Statistical Analysis of Computer Network Security

Goran Kap and Dana Ali

October 7, 2013

Abstract

In this thesis it is shown how to measure the annual loss expectancy of computer networks due to the risk of cyber attacks. With the development of metrics for measuring the exploitation difficulty of identified software vulnerabilities, it is possible to make a measurement of the annual loss expectancy for computer networks using Bayesian networks. To enable the computations, computer network vulnerability data in the form of vulnerability model descriptions, vulnerable data connectivity relations and intrusion detection system measurements are transformed into vector based numerical form. This data is then used to generate a probabilistic attack graph which is a Bayesian network of an attack graph. The probabilistic attack graph forms the basis for computing the annualized loss expectancy of a computer network. Further, it is shown how to compute an optimized order of vulnerability patching to mitigate the annual loss expectancy. An example of computation of the annual loss expectancy is provided for a small invented example network.

Acknowledgements

We would like to thank our supervisor Timo Koski at KTH for his valuable feedback and guidance on this thesis. We would also like to thank our friends and families for their support during the thesis.

Contents

Chapter 1

Introduction

Today information is increasingly being stored in electronic form on computers and computer networks. With the onset of the world wide web, computers that where once disconnected from each other are now globally connected through the world wide web. This has meant huge efficiency increases in information exchange and availability of information but has also meant that computer networks are more vulnerable to the security of information being attacked and compromised. At the same time, more and more functions and services in the society, some being very vital, both public and private, rely on computers and computer networks to store their information. For both of these reasons, the importance of information security in computer networks only grows.

Organizations of different kinds, such as government agencies, corporations and financial institutions amass a great deal of confidential information. Owners of information are concerned with protecting their information and information systems from unauthorized access, modification, use, disclosure, disruption and destruction. These goals can be formulated by the goal to protect the **confidentiality**, **integrity** and **availability** of information, also known as the CIA triad of information security. Confidentiality of information means that the information is secure from disclosure to unauthorized individuals. Integrity of information means that the information cannot be modified by unauthorized users. Availability of information means that the information is available when it is needed.

More formally, the protection of confidentiality, integrity and availability of information in information technology has been defined in [1] as:

Confidentiality - The security goal that generates the requirement for protection from intentional or accidental attempts to perform unauthorized data reads. Confidentiality covers data in storage, during processing, and while in transit.

Integrity - The security goal that generates the requirement for protection against either intentional or accidental attempts to violate data integrity (the

property that data has not been altered in an unauthorized manner) or system integrity (the quality that a system has when it performs its intended function in an unimpaired manner, free from unauthorized manipulation).

Availability - The security goal that generates the requirement for protection against intentional or accidental attempts to (1) perform unauthorized deletion of data or (2) otherwise cause a denial of service of data.

The risks posed by cyber attacks, both from attackers on the internet and from malicious insiders inside the internal network have been hard to measure and for this reason organizations have had difficulties quantifying these risks [3, p. 13]. The aim of this thesis is to investigate how one can quantify the risks of cyber attacks posed to computer networks and also quantitatively assess how modifications of the computer network can decrease these risks in an efficient manner.

1.1 How to measure the risk of cyber attacks on a computer network

First, we have to define how to measure the risk of successful attacks on a computer network. We use a measure that can be used directly in a cost benefit analysis since it says how much money it's worth to spend on IT security, the Annualized Loss Expectancy, ALE [7, p. 32]:

$$ALE = SLE * ARO \tag{1.1}$$

where SLE is the single loss expectancy and ARO is the annualized rate of occurrence. Generally speaking, the SLE is, as the expression implies, a measure of the expected monetary loss when a specific event occurs. The ARO is the estimated expected number of times for this event to occur in a given year. In our problem, the event is a certain kind of violation of the CIA of information of a host in a computer network.

Let the function ALE($h_{i,j}$) define the annual loss expectancy from a certain kind of violation j of the CIA of information on host h_i. Let the event $h_{i,j}$ be a random variable with a probability distribution that gives the probability that the event will happen a certain number of times in a given year. Then the ALE of the whole computer network is ALE(network):

$$ALE(network) = \sum_{i=1}^{n}\sum_{j=1}^{m} ALE(h_{i,j}) = \sum_{i=1}^{n}\sum_{j=1}^{m} SLE(h_{i,j}) * E(h_{i,j}) \tag{1.2}$$

For us, to estimate this value we must first model the vulnerability of computer networks from cyber attacks.

Chapter 2

Theory on how to model computer network vulnerability

2.1 Modeling the vulnerability of computer networks

Due to the complexity of software it is difficult to guarantee that they don't contain any flaws. Any flaw in the software that can be used to compromise the security of information by an intruder is a software vulnerability [9, p. 157]. In this thesis we will refer to software vulnerabilities as vulnerabilities. In computer security, the noun **exploit** is a piece of software, data, or sequence of commands that make use of a software vulnerability to compromise the information security of the host with the vulnerability. One particular meaning of the verb exploit, from the Oxford Dictionaries [14] is make use of (a situation) in a way considered unfair or underhand. In the context of computer security, an attacker exploits a vulnerability to gain illegal privilege level on the vulnerable host he is attacking. In computing, a **privilege level** defines the level of access to computer resources on a host by an individual. Examples of more limited privilege levels include the ability to view and edit files, modify system files, install and use programs or ability to read a user's credential. A **credential** is an attestation of authority to individuals, giving an individual the right to access certain computer resources on one or several hosts. The highest possible privilege level that an individual can have to a hosts computer's resources is given by the administrator privilege level on Windows hosts and root privilege level on UNIX hosts.

An attacker gaining illegal privilege on a host is called **privilege escalation** and can be measured in the degree of violation to the CIA of information [2]. When an attacker has enough privilege level to a host's computer resources, it can be used to attack other hosts that it is connected to. The **locality** of

a vulnerability defines from which hosts it can be exploited. Vulnerabilities that can be exploited from other hosts then the host where the vulnerability exists are said to be **remotely exploitable**, whereas vulnerabilities that can only be exploited from the host where the vulnerability exists are said to be **locally exploitable**.

Therefore, to model the privilege level that an attacker can gain on the hosts in a computer network, we need to know the data connectivity between hosts and the vulnerabilities present on them and their characteristics. A comprehensive network vulnerability model is made up of the following six components:

1. Hosts, a model of the hosts connected to the network and the set of vulnerabilities on each host
2. Vulnerabilities, a model of vulnerabilities
3. Vulnerable data connectivity relations, a model of the set of vulnerable data connectivity relations between hosts
4. Attackers, a model of the attackers that try to gain illegal privilege levels in the computer network
5. Attacks, a model of exploits that an attacker can use to attack vulnerabilities
6. IDS, a model of intrusion detection systems

2.1.1 Hosts

Hosts are identified by their network address, and consist of a list of vulnerabilities. Services are software bound to ports that enable the software to send and receive data to ports on other hosts [4, p. 23], [11, p. 61]. If the service has a vulnerability, the vulnerability can be accessed and exploited on that port from other hosts that have a data connectivity relation with that port [3, p. 21], [11, p. 61, pp. 80-83], we will later return with a definition of a vulnerable data connectivity relation in subsection 2.1.3. Remotely exploitable vulnerabilities are therefore associated with a port number, defining from which other hosts the vulnerability can be exploited. The network vulnerability model only consists of vulnerabilities that are known, it is not possible to model vulnerabilities whose existence are not known at the time of the network vulnerability analysis. As Daniel Bihar states in [4, p. 15], known vulnerabilities are the main entry point into computer networks, consequently it is not a meaningless task to model the vulnerability of computer networks because there are unknown vulnerabilities in computer networks. Attacks on unknown vulnerabilities are called zero day attacks because network defenders have had zero days to apply a patch on the vulnerability that is being attacked. In the context of cumputer network security, a **patch** is a piece of software that removes a vulnerability from that software.

Known vulnerabilities are identified by their CVE-identifier. These are issued by the Common Vulnerabilities and Exposures (CVE) list, a meta vulnerability database which is independent from individual vulnerability databases, maintained by the US government through the not-for-profit corporation MITRE that manages US government funded research and development centers. CVE was launched in 1999 and provides a common identifier for known vulnerabilities which can be used across various vulnerability databases. This common identifier facilitates easier searching and finding of information about common vulnerabilities from different vulnerability databases and also gives each known vulnerability a universally accepted name.

2.1.2 Vulnerabilities

The characteristics of a vulnerability are defined by the preconditions and postcondition of the vulnerability.

Vulnerability preconditions are: The locality of the vulnerability and the set of attack precondition privilege levels that enable the attack on the vulnerability from a host.

Vulnerability postcondition is: The attack postcondition privilege level obtained on the host with the vulnerability when the vulnerability has been successfully attacked.

In the generic vulnerability description model, a vulnerability description is defined as a set of 4-tuples:

Definition 2.1 (**Generic vulnerability model description**). *Let* $H = \{h_0, h_1, \ldots, h_n\}$ *be the set of vulnerable hosts in a network that can potentially be targeted by an attack in addition to the internet host* h_0. *Let* $V_{i,j}$ *be the vulnerability description of vulnerability* $vl_{i,j} \in VL$. *Let* $VL_i = \{vl_{i,1}, \ldots, vl_{i,m}\}$ *be the set of vulnerabilities on vulnerable host* $h_i \in H$ *and let* $VL = VL_1 \cup VL_2 \cup \ldots \cup VL_n$ *be the total set of vulnerabilities in the network on all hosts H. Let the set of vulnerability descriptions of all vulnerabilities* $vl_i \in VL_i$ *on host* $h_i \in H$ *be given by* $V_i = V_{i,1} \cup V_{i,2} \cup \ldots \cup V_{i,m}$ *and the total set of all vulnerability descriptions of all vulnerabilities in a network be given by* $V = V_1 \cup V_2 \cup \ldots \cup V_n$.

Let PT be the set of possible postcondition privilege levels that an attacker can obtain when successfully attacking a vulnerability $vl \in VL$ *on a host* $h \in H$ *and let* $L = \{local, remote\}$ *be the locality set, giving the locality of a vulnerability* $vl \in VL$. *Let PS be the set of privilege levels on a host* $h \in H$ *that enables an attacker to attack a vulnerability* $vl \in VL$ *from the host* $h \in H$.

A generic vulnerability description $V_{i,j}$ *of vulnerability* $vl_{i,j} \in VL$ *is modeled as the 4-ary Cartesian product over the following 4 sets:*

- *the host* $h_i \in H$ *with the vulnerability* $vl_{i,j} \in VL$, *given by the one-element set* $H_i = \{h_i \in H\}$
- *the locality* $l \in L$ *of the vulnerability* $vl_{i,j} \in VL$, *given by the one-element set* $L_{i,j} = \{l \in L\}$

- *the set of attack precondition privilege levels* $PS_{i,j} \subseteq PS$ *that enable an attacker to attack vulnerability* $vl_{i,j} \in VL$ *from a host* $h \in H$

- *the attack postcondition privilege level, given by the one-element set* $PT_{i,j} = \{pt \in PT\}$ *that an attacker obtains on host* $h_i \in H$ *when successfully exploiting vulnerability* $vl_{i,j} \in VL$

In the generic vulnerability model, a vulnerability description $V_{i,j}$ *of vulnerability* $vl_{i,j} \in VL$ *is modeled as a Cartesian product over the sets* $H_i,\ L_{i,j},\ PS_{i,j}$ *and* $PT_{i,j}$*, it is a set of 4-tuples:*

$V_{i,j} = H_i \times L_{i,j} \times PS_{i,j} \times PT_{i,j} = \{(h_i,\ l_{i,j},\ ps_{i,j},\ pt_{i,j}) : h_i \in H_i,\ l_{i,j} \in L_{i,j},\ ps_{i,j} \in PS_{i,j},\ pt_{i,j} \in PT_{i,j}\}$

meaning that:

- *the vulnerability* $vl_{i,j} \in VL$ *is located on host* $h_i \in H$

- *the vulnerability* $vl_{i,j} \in VL$ *has locality* $l_{i,j} \in L_{i,j}$ *where the one-element set* $L_{i,j} = \{l \in L\}$

- *the vulnerability* $vl_{i,j} \in VL$ *can be attacked with attack precondition privilege level* $ps_{i,j} \in PS_{i,j}$ *where* $PS_{i,j} \subseteq PS$ *from a host* $h \in H$

- *the vulnerability* $vl_{i,j} \in VL$ *gives an attacker attack postcondition privilege level* $pt_{i,j} \in PT_{i,j}$*, where the one-element set* $PT_{i,j} = \{pt \in PT\}$*, on host* $hi \in H$ *when exploited successfully*

2.1.3 Vulnerable data connectivity relations between hosts

To know which vulnerabilities on other hosts an attacker can attack when an attacker has obtained one of the attack precondition privilege levels $ps \in PS$ on a host $h_i \in H$ that enable an attack on other vulnerabilities, we need to know the set of vulnerable data connectivity relations C_i of host $h_i \in H$ to port numbers with vulnerabilities on other hosts $h_j \in H$, $h_i \in H \neq h_j \in H$. Data connectivity relations in general are not of interest to a computer network vulnerability model since they don't affect the vulnerability of the network to computer attacks, therefore we only need to model vulnerable data connectivity relations in the network vulnerability model. The set of vulnerable data connectivity relations from host $h_i \in H$ is given by a set of triples [11, p. 60]:

Definition 2.3 (**Set of vulnerable data connectivity relations from host hi** $\in$ **H**). *Let the set of vulnerable data connectivity relations from host* $h_i \in H$ *be denoted by* C_i *and the total set of vulnerable data connectivity relations in a network be given by* $C = C_1 \cup C_2 \cup \ldots \cup C_n$*. Let the host* $h_i \in H$ *be given by the one-element set* $H_i = \{h_i \in H\}$*.*

We define the set of vulnerable data connectivity relations C_i *of host* $h_i \in H$ *as a subset of the 3-ary Cartesian product over the 3 sets* $H_i,\ H \setminus H_i$ *and* $VL \setminus$

VL_i, it is a set of triples:

$$C_i \subseteq H_i \times H \setminus H_i \times VL \setminus VL_i = \{(h_i,\ h,\ vl) : h_i \in H_i,\ h \in H \setminus H_i,\ vl \in VL \setminus VL_i\}$$

meaning that host $h_i \in H_i$ has a vulnerable data connectivity relation to a port on host $h \in H \setminus H_i$, where a remotely exploitable vulnerability $vl \in VL \setminus VL_i$ can be targeted by an attack.

2.1.4 Attackers

The attackers are associated with a set of attacker states AS that they can potentially obtain. The set of attacker states AS of the attackers tells us the privilege level that the attackers can potentially obtain on each host in the network. The set of possible attacker states AS of the attackers in a computer network is given by a set of doubles.

Definition 2.4 (**Set of possible attacker states**). *We define the set of possible attacker states AS as a subset of the two-ary Cartesian product over the two sets H and PT, it is a set of doubles:*

$$AS \subseteq H \times PT = \{(h, pt) : h \in H,\ pt \in PT\}$$

meaning that an attacker has obtained the attack postcondition privilege level $pt \in PT$ on host $h \in H$.

2.1.5 Attacks

An attack is an event that is triggered when certain conditions in its environment are met, the preconditions of the event. The event has a certain effect on its environment, the postconditions of the event. Our network vulnerability model contains the attributes of a computer network that are relevant for modelling computer attacks. Attacks are defined by the preconditions and postcondition of the vulnerability $vl_{f,g} \in VL$ that is being attacked. Thus if an attack $a_i \in A_i$ targets target vulnerability $vt_i \in VT_i$ on host $h_j \in H$, the attack precondition privilege level $ps \in PS$ of attack $ai \in Ai$ is the same as the attack precondition privilege level $ps \in PS$ of target vulnerability $vti \in VTi$. In the same way, the attack postcondition privilege level $pt \in PT$ of the attack $a_i \in A_i$ on target vulnerability $vt_i \in VT_i$ is the same as the attack postcondition privilege level $pt \in PT$ of target vulnerability $vt_i \in VT_i$.

The preconditions and postcondition of an attack are:

Attack preconditions are: Attacker state $(h_i, ps) \in AS$ from where the attack is launched, vulnerable connectivity relation $c_i \in C_i$ if the attack is launched against a remote vulnerability and existence of vulnerability $vl_{f,g} \in VL$ on attacked host $h_f \in H$.

Attack postcondition is: Attacker state $(h_f, pt) \in AS$ gained on the attacked host $h_f \in H$ when the vulnerability $vl_{f,g} \in VL$ has been successfully attacked

[11, p. 70].

We define the set of possible attacks A_i from host $h_i \in H$ that can be launched by an attacker by obtaining attack precondition privilege level $ps \in PS$ on host $h_i \in H$ as follows:

Definition 2.4 (**Set of possible attacks from host $\mathbf{h_i} \in \mathbf{H}$**). *Let A_i be the set of possible attacks from host $h_i \in H$ and let $A = A_1 \cup A_2 \cup \ldots \cup A_n$ be the set of possible attacks in the whole network from all hosts H.*

We define the set of possible attacks A_i launched from host $h_i \in H$ as a subset of the 5-ary Cartesian product over the following 5 sets:

- *the host $hi \in H$ from where the attack is launched, given by the one-element set $H_i = \{h_i \in H\}$*
- *the set of attack precondition privilege levels PS on the host $h_i \in H$ that enable an attacker to attack a vulnerability $vl \in VL$ from the host $h_i \in H$*
- *the set of target hosts $HT_i \subseteq H$ with a target vulnerability $vt_i \in VT_i$ that an attacker can attack from host $h_i \in H$*
- *the set of target vulnerabilities $VT_i \subseteq VL$ that an attacker can attack from host $h_i \in H$*
- *the set of attack postcondition privilege levels PT on the target host $ht_i \in HT_i$ that an attacker obtains when successfully attacking target vulnerability $vt_i \in VT_i$ on target host $ht_i \in HT_i$*

We define the set of possible attacks A_i from source host $h_i \in H$ as a subset of the 5-ary Cartesian product over the 5 sets, H_i, PS, HT_i, VT_i and PT, it is a set of 5-tuples:

$A_i \subseteq H_i \times PS \times HT_i \times VT_i \times PT = \{(h_i,\ ps,\ ht_i,\ vt_i,\ pt) :\ h_i \in H_i,\ ps \in PS,\ ht_i \in HT_i,\ vt_i \in VT_i,\ pt \in PT\}$

meaning that:

- *an attack can be launched from host $h_i \in H$ where an attacker has attack precondition privilege level $ps \in PS$ on target vulnerability $vt_i \in VT_i$ on target host $ht_i \in HT_i$*
- *the attack gives an attacker attack postcondition privilege level $pt \in PT$ on target host $ht_i \in HT_i$ when target vulnerability $vt_i \in VT_i$ is successfully attacked*

Like vulnerabilities, attacks can be defined by their locality, defining if the attack $a_i \in A_i$ is launched against a vulnerability $vl_{i,j} \in VL$ on the same host $h_i \in H$ or against a vulnerability $vl_{f,g} \in VL$ on another host $h_f \in H$, $h_f \in H \neq h_i \in H$.

The attack postcondition privilege level $pt \in PT$ of an attack $a_i \in A_i$ on a host $h_i \in H$ can be the attack precondition privilege level $ps \in PS$ of another attack $a_j \in A_j$ from host $h_j \in H$. By knowing the characteristics of vulnerabilities, the preconditions required to exploit them, and the postcondition of exploiting them, it becomes possible to chain possible attacks $a \in A$ together into a sequence of attacks that achieve a certain goal attacker state $as \in AS$. This is the information that attack graphs represent.

2.1.6 Intrusion detection systems

An intrusion detection system (IDS) is a device that detects attacks $a_i \in A_i$ in computer networks. The IDS produces alarms to the person monitoring the security of the computer network, a security administrator when it makes an estimate that an attack is ongoing and records the estimated number of attempted attacks $ai \in Ai$ from host $hi \in H$ on a vulnerability $vl_{j,k} \in VL$ on host $h_j \in H$ [11, p. 61].

IDS devices suffer from two flaws, one is that they don't detect all attacks that they are supposed to detect. A **false negative** is a failure of the IDS to produce an alarm when an actual attack has taken place. The other flaw is that an IDS sometimes produces an alarm when no real attack has taken place, this is called a **false positive**. To estimate the true number of attempted attacks $a_i \in A_i$ from host $h_i \in H$ on a vulnerability $vl_{j,k} \in VL$ on host $h_j \in H$ one has to take into account both the number of false negatives that puts the IDS estimate downward from the true number and the number of false positives that puts the IDS estimate upward of the true number. The developers of IDS devices are of course aware of both these flaws and we assume that they develop IDS devices that come as close as possible to the true count so that these IDS devices make the best estimates available.

We define the estimated historical average number of attacks in a given time interval measured by an intrusion detection system of an attack $a_i \in A_i$ from host $h_i \in H$ on a vulnerability $vl_{j,k} \in VL$ on host $h_j \in H$ as a function named IDSf:

Definition 2.6 (**Intrusion detection system values set IDSS**). *The function $IDSf(x1, x2, y)$ where $x1 \in \mathbb{N}$, $x2 \in \mathbb{N}$, $y \in \mathbb{N}$ and $\mathbb{N}$ is the set of natural numbers, gives the best estimated historical average number of attempted attacks from host $h_{x1} \in H$ on vulnerability $vl_{x2,y} \in VL$ on host $h_{x2} \in H$ in a given time interval. The estimate is made by an intrusion detection system. The set of values of the function IDSf(x1, x2, y) that exist in a network is given by the set IDSS.*

2.2 Attack graphs

An attack graph is a directed graph that represents the dependency of attacks that lead to a goal attacker state $as \in AS$ [20, p. 3], [10, p. 10]. A directed

graph is defined as [25, p. 41, p. 45]:

Definition 2.6 (**Directed graph**). *A directed graph $G = (V, D)$ consists of a finite set of nodes V and a directed edge set D. For any two distinct nodes $\alpha \in V$ and $\beta \in V$, the ordered pair $(\alpha, \beta) \in D$ if and only if there is a directed edge from α to β. The edge set D therefore consists of ordered pairs of nodes. Let $V = \alpha_1, \ldots, \alpha_d$. The directed graph does not contain any directed edges of the form (α_j, α_j) (that is a loop from the node to itself) and any edge $(\alpha_j, \alpha_k) \in D$ appears exactly once. That is, multiple edges are not permitted.*

Based on the definition of a directed graph a general definition of an attack graph can be given [16, p. 101]:

Definition 2.7 (Attack Graph). *Given a set of attacks A and a set of conditions C and two types of edge sets $R \subseteq C \times A$ and $S \subseteq A \times C$, an attack graph G is a directed graph $G(A \cup C,\ R \cup S)$ ($A \cup C$ is the set of nodes and $R \cup S$ is the set of edges in the directed graph G). Condition nodes represent either precondition or postcondition attacker states as $\in AS$, the whole set of attack $a \in A$ preconditions, or a single attack $a \in A$ precondition in the set of attack $a \in A$ preconditions.*

The goal attacker state $as \in AS$ can for example be administrative access on a particular host or access to a database. An attack graph can also show all possible attacker states AS in the network, showing all possible attack postcondition privilege levels $pt \in PT$ of an attacker on all hosts in the network by an attacker from outside the network [3, p. 2]. Xinming Ou stated in [10] that there are two basic approaches of representing attack graphs, the network state and the exploit dependency attack graph [10, p. 11]. Since then a third has been proposed and developed by various researchers, a kind of attack graph that we call the probabilistic attack graph.

2.3 Network state attack graphs

2.3.1 Modeling network vulnerability using model checkers

The initial approach in the research community to model the vulnerability of computer networks was using model checking techniques. This method was pioneered by Ritchey and Ammann in [9] [10, p. 7]. A model checker is a tool that assists engineers to automatically and exhaustively identify individual design flaws in a model of a system. Using off-the-shelf model checkers, the researchers could avoid custom building special purpose tools for attack graph generation [10, p. 15]. To check if the system has a flaw, the model is checked whether it meets a correctness specification. The model is a state machine defined by variables, initial values for the variables and a description of the

conditions under which variables may change value. When the variables change value they cause a state transition. The sum of all possible states of a state machine is the state space.

The model can be automatically checked by a model checker against a correctness specification if the model has any flaws. The correctness specifications are expressed in propositional temporal logic. The model checker performs an exhaustive search through the state space to determine that each state satisfies the correctness specification. If the correctness specification is not satisfied, the model checker will give a counterexample execution, showing the sequence of states that lead to the violation of the correctness specification [11, p. 3, p. 47], [9, p. 158], [12, p. 256].

When modeling computer network vulnerability as a state machine, the model typically looks similar to our model in section 2.2 [9, pp. 159-160], [11, pp. 59-62]. The main action that triggers other variables to change value, causing a state transition, is an attacker launching new attacks $a \in A$. The main variable that changes value after an attack is the set of attacker states AS of an attacker, when a new attacker state $as \in AS$ is added to an attacker's set of attacker states AS. The initial values for the variables is the initial attacker state when the attack starts. For an external attack from the internet, the privilege level will be set to none on all hosts in the network. Modeling attack's by an employee or other trusted individuals, starting privilege levels should be higher on some hosts reflecting the person's user privilege on those hosts [9, p. 160].

When applying model checking to a model of network security, the counterexample execution is an attack path to a goal attacker state. The model needs a failure definition defining which state constitutes a violation of the correctness specification. The correctness property could be that an external attacker can never get access to the file server, for example. When the model checker visits a state where this property is false, the model checker outputs the sequence of states from the initial state leading to the state that violates that property. This represents an attack scenario for the network. Eventually a state will be reached where the correctness property is false or will continue until no more exploits can be employed, showing that there are no attack scenarios that violate the given security property [9, p. 161].

2.3.2 How network state attack graphs are generated

In his PhD thesis on scenario graphs and attack graphs [11], Sheyner uses an adapted model checking technique, a formalism known as a *failure scenario graph* to model the vulnerability of computer networks. Unlike regular model checker's that give one failure scenario at a time that violate a given correctness specification in the model, failure scenario graphs show all sequences of states that lead to a violation of the correctness specification. A particular kind of failure scenario graph is the network state attack graph. The network state attack graph shows all possible attack paths to a particular goal attacker state. This gives the user the ability to prioritize the problem fixing as appropriate. The first formal treatment of attack graphs in [12] used this approach to define

an attack graph [13, p. 217]. In the network state attack graph, the nodes are the states of the network and the edges are the state transitions between these states. In each state, an attacker's set of attacker states is included as is information on all other components of the network vulnerability model. Thus, the privilege level of an attacker on all hosts in the network is represented in each node.

2.3.3 Scalability and performance of network state attack graphs

The network state attack graph suffers from serious scalability problems. For example, Sheyner et al.'s work in [12] describe a network state attack graph of a network with 5 hosts and 8 types of vulnerabilities. The modified model checker NuSMV took 2 hours to generate the attack graph for this network. The resulting attack graph had 5948 nodes and 68364 edges. The problem is common for model checker's and is commonly known as the state explosion problem. The state explosion problem arises from the combinatorial blow up of the state space, causing the number of possible states to be exponential to the number of variables in the model [13, p. 218]. Thus, the number of nodes in the network state attack graph scales exponentially to the number of vulnerabilities in the network and there is good reason to doubt whether model checker's will ever be able to scale the network vulnerability analysis to networks of even modest size [13, p. 223]. In [11, p. 64] Sheyner developed an algorithm for his network state attack graph generator tool that finds a minimal set of defensive measures that will completely disconnect the initial and final states of the attack graph. A defensive measure is a measure that renders an attack ineffective. This can mean removing connectivity relations by adding firewall rules, patching vulnerabilities or changing privilege levels for users. The algorithm will output a set of possible attacks that, if removed by defensive measures, will make it impossible for an attacker to reach his goal attacker state.

2.4 Exploit dependency attack graphs

2.4.1 The monotonicity assumption

Having observed the scalability issues with the model checking approach to generate network state attack graphs, Ammann et al. were first to propose a more efficient representation of the attack graph [10, p. 9], the exploit dependency attack graph in [13]. By making the simple assumption of monotonicity, which states that the preconditions of a given attack are never invalidated by another attack, the authors were able to model the attack graph as a directed graph where the nodes represent a single attacker state $as \in AS$ instead of the whole set of obtained attacker states AS of an attacker as in the network state attack graph. In the network state attack graph, each order in which the attacker carries out the exploitations to his goal attacker state is explicitly shown in the

attack graph, which results in an exponential number of redundant attack paths that differ only in the order of the attack steps. In the exploit dependency attack graph on the other hand, once an attacker gains a certain attacker state, the fact can remain true for the remainder of the vulnerability analysis process [10, p. 9]. Since the nodes in the exploit dependency attack graph only represent a single attacker state the representation of the attack graph becomes more efficient. The result is that the number of nodes in the exploit dependency attack graph scales linearly to the number of vulnerabilities in the network instead of exponentially as in the network state attack graph [13, p. 223].

2.4.2 Different types of exploit dependency attack graphs

The exploit dependency attack graph shows the relationship between attacks $a \in A$ and attacker states $as \in AS$. The original exploit dependency attack graph in [13] is a directed graph where the edges are attacks $a \in A$ and the nodes are attacker states $as \in AS$ enabling new attacks $a \in A$. Attacker state nodes are called condition nodes, since an attacker state can be both a precondition and a postcondition of an attack. Since the number of nodes in the exploit dependency attack graph scales linearly to the number of vulnerabilities in the network [10, p. 11], it scales much better then the network state attack graph. Thanks to this crucial property of exploit dependency attack graphs, the research community has largely adopted this approach ever since it was proposed in [13]. Currently, there are three software systems that have been developed by researchers at universities that manage to generate exploit dependency attack graphs for large networks, Topological Vulnerability Analysis (TVA) [15], Multihost, multistage Vulnerability AnaLysis (MulVAL) [10] and Network Security Planning Architecture (NetSPA) [3]. Exploit dependency attack graphs show the dependency between attacks $a \in A$ and possible attacker states $as \in AS$. This can be done in many different ways however. We will present three types of exploit dependency attack graphs used by the three main attack graph generation systems. In all three types, edges are contentless and are only used to connect nodes of different types in a directed graph.

2.4.3 TVA attack graphs

The attack graph model used in Topological Vulnerability Analysis (TVA) uses two types of nodes, exploit nodes and security condition nodes [16]. Exploit nodes represent attacks $a \in A$ and security condition nodes represent the attack postcondition attacker state $as \in AS$ of an attack or a single precondition in the set of preconditions of an attack. The nodes look different depending on type, exploit nodes are ovals and security condition nodes are clear text [16, p. 4]. Security condition nodes are the elements in the set of preconditions to an attack $a \in A$, they are attacker states $as \in AS$, the existence of a vulnerability $vl_{f,g} \in VL$ on attacked host $h_f \in H$ or vulnerable data connectivity relation $c_i \in C_i$ on host $hi \in H$ from where the attack is launched. Since sets of security condition nodes imply possible attacks and attack nodes imply the attack postcondition

attacker state $as \in AS$ of the attack, no edge goes directly between two security condition nodes or between two exploit nodes, directed edges only inter-connect security condition nodes and exploit nodes [16, p. 4]. Exploit nodes have only one outbound edge pointing to a single security condition node which is the attack postcondition attacker state $as \in AS$ of the attack. Similarly, security condition nodes that are the set of preconditions to an attack point to exploit nodes that they are the preconditions to. An example of a TVA attack graph is depicted in figure 2.1 which has been taken from [16, p. 112].

2.4.4 MulVAL attack graphs

The MulVAL attack graph has three types of nodes, attack-step nodes represented as oval shaped nodes, privilege nodes represented as diamond shaped nodes and configuration nodes which are represented as rectangles [18, p. 2]. Attack step nodes represent attacks $a \in A$ and privilege nodes represent attacker states $as \in AS$. Configuration nodes represent connectivity relations $ci \in Ci$, vulnerability descriptions $V_{f,g}$ of vulnerability $vl_{f,g} \in VL$ on attacked host $h_f \in H$ and descriptions of certain services on attacked host $h_f \in H$ for example. Configuration nodes are network configuration objects that are a part of the preconditions of an attack $a \in A$ and are known to exist in the network regardless of an attacker and that enable the attack-step nodes. Attack-step nodes and privilege nodes on the other hand represent possibilities that something can happen, i.e. that an attack $a \in A$ can happen or an attacker state $as \in AS$ can be obtained by an attacker. Privilege nodes point to attack-step nodes, attack-step nodes point to configuration nodes and privilege nodes and configuration nodes have no outbound edges. This is counter-intuitive, since configuration nodes imply attack-step nodes and thus should point to these, privilege nodes imply attack-step nodes and thus should point to these and configuration nodes don't depend on the possibility that something must happen and thus should not have inbound edges. An example of a MulVAL attack graph is depicted in figure 2.2 which has been taken from [22, p. 10].

2.4.5 NetSPA multi-prerequisite attack graphs

In [5], Lippmann et. al developed the multi-prerequisite graph for their attack graph generation system NetSPA that scales nearly linearly with the number of hosts in the network. The multi-prerequisite graph also has three types of nodes, state nodes, represented as circles, prerequisite nodes, represented as rectangles and vulnerability instance nodes, represented as triangles. In the multiple-prerequisite attack graph, state nodes represent attacker states $as \in AS$. Prerequisite nodes represent the set of preconditions of one or several attacks $a \in A$. Vulnerability instance nodes represent attacks $a \in A$. A prerequisite node can contain several attacker states $as \in AS$ if they imply the same set of attacks $a \in A$. Thus, several state nodes can point to the same prerequisite nodes. Prerequisite nodes point to vulnerability instance nodes that represent the set of attacks $a \in A$ that the prerequisite node enables. In this way, by intro-

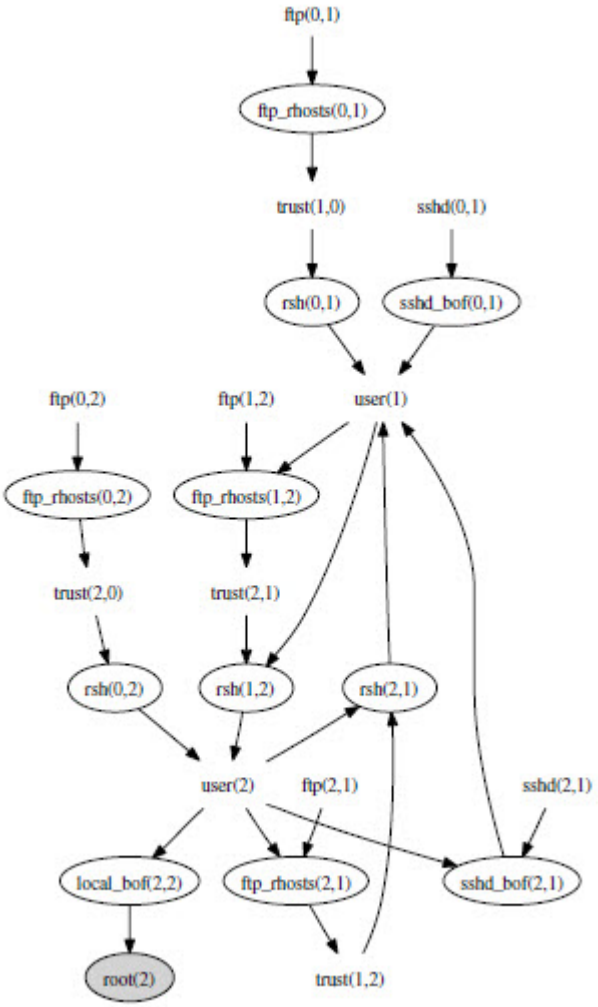


Figure 2.1: A TVA attack graph

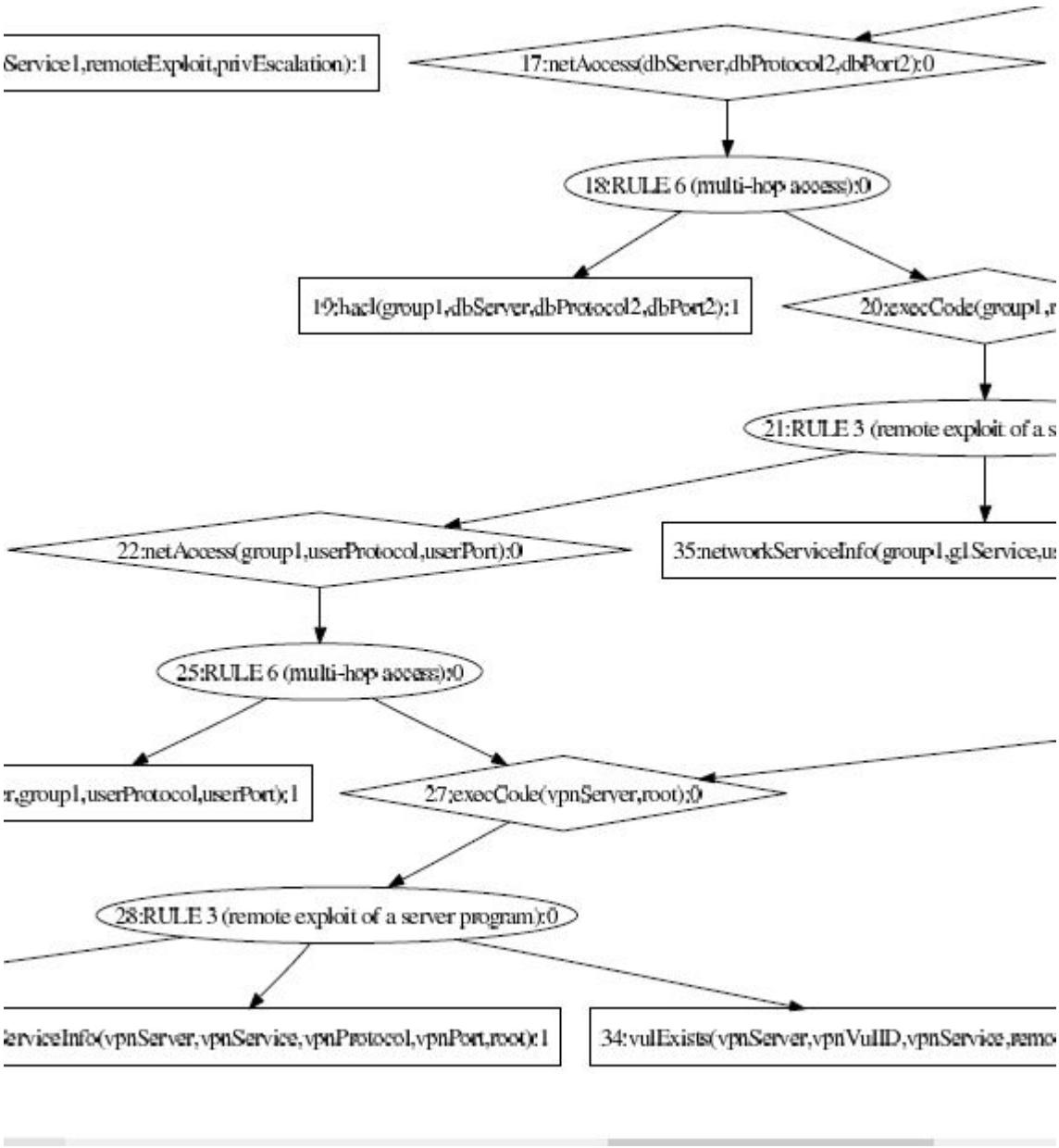


Figure 2.2: Part of a MulVAL attack graph

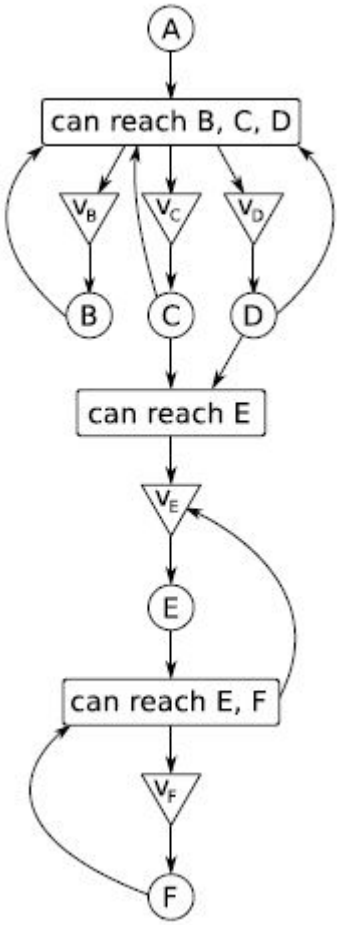


Figure 2.3: A NetSPA Multiple prerequisite attack graph

ducing prerequisite nodes, the number of edges is reduced compared to having state nodes pointing directly to vulnerability instance nodes, since many state nodes can imply the same set of attacks. Finally, vulnerability instance nodes point to a single state node, the attacker state $as \in AS$ obtained by launching the attack represented by the vulnerability instance node. An example of a multiple-prerequisite graph is depicted in figure 2.3 which has been taken from [5, p. 124].

2.4.6 Scalability and performance of exploit dependency attack graphs

Scalability depends on the software system used, computation power of the computer used by the tool and the complexity of the network analyzed. Computer power is largely dependent on the time when the test was made, according to Moore's law. The systems are tested on large simulated network environments for two main reasons [3, p. 45]. First, real data are sensitive and reveal network weaknesses that are valuable to an attacker. Therefore, as Lippmann et al. state in [3, p. 45]: "As noted above, most system administrators require us to perform analyses of real networks on site in a physically protected area and preferably on a computer not connected to any network. They also do not permit release of attack graphs for real networks. None of these restrictions apply to simulated networks." This means that the acquiring of data is difficult and that attack graph generation results is difficult to present to anyone outside the organization. Second, simulation studies allow evaluation of attack graph generation system performance, how much time the system requires to generate the attack graph and make network security enhancing recommendations, depending on the size of the network.

In a test of the TVA attack graph generation system, tests are conducted with relation to the number of hosts in the network [15, p. 153]. Hosts are grouped into subnets, a subnet contains of 200 hosts, and each simulated host has the same set of 5 vulnerabilities, and each vulnerability $vl_{f,g} \in VL$ can be attacked remotely from hosts $h_i \in H$ with vulnerable connectivity relation $(h_i, h_f, vl_{f,g})$. Each subnet has incoming vulnerable connectivity relations from two other subnets, and symmetrically, outgoing vulnerable connectivity relations to two other subnets. From one subnet to another, there are 500 vulnerable connectivity relations to vulnerabilities in the other subnet. Thus there are 2*500 = 1000 incoming and 2 * 500 = 1000 outgoing vulnerable connectivity relations. The number of subnets is increased to test the scalability of the attack graph generation system. With this type of network complexity, computation time grows linearly to the number of hosts. For 40 000 hosts in this type of network complexity, the TVA attack graph generation system takes 20 s to generate the attack graph. TVA can automatically compute the minimum set of attacks that separates starting attacker state $as \in AS$ from goal attacker state $as \in AS$ of an attacker and give optimal network security enhancing recommendations [15, p. 151].

For a simulated network with a complex network environment, MulVAL generated the attack graph for a simulated network of 1000 hosts in around 1000 s in [17, p. 343]. The computation time is shown to scale between $O(n^2)$ and $O(n^3)$ to the number of hosts n for a complex network environment [17, p. 343].

In simulated network environments in NetSPA, hosts are grouped into subnets and tenants. A tenant is a subnet with a firewall between its hosts and the rest of the network. A firewall is a device that restricts data connectivity relations between hosts. All hosts in subnet and tenant groups are configured

alike, meaning that all hosts $h_i \in H$ in a tenant or subnet group have the same vulnerabilities $vl_{i,j} \in VL$ and connectivity relations $c_i \in C$ to other hosts. The number of hosts and the number of vulnerabilities per host can be specified separately for each tenant and subnet [3, p. 48].

For a simulated network with a complex network environment, NetSPA generated the attack graph for a simulated network of 10000 hosts in around 3 h in [19, p. 989]. The computation time is shown to grow less then quadratically for both the simple and complex network type.

By defining the host asset value to each host in the network, reflecting the monetary value of the information resources on the host to the owners, the NetSPA system is able to automatically compute the percentage of host asset values that can be compromised, the network compromised percentage (NCP):

$$Network\ Compromised\ Percentage = \\ 100 * \frac{\sum_{Compromised\ hosts} Host\ asset\ value}{\sum_{All\ hosts} Host\ asset\ value} \tag{2.1}$$

A host is considered compromised if an attacker has gained either user or administrator privilege level on the host or is able to cause a denial of service attack, making the hosts computer resources unavailable to its intended users [3, p. 14]. NetSPA can also make a prioritized list of recommended changes to the network based on what vulnerability removals cause the greatest reduction in NCP [3, pp. 41-42]. The performance of these systems are marked improvements to the tools that generate network state attack graphs that take much more time to analyze networks since the number of nodes in network state attack graphs grow exponentially to the number of vulnerabilities in the network.

2.5 Probabilistic attack graphs

The exploit dependency attack graph shows us the set of possible attacker states $as \in AS$ and possible attack paths into a computer network. It gives us a qualitative view of a networks vulnerability, it says that something is possible, an attacker has a possibility to obtain an attacker state $as \in AS$ in the attack graph [8, p. 284]. In reality however, some attacker states $as \in AS$ are easier to reach then others [18, p. 1]. To answer the question of how big the possibility of an attacker to reach a certain attacker state $as \in AS$ is, quantitative measures of network security are necessary instead of the qualitative, either secure or insecure view of the regular exploit dependency attack graphs [20, p. 284]. To quantify the risk of attacks to computer networks, various researchers have proposed what we call the probabilistic attack graph that uses the exploit dependency attack graph to model the probability distribution for each attacker state in the attack graph [20], [21], [22]. Modeling these probability distributions through the probabilistic attack graph will enable us to obtain an estimation of the ALE(network) which was the goal of this thesis. To model the probabilistic attack graph, any cycles in the directed graph that represents

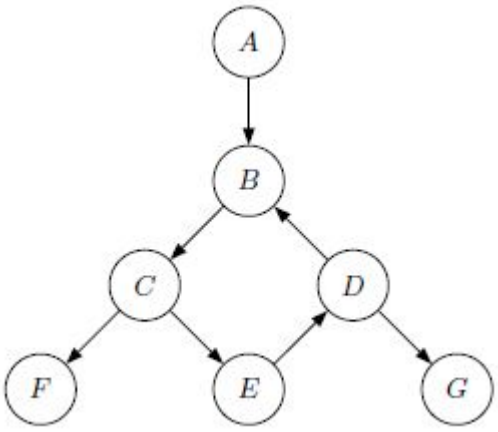


Figure 2.4: A directed graph with a cycle. This is not allowed in Bayesian networks.

the exploit dependency attack graph have to be removed, thereby creating an acyclic directed graph. To define a cycle in a directed graph, we first have to define what a directed path is in a directed graph [25, p. 44]:

Definition 2.8 (**Directed path**). *Let $G = (V, D)$ be a directed graph. A path of length m from a node α to a node β is a sequence of distinct nodes $(\tau_0, \tau_1, \ldots, \tau_m)$ such that $\tau_0 = \alpha$ and $\tau_m = \beta$ such that $(\tau_{i-1}, \tau_i) \in D$ for each $i = 1, \ldots, m$.*

A cycle is defined as:

Definition 2.9 (**Cycle**). *Let $G = (V, D)$ be a directed graph. An m-cycle in G is a sequence of distinct nodes*

$$\tau_0, \ldots, \tau_{m-1}$$

such that $\tau_0, \ldots, \tau_{m-1}, \tau_0$ is a path (Definition 2.9). [25, p. 44]

An example of a cycle in a directed graph is given in figure 2.4.

Based on these definitions, a definition of a directed acyclic graph can be given:

Definition 2.10 (**Directed acyclic graph** (**DAG**)). *A graph $G(V, D)$ is said to be a directed acyclic graph if G is a directed graph and there are no m-cycles in G for any $m \geq 1$.*

Further, all nodes in the resulting acyclic directed graph have to be transformed into random variables with probability distributions or conditional probability

distributions. The directed edges in the directed acyclic graph indicate the influence between variables, how probability distributions of variables are conditioned on probability distributions of other variables, indicated by inbound edges from the influencing variables. The conditional probability distribution of a variable gives the probability that a variable will be in a certain state conditioned on the state of other variables. The state gives the number of times an attacker obtains an attacker state $as \in AS$ in a given period of time. Thus, the probabilistic attack graph can be defined as an exploit dependency attack graph where the nodes are variables with discrete probability distributions or discrete conditional probability distributions and where any cycles have been removed, thereby creating an acyclic directed graph.

Acyclic directed graphs with nodes that are random variables with probability distributions or conditional probability distributions have formally been described as Bayesian networks. Therefore, the probabilistic attack graph is a type of Bayesian network, a Bayesian network of an attack graph. We will return with a more formal definition of Bayesian networks in chapter 3.

By modeling the attack graph as a Bayesian network, the expected number of times an attacker state $as \in AS$ will be reached in a given period of time can be computed for any attacker state $as \in AS$. To enable the modelling of attack graphs as a Bayesian network, each attack random variable in the probabilistic attack graph depends on the estimated probability that the vulnerability will be attacked successfully given that all preconditions are satisfied. This has been enabled by the introduction of standardized vulnerability metrics in vulnerability databases that measure exploit difficulty of individual vulnerabilities. One such standard to measure the exploit difficulty and severity impact of vulnerabilities on the information security of the vulnerable host is given by the Common Vulnerability Scoring System.

2.5.1 The Common Vulnerability Scoring System

The Common Vulnerability Scoring System (CVSS) vulnerability metrics standard has been developed by FIRST, the Forum of Incident Response and Security Teams to measure vulnerability severity impact and exploit difficulty and was introduced in 2007 [2]. FIRST is an organization that brings together the government, commercial and academic sectors to improve the computer network security globally. CVSS has been adopted by a number of vulnerability database providers, each giving their different subjective scores of the different vulnerability metrics in CVSS, this means that different vulnerability database providers can possibly provide different CVSS scores for the same vulnerability [28, p. 56]. CVSS consists of the base score, the temporal score and the environmental score. The most commonly provided CVSS vulnerability score is the CVSS base score which consists of exploitability metrics and impact metrics. In the exploitability metrics there are three metrics for Access Vector (AV), Attack Complexity (AC) and Authentication (Au). The exploitability metrics measure characteristics of the vulnerability that affect the difficulty of exploitation of the vulnerability. The Access Vector defines the locality of the vulnerability

and has three values, local, adjacent network and network, which is the same as remote locality in the terminology of this thesis. Access Complexity measures the complexity of exploiting the vulnerability and has three levels, low, medium and high. Authentication defines the number of required authentication steps required to exploit the vulnerability and has three values, none, single instance and multiple instances. The impact metric measures the severity of compromise on each goal of information security, confidentiality, integrity and availability of information. All three metrics, Confidentiality Impact (C), Integrity Impact (I) and Availability Impact (A) have three levels, none, partial and complete impact. Details of what the different levels of vulnerability metrics in the CVSS vulnerability metric standard mean and what they measure are given in [2].

2.5.2 How to measure the exploit difficulty of vulnerabilities

To model the probability of exploitation of a given vulnerability given that all the vulnerability preconditions are satisfied, the difficulty for an attacker of exploiting a vulnerability must be measured. CVSS has two vulnerability metrics that impact this difficulty, *Access complexity (Ac)* from the base score and *Exploitability (E)* from the temporal score [27, p. 213]. As stated in section 2.6.1, Access Complexity measures the complexity of exploiting the vulnerability and has three levels, low, medium and high. The Exploitability metric describes the availability of exploit code and how functional it is. There are four levels of Exploitability, from lowest to highest level they are unproven (U), proof-of-concept (PoC), functional (F) and high (H). Details of what the different Exploitability and Access complexity levels mean are given in [2]. In his examination of information provided by the ten most popular commercial and non-profit vulnerability information providers, Schuppenies shows in [28, p. 52] that four vulnerability information providers provide CVSS impact metrics and that only one, the National Vulnerability Database (NVD) provides the Access Complexity (AC) metric of an attack as well as level of Authentication (Au) needed [28, p. 52]. The National Vulnerability Database (NVD) provides vulnerability information in XML files and is therefore easily parsed [28, p. 53]. Only the X-force vulnerability database provides CVSS temporal scores and thus the Exploitability metric (E) [28, p. 58]. X-force vulnerability information is not publicly provided in a single file format, such as XML and is instead distributed over thousands of web pages, making information extraction more difficult. The right vulnerability can be found from reference links in NVD and in this way vulnerability information can be extracted in a web-crawling like fashion [28, p. 53]. High Exploitability level will increase the probability of exploitation and high Access Complexity level will decrease the probability of exploitation of a vulnerability given that the vulnerability preconditions are satisfied.

Another rich resource of vulnerability information is provided by the non-public Symantec DeepSight Threat Management System [23, p. 5]. This vulnerability database also provides vulnerability metrics that affect the exploit difficulty of vulnerabilities. One such metric is called Ease of Exploit and is

similar to the Exploitability metric (E) in the CVSS temporal score in that it measures the availability of exploit code and if it is needed at all. It has three levels, from lowest to highest level they are no exploit available, exploit available and no exploit required. Another metric that affects the difficulty to exploit a vulnerability is called Availability and measures the likelihood that the vulnerable software is running and vulnerable of exploitation at the time of attack. From lowest to highest level they are circumstantial, user initiated, time dependent and always. Both high levels of Ease of Exploit and Availability will increase the probability of exploitation of a vulnerability given that the vulnerability preconditions are satisfied. Details of what the different levels of vulnerability metrics in Symantec DeepSight Threat Management System mean and what they measure are given in [29].

2.6 Chaining together preconditions and postcondition of attacks into a sequence of attacks

2.6.1 Modeling vulnerabilities for attack graph generation in TVA and NetSPA

Chaining together attack precondition privilege levels and attack postcondition privilege levels of attacks is a big issue in attack graph generation research. The attack postcondition privilege level $pt \in PT$ of one attack $a \in A$ can be the attack precondition privilege level $ps \in PS$ of another attack $a \in A$, thereby enabling a sequence of attacks. All possible sequences of attacks $a \in A$ and the consequence attacker states $as \in AS$ by these attacks are represented in the attack graph. The attack precondition privilege level $ps \in PS$ and attack postcondition privilege level $pt \in PT$ of an attack $a \in A$ is defined by the vulnerability $vl_{g,h} \in VL$ that is being attacked. Therefore information on vulnerability preconditions and postcondition is a crucial issue for attack graph generation research, especially the automatic extraction of this vulnerability information to avoid labour intensive manual analysis [6, p. 1].

Modeling preconditions and postcondition of vulnerabilities for the TVA attack graph generation system, Jajodia et al. state in [26] that it is very difficult to automatically extract vulnerability preconditions and postcondition descriptions in vulnerability databases because the vulnerability reporting community has not defined any standard formal language for specifying such descriptions. Instead, vulnerability databases usually rely on natural language text to describe vulnerabilities [26, p. 6]. Therefore, vulnerabilities are modeled manually in TVA and they state that what is needed is vulnerability descriptions "written in a standard, machine-understandable language" [26, p. 17].

In [3, p. 9], Lippmann et al. state that an analysis of the many online vulnerability databases show that detailed vulnerability descriptions are hard to obtain. For this reason, the set of NetSPA modeled vulnerability postcondition

privilege levels $PTN = \{user,\ admin,\ DoS,\ other\}$ where chosen because vulnerability information on these postcondition privilege levels where easily obtained and verified. The four postcondition privilege levels $ptn \in PTN$ are defined as follows:

- *user* privilege level provides the privilege level of a typical user
- *admin* privilege level provides the privilege level of an administrator on Windows hosts and root privilege level on UNIX hosts. Administrator privilege level is the highest form of privilege level and provides the user full access to a hosts computer resources.
- *DoS* or denial of service privilege level gives an attacker the ability to make the hosts computer resources unavailable to its intended users [3, p. 10]
- *other* privilege level defines loss of confidentiality and/or integrity for specific programs or data [3, p. 13]

In NetSPA, the set of modeled attack precondition privilege levels PSN on a host $h_i \in H$ that enable an attacker to attack vulnerabilities from that host $h_i \in H$ is given by $PSN = \{user, admin\}$ [5, p. 122] and vulnerability locality is given by $L = \{local, remote\}$ [3, p. 10]. Further, in NetSPA it is assumed that the same set of target vulnerabilities VT_i can be attacked from a host $h_i \in H$ with attack precondition privilege levels "user" and "admin" [5, p. 122]. The same assumption is made in the MulVAL system [10, p. 44, p. 104].

Based on the vulnerability precondition privilege level set PSN, postcondition privilege level set PTN and locality set L, in the NetSPA attack graph generation system, a NetSPA vulnerability description $VN_{i,j}$ of vulnerability $vl_{i,j} \in VL$ in a computer network is modeled in the following way:

Definition 2.11 (NetSPA vulnerability model description). *In the NetSPA vulnerability model description, a vulnerability description $VN_{i,j}$ of vulnerability $vl_{i,j} \in VL$ is modeled as a 4-ary Cartesian product over the following four sets:*

- *the vulnerable host $h_i \in H$ with the vulnerability $vl_{i,j} \in VL$, given by the one-element set $H_i = \{h_i \in H\}$*
- *the locality of the vulnerability $l \in L$, where $L = \{local,\ remote\}$, given by the one-element set $L_{i,j} = \{l \in L\}$*
- *the set of attack precondition privilege levels PSN on a host $h_i \in H$ that enable an attacker to attack vulnerability $vl_{i,j} \in VL$ from the host $h_i \in H$ is given by $PSN = \{user,\ admin\}$*
- *the attack postcondition privilege level $ptn \in PTN$, where $PTN = \{user,\ admin,\ DoS,\ other\}$, that an attacker obtains when successfully exploiting vulnerability $vl_{i,j} \in VL$ on host $h_i \in H$ is given by the one-element set $PTN_{i,j} = \{ptn \in PTN\}$*

In the NetSPA vulnerability description model, a vulnerability description $VN_{i,j}$ is modeled as a 4-ary Cartesian product over the 4 sets H_i, $L_{i,j}$, PSN and $PTN_{i,j}$, it is a set of 4-tuples:

$V_{i,j} \subseteq H_i \times L_{i,j} \times PSN \times PTN_{i,j} = \{(h_i,\ l_{i,j},\ psn,\ ptn_{i,j}) :\ h_i \in H_i,\ l_{i,j} \in L_{i,j},\ psn \in PSN,\ ptn_{i,j} \in PTN_{i,j}\}$

meaning that

- *the vulnerability $vl_{i,j} \in VL$ is located on host $h_i \in H$*
- *the vulnerability $vl_{i,j} \in VL$ has locality $l_{i,j} \in L_{i,j}$ where the one-element set $L_{i,j} = \{l \in L\}$*
- *the vulnerability $vl_{i,j} \in VL$ can be attacked with attack precondition privilege level $psn \in PSN$ from a host $h \in H$*
- *the vulnerability $vl_{i,j} \in VL$ gives an attacker attack postcondition privilege level $ptn_{i,j} \in PTN_{i,j}$, where the one element-set $PTN_{i,j} = \{ptn \in PTN\}$, on host $h_i \in H$ when exploited successfully*

The developers of NetSPA face the same problems as the developers of TVA in automatic extraction of vulnerability information [3, p. 11]. Because manual analysis of vulnerability locality $l \in L$ and postcondition privilege level $pt \in PT$ is labour intensive, the researchers developed an automated pattern classifier that automatically extracts vulnerability information from different sources on vulnerability locality $l \in L$ and postcondition privilege level $pt \in PT$ and provides accurate decisions on correct vulnerability description from multiple data sources, including textual descriptions of vulnerabilities. For the NetSPA version in [3], Lippmann et al. used three vulnerability information sources, the Nessus vulnerability scanner, the ICAT database and the CVE vulnerability dictionary. The CVE identifier provides the means to identify a vulnerability across the different vulnerability databases.

The ICAT database was launched in 1999 by the US government through the National Institute of Standards and Technology (NIST) and was superseded by the publicly available National Vulnerability Database (NVD) in 2005. The NVD can be accessed and searched for CVE-identified vulnerabilities on nvd.nist.com. The vulnerability descriptions are available in XML format which makes automatic vulnerability information extraction easier then on regular web pages.

At the time of writing [3], the Nessus vulnerability scanner provided both textual vulnerability descriptions and well defined categorical information on vulnerability locality and postcondition privilege level while the ICAT database just provides categorical vulnerability information and the CVE vulnerability dictionary just provides textual vulnerability descriptions [3, p. 11]. Categorical values are well defined and easily extracted while in textual descriptions the automated pattern classifier searches for common phrases that indicate categorical

values in textual descriptions. Phrases are collected into groups for each category. For example, the administrator privilege level category can be indicated by phrases including "execute arbitrary code" and "gain system privileges" [3, p. 11]. In this way, postcondition privilege level $pt \in PT$ for vulnerabilities can be inferred from the textual descriptions.

2.6.2 Extracting vulnerability information from the National Vulnerability Database for attack graph generation

In the NetSPA attack graph generation system as well as in the MulVAL attack graph generation system it is assumed that the same set of target vulnerabilities VT_i can be attacked from a host $h_i \in H$ with attack precondition privilege levels "user" and "admin". Therefore it is especially important to know which vulnerabilities have postcondition privilege level "$user$" $\in PTN$ or "$admin$" $\in PTN$, allowing an attacker to launch further attacks $a \in A$. In [3, pp. 10-12] Lippmann et al. show how the information on the attack postcondition privilege levels $\{user, admin, DoS, other\} \in PTN$ of vulnerabilities can be found using three sources, Nessus vulnerability scanner, the ICAT database and the CVE vulnerability dictionary. However, ICAT was superseded by the publicly available National Vulnerability Database (NVD) in 2005. Although the method of vulnerability description information extraction stays relevant, one of the sources of these descriptions no longer exists. In [6], Franqueira and Keulen show how information on vulnerability locality $l \in L$ and attack postcondition privilege level $pt \in PT$ can be found in the publicly available National Vulnerability Database (NVD). Because of the CVSS impact metric which was launched with the CVSS metric standard in 2007, better and more detailed information on vulnerability postcondition can be found in NVD today then in the old ICAT database. For non CVSS CIA impact level categories of postcondition privilege levels $pt \in PT$ the authors where able to find the postcondition privilege level categories $PTF = \{user, admin, runCode, DoS, obtainCred\}$ from the NVD. These are defined as

- *user* privilege level provides the privilege level of a typical user
- *admin* privilege level means an attacker gains privilege level of an administrator on Windows hosts and root privilege level on UNIX hosts
- *runCode* privilege level means an attacker gains the ability to execute arbitrary code on the vulnerable host
- *DoS* or denial of service privilege level means an attacker gains the ability to make the vulnerable host unavailable to its legitimate users
- *obtainCred* means an attacker gains the ability to obtain credentials for the vulnerable host

The NVD also provides the CVSS base score where the postcondition privilege level categories are given by the impact metric. It defines the postcondition privilege level obtained on a host as the level of impact on the confidentiality, integrity and availability of information security. The level of impact on a host on each category of information security is given by three levels of severity of impact, none, partial and complete.

The authors extracted relevant vulnerability information on locality and postcondition privilege level $pt \in PT$ for attack graph generation on 27,273 CVE-identified vulnerabilities in NVD created between 1999 and 2007 by loading the XML files into an XML database. Information on what phrases indicate what attack postcondition privilege level and the XML tag where these phrases can be found in the NVD XML files is described in table 1 in [6, p. 9]. This information can be used when developing a parser that automatically extracts vulnerability information from NVD on vulnerability locality and postconditon privilege level for vulnerabilities found in a network.

2.6.3 Statistical analysis of vulnerability information in the National Vulnerability Database

In [6], Franqueira and Keulen analyze how privilege levels inferred from textual descriptions relate to the CVSS impact scores and also how common the different attack postcondition privilege levels and localities are for the total set of CVE-identified vulnerabilities in NVD between 1999 and 2007. All vulnerabilities with attack postcondition privilege level "admin" have complete impact on all categories of CIA in the CVSS base score impact metrics. Thus, vulnerabilities that are found to have complete CIA impact in the CVSS base score impact metric can be classified as vulnerabilities with postcondition privilege level "admin" even though this can not be inferred from textual descriptions [6, p. 13]. DoS impacts heavily on availability, thus 3695 out of 3964 (93.2%) of complete Availability impact metric category vulnerabilities cause "DoS" in textual descriptions. From this strong correlation the authors assume vulnerabilities with complete Availability impact can be classified as vulnerabilities with "DoS" attack postcondition privilege level even though this can not be inferred from textual descriptions [6, p. 14]. The "user" privilege level always results in partial CIA impact where all three categories of impact metrics, confidentiality, integrity and availability have partial impact, although partial CIA impact does not necessarily result in "user" privilege level, it can also result in "no privilege" and "other" privilege level. Thus, nothing can be inferred from CVSS impact metrics on postcodition privilege level "user".

Further interesting results from their analysis of the NVD show that 97.1% of vulnerabilities require no credentials, the privilege level given by "obtainCred", to be exploited based on the authentication metric category in CVSS. Only 0.5% of vulnerabilities have the postconditon privilege level "obtainCred" as inferred from phrases from their textual description [6, p. 9].

By the definition of the NetSPA vulnerability model of the set of attack precondition privilege levels PSN = {user, admin}, an interesting question

is how common it is for vulnerabilities to have attack postcondition privilege level "user" $\in PTN$ or "admin" $\in PTN$, resulting in the possibility to launch new attacks $a \in A$. For vulnerability descriptions $V_{i,j}$ of a vulnerability $vl_{i,j} \in VL$, their analysis shows that 9.8% of vulnerabilities have vulnerability descriptions that contain the 4-tuple $(h_i, remote, admin, admin) \in V_{i,j}$ and 6.8 % of vulnerabilities have vulnerability descriptions that contain the 4-tuple $(h_i, remote, admin, user) \in V_{i,j}$. This shows that 16.6 % of vulnerabilities can be exploited remotely and result in "user" or "admin" privilege level.

This is a pretty large share and a conclusion can be made that vulnerabilities that have one of the attack postcondition privilege levels "user" $\in PT$ or "admin" $\in PT$ that result in the possibility to launch new attacks $a \in A$ and deeper penetration into computer networks and are remotely exploitable is pretty large and that these vulnerabilities indeed pose a great risk to computer networks. The danger from this kind of vulnerabilities to the security of computer networks should be analyzed with the help of attack graphs or better, probabilistic attack graphs since they affect the security of the whole network and not just the security of the vulnerable host.

2.6.4 Modeling privilege levels "read credentials" and "ability to change firewall rules"

When an attacker has obtained postcondition privilege level $pt \in PT$ ability to "read credentials" on a host, an attacker can use the credentials to obtain trust relationships between hosts in the network [19, p. 990]. If a host $h_j \in H$ trusts another host $h_i \in H$ it means that a user on host $h_i \in H$ can access another host $h_j \in H$ without authentication [11, p. 61]. The result is that an attacker has obtained new possibilities of remote attacks $a \in A$ from other attacker states $as \in AS$ where the privilege level is $ps \in PS$ on other hosts $h \in H$ in the network.

The privilege level "change firewall rules" has the same effect on the attack graph. By changing the firewall rules, an attacker presumably changes the firewall rules so that new vulnerable data connectivity relations $c \in C$ are added in the network, thereby creating opportunities for new remote attacks $a \in A$ from other attacker states $as \in AS$ where the privilege level is $ps \in PS$ on other hosts $h \in H$ in the network [19, p. 10].

Both these privilege levels affect the privilege levels on other possible attacker states $as \in AS$ in the attack graph unlike other attack postcondition privilege levels that do not affect the privilege levels at other possible attacker states.

Although it is possible to manually determine where credentials are and what they protect, Lippmann et al. have not developed any tool that can automatically determine where credentials are and what they protect for the NetSPA system version in [5, p. 122]. Nor does NetSPA model the privilege level that enables an attacker to change firewall rules because of the difficulty of automatically model both these postcondition privilege levels [19, p. 990].

Chapter 3

Bayesian networks

3.1 The causal network

A causal network is a directed graph where the nodes represent variables and the edges indicate how the variables influence each other [24, p. 26]. It gives a qualitative view of the dependencies among variables [25, p. 38]. The directed edges connecting the nodes represent the causality between the variables in the network. Outbound edges from a variable A indicate which other variables that the variable A influences. Inbound edges to variable A indicate what other variables influence the state of variable A. When talking about the influence between variables in a causal network, the terminology of family relations is used, if there is an edge from B to A then B is a parent of A, and A is a child of B [24, p. 26]. The set of parent variables of a variable C is denoted pa(C) [24, p. 36]. Influence between variables are quantified with conditional probability distributions [24, p. 32].

3.2 Discrete probability distributions and discrete conditional probability distributions

Since probabilistic attack graphs are Bayesian networks consisting of discrete random variables, we will here describe the theory of discrete random variables. A random variable is an object that unlike other variables does not have a single, fixed value, instead it can take on a set of possible different values where each value is associated with a probability that the random variable will have that value. A random variable must be in one of its *mutually exclusive* and *exhaustive* states. For discrete random variables the set of states associated with a random variable is finite and the state space of a discrete random variable is denoted by $sp(A) = (a_1,\ a_2,\ \ldots,\ a_n)$. That the states are mutually exclusive ensures that the random variable is in only one state, and that the states are exhaustive ensures that the variable is in one of its states [24, p. 7]. The probability that

	B		
A	b_1	b_2	b_3
a_1	0.4	0.3	0.6
a_2	0.6	0.7	0.4

Table 3.1: An example of a conditional probability table $P(A \mid B)$ for the binary variable A given the ternary variable B. For each state of B the probabilities of A must sum up to 1.

a discrete random variable A with state space $sp(A) = (a_1,\ a_2,\ \ldots,\ a_n)$ is in each of its states is given by the discrete probability distribution P(A):

$$P(A) = (x_1,\ \ldots,\ x_n);\ x_1 \geq 0;\ \sum_{i=1}^{n} x_i = 1, \tag{3.1}$$

where x_i is the probability of A being in state a_i [24, p. 7].

The influence between variables is quantified with conditional probability distributions. A conditional probability distribution $P(A \mid B)$ defines the outcome of a variable A given the outcome of a variable B. Let X be a discrete random variable, then $\mid sp(X) \mid$ denotes the number of states of X [24, p. 135]. The discrete conditional probability distribution $P(A \mid B)$ contains $\mid sp(A) \mid * \mid sp(B) \mid$ conditional probabilities that specify the probabilities of the outcome of each state $a_i \in sp(A)$ given the outcome of each state $b_j \in sp(B)$. For each state $b_j \in sp(B)$, the probabilities of A must sum to 1:

$$\sum_{i=1}^{n} P(A = a_i \mid B = b_j) = 1\ for\ each\ b_j \in sp(B) \tag{3.2}$$

Discrete conditional probability distributions are usually represented in tables, table 3.1 gives an example of a conditional probability distribution table $P(A \mid B)$.

When the state of a variable is known, it is said that the random variable is instantiated, affecting the probability distribution of other random variables that it influences [24, p. 26]. If a variable is instantiated it is also said that evidence of the state of a variable has been received. For example in the above example if it is known that the discrete random variable B is in the state b_1, then we know that the discrete probability distribution of the variable becomes

$$P(A \mid B = b_1) = (0.4,\ 0.6)$$

More generally, the discrete conditional probability distribution $P(A \mid B_1,\ B_2,\ \ldots,\ B_n)$ defines the outcome of a variable A given the outcome of the variables $B_1,\ B_2,\ \ldots,\ B_n$. The discrete conditional probability distribution $P(A \mid B_1,\ B_2,\ \ldots,\ B_n)$ contains $\mid sp(A) \mid * \prod_{i=1}^{n} \mid sp(B_i) \mid$ conditional probabilities that specify the probabilities of the outcome of each state $a_i \in sp(A)$

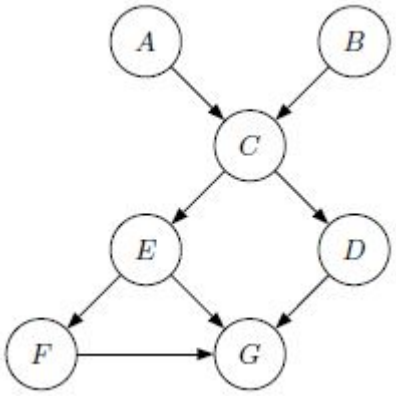


Figure 3.1: A directed acyclic graph (DAG), representing a Bayesian network. The probabilities to specify are $P(A)$, $P(B)$, $P(C \mid A,\ B)$, $P(E \mid C)$, $P(D \mid C)$, $P(F \mid E)$ and $P(G \mid D,\ E,\ F)$.

given each configuration of the states $b_{1,j} \in sp(B_1),\ b_{2,k} \in sp(B_2),\ \ldots,\ b_{n,f} \in sp(B_n)$ [24, p. 34].

3.3 Definition of Bayesian networks

A causal network with no cycles and conditional probability tables attached to each variable in the causal network is a Bayesian network (see figure 4.1).

Definition 3.1 (Bayesian network). *A Bayesian network consists of the following [24, p. 33]:*

- *A set of random variables and a set of directed edges between the random variables.*
- *Each random variable has a set of mutually exclusive states.*
- *The random variables together with the directed edges form an acyclic directed graph (traditionally abbreviated DAG); a directed graph is acyclic if there is no directed path $A_1,\ \ldots,\ A_n$ so that $A_1 = A_n$.*
- *To each random variable A with parents $B_1,\ \ldots,\ B_n$, a conditional probability table $P(A \mid B_1,\ \ldots,\ B_n)$ is attached.*

An example of a Bayesian network is given in fig. 3.1 which has been taken from [24, p. 34].

Chapter 4

Probabilistic attack graph generation model

4.1 Vulnerability model for probabilistic attack graph generation

In our probabilistic attack graph vulnerability model, we model vulnerability descriptions based on two criteria. The vulnerability model should be as precise as possible in defining attack postcondition privilege level and be available from the publicly available NVD as described by Frankueira and Keulen in [6, p. 9].

4.1.1 Modeling precondition privilege levels $psp \in PSP$ and locality $l \in L$

We model vulnerability attack precondition privilege level and locality in the same way as in the NetSPA attack graph generation system. Thus, as in NetSPA, the set of modeled precondition privilege levels PSP on a host $h \in H$ that enable an attacker to attack vulnerabilities from that host $h \in H$ is given by the set $PSP = \{user,\ admin\}$ in our vulnerability model [5, p. 122]. Further, as in NetSPA and MulVAL, it is assumed that the same set of target vulnerabilities $vt_i \in VT_i$ on target hosts $ht_i \in HT_i$ can be attacked by attack precondition privilege levels "$user$" $\in PSN$ and "$admin$" $\in PSN$ from a host $h_i \in H$ [5, p. 2], [10, p. 44, p. 104]. Following the NetSPA model on the vulnerability locality set, we model two vulnerability localities, "local" and "remote", given by the locality set $L = \{local,\ remote\}$. The locality of vulnerabilities can be found in the access vector in CVSS in NVD that consists of the three categories "local", "adjacent network" and "network". Here, the "network" category means "remote" locality and the "adjacent network" category refers to vulnerabilities that can be accessed with wireless technologies like Bluetooth and print drivers. Vulnerabilities that have locality "adjacent network" in CVSS consist of only

0.2 % of vulnerabilities found in the NVD vulnerability database between 1999 and 2007 and are modeled as remotely exploitable in our model.

4.1.2 Modeling postcondition privilege levels $pt \in PT$

We model the set of vulnerability postconditions as they do in the NetSPA model where the NetSPA model set of vulnerability postcondition privilege levels is given by $PTN = \{user,\ admin,\ DoS,\ other\}$ [3, p. 10] where the privilege levels in PTN are defined as in section 2.6.1. In our model the "other" postcondition privilege level category is replaced by the much more detailed CIA impact level combinations from CVSS CIA impact levels.

A privilege level as defined by the CVSS impact metric is given by the level of violation on the CIA of information security from the illegal privilege level. To describe CVSS impact level descriptions we use abbreviations for the CIA category that is impacted followed by a colon followed by an abbreviation for the impact level, this impact level description is done for all categories of CIA. The CIA category confidentiality has the abbreviation C, integrity has the abbreviation I and availability has the abbreviation A, among impact levels, impact level none has abbreviation N, impact level partial has abbreviation P and impact level complete has abbreviation C. In this way, an impact level none on confidentiality, complete on integrity and partial on availability is described as "C: N - I: C - A: P". Since there is a strong correlation between CIA impact level "C: N - I: N - A: C" and privilege level "DoS" in the NVD database [6, p. 14], all vulnerabilities with CIA impact level descriptions "C: N - I: N - A: C" are classified as "DoS" privilege level. In the same way, vulnerabilities with CIA impact level descriptions "C: C - I: C - A: C" are classified as privilege level "admin" [6, p. 13].

The set of postcondition privilege levels as defined by CVSS CIA impact level descriptions is given by all combinations of CVSS CIA impact levels except for the impact level combinations "C: N - I: N - A: C" and "C: C - I: C - A: C" that if found in a vulnerability description of a vulnerability is reclassified as attack postcondition privilege level "DoS" and "admin" respectively. In all cases of defining the attack postcondition privilege level of a vulnerability, if a textual description is found on a vulnerability that implies on attack postcondition privilege levels "user", "admin" and "DoS", then this vulnerability is classified with postcondition privilege level categories "user", "admin" and "DoS" instead of the CVSS CIA impact level description.

Let the set of privilege levels that are defined by all combinations of CVSS CIA impact levels except for the CIA impact level combinations 'C: N - I: N - A: C' and 'C: C - I: C - A: C' be given by the set CIAI. In our vulnerability model, the set of possible attack postcondition privilege levels that an attacker can obtain when successfully attacking a vulnerability $vl \in VL$ on a host $h \in H$ is given by the postcondition privilege level set $PTP = CIAI \cup \{user,\ admin,\ DoS\}$.

The privilege levels "read credentials" and "change firewall rules" affect the privilege levels at other attacker states in the attack graph. Both privilege levels are very dangerous since they enable remote attacks $a \in A$ from other attacker

states $as \in AS$ then the one where the privilege level is obtained. In a simplified way they could be modeled by simply placing the monetary loss value on the attacker states where the privilege levels are "read credential" or "change firewall rules" at a very high level. The information necessary to know which vulnerabilities in the network that have these attack postcondition privilege levels is not provided in NVD. It can be a vulnerability that for example provides administrative access on a host and thereby gives an attacker the ability to change firewall rules or read credentials, but the information on which vulnerabilities provide these attack postcondition privilege levels is network specific and not general, therefore this information has to be provided by the network security administrator.

4.1.3 Modeling the exploit difficulty of vulnerabilities

To model the conditional probability of exploiting a vulnerability for probabilistic attack graph generation we need to model the exploit difficulty of vulnerabilities. Here, we use the exploitability metric from the CVSS temporal score and access complexity metric from the CVSS base score. We use the combination of CVSS exploitability metric and CVSS access complexity metric to give a vulnerability an exploit difficulty value. A high exploit difficulty value means that the vulnerability is hard to exploit while a low exploit difficulty value means that a vulnerability is easy to exploit. The exploitability metric is only provided by the X-force vulnerability database and is publicly provided, but distributed on thousands of web pages. Therefore it is harder to extract this information then the access complexity metric which is provided by NVD in XML file format. But, ideally, an exploit difficulty value should be a function of both these metric values to be as accurate as possible.

Definition 4.1 (Exploit difficulty function ED). *To define the exploit difficulty function ED we first need to define the exploitability value function E and the access complexity value function AC. The exploitability function E(x, y) where $x \in \mathbb{N}$ and $y \in \mathbb{N}$ maps the exploitability metric value of vulnerability $vl_{x,y} \in VL$ into a numerical exploitability value. The range of E(x, y) is given by the set $R(E) = \{1.5,\ 3,\ 4.5,\ 6\}$.*

If vulnerability $vl_{x,y} \in VL$ has exploitability metric value high, then E(x, y) = 1.5
If vulnerability $vl_{x,y} \in VL$ has exploitability metric value functional, then E(x, y) = 3
If vulnerability $vl_{x,y} \in VL$ has exploitability metric value proof-of-concept, then E(x, y) = 4.5
If vulnerability $vl_{x,y} \in VL$ has exploitability metric value unproven, then E(x, y) = 6

The access complexity function AC(x, y) where $x \in \mathbb{N}$ and $y \in \mathbb{N}$ maps the access complexity metric value of vulnerability $vl_{x,y} \in VL$ into a numerical access

complexity value. The range of AC(x, y) is given by the set $R(AC) = \{2,\ 4,\ 6\}$.

If vulnerability $vl_{x,y} \in VL$ *has access complexity metric value low, then AC(x, y) = 2*
If vulnerability $vl_{x,y} \in VL$ *has access complexity metric value medium, then AC(x, y) = 4*
If vulnerability $vl_{x,y} \in VL$ *has access complexity metric value high, then AC(x, y) = 6*

The exploit difficulty function ED(x, y) where $x \in \mathbb{N}$ *and* $y \in \mathbb{N}$ *gives the exploit difficulty value of vulnerability* $vl_{x,y} \in VL$. *The exploit difficulty function ED(x, y) is given by*

$$ED(x, y) = (E(x, y) + AC(x, y))/1.2 \tag{4.1}$$

The range of ED is given by $R(ED) \subseteq [35/12, 10]$

4.1.4 Probabilistic vulnerability model description

Based on the vulnerability locality set $L = \{local,\ remote\}$, attack precondition privilege level set $PSP = \{user,\ admin\}$, attack postcondition privilege level set $PTP = CIAI \cup \{user,\ admin,\ DoS\}$, and range of the exploit difficulty function ED, given by the numerical values in the range R(ED) of the function ED, our probabilistic vulnerability model description for probabilistic attack graph generation is defined in the following way:

Definition 4.2 (Probabilistic vulnerability model description). *Let the set of privilege levels that are defined by all combinations of CVSS CIA impact levels except for the CIA impact level combinations 'C: N - I: N - A: C' and 'C: C - I: C - A: C' be given by the set CIAI. In our vulnerability model, the set of possible attack postcondition privilege levels that an attacker can obtain when successfully attacking a vulnerability* $vl \in VL$ *on a host* $h \in H$ *is given by the attack postcondition privilege level set* $PTP = CIAI \cup \{user,\ admin,\ DoS\}$. *Let the set of probabilistic vulnerability descriptions of all vulnerabilities* $vl \in VLi$ *on host* $h_i \in H$ *be given by* $VP_i = VP_{i,1} \cup VP_{i,2} \cup \ldots \cup VP_{i,m}$ *and the total set of probabilistic vulnerability descriptions of all vulnerabilities* $vl \in VL$ *in a network be given by* $VP = VP_1 \cup VP_2 \cup \ldots \cup VP_n$.

We define a probabilistic vulnerability model description $VP_{i,j}$ *of vulnerability* $vl_{i,j} \in VL$ *as a 5-ary Cartesian product over the following 5 sets:*

- *the vulnerable host* $h_i \in H$ *with the vulnerability* $vl_{i,j} \in VL$, *given by the one-element set* $H_i = \{h_i \in H\}$
- *the locality* $l \in L$ *of the vulnerability* $vl_{i,j} \in VL$, *where* $L = \{local,\ remote\}$, *given by the one-element set* $L_{i,j} = \{l \in L\}$

- *the set of attack precondition privilege levels PSP on a host hi* $\in H$ *that enable an attacker to attack a vulnerability vl* $\in VL$ *from the host hi* $\in H$ *is given by* $PSP = \{user,\ admin\}$

- *the attack postcondition privilege level, given by the one-element set* $PTP_{i,j} = \{ptp \in PTP\}$ *that an attacker obtains when successfully exploiting vulnerability* $vl_{i,j} \in VL$ *on host* $h_i \in H$

- *the exploit difficulty value* $ED(i,j) \in R(ED)$*, given by the one-element set* $ED_{i,j} = \{ED(i,j) \in R(ED)\}$

We define a probabilistic vulnerability model description $VP_{i,j} \in VP$ *of vulnerability* $vl_{i,j} \in VL$ *as a 5-ary Cartesian product over the 5 sets* $H_i,\ L_{i,j},\ PSP,$ $PTP_{i,j}$ *and* $ED_{i,j}$*, it is a set of 5-tuples:*

$VP_{i,j} = H_i \times L_{i,j} \times PSP \times PTP_{i,j} \times ED_{i,j} = \{(h_i,\ l_{i,j},\ psp,\ ptp_{i,j},\ ed_{i,j}) :$ $h_i \in H_i,\ l_{i,j} \in L_{i,j},\ psp \in PSP,\ ptp_{i,j} \in PTP_{i,j},\ ed_{i,j} \in ED_{i,j}\}$

meaning that

- *the vulnerability* $vl_{i,j} \in VL$ *is located on host* $h_i \in H$

- *the vulnerability* $vl_{i,j} \in VL$ *has locality* $l_{i,j} \in L_{i,j}$ *where the one-element set* $L_{i,j} = \{l \in L\}$

- *the vulnerability* $vl_{i,j} \in VL$ *can be attacked with attack precondition privilege level psn* $\in PSP$ *from a host* $h \in H$

- *the vulnerability* $vl_{i,j} \in VL$ *gives an attacker attack postcondition privilege level* $ptp_{i,j} \in PTP_{i,j}$ *where the one element-set* $PTP_{i,j} = \{ptp \in PTP\}$ *on host* $h_i \in H$ *when exploited successfully*

- *the vulnerability* $vl_{i,j} \in VL$ *has exploit difficulty value ED(i ,j)*

4.2 Generating a probabilistic attack graph

4.2.1 Probabilistic attacks

Attacks are defined by the vulnerability that is attacked, thus the set of possible attacks A_i from a host $h_i \in H$ is defined by the vulnerability model that is being used. Defined by our vulnerability model description for probabilistic attack graph generation, the set of possible attacks AP_i from a host $h_i \in H$ is defined as:

Definition 4.3 (Set of possible probabilistic attacks from host $h_i \in H$**).** *Let* AP_i *be the set of possible probabilistic attacks from host* $h_i \in H$ *and let* $AP = AP_1 \cup AP_2 \cup \ldots \cup AP_n$ *be the set of possible probabilistic attacks in the whole network from all hosts H.*

We define the set of possible probabilistic attacks AP_i launched from host $hi \in H$ as a subset of the 6-ary Cartesian product over the following 6 sets:

- *the vulnerable host $hi \in H$ from where the attack is launched, given by the one-element set $H_i = \{h_i \in H\}$*
- *the set of attack precondition privilege levels PSP on the host $h_i \in H$ that enable an attacker to attack a vulnerability $vl_{f,g} \in VL$ from the host $h_i \in H$*
- *the set of target vulnerabilities $VT_i \subseteq VL$ that an attacker can attack from host $h_i \in H$*
- *the set of target hosts $HT_i \subseteq H$ with a vulnerability $vt_i \in VT_i$ that an attacker can attack from host $h_i \in H$*
- *the set of attack postcondition privilege levels PTP on the host $ht_i \in HT_i$ that an attacker can obtain when successfully attacking target vulnerability $vt_i \in VT_i$ on host $ht_i \in HT_i$*
- *the set of possible exploit difficulty values, given by the range of ED, R(ED)*

We define the set of possible attacks A_i from source host $h_i \in H$ as a subset of the 6-ary Cartesian product over the 6 sets, H_i, PSP, VT_i, HT_i, PTP and R(ED), it is a set of 6-tuples:

$AP_i \subseteq H_i \times PSP \times HT_i \times VT_i \times PTP \times R(ED) = \{(h_i,\ psp,\ ht_i,\ vt_i,\ ptp,\ ED(f,g)) :$
$h_i \in H_i,\ psp \in PSP,\ ht_i \in HT_i,\ vt_i \in VT_i,\ ptp \in PTP,\ ED(f,g) \in R(ED)\}$

meaning that

- *a probabilistic attack can be launched from host $h_i \in H$ where an attacker has attack precondition privilege level $ps \in PS$ on target vulnerability $vt_i \in VT_i$ on target host $ht_i \in HT_i$ that has exploit difficulty value ED(f, g)*
- *the probabilistic attack gives an attacker attack postcondition privilege level $pt \in PT$ on target host $ht_i \in HT_i$ when vulnerability $vt_i \in VT_i$ is successfully attacked*

The probabilistic attack graph is solely determined by the set of vulnerabilities VL in a network, the set of vulnerability descriptions of all vulnerabilities in a network VP, the set of IDS values in a network given by the set IDS and the set of vulnerable data connectivity relations C in a network.

This can partly be understood by studying the attack preconditions of an attack. From section 2.1.5 we know that the preconditions of an attack are defined as:

Attack preconditions are: Precondition privilege level $psp \in PSP$ on host $h_i \in H$ from where the attack is launched, vulnerable data connectivity relation

$c_i \in C_i$ if the attack is launched against a remote host and existence of a vulnerability $vl_{f,g} \in VL$ on attacked host $h_f \in H$.

Both vulnerabilities $vl \in VL$ and vulnerable data connectivity relations $c \in C$ are known to exist in a network unlike attacks $ap \in AP$ and attacker states $as \in AS$ that represent possibilities of events and conditions respectively that are enabled by the existence of vulnerabilities $vl \in VL$, their descriptions $VP_{i,j}$ and vulnerable data connectivity relations $c \in C$. The event of an attack $ap \in AP$ creates a condition in the environment when an attacker obtains an attacker state $as \in AS$, enabling a new attack $ap \in AP$, this sequence of attacks is repeated until the whole probabilistic attack graph has been generated where all possible sequences of attacks and the resulting attacker states from these attacks are represented in an acyclic directed graph. As described in section 2.5, a probabilistic attack graph is an exploit dependency attack graph where the attack and attacker state nodes have been transformed into discrete random variables with discrete probability distributions or discrete conditional probability distributions and where all cycles have been removed.

To be able to chain together all possible attacks $ap \in AP$ into a sequence of attacks and find all possible attacker states $as \in AS$ that are a result of these attacks we need to find out how the set of possible attacks AP_i from a host $h_i \in H$ in the network is implied based on network vulnerability data.

4.2.2 Implying attacks $ap \in AP$ and attacker states $as \in AS$ from network vulnerability data

As described in section 2.1.5, attacks can be of two kinds, local and remote, where local attacks are attacks that are launched against the same host $h_i \in H$ as the host $hi \in H$ that they are launched from and remote attacks are attacks that are launched against other hosts $h_j \in H$ then the host $hi \in H$, $h_j \in H \neq h_i \in H$, from where the attack is launched.

From each host where an attacker has attack precondition privilege level $psp \in PSP$ where $PSP = \{user,\ admin\}$, the set of possible attacks $ap_i \in AP_i$ is the same for the attacker when having attack precondition privilege level "user" and "admin". However, for an attacker there is no reason to launch a local attack if he has obtained privilege level "admin" on host $h_i \in H$ since the "admin" privilege level provides the highest privilege level available and launching new attacks on the same host to gain a higher privilege level is impossible. The privilege level "user" on the other hand does not provide an attacker with full access to the computer's resources (at least not in all cases) and thus, local attacks will only be launched if an attacker has privilege level "user", for example to obtain the higher privilege level "admin". Thus, from an attacker state $(h_i, user) \in AS$, the whole set of possible attacks $ap_i \in AP_i$ from host $h_i \in H$ can be launched, while from attacker states $(h_i,\ admin) \in AS$, only the subset of remote attacks will be launched from host $h_i \in H$. Attacks $ap \in AP$ are implied in the following way from network vulnerability data:

Definition 4.4 (Implying a possible attack $ap_i \in AP_i$ from host $h_i \in H$

from network vulnerability data). *Let the set of possible local attacks from host $h_i \in H$ be given by AL_i and the set of possible remote attacks from host $h_i \in H$ be given by AR_i. The total set of attacks from host $h_i \in H$ is given by $AP_i = AL_i \cup AR_i$.*

A local attack $al_i \in AL_i$ is implied in the following way:

$(h_i,\ user) \in AS\ \wedge\ vl_{i,j} \in VL_i\ \wedge\ (h_i,\ local,\ user,\ ptp_{i,j}, ed_{i,j}) \in VP_{i,j} \longrightarrow$
$(h_i, user, vl_{i,j}, h_i, ptp_{i,j}, ed_{i,j}) \in AL_i$

A remote attack $ar_i \in AR_i$ is implied in the following way:

$(h_i,\ prp) \in AS\ \wedge\ (h_i,\ h_f,\ vl_{f,g}) \in C_i\ \wedge\ vl_{f,g} \in VL \setminus VL_i\ \wedge$
$(h_f,\ remote,\ prp,\ ptp_{f,g},\ ed_{f,g}) \in VP_{f,g}\ \wedge\ prp \in \{user,\ admin\} \longrightarrow$
$(h_i,\ prp,\ h_f,\ vl_{f,g},\ ptp_{f,g},\ ed_{f,g}) \in AR_i\ \wedge\ prp \in \{user,\ admin\}$

From an attack $ap \in AP$, the resulting attacker state $as \in AS$ is implied in the following way:

Definition 4.5 (Implying a resulting attacker state $as \in AS$ from an attack $ap \in AP$). *An attacker state $as \in AS$ is implied in the following way:*

$(h_i,\ prp,\ h_f,\ vl_{f,g},\ ptp_{f,g},\ ed_{f,g}) \in AP_i \longrightarrow (h_f,\ ptp_{f,g}) \in AS$

4.2.3 Generating a probabilistic attack graph from network vulnerability data

The set of vulnerabilities that exist in a network VL are described by the vulnerability description data set VP and therefore don't have to be taken into account when defining the vulnerability of a network. Therefore, a collection of data that determines the probabilistic attack graph of a network is a set VD, the network vulnerability data set defined as $VD = VP \cup C \cup IDSS$. It is the union of the descriptions of the vulnerabilities that are known to exist in a network, the vulnerable data connectivity relations that are known to exist in a network and the set of IDS measurements in a network.

Attack graphs are of most interest when showing the set of possible attacker states AS that an attacker can obtain from outside although they can be generated with starting privilege level $psp \in PSP$ from a host inside the network modeling the starting privilege level of an employee or other trusted individual with access to some hosts or several hosts computer resources. But most commonly, the attack graphs of biggest interest are those that show the sequence of attacks that can be made from outside the network.

From the perspective of the attack graph, a group of hosts that have the same set of vulnerabilities and the same set of inbound and outbound vulnerable data connectivity relations are treated as a single host [3, pp. 37-38]. Such a group of hosts is called a host group. If one of the hosts in a host group can be compromised then all of the hosts in a host group can be compromised.

Therefore an attacker state $as \in AS$ can either represent the privilege level of the attacker on one host or on a whole host group.

A probabilistic attack graph generation process must start by generating all possible attacks from the host where an attacker has precondition privilege level $psp \in PSP$ from the start. From these attacks, the set of implied attacker states must be found. In this set of attacker states new attacks must be implied from the attacker states where the privilege level is "user" or "admin".

This process is repeated so that new layers of attacks and attacker states are added to the probabilistic attack graph. Each attack $ap \in AP$ in a new layer of attacks must be checked if it hasn't been launched already, if it has, this attack must be removed. In the same way, each attacker state $as \in AS$ in a new layer must be checked, if the attacker state doesn't exist already, a new node is created in the acyclic directed graph. If it already exists, the edge is simply drawn to the existing node and in this case it has to be controlled if the new edge has created a cycle in the directed graph. This is done by controlling if the attacker state node to which the edge is drawn becomes an ancestor node to itself, if it does, a cycle has been created and this edge has to be removed. A node γ is an ancestor of a node α if and only if there is a directed path from γ to α. The set of ancestor nodes $A(\alpha)$ of a node α is defined as [25, p. 44]:

Definition 4.6 (Ancestor). *Let G = (V, D) be a directed graph. A node γ is an ancestor of a node α if and only if there is a directed path from γ to α. The set of ancestors $A(\alpha)$ of a node α is defined as*

$$A(\alpha) = \{\beta \in V \mid \exists \tau = (\tau_o, \ \ldots, \ \tau_k) : \ \tau_0 = \beta, \ \tau_k = \alpha, \ (\tau_j, \tau_{j+1}) \in D, \ j = 0, \ 1, \ \ldots, \ k\}.$$

That an edge that is drawn to a node α so that the node α becomes an ancestor node to itself creates a cycle can be proven in the following way. Let G = (V, D) be a directed graph. Let $A(\alpha)$ be the set of ancestor nodes of a node α and let $\beta \in A(\alpha)$. It can be proven that the creation of a directed edge $(\alpha, \beta) \in D$ will create a cycle in G:

From definition 4.6 we know that an ancestor node $\beta \in A(\alpha)$ is the node β in a path $\beta, \ldots, \ \alpha$. If a directed edge $(\alpha, \ \beta) \in D$ is added to D, then a path $\beta, \ \ldots, \ \alpha, \ \beta$ is created.

The path $\beta, \ \ldots, \ \alpha, \ \beta$ is a cycle and is not allowed in Bayesian networks and therefore not allowed in the probabilistic attack graph. Therefore it must be removed in the probabilistic attack graph generation process.

Layers of new attacks and attacker states are added in the probabilistic attack graph generation process until no new attacks and attacker states can be added according to the rules of the probabilistic attack graph generation process. When no new attacks can be added, the probabilistic attack graph has been generated for the network based on the network vulnerability data set VD. The result is the probabilistic attack graph where attack $ap \in AP$ nodes and attacker states $as \in AS$ nodes are discrete random variables:

Definition 4.7 (Probabilistic attack graph). *Given a set of attack discrete random variables X and a set of attacker state discrete random variables Y where $X \cap Y = \oslash$ and two directed edge sets $U \subseteq Y \times X$ and $V \subseteq X \times Y$, a probabilistic attack graph is an acyclic directed graph $G(X \cup Y,\ U \cup V)$.*

4.2.4 Discrete conditional probability distributions for attack $ap \in AP$ and attacker state $as \in AS$ random variables

$X(h_i, psp, h_j, vl_{j,k}, ptp, ED(j,k))$ denotes the attack random variable that gives the probability that the attack $(h_i, psp, h_j, vl_{j,k}, ptp, ED(j,k)) \in AP$ will be launched a certain number of times in a certain time interval. $Y(h_i, ptp)$ denotes the attacker state random variable that gives the probability that the attacker state $(h_i, ptp) \in AS$ will be reached a certain number of times in a certain time interval. Both attack random variables and attacker state random variables have Poisson distributions. A Poisson distribution $X \sim Po(\lambda)$ with parameter $\lambda > 0$ is a discrete probability distribution that expresses the probability of a given number of events k occurring in a fixed interval of time if these events occur with a known average rate $\lambda = E(X)$ and independently of the time since the last time. A random variable X that has a Poisson distribution is defined as [30, p. 55]:

Definition 4.8 (Poisson distribution). *If a random variabe X has the probability distribution*

$$P(X = k) = \lambda^k \cdot e^{-\lambda}/k!,\ k = 0,\ 1,\ 2,\ \ldots,\ \lambda > 0 \tag{4.2}$$

it is said that X has a Poisson distribution with paramter $\lambda > 0$, $X \sim Po(\lambda)$.

The positive real number λ is equal to the expected value E(X) of X, $\lambda = E(X)$.

For attacker state random variables the Poisson distribution gives us the probability that an attacker state $as \in AS$ will be reached a certain number of times in a given period of time by an attacker. For attack random variables the Poisson distribution gives us the probability that an attack $a \in A$ will be launched a certain number of times in a given period of time.

Attack random variables are always implied by a single attacker state $as \in AS$ and therefore have conditional probability distributions that are conditioned on the single attacker state random variable that implies the attack $ap \in AP$. Attack random variables are defined in the following way:

Definition 4.9 (Discrete conditional probability distribution for attack random variables $X(h_i,\ psp,\ h_j,\ vl_{j,k},\ ptp,\ ED(j,k))$). *Suppose that the attack $(h_i,\ psp,\ h_j,\ vl_{j,k},\ ptp,\ ED(j,k)) \in AP$ is launched successfully with average probability k given that an attacker has obtained the precondition attacker state $(h_i, psp) \in AS$ for the attack $(h_i,\ psp,\ h_j,\ vl_{j,k},\ ptp,\ ED(j,k)) \in AP$. In*

the probabilistic attack graph $(h_i, psp) \in AS$ *is the parent of* $(h_i,\ psp,\ h_j,\ vl_{j,k},\ ptp,\ ED(j,k)) \in AP$. *Let* $y = E(Y(h_i,\ psp))$, *meaning that the attacker state* $(h_i,\ psp) \in AS$ *will be reached on average y number of times in a certain time interval. Then* $(X(h_i,\ psp,\ h_j,\ vl_{j,k},\ ptp,\ ED(j,k)) \mid Y(h_i,\ psp)) \sim Po(k \cdot y)$.

Attacker state random variables can be implied by one or more attack random variables. Let the set of attacks $ap \in AP$ that imply the attacker state $as \in AS$ be given by the set $PRE(as \in AS)$. If an attack $ap \in AP$ that implies the attacker state $as \in AS$ is launched successfully, the attacker state $as \in AS$ is obtained with probability one. Since attack random variables of the attacks in $PRE(as \in AS)$ have Poisson distributions, the attacker state random variable implied by the attack random variables of the attacks in $PRE(as \in AS)$ gets a Poisson distribution that is the sum of the Poisson distributions of the attack random variables of the attacks in $PRE(as \in AS)$. A Poisson distribution that is the sum of other Poisson distributed variables is defined in the following way:

Definition 4.10 (Sum of Poisson distributed variables). *If* $X_i \sim Po(\lambda_i)$, $i = 1,\ \ldots, n$ *are independent, and* $\lambda = \sum_{i=1}^{n} \lambda_i$, *then* $Y = \sum_{i=1}^{n} X_i \sim Po(\lambda)$.

Based on the definition of the sum of Poisson distributed variables, attacker state random variables are defined in the following way:

Definition 4.11 (Discrete conditional probability distributions for attacker state random variables $Y(h_j,\ ptp)$**).** *Let the set of attacks that imply the attacker state* $(h_j, ptp) \in AS$ *be given by the set* $PRE((h_j, ptp) \in AS)$. *In the probabilistic attack graph, the attacks in* $PRE((h_j, ptp) \in AS)$ *are the parents of* $(h_j, ptp) \in AS$. *Let* $X_1,\ X_2,\ \ldots,\ X_n$ *be the attack random variables of the attacks in* $PRE((h_j, ptp) \in AS)$. $X_i \sim Po(\lambda_i),\ i = 1,\ \ldots,\ n$ *and let* $\lambda = \sum_{i=1}^{n} \lambda_i$. *Then* $(Y(h_j,\ ptp) \mid X_1,\ X_2,\ \ldots,\ X_n) = (\sum_{i=1}^{n} X_i) \sim Po(\lambda)$.

In a small example of a probabilistic attack graph in figure 4.1 it is shown how both attack and attacker state discrete random variables have discrete conditional probability distributions that are conditioned on the discrete probability distributions of the parents.

4.2.5 Calibrating the discrete conditional probability distributions $P(X(h_i, psp, h_j, vl_{j,k}, ptp, ED(j,k)) \mid Y(h_i, psp))$ of attacks with information from IDS devices

We recall from definition 4.9 that an attack random variable has a conditional probability distribution $(X(h_i,\ psp,\ h_j,\ vl_{j,k},\ ptp,\ ED(j,k)) \mid Y(h_i,\ psp)) \sim Po(k \cdot y)$. The parameter k is the average probability that an attack will be successful given that the attack preconditions are satisfied. It is obvious that the value of the probability parameter k depends on the skill of the attacker launching the attack. Of course the skill depends on the attacker and the

probability parameter k should reflect the skill of the average attacker.

Given this logic, the question is, how do we know the different values of k for attack random variable conditional probability distributions? We know that the success of an attack depends on the exploit difficulty value ED(i,j) of the vulnerability that is being attacked. A high exploit difficulty value should affect the probability parameter k downwards. This means that the value k should correspond to an exploit difficulty value ED(i,j), a given exploit difficulty value always implies the same probability parameter k and also the higher exploit difficulty value ED(i,j) the lower the probability parameter k. But this still doesn't tell us what a good estimate of k is exactly.

Here, historical data of attack intensity from IDS devices can help us to get the right expected value of attack intensity $E(X(h_i,\ psp,\ h_j,\ vl_{j,k},\ ptp,\ ED(j,k)) \mid Y(h_i,\ psp))$. We recall from section 2.6.1 that the function IDSf(i, f, g) gives a count of the estimated historical average number of attacks from host $h_i \in H$ on vulnerability $vl_{f,g} \in VL$ on host $h_f \in H$ over a given time interval. Since the attack precondition privilege levels are given by the set $PSP = \{user, admin\}$, there are two attacker states, $(h_i, user) \in AS$ and $(h_i, admin) \in AS$ that give an attacker the ability to launch attacks from a host $h_i \in H$. Thus, the expected number of attacks from host $h_i \in H$ on vulnerability $vl_{f,g} \in VL$ on host $h_f \in H$ is given by the expected value $E(X(h_i,\ user,\ h_f,\ vl_{f,g},\ ptp,\ ED(f,g)) + X(h_i,\ admin,\ h_f,\ vl_{f,g},\ ptp,\ ED(f,g)))$.

We have established that the attack probability parameter k corresponds to an exploit difficulty value ED(f, g), where a higher exploit difficulty value ED(f, g) implies a lower attack probability value k. Let the set of attack probability parameters k be given by the set $K = \{k_1,\ k_2,\ \ldots,\ k_{12}\}$, where $k_1 < k_2 < \ldots < k_{12}$, there are twelve attack probability parameter values since each attack probability parameter value $k \in K$ corresponds to an exploit difficulty value and there are twelve different exploit difficulty values in R(ED). Here k_1 corresponds to the highest exploit difficulty value, k_2 corresponds to the next highest exploit difficulty value and so on until k_{12} that corresponds to the lowest exploit difficulty value. We do not know the attack probability parameter values $k \in K$, but the probability parameters $k \in K$ must be chosen so that the expected number of attacks $E(X(h_i, user, vl_{f,g}, h_f, ptp, ED(f,g)) + X(h_i, admin, vl_{f,g}, h_f, ptp, ED(f,g)))$ from host $h_i \in H$ on vulnerability $vl_{f,g} \in VL$ on host $h_f \in H$ is as close as possible to historical data on attack intensity IDSf(i, f, g) from IDS devices where IDS data is available. This can be done by testing many different values for all $k \in K$, and in that way see which combination of values gives the best fit with historical data from IDS devices. Different values of the probability parameters $k \in K$ as long as the probability parameters $k \in K$ remain in the right order can be tested by using Monte Carlo methods, where each probability $k \in K$ has a defined domain of possible probability values and these values are generated randomly from a probabilistic distribution over the domain.

Let the expected number of attacks function

$$APE(i,\ f,\ g) = E(X(h_i,\ user, h_f,\ vl_{f,g},\ ptp,\ ED(f,g))) +$$

$$E(X(h_i,\ admin,\ h_f,\ vl_{f,g},\ ptp,\ ED(f,g))) \tag{4.3}$$

For the network vulnerability data set VD, the attack probabilities $k \in K$ must be chosen so that the function

$$f1 = \sum_{IDSf(i,f,g) \in IDSS} (IDSf(i,\ f,\ g) - APE(i,\ f,\ g))^2 \ is\ minimized \tag{4.4}$$

$while\ k_1 < k_2,\ k_2 < k_3,\ k_3 < k_4,\ k_4 < k_5,\ k_5 < k_6,\ k_6 < k_7,\ k_7 < k_8,\ k_8 < k_9, k_9 < k_{10},\ k_{10} < k_{11},\ k_{11} < k_{12}$

In this way the probability parameters $k \in K$ are calibrated so that the expected number of attacks values APE(i, f, g) correspond as closely as possible to the historical data from the IDS device.

We show what the minimization function value is for a small example probabilistic attack graph in figure 4.1. In this example the set IDSS consists of the following IDSf(i,f,g) function values: $IDSS = \{IDSf(0,1,1) = 7/year,\ IDSf(0,1,2) = 5/year,\ IDSf(0,1,3) = 4/year\}$. Here k_4, k_6 and k_9 must be chosen so that the function

$$f_1 = \sum_{IDSf(i,f,g) \in IDSS} (IDSf(i,f,g) - APE(i,f,g))^2 =$$

$$= (IDSf(0,1,1) - E(X(h_0,\ admin,\ h_1,\ vl_{1,1},\ user,\ 5.42)))^2 +$$

$$(IDSf(0,1,2) - E(X(h_0,\ admin,\ h_1,\ vl_{1,2},\ user,\ 6.67)))^2 +$$

$$(IDSf(0,1,3) - E(X(h_0,\ admin,\ h_1,\ vl_{1,3},\ user,\ 7.50)))^2 =$$

$$= (7 - k_9 \cdot \lambda)^2 + (5 - k_6 \cdot \lambda)^2 + (4 - k_4 \cdot \lambda)^2\ is\ minimized$$

$$while\ k_4 < k_6,\ k_6 < k_9$$

4.2.6 Computing optimized order of vulnerability patching to mitigate monetary loss from cyber attacks

While the exploit dependency attack graph shows the relationship between possible attacks $ap \in AP$ and possible attacker states $as \in AS$, it shows what is possible without any quantification of these possibilities, the probabilistic attack graph quantifies the expected number of times these attacks $ap \in AP$ are launched and the number of times the attacker states $as \in AS$ are reached. By disabling a vulnerability $vl_{f,g} \in VL$, and thereby removing it from the network

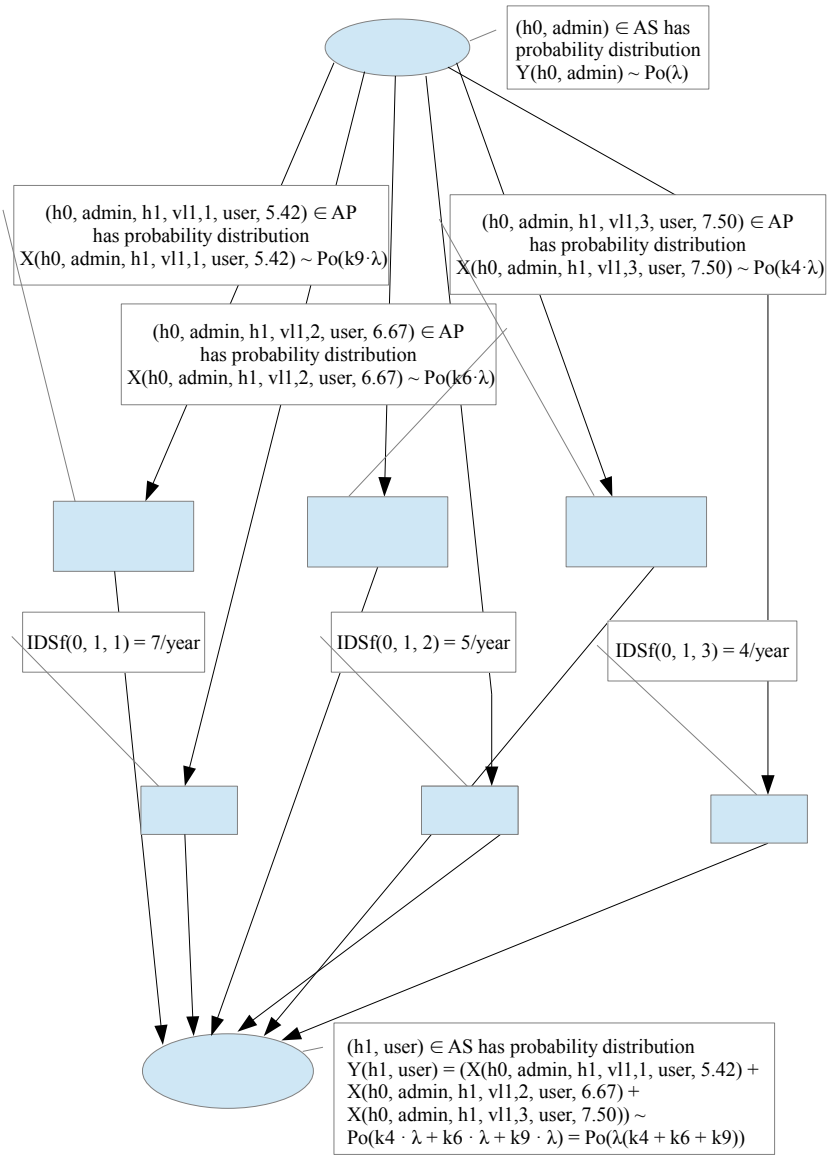


Figure 4.1: In this small example of a probabilistic attack graph it is shown how both attacks and attacker state random variables have discrete probability distributions that are conditioned on the discrete probability distributions of the parents. The probability parameter values k_4, k_6 and k_9, each corresponding to an exploit difficulty value must be calibrated with data from IDS devices.

vulnerability data set VD, it is possible to compute how $E(Y(h_i, ptp))$ changes for the different attacker state random variables in the network. If each possible attacker state random variable in the network gets a monetary loss value, a single loss expectancy, it also becomes possible to compute the annual loss expectancy for the network, ALE(network). Let the single loss expectancy if an attacker reaches an attacker state $(h_i, ptp) \in AS$ be given by $SLE((h_i, ptp) \in AS)$. The annualized rate of occurrence for an attacker to reach attacker state $as \in AS$ is given by $E(Y(h_i, ptp))$. Thus,

$$ALE(network) = \sum_{(h_i,ptp)\in AS} SLE((h_i, ptp) \in AS) * E(Y(h_i, ptp)) \tag{4.5}$$

Here, the network security administrator must give an estimate of $SLE((h_i, ptp) \in AS)$ for each attacker state $(h_i, ptp) \in AS$. By patching one vulnerability $vl_{f,g} \in VL$ at a time, thereby disabling the corresponding attacks $ap \in AP$ on that vulnerability $vl_{f,g} \in VL$, it is possible to compute how the patch affects the expected number of times each attacker state $as \in AS$ in the network is reached $E(Y(h_i, ptp))$ by an attacker. In this way it is possible to compute which patch affects ALE(network) most, giving the network security administrator the ability to know with mathematical certainty which vulnerabilities carry the biggest monetary loss on the network. At the same time, ALE(network) quantifies how much money it is worth spending on security measures to mitigate monetary loss. In this way it is possible to compute a priority list of vulnerabilities to patch and how much money it is worth spending on network security ALE(network) only by knowing the network vulnerability data set VL.

4.3 Mathematical representation of network vulnerability data

The probabilistic attack graph is solely determined by the network vulnerability data set $VD = VP \cup C \cup IDSS$. To enable the computation of the probabilistic attack graph and be able to compute the probability distribution of each attacker state $as \in AS$ in the probabilistic attack graph, the network vulnerability data set of a given network must be transformed into numerical form. Since vulnerability descriptions $VP_{i,j}$ are sets of tuples and vulnerable data connectivity relations $c \in C$ are tuples, it is easy to transform these into vector form. This is done by simply transforming the sets in the Cartesian products that the vulnerability data tuples $vd \in VD$ are members of into numerical form. Then the corresponding Cartesian product that the network vulnerability data tuples are members of become transformed into sets of tuples in numerical form. After this stage a tuple is obtained where all elements are numbers, this tuple is then transformed into vector form. A set that is transformed into numerical form will simply get the N character at the end of the set name meaning that for

example the numerical representation of the PTP set is denoted by PTPN. Network vulnerability data sets that are in vector form have a V at the end of the set name. For example a vulnerability description set $VP_{i,j}$ that is transformed into vector form gets the name $VPV_{i,j}$.

4.3.1 Transformation of vulnerability description tuples $vp_{i,j} \in VP_{i,j}$ into vector form

From definition 4.2 we know that a vulnerability description element $vp_{i,j} \in VP_{i,j}$ is a 5-tuple and a member of the following Cartesian product:

$$vp_{i,j} \in H \times L \times PSP \times PTP \times R(ED)$$

H is the set of hosts in the network and hosts are enumerated where $h_i \in H$ denotes host number i in the network, thus the numerical representation of host $h_i \in H$ is $i \in HN$. To denote the transformation of sets into numerical representations of sets or the transformation of set elements into numerical representation of set elements the imply symbol $\longrightarrow$ is used, for example $h_i \in H \longrightarrow i \in HN$.

The first element in the numerical form of a network vulnerability data tuple $vd \in VD$ is given by the set named type denoted by T that gives the type of computer network vulnerability data used, $T = \{vp,\ c,\ ids,\ ap,\ as\}$ where vp denotes that it is a vulnerability description $vp \in VP$ data type, c denotes that it is a vulnerable data connectivity relation $c \in C$ data type, ids denotes that it is an $IDSf(i, f, g) \in IDSS$ value data type, ap denotes that it is an attack $ap \in AP$ data type and as denotes that it is an attacker state $as \in AS$ data type. The type set transformation $T \longrightarrow TN$ is given by:

$vp \in T \longrightarrow 1 \in TN$
$c \in T \longrightarrow 2 \in TN$
$ids \in T \longrightarrow 3 \in TN$
$ap \in T \longrightarrow 4 \in TN$
$as \in T \longrightarrow 5 \in TN$

The host set transformation $H \longrightarrow HN$ is given by:

$h_i \in H \longrightarrow i \in HN$

The locality set transformation $L \longrightarrow LN$ is given by:

$local \in L \longrightarrow 1 \in LN$
$remote \in L \longrightarrow 2 \in LN$

The attack precondition privilege level set transformation $PSP \longrightarrow PSPN$ is given by:

"$user$" $\in PSP \longrightarrow 1 \in PSPN$
"$admin$" $\in PSP \longrightarrow 2 \in PSPN$

The attack postcondition privilege level set transformation $PTP \longrightarrow PTPN$ is given by:

"$user$" $\in PTP \longrightarrow 1 \in PTPN$
"$admin$" $\in PTP \longrightarrow 2 \in PTPN$
"DoS" $\in PTP \longrightarrow 3 \in PTPN$
"$C:N-I:N-A:P$" $\in PTP \longrightarrow 4 \in PTPN$
"$C:N-I:P-A:N$" $\in PTP \longrightarrow 5 \in PTPN$
"$C:N-I:P-A:P$" $\in PTP \longrightarrow 6 \in PTPN$
"$C:N-I:P-A:C$" $\in PTP \longrightarrow 7 \in PTPN$
"$C:N-I:C-A:N$" $\in PTP \longrightarrow 8 \in PTPN$
"$C:N-I:C-A:P$" $\in PTP \longrightarrow 9 \in PTPN$
"$C:N-I:C-A:C$" $\in PTP \longrightarrow 10 \in PTPN$
"$C:P-I:N-A:N$" $\in PTP \longrightarrow 11 \in PTPN$
"$C:P-I:N-A:P$" $\in PTP \longrightarrow 12 \in PTPN$
"$C:P-I:N-A:C$" $\in PTP \longrightarrow 13 \in PTPN$
"$C:P-I:P-A:N$" $\in PTP \longrightarrow 14 \in PTPN$
"$C:P-I:P-A:P$" $\in PTP \longrightarrow 15 \in PTPN$
"$C:P-I:P-A:C$" $\in PTP \longrightarrow 16 \in PTPN$
"$C:P-I:C-A:N$" $\in PTP \longrightarrow 17 \in PTPN$
"$C:P-I:C-A:P$" $\in PTP \longrightarrow 18 \in PTPN$
"$C:P-I:C-A:C$" $\in PTP \longrightarrow 19 \in PTPN$
"$C:C-I:N-A:N$" $\in PTP \longrightarrow 20 \in PTPN$
"$C:C-I:N-A:P$" $\in PTP \longrightarrow 21 \in PTPN$
"$C:C-I:N-A:C$" $\in PTP \longrightarrow 22 \in PTPN$
"$C:C-I:P-A:N$" $\in PTP \longrightarrow 23 \in PTPN$
"$C:C-I:P-A:P$" $\in PTP \longrightarrow 24 \in PTPN$
"$C:C-I:P-A:C$" $\in PTP \longrightarrow 25 \in PTPN$
"$C:C-I:C-A:N$" $\in PTP \longrightarrow 26 \in PTPN$
"$C:C-I:C-A:P$" $\in PTP \longrightarrow 27 \in PTPN$

In the exploit difficulty range set R(ED) the values are already numerical and thus this set is already in numerical form and doesn't have to be transformed.

We introduce a set VLN that gives the number of the vulnerability $vl_{i,j} \in VL$, it gives the number j of a vulnerability $vl_{i,j} \in VL$ on host $h_i \in H$, $VLN \subseteq \mathbb{N}$.

The numerical representation of a vulnerability description element $vpn_{f,g} \in VPN_{f,g}$ is a member of the 7-ary Cartesian product over the 7 sets, $TN = \{1, 2, 3, 4, 5\}$, $HN \subseteq \mathbb{N}$, $VLN \subseteq \mathbb{N}$, $LN = \{1, 2\}$, $PSPN = \{1, 2\}$, $PTPN = [1,\ 27] \cap \mathbb{N}$ and $R(ED) \subseteq [35/12,\ 10]$:

$vpn_{f,g} \in TN \times HN \times VLN \times LN \times PSPN \times PTPN \times R(ED)$

To exemplify the transformation of a vulnerability description element $vp_{i,j} \in VP_{i,j}$ into its numerical representation $vpn_{i,j} \in VPN_{i,j}$ and then into vector form $vpv_{i,j} \in VPV_{i,j}$ we give two examples:

$(h_1,\ remote,\ admin,\ admin,\ 5.83) \in VP_{1,2} \longrightarrow (1,\ 1,\ 2,\ 2,\ 2,\ 2,\ 5.83) \in VPN_{1,2} \longrightarrow [1\ 1\ 2\ 2\ 2\ 2\ 5.83] \in VPV_{1,2}$

$(h_5,\ remote,\ admin,\ 'C : NI : CA : C',\ 8.33) \in VP_{5,3} \longrightarrow (1,\ 5,\ 3,\ 2,\ 2,\ 10,\ 8.33) \in VPN_{5,3} \longrightarrow [1\ 5\ 3\ 2\ 2\ 10\ 8.33] \in VPV_{5,3}$

4.3.2 Transformation of vulnerable data connectivity relation tuples $c \in C$ into vector form

From definition 2.3 we know that a vulnerable data connectivity relation $c_i \in C_i$ is a 3-tuple and a member of the following Cartesian product:

$c_i \in H_i \times H \setminus H_i \times VL \setminus VL_i$

The one-element host set transformation $H_i \longrightarrow HN_i$ is given by:

$h_i \in H_i \longrightarrow i \in HN_i$

The host set transformation $H \setminus Hi \longrightarrow HN \setminus HN_i$ is given by:

$h_f \in H \setminus H_i \longrightarrow f \in HN \setminus HN_i$

The vulnerability set transformation $VL \setminus VL_i \longrightarrow VLN \setminus VLN_i$ is given by:

$vl_{f,g} \in VL \setminus VL_i \longrightarrow g \in VLN \setminus VLN_i$

We have that $HN_i = \{i\},\ HN \setminus HN_i \subseteq \mathbb{N} \setminus HN_i$ and $VLN \setminus VLN_i \subseteq \mathbb{N}$.

The numerical representation of a vulnerable data connectivity relation $c_i \in C_i$ is a member of the 4-ary Cartesian product over the 4 sets, $TN = \{1,\ 2,\ 3\},\ HN_i = \{i\},\ HN \setminus HN_i \subseteq \mathbb{N} \setminus HN_i$ and $VLN \setminus VLN_i \subseteq \mathbb{N}$:

$cn_i \in TN \times HN_i \times HN \setminus HN_i \times VLN \setminus VLN_i$

To exemplify the transformation of a vulnerable data connectivity relation $c_i \in C_i$ into its numerical representation $cn_i \in CN_i$ and then into vector form $cv_i \in CV_i$ we give two examples:

$(h_1,\ h_5,\ vl_{5,4}) \in C_1 \longrightarrow (2,\ 1,\ 5,\ 4) \in CN_1 \longrightarrow [2\ 1\ 5\ 4] \in CV_1$

$(h_2,\ h_3,\ vl_{3,3}) \in C_2 \longrightarrow (2,\ 2,\ 3,\ 3) \in CN_2 \longrightarrow [2\ 2\ 3\ 3] \in CV_2$

4.3.3 Transformation of IDSf(i, f, g) values into vector form

The numerical representation of an IDS value $IDSf(i, f, g) = 4/year \in IDSS$ is given by the tuple $(3, i, f, g, 4) \in IDSSN$ where the first element denotes that it is an IDS type of data tuple and the fifth element gives the number of attacks in a given historical period of time. To exemplify the transformation of an IDS value $IDSf(i, f, g) = y/year$ into its numerical representation $idsn \in IDSSN$ and then into vector form $idsv \in IDSSV$ we give two examples:

$IDSf(2,\ 5,\ 3) = 5/year \in IDSS \longrightarrow (3,\ 2,\ 5,\ 3,\ 5) \in IDSSN \longrightarrow [3\,2\,5\,3\,5] \in IDSSV$

$IDSf(4,\ 2,\ 2) = 3/year \in IDSS \longrightarrow (3,\ 4,\ 2,\ 2,\ 3) \in IDSSN \longrightarrow [3\,4\,2\,2\,3] \in IDSSV$

4.3.4 Mathematical representation of a probabilistic attack graph

A probabilistic attack graph is solely determined by the set of possible attacker states AS and set of possible probabilistic attacks AP, implied by the network vulnerability data set VD according to definition 4.4 and 4.5. Therefore a probabilistic attack graph can be represented mathematically by representing the set of probabilistic attacks AP and attacker states AS in mathematical form.

4.3.5 Transformation of attacks $ap \in AP$ into vector form

From definition 4.3 we know that a probabilistic attack is a 6-tuple and a member of the following Cartesian product:

$ap_i \in H_i \times PSP \times HT_i \times VT_i \times PTP \times R(ED)$

The transformations of the sets PSP, PTP, R(ED) into numerical form is given in section 4.2.6. The one-element host set transformation $H_i \longrightarrow HN_i$ is given by:

$h_i \in H_i \longrightarrow i \in HN_i$

The host set transformation $HT_i \longrightarrow HTN_i$ is given by:

$h_f \in HT_i \longrightarrow f \in HTN_i$

The vulnerability set transformation $VT_i \longrightarrow VTN_i$ is given by:

$vl_{f,g} \in VT_i \longrightarrow g \in VTN_i$

The first element of an attack in numerical form is given by the type set TN. To be able to represent the probabilistic attack graph in numerical form it is necessary to define in which layer in the probabilistic attack graph the attacks $ap \in AP$ and attacker states $as \in AS$ are situated, therefore the second element of an attack in numerical form $apn \in APN$ is given by the layer set LA that gives the layer of attack $ap \in AP$ in the probabilistic attack graph. To know the expected number of times that an attack $ap \in AP$ is launched over a given period of time, the ninth element in an attack $ap \in AP$ in numerical form $apn \in APN$ is given by the set EAP which gives the expected number of times an attack $ap \in AP$ is launched in a given period of time. In this way the probability distribution $P(X(h_i, psp, h_j, vl_{j,k}, ptp, ED(j,k)))$ of the attack $(h_i, psp, h_j, vl_{j,k}, ptp, ED(j,k)) \in AP$ is given, since $E(X(h_i, psp, h_j, vl_{j,k}, ptp, ED(j,k)) = \lambda$ and $X(h_i, psp, h_j, vl_{j,k}, ptp, ED(j,k)) \sim Po(\lambda)$. Thus, the numerical representation of an attack $ap \in AP$ is a stochastic variable with a given probability distribution and expected value of an attack $ap \in AP$. It is a member of the 9-ary Cartesian product over the 9 sets, $TN = \{1,\ 2,\ 3,\ 4,\ 5\}, LA \subseteq \mathbb{N},\ HN_i \subseteq \mathbb{N},\ PSPN = \{1,2\}, HTNi \subseteq \mathbb{N},\ VTN_i \subseteq \mathbb{N},\ PTPN = [1,27] \cap \mathbb{N},\ R(ED) \subseteq [35/12, 10]$ and $EAP \subseteq \mathbb{R}+$:

$$apn_i \in TN \times LA \times HNi \times PSPN \times HTNi \times VTNi \times PTPN \times R(ED) \times EAP$$

To exemplify the transformation of an attack $ap \in AP$ into its numerical representation $apn \in APN$ and then into vector form $apv \in APV$ we give two examples:

$(h_i,\ user,\ h_f,\ vl_{f,g},\ "C:N-I:C-A:N",\ 5.83) \in AP_i \longrightarrow$
$(4,\ x,\ i,\ 1,\ f,\ g,\ 8,\ 5.83,\ y) \in APNi \longrightarrow [4\ x\ i\ 1\ f\ g\ 8\ 5.83\ y] \in APV_i$

$(h_i,\ admin,\ h_f,\ vl_{g,f},\ "C:C-I:N-A:P", 6.25) \in AP_i \longrightarrow$
$(4,\ x,\ i,\ 1,\ f,\ g,\ 21,\ 6.25,\ y) \in APN_i \longrightarrow [4\ x\ i\ 1\ f\ g\ 21\ 6.25\ y] \in APV_i$

4.3.6 Transformation of attacker states $as \in AS$ into vector form

From definition 2.4 we know that an attacker state is a double and a member of the following Cartesian product:

$$as \in H \times PTP$$

The transformation of the postcondition privilege level set PTP and vulnerable host set H is given in section 4.2.6. The first element in the attacker state AS in numerical form is given by the type set T and the second element is given by the layer set LA that gives the layer of attacker state $as \in AS$ in the probabilistic attack graph. The fifth element is given by the set EAS which gives the

expected number of times an attacker state $as \in AS$ is reached in a given period of time. In this way the probability distribution $P(Y(h_i, ptp))$ of the attacker state $(h_i, ptp) \in AS$ is known since $E(Y(h_i, ptp)) = \lambda$ and $Y(h_i, ptp) \sim Po(\lambda)$.

Thus, the numerical representation of an attacker state $as \in AS$ is a stochastic variable with a given probability distribution and expected value of an attacker state $as \in AS$. It is a member of the 5-ary Cartesian product over the 5 sets, $TN = \{1,\ 2,\ 3,\ 4,\ 5\}$, $LA \subseteq \mathbb{N}$, $HN \subseteq \mathbb{N}$, $PTPN = [1, 27] \cap \mathbb{N}$ and $EAS \subseteq \mathbb{R}+$:

$asn \in TN \times LA \times HN \times PTPN \times EAS$

To exemplify the transformation of an attacker state $as \in AS$ into its numerical representation $asn \in ASN$ and then into vector form $asv \in ASV$ we give two examples:

$(h_i,\ DoS) \in AS \longrightarrow (5,\ x,\ i,\ 3,\ y) \in ASN \longrightarrow [5\ x\ i\ 3\ y] \in ASV$

$(h_i,\ 'C : P-I : N-A : P') \in AS \longrightarrow (5,\ x,\ i,\ 12,\ y) \in ASN \longrightarrow [5\ x\ i\ 12\ y] \in ASV$

4.3.7 Implying possible attacks $ap \in AP$ and attacker states $as \in AS$ from network vulnerability data in numerical form

Using the numerical representation of network vulnerability data, attacks $ap \in AP$ and attacker states $as \in AS$, we give one example each of remote attacks $ar_i \in AR_i$ and attacker states $as \in AS$ implied from the network vulnerability data set VD, attacks $ap_i \in AP_i$ and attacker states $as \in AS$ in numerical form. Using regular representation of vulnerable data connectivity relation $c_i \in C_i$, vulnerability description element $vp_{f,g} \in VP_{f,g}$ and attacker states $as \in AS$ for implying a remote attack:

$(h_i,\ user) \in AS \wedge (h_i,\ h_f,\ vl_{f,g}) \in C_i \wedge (h_f,\ remote,\ user,\ 'C : P - I : C - A : N',\ 6.25) \in VP_{f,g} \longrightarrow (h_i,\ user,\ h_f,\ vl_{f,g},\ 'C : P - I : C - A : N', 6.25, y) \in AR_i$

Using numerical representation for implying the same remote attack $ar_i \in AR_i$ in numerical form $arn_i \in ARN_i$:

$(5,\ x,\ i,\ 1,\ y) \in ASN \wedge (2,\ i,\ f,\ g) \in CN_i \wedge (1,\ f,\ 2,\ 1,\ 17,\ 6.25) \in VPN_{f,g} \longrightarrow (4,\ x,\ i,\ 1,\ f,\ g,\ 17,\ 6.25,\ y) \in ARN_i$

Using regular representation of attacks $ap_i \in AP_i$ for implying an attacker state $as \in AS$:

$(h_i,\ user,\ h_f,\ vl_{f,g},\ 'C : P - I : C - A : N',\ 6.25) \in AP_i \longrightarrow (h_f,\ 'C : P - I : C - A : N') \in AS$

Using numerical representation for implying the same attacker state $as \in AS$ in numerical form $asn \in ASN$:

$(4,\ i,\ 1,\ f,\ g,\ 17,\ 6.25) \in APNi \longrightarrow (5,\ x,\ f,\ 17,\ y) \in ASN$

Chapter 5

Probabilistic attack graph for an invented example network

5.1 Network vulnerability data in vector form for an invented example network

We will test our method by computing the expected number of times each attacker state $as \in AS$ is reached in a year in an invented example network. The graphical representation of the example network is given in figure 5.1. We will use sections 4.3.1 - 4.3.3 to transform the vulnerable data set $VD = VP \cup C \cup IDSS$ from the invented example network to vector form and use this data to generate the probabilistic attack graph and ALE(network).

Transformation of vulnerability description tuples $vp_{i,j} \in VP_{i,j}$ in the invented example network into vector form $vpv_{i,j} \in VPV_{i,j}$:

$(h_1,\ remote,\ admin,\ 'C : P{-}I : C{-}A : N',\ 7.08) \in VP_{1,1} \longrightarrow [1\ 1\ 1\ 2\ 2\ 17\ 7.08] \in VPV_{1,1}$
$(h_1,\ remote,\ admin,\ admin,\ 5.83) \in VP_{1,2} \longrightarrow [1\ 1\ 2\ 2\ 2\ 2\ 5.83] \in VPV_{1,2}$
$(h_2,\ remote,\ admin,\ 'C : P{-}I : P{-}A : P',\ 7.5) \in VP_{2,1} \longrightarrow [1\ 2\ 1\ 2\ 2\ 15\ 7.5] \in VPV_{2,1}$
$(h_2,\ remote,\ admin,\ 'C : P{-}I : N{-}A : C',\ 4.58) \in VP_{2,2} \longrightarrow [1\ 2\ 2\ 2\ 2\ 13\ 4.58] \in VPV_{2,2}$
$(h_2,\ remote,\ admin,\ user,\ 6.25) \in VP_{2.3} \longrightarrow [1\ 2\ 3\ 2\ 2\ 1\ 6.25] \in VPV_{2,3}$
$(h_3,\ remote,\ admin,\ user,\ 5.42) \in VP_{3.1} \longrightarrow [1\ 3\ 1\ 2\ 2\ 1\ 5.42] \in VPV_{3,1}$
$(h_3,\ remote,\ admin,\ 'C : N{-}I : C{-}A : C',\ 8.33) \in VP_{3,2} \longrightarrow [1\ 3\ 2\ 2\ 2\ 10\ 8.33] \in VPV_{3,2}$
$(h_4,\ remote,\ admin,\ admin,\ 2.92) \in VP_{4,1} \longrightarrow [1\ 4\ 1\ 2\ 2\ 2\ 2.92] \in VPV_{4,1}$
$(h_4,\ remote,\ admin,\ 'C : C{-}I : N{-}A : P', 4.58) \in VP_{4,2} \longrightarrow [1\ 4\ 2\ 2\ 2\ 21\ 4.58] \in$

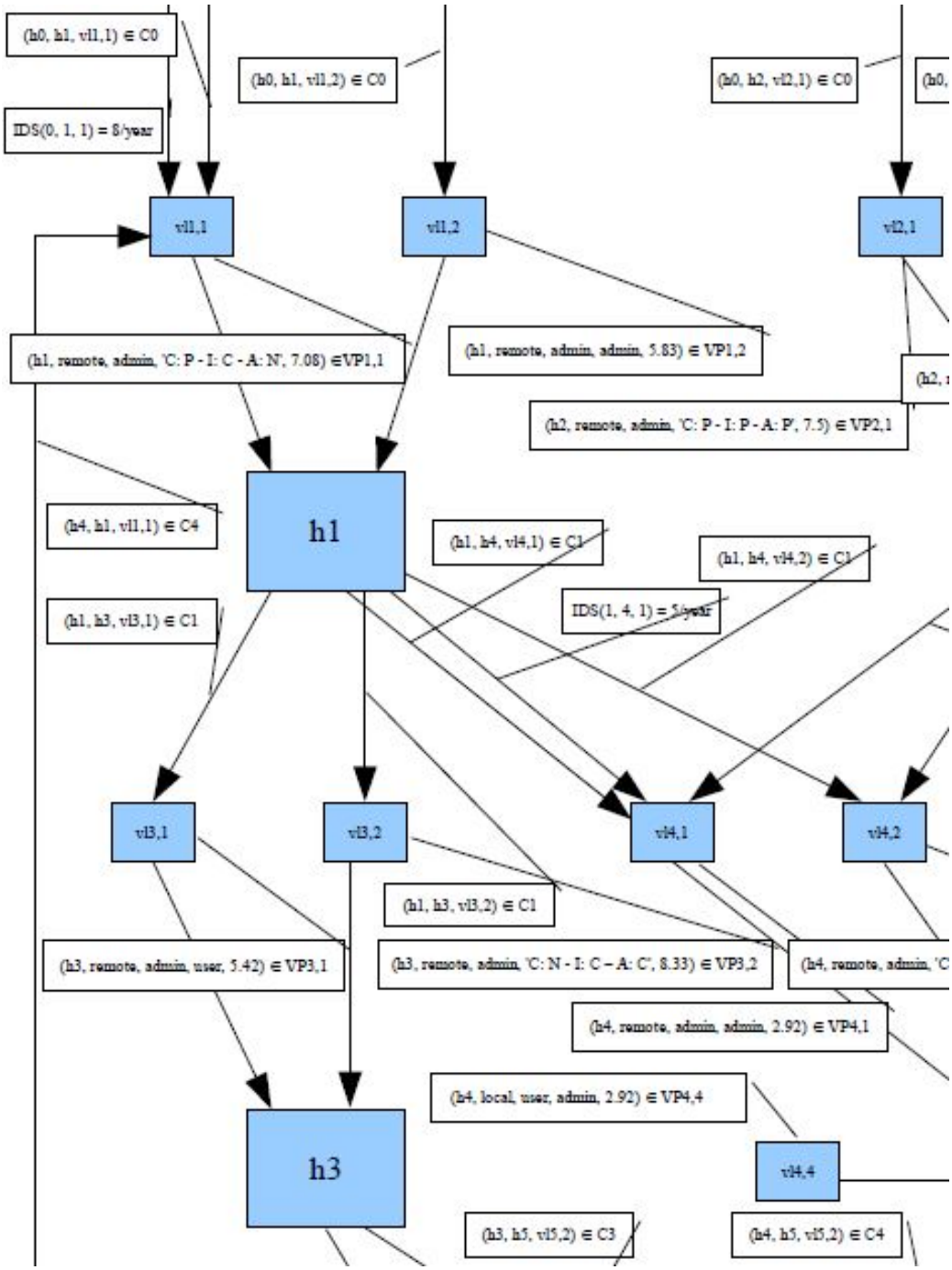


Figure 5.1: Part of a graphical representation of network vulnerability data for an invented example network

$VPV_{4,2}$
$(h_4,\ remote,\ admin,\ user,\ 8.33) \in VP_{4,3} \longrightarrow [1\ 4\ 3\ 2\ 2\ 1\ 8.33] \in VPV_{4,3}$
$(h_4,\ local,\ user,\ admin,\ 2.92) \in VP_{4,4} \longrightarrow [1\ 4\ 4\ 1\ 2\ 1\ 2.92] \in VPV_{4,4}$
$(h_5,\ remote,\ admin,\ DoS,\ 4.17) \in VP_{5,1} \longrightarrow [1\ 5\ 1\ 2\ 2\ 3\ 4.17] \in VPV_{5,1}$
$(h_5,\ remote,\ admin,\ admin,\ 7.08) \in VP_{5,2} \longrightarrow [1\ 5\ 2\ 2\ 2\ 2\ 7.08] \in VPV_{5,2}$
$(h_5,\ remote,\ admin,\ 'C:C{-}I:P{-}A:N',\ 4.58) \in VP_{5,3} \longrightarrow [1\ 5\ 3\ 2\ 2\ 23\ 4.58] \in VPV_{5,3}$

Transformation of vulnerable data connectivity relations $c \in C$ of the invented example network into vector form $cv \in CV$:

$(h_0,\ h_1,\ vl_{1,1}) \in C \longrightarrow [2\ 0\ 1\ 1] \in CV$
$(h_0,\ h_1,\ vl_{1,2}) \in C \longrightarrow [2\ 0\ 1\ 2] \in CV$
$(h_0,\ h_2,\ vl_{2,1}) \in C \longrightarrow [2\ 0\ 2\ 1] \in CV$
$(h_0,\ h_2,\ vl_{2,2}) \in C \longrightarrow [2\ 0\ 2\ 2] \in CV$
$(h_0,\ h_2,\ vl_{2,3}) \in C \longrightarrow [2\ 0\ 2\ 3] \in CV$
$(h_1,\ h_3,\ vl_{3,1}) \in C \longrightarrow [2\ 1\ 3\ 1] \in CV$
$(h_1,\ h_3,\ vl_{3,2}) \in C \longrightarrow [2\ 1\ 3\ 2] \in CV$
$(h_1,\ h_4,\ vl_{4,1}) \in C \longrightarrow [2\ 1\ 4\ 1] \in CV$
$(h_1,\ h_4,\ vl_{4,2}) \in C \longrightarrow [2\ 1\ 4\ 2] \in CV$
$(h_2,\ h_4,\ vl_{4,1}) \in C \longrightarrow [2\ 2\ 4\ 1] \in CV$
$(h_2,\ h_4,\ vl_{4,2}) \in C \longrightarrow [2\ 2\ 4\ 2] \in CV$
$(h_2,\ h_4,\ vl_{4,3}) \in C \longrightarrow [2\ 2\ 4\ 3] \in CV$
$(h_3,\ h_5,\ vl_{5,1}) \in C \longrightarrow [2\ 3\ 5\ 1] \in CV$
$(h_3,\ h_5,\ vl_{5,2}) \in C \longrightarrow [2\ 3\ 5\ 2] \in CV$
$(h_4,\ h_5,\ vl_{5,2}) \in C \longrightarrow [2\ 4\ 5\ 2] \in CV$
$(h_4,\ h_5,\ vl_{5,3}) \in C \longrightarrow [2\ 4\ 5\ 3] \in CV$
$(h_4,\ h_1,\ vl_{1,1}) \in C \longrightarrow [2\ 4\ 1\ 1] \in CV$

Transformation of $IDSf(i, f, g) \in IDSS$ values of the invented example network into vector form:

$IDSf(0,\ 1,\ 1) = 8/year \in IDSS \longrightarrow [3\ 0\ 1\ 1\ 8] \in IDSSV$
$IDSf(1,\ 4,\ 1) = 5/year \in IDSS \longrightarrow [3\ 1\ 4\ 1\ 5] \in IDSSV$
$IDSf(4,\ 5,\ 3) = 3/year \in IDSS \longrightarrow [3\ 4\ 5\ 3\ 3] \in IDSSV$

Using these transformations of the network vulnerability data set VD, the invented example network is described in mathematical form by the following network vulnerability data set in vector form VDV:

VDV = {[1 1 1 2 2 17 7.08], [1 1 2 2 2 2 5.83], [1 2 1 2 2 15 7.5], [1 2 2 2 2 13 4.58], [1 2 3 2 2 2 6.25], [1 3 1 2 2 1 5.42], [1 3 2 2 2 10 8.33], [1 4 1 2 2 2 2.92], [1 4 2 2 2 21 4.58], [1 4 3 2 2 1 8.33], [1 5 1 2 2 3 4.17], [1 5 2 2 2 2 7.08], [1 5 3 2 2 23 4.58], [2 0 1 1], [2 0 1 2], [2 0 2 1], [2 0 2 2], [2 0 2 3], [2 1 3 1], [2 1 3 2], [2 1 4 1], [2 1 4 2], [2 2 4 1], [2 2 4 2], [2 2 4 3], [2 3 5 1], [2 3 5 2], [2 4 5 2], [2 4 5 3], [3 0 1 1 8], [3 1 4 1 5], [3 4 5 3 3]}

5.2 Using the network vulnerability data set in vector form VDV to imply the probabilistic attack graph for the invented example network

By using sections 4.2.1, 4.2.2 and 4.3 we will use the network vulnerability data set in vector form VDV of the example network in section 5.1 to generate a probabilistic attack graph for an attacker that starts the attack from the internet. The probabilistic attack graph is represented mathematically by presenting the expected number of times attacks $ap \in AP$ can be launched and attacker states $as \in AS$ can be reached in a given period of time.

The probabilistic attack graph set in vector form PAV for a computer network is solely defined by the set of attacker states AS and set of attacks AP. To be able to compute the expected number of times each attacker state $as \in AS$ can be reached in the probabilistic attack graph we need to define the attack probabilities K. The range of the exploit difficulty function ED is given by the set R(ED), each value in R(ED) corresponds to an attack probability $k \in K$. The range of the exploit difficulty function ED is given by the exploit difficulty range set

R(ED) = {2.92, 4.17, 4.58, 5.42, 5.83, 6.25, 6.67, 7.08, 7.50, 8.33, 8.75, 10}

In our probabilistic attack graph for the invented example network in section 5.1 each exploit difficulty value ED(i, j) corresponds to an attack probability $k \in K$ of a successful attack in the following way:

k = (11 - ED(i,j))/10

Thus the attack probabilities set K in ascending order is given by:

K = {0.10, 0.23, 0.27, 0.35, 0.39, 0.43, 0.48, 0.52, 0.56, 0.64, 0.68, 0.81}

The attack probability values $k \in K$ are implied in the following way by the exploit difficulty values ED(i,j) for attacks $ap \in AP$:

$10 \in R(ED) \longrightarrow 0.10 \in K$
$8.75 \in R(ED) \longrightarrow 0.23 \in K$
$8.33 \in R(ED) \longrightarrow 0.27 \in K$
$7.50 \in R(ED) \longrightarrow 0.35 \in K$
$7.08 \in R(ED) \longrightarrow 0.39 \in K$
$6.67 \in R(ED) \longrightarrow 0.43 \in K$
$6.25 \in R(ED) \longrightarrow 0.48 \in K$
$5.83 \in R(ED) \longrightarrow 0.52 \in K$
$5.42 \in R(ED) \longrightarrow 0.56 \in K$

$4.58 \in R(ED) \longrightarrow 0.64 \in K$
$4.17 \in R(ED) \longrightarrow 0.68 \in K$
$2.92 \in R(ED) \longrightarrow 0.81 \in K$

With these attack probabilities that correspond to each possible exploit difficulty value ED(i,j) we can compute the average number of times each possible attacker state $as \in AS$ can be reached in the probabilistic attack graph. The probabilistic attack graph is depicted graphically in figure 4.2. The whole probabilistic attack graph for the invented computer network is implied mathematically below. Expected values given by the fifth value for attacker states in vector form and the ninth element for attacks in vector form give the expected number of attacks $ap \in AP$ launched and number of attacker states $as \in AS$ reached in a year by an attacker.

Layer 1 of possible attacks ap $\in AP$:

[5 1 0 1 10] $\wedge$[2 0 1 1] $\wedge$ [1 1 1 2 2 17 7.08] $\longrightarrow$ [4 1 0 1 1 1 17 7.08 3.920]
[5 1 0 1 10] $\wedge$ [2 0 1 2] $\wedge$ [1 1 2 2 2 2 5.83] $\longrightarrow$ [4 1 0 1 1 2 2 5.83 5.170]
[5 1 0 1 10] $\wedge$ [2 0 2 1] $\wedge$ [1 2 1 2 2 15 7.5] $\longrightarrow$ [4 1 0 1 2 1 15 7.5 3.500]
[5 1 0 1 10] $\wedge$ [2 0 2 2] $\wedge$ [1 2 2 2 2 13 4.58] $\longrightarrow$ [4 1 0 1 2 2 13 4.58 6.420]
[5 1 0 1 10] $\wedge$ [2 0 2 3] $\wedge$ [1 2 3 2 2 2 6.25] $\longrightarrow$ [4 1 0 1 2 3 2 6.25 4.750]

Layer 2 of possible attacker states $as \in AS$:

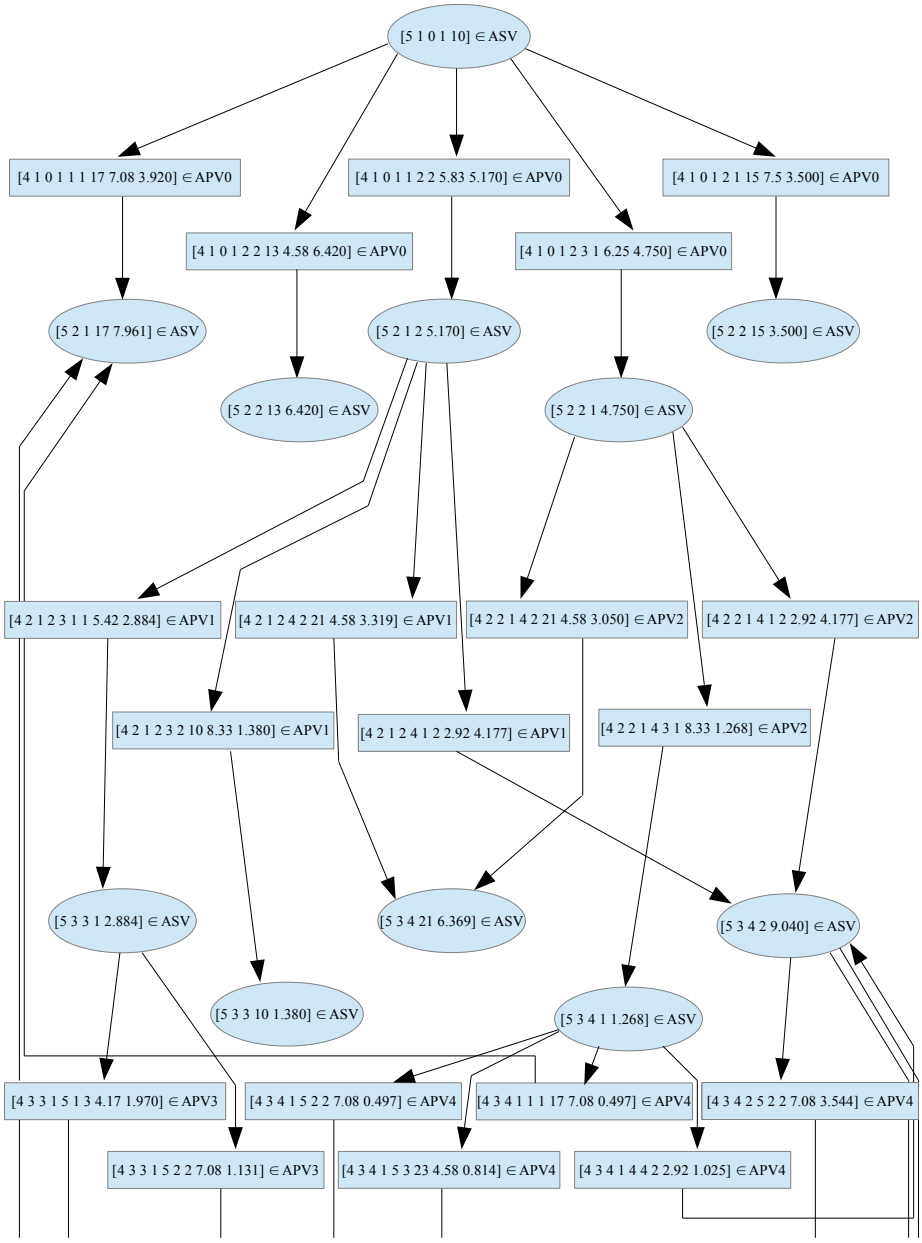
[4 1 0 1 1 1 17 7.08 3.920] $\longrightarrow$ [5 2 1 17 7.961]
[4 1 0 1 1 2 2 5.83 5.170] $\longrightarrow$ [5 2 1 2 5.170]
[4 1 0 1 2 1 15 7.5 3.500] $\longrightarrow$ [5 2 2 15 3.500]
[4 1 0 1 2 2 13 4.58 6.420] $\longrightarrow$ [5 2 2 13 6.420]
[4 1 0 1 2 3 2 6.25 4.750] $\longrightarrow$ [5 2 2 2 4.750]

Layer 2 of possible attacks $ap \in AP$:

[5 2 1 2 5.170] $\wedge$ [2 1 3 1] $\wedge$ [1 3 1 2 2 1 5.42] $\longrightarrow$ [4 2 1 2 3 1 1 5.42 2.884]
[5 2 1 2 5.170] $\wedge$ [2 1 3 2] $\wedge$ [1 3 2 2 2 10 8.33] $\longrightarrow$ [4 2 1 2 3 2 10 8.33 1.380]
[5 2 1 2 5.170] $\wedge$ [2 1 4 1] $\wedge$ [1 4 1 2 2 2 2.92] $\longrightarrow$ [4 2 1 2 4 1 2 2.92 4.177]
[5 2 1 2 5.170] $\wedge$ [2 1 4 2] $\wedge$ [1 4 2 2 2 21 4.58] $\longrightarrow$ [4 2 1 2 4 2 21 4.58 3.319]
[5 2 2 2 4.750] $\wedge$ [2 2 4 1] $\wedge$ [1 4 1 2 2 2 2.92] $\longrightarrow$ [4 2 2 2 4 1 2 2.92 3.838]
[5 2 2 2 4.750] $\wedge$ [2 2 4 2] $\wedge$ [1 4 2 2 2 21 4.58] $\longrightarrow$ [4 2 2 2 4 2 21 4.58 3.050]
[5 2 2 2 4.750] $\wedge$ [2 2 4 3] $\wedge$ [1 4 3 2 2 1 8.33] $\longrightarrow$ [4 2 2 2 4 3 1 8.33 1.268]

Layer 3 of possible attacker states $as \in AS$:

[4 2 1 2 3 1 1 5.42 2.884] $\longrightarrow$ [5 3 3 1 2.884]
[4 2 1 2 3 2 10 8.33 1.380] $\longrightarrow$ [5 3 3 10 1.380]
[4 2 1 2 4 1 2 2.92 4.177] $\longrightarrow$ [5 3 4 2 9.040]


[5 1 0 1 10] ∈ ASV
[4 1 0 1 1 1 17 7.08 3.920] ∈ APV0
[4 1 0 1 1 2 2 5.83 5.170] ∈ APV0
[4 1 0 1 2 1 15 7.5 3.500] ∈ APV0
[4 1 0 1 2 2 13 4.58 6.420] ∈ APV0
[4 1 0 1 2 3 1 6.25 4.750] ∈ APV0
[5 2 1 17 7.961] ∈ ASV
[5 2 1 2 5.170] ∈ ASV
[5 2 2 15 3.500] ∈ ASV
[5 2 2 13 6.420] ∈ ASV
[5 2 2 1 4.750] ∈ ASV
[4 2 1 2 3 1 1 5.42 2.884] ∈ APV1
[4 2 1 2 4 2 21 4.58 3.319] ∈ APV1
[4 2 2 1 4 2 21 4.58 3.050] ∈ APV2
[4 2 2 1 4 1 2 2.92 4.177] ∈ APV2
[4 2 1 2 3 2 10 8.33 1.380] ∈ APV1
[4 2 1 2 4 1 2 2.92 4.177] ∈ APV1
[4 2 2 1 4 3 1 8.33 1.268] ∈ APV2
[5 3 3 1 2.884] ∈ ASV
[5 3 4 21 6.369] ∈ ASV
[5 3 4 2 9.040] ∈ ASV
[5 3 3 10 1.380] ∈ ASV
[5 3 4 1 1.268] ∈ ASV
[4 3 3 1 5 1 3 4.17 1.970] ∈ APV3
[4 3 4 1 5 2 2 7.08 0.497] ∈ APV4
[4 3 4 1 1 1 17 7.08 0.497] ∈ APV4
[4 3 4 2 5 2 2 7.08 3.544] ∈ APV4
[4 3 3 1 5 2 2 7.08 1.131] ∈ APV3
[4 3 4 1 5 3 23 4.58 0.814] ∈ APV4
[4 3 4 1 4 4 2 2.92 1.025] ∈ APV4

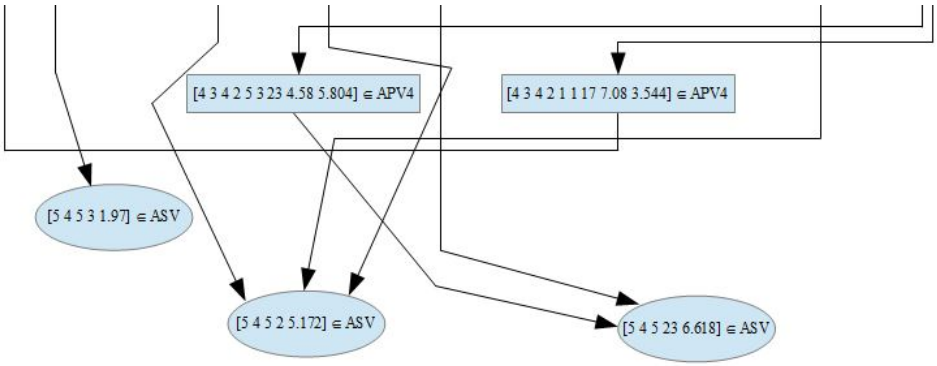


Figure 5.2: Graphical representation of the probabilistic attack graph for the invented example network

[4 2 1 2 4 2 21 4.58 3.319] ⟶ [5 3 4 21 6.369]
[4 2 2 1 4 1 2 2.92 3.838] ⟶ [5 3 4 2 9.040]
[4 2 2 1 4 2 21 4.58 3.050] ⟶ [5 3 4 21 6.369]
[4 2 2 1 4 3 1 8.33 1.268] ⟶ [5 3 4 1 1.268]

Layer 3 of possible attacks $ap \in AP$:

[5 3 3 1 2.884] ∧ [2 3 5 1] ∧ [1 5 1 2 2 3 4.17] ⟶ [4 3 3 1 5 1 3 4.17 1.970]
[5 3 3 1 2.884] ∧ [2 3 5 2] ∧ [1 5 2 2 2 2 7.08] ⟶ [4 3 3 1 5 2 2 7.08 1.131]
[5 3 4 1 1.268] ∧ [2 4 5 2] ∧ [1 5 2 2 2 2 7.08] ⟶ [4 3 4 1 5 2 2 7.08 0.497]
[5 3 4 2 9.040] ∧ [2 4 5 2] ∧ [1 5 2 2 2 2 7.08] ⟶ [4 3 4 2 5 2 2 7.08 3.544]
[5 3 4 1 1.268] ∧ [2 4 5 3] ∧ [1 5 3 2 2 23 4.58] ⟶ [4 3 4 1 5 3 23 4.58 0.814]
[5 3 4 2 9.040] ∧ [2 4 5 3] ∧ [1 5 3 2 2 23 4.58] ⟶ [4 3 4 2 5 3 23 4.58 5.804]
[5 3 4 1 1.268] ∧ [2 4 1 1] ∧ [1 1 1 2 2 17 7.08] ⟶ [4 3 4 1 1 1 17 7.08 0.497]
[5 3 4 2 9.040] ∧ [2 4 1 1] ∧ [1 1 1 2 2 17 7.08] ⟶ [4 3 4 2 1 1 17 7.08 3.544]
[5 3 4 1 1.268] ∧ [1 4 4 1 1 2 2.92] ⟶ [4 3 4 1 4 4 2 2.92 1.025]

Layer 4 of possible attacker states $as \in AS$:

[4 3 3 1 5 1 3 4.17 1.970] ⟶ [5 4 5 3 1.970]
[4 3 3 1 5 2 2 7.08 1.131] ⟶ [5 4 5 2 5.172]
[4 3 4 1 5 2 2 7.08 0.497] ⟶ [5 4 5 2 5.172]
[4 3 4 2 5 2 2 7.08 3.544] ⟶ [5 4 5 2 5.172]
[4 3 4 1 5 3 23 4.58 0.814] ⟶ [5 4 5 23 6.618]
[4 3 4 2 5 3 23 4.58 5.804] ⟶ [5 4 5 23 6.618]
[4 3 4 1 1 1 17 7.08 0.497] ⟶ [5 2 1 17 7.961]
[4 3 4 2 1 1 17 7.08 3.544] ⟶ [5 2 1 17 7.961]
[4 3 4 1 4 4 2 2.92 1.025] ⟶ [5 3 4 2 9.040]

$asn \in ASN$	Single loss expectancy in USD	Annualized rate of occurence
$(5, x, 1, 17, y) \in ASN$	150	8.0
$(5, x, 1, 2, y) \in ASN$	300	5.2
$(5, x, 2, 15, y) \in ASN$	150	3.5
$(5, x, 2, 13, y) \in ASN$	150	6.4
$(5, x, 2, 2, y) \in ASN$	300	4.8
$(5, x, 3, 1, y) \in ASN$	150	2.9
$(5, x, 3, 10, y) \in ASN$	200	1.4
$(5, x, 4, 2, y) \in ASN$	300	9.0
$(5, x, 4, 21, y) \in ASN$	150	6.4
$(5, x, 4, 1, y) \in ASN$	150	1.3
$(5, x, 5, 3, y) \in ASN$	200	2.0
$(5, x, 5, 2, y) \in ASN$	600	5.2
$(5, x, 5, 23, y) \in ASN$	300	6.6

Table 5.1: Single loss expectancy and annualized rate of occurrence for possible attacker states in the invented example network

To compute ALE(network) we need to put a single loss expectancy on each attacker state in the probabilistic attack graph for our invented network. The single loss expectancy and annualized rate of occurrence are given in table 4.1.

Based on these values we can compute ALE(network) for our invented example network:

ALE(network)= 150*7.961+300*5.17+150*3.5+150*6.42+300*4.75+150*2.884 +200*1.38+300*9.04+150*6.369+150*1.268+200*1.97+600*5.172+300*6.618 $\approx$ 16000 USD

5.3 Conclusion and future improvements

We have shown what data is needed and where it can be found to compute ALE(network) for any computer network and also how to compute the optimized order of vulnerability patching. We have been able to compute the ALE(network) for a small invented example network. For the future, work remains on the automation and scalability of these computations, so that the computer network vulnerability analysis process becomes mathematically precise and automatic instead of labour intensive and based on human intuition and experience as it is today.

Appendix A

Implying the probabilistic attack graph from network vulnerability data

A.1 MATLAB code for automatic generation of attacks $apv \in APV$ and attacker states $asv \in ASV$ in vector form except for the expected value element

```
AS1=[5 1 0 1];

CV = [2 0 1 1; 2 0 1 2; 2 0 2 1; 2 0 2 2; 2 0 2 3; 2 1 3 1;...
2 1 3 2; 2 1 4 1; 2 1 4 2; 2 2 4 1; 2 2 4 2; 2 2 4 3; ...
2 3 5 1; 2 3 5 2; 2 4 5 2; 2 4 5 3; 2 4 1 1];

VPV = [1 1 1 2 2 17 7.08; 1 1 2 2 2 2 5.83; 1 2 1 2 2 15 7.5;...
1 2 2 2 2 13 4.58; 1 2 3 2 2 2 6.25; 1 3 1 2 2 1 5.42;...
1 3 2 2 2 10 8.33; 1 4 1 2 2 2 2.92; 1 4 2 2 2 21 4.58;...
1 4 3 2 2 1 8.33; 1 4 4 1 2 1 2.92; 1 5 1 2 2 3 4.17;...
1 5 2 2 2 2 7.08; 1 5 3 2 2 23 4.58];

CV1=[]; VPV1=[]; AS1i=[]; AP1=[];

for i=1:length(CV(:,1))
if CV(i,2)==AS1(3)
CV1 = [CV1; CV(i,:)];
else
CV1 = CV1;
```

```
end
end
CV1

for i=1:length(CV1(:,1))
for j=1:length(VPV(:,1))
if CV1(i,3)==VPV(j,2) & CV1(i,4)==VPV(j,3)
VPV1=[VPV1; VPV(j,:)];
else
VPV1=VPV1;
end
end
end
VPV1

for i=1:length(AS1(:,1))
for j=1:length(CV1(:,1))
if AS1(i,3)==CV1(j,2) & AS1(i,4)==1
AS1i=[AS1i; AS1(i,:)];
elseif AS1(i,3)==CV1(j,2) & AS1(i,4)==2
AS1i=[AS1i; AS1(i,:)];
else
AS1i=AS1i;
end
end
end
AS1i

for i=1:5
AP1 = [AP1; 4, 1, AS1i(i,3), AS1i(i,4), CV1(i,3), CV1(i,4), ...
VPV1(i,6), VPV1(i,7)];
end
AP1

CV2=[]; VPV2=[]; AS2=[]; AP2=[]; AS2i=[];

for j=1:length(AP1(:,1))
AS2 = [AS2; 5 2 AP1(j,5) AP1(j,7)];
end
AS2

for i=1:length(CV(:,1))
for j=1:length(AS2u(:,1))
if CV(i,2)==AS2u(j,3) & AS2u(j,4)==1
CV2=[CV2; CV(i,:)];
elseif CV(i,2)==AS2u(j,3) & AS2u(j,4)==2
```

```
CV2=[CV2; CV(i,:)];
else
CV2=CV2;
end
end
end
CV2

for i=1:length(CV2(:,1))
for j=1:length(VPV(:,1))
if CV2(i,3)==VPV(j,2) & CV2(i,4)==VPV(j,3)
VPV2=[VPV2; VPV(j,:)];
else
VPV2=VPV2;
end
end
end
VPV2

AS2u=unique(AS2,'rows');

for i=1:length(AS2u(:,1))
for j=1:length(CV2(:,1))
if AS2u(i,3)==CV2(j,2) & AS2u(i,4)==1
AS2i=[AS2i; AS2u(i,:)];
elseif AS2u(i,3)==CV2(j,2) & AS2u(i,4)==2
AS2i=[AS2i; AS2u(i,:)];
else
AS2i=AS2i;
end
end
end
AS2i

for i=1:length(CV2(:,1))
AP2 = [AP2; 4, 2, AS2i(i,3), AS2i(i,4), CV2(i,3), CV2(i,4), ...
VPV2(i,6), VPV2(i,7)];
end
AP2

CV3=[]; VPV3=[]; AS3=[]; AP3=[]; AS3i=[];

for j=1:length(AP2(:,1))
AS3 = [AS3; 5 3 AP2(j,5) AP2(j,7)];
end
AS3
```

```
AS3u=unique(AS3,'rows'),

for i=1:length(CV(:,1))
for j=1:length(AS3u(:,1))
if CV(i,2)==AS3u(j,3) & AS3u(j,4)==1
CV3=[CV3; CV(i,:)];
elseif CV(i,2)==AS3u(j,3) & AS3u(j,4)==2
CV3=[CV3; CV(i,:)];
else
CV3=CV3;
end
end
end
CV3

for i=1:length(CV3(:,1))
for j=1:length(VPV(:,1))
if CV3(i,3)==VPV(j,2) & CV3(i,4)==VPV(j,3)
VPV3=[VPV3; VPV(j,:)];
else
VPV3=VPV3;
end
end
end
VPV3

CV3u=unique(CV3,'rows')

for i=1:length(CV3u(:,1))
for j=1:length(AS3u(:,1))
if AS3u(j,3)==CV3u(i,2) & AS3u(j,4)==1
AS3i=[AS3i; AS3u(j,:)];
elseif AS3u(j,3)==CV3u(i,2) & AS3u(j,4)==2
AS3i=[AS3i; AS3u(j,:)];
else
AS3i=AS3i;
end
end
end
AS3i

for i=1:length(CV3(:,1))
AP3 = [AP3; 4, 3, AS3i(i,3), AS3i(i,4), CV3(i,3), CV3(i,4), ...
VPV3(i,6), VPV3(i,7)];
end
```

AP3

AS4=[];

for j=1:length(AP3(:,1))
AS4 = [AS4; 5 4 AP3(j,5) AP3(j,7)];
end
AS4

AS1, AP1, AS2, AP2, AS3, AP3, AS4

A.2 Computation of expected values for attacker state $as \in AS$ and attack $ap \in AP$ random variables

Computation of expected values for attacks $ap \in AP$ and attacker states $as \in AS$. Sometimes attacks $ap \in AP$ imply attacker states $as \in AS$ that have already been implied by other attacks $ap \in AP$ in earlier layers. In this case new computations have to be made of the expected value of the implied attacker state $as \in AS$. Therefore each expected value, both for attacker state $as \in AS$ and attack $ap \in AP$ random variables, has a number in brackets, showing for each expected value how many times computations have been made up to that computation for that expected value. When all computations have been made, the expected value with the highest value in brackets gives the true attacker state $as \in AS$ or attack $ap \in AP$ expected value.

E([4 1 0 1 1 1 17 7.08 y] $\in APV$) (1) = 10*(11-7.08)/10 = 3.92
E([4 1 0 1 1 2 2 5.83 y] $\in APV$) (1) = 10*(11-5.83)/10 = 5.17
E([4 1 0 1 2 1 15 7.5 y] $\in APV$) (1) = 10*(11-7.5)/10 = 3.5
E([4 1 0 1 2 2 13 4.58 y] $\in APV$) (1) = 10*(11-4.58)/10 = 6.42
E([4 1 0 1 2 3 2 6.25 y] $\in APV$) (1) = 10*(11-6.25)/10 = 4.75

E([5 2 1 17 y] $\in ASV$) (1) = 3.92
E([5 2 1 2 y] $\in ASV$) (1) = 5.17
E([5 2 2 15 y] $\in ASV$) (1) = 3.5
E([5 2 2 13 y] $\in ASV$) (1) = 6.42
E([5 2 2 2 y] $\in ASV$) (1) = 4.75

E([4 2 1 2 3 1 1 5.42 y] $\in APV$) (1) = 5.17*(11-5.42)/10 = 2.885
E([4 2 1 2 3 2 10 8.33 y] $\in APV$) (1) = 5.17*(11-8.33)/10 = 1.380
E([4 2 1 2 4 1 2 2.92 y] $\in APV$) (1) = 5.17*(11-2.92)/10 = 4.177
E([4 2 1 2 4 2 21 4.58 y] $\in APV$) (1) = 5.17*(11-4.58)/10 = 3.319
E([4 2 2 1 4 1 2 2.92 y] $\in APV$) (1) = 4.75*(11-2.92)/10 = 3.838
E([4 2 2 1 4 2 21 4.58 y] $\in APV$) (1) = 4.75*(11-4.58)/10 = 3.050

E([4 2 2 1 4 3 1 8.33 y] $\in APV$) (1) = 4.75*(11-8.33)/10 = 1.268

E([5 3 3 1 y] $\in ASV$) (1) = 2.885
E([5 3 3 10 y] $\in ASV$) (1) = 1.380
E([5 3 4 2 y] $\in ASV$) (1) = 4.177 + 3.838 = 8.015
E([5 3 4 21 y] $\in ASV$) (1) = 3.319 + 3.050 = 6.369
E([5 3 4 1 y] $\in ASV$) (1) = 1.268

E([4 3 3 1 5 1 3 4.17 y] $\in APV$) (1) = 2.885*(11-4.17)/10 = 1.971
E([4 3 3 1 5 2 2 7.08 y] $\in APV$) (1) = 2.885*(11-7.08)/10 = 1.131
E([4 3 4 1 5 2 2 7.08 y] $\in APV$) (1) = 1.268*(11-7.08)/10 = 0.497
E([4 3 4 2 5 2 2 7.08 y] $\in APV$) (1) = 8.015*(11-7.08)/10 = 3.142
E([4 3 4 1 5 3 23 4.58 y] $\in APV$) (1) = 1.268*(11-4.58)/10 = 0.814
E([4 3 4 2 5 3 23 4.58 y] $\in APV$) (1) = 8.015*(11-4.58)/10 = 5.146
E([4 3 4 1 1 1 17 7.08 y] $\in APV$) (1) = 1.268*(11-7.08)/10 = 0.497
E([4 3 4 2 1 1 17 7.08 y] $\in APV$) (1) = 8.015*(11-7.08)/10 = 3.142
E([4 3 4 1 4 4 2 2.92 y] $\in APV$) (1) = 1.268*(11-2.92)/10 = 1.025

E([5 4 5 3 y] $\in ASV$) (1) = 1.971
E([5 4 5 2 y] $\in ASV$) (1) = 1.131 + 0.497 + 3.142 = 4.770
E([5 4 5 23 y] $\in ASV$) (1) = 0.814 + 5.146 = 5.960
E([5 2 1 17 y] $\in ASV$) (2) = 3.92 + 0.497 + 3.142 = 7.559
E([5 3 4 2 y] $\in ASV$) (2) = 4.177 + 3.838 + 1.025 = 9.040

E([4 3 4 2 5 2 2 7.08 y] $\in APV$) (2) = 9.040*(11-7.08)/10 = 3.544
E([4 3 4 2 5 3 23 4.58 y] $\in APV$) (2) = 9.040*(11-4.58)/10 = 5.804
E([4 3 4 2 1 1 17 7.08 y] $\in APV$) (2) = 9.040*(11-7.08)/10 = 3.544

E([5 4 5 2 y] $\in ASV$) (2) = 1.131 + 0.497 + 3.544 = 5.172
E([5 4 5 23 y] $\in ASV$) (2) = 5.804 + 0.814 = 6.618
E([5 2 1 17 y] $\in ASV$) (3) = 3.92 + 0.497 + 3.544 = 7.961

Bibliography

[1] Alexis Feringa, Clark Hayden and Gary Stoneburner. National Institute of Standards and Technology. Engineering principles for information technology security (a baseline for achieving security), 2004.

[2] Peter Mell, Sasha Romanosky and Karen Scarfone. A complete guide to the common vulnerability scoring system version 2.0. http://www.first.org/cvss/cvss-guide, 2007.

[3] M. Artz, R. K. Cunningham, K. W. Ingols, R. P. Lippmann, K. J. Kratkiewicz, K. Piwowarski, C. Scott. Evaluating and strengthening enterprise network security using attack graphs. Technical Report ESC-TR-2005-064, MIT Lincoln Laboratory, Massachusetts, USA, 2005.

[4] Daniel Bihar. *Quantitative risk analysis of computer networks*. PhD thesis, Dartmouth College, New Hampshire, USA, 2003.

[5] Key Ingols, Richard Lippmann and Keith Piwowarski. Practical Attack Graph Generation for Network Defense. In *ACSAC '06 Proceedings of the 22nd Annual Computer Security Applications Conference*, pp. 121-130, 2006.

[6] Virginia N. L. Franqueira and Maurice van Keulen. Analysis of the NIST database towards the composition of vulnerabilities in attack scenarios. Technical Report TR-CTIT-08-08, University of Twente, Enschede, Netherlands, 2008.

[7] Andrew Jaquith. *Security metrics: Replacing fear, uncertainty, and doubt*. Addison-Wesley Professional, 2007.

[8] Tania Islam, Sushil Jajodia, Tao Long, Anoop Singhal and Lingyu Wang. An attack graph- based probabilistic security metric. In *Proceedings of The 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, pp. 283 296, 2008.

[9] Paul Ammann and Ronald W. Ritchey. Using model checking to analyze network vulnerabilities. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, pp. 156165, 2000.

[10] Xinming Ou. *A logic-programming approach to network security analysis*. PhD thesis, Princeton University, New Jersey, USA, 2005.

[11] Oleg Mikhail Sheyner. *Scenario graphs and attack graphs*. PhD thesis, Carnegie Mellon University, Pennsylvania, USA, 2004.

[12] Joshua Haines, Somesh Jha, Richard Lippmann, Oleg Sheyner and Jeannette M. Wing. Automated generation and analysis of attack graphs. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pp. 254265, 2002.

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

[14] http://oxforddictionaries.com

[15] Sushil Jajodia and Steven Noel. Topological Vulnerability Analysis. In *Cyber Situational Awareness, Advances in Information Security, Volume 46*, p. 139, 2010

[16] Sushil Jajodia, Anoop Singhal and Lingyu Wang. Measuring the overall security of network configurations using attack graphs. In *Proceedings of the 21st annual IFIP WG 11.3 working conference on Data and applications security*, pp. 98 112, 2007.

[17] Wayne Boyer, Miles McQueen and Ximning Ou. A scalable approach to attack graph generation. In *Proceedings of the 13th ACM conference on Computer and communications security*, pp. 336 345, 2006.

[18] John Homer, Xinming Ou, Anoop Singhal and Su Zhang. An empirical study of a vulnerability metric aggregation method. In *Mission Assurance and Critical Infrastructure Protection*, 2011.

[19] R. P. Lippmann, K. W. Ingols, C. Scott, K. Piwowarski, K. J. Kratkiewicz, M. Artz and R. K. Cunningham. Validating and Restoring Defense in depth Using Attack Graphs. In *Proceedings of the 2006 IEEE conference on Military communications*, pp. 981 990, 2006.

[20] Lingyu Wang, Tania Islam, Tao Long, Anoop Singhal, Sushil Jajodia. An Attack Graph-Based Probabilistic Security Metric. In *Proceeedings of the 22nd annual IFIP WG 11.3 working conference on Data and Applications Security*, pp. 283 296, 2008.

[21] Marcel Frigault and Lingyu Wang. Measuring Network Security Using Bayesian Network-Based Attack Graphs. In *Proceedings of the 2008 32nd Annual IEEE International Computer Software and Applications Conference*, pp. 698 -703, 2008.

[22] John Homer, Xinming Ou and David Schmidt. A Sound and Practical Approach to Quantifying Security Risk in Enterprise Networks. 2009.

[23] Steven Noel, Sushil Jajodia, Lingyu Wang and Anoop Singhal. Measuring Security Risk of Networks Using Attack Graphs. In *International Journal of Next-Generation Computing*, Vol. 1, No. 1, 2010.

[24] Finn V. Jensen and Thomas D. Nielsen. *Bayesian Networks and Decision Graphs*. Springer Verlag, 2007.

[25] Timo Koski and John Noble. *Bayesian Networks: An Introduction*. Wiley, 2009.

[26] Sushil Jajodia, Steven Noel and Brian OBerry. Topological Analysis of Network Attack Vulnerability. In *ASIACCS '07 Proceedings of the 2nd ACM symposium on Information, computer and communications security*, 2005.

[27] Peng Xie, Jason H Li, Xinming Ou, Peng Liu and Renato Levy. Using Bayesian Networks for Cyber Security Analysis. In 2010 *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 211 220, 2010.

[28] Robert Schuppenies. *Automatic Extraction of Vulnerability Information for Attack Graphs*. Master of Science thesis, Potsdam, Germany, 2009.

[29] Symantec DeepSight Threat Ratings Guide.

[30] Gunnar Blom, Jan Enger, Gunnar Englund, Jan Grandell and Lars Holst. *Sannolikhetsteori och statistikteori med tillmpningar*. Studentlitteratur, 2005.