Impact Estimation using Data Flows over Attack Graphs

Tomas Olsson

Swedish Institute of Computer Science
P.O. Box 1263, SE-164 29 Kista, Sweden
`tol@sics.se`

Abstract. We propose a novel approach to estimating the impact of an attack using a data model and an impact model on top of an attack graph. The data model describes how data flows between nodes in the network – how it is copied and processed by softwares and hosts – while the impact model models how exploitation of vulnerabilities affects the data flows with respect to the confidentiality, integrity and availability of the data. In addition, by assigning a loss value to a compromised data set, we can estimate the cost of a successful attack. We show that our algorithm not only subsumes the simple impact estimation used in the literature but also improves it by explicitly modeling loss value dependencies between network nodes. With our model, the operator will be able to use less time when comparing different security patches to a network.

1 Introduction

In order to manage the dynamic nature of the security of a network, where new nodes are added and new softwares are installed, operators regularly run scanning tools such as Nessus to discover the network topology and existing vulnerabilities [1]. However these tools do not put vulnerabilities into the context of how these vulnerabilities can be exploited to obtain illegal access to resources in the network. Thus, the process of setting attacks into context and determining the impact of attacks is still largely a manual process.

Due to these deficiencies, a great number of approaches to automate the analysis of network security have been proposed during the last decade [2]. Some of these automated approaches define security metrics over the paths of an attack graph to measure the security of the network [3,4,5,6,7,8] while others define a security metric in terms of security risk [9,10]. Most of these papers use a simple model for assessing the impact to the network. They either do not use an explicit model, for instance, only using probabilities, or sum the weights of the set of compromised nodes. As an example, in [8], the security of two networks, one patched and one unpatched, is compared by either computing the number of hosts removed from the attack graph or by in addition using a weight of importance for each host assigned by an expert. This would be equivalent to estimating the impact by counting the number of compromised hosts in the attack graph or by computing the sum of their weights.

In our model, we estimate the impact to a network using a data model in combination with an impact model. The data model models how data is copied between softwares in a network and thereby taking into account the dependencies between different hosts and the dependencies between different data flows. The impact model describes what impact individual vulnerabilities have on the data sets with respect to confidentiality, integrity and availability as well as the impact propagation from compromised data sets. By assigning loss values to data sets, we can estimate the cost of a successful attack. Thus, in our work, an operator does not need to understand the importance of each node in a network as is the case in the papers referred to above.

For our model, we follow the terminology of the multi-prerequisite attack graph (MP attack graph) presented in [8], but we introduce our own notation. However, we are not bound to any specific attack graph as long as we can model the flow of data.

The Common Vulnerability Scoring System (CVSS) is an open framework for communicating the impact of vulnerabilities to an IT-system [11]. We take advantage of the expert knowledge gathered through CVSS by using the impact elements from CVSS version 2.

A simple network from [8] illustrates what sort of problem we want to address with our modeling framework. In the example, there are seven hosts, A, B, C, FW (Firewall) $, D, E, F$. The attacker has root access to host A. All other hosts, but the firewall, have a single vulnerability instance, e.g. v_B, that can be exploited remotely via one single open port. The firewall will accept that hosts C and D communicate with host E and will deny all other communication. In Fig. 1, we show the MP attack graph where the triangles indicate vulnerability nodes, circles indicate attack states and rectangles indicate what nodes can be reached from an attack state. The problem we are addressing is how to compare different networks with respect to their vulnerabilities using our impact estimation.

The rest of the paper is organized as follows. Sect. 2 introduces our formalization of the network model and the attack graph presented in [8]. Sect. 3 describes the model of the logic of the impact to a network. Sect. 4 presents the impact estimation computation and shows that it subsumes and extends the previous simple approaches. Sect. 5 presents related work in comparison with the proposed framework.

2 Introducing the MP Attack Graph Model

Our approach begins with constructing a model of the network containing elements such as a network topology, and traffic rules. Then, we define the MP attack graph. In the next sections, we follow the terminology of the NetSPA tool presented in [8]. However, the notation we will introduce is our own contribution.

Basic Network Model A *network topology* consists of a set of *hosts* N and a set of *links* E. Each host $h \in N$ has a set of *interfaces* $i(h) \subseteq I$ where I is the set of all interfaces in the network. The hosts are connected via a set of links such that each link $l \in E$ connects a set of interfaces $i(l) \subseteq I$. In addition, each interface $i \in i(h)$ has a set of *ports* $p(h, i)$.

A *traffic rule* allows a source and a destination software instance to communicate via two interfaces of a host. Each host $h \in N$ has a set of traffic rules $r(h)$. For each $r \in r(h)$, r has two interfaces in $i(h)$ and two *location identifiers* for source and destination. A location identifier consists of a host, an interface and a port. By setting the second interface in a traffic rule equal to the first, we can allow traffic to only flow from or to the host of the rule.

MPAttack Graph The MP attack graph consists of three types of nodes: *prerequisites*, *attack states* and *vulnerability instances.*

A *prerequisite* is the means needed to gain access to a vulnerability instance. It is either a credential; any piece of information such as a password or a private key used for access control or a reachability group that points to a set of reachable hosts. We use Q to denote the set of all prerequisites in a network. For each prerequisite $q \in Q$, $v(q)$ denotes the set of (reachable) vulnerability instances of q.

An *attack state* a consists of a host $h \in N$ and an attack level $o \in \{root, user, dos, other\}$. Notice that the attack state is only used as a model of possible attacks, and the attack level

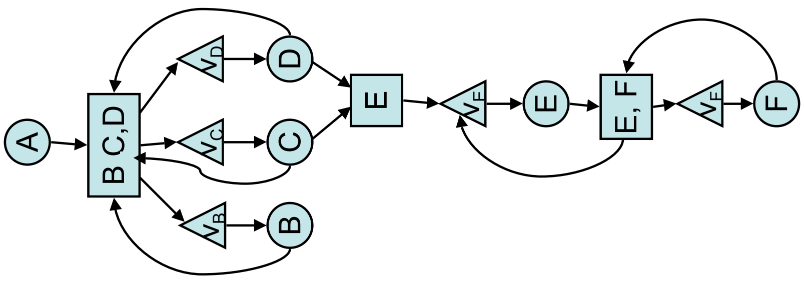

Fig. 1. The multi-prerequisites attack graph from [8].

is not used for modeling impact directly. We model the impact in the next section. We use A to denote the set of all attack states. For each $a \in A$, an attacker will, by obtaining attack state a, also obtain a set of prerequisites $q(a) \subseteq Q$ available at a.

A *vulnerability instance* is any means that an attacker can use to gain access to a system, such as a password, a software flaw, a trust relationship or a configuration error. We use V to denote the set of all vulnerability instances. Each software instance s at a host h has a set of vulnerability instances $v(s) \subseteq V$ affecting s. By successfully exploiting a vulnerability instance $v \in V$, an attacker will obtain a single attack state $a(v) \in A$.

In Table 1 and Table 2, we use our definitions to model the network model and the attack graph shown in Fig. 1 respectively, for the simple network example from Sect. 1.

Table 1. Network model.

$N = \{A, B, C, D, FW, E, F\}$
$E = \{l_1, l_2\}, i(FW) = \{i_{FW1}, i_{FW2}\}$
$\forall n \in N - \{FW\} : i(n) = \{i_n\}$
$i(l_1) = \{i_A, i_B, i_C, i_D, i_{FW1}\}$
$i(l_2) = \{i_{FW2}, i_E, i_F\}$
$r(FW) = \{\langle i_{FW1}, i_{FW2}, [C, i_C, 0], [E, i_E, 0]\rangle,$
$\langle i_{FW1}, i_{FW2}, [D, i_D, 0], [E, i_E, 0]\rangle\}$
$r(C) = \{\langle i_C, i_C, [C, i_C, 0], [E, i_E, 0]\rangle\}$
$r(D) = \{\langle i_D, i_D, [D, i_D, 0], [E, i_E, 0]\rangle\}$
$r(E) = \{\langle i_E, i_E, [C, i_C, 0], [E, i_E, 0]\rangle,$
$\langle i_E, i_E, [D, i_D, 0], [E, i_E, 0]\rangle\}$

Table 2. MP attack graph

$A = \{a_A, a_B, a_C, a_D, a_E, a_F\}$
$Q = \{q_{BCD}, q_E, q_{EF}\}$
$V = \{v_B, v_C, v_D, v_E, v_F\}$
$v(q_{BCD}) = \{v_B, v_C, v_D\}$
$v(q_E) = \{v_E\}, v(q_{EF}) = \{v_E, v_F\}$
$q(a_A) = q(a_B) = \{q_{BCD}\}$
$q(a_C) = \{q_{BCD}, q_E\}$
$q(a_D) = \{q_{BCD}, q_E\}$
$q(a_E) = \{q_{EF}\}, q(a_F) = \{q_{EF}\}$
$a(v_B) = a_B, a(v_C) = a_C, a(v_D) = a_D$
$a(v_E) = a_E, a(v_F) = a_F$

3 Modeling the Impact to a Network using Data flows

After defining the basic network model and the attack graph, we define the data model and the impact model. These are our own contributions, except for some borrowed notions from [11].

3.1 Defining the Data Model

We start by defining a *data set instance* to be a copy of any set of data that is being protected with respect to the three security aspects – confidentiality, integrity and availability. We use D to denote the set of all data set instances.

In order to locate a data set instance in a network, we use a *data set instance location* that consists of a data set instance $d \in D$ and a location identifier with a host $h \in N$, an interface $i \in i(h)$ of host h and a port $p \in p(h, i)$ of interface i.

We assume that all data set instances are somehow processed by *software instances* located at hosts in the network. We denote the set of all software instances S. Thus, each host $h \in N$ has a set of software instances $s(h) \subseteq S$. In order to be the originator or creator of a data set instance, each software instance $s \in s(h)$ has a set of data set instances located at s denoted $store(s)$. Examples of stores are a database, an HTML file or the RAM and examples of software instances are a SQL engine for accessing a database, a web server for accessing HTML pages and a search engine processing queries that caches recent search results in RAM.

If the data is not already in the store, then it must be retrievable from somewhere else. Thus for each software instance $s \in s(h)$ and for each interface $i \in i(h)$, $inputs(s, i)$ denotes the set of data set locations from where s retrieves data set instances via interface i. Similarly, $outputs(s, i)$ denotes the set of data set instance locations with data set instances produced by s that can be retrieved from interface i. Therefore, for each data set location $u \in outputs(s, i)$ at a host h, the interface of u is i, the host of u is h, the port of u is in $p(h, i)$.

We model dependencies between data set instances and thereby between data flows as follows. We let $depends(s, d)$ denote the set of data set instances used by s to produce data set d. Then, either each data set instance $d \in depends(s, d')$ is already stored at s (that is, $d \in store(s)$) or it can be retrieved from somewhere else (that is, there is an interface $i \in i(h)$ such that d has a data set instance location in $inputs(s, i)$). For instance, a distributed search engine would retrieve data from different databases and then present them uniformly by removing duplicates.

Then, we define a *data flow* for a data set instance $d \in D$ from a producer software instance $s_0 \in S$ to a retriever software instance $s_n \in S$ as a sequence of hosts $(h_0, h_1, \ldots, h_{n-1}, h_n)$ in N where: (a) $n > 0$, (b) $outputs(s_0, i_0)$ and $inputs(s_n, i_n)$ contain the same data set instance location with host h_0, interface i_0, port $p_0 \in p(h_0, i_0)$ and data set instance d, (c) there exists links with interfaces connecting all hosts in the sequence, and (d) there exists a traffic rule at each host allowing traffic from source host h_0 to destination host h_n using as source and destination the location identifiers of the producer and the retriever respectively.

Finally, a data flow f is *active* if the retriever actually can retrieve the data set instance. Thus, if the data set instance is depending on other data set instances through the dependencies relation then either the data sets are stored at the producer or there must exist an active data flow f' for each retrieved data set instance with the producer of f as the receiver of f'.

In Table 3, we show a data model for the simple network from Sect. 1 and Table 1.

3.2 Defining the Impact Model

As a means to define the local impact, we define that a data set instance $d \in D$ is *accessible* to a software instance $s \in S$ if either $d \in store(s)$ (d is stored), there exists an active data flow for d with s as receiver (d is retrieved) or there exists an interface i such that d has a data set instance location in $outputs(s, i)$ while for each $d' \in depends(s, d)$, either $d' \in store(s, d)$ or there exists an active data flow for d' with s as receiver (d is produced). Similarly, we define that a data set instance $d \in D$ is *accessible* to a host $h \in N$ if there exists a software instance $s' \in s(h)$ such that d is accessible to s' or there exists an active data flow for d with a sequence of hosts containing h. Hence it is accessible if either the data set instance is stored/produced at the software instance/host or that it can flow to the software instance/host.

Since we want to model the impact of a successful attack to the three security aspects, confidentiality, integrity and availability (denoted c,i,a), we define that for each vulnerability instances

$v \in V$ and for each security aspect $e \in \{c, i, a\}$, $impact(v, e) \in \{None, Partial, Complete\}$ denotes the local impact that vulnerability instance v has on security aspect e at a host h. We borrow the values $None, Partial, Complete$ for the three security aspects from CVSSv2 [11].

For our purpose, we interpret the impact values of CVSSv2 such that for each security aspect $e \in \{c, i, a\}$, an attacker will *locally violate* the security aspect e of, in the case of impact value: (a) $None$, no data set instances, (b) $Partial$, all data set instances *accessible* to the software instance with an exploited vulnerability instance, and (c) $Complete$, all data set instances *accessible* to the host of the software instance with the exploited vulnerability instance. Then, we define that for each data instance $d \in D$, for each security aspect $e \in \{c, i, a\}$, for each subset of exploited vulnerability instances $V_{exploit} \subseteq V$ and for each host $h \in N$, $locallyViolated(e, d, h, V_{exp})$ denotes that d was locally violated by an attacker at host h with respect to e given $V_{Exploit}$.

Then, by assigning a *loss value*, we can estimate a cost of exploiting a vulnerability instance with a certain loss of security. For each data set instance $d \in D$, and for each security aspect $e \in \{c, i, a\}$, $cost(d, e)$ denotes the loss value of data set d with respect to e.

Now, we define predicates for whether a data set instance is violated at the network level:

Loss of confidentiality For each data set instance $d \in D$, for each host $h \in N$ and for each subset of vulnerability instances $V_{exp} \subseteq V$, $lossOfConf(d, h, V_{exp})$ holds true if *locally-*$Violated($ c, d, h, $V_{exp})$ holds true.

Loss of integrity For each data set instance $d \in D$, for each host $h \in N$ and for each set of vulnerability instances $V_{exp} \subseteq V$, $lossOfInteg(d,$ h, $V_{exp})$ holds true if either (a) $locallyViolated(i,$ $d, h, V_{exp})$ holds true, or (b) there exists $s \in S$ such that $d' \in depends(s, d)$ and there exists an active data flow f for d' where the receiver is s and there exists a host h' in the sequence of hosts of f where $h' \neq h$ and $lossOfInteg(d'$, h', $V_{exp})$ holds true.

Loss of availability For each data set instance $d \in D$, for each host $h \in N$ and for each set of vulnerability instances $V_{exp} \subseteq V$, $lossOfAvailability($ d, h, $V_{exp})$ holds true if either (a) *locally-* $Violated$ (a, d, h, V_{exp}) holds true, or (b) there exists $s \in S$ such that $d' \in depends(s, d)$ and for each active data flow f for d' where the receiver is s there exists a host h' in the sequence of hosts of f where $h' \neq h$ and $lossOfAvail(d'$, h', $V_{exp})$ holds true.

Generic loss of security For each security aspect $e \in \{c, i, a\}$, for each data set instance $d \in D$ and for each set of vulnerability instances $V_{exp} \subseteq V$, $lossOfSecurity(e, d, V_{exp})$ holds true if there exits $h \in N$ such that either $e = c$ and $lossOfConf(d, h, V_{exp})$ or $e = i$ and $lossOfInteg(d, h, V_{exp})$ or $e = a$ and $lossOfAvail(d,$ h, $V_{exp})$ holds true.

Notice that we have assumed that the loss of integrity and loss of availability are recursive in nature, while loss of confidentiality is not. This is because, apart from being violated locally at a host, a data set instance d will also suffer a loss of integrity if there is a loss of integrity of any data set instance d' that d is depending on for its existence. If d' was changed and d' is used to produce d then d has also been changed. The same is reasonable for the availability of data. However, in case of integrity, it is enough that there exists a *single* data flow for d' to the producer of d such that d' has been compromised, while in case of loss of availability, *all* data flows for d' to the producer of d must be compromised. If d can be accessed through a remaining data flow then d' can still be produced. In case of loss of confidentiality, we assume that a loss does not propagate to other data set instances. Though this is not necessarily the case since if d is an aggregation of, for instance, two data set instances d' and d'' then there is some dependency between these three data set instances d, d' and d''. We leave this extension to future work.

In Table 3, we show an example of a data model for the simple example in Sect. 1. There is a single vulnerability instance at host $B - F$ and data set instance d_F is most valuable. The data model models that data sets d_E is located at host E, d_F is located at F and that $d_{E'}$ is dependent of d_E for its existence and that $d_{E'}$ flows from E to C using a set of software instances.

Table 3. Data model.

$S = \{s_B, s_C, s_D, s_E, s_F\}$
$\forall h \in N - \{A, FW\} : s(h) = \{s_h\}$
$D = \{d_E, d_{E'}, d_F\}, \mathit{store}(s_F) = \{d_F\}$
$\mathit{store}(s_E) = \{d_E\}, \mathit{depends}(s_E, d_{E'}) = \{d_E\}$
$\mathit{outputs}(s_E, i_E) = \{\langle d_{E'}, E, i_E, 0\rangle\}$
$\mathit{inputs}(s_C, i_C) = \{\langle d_{E'}, E, i_E, 0\rangle\}$

Table 4. Impact Model.

$V = \{v_B, v_C, v_D, v_E, v_F\}$
$\forall h \in N - \{A, FW\}, v(s_h) = \{v_h\}$, $\forall e \in \{c, i, a\}, \mathit{impact}(v_h, e) = \mathit{Complete}$
$\forall e \in \{c, i, a\} : \mathit{cost}(d_E, e) = 1$
$\forall e \in \{c, i, a\} : \mathit{cost}(d_{E'}, e) = 0.5$
$\forall e \in \{c, i, a\} : \mathit{cost}(d_F, e) = 2$

Table 5. Summary of notation

N	Set of hosts: a typical element is $h \in N$
D	Set of data set instances: a typical element is $d \in D$
V	Set of vulnerability instances: a typical el. is $v \in V$
c, i, a	Confidentiality, integrity, availability: $e \in \{c, i, a\}$
$\mathit{cost}(d, e)$	Loss value of d for $e \in \{c, i, a\}$
S	Set of software instances: a typical element is $s \in S$
$s(h)$	Set of software instances at host h
$\mathit{store}(s)$	Set of data set instances stored at s
$\mathit{lossOfSecurity}(e, d, V)$	holds if d has been locally violated for e given V

4 Defining and Applying the Impact Estimation

In Table 5, we have summarized the definitions from the previous section needed to understand this section. Recall from Sect. 1 that the impact estimation for [8] and many other papers could be computed as the sum of the weights over the number of compromised hosts in the attack graph $\mathcal{A}$ as in (1) where $P(h)$ is zero if host h is not in $\mathcal{A}$ or one otherwise, and $\mathit{cost}(h)$ is the importance weight. The corresponding impact estimation for our model is the sum over the data sets is shown in (2) where $P_e(d)$ is zero if $\mathit{lossOfSecurity}(e, d, V)$ does not hold or one otherwise.

$$\mathit{loss}_0(\mathcal{A}) = \sum_{h \in N} \mathit{cost}(h) \cdot P(h) \qquad (1)$$

$$\mathit{loss}(\mathcal{A}) = \sum_{d \in D, e \in \{c,i,a\}} \mathit{cost}(d, e) \cdot P_e(d) \qquad (2)$$

Now, we can show that (2) can express an equivalent impact estimation as (1): Let $D = \{d_h\}_{h \in N}$ and for each $h \in N$, $s(h) = \{s_h\}$, $\mathit{store}(s_h) = \{d_h\}$, $\mathit{cost}(d_h, c) = \mathit{cost}(h)$, $\mathit{cost}(d_h, i) = \mathit{cost}(d_h, a) = 0$, then, since for each $h \in N$, $\mathit{lossOfSecurity}(c, d_h, V)$ holds true, according to its definition, such that $P_c(d_h) = P(h)$:

$$\mathit{loss}(\mathcal{A}) = \sum_{d \in D} \mathit{cost}(d_h, c) P_c(d_h) = \sum_{h \in N} \mathit{cost}(h) P_c(d_h) = \sum_{h \in N} \mathit{cost}(h) P(h) = \mathit{loss}_0(\mathcal{A}) \quad \mathit{Q.E.D.} \qquad (3)$$

Typically, we have an original attack graph $\mathcal{A}$ and we want to use (1) or (2) to compare modifications to $\mathcal{A}$ to determine which vulnerability to patch first. Thus, if $\mathbf{A} = \{\mathcal{A}^0, \ldots, \mathcal{A}^k\}$ is a set of modified attack graphs then we want to choose the graph with lowest estimated impact:

$$\hat{\mathcal{A}} = \min_{\mathcal{A}^* \in \mathbf{A}} (\mathit{loss}(\mathcal{A}^*)) \qquad (4)$$

Table 6 shows the impact estimation for the simple example from [8] that was described in Sect. 1, Fig. 1 and Table 1-4. We show the values for three different data models and four different patch choices. In [8], patches of set of vulnerabilities are compared. For this simple example, we only patch a single vulnerability instance. The first column shows the impact estimation of the models in Table 3 and 4. The second column shows the values when we remove the data flow, in which case, $d_{E'}$ cannot be reached anymore when v_E is patched. The third column have the values when each host $B-F$ has weight 1. Notice that patching v_E is the best choice in all cases.

In the case that a data model lacks data flows, (1) can be used to model the same impact estimations as (2). However, we can show that in order to use (1) for comparing graphs where at least one compromised host affects a data flow, we will most likely have to change the weights of the hosts for that case. The weight assignment can also be rather counterintuitive.

Assume that we have a simple network: $N = \{E, F, G\}$ with the three attack graphs $\mathcal{A}$, $\mathcal{A}^0$, $\mathcal{A}^1$ shown left in Table 7 where the attacker starts in E with root access. There is also an active data flow with the sequence of hosts (G, F) for d. Let the impact value be *Complete* for all vulnerabilities and $cost(d, c) = 1$, $cost(d, i) = cost(d, a) = 0$. Then in case of (2), $loss(\mathcal{A}) = loss(A^0) = 1$, $loss(A^1) = 0$. For (1), reasonable weights for $\mathcal{A}$ would be $cost(F) = 0$, $cost(G) = 1$, making $loss_0(\mathcal{A}) = 1$, but $loss_0(A^0) = loss_0(A^1) = 0$. Thus, we have to change the weights. Counterintuitively, we get the desired result for $cost(F) = 1$, $cost(G) = 0$! The $cost(G)$ is 0 because compromising F is enough – there is no gain by compromising G as well. Consider what happens when we have a third vulnerability $v_{G'}$ accessible from E to G. Then, we have the three new attack graphs $\mathcal{A}'$, $\mathcal{A}^{0'}(= \mathcal{A})$, $\mathcal{A}^{1'}$ and $\mathcal{A}^{2'}$ in Table 7. Now $loss(\mathcal{A}') = loss(\mathcal{A}^{0'}) = loss(\mathcal{A}^{1'}) = loss(\mathcal{A}^{2'}) = 1$ and $loss_0(\mathcal{A}') = loss_0(\mathcal{A}^{0'}) = loss_0(\mathcal{A}^{1'}) = 1$ but $loss_0(\mathcal{A}^{2'}) = 0$. To get the desired result, we revert to $cost(G) = 1$, $cost(F) = 0$. Thus, in contrast to (2), the importance of a host depends on the attack graph.

Table 6. The impact estimation for three different data models.

Patch	Unmod	No Flow	Uniform
none	10.5	10.5	5
v_B	10.5	10.5	4
v_C	10.5	10.5	4
v_D	10.5	10.5	4
v_E	1.5	0	3
v_F	4.5	4.5	4

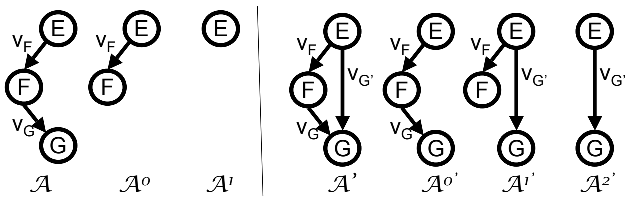

Table 7. Simplified attack graphs for original and modified cases, where one vulnerability is removed.

5 Related work

A set of papers [3,4,5,6,7] define metrics to measure the impact to network based primarily on different ways of weighing paths in an attack graph. Other approaches – like our work – use the cost or loss of an intrusion as the basis for the impact metric [9,10].

An early work is presented in [3,4]. Instead of estimating the impact of an attacker, the authors estimate cost of the attacker in terms of the *time* from an attack starts until it succeeds and the *effort* required by the attacker to succeed. Thus, their work complements our work.

A work inspired by reliability analysis is presented in [5,6]. The authors use a model checker to construct an attack graph from a detailed system model. They can compute the probability of ending up in an unsafe system state. They do not consider any cost of reaching an unsafe state.

In [7] the authors use the PageRank algorithm of Google to measure the security of a network from an attack graph. Their hypothesis is that an attacker behaves quite similarly to a person browsing the web. In contrast to our work, they do not consider the value of protected assets or dependencies between attack states.

In [9], the authors present a framework called RheoStat that uses a cost-benefit analysis to select responses to attacks. A cost is assigned to each asset, but no dependencies between assets are considered and only one host is considered.

The work in [10] uses a Hidden Markov Model (HMM) to estimate the probability for a network given observed intrusion alerts from an IDS to be in malicious states. In contrast to our work, the authors only assign costs of each host being in a state. The impact is estimated either computed as the total expected cost for all hosts or the average expected cost per host.

6 Summary and Concluding Remarks

We have presented a framework for assessing the impact to a network using data flows and an impact model. By modeling the flow of data, we take into account dependencies between different software applications and different hosts as well as dependencies between different data flows. Thus, the operator does not have to consider the importance of different hosts. However, more work is needed to really understand which services exists and how they retrieve and use the data.

An interesting extension, which we have been working on, is to use historical attack data to estimate the probability of a successful exploitation. Thus, turning the P_e function of Sect. 4 into a continuous function in [0,1]. We will leave this work to a future publication.

Acknowledgements. This work has been performed within the SICS Center for Networked Systems funded by VINNOVA, SSF, KKS, ABB, Ericsson, Saab Systems, TeliaSonera and T2Data.

References

1. Nessus: The network vulnerability scanner. http://www.nessus.org (April 2009)
2. Lippman, R., Ingols, K.: An annotated review of past papers on attack graphs. Project Report PR-IA-1, MIT Lincoln laboratory (March 2005)
3. Dacier, M., Deswarte, Y., Kaaniche, M.: Quantitative assessment of operational security: Models and tools (1996)
4. Ortalo, R., Deswarte, Y.: Experimenting with quantitative evaluation tools for monitoring operational security. IEEE Transactions on Software Engineering **25** (1999) 633–650
5. Jha, S., Sheyner, O., Wing, J.M.: Minimization and reliability analyses of attack graphs. Technical Report CMU-CS-02-109, School of Computer Science, Carnegie Mellon University (2002)
6. Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.M.: Automated generation and analysis of attack graphs. In: IEEE Symposium on Security and Privacy. (2002) 273– 284
7. Mehta, V., Bartzis, C., Zhu, H., Clarke, E., Wing, J.: Ranking attack graphs. In: Proceedings of Recent Advances in Intrusion Detection, Springer (2006)
8. Ingols, K., Lippmann, R., Piwowarski, K.: Practical attack graph generation for network defense. In: Computer Security Applications Conference. (2006) 121–130
9. Gehani, A., Kedem, G.: Rheostat : Real-time risk management. In: Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection. (2004) 15–17
10. Årnes, A., Valeur, F., Vigna, G., Kemmerer, R.A.: Using hidden markov models to evaluate the risks of intrusions - system architecture and model validation. In: Proceedings of Recent Advances in Intrusion Detection, Springer (2006)
11. CVSS: Common vulnerability scoring system. http://www.first.org/cvss/ (April 2009)