A Framework for Predicting Security and Dependability Measures in Real-time

Karin Sallhammar, Bjarne E. Helvik and Svein J. Knapskog

"Centre for Quantifiable Quality of Service in Communication Systems",
Norwegian University of Science and Technology, Trondheim, Norway

Summary
The complex networked systems of today that our technological and social society relies upon are vulnerable to a large number of failures, accidental as well as intentional. Ideally, the service delivered by such a system should be both dependable and secure. This paper presents a framework for integrated security and dependability assessment. The proposed model is based on traditional stochastic analysis techniques, supported by live data from network sensors, which is used to estimate the current state and predict the future behavior of a system in real-time. The method is demonstrated by a small case study.

Key words:
Stochastic modeling, integrating security and dependability, security measures, real-time prediction, hidden Markov models.

1. Introduction

The new paradigms of ubiquitous computing and high capacity data transfer has led to an explosive growth in the number and complexity of computing systems used for critical applications, such as electronic commerce, health-care and urgent information interchange. Since the modern society of today is highly dependent on these services, the computing systems need to function properly despite not only accidental failures but also malicious attacks. To increase the trustworthiness of the implemented services, it should be possible to monitor the systems' current robustness towards these impairments, as well as assess and predict their current and near future behavior. This paper deals with a method for such monitoring and prediction.

A system's ability to provide a correct and timely service can be described in terms of its dependability and security. Dependability is the ability to deliver service that can justifiably be trusted, and can be stated as an integrative concept that encompasses the attributes availability, reliability, safety, integrity and maintainability [1]. Security, on the other hand, is defined as a concept addressing the attributes confidentiality, integrity and availability [7]. To function properly, critical applications and systems need to be both dependable *and* secure. However, despite the fact that a system cannot be considered trustworthy without a rigorous analysis comprising a joint consideration of these two concepts, dependability and security have tended to be treated separately. To allow continuous estimation of the trustworthiness of the services provided by today's computing systems, there is an urgent need of new modeling methods that treats both security and dependability.

As pointed out in [10], to model, analyze and evaluate systems that are yet to be built or systems whose specific vulnerabilities remain unknown, stochastic assumptions are needed. During the last decade, probabilistic modeling of security has gained a lot of interest [2, 8–10, 13, 17]. Such models, which are inspired by the traditional dependability analysis techniques, can be used to provide quantitative measures of a system's operational security. However, most of the recent research efforts have focus on either security or dependability analysis, rather than aiming for a unified evaluation framework. In [12] we described a method for integrated security and dependability evaluation, which uses stochastic analysis techniques to model and compute expected failure times for a computing system, regardless of whether the failure cause is intentional or not. To incorporate malicious behavior in the stochastic model, a game theoretic approach is applied. The method can be used for both predicting a system's future security and dependability behavior, as well as for trade-off analysis of possible countermeasures. This paper extends our previously published results by integrating the proposed model in a distributed network monitoring environment. By using observations provided by network sensors, the probability of the current system state can be estimated, which makes it possible to use the stochastic model to predict the future behavior of the monitored computer system in real-time. The overall concepts used in our framework are depicted in Fig. 1. The system that is to be assessed, together with its operational environment, is described by a stochastic model. There is a monitor that continuously surveys the system and gives warnings of possible disturbances. By

Manuscript received March 5, 2007
Manuscript revised March 25, 2007

using the stochastic model together with real-time data from the monitor our model can estimate the current state of the system, and predict its future behavior.

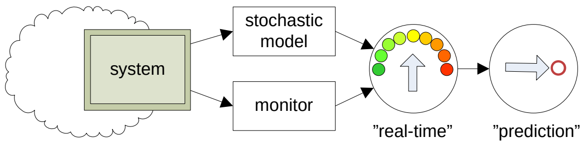

Fig. 1: The overall concepts.

This paper is organized as follows. Section 2 starts by reviewing the basis of stochastic modeling, and introduces the proposed security and dependability measures. Section 3 discusses how security analysis differs from traditional dependability evaluation, and explains how these issues are treated in our integrated framework. In Section 4 the proposed security and dependability assessment architecture is presented. In Section 5 we explain how a monitoring architecture can be used to collect sensor data and how this information is interpreted to estimate the current system state. Section 6 provides the methodology for computing real-time measures of the system. In Section 7 the approach is demonstrated by an illustrative example. Finally, Section 8 concludes the paper and points to future work.

2. Predicting Security and Dependability

When operating a system with security and dependability requirements it is of interest to be able answer questions like: "Is the system already compromised?", "What is the probability that the system is currently under attack?", "What is the probability that the system will operate without security breaches or failures for the next 24 hours?", or "What is the expected time until such an event occurs?". The objective of this paper is to provide a methodology that can be used to provide answers to such questions. For the simplicity of the presentation we denote any undesired event in or state of the system as a *failure*, in accordance with [1], without discrimination with respect to kind of failure. A refinement is straight forward. Hence, we denote the time from now and until the next failure by T_F and seek to obtain:

- $P_F(t) = P(T_F > t)$
  The probability that the time until the next failure is greater than t[1].

- $MTNF = \frac{1}{P_F(0)} \int_0^\infty P_F(t)dt$
  The mean time to next failure[2], assuming that the system will sooner or later fail (i.e., $\lim_{t\to\infty} P_F(t) = 0$).

Dealing with security issues, it is not necessarily evident when the system is failed (compromised), so we will in general have $P_F(0) \le 1$, where $1 - P_F(0)$ represents the probability that the system is already failed (compromised).

To obtain the above measures, two important elements are needed:

- The ability to predict the current state of the system, and
- For a given state, the ability to predict the future behavior of the system.

We will return to these elements later on in this paper. First, we introduce the overall modeling approach applying Markov models and discuss how security issues may be included in these.

2.1 The Stochastic Modeling Approach

Above we have informally introduced the concept of state. By a *state* in this context is meant an operational mode of the system characterized by which units of the system that are operational or failed, whether there are ongoing attacks, active countermeasures, operational and maintenance activities, whether parts of the system have been compromised or not, etc. The decision of what to include or not in the state definition is a trade-off between model representativeness and complexity, and is a salient part of every modeling and analysis effort. The more system states that are taken into consideration, the more fine-

[1] This expression corresponds to the reliability function in traditional dependability analysis (where only random failures are regarded), but this term is not used to avoid misinterpretation.
[2] The *MTNF* measure differs from the *MTTF* (mean time to failure) and *MTFF* (mean time to first failure) traditionally used in dependability analysis. In contrast to *MTTF* and *MTFF*, the *MTNF* measure is conditioned on $P_F(0)$. The measure will be further explained in Section 6.

granular the model becomes. Since the model needs to be parameterized, the granulation of the state space must be carefully considered. Too simple models will not provide any valuable insight into the system behavior, whereas too complex models will quickly lead to state space explosion. An example primarily for illustration will be presented in Section 7.

Let say the system has N disjoint states and that at any time it is in one of these. The behavior of the system is characterized by the *transitions* between the states, each transition triggered by an event. The events that will occur next, as well as the time until next event, are random. Hence, the behavior of the system is a stochastic process. For the sake of simplicity, assume that the occurrence rates of events depend only on the state the system is in, i.e., the system is modeled by a continuous time Markov chain (CTMC) (for an introduction to Markov modeling for dependability analysis, see for instance [5]). A CTMC is characterized by its rate matrix, whose elements represent the transition rates between the system states.

2.2 System Equations

Assume that the system has a finite number of states, denoted $S = \{S_1, \dots, S_N\}$. This state set can be split into two subsets $S = \{S_G, S_F\}$, such that S_G contains the good system states and S_F contains the failed system states. Let

$$X(t) = \{X_1(t), \dots, X_N(t)\}, \tag{1}$$

where $X_i(t)$ denotes the probability that the system is in state i at time t. The state equation describing the system behavior is then

$$\frac{d}{dt}X(t) = X(t)Q, \tag{2}$$

where $Q = \{q_{ij}\}$ is the $N \times N$ state transition rate matrix of the system. The element q_{ij} represents the transition rate from state i to state j. Note that $q_{ii} = -\sum_{i \neq j} q_{ij}$. The state equation can be solved if the initial state of the system $X(0)$ is known. Then

$$X(t) = X(0)e^{Qt}. \tag{3}$$

The solution to (3) provides the transient state probabilities for the system. See for instance [15]. However, the probability that a CTMC will be in state i at time t often converges to a limiting value, which is independent of the initial state. The steady state probabilities

$$X = \{X_1, \dots, X_N\}, \tag{4}$$

whose elements $X_i = \lim_{t\to\infty} X_i(t), i = 1, \dots, N,$ can then be obtained by solving the set of N equations given by N - *1* of the N equations

$$XQ = 0, \tag{5}$$

and with the N'th equation

$$\sum_{l=1}^{N} X_l = 1. \tag{6}$$

As is common practice in dependability analysis, the CTMC can be used as a basis for obtaining various measures of the system, such as the *MTNF* or $P_F(t)$ previously discussed. For computation of measures relating to T_F, the failure states may be made absorbing, i.e., $q^*_{ij} = 0$ when $i \in S_F, j \in S_G$, otherwise $q^*_{ij} = q_{ij}$. Let $Q^* = \{q^*_{ij}\}$ be the modified state transition rate matrix with absorbing failure states and denote by $X^*(t)$ the corresponding state probabilities. Hence,

$$P_F(t) = \sum_{i \in S_G} X^*_i(t), \tag{7}$$

from which the *MTNF* can be computed. Rather than integrating $\int_0^\infty P_F(t)dt$ to obtain *MTNF* we will adopt a computationally more efficient approach, based on [3]. The details together with computational issues will be further explained in Section 6.

3. The Challenges with Security Modeling

In dependability analysis it is very common to use the stochastic modeling approach described in Section 2 to quantify the reliability or availability of systems. In that case, the states are classified as either "up" states (the good states in S_G) or "down" states (the failed states in S_F), depending on whether the required service is delivered or not. In theory, by associating down states with failures of confidentiality or integrity one can use these methods for evaluating also the security properties of a system. A simple example is depicted in Fig. 2, where S

= {G,C,F} = {"good", "compromised", "failed"}. Here, it is assumed that a large number of attackers are targeting the system in state *G*, with accumulated attack intensity λ. In contrast to attack graphs (as used in e.g., [8]) where each state transition corresponds to a single atomic step of a penetration, our model aim to be more high-level and focus on the *impact* of the attacks on the system rather than on the specific attack procedures themselves. This facilitates the modeling of unknown attacks in terms of generic state transitions. For example, in Fig. 2 the attack is merely defined as "the action that seeks to transfer the system from a good state to a compromised state" and nothing more.

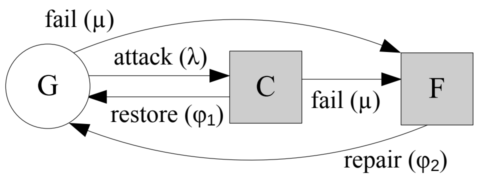

Fig. 2: A simple Markov model, including a compromised system state.

However, for real-world cases where more complex models are needed two problems quickly arise:

1. The attacks are *intentional*, rather than purely random.
2. Regarding security, the current system state may be *unobservable*.

The remainder of this section will discuss these two problems and explain our proposed solutions.

3.1 Incorporating Attacker Behavior

When using the traditional Markov approach it is (in most cases) straightforward to model accidental failures as state transitions. However, since attacks are intentional they may not always be well characterized by models of random nature. Hence, a more sophisticated approach than the simple model depicted in Fig. 2 is needed. To be able to model the effect of a successful attack as a transition between system states one needs to consider the two underlying causes of any attack. As pointed out in [12], there must be (at least) one vulnerability in the system, and a malicious action that tries to exploit that vulnerability, for an attack to be successful. Even though the time an attacker needs to perform an attack action may be modeled as randomly distributed, the *decision* to perform the action will also influence the system failure rate. Therefore, attacker behavior must be represented in the state transitions. In this paper we follow the approach in [12] and define $\pi_i(a)$ as the probability that an attacker will choose action *a* when the system is in (the vulnerable) state *i*. The failure rate between state *i* and *j* when incorporating malicious behavior can therefore be computed as

$$q_{ij} = \pi_i(a)\lambda_{ij}(a), \tag{8}$$

where $\lambda_{ij}(a)$ is the accumulated intensity if all potential attackers always take action *a*. By introducing the attack probability $\pi_i(a)$ as an element in the rate value q_{ij}, the result from a successful attack can be modeled as one or more *intentional state changes* of the underlying stochastic process, which represents the dynamic behavior of the system. To compute the attack probabilities we use the game model proposed in [12]. The game model is based on a reward- and cost concept, which makes it possible to predict the expected attacker behavior, in terms of attack probabilities, for a number of different attacker profiles. The game theoretic approach will not be further explained in this paper; the reader is referred to [12] for the exact details.

3.2 Observing the System Security State

In dependability analysis, the system state set *S* is usually considered known. Moreover, all states are assumed to be deterministically observable, in that the current system state is well defined and perceptible, at all times. However, in a security context the degree of unobservability may be quite high. A system might very well seem to be in a good state even though it is compromised, e.g., due to a stealthy attack. How can one compute measures such as $P_F(t)$ or *MTNF* if one does not know the initial state of the system with certainty? Our solution is to use information from network sensors monitoring the system to estimate its current state probability. We then replace $X(0)$ in (3) with the most likely state probability at that particular time instant, which provides us with the possibility of re-computing the system measures in real-time. This procedure will be further explained in the subsequent three sections of this paper.

4. The Prediction Framework

The proposed real-time security and dependability prediction framework is illustrated in Fig. 3. As depicted in the figure, the system is described by a three-part stochastic model, which consists of the state space *S*, the game model Γ and the corresponding state transition rate matrix *Q*. Naturally, the system behavior will depend on its operating environment, such as user behavior, administrative activities, the possible intrusions and exploits, and random software and hardware failures. Note that we include attacker profile data as a separate part of the system environment. As mentioned in the previous section, by using cost-and reward values from the attacker profile, the game model is used to compute the attack probabilities that are to be incorporated in the rate matrix. As Fig. 3 indicates, the purpose of the stochastic model is to provide the system rate values needed to predict the $P_F(t)$ and *MTNF* measures.

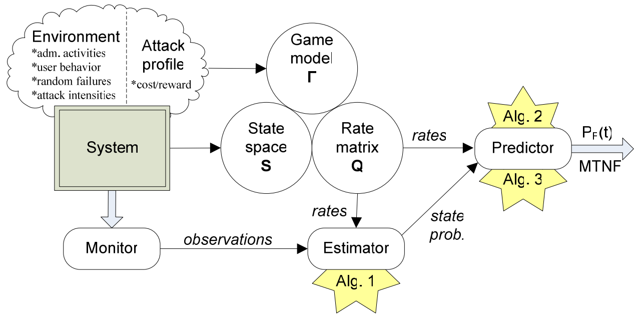

Fig. 3: The security and dependability assessment architecture.

From Fig. 3 it is clear that to perform real-time security and dependability assessment for the system, there are three main tasks that has to be performed; *monitoring*, *estimation* and *predicting*. The main task of the monitor is to provide the estimator with observations regarding the system security and dependability behavior. The monitor is implemented as a distributed architecture consisting of agents that observe the system by means of network sensors. The estimator then uses the observations provided by the monitor to estimate the current state probability of the system. To be able to obtain the goal of real-time security and dependability measurements, the current state probability is updated and forwarded to the predictor as soon as a new observation has been received and interpreted. Finally, the predictor computes measures for the observed system, based on the transition rates from the stochastic model together with the estimated state probability. As previously discussed, the predictor uses well-known Markov analysis techniques to compute the $P_F(t)$ and *MTNF* for the system. The implementation details of the monitoring and estimation architecture (Alg. 1) will be further described in Section 5 and the exact procedure for computing the predicted measures (Alg. 2-3) will be explained in Section 6.

5. The Monitoring and Estimation Architecture

The proposed monitor and estimator in Fig. 3 are both based on the results published in [11]. In this paper we restrict ourselves to an overall description of the architecture. The reader is referred to [11] for more details.

5.1 The Monitor

In [11], the monitor is implemented as a distributed monitoring architecture consisting of agents that observe one or more systems using network sensors. A *sensor* can be any information-gathering program or device, including network sniffers using sampling or filtering, different types of intrusion detection systems (IDS), logging systems, virus detectors, etc. The main task of a sensor is to gather information regarding the current state of one or more systems. The assumed monitoring architecture is hybrid in the sense that it supports any type of sensor. However, it is assumed that the sensors are able to classify and send standardized observations according to the state estimation model described in this paper. An *agent* is a computer program capable of a certain degree of autonomous action. An agent is responsible for collecting and aggregating sensor data from a set of sensors that monitor one or more systems and to forward these data to the estimator. In a multi-agent system, agents are capable of communicating and cooperating with other agents. A multi-agent architecture is preferable over a single agent implementation, due to its flexibility and scalability. The case study presented later on in this paper makes use of a single agent only.

In real-life distributed agent-sensor implementations, observations often arrive in bursts, and there will also be silent periods without any activity at all. In this paper we let the agent adopt a sampling process of the sensors monitoring a particular system, similarly to the approach in [18]. By providing the estimator with observations at regular time intervals, the predicted system security and dependability measures can be updated at a predefined frequency. The sampling process will be further explained in the next subsection.

5.2 The Discrete Sampling Process

Recall that we use a CTMC to model the security and dependability behavior of a system. Due to its stochastic behavior, the system may be in any of the states in *S* when sampled. As Fig. 3 indicates, the purpose of the estimator is to use the sampled observations to estimate the current system state. To formalize, let z_τ be the (possibly unobservable) system state at sampling instant τ. The *sequence* of states that a system is in during the sampling instants will then be denoted $Z = (z_1, z_2, \ldots)$. Let

$$X^\tau = \{X_1^\tau, \ldots, X_N^\tau\}, \tag{9}$$

where X_i^τ denotes the probability that the system is in state *i* at the τ'th sample. Since one cannot assume that *Z* is known (recall the unobservability problem discussed in Section 3.2), it is this state probability that will be estimated and used to predict new system measures, at each sampling instant τ.

Recall the system rate matrix *Q*. Assume that the interval between two different adjacent samples is fixed to Δ. Now, let $P(\Delta)$ denote the one-step transition probability matrix with elements $p_{ij}(\Delta)$, such that $p_{ij}(\Delta) = P(z_{\tau+1} = j \mid z_\tau = i),\ 1 \le i, j \le N$. Hence, $p_{ij}(\Delta)$ represents the probability that the system will, given that its current state at sampling instant τ is *i*, be in state *j* after an additional time Δ, i.e., at the next sample τ +*1*. By using (3), $P(\Delta)$ can be derived from *Q* as

$$P(\Delta) = I \cdot e^{Q\Delta}, \tag{10}$$

where *I* is the identity matrix. For simplicity, we let *P* represents $P(\Delta)$ in the rest of this paper. It is important to notice that even though we used a fixed sampling interval Δ in this paper, this is not a requirement for the model to work. One can easily imagine scenarios where it is desirable to sample the sensors and predict new system measures at irregular time intervals. In that case $P(\Delta)$ needs to be recomputed at each sampling instant τ

5.3 Interpreting Observations

Due to the inhomogeneity of sensor types, the observations can consist of a variety of information; different types of alarms, suspect traffic patterns, entries in log data files, input from network administrators, indications of system elements up and down, ongoing operational and maintenance activities, and so on. To formalize, we assume that any observation can be classified as one of the symbols in the finite symbol set $V = \{v_1, \ldots, v_M\}$. The sequence of observations that the monitor forwards to the estimator is denoted $Y = (y_1, y_2, \ldots)$, where $y_\tau \in V$ is the observation received at sampling instant τ. Based on *Y*, the estimator will estimate the system's current state, in terms of the state probability X^τ in (9). The estimator will receive observations originating from more than one sensor, and these sensors may provide different types of data or even inconsistent data. All sensors will not be able to register all kinds of activities, so one cannot assume that the estimator is able to resolve the correct state of the monitored system at all times. The observation symbols are therefore probabilistic functions of the system's Markov chain, i.e., the system's true state will be *hidden* from the estimator. This is consistent with the basic idea of hidden Markov models (HMMs), as described in [14].

5.4 State Probability Estimation

Each monitored system can be represented by a HMM, defined by the three tuple $\Lambda = (P, X^1, O)$. As previously discussed, $P = \{p_{ij}\}$ is the one-step transition probability matrix for the system, $X^1 = \{X_1^1, \ldots, X_N^1\}$ is the state probability distribution of the system when the sampling starts, i.e., at sample instant τ = *1*. If one does not know the initial state probability of the system, the elements in X^1 have to be estimated, for instance by using the system steady state probabilities in (4). $O = \{o_j(l)\}$ is the observation symbol probability matrix for a system during sampling. Its elements are $o_j(l) = P(y_\tau = v_l \mid z_\tau = j),\ 1 \le j \le N,\ 1 \le l \le M$ i.e., $o_j(l)$ represents the probability that a sensor will provide the observation symbol v_l when sampled, given that the system is in state *j*. The elements of *O* will therefore give an indication of the sensor's false-positive and false-negative effect on the security and dependability prediction process. Note that if there is more than one sensor monitoring a particular system, one should define a separate observation symbol probability vector O_k for each sensor *k*.

By using an observation y_τ and the HMM Λ, the estimator will compute and replace X^τ in (9) with $\hat{X}^\tau$, where $\hat{X}^\tau$ is the system's *most likely state probability* at sampling instant τ. This is done by means of Alg. 1. The complexity of the algorithm is $O(N^2)$ where *N* is the number of system states.

Algorithm 1 Estimate the current state probability

Require: y_τ, Λ {an obs. at sampl. instant τ, the HMM}

Ensure: $\hat{X}^\tau$ {the estimated state prob. at sampl. inst. τ }

```
if $\tau = 1$ then
  for $i=1$ to $N$ do
```

```
      $\alpha_i^\tau \leftarrow o_i(y_1)X_i^1$
      $\hat{X}_i^\tau \leftarrow \frac{\alpha_i^\tau}{\sum_{j=1}^N \alpha_j^\tau}$
    end for
  else
    for i=1 to N do
      $\alpha_i^\tau \leftarrow o_i(y_\tau)\sum_{j=1}^N \alpha_j^{\tau-1} p_{ji}$
      $\hat{X}_i^\tau \leftarrow \frac{\alpha_i^\tau}{\sum_{j=1}^N \alpha_j^\tau}$
    end for
  end if
return $\hat{X}^\tau = \left\{\hat{X}_1^\tau, \ldots, \hat{X}_N^\tau\right\}$
```

To see why Alg. 1 works, note that, given the first observation y_1 at τ = *1*, and the HMM $\Lambda = (P, X^1, O)$, the elements in a new initial state probability $\hat{X}^1$ can be estimated as

$$\hat{X}_i^1 = P(z_1 = i \mid y_1, \Lambda) = \frac{P(y_1, z_1 = i \mid \Lambda)}{P(y_1 \mid \Lambda)} = \frac{P(y_1 \mid z_1 = i, \Lambda)P(z_1 = i \mid \Lambda)}{P(y_1 \mid \Lambda)} \tag{11}$$

To find the denominator, one can condition on the first visited state and sum over all possible states

$$P(y_1 \mid \Lambda) = \sum_{j=1}^N P(y_1 \mid z_1 = j, \Lambda)P(z_1 = j \mid \Lambda) = \sum_{j=1}^N o_j(y_1)X_j^1. \tag{12}$$

Hence, by combining (11) and (12)

$$\hat{X}_i^1 = \frac{o_i(y_1)X_i^1}{\sum_{j=1}^N o_j(y_1)X_j^1}. \tag{13}$$

To simplify the computation of the estimated state probability at the τ't observation we use the *forward-variable* $\alpha_i^\tau = P(y_1 \ldots y_\tau, z_\tau = i \mid \Lambda)$, as defined in [14]. By using recursion, this variable can be calculated in an efficient way as

$$\alpha_i^\tau = o_i(y_\tau)\sum_{j=1}^N \alpha_j^{\tau-1} p_{ji}, \tag{14}$$

for τ>1. In the derivation of α_i^τ we assumed that y_τ depends on z_τ only, and that the Markov property holds. From (11) and (13) we find the initial forward variable

$$\alpha_i^1 = o_i(y_1)X_i^1 \tag{15}$$

when τ=1. Now we can use the forward variable α_i^τ to update the estimated state probability distribution by new observations. This is done by

$$\hat{X}_i^\tau = P(z_\tau = i \mid y_1 \ldots y_\tau, \Lambda) = \frac{P(y_1 \ldots y_\tau, z_\tau = i \mid \Lambda)}{P(y_1 \ldots y_\tau \mid \Lambda)} = \frac{P(y_1 \ldots y_\tau, z_\tau = i \mid \Lambda)}{\sum_{j=1}^N P(y_1 \ldots y_\tau, z_\tau = j \mid \Lambda)} = \frac{\alpha_i^\tau}{\sum_{j=1}^N \alpha_j^\tau}. \tag{16}$$

6. Making the System Predictions

The final step in the security and dependability assessment process illustrated in Fig 3 is that the predictor uses the estimated state probability distribution together with the state transition rate matrix to compute system measures. This section provides the algorithms (Alg. 2-3) together with a detailed explanation of the mathematical equations that are used to compute the $P_F(t)$ and *MTNF* measures.

6.1 Computing $P_F^\tau(t)$

Recall the definition of $P_F(t)$ provided in (7). To use the estimated state probabilities to compute the function at sample instant τ, Alg. 2 can be used.

Algorithm 2 Predict the $P_F^\tau(t)$ measures

```
Require: $Q^*$, $\hat{X}^\tau$ {the modified rate matrix, the estimated state prob. at τ}
Ensure: $P_F^\tau(t)$ {the predicted $P_F(t)$ at sample τ }
  for i=1 to N do
    $X_i^*(0) \leftarrow \hat{X}_i^\tau$
    $X_i^*(t) \leftarrow X_i^*(0)e^{Q^* t}$
    if $i \le K$ then
```

```
      $P_F^\tau(t) += X_i^*(t)$
    end if
  end for
return $P_F^\tau(t)$
```

As can be seen from the algorithm, at each sampling instant τ there are three main steps to perform. First, the algorithm sets the initial state probability equal to the estimated, i.e., $X^*(0) \leftarrow \hat{X}^\tau$. The $X^*(0)$ vector is then used when solving the system state equation defined in (2), i.e., $X^*(t) = X(0)^* e^{Q^* t}$. To solve the system state equation, the algorithm uses the Mathematica package "StateDiagrams.m" [6], which implements the theory in [3]. Then, in accordance to (7), the $P_F^\tau(t)$ function is computed as

$$P_F^\tau(t) = \sum_{i \in S_G} X_i^*(t). \qquad (17)$$

Even though this definition of $P_F^\tau(t)$ is very similar to the traditional definition of the system reliability function (see e.g., [5]), there is a crucial difference. As a consequence of the estimation process, we cannot make the usual assumption that the system state is good when computing (17). Because $\hat{X}^\tau$ is used to determine $X^*(0)$ in Alg. 2 it might be that $\sum_{i \in S_G} X_i^*(0) \neq 1$. Hence, to validate the predicted system measures, one should also use (17) to compute $P_F^\tau(0) = \sum_{i \in S_G} X_i^*(0)$, i.e., the probability that the system actually is in a good state at sampling instant τ.

6.2 Computing $MTNF^\tau$

To compute the mean time to next failure (*MTNF*) measure at sampling instant τ, Alg. 3 is used.

Algorithm 3 Predict the $MTNF^\tau$ measures

```
Require: $Q_1, \hat{X}^\tau$ {(a part of) the rate matrix, the estimated state prob. at τ}
Ensure: $MTNF^\tau$ {the predicted MTNF at sample τ }
  for j=1 to K do
    $\hat{\hat{X}}_j^\tau \leftarrow \frac{\hat{X}_j^\tau}{\sum_{j=1}^K \hat{X}_j^\tau}$
    define $-\sum_{i=1}^K T_i q_{ij} = \hat{\hat{X}}_j^\tau$
  end for
  solve for all $T_i$
  return $MTNF^\tau = \sum_{i=1}^K T_i$
```

The algorithm has been implemented in accordance to methodology in [3], slightly modified to fit into the context of the proposed security and dependability assessment architecture. Suppose the states are ordered, such that $S_G = \{S_1, \ldots S_K\}$ and $S_F = \{S_{K+1}, \ldots S_N\}$. Then Q can be written in partitioned form as

$$Q = \begin{pmatrix} Q_1 & Q_2 \\ Q_3 & Q_4 \end{pmatrix}, \qquad (18)$$

where the size of Q_1 is $K \times K$, the size of Q_2 is $K \times (N-K)$ and so forth. Now also the estimated state probability vector can be partitioned as $\hat{X}^\tau = \left\{\hat{X}_G^\tau, \hat{X}_F^\tau\right\}$ where $\hat{X}_G^\tau = \left\{\hat{X}_1^\tau, \ldots, \hat{X}_K^\tau\right\}$ and $\hat{X}_F^\tau = \left\{\hat{X}_{K+1}^\tau, \ldots, \hat{X}_N^\tau\right\}$.

To compute the system's expected time to next failure, one has to assume that the system is in one of the good states in S_G at sampling instant τ (otherwise *MTNF* = 0, since the system already has failed). Therefore, the estimated state probabilities in $\hat{X}_G^\tau$ must be re-normalized such that

$$\hat{\hat{X}}_G^\tau = \frac{\hat{X}_G^\tau}{\hat{X}_G^\tau h_K} \qquad (19)$$

where h_K is a column vector of *K* ones. Define $T = \{T_1, \ldots, T_K\}$. By solving

$$-TQ_1 = \hat{\hat{X}}_G^\tau \qquad (20)$$

the mean time to next failure for the system at the particular instant τ can be computed as

$$MTNF^\tau = \sum_{i=1}^{K} T_i \, , \qquad (21)$$

provided that the system is in any of the good states in S_G when sampled. The main difference between the $MTNF^\tau$ measure used in this paper and the *MTFF* (mean time to first failure) measure used in traditional dependability analysis, is that when computing *MTFF* the system is considered new when it starts to deliver its intended service, i.e., $X(0) = \{1,0,\ldots,0\}$. In contrast, $MTNF^\tau$ is computed from the estimated state probabilities rather than the initial system state probabilities. The advantage with our approach is that by computing $MTNF^\tau$ as proposed in (21) one can use the real-time observations provided by the monitoring architecture to make a better prediction of the system's expected time to next failure, and update this prediction whenever new information arrives. Hence, in contrast to the static *MTFF*, the $MTNF^\tau$ will be a *dynamic* system measure, more suitable for a real-time system assessment architecture. However, as previously discussed, since the $MTNF^\tau$ is conditioned on a good system state at sampling instant τ (i.e., that $z_\tau \in S_G$) the measure should always be evaluated together with the corresponding $P_F^\tau(0)$ to make sense. This will be illustrated in the case study in the next section.

7. Case Study: A Database Server

To illustrate the proposed approach, we model and simulate the security and dependability assessment process for a typical network service configuration consisting of a database service for a local area network (LAN). In this paper we consider a single server implementation only; however, the example model can easily be extended to the more commonly used distributed server implementation (see [16] for an example). Fig. 4 illustrates the database server that is to be assessed in this study.

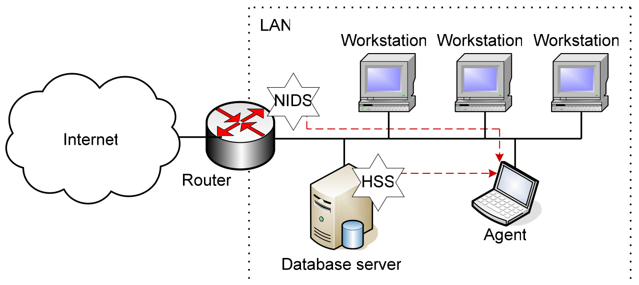

Fig. 4: The database server in its network environment.

In this example, the database server is assumed to be subject to accidental software-and hardware failures, as well as packet flooding Denial of Service (DoS) attacks originating from outside the LAN. As can be seen, the database server is monitored by a distributed agent-sensor system consisting of one agent that samples and interprets information from two different kinds of sensors; a network intrusion detection system (NIDS) and a host-based sensor system (HSS). The NIDS monitors traffic between the outside network and the internal LAN, and the HSS processes log files and checks system status locally at the database server.

7.1 The Stochastic Model

The database server can be modeled by a four-state CTMC. State *G* means that the server is fully operational, i.e., it is a system "up" state. In state *A* the server is subject to an ongoing DoS attack, which means that its performance is degraded so that the service is only partially available. Still, the *A* state is considered a system "up" state. If the DoS attack is detected and reacted to before the server crashes, the system will return to state *G*. In the "down" states *SF* and *HF* the server is subject to software and hardware failures, respectively. A hardware failure requires a manual repair. To recover from a software failure, only a server reboot is needed. Note that since also the effect of a successful DoS attack is a software failure requiring a reboot, we do not need to distinguish between accidental and malicious software failure modes in the stochastic model. Hence, the complete state set is $S = \{G, A, SF, HF\}$ whereof $S_G = \{G, A\}$ and $S_F = \{SF, HF\}$, as illustrated in Fig. 5.

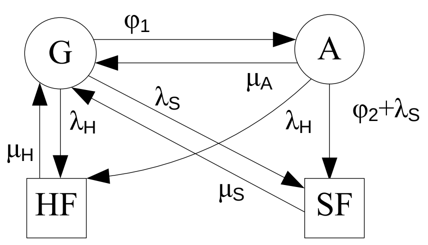

Fig. 5: The state transition diagram for the database server.

The time to failure, attack and repair are assumed to follow the exponential distributions $\lambda e^{-\lambda t}$, $\varphi e^{-\varphi t}$ and $\mu e^{-\mu t}$, respectively. The specific rates used in this example are $\lambda_S = 0.005$, $\lambda_H = 0.0003$, $\varphi_1 = 0.002$, $\varphi_2 = 60$, $\mu_A = 15$, $\mu_S = 0.25$ and $\mu_H = 0.04$ (h^{-1}). The rate transition matrix for the server, denoted Q_{server}, is

$$Q_{server} = \begin{pmatrix} -(\varphi_1+\lambda_S+\lambda_H) & \varphi_1 & \lambda_S & \lambda_H \\ \mu_A & -(\mu_A+\varphi_2+\lambda_S+\lambda_H) & \varphi_2+\lambda_S & \lambda_H \\ \mu_S & 0 & -(\mu_S+\lambda_H) & \lambda_H \\ \mu_H & 0 & 0 & -\mu_H \end{pmatrix}$$
$$= \begin{pmatrix} -0.0073 & 0.002 & 0.005 & 0.0003 \\ 15 & -75.0053 & 60.005 & 0.0003 \\ 0.25 & 0 & -0.2503 & 0.0003 \\ 0.04 & 0 & 0 & -0.04 \end{pmatrix}.$$

In this example we chose not to demonstrate how to incorporate attacker behavior in the rate matrix, but assume that a game model, Γ_{server}, has already been applied to obtain the attack probability parts of φ_1 and φ_2. The result from the game model affects the numerical values of the predicted system measures in the final step of the system assessment model but is otherwise not substantially important for understanding the functionality of the prediction architecture. The reader is therefore referred to the previously published paper [12] for a more illustrative case-study on this particular topic.

7.2 The Monitoring System

As illustrated in Fig. 4, the agent collects and interprets data from both the NIDS and the HSS. The observations are then forwarded to the estimator (not illustrated in the figure). In this example the observation symbol set is $V = \{g, a, sf, hf\}$ where symbol g is an indication of system state G, symbol a an indication of state A, and so forth. In this paper we do not focus on *how* the NIDS and HSS data is interpreted; we simple assume that the agent is able to map sensor data into symbols representing states.

The HMM representing this monitoring system is defined by the three-tuple $\Lambda = \left(P_{server}, X^1_{server}, O_{server}\right)$. The sampling interval is fixed to $\Delta = 15$ min. By using (10) we compute the one-step transition probability matrix as

$$P_{server} = \begin{pmatrix} p_{G,G} & p_{G,A} & p_{G,SF} & p_{G,HF} \\ p_{A,G} & p_{A,A} & p_{A,SF} & p_{A,HF} \\ p_{SF,G} & p_{SF,A} & p_{SF,SF} & p_{SF,HF} \\ p_{HF,G} & p_{HF,A} & p_{HF,SF} & p_{HF,HF} \end{pmatrix}$$
$$= \begin{pmatrix} 0.998 & 2.66\cdot10^{-5} & 1.58\cdot10^{-3} & 7.46\cdot10^{-5} \\ 0.246 & 6.49\cdot10^{-6} & 0.754 & 7.46\cdot10^{-5} \\ 0.061 & 1.53\cdot10^{-6} & 0.939 & 7.46\cdot10^{-5} \\ 9.94\cdot10^{-3} & 2.51\cdot10^{-7} & 7.85\cdot10^{-6} & 0.990 \end{pmatrix}.$$

As the initial sampling state distribution we use the steady state probabilities of the system

$$X^1_{server} = \left\{X^1_G, X^1_A, X^1_{SF}, X^1_{HF}\right\}$$
$$= \left\{0.967, 2.58\cdot10^{-5}, 2.55\cdot10^{-2}, 7.44\cdot10^{-3}\right\},$$

found by solving (5)-(6). One can see that the database most likely will be in state *G* when the sampling process starts. The observation symbol probability matrix for the database server is

$$O_{server} = \begin{pmatrix} o_G(g) & o_G(a) & o_G(sf) & o_G(hf) \\ o_A(g) & o_A(a) & o_A(sf) & o_A(hf) \\ o_{SF}(g) & o_{SF}(a) & o_{SF}(sf) & o_{SF}(hf) \\ o_{HF}(g) & o_{HF}(a) & o_{HF}(sf) & o_{HF}(hf) \end{pmatrix}$$
$$= \begin{pmatrix} 0.80 & 0.10 & 0.06 & 0.04 \\ 0.30 & 0.55 & 0.10 & 0.05 \\ 0.08 & 0.02 & 0.70 & 0.20 \\ 0.01 & 0.01 & 0.10 & 0.88 \end{pmatrix}.$$

Since both $o_G(sf)$, $o_G(hf)$, $o_A(sf)$ and $o_A(hf) \neq 0$, and $o_{SF}(g)$, $o_{SF}(a)$, $o_{HF}(g)$ and $o_{HF}(a) \neq 0$ in O_{server}, one can see that even though the sensors in this case study have relatively low false-positive and false-negative rates there is still room for the possibility of misleading observations. Note that we use a single observation symbol probability matrix to represent the trustworthiness of the (merged) data from both the NIDS and the HSS sensor. See e.g., [11] for an example of round robin sampling of sensors, or [4] for an algorithm for optimal selection of data from multiple sensors.

7.3 Simulation Results

To evaluate the effectiveness of the proposed prediction method, we simulate the following three different observation sequences

$$\begin{aligned} Y_1 &= (g,g,g,a,g,g,g,g,g,g), \\ Y_2 &= (g,g,a,sf,a,sf,sf,g,g,g), \\ Y_3 &= (g,g,sf,g,sf,g,hf,hf,hf,hf). \end{aligned}$$

The purpose of the first simulated sequence (Y_1) is to demonstrate how the prediction process reacts to a single ”attack” warning observation (*a*) that are preceded and followed by a number of ”good” observations (*g*). The second simulation (Y_2) demonstrates how the prediction algorithm reacts to alternate *a* and *sf* observations. The third sequence (Y_3) simulates a number of software failure observations that are indicated to be repaired, and finally followed by a protracted hardware failure.

The $P_F^\tau(t)$ functions

First, we discuss the predicted $P_F^\tau(t)$ functions for the three cases. From Fig. 6 it appears that $P_F^\tau(0)$ is very close to 1 for all samples $\tau = 1,\ldots,10$ when simulating Y_1. Fig. 7 shows a more detailed view of the simulation results from Y_1. One can see that the $P_F^\tau(t)$ graph is lower for the first sample, i.e., when τ= 1. This is because, in accordance to X^1_{server}, the server is assumed to be in state *G* with only 96.7% certainty when the sampling process starts. As the estimator receives more *g* symbols, the estimated probability of state *G* will rise, and hence, the corresponding $P_F^\tau(t)$ graph will rise. Note that since the fourth observation $y_4 = a$ in the first simulation, the $P_F^4(t)$ graph will be slightly lower than the subsequent predicted graphs.

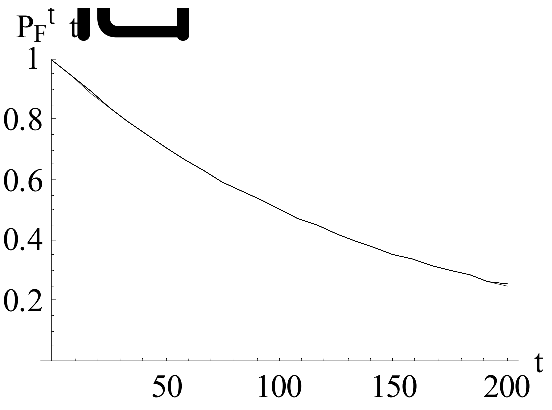

Fig. 6: An overview of the $P_F^\tau(t)$ graphs when simulating Y_1. The figure depicts the predicted $P_F^\tau(t)$ graphs at the sampling instants $\tau = 1,\ldots,10$.

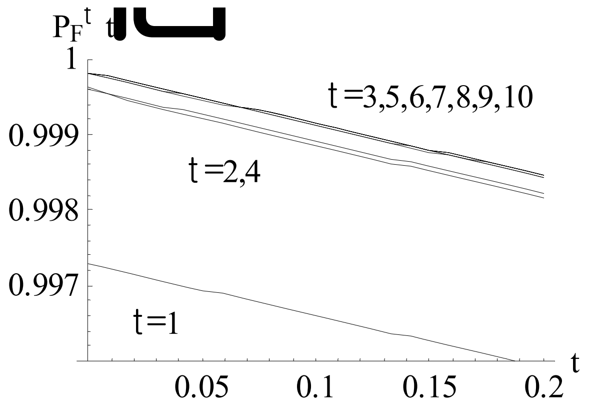

Fig. 7: A closer look at the $P_F^\tau(t)$ graphs when simulating $Y_1 = (g,g,g,a,g,g,g,g,g,g)$.

As can be seen from Fig. 8 and Fig. 9, $P_F^\tau(t)$ for the second simulation will be quite high until the estimator receives the first *sf* symbol at sampling instant τ = 4. Even though the next observation ($y_5 = a$) will rise the predicted graph, the next observation after that ($y_6 = sf$) will lower it even more. The lowest graph of them all will appear at sampling instant τ = 7, which is due to the two successive *sf* observations. Note that for the same reason $P_F^7(0) \approx 0.55$, since the system with a high probability already has failed.

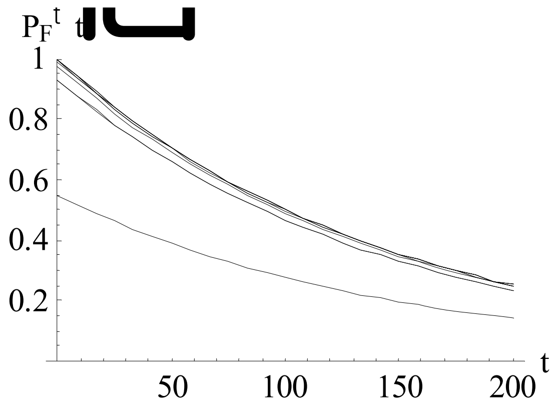

Fig. 8: An overview of the $P_F^{\tau}(t)$ graphs when simulating Y_2. The figure depicts the predicted $P_F^{\tau}(t)$ graphs at the sampling instants $\tau = 1,\ldots,10$.

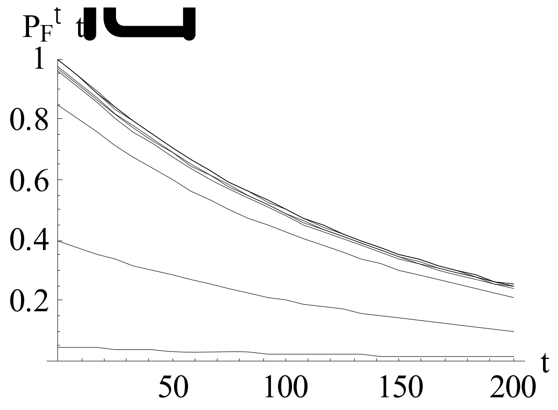

Fig. 10: An overview of the $P_F^{\tau}(t)$ graphs when simulating Y_3. The figure depicts the predicted $P_F^{\tau}(t)$ graphs at the sampling instants $\tau = 1,\ldots,10$.

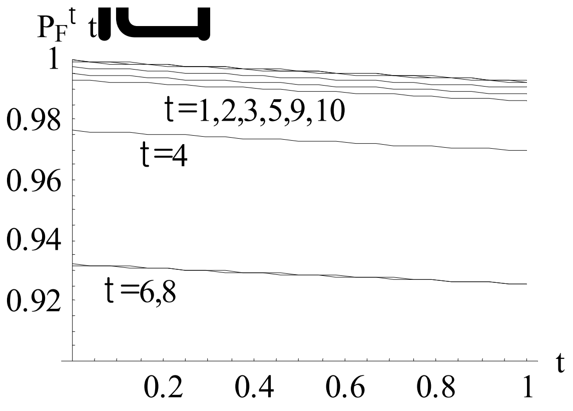

Fig. 9: A closer look at the $P_F^{\tau}(t)$ graphs when simulating $Y_2 = (g, g, a, sf, a, sf, sf, g, g, g)$.

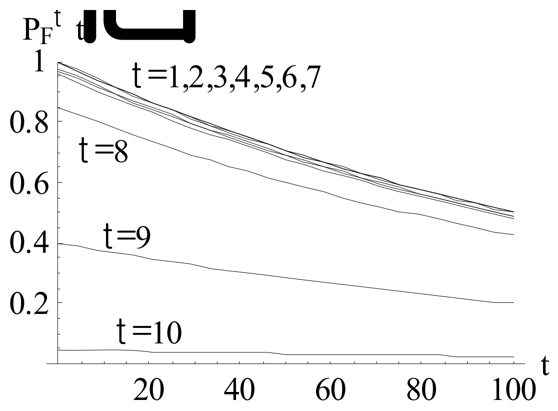

Fig. 11: A closer look at the $P_F^{\tau}(t)$ graphs when simulating $Y_3 = (g, g, sf, g, sf, g, hf, hf, hf, hf)$.

The result from the third simulation (Fig. 10, Fig. 11 and Fig. 12) shows that the alternating *g* and *sf* observations will give rise to corresponding $P_F^{\tau}(t)$ graphs. As the agent starts to receive *hf* (hardware failure) symbols, the predicted $P_F^{\tau}(t)$ graphs will decrease even more. Also $P_F^{\tau}(0) \to 0$ as $\tau \to 10$.

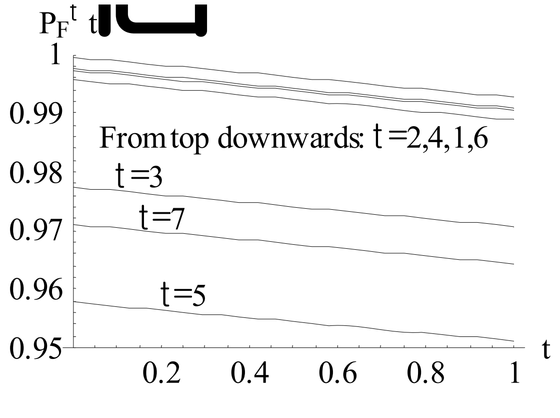

Fig. 12: A closer look at the top seven $P_F^{\tau}(t)$ graphs when simulating Y_3.

As indicated in Fig. 6, Fig. 8 and Fig. 10, $P_F^\tau(t) \to 0$ as $t \to \infty$ for all simulated graphs, i.e., even though the estimated state during sampling is likely to be good, the system will sooner or later fail.

The *MTNF* measures

The predicted $MTNF^\tau$ measures, together with the corresponding $P_F^\tau(0)$'s, are depicted in Fig. 13. During the first simulation (Y_1), the predicted *MTNF* measure drops as the *a* symbol is received (at sampling instant τ = 4), but returns to the same level as more *g* symbols are received. The corresponding $P_F^\tau(0)$ graph indicates that the predicted *MTNF* measures are very reliable ($P_F^\tau(0) \approx 1$), with an exception for the first sample ($P_F^1(0) = 0.998$).

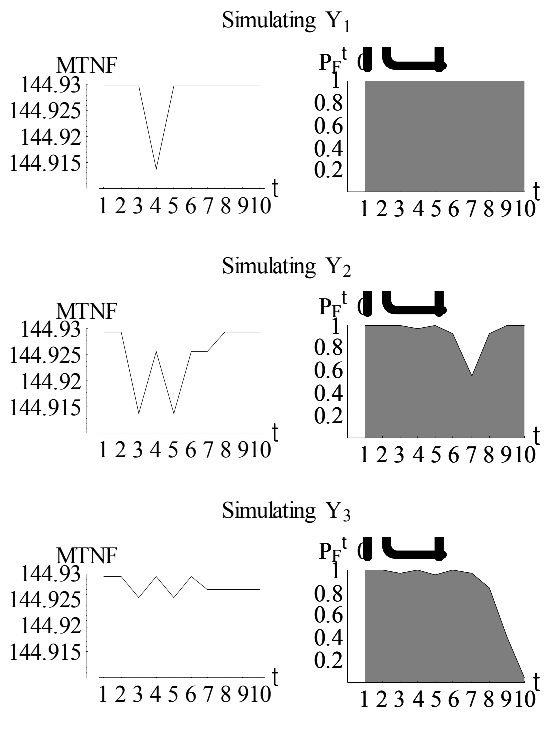

Fig. 13: The *MTNF* measures together with the corresponding $P_F^\tau(0)$'s.

From the second simulation we observe that since $y_3 = y_5 = a$, the predicted *MTNF* measure will be slightly lower at τ = 3 and τ = 5. Interestingly, *MTNF* will rise at τ = 4, τ = 6 and τ = 7, even though $y_4 = y_6 = y_7 = sf$. The corresponding $P_F^\tau(0)$ graph explains this phenomenon; since the graph is lower at τ = 4, 6, 7 the system may already be in a failed state at these particular sampling instants.

The results from the third simulation indicates that *sf* and *hf* symbols will lower the predicted *MTNF* measures. Since the simulated trace ends with four subsequent *hf* symbols (at τ =7, 8, 9, 10), $P_F^\tau(0) \to 0$ as $\tau \to 10$.

8. Concluding Remarks

This paper presents a framework for integrated security and dependability assessment of computing systems. By using data provided by a monitoring architecture, the current system state and its future behavior can be predicted in real-time. The proposed method for computing system measures is based on Markov analysis, where some of the model parameters are determined by a game theoretic analysis of attacker behavior. To demonstrate the feasibility of the proposed prediction architecture, we performed three simulation experiments for a small case study.

The stochastic modeling approach used in this paper relies on a few assumptions. First, we assume that the security and dependability behavior of the system can be modeled as a Markov process, which means that the conditional probability distribution of future states of the process depends only upon the current state. Even though it is very common to assume these properties when modeling and analyzing systems in a traditional dependability context (considering accidental failures only), it is not (yet) a well established practice in the security community. Second, the HMM approach relies on independent observations, which means that the observations that a sensor produces depend on the current system state only, and not on any previous observations. The main drawback of this approach is that, because security indications and alerts can be highly correlated, the sampling interval must be large enough so that the observations received by the estimator can be considered independent, for the model to be valid. Of course the exact lower limit for the sampling interval will depend on the particular system that is to be assessed, and on the types of sensors that monitors the system. As an example, for the database server in the case study 15 minutes was suggested as a reasonable sampling interval.

The case study used to demonstrate the approach in Section 7 is kept simple for illustration. In the future we plan to model and simulate the security and dependability assessment process for a more extensive example. A validation by a prototype system also remains.

References

[1] Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. IEEE Transactions on Dependable and Secure Computing, 1:11–33, January-March 2004

[2] K. B. B. Madan, K. Vaidyanathan Goseva-Popstojanova, and K. S. Trivedi. A method for modeling and quantifying the security attributes of intrusion tolerant systems. In Performance Evaluation, volume 56, 2004.

[3] John A. Buzacott. Markov approach to finding failure times of repairable systems. IEEE Transactions on Reliability, R-19:128–134, November 1970.

[4] R. Evans, V. Krishnamurthy, G. Nair, and L. Sciacca. Networked sensor management and data rate control for tracking maneuvering targets. IEEE Transactions on Signal Processing, 53:1979–1991, June 2005.

[5] Bjarne E. Helvik. Dependable Computing Systems and Communication Networks, Design and Evaluation. Draft lecture notes (256 p.), Department of Telematics, NTNU, Nov. 2003.

[6] Bjarne E. Helvik. Statediagrams.m; a Mathematica package for dependability analysis of systems by CTMC. Unpublished NTNU, 2002.

[7] ISO/IEC 13335: Information Technology -Guidelines for the management of IT Security. http://www.iso.org

[8] S. Jha, O. Sheyner, and J. Wing. Two formal analyses of attack graphs. In Proceedings of the 2002 Computer Security Foundations Workshop, 2002.

[9] 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, Oct 1993.

[10] David M. Nicol, William H. Sanders, and Kishor S. Trivedi. Model-based evaluation: From dependability to security. IEEE Transactions on Dependable and Secure Computing, 1:48–65, January-March 2004.

[11] Andre Årnes, Karin Sallhammar, Kjetil Haslum, Tønnes Brekne, Marie Elisabeth Gaup Moe, Svein Johan Knapskog. Real-time Risk Assessment with Network Sensors and Intrusion Detection Systems. In International Conference on Computational Intelligence and Security (CIS), Dec 2005

[12] Karin Sallhammar, Bjarne E. Helvik and Svein J. Knapskog. Towards a stochastic model for integrated security and dependability evaluation. In Proceedings of the First International Conference on Availability, Reliability and Security (AReS), 2006.

[13] R. Ortalo and Y. Deswarte. Experimenting with quantitative evaluation tools for monitoring operational security. IEEE Transactions on Software Engineering, 25(5):633–650, Sept/Oct 1999.

[14] Lawrence R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Readings in speech recognition, pages 267–296, 1990.

[15] Sheldon M. Ross. Introduction to Probability Models. Academic Press, 8th edition, 2003.

[16] Mehmet Savsar and Fawaz S. Al-Anzi. Reliability of data allocation on a centralized server configuration with distributed servers. The Computer Journal, 49:258–267, 2005.

[17] F. Stevens, T. Courtney, S. Singh, A. Agbaria, J. F. Meyer, W. H. Sanders, and P. Pal. Model-based validation of an intrusion-tolerant information system. In Proceedings of the 23rd Symposium on Reliable Distributed Systems (SRDS 2004), Oct 18-20 2004.

[18] Wei Wei, Bing Wang, and Don Towsley. Continuous-time Hidden Markov Models for Network Performance Evaluation. Performance Evaluation, 49:129–146, 2002.

Karin Sallhammar received her Civ.ing. degree (M.S.) in Media Technology and Engineering from the Linköping University, Sweden in 2003. She is currently a Ph.D. candidate at the Norwegian University of Science and Technology (NTNU), the Department of Telematics, and is associated with the Norwegian Centre of Excellence (CoE) for Quantifiable Quality of Service in Communication Systems (Q2S).

Her field of interests includes information security primitives and protocols, network monitoring and intrusion detection, as well as dependability modeling and analysis. Her current research focus is on stochastic models for security and dependability evaluation.

Bjarne E. Helvik received his Siv.ing. degree (M.S.E.E.) from the Norwegian Institute of Technology (NTH), Trondheim, Norway in 1975. He was awarded the degree Dr. Techn. from NTH in 1982. He has since 1997 been Professor at the Norwegian University of Science and Technology (NTNU), the Department of Telematics. He is principal academic at the Norwegian Centre of Excellence (CoE) for Quantifiable Quality of Service in Communication Systems (Q2S). He has previously held various positions at ELAB and SINTEF Telecom and Informatics. In the period 1988-1997 he was appointed as Adjunct Professor at the Department of Computer Engineering and Telematics at NTH.

His field of interests includes QoS, dependability modeling, measurements, analysis and simulation, fault-tolerant computing systems and survivable networks as well as related communication system architectural issues. His current research focus is on distributed, autonomous and adaptive fault-management in telecommunication systems, networks and services.

Prof. Helvik is a member of the IEEE, has been active in RACE, ACTS, IST and EURESCOM collaborations, and in a number of program committees. Among them committees for the International Symposium on Fault-Tolerant Computing (FTCS), the European Dependable Computing Conferences (EDCC), the ITC Specialist Seminars, the International Teletraffic Congress (ITC), GlobeCom and is co-chairing the EURONGI 2007 conference. He has authored/co-authored a number of research papers and textbooks.
.

Svein J. Knapskog received his Siv.ing. degree (M.S.E.E.) from the Norwegian Institute of Technology (NTH), Trondheim, Norway in 1972. Since 2001, he has been Professor at the Norwegian University of Science and Technology (NTNU), the Department of Telematics. He is presently principal academic at the Norwegian Centre of Excellence (CoE) for Quantifiable Quality of Service in Communication Systems (Q2S). He has previously held various positions in Norwegian public sector, SINTEF and industry. From 1982 - 2000, he was Associate Professor at NTH (later NTNU), Department of Computer Science and Telematics, where he also served a three year term as Head of Department. In the academic year 2005-2006 he has been acting Head of Department of the Department of Telematics.

His field of interests includes information security and QoS as well as related communication system architectural issues. His current research focus is on information security primitives, protocols and services in distributed autonomous telecommunication systems and networks, and security evaluation thereof.

Prof. Knapskog has been active in a number of conference program committees and has authored/co-authored a number of technical reports, research papers and a textbook (in Norwegian)