To appear in Proc. Symp. on Risk Management and Cyber-Informatics (RMCI 2005).

Predicting Availability of Systems using BBN in Aspect-Oriented Risk-Driven Development (AORDD)

Siv Hilde Houmb
Department of Computer Science and Information Science
Norwegian University of Science and Technology
Sem Selands Vei 7-9, NO-7034 Trondheim, Norway
sivhoumb@idi.ntnu.no

Geri Georg, Robert France, Raghu Reddy, and James Bieman
Software Assurance Laboratory
Department of Computer Science, Colorado State University
601 S. Howes St., Fort Collins, CO 80523-1873
(georg/france/raghu/bieman)@CS.colostate.edu

Abstract

Several standards exist that target security assessment and certification. However, these standards support qualitative evaluation according to predefined security levels. To trade-off between treatment strategies, we also need quantitative estimates of operational security. Quantitative evaluation, such as probabilistic analysis, is frequently used within the dependability domain. The need to quantify security attributes has recently been raised. This relates both to security requirements in Quality of Service (QoS) architectures, and input requirements to trade-off decisions regarding the design and choice of security mechanisms. To measure and make trade-off decisions regarding security mechanisms, we separate treatments from the primary functionality model, and model treatment strategies as aspects using Aspect-Oriented Modeling (AOM). In this paper, we develop a Bayesian Belief Network (BBN) based prediction system for estimating system availability. Availability is estimated using the variables mean time to mis-use (MTTM), mean effort to mis-use (METM), impact of misuse (MI), and frequency of misuse (MF). Misuses are addressed using treatment strategies. The quality of treatment strategies is estimated using the variables treatment cost (TC) and treatment effect (TE).

Keywords: *Security management, prediction system, trade-off analysis, BBN, and AOM*

1 Introduction

In security management there is no single correct software design solution, rather choices and trade-offs. Furthermore, the multidisciplinary nature of these choices involves trade-off among conflicting objectives. This means that we need techniques to rank available solutions. The Aspect Oriented Risk Driven (AORDD) Bayesian Belief Network (BBN) based cost-benefit trade-off analysis provides decision support for resolving conflicts, by computing the Return of Security Investment (RoSI) for each solution. In the context of this paper we are looking into treatment strategies, which are solutions to misuses. We make use of Aspect Oriented Modeling (AOM), and model treatment strategies as aspects to separate security concerns from the primary model. This makes it easy to swap treatment solutions in and out when computing RoSI.

To compute RoSI and rank the different solutions we need to specify misuses and solutions in terms of quality attributes. Quality attributes are non-functional requirements, such as requirements related to the reliability, availability, safety, and security [19] of a system. In the context of this paper we build a prediction system to estimate the availability of a system, as part of a estimation repository in the AORDD framework.

The paper is structured as follows. Section 2 gives an overview of related work. Section 3 gives a brief description of the AORDD framework, while Section 4 presents the prediction system for system availability. Section 5 presents the BBN topology, which is the implementation of the prediction system. Section 6 presents the information sources for quantifying variables used to predict availability. Section 7 gives an example of the approach using the e-commerce platform ACTIVE. We conclude the paper by looking into some of the problems with the approach, as well as presenting further work.

2 Related work

In Littlewood et al. [20] a first step towards operational measures of computer security is discussed. The authors point to the lack of quantitative measures for determining operational security, and relate security assessment to traditional probability theory. They make use of knowledge from the dependability domain, and re-define the input space and usage environment of this domaine to include intentional attacks posed upon the system. Furthermore, Ortalo et al. [23] present a quantitative model to measure known Unix security vulnerabilities using a privilege graph, which is transformed into a Markov chain. The model allows the characterization of operational security to be expressed as *mean effort to security failure* as proposed in Littlewood et al.

Madan et al. [21] consider security to be a Quality of Service (QoS) attribute and, based on ideas from Littlewood et al. [20], they present an approach to quantify security attributes of intrusion-tolerant software systems using stochastic modeling techniques. Wang et. al. [35] extend the state transition approach of Mandan et al. They present the difficulty of capturing details of real architectures in manually constructed Markov models, and advocate the use of Stochastic Petri Nets (SPN). A similar approach is used in Singh et al. [28], who describe an approach for probabilistic validation of an intrusion-tolerant replication system. Here, a hierarchical Stochastic Activity Nets (SAN) model is used to validate intrusion-tolerant systems and to evaluate the merits of various design choices.

Jonsson and Olovsson [18] present a quantitative analysis of attacker behavior based on empirical data collected from intrusion experiments performed by undergraduate students at Chalmers University in Sweden. The results of the experiment show that a typical attacker behavior comprises three phases; the learning phase, the standard attack phase, which appears to be exponentially distributed, and the innovative attack phase.

Our approach is based on the initial concepts discussed in Littlewood et al. [20] and adopted in Madan et al. [21]. We model security risk as a random process, and base our analysis on the concepts of stochastic modeling, and in particular Markov analysis, as presented by Madan et al. [21], Ortalo and Deswarte [23], and Jonsson and Olovsson [18]. We use aspects to model treatment strategies in order to separate concerns and ease trade-off analysis by making it easier to incorporate different security treatments into the system under consideration. We also discuss how to combine empirical data and subjective expert judgment when predicting availability.

3 AORDD Framework

The AORDD framework combines risk-driven development (RDD) [30] with aspect-oriented modeling (AOM) [9]. The framework consists of the AORDD process [12], an iterative development process, a security treatment aspect repository, an estimation repository, rules for how to annotate UML models with information used for estimation, a BBN- based cost-benefit trade-off analysis, and rules for how to transfer information from the annotated UML models into the BBN topology. In this paper we focus on estimation, and in particular how to measure availability. We also describe how to apply the information as input to the BBN topology.

Figure 1 gives an overview of the activities in the AORDD cost-benefit trade-off analysis. Whenever a decision request is initiated, the composer integrates the aspect models with the primary model. Then the prediction manager examines the composed model. The main task for the prediction manager is to identify the type of system, and the quality attributes that need to be provided to the BBN. These variables are provided as annotations in the UML models. The appropriate estimation set is requested from the repository, and fed into the BBN topology. The BBN topology is the implementation of the trade-off analysis. The estimation repository is connected to the security treatment aspect repository, where the aspect models are stored. Furthermore, for both repositories we have company confidential and public versions. These two repositories are updated whenever new information is available.

4 Prediction system for availability

Availability is the property of being accessible and usable upon demand by an authorized entity [14]. This means that system assets should be available to the proper system user upon request. It also means that system assets should not be available to a non-user upon request. To describe availability we need to specify not only expected system behavior, the services that the system delivers to the environment, but also the system's ability to resist external faults, and in particular intentional faults [16]. Figure 2 gives an overview of the relations between requests and preservation of system availability.

This model of availability is based on work by Laprie [19] and Jonsson [16]. A *fault* occurs when user or non-user input causes an error in the system. A *failure* is defined as an undesirable state. A failure may lead to degradation of a system service, and thereby reduce system availability.

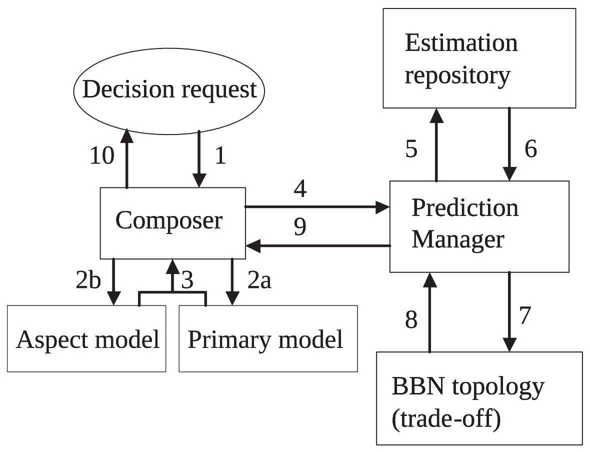


Figure 1. Overview of the activities of the AORDD cost-benefit trade-off analysis

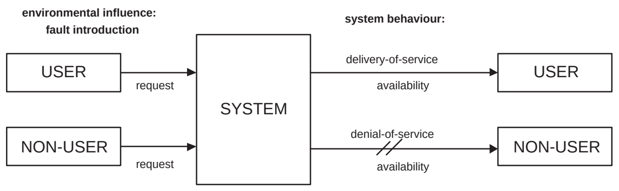


Figure 2. Effect of environmental and internal influence on the system

We measure availability using service levels [17], which is modeled using continuous-time Markov chains [34]. A service level is defined as a group of system states, each denoting a specified degree of normal system accomplishment. The service levels are dependent on the design and implementation of the system (structure of the system), as well as on the application of the system; how the system is used. A degraded service delivered by a system may be regarded as a full service in a certain application, or by a certain user. Service levels are therefore domain and stakeholder specific.

The highest service level is **service level 0 (SL0)**, or full service level. This level is comprised of system states that describe the complete fulfillment of all requirements in the specification, and at the same time preserve availability; the fully operational states. The lowest service level is **service level x (SLx)**, or no service level. This level is comprised of states where the system does not deliver any service to the user, or delivers service to the non-user, which means that the system availability is compromised. In security critical systems some failures are more serious than others, implying severe economic loss. These failures are denoted catastrophic. Such failures are modeled as **service level x (SLx)**, no service level, and treated separately using fail safe states. The fail safe state for the example, given in Section 7, is that whenever a catastrophic failure occurs, the system is shut down. An example of a catastrophic failure is when a non-user has gained full access to the system. We also describe levels between these two service levels, each denoting different degraded operational states.

Service levels depend on system states and we model the behavior of the system, including the identified mis-uses, as a state transition diagram. Operational states (normal system behavior) are modeled as normal behavior states (NB), and states associated with a misuse are modeled as misuse states (MU). Each state in the state diagram is annotated with the type of state and the service level to which it belongs. Figure 3 shows a general example of a state transition diagram, while Figure 4 depicts the related state transition matrix. Both figures use the NB, MU, and FS state name notation. The matrix in Figure 4 shows transition probabilities from each state to every other state in the system.

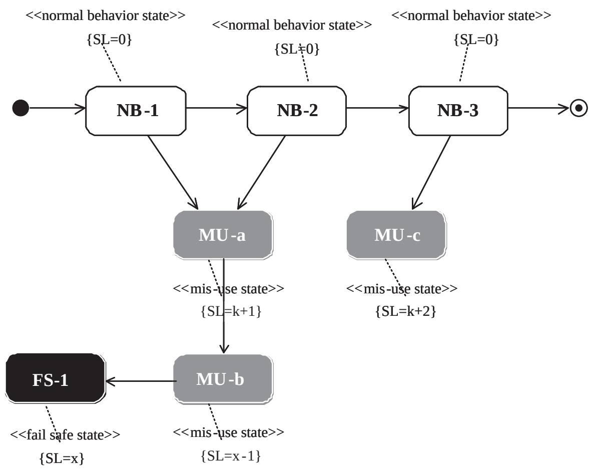
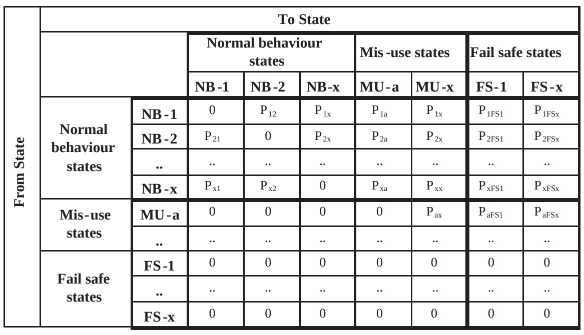


Figure 3. State transition diagram

Transitions between states can be modeled using any set of UML diagrams, given that the UML diagrams are annotated appropriately. As shown in the example in Section 7, we compute the transition probabilities and associated quality attributes (in this case the quality attribute availability) using sequence diagrams.

4.1 Computational procedure for availability

The first step in the computational procedure is to define the service levels, along with specifying those functions or



			To State						
			Normal behaviour states			Mis-use states		Fail safe states	
			NB-1	NB-2	NB-x	MU-a	MU-x	FS-1	FS-x
From State	Normal behaviour states	NB-1	0	P_{12}	P_{1x}	P_{1a}	P_{1x}	P_{1FS1}	P_{1FSx}
		NB-2	P_{21}	0	P_{2x}	P_{2a}	P_{2x}	P_{2FS1}	P_{2FSx}
		..	..	..	..	..	..	..	..
		NB-x	P_{x1}	P_{x2}	0	P_{xa}	P_{xx}	P_{xFS1}	P_{xFSx}
	Mis-use states	MU-a	0	0	0	0	P_{ax}	P_{aFS1}	P_{aFSx}
		..	..	..	..	..	..	..	..
	Fail safe states	FS-1	0	0	0	0	0	0	0
		..	..	..	..	..	..	..	..
		FS-x	0	0	0	0	0	0	0

Figure 4. State transition matrix

sub-services that belong to each level. The second step includes the construction of a availability state transition diagram for each failed service level. This implies modeling the system behavior using annotated UML interaction diagrams and the system architecture, including hardware as well as software, using annotated UML structural diagrams. In the third step, failure rates, for all the transitions in the availability transition diagram, are calculated by compiling the influence of the various faults, and then integrating them for each service level. In security critical systems failure rates for each level are not constant, but are rather functions over time.

4.2 Definition of the availability prediction system

We model the state of a system as a continuous-time Markov process, $\{X_t\}_{t \geq 0}$, with a finite state space *E*, in which each service level, *SLn*, can be identified with a subset of states in *E*. Thus, *E* is the disjoint union *SL0 + ... + SLx*, where *x* is the number of service levels. Furthermore, service levels *0,...,k* correspond to **operational** states *O*; states giving the minimum required availability for the system, in the sense that the system delivers a full or degraded service to the user. Service levels *k+1,...,x* denotes the **failed** states. That is, *E = O + F* where, *O = SL0+ ... +SLk* and *F = SL(k+1)+ ... +SLx*.

Transitions $i \rightarrow j$ have intensity λ_{ij} ($i,j \varepsilon E$, $i \neq j$). The initial probability $P(X_0 = i)$ is denoted by πi. In most situations, the system will always start in a fixed state i_0 so that: π_i = 1 for $j = i_0$ and 0 for $j \neq i_0$.

We assume that the system starts at the highest service level; $i_0 \varepsilon SL0$. Transitions between operational states represent degradations, and transitions to a failed state represent failures. We do not consider repair or online implementation of treatment strategies (such as dynamic allocation of treatments), meaning that no transitions take place from a failed state; λ_{fj} = 0 for $f \varepsilon F$ and all $j \varepsilon E$. Failed states are absorbing.

For an acyclic Markov state graph (meaning that a state can only be visited once) the mean time to mis-use, MTTM, is given by the sum of the mean sojourn times in the states leading to the target, weighted by the probability of visiting these states. The mean sojourn time in state j, denoted as T_j, is given by the inverse of the sum of state j's output transition rates: T_j = 1/λ. $MTTM_i$ = T_i + $\sum_{l \varepsilon out(i)}$ P_{il} * $MTTM_{il}$; P_{il} = λ_{il} * T_i, where i is the initial state and l is the target state. The mean effort to misuse, METM; $METM_i$ = E_i + $\sum_{l \varepsilon out(i)}$ P_{il} * $METM_{il}$; P_{il} = λ_{il} * E_i, where E_i is the sojourn effort spent in state i.

For a set of n states system availability is given as SA = $\int_{1 \leq SLn \leq x}$ ($\int_{0 \leq i \leq k}$ ($MTTM_i$ + $METM_i$ *di* + $\sum_{0 \leq i \leq k}$ $MF_i|MTTM_i$, $METM_i$, TE_i + $\sum_{0 \leq i \leq k}$ $MI_i|TE_i$), where MF is misuse frequency and TE is treatment effect. When doing trade-off between treatment strategies we also need to take treatment cost, TE, into account. This is measured using Return of Security Investment, RoSI; ROSI = TC * $\int_{1 \leq SLn \leq x}$ ($\int_{0 \leq i \leq k}$ $MTTM_i$ + $METM_i$ *di* + $\sum_{0 \leq i \leq k}$ $MF_i|MTTM_i$, $METM_i$, TE_i + $\sum_{0 \leq i \leq k}$ $MI_i|TE_i$).

Please note that the method is based on a pre-defined set of service levels (see Section 4.1), and a set of corresponding failure rates estimating the rate of transitions between levels. These sets are further explored in the next sections.

5 Implementing the availability prediction system using BBN

To predict availability we need information on which to base estimates. There are rarely sufficient amounts of empirical data available, so we extend the method with tools to reason under uncertainty. We also need to combine disparate information sources for assessing the quality of treatment strategies in terms of RoSI, and to compute the system availability.

BBN have proven to be a powerful technique for reasoning under uncertainty, and have been successfully applied to the assessment of system safety [5], [6], [7], [27], [33], and [8]. The BBN methodology [26] is based on Bayes rule. HUGIN [13] is the leading tool supporting BBN.

Bayes rule calculates conditional probabilities. Given the two variables X and Y, the probability P for the variable X given the variable Y, can be calculated from: P(X|Y)=P(Y|X)*P(X)/P(Y). By allowing X_i to be a complete set of mutually exclusive instances of X, Bayes formula can be extended to calculate the conditional probability of X_i given Y. A BBN is a connected and directed graph consisting of a set of nodes and a set of directed arcs (or links). Nodes are defined as stochastic or decision variables, and multiple variables may be used to determine the state of a node. Each state of each node is expressed using probability density. The probability density expresses our confidence to the various outcomes of the set of variables connected to a node, and depends conditionally on the status of the parent node at the incoming edges. For more information on BBN and variables in BBN the reader is referred to [15].

Figure 5 presents a portion of the BBN topology for assessing system availability based on treatment strategies. Note that this is just a small part of the complete BBN for computing RoSI, which represents the implementation of the availability prediction system. The decision variables are mean time to mis-use (MTTM), mean effort to misuse (METM), and treatment effect (TE). These variables affect state nodes mis-use frequency (MF) and mis-use impact (MI). The resulting states of these nodes are combined with TE using a BBN utility node ($SA_utility$) and are also used to calculate the value of the node of interest, system availability (SA).

Treatment effect (TE) is related to misuse impact (MI) and frequency (MF). We can impact TE in three ways; by reducing MI, reducing MF, or both. To calculate a reduction in MI we estimate the effect using the same unit as for misuse impact, e.g. asset value loss for MI and asset value regain for TE, or by using a qualitative scale such as low, medium, and high. The main asset in the example is the availability of the system, and MI and TE are estimated in terms of loss and regain of system availability. The change in system availability does therefore need to be included in the definition of the different service levels. When calculating a reduction in MF the values are given as reduced frequency or probability, depending on the unit used to describe misuse frequency. Treatment cost represents the estimated cost to implement the treatment strategy. This cost is not shown in the BBN fragment of Figure 5. In the example of Section 7 we estimate cost as economical expenses, but one may also use other value types, such as resources or time to implement the treatment during development or system downtime, if the system is a running system. BBN has the ability to use different units for different values, as long as their relations are defined; the conditional probabilities, e.g. P((MF=high)=0.8|METM=low, MTTM ¡ 1 hour, TE=none).

Misuse frequency depends on the time and effort to misuse and the treatment effect (MF|MTTM, METM, TE). MTTM and METM are based on the model for quantification of security attributes in software systems as discussed by Mandan et al. [21] and Wang et al. [35]. Mandan et al. [21] measure these two variables using stochastic processes, a semi-Markov process, since the time and effort distribution may be non-exponential (a novice attacker uses more effort and time before a successful attack). It should also be noted that an attacker may not always be successful in causing a misuse, i.e., probability of success ≤ 1. Jonsson and Olovsson [18] describe an attack using three phases; learning phase, standard attack phase, and innovative attack phase. Time and effort between successful attacks are shown to be exponential for the standard attack phase (standard attacks for an average attacker). In this paper we only consider the standard attack phase, and assume an exponential distribution between misuses (successful attacks), as given in Section 4.2.

6 Information sources

Since there are limited amount of relevant empirical data available for the variables MI, MF, and TE, we need to combine information sources and reason under uncertainty using the BBN topology. We have two main types of information sources for misuse and treatment variable estimation; empirical or observable information ("objective) and subjective expert judgments.

6.1 Empirical or observable information sources

One type of observable information source is real-time information sources, such as Intrusion Detection Systems (IDS) and honeypots [29]. Other observable information sources are company experience repositories, public repositories, and experience from similar systems. Examples of public repositories are the quarterly reports from Senter for Informasjonssikkerhet (SIS) in Norway, CERT.com, reports from the Honeynet-project [31], and other attack trend reports.

Another type of empirical source is available When using information from similar systems. In this case we use experts to reason about system differences and their effect on the configuration being assessed. The result of this reasoning is documented as annotated UML models, and a document describing the experts and their discussion of the provided opinions. This process is formalized using a documentation template.

6.2 Subjective expert judgments

The two most common expert collections methods were developed by RAND; the scenario method and the Delphi method [22]. A problematic aspect with the Delphi method is that the result from the expert judgments represents their agreed upon values, rather than aggregated and traceable individual values. When estimating misuse impact (MI), treatment strategy effect (TE), and cost (TC) we need to make use of experts with different knowledge domains, as illustrated in Figure 6.

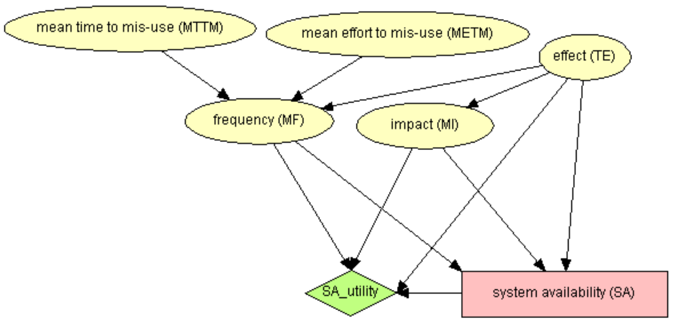


Figure 5. BBN topology for assessing system availability

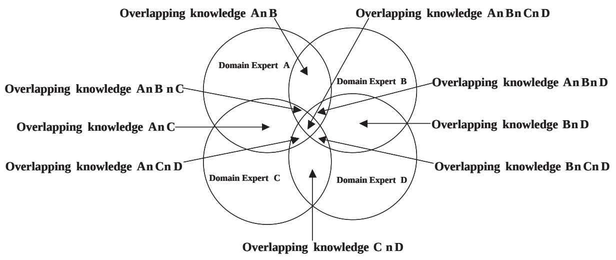

Overlapping knowledge AnB
Overlapping knowledge AnBnCnD
Domain Expert A
Domain Expert B
Overlapping knowledge AnB n C
Overlapping knowledge AnBnD
Overlapping knowledge AnC
Overlapping knowledge BnD
Overlapping knowledge AnCn D
Overlapping knowledge BnCnD
Domain Expert C
Domain Expert D
Overlapping knowledge C n D

Figure 6. Relation between domain expert knowledge

7 Predicting availability of ACTIVE

The e-Commerce platform ACTIVE [1] consists of a web server running Microsoft Internet Information Server (IIS), an Java application server (Allaire JSP Engine), and a database server running Microsoft SQL Server.

The IST EU-project CORAS [4] performed three risk assessments of ACTIVE in the period 2000–2003. The project looked into security risks of the user authentification mechanism, secure payment mechanism, and the agent negotiation mechanisms of ACTIVE. In the context of this paper we will use the results from the risk assessment of the user authentification mechanism.

To access any of the services in ACTIVE, users must either login as a registered user or a visitor. The authentification mechanism is a cross-cutting concern and can be viewed separately from the main functionality of the ACTIVE platform. We term this main functionality as the primary model [11].

One of the misuse types identified for the authentification mechanism was different types of denial of service (DoS) attacks, and in particular TCP-SYN flooding [3] and IP spoofing attacks [2]. Service levels used are; SL0, SL1, SL2, SL(x-1), and the fail safe service level SLx. SL0 denotes the set of states describing a full operational system. SL1 denotes the set of states with degraded availability for users, and no access to services for non-users. SL2 denotes a further degradation in the services provided to users, and still no access for non-users. In our example, SL(x-1) denotes the set of services where there is no availability of services for users, as well as no access to services for non-users. SLx denotes the set of fail safe states. For this example we have one fail safe state FS-1; full access to services for non-users and the system is taken offline.

Figure 7 depicts a UML sequence diagram of the original login sequence for ACTIVE using a TCP connection. The sequence diagram has been annotated to include availability variables for the TCP connection setup service. The highest level of service begins at the top left of the diagram, and ends with the return of a login page from the e-commerce system web server. The level of service is given in terms of the physical deployment. Thus, if the client and server are connected over a LAN the level of service will be SL0. However if the client and server communicate over internet or a dial-up connection, the service will be somewhat degraded, to SL1. (These service levels are defined in an associated deployment diagram, which is not included in this paper due to space constraints.)

The degraded service, where service is not available to users or to non-users, begins at the ’else’ clause in the NetworkStack-Server, when the data structures for partially opened TCP connections have been filled. At this point, no more connections can be accepted, with the result that no one can gain new access to the system. This is service SL(x-1). The service ends when an error is returned to the web client. The availability variables associated with the service level are shown to the right of the ’else’ guard. These variables are shown as an instance in an array of possible expert data. In this example, information from the ith expert is shown in the figure. Since the data is stored in an

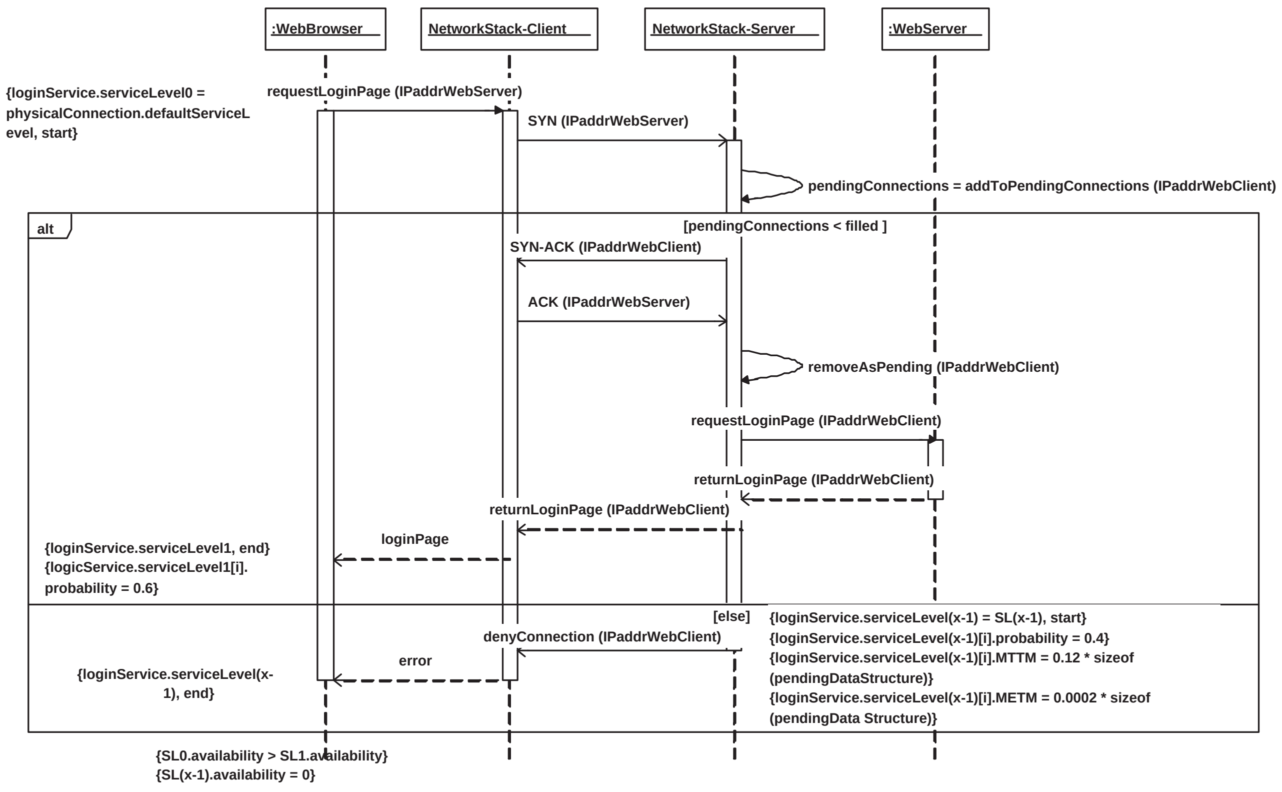

:WebBrowser
NetworkStack-Client
NetworkStack-Server
:WebServer
{loginService.serviceLevel0 = physicalConnection.defaultServiceLevel, start}
requestLoginPage (IPaddrWebServer)
SYN (IPaddrWebServer)
pendingConnections = addToPendingConnections (IPaddrWebClient)
alt
[pendingConnections < filled]
SYN-ACK (IPaddrWebClient)
ACK (IPaddrWebServer)
removeAsPending (IPaddrWebClient)
requestLoginPage (IPaddrWebClient)
returnLoginPage (IPaddrWebClient)
returnLoginPage (IPaddrWebClient)
loginPage
{loginService.serviceLevel1, end}
{logicService.serviceLevel1[i].probability = 0.6}
[else]
{loginService.serviceLevel(x-1) = SL(x-1), start}
{loginService.serviceLevel(x-1)[i].probability = 0.4}
{loginService.serviceLevel(x-1)[i].MTTM = 0.12 * sizeof (pendingDataStructure)}
{loginService.serviceLevel(x-1)[i].METM = 0.0002 * sizeof (pendingData Structure)}
denyConnection (IPaddrWebClient)
error
{loginService.serviceLevel(x-1), end}
{SL0.availability > SL1.availability}
{SL(x-1).availability = 0}

Figure 7. Login service using a TCP connection

array, the information from different experts can be combined according to averages, a weighting scheme or other types of algorithms, before it is used in a BBN computation. The variables are the probability that the attack will occur (given as 0.4), the time needed to reach a successful attack, and the effort needed to reach a successful attack. The two last variables are given in terms of the number of connections that the server can handle. Based on honeypot observations of a prototype of the e-commerce system, we assume the server can handle up to 1000 simultaneously TCP connections. We set the time to a successful attack to 0.12 times the number of connections the server can handle, and the effort as 0.0002 times the number of connections. We thus annotate the diagram based on empirical data from the honeypot.

There are several available treatment strategies for the misuse TCP-SYN flooding. One possible treatment strategy is a patch to the network stack software that adds another message layer once the partial connection data structures becomes close to full. In this situation, a cookie is sent to the client and the pending connection is removed from the data structure. If the client does not respond within a short period of time, the cookie expires and the client must restart the request for a connection. If the client responds in time the SYN-ACK message is sent, and the connection is set up. Adding the cookie message makes it unlikely that an attacker can respond in time, and if the client address has been spoofed, the client will not respond in any event. In both cases, the cookie will expire on the server, without taking up any storage in the pending connections data structures. Thus, these structures will not fill due to flooding and cause a failure in the delivery of the service. This treatment sequence can be composed with the original login sequence to produce a model of the protected system. This composition is shown in Figure 8.

The woven model is also annotated with service availability variables. As before, the highest service level begins at the upper left corner of the sequence. It ends at the bottom left corner of the sequence, when the client receives a login page. The highest level of service is given in terms of the physical connection between the client and server.

The degraded service level begins when the pending connection data structure is close to full, and ends when the client receives a login page. The degraded service level is given in terms of the original highest service level. The degraded service availability variables are shown to the right of the guard that starts the degraded service. Again, they are shown as an instance of an array, so that the variables for

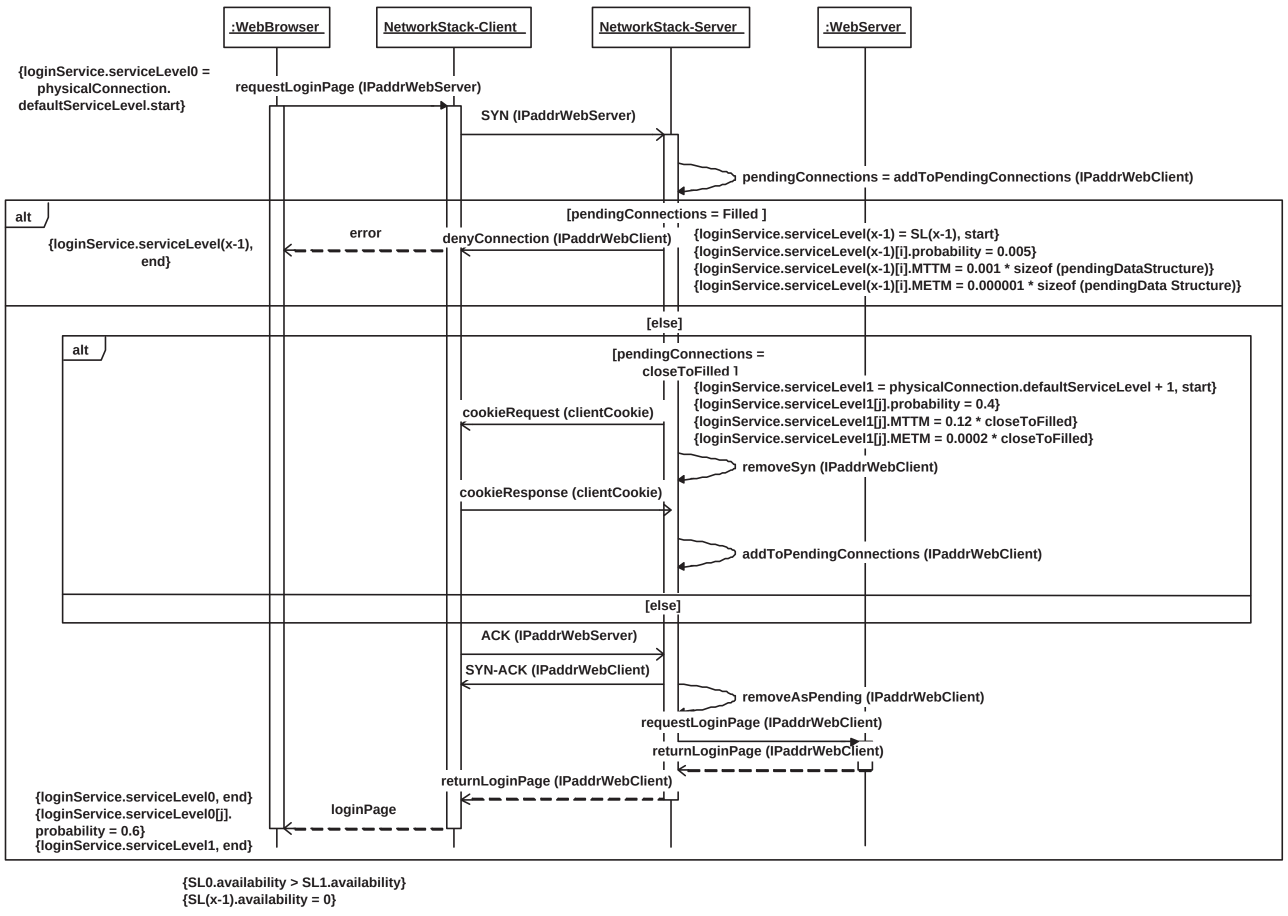

:WebBrowser
NetworkStack-Client
NetworkStack-Server
:WebServer
{loginService.serviceLevel0 = physicalConnection. defaultServiceLevel.start}
requestLoginPage (IPaddrWebServer)
SYN (IPaddrWebServer)
pendingConnections = addToPendingConnections (IPaddrWebClient)
alt
[pendingConnections = Filled]
error
denyConnection (IPaddrWebClient)
{loginService.serviceLevel(x-1), end}
{loginService.serviceLevel(x-1) = SL(x-1), start}
{loginService.serviceLevel(x-1)[i].probability = 0.005}
{loginService.serviceLevel(x-1)[i].MTTM = 0.001 * sizeof (pendingDataStructure)}
{loginService.serviceLevel(x-1)[i].METM = 0.000001 * sizeof (pendingData Structure)}
[else]
alt
[pendingConnections = closeToFilled]
{loginService.serviceLevel1 = physicalConnection.defaultServiceLevel + 1, start}
{loginService.serviceLevel1[j].probability = 0.4}
{loginService.serviceLevel1[j].MTTM = 0.12 * closeToFilled}
{loginService.serviceLevel1[j].METM = 0.0002 * closeToFilled}
cookieRequest (clientCookie)
removeSyn (IPaddrWebClient)
cookieResponse (clientCookie)
addToPendingConnections (IPaddrWebClient)
[else]
ACK (IPaddrWebServer)
SYN-ACK (IPaddrWebClient)
removeAsPending (IPaddrWebClient)
requestLoginPage (IPaddrWebClient)
returnLoginPage (IPaddrWebClient)
returnLoginPage (IPaddrWebClient)
{loginService.serviceLevel0, end}
{loginService.serviceLevel0[j]. probability = 0.6}
{loginService.serviceLevel1, end}
loginPage
{SL0.availability > SL1.availability}
{SL(x-1).availability = 0}

Figure 8. Woven “OS·· patch aspect model and primary model

multiple experts can be included in the BBN computations. The availability variables are similar to the values as in the original login service sequence depicted in Figure 7, since the attack is the same. One difference is that the number of connections that are needed before the service degrades is based on the limit ’closeToFilled’. Another difference is that the service level only degrades once this threshold has been reached. The service does completely fail if the data structure for partially opened TCP connections is full. This case will not occur because of a flooding attact, so the variables associated with this failure are much lower than in the unprotected sequence shown in Figure 7.

7.1 Information sources used for predicting availability

Information sources used for estimating availability include log files from a honeypot configured with Honeyd [32]. Honeyd simulated Windows NT 4.0 server with Internet Information Server 4.0 (IIS) [24]. As a second layer of logging we used the Network IDS version of Snort.

In this example, we will only look at connection attempts to TCP port 80, which is intended for the IIS server. During a period of 24 hours Snort detected 470 different IP-addresses trying to open connections to port 80, which gives 470/24=19,8 attack tries per hour. This was done by penetrating the network sending SYN requests to ranges of IP-addresses. In this case we are most interested in the attack attempts where outsiders sent several SYN requests to the same source. 140 of the 470 IP-addresses sent a series of SYN requests to the same source within 24 hours, which gives 140/24=5,8 SYN flooding attack tries per hour. 2 of the attack tries was successful, which gives MTTM=12 hours. The mean attack time for the successful attacks where; METM=0,2 hours.

7.2 Computing system availability using BBN

Recall the BBN topology from Section 5. To compute system availability for the treatment strategy “OS patch” we insert the annotated information from the UML sequence diagram modeling the composed treatment and primary model for "OS patch" (see Figure 8). Figure 9 shows a portion of the BBN that computes system availability of "OS patch" with the annotated information.

We have made a few simplifications and used qualitative scales; low, medium, and high for MI and MF, and none, low, medium, and high for TE. The translation scale used for MF is; low=0.0-0.4, medium=0.4-0.8, and high=0.8-1.0. In order to illustrate the use of the BBN topology we assumes that MTTM=12 hours equals medium on the qualitative scale, and that METM=0,2 hours equals low on the qualitative scale.

8 Conclusion and further work

The second-generation honeynet, GENII, has the possibility to provide real-time and realistic empirical data both during the development and maintenance of a system. By configuring the Honeynet with the current design and solutions, we can test the current design for possible vulnerabilities, as well as assessing the effect of different treatment strategies. When doing trade-off between treatment strategies, GENII has the possibility to collect estimates by configuring the Honeynet using one treatment strategy, record data, and then swap treatment strategy and record data. This can also be automated and connected directly with the BBN topology. By preparing a set of configurations for the different treatment strategies, the Honeynet is configured automatically using a predefined time interval. The data recorded by the log-files is examined and inserted into the appropriate observable nodes in the BBN topology, which then computes the RoSI for each treatment strategy. However, the quality of the data collected depends on the quality of the filtering mechanism in the Honeynet, the Honeywall, which distinguishes authorized use from unauthorised use. This is further discussed by Ostvang [25].

In this paper we have focused on building a prediction system for estimating the availability of a system. To compute RoSI for treatment strategies, or to assess the overall security level of a system, one needs to include more quality attributes in the prediction system. Laprie [19] describes trustworthiness of a system using a dependability framework. Dependability is described by four basic attributes; reliability, availability, safety, and security. These four attributes are primary related to non-degradable systems. Jonsson [16] extends the dependability framework to include security specific attributes, and in particular effects of activities from non-users. Further work includes extending the estimation repository of AORDD to include the quality attributes described in [16].

When assessing treatment strategies or the availability of a system, we need to combine disparate information sources and aggregate expert opinions. There exist a set of methods to assess safety of software systems using BBN [10]. Further work includes looking into how to utilize BBN for aggregating expert opinions and combining information sources for RoSI.

References

[1] EP-27046-ACTIVE, Final Prototype and User Manual, D4.2.2, Ver. 2.0, 2001-02-22., 2001.

[2] CERT Advisory CA-1995-01. Ip spoofing attacks and hijacked terminal connections, September 1997. CERT Coordination Centre, http://www.cert.org/advisories/CA-1995-01.html.

[3] CERT Advisory CA-1996-21. Tcp syn flooding and ip spoofing attacks, November 2000. CERT Coordination Centre, http://www.cert.org/advisories/CA-1996-21.html.

[4] CORAS (2000–2003). A platform for risk analysis of security critical systems. IST-2000-25031, http://www.sourceforge.net/coras/, 29. November 2004.

[5] P.-J. Courtois, N. E. Fenton, B. Littlewood, M. Neil, L. Strigini, and D. R. Wright. Bayesian belief network model for the safety assessment of nuclear computer-based systems. Second year report part 2, Esprit Long Term Research Project 20072-DeVa, 1998.

[6] K. Delic, M. Mazzanti, and L. Stringini. Formilizing engineering judgment on software dependability via belied netowrks. In *DCCA-6, Sixth IFIP International Working Conference on Dependable Computing for Critical Applications, "Can We Rely on Computers?"*, Garmisch-Partenkirchen, Germany, 1997.

[7] N. Fenton, B. Littlewood, M. Neil, L. Strigini, A. Sutcliffe, and D. Wright. Assessing dependability of safety critical systems using diverse evidence. *IEEE Proceedings Software Engineering*, 145(1), 1998.

[8] N. Fenton and M. Neil. A critique of software defect prediction models. *IEEE Transaction of Software Engineering*, 25(5):675–689, 1999.

[9] G. Georg, R. France, and I. Ray. An aspect-based approach to modeling security concerns. In *Workshop on Critical Systems Development with UML (CSDUML'02)*. Dresden, Germany, October 2002.

[10] B. A. Gran. *The use of Bayesian Belief Networks for combining dispparate sources of information in the safety assessment of software based systems*. Doxtoral of engineering thesis 2002:35, Department of Mathematical Science, Norwegian University if Science and Technology, 2002. 2002:35.

[11] S. H. Houmb, G. Georg, R. France, J. Bieman, and J. Jürjens. Cost-benefit trade-off analysis using bbn for aspect-oriented risk-driven development. In *Accepted for the International Conference on Engineering of Complex Computer System (ICECCS2005) in Shanghai, China, 16-20 June 2005*, 2005.

[12] S. H. Houmb, G. Georg, R. France, and D. Matheson. Using aspects to manage security risks in risk-driven development. In *3rd International Workshop on Critical Systems Development with UML*, number TUM-I0415, pages 71–84. TUM, 2004.

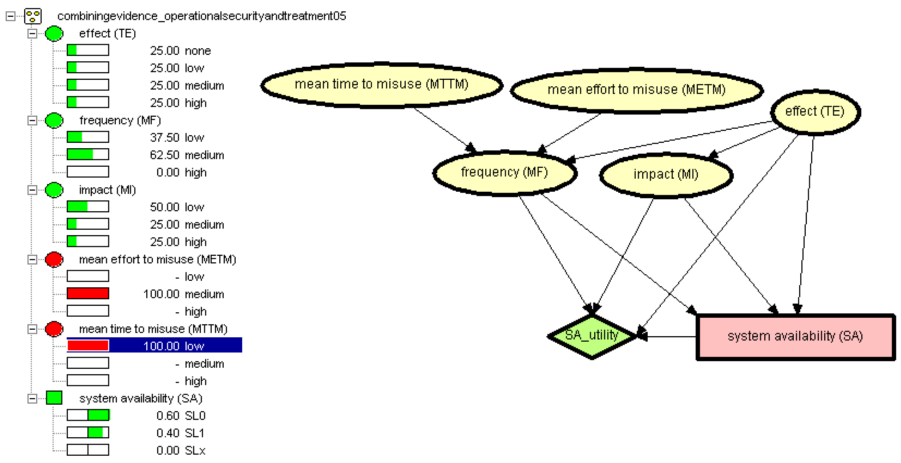

```
combiningevidence_operationalsecurityandtreatment05
  effect (TE)
      25.00 none
      25.00 low
      25.00 medium
      25.00 high
  frequency (MF)
      37.50 low
      62.50 medium
       0.00 high
  impact (MI)
      50.00 low
      25.00 medium
      25.00 high
  mean effort to misuse (METM)
          - low
     100.00 medium
          - high
  mean time to misuse (MTTM)
     100.00 low
          - medium
          - high
  system availability (SA)
       0.60 SL0
       0.40 SL1
       0.00 SLx
```

Figure 9. Computing system availability using BBN

[13] HUGIN: Tool made by Hugin Expert a/s, Alborg, Denmark, 2004. http://www.hugin.dk.

[14] ISO/IEC. *ISO/IEC 13335: Information technology – Guidelines for management of IT Security*, 2001.

[15] F. Jensen. *An introduction to Bayesian Network*. UCL Press, University College London, 1996.

[16] E. Jonsson. On the integration of security and dependability in computer systems. In *IASTED Int'l Conf. Reliability, Quality Control and Risk Assessment*, pages 93–97. Washington, Nov. 4–6 1992.

[17] E. Jonsson and S. Asmussen. A practical measure for some dependability attributes in degradable computing systems. In *In Proceedings of NSDCS92*, 1992.

[18] E. Jonsson and T. Olovsson. A quantitative model of the security intrusion process based on attacker behavior. *IEEE Trans. Software Eng.*, 4(25):235, April 1997.

[19] J. C. Laprie. *Dependability: Basic Concepts and Terminology*. Springer-Verlage, 1992.

[20] B. Littlewood, S. Brocklehurst, N. Fenton, P. Mellor, S. Page, D. Wright, J. Dobson, McDermid J., and D. Gollmann. Towards operational measures of computer security. *Journal of Computer Security*, 2:211–229, 1993.

[21] B. Madan, K. Vaidyanathan, and K. Trivedi. Modeling and quantification of security attributes of software systems. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN'02)*, 2000.

[22] K. Øien and P. R. Hokstad. Handbook for performing expert judgment. Technical report, SINTEF, 1998.

[23] R. Ortalo and Y. Deswarte. Experiments with quantitative evaluation tools for monitoring operational security. *IEEE Trans. Software Eng.*, 5(25):633–650, Sept/Oct 1999.

[24] M. E. Østvang. The honeynet project, phase 1: Installing and tuning honeyd using lids. Master's thesis, Norwegian University of Science and Technology, 2003.

[25] M. E. Østvang. Using honeynet as an information source in a business perspective: What are the benefits and what are the risks? Master's thesis, Norwegian University of Science and Technology, 2004.

[26] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Network forPlausible Inference*. Morgan Kaufmann, 1988.

[27] SERENE: Safety and Risk Evaluation using Bayesian Nets. ESPIRIT Framework IV nr. 22187, 1999. http://www.hugin.dk/serene/.

[28] S. Singh, M. Cukier, and W. H. Sanders. Probabilistic validation of an intrusion-tolerant replication system. In de Bakker, J.W., de Roever, W.-P., and Rozenberg, G., editors, *International Conference on Dependable Systems and Networks (DSN'03)*, June 2001.

[29] L. Spitzner. *Honeypot - tracking hackers*. Addison-Wesley, 2003.

[30] K. Stølen, F. den Braber, T. Dimitrakos, R. Fredriksen, B. A. Gran, S. H. Houmb, Y. C. Stamatiou, and J. Ø. Aagedal. Model-based risk assessment in a component-based software engineering process: The CORAS approach to identify security risks. In F. Barbier, editor, *Business Component-Based Software Engineering*, pages 189–207. Kluwer, 2002. ISBN: 1-4020-7207-4.

[31] The Honeynet Project. The web page for the honeynet project. http://www.honeynet.org/. 15 December 2004.

[32] The Honeynet Project. *Know your enemy, Revealing the sescurity tools, tactics, and motives of the blackhat community*. Addison-Wesley, 2002. ISBN: 0201746131.

[33] TRACS: Transport Reliability Assessment Calculation System, DERA project E20262, 1999. http://www.agena.co.uk/tracs/index.html.

[34] D. Vose. *Risk Analysis: A Quantitative Guide*. John Wiley & Sons Ltd., 2000.

[35] D. Wang, B. B. Madan, and K. S. Trivedi. Security analysis of sitar intrusion tolerance system. In *ACM SSRS'03*. ACM Press, 2003.