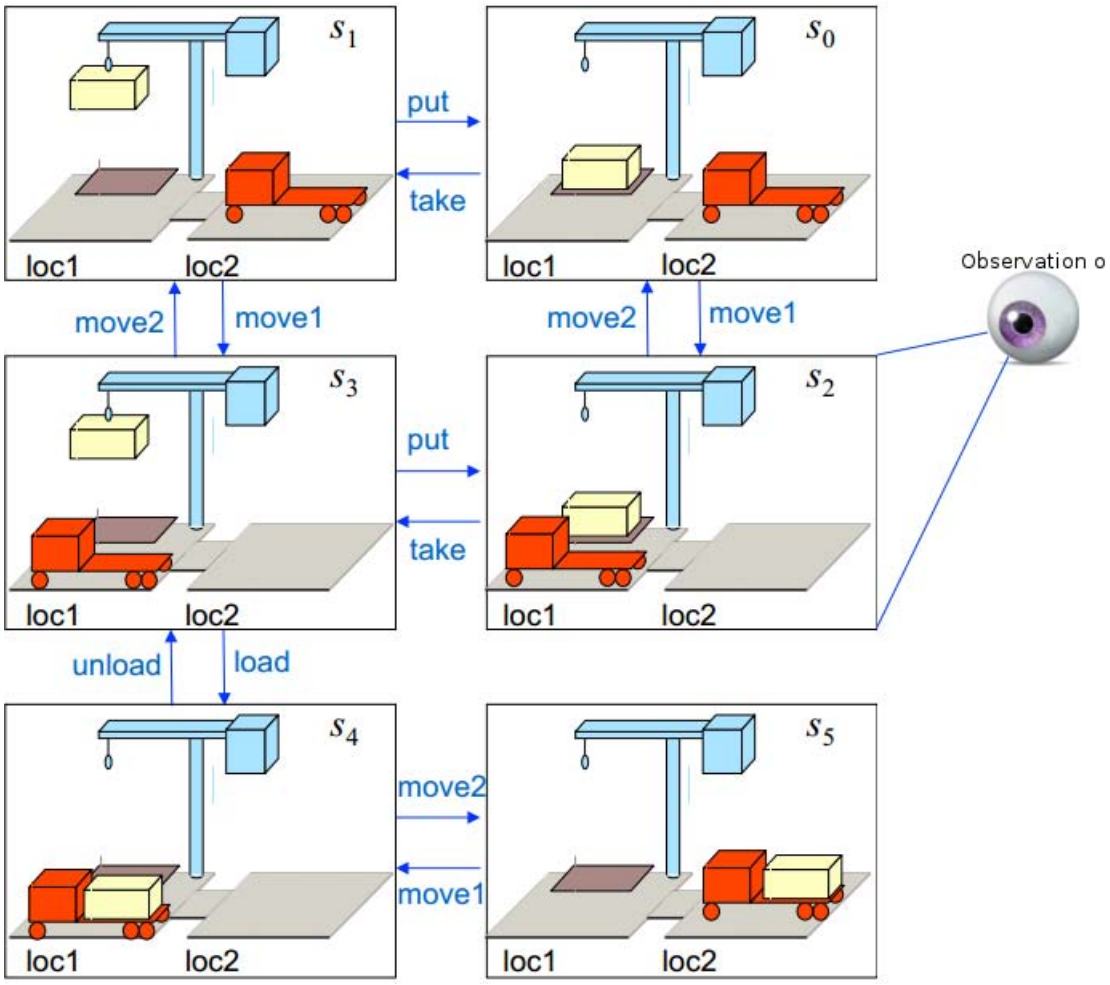


Figure 2.4: *State-transitions for a dock system involving a crane and transport robot[22].*

limited by time or external events acting on the system. This planning technique imposes time constraints on actions, which could symbolise the minimum or maximum duration taken to complete a task. *Probabilistic Planning* uses information on the probabilities of non-deterministic events, maximizing the probability of reaching designated goal sets. *Classical Planning* makes use of a unique known initial state and has durationless deterministic actions, making it most suitable to the application of attack graph generation. For this reason, *Classical Planning* will be elaborated on in the following sections.

2.3.4 Restricted Planning Model

The conceptual planning model stated above represents an abstraction which can be developed upon to create a more operational model. In reality, the world is extremely complex and can rarely be represented in states for a state-transition machine. Abstraction is key to simplifying real environments into forms which can be more easily modelled. These assumptions are described for the Restricted Planning Model below[20, p. 10]:

- **Assumption 1:** The state-transition system Σ has a finite number of states.
- **Assumption 2:** An observation of the state-transition system Σ contains complete knowledge its state. At the time of execution, the planner will know the state of the domain precisely.
- **Assumption 3:** The state-transition system Σ is deterministic, implying that there can be only one outcome for each action.
- **Assumption 4:** The state-transition system Σ is static, such that no changes occur until the controller issues an action.
- **Assumption 5:** Only restricted goals are handled by the planner, which are explicitly specified. Goals including states to be avoided, utility functions or constraints are not handled by the planner.
- **Assumption 6:** Solutions are an enumeration of a finite sequence of actions.
- **Assumption 7:** Actions and events are timeless and occur instantaneously.
- **Assumption 8:** The planner does not concern itself with any changes to the environment and reacts only to the initial and goal states provided to it.

Specification of these axioms becomes important for construction and abstraction of an operational model, which can be used to evaluate an environment.

2.3.5 Classical Planning

Classical Planning pertains to planning models which are restricted and follow **Assumption 1** to **Assumption 8**[20, p. 17]. The state-transition system is based on the *Restricted Planning Model*, with the exclusion of the set of events E, since the restrictive nature of the

model has no unexpected or unpredictable events in it.

The state-transition system Σ is thus reduced to a triple defined by $\Sigma = (S, A, \gamma)$, where S represents a set of all possible states, A a set of all possible actions the planner can execute and γ a set of all state transitions. Furthermore, a planing problem is defined as $\mathcal{P} = (\Sigma, s_0, g)$, where s_0 is an initial state and g is a set of goal states. A solution to $\mathcal{P}$ can be written as a sequence of actions $\langle a_1, a_2, ..., a_k \rangle$ corresponding to a sequence $\langle s_0, s_1, ..., s_k \rangle$ of state transitions, such that $s_1 = \gamma(s_0, a_1)$, $s_2 = (\gamma s_1, a_2), ..., s_k = \gamma(s_{k-1}, a_k)$, where s_k is a valid goal state[20, p. 17].

States

A state is in essence a set S of basic atoms which are used to define the details of a domain. Atoms represent facts which hold true for a state $s \in S$. A set S is guaranteed to be finite, since a domain is constructed of a finite number of atoms which may not be functions.

Operators

An operator consists of a triple $o = (\text{name}(o), \text{precondition}(o), \text{effect}(o))$. A name($o$) comprises of a unique operator symbol and every variable used in o, a precondition(o) includes literals which are required to be true for operator usage. For an operator o, $precondition^+(o)$ represents a positive precondition(a precondition which is true), while $precondition^-(o)$ a negative precondition(a precondition which is false). Lastly, an effect(o) contains literals which are made to be true after the successful execution of an operator.

Actions

The notion of an action can be understood to mean an instance of a particular operator. Consider a state s where an *action* a exists such that $precondition^+(o) \subseteq s$ and $precondition^-(o) = \emptyset$. Action a is defined as valid resulting in the evolution of s to state s' via the state-transition function $\gamma(s, a)$.

Representation of Planning Problems

A planning problem $\mathcal{P} = (\Sigma, s_0, g)$ can be syntactically defined as a planning statement $P = (O, s_0, g)$. P is in essence an instantiation of $\mathcal{P}$.

Plan Solutions

A plan is defined as a sequence of actions $\pi = \langle a_1, a_2, a_3,...,a_n \rangle \; \forall \; n \in \mathbb{N}$. Each a_i corresponds with an operator in O. A plan π is a solution to a problem P if and only if the action sequence of π can be successfully executed and also achieves some goal g[22].

2.3.6 State-Space Exploration

In classical planning there are several algorithms used in the exploration of state spaces. These search algorithms use nodes to represent domain states and arcs to indicate state transitions between states. The following sections will discuss two of the most commonly used algorithms; namely *Forward Chaining State Space Search* and *Backward Chaining State Space Search* or *Forward-Search* and *Backward-Search* respectively.

Forward Chaining State Space Search

Forward Chaining State Space Search is one of the two main techniques used in state space exploration, based on the usage of inference rules for its reasoning. Forward-Search is said to be *complete*, since if a solution exists to a problem P then a solution will be returned[22]. It takes as an input statement the instantiation $P = (O, s_0, g)$ of $\mathcal{P}$ and performs a Forward-Search returning a solution sequence, should the problem be solvable. It starts using available data and uses inference rules to extract more data until a designated goal is achieved. In Forward-Search, each new state s' is generated using $\gamma(s, a)$. Several implementations of Forward-Search include:

- Breadth-first search
- Depth-first search
- Best-first search

- Greedy search

The opposite state space search technique is *Backward Chaining State Space Search*, which will be further described in the next section.

Backward Chaining State Space Search

The concept of *Backward-Search* is to start at the goal and apply the inverse of operators in order to create subgoals, terminating at the generation of a set of subgoals satisfying the initial state specified[20, p. 73]. In Backward-Search, each subgoal or previous state s is generated using inverse transition function $\gamma^{-1}(s',\ a)$. An action is selected and a set of predecessor states is computed by applying the inverse transition function. A plan has been found once the inverse transition function generates the initial state.

2.3.7 GRAPHPLAN Planning

A drawback of previous search algorithms is the large branching factor, affecting state space search performance. This issue can be outlined in *Backward-Searching* where many actions actions are attempted which cannot be reach from an initial state. Aiming to address this issue and improve on planning techniques, A.L. Blum and M. L. Furst[23] with their planner, named GRAPHPLAN. The algorithm begins by constructing a *Planning Graph*, which encodes the planning problem in a way that offers constraints in an explicit manner, reducing search time. Planning Graphs incorporate domain information, goals, initial problem conditions and the concept of time. GRAPHPLAN uses the Planning Graph to guide its search in identifying solutions. It creates a relaxed problem[4], removing some restrictions from the original time, such that the relaxed problem can be solved in polynomial time. All solutions identified in the relaxed problem will included those solutions to the original problem. A modified search is performed on the original search, including only those actions occurring in solutions to the relaxed problem.

[4]A relaxed planning problem essentially simplifies problem by reducing the computational costs of calculating a solution by ignoring certain parts of its specification.

2.3.8 Heuristic Search Planning

The concept of HSP, developed by B. Bonet and H. Geffner[24], makes use of heuristics to guide its search for a goal. A heuristic function h is used to try solve problem $\mathcal{P}$, by focusing on a relaxed version of the problem $\mathcal{P}'$, which ignores *delete lists*.[5] The heuristic function $h(s)$ of a state s is set as an estimation of the optimal cost from a state s to a goal state. It guides the planner through a hill-climbing search and terminates once it has identified the goal. A hill-climbing search involves the selection of the best child nodes at each interval until a goal is achieved. In the application of HSP, the best child nodes are those which minimize $h(s)$. Those states which have already been computed are stored in rapidly accessible memory so that their values do not have to be recomputed and so that they are excluded from the remainder of the search.

2.3.9 Fast-Forward Planning

J.Hoffmann[26] introduced the *Fast-Forward* planner, which was the most successful automated planner in the AIPS-2000 planning systems competition.[6] The basic principle of Fast-Forward is based on the HSP system, differing mainly in the technique used in obtaining the heuristic function.

Fast-Forward approaches planning using a Forward-Search algorithm to explore the state space, guiding it using a heuristic function. Furthermore, the heuristic function is also derived by analysing a relaxed version of the problem, ignoring certain specifications, such as *delete lists*. Hoffmann went by obtaining a heuristic function $h_{FF}(S)$ of state S by running a relaxed version of GRAPHPLAN. This relaxed version first builds the *planning graph* until reaching all goals identified. The graph consists of layers of facts and actions, where the initial layer contains the facts which are applicable to state S. After applying all actions to these facts, the *add lists*[7] effects are added to those which already exist to form the second layer. This process is repeated to generate as many fact layers as required until all goals are

[5] Some planning systems refer to the set of negative literals or negative atoms as a delete lists[25].

[6] The Artificial Intelligence Planning and Scheduling Systems (AIPS) brings together researchers working on various fields such as planning, scheduling, learning and plan execution. They play host to the world planning competition, challenging the robustness of competitor planning software against a variety of problem types[27].

[7] The add list can be viewed as the counterpart to the delete list, containing a set of the positive literals or positive atoms.

reached. A relaxed plan can then be extracted by starting at the very last fact layer m of those including goals. Then for each layer i, should a goal exist in the previous layer i - 1, it is added as a goal to achieve for the layer i - 1. If no goal exists at layer i - 1, an action is selected from i - 1 which adds the goal and the action's preconditions into the goals of layer i - 1. Termination occurs upon reaching the very first layer, resulting in a relaxed action sequence $\langle O_0,..., O_{m-1}\rangle$. The solution length is estimated based on this sequence by using the function:

$$h_{ff}(S) = \sum_{i=1,...,m-1} |O_i| \forall i \in \mathbb{N}$$

While the heuristic function can obtain a solution in polynomial time, the actual process of searching the state space is still costly, prompting the usage of a more straight-forward method. Fast-Forward uses and *enforced* hill-climbing search, which evaluates all successors of a search state S for a state with a more optimal heuristic. If a better heuristic than that of S is not discovered, the successor's successors are evaluated and so forth. A state S' with a better heuristic than S is then added to the current plan, with the search continuing from S' as the new starting state. In summary, Fast-Forward searches through states using a breadth-first search which enforces solutions to have strictly better results. Should this technique fail to find a solution, Fast-Forward switches to a weighted A* algorithm.[8]

An analysis comparing the performance of the Fast-Forward system and HSP was done. Three main fields were focused on, including heuristic function acquisition, searching method and pruning techniques, revealing an improved performance with Fast-Forward for each domain tested.

2.3.10 Planning Domain Definition Language

The Planning Domain Definition Language (PDDL) is an action-centered language used in automated planning to describe the environment of a domain. It was initially developed for the International Planning Competition as a universal planning language to be used in benchmarking planners. Such descriptions entail the kinds of predicates or literals available,

[8] A* uses a best-first search to find a least-cost path from an initial node to a goal node. It uses a heuristic to follow the path of lowest cost, keeping a sorted queue of alternate paths as the algorithm progresses. The algorithm differs from a greedy best-first search in that it tracks the distance already traveled.

the structure of actions and the effects they produce[28]. PDDL was developed by deriving certain concepts from several previous planning languages including ADL[29], SIPE-2[30], Prodigy-4.0[31], UMCP[32], UNPOP[33] and UCPOP[34].

Domain Definition

In general, the domain definition describes the domain predicates and operators(known as actions in PDDL), as well as constants and static axioms. In the context of a domain, predicates have no intrinsic meaning and are use only to specify predicate names used in a domain and their parameters. A predicate evolves according to the evaluation of an action, whose effect can modify it, and by its initial instantiation in the problem definition. Actions take in a list of predicates as parameters on which they operate. An action can be separated into a *precondition* and *effect* section. The precondition section specifies a goal that must be satisfied before an action is executed. These goals are based on the states of precondition parameters specified as action input. Preconditions which are evaluated as *true* lead to the execution of effects, which modify the current state of the domain by altering its predicates.

Problem Definition

The problem definition is responsible for specifying initial states of object in the problem instance as well as the overall goal to search for. All atoms stated in the initialisation section are set as true in the initial state, while all others are set as false. The goal description is used to assign desired goals to a planning problem. These goals correspond with objects or constant names rather than quantities.

2.4 Vulnerability Data Sources

In order to determine whether an exploit action can be executed during a specific state, certain requirements must be satisfied. One of these preconditions is that there exists a vulnerability on a reachable host, which would allow an attacker to either elevate his privilege level on his currently compromised machine or gain privileges on an adjacent machine. Vulnerable hosts are discovered based on software installed on these hosts. Information regarding the software vendor, name and version are cross-referenced with publicly available

vulnerability information to determine the types of results an attacker could achieve. Several vulnerability databases are available online, which would be suited for this solution. These are:

- National Vulnerability Database (NVD)[11]
- Open-Source Vulnerability Database (OSVDB)[35]
- CERIAS Cooperative Web Vulnerability Database[36]
- Greedy search

In the Design Chapter we will compare the pros and cons of each of these databases, providing insight into the most suitable choice to use for this research.

2.5 Chapter Summary

This chapter outlined three fields of interest, namely Security Information and Event Management (SIEM), security attack graphs and automated planning. SIEM has been widely utilized within large enterprise networks as a means of centralised log management and real-time analysis. A predictive security analysis approach could benefit from the pre-collected and normalized log data of SIEMs, offering administrators an opportunity to identify potential threats against a monitored network. These potential threats can be represented on a network-wide scale by using attack graphs. Attack graphs offer a graph-based approach to analysing network vulnerabilities, by showing how individual attacks can be composed to create larger attack paths. Searching through entire state spaces for medium or large networks quickly becomes infeasible due to the state-space explosion problem, prompting the use of more practical means of graph exploration. Automated planning algorithms aid in state-space exploration by guiding the search using heuristics rather than performing exhaustive searches. In the following chapter we will consider how these techniques can be adapted and improved. We will provide a design for a prototype and specify an approach for the extraction of exploit information from publicly available vulnerability databases.

Chapter 3

Design

In this chapter, we look at the design of the system architecture and outline the process employed in evaluating network security. The chapter is further broken down into three sections, representing design layers, which serve to demonstrate the information flow throughout the framework.

3.1 System Overview

In this chapter, the proof of concept framework is presented with an outline of its design specifications. Network Vulnerability Analyser (NVA) uses a topological model of a network to simulate the mechanics of an attack on it. In essence, NVA produces a specification for a multi-step attack based on a network's structure, device connectivity relations and software installed on devices. It is aimed at benefiting network administrators, since by outlining an attacker's most likely strategy of attack one can anticipate potential attacks and apply suitable countermeasures.

3.1.1 Typical Strategies of Attack

It is crucial to understand the typical methods employed by malicious users to compromise critical network resources, in order to extrapolate a basis for the requirements which drive the solution. Throughout an attack on network infrastructure, an attacker constantly

performs reconnaissance on newly discovered devices as his knowledge of the infrastructure increases. Reconnaissance incorporates enumeration of device information which could further aid in the attack, such as services exposed to him. Once versioning information is retrieved regarding these exposed services, it can be cross-referenced with a vulnerability database to explore potential weaknesses which may allow him unauthorized access. Successful exploitation of discovered vulnerabilities may lead to access to the vulnerable device, with administrative or user-level permissions. It may not be necessary for an attacker to gain administrative access to a target if the goal is to merely use it as a pivoting point to launch attacks against other devices. This process of vulnerability enumeration and exploitation is the driving factor behind the design of NVA. While in reality there are many more factors to take into account when exploiting a vulnerability, these complexities were ignored in order to focus on the application of the automated planning aspect of attack path generation.

3.1.2 Considerations for Design Requirements

NVA requires information from numerous sources in order to be able to perform attack path computations. The solution is dependent on a computer-readable description of the network topology being assessed and access to vulnerability information. Conducting automated discovery and mapping of network topology requires the discovery of routers, switches, firewalls, load balancers, servers and workstations. This task is requires a significant amount of time and effort and was not deemed as part of the scope of this thesis. Consequently, static XML descriptions of network topologies were generated for testing purposes. The potential for NVA to receive topological descriptions from automated mapping and discovery tools has been left as future work.

The solution focusses on the exploitation of vulnerabilities which result in unauthorised access to devices. As such it can only evaluate a subset of scenario, since it does not model attacks which affect the availability of devices or confidentiality of data.

The architecture of NVA is illustrated in Figure 3.1, and is comprised of three layers: **(I) Information Aggregation**, **(II) Attack Graph Computation** and **(III) Visualization**.

Information Aggregation is performed by the *Network Configuration Transformer*, whose responsibility is taking an XML network configuration file and parsing it to create an object-

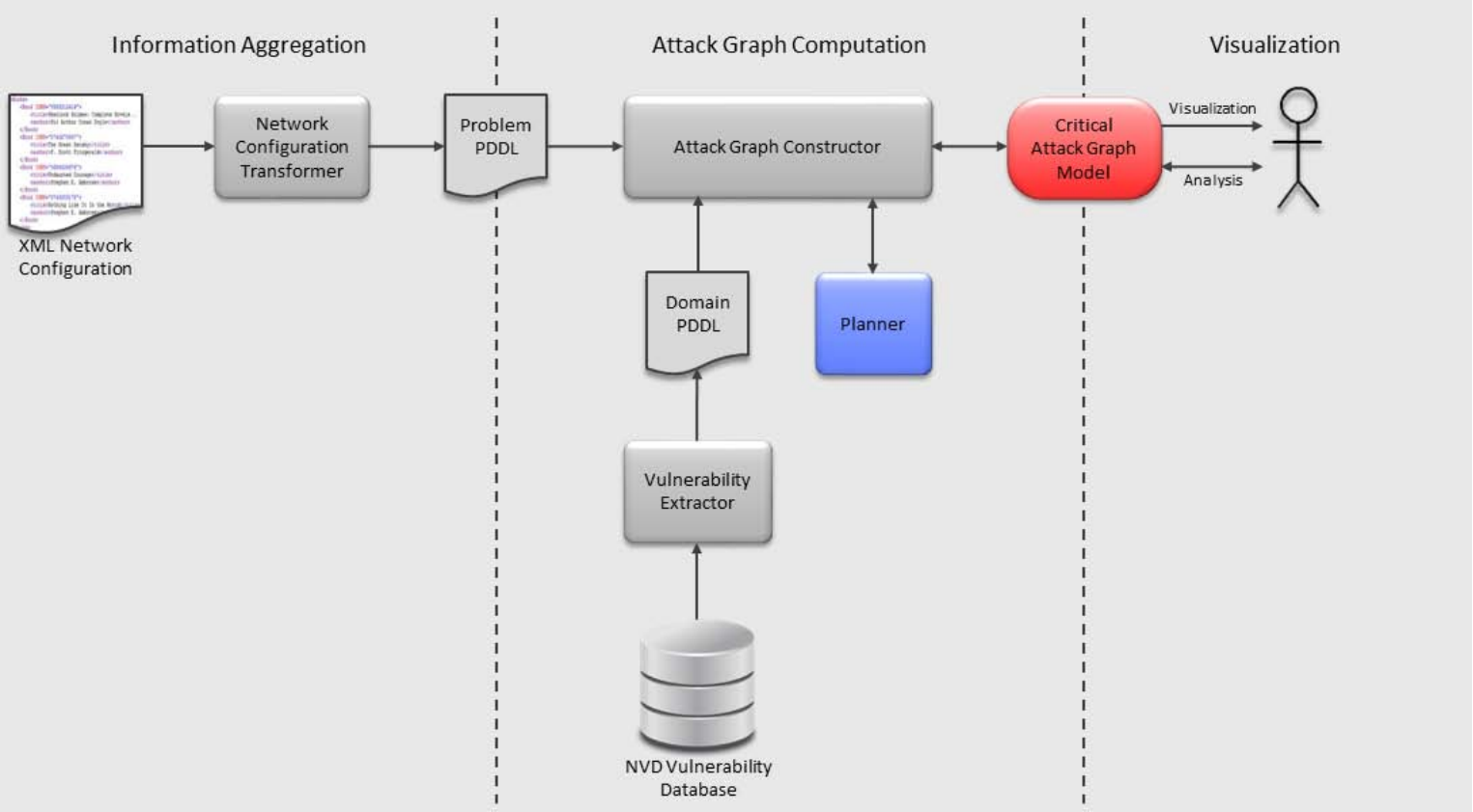


Figure 3.1: *An overview of NVA architecture, including various layers involved in computation of attack graphs.*

based model of the network. The network model is stored in memory so that it can be easily manipulated for graph analysis operations. The *Network Configuration Transformer* also generates an artefact of the network model called *Problem PDDL*, which acts as an interface between the **Information Aggregation** layer and **Attack Graph Computation** layer, as well as input for the *Attack Graph Constructor* module. The *Problem PDDL* is basically a description of the scenario to be processed by the planner, described in more detail in section 3.5.6.

The **Attack Graph Computation** layer forms the bulk of the processing load, incorporating the most vital functions of NVA in the attack graph generation process. At the center of this layer is the *Attack Graph Constructor* module, responsible for managing the network model and all generated attack graph models for the network, storing all executable exploit actions and most importantly, interfacing with the planner. Before the *Attack Graph Con-*

structor can perform any processes it needs to retrieve a detailed description of possible executable exploits it can use, obtained directly from the vulnerability database. Vulnerabilities are extracted from the vulnerability database using the *Vulnerability Extractor*. The *Vulnerability Extractor* parses the vulnerability database and creates a hash table of vulnerability objects using unique vulnerability identifiers as keys. The vast amount of vulnerabilities stored in the database are not all required in computation of attack graphs. Therefore, the *Vulnerability Extractor* cross-references the software installed on devices in the network with extracted vulnerabilities to create a subset of vulnerabilities, used to generate atomic exploit actions. By creating a subset of vulnerabilities it minimizes the amount of possible actions planner has to perform, improving performance time. The *Vulnerability Extractor* then uses the metrics available within the extracted vulnerability subset to create atomic exploit actions by enumerating preconditions for successful exploitation and the postconditions thereafter. These exploits are listed in the *domain PDDL*, which is fed to the *Attack Graph Computation* module. The *Domain PDDL*, *Problem PDDL* and an attacker's target are crucial elements in producing an attack plan. They are used as input to the planner, where the *Problem PDDL* describes the environment of the problem and the *Domain PDDL* lists the kinds of actions its capable of performing.

Should the planner identify a solution to the problem it outputs a *Critical Attack Graph Model*. This model acts as an interfacing component between the **Attack Graph Computation** layer and **Visualization** layer and is displayed to the user in NVA's attack graph visualizer. The user can interact with the *Critical Attack Graph Model* via the GUI and regenerate attack graphs dynamically, taking into account any modifications.

3.2 Extracting Data from a Vulnerability Database - Layer I

3.2.1 Sources of Vulnerability Data

Vulnerability information is generally available from two sources; namely commercial and non-profit organizations. Since commercial security organizations often do not publish all of their vulnerability information, non-profit sources are focused on. This section is involved

in **Information Aggregation** of vulnerability data required in device product evaluation. It also examines some of the most popular non-profit vulnerability providers. The type of vulnerability information each of them offers is analysed to determine which source may be the most beneficial in the context of attack graph generation.

Open-Source Vulnerability Database (OSVDB)

OSVDB[1] is geared towards offering detailed and current information to the online community. Launched in 2002, the project aimed to implement a vulnerability database which is open for free use and has no restrictions on its content. OSVDB runs as a web application with a front-end offering vulnerability searches and a back-end whose content is openly available to updates[35]. A relational database is used to store vulnerability information and is maintained entirely by the open-source community. OSVDB essentially relies on security experts and the power of the open-source development model to identify and document vulnerability information in its database.

CERIAS Cooperative Web Vulnerability Database

Based on a previous vulnerability database project whose information was kept internally, the CERIAS Cooperative Web Vulnerability Database (WebVDB) was built into a more robust system made available to the public. Similar to OSVDB, WebVDB uses a relational database engine efficient for storage and retrieval of large amount of vulnerability data. However, WebVDB makes use of a voting mechanism utilized by groups of reviewers to verify the correctness of vulnerabilities submitted by users. By voting a reviewer indicates their level of confidence in a vulnerability's correctness. The accumulation of these votes decide whether a vulnerability and its information is released to the public or if any corrections are necessary. Furthermore, unlike OSVDB, WebVDB is not open-source and relies on two types of user groups to manage the vulnerabilities in the database. The first group, known as *normal users*, have sufficient privileges to submit vulnerabilities and edit their submissions. The second group, known as *editors*, have a privilege level which comprises of *normal user* privileges as well as the ability to edit any vulnerabilities and vote for them[36].

[1] http://www.osvdb.org

National Vulnerability Database (NVD)

NVD[2] was created as a repository of standardised vulnerability data for the U.S. government. Structured vulnerability data enables the automation of vulnerability management, security measurement and compliance. It incorporates databases of security checklists, security related software flaws, misconfigurations, product names and impact metrics. NVD is a project sponsored by the U.S. Department of Homeland Security's National Cyber Security Division and is geared towards helping agencies such as the National Security Agency (NSA) and the National Institute of Standards and Technology (NIST).

3.2.2 Motivation for Vulnerability Source Selection

In this section we examine the information provided by the aforementioned vulnerability databases. A comparison is made based on the information they provide on a vulnerability, to help make an informed decision about which one will be most beneficial for building attack graphs. The key aspects considered for analysing how databases present vulnerabilities include naming conventions, descriptions, Common Vulnerabilities and Exposures (CVE)[37] reference, operating system, software, access vector, access complexity, authentication, CVSS, impact, references, format, release date and last update. Table 3.1 provides an overview of the information offered by OSVDB, WebVDB and NVD.

[2]http://nvd.nist.gov

Table 3.1 A comparison of data offered by several popular vulnerability databases[38].

	OSVDB	**WebVDB**	**NVD**
Title	✓	✓	
Description	✓	✓	✓
CVE Reference	✓	✓	✓
OS	✓	✓	✓
Software		✓	✓
Access Vector	✓	✓	✓
Access Complexity			✓
Authentication			✓
CVSS			✓
Impact	✓	✓	✓
References	✓	✓	✓
Format	HTML	CSV/HTML/MySQL/XML	HTML/XML
Release Date		✓	✓
Last Update		✓	✓

When considering which database to use as a data source, the extractability of data must be examined. For example, a database which uses easily identifiable fields makes for easier extraction with a parser, as opposed to entries which are only human-readable. This has been a deciding factor in the selection of OSVDB, WebVDB and NVD, since they present vulnerability data in common formats which aide extraction. Fields used in comparing each of the vulnerability databases are further described below.

The first group of fields is used in identifying vulnerabilities. Human-readable titles are provided by OSVDB and WebVDB to provide a brief explanation of what a vulnerability is, however this information is not useful in terms of attack graph construction. A more thorough description of a vulnerability is offered by all three databases, providing human-readable information to better understand the detail behind it. Vulnerabilities are described using a standardised format using the Common Vulnerabilities and Exposures list, which provides identifiers for all known vulnerabilities[37]. This standardised identifier is used widely amongst the majority of vulnerability databases and is the generally accepted format for labelling vulnerabilities. Universal usage of CVE identifiers is key since it supplies unique identifier for each vulnerability which is computer-readable. All three databases utilize CVE identifiers as a naming convention for their vulnerabilities.

The second group of fields is concerned with information which NVA can use in attack graph construction and reachability of targets. The access vector field describes how an attacker would need to access a vulnerable device to exploit its vulnerability. Access complexity defines the difficulty an attacker faces in successfully executing an exploit against a vulnerability. Authentication defines the amount of times an attacker is required to authenticate with a target device to exploit it. All of the fields in this group are further explained in the Common Vulnerability Scoring System (CVSS) in the next section.

The third group contains information which NVA is able to use in identifying critical attack paths within networks. The CVSS severity score defines the risk of a vulnerability and is used in determining the overall security level of a network. The impact field can be used to gather information on the postconditions of an attack. For example, it may state that successful exploitation of a vulnerability offers an attacker root access to the target device, providing the attacker with access to every given resource. NVA uses this kind of information in determining threat mitigation strategies, aggregating the impact of vulnerabilities to compute the maximum damage an attacker may be able to cause on a network.

In the fourth group we consider any references provided by vendors of vulnerable software. They can supply more information for a vulnerability, including patches and all affected software versions. These references can help a user to find more information on updates or patches for vulnerable software. Another advantage of these references is the ability to find information for a vulnerability on other databases without having to know the vendor-specific ID for the weakness. By providing URLs to other sites hosting information about a particular vulnerability, it allows pages to be found more easily.

Most vulnerability information is only offered in HTML format as descriptions. The disadvantage of HTML is that it makes data extraction challenging, since a site-specific parser needs to be built to scrape vulnerability data out. This process can be computationally expensive and detracts from actual problem at hand; attack graph construction. Only the NVD and OSVDB offer data in additional formats to HTML, such as XML.

Finally, the last group presents the release date and latest update date. While this information provides no extra value in terms of attack graph construction it helps in determining

the freshness of a report and whether a vulnerability description is outdated.

From Table 3.1 it can be seen that the NVD offers most of the required vulnerability information. A crucial advantage it also provides is that it makes data available in an XML format, drastically reducing the amount of work needed for data extraction. Furthermore, NVD includes vulnerability information using the CVSS, which includes all of the necessary fields required for NVA to build attack graphs. For these reasons, NVD was selected as the source for vulnerability information to use in attack graph generation.

3.3 Common Vulnerability Scoring System

Vulnerability assessment has been challenging in the past, since vulnerabilities need to be identified and analysed across a variety of disparate hardware and software platforms. An organization may be host to a mountain of vulnerabilities, where each vulnerability poses a different level of risk. When there are so many to fix it can be difficult for an IT manager to know where to start. The Common Vulnerability Scoring System (CVSS) helps to address this issue by offering an open framework which can aide in prioritizing vulnerability resolution, according to a severity score[39]. By using an open framework it allows any user to see individual characteristics of a vulnerability used in deriving its severity score.

While no organization owns CVSS, it is under custodial care of the Forum of Incident Response and Security Teams (FIRST).[3] Organizations that make use of CVSS include vulnerability bulletin providers, software application vendors, private-sector user organizations vulnerability scanning and management organizations, security risk management firms and researchers.

3.3.1 Ranking Vulnerabilities Using Metrics

CVSS can be broken down into three groups of metrics; namely Base, Temporal and Environmental, each of which contain subsets of metrics. Figure 3.2 shows these groups as well as the metrics they comprise of.

[3] www.first.org/cvss

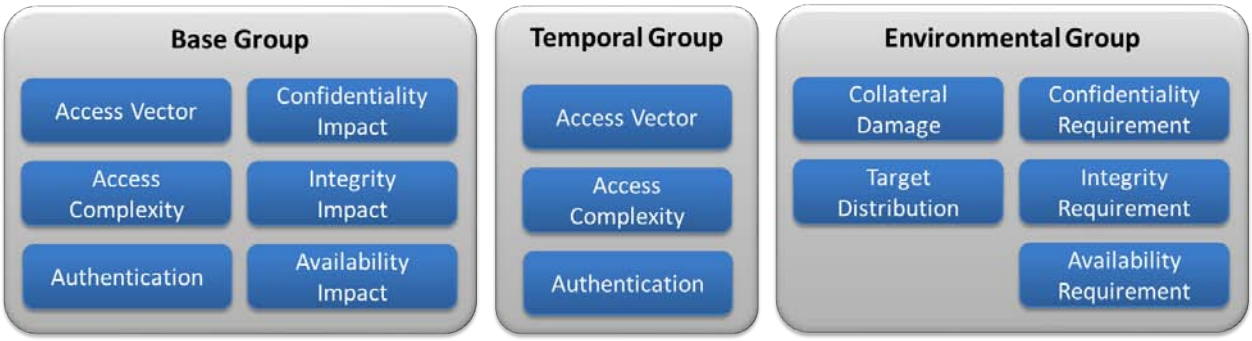


Figure 3.2: *CVSS groups comprising of several metrics which determine a vulnerability's severity score.*

The CVSS score is computed using a function which is a combination of the base, temporal and environmental metric groups. Each metric group is represented as a vector function, comprised of the values for each metric shown in Figure 3.2. This not only provides the user with an overall CVSS score but also a more detailed break down which can be used to communicate exactly how the score for a vulnerability is derived.

3.3.2 Base Metrics

The base metric group looks at those characteristics of a vulnerability which are static and independent of time and the infrastructure environment. A description of how a vulnerability is accessed and the conditions required to do so is provided by the Access Vector, Access Complexity and Authentication metrics. On the other hand, Confidentiality Impact, Integrity Impact and Availability Impact define the effect of the successful exploitation of a vulnerability. The three impact metrics help to define the degree of loss of confidentiality, integrity and availability.

Access Vector

This metric describes the type of access an attacker requires to exploit a vulnerability. Table 3.2 shows the values available to the type of access to a target host. The ability for an attacker to exploit a vulnerability remotely will achieve a much higher score than one that requires local access to a targeted device. This owes to the fact that a remote vulnerability allows access to a host across a network, without having to perform any preparatory attack

steps to initially gain access to the host via another vulnerability.

Table 3.2 Access vector values[39].

Metric Value	Description
Local	This requires the attacker to have local access to the target device. What this entails is either physical access or access via a shell account.
Adjacent Network	Vulnerabilities which are exploitable via an adjacent network offer an attacker access if they have access to the broadcast domain or IP subnet of the target device.
Network	A vulnerability which requires network access is dangerous since it does not require local network access or physical access. It is bound to the network, making it increasingly exploitable since devices are often connected to the network. Without firewalls in place within a network infrastructure, these vulnerable devices may be visible to anyone on the internet. This danger achieves network accessible vulnerabilities the highest score.

Access Complexity

The complexity of an attack on a vulnerable device is categorized with this metric, once the attacker has gained access using the aforementioned access vector. The degree of access complexity is defined by the number of steps required to successfully exploit a vulnerability. For example, a buffer overflow vulnerability in a Microsoft Internet Information Server (IIS) may be executed as soon as the attacker gains access to the target device. Conversely, a prerequisite for a vulnerable chat client may be that a user needs to interact with the attack, such as clicking on a link provided by the attacker. Such a scenario would achieve a lower score than the first example. Degrees of access complexity are further explained in Table 3.3.

Table 3.3 Access complexity values[39].

Metric Value	Description
High	Require specialised access conditions before a vulnerability can successfully be exploited. This may entail the attacker already having a certain privilege level or making use of social engineering to carry out an attack. Furthermore, it might require certain rare and vulnerable conditions to be in place.
Medium	Access conditions are fairly intricate but not as difficult to satisfy. This level of complexity might involve an attacker gaining access to the target device within a broad usergroup and could even include untrusted users. It could also require that some information be obtained to carry out and exploit.
Low	This degree of access complexity does not require any specialised conditions to be in place. It is likely to be present in scenarios where a vulnerable product requires access to a wide range of systems and usergroups or an attack require little skill or additional information to be gathered.

Authentication

This metric looks at the amount of times an attacker is required to authenticate with the target device in order to exploit a vulnerability on it. For simplicity's sake, it does not take into account the complexity of the authentication process, only the number of authentications required. The scale of authentication values consists of three values shown in Table 3.4.

Table 3.4 Authentication values[39].

Metric Value	Description
Multiple	The attacker needs to authenticate with the target device more than two times.
Single	The authentication process only needs to be executed once for an attacker to exploit a vulnerability.
None	No authentication is required to exploit a vulnerability.

Confidentiality Impact

Data confidentiality is whether information stored on a system is protected against unintended or unauthorized access. In this context, confidentiality impact is the measure of

damage on confidentiality should an attacker successfully exploit a vulnerability on a system. Possible values for this metric are listed in Table 3.5, decreasing in confidentiality impact score.

Table 3.5 Confidentiality impact values[39].

Metric Value	Description
Complete	An attacker has total information disclosure of the target device, allowing him to read all of its data.
Partial	Information is leaked to the attacker, however the attacker does not have control over what is revealed. In such a scenario only limited amounts of information are disclosed.
None	There is no risk of any information being compromised on the target device.

Integrity Impact

Integrity refers to maintaining and assuring the accuracy and consistency of data in a system. Integrity impact describes the level of compromise and effect caused by an attacker on a target device's data integrity. Table 3.6 shows the values for this metric, decreasing in score.

Table 3.6 Integrity impact values[39].

Metric Value	Description
Complete	The entire system is compromised and all data is at risk of being altered. No data on the target device is protected from modification.
Partial	The system is partially impacted by data integrity issues. The attack is capable of modifying some data, however does not have complete control over what can be modified.
None	No impact on data integrity is caused.

Availability Impact

Availability describes the level of accessibility of information resources. Attacks against the availability of systems are referred to as Denial of Service (DoS) attacks, and include the consumption of network bandwidth, processor cycles, disk space and shutting down services. Availability impact is a metric which measures the impact a vulnerability has

on the availability of services if successfully exploited. Values defined for this metric are described further in Table 3.7.

Table 3.7 Availability impact values[39].

Metric Value	Description
Complete	The attacker has complete control of the target device's resources and can render them totally unavailable.
Partial	Resource availability and performance is drastically reduced. Resource availability is interrupted or limited access is available.
None	No impact on data availability is caused.

3.3.3 Temporal Metrics

The temporal metrics group is concerned with elements of vulnerabilities which change over time. Factors that are considered in this group included the exploitability of a vulnerability, the remediation status of it and the level of confidence in a vulnerability's report. The temporal metric group is optional, therefore each metric group has a value which can be used that has no effect on the overall score.

Exploitability

Exploitability looks at the current availability of methods or source code which can be used to exploit a vulnerability. Furthermore, it considers how easy publicly available source code or methods are to use, increasing the score if they require little skill. The possible values used in the exploitability scoring system are listed in Table 3.8.

Table 3.8 Exploitability values[39].

Metric Value	Description
Unproven	No exploit code or methods are available.
Proof-of-Concept	When vulnerabilities are initially discovered in the real world, proof-of-concept code is developed, which acts as very functional exploit code for demonstration purposes only. This code is not functional in all scenarios and generally requires a large amount of modification to suit specific situations.
High	The vulnerability has widely available exploit code or needs no code at all to execute thoroughly documented steps manually. In such cases, coded or methods work every time.
Not Defined	Assignment of this value will exclude this metric from the overall score.

Remediation Level

This metric looks at the state of a vulnerability's patch. When a vulnerability is first discovered it usually requires that the vendor or a third-party organization release a patch or upgrade. Prior to this, hotfixes or workaround may be released to address the issue in the interim. The remediation level score is affected by the stage of remediation a vulnerability is in, where an unpatched vulnerability attains the highest score and decreases throughout each stage. Table 3.9 shows the possible values available for this metric.

Table 3.9 Remediation level values[39].

Metric Value	Description
Official Fix	A total update to the issue is provided by the software vendor.
Temporary Fix	The official vendor provides a temporary fix while a more steady fix is developed on.
Workaround	There is a non-official workaround or patch provided by other users who have encountered the vulnerability.
Unavailable	No solution is available to the vulnerability or it is impossible to fix.
Not Defined	Assignment of this value will exclude this metric from the overall score.

Report Confidence

Report confidence looks at how credible the official report of a vulnerability really is and the level of detail provided on it. Often, vulnerabilities are publicised without any technical details outlining the problem. At a later stage these vulnerabilities may be confirmed by the software vendors or author of the vulnerability. The degree of confidence in a report is broken down and shown in Table 3.10, where confirmed reports gain the highest score possible.

Table 3.10 Report confidence values[39].

Metric Value	Description
Confirmed	A vulnerability is confirmed to exist by either the software vendor or author of the vulnerability. High levels of confidence are associated with these reports and they may include proof-of-concept exploit code.
Uncorroborated	Multiple non-official reports exist for a vulnerability, which may have conflicting technical details.
Unconfirmed	A single unconfirmed report exists or multiple conflicting reports exist. There is a very low level of confidence in such a report and they usually surface from hacker sources.
Not Defined	Assignment of this value will exclude this metric from the overall score.

3.3.4 Environmental Metrics

The environmental group, similar to the temporal group, is used to paint a more intricate picture of a vulnerability and the effects it has on a computing environment. It tries to capture the details of a vulnerability in the environment, since different environments can affect the risk level a vulnerability poses to an organization. Moreover, the environmental metrics group is also optional and can be disregarded from the total severity score. Included in this group is the collateral damage potential, target distribution and security requirements of an environment.

Collateral Damage Potential

Collateral damage potential measures the danger a vulnerability can potentially have on a working environment, ranging from loss of life or damage physical assets to theft of data. Incorporated into this metric is the potential for a vulnerability to affect an organization's productivity or revenue. Vulnerabilities with a higher collateral damage potential in a particular organization gain a higher score. Table 3.11 reflects the different kinds of values available for rating this metric.

Table 3.11 Collateral damage potential values[39].

Metric Value	Description
High	Successful execution of an exploit against the organization has disastrous consequences, with large-scale physical damage or property loss. The repercussions of an attack on such a vulnerability will affect the organization's productivity or revenue.
Medium-High	Exploitation of such a vulnerability results in a significant degree of physical damage or property loss.
Low-Medium	A moderate degree of physical damage or property loss will result from the successful exploitation of the vulnerability.
Low	Exploitation of this vulnerability results in slight loss of property or physical damage.
None	No loss of property or physical damage is experience by successful exploitation of this vulnerability.
Not Defined	Assignment of this value will exclude this metric from the overall score.

Target Distribution

This metric is used as an indicator for the quantity of devices in an organization which are affected by a vulnerability. The target distribution metric is very much environment-specific and depends on the types of software and services present on devices in a network. The proportion of devices with an existing target vulnerability determines the score level for this metric, where a larger proportion implies a higher score. Table 3.12 indicates the proportion ranges used in deciding this score.

Table 3.12 Target distribution values[39].

Metric Value	Description
High	Between 76% and 100% of devices are at risk in the environment.
Medium	Between 26% and 75% of devices are at risk in the environment.
Low	No target devices exist in the environment.
Not Defined	Assignment of this value will exclude this metric from the overall score.

Security Requirements

Three submetrics comprise security requirements; namely availability, confidentiality and integrity. Base metric scores can be adjusted by modifying the availability, confidentiality and integrity of assets in an organization, depending on the organization's priorities. For example, an organization with assets which are key to business functions may be assigned higher priority in availability, rather than confidentiality and integrity. Security requirement metrics correspond with their counterpart impact metrics: availability impact, confidentiality impact and integrity impact. These metrics are linked, in that a higher availability requirement increases the weighting of the availability impact and so forth. Table 3.13 shows the possible values used in this metric.

Table 3.13 Security requirements values[39].

Metric Value	Description
High	Loss of confidentiality, integrity or availability will have disastrous effects on the organization.
Medium	Loss of confidentiality, integrity or availability will have serious effects on the organization and may affect users.
Low	Loss of confidentiality, integrity or availability will only have limited effects on the organization.
Not Defined	Assignment of this value will exclude this metric from the overall score.

3.3.5 Calculation of Vulnerability Severity

CVSS provides a framework for the calculation of a vulnerability's severity rating, between one and ten. Each of the scores for the above mentioned metric groups is calculated using a set of independent equations. Using the generated scores, an administrator can make

an informed decision on which devices and software to patch in a network. This project looks at the automated aspects of identifying critical paths of devices and software to patch. For this reason, the temporal and environmental metric groups have been neglected, since they require tailored input from and administrator or security expert. The main source of vulnerability information will be deduced from the base metric group. CVSS states the formula to compute the base metric group as follows.

```
BaseScore = round_to_1_decimal(((0.6*Impact)+(0.4*Exploitability)1.5)*f(Impact))

Impact = 10.41*(1-(1-ConfImpact)*(1-IntegImpact)*(1-AvailImpact))

Exploitability = 20*AccessVector*AccessComplexity*Authentication

f(impact) = 0 if Impact=0, 1.176 otherwise

AccessVector = Local Accessible: 0.395
               Adjacent Network Accessible: 0.646
               Network Accessible: 1.0

AccessComplexity = High: 0.35
                   Medium: 0.61
                   Low: 0.71

Authentication = Requires Multiple Authentications: 0.45
                 Requires Single Authentication: 0.56
                 No Authentication Required: 0.704

ConfImpact = None: 0.0
             Partial: 0.275
             Complete: 0.660

IntegImpact = None: 0.0
              Partial: 0.275
              Complete: 0.660
```

```
AvailImpact = None: 0.0
              Partial: 0.275
              Complete: 0.660
```

The above equations are further illustrated using an example of a vulnerability taken directly from NVD. For the purpose of this project only the base metric group is used to provide a base severity score, using information provided with the vulnerability report. NVA uses base metric scores, which have already been precomputed and stored in NVD.

3.3.6 Example of Vulnerability Severity Score Calculation

This example will look at a vulnerability called **CVE-2002-0392**, discovered in June 2002. It was a vulnerability in the Apache Web Server which, when exploited successfully, could lead to a denial of service attack or the execution of arbitrary code with web server level privileges.

This vulnerability can be exploited remotely, meaning that the *Access Vector* is *Network Accessible*. No additional circumstances, such as social engineering attack vectors, are required in the execution of an exploit against this vulnerability. This infers that the vulnerability's *Access Complexity* is *Low*. The *Authentication* metric is set to *None* since no authentication with the web server to remotely exploit this vulnerability.

Exploitation of this vulnerability can result in two outcomes, namely denial of service or execution of arbitrary code with web server permission privileges. Each of these outcomes affects the base score differently since they incorporate different metrics into the base score equations. In computing the base score, the outcome which produces the greater base score is selected.

If the vulnerability is exploited with the intent to execute arbitrary code with the web server permission level, the *Confidentiality* and *Integrity Impact* metrics are set to *Partial*. This is due to the fact that the attacker doesn't have complete control over what is accessible. Moreover, the attacker only has the privilege level of the web server, which may not be the root or administrator privilege level, usually required for accessing sensitive system files and credential information.

If the vulnerability if exploited in order to cause a denial of service attack, the *Availability*

Impact is set to *Complete*. A denial of service is highly effective against this version of Apache Web Server, rendering it unavailable.

Below is a comparison of the results generated by the above mentioned attack vectors, illustrating how each metric is combined to produce the base score. Table 3.14 shows the results of an exploit successfully run against **CVE-2002-0392** to exploit arbitrary code at web server level.

Table 3.14 Calculation of CVE-2002-0392 severity score in an attack geared towards arbitrary code execution.

Base Metric	**Evaluation**	**Score**
Access Vector	Network	1.00
Access Complexity	Low	0.71
Authentication	None	0.704
Confidentiality Impact	Partial	0.275
Integrity Impact	Partial	0.275
Availability Impact	None	0.00

```
Impact = 10.41 * (1 - (0.725) * (0.725) * (1)) = 4.9
Exploitability = 20 * 1 * 0.71 * 0.704 = 10.0
f(Impact) = 1.176
BaseScore = ((0.6 * 4.9) + (0.4 * 10) - 1.5) * 1.176 = 6.4
```

Table 3.15 shows the results of an exploit geared towards a denial of service attack against the Apache web server.

Table 3.15 Calculation of CVE-2002-0392 severity score in a denial of service scenario.

Base Metric	**Evaluation**	**Score**
Access Vector	Network	1.00
Access Complexity	Low	0.71
Authentication	None	0.704
Confidentiality Impact	Partial	0.00
Integrity Impact	Partial	0.00
Availability Impact	None	0.66

```
Impact = 10.41 * (1 - (1) * (1) * (0.34)) = 6.9
Exploitability = 20 * 0.71 * 0.704 * 1 = 10.0
f(Impact) = 1.176
BaseScore = ((0.6 * 6.9) + (0.4 * 10.0)  1.5) * 1.176 = 7.8
```

The comparison above reveals that exploiting **CVE-2002-0392** to execute arbitrary code achieves a base score of 6.4, while exploiting it as a denial of service attack achieves 7.8. The highest score of the two is selected as the official severity rating, displayed on the vulnerability report.

3.4 Network Configuration Sources - Layer I

One of the key inputs required in the **Information Aggregation** layer is a layout of the network being evaluated. In order to map out the structure of the network being modelled, topological information is required, incorporating facts about network devices and the connectivity relations between them. Moreover, it is crucial to supply a detailed list of the kinds of products installed on each device to support the discovery of vulnerability identification. This information is required to gather a vulnerability's security metric data from a vulnerability database as well as extract details required to construct exploit actions. Currently NVA does not facilitate the transformation of network data from other sources into the NVA file format, however this is deemed as an opportunity for future work. The conversion of this information from external sources could be performed by constructing plugins for each source. This thesis assumes that modelling of networks will be done independently of NVA, using third-party tools. Listed below are some of the more popular tools used in network discovery and vulnerability identification:

1. NMAP is a security scanner used to discover devices and services on networks, while also mapping out the network structure[40]. It achieves this by transmitting specially crafted packets to target devices and analysing the responses. Some of the key features offered by NMAP include host discovery, port scanning, application version detection and operating system detection.

2. Nessus is a vulnerability scanning program whose goal is to detect vulnerabilities on tested systems[41]. Nessus scans search for vulnerabilities based on software installed on devices, misconfigurations and also default password usage. Using tools such as Nessus can help provide NVA with an enumeration of existing vulnerabilities on devices.

3.4.1 Using a SIEM to Obtain Network Information

In the Background Chapter, we discussed the concept of SIEM, which aims at providing reporting, event archiving, real-time security analysis and alert generation. These functions draw knowledge from events generated by devices in a network and the applications installed on them. Edge-side agents are installed on devices which report back to event aggregation modules, which communicates with a central management system. This architecture provides the opportunity to perform automated network discovery and aggregate device configuration information. This SIEM process is highly beneficial, since although this information could be gathered manually, it quickly becomes impractical for the framework to gather information for a large-scale network of more than 50 devices.

3.4.2 Input File Format

NVA reads network configuration files stored in an Extensible Markup Language (XML) format. XML is easy for computers to read and its tree-based structure is suitable to this scenario's structure. In Figure 3.3 the structure of data entries in a network configuration file is shown, illustrating the relationship between elements of a network and their information.

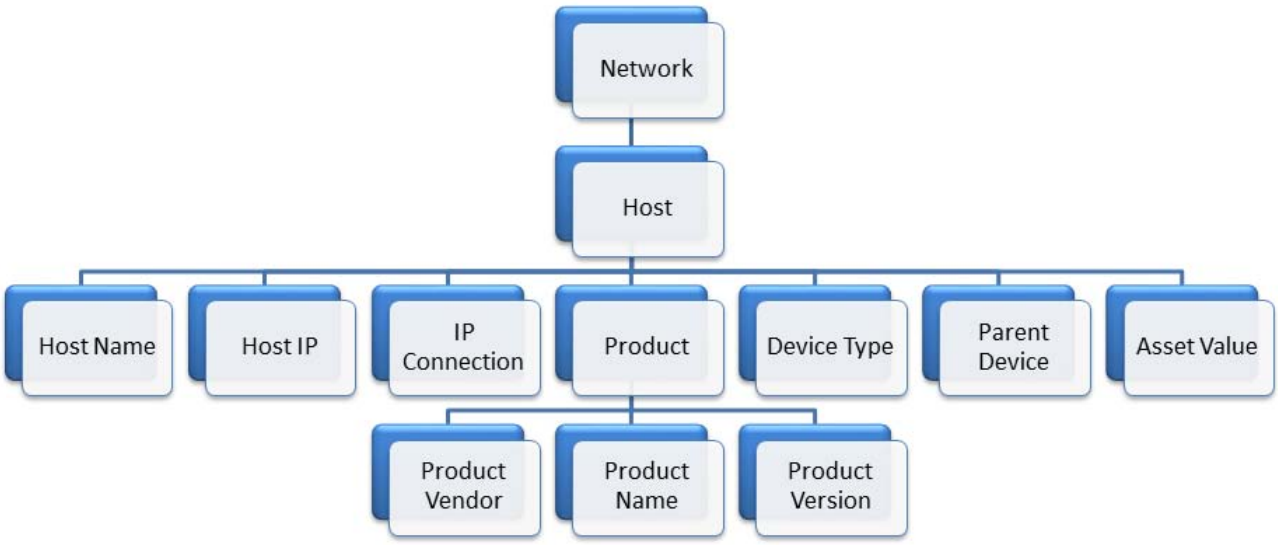


Figure 3.3: *The hierachical structure of elements in a network configuration file.*

Another benefit of the XML format is the fact that it can be used on a wide variety of platforms and can be interpreted by many tools. This translates to high interoperability

owing to the fact that information supplied from other applications can be easily translated into the required XML structure of an NVA network configuration file via a simple parser plugin.

3.5 Constructing Multi-Step Network Attacks as Plannable Problems - Layer II

3.5.1 Modelling of Devices on the Network and Topology

This section discusses what information is required to model a network and how elements relate to each other. Attack graph construction techniques are currently restricted to model attacker privilege levels on exploited hosts, since the most abundant sources of vulnerability information only offer postconditions in terms of privilege escalation levels. All entities described here are considered part of the problem or scenario description.

Network Devices

All devices modelled, including *hosts*, *routers*, *switches* and the *attacker* are incorporated into a larger *network model*. *Network models* are based on input which describes the topology of the network to be tested, provided in an XML file. The aggregation of this information is not within the scope of NVA, however it can be gathered using other third party network discovery tools, mentioned in Section 3.4. The *attacker* is modelled as a *host* which is initially compromised, and has the ability to connect to a variety of specified *hosts* in the network. Information stored within the *network* and *attack* models is later used in computation of attack graphs by transforming the model into a PDDL representation called the *problem*.

Device Connectivity Relationships

Network connections between devices in a network are described using a pair-wise relationship called an *IP connection*. Devices can only access each other if there exists an *IP connection* between them to facilitate communication. In general, all hosts within a subnetwork have connectivity between each other since communication within subnetworks are

typically unrestricted, unless explicitly restricted by a firewall. In reality, there are a lot more intricacies which govern how hosts are connected. However, it is felt that this simplification is acceptable for this research, since the focus is around attack path optimization and not information aggregation. Future work may entail information aggregation from routers and firewalls to help construct connectivity relations between hosts. *Hosts* may however be restricted to what they can access in other subnets, based on firewall constraints. It is important to note that while firewalls are not explicitly modelled, their connection rules are by stating what each device is capable of accessing. Future work may incorporate rules from actual firewalls to add restraints on device accessibility within networks.

Host Models

The core focus in computing attack graphs from the aforementioned model revolves around the vulnerability of *hosts* and their accessibility from an attacker. *Hosts* in the network offer crucial pieces of information which can be used to discover likely attack vectors. Each *host* has a unique IP address and hostname, used as identifiers for both administrators analysing the network as well as the planner, which requires that each object be uniquely identifiable. Additionally, they can be assigned asset values, between one and ten, based on the importance of their function in the network. This is used in identifying suitable hosts to patch within critical attack paths while also prioritizing the search based their asset values. For example, in enterprise network where client information is stored in a database, a high asset value would be assigned to its database server while assigning a lower asset value would be assigned to a workstation in the accounting department. This is because the database server stores important and confidential client information and requires high availability.

Product Models

Products running on hosts are enumerated within the network description and provide insight into software weaknesses. A *product* comprises of a vendor name, product name and version. This information is used to perform a vulnerability assessment on each host within the network, by comparing products with vulnerability entries in NVD.

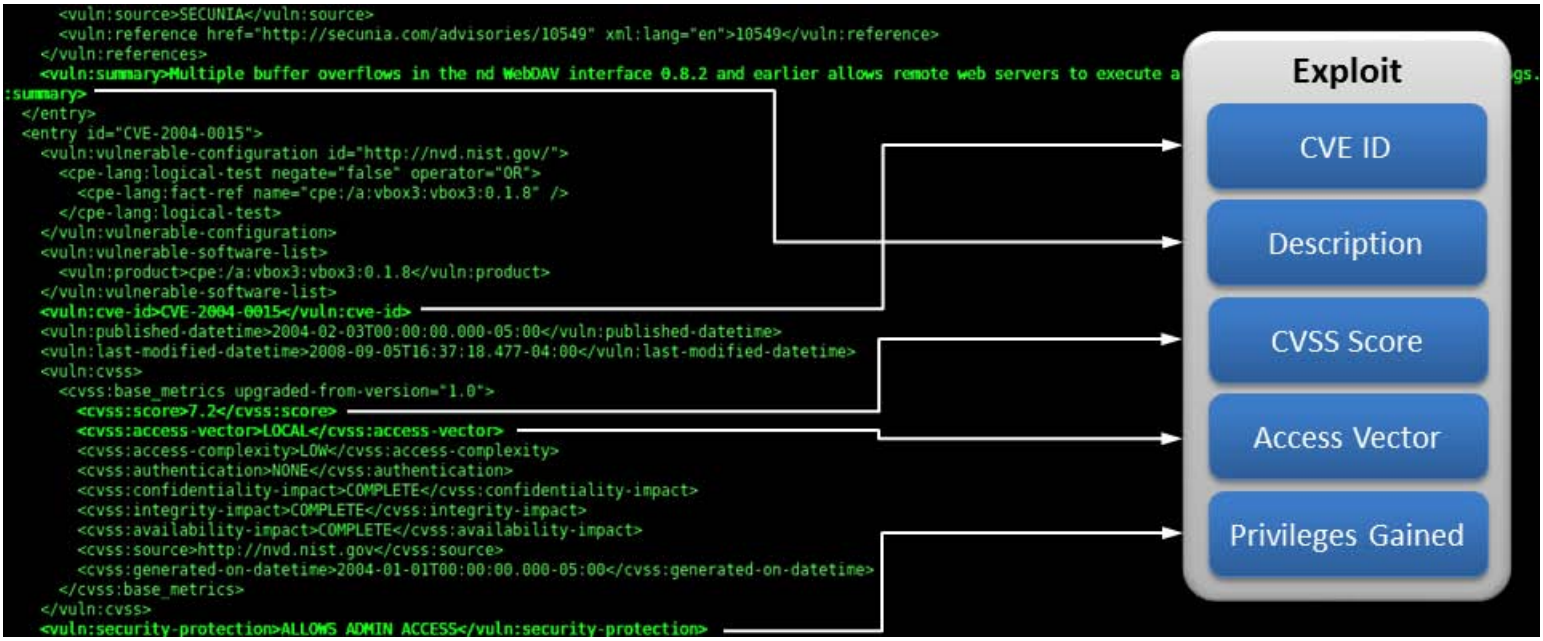


Figure 3.4: *An illustration of fields used in vulnerability entries from NVD which are extracted for usage in exploit action encoding.*

Vulnerability Models and Assessment of Vulnerable Products on Modelled Devices

Vulnerabilities discovered in performing a vulnerability assessment are stored within each host model. Prior versions of NVA provided product information to the planner, requiring it to perform cross-correlation with listed *vulnerabilities*. The issue with this technique is that it entailed explicit enumeration of every possible vulnerability in the planner's domain description. This meant that every potential vulnerability was tested for each step of an attack, resulting in the planner rapidly running out of memory and processing resources. Performing vulnerability assessments as a pre-processing task takes the load off of the planner and also means that only *vulnerabilities* related to the problem need to be encoded into *exploits* and enumerated in the domain description. In doing so, the amount of potential exploit actions the planner needs to test for each step of the attack process is drastically reduced. Discovered *vulnerabilities* are stored within host models using their CVE identifiers to reduce resource consumption. Figure 3.4 summarises the fields from vulnerability entries in NVD that are used to encode exploit actions.

Exploit Models

Exploit actions are constructed by drawing on information provided by NVD, used in defining potential actions for the planner to utilize. State-transitions are performed by the planner by selecting appropriate actions in the domain description to shift to a new state. Actions are

defined by preconditions required for execution and the resulting postconditions there of. In this case, actions are encoded exploits, based on *vulnerabilities*, indicating the requirements for an attack to be carried out and the resulting state the attack leaves the network in. Exploits are stored within the domain file using the syntax defined for actions in PDDL, illustrated in Figure 3.5.

```
(:action CVE-2004-1619
        :parameters (?attacker - host ?source - host ?target - host)
        :precondition (
                and (or (has_priv NO_PRIV ?target)(has_priv ALLOWS_USER_ACCESS ?target))
                (compromised ?source)
                (has_vuln ?target CVE-2004-1619)
                (has_ip_connectivity ?source ?target)
        )
        :effect (
                and (has_priv ALLOWS_USER_ACCESS ?target)
                (compromised ?target)
                (decrease (cvss_total) 7.5)
        )
)
```

Figure 3.5: *Exploits are structured into preconditions and postconditions, based on information provided by NVD vulnerability entries.*

An exploit action consists of a unique identifier, device parameters, preconditions that need to be satisfied to perform it and the resultant postconditions. Exploit actions use the CVE identifier as an exploit action identifier, since they are unique and machine-readable from NVD. Below the exploit action identifier is a list of parameters which are used in both the preconditions and postconditions. These parameters include the attacker's host, the source host and the target host. The structure and construction of exploit models will be further elaborated on below, in Section 3.5.2.

Attacker Models

An *attacker* is based on the same model as a standard host, differing in that products are omitted. Products can be omitted since an *attacker* is the source of an attack, targeting other hosts with software vulnerabilities. Device connectivity for an *attacker* is defined similarly to any other network device, using *IP Connections* to describe what it can connect to. Furthermore, an *attacker's* set of *IP Connections* also imply the visibility of certain network devices to him. For example, an *attacker* based in the internet may only have vision of a

NATed enterprise network's[4] web server, since it is usually the only device publicly exposed to the internet. By compromising devices in this network and pivoting through them an *attacker* can gain vision of other devices. An *attacker* can have three different levels of privileges on hosts in a network, namely *no privileges*, *user level privileges* or *administrator level privileges*. An attacker who has no privileges on a target machine is not capable of initiating an attack on another machine from the target, because he does not have sufficient access levels to the target. An attacker with *user level privileges* on a target has the most basic level of control, and can pivot through the machine to attack another target. An attack who has *administrator level privileges* to a target has the highest level of access and can initiate attacks on other targets as well as secure means of persistent access to the target, by installing a backdoor on it. By exploiting vulnerabilities in products, an *attacker* can compromise a host and escalate his privileges on it. This allows to him 'pivot' through a network, further compromising other hosts which he might not have had access to previously, until a designated goal host is compromised. In this context, *pivoting* refers to a shift in an attacker's position by relocating to a compromised machine. For the purpose of this study, a goal is strictly defined as a certain degree of privilege gained on a host compromised by an *attacker*.

3.5.2 Modelling Exploit Actions

In graph domains, operators or actions are used to perform transitions between logical states. Let us represent a deterministic transition system $\Sigma = \langle S, I, O, G \rangle$ is used to encompass all elements required for transitions between states. The transition system is deterministic since there is only one initial state and all actions are deterministic. This implies that all potential states of this networking environment are entirely predictable. The deterministic transition system Σ is defined such that:

- S is a finite set of network states
- Initial network state $I \in S$

[4]Network Address Translation (NAT), described in simplified terms, is commonly used to hide an entire subnet of private IP addresses behind a single public IP address. This is achieved by splitting a network into an internal and external network, whereby a subnet of internal IP addresses are mapped to a single externally accessible IP address. Communication is initiated from within the private network with outside parties to establish the routing table for the external party to route packets to within the NATed network. The result of this technique is that someone outside of the masqueraded network cannot access a private IP directly without the private IP initiating communication first.

- Exploit actions $a \in O$
- Goal network state set $G \subseteq S$ is finite

Provided with an initial state $s \in I$ and an exploit action $a \in S$ such that a is applicable to state s, a successor state s' can be found. The successor state is defined within set S, since the transition system is deterministic, and can be obtained using a transition function γ with $s' = \gamma(s,\ a)$. Essentially, for a network state s to transition into a new state s', an exploit action must be able to take advantage of an existing vulnerability by satisfying certain criteria. A plan P generated for a network is a critical attack path and is comprised of a sequence $\pi = a_1,..., a_n$ of exploit actions. A plan P is valid iff for π there exists a sequence of states $s_0,...,s_n$ such that $s_0 = P_{Init}$ and $s_n = P_{Goal}$. Below we define all of elements of an exploit action used in transitioning network states.

Exploit Action Syntax

In the PDDL language an action is composed of three parts; a list of parameters, a precondition and an effect. Using the Extended Backus-Naur Form (EBNF) description of PDDL 3.0[42], the syntax of an action is as follows:

```
(:action <action name>
 :parameters <typed list of parameters>
 :precondition <precondition propositions>
 :effect <list of effects>
)
```

Below we examine an example of an exploit action based on vulnerability **CVE-2009-0184**, defined for the domain of a network. Each exploit action is based on vulnerability information extracted from NVD which is encoded into the PDDL syntax.

```
1.  (:action CVE-2009-0184
2.   :parameters (?attacker - host ?source - host ?target - host)
3.   :precondition (
4.    and (or (has_priv NO_PRIV ?target)(has_priv ALLOWS_USER_ACCESS ?target))
5.    (compromised ?source)
6.    (has_vuln ?target CVE-2009-0184)
7.    (has_ip_connectivity ?source ?target)
```

```
8.   )
9.   :effect (
10.   and (has_priv ALLOWS_ADMIN_ACCESS ?target)
11.   (compromised ?target)
12.   (increase (cvss_total) 9.3)
13.   )
14. )
```

The `:parameters` list is made up of a typed list of variables on which the action operates. These variables are used in both the `:precondition` and `:effect` sections to modify the defined domain for the problem space. Exploit actions use three parameters typed as host objects; an attacker, a source device from where the attack is originating and a device being targeted by the attacker.

Exploit Preconditions

A precondition is a first order logic which must be valid in the current state for an action to be applicable to that current state[43]. Logical operators `and`, `or` and `not` can be applied to atomic propositions defined in the predicate definitions, or a group of operators and atomic propositions.

The domain defines three important predicates used in validating any exploit action; namely `(has_ip_connectivity ?source - host ?target - host)`, `(compromised ?target - host)` and `(has_priv ?level - privilege ?target - host)`. The `(has_ip_connectivity ?source - host ?target - host)` predicate is `true` if the source and target hosts are defined to have IP connectivity in the problem specification. The `(compromised ?target - host)` predicate returns `true` if the attacker has compromised the host in question. The privilege level that an attacker has gained for a host is checked using the `(has_priv ?level - privilege ?target - host)` predicate.

From action **CVE-2009-0184**, several atomic propositions are listed in `:precondition`, based on defined predicates for the domain. The `and` operator in line 4 wraps around the entire precondition and denotes that it is conjugated. In the first line the planner evaluates whether the attacker has not got any privileges or has user level privileges on

the target host. By checking this it prevents the action from being further evaluated if the attacker already has the highest privilege level on the target, reducing processing time. Line 5 determines whether the source host has already been compromised by the attacker, since to successfully compromise a target the attacker must already have pivoted his position to the source host. Next the target host is examined to see whether it is in fact vulnerable to **CVE-2009-0184**. In previous versions of NVA additional predicates were included for product vendors, names and versions and compared with vulnerability information in the domain to determine whether a host was vulnerable or not. However, it was found that this method was too computationally expensive for the planner, leading to exhaustion of memory for the simplest of networks. This lead to the refinement of techniques, performing host vulnerability assessments as pre-processing, effectively taking a large computational load off of the planner itself. Finally, line 7 checks whether there is in fact an *IP connection* between the source and target hosts, since the attack cannot occurr if there is no link between the two. If all statements listed in `:precondition` evaluate to `true` the preconditions for the actions are satisfied and the postconditions can be applied.

Exploit Postconditions

Much like the precondition, the effect also uses first order logic together with the same notation and operators. Atomic propositions can be set to `true` if they are not negated, while negated atomic propositions are set to `false` in the resulting new state. The effect can also act on numeric values which are defined as functions.

Functions act as tuples which associate values with domain objects, which can be modified in `:effects` using primitive numeric expressions. Effects can modify a function value by using direct assignment or `increase` and `decrease` operators. In the network domain, a function defined as `(cvss_total)` is used to keep track of the aggregated CVSS rating for a sequence of exploit actions. This is important since this metric is used by the planner to discover critical attack paths by identifying attack paths which yield the highest CVSS rating. CVSS severity scores were selected as a means of guiding state-space exploration since they are currently the only publicly available metric which can measure the impact of an exploited vulnerability.

In the `:effect`, the first line issues the assignment of a privilege that the attacker will have on the target host. Line 11 sets the `compromised` predicate as `true` for the target host. This predicate is used to prevent the planner's search algorithm from backtracking along previously explored paths. Lastly, the `(cvss_total)` is increased by the value of the vulnerability's CVSS score, which in this case is 9.3. The combination of these statement modifications yields a new domain state.

3.5.3 Deterministic Attack Planning

In *deterministic planning* two simple assumptions are made. Firstly, all actions are deterministic which implies that an action can result in only one successor state. What this means in terms of NVA is that the successful execution of an exploit action results in the compromise of only one host device. Secondly, there can be only one initial state meaning that the network can initially only have one state; a network of uncompromised host devices which an attacker could potentially connect to.

Motivation for Using a Planner to Guide State-Space Searching

The most basic planning algorithm generates every possible state and then performs an exhaustive search to find the shortest path from an initial state to a goal state. While this technique may be practical for smaller state-spaces, it quickly becomes impractical when the number of state variables is above 20 or 30[44]. Consider that for 20 boolean state variables the state-space would be $2^{20} = 1048576$, while 30 boolean state variables compute to $2^{30} = 1073741824$ states. Analysis on previous attack modelling approaches proved to have exponential complexity due to the aforementioned problem. In an attempt to address this issue, some approaches [45][46] used the assumption of monotonicity, whereby a compromised host cannot be uncompromised. Fundamentally, this prevents back-tracking through an attack graph during a state-space search, improving the problem from exponential complexity to polynomial one[47]. This concept was employed in combination with the use of a planner, in the design of NVA, to further reduce computation time. To avoid the state-space explosion problem planners avoid generation of the entire state-space and rather focus on only producing successor states to those under consideration. Selection of successor states can be guided by using a heuristic approach to compute optimal solutions. Furthermore, planners offer the following benefits:

- Pruning of unnecessary actions and finds the shortest path to a goal state
- Allowance for the addition of actions to a plan at any time
- Use of a rich input language allows domains to described in detail
- Does not suffer from state-space explosion

Planners manage to achieve the aforementioned points by tackling state-space searching using a more intelligent approach. They employ heuristics from artificial intelligence algorithms which help by delaying or ruling out the exploration of unpromising regions of a graph. A heuristic function $h^*(s)$ is the cost of the true lowest-cost path from state s to a goal state. Some heuristic function h is defined as *admissible* if:

$$h(s) \leq h^*(s) \qquad \forall s \in S \tag{3.1}$$

An *admissible* heuristic can be used to compute an optimal solution for a state-space. Different planners adopt different heuristic functions, affecting the amount of states generated and the speed of solution discovery, encouraging further investigation into, and comparisons of automated planners.

3.5.4 A Comparison of Automated Planners

After confirming the advantage that planners can provide in solving problems that require the discovery of shortest paths, it was important to establish a suitable planner to use for the network security domain. It is important to note that planners can be used interchangeably since all modern planners use PDDL as a standardised method for representing domains and problems.

The Fifth International Planning Competition (IPC-5)[48] acts as a benchmark for automated planners by evaluating different planning approaches. The competition is broken down into two sections, a deterministic track and a non-deterministic track. Results from the deterministic track were analysed given, the deterministic nature of this study, revealing strengths and weaknesses of several planners. Benchmark domains for the deterministic track included a number domains, some of which were inspired by real world applications,

and others by known problems in computer science. The domains or tracks studied were `storage`, `trucks`, `pathways`, `travelling purchaser problem` and `openstacks`.

Travelling Purchaser Problem Domain

The `travelling purchaser problem(TPP)` is a well researched domain which is a generalized version of the Travelling Purchaser Problem. On this track there is a set of products and a set of markets, where markets have limited product stocks at a known price. The problem entails selecting a subset of markets such that the travel time and purchase cost is minimized, while there is demand for each product. It is considered a challenging problem since it is in fact NP-hard.

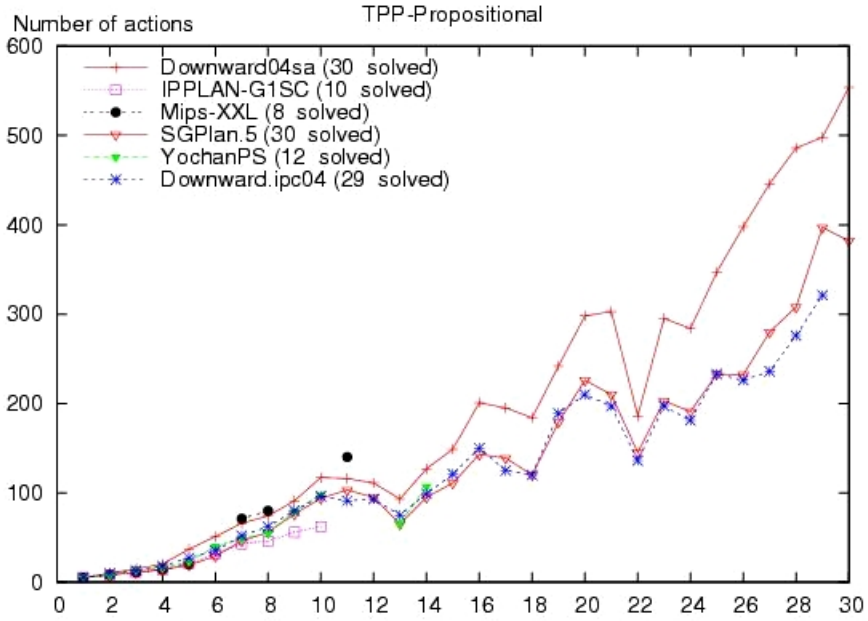


Figure 3.6: *In the TPP track there are multiple depots and trucks, goods have uniform prices and operators were added for loading and unloading goods to and from trucks[48].*

Openstacks Domain

In `openstacks`, a manufacturer has a number of product orders for a variety of products. Furthermore, the manufacturer can only make one product at a time, and the quantity of each product is initialised at the same time. An order is said to be *open* from the creation of

the first product of an order to the completion of all products in an order. While an order is *open* created products are stored in a storage space known as a *stack*. The challenge is to minimize the the maximum number of *stacks* being used simultaneously during an order of different products, or minimize the number of orders made in simultaneous production. This track simulates a realistic manufacturing problem and is also known to be NP-hard.

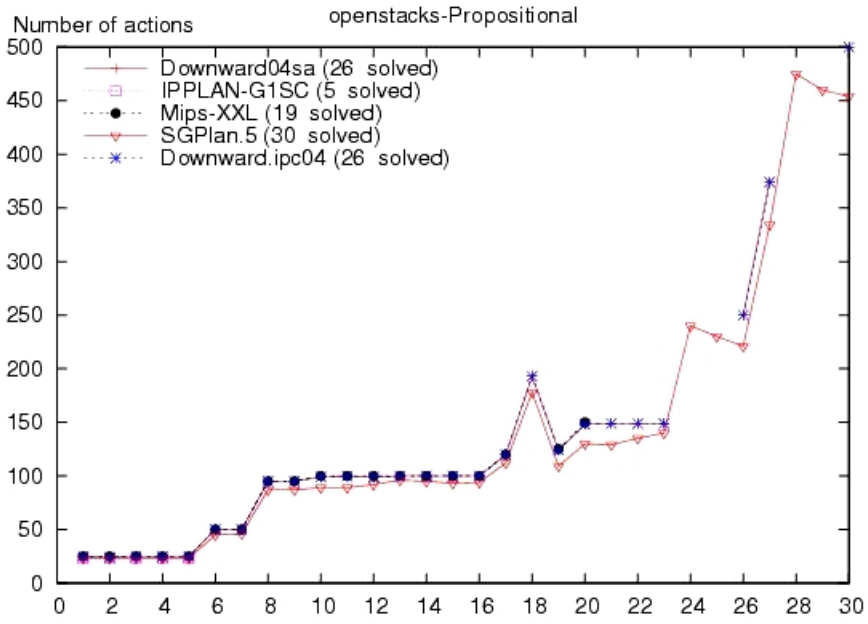


Figure 3.7: *The objective function of this track is to minimize the maximum number of simultaneously open stacks. Operators used perform simple functions including starting orders, making products, shipping completed orders and stack opening action[48].*

Storage Domain

This domain is involved in moving a certain number of crates from containers to depots, using hoists. Hoists are governed by certain spatial reasoning and can move according to certain mappings to connect areas of a depot. Problems generated for this domain aim at creating different configurations based on different numbers of depots, crates, containers and hoists.

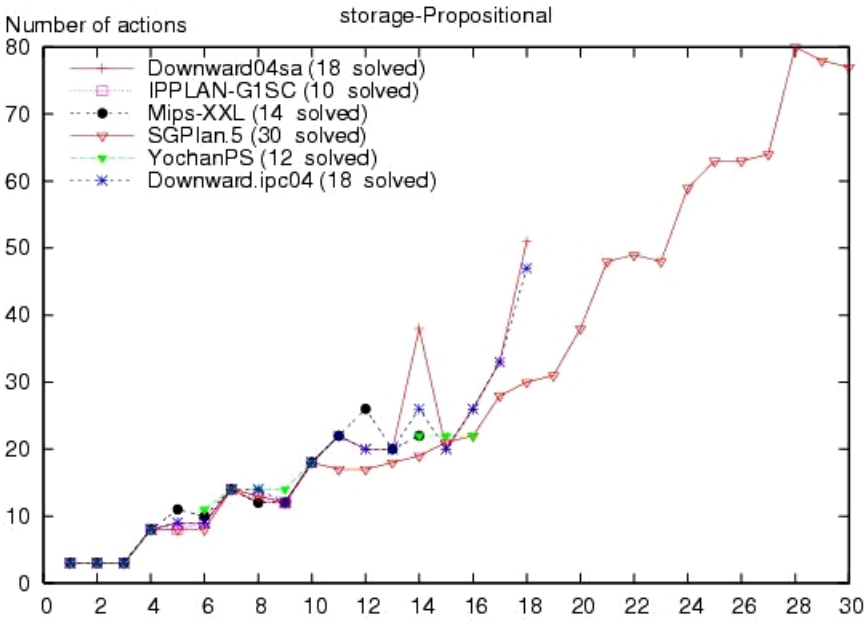


Figure 3.8: *Domain operators for this track include lifting a crate with a hoist, dropping a crate with the hoist, moving a hoist into a depot, moving a hoist from different areas within a depot and moving a hoist outside of a depot.[48].*

Trucks Domain

`Trucks` is concerned with logistical problems, whereby trucks with storage space limitations, need to transport packages between various locations under certain constraints. Packages need to be arranged in the storage area of a truck, keeping in mind that a package can only be loaded into or unloaded from an area only if there is free space between the truck door and the targeted area. A further restriction on this domain is a time constraint, used in determining an optimal solution.

Pathways Domain

Inspired by molecular biology, `pathways` looks at constructing a sequence of chemical reactions that generates one or more substances, limited by a certain number of input substances. Duration times are assigned to reactions and the aim is to produce a valid plan in the minimum amount of time.

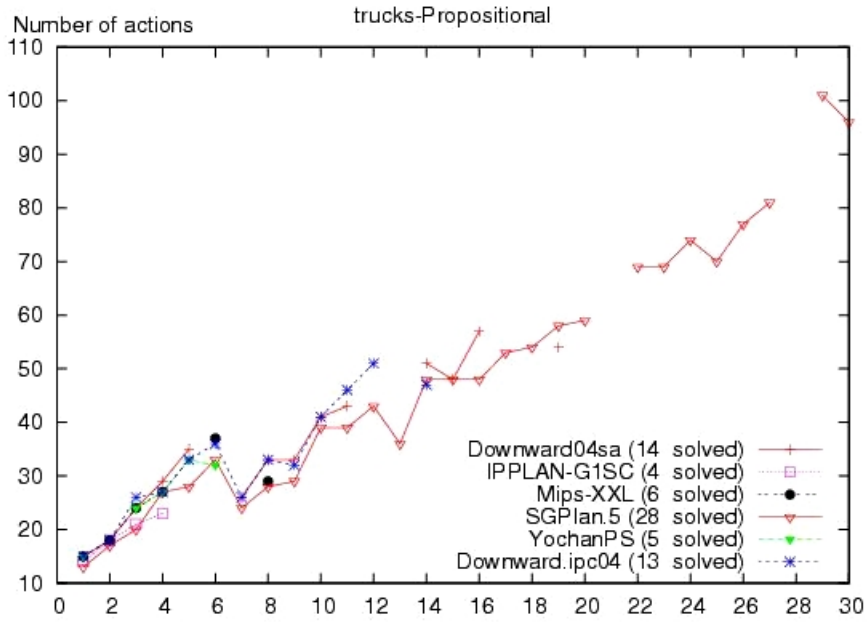


Figure 3.9: *Four different operators are encoded into this track; loading a package into a truck, unloading a package from a truck, moving a truck and delivering a package. The objective function is to minimize the time taken to complete a problem as well as the distance of the logistical problem[48].*

Optimal and Suboptimal Tracks

Evaluation criteria is narrowed down further, splitting competing planners into two categories; *optimal* and *suboptimal* planners. Evaluation of planners in the *optimal* class looks at the number of solved problems and the amount of CPU time taken. In the *suboptimal* class the number of problems solved and plan quality is considered as the top priority, while CPU time is considered as only a secondary measure. We focused on *suboptimal* planners only, because the quality of plans as well as the number of solutions is of utmost importance when determining the overall security of a modelled network. Although the computational speed advantage offered by an *optimal* planner may processing of larger network models faster, it may neglect to identify crucial aspects of information regarding attack vectors.

Summary of IPC-5 Suboptimal Track Results

In this section we evaluate the results for all planners competing in IPC-5's *suboptimal* tracks, focusing on the quality of plans and the number of valid solutions identified. Problems

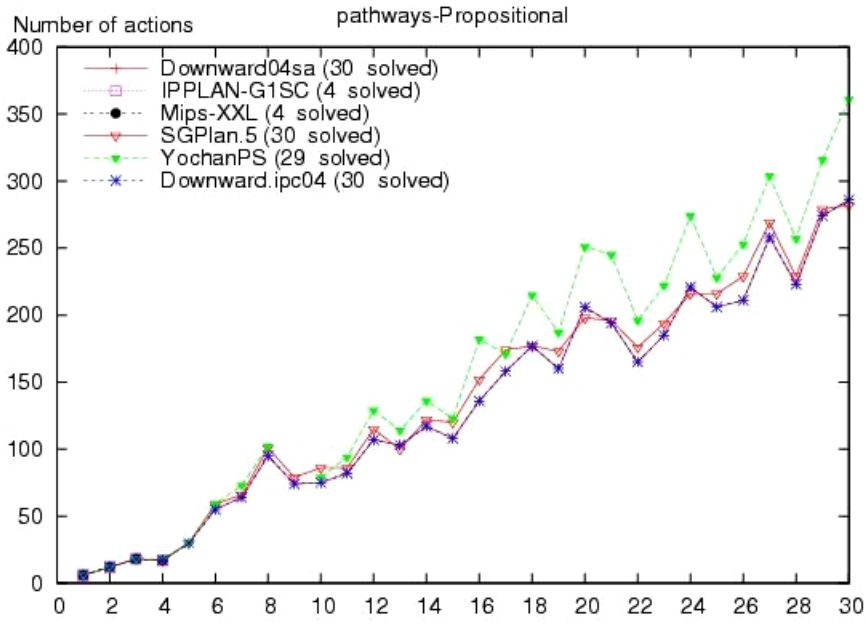


Figure 3.10: *This track's domain has five basic actions; an action to select an initial substance, increasing the quantity of a substance, biochemical reactions, biochemical reactions including catalysts and modelling biochemical synthesis reactions[48].*

for each of the tracks were generated automatically by dedicated software tools for each domain[48]. In each of the domains mentioned above, 30 different problems were tested on each planner.

SGPlan5 placed first, with the best overall performance for the *suboptimal* tracks in IPC-5[48]. The results from *suboptimal* tracks, displayed above, reflect that SGPlan5 was very consistent at tackling each domain. For each track it managed to solve all 30 problems, excepting for `trucks` where it solved 28 problems of the 30. Other distinguishing performers which deserve mention include Downward04sa and Downward.IPC4, which were also fairly consistent in completing the majority of the problems presented for *suboptimal* tracks. The overall consistency and ability to manage both complex and simple problems, while producing plans of optimal quality, made SGPlan5 the prime candidate for use in NVA.

3.5.5 Further Insight into SGPlan5 and Subgoal Partition Planning

SGPlan5's success in tackling large planning problems can be attributed to its problem solving approach. SGPlan5 is capable of solving both *temporal* and *non-temporal* problems and can understand problems written in both PDDL2 and PDDL3[49]. In *temporal* planning, actions have durations meaning that the planner needs to take into account overlapping actions as well as how far the execution of an actions has proceeded. On the other hand, *non-temporal* planning does not take into account the duration of actions an only measures the total time taken to compute a plan. Next we discuss the architecture of SGPlan5 and the key processes involved in plan computation.

SGPlan5 Overview and Architecture

In this subsection we discuss SGPlan5's architecture and describe the process used in subgoal partition planning to reach a solution. The architecture of SGPlan5 is shown in Figure 3.11, illustrating numerous steps used to reach a final plan.

Three main steps are used in solving a problem; namely **parallel decomposition**, **constraint resolution** and **subproblem solving**[50]. Below we further elaborate on these steps and provide insight into how SGPlan5 goes about plan computation.

Parallel Decomposition: This step takes the initial problem and partitions the state-space into subproblems which can be solved more easily. One of the results of this is that variables of the problem are also partitioned into subsets, leading to the potential for variable subsets of different subproblems to overlap. These overlapping variable subsets are referred to as *complicating variables*, while variables in subsets which have no overlapping issues are known as *local variables*. Two sets of attributes have been identified which lead to improved success in partitioning problems; *guidance variables* and *bottle-neck state variables*. *Guidance variables* are multi-valued state variables that are in goal-state constraints. *Bottle-neck state variables* represent variables which restrict plan parallelisation, such as object mutexes which prevent actions from accessing them simultaneously. *Parallel decomposition* is performed by partitioning the problem, such that the number of partitions is the lower bound between the number of *guidance variables* and *bottle-neck state variables*. A graph partitioning problem is

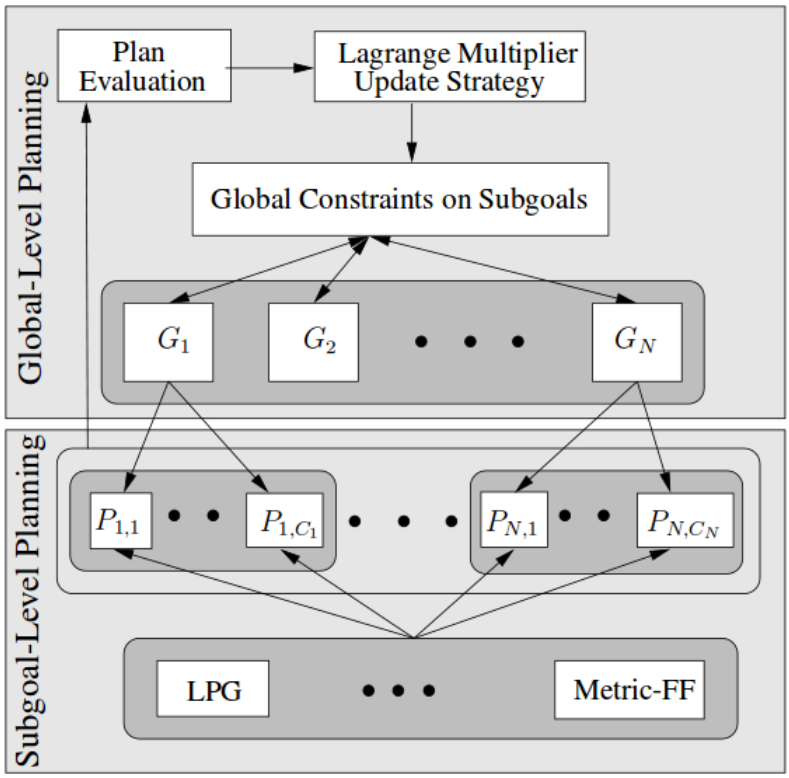


Figure 3.11: *SGPlan architecture[50].*

created where *guidance variables* are clustered together, such that nodes represent *guidance variables* and edges represent a constraint on both *guidance variables*.

Constraint Resolution: A further result of problem partitioning is the partitioning of problem constraints into subsets. Similarly to variables, constraint subsets which involve variables in one subset are called *local constraints*, while those using variables in other subsets are defined as *complicating constraints*. Another step in solving problems is *constraint resolution*, which is concerned with resolving the aforementioned *complicating constraints*. Decomposition of problems and the partitioning of constraints into subsets can reduce the complexity of problems, since the majority of planning domains have been observed to have more *local constraints* than *complicating constraints*. *Parallel decomposition* results in mutually excluded *complicating constraints* across subproblems, also known as *violating constraints*. These *violating constraints* are resolved using Extended Saddle Point Conditions, transforming them into valid *complicating constraints*.

Subproblem Solving: Once violating *complicating constraints* have been resolved, the next step is to focus on solving each subproblem. Firstly, a function derived from each subproblem's objective function and *complicating constraint* functions are weighted with penalty metrics to guide the search of each subproblem. Secondly, SGPlan5 uses a modified version of Metric-FF as a basic planner to solve these subproblems individually, performing multiple iterations. For each iteration the basic planner generates a valid solution which satisfies the *local constraints* and minimizes the weighted violations *complicating constraints*. After completing an iteration, penalty metrics are increased until all *complicating constraints* are successfully resolved. The final plan for the original problem is built from solutions computed for each of the subproblems.

3.5.6 Attack Planning Process

With and understanding of the techniques employed by SGPlan5 with subgoal partitioning, we can look at how attack plans are computed for NVA. Attack planning draws on multiple knowledge sources about a network as well as existing vulnerabilities from NVD. Before the planning process can begin, this information needs to be translated into a format that SGPlan5 can comprehend. The PDDL description language fills this role by acting as an interface between network topological information and vulnerability information supplied to NVD and the planner.

Generation of Attack Paths

We derive the planning problem $\mathcal{P} = (A, S, I, G)$ from the the definition for a Classical Planning Problem, to represent the network security problem, which aims to identify whether an attacker can reach a target device as well as the most likely approach used to do so. In planning problem $\mathcal{P}$, A is a set of exploit actions, S is a set of all state variables, I is the initial network configuration or state that defines the original values for all state variables and G the goal or target device of the attack. Since S is not explicitly defined and constructed by NVA itself, we will focus on further describing the triple (A, I, G).

The set of all exploit actions A is used in constructing the domain for the network security problem $\mathcal{P}$. As described in Section 3.5.2, exploit actions are constructed using information

based on vulnerability information stored in the NVD database.

An initial state I acts as a starting point for SGPlan5 to use in computing a plan. In essence, I describes the initial configuration of the network in a PDDL problem file, based on network topological information provided as input to NVA. To further illustrate the types of information encoded in I, an extract of a problem file is provided below:

```
1. (:objects
2.
3.  ;ID01 Host machines
4.   Attacker - host
5.   Workstation1 - host
6.   Workstation2 - host
7.   Workstation3 - host
8.   Web_Server - host
9.
10. ;ID02 Vulnerability names
11.  CVE-2009-0184 - vulnerability
12.  CVE-2008-2551 - vulnerability
13.  CVE-2008-0082 - vulnerability
14. )
15.
16. (:init [Defined as I in P=(A,S,I,G)]
17.
18. ;ID03 Define IP connectivity relations
19.  (has_ip_connectivity Attacker Web_Server)
20.  (has_ip_connectivity Workstation1 Web_Server)
21.  (has_ip_connectivity Workstation1 Workstation2)
22.  (has_ip_connectivity Workstation1 Workstation3)
23.  (has_ip_connectivity Workstation1 Web_Server)
24.
25. ;ID04 Define privilege relations
26.  (has_priv ALLOWS_ADMIN_ACCESS Attacker)
27.  (has_priv NO_PRIV Workstation1)
28.  (has_priv NO_PRIV Workstation2)
```

```
29.  (has_priv NO_PRIV Workstation3)
30.  (has_priv NO_PRIV Web_Server)
31.
32. ;ID05 Define vulnerabilities on hosts
33.  (has_vuln Workstation1 CVE-2009-0184)
34.  (has_vuln Workstation2 CVE-2008-2551)
35.  (has_vuln Web_Server CVE-2008-0082)
36. )
37.
38. ;Define metrics
39. (:metric
40.  maximize (cvss_total)
41. )
42.
43. ;ID06 Define goal [Defined as G in P=(A,S,I,G)]
44. (:goal
45.  (and(has_priv ALLOWS_ADMIN_ACCESS Workstation1)))
46. )
```

In the above extract we can see two sections containing information about the initial network configuration I; namely the `objects`, `init`, `metric` and `goal` sections. The `objects` section is used to define any object variables for the planning problem. In this case it is a list of all the hosts in the network as well as all the vulnerabilities discovered during the vulnerability assessment phase. Similarly to other object oriented languages, PDDL allows the declaration of object types. This is achieved by adding the `:typing` requirements to the requirement sections at the top of both the domain and problem files. The benefit `:typing` provides is that objects of invalid type parsed to actions are dismissed and not considered in the computation of valid solutions, eliminating irrelevant calculations. In the `init` section, predicates are initialised using information extracted from the network configuration file. Predicates that need to be defined include the IP connectivity relations between network devices, the initial type of privilege level that the attacker has to all specified devices, and finally enumerating existing vulnerabilities on network devices. In NVA the user is presented with a graphic view of the network which allows the specification of a goal G. A goal state is the compromise of a specified host, resulting in the attacker gaining a particular privilege level to that host. This is detailed between lines 43 and 45, where within the `goal` sec-

tion the `has_priv` predicate defines a goal state. In the above extract we see that the goal is in fact for the attacker to compromise Workstation1 and gain administrator privileges to it.

A plan P can be arrived at for problem $\mathcal{P}$, outlining the shortest path identified from initial state I to goal state G. Each node in a plan is comprised of a triplet (E, S, T), where E is an exploit action, S is the source host from which the attack originates for the current step and T is the target host. However, this plan may not be the best solution to the problem since other plans may exist which may be more damaging to the network's security as a whole. This prompts the introduction of a metric function to guide the search for a plan based on CVSS severity scores, consequently discovering the most dangerous attack path to reach a target. A metric is defined in the `:metric` section, where a maximization function is applied to a numerical variable called `cvss_total`. This variable is modified as the result of successful execution of an exploit action, increasing `cvss_total` by the CVSS score of the vulnerability exploited. The framework relies on SGPlan5's ability to search efficiently for an optimal attack path, which can be used for analysis.

The Evolution of States During an Attack

A state is defined as a set of atoms or facts pertaining to the configuration of a network at a specific point in time. This set of facts is an accumulation of information for all of the devices in the network, including host names, vulnerabilities on hosts, IP connectivity relationships and whether a host has been compromised, as well as the privilege level gained by the attacker. These atomic facts, which are either `true` or `false`, affect the kinds of actions that are applicable to a state. A state-transition is the evolution of a state's facts from the current configuration to a new configuration via the modification of facts. State-transitions can take place when all expressions in the preconditions for a particular exploit action are satisfied. The satisfaction of preconditions allows yields in the execution of a set of expressions in the postconditions. When an exploit action is successfully executed the target machine is marked as compromised and the attacker gains access to it with the privilege level associated with the vulnerability. This results in a modification of the network's facts which in turn changes the access that the attacker has to the network. In Figure 3.12 we can see the effects of several state-transitions of a network model.

During an attack on a network, an attacker may initially have restricted access to devices in

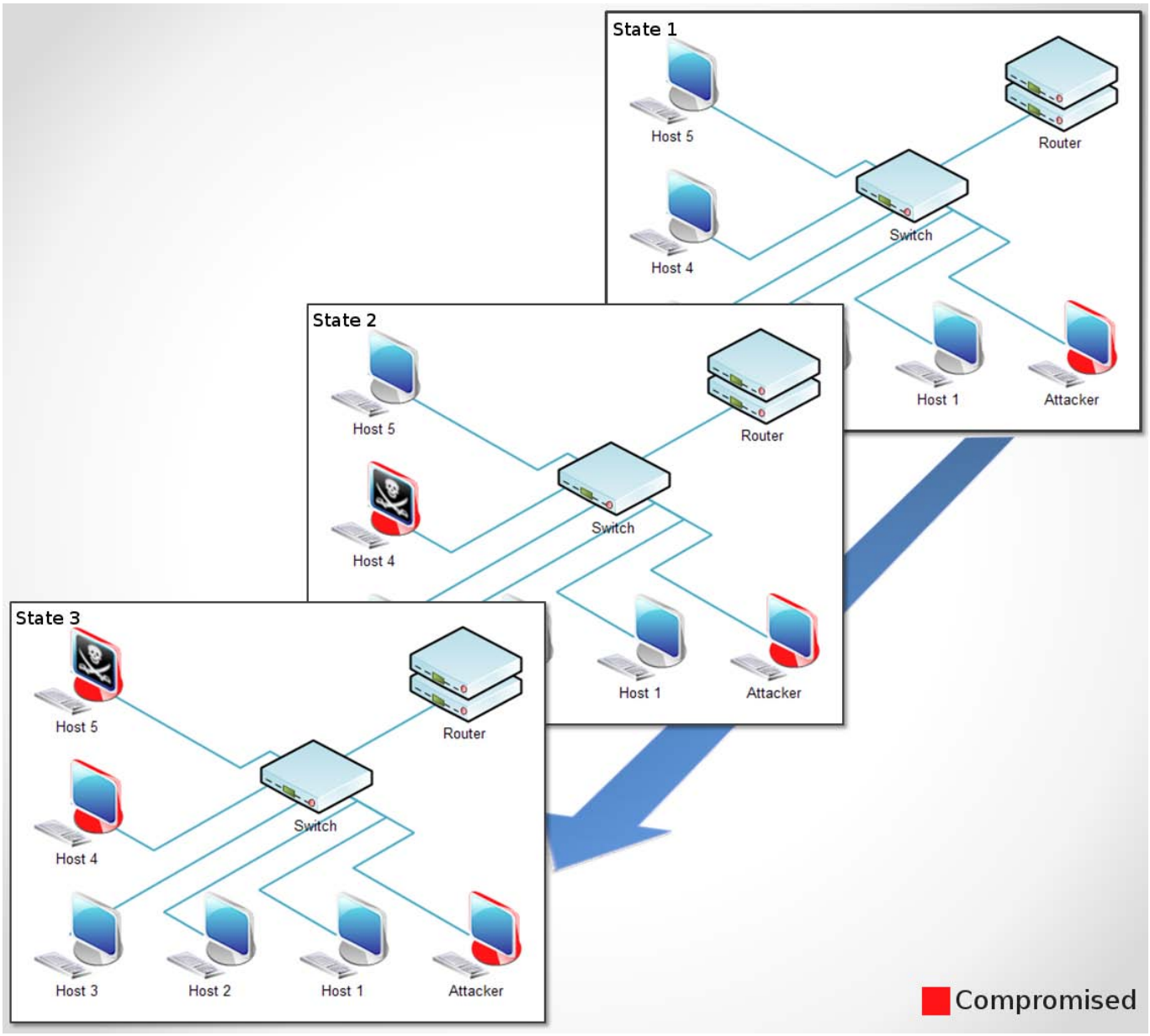


Figure 3.12: *The evolution of an attack on a network using state-transitions.*

the network, as well as a limited view of the network due to devices being concealed within various subnets by routers, firewalls or switches. However, by "pivoting" from one host to another by gaining access privileges to other hosts the attacker is able to improve his stance by not only increasing network visibility, but also expanding his potential to connect to other hosts via newly compromised hosts. Figure 3.12 illustrates the evolution of an attack with each state-transition, showing how the attacker can effectively develop his position and

perspective. Much like a game of chess, newly attained positions within a network offer the attacker an improved chance of reaching their final target.

3.6 Attack Plan Analysis Techniques - Layer II

The aforementioned approach in Section 3.5.6 has the potential to produce a plan known as a *critical attack path*, should it exist, which acts as a conduit for an attacker to reach a target. When analysing the results of an *critical attack path*, one must consider the perspective of a network administrator or security specialist who is attempting to harden the network in question. While a *critical attack path* provides insight into the most likely route of an attack, it does not answer the question, "How do I prevent this attack from happening?" The two analysis algorithms aim to address this question by presenting the user with the most prudent solution in terms of threat impact and asset damage minimization.

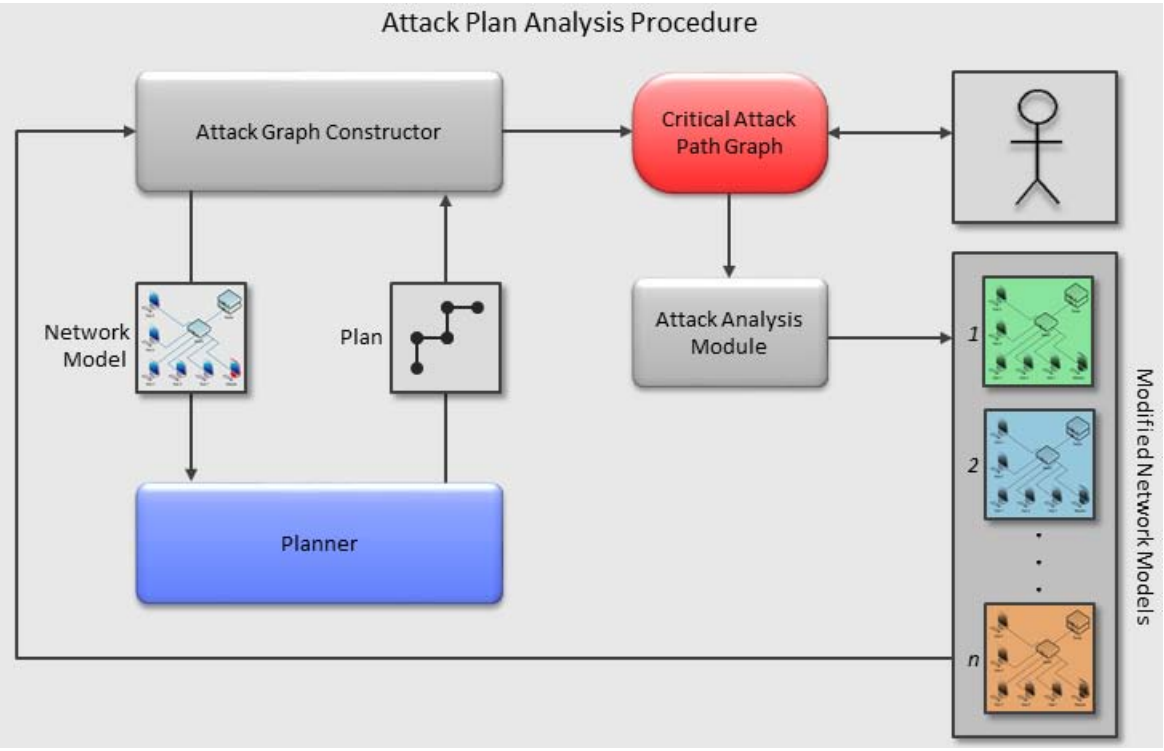


Figure 3.13: *Analysis of attack plans is performed by evaluating modifications of the initial network.*

In Figure 3.13 we illustrate the process used in performing both *Critical Action Analysis* and

Asset Action Analysis within NVA. The *Attack Analysis Module* houses all the algorithms involved in both analysing generated critical attack paths and identifying optimal solutions. Both the *Attack Graph Constructor* and *Planner* play key roles in analysis and are relied on heavily in this iterative process. These two analysis techniques aim to identify a node from a critical attack path, which when removed, completely break up the graph and prevent an attacker entirely from reaching the target. Following the aforementioned *Attack Planning Process*, a graph is constructed from the critical attack path which is presented to the user visually in the NVA GUI. We further describe the analysis process performed after a graph is generated:

1. The *Attack Analysis Module* uses this graph to create several clones of the original network model, denoted as N_i. These clones act as separate network cases associated with each node in the critical attack path.

2. For each N_i, the corresponding node is removed from the network, essentially removing the vulnerability from the exploited host.

3. Each network N_i is sent to the *Attack Graph Constructor* which attempts to generate a new critical attack path to the same target host. A candidate solution has been identified if no path can be generated, since the removal of the graph node from network N_i prevented the attack from occurring.

4. The *Attack Analysis Module* stores a list of all of the candidate solutions identified, which are then arranged based on the type of analysis technique being used.

The two analysis techniques available will be further elaborated on, comparing the types of sorting algorithms used on candidate solutions as well as highlighting the benefits of each method.

3.6.1 Critical Action Analysis

Once a list of candidate solutions have been identified, this algorithm proceeds by ranking each solution according to their position within the critical attack path. Once solution nodes have been ranked and sorted, the node with the lowest ranking (earliest in the graph) is selected as the final solution. The aim of this is to halt the attack by preventing the attack from progressing as soon as possible. Thus, it eliminates the potential for the attack to penetrate any deeper into the network.

3.6.2 Asset Action Analysis

This algorithm behaves similarly to *Critical Action Analysis*, also aiming to identify a solution which prevents an attack from occurring. However, it differs in that rather than focusing on identifying a solution node which causes the largest break in the graph, it takes into account the asset values of hosts. Subsequently, solution nodes are ranked according to the value of the asset affected by an exploit action. The reason for considering this analysis technique is when one wants to, at all costs, secure the most valuable devices whose downtime would greatly affect the productivity of an organization. It should be noted that in both *Critical Action Analysis* and *Asset Action Analysis*, the first node is ignored. The first node represents the point of entry and device initially compromised.

While it is typically most prudent to eliminate the point of origin of the attack to prevent it from progressing, this would make it impossible to generate attack paths. As such, the first node in an attack path is removed from the list of analysed nodes to aid in graph exploration and discovery of an optimal solution. In a scenario whereby the attacker is based externally and is required to breach perimeter security devices in order to gain access, it would best practice to secure these devices. However, from an internal perspective it may not be feasible to patch every workstation from which an attack may originate. Thus, it is important to take into account the resolution of other vulnerabilities which prevent the compromise of mission critical devices.

3.6.3 Criteria for Success

The criteria for success is governed by the ability for NVA to accurately discover attack paths which yield the highest severity scores. In the context of NVA, these paths are known as critical attack paths and denote the most likely attack vector, used to compromise a targeted device in the network being assessed. Moreover, the criteria for success is also dictated by the solution's ability to correctly select nodes from critical attack paths, which when patched, eliminate the potential for compromise on the targeted device.

3.7 Visualization Design - Layer III

The visualization aspect of NVA was designed to be simple yet robust, providing the user with several interactive graph views. The **Visualization** layer was decoupled from the rest of the framework's core, ensuring that any GUI related processing does not interfere with any Layer II activities. NVA performs some memory and process intensive computations while simultaneously providing feedback in the form of visual graphs. Extensive usage of *worker threads* were made in order to prevent the GUI from becoming unresponsive, during computation intensive periods. This approach assigns all **Attack Graph Computation** layer activities to a *background worker thread*, while assigning all **Visualization** layer activities to a *foreground thread*.

Synchronous communication between layers was chosen to better support the potentially high system overhead. Furthermore, the order in which certain method calls are performed needs to be guaranteed, since other methods may be heavily reliant on generated artefacts. Communication between Layer II and Layer III was designed to be achieved using direct method calls which exchange data objects.

3.8 Chapter Summary

This chapter highlighted elements of NVA's design, breaking down the computation of critical attack paths into three well-defined layers. In the first layer, we described two information sources which are fundamental in constructing network models; namely network topology and configuration input, and a vulnerability database. The second layer covered the core aspects of NVA, elaborating on the transformation of information from the first layer into plannable models. Attack planning was described, highlighting the role of a planner in computing critical attack paths, based on predefined models. It was also shown how these critical attack paths could be analysed to produce meaningful solutions for those conducting a security evaluation of a network. Finally, in the third layer the approach used for visualization was described, elaborating on the decouplement of the visualization from logic. In this next chapter we will outline the implementation of NVA and showcase algorithms used in the attack path computation and solution identification.

Chapter 4

Implementation

This chapter describes the implementation of NVA, based on the design proposed in the previous chapter. We outline the algorithms used for network generation, network information and vulnerability extraction, and attack plan analysis. Moreover, the implementation of NVA's graphical user interface is elaborated on, illustrating the core functionality offered to users.

4.1 Overview

In this chapter we present the implementation of NVA, whose purpose is to demonstrate the validity and usefulness to threat modelling. NVA is a proof-of-concept (PoC) framework which utilizes an existing vulnerability database to model threats against hosts in a network. The PoC extracts vulnerability information from the vulnerability database, transforming them into atomic exploit actions which a planner can use to test against a network model. Two analysis techniques have been implemented to further support the user in making an informed decision as to which hosts need to be patched. A network generation procedure was also implemented to assist in evaluation, during NVA's design process.

The implementation of the proposed design was done using a combination of Planning Domain Definition Language (PDDL)[19] and Java.[1] PDDL is a language is a common formal-

[1] http://www.java.com

ism used for describing planning domains, and is the most widely used language for planners. On the other hand, Java is an object-oriented language which can operate across different platforms. This is highly beneficial since writing an application in Java isolates it from platform dependent aspects of the system on which it is deployed. Other benefits include the ability to represent model elements as objects, multi-processing using threads, automatic garbage collection and the abundance of available libraries for building user interfaces and XML document parsing.

4.2 Generation of Networks for Simulation

During earlier stages of development, manual construction of small sample networks could be done within an acceptable amount of time. However, generation of larger networks proved challenging to perform manually, due to the interconnected nature of network models. One of the objectives of NVA is to prove that analysis can be performed on large networks, prompting investigation into network generation techniques. Simplistic network configuration requirements lead to creation of a custom network generation engine rather than relying on more cumbersome network simulators, such as NS-3[2] and OMNeT++[3].

The network generator constructs networks which adhere to the campus network architecture. That is, a collection of subnetworks connected by switches and routers to form a larger network. Large organizations make use of campus network architecture because its modular design makes networks highly scalable, flexible and administrable. Networks are generated by creating clusters of workstations and servers within subnets, connected with the rest of the network using switches and routers.

4.2.1 Network Generation Procedure

In Algorithm 1, the generation procedure has been annotated, revealing several distinct routines. Its main routine is concerned with generating a number of switches, specified using

[2]NS-3 is a tool used for network simulation, primarily geared towards research and educational use. Its infrastructure allows the simulation of realistic network models which can operate at real-time and interact with real-world protocols[51].

[3]OMNeT++ is designed to simulate communication networks, but can also be used to simulate IT systems or hardware architecture[52].

the parameter s. Every network is required to have a router, allowing communication with other networks or the internet. Switch generation is further subdivided into segments which govern the generation of workstations and servers, ensuring that each switch acts as a parent node to a number of hosts in a subnet. A random integer n, between one and five, is used as an upper bound for the amount of servers or workstations to create and add to a subnet. Not shown in Algorithm 1, is the generation of individual hosts, which entails the possible random selection of products from NVD. In doing this, it ensures that some hosts have vulnerable products installed which could potentially be exploited by the attacker. Next, the newly populated network is parsed to a function, shown in line 32, which generates IP connectivity relations between randomly selected hosts. The connection generation function ensures that all hosts within a subnet have connectivity with each other, since it is uncommon for a switch to be configured to restrict connectivity between hosts in its subnet. Next, the attacker's host is added to the network and is connected to a random number of hosts, based on the network size. Finally, the fully populated network is returned and is ready to be written to a file.

This feature has been integrated into NVA to aide in research and testing. Shown in Figure 4.1, is the wizard presented to the user to help construct a new network.

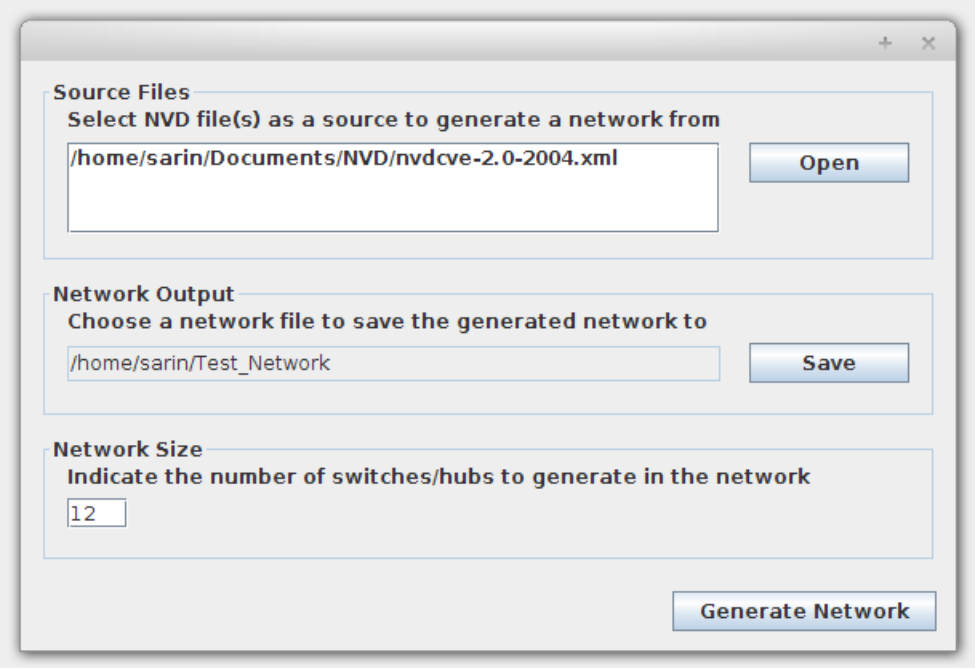


Figure 4.1: *The network generation wizard simplifies the construction of networks by requesting only essential information required.*

Algorithm 1 Generation Procedure for Sample Networks

```
1: procedure NETWORK_GENERATOR(s)
2:     switchCount ← s
3:     function GENERATE_HOSTS(switchCount)
4:         Add router to network
5:         if switchCount > 2 then
6:             Add centralSwitch to network
7:         end if
8:         for i ← 0, switchCount do                    ▷ Add switches to network
9:             Create switch
10:            if switchCount ≤ 2 then
11:                Set router as parent
12:            else
13:                Set centralSwitch as parent
14:            end if
15:            Add switch to network
16:            n ← RANDOM_NUMBER_GENERATOR(1, 5)
17:            for j ← 0, n do                          ▷ Add server host to subnet
18:                Create server
19:                ADD_PRODUCTS(server)
20:                Add server to network
21:                Set switch as parent
22:            end for
23:            n ← RANDOM_NUMBER_GENERATOR(1, 5)
24:            for j ← 0, n do                          ▷ Add workstation host to subnet
25:                Create workstation
26:                ADD_PRODUCTS(workstation)
27:                Add workstation to network
28:                Set switch as parent
29:            end for
30:        end for
31:    end function
32:    GENERATE_HOST_CONNECTIONS(network)
33:    GENERATE_ATTACKER(network)
34:    return network
35: end procedure
```

The wizard follows a simple 3-step process which allows the user to select NVD files with which the generator selects products, a file to save the network as and the desired amount of switches in the network. This NVA feature was used throughout the evaluation phase and drastically helped the speed at which networks of varied size could be built for testing. It has the capability of generating small networks with just one switch or extensive networks with tens or hundreds of switches.

4.3 Vulnerability Extraction from NVD

Upon first usage, the user is presented with the option to select a directory containing NVD vulnerability files. This location is written to NVA's configuration file and can be altered by the user at any time. In the Design Chapter we detailed the structure of entries in NVD and also outlined which were necessary to build exploit actions. Further investigation was conducted to identify an optimal technique for parsing large XML files within a reasonable time frame.

4.3.1 A Comparison of XML Parsers

Considering that NVA was written in Java, the two most popular Java XML parsers were compared; SAX (Simple API for XML Parsing) and DOM (Document Object Model) Parser. Both parsers have there own set of advantage and disadvantages which will be further elaborated on below.

- **DOM Parser** converts XML documents into tree models, having elements represent branches of the tree. The entire document is converted into a tree which is stored in memory.
- **SAX Parser** works by using events triggered by elements in XML documents. When an opening element of a tag is encountered, an event fires to begin the parsing process of the contained data. SAX Parser parses XML documents in sections and does not load the entire document into memory.

There are several differences in the way each parser functions, which affect how suitable they are at parsing NVD files. In Table 4.1, we provide a comparison, highlighting some of the pros and cons of each parser.

Table 4.1 A high-level comparison between SAX and DOM Parser[53].

	SAX	**DOM**
Memory Consumption	Low - Only small parts of a document is loaded into memory.	High - The entire document is loaded into memory.
Access Speed	Slow - Access speed is limited by the time taken to read the document in.	Fast - DOM can access a file quickly in memory.
Optimal File Size	Large - SAX does not require much memory so it can read large documents easily.	Smaller - Small files can be parsed and accessed very quickly.

When comparing the differences between these two parsers it was important to consider the large size of NVD files. SAX Parser was selected for this reason and was implemented in the process responsible domain transformation. This information is later used in the construction of domain files, tailored to each specific network.

4.4 Network Configuration Information Extraction

Network configuration information is stored in an XML file, in a tree-like structure which links hosts to a network. Information extracted from this file is structured into a network object, containing each of the hosts. A network object is later used in the construction of a problem description file for the planner. SAX Parser was implemented for the network configuration extraction process and functions similarly to the algorithm used to extract vulnerabilities from NVD.

4.4.1 Network Configuration Extraction Procedure

The extraction procedure reads the XML file in steps, using the SAX Parser. Algorithm 2 breaks the procedure down into two key functions which also highlight SAX Parser's event-triggered approach. The first function, named `Start_Element`, is triggered whenever a start element is read from a line by SAX Parser. It then proceeds to identify and match the element and contained data with the appropriate *host* variable. It is important to note that *HOSTNAME* or *PRODUCT* elements do not contain single data entries, but rather a series of subelements. When one of these elements are triggered in `Start_Element`, a new *host* or

product object is created, to which extracted data can be assigned. The second function, named `End_Element`, finalizes the parsing process for a set of elements and is responsible for linking *product* objects to *host* objects, and *host* objects to a *network*.

4.5 Transformation of Models into PDDL

In order to compute critical attack paths, the planner must be provided with information about the scenario and its environment. The scenario is described in the problem PDDL, while the environment is described in the domain PDDL. The planner relies on both of these files, which need to be constructed based on existing models created by NVA.

4.5.1 Constructing the Problem Description

The problem is derived from the network model, which was established from the network configuration file provided as input to NVA. Moreover, the goal is specified as the target of attack, selected by the user via NVA's network graph GUI. A problem template was created, which includes the necessary constructs of every problem file, as well as markers which indicate code-injection points for NVA to write model information to. These markers guide NVA to write the following model information:

- Hosts
- Vulnerability names
- IP connectivity relations
- Attacker privileges on hosts
- Vulnerabilities associated with specific hosts
- Attack goal

Essentially, the problem file is a way of specifying the initial state I and goal state G for the planning problem $\mathcal{P} = (A, I, G)$.

Algorithm 2 Network Configuration Extraction Procedure

```
1: procedure PROBLEM_PARSER(networkFile, network)
2:     SAXPARSER(networkFile)
3:     function START_ELEMENT(SAXParser)
4:         startElement ← SAXParser.start
5:         lineData ← SAXParser.data
6:         if startElement = HOST then                    ▷ Instantiate a new host
7:             Create host
8:         else if startElement = HOSTNAME then
9:             host.name ← lineData
10:        else if startElement = HOSTIP then
11:            host.ip ← lineData
12:        else if startElement = IPCONNECTIVITY then
13:            host.connectivityList ← lineData
14:        else if startElement = DEVICETYPE then
15:            host.type ← lineData
16:        else if startElement = ASSETVALUE then
17:            host.value ← lineData
18:        else if startElement = PRODUCT then            ▷ Instantiate a new product
19:            Create product
20:        else if startElement = PRODUCTVENDOR then
21:            product.vendor ← lineData
22:        else if startElement = PRODUCTNAME then
23:            product.name ← lineData
24:        else if startElement = PRODUCTVERSION then
25:            product.version ← lineData
26:        end if
27:    end function
28:    function END_ELEMENT(SAXParser)
29:        endElement ← SAXParser.end
30:        if endElement = HOST then
31:            Add host to network
32:        else if endElement = PRODUCT then
33:            Add product to host
34:        end if
35:    end function
36:    return network
37: end procedure
```

4.5.2 Constructing the Domain Description

The domain is the result of transforming vulnerabilities into exploit models, pertaining to a particular network problem. As mentioned previously, it is not practical to model every vulnerability into exploits because it slows down plan computation. A palette of vulnerabilities are modelled into exploits, based on a vulnerability scan of host products. The following information is converted from exploit models in NVA to PDDL actions:

- Initial attacker privilege level on the target
- Whether the source host has been compromised
- Whether the target is affected by the vulnerability
- Whether there is an IP connectivity relation between the source and target
- The resulting attacker privilege level on the target
- Compromise of the target
- Modifying the aggregated CVSS severity of the attack path

The aforementioned items are encoded into PDDL predicates which can be processed by SGPlan5. Similarly to writing the problem file, the domain file template is marked with a code-injection point which directs NVA to the correction position to write actions to. The combination of the problem and domain files are parsed to SGPlan5 as input parameters. NVA executes SGPlan5 in a background thread and waits for the results to be generated in a plan output file. The content of this file is 'scraped out' by NVA to see whether a valid plan was identified, parsing the result into an attack plan object, should it exist.

4.6 Attack Plan Analysis

Both attack plan analysis techniques offered in NVA aim to identify a host-vulnerability combination, which when patched, completely prevents an attacker from being able to compromise a target. Moreover, both techniques utilize the same base procedure, shown in Algorithm 3, to arrive at a list of candidate solutions.

In Algorithm 3 we see how a clone network is made for each critical attack path node. A node is symbolic of a host-vulnerability combination, identifying an exploited vulnerability

Algorithm 3 Attack Analysis Procedure

```
1: procedure COMPUTE_SOLUTION(network, attackPath, switch)
2:     pathSize ← attackPath.size
3:     for i ← 1, pathSize do
4:         node ← attackPath[i]
5:         cloneNetwork ← NETWORK_CLONER(network, node)
6:         cloneAttackPath ← ATTACK_GRAPH_CONSTRUCTOR(cloneNetwork)
7:         Add node to vulnerabilityList
8:         Add cloneNetwork to networkList
9:     end for
10:    solutionList ← FIND_SOLUTION_ACTION(vulnerabilityList, networkList)
11:    SORT_SOLUTIONS(solutionList, attackPath)
12:    if switch = 0 then
13:        FILTER_SOLUTIONS(solutionList, attackPath)
14:    else
15:        FILTER_SOLUTIONS_BY_ASSET(solutionList, attackPath)
16:    end if
17:    return solutionList
18: end procedure
```

on a particular host, from the critical attack path. This node is removed from the cloned network, essentially patching the vulnerable device. By treating each cloned network as a new test case we can determine whether the modification in fact prevents the attack from being successfully carried out. The algorithm branches into either *Critical Action Analysis* or *Asset Value Analysis*, based on a switch parameter. These two algorithms use contrasting methods for filtering candidate solutions, based on differing priorities.

4.6.1 Critical Action Analysis

Although each candidate solution discovered in Algorithm 3 would prevent the attacker from reaching their target, they have different implications on the amount of damage an attacker could still cause. An attacker which can penetrate deeper into a network can potentially inflict more damage on a company. Each host in an attack path could have sensitive data compromised or corrupted, or even be rendered unavailable. By using *Critical Action Analysis* the attack is thwarted as early as possible to minimize damage impacted on the network.

Algorithm 4 identifies an optimal solution by taking the list of candidate solutions and

ranking each of them based on their position in the originally identified critical attack path. This means that a candidate solution which is earlier in the critical attack path will have a lower rank than one which is later. The solution list is pruned by eliminating all other solutions which have a higher rank than the optimal solution.

Algorithm 4 Action Analysis Procedure

```
1: procedure FILTER_SOLUTIONS(solutionList, attackPath)
2:     if solutionList is Empty then
3:         return
4:     else
5:         solutionsSize ← solutionList.size
6:         solution ← solutionList[0]
7:         bestRank ← index of solution in attackPath
8:         for i ← 1, solutionsSize do
9:             solution ← solutionList[i]
10:            rank ← index of solution in attackPath
11:            if rank > bestRank then
12:                Remove solution from solutionList
13:            end if
14:        end for
15:    end if
16: end procedure
```

4.6.2 Asset Value Analysis

Asset Value Analysis considers the asset values of hosts the top priority when identifying an optimal solution. This technique is aimed at securing high priority hosts, whose downtime would negatively affect an organization's ability to function. The algorithm addresses this issue by sorting the list of candidate solutions by asset value. A solution whose host has a lower asset value than the optimal solution is removed from the solution list, resulting in top priority solution candidates.

4.7 Graph Visualization and User Interface

NVA's interface is divided up into a number of sections, with the main focal point being the network graph itself. Appendix B shows a screenshot taken from NVA of a simple network,

illustrating the various sections presented to the user. Along the main menu, we provide a number of functions, listed and elaborated on further below:

1. **Open** a network topology file. NVA can only read XML network configuration files which adhere to NVA's file format.
2. **Generate Attack Path** transforms the network model into a PDDL model, using a user-specified target to compute a critical attack path.
3. **Network Scan** uses the same process as in the generation of a single critical attack path, however considers every host in the network as an attacker's target. The outcome of this highlights each host's security level from the attacker's perspective.
4. **Critical Action Analysis** becomes available only after a critical attack path has been discovered and aims to provider the user with a solution to the attack by suggesting the most optimal software to patch on a host.
5. **Asset Action Analysis** Functions much like **Critical Action Analysis** excepting that it prioritizes hosts with higher asset values.
6. **Zoom In** increases the size of the graphic displayed.
7. **Zoom Out** decreases the size of the graphic displayed.
8. **Snapshot** allows the user to take a screenshot of the graphic being displayed so that it can be saved.

The left-most panel houses the *Properties* and *Device Connection List* windows. When a user selects a device from the graph it displays all of the information pertaining to that device in the *Properties* window, including its host name, IP address, device type, asset value and the products installed on it. Moreover, the selection of a product, should it have an existing vulnerability, displays its CVSS information.

The network graphic illustrates the hierarchical structure of the network but neglects to reveal the connectivity relations between devices. The reason for omitting this information is mainly because it tends to clutter up the network graphic, making it difficult to interact with it, especially in larger networks. To address this issue, the *Device Connection List* window was introduced to the interface, containing a tree structure indicating a network's underlying connectivity relations.

The center-most pane houses the network graph and all generated critical attack paths. Earlier version of NVA used custom-created canvases to paint graphs onto, however a more extensive graphing framework was discovered. Java Universal Network Graph(JUNG)[54] is a free open-source software library which allows the visualization and modification of graphs in Java. The ease of use and multitude of functionality offered by this library motivated its selection, one of which being the ability to cluster several nodes around a central node in a wagon wheel configuration. This configuration suitably represents the clustering of several hosts around devices such as switches or routers, commonly used in computer network diagrams.

Below the graphing area a contextual window offers details to the user about the current graphic pane being viewed. In the network view it enumerates information about the total number of devices in the network, a break down of these devices into their device types and also the number of vulnerabilities discovered in the entire network, as a result of the automated vulnerability scan. In the attack view it lists information about the critical attack path discovered as well as its total severity score.

4.8 Chapter Summary

The implementation of the proof of concept framework, presented in this chapter, was divided into six sections. These sections highlight fundamental procedures required to model threats and identify solutions to these threats. Java was used to implement the framework because it is object-oriented, manages garbage collection, is platform independent and has an abundance of libraries both required for XML document parsing and graphing on the GUI. PDDL was used to model network domain information since it is the most modern and standardised language used by automated planners. The following chapter will discuss evaluation methods and demonstrate the usage of NVA in two realistic scenarios.

Chapter 5

Evaluation

In this chapter we will evaluate the effectiveness of NVA by subjecting it to two different kinds of evaluation methods. Firstly, the basic functionality and versatility will be demonstrated with in-depth walkthroughs of two different case studies. The aims of these two case studies are to reflect the application of NVA in a variety of scenarios featuring different types of networks. Moreover, NVA will use identified threats to try and identify the most beneficial solution for each type of scenario. Secondly, the performance and scalability of NVA will be analysed to determine whether the framework scales well for large networks.

5.1 Case Study Demonstration

5.1.1 Case Study 1

This case focuses on a small scale network and acts as a simple proof of concept demonstration for NVA. The simplicity of this case aims to demonstrate how a sequence of actions can be identified which could lead to the compromise of critical network resources.

Network Topology

The example network illustrated in Figure 5.1 shows a network likely to be installed in a small organization. This case study models a small school network and shows how an attacker can target internal network resources from the internet. The devices this small

network contains include workstations, servers, switches, routers and firewalls. The network infrastructure is broken up into three different subnets, two of which contains workstations and the other a cluster of servers. Subnet 1 is situated inside a computer lab and is used by school students only. The student lab contains basic workstations which offer students access to basic document editing software and internet, accessed via the web server, to help them in doing research for assignments. Student workstations are not permitted to access the file server nor database server, since the information they contain is sensitive. Subnet 2 is situated inside a staff computer lab and is restricted to use by school teachers. Workstations in Subnet 2 are used by teachers to perform various administrative functions, including accessing and editing student information in the database server, as well as storing test and exam papers on the file server. Subnet 3 is comprised of a small cluster of servers, including a File Server, Web Server and Database Server. The Web Server is responsible for hosting the school website and acting as a web proxy for all workstations in the internal network to access the internet. The File Server is used by staff members for storing test and exam papers and preliminary student mark information before being finalised and committed to the database. Lastly, the Database Server contains detailed student records with information such as their finalised quarterly marks, financial records and workstation usernames and passwords for both students and staff members. Firewalls are used to filter traffic between subnets within the network and also restrict access to the internal network from public networks such as the internet. Note that for these firewall configurations are a simplification of the ingress and egress filtering performed on firewalls in reality, and as such the actual port filtering rules are not provided. This simplification is deemed as acceptable considering that the overall effect, in terms of connectivity relations, is what is being used to perform attack path generation. Firewalls FW1 and FW2 restrict network traffic as follows:

- FW1: Allows external access from the internet to the Web Server
- FW2: Allows access to the Web Server from both Subnet 1 and Subnet 2 and only allows access to the File Server and Database Server from Subnet 2.

Network Device Configuration Information

Devices in the network may contain a number of vulnerable products installed on them, each of them offering different kinds of benefits to the intruder. A detailed outline of the products installed on several devices in the network is provided in Appendix A, including

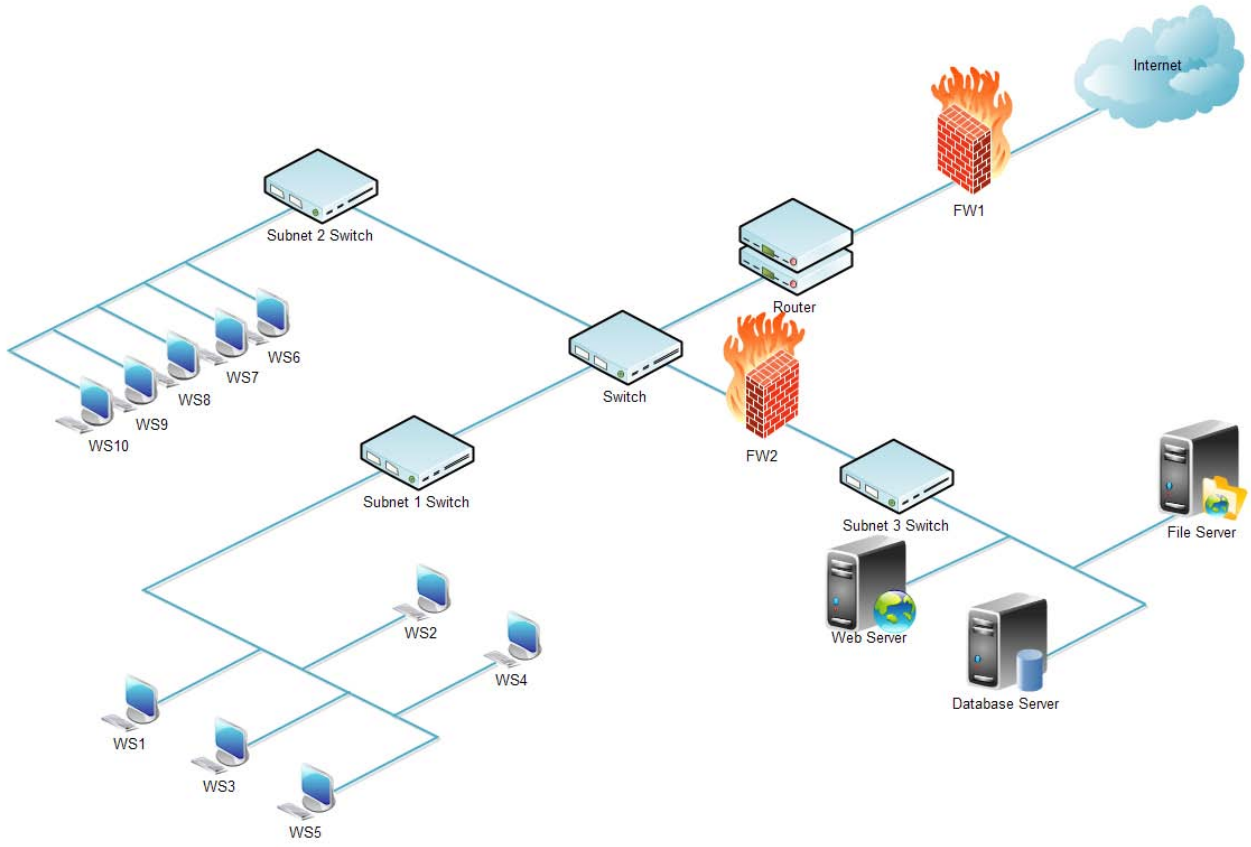


Figure 5.1: *A network structure inspired by a small school network.*

a vulnerability identifier, description, access vector required for exploitation, CVSS severity rating and the privilege level gained from successful exploitation.

Scenario Description

In this scenario, the attacker is a school student named Bob seeking to get hold of exam papers in order to sell them to other students. Bob is based outside of the school network and only has access to the Web Server. From this perspective Bob would use a network scanner, such as NMAP[40] to discover the version of the web service running on the Web Server, in order to select an applicable exploit to increase is privilege level. A common tool in the attacker's arsenal, NMAP performs TCP and UDP scans to discover services running on a target host. Moreover, upon completion it can probe discovered ports to determine what

exactly is running, including service names and version numbers, both valuable information in formulating an attack vector. While firewall FW1 is installed to prevent unwanted access to the internal network, it fails to prevent this since it allows traffic to reach the Web Server. The Web Server, running Microsoft IIS 7.5, is vulnerable to a remote privilege escalation exploit granting Bob root or administrative access to it. By gaining access to an internal device of the school network, Bob now has the ability to pivot from the Web Server into another attack on several other devices he now has connectivity with from his new position. While the workstations Bob now has vision of may be an option to launch a new attack against from his newly attained vantage point, the offer little value to him in terms of the information they may contain. The primary target of his multi-step attack is the File Server, which contains the exam papers along with other sensitive information. However, the Web Server does not have access to the File Server initially, prompting Bob to investigate other means of attack which could improve his stance in attaining his goal. From the perspective of a network administrator, it is crucial to prevent the compromise of such high value assets, such as the File Server and Database Server in this scenario. NVA helps to identify critical paths leading to the compromise of specific targets, allowing the network administrator to take the appropriate actions to solve the problem.

NVA Usage Walkthrough

In order to discover if any attack paths exist between the attacker and the File Server, the administrator uses NVA to analyse the network. The administrator opens the network configuration up in NVA and selects the File Server as its target. It is assumed that Bob will aim to gain root privileges to the File Server so that he can install a backdoor to allow for easier means of access in the future. The File Server is selected as a root target, as illustrated in Figure 5.2, and an attack graph is attempted to be generated by clicking the button shown in Figure 5.3. The selected target is not as merely being able to connect to the File Server but rather as a system state. This state is defined as the attacker having successfully compromised the File Server and attaining root privileges on it. Note the round green icon on the right of the NVA main menu, indicating that not attack graphs have been identified currently. Should an attack graph be identified, this icon changes to red to indicate that the target is in danger of being exploited.

The planner is not always capable of producing an attack path to a target device. A plan or attack path can only be generated if there exists an sequence of atomic actions(exploits)

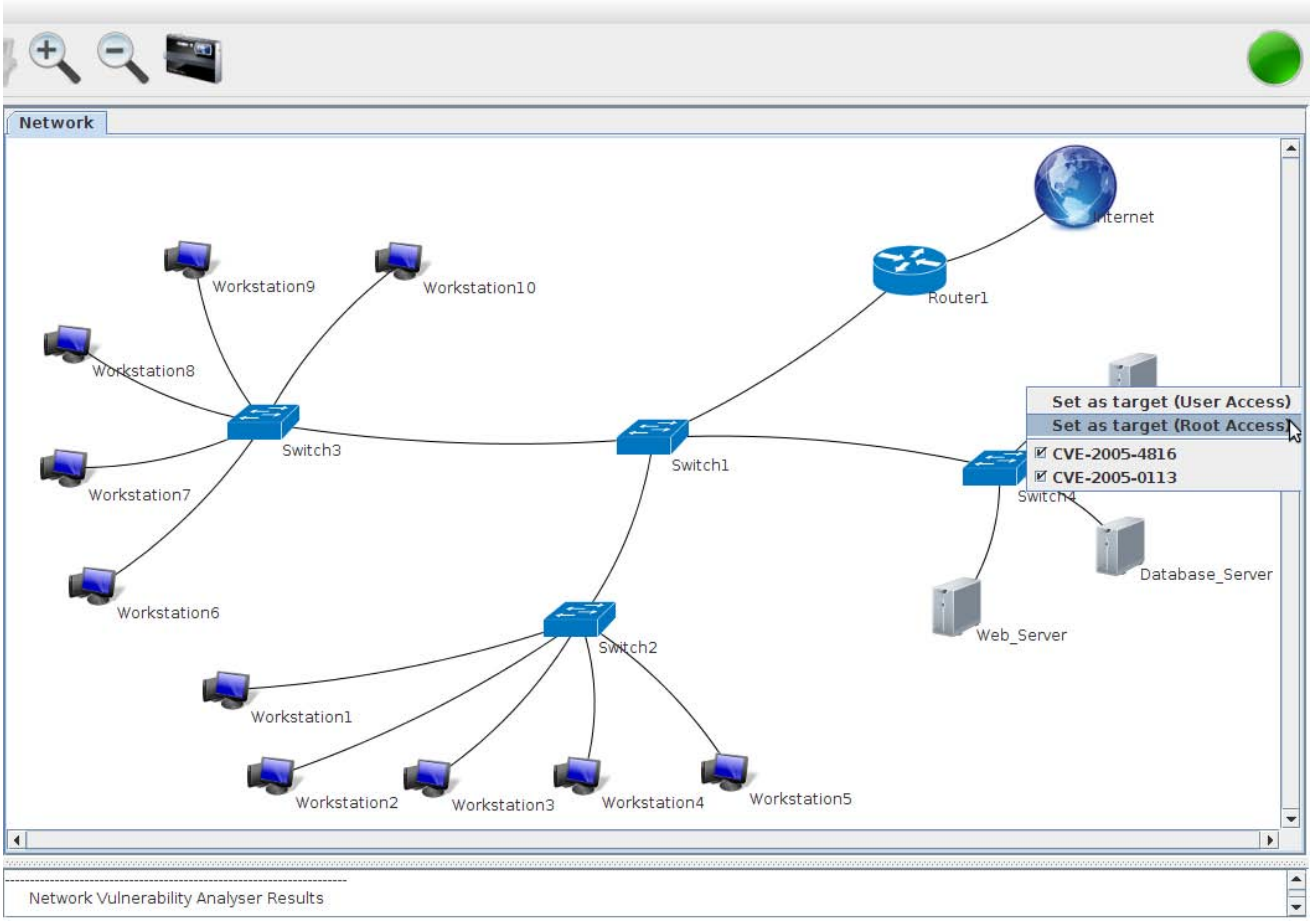


Figure 5.2: *A host is selected as a target for attack. NVA will try to find a critical attack path which results in the attacker obtaining root privileges to the target.*

Figure 5.3: *The model is sent to the planner to be processed and returns an optimal attack graph to a target devices, should it exist.*

which can transition the attacker from one state to another through out the network. In a network security model, state-transitions correspond with an attacker gaining access to various devices and elevating his privilege levels until he reaches a goal device. However, in this scenario NVA was able to produce an attack path reaching the File Server, illustrating how Bob could pivot through a combination of workstations and servers to reach his goal. In the Network View, we can now see that all devices involved in this attack have turned

red to outline critical devices which need to be addressed by the administrator. By selecting the Attack Graph Tab we can now see an ordered sequence of devices compromised in an attack on the File Server. Furthermore, the results panel below lists an enumeration of the vulnerabilities exploited to reach the target, illustrated in Figure 5.4. The results show that the most probable attack vector leverages vulnerabilities in the Web Server, Workstation9 and the File Server, achieving a CVSS total vulnerability severity rating of 33.2.

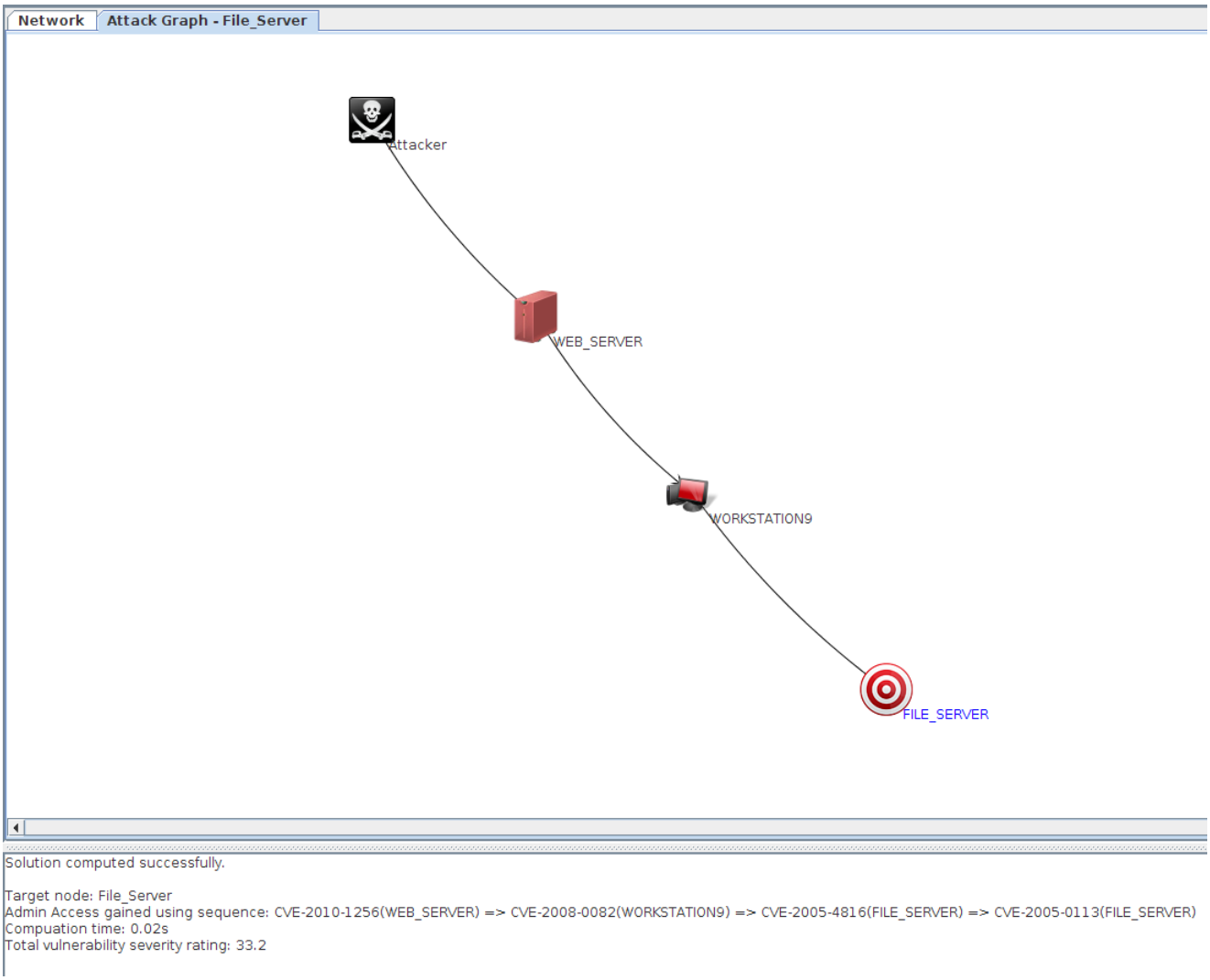


Figure 5.4: *A generated attack path, targetting the File Server of a small network.*

Presented with this critical attack path, the administrator needs to make a decision as to which devices to patch to prevent Bob from gaining root privileges to the File Server. One

may ask "Why not patch all vulnerable software running on the devices in the network?" The reason for dismissing this approach is that it may be financially infeasible to patch every single device in a network. Consider a large enterprise network running thousands of workstations which happens to all host a vulnerable version of a software product. To patch all of these devices may cost the company a large sum of money, while the vulnerability of these devices may play no critical role on the attack of high priority company assets. With this in mind, NVA uses a technique called Critical Action Analysis, geared towards highlighting the most crucial vulnerabilities, which when removed, eliminate all paths of attack to a target. By performing Critical Action Analysis on this attack path, we see in Figure 5.5 that the solution is to patch the Web Server, running Microsoft IIS 7.5 with vulnerability CVE-2010-1256.

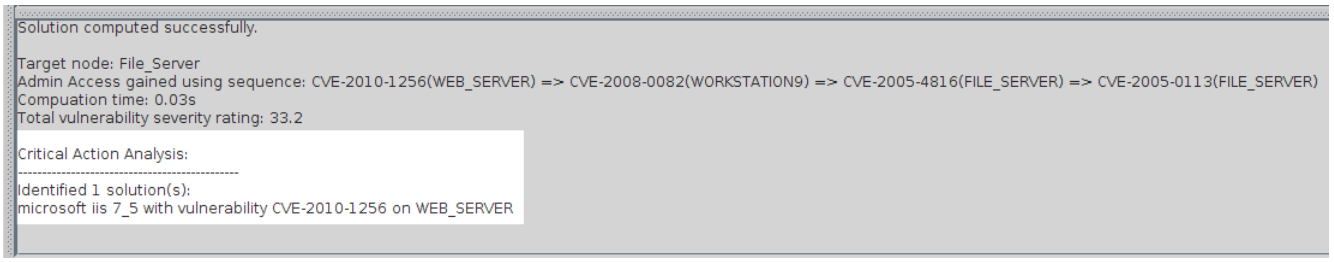


Figure 5.5: *Results computing the most practical software to patch in order to prevent a specific attack.*

The administrator is able to simulate the patching process in NVA by right clicking the Web Server and deselecting vulnerability CVE-2010-1256, shown in Figure 5.6. Now regenerating the attack model, NVA is unable to produce any attack paths leading to the File Server, demonstrating that the problem has been solved.

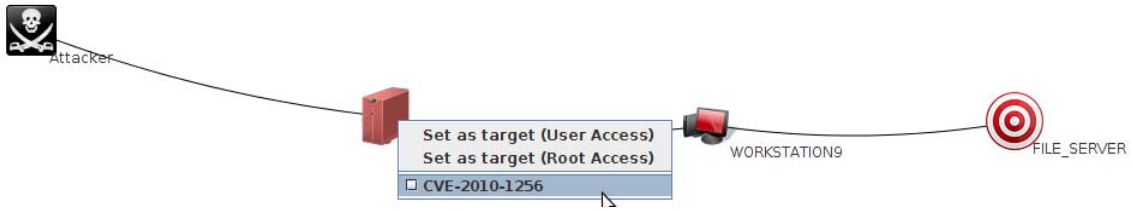


Figure 5.6: *By deselecting a vulnerability from a network device it is no longer vulnerable to it, therefore eliminating the threat it poses.*

Critical Action Analysis Computation

Dissecting the Critical Action Analysis process allows us to see why CVE-2010-1256 was selected for patching over any of the other vulnerabilities in the discovered attack path of critical actions. The algorithm aims to find the optimum exploit action to remove, such that when removed it prevents any routes to the target state. Simply removing an exploit action from the attack path identified does not guarantee prevention of the attack, since the attacker may be able to reroute the attack, pivoting through other devices not necessarily in the originally identified attack path, thus resulting in a new attack path. For this reason, the algorithm tries to find a solution by systematically removing each vulnerability and analysing the results. In this scenario we can visualize these results in Table 5.1:

Table 5.1 Each of the above vulnerabilities is sequentially removed from the network model and the attack paths regenerated to determine a solution.

Vulnerability ID	**Resultant Attack Path Generated**
CVE-2010-1256	×
CVE-2008-0082	CVE-2010-1256(WEB SERVER) $\Longrightarrow$ CVE-2008-2551(WORKSTATION8) $\Longrightarrow$ CVE-2005-4816(FILE SERVER) $\Longrightarrow$ CVE-2005-0113(FILE SERVER)
CVE-2005-4816	×
CVE-2005-0113	×

From Table 5.1 we notice that a set of 3 solutions have been identified; namely the removal of CVE-2010-1256, CVE-2005-4816 and CVE-2005-0113. The removal of CVE-2008-0082 results in another attack path being generated, circumventing the usage of CVE-2008-0082 on WORKSTATION9. The existence of an alternate route means that this is not a valid solution. The solution set is then arranged in order of their position in the attack path, providing each solution with a ranking. In ranking solutions we can select a solution which eliminates the threat as early as possible in the attack vector. The algorithm ranks the set of solutions as follows:

1. CVE-2010-1256 (WEB SERVER)
2. CVE-2005-4816 (FILE SERVER)
3. CVE-2005-0113 (FILE SERVER)

By patching Microsoft IIS 7.5 on the Web Server, vulnerability CVE-2010-1256 is removed from the network effectively eliminating Bob's point of access into the network. With no

means of further access into the network, Bob is no longer a threat to the network from an external perspective (from the internet).

Network-Wide Security Scans

Another element which offers value to securing the network as a whole is the Network Scan feature. Rather than the administrator individually selecting potential targets to test for attack paths he can use the Network Scan feature to reveal critical attack paths to all devices. This feature computes any possible attack path Bob may be able to use to get into the network, allowing the administrator to further investigate any results of concern. After running the Network Scan option, searching for vulnerabilities offering root privileges, we see that several attack paths have been identified, illustrated in Figure 5.7.

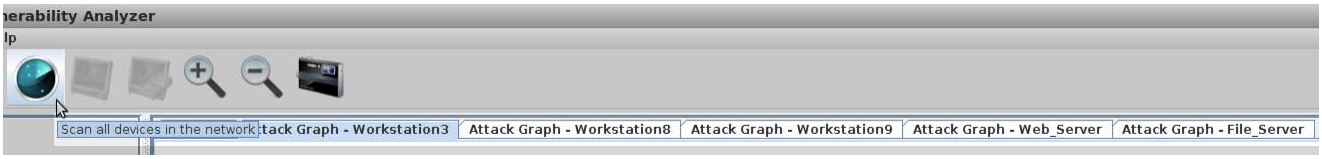


Figure 5.7: *A global scan on the network allows one to see all devices vulnerable to attack and the attack paths involved.*

Summary

In Case Study 1 we have shown how to identify potential targets of attacks using the Network Scan feature as well as the discovery of critical attack paths for specifically targeted devices. Furthermore, it was shown that solutions to these critical attack paths can be computed in order to secure the school network in a practical manner.

5.1.2 Case Study 2

In the previous case we looked at a small school network environment and revealed potential attack paths to critical systems. Case Study 2 focuses on a more realistic construct, showcasing typical network infrastructure of large enterprise environments. A larger network is a more challenging problem to solve since it adds to the complexity of identifying atomic steps in an attack. This is due to an increase in potential connectivity relations between

devices, more intricate topology structures and additionally, difficulty in determining critical paths due to an increase in the number of potential attack paths generated. Such a scenario provides an opportunity to test the applicability of NVA to real-world problems, since networks requiring a high degree of security are most likely to be larger organizations with many devices.

Network Topology

A fictitious marketing firm, named Expand Media, is analysed in this case, looking to secure their network from hackers after previously identified attempts. NVA was employed to identify potential paths of attack, aimed at the compromise of mission critical services. Similarly to Case Study 1, the devices this enterprise network contains include workstations, servers, switches, routers and firewalls. Figure 5.8 illustrates the network's topology, showing the campus network design. This design is used for large networks since it affords high-performance and improved reliability and manageability due to its scalability potential. Furthermore, it allows a network to span across several buildings or departments with its distributed multilayer architecture[55]. This suits Expand Media, considering that it contains a number of different departments based on different floors of the building.

A router connects the internal network to the internet and uses a firewall, FW1, to moderate ingress traffic from the internet. This acts as a first line of defence, filtering out any inbound traffic attempting to access any services not intended to be exposed to the public. Services exposed to the internet include the Exchange Server, Web Server and Proxy Server, housed within the demilitarized zone(DMZ). These services are typical features within an enterprise network and are kept separate from the intranet due to their exposure to the internet, for safety measures. An external user only has access to devices within the DMZ, making them an ideal target to attackers. Consequently, the Exchange Server, Web Server and Proxy Server have been placed in their own DMZ subnetwork, managed by the DMZ Switch, interfacing between the intranet and internet. A Core Switch acts as the central manager of intranet data flow and is responsible for linking several subnets, allowing internal communication between all departments and mission critical services.

Floor 1 has its own switch which manages the Sales Department and Services subnets. The Services Switch hosts internal services including the Database Server, File Server and Print

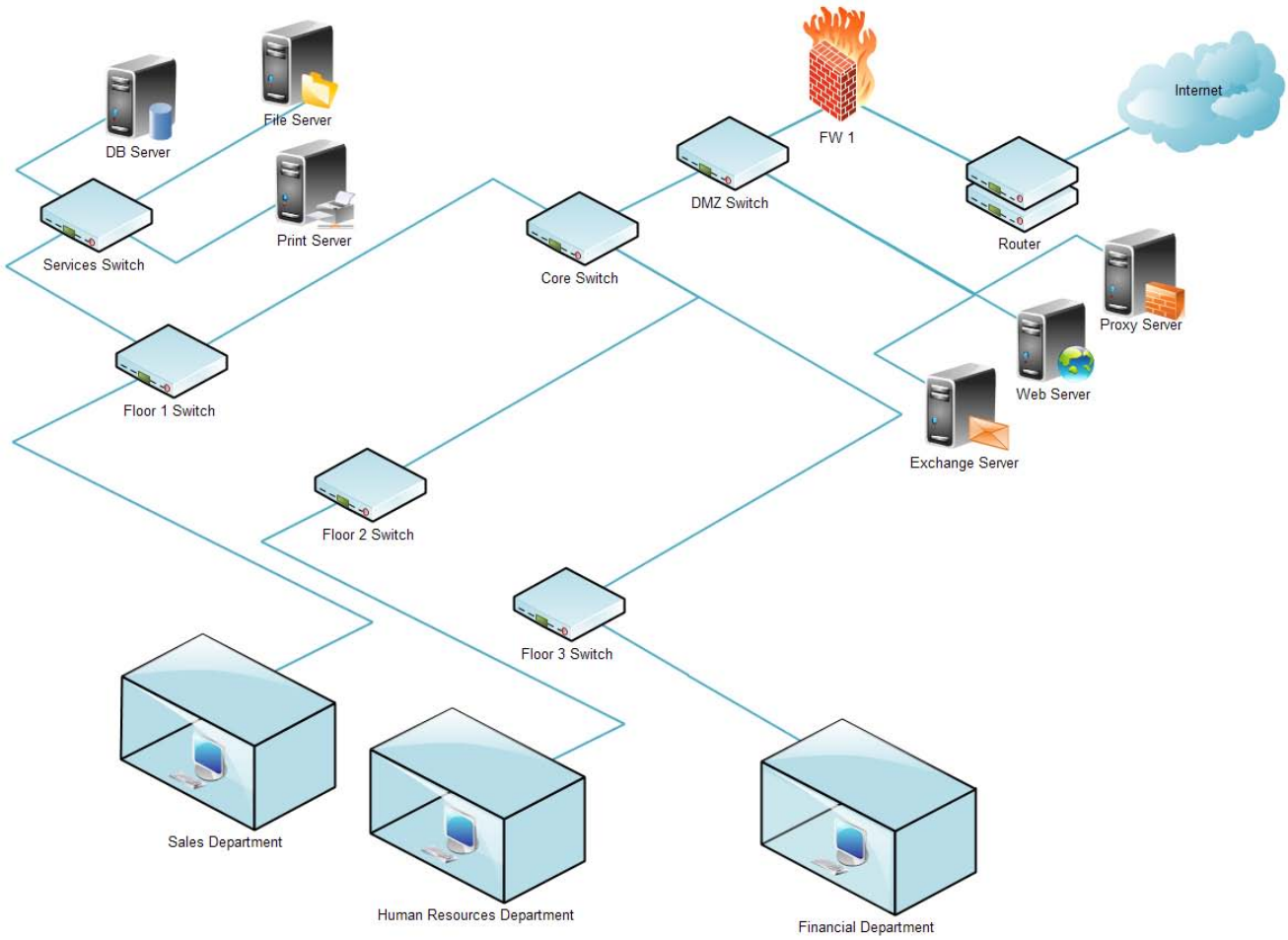


Figure 5.8: *Expand Media's network topology, showing the layout of a typical enterprise network.*

Server, which are crucial to the operation of the marketing firm. These are deemed as high value assets and are the focus of the firm's security audit using NVA. The Database Server stores information regarding the firm's clientele, employee records, payrole data and user credentials for management and departmental users. The nature of the content of this server yields an asset value of 10, where asset values can be ranked between 1 and 10. The compromise of such sensitive data would be devastating to the functioning of Expand Media's business. The File Server is used to store everything related to clients' products, ranging from completed, archived and incomplete products. Furthermore, it is used a repository for all the material required by employees to design and implement anything required for projects. This server is also considered as high value and is assigned an asset value of 10, since its compromise can lead to theft of property and subsequently a loss of revenue. The

Print Server offers the ability to print documents to all of the departments. While this is a commonly performed activity it is not deemed a mission critical function, since it is not an absolute necessity in the daily functioning of business. Consequently, the asset value assigned to it is much lower than that of the Database Server and File Server, ranked with a value of 3. It is important to note that access control lists (ACL) manage which devices have access to these services. These lists, stored within switches, are set in accordance with company policies, limiting which groups have access to certain networking resources. Departmental access will be further elaborated on in the following paragraphs, describing the connectivity with network resources as well as the implications there of.

The first floor plays host to Expand Media's Sales Department, mediated by Floor 1 Switch which controls a subnet of 10 workstations. These 10 workstation are dedicated to employees working in the Sales Department only, and as such only have access to networking resources required for sales-related tasks and projects. The ACL has been illustrated below in Table 5.2 to further describe visibility that devices within the Sales Department subnet have with devices throughout the network. This information is of importance since it outlines potential attack vectors a hacker can use by pivoting through exploited devices.

Table 5.2 A description of the ACL for the Sales Department.

Network Devices	**Workstation Connectivity**
Proxy Server	✓
Web Server	✓
Exchange Server	✓
Database Server	×
File Server	✓
Print Server	✓
Floor 1 Subnet	✓
Floor 2 Subnet	✓
Floor 3 Subnet	✓

It is important to note from Table 5.2 that sales employees have access to mission critical network services; namely the File Server. Projects and finished products are accessed from and stored on the File Server, making this a fundamental network resource to the functioning of Expand Media. The Database Server is not accessible by Sales Department employees owing to the fact that they should not have access to any of its content.

The second floor is managed by the Floor 2 Switch and regulates communication between the Human Resources subnet and the rest of the internal network. The Human Resources departments is comprised of 10 workstations designated to Human Resources employees only. Table 5.3 depicts the connectivity relations workstations within this subnet have within Expand Media's network.

Table 5.3 A description of the ACL for the Human Resources Department.

Network Devices	Workstation Connectivity
Proxy Server	✓
Web Server	✓
Exchange Server	✓
Database Server	×
File Server	×
Print Server	✓
Floor 1 Subnet	✓
Floor 2 Subnet	✓
Floor 3 Subnet	✓

The ACL restricts these employees from accessing key mission critical resources such as the Database Server or File Server on account of this department storing printed copies of records. This reduces the probability of workstations within this subnet being targeted since they offer no direct route to primary targets; notably the Database or File Servers. However, attacks on this subnet should not be ruled out considering that they do have connectivity to other departmental subnets and the publicly exposed DMZ.

The Financial Department is on the third floor and is managed by the Floor 3 Switch. Floor 3 Switch controls a subnet of 10 workstations whose usages revolve around all activities expected of the Financial Department. Employees in this department cover a wide range of bookkeeping activities including, day to day transactional accounting, management of the organization's cashflow, budget management, sales and expenditure forecasts, and reporting back to management to assist in making key strategic decisions. The Financial Department play an indispensable role in any enterprise, providing essential information used to maximize profits. Table 5.4 illustrates the connectivity that workstations in this subnet have within the network.

The ACL has some interesting differences when compared with those of the Sales Department and Human Resources Department. Financial Department users have access to sensitive

Table 5.4 A description of the ACL for the Financial Department.

Network Devices	Workstation Connectivity
Proxy Server	×
Web Server	×
Exchange Server	×
Database Server	✓
File Server	✓
Print Server	✓
Floor 1 Subnet	✓
Floor 2 Subnet	✓
Floor 3 Subnet	✓

financial information stored within the Database Server, which should not be made available to any other users for security reasons. In an effort to enhance security in this department, workstations do not have access to any publicly exposed devices, such as the Proxy Server, Web Server or Exchange Server. The aim of this is to avoid the threat of a direct attack on this high priority subnet from a compromised DMZ service.

Network Device Configuration Information

In this section we elaborate on vulnerable products identified on devices within the network. Enumeration of this information is a necessity while auditing the overall security of any network, since it provides insight into potential weaknesses which can be exploited. A detailed overview of this information is provided in Appendix B, highlighting the most important elements of each vulnerability. This information is used in constructing a more comprehensive security assessment of Expand Media's network by constructing attack graphs.

Scenario Description

Expand Media worked its way up from humble beginnings to become a leader in the advertising world. Having established themselves as a dominant force in their field, their success has lead them to expand into a large enterprise. Steeped in fierce competition, Expand Media has developed a number of rivals in the battle for high-end customers. In recent events, Expand Media managed to capture some key clientele, creating a stir amongst their competition. Rival company Apex Marketing, aiming to win some clients of Expand Media,

hired a hacker to break into their network and steal the customer database and any intellectual property of interest. Initial attempts at this raised the alarms, prompting the network administrator to look into better securing the network.

The network administrator realised that patching every vulnerable software device in the enterprise network would be costly and impractical. Moreover, the downtime involved in patching every server and workstation would be costly for Expand Media since it interfere with users' work flow. NVA was employed to identify key points of interest, which when secured could help to eliminate threats posed against high value organizational assets. The compromise of the customer database would be devastating to Expand Media and will likely lead to large losses in terms of clientele and revenue. Consequently, the overall security of the network is also crucial in protecting the company's financial status.

NVA Usage Walkthrough

An assumption was made for this case that the attacker was targeting the network from the internet. In terms of the network configuration this suggested that initially the attacker would only have access to publicly exposed devices; namely the DMZ. Network fingerprinting would reveal to the attacker the 3 services contained within the DMZ. Using NMAP, the attacker can scan Expand Media's website to retrieve its public IP address. Once this IP is obtained further probing can be performed by consulting the WHOIS service. WHOIS is a protocol used for querying online databases of registered domain names and internet resources[56]. A WHOIS query can provide information to the attacker in a human readable format on other domains registered to Expand Media, such as he Proxy Server and Exchange Server.

An automated preliminary vulnerability scan performed by NVA on individual devices produced the following useful information:

```
-----------------------------------------------------------------
     Network Vulnerability Analyser Results
-----------------------------------------------------------------

Network contains 45 devices, including:
```

```
PCs: 31
Servers: 6
Switches: 6
Routers: 1

Network contains 10 vulnerabilities.
```

As mentioned previously, the focus of this audit was to help secure mission critical resources; namely the Database Server and File Server. The Database Server was marked as the first target for attack, in an attempt to gain root privileges. Hitting the Generate Graph button in NVA revealed that the Database Server was reachable with a multi-step attack. Figure 5.9 shows several devices highlighted in red, which were pivoted through in this attack to reach the final target.

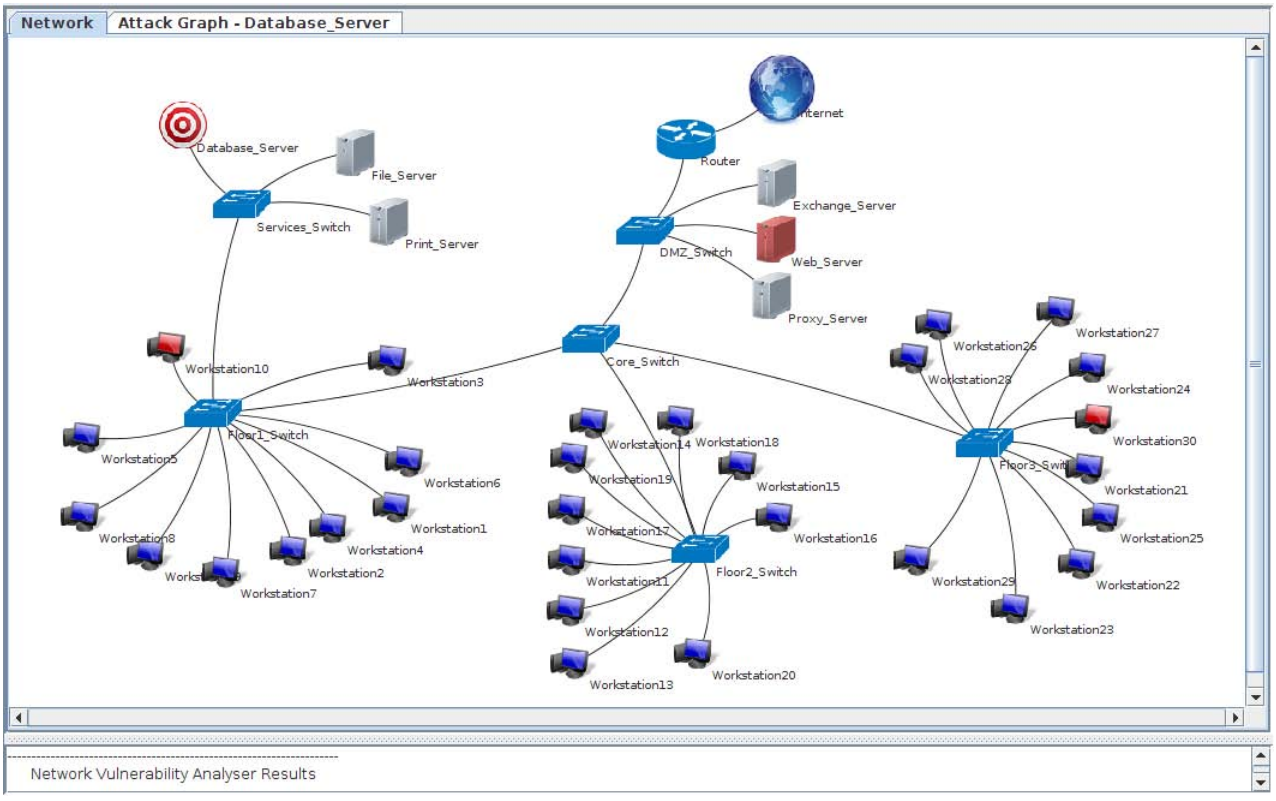


Figure 5.9: *Scan results of the Database Server reveal that it is not safe from attackers based on external networks.*

These results are concerning since they demonstrate that the Database Server is not safe from externally based attackers. Looking at the Attack Graph tab shows the critical attack

path generated by the planner. It illustrates the sequence of devices compromised in this attack, including both publicly exposed devices as well as devices which should only be accessible within the local area network. Figure 5.10 illustrates the critical attack path below.

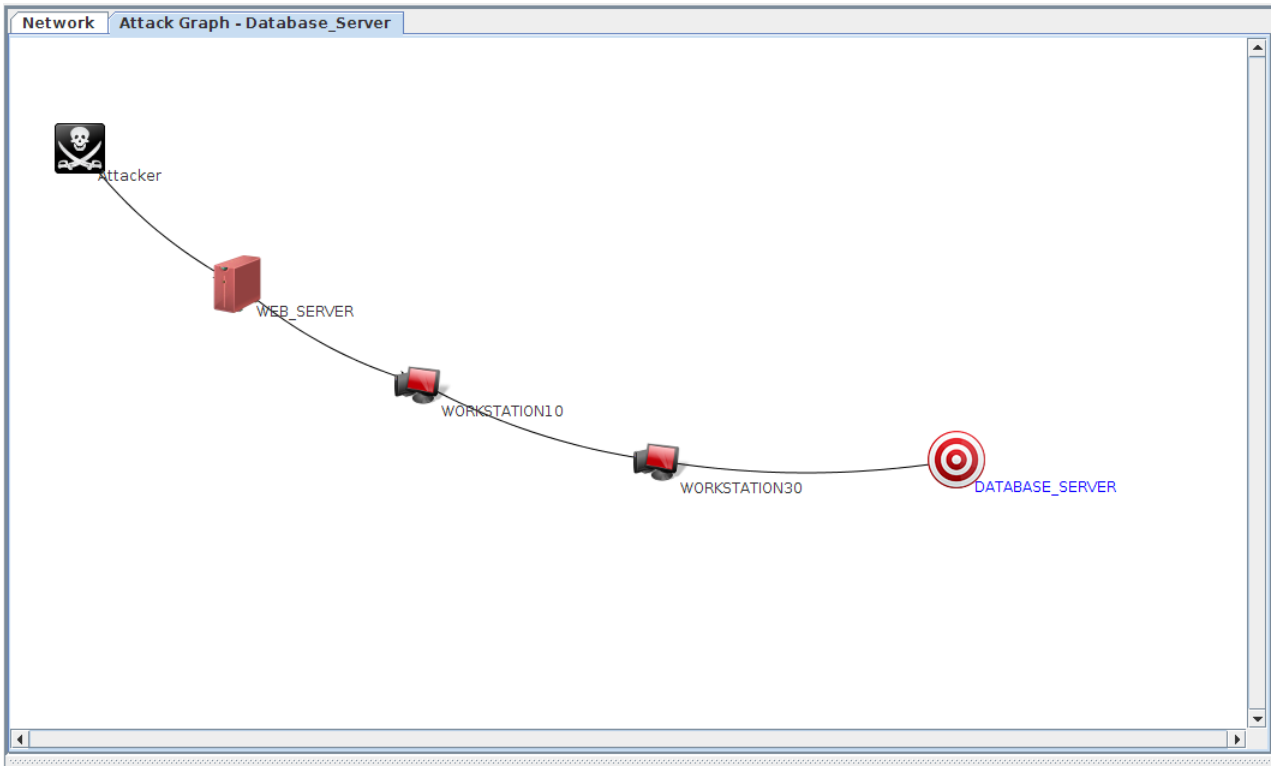


Figure 5.10: *The Database Server is vulnerable to attack and can be reached by pivoting through a number of network devices.*

Analysing results produced by the planner reveals several interesting metrics about the generated critical attack path. These results are shown below:

```
Target node: Database_Server
Admin Access gained using sequence: CVE-2008-0075(WEB_SERVER)
=> CVE-2004-0409(WORKSTATION10) => CVE-2004-0409(WORKSTATION30)
=> CVE-2004-1351(DATABASE_SERVER)
Compuation time: 0.04s
Total vulnerability severity rating: 35.0
```

The aforementioned attack sequence shows both the devices compromised as well as the vulnerabilities which would have been exploited in the process. Walking through this se-

quence it was discovered that the attacker's point of entry was the Web Server, exploiting a vulnerability in the Microsoft IIS 6.0 software. This vulnerability can be executed remotely and grants root privileges to the attacker. With a compromised device which acts as an interface between the internet and internal network the attacker could then gain visibility of the internal network by performing broadcast scans to detect live hosts. According to Table 5.2, Table 5.3 and Table 5.4 the attacker would only have access to workstation within the Sales Department and Human Resources Department. While the compromise of devices within these subnets would not provide direct access to the Services subnet, they do present an opportunity to further explore the internal network for other possibilities. The next victim was Workstation10, compromised via a vulnerability in a chat application the user was running. A port scan of Workstation10, from the compromised Web Server, would reveal to the attacker an open port associated with XChat 1.8.0. The exploit offers user privileges to the device, which is sufficient for him to continue further penetration into the network. This also outlines the importance of restricting the kinds of applications allowed to be run on workstations or eliminating the ability to install any applications. An employee who is able to install software on workstations may inadvertently be creating an opportunity for an attacker to gain entrance to the corporate network. The attacker discovers that the user working on Workstation10 was in fact chatting to another user in the Financial Department's subnet, on Workstation30. Executing the same exploit against Workstation30 gained the attacker user level privileges, but more importantly visibility of the main target, since workstations in this subnet are able to connect to the Database Server. By pivoting to Workstation30 the attacker would have the Database Server in his cross-hair, with direct accessibility. Performing an NMAP scan on the Database Server exposed useful information to the attacker. An outdated operating system, Solaris 8, with known security holes provided the foundation for the Database Server. These favourable circumstances allowed the Database Server to be exploited via a vulnerability in the *rwho* daemon. Successfully executing this exploit presented the attacker with root access to the final target and a means to steal sensitive client information for Apex Marketing.

A second audit was also performed on the File Server to identify whether there was any threat of it being compromised. This server is of interest to the attacker, considering that the intellectual property stored on it could be useful to a rival organization if obtained. Marking the File Server as a target and attempting to gain root privileges revealed similar feedback to the previous target. The output below provides some insight into the most likely

attack path an attacker would use to compromise the target.

```
Target node: File_Server
Admin Access gained using sequence: CVE-2008-0075(WEB_SERVER)
=> CVE-2004-0409(WORKSTATION10) => CVE-2004-1351(FILE_SERVER)
Compuation time: 0.04s
Total vulnerability severity rating: 27.5
```

The pattern of the attack above is quite similar to that of the Database Server, with an identical point of entry through the Web Server, leading to the compromise of Workstation10. However, the attack path differs from that point onwards, requiring one attack step less to reach the final target. If we look at Table 5.2 we notice that Workstation10 in the Sales Department does in fact have direct access to the File Server. The File Server, running the same vulnerable version of Solaris 8 as an operating system, was exploited after pivoting through Workstation10, providing the attacker with root privileges and access to sensitive files.

Critical Action Analysis Computation

In order to rectify the security issues identified above it is important to patch vulnerable software on devices which acted as conduits for an attacker to reach his targets. Critical Action Analysis points the administrator to this solution by computing points in the critical attack path, which when eliminated completely prevent the attacker from reaching his goal. Performing this calculation on the critical attack path identified for the Database Server produced the following result:

```
Critical Action Analysis:
-------------------------------------------
Identified 1 solution(s):
xchat xchat 1_8_0 with vulnerability CVE-2004-0409 on WORKSTATION10
```

In essence, by eliminating vulnerability CVE-2004-0409 on Workstation10, by either patching the software or uninstalling it completely, prevents the compromise of the Database Server. Asset Action Analysis was also performed on the critical attack path, which is based on the same algorithm as Critical Action Analysis, but sorts potential solutions from the highest asset value to the lowest. Results of Asset Action Analysis provided the following information:

```
Asset Action Analysis:
-----------------------------------------------
Identified 1 solution(s):
sun solaris 8_0 with vulnerability CVE-2004-1351 on DATABASE_SERVER
```

Here we see that the solution selected was vulnerability CVE-2004-1351 on the Database Server, as opposed to CVE-2004-0409 on Workstation10. While both of these solutions prevent the attacker entirely from reaching the target, the Database Server's vulnerability is selected in Asset Action Analysis since the device's value is 10 while Workstation10's is only 1. In this case study the Asset Action Analysis is of more interest in producing a solution since the compromise of a high valued asset can have a very large impact on the company. In an enterprise network environment where hundreds or thousands of devices may be reliant on the availability of certain services it is more beneficial to prioritise the patching of higher valued assets.

Summary

In Case Study 2 we looked at the applicability of NVA to much larger networks such as those present in enterprises. A campus network topology was analysed, typical in enterprise networks, which added made the discovery of critical attack paths more challenging due to an increase in potential actions at each step of attack. NVA proved that it was fully capable of identifying critical attack paths in a network containing 45 network devices and could correctly compute solutions to prevent identified attack vectors.

5.2 Performance and Scalability of NVA Framework

In Case Study 1 the Network Scan feature was introduced for computing network-wide scans of all existing critical attack paths. From a single point of entry, it tries to determine whether each device in the network is reachable by an attacker. While this feature adds benefit in terms of identifying potential threats which may not have been previously discovered, it can become computationally expensive as the network grows in size and complexity. The practicality of the Network Scan feature is analysed in this section to determine whether it may be of use in scenarios with large networks. An issue which was acknowledged is that the amount of time taken for a Network Scan to complete should be within reasonable bounds

to make this feature useful. Previous techniques which used exhaustive searches to generate complete attack graphs would become impractical due to the computation time required to analyse larger networks. The aim of this section was to see whether a planner can be applied to this problem to perform partially exhaustive searches by guiding attacks on each device using heuristics.

Evaluation methods used involved analysing the time taken to scan randomly generating networks of increasing size. Network generation was done using the network generator which was built into NVA. A basic campus network design was used for each network, where workstations or servers form clusters, connected to switches and routers. Vulnerable software were selected at random from NVD and assigned to randomly selected hosts, thus adding to the complexity by making not every host exploitable. In Figure 5.11 networks of increasing size are depicted with the amount of time taken to compute a Network Scan. Scans were focused on finding critical attack paths that resulted in the attacker gaining root privileges only. Tests were performed on a laptop with Intel Core2 Duo CPU T6670 @ 2.20GHz x 2 and 2.8GB memory, running 32 bit Ubuntu Linux.

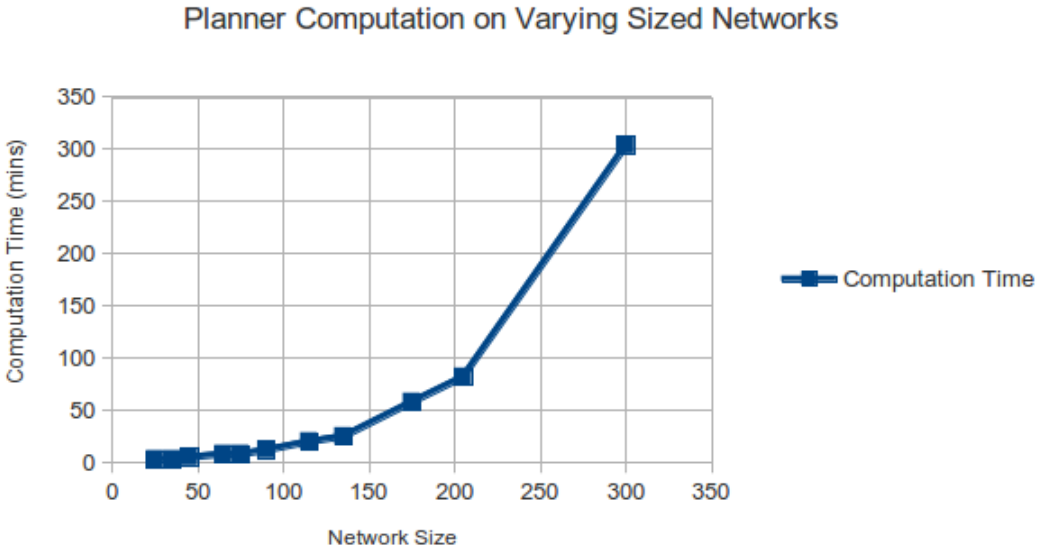


Figure 5.11: *The total time taken to perform a Network Scan is the summation of the computation of every critical attack path for a network's devices.*

From Figure 5.11 we can see a significant rise in the amount of time taken to compute critical attack paths as the network size increases. This is mainly due to the drastic increase in complexity caused by increasing the size. Essentially, the size increase translates into more

planner actions during the evaluation of each attack step. This is due to the fact that a larger network presents the attacker with more possibilities in terms of device connections and vulnerabilities to exploit. In larger networks the primary function of NVA is to assess the risk of attack posed against selected devices, by identifying high priority assets which are likely targets. However, as shown in Figure 5.11 it is still possible to do network wide scans of medium size networks in a reasonable amount of time.

NVA was intended to be used to plan attacks against realistically sized networks, encouraging the analysis of NVA performance on medium to large networks. In order to simplify the generation of networks certain adjustments were made to the initial algorithm to speed up the process. Typically, network generation selects vulnerable software from NVD at random and randomly assigns them to PCs or servers. Vulnerability selection was restricted by creating a pool of 20 different vulnerable software, from NVD, for selection and assignment to devices. In large enterprise networks workstations usually have a uniform set of software installed on them, differing only between departments where users may require different types of software. This restricted range of software installations means that there may be a large amount of vulnerable devices present in a network with the same vulnerability sets. This rationale suggested the restriction of the vulnerability database to a small pool of products while generating large scale networks. The results of several large networks tested are displayed in Table 5.5 below.

Table 5.5 Network wide scans performed on large scale networks.

Network Size	**Computation Time**
300 Devices	3m 4s
500 Devices	13m 2s
1000 Devices	1h 33m 55s

The implications of reducing the pool of selectable vulnerable software for the generated network resulted in a reduced amount of operators in the planning domain. This directly affects memory consumption since much fewer exploit actions are required to be performed at each attack step during the plan construction phase. Table 5.5 demonstrates the Network Scan feature can be used successfully on large enterprise networks within a reasonable amount of time if the number of different exploit actions is limited to a smaller range.

5.3 Assessment of Research Objectives

We set out to determine whether this research could address a number issues which have greatly limited attack graph application to real world scenarios. While previous approaches were promising, they focused mainly on implementing proof of concept prototypes which could evaluate the security of small-scale networks. Challenges in computing attack graphs are listed below.

- Exhaustive enumeration of each possible state of attack leads to state-space explosion, due to the exponential nature of the problem. This means that it is impractical to build attack graphs of large and intricate networks.
- Generation of complete attack graphs are difficult to visualise since even the smallest networks can produce massive graphs, which quickly expand latitudinally.
- Exploit actions need to be strictly defined in terms of preconditions and postconditions. Currently, no publicly available databases exist which specify exploits in this standardised format.

C. Sarraute et al.[18] have proposed the closest related work, which integrates an automated planner into the Core Impact penetration testing framework.[1] They extract the description of attack exploit actions, in terms of requirements and results, from the Core Impact's extensive exploit database.

Having now done an evaluation on two scenarios, as well as testing NVA's performance and scalability potential, we can now report on its effectiveness at tackling the aforementioned research objectives. Each of the research objectives stated in Table 5.6 were achieved, illustrating NVA's benefit in identifying points of weakness in medium and large networks. The prototype was successful in discovering the expected attack vectors for targeted devices in both case studies, and could present optimal solutions which eliminated the threats.

[1] http://www.coresecurity.com/core-impact-pro

5.4 Chapter Summary

In this chapter we demonstrated the applicability of NVA to both small networks and large scale enterprise networks. In Case Study 1 it was shown that critical attack paths could be generated for a simple network featuring only 13 hosts. Moreover, NVA was shown to produce the most beneficial solution to protect the targeted assets from an attacker. In Case Study 2 a much larger enterprise network was audited by NVA, to reveal critical paths which would lead to the compromise of high value hosts. Both Critical Action Analysis and Asset Action Analysis were used to produce a solution to these threats, outlining the importance of prioritising asset values in these solutions for such a scenario. In the Performance and Scalability section we looked at how well the Network Scan feature could scale up for larger networks. It was shown that while this feature is more useful for smaller networks, it can scale up for large enterprise networks of 1000 hosts in a reasonable amount of time.

Table 5.6 An analysis measuring how effectively the research objectives were satisfied.

Research Objective	**Result**
Attack Type Identification	NVA sources exploit information from the publicly available vulnerability database NVD, which presents vulnerabilities using the Common Vulnerability Scoring System (CVSS). Analysis of the type of data and structure of entries in vulnerability databases revealed that planners can currently best model exploits which result in the elevation of an attacker's privileges on a machine. Denial of service attacks can be modelled by simply switching off the availability of a service, after the compromise of its host. However, at this point in time, vulnerability databases do not supply a machine-readable field which can specify whether an attack results in privilege escalation, denial of service or information disclosure. For this reason, NVA focused on vulnerabilities which result in the escalation of privileges, since they clearly define the preconditions and postconditions of their exploitation.
Critical Path Enumeration	By incorporating and automated planner (SGPlan5) into NVA, attacks are searched for with a heuristic, without constructing complete attack graphs, thus solving the state-space explosion issue. SGPlan5 consistently identifies the optimum critical path of attack by using a heuristic function that optimizes the aggregated CVSS severity score of attack paths. A by-product of critical attack path computation is the solution to the visualization issue which plagued previous attack graph approaches. The linear nature of a critical attack path simplifies the visualization of attacks and also makes it easier for users to comprehend threats.
Network Strengthening Analysis	Two analysis techniques were developed to aid in strengthening networks in which critical attack paths are identified. The graph analysis algorithms successfully compute solutions which prevent an attack from occurring, each based on differing priorities. The first algorithm eliminates an attack threat by causing the maximum disconnection in a critical attack path, by removing a single node to prevent an attack from further progressing to its target. Similarly, the second algorithm produces the maximum disconnection in a critical attack path, while prioritising the protection of machines with high asset values.
Performance and Scalability	SGPlan5 consistently computed the expected critical attack paths for each test case within reasonable time frames. Test cases with networks up to 1000 hosts were computed by SGPlan5 without the framework crashing or becoming non-responsive.

Chapter 6

Conclusion

This chapter starts off by briefly summarizing some of the key topics covered in the preceding chapters. It also compares the results discovered with initially proposed objectives, to determine the degree of success of this thesis. Lastly, it provides perspective on potential propositions for future work.

6.1 Summary

Overwhelming amounts of network information make it very difficult for security administrators to manually determine weaknesses in IT infrastructure as a whole. Attack graphs use network configuration information, including topological data and device information, as well as vulnerability information to derive an overview of potential multi-step attacks. Many large-scale enterprise networks are monitored by Security Information and Event Management (SIEM) systems, which aggregate and correlate event data to alert security administrators of possible attacks in progress. These systems have detailed knowledge about the infrastructure being monitored and can be enhanced with an automated attack modelling tool. The amalgamation of these two technologies benefit security administrators during risk assessment phases, which typically should be conducted whenever a network is updated. This thesis researched the issue with current attack graphing approaches and found that they are severely limited in their abilities due to a combinatorial problem, whereby an exhaustive search of every possible attack is not feasible for medium- to large-scale networks. This thesis has identified four objectives for its work. They were:

- Identification of the kind of attacks that can be modelled using an attack planning approach.
- Implementation of a proof of concept prototype to determine whether attack paths can in fact be enumerated for selected network devices.
- Investigation into analysis techniques which can best be combined with automated planning to strengthen points of weakness in networks.
- Measurement of scalability and performance of the prototype against networks of increasing size and complexity.

Previous attack graphing approaches have not been successful in providing an implementation which is practical to real-world application. Proof of concept implementations were applied to small networks and only modelled small sets of exploits. These techniques built complete attack graphs, exhaustively searching for every possible attack path. Analysis of these approaches has shown that they have exponential complexity, suffering severely from the state-space explosion problem, also known as the combinatorial problem. Another issue plaguing previous approaches is the lack of up-to-date and complete exploit information for vulnerable software products. To tackle the complexity issue, investigation was done into a field of Artificial Intelligence known as Automated Planning. It concerns itself with a deliberation process of action sequences by measuring their expected outcomes. This deliberation process aims to find an optimal solution path to a goal by guiding a search through a state-space with a heuristic, rather than building a complete graph.

A design was presented, which incorporated automated planning with publicly available vulnerability information from NVD into a proof of concept framework named Network Vulnerability Analyser (NVA). Using a network configuration information model, NVA can cross-examine installed software products with a vulnerability database to determine what kind of vulnerabilities exist. Vulnerabilities present in a network are transcoded from their NVD database entries into exploit action models, based on vulnerability preconditions and postconditions. Both the set of exploit action models and network model are converted into PDDL, a standardised automated planner language. Further research was conducted to decide which automated planner would be most suited to computing network security plans. Results from The Fifth International Planning Competition (IPC-5), an automated planner benchmarking competition, concluded that SGPlan5 is most suitable due to its overall

consistency and ability to manage both complex and simple problems. Upon further consideration, it was decided that a resultant critical attack path from the planner was not sufficient information for a security administrator to use as a solution. Two analysis algorithms were developed to aid the user in making informed decisions as to which devices needed to be patched to ultimately prevent the attack from occurring. The first algorithm was designed to single out a device, which when patched, eliminates the threat and also minimizes the impact of an attack on the network. On the other hand, the second algorithm was designed to identify a device to patch based on the asset values of devices in the network. NVA was designed to accept network configuration information in XML documents. The decision behind this was to support information input from other systems, such as SIEMs, where network configuration information is readily available.

Evaluation of NVA was broken down into two main sections: case study demonstrations and scalability analysis. Two case studies were experimented with, each of which aimed to highlight different functions offered by NVA. In the first case study a small scale network is considered, with a simple topological layout. The network was divided up into three different subnets, two of play host to workstations, while the other houses a set of servers. Servers running in the third subnet include a web server, file server and database server. Tests against the file server demonstrated that NVA was able to discover a critical attack path which lead to its compromise. Furthermore, a solution to the attack was identified using critical action analysis. It showed that patching a vulnerability in the web server would not only prevent the file server from being compromised, but also prevent an external attacker from gaining any access to the internal network at all. The second case study examined a large scale network which was a lot more complex in nature. This network simulated the structure of an enterprise network, constituted of a hierarchical structure of routers and switches to emulate a campus network topology. A DMZ subnetwork, interfacing with both the internet and internal network, comprised of an exchange server, web server and proxy server. The internal network was divided into three subnetworks of workstations, each of which were used by different organizational departments. Moreover, a services subnetwork on the internal network hosted a database server, file server and print server. In this network the connectivity relationships between devices was restricted by firewall ingress and egress rules, which add to the complexity of computing critical attack paths. NVA proved that it was fully capable of identifying critical attack paths in a network containing 45 network devices and could correctly compute solutions to prevent identified attack vectors. It also

demonstrated how a solution could be identified by prioritizing the asset values of devices in the network. NVA was intended to be used to plan attacks against realisticly sized networks, which encouraged the testing of its performance on networks of increasingly large scale. Scalability tests were conducted which aimed to identify critical attack paths for every host in a network. Results showed that it can scale up for large enterprise networks of 1000 hosts in a reasonable amount of time.

In regard to the proposed research objectives, this thesis has succeeded in what it aimed to achieve. The results of the prototype showed the following:

- Analysis of the type of data and structure of entries in vulnerability databases revealed that planners can currently best model exploits which result in the elevation of an attacker's privileges on a machine. Denial of service attacks can be modelled by simply switching off the availability of a service, after the compromise of its host. However, at this point in time, vulnerability databases do not supply a machine-readable field which can specify whether an attack results in privilege escalation, denial of service or information disclosure. For this reason, NVA focused on vulnerabilities which result in the escalation of privileges, since they clearly define the preconditions and postconditions of their exploitation.

- Evaluation of case studies illustrated how NVA can enumerate critical attack paths, using an automated planner (SGPlan5) to guide the state-space search with a heuristic function. The heuristic function guided searches by discovering paths which yield the maximum aggregated CVSS severity score of attacks.

- Two analysis techniques were developed to aid in strengthening networks in which critical attack paths are identified. The graph analysis algorithms successfully compute solutions which prevent an attack from occurring, each based on differing priorities. The first algorithm eliminates an attack threat by causing the maximum disconnection in a critical attack path, by removing a single node to prevent an attack from further progressing to its target. Similarly, the second algorithm produces the maximum disconnection in a critical attack path, while prioritizing the protection of machines with high asset values.

- SGPlan5 consistently computed the expected critical attack paths for each test case within reasonable time frames. Test cases with networks up to 1000 hosts were com-

puted by SGPlan5 without the framework crashing or becoming non-responsive.

SIEM can enhance the effectiveness of NVA since it aggregates extensive knowledge about the network it monitors. It is highly suitable as a knowledge source for the topological structure of networks, connectivity relations between devices, as well as host configuration information such as software products installed. The highly integrated nature of tools incorporated into SIEM, which manage things such as network discovery and device configuration information aggregation, can streamline NVA's security analysis by providing it with network information.

6.2 Future Research

Contributions made by this thesis concentrated on expressing the potential that automated planning can bring to the attack modelling field. Shortcomings of this thesis stem mainly from the lack of availability of exploit information. Currently, no publicly available databases exist which express exploits in terms of their requirements for successful execution, and resultant outcomes. Details which describe the difficulty of execution, the average execution time, the type of exploit, code availability and effects it has on a network, would be highly beneficial for attack modelling. It would allow attack probabilities to be factored in, which in combination with exploit severity ratings, could optimize attack path computation even further. While the construction of such a database would have its challenges, further research into such endeavours may prove fruitful. Moreover, it is envisioned that a set of software Application Programming Interfaces (API) be developed to better interface with SIEMs and network discovery tools. Integration of these APIs into NVA would simplify the network configuration input process. Additionally, it could aide in providing alerts and security reports back to a SIEM's main dashboard. Another consideration for future work, looks at the introduction of Common Configuration Enumerations (CCE)[1] into the framework. The CCE database enumerates weaknesses in system and software configurations. Consequently, this would also require the input of richer device information which provides configurations for installed products.

[1] http://nvd.nist.gov/cce/index.cfm

References

[1] ZOHO Corp. Analyzing logs for security information event management. Technical report, ZOHO Corp, 2007.

[2] Amrit Williams. The future of SIEMthe market will begin to diverge. Available at: `http://techbuddha.wordpress.com/2007/01/01/the-future-of-siem`.

[3] Joey Hernandez. Security information and event management:business benefits and security,governance and assurance perspectives. *ISACA Journal*, 2010.

[4] CRN. Siem: A market snapshot. Available at: `http://www.crn.com/news/security/197002909/siem-a-market-snapshot.htm`.

[5] HENRIK KARLZN. An analysis of security information and event management systems. Master's thesis, UNIVERSITY OF GOTHENBURG, 2009.

[6] M. Nicolett and K.M. Kavanagh. Magic quadrant for security information and event management. *Gartner RAS Core Reasearch Note (May 2009)*, 2011.

[7] J. Oltsik. The SIEM architecture. Technical report, The Enterprise Strategy Group Inc., 2007.

[8] Sensage. A practical guide to next-generatioin siem. 2012.

[9] R. Ortalo, Y. Deswarte, and M. Kaaniche. Experimenting with quantitative evaluation tools for monitoring operational security. *Software Engineering, IEEE Transactions on*, 25(5):633 –650, sep/oct 1999.

[10] P. Mell, K. Scarfone, and S. Romanosky. A complete guide to the common vulnerability scoring system version 2.0. In *Published by FIRST-Forum of Incident Response and Security Teams*, pages 1–23, 2007.

[11] P. Meil, T. Grance, et al. Nvd national vulnerability database.

[12] L.P. Swiler, C. Phillips, D. Ellis, and S. Chakerian. Computer-attack graph generation tool. In *DARPA Information Survivability Conference & Exposition II, 2001. DISCEX'01. Proceedings*, volume 2, pages 307–321. IEEE, 2001.

[13] R.W. Ritchey and P. Ammann. Using model checking to analyze network vulnerabilities. In *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*, pages 156–165. IEEE, 2000.

[14] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J.M. Wing. Automated generation and analysis of attack graphs. In *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pages 273–284. IEEE, 2002.

[15] R. Rieke. Tool based formal modelling, analysis and visualisation of enterprise network vulnerabilities utilising attack graph exploration. In *Eicar 2004 Conference CD-rom: Best Paper Proceedings, Copenhagen, EICAR eV*, 2004.

[16] K. Ingols, R. Lippmann, and K. Piwowarski. Practical attack graph generation for network defense. In *Computer Security Applications Conference, 2006. ACSAC'06. 22nd Annual*, pages 121–130. IEEE, 2006.

[17] Igor Kotenko and Andrey Chechulin. Common framework for attack modeling and security evaluation in SIEM systems. In *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*, pages 94–101. IEEE, 2012.

[18] C. Sarraute, J. Lucangeli, and G. Richarte. Attack planning in the real world. In *Workshop on Intelligent Security (SecArt 2010)*, 2010.

[19] M. Fox and D. Long. PDDL2.1: An extension to pddl for expressing temporal planning domains. *J. Artif. Intell. Res. (JAIR)*, 20:61–124, 2003.

[20] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: Theory & Practice*. Morgan Kaufmann, 2004.

[21] T.J. Li, G.Q. Chen, and G.F. Shao. Action control of soccer robots based on simulated human intelligence. *International Journal of Automation and Computing*, 7(1):55–63, 2010.

[22] D.S. Nau. Automated planning: Theory and practice. University Lecture, 2012.

[23] A.L. Blum and M.L. Furst. Fast planning through planning graph analysis. *Artificial intelligence*, 90(1):281–300, 1997.

[24] B. Bonet and H. Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1):5–33, 2001.

[25] *Artificial Intelligence: A Modern Approach, 2nd Edition.* Pearson, 2003.

[26] J. Hoffmann. FF: The fast-forward planning system. *AI magazine*, 22(3):57, 2001.

[27] D. Long, H. Kautz, B. Selman, B. Bonet, H. Geffner, J. Koehler, M. Brenner, J. Hoffmann, F. Rittinger, C.R. Anderson, et al. The AIPS-98 planning competition. *AI magazine*, 21(2):13, 2000.

[28] C. Aeronautiques, A. Howe, C. Knoblock, I.S.I.D. McDermott, A. Ram, M. Veloso, D. Weld, D.W. SRI, A. Barrett, D. Christianson, et al. PDDL— the planning domain definition language. 1998.

[29] E.P.D. Pednault. ADL: Exploring the middle ground between STRIPS and the situation calculus. In *Proceedings of the first international conference on Principles of knowledge representation and reasoning*, pages 324–332. Morgan Kaufmann Publishers Inc., 1989.

[30] D.E. Wilkins. *Practical planning: Extending the classical AI planning paradigm*. Morgan Kaufmann Publishers Inc., 1988.

[31] J.G. Carbonell, J. Blythe, O. Etzioni, Y. Gil, R. Joseph, D. Kahn, C. Knoblock, S. Minton, A. Pérez, S. Reilly, et al. Prodigy4. 0: The manual and tutorial. Technical report, DTIC Document, 1992.

[32] K. Erol, J. Hendler, and D.S. Nau. Umcp: A sound and complete procedure for hierarchical task-network planning. In *Proc. 2nd Intl. Conf. on AI Planning Systems*, pages 249–254, 1994.

[33] D. McDermott. A heuristic estimator for means-ends analysis in planning. In *Proceedings of the 3rd International Conference on Artificial Intelligence Planning Systems (AIPS-96)*, pages 142–149, 1996.

[34] A. Barrett, K. Golden, JS Penberthy, and D. Weld. UCPOP user's manual. *Computer Science and Engineering*, 1995.

[35] W.A.S. Ward and J.L. Kouns. OSVDB project aims. 2004.

[36] Lingfeng Ma, Salvador Mandujano, Guanfeng Song, L. Ma, G. Song, Pascal Meunier, L. Ma, G. Song, and P. Meunier. Sharing vulnerability information using a taxonomically-correct, web-based cooperative database, 2001.

[37] CVE MITRE. Common vulnerabilities and exposures, 2005.

[38] Robert Schuppenies. Automatic extraction of vulnerability information for attack graphs.

[39] P. Mell, K. Scarfone, and S. Romanosky. A complete guide to the common vulnerability scoring system (CVSS), version 2.0, forum of incident response and security teams, 2007.

[40] G.F. Lyon. Nmap network scanning: The official nmap project guide to network discovery and security scanning author: Gordon Fyodor L. 2009.

[41] R. Deraison et al. The Nessus Project. see `http://www.nessus.org`, 2002.

[42] Alfonso Gerevini and Derek Long. BNF description of PDDL3.0. *Unpublished manuscript from the IPC-5 website*, 2005.

[43] Nicklas Bornstein and Simon Svane. *Heuristics in Generic Planning using PDDL*. PhD thesis, Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark, 2011.

[44] Jussi Rintanen. State-space traversal techniques for planning, 2005.

[45] Paul Ammann, Duminda Wijesekera, and Saket Kaushik. Scalable, graph-based network vulnerability analysis. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 217–224. ACM, 2002.

[46] Steven Noel, Sushil Jajodia, Brian O'Berry, and Michael Jacobs. Efficient minimum-cost network hardening via exploit dependency graphs. In *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*, pages 86–95. IEEE, 2003.

[47] Nirnay Ghosh and SK Ghosh. An intelligent technique for generating minimal attack graph. In *First Workshop on Intelligent Security (Security and Artificial Intelligence)(SecArt09)*, 2009.

[48] Yannis Dimopoulos, Alfonso Gerevini, Patrik Haslum, and Alessandro Saetti. The benchmark domains of the deterministic part of IPC-5. *Abstract Booklet of the competing planners of ICAPS-06*, pages 14–19, 2006.

[49] Yixin Chen, Chih-Wei Hsu, and Benjamin W Wah. Sgplan: Subgoal partitioning and resolution in planning. *Edelkamp et al.(Edelkamp, Hoffmann, Littman, & Younes, 2004)*, 2004.

[50] Chih-Wei Hsu and Benjamin W Wah. The sgplan planning system in ipc-6. *Sixth International Planning Competition, Sydney, Australia (Sepember 2008)*, 2008.

[51] Thomas R Henderson, Mathieu Lacage, George F Riley, C Dowell, and JB Kopena. Network simulations with the ns-3 simulator. *SIGCOMM demonstration*, 2008.

[52] András Varga and Rudolf Hornig. An overview of the omnet++ simulation environment. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, page 60. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[53] Javin Paul. Difference between dom and sax parsers in java. Available at: `http://javarevisited.blogspot.com/2011/12/difference-between-dom-and-sax-parsers.html`.

[54] Scott White, Joshua OMadadhain, Danyel Fisher, and Yan-Biao Boey. Jung: Java universal network/graph framework. Available now at: `http://jung.sourceforge.net/index.html`, 2004.

[55] Miguel Procopio Castellar de Araujo, Amit Kumar Jha, Mohammed Nur, and Victor Ighalo. Cisco gigabit campus network design. *Group*, 2012.

[56] Leslie Daigle. Whois protocol specification. 2004.

Appendix A

Case Study 1 Network Product Installations

Cross-referencing the products installed on hosts in Case Study 1 with NVD produced a number of vulnerabilities, which could be leveraged by an attacker. These results are reflected in Appendix A, listing vulnerability information sourced from NVD entries with the corresponding CVE identifier. Most importantly, these entries denote the CVSS severity of a successfully exploited vulnerability, the type of access required to the targeted host, and the resultant privilege level gained from exploitation. This information is important for generation of critical attack paths, as well as the computation of solutions to these paths.

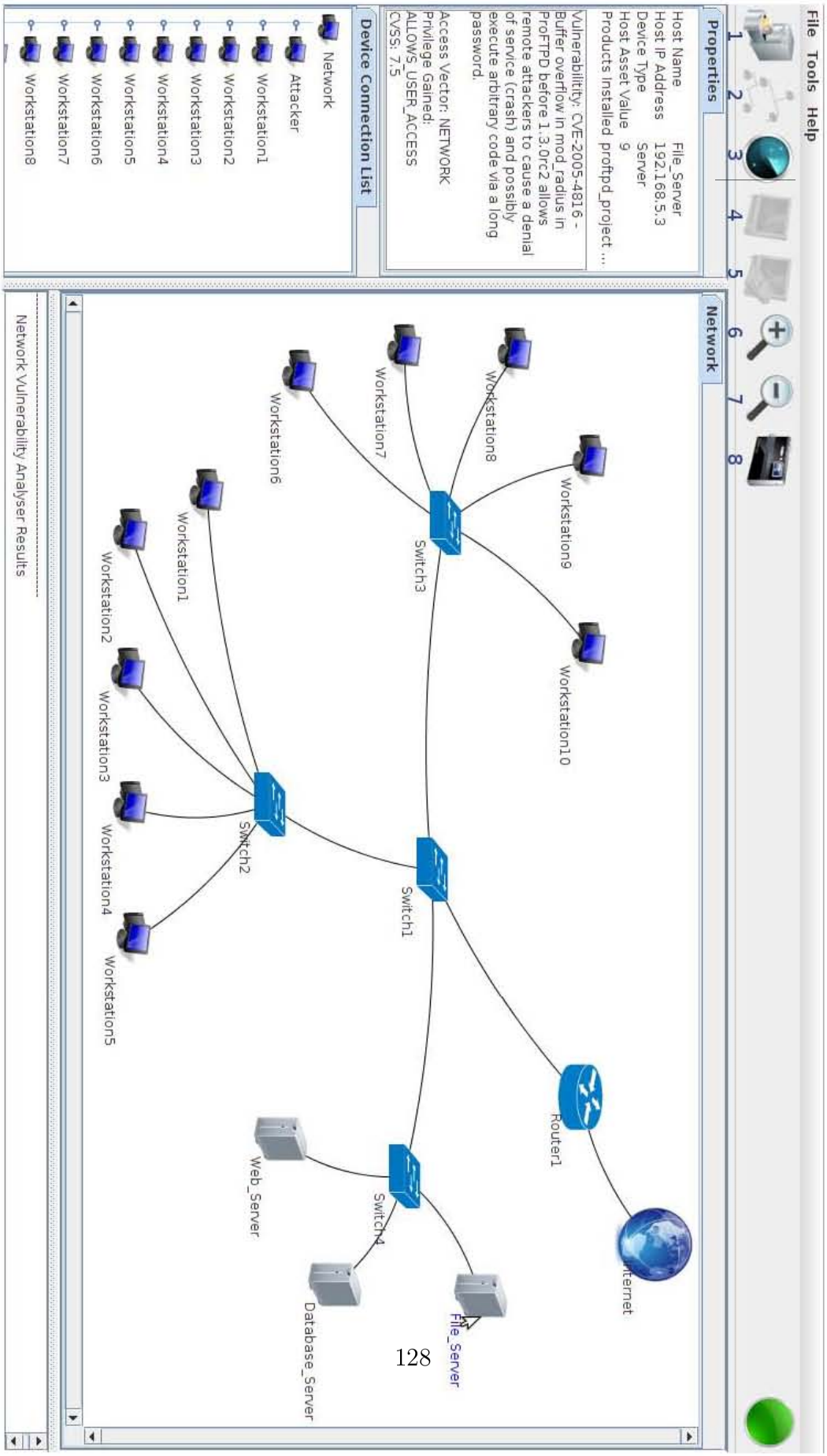
Device	**Installed Product**	**CVE ID**	**Access Vector**	**Privilege Gained**	**CVSS**	**Vulnerability Description**
Web Server	Microsoft IIS 7.5	CVE-2010-1256	Network	Admin	8.5	Unspecified vulnerability in Microsoft IIS 7.5, when Extended Protection for Authentication is enabled, allows remote authenticated users to execute arbitrary code via unknown vectors related to "token checking."
File Server	ProFTPD 1.2.pre1	CVE-2005-4816	Network	User	7.5	Buffer overflow in mod radius in ProFTPD before 1.3.0rc2 allows remote attackers to cause a denial of service (crash) and possibly execute arbitrary code via a long password.

	SGI Irix 6.5	CVE-2005-0113	Local	Admin	7.2	inpview in SGI IRIX allows local users to execute arbitrary commands via the SUN TTSESSION CMD environment variable, which is executed by inpview without dropping privileges.
Workstation3	Free Download Manager 2.5	CVE-2009-0184	Network	Admin	9.3	Multiple buffer overflows in the torrent parsing implementation in Free Download Manager (FDM) 2.5 allow remote attackers to execute arbitrary code via a long file name within a torrent file or a long tracker URL in a torrent file.
Workstation8	Icona Instant Messenger 1.0.0.1	CVE-2008-2551	Network	Admin	9.3	The DownloaderActiveX Control in Icona SpA C6 Messenger 1.0.0.1 allows remote attackers to force the download and execution of arbitrary files via a URL.

Appendix B

Visualization of Networks on the NVA GUI

The network in Case Study 1 was plotted in NVA, illustrating the network's topological layout, device properties and connectivity relations between devices. NVA provides the tools to determine an attacker's most likely path of attack, highlighting the weakest points in a network's security. Analysis of attack vectors support users in making decisions regarding the most practical ways to address threats.


File Tools Help
Properties
Host Name File_Server
Host IP Address 192.168.5.3
Device Type Server
Host Asset Value 9
Products Installed proftpd_project ...
Vulnerability: CVE-2005-4816 - Buffer overflow in mod_radius in ProFTPD before 1.3.0rc2 allows remote attackers to cause a denial of service (crash) and possibly execute arbitrary code via a long password.
Access Vector: NETWORK
Privilege Gained: ALLOWS_USER_ACCESS
CVSS: 7.5
Device Connection List
Network
Attacker
Workstation1
Workstation2
Workstation3
Workstation4
Workstation5
Workstation6
Workstation7
Workstation8
Network
Workstation8
Workstation7
Workstation6
Workstation9
Workstation10
Switch3
Workstation1
Workstation2
Workstation3
Workstation4
Workstation5
Switch2
Switch1
Router1
Web_Server
Switch4
Internet
File_Server
Database_Server
Network Vulnerability Analyser Results