REAL-TIME NETWORK INTRUSION PREVENTION

Thesis for the degree of philosophiae doctor

KJETIL HASLUM

Department of Telematics
Norwegian University of Science and Technology
May 2010

Abstract

It is not economically or technically feasible to make complex computer systems that are completely secure. New attacks are constantly developed by attackers and the security situation can therefore rapidly change. In order to detect and stop attackers before any damage is done, automated tools have to be deployed because there is not enough time for manual intervention. Therefore there is a need for online risk assessment and proactive defense mechanisms like Intrusion Prevention System (IPS). In the area of computer security there have been only a few quantitative security measures until now, and there are few published cases for methods and tools based on such measures.

The main areas of this thesis are: Quantitative characterization of risk and security in computer systems or networks; and dynamic risk and security assessment based on network monitoring. During our research, the focus has been narrowed down to look for answers to the following problems:

- Is it possible (and practical) to reuse some of the stochastic modeling techniques used to model dependable systems?
- Can Hidden Markov Models (HMMs) be successfully used in real time risk assessment?
- Is it feasible to prevent attacks against systems and networks based on risk assessment?

For these problems a Markov model describing the interaction between the system and attackers in a quantitative manner is proposed. The Markov model describes the different security states of a network, and the transitions between them.

Based on the initial Markov model, a HMM modeling the trustworthiness of sensors collecting security relevant information in a computer network is proposed. The sensor model is used for online risk assessment based on observations from sensors in a network. A security measure called intrusion frequency is used. The intrusion frequency is estimated from the state distribution estimated by the HMM. The sensor model has been validated through simulations, and through experiment with synthetic and real network traffic.

Two different approaches to online risk assessment are proposed: one based on costs associated width states and one based on a hierarchical fuzzy inference system. Three different methods for aggregation of alerts from multiple network sensors are discussed. The first method was to use the average of the risk estimated by each sensor, this solution have some obvious drawbacks e.g. when the risk from two sensors are aggregated where one is very trustworthy and one is very little trustworthy, in this case we would have been better off using only the risk from the most trustworthy sensor instead of the average. The second method produces a minimum variance estimator of the risk. This solution is based on a strict assumption on independence between sensors. In the third proposal, one common distribution over the security state space is maintained. The distribution is updated when an observation is received, using the sensors of the corresponding HMM. The fine tuning of the fuzzy logic based risk assessment is achieved using a neural network learning technique. A Distributed Intrusion Prevention System (DIPS) architecture based on fuzzy online risk assessment is presented as a practical application of the models developed in thesis.

Preface

This thesis is submitted in partial fulfillment of the requirements of the degree of philosophiae doctor (PhD) at the Norwegian University of Science and Technology (NTNU). The PhD study was conducted in the period January 2004 to April 2008. During the study period, I have been hosted by the Centre for Quantifiable Quality of Service in Communication Systems (Q2S), Centre of Excellence. Q2S is established and fully funded by the Research Council of Norway, NTNU, UNINETT, and Telenor. Academically, the PhD study was under the responsibility of the Department of Telematics, NTNU, and has been supervised by Professor Svein Johan Knapskog.

The document has been formatted in LaTeX using a modified version of the document class *kapproc.cls* provided by Kluwer Academic Publishers.

Acknowledgements

First of all I would like to thank my supervisor Professor Svein J. Knapskog, who has been a great support during my PhD study at Q2S, and Ajith Abraham for being my Co-supervisor in part of my PhD study. I would like to thank my colleague and Co-author Marie Elisabeth Gaup Moe for good cooperation during my stay at Q2S, and the rest of my Co-authors: André Årnes, Karin Sallhammar and Tønnes Brekne, all being very supportive and encouraging during the whole research period and the subsequent writing of the papers included in this thesis. I would also thank Professor Bjarne Helvik for his insightful feedback to some of the papers in this thesis.

The administrative and technical staff have been very helpful during my stay at Q2S, and I would give a great thank to Anniken Skotvoll for helping me with administrative tasks; Pål Sæther and Hans Almåsbakk for excellent technical support, particularly during my teaching and supervising activities at the Department of Telematics. It has been a great time working at the Q2S Centre, and I would like to thank all my colleagues for a friendly working environment, and my friends for fruitful discussions.

I would also like to thank Telenor for financially supporting my PhD position at Q2S.

Contents

List of Papers

- PAPER A:
  André Årnes, Karin Sallhammar, Kjetil Haslum, Tønnes Brekne, Marie Elisabeth Gaup Moe and Svein Johan Knapskog. *Real-time Risk Assessment with Network Sensors and Intrusion Detection Systems.* In Proceedings of the 2005 International Conference on Computational Intelligence and Security (CIS'05). Xian, China. December 15-19, 2005.

- PAPER B:
  André Årnes, Karin Sallhammar, Kjetil Haslum, and Svein Johan Knapskog. *Real-time Risk Assessment with Network Sensors and Hidden Markov Models.* In Proceedings of the 11th Nordic Workshop on Secure IT-systems. Linköping, Sweden. October 19-20, 2006.

- PAPER C:
  Kjetil Haslum, André Årnes. *Multisensor Real-time Risk Assessment using Continuous-time Hidden Markov Models.* In Proceedings of the 2006 International Conference on Computational Intelligence and Security (CIS'06). Guangzhou, China. November 3-6, 2006.

- PAPER D:
  Kjetil Haslum, Ajith Abraham, and Svein Knapskog. *DIPS: A Framework for Distributed Intrusion Prediction and Prevention Using Hidden Markov Models and Online Fuzzy Risk Assessment.* In Proceedings of the Third International Symposium on Information Assurance and Security. Manchester, United Kingdom. August 29-31, 2007.

- PAPER E:
  Kjetil Haslum, Ajith Abraham, and Svein Knapskog. *Fuzzy Online Risk Assessment for Distributed Intrusion Prediction and Prevention Systems.* In Proceedings of EUROSIM/UKSIM. Cambridge, England. April 1-3, 2008.

- PAPER F:
  Kjetil Haslum, Ajith Abraham and Svein Knapskog. *HiNFRA: Hierarchical Neuro-Fuzzy Learning for Online Risk Assessment.* In Proceedings

of the Second Asia International Conference on Modeling & Simulation. Kuala Lumpur, Malaysia. May 13-15, 2008.

- PAPER G:
  Kjetil Haslum, Marie Elisabeth Gaup Moe, and Svein Knapskog. *Real-time Intrusion Prevention and Security Analysis of Networks using HMMs.* In Proceedings of the Fourth IEEE LCN Workshop on Network Security (WNS 2008) Montreal, Canada. October 17, 2008.

Abbreviations

AAFID Autonomous Agents for Intrusion Detection

ANFIS Adaptive Network Based Fuzzy Inference System

ANN Artificial Neural Network

CC Common Criteria

CTMM Continuous Time Markov Model

DIDS Distributed Intrusion Detection System

DIPS Distributed Intrusion Prevention System

DMZ Demilitarized Zone

DNS Domain Name System

DOS attack Denial-of-service attack

DP Dynamic Positioning

DTMC Discrete Time Markov Chain

EAL Evaluation Assurance Level

EMERALD Event Monitoring Enabling Responses to Anomalous Live Disturbances

FAIR Factorial Analysis of Information Risk

FIS Fuzzy Inference System

GPS Global Positioning System

GSPN Generalized Stochastic Petri Net

HIDS Host based IDS

HMM Hidden Markov Model

IA Intrusion Attempt

IDMF Intrusion Detection Message Exchange Format

IDS Intrusion Detection System

IP Intrusion in Progress

IPS Intrusion Prevention System

ITUA Intrusion Tolerant by Unpredictable Adoption

KF Kalman Filter

LAN Local Area Network

METF Mean Effort To Security Failure

MTFF Mean Time To First Failure

MTNSF Mean Time to Next Security Failure

MTSF Mean Time To Security Failure

NIDS Network based IDS

NIST National Institute of Standards and Technology

N Normal

NVD National Vulnerability Database

PID Proportional Integral differential

QoS Quality of Service

RIPPER Repeated Incremental Pruning to Produce Error Reduction

SA Successful Attack

SITAR scalable intrusion tolerant architecture

SLA Service Level Agreement

SRN Stochastic Reward Nets

STAT State Transition Analysis Technique

STDE Sequence Time-Delay Embedding

TCP Transmission Control Protocol

TSK Takagi-Sugeno-Kang

I

THESIS SUMMARY

1. Introduction

In today's society most critical infrastructures are based on dependable computer networks. However in many cases the networks can be reachable by attackers. As a consequence, the security of these critical systems, in addition to traditional Quality of Service (QoS) aspects like availability, reliability and performance, become crucial. In order to specify necessary security requirements, there is a need for quantifying of security attributes. We also need to have methods for monitoring the security of the system, and means to detect security breaches and possibly prevent intrusions.

Firewalls and Intrusion Detection Systems (IDSs) are traditionally used for stopping suspicious traffic at the network perimeter and for detecting of suspicious activity inside the network. Firewalls make use of static filtering rules, and with a proper configuration, the most obvious malicious traffic can be detected and denied to enter a network. However, clever attackers can always find ways to circumvent these rules. Attacks launched from inside an organization are also a serious concern, since they can not be prevented by a firewall.

IDSs may discover attacks based on anomaly detection or signature matching, but have no protection mechanisms against these attacks. Intrusion Prevention System (IPS), are proactive defense mechanisms designed to detect malicious packets embedded in normal network traffic and stop intrusions, blocking the offending traffic.

IPSs have the same detection capabilities as IDSs, with the addition of mechanisms that allow for stopping detected intrusions and introduce defense mechanisms to reduce the probability of success of future attacks. Their detection capabilities can be based on blacklists, whitelists and thresholds. Examples of prevention capabilities are response techniques that stop the attack itself by terminating a network connection or a user session and response techniques that block access to a specific target. The IPS can also change the security environment by reconfiguring a firewall, router or switch to prevent access from a specific attacker IP address. Some IPS technologies can even remove or replace malicious portions of traffic possibly constituting an attack.

The use of tools for monitoring the security state of assets in a network is an essential part of network management. Traditional risk assessment methodologies provide a framework for manually determining the risks of assets, and intrusion detection systems can provide alerts regarding security incidents, but these approaches do not provide a real-time high level overview of the risk level of assets.

2. Thesis Idea and Motivation

Experience has shown that it is hard, maybe even impossible, to design and implement large systems without any weaknesses. Starting with this fact, the motivation of our work is that the only viable option is to make networked systems reasonable secure. The security measures should be relevant to the threats against the values handled by the system, as well as the expected knowledge level of the potential intruders or attackers.

The security of computer systems has traditionally been assessed from a qualitative point of view, using some criteria. The most well known of these are the Trusted Computer System Evaluation Criteria published by Department of Defense [Def83] (also known as the Orange Book), and more recently the Common Criteria (CC), published by both NIST and ISO [ISO05]. Assessment based on criteria focus on the presence and absence of certain functional characteristics and the use of specific development techniques.

There are some obvious weaknesses with this type of assessments: it is qualitative and therefore gives no quantitative measures of the security. It is usually done during or immediately after the design and production phases (pre-deployment) and is therefore not assessing the security situation experienced by an operational system on a day to day basis.

Existence of effective methods for quantifying and characterizing security will make it much easier to specify and measure security. The existence of quantitative security measures will enable us to specify security requirements in a Service Level Agreement (SLA), similar to how dependability requirements like availability and reliability are specified today. When the system or service is operational, it should be possible to measure the actual security of the system in order to check that it is within the requirements specified in the SLA.

2.1 Research Questions

The focus of this thesis was intended to be on methods for characterizing and quantifying security in dynamic and complex computer systems. The main research areas investigated in the thesis to achieve this are:

- Quantitative characterization of risk and security in computer systems or networks.
- Dynamic risk and security assessment based on network monitoring.

During our research, the focus has been narrowed down to look for answers to the following problems:

- Is it possible (and practical) to reuse some of the stochastic modeling techniques used to model dependable systems?
- Can Hidden Markov Models (HMMs) be successfully used in real time risk assessment?
- Is it feasible to prevent attacks against systems and networks based on risk assessment?

3. Research Assumptions

When using a simplified model of a real system we have to make some assumptions. This section will mention the most important assumptions made for the research described in this thesis, and give some arguments for the validity of the assumptions.

- When using a Markov model to model the interaction between attackers and a system, we assumed the history of the system to be contained in the state of the system. Jonsson and Olovsson [JO97] suggest that a typical attack can be divided into three different phases based on the attacker's behavior: the learning phase, the standard attack phases and the innovative attack phase. In our research we have taken a more system centric approach and models the impacts on the system rather than the specific attack phase. A similar model with only three states is used in a HMM based IDS in [KL07]. Gong et al. [GGPW+01] uses a state transition model with nine states to model an intrusion tolerant system. One approach could be to use automated tools for making more realistic models, and then apply lumping and aggregation methods to reduce the number of states. The validity of the assumption that it is possible to model the system's security state using only a limited number of states is further discussed in Section 7.

- State occupation times are assumed to be exponentially distributed, when using the continuous time Markov model. We have also used a discrete time Markov model which implies that the state occupation times have to be geometrically distributed. This assumption is discussed without a clear conclusion in [LBF+93], they discuses different reason whey the failure rate might be constant (corresponding to an exponential distribution) decreasing or increasing. An argument of a constant failure rate could be that the population of possible attackers are big, and therefore will have a more random nature than an individual attacker. We also assume model parameters like transition rates to only be valid for shorter periods like hours, to take into account that there might be an increased attacker activity on specific time and dates. Jonsson and Olovsson [JO97] have made experiments that indicate that time between breaches is exponentially distributed. Results from experiments with honeypots by Kaâniche et al. [KAN+06] indicate that it may actually be better to use a semi-Markov model. This comes at a cost of increased algorithm complexity, and we see this as a possible extension of our work when real life data are available.

- By using a HMM to model sensors, we have assumed observations to be independent given the current state. This assumption is justified by assuming that there are some prepossessing of alerts in order to group together alerts originating from the same root cause, using some correlation and aggregation technique. For more details see Paper G.

- In one of the papers we used minimum variance to combine risk from individual sensors. We assume that the risk estimated from each sensor is unbiased and independent. As this assumption is questionable, we have also investigated other methods for combining data from multiple sensors, as seen in Paper G.

In addition to these basic assumptions there are some assumptions about attackers that are discussed in Section 4.2, and some assumptions about sensors that are discussed in Section 4.4.

4. Background

4.1 Stochastic Modeling of Security

This section defines some basic concepts and measures related to security, and introduce the stochastic modeling approach used in this thesis. The modeling approach is state based and similar to the approach used to model dependable systems. On the other hand, static methods like fault trees [NST04] and block diagrams [NST04] are not considered in this thesis.

We start with some basic definitions that are used in the description of the stochastic models the security in computer networks, introduced later in this section.

Definition 1 *A security policy is a set of security-motivated constraints that are to be adhered to by, for example, an organization or a computer system [ALRL04].*

The security policy may be enforced by technical means, management actions or operational control, or a combination of these.

Definition 2 *A deviation from the security policy will be regarded as a security failure [ALRL04].*

A consequence of this definition is that the same event within two different organizations may be considered as a security failure in one organization and as a normal situation in another organization, depending on the organization's security policy.

Definition 3 *The adjudged or hypothesized cause of an error is called a fault [ALRL04].*

Faults can be classified as [ALRL04]:

- Development faults which include all fault classes occurring during development.

- Physical faults which include all fault classes that affect hardware.
- Interaction faults which include all external faults.

Faults are divided into internal and external faults. Internal faults include the development and physical faults and will not affect the external state of the system in such a way that it will influence the system's ability to deliver a correct service. Conversely, external faults or interaction faults will affect the service delivered by the system.

Definition 4 *A vulnerability is an internal fault that enables an external fault to harm the system [ALRL04].*

A vulnerability may be located in the design, the code or the configuration of the system. Some examples of vulnerabilities are: weak passwords, configuration mistakes and software bugs.

Definition 5 *The security state of a system are described by several aspects of the computer network and its environment.*

These aspects could for instance be the security condition of the network, if security updates and patches have been installed, how the security policies are implemented, what attack countermeasures are in place, the existence of intrusion detection systems and so on.

In any modeling effort there is always a trade-off between model complexity and model precision.In our approach, we seek to model the interaction between the system and the attackers by describing the security state of the system in terms of the different stages of a typical attack against a computer network.

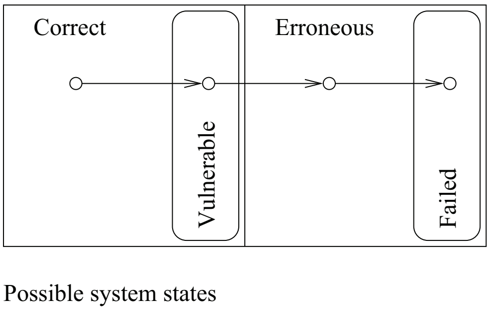


Figure 1. Partitioning of system states

Figure 1 illustrates a classification of system states, where all system states are either classified as correct or erroneous. The correct states contain a subset of vulnerable states, and the erroneous states contain a subset of failed states. The correct states are the states where there are no internal or external errors in the system. The vulnerable states are those states where the system contains one or more unexploited vulnerability. The set of erroneous states are those states where a vulnerability has been exploited by an attacker and there is an internal error in the system. The system may propagate through one or more erroneous states before the security policy is broken and the system enters a failed state. It is also possible that the system goes back into one of the correct states without entering a failed state. The system may already be in a vulnerable state when it starts operating, because of vulnerabilities introduced during the system design. Two faults are needed in order to transfer the system into an erroneous state from a correct and non vulnerable state: one for transferring the system into a vulnerable state and one more for transferring the system from a vulnerable state and into an erroneous state. The state model is based on the fault-error-failure terminology known from the dependability community [ALRL04].

The relation between security and dependability attributes is illustrated in Figure 2.

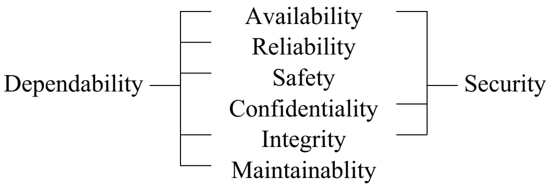


Figure 2. Attributes of security and dependability.

Definition 6 *Dependability is an integrated concept of the five attributes: availability, reliability, safety, integrity and maintainability [ALRL04].*

Definition 7 *Security can be defined as a composite of three attributes: confidentiality, integrity and availability [ALRL04].*

The attributes of availability and security is future explained here:

- Availability: readiness for correct service.
- Confidentiality: absence of unauthorized disclosure of information.
- Integrity: absence of improper system alternation.
- Maintainability: ability to undergo modifications and repairs.
- Reliability: continuity of correct service.
- Safety: absence of catastrophic consequences on the user(s) and the environment.

Availability could be measured asymptotically as the percentage of time the system is available, or as instantaneous availability which is the probability that the system is available at a specific time. The system is available in a dependability context if it is within the service specification, and it is available in a security context if the security policy is not broken. Typical breaches to the availability attribute of a system could be a Denial-of-service attack (DOS attack).

Confidentiality is one of the most well known security attributes, and is often the main motivation for using cryptography. Breaches on a system's confidentiality property could typically happen if someone breaks into a system and copies or reads unauthorized information like a document containing sensitive information or sensitive system information, e.g. a password. It is in general not possible to repair a confidentiality failure, because it is not possible to get people to forget or delete all copies of the information once they are obtained.

The integrity of a system is broken if the system is altered in an unauthorized way, e.g. someone breaks into a system and deletes or modifies a file or document. Integrity failures could also be caused by viruses or insiders with malicious intentions. Integrity failures can often be detected and repaired, e.g. by restoring a backup.

The reliability of a system can be measured by its reliability function, the failure rate or Mean Time To First Failure (MTFF). Reliability could also be considered in a security context. We will do that in Paper F, where we will calculate Mean Time To Security Failure (MTSF). Taking the inverse of the MTSF we get what we call the intrusion frequency.

In this thesis, we consider only first order Markov processes, where next state transition only depend on current the state. This type of process is also known as a Markov chain. The term Markov model and Markov chain will be used interchangeably both referring to discrete and continues time first order Markov processes.

We will now discuss the dynamic aspects of the system model that is related to time between state transitions. In this thesis, we apply the Markov modeling approach, which is also frequently used to model dependability attributes of a system. We use the Markov model to model the interaction between the system and an attacker. A Markov model is a stochastic process that has a set of states, as described earlier in this section. The name Markov models refers to the Markov assumption, which states that the next state transition only depend on the current state. By making this assumption, the behavior of the model can be described by a transition rate matrix describing how the system moves from state to state, with an initial distribution describing the starting conditions.

Figure 3 shows the relation between the partitioning of the system states shown in Figure 1, and the states in the Markov model used in the four last papers (Paper D-G). In this model all system states are lumped into the following four super states:

1 Normal (N) indicates no suspicious activity against the network

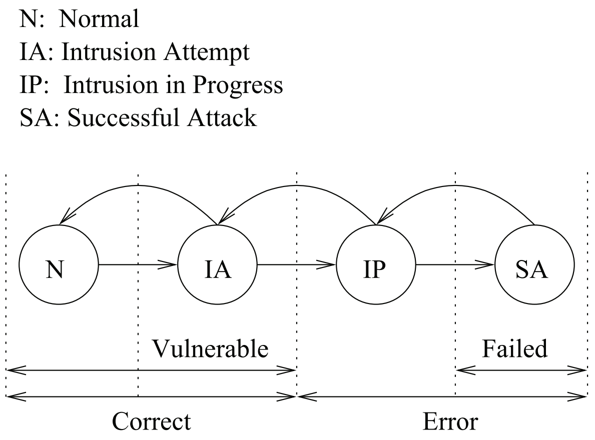


Figure 3. A Markov model used in security modeling of computer systems

2 Intrusion Attempt (IA) indicates that one or more attackers are trying to gather information about the system (for possible use in a future intrusion attempt)

3 Intrusion in Progress (IP) indicates that one or more attackers have started an attack against the system. The system is still functioning correctly and no confidentiality or integrity breaches have occurred

4 Successful Attack (SA) indicates that one or more attackers have broken into the system and may have full control over the system

In the three first papers (Paper A-C) a model consisting of three states was used. The states were called: Good (G), Attacked (A) and Compromised (C). The difference between the two models are the splitting of the attack state into the two states IA and IP, making the model more flexible.

The Markov model of the system's security states is not intended to model the system during the designs phases, and the model is therefore only valid after the system has been deployed. For this thesis it is assumed that there are introduced some vulnerabilities into the system during design and deployment of the system. The system is therefore assumed to be vulnerable when it is in the N state.

When one or more attackers have started investigating the system in order to find some vulnerabilities that could be exploited, the system moves into the IA state.

When the attacker finds a vulnerability and starts an active attack which leads to an internal error state, the system moves into the IP state.

If the attack is successful and the system moves into an internal error state such that the security policy is broken, the system moves into the SA state.

4.2 Attacker Model

This section describes the attacker model used in this thesis. Specific attacks or individual attackers are not modeled. Only the overall system state is modeled with respect to security. The number of attackers is not included in the model, i.e. we assume that the number of attackers that have broken into the system is irrelevant. One attacker can do as much damage as many attackers. Another reason for not modeling the specific number of attackers is that it is very difficult to distinguish between attackers given that they can use a bot-net or cooperate otherwise. An increasing activity from attackers will of course be reflected in an increased probability of being in one of the attacking states. The intelligence of the attacker does not enter into the model, i.e. the attackers's knowledge about the system, cooperation with other attackers, memory constraints or constraints in his/her processing capacity are not taken into account. The reason for not modeling these factors is mainly to keep the model complexity at a reasonable level.

The following assumptions have been made about the attacker's behavior:

- There are always some vulnerabilities in the system that the attacker can benefit from. This assumption is made in Paper A, indicating that it is unrealistic to have a large computer system without some vulnerabilities.
- Time between attacks is assumed to be exponentially distributed within a shorter time period. On a longer perspective the parameters are assumed to drift, but this could be remedied by re-estimation.
- The attacker may give up during an attack. The probability that an attacker gives up or the attack is stopped by some prevention mechanisms is merged, since the effect on the system is the same.
- There could be unsuccessful repair, leading to known vulnerabilities remaining in the system after it has been repaired.
- Time is assumed to be a representative measure of the attacker's effort.

4.3 Sensor Model

Section 4.1 explained how a system's security state can be modeled by a Markov model. In general we assume the security state of the system to be unobservable. In order to do online risk assessment a sensor model that describes the trustworthiness of the sensors is needed. This is the basic motivation for using HMMs to model the system. This idea is not new, since it has been used in process control applications for a long time. A Kalman Filter (KF) is a specific type of HMM having continuous state space and continuous or discrete time. In a KF the process and measurement noise are assumed to have a Gaussian distribution.

One classic use of a KF is as a component in a system for Dynamic Positioning (DP) of ships [Ven96]. The hidden states could e.g. be the

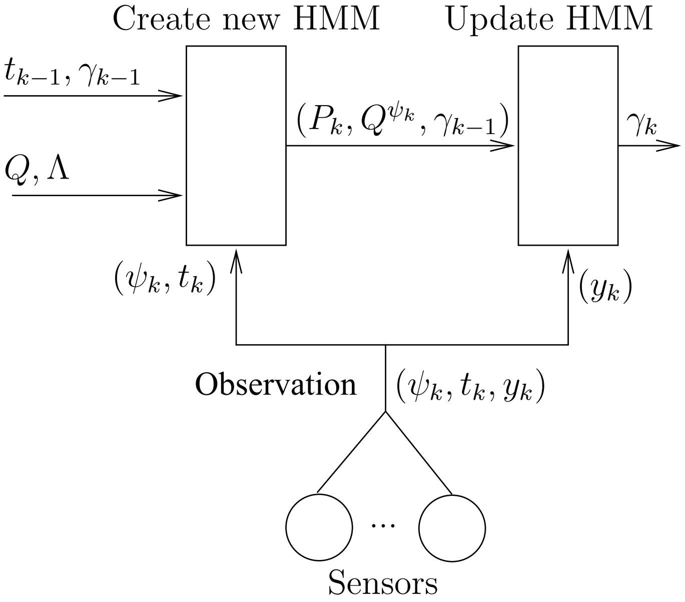


Figure 4. Observation processing

heading, position, speed, and acceleration of the ship. Sensors in this setting are typically Global Positioning System (GPS) receivers, wind sensors, and accelerometers. The KF models the ship's motion and the relation between the system state and measurements collected by the sensors.

The model is used to predict the next position. The predicted position is then corrected by the observation to form a minimum variance estimate of the position.

k observation index

t_k time when observation k is received

δ_k time between observation $k-1$ and observation k

γ_k state distribution after observation k is processed

$Q = \{Q^1, Q^2, \ldots\}$ observation probability matrices, describing the trustworthiness of the sensors

Λ transition rate matrix, describing the dynamics of the CTMM of the interaction between the system and the environment

P_k transition probability matrix used for processing observation k

ψ_k index of the sensor that has produced observation k

y_k observation received at time t_k.

In order to use Markov models for prediction and detection of attacks, we need to process observations received from 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, honeypots, etc. Time between alerts produced by sensors is not modeled, we are therefore able to model active monitoring as well as alerts being produced at random times. A specific consequence of this is that it can be produced zero, one or more observations while the system is in the same state.

As an alternative to the KF we now consider the discrete state space shown in Figure 3. Observations are not necessarily Gaussian distributed. They can in fact have an arbitrary distribution described by an observation probability matrix. Each sensor is modeled by an HMM, described by a parameter set indicating the trustworthiness of the sensor.

One common state distribution γ_k is maintained for all sensors, and it is updated each time an observation is received from one of the sensors. Figure 4 shows the two main steps in the observation processing:

1 Create HMM: First a new HMM is created, which is specific for the sensor and time since last observation was received δ_k. The HMM is represented by a parameter set consisting of two matrices and one vector. The first matrix is the transition probability matrix $P_k = e^{\Lambda\delta_k}$, estimated from the transition rate matrix Λ and time since last observation $\delta_t = t_{k-1} - t_k$. The second matrix is the observation probability matrix Q^{ψ_k}, that is selected based on the index ψ_k of the sensor that has produced the observation. Current state distribution γ_{k-1} is used as the initial distribution.

2 Update HMM: The newly created HMM is updated using a standard update algorithm, that e.g. can be found in [Rab90]. The update algorithm takes the model created in step 1 and the observation y_k as input. The output from this algorithm is the updated state distribution γ_k.

4.4 Assumption About Sensors

We have made the following basic assumptions about the sensors:

- Sensors are not perfect, false alerts could occur, e.g. as a consequence of underspecified signatures.
- The probability of producing different alerts depends on the security state of the assets that are monitored. In general we assume that more severe alerts are produced when the security state of the network is more severe.
- Observations are assumed to be independent, given current state. This requires some form of preprocessing and aggregation of alerts in order to

collect all observations coming from the same sensor, if they are based on the same underlying events. This is described in more details in Paper G

- Parameters are assumed to be stable during short time intervals. This can be tackled by using some form of drift detection as proposed in paper [KL07].

We assume filtering and correlation of alerts before they are used for risk assessment in order to remove alerts representing attacks that the system is protected against.

4.5 Multi Sensor Risk Assessment

Three different methods for aggregating risk from multiple sensors have been proposed in the papers in this thesis:

1 Averaging risk, introduced in Paper C.

2 Minimum variance estimator of the risk, introduced in Paper B.

3 One common state distribution with different observation matrixes for each sensor. This method is presented in Section 4.3. It was first introduced in Paper A, and later extended to a continuous time model in Paper G.

The averaging approach presented in Paper C, is based on using one HMM for each sensor which takes into account the trustworthiness of that sensor when the risk is estimated for it. When the risk from each sensor is aggregated, an averaging operation is used, weighting the risk from all sensors equally. This approach is easy to implement, and could be a good approach if the sensors have similar trustworthiness. In a situation where the estimated risk from one very trustworthy sensor and one that is not very trustworthy are aggregated, it would have been better to use only the risk from the most trustworthy sensor, and not the average. More details on this method can be found in Section 4 in Paper C.

The minimum variance approach is based on forming a minimum variance estimator of the risk by weighting the risk estimated from each sensor with the inverse of its variance. This approach takes into account the trustworthiness of each sensor and therefore is a better estimator of the risk than the average. A drawback with this approach is that it depends on the assumption that the risk estimated from each sensor is independent and unbiased. More details on this method can be found in Section 3 in Paper B.

The last approach presented in Paper A and later improved in Paper G, uses one common distribution for all sensors in monitoring the system. When an observation is received from one of the sensors monitoring the system, the distribution is updated using that sensor's observation probability matrix which represents the trustworthiness of that sensor. This approach have some advantages over the first two approaches: it takes into account the trustworthiness of individual sensors when information from different sensors are combined; it makes no assumption about independence between sensors except for assuming independent observations given current state, as explained in Section 3; and it is based on a Continuous Time Markov Model (CTMM) in order to tackle the fact that observations are produced at irregular time intervals.

4.6 Modeling Risk by Costs

Two different risk assessment methods are presented in this thesis. The first which is introduced in Paper A is based on cost associated with states, and the second method first introduced in Paper D is based on fuzzy logic and will be explained in Section 4.8.

Following the terminology in [Sta04], risk is measured in terms of consequences and likelihood. A consequence is the (qualitative or quantitative) outcome of an event and the likelihood is a description of the probability of the event. To perform dynamic risk assessment, we need a mapping: $\mathcal{C} : S \to \mathbb{R}$, describing the expected cost (due to loss of confidentiality, integrity and availability) for each object. The total risk $\mathcal{R}_t$ for an object at time t is

$$\mathcal{R}_t = \sum_{i=1}^{N} \mathcal{R}_t(i) = \sum_{i=1}^{N} \gamma_t(i)\mathcal{C}(i) \tag{1}$$

where $\gamma_t(i)$ is the probability that the object is in security state s_i at time t, and $\mathcal{C}(i)$ is the cost value associated with state s_i. For further details, see Paper A.

4.7 Intrusion Frequency

The idea behind intrusion frequency is to estimate mean time to absorption given the distribution γ_k estimated by the HMM. An absorbing state is a state if once entered is never left [Ros03]. We propose to use mean time to absorption as a security measure and call it Mean Time to Next Security Failure (MTNSF).

In order to do the MTNSF analysis we partition the system states into two subsets; the set of good states $S_G = \{N, IA, IP\}$, and the set of failed states $S_F = \{SA\}$. The MTNSF analysis is based on the embedded Markov chain, which is a Discrete Time Markov Chain (DTMC). We then modify the Markov model so that the failed state SA is made absorbing, and the good states are then transient.

The first step in the MTNSF analysis is to estimate the average number of times each of the transient states S_G is visited before the DTMC reaches one of the absorbing states S_F. This number is called w_i.

MTNSF can now be estimated by summing the product of w_i multiplied by the mean state occupation time h_i for each of the transient states S_G, giving $MTNSF = \sum_{i \in S_G} w_i h_i$. The intrusion frequency f_k can now be estimated by inverting MTNSF.

Based on the observation y_k, the distribution γ_k is estimated and then used to estimate f_k, which is the input to the online fuzzy risk assessment described in the next section.

4.8 Modeling Risk by Fuzzy Logic Controllers

The model presented in Section 4.6 is not taking into account specific details about the attacker. We have therefore developed a more comprehensive risk model that takes into account information abut specific attacks and assets. This model is mainly based on the Factorial Analysis of Information Risk (FAIR) model [Jon06], but we have included fuzzy logic in our model.

Traditionally, risk assessment has been done by human experts, because there is no exact and mathematical model which can be used to represent risk for a given ICT system. Our approach to risk assessment in ICT systems is based on an dynamically estimated intrusion frequency, and some additional static system parameters.

We propose an online risk assessment system that can continuously estimate the risk of a computer system or network, based on observations collected from sensors in the network. Sensors in this setting could e.g. be IDS or virus scanners. Observations are processed using our sensor model, and the intrusion frequency estimated each time a new observation is received. We have based our risk assessment on fuzzy logic that is used to represent knowledge collected from security experts. This knowledge is then represented as *if-then* rules in a fuzzy inference system.

The risk assessment is based on altogether nine linguistic variables which are processed using three Fuzzy Logic Controllers ($FLC_1 - FLC_3$). The three FLC's represent *Threat Level*, *Vulnerability* and *Asset Value*, which are three derived linguistic variables. The derived linguistic variables are then combined using FLC_4 to compute the net *Asset Risk*. This forms a hierarchical fuzzy system.

Of these nine linguistic variables, all except the intrusion frequency f_k obtained from the HMM model, are considered to be of a static nature during a short time period. Eight of the nine linguistic variables can be obtained from a lookup table, indexed by asset and attack category. We have used both a Mamdani- and a Takagi-Sugeno-Kang fuzzy inference system.

4.9 Fuzzy Risk Model Optimization Using Neural Learning

Two challenges when modeling the risk of assets with fuzzy logic controllers (a Fuzzy Inference System (FIS)) are the construction of if-then rules and the parameter estimation. We therefore proposed to use an integrated model combining fuzzy logic and neural learning.

On the other hand, in an integrated model, neural network learning algorithms are used to determine the parameters of fuzzy inference systems. Integrated neuro-fuzzy systems share data structures and knowledge representations. A fuzzy inference system can utilize human expertise by storing its essential components in a rule base and a database, and perform fuzzy reasoning to estimate the overall output.

The derivation of *if-then* rules and corresponding membership functions depends heavily on the *a priori* knowledge about the system under consideration. However, there is no systematic way to transform experiences or knowledge of human experts to the knowledge base of a fuzzy inference system.

There is also a need for adaptability or some learning algorithms to produce outputs within the required error rate. On the other hand, an Artificial Neural Network (ANN) learning mechanism does not rely on human expertise. Due to the homogenous structure of an ANN, it is hard to extract structured knowledge from either the weights or the configuration of the network. The weights of the neural network represent the coefficients of the hyper-plane that partition the input space into two regions with different output values. If we can visualize this hyper-plane structure from the training data, then the subsequent learning procedures in a neural network can be reduced. However, in reality, the a priori knowledge is usually obtained from human experts, and it is most appropriate to express the knowledge as a set of fuzzy if-then rules. However, as already mentioned it is considered a challenge to encode the knowledge into a neural network.

To a large extent, the drawbacks pertaining to these two approaches seem complementary. As a consequence, it seems natural to consider building an integrated system, combining the concepts of FIS and ANN modeling. A common way to apply a learning algorithm to a fuzzy system is to represent it in a special neural network like architecture. However, the conventional neural network learning algorithms (e.g. gradient descent) cannot be applied directly to such a system since the functions used in the inference process are usually non differentiable. This problem can be tackled by using differentiable functions in the inference system or by not using the standard neural learning algorithm. In our simulation, we used the Adaptive Network Based Fuzzy Inference System (ANFIS) [Jan93]. ANFIS implements a Takagi-Sugeno-Kang (TSK) fuzzy inference system [Sug85] in which the conclusion of a fuzzy rule is constituted by a weighted linear combination of crisp inputs rather than a fuzzy set.

4.10 Intrusion Detection Systems

In [DDW99] Debar et al. made a taxonomy for IDSs, defining the two main categories of IDS. We will refer to behavior-based as anomaly-based and knowledge-based as misuse or signature-based. Anomaly-based IDSs use information about the normal behavior of the system it monitors in order to search for unusual behavior while signature-based IDS search for evidence of attacks based on knowledge accumulated from known attacks. Signature-based IDSs have usually a lower false positive rates than anomaly-based IDSs, but they have some drawbacks like difficulties to gather all necessary information to detect an attack and to keep the database of signatures up to date. Anomaly-based IDSs can detect attempts to exploit new and unforeseen vulnerabilities, but this comes at a cost of a higher false alarm rate.

Debar et al. [DDW99] define two types of IDS: Host based IDS (HIDS) and Network based IDS (NIDS). An HIDS is a tool for analyzing of audit data available at the host in order to detect attacks, while a NIDS is a tool for sniffing and analyzing network traffic in order to detect attacks. There are several types of network attacks that are impossible or at least very difficult to detect by an HIDS, e.g. Domain Name System (DNS) spoofing, Transmission Control Protocol (TCP) hijacking, port scanning and Ping-Of-Death. These attacks are usually better detected using a NIDS. The type of IDS is in general not dependent on the category, so both types of Intrusion Detection System (IDS) (HIDS and NIDS) could be either anomaly-based or signature-based.

In a Distributed Intrusion Detection System (DIDS), a conventional IDS is embedded inside intelligent agents which are deployed over a network. IDS agents communicate with each other in order to detect planned and coordinated intrusion attempts.

An IDS will not do any attempts to prevent an intrusion, it will only detect it and raise an alarm, and possibly report it to the operator.

4.11 Online Risk Assessment and Intrusion Prevention

We propose an online risk assessment and Distributed Intrusion Prevention System (DIPS) based on our fuzzy risk model.

Definition 8 *An IPS is defined as the complete process of identifying and responding to attempts to compromise the security policy. A DIPS is an intrusion prevention system implemented in a distributed environment.*

We consider the IPS as an integrated IDS with many more functions. Due to the distributed nature of the IPS, the implementation poses several challenges. IDSs are embedded inside software mobile agents and placed in the network to be monitored. The individual IDS may be configured to detect a single attack, or they may detect several types of attacks.

Figure 5 illustrates the basic architecture of a DIPS element, which is controlled by a local controller. In a large network, each DIPS element communicates/coordinates with other DIPS local controllers and/or a central controller [AJTH07]. Online risk assessment for assets protected by the DIPS are performed by the HMM and ANFIS system which is a central part of each DIPS element. An IDS deployed are capable of detecting events spanning

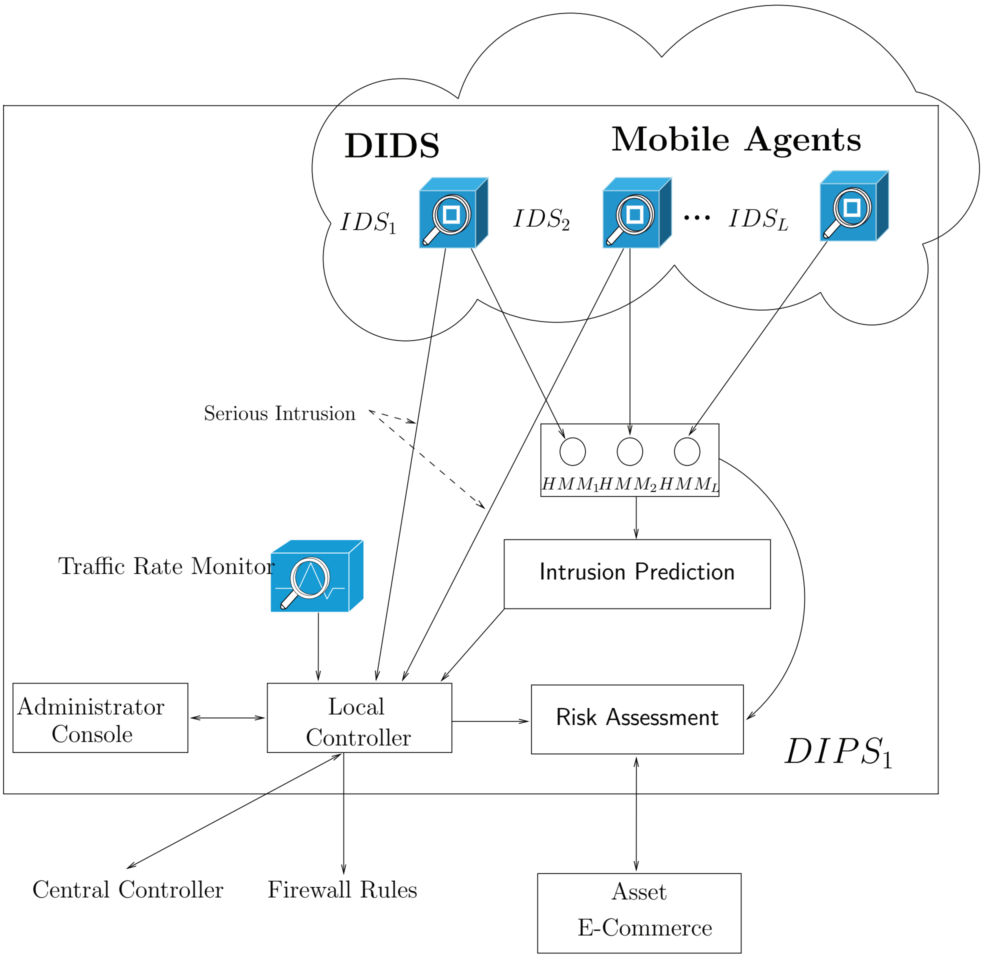


Figure 5. Architecture of a DIPS element.

from simple problems to serious denial-of-service type of attacks. The central controller takes the following actions based on the nature of the detected attack:

1 If the detected attack is simply a port scan or a probe, the HMM model will estimate the intrusion frequency of a possible future attack based on the current distributed attack pattern. Based on the estimated intrusion frequency, the central controller (or administrator) would take precautionary measures to prevent future attacks. The central controller would also make use of online assessment of the risk of the assets being subjected to possible serious attacks in the future.

2 If the detected attack is very serious, the central controller would take necessary actions to re-configure firewall rules or notify the administrator. Such serious attacks would bypass the HMM model.

3 If at any time any abnormal traffic rate is noted by the monitor, then again the central controller would take necessary actions to re-configure firewall rules or notify the administrator etc.

5. Related Work and Foundation

5.1 Security Assessment Based on Criteria

The security of computer systems has traditionally been assessed from a qualitative point of view, using a set of criteria. Assessment based on criteria focus on the presence and absence of certain functional characteristics and the use of specific development techniques. A well known set of qualitative evaluation criteria are the “Trusted Computer System Evaluation Criteria” [Def83] also known as the Orange Book published by US Department of Defense, and it’s interpretations. The Red Book [Cen90], entitled “Trusted Network Interpretation”, is divided into four divisions: A, B, C and D. Where (A) is the highest division reserved for systems providing the most comprehensive security. The Orange Book has subsequently been replaced by the Common Criteria (CC), also standardized by ISO/IEC [ISO05]. The CC Part 3 is organized in Evaluation Assurance Levels (EALs), where the numerical rating is describing the depth and rigor of an evaluation. EAL 1 is the most basic and EAL 7 is the highest level. These types of assessment represent a level of trust, or an un-quantified belief about security, but give no quantitative measures of security. A problem with these types of assessment is that there is no guarantee that a system evaluated to a higher trust level is truly more secure, despite the fact that the confidence in the correctness of the system relative to its design, construction and, implementation and documentation may be higher.

5.2 Stochastic Modeling of the Attack Process

In [LBF^{+}93] Littlewood et al. introduced the concept of operational measures of computer security, with the intention of starting a discussion on the relation between reliability and security. Many open questions that need to be answered before the quantitative approach could be taken further were identified. The idea of using mean effort spent by an attacker to a security breach as a security measure was launched, and the relation between time and effort was discussed. The authors argued that an assumption of breaches being accidental seems most amenable to the reliability approach. They also discuss the validity of using the exponential distribution as an approximation for distribution of the time to next security breach. In our research we have chosen to use time instead of effort, but we do assume that regular re-estimation of the transition rates will to some extent take into account the relation between time and effort.

Jonsson and Olovsson [JO97] conducted an intrusion experiment where students attacked a real system under controlled conditions. From their experiment they concluded that the attacking process can be split into three different phases: the learning phase, the standard attack phase and the innovative phase. The probability of a successful attack during the standard attack phase is much higher than in the learning and innovative phases. The collected data indicate that the time between breaches during the standard attack phase is exponentially distributed, and this implies that methods that otherwise are used for reliability actually could be applied for security threats as well. The experiment did not focus on the effect of a large pool of possible attackers; only a fixed number of attackers consisting of 12 groups of 2 attackers were considered. The experiment focuses mainly on introduction and removal of vulnerabilities caused by installation of new software and software upgrades. One observation was that skilled attackers after some time run out of vulnerabilities that are easy to exploit and enters the innovative phase. We assumed that in a real system this effect would be less noticeable since more complex systems are under constant change, often resulting in new vulnerabilities being introduced through the installation of new software, software updates, and reconfiguration of existing software and hardware.

Kaâniche et al. [KAN$^+$06] have performed an empirical analysis of data collected from thirty-five honeypots located in twenty-five countries. They found that the distribution of times between attacks is best described by a mixed combination of a Pareto distribution and an exponential distribution. This is an argument for using a semi-Markov model to model the intrusion process, where state occupation times can have arbitrary distributions, and a hidden semi-Markov model (HSMM) to model a sensor. We have chosen to not use a HSMM, in order to keep the presentation as clear as possible. The reason for this is the complexity of statistical inference in HSMMs. In future works the use of different probability distributions in a HSMM may be considered.

In [SKH05, SHK05] Sallhammar et al. introduce stochastic game theory as a tool for modeling an attacker's behavior, and in [SHK06] a method for integrating dependability and security is presented. In [SHK07] they use their model for online security and dependability assessment. They model the interaction between the system and the attacker as a game, assuming that the system is trying to minimize the loss and the attacker to maximize his/her gain. By solving the game, a complete attack strategy can be calculated.

5.3 Stochastic Models of Intrusion Tolerant Systems

In [GGPW$^+$01] Gong et al. present a state model representing the dynamic behavior of an intrusion tolerant system. Their model consists of nine states aimed at modeling the different actions an intrusion tolerant system can take against an attacker, and the attacker's response. They claim that their model covers both known and unknown vulnerabilities by focusing on impact rather than specific attack procedures. In the paper they do not present any security measures, but their approach on modeling the interaction between the attacker and the system have been an inspiration for our research. One important model feature that we have adopted in our research, is the modeling of attackers giving up attacks and the removal of known vulnerabilities from the system. Their system model still differs from ours in that they are modeling an intrusion tolerant system, specifically their model contain some extra states for this purpose. Another difference is that they do not take into account unsuccessful removal of vulnerabilities or unsuccessful reconfiguration that we believe to be quite important in a real system.

Madan et al. [MVT02, BBMGPT04] extend the state model presented by Gong et al. to a semi-Markov process, enabling them to compute some quantitative security measures like steady state availability and mean time to security failure. They do not model the relation between time and effort, which corresponds to the approach taken in our research. Still, when comparing their model to ours, some noticeable differences appear. The main difference is that they model a system while we model sensors comprising a system model. Furthermore they use a semi-Markov model to model the system, while we use a Markov model. They state that the use of a semi-Markov model have no implications on the analysis since they only do a steady state analysis and not a transient analysis. Their mean time to security failure is different from our mean time to next security failure in that they assume the system starts in the good state, but we use the distribution estimated by the HMM as the initial distribution for the analysis. They model availability, integrity and confidentiality, based on the steady state probabilities. This analysis do not depend on the initial or current state distribution that we estimate with the HMM, and this type of security measure is therefore considered to be of less importance in a real time risk assessment system. We note, however, that the situation may change with frequent re-estimation of model parameters.

In [WMT03] Wang et al. use Stochastic Reward Nets (SRN), which is an extension of the Generalized Stochastic Petri Net (GSPN), to model the dynamics of an intrusion tolerant system. This was developed in the scalable intrusion tolerant architecture (SITAR) research project. Their work is based on research done by Madan et al. and is built on the same basic model as used in [MVT02], but the new model is more specific for the SITAR system. The model takes into account the multiple vulnerabilities in the system, the unsuccessful removal of vulnerabilities and the state of three individual servers. These extra model features come at a price; there is the large increase in the number of states. In the resulting CTMM the number of states increases from 9 in [MVT02] to a few hundred. The approach is therefore not very attractive in our online estimation research, since the computation of our security measures may become too time- and memory-consuming. The difficulties of doing parameter estimation and parameter re-estimation will also increase. This type of model is therefore more attractive in offline settings.

Different variations of Stochastic Petri Nets have also been used, e.g. in [SCS03] where Singh et al. use stochastic activity networks to model the Intrusion Tolerant by Unpredictable Adoption (ITUA) architecture.

5.4 Attack and Privilege Graphs

Another interesting approach to quantifying security is [OD99], where Ortalo et al. propose to model a computer system as a privilege graph which exhibits security vulnerabilities. In this graph each node represents a set of privileges owned by the user, while edges represent vulnerabilities. In [OD99] a tool called Automatic Security Adviser exists that looks for known vulnerabilities in the system and builds the privilege graph. A Markov model is constructed from the privilege graph, where each edge is assigned a rate corresponding to the amount of effort that an attacker has to spend in order to exploit the corresponding vulnerability. They propose a quantitative measure called Mean Effort To Security Failure (METF), representing mean effort until one of the privileges that must be protected according to the security policy is obtained by an attacker. They propose different versions of METF distinguished by the attackers memory of visited states: Total memory assumption, and the memory-less assumption. There are some important differences between their model and our model. Their model will only take into account known vulnerabilities, but we are more focused on the unknown vulnerabilities. In our model we assume that when a vulnerability is detected, some effort will be made to remove it. They also do not model the fact that an attacker may give up an attack which we consider an important property of our model.

In their paper the security measure Mean Time To Failure is not taking into account the security state of the system. Our security measure do this, by using the output from the HMM (the estimated distribution) as the initial distribution for the Mean Time To Security Failure estimate.

Sheyner et al. propose to use automated generation and analysis of attack graphs [SHJ^{+}02] as a part of a probabilistic reliability analysis. They model the network as a finite state machine, where atomic attacks correspond to transitions. They then use a model-checker to construct and analyze an attack graph, and estimate likelihood of intrusion. This takes advantage of the efficient model representation that exists in the field of model checking, increasing the size of a system that can be analyzed. Their modeling approach is quite similar to the privilege graph approach. Sheyner et al. claim that the privilege graph approach is more attack centric than the approach used in their paper. They find the minimal set of atomic attacks that must be prevented in order to guarantee that the intruder can not reach his goal. By assigning probabilities for certain events to the graph, and interpreting the graph as a Markov Decision Process, they are able to estimate the effect of adding new security mechanisms in terms of the probability of detecting the intrusion. The model presented by Sheyner et al. is quite different from our model since they do not take time or effort into account, resulting in a static model. To some extent, their model takes into account the fact that sensors in the network are not perfect and some attacks are not detected. That is similar to what we try to model.

5.5 Aggregation and Correlation of IDS Alerts

There has been a research focus lately on different types of aggregation and correlation of IDS alerts. The main objectives of this research have been to reduce the amount of alerts seen by the operator, group together alerts arising from the same event, and the removal of false alerts. The output of this type of algorithms are usually new alerts (e.g. referred to as meta-alerts), which contain a high level description of the attack. Some interesting approaches are:

In [DW01], Debar and Wespi propose a system that performs both aggregation and correlation of IDS alerts from many different sensors. One important feature of their system is its ability to aggregate alerts into so called situations according to the source and target. They also propose a method for detecting unauthorized scans.

Data mining and clustering [Jul01] is another approach that is used for extracting the true alerts in a large data set of IDS output by clustering alerts according to their root cause. False positives can be reduced considerably, but at the cost of additional increase in false negatives for infrequent alerts.

A comprehensive approach for intrusion detection alert correlation is presented by Valeur et al. in [VVKK04]. One notable feature of the proposed system is its ability to do impact analysis of alerts in order to prioritize alerts according to their severity, based on the security policies and security requirements of the sites.

Perdisci et al. propose a strategy for alarm clustering [PGR06]. They use a learning phase to extract the attack class(es) an attack description belongs to. Their clustering method is based on a distance measure expressing the distance between alarms and meta alarms. As before, the meta-alarms produced are a high level description of the attacks.

We have considered these techniques a necessary part of a risk or intrusion prevention system based on IDS sensors. In Paper G we have described how alerts are transformed into observations by applying some correlation and aggregation techniques. This is done to avoid dependencies between observations, and to have all alerts in some standardized format. We have proposed that the alert severity ratings, together with the timestamps, are used as observations in the HMM. In [PGR06], the approach differs from our approach in that they do not model the dynamic aspects of the interaction between the attacker and the system, neither do they produce any quantitative security measures, based on the security state or situation of the asset.

5.6 Risk Assessment

Risk assessment has traditionally been a manual analysis process based on a standardized framework, such as those recommended by National Institute of Standards and Technology (NIST) [SGF02] and AS/NZS [Sta04]. The main drawback with this type of methods is costs associated with hiring experts, and the fact that it takes some time to do the actual assessment. During this time an attack could already have occurred. Therefore there is a need for some online risk assessment methods making a fast response possible.

In [Jon06] Jones describes a risk assessment framework called FAIR. FAIR is an offline risk assessment method, and has been an inspiration for our fuzzy logic risk assessment method. Our method is a simplified and modified version of FAIR in terms of risk factors involved in the assessment; this simplification makes it more suitable for online risk assessment. The FAIR model consists of four basic components: threats, assets, the organization itself, and the external environment. In comparison, our model takes into consideration threat level, vulnerability, and asset value. As has FAIR, we have adopted a hierarchical structure. However, the way individual risk factors are combined is quite different. They use a matrix to map combination of input variables to output variables, while we use fuzzy logic controllers which, although arranged in a similar matrix structure, are much more flexible since we can use different membership functions for the input and output variables. The principal difference is that we use fuzzy sets while they use crisp sets for the input and output variables. From our point of view this type of risk models seems suitable for an implementation using fuzzy logic, since human expert knowledge and opinions about risk can be embedded in the model.

In [GK04], Gehain et al. present a method for intrusion prevention called RheoStat. This is a formal model to characterize the risk faced by a host, and describe how risk can be managed in real-time by adapting the host's exposure to perceived threats. The RheoStat method is implemented by modifying the access control system in the java runtime environment, in order to tighten the access to resources in response to threats detected by an integrated intrusion detection system running on the host. The decision of whether to tighten or loosen the access to resources is based on a cost-benefit analysis, where costs are measured by the frequency of which permissions are checked. After each event reported by the IDS, the system tries to find the most cost efficient subset of permissions that will bring the risk under some predefined limit. This problem is equivalent to the NP-complete Knapsack Problem. The risk model consists of three basic risk factors: threat, vulnerabilities, and assets. Threats are associated with IDS signatures, i.e. the likelihood of the threat are derived based on how completely the signature is matched. Assets are items with value, and the consequence is the harm that an asset may suffer in terms of confidentiality, integrity and/or availability loss. The NIST National Vulnerability Database (NVD) [NCSD09] is used to get information about vulnerabilities. The actual risk estimation is based on combining these factors using products and sums. This risk model is in some sense similar to our model, but with some important differences. We model and measure the activity of the attacker, in terms of intrusion frequency. We also assume that it is important to consider time when estimating risk; our risk model will fade out the risk over time if nothing is happening. We have also developed a sensor model, modeling the trustworthiness of different sensors so we can handle input from more than one sensor. This factor seems to be important in a real network. These considerations are valid for both of our risk models, the ones presented in Paper A to Paper D that are based on cost associated with states, and for the models presented in Paper E to Paper G using parameters for different attack/categories stored in a look-up table. We have not done any research on specific methods for reaction against attacks.

5.7 HMM and IDS

A relevant introduction to HMM can be found in [Rab90]. We have made some modifications to the algorithms presented in that paper, in order to use a continuous time model instead of a discrete time model. We also introduce the idea of using more than one observation matrix in order to model more than one sensor.

Using system call data sets generated by different programs, Warrender et al. compared different data modeling method's ability to model normal behavior accurately with the intent to use anomaly detection to recognize intrusions [WFP99]. They proposed an HMM based IDS, and compared it to a model using techniques described as Sequence Time-Delay Embedding (STDE), and Repeated Incremental Pruning to Produce Error Reduction (RIPPER). Registered user behavior for different UNIX programs (nemd, ps, sendmail,...) has been used as basis for the modeling of normal behavior. For the HMM they have suggested using roughly the same number of states as unique system calls used by the program, ranging from 40 to 60. For each system call, an anomaly decision is taken, based on what state transitions and output would have been required of the HMM to produce that output. If the system call only could have been produced using unlikely transitions and outputs, it is marked as a mismatch. They concluded that the system call data was regular enough for simpler modeling methods than HMM to work well.

Wang et al. [WGZ04] repeated some of the experiments done by Warrender et al. using traces of system calls as observables for an HMM instead of individual system calls, and reported a better detection accuracy. They argue that this method is suitable for online detection, since it is only the training phase that is computationally expensive.

HMMs have been used to detect multi stage attacks [OMSH03]. Ourston et al. model each stage of an attack as a state in the HMM, and use transitions to represent atomic attacks. They implemented their system using a separate HMM for each attack category. The HMM model has one state for each attack step (ip scan, port probe). They collect alerts from multiple sensors in the network and sort them according to their connections (source, destination). All alerts in one connection are subsequently analyzed by each HMM in order to check if there is a match with one of the attack categories.

In [GSW03] GAO et al. use an HMM to forecast attacks in an IDS, mainly operating on the application layer. They also use fuzzy logic in the decision making. In [KL06], Khanna et al. propose to build an IDS based on an HMM with multivariate Gaussian distributed observations, and dynamical re-estimation of parameters. In [KL07], Khanna et al. use distributed HMM processing in combination with a Proportional Integral differential (PID) control engine to make a distributed IDS for ad hoc networks. In essence, their approach is different from ours as they use continuous observations and are only modeling one sensor at a time.

5.8 DIDS

The detection of certain attacks against computer networks requires information from multiple sources. In response to that fact, several different frameworks for DIDS have been proposed. The concept of a DIDS was first proposed by Snapp et al. [SBD+91] in 1991. A major problem considered in that paper is the Network-user Identification problem. That is the problem of tracking users moving across the network, possibly with a new user-id on each computer. They designed and implemented a prototype DIDS that combined distributed monitoring and data reduction through individual host and Local Area Network (LAN) monitors with centralized data analysis through the DIDS director. This was used to monitor a heterogeneous network of computers. Another concept which is discussed in the paper is the security state of the network represented by a number between 1 and 100. This number represents all threats against all subjects in the network.

The use of a centralized data analyzer represents a single point of failure. To make the system more robust, Crosbie and Spafford [CS96] propose a DIDS architecture based on Autonomous Software Agents. This architecture was further developed in the Autonomous Agents for Intrusion Detection (AAFID) project. In this project a prototype of a DIDS has been developed based on Autonomous Software Agents, and is claimed to be more robust because of the following:

- there is no single point of failure
- agents are easier to reconfigure since individual agents can be added or reconfigured without restarting
- if the agent is running on the host it is monitoring, the chance of insertion and evasion attacks are reduced, DIDS [BGFI+98].

Their design consists of three essential components: agents, transceivers, monitors. An agent is an independently running entity which monitors specific aspects of a host; interesting events are reported to a transceiver. Each host has one transceiver which is responsible for control and communication with agents running on the host, for processing and aggregation of information, and for communication with monitors. Monitors are the highest level of entities in the AAFID architecture. They can control entities on other hosts. Monitors are responsible for high level correlation of data and have an API which can be accessed via a user interface. Agents are mobile and allowed to move from host to host.

Another DIDS is the GrIDS project at UC Davis [SCCC$^+$96]. They have modules running on each host to send information to an engine which builds a graph representation of the activity in the network. The graphical representation is used to report possible intrusions. GrIDS is designed to scale in large networks, by building graphs of activity in a hierarchical structure organized in hosts and several levels of departments.

Helmer et al. propose to use lightweight mobile agents for intrusion detection [HWH$^+$03]. In this system, software agents can move around in the network and be upgraded and updated. The main focus of this system is the use of learning algorithms, data warehousing, and mobile agents. A prototype of this system has been developed in Java.

Another DIDS architecture is the Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD) project [PN97] which is based on a recursive structure allowing generic components to be deployed in a distributed manner.

Our work is more focused on modeling of the system and estimation of risk, while other DIDS frameworks presented in this section are more focused on how to collect and aggregate information from agents in order to detect specific attacks. Our DIPS architecture proposed in Paper D is based on extending a DIDS architecture to also include intrusion prevention based on risk assessment. Intrusion prevention mechanisms add functionality to an IDS. This is functionality like: stopping suspicious network traffic, shutting down servers and processes that have a high risk of being under attack, and blocking of users that behave suspiciously.

The State Transition Analysis Technique (STAT) framework of [VVK03, VKB01] is a framework for administration and configuration of sensors in a DIDS based on five key concepts:

- STAT Technique for representing high-level descriptions of computer attacks,
- STATL Language which is an extendable language used to represent STAT attack scenarios,
- STAT Core representing the runtime of the STATL language,

- CommSTAT used by sensors to communicate in a secure way, following the Intrusion Detection Message Exchange Format (IDMF) [DCF07] and
- MetaSTAT which use CommSTAT to communicate with and control a set of sensors.

The STAT framework also includes an alert correlating component [VVKK04] and the infrastructure MetaSTAT which manages alerts and enables dynamic reconfiguration of sensors. In [ÅVVK06] Årnes et al. integrate our HMM model of risk into the STAT framework as a tool for prioritizing alerts.

6. Research Methodology

The initial idea of using HMMs in risk assessment was developed during the summer of 2005, as a result of cooperation between several researchers working in the security group at NTNU/Q2S. This work was published later that year, see Paper A. The idea of using Markov models to model the security state of a computer network is inspired by the stochastic methods used in the analysis of dependable systems. All papers in this thesis have multiple authors and are the result of joint work and discussions.

Given the fundamental nature of the research questions stated in Section 2.1, we believed the best method for tackling these questions was to start with a theoretical approach in order to build a theoretical foundation for future research. The work presented in this thesis is based on a broad literature study, discussions and cooperation between the authors in the development of theoretical models. The research is aimed at developing fundamental concepts and models to form a framework for risk assessment in operational computer systems and networks. During the literature study, different types of stochastic models have been investigated with a main focus on models used in dependability research. The emphasis in our research has been on the development of a sensor model that comprises a system model in addition to specific properties of individual sensors. This modeling technique enable us to model the relation between the system state and observations made by the sensors.

7. Validation of Work

The work presented in this thesis has mainly been validated by means of mathematical analysis in order to check that the model behaves as expected. The results have been presented at international conferences.

Simulation has been used to validate parts of the theoretical work. A simulation experiment is presented in Paper C, where a discrete-event simulation is implemented using JSIM [Mil]. JSIM is a discrete-event simulator framework for Java. The simulated network consists of an Internet gateway (router), two publicly available web-servers in a Demilitarized Zone (DMZ), two protected file-servers, as well as ten workstations and ten laptops. All assets with, the exception of the router, are monitored by a NIDS and a HIDS. State transitions and observations were generated using an HMM. This enables us to compare the true risk, with the estimated risk. True risk referees to the risk estimated from the true security state, estimated risk means risk estimated from the state distributions estimated by the HMM. Results shows that there is a high correlation between the estimated risk and the true risk, even if the HMM parameters are inaccurate.

In [ÅVVK06] Årnes et al. present an experiment where our risk assessment method have been integrated into the STAT framework [VKB01], and they have performed tests using training data from the Lincoln Laboratory [Lab00] and real network traffic from the Technical University of Vienna [KKM$^+$05]. The Lincoln Laboratory dataset consists of 1016 IP addresses each modeled by a separate HMM. The same transition and observation probability matrix where used for all HMMs except for two nodes in the network. The reason for this is that all hosts in the network are protected by NIDSs in addition the two nodes are protected by HIDSs, reflected in the parameter set. Alerts are fused and aggregated by MetaStat before they are fed into the risk assessment system. The risk assessment method used in this paper is similar to the method we have presented in Paper B. The entire data set has a time span of 11386 sec and consists of 36635 alerts and a truth file with information about security events in the network. In [VVK06] it is clearly demonstrated that the risk assessment method was able to assess the risk and detect several of the security relevant events in the data set. The system was able to assign maximum risk to the two most critical incidents. This experiment shows that our risk assessment method can be suitable for deployment in relatively large networks.

Implementation and testing of the risk assessment method in an operational computer network would of course have been preferable, but there are some problems with this type of validation. As already mentioned Årnes et al. tested the risk assessment method on network traffic data from the Technical University of Vienna. The result was in some sense inconclusive, because there were no known attacks in the test data. Also since real traffic data was used, there where no possibility to check the assessed risk against the truth state of the network. Another problem with performing tests in an operational network is the reluctance many system administrators have against deploying new experimental software that could have security and privacy implications in their systems. We believe that the first step in validating this type of risk assessment methods should remain theoretic, albeit followed up by simulations and experiments later, and this corresponds to what we have done so far. For progressing further, we recommend experiments to be performed in a closed and tightly controlled environment similar to what is presented in [Jon06]. Only after one is convinced by these types of experiments that the IDS/IPS is successful, a full implementation in an operational network which would be the final step in validation of the method is envisaged.

8. Summary of Papers

This section describes the contribution of each paper and the relation between the papers included in this thesis. A short description of my personal contribution to each paper is also included. Figure 6 shows a graphical representation of the relation between the papers. The indexing of papers is chronological after when they have been written. Paper A-C and G gradually refine the HMM modeling approach to online risk assessment and are closely related. Paper D-F are also closely related and presents an IPS architecture based on fuzzy logic. Paper G shows how the intrusion frequency needed as input to the risk assessment method presented in Paper D-F can be calculated, from the HMM model proposed in Paper A-C.

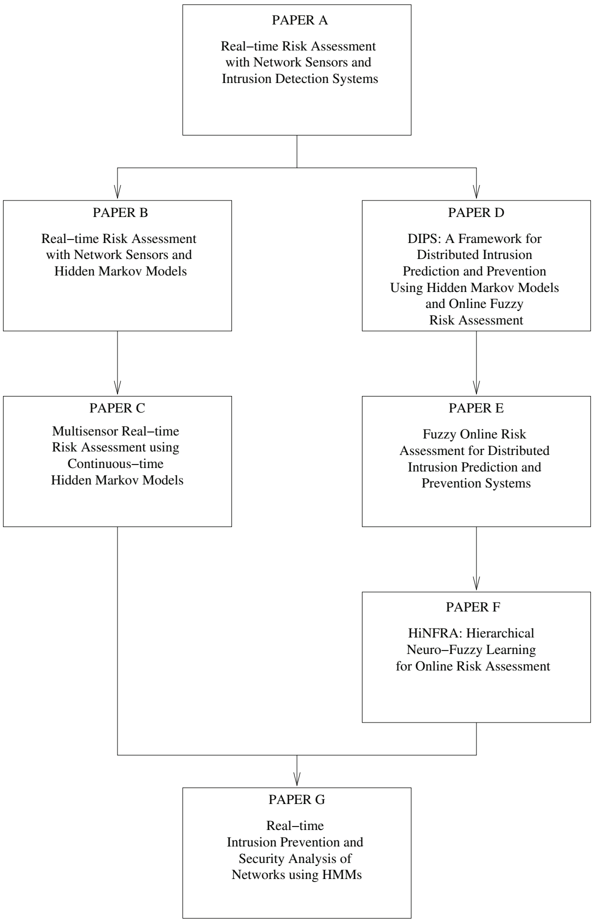

PAPER A
Real−time Risk Assessment with Network Sensors and Intrusion Detection Systems
PAPER B
Real−time Risk Assessment with Network Sensors and Hidden Markov Models
PAPER D
DIPS: A Framework for Distributed Intrusion Prediction and Prevention Using Hidden Markov Models and Online Fuzzy Risk Assessment
PAPER C
Multisensor Real−time Risk Assessment using Continuous−time Hidden Markov Models
PAPER E
Fuzzy Online Risk Assessment for Distributed Intrusion Prediction and Prevention Systems
PAPER F
HiNFRA: Hierarchical Neuro−Fuzzy Learning for Online Risk Assessment
PAPER G
Real−time Intrusion Prevention and Security Analysis of Networks using HMMs

Figure 6. The relationship between papers included in Part II

8.1 Paper A

Real-time Risk Assessment with Network Sensors and Intrusion Detection Systems.

This paper considers a real-time risk assessment method for information systems and networks based on observations from networks sensors such as intrusion detection systems. In this paper, we introduce the idea of using HMM for real-time assessment. We assume that the security state of a computer system can be modeled using a first order Markov model, and that the trustworthiness of sensors can be modeled by an HMM. The system risk is dynamically evaluated, providing a mechanism for handling data from sensors with different trustworthiness in terms of false positives and negatives. The risk is estimated by associating costs with states and using the state distribution estimated by the HMM. In this paper, we use an HMM to aggregate observations from all the sensors participating in the risk assessment. The paper contains an example with two sensors, showing parameters and observations combined with the estimated risk.

My contribution to paper A is the idea of using an HMM model to model the security state of a network. I was also the main contributor to the development of the model used in the paper, and in the implementation of the simulations.

8.2 Paper B

Real-time Risk Assessment with Network Sensors and Hidden Markov Models

This paper extends Paper A and shows simulation results. Specifically, it describes a method for handling data from multiple, heterogeneous sensors with different levels of trustworthiness. For each particular sensor, the risk is estimated based only on observations from that particular sensor. The total risk for the system is then estimated by using the mean of the risk from each sensor.

A discrete-event simulator is implemented, demonstrating the real-time risk assessment based on observations from intrusion detection systems. Using this simulator, we model and simulate the risk assessment of a large network and compare the results to the true values in order to validate the proposed approach.

My contribution to paper B was that I had main responsibility for researching and writing the risk assessment section.

8.3 Paper C

Multisensor Real-time Risk Assessment using Continuous-time Hidden Markov Models

In this paper we further extend ideas presented in Paper A and Paper B to facilitate more flexible risk modeling with support for a wide range of sensors.

For each sensor we maintain a state distribution estimated by an HMM, this distribution was then used to estimate the risk. The overall risk is estimated by aggregating the risk estimates from each sensor using the inverse variance as weights. This forms a minimum variance estimator of the system risk. This method is an improvement to the aggregation method presented in Paper B.

We use a continuous time Markov model to model the evolution of the system state. When an observation is received from a sensor, a discrete Markov model is created from the continuous time model based on time since last observation. By using this method we take into account the fact that most sensors produce observations at irregular time intervals, e.g. an IDS.

My contribution to paper C is to be found in section 2, 3 and 4 where I was the main contributor to the development of the continuous time risk model, and the multi sensor model. I was also responsible for the examples.

8.4 Paper D

DIPS: A Framework for Distributed Intrusion Prediction and Prevention Using Hidden Markov Models and Online Fuzzy Risk Assessment

In this paper, a DIPS is proposed. The system consists of several IPSs over a large network, all of which communicate with each other or with a central server to facilitate advanced network monitoring. A Hidden Markov Model, as presented in Paper A, is used for sensing intrusions in a distributed environment and to make a one step ahead prediction against possible serious intrusions. The DIPS is activated based on the predicted threat level and risk assessment of the protected assets. Intrusion attempts are blocked based on (1) a serious attack that has already occurred (2) the rate of packet flows (3) prediction of possible serious intrusions and (4) online risk assessment of the assets possibly available to the intruder. The paper focus on three areas: (1) the distributed monitoring of intrusion attempts, (2) the one step ahead prediction of such attempts and (3) online risk assessment using fuzzy inference systems.

The paper is based on the discrete HMM model presented in Paper A and not the continuous HMM model presented in Paper C. The reason for this is that we want the presentation to be clearer and not unnecessary complicated.

Professor Ajith Abraham first came up with the idea of using hierarchical fuzzy logic for risk modeling and the use of risk as input to an IPS. I was made responsible for the experiments and the writing of Paper D with some contribution from Professor Abraham in the parts about fuzzy logic and distributed intrusion prevention systems.

8.5 Paper E

Fuzzy Online Risk Assessment for Distributed Intrusion Prediction and Prevention Systems

The novelty of this paper compared to Paper D is the detailed development of Fuzzy Logic Controllers to estimate the various risk(s) that are dependent on several other variables based on the inputs from HMM modules and the DIDS agents. To develop the fuzzy risk expert system, *if-then* fuzzy rules were formulated based on interviews with colleagues. Preliminary results indicate that such a system is very practical for protecting assets which are prone to attacks or misuse, i.e. highly at risk.

I had the overall responsibility for writing the paper and for the experiments. Professor Abraham contributed by writing the introduction and supported me during the experiments.

8.6 Paper F

HiNFRA: Hierarchical Neuro-Fuzzy Learning for Online Risk Assessment

Paper D and Paper E illustrate the design of fuzzy logic based online risk assessment for DIPS. Paper F on the other hand proposes a Hierarchical Neuro-Fuzzy online Risk Assessment (HiNFRA) model to aid the decision making process of a DIPS. The fine tuning of fuzzy logic based risk assessment model is achieved using a neural network learning technique. Preliminary results indicated that the neural learning technique could improve the fuzzy controller performance and make the risk assessment model more robust.

I was responsible for the overall writing of paper F and for the experiments, while Professor Abraham offered some input and advise on using Adaptive Network Based Fuzzy Inference System. He was also responsible for writing section 2.2.

8.7 Paper G

Real-time Intrusion Prevention and Security Analysis of Networks using HMMs

In Paper G, we further develop the idea of using a continuous time Markov model to model a sensor that was first introduced in Paper C. However, Paper G does not store one state distribution for each sensor as proposed in Paper C. Rather a new HMM is created and used for updating the estimated system state for each observation, based on the sensor trustworthiness, the continuous time Markov model and the time since last observation processed. Observations from different sensors are aggregated in the HMM and two different security metrics are estimated. Our objective in this paper was to calculate and maintain a state probability distribution that can be used for intrusion prediction and prevention needs. We showed how our sensor model can be used in the IPS architecture based on IDS sensors as presented in Paper D-F. Our approach is illustrated by a case study presenting a small computer network, where we construct an attack scenario and show how the security analysis is done.

My main contribution to Paper G was in the development of the mathematical model used in the paper. I was also responsible for writing sections 3.2, 3.3 and 3.4.

8.8 Contributions of the papers

The papers included in this thesis have contributed with the following:

- A model describing the different security states of a network and the transitions between them. This model is based on probabilistic methods used in dependability modeling (Markov models) that have been extended to model security.
- Extension of the system model to a sensor model based on an HMM. The sensor model models the trustworthiness of sensors for IDS and IPS.
- Introduction of a security measure called MTNSF and its inverse called the intrusion frequency.
- Two different risk models: One based on costs associated with states and one combining a hierarchical fuzzy inference system with the probabilistic system model to model risk.
- Two different methods for aggregation of information from multiple sensors. The first method combines the risk estimate from each sensor to produce a minimum variance estimator of the risk, where the variance represents the trustworthiness of the sensor. In the second method, one common distribution is maintained, and when an observation is received, the risk estimate is updated taking into account the sensors trustworthiness.
- The use of neural network learning to estimate parameters for the fuzzy logic based risk assessment model.
- A DIPS architecture as an application of our online risk assessment model.

8.9 My Contributions to the Papers Included in this Thesis

All papers in this thesis are authored by multiple authors. I will therefore in more detail explain my specific contribution to the papers included in this thesis. Svein J. Knapskog has been my supervisor and has therefore been a co-author of all papers except Paper B. He has been active in reviewing the text and has given important feedback on how to improve the text. Ajith

Abraham have introduced me to the idea of using Fuzzy logic in combination with an HMM to model risk. We have had some fruitful discussions about the modeling approach and he functioned as a co-supervisor on Paper D, Paper E and Paper F. Specifically my contributions to the paper in this thesis are:

- The initial idea of using an HMM to model the fact that the true security state of a network is unobservable or hidden.
- Forming the mathematical basis for Paper A, which is refined and extended in the papers following Paper A.
- Responsible for the simulations in all papers except Paper C.
- The idea of using a minimum variance estimation of risk, presented in Paper C. André Årnes was mostly involved in the writing of the Paper C.
- Introducing the idea of using one new observation probability matrix for each observation, in order to take into account the time between observations and the trustworthiness of different sensors.
- The idea of using the security measures MTNSF analogous to the MTFF frequently used as a dependability measure.
- Writing the sections about the system and sensor model in all 6 papers.
- The drawing of figures and plotting of graphs in all papers except Paper C.
- The writing of Paper E and Paper F, with some support from Ajith Abraham.

I have received adequate support in the writing the papers as my skills are more towards understanding and using mathematics and physics, than towards the writing of running text. It has been more natural for me to concentrate on to develop models, perform simulations, and illustrate proposed solutions by drawing figures rather that using comprehensive textual descriptions and explanations of the performed research.

9. Discussion and Conclusion

This thesis has focused on problems related to quantitative characterization of risk and security in computer systems or networks and dynamic risk and security assessment based on network monitoring. The result of the research is the development and design of a Markov model describing the security state of a computer network. This model is used to obtain quantitative security measures like the intrusion frequency and system risk. The system model is extended to include estimation of the trustworthiness of sensors collecting security relevant information in computer networks. An online risk assessment method based on the sensor model is proposed and used as a basis for an IPS architecture.

The work has been validated through mathematical analysis, simulations and experiments. The theoretical results presented in this thesis indicate that our risk assessment method could be useful in predicting and preventing attacks against computer networks. Simulations have been used to validate parts of the theoretical work. A simulation experiment is presented in Paper C, where a discreet-event simulation is implemented using the JSIM [Mil], a discrete-event simulator framework for Java. In [ÅVVK06] Årnes et al. present an experiment where our risk assessment method have been integrated into the STAT framework [VKB01], and they have performed tests using training data from Lincoln Laboratory [Lab00] and real network traffic from the Technical University of Vienna [KKM$^+$05].

The main challenges in implementing and testing the proposed risk assessment method have been parameter estimation and verification of the HMM. One specific characteristic of the parameter estimation process is that it is impossible to directly observe the security state of a computer network, e.g. the most dangerous attacks are those that never are detected. Another challenge concerning parameter estimation is that it is difficult to get security relevant incident data from real systems, because companies and organizations are reluctant to reveal any security and possibly privacy related information. A possible solution may be to use honeypots to collect data for parameter estimation, but this solution is not ideal, since a honey-pot is not a real production system.

The number of states in the models we have investigated are quite small. This is a conscious choice, mainly to reduce the number of parameters and to increase the chance of realistic validation. New and more expressive models which also take into account availability may be a topic for future research. Another research topic could be to extend the system model from a Markov model where state occupation times have to be exponentially distributed, to a semi-Markov model where the state occupation times can have a general distribution.

There are some important problems related to IPS that need focus: self denial of service, false positives, and the fact that a DIPS is by itself an attractive target. These problems have not been investigated in this research and may also be a topic for future investigations.

Bibliography

[AGMV07] A. Abraham, C. Grosan, and C. Martin-Vide. Evolutionary design of intrusion detection programs. *International Journal of Network Security*, 4(3):328–339, 2007.

[AJTH07] A. Abraham, R. Jain, J. Thomas, and S.Y. Han. D-scids: Distributed soft computing intrusion detection systems. *Journal of Network and Computer Applications, Elsevier Science*, 30(1):81–98, 2007.

[ALRL04] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 01(1):11–33, 2004.

[ANK$^+$06] Eric Alata, V Nicomette, M Kaâniche, Marc Dacier, and M Herrb. Lessons learned from the deployment of a high-interaction honeypot. In *EDCC'06, 6th European Dependable Computing Conference, October 18-20, 2006, Coimbra, Portugal*, Oct 2006.

[ÅSH$^+$05] André Årnes, Karin Sallhammar, Kjetil Haslum, Tønnes Brekne, Marie Elisabeth Gaup Moe, and 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.

[ÅSHK06] André Årnes, Karin Sallhammar, Kjetil Haslum, and Svein Johan Knapskog. Real-time risk assessment with network sensors and hidden markov model. In *Proceedings of the 11th Nordic Workshop on Secure IT-systems (NORDSEC 2006)*, Oct 2006.

[ÅVVK06] André Årnes, Fredrik Valeur, Giovanni Vigna, and Richard A. Kemmerer. Using hidden markov models to evaluate the risk of intrusions. In *Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection, RAID 2006, Hamburg, Germany, September 20 – 22, 2006.*, September 2006.

[BBMGPT04] 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.

[BGFI$^+$98] J. S. Balasubramaniyan, J. O. Garcia-Fernandez, D. Isacoff, E. Spafford, and D. Zamboni. An architecture for intrusion detection using autonomous agents. In *Proceedings of the 14th Annual Computer Security Applications Conference*, page 13. IEEE Computer Society, 1998.

[Cen90] National Computer Security Center. *Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria.* NSA/NCSC Rainbow Series. 1990.

[CHSP00] Curtis A. Carver Jr., John M.D. Hill, John R. Surdu, and Udo W. Pooch. A methodology for using intelligent agents to provide automated intrusion response. In *Proceedings of the IEEE Workshop on Information Assurance and Security*, 2000.

[CS96] Mark Crosbie and Gene Spafford. Defending a computer system using autonomous agents. In *In Proceedings of the 18th National Information Systems Security Conference*, 1996.

[DCF05] H. Debar, D. Curry, and B. Feinstein. Intrusion detection message exchange format (IDMEF) – Internet-Draft, 2005.

[DCF07] H. Debar, D. Curry, and B. Feinstein. The intrusion detection message exchange format (IDMEF), 2007. IETF RFC 4765.

[DDW99] Hervé Debar, Marc Dacier, and Andreas Wespi. Towards a taxonomy of intrusion-detection systems. *Computer Networks*, 31(8):805–822, 1999.

[Def83] Department Of Defense. *CSC-STD-001-83: Department of Defense Trusted Computer System Evaluation Criteria [DOD 5200.28].* Defense Department Rainbow Series. 1983.

[DTBCC06] H. Debar, Y. Thomas, N. Boulahia-Cuppens, and F. Cuppens. Using contextual security policies for threat response. In *DIMVA '06: Proceedings of the 3th International Conference on Detection of Intrusions and Malware & Vulnerability Assessment, LNCS 4064*, pages 109–128. Springer-Verlag, 2006.

[DW01] Hervé Debar and Andreas Wespi. Aggregation and correlation of intrusion-detection alerts. In *RAID '00: Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection*, pages 85–103, London, UK, 2001. Springer-Verlag.

[GGPW$^+$01] Fengmin Gong, Katerina Goseva-Popstojanova, Feiyi Wang, Rong Wang, Kalyanaraman Vaidyanathan, Kishor Trivedi, and Balamurugan Muthusamy. Characterizing intrusion tolerant systems using a state transition model. In *DARPA Information Survivability Conference and Exposition (DISCEX II)*, volume 2, 2001.

[GK04] Ashish Gehani and Gershon Kedem. Rheostat: Real-time risk management. In *Recent Advances in Intrusion Detection: 7th International Symposium, RAID 2004, Sophia Antipolis, France, September 15-17, 2004. Proceedings*, pages 296–314. Springer, 2004.

[GSW03] Fei Gao, Jizhou Sun, and Zunce Wei. The prediction role of hidden markov model in intrusion detection. In *IEEE CCECE 2003: Canadian Conference on Electrical and Computer Engineering*, volume 2, pages 893–896, 2003.

[HAK07] Kjetil Haslum, Ajith Abraham, and Svein Knapskog. Dips: A framework for distributed intrusion prediction and prevention using hidden markov models and online fuzzy risk assessment. In *Third International Symposium* *on Information Assurance and Security, IEEE Computer Society press*, volume I, pages 183–188, 2007.

[HAK08] Kjetil Haslum, Ajith Abraham, and Svein Knapskog. Fuzzy online risk assessment for distributed intrusion prediction and prevention systems. In *Tenth International Conference on Modeling and Simulation, IEEE Computer Society press*, volume I, page in press, 2008.

[HWH$^+$03] Guy Helmer, Johnny S. K. Wong, Vasant G. Honavar, Les Miller, and Yanxin Wang. Lightweight agents for intrusion detection. *J. Syst. Softw.*, 67(2):109–122, 2003.

[ISO05] ISO/IEC JTC 1, SC 27. *ISO/IEC 15408 Part 1-3:2005, Common Criteria for Information Technology Security Evaluation.* ISO, Geneva, Switzerland, 2005.

[Jan93] J.-S.R. Jang. Anfis: adaptive-network-based fuzzy inference system. *Systems, Man and Cybernetics, IEEE Transactions on*, 23(3):665–685, May/Jun 1993.

[JO97] Erland Jonsson and Tomas Olovsson. A quantitative model of the security intrusion process based on attacker behavior. *IEEE Transactions on Software Engineering*, 23(4):235–245, 1997.

[Jon06] Jack Jones. An introduction to factor analysis of information risk (fair). *Norwich Journal of Information Assurance*, 2(1):67, 2006.

[Jul01] K. Julisch. Mining alarm clusters to improve alarm handling efficiency. In *ACSAC '01: Proceedings of the 17th Annual Computer Security Applications Conference*, page 12, Washington, DC, USA, 2001. IEEE Computer Society.

[KAN$^+$06] M Kaâniche, E Alata, V Nicomette, Yves Deswarte, and Marc Dacier. Empirical analysis and statistical modeling of attack processes based on honeypots. In *WEEDS 2006 - Workshop on empirical evaluation of dependability and security (in conjunction with the international conference on dependable systems and networks, DSN 2006), June 25 - 28, 2006, Philadelphia,USA*, Jun 2006.

[Ken99] Kristopher Kendall. A database of computer attacks for the evaluation of intrusion detection systems. Master's thesis, MIT, USA, June 1999.

[KKM$^+$05] Christopher Krügel, Engin Kirda, Darren Mutz, William K. Robertson, and Giovanni Vigna. Polymorphic worm detection using structural information of executables. In Alfonso Valdes and Diego Zamboni, editors, *RAID*, volume 3858 of *Lecture Notes in Computer Science*, pages 207–226. Springer, 2005.

[KL06] Rahul Khanna and Huaping Liu. System approach to intrusion detection using hidden markov model. In *IWCMC '06: Proceeding of the 2006 international conference on Communications and mobile computing*, pages 349–354, New York, NY, USA, 2006. ACM Press.

[KL07] Rahul Khanna and Huaping Liu. Distributed and control theoretic approach to intrusion detection. In *IWCMC '07: Proceedings of the 2007* *international conference on Wireless communications and mobile computing*, pages 115–120, New York, NY, USA, 2007. ACM.

[Lab00] Lincoln Laboratory. Lincoln laboratory scenario (ddos) 1.0, 2000.

[LBF$^+$93] B. Littlewood, S. Brocklehurst, N. Fenton, P. Mellor, S. Page, D. Wright, J. Dobson, J. McDermid, and D. Gollmann. Towards operational measures of computer security. *Journal of Computer Security*, 2:211–229, 1993.

[MA75] E.H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7(1):1–13, 1975.

[Mil] John A. Miller. Jsim: A java-based simulation and animation environment. `http://chief.cs.uga.edu/~jam/jsim/`.

[MVT02] B.B. Madan, K. Vaidyanathan, and K.S. Trivedi. Modeling and quantification of security attributes of software systems. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN'02)*, 2002.

[NCSD09] Information Technology Laboratory NIST Computer Security Division. National vulnerability database version 2.2, 2009. http://nvd.nist.gov/.

[NST04] David M. Nicol, William H. Sanders, and Kishor S. Trivedi. Model-based evaluation: From dependability to security. *IEEE Transactions on Dependable and Secure Computing*, 01(1):48–65, 2004.

[OD99] Rodolphe Ortalo and Yves Deswarte. Experimenting with quantitative evaluation tools for monitoring operational security. *IEEE Transactions on Software Engineering*, 25:633–650, 1999.

[OMSH03] Dirk Ourston, Sara Matzner, William Stump, and Bryan Hopkins. Applications of hidden markov models to detecting multi-stage network attacks. In *Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS)*, 2003.

[PGR06] Roberto Perdisci, Giorgio Giacinto, and Fabio Roli. Alarm clustering for intrusion detection systems in computer networks. *Engineering Applications of Artificial Intelligence*, 19(4):429 – 438, 2006. Recent Advances in Data Mining.

[PN97] P. A. Porras and P. G. Neumann. EMERALD: Event monitoring enabling responses to anomalous live disturbances. In *Proc. 20th NIST-NCSC National Information Systems Security Conference*, pages 353–365, 1997.

[Rab90] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Readings in speech recognition*, pages 267–296, 1990.

[Ros03] Sheldon M. Ross. *Introduction to Probability Models.* Academic Press, New York, 8th edition, 2003.

[SBD$^+$91] Steven R. Snapp, James Brentano, Gihan V. Dias, Terrance L. Goan, L. Todd Heberlein, Che lin Ho, Karl N. Levitt, Biswanath Mukherjee, Stephen E. Smaha, Tim Grance, Daniel M. Teal, and Doug Mansur. DIDS (distributed intrusion detection system) - motivation, architecture, and an early prototype. In *Proceedings of the 14th National Computer Security Conference*, pages 167–176, Washington, DC, 1991.

[SCCC$^+$96] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle. GrIDS – A graph-based intrusion detection system for large networks. In *Proceedings of the 19th National Information Systems Security Conference*, 1996.

[SCS03] 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 2003.

[SGF02] Gary Stonebumer, Alice Goguen, and Alexis Feringa. Risk management guide for information technology systems, National Institute of Standards and Technology, special publication 800-30, 2002. http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf.

[SHJ$^+$02] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J.M. Wing. Automated generation and analysis of attack graphs. *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pages 273–284, 2002.

[SHK05] Karin Sallhammar, Bjarne E. Helvik, and Svein J. Knapskog. Incorporating attacker behavior in stochastic models of security. In Hamid R. Arabnia, editor, *Security and Management*, pages 79–85. CSREA Press, 2005.

[SHK06] Karin Sallhammar, Bjarne E. Helvik, and Svein J. Knapskog. A game-theoretic approach to stochastic security and dependability evaluation. In *DASC*, pages 61–68. IEEE Computer Society, 2006.

[SHK07] Karin Sallhammar, Bjarne E. Helvik, and Svein J. Knapskog. A framework for predicting security and dependability measures in real-time. *International Journal of Computer Science and Network Security (IJCSNS)*, 7(3):169–183, 2007.

[SKH05] Karin Sallhammar, Svein J. Knapskog, and Bjarne E. Helvik. Using stochastic game theory to compute the expected behavior of attackers. In *SAINT Workshops*, pages 102–105. IEEE Computer Society, 2005.

[Sta04] Standards Australia and Standards New Zealand. AS/NZS 4360: 2004 risk management, 2004.

[Sug85] Michio Sugeno. *Industrial Applications of Fuzzy Control*. Elsevier Science Inc., New York, NY, USA, 1985.

[Ven96] D.E. Ventzas. Kalman filters for dynamic position control of large scale systems. *Industrial Electronics, Control, and Instrumentation, 1996., Proceedings of the 1996 IEEE IECON 22nd International Conference on*, 2:647–652 vol.2, 5-10 Aug 1996.

[VKB01] Giovanni Vigna, Richard A. Kemmerer, and Per Blix. Designing a web of highly-configurable intrusion detection sensors. In *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID 2000)*, pages 69–84, London, UK, 2001. Springer-Verlag.

[VVK03] Giovanni Vigna, Fredrik Valeur, and Richard A. Kemmerer. Designing and implementing a family of intrusion detection systems. *SIGSOFT Softw. Eng. Notes*, 28(5):88–97, 2003.

[VVKK04] Fredrik Valeur, Giovanni Vigna, Christopher Kruegel, and Richard A. Kemmerer. A comprehensive approach to intrusion detection alert correlation. *IEEE Transactions on Dependable and Secure Computing*, 1(3):146–169, 2004.

[WFP99] C. Warrender, S. Forrest, and B. Pearlmutter. Detecting intrusions using system calls: Alternative data models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, 1999.

[WGZ04] Wei Wang, Xiao-Hong Guan, and Xiang-Liang Zhang. Modeling program behaviors by hidden markov models for intrusion detection. *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, 5:2830–2835 vol.5, Aug. 2004.

[WMT03] Dazhi Wang, Bharat B. Madan, and Kishor S. Trivedi. Security analysis of sitar intrusion tolerance system. In *SSRS '03: Proceedings of the 2003 ACM workshop on Survivable and self-regenerative systems*, pages 23–32, New York, NY, USA, 2003. ACM.

[WWT02] Wei Wei, Bing Wang, and Don Towsley. Continuous-time hidden markov models for network performance evaluation. *Performance Evaluation*, 49:129–146, 2002.

[Zad65] L.A. Zadeh. Fuzzy sets. *Info. & Ctl.*, 8:338–353, 1965.

II

INCLUDED PAPERS

Paper A

Real-time Risk Assessment with Network Sensors and Intrusion Detection Systems

André Årnes, Karin Sallhammar, Kjetil Haslum, Tønnes Brekne, Marie Elisabeth Gaup Moe and Svein Johan Knapskog.

In Proceedings of the 2005 International Conference on Computational Intelligence and Security (CIS'05)

Xian, China. December 15-19, 2005

REAL-TIME RISK ASSESSMENT WITH NETWORK SENSORS AND INTRUSION DETECTION SYSTEMS

André Årnes, Karin Sallhammar, Kjetil Haslum, Tønnes Brekne,
Marie Elisabeth Gaup Moe, and Svein Johan Knapskog
Centre for Quantifiable Quality of Service in Communication Systems
Norwegian University of Science and Technology
O.S. Bragstads plass 2E, N-7491 Trondheim, Norway
{andrearn,sallhamm,haslum,tonnes,marieeli,knapskog}@q2s.ntnu.no

Abstract This paper considers a real-time risk assessment method for information systems and networks based on observations from networks sensors such as intrusion detection systems. The system risk is dynamically evaluated using hidden Markov models, providing a mechanism for handling data from sensors with different trustworthiness in terms of false positives and negatives. The method provides a higher level of abstraction for monitoring network security, suitable for risk management and intrusion response applications.

1. Introduction

Risk assessment is a central issue in management of large-scale networks. However, current risk assessment methodologies focus on manual risk analysis of networks during system design or through periodic reviews. Techniques for real-time risk assessment are scarce, and network monitoring systems and intrusion detection systems (IDS) are the typical approaches. In this paper, we present a real-time risk assessment method for large scale networks that build upon existing network monitoring and intrusion detection systems. An additional level of abstraction is added to the network monitoring process, focusing on risk rather than individual warnings and alerts. The method enables the assessment of risk both on a system-wide level, as well as for individual objects.

The main benefit of our approach is the ability to aggregate data from different sensors with different weighting according to the trustworthiness of the sensors. This focus on an aggregate risk level is deemed more suitable for network management and automated response than individual intrusion detection alerts. By using hidden Markov models (HMM), we can find the most likely state probability distribution of monitored objects, considering the trustworthiness of the IDS. We do not make any assumptions on the types of sensors used in our monitoring architecture, other than that they are capable of providing standardized output as required by the model parameters presented in this paper.

1.1 Target Network Architecture

The target of the risk assessment described in this paper is a generic network consisting of computers, network components, services, users, etc. The network can be arbitrarily complex, with wireless ad-hoc devices as well as ubiquitous services. The network consists of entities that are either *subjects* or *objects*. Subjects are capable of performing actions on the objects. A subject can be either users or programs, whereas objects are the targets of the risk assessment. An asset may be considered an object. The unknown factors in such a network may represent vulnerabilities that can be exploited by a malicious attacker or computer program and result in unwanted incidents. The potential exploitation of a vulnerability is described as threats to assets. The *risk* of a system can be identified through the evaluation of the probability and consequence of unwanted incidents.

1.2 Monitoring and Assessment Architecture

We assume a multiagent system architecture consisting of agents that observe objects in a network using sensors. The architecture of a multiagent risk assessment system per se is not the focus of this paper, but a description is included as a context.

An *agent* is a computer program capable of a certain degree of autonomous actions. In a multiagent system, agents are capable of communicating and cooperating with other agents. In this paper, an agent is responsible for collecting and aggregating sensor data from a set of sensors that monitor a set of objects. The main task of the agent is to perform real-time risk assessment based on these data. A multiagent architecture has been chosen for its flexibility and scalability, and in order to support distributed automated response.

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, honeypots, etc. The main task of the sensors is to gather information regarding the security state of objects. 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 risk assessment model described in this paper.

1.3 Related Work

Risk assessment has traditionally been a manual analysis process based on a standardized framework, such as [Sta04]. A notable example of real-time risk assessment is presented in [GK04], which introduces a formal model for the real time characterization of risk faced by a host. *Distributed intrusion detection systems* have been demonstrated in several prototypes and research papers, such as [SCCC$^+$96, SBD$^+$91]. Multiagent systems for intrusion detection, as proposed in [BGFI$^+$98] and demonstrated in e.g. [HWH$^+$03] (an IDS prototype based on lightweight mobile agents) are of particular relevance for this paper. An important development in distributed intrusion detection is the recent IDMEF (Intrusion Detection Message Exchange Format) IETF Internet draft [DCF05]. *Hidden Markov models* have recently been used in IDS architectures to detect multi-stage attacks [OMSH03], and as a tool to detect misuse based on operating system calls [WFP99]. *Intrusion tolerance* is a recent research field in information security related to the field of fault tolerance in networks. The research project SITAR [GGPW$^+$01] presents a generic state transition model, similar to the model used in this paper, to describe the dynamics of intrusion tolerant systems. Probabilistic validation of intrusion tolerant systems is presented in [SCS03].

2. Risk Assessment Model

In order to be able to perform dynamic risk assessment of a system, we formalize the distributed network sensor architecture described in the previous section. Let $O = \{o_1, o_2, \ldots\}$ be the set of objects that are monitored by an agent. This set of objects represents the part of the network that the agent is responsible for. To describe the security state of each object, we use discrete-time Markov chains. Assume that each object consisting of N states, denoted $S = \{s_1, s_2, \ldots, s_N\}$.

As the security state of an object changes over time, it will move between the states in S. The sequence of states that an object visits is denoted $X = x_1, x_2, \ldots, x_T$, where $x_t \in S$ is the state visited at time t. For the purpose of this paper, we assume that the state space can be represented by a general model consisting of three states: Good (G), Attacked (A) and Compromised (C), i.e. $S = \{G, A, C\}$. State G means that the object is up and running securely and that it is not subject to any kind of attack actions. In contrast to [GGPW$^+$01], we assume that objects always are vulnerable to attacks, even in state G. As an attack against an object is initiated, it will move to security state A. An object in state A is subject to an ongoing attack, possibly affecting its behavior with regard to security. Finally, an object enters state C if it has been successfully compromised by an attacker. An object in state C is assumed to be completely at the mercy of an attacker and subject to any kind of confidentiality, integrity and/or availability breaches.

The security observations are provided by the sensors that monitor the objects. These *observation messages* are processed by agents, and it is assumed that the messages are received or collected at *discrete time intervals*. An observation message can consist of any of the symbols $V = \{v_1, v_2, \ldots, v_M\}$. These symbols may be used to represent different types of alarms, suspect traffic patterns, entries in log data files, input from network administrators, and so on. The *sequence* of observed messages that an agent receives is denoted $Y = y_1, y_2, \ldots, y_T$, where $y_t \in V$ is the observation message received at time t. Based on the sequence of observation messages, the agent performs dynamic risk assessment. The agent will often receive observation messages 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 attacks, so we cannot assume that an agent is able to resolve the correct state of the monitored objects at all times. The observation symbols are therefore probabilistic functions of the object's Markov chain, the object's true security state will be *hidden* from the agent. This is consistent with the basic idea of HMM [Rab90].

2.1 Modeling Objects as Hidden Markov Models

Each monitored object can be represented by a HMM, defined by $\lambda = \{\mathbf{P}, \mathbf{Q}, \pi\}$.

$\mathbf{P} = \{p_{ij}\}$ is the state transition probability distribution matrix for object o, where $p_{ij} = P(x_{t+1} = s_j | x_t = s_i), 1 \leq i, j \leq N$. Hence, p_{ij} represents the probability that object o will transfer into state s_j next, given that its current state is s_i. To be able to estimate $\mathbf{P}$ for real-life objects, one may use either statistical attack data from production or experimental systems or the subjective opinion of experts. Learning algorithms may be employed in order to provide a better estimate of $\mathbf{P}$ over time.

$\mathbf{Q} = \{q_j(l)\}$ is the observation symbol probability distribution matrix for object o in state s_j, whose elements are $q_j(l) = P(y_t = v_l | x_t = s_j), 1 \leq j \leq N, 1 \leq l \leq M$. In our model, the element $q_j(l)$ in $\mathbf{Q}$ represents the probability that a sensor will send the observation symbol v_l at time t, given that the object is in state s_j at time t. $\mathbf{Q}$ therefore indicates the sensor's false-positive and false-negative effect on the agents risk assessments.

$\pi = \{\pi_i\}$ is the initial state distribution for the object. Hence, $\pi_i = P(x_1 = s_i)$ is the probability that s_i was the initial state of the object.

2.2 Quantitative Risk Assessment

Following the terminology in [Sta04], risk is measured in terms of *consequences* and *likelihood*. A consequence is the (qualitative or quantitative) outcome of an event and the likelihood is a description of the probability of the event. To perform dynamic risk assessment, we need a mapping: $\mathcal{C} : S \rightarrow \mathbb{R}$, describing the expected cost (due to loss of confidentiality, integrity and availability) for each object. The total risk $\mathcal{R}_t$ for an object at time t is

$$\mathcal{R}_t = \sum_{i=1}^{N} \mathcal{R}_t(i) = \sum_{i=1}^{N} \gamma_t(i)\mathcal{C}(i) \tag{1}$$

where $\gamma_t(i)$ is the probability that the object is in security state s_i at time t, and $\mathcal{C}(i)$ is the cost value associated with state s_i.

In order to perform real-time risk assessment for an object, an agent has to dynamically update the object's state probability $\gamma_t = \{\gamma_t(i)\}$. Given an observation y_t, and the HMM λ, the agent can update the state probability γ_t of an object using Algorithm 1. The complexity of the algorithm is $O(N^2)$. For further details, see the Appendix.

Algorithm 1 Update state probability distribution

```
Require: $y_t, \lambda$ {the observation at time $t$, the hidden Markov model}
Ensure: $\gamma_t$ {the security state probability at time $t$}
  if $t = 1$ then
    for $i = 1$ to $N$ do
      $\alpha_1(i) \leftarrow q_i(y_1)\pi_i$
      $\gamma_1(i) \leftarrow \frac{q_i(y_1)\pi_i}{\sum_{j=1}^{N} q_j(y_1)\pi_j}$
    end for
  else
    for $i = 1$ to $N$ do
      $\alpha_t(i) \leftarrow q_i(y_t) \sum_{j=1}^{N} \alpha_{t-1}(j)p_{ji}$
      $\gamma_t(i) \leftarrow \frac{\alpha_t(i)}{\sum_{j=1}^{N} \alpha_t(j)}$
    end for
  end if
  return $\gamma_t$
```

3. Case – Real-time Risk Assessment for a Home Office

To illustrate the theory, we perform real-time risk assessment of a typical home office network, consisting of an Internet router/WLAN access point, a stationary computer with disk and printer sharing, a laptop using WLAN, and a cell phone connected to the laptop using Bluetooth. Each of the objects (hosts) in the home office network has a sensor that processes log files and checks system integrity (a host IDS). In addition, the access point has a network monitoring sensor that is capable of monitoring traffic between the outside network and the internal hosts (a network IDS).

For all objects, we use the state set $S = \{G, A, C\}$. The sensors provide observations in a standardized message format, such as IDMEF, and they are capable of classifying observations as indications of the object state. Each sensor is equipped with a database of signatures of potential attacks. For the purpose of this example, each signature is associated with a particular state in S. We define the observation symbols set as $V = \{g, a, c\}$, where the symbol g is an indication of state G and so forth. Note that we have to preserve the discrete-time property of the HMM by sampling sensor data periodically. If there are multiple observations during a period, we sample one at random. If there are no observations, we assume the observation symbol to be g. In order to use multiple sensors for a single object, a round-robin sampling is used to process only one observation for each period. This is demonstrated in example 3.

The home network is monitored by an agent that regularly receives observation symbols from the sensors. For each new symbol, the agent uses Algorithm 1 to update the objects' security state probability, and (1) to compute its corresponding risk value. Estimating the matrices $\mathbf{P}$ and $\mathbf{Q}$, as well as the cost $\mathcal{C}$ associated with the different states, for the objects in this network is a non-trivial task that is out of scope for this paper.

The parameter values in these examples are therefore chosen for illustration purposes only. Also, we only demonstrate how to perform dynamic risk assessment of the laptop.

3.1 Example 1: Laptop Risk Assessment by HIDS Observations

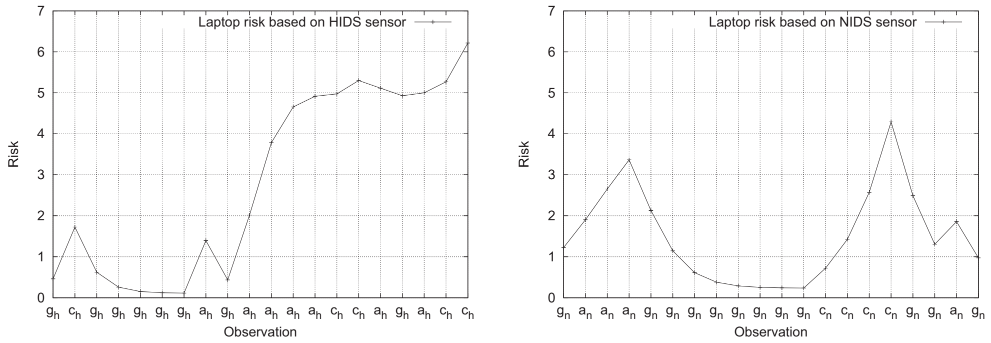
First, we assess the risk of the laptop, based on an observation sequence Y_{HIDS-L}, containing 20 samples collected from the laptop HIDS. We use the HMM $\lambda_L = \{\mathbf{P}_L, \mathbf{Q}_{HIDS-L}, \pi_L\}$, where

$$\mathbf{P}_L = \begin{pmatrix} p_{GG} & p_{GA} & p_{GC} \\ p_{AG} & p_{AA} & p_{AC} \\ p_{CG} & p_{CA} & p_{CC} \end{pmatrix} = \begin{pmatrix} 0.995 & 0.004 & 0.001 \\ 0.060 & 0.900 & 0.040 \\ 0.008 & 0.002 & 0.990 \end{pmatrix}, \tag{2}$$

$$\mathbf{Q}_{HIDS-L} = \begin{pmatrix} q_G(g) & q_G(a) & q_G(c) \\ q_A(g) & q_A(a) & q_A(c) \\ q_C(g) & q_C(a) & q_C(c) \end{pmatrix} = \begin{pmatrix} 0.70 & 0.15 & 0.15 \\ 0.15 & 0.70 & 0.15 \\ 0.20 & 0.20 & 0.60 \end{pmatrix}, \tag{3}$$

$$\pi_L = (\pi_G, \pi_A, \pi_C) = (0.8, 0.1, 0.1). \tag{4}$$

Since the HIDS is assumed to have low false-positive and false-negative rates, both $q_G(a), q_G(c), q_A(c) \ll 1$ and $q_A(g), q_C(g), q_C(a) \ll 1$ in $\mathbf{Q}_{HIDS-L}$. The dynamic risk in Figure 1(a) is computed based on the observation sequence Y (as shown on the x-axis of the figure) and a security state cost estimate measured as $\mathcal{C}_L = (0, 5, 10)$.



(a) HIDS sensor

(b) NIDS sensor

Figure 1. Laptop risk assessment

3.2 Example 2: Laptop Risk Assessment by NIDS Observations

Now, we let the risk assessment process of the laptop be based on another observation sequence, Y_{NIDS-L}, collected from the NIDS. A new observation symbol probability distribution is created for the NIDS

$$\mathbf{Q}_{NIDS-L} = \begin{pmatrix} 0.5 & 0.3 & 0.2 \\ 0.2 & 0.6 & 0.2 \\ 0.2 & 0.2 & 0.6 \end{pmatrix}. \tag{5}$$

One can see that the NIDS has higher false-positive and false-negative rates, compared to the HIDS. Figure 1(b) shows the laptop risk when using the HMM $\lambda_L = \{\mathbf{P}_L, \mathbf{Q}_{NIDS-L}, \pi_L\}$. Note that the observation sequence is not identical to the one in example 1, as the two sensors are not necessarily consistent.

3.3 Example 3: Aggregating HIDS and NIDS Observations

The agent now aggregates the observations from the HIDS and NIDS sensors by sampling from the observation sequences Y_{HIDS-L} and Y_{NIDS-L} in a round-robin fashion. To update the current state probability γ_t, the agent therefore chooses the observation symbol probability distribution corresponding to the sampled sensor, i.e the HMM will be

$$\lambda_L = \{\mathbf{P}_L, \mathbf{Q}^*, \pi_L\}, \text{where } \mathbf{Q}^* = \begin{cases} \mathbf{Q}_{HIDS-L} \text{ if } y_t \in Y_{HIDS} \\ \mathbf{Q}_{NIDS-L} \text{ if } y_t \in Y_{NIDS} \end{cases}. \tag{6}$$

The calculated risk is illustrated in Figure 2. The graph shows that some properties of the individual observation sequences are retained.

4. Managing Risk with Automated Response

In order to achieve effective incident response, it must be possible to effectively initiate defensive measures, for example by reconfiguring the security services and mechanisms in order to mitigate risk. Such measures may be manual or automatic. An information system or network can be automatically reconfigured in order to reduce an identified risk, or the system can act as a support system for system and network administrators by providing relevant information and recommending specific actions. To facilitate such an approach, it is necessary to provide a mechanism that relates a detected security incidence to an appropriate response, based on the underlying risk model. Such a mechanism should include a policy for what reactions should be taken in the case of a particular incident, as well as information on who has the authority to initiate or authorize the response. Examples of distributed intrusion detection and response systems have been published in [CHSP00, PN97].

The dynamic risk-assessment method described in this paper can provide a basis for automated response. If the risk reaches a certain level, an agent may initiate an automated response in order to control the risk level. Such a response may be performed both for individual objects (e.g. a compromised host) or on a network-wide level (if the network risk level is to high). Examples of a local response may be firewall reconfigurations for a host, changing logging granularity, or shutting down a system. Examples of a global response may be the revocation of a user certificate, the reconfiguration of central access control configurations, or firewall reconfigurations. Other examples include traffic rerouting or manipulation, and honeypot technologies. Note that such adaptive measures has to be supervised by human intelligence, as they necessarily introduce a risk in their own right. A firewall reconfiguration mechanism can, for example, be exploited as part of a denial-of-service attack.

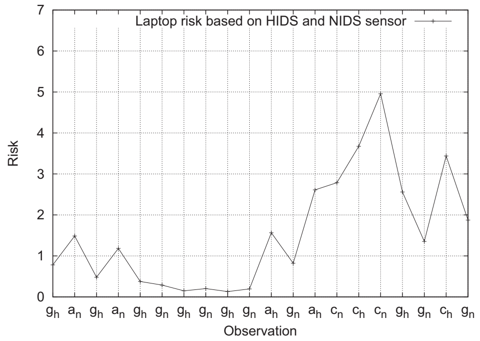


Figure 2. Laptop risk assessment based on two sensors (HIDS and NIDS)

5. Conclusion

We present a real-time risk-assessment method using HMM. The method provides a mechanism for aggregating data from multiple sensors, with different weightings according to sensor trustworthiness. The proposed discrete-time model relies on periodic messages from sensors, which implies the use of sampling of alert data. For the purpose of real-life applications, we propose further development using continuous-time models in order to be able to handle highly variable alert rates from multiple sensors. We also give an indication as to how this work can be extended into a multiagent system with automated response, where agents are responsible for assessing and responding to the risk for a number of objects.

Appendix: On Algorithm 1

Given the first observation y_1 and the hidden Markov model λ, the initial state distribution $\gamma_1(i)$ can be calculated as

$$\gamma_1(i) = P(x_1 = s_i|y_1, \lambda) = \frac{P(y_1, x_1 = s_i|\lambda)}{P(y_1|\lambda)} = \frac{P(y_1|x_1 = s_i, \lambda)P(x_1 = s_i|\lambda)}{P(y_1|\lambda)}. \tag{A.1}$$

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

$$P(y_1|\lambda) = \sum_{j=1}^{N} P(y_1|x_1 = s_j, \lambda)P(x_1 = s_j|\lambda) = \sum_{j=1}^{N} q_j(y_1)\pi_j. \tag{A.2}$$

Hence, by combining (A.1) and (A.2)

$$\gamma_1(i) \quad = \quad \frac{q_i(y_1)\pi_i}{\sum_{j=1}^{N} q_j(y_1)\pi_j}, \tag{A.3}$$

where $q_j(y_1)$ is the probability of observing symbol y_1 in state s_j, and π is the initial state probability. To simplify the calculation of the state distribution after t observations we use the *forward-variable* $\alpha_t(i) = P(y_1 y_2 \cdots y_t, x_t = s_i|\lambda)$, as defined in [Rab90]. By using recursion, this variable can be calculated in an efficient way as

$$\alpha_t(i) = q_i(y_t) \sum_{j=1}^{N} \alpha_{t-1}(j)p_{ji}, \qquad t > 1. \tag{A.4}$$

From (A.1) and (A.3) we find the initial forward variable

$$\alpha_1(i) = q_i(y_1)\pi_i, \qquad t = 1. \tag{A.5}$$

In the derivation of $\alpha_t(i)$ we assumed that y_t only depend on x_t and that the Markov property holds.

Now we can use the forward variable $\alpha_t(i)$ to update the state probability distribution by new observations. This is done by

$$\begin{aligned}\gamma_t(i) = P(x_t = s_i|y_1y_2\cdots y_t,\lambda) &= \frac{P(y_1y_2\cdots y_t, x_t = s_i|\lambda)}{P(y_1y_2\cdots y_t|\lambda)} \\ &= \frac{P(y_1y_2\cdots y_t, x_t = s_i|\lambda)}{\sum_{j=1}^{N} P(y_1y_2\cdots y_t, x_t = s_j|\lambda)} = \frac{\alpha_t(i)}{\sum_{j=1}^{N}\alpha_t(j)}.\end{aligned} \tag{A.6}$$

Note that (A.6) is similar to Eq. 27 in [Rab90], with the exception that we do not account for observations that occur after t, as our main interest is to calculate the object's state distribution after a number of observations.

Paper B

Real-time Risk Assessment with Network Sensors and Hidden Markov Models

André Årnes, Karin Sallhammar, Kjetil Haslum, and Svein Johan Knapskog.

In Proceedings of the 11th Nordic Workshop on Secure IT-systems

Linköping, Sweden. October 19-20, 2006

REAL-TIME RISK ASSESSMENT WITH NETWORK SENSORS AND HIDDEN MARKOV MODELS

André Årnes, Karin Sallhammar,
Kjetil Haslum, and Svein Johan Knapskog
Centre for Quantifiable Quality of Service in Communication Systems
Norwegian University of Science and Technology
O.S. Bragstads plass 2E, N-7491 Trondheim, Norway
{andrearn,sallhamm,haslum,knapskog}@q2s.ntnu.no

Abstract This paper presents a method for real-time risk assessment of large-scale networks. The method provides a mechanism for handling data from multiple, heterogeneous sensors with different levels of trustworthiness. It aims to serve as a higher level of abstraction for applications, such as risk management, network monitoring and incident response. In order to assess network risk in a real-time setting, the method is adapted to approximate continuous time system behavior. In addition, design issues, such as the use of multiple sensors and the queuing of sensor observations, are addressed. A discrete-event simulator is implemented, demonstrating the real-time risk assessment based on observations from intrusion detection systems. Using this simulator, we model and simulate the risk assessment of a large network and compare the results to the true values in order to validate the proposed approach.

1. Introduction

The practice of risk management is becoming increasingly important as information systems and networks are growing more complex and interconnected. Traditional risk assessment techniques focus on manual analysis of network components during system design, but do not address the dynamic assessment of network risk. On the other hand, intrusion detection systems (IDS) focus on the detection and reporting of security-related events, but often fail to provide an overview of the overall network risk or the priorities of observed alerts. In order to handle incidents in a more appropriate and efficient manner, methods for automated real-time risk assessment are needed. In this paper, we present and further develop a method for real-time risk assessment based on hidden Markov models (HMMs). The purpose of this approach is ultimately to effectively identify, prioritize, and respond to threats to critical assets.

This paper begins by presenting previous and related work on distributed IDS and real-time risk assessment. The proposed reference model and the basic risk assessment method are presented in Section 2 and 3. Section 3 also explains how the model can be used to approximate a continuous time setting, adapted to deal with bursts of observation data. Section 4 describes the implementation of a discrete-time, discrete-event simulator. Section 5 we provide results from simulation experiments. Finally, Section 6 concludes the paper and points to future work.

1.1 Previous Work

We have previously outlined a methodology for real-time risk assessment using HMMs in a multiagent system architecture [ÅSH$^+$05]. The method provides a higher level of abstraction for network security monitoring, suitable for risk management and incident response applications. The system relies on input from a number of heterogeneous sensors, typically IDS, with different trustworthiness in terms of false positives and negatives. This paper further develops and addresses some of the limitations of this initial approach.

HMMs are discrete-time models, inherently not suitable for continuous time sensor data. Moreover, the use of multiple sensors is not directly supported by HMMs. In [ÅSH$^+$05], these limitations were addressed by using a round-robin sampling of observations from sensors. A major drawback of this approach is that it cannot, without loss, handle the arrival of simultaneous observations. In real-life applications, one must be able to handle and correctly interpret bursts of alerts from several sensors, as well as observations arriving sparsely distributed in time. Also, [ÅSH$^+$05] did not consider parameter estimation. A good model parameterization is crucial in order to obtain accurate results from the risk assessment process. In order to reduce the number of individual initial parameter evaluations, it is desirable to generalize the parameters for similar objects through the use of parameter profiles. We address these issues by estimating the security states of the monitored assets at the sensor level, and by using generalized profiles to simplify the model parameterization task. The security state probabilities for an asset are computed for each sensor using the HMM method, and the risk is in turn computed for each asset as a function of all its sensor input. A profile represents a class of assets or sensors with common attributes. Simulation experiments are provided to demonstrate and validate the method using a realistic scenario.

1.2 Related Work

Risk assessment has traditionally been a manual analysis process based on a standardized framework, such as those recommended by NIST [SGF02] and AS/NZS [Sta04]. A notable example of real-time risk assessment is presented in [GK04], which introduces a formal model for real-time characterization of the risk faced by a host. *Intrusion tolerance* is a recent research field in information security related to the field of fault tolerance in network dependability. The research project SITAR [GGPW$^+$01] presents a generic state transition model, similar to the model used in this paper, to describe the dynamics of intrusion tolerant systems. Probabilistic validation of intrusion tolerant systems is presented in e.g., [SCS03].

Distributed intrusion detection systems are discussed in several research papers, such as DIDS [SBD$^+$91] and GrIDS [SCCC$^+$96], and a multi-agent system for intrusion detection was proposed in [BGFI$^+$98]. STAT [VKB01] is a state-based IDS that uses state modeling to describe and detect attacks. An alert correlation framework providing for STAT is presented in [VVKK04]. *Hidden Markov models* have been used in IDS architectures to detect multi-stage attacks [OMSH03], and as a tool to detect misuse based on operating system calls [WFP99]. Our approach shares some attributes with several of these systems, but we attempt to study the network risk at a higher abstraction level, rather than to detect specific attacks and intrusions. The recent IDMEF (Intrusion Detection Message Exchange Format) IETF Internet draft [DCF05] is expected to be a suitable language for message exchange between IDS sensors and the risk assessment system proposed in this paper.

2. Terminology and Reference Model

This section provides a brief description of the terminology and the reference model used in this paper. We discuss both the *target network architecture* and the *monitoring and assessment architecture*. Ideally, these systems can be assumed to be independent. See [ÅSH$^+$05] for a more detailed description.

2.1 Target Network Architecture

The target of the risk assessment process is a generic, arbitrarily complex computer network, consisting of *entities* that are either *subjects* or *objects*. The subjects are capable of performing actions on the objects. For the purpose of the risk assessment in this paper, an object is considered to be an *asset*. Unknown factors in such a network may represent *vulnerabilities* that in turn can be *exploited* by a malicious attacker or computer program, causing *unwanted incidents*. The potential exploitation of vulnerabilities can be described as *threats* to the assets. The *risk* of the network can be estimated by evaluating the probability and consequence of unwanted incidents.

2.2 Monitoring and Assessment Architecture

As in [ÅSH$^+$05], we assume a multiagent system architecture consisting of *agents* and *sensors*. A *sensor* primarily refers to an IDS, but can be any information-gathering program or device capable of collecting security relevant data, such as logging systems, virus detectors, honeypots, and network sniffers using sampling or filtering. Their main task is to gather information about the security state of assets and to send standardized observation messages to the agents. An *agent* is a computer program capable of a certain degree of autonomous actions. In this paper, agents are responsible for performing real-time risk assessment based on data collected from a number of sensors monitoring one or more assets. The multiagent architecture has been chosen for its flexibility and scalability, in order to support future applications, such as distributed automated response.

3. The Risk Assessment Model

This section formalizes the proposed risk assessment model. The model is based on [ÅSH$^+$05], but proposes some modifications and improvements.

3.1 Modeling Assets as Hidden Markov Models

Assume that the security of an asset can be modeled by N states, denoted $S = \{s_1, \ldots, s_N\}$. A state refers to an operational mode of the asset characterized by which units of the assets that are operational or failed, whether there are ongoing attacks, active countermeasures, operational or maintenance activities, whether the asset is compromised or not, etc. The decision of what to include in the state definition is a trade-off between model expressiveness and complexity. Different applications will likely require different state models. An example primary for illustration will be presented in Section 5. The behavior of an asset is characterized by the transitions between its states. Due to attack attempts and compromises, or administrative activities, the security state of an asset will change over time. The sequence of states visited is denoted $X = x_1, x_2, \ldots$, where $x_t \in S$ is the state visited at time t. We assume that the probability of future states depend only on the current system state, i.e., $P(x_{t+1} = s_i | x_t, x_{t-1}, \ldots, x_1) = P(x_{t+1} = s_i | x_t)$. Hence, the security behavior of an asset can be modeled by a Markov chain.

The risk *observation messages* are provided by the K sensors monitoring an asset, indexed by $k \in \{1, \ldots, K\}$. An observation message from sensor k can consist of any of the symbols in the observation symbol set $V^k = \{v_1^k, \ldots, v_{M_k}^k\}$. Different sensors may therefore produce observation messages from different observation symbol sets, depending on the sensor type. We assume that the observation messages are independent variables, i.e., an observation message will depend on the asset's current state only and not on any previous observation messages. The *sequence* of messages received from sensor k is denoted $Y^k = y_1^k, y_2^k, \ldots$, where $y_t^k \in V^k$ is the observation message received from sensor k at time t. Based on the observation messages, an agent performs real-time risk assessment. The observation messages can be received from several sensors simultaneously, and they may contain conflicting information. As one cannot assume that it is possible to resolve the correct state of the monitored assets at all times, the observation symbols are probabilistic functions of the asset's security state. The asset's true state is *hidden*. This is consistent with the basic idea of HMMs [Rab90].

For each sensor k monitoring an asset, there is an HMM described by the parameter vector $\lambda^k = (\mathbf{P}, \mathbf{Q}^k, \pi)$. The parameter vector for one object is $\lambda = (\mathbf{P}, (k, \mathbf{Q}^k), \pi)$, where $k = 1, \ldots, K$, and K is the number of sensors for this object. $\mathbf{P} = \{p_{ij}\}$ is the state transition probability distribution matrix for an asset, where $p_{ij} = P(x_{t+1} = s_j | x_t = s_i), 1 \leq i, j \leq N$. Hence, p_{ij} represents the probability that the asset will transfer into state s_j next, given that its current state is s_i. $\pi = \{\pi_i\}$ is the initial state distribution for the asset. Hence, $\pi_i = P(x_1 = s_i)$ is the probability that the asset was in state s_i when the risk assessment process started.

For each asset, there are K observation symbol probability distribution matrices, one for each sensor monitoring the asset. The observation symbol probability distribution matrix $\mathbf{Q}^k = \{q_j^k(l)\}$ is a probability distribution for an asset in state s_j over the observation symbols from sensor k, whose elements are $q_j^k(l) = P(y_t^k = v_l^k | x_t = s_j), 1 \leq j \leq N, 1 \leq k \leq K, 1 \leq l \leq M_k$. The element $q_j^k(l)$ in $\mathbf{Q}^k$ represents the probability that sensor k will send the observation symbol v_l^k given that the asset is in state s_j. $\mathbf{Q}^k$ therefore indicates sensor k's false-positive and false-negative effects on the agents risk assessments.

The π vector and the $\mathbf{P}$ matrix describe the initial state and security behavior of an asset, and must be the same for all sensors monitoring the same asset. Since each sensor may produce a unique set of observation symbols, the $\mathbf{Q}$ matrix depends on the sensor k. to describing the relation between the observation symbols and the state of the object.

3.2 Quantitative Risk Assessment

Following the terminology in [Sta04], risk can be measured in terms of *consequences* and *likelihoods*. A consequence is the qualitative or quantitative outcome of an event, and the likelihood is the probability of the event. To perform risk assessment, we use a mapping: $\mathcal{C} : S \to \mathbb{R}$, describing the cost due to loss of confidentiality, integrity and availability associated with each state of an asset. The total risk $\mathcal{R}_t$ for an asset at time t is

$$\mathcal{R}_t = \sum_{i=1}^{N} \mathcal{R}_t(i) = K^{-1} \sum_{k=1}^{K} \sum_{i=1}^{N} \mathcal{C}(i) \gamma_t^k(i) \tag{1}$$

where $\gamma_t^k(i)$ is the (estimated) probability that the asset is in security state s_i at time t, based on observations from sensor k. N is the number of states for the asset, K is the number of sensors, and $\mathcal{C}(i)$ is the cost value associated with state s_i. Here, the sum of the estimates γ_t^k from the sensors are weighted equally by K^{-1}. Ideally, the estimates should be weighted in accordance to the reliability of the sensor data so that estimates from unbiased sensors with low variance will be given higher priority.

The risk value obtained from (1) represents the current asset risk at time t. In order to perform real-time risk assessment, $\mathcal{R}_t$ needs to be regularly updated. For each sensor k the agent computes the asset's current state probability $\gamma_t^k = \{\gamma_t^k(1), \ldots, \gamma_t^k(N)\}$, at each time instant t. Given an observation y_t^k, and the HMM $\lambda^k = (\mathbf{P}, \mathbf{Q}^k, \pi)$, the agent can update the state probability by using Alg. 2. This algorithm relies on a *forward variable*, computed by means of Alg. 3. To simplify the notation the sensor index k has been omitted in these algorithms. For further details on the algorithms, see the Appendix.

```
Algorithm 2 Update state probability distribution $\gamma_t$
Require: $y_t, \alpha_{t-1}, \lambda$ {observation at time $t$, forward variable at time $t-1$,
  the HMM}
Ensure: $\gamma_t$ {the security state probability at time $t$}
  use Alg. 3 to compute the forward variable $\alpha_t$
  for $i = 1$ to $N$ do
    $\gamma_t(i) \leftarrow \frac{\alpha_t(i)}{\sum_{j=1}^{N} \alpha_t(j)}$
  end for
  return $\gamma_t = \{\gamma_t(i)\}$
```

```
Algorithm 3 Compute forward variable $\alpha_t$
Require: $y_t, \alpha_{t-1}, \lambda$ {observation at time $t$, forward variable at time $t-1$,
  the HMM}
Ensure: $\alpha_t$ {the forward variable at time $t$}
  for $i = 1$ to $N$ do
    if $t = 1$ then
      $\alpha_t(i) \leftarrow q_i(y_t)\pi_i$
    else
      $\alpha_t(i) \leftarrow q_i(y_t) \sum_{j=1}^{N} \alpha_{t-1}(j) p_{ji}$
    end if
  end for
  return $\alpha_t = \{\alpha_t(i)\}$
```

3.3 A Continuous Time Approximation

The HMM defined in Section 3.1 is a discrete-time model, inherently not suitable for continuous-time observation data. A model for real-time risk assessment must be able to handle bursts of alerts, as well as silent periods without alerts. Ideally, no alerts should be discarded at any time. To correctly interpret alerts by malicious events as an indication of an ongoing attack, the time interval between subsequent alerts must be considered in the model. To solve this problem we make a continuous-time approximation, similar to [WWT02]. By using a fixed, sufficiently short, time period between events in the discrete-time model, the intervals between observations will be multiples of this period.

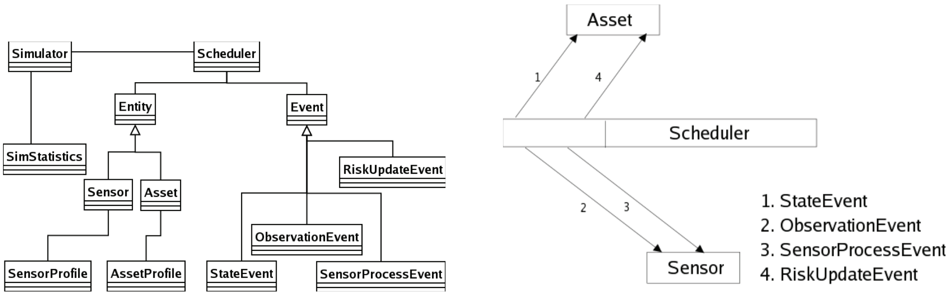
Recall that an agent will process a sequence of discrete-time observation messages Y^k, where $y_t^k \in V^k$ is the observation message received from sensor k at time t. We define the time between two subsequent observation messages as Δ, where Δ is a fixed value. Hence, in a continuous-time context, $p_{ij}(\Delta) = P(x_{t+\Delta} = s_j | x_t = s_i)$ represents the probability that an asset will be in state s_j after an additional time Δ, given that its current state at time t is s_i. For simplicity we let p_{ij} represent $p_{ij}(\Delta)$. In case there are no observation messages during Δ, a "null" message is generated. When two or more observation messages arrive within this time interval, they are placed in a queue and processed at time $t + \Delta, t + 2\Delta, \ldots$. The queue will necessarily introduce a delay in the risk computation. Δ should therefore be small enough so that the agent can handle the alert frequency of the monitored asset in real-time with minimal loss of alerts due to a full queue. The queue size must, however, not be so large that the system looses its ability to assess risk in real-time. As an example, a queue size of 1200 alerts and $\Delta = 1$ second introduces a maximum delay of 20 minutes, which is unacceptable for most applications. On the other hand, the processing capacity of the agent should not be exceeded; it must be able to update the state probability (i.e., execute Alg. 2 and 3) in less than Δ. The selection of a suitable time interval is a configuration issue that depends on the actual implementation.

4. The Simulator

In order to demonstrate and validate the theory in a realistic setting, we implemented a discrete-time, discrete-event simulator. This enabled us to simulate the security events and risk assessment process of large networks over a longer period of time. We refer to the states generated by the simulator as the *true security states* of the assets, whereas the *estimated security state distribution* refers to the state distribution estimated by Alg. 1. Consequently, by applying (1), the *true risk* refers to the risk value computed from the true security state, and the *estimated risk* refers to the risk value computed from the estimated security state distribution. The true and estimated risk of the simulated systems are compared in order to study the validity of the method.

4.1 Simulator Design

The simulator was implemented using the JSIM [Mil] discrete-event simulation framework for Java. JSIM consists of a `Scheduler`, where `Events` are scheduled to be performed on `Entities`. The entities of the risk-assessment simulation are `Assets` (representing hosts) and `Sensors` (representing IDS sensors), and the events are the `StateEvent`, the `SensorProcessEvent`, the `ObservationEvent`, and the `RiskUpdateEvent`. The simulator can be divided into three phases: initialization, execution, and reporting. A class diagram showing an overview of the simulator classes is depicted in Fig. 1(a). Fig. 1(b) depicts the scheduling of the `Events`.



(a) Simulator UML class diagram.

(b) Overview of scheduler, events, and entities.

Figure 1. Simulator design.

During initialization, each `Asset` and `Sensor` is initialized with appropriate HMM model parameters, i.e., $\mathbf{P}$ and π for `Assets` and $\mathbf{Q}^k$ for `Sensors`. For each `Asset`, an initial state $x_1 \in S$ is chosen, according to its initial state probability distribution π. `RiskUpdateEvents` (events that cause an update of the true security state of the `Assets`), `SensorProcessEvents` (events that cause sensors to estimate a new security state distribution by using Alg. 1), and `RiskUpdateEvents` (events that cause `Assets` to update their assessed risk according to (1)), are scheduled for each time interval of the simulation.

At each time t during the execution, the `StateEvents` cause `Assets` to transfer to their next state x_{t+1}, based on their transition probability matrix $\mathbf{P}$. These states are sensed by the `Sensors`, that in turn schedule `ObservationEvents`, representing a `Sensor` observations y_t^k based on the true state and the observation probability matrix $\mathbf{Q}^k$. The `ObservationEvents` cause the `Sensors` to read and queue the observations for further processing. A `SensorProcessEvent` for every sensor is scheduled for each time interval and causes each `Sensor` to process the first `Observation` in its queue and update its state distribution using Alg. 1. Finally, for each time instant t, the `RiskUpdateEvents` cause every `Asset` to update their risk value based on the input from one or more `Sensors`. The current risk value $\mathcal{R}_t$ is computed in accordance to (1) and stored in the `SimStatistics` class. Fig. 1(b) shows the sequence of events acting on the entities (assets and sensors).

The simulation results are stored in the `SimStatistics` object during the simulation and written to file for further analysis when the simulation has executed. The risk values for the `Assets`, as well as the aggregated risk level of the entire network, is stored for for each time instant t. Additional processing, such as correlation analysis, is also performed at this stage.

4.2 Implementation Issues

This section provides a discussion of some design considerations for the risk-assessment simulation implementation.

Observation Message Queues. As discussed in Section 3.3, each `Sensor` must be associated with an observation message queue, in order to handle bursts of alerts without data loss. Whenever a `Sensor` receives an observation message for a particular asset, an observation is put in a queue and processed on a first-come first-serve basis. Only the first observation in the queue is processed by each sensor in each time interval Δ. These mechanisms are implemented in the simulator, but they would be best studied using experimental or real traffic data. The discrete-time simulator described in this paper consequently does not simulate alert burst, and using the method on real sensor data is left for future work.

Null Observations. Most IDS sensors do not provide observations indicating a good state; they only provide warnings and alerts. In this implementation, the risk assessment process therefore produces and intepretets a "null" observation whenever the message queue of a sensor is empty. As will be seen in the simulated example in Section 5, one can usually assume that the null observation indicates a good state.

Profiles. For large networks, estimating initial parameters for all assets and sensors can become very time-consuming. To address this, we implemented the `AssetProfiles` and `SensorProfiles` java classes, which contain sets of HMM parameters that are common to several assets and sensors. As will be seen in Section 5, there can be profiles for different types of hosts (such as webservers, routers, workstations, and laptops), as well as for different types of sensors (such as network and host IDS). For now, the profiles are implemented directly in Java as part of the simulator, but ideally the profiles should be described as part of an overall network model using a suitable language, such as XML.

Scaling. In the actual implementation of Alg. 1 and 2 we used a scaled version of the forward variable: $\bar{\alpha}_t(i) = C_t \alpha_t(i)$, where $C_t = (\sum_{i=1}^{N} \alpha_t(i))^{-1}$. The purpose is to keep the computations within the precision range of the computer. It can be shown that these scaling coefficients cancel out [Rab90, pp. 272].

5. Examples and Simulation Results

The predecessor of this paper [ÅSH+05] included a simple example, which demonstrated how the assessed risk value of an asset varies as an agent receives and processes a predefined observation sequence. To demonstrate the method in a real-world context, we now simulate the risk assessment process of a large network with multiple sensors throughout the network. In order to efficiently manage a high number of hosts, `SensorProfiles` and `AssetProfiles` are defined for the different types of sensors and assets.

The network consists of an Internet gateway (router), two publicly available web-servers on a demilitarized zone (DMZ), two protected file-servers, as well as ten workstations and ten laptops (see Fig. 2). Each host type is described by an `AssetProfile`, as discussed above. The profiles represent different levels of exposure to attacks and compromises, as well as the particular costs associated to the assets' states. For the purpose of this example, we assume that the state space of each asset can be represented by a simple Markov model with the states G (good), A (under attack), and C (compromised). State G means that the asset is up and running securely and that it is not subject to any kind of attack activity. In contrast to [GGPW^{+}01], we assume that assets are always vulnerable to attacks, even in state G. As an attack against an asset is initiated, it will move to security state A. An asset in state A is subject to an ongoing attack, possibly affecting its behavior with regard to security. Finally, an asset enters state C if it has been successfully compromised by an attacker. An asset in state C is assumed to be completely at the mercy of an attacker and subject to any kind of confidentiality, integrity and/or availability breaches.

We assume that the router and file servers are configured to be relatively secure (i.e., the transition probabilities to state C are small), and that the laptops, workstations and web servers are particularly susceptible to attacks (i.e., the transition probabilities to state A are relatively high). All assets, with the exception of the router, are monitored by a network intrusion detection system (NIDS) and a host intrusion detection system (HIDS), as generalized by `SensorProfiles`. The router is only monitored by a NIDS. The observation symbols sets are the same for both the NIDS and the HIDS: $V^{NIDS} = V^{HIDS} = \{\phi, a, c\}$, where symbol a is an indication of state A, c an indication of state C, and ϕ (the "null" observation) an indication of the good state G. In the examples beneath, we differentiate between λ_{gen}, the underlying HMM that generates the true state transitions of an asset and controls the behavior of its sensors, and λ_{est}, the estimated HMM used in the risk assessment procedure. As pointed out in Section 3.3, the choice of an appropriate time interval is essential. For the purpose of this simulation, we use $\Delta = 1$ s.

We present two simulation experiments, based on randomly generated state sequences and corresponding observation messages, according to λ_{gen}. Both simulations have a time-span of 24 hours (86400 s.). The cost value vectors $\mathcal{C} = (\mathcal{C}(G), \mathcal{C}(A), \mathcal{C}(C))$ for the assets are $\mathcal{C}_{router} = (0, 4, 8)$, $\mathcal{C}_{webserver} = (0, 3, 6)$, $\mathcal{C}_{fileserver} = (0, 1, 10)$, $\mathcal{C}_{workstation} = (0, 2, 4)$ and $\mathcal{C}_{laptop} = (0, 1, 2)$, so that the total maximum risk for the network is $\mathcal{R}_t = 100$. The HMM model parameters $\mathbf{P}$, $\mathbf{Q}^k$, and π for the assets and sensors have been assigned manually.

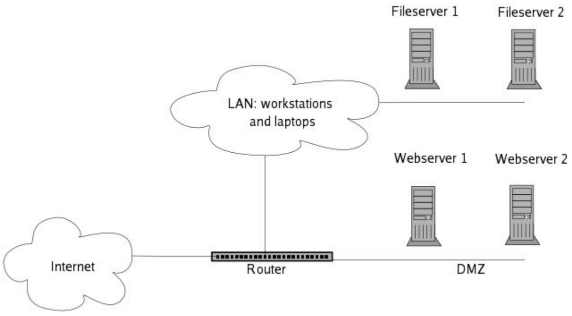


Figure 2. Overview of example network topology.

Algorithms for estimating and learning these parameters are needed, but this is not considered in this paper.

5.1 Example A: Risk Assessment with Known HMM Parameters

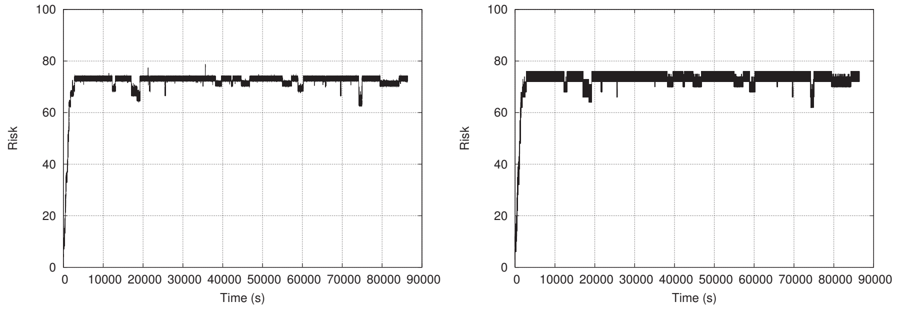
In the first example, $\lambda_{est} = \lambda_{gen}$ for all assets and sensors, i.e., we use the same HMM both for generating state transitions and observations and for assessing the current risk. In other words, the risk-assessment in this example is based on perfect knowledge of the state and observation generation parameters. This is obviously not a realistic scenario, but it allows us to study the performance of the method under optimal circumstances. As an example of the model parameters used in the simulation experiment, the HMM parameters used for the NIDS `SensorProfile` and the router `AssetProfile` are

$$\mathbf{Q}^{NIDS}_{router-gen} = \begin{pmatrix} q_G(\phi) & q_G(a) & q_G(c) \\ q_A(\phi) & q_A(a) & q_A(c) \\ q_C(\phi) & q_C(a) & q_C(c) \end{pmatrix} = \begin{pmatrix} 0.6 & 0.2 & 0.2 \\ 0.2 & 0.5 & 0.3 \\ 0.1 & 0.1 & 0.8 \end{pmatrix},$$

$$\pi_{router-gen} = (\pi_G, \pi_A, \pi_C) = (1, 0, 0),$$

$$\mathbf{P}_{router-gen} = \begin{pmatrix} p_{GG} & p_{GA} & p_{GC} \\ p_{AG} & p_{AA} & p_{AC} \\ p_{CG} & p_{CA} & p_{CC} \end{pmatrix} = \begin{pmatrix} 0.800000 & 0.199995 & 0.000005 \\ 0.700000 & 0.299995 & 0.000005 \\ 0.000005 & 0.000005 & 0.999990 \end{pmatrix}.$$

The laptops, workstations and web servers are likely to get compromised early on during the simulation, whereas the file servers and the router are more resistant to successful attacks. Fig. 3(a) depicts the assessed risk for the network described above, simulated over a period of 24 hours (86400 s.). All hosts are assumed to start in the state G, i.e., $\pi = (1, 0, 0)$ for all assets. Naturally, the development of the network risk varies between simulation executions, as the state generation is probabilistic. Since all assets have a close to absorbing state C, the risk level tends to increase over time, approaching the total maximum risk level.



(a) Assessed risk with perfect knowledge of HMM parameters.

(b) True risk computed from the generating HMM.

Figure 3. Assessed and true risk for Example A.

Based on a comparison between the estimated risk level (see Fig. 3(a)) and the true risk level (see Fig. 3(b)), it is possible to compute the correlation coefficient as a measure of the degree to which the two data sets correlate. Based on 20 simulation runs, the mean correlation coefficient is 0.969 with variance 0.0003 and standard deviation 0.0179. This indicates that the estimation is highly accurate with a high certainty. This is to be expected, as the HMM parameters are known in advance (i.e., $\lambda_{est} = \lambda_{gen}$).

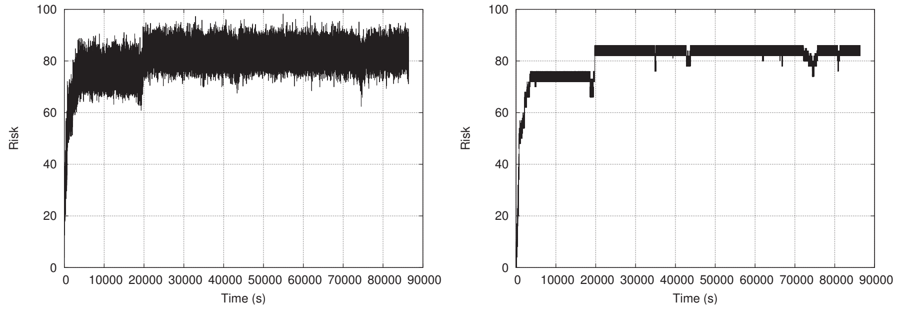
5.2 Example B: Risk Assessment with Estimated HMM Parameters

We now assume that the exact HMM parameters used to generate the state transitions and produce observation messages, λ_{gen}, is unknown, and the HMM parameters for the risk assessment, λ_{est}, have to be estimated. In this way, we can study the systems ability to assess risk under inaccurate estimation parameters, i.e., when $\lambda_{est} \neq \lambda_{gen}$. An example of the estimated parameters is

$$\begin{aligned}
\mathbf{Q}^{NIDS}_{router-est} &= \begin{pmatrix} 0.950 & 0.030 & 0.020 \\ 0.050 & 0.900 & 0.050 \\ 0.020 & 0.020 & 0.960 \end{pmatrix}, \\
\pi_{router-est} &= (0.7, 0.2, 0.1), \\
\mathbf{P}_{router-est} &= \begin{pmatrix} 0.700 & 0.200 & 0.100 \\ 0.500 & 0.450 & 0.050 \\ 0.002 & 0.002 & 0.996 \end{pmatrix}.
\end{aligned}$$

Note that in order to make the results of the two simulation experiments comparable, the parameters used for state generation and for producing observation messages in this example (λ_{gen}) are identical to the ones in the previous example.

Fig. 4(a) shows the assessed risk when using the estimated λ_{est}, and Fig. 4(b) shows the true risk generated during the simulation according to λ_{gen}. Fig. 5(a)-5(b) show the same results, but for a shorter period of time (30 min.). By comparing these graphs, it is possible to see how close the assessed risk value is to the true risk level of the network. Although the estimation parameters in λ_{est} differ from the underlying HMM λ_{gen}, one can conclude from Fig. 4(a)-5(b) that the estimated risk generally follows the true risk. Note that the reason why the estimated risk is higher than the true risk during the first 60 s. of the simulation (Fig.6(a)-6(b)) is the inaccurate estimated initial state distributions π_{est} for the assets. However, as can be seen in Fig. 4(a)-4(b), the total risk value for the network will approach the true risk value over time, regardless of the initial states of the assets.



(a) Assessed risk based on estimated parameters.

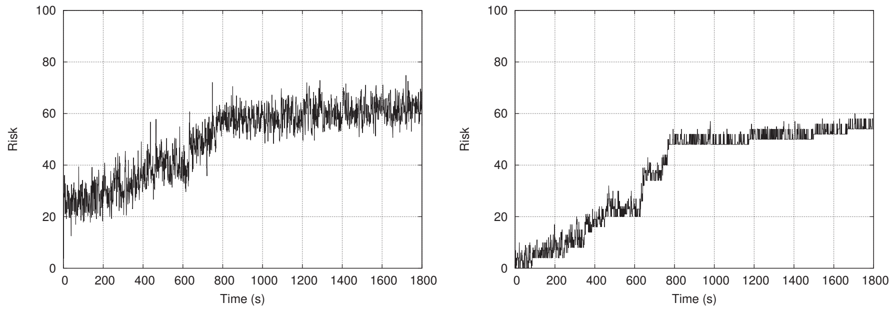
(b) True risk computed from the generating HMM.

Figure 4. Assessed and true risk for Example B (24 h)

Based on 20 simulation runs with the same model parameters, the mean correlation coefficient for the estimated risk value in this example is 0.777, with variance 0.010 and standard deviation 0.102. Compared to the previous example, the correlation coefficient is lower, but it still indicates a high positive correlation. Note that the variance and the standard deviation are higher than in the previous example.

6. Conclusion and Further Work

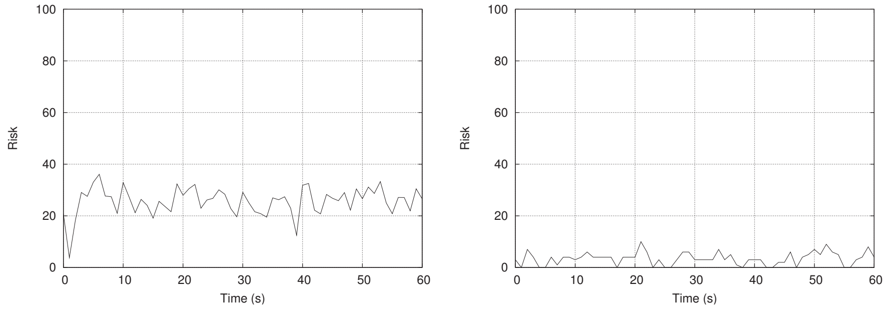
In this paper, we demonstrate how hidden Markov models can be used to perform real time risk assessment of large-scale computer networks. Under the Markov assumption, we model and simulate the risk level of a large number of assets, based on a predefined state transition model with corresponding HMM parameters for each asset and sensor. The simulations indicate that



(a) Assessed risk based on estimated parameters.

(b) True risk computed from the generating HMM.

Figure 5. Assessed and true risk for Example B (30 min)



(a) Assessed risk based on estimated parameters.

(b) True risk computed from the generating HMM.

Figure 6. Assessed and true risk for Example B (60 s)

the method provides insightful results about the security state and the risk level of hosts in a network, even when the estimated model parameters are inaccurate.

Although the initial approach described in [ÅSH$^+$05] has been significantly extended, there are still some open research questions that remain to be solved. A natural extension of this work is to perform analysis based on real network data. The possibility of modeling asset interdependencies must be considered. For the proposed approach to be useful in practice, a method for automated parameter reestimation is needed. Finally, the simulation framework could be extended to simulate different types of threats and attackers, in order to study the performance of the proposed method in a more realistic context.

Acknowledgments

The Centre for Quantifiable Quality of Service in Communication Systems, Centre of Excellence, is appointed by the Research Council of Norway, and funded by the Research Council, NTNU, UNINETT, and Telenor. See `http://www.q2s.ntnu.no/` for more information.

Appendix: Computing the State Distributions

In this appendix we explain the background of Alg. 2 and 2, i.e., how the security state probabilities of an asset can be estimated. Note that the computations are independent of the sensor type, hence, the k index has been omitted from the equations in this appendix.

Recall the sequence of observed messages $Y = y_1, y_2, \ldots$. Given the first observation y_1 and the hidden Markov model $\lambda = (\mathbf{P}, \mathbf{Q}, \pi)$, the initial estimated state distribution $\gamma_1(i)$ can be calculated as

$$\gamma_1(i) = P(x_1 = s_i | y_1, \lambda) = \frac{P(y_1, x_1 = s_i|\lambda)}{P(y_1|\lambda)} = \frac{P(y_1|x_1 = s_i, \lambda)P(x_1 = s_i|\lambda)}{P(y_1|\lambda)}. \tag{A.1}$$

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

$$P(y_1|\lambda) = \sum_{j=1}^{N} P(y_1|x_1 = s_j, \lambda)P(x_1 = s_j|\lambda) = \sum_{j=1}^{N} q_j(y_1)\pi_j. \tag{A.2}$$

Hence, by combining (A.1) and (A.2)

$$\gamma_1(i) \quad = \quad \frac{q_i(y_1)\pi_i}{\sum_{j=1}^{N} q_j(y_1)\pi_j}, \tag{A.3}$$

where $q_j(y_1)$ is the probability of observing symbol y_1 in state s_j, and π is the initial state probability. To simplify the calculation of the state distribution after t observations we use the *forward-variable*

$$\alpha_t(i) = P(y_1 \cdots y_t, x_t = s_i|\lambda), \tag{A.4}$$

as defined in [Rab90]. By using recursion, this variable can be calculated in an efficient way as

$$\alpha_t(i) = \begin{cases} q_i(y_1)\pi_i, & t = 1 \\ q_i(y_t)\sum_{j=1}^{N} \alpha_{t-1}(j)p_{ji}, & t > 1 \end{cases} \tag{A.5}$$

where the initial forward variable $\alpha_1(i)$ was found from (A.1) and (A.3) In the derivation of $\alpha_t(i)$ we assumed that y_t only depend on x_t and that the Markov property holds. Now we can use the forward variable $\alpha_t(i)$ to update the state probability distribution by new observations. This is done by

$$\begin{aligned} \gamma_t(i) = P(x_t = s_i|y_1 \cdots y_t, \lambda) &= \frac{P(y_1 \cdots y_t, x_t = s_i|\lambda)}{P(y_1 \cdots y_t|\lambda)} \\ &= \frac{P(y_1 \cdots y_t, x_t = s_i|\lambda)}{\sum_{j=1}^{N} P(y_1 \cdots y_t, x_t = s_j|\lambda)} = \frac{\alpha_t(i)}{\sum_{j=1}^{N} \alpha_t(j)}. \end{aligned} \tag{A.6}$$

Note that (A.6) is similar to Eq. 27 in [Rab90], with the exception that we do not account for observations that occur after t.

References

[ÅSH$^+$05] André Årnes, Karin Sallhammar, Kjetil Haslum, Tønnes Brekne, Marie Elisabeth Gaup Moe, and 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.

[BGFI$^+$98] J. S. Balasubramaniyan, J. O. Garcia-Fernandez, D. Isacoff, E. Spafford, and D. Zamboni. An architecture for intrusion detection using autonomous agents. In *Proceedings of the 14th Annual Computer Security Applications Conference*, page 13. IEEE Computer Society, 1998.

[DCF05] H. Debar, D. Curry, and B. Feinstein. Intrusion detection message exchange format (IDMEF) – Internet-Draft, 2005.

[GGPW$^+$01] Fengmin Gong, Katerina Goseva-Popstojanova, Feiyi Wang, Rong Wang, Kalyanaraman Vaidyanathan, Kishor Trivedi, and Balamurugan Muthusamy. Characterizing intrusion tolerant systems using a state transition model. In *DARPA Information Survivability Conference and Exposition (DISCEX II)*, volume 2, 2001.

[GK04] Ashish Gehani and Gershon Kedem. Rheostat: Real-time risk management. In *Recent Advances in Intrusion Detection: 7th International Symposium, RAID 2004, Sophia Antipolis, France, September 15-17, 2004. Proceedings*, pages 296–314. Springer, 2004.

[Mil] John A. Miller. Jsim: A java-based simulation and animation environment. `http://chief.cs.uga.edu/~jam/jsim/`.

[OMSH03] Dirk Ourston, Sara Matzner, William Stump, and Bryan Hopkins. Applications of hidden markov models to detecting multi-stage network attacks. In *Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS)*, 2003.

[Rab90] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Readings in speech recognition*, pages 267–296, 1990.

[SBD$^+$91] Steven R. Snapp, James Brentano, Gihan V. Dias, Terrance L. Goan, L. Todd Heberlein, Che lin Ho, Karl N. Levitt, Biswanath Mukherjee, Stephen E. Smaha, Tim Grance, Daniel M. Teal, and Doug Mansur. DIDS (distributed intrusion detection system) - motivation, architecture, and an early prototype. In *Proceedings of the 14th National Computer Security Conference*, pages 167–176, Washington, DC, 1991.

[SCCC$^+$96] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle. GrIDS – A graph-based intrusion detection system for large networks. In *Proceedings of the 19th National Information Systems Security Conference*, 1996.

[SCS03] 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 2003.

[SGF02] Gary Stonebumer, Alice Goguen, and Alexis Feringa. Risk management guide for information technology systems, National Institute of Standards and Technology, special publication 800-30, 2002. http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf.

[Sta04] Standards Australia and Standards New Zealand. AS/NZS 4360: 2004 risk management, 2004.

[VKB01] Giovanni Vigna, Richard A. Kemmerer, and Per Blix. Designing a web of highly-configurable intrusion detection sensors. In *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID 2000)*, pages 69–84, London, UK, 2001. Springer-Verlag.

[VVKK04] Fredrik Valeur, Giovanni Vigna, Christopher Kruegel, and Richard A. Kemmerer. A comprehensive approach to intrusion detection alert correlation. *IEEE Transactions on Dependable and Secure Computing*, 1(3):146–169, 2004.

[WFP99] C. Warrender, S. Forrest, and B. Pearlmutter. Detecting intrusions using system calls: Alternative data models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, 1999.

[WWT02] Wei Wei, Bing Wang, and Don Towsley. Continuous-time hidden markov models for network performance evaluation. *Performance Evaluation*, 49:129–146, 2002.

Paper C

Multisensor Real-time Risk Assessment using Continuous-time Hidden Markov Models

Kjetil Haslum, André Årnes

In Proceedings of the 2006 International Conference on Computational Intelligence and Security (CIS'06).

Guangzhou, China. November 3-6, 2006

MULTISENSOR REAL-TIME RISK ASSESSMENT USING CONTINUOUS-TIME HIDDEN MARKOV MODELS

Kjetil Haslum, and André Årnes
Center for Quantifiable Quality of Service in Communication Systems[*]
Norwegian University of Science and Technology
O.S. Bragstads plass 2E, N-7491 Trondheim, Norway
{haslum,andrearn}@q2s.ntnu.no

Abstract The use of tools for monitoring the security state of assets in a network is an essential part of network management. Traditional risk assessment methodologies provide a framework for manually determining the risks of assets, and intrusion detection systems can provide alerts regarding security incidents, but these approaches do not provide a real-time high level overview of the risk level of assets. In this paper we further extend a previously proposed real-time risk assessment method to facilitate more flexible modeling with support for a wide range of sensors. Specifically, the paper develops a method for handling continuous-time sensor data and for determining a weighted aggregate of multisensor input.

1. Introduction

With the complexity of technologies in todays society, we are exposed to an increasing amount of unknown vulnerabilities and threats. For a system or network administrator, it is vital to have access to automated systems for identifying risks and threats and for prioritizing security incidents. In this paper we study and extend a previously proposed system for real-time risk assessment. The proposed system computes a quantitative risk measure for all assets based on input from sensors such as network-based intrusion detection systems (IDS). The approach was first proposed in [ÅSH$^+$05], and it has been validated using simulations in [ÅSHK06] and real-life data in [ÅVVK06]. During this work, several open research issues have been identified. There is a need for more flexible security state modeling, and the wide range of potential sensor types require different modeling schemes. In particular, a typical

[*] The Centre for Quantifiable Quality of Service in Communication Systems, Centre of Excellence, is appointed by the Research Council of Norway, and funded by the Research Council, NTNU, UNINETT, and Telenor.

signature-based IDS can be much better modeled using a continuous-time hidden Markov model (HMM) than the discrete-time HMM in [ÅSH$^+$05].

The contributions of this paper consist of a method for continuous-time estimation using transition rates rather than transition probabilities, as well as a method for computing risk as a weighted sum of sensor input, taking into consideration the fact that some sensors are statistically more reliable and significant than others.

In Section 2 we revisit the proposed risk assessment approach and provide explanations of the necessary terminology. In Section 3 and 4 we present various ways of HMM modeling for a flexible real-time risk assessment system, with particular focus on continuous-time HMMs and the aggregation of input from multiple sensors. In Section 5 we discuss the results and provide directions for further work.

2. Real-time Risk Assessment

Risk assessment is typically a manual analysis process based on standardized frameworks, such as those recommended by NIST [SGF02] and AS/NZS [Sta04]. Such methodologies are suitable for evaluating threats and vulnerabilities, but they are not designed to support operational network management. A notable exception is the real-time risk assessment system presented in [GK04], which introduces a formal model for real-time characterization of the risk faced by a host. In [ÅSH$^+$05], we presented another real-rime risk assessment system employing HMMs. An HMM enables the estimation of a *hidden* state based on *observations* that are not necessarily accurate. An important feature of this approach is that it is able to model the probability of false positives and false negatives associated with the observations. The method is based on Rabiner's work on HMMs [Rab90]. This section reviews the model presented in [ÅSH$^+$05]. Some adaptations have been introduced for the purpose of this paper.

The target of the risk assessment is a generic computer network, consisting of *assets*. Unknown factors in such a network may represent *vulnerabilities* that in turn can be *exploited* by a malicious attacker or computer program, causing *unwanted incidents*. The potential exploitation of a vulnerability can be described as *threats* to the assets. The *risk* of the network is evaluated as the probability and consequence of unwanted incidents. The consequences of an unwanted incident is referred to as the *cost* of the incident. As in [ÅSH$^+$05], we assume a multiagent system architecture consisting of *agents* and *sensors*. A *sensor* typically refers to an IDS, but it could be any information-gathering program or device capable of collecting security relevant data, such as logging systems, virus detectors, honeypots, and network sniffers using sampling or filtering. The main task of a sensor is to gather information about the security state of assets and to send standardized observation messages to the agents. An *agent* is responsible for performing real-time risk assessment based on data collected from a number of sensors. The multiagent architecture has been chosen for its flexibility and scalability, in order to support future applications, such as distributed automated response.

Assume that the security of an asset can be modeled by N states, denoted $S = \{s_1, \ldots, s_N\}$. Due to security incidents such as attack attempts and compromises, the security state of an asset will change over time. The sequence of states visited is denoted $X = x_1, \ldots, x_T$, where $x_t \in S$ is the state visited at time t. As in [ÅSH$^+$05], we assume that the state space can be represented by a fully connected Markov model with the states G (good), A (under attack), and C (compromised), i.e., $S = \{G, A, C\}$, as shown in Fig. 1. State G means that the asset is up and running securely and that it is not subject to any kind of attack activity. As an attack against an asset is initiated, it will move to security state A. An asset in state A is subject to an ongoing attack, possibly affecting its behavior with regard to security. Finally, an asset enters state C if it has been successfully compromised by an attacker. It is then assumed to be completely at the mercy of an attacker and subject to any kind of confidentiality, integrity, and/or availability breaches. The risk-assessment method is general and independent of the specific states used. Two alternative ways of modeling the security states of assets are presented in Fig. 2(a) and 2(b). In Fig. 2(a) we show how an asset can be represented by three separate Markov models indicating the security state with respect to *confidentiality*, *integrity*, and *availability*. In Fig. 2(b) we show a left-right model, where the asset can only transfer to a more serious state, with C as an absorbing state.

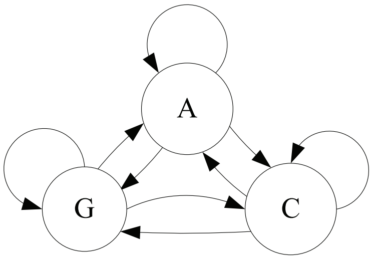
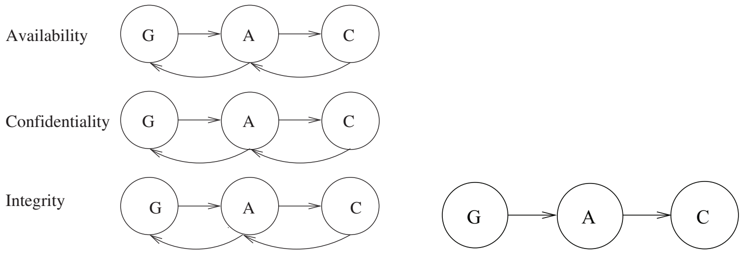


Figure 1. Fully connected Markov model.

The risk *observation messages* are provided by the K sensors monitoring an asset, indexed by $k \in \{1, \ldots, K\}$. An observation message from sensor k can consist of any of the symbols in the observation symbol set $V^k = \{v_1^k, \ldots, v_M^k\}$. Different sensor types may produce observation messages from different observation symbol sets. We assume that the observation messages are independent, i.e., an observation message will depend on the asset's current state only and not on any previous observation messages. The *sequence* of messages received from sensor k is denoted $Y_t^k = y_1^k, \ldots, y_t^k$, where $y_t^k \in V^k$ is the observation message received from sensor k at time t. For the purpose of this paper, we



(a) A risk model consisting of tree submodels

(b) A pure birth process

Figure 2. Alternative security state models.

assume an observation symbol set $V^k = \{g^k, a^k, c^k\}, \forall k$, corresponding to the states in $S = \{G, A, C\}$. Based on the observation messages, an agent performs real-time risk assessment. As one cannot assume that it is possible to resolve the correct state of the monitored assets at all times, the observation symbols are probabilistic functions of the asset's security state. The asset's true state is *hidden*, consistent with the basic idea of HMM [Rab90].

For each sensor k monitoring an asset, there is an HMM described by the parameter vector $\lambda^k = (\mathbf{P}, \mathbf{Q}^k, \pi)$. The parameter vector for one object is $\lambda = (\mathbf{P}, (k, \mathbf{Q}^k), \pi)$, where $k = 1, \ldots, K$, and K is the number of sensors for this object. $\mathbf{P} = \{p_{ij}\}$ is the state transition probability distribution matrix for an asset, where $p_{ij} = P(x_{t+1} = s_j | x_t = s_i), 1 \leq i, j \leq N$. Hence, p_{ij} represents the probability that the asset will transfer into state s_j next, given that its current state is s_i. $\pi = \{\pi_i\}_{i \in S}$ is the initial state distribution for the asset. Hence, $\pi_i = P(x_1 = s_i)$ is the probability that s_i was the initial state of an asset.

For each asset, there are K observation symbol probability distribution matrices, one for each sensor. Each row i in the observation symbol probability distribution matrix $\mathbf{Q}^k = \{q_i^k(m)\}$ is a probability distribution for an asset in state s_i over the observation symbols from sensor k, whose elements are $q_i^k(m) = P(y_t^k = v_m^k | x_t = s_i), 1 \leq i \leq N, 1 \leq k \leq K, 1 \leq m \leq M$. The element $q_i^k(m)$ in $\mathbf{Q}^k$ represents the probability that sensor k will send the observation symbol v_m^k at time t, given that the asset is in state s_i at time t. $\mathbf{Q}^k$ therefore indicates sensor k's false-positive and false-negative effects on the agents risk assessments.

The π vector and the $\mathbf{P}$ matrix describe the initial state and the security behavior of an asset, and they must be the same for all sensors monitoring the same asset. Since each sensor may produce a unique set of observation symbols, the $\mathbf{Q}^k$ matrix depends on the sensor k. to describing the relation between the observation symbols and the state of the object.For each sensor the agent updates the probability distribution $\gamma_t^k = \{\gamma_t^k(i)\}$, where $\gamma_t^k(i) = P(x_t = s_i | Y_t^k)$, by using the method presented in [ÅSH$^+$05]. In [ÅSH$^+$05], the risk of an asset was then evaluated as $\mathcal{R}_t^k = \sum_{i=1}^N \gamma_t^k(i)\mathcal{C}(s_i)$, where t is the time of the evaluation, k is the sensor used, and $\mathcal{C}(s_i)$ describing the cost due to loss of confidentiality, integrity, and availability for each state of an asset. In Section 4 we present a new method for multisensor assessment using a weighted sum of the results from multiple sensors.

3. Continuous-time Markov Chains

There is a multitude of sensors that can provide security relevant information, such as IDS, network logs, network traffic measurements, virus detectors, etc. In our previous work, we have only considered the use of discrete-time HMMs, but we have seen the need for continuous-time HMMs allowing for transition rates rather than probabilities. The two HMM types complement each other, and they are suitable for different types of sensors. Let us consider some example sensor types. A signature based IDS matches network traffic (network IDS) or host activity (host IDS) with signatures of known attacks and generates alerts. Virus detection systems use a similar technique. The alert stream of a signature based IDS is typically highly varying, and a continuous time HMM approach is preferable. An active measurement systems can be used to perform periodical measurements of the availability of hosts and services, for example based on delay measurements. Such a measurement system is an example of an active sensor suitable for a discrete-time HMM that is updated periodically. An anomaly based IDS uses statistical analysis to identify deviation from a behavior that is presumed to be normal. Such a sensor could be used with either a continuous- or a discrete- time model. If the sensor is used to produce alerts in case of detected anomalies, it can be used in a fashion similar to the signature based sensors. If the sensor is used to compute a measure of the normality of a network or system, it can be used as a basis for periodic computations using a discrete time model.

We assume that a continuous-time Markov chain $(x(t), t \geq 0)$ can be used to model the security of an asset. The model consists of the set of states $S = \{s_1, \ldots, s_N\}$, the initial state distribution π, and a transition rate matrix $\mathbf{\Lambda} = \{\lambda_{ij}\}$, $1 \leq i, j \leq N$. When the system is in state s_i, it will make λ_{ij} transitions to state s_j per time unit. The time spent in state s_i is exponentially distributed with mean u_i^{-1} (sojourn time), where $u_i = \sum_{j \neq i} \lambda_{ij}$ is the total rate out of state s_i. The rate in and out of a state must be equal and therefore $\sum_j \lambda_{ij} = 0$, where $\lambda_{ii} = -u_i$ represent the rate of transitions into state s_i. The new HMM for sensor k, based on the transition rates, is then $\lambda^k = (\mathbf{\Lambda}, \mathbf{Q}^k, \pi)$.

The time between observations is not constant, so for each new observation, a transition probability matrix $\mathbf{P}(\Delta_t) = \{p_{ij}(\Delta_t)\}$ have to be calculated, where Δ_t is the time since last observation was received. Suppose that the process $x(t)$ is in state s_i at time t, then the probability that the process is in state s_j at time $t + \Delta_t$ is given by $p_{ij}(\Delta_t) = P(x(t+\Delta_t) = s_j | x(t) = s_i)$. If the transition probability from state s_i to s_j is independent of t, the process is said to be a homogeneous Markov process. The transitions probability matrix $\mathbf{P}(\Delta_t)$ can be calculated by

$$\mathbf{P}(\Delta_t) = e^{\mathbf{\Lambda}\Delta_t},$$

and approximated by

$$\mathbf{P}(\Delta_t) \approx \lim_{n\to\infty} \left(\mathbf{I} + \mathbf{\Lambda}\frac{t}{n} \right)^n . \tag{1}$$

More details on computing the transition probability matrix can be found in [Ros03], pages 388 – 389.

Example 1.. Consider a network with continuous-time sensors monitoring a central server. Through a manual risk assessment process, the administrators have estimated the initial state distribution and the transition rates for the system per day. Given a set of states $S = \{G, A, C\}$, the transition rate matrix is set to

$$\mathbf{\Lambda} = \begin{pmatrix} \lambda_{GG} & \lambda_{GA} & \lambda_{GC} \\ \lambda_{AG} & \lambda_{AA} & \lambda_{AC} \\ \lambda_{CG} & \lambda_{CA} & \lambda_{CC} \end{pmatrix} = \begin{pmatrix} -1.1 & 1.0 & 0.1 \\ 4 & -5 & 1 \\ 3 & 1 & -4 \end{pmatrix}.$$

As noted above, the values indicate the transition rate per day. However, the numbers in the diagonal of the matrix is the rate into the state, which is equal to the sum of the rates out of the state. The first row represents the rates in and out of state G, indicating that the rate of transitions to state A (1 transition per day) is greater than the rate of transitions to state C (0.1 transitions per day). The bottom row of the matrix represents state C, and it indicates that the most probable development is a return to state G due to a successful repair.

First, we calculate the rate at which the system leaves each state

$$\begin{aligned} u_G &= \lambda_{GA} + \lambda_{GC} = 1 + 0.1 = 1.1 = -\lambda_{GG}, \\ u_A &= \lambda_{AG} + \lambda_{AC} = 4 + 1 = 5 = -\lambda_{AA}, \\ u_C &= \lambda_{CG} + \lambda_{CA} = 3 + 1 = 4 = -\lambda_{CC}. \end{aligned}$$

From this we can calculate the sojourn time for each state

$$u_G^{-1} = \frac{10}{11},\ u_A^{-1} = \frac{1}{5},\ u_C^{-1} = \frac{1}{4}.$$

If observations are received at $t_0, t_1, t_2, t_3 = 0, 0.01, 0.11, 0.13$, we have to calculate the time between successive observations $\Delta_l = t_l - t_{l-1}$. This gives

$$\Delta_1, \Delta_2, \Delta_3 = 0.01, 0.1, 0.02.$$

If we apply Equation 1 for computing the transition probabilities, using $n = 2^{10} = 1024$ in the approximation, we get the following transition matrix

$$\mathbf{P}(\Delta_1) = \mathbf{P}(0.01) = \begin{pmatrix} 0.9893 & 0.0097 & 0.0010 \\ 0.0390 & 0.9515 & 0.0096 \\ 0.0294 & 0.0097 & 0.9609 \end{pmatrix},$$

$$\mathbf{P}(\Delta_2) = \mathbf{P}(0.1) = \begin{pmatrix} 0.9133 & 0.0752 & 0.0114 \\ 0.3102 & 0.6239 & 0.0659 \\ 0.2497 & 0.0752 & 0.6750 \end{pmatrix},$$

$$\mathbf{P}(\Delta_3) = \mathbf{P}(0.02) = \begin{pmatrix} 0.9791 & 0.0188 & 0.0021 \\ 0.0759 & 0.9058 & 0.0184 \\ 0.0578 & 0.0188 & 0.9234 \end{pmatrix}.$$

We see from the matrices above that the probability of transferring to another state increases as the period between observations Δ increases. For the special case $\Delta = 0$, the probability of staying in the same state would be 1. Furthermore, we can see from the matrices that the rows sums to 1, as expected for a probability distribution. The computations were performed in Matlab. Only 10 matrix multiplications were necessary in order to compute a matrix to the power of 1024.

4. Multisensor Quantitative Risk Assessment

Following the terminology in [Sta04], risk can be measured in terms of *consequences* and *likelihoods*. A consequence is the qualitative or quantitative outcome of an event, and the likelihood is the probability of the event. To perform risk assessment, we need a mapping: $\mathcal{C} : S \rightarrow \mathbb{R}$, describing the cost due to loss of confidentiality, integrity, and availability for each state of an asset.

The risk $\mathcal{R}_t = E[\mathcal{C}(x_t)]$ is the expected cost at time t, and it is a function of the hidden state x_t of an asset. The only information available about x_t is the distribution γ_t estimated by the HMM. The risk $\mathcal{R}_t^k$ estimated by sensor k is based on the observations Y_t^k from sensor k

$$\mathcal{R}_t^k = E[\mathcal{C}(x_t)|Y_t^k] = \sum_{i=1}^{N} \gamma_t^k(i)\mathcal{C}(s_i),$$

and the estimated variance $\sigma_t^2(k)$ of $\mathcal{R}_t^k$ is

$$\sigma_t^2(k) = Var[\mathcal{R}_t^k] = \sum_{i=1}^{N} \gamma_t^k(i)(\mathcal{C}(s_i) - \mathcal{R}_t^k)^2.$$

A new estimate of the risk $\mathcal{R}_t^0$ based on observations from all the K sensors, is formed by taking a weighted sum of the estimated risk from each sensor. Assuming the estimated risk from each sensor to be unbiased and independent random variables, we can then use the inverse of the variance as weights to get an unbiased minimum variance estimator of the risk. This can be shown by applying the Lagrange multiplier method, see Appendix 5.

$$\begin{aligned}\mathcal{R}_t^0 &= E[\mathcal{C}(x_t)|Y_t^1, Y_t^2, \ldots, Y_t^K] \\ &= \frac{\sum_{k=1}^K (\sigma_t^2(k))^{-1} \mathcal{R}_t^k}{\sum_{k=1}^K (\sigma_t^2(k))^{-1}},\end{aligned} \tag{2}$$

and the variance $\sigma_t^2(0)$ of $\mathcal{R}_t^0$ can be estimated as follows

$$\sigma_t^2(0) = Var[\mathcal{R}_t^0] = \frac{1}{\sum_{k=1}^K \frac{1}{\sigma_t^2(k)}}. \tag{3}$$

A derivation of equation 3 is shown in Appendix 5.

Example 2.. Consider the same network as in Example 1. Assume that the server is monitored by two different sensors with the following states and cost values

$$\begin{aligned}S &= \{G, A, C\}, \\ \mathcal{C} &= (\mathcal{C}(G), \mathcal{C}(A), \mathcal{C}(C)) = (0, 5, 20).\end{aligned}$$

At time t, assume that the two HMMs of the two sensors have the following estimated state distributions

$$\begin{aligned}\gamma_t^1 &= (0.90, 0.09, 0.01), \\ \gamma_t^2 &= (0.70, 0.20, 0.10).\end{aligned}$$

We are interested in finding an estimator for the risk of the monitored asset based on the input from the two sensors. As this estimator should have as little variance as possible, we wish to give more weight to the sensor with the best estimate, i.e., the sensor with the least variance. The weight is computed as the inverse of the variance from the two sensors. We compute the mean and variance of the risk from each sensor

$$\begin{aligned}
\mathcal{R}_t^1 &= 0.9 \times 0 + 0.09 \times 5 + 0.01 \times 20 = 0.650, \\
\mathcal{R}_t^2 &= 0.7 \times 0 + 0.2 \times 5 + 0.1 \times 20 = 3.000, \\
\sigma_t^2(1) &= 0.9(0-0.65)^2 + 0.09(5-0.65)^2 \\
&\quad + 0.01(20-0.65)^2 = 5.826, \\
\sigma_t^2(2) &= 0.7(0-3)^2 + 0.2(5-3)^2 + 0.1(20-3)^2 \\
&= 36.00.
\end{aligned}$$

We now combine the risk from each sensor to get a minimum variance estimate of the risk

$$\begin{aligned}
\mathcal{R}^0 &= \frac{\frac{1}{5.8275}0.65 + \frac{1}{36}3}{\frac{1}{5.8275} + \frac{1}{36}} && = 0.977, \\
\sigma_t^2(0) &= \frac{1}{\frac{1}{5.8275} + \frac{1}{36}} && = 5.016.
\end{aligned}$$

We see that the mean for the weighted risk is close to the mean for sensor 1. This is intuitive, as sensor 1 has the least variance. We can also see that the variance of the weighted risk is smaller than that of the individual sensors.

5. Conclusions and Further Work

We have addressed several issues to improve the proposed method for real-time risk assessment. The rate-based assessment is proposed as an alternative for some common sensors, and the weighted multisensor risk assessment method provides a mechanism for integrating sensors with varying accuracy and reliability into the system. The mechanisms proposed in this paper should be implemented and tested using real-life data and simulations, as previously done in [ÅVVK06]. Another issue that still remains is the problem of parameter estimation and learning. It is possible to set the model parameters using expert knowledge, but this is a cumbersome process, and it would be preferable to automate the process of estimating and learning the parameters.

Appendix: Minimum Variance Estimator

Assume that we have K independent random variables $(x_k,\ k = 1, \ldots, K)$ with the same mean μ, and variance $Var[x_k] = \sigma_k^2$. A new random variable $x = \sum_{k=1}^K a_k x_k$ is constructed from $(x_k\ k = 1, \ldots, K)$, this new random variable should be unbiased $E[x] = \mu$ and have minimum variance

$$Var[x] = Var[\sum_{k=1}^K a_k x_k] = \sum_{k=1}^K a_k^2 Var[x_k] = \sum_{k=1}^K a_k^2 \sigma_k^2,$$

$$E[x] = E[\sum_{k=1}^{K} a_k x_k] = \sum_{k=1}^{K} a_k \mu = \mu \Rightarrow \sum_{k=1}^{K} a_k = 1$$

To find the optimal weights ($\bar{a}_k,\ k = 1, \ldots, K$) we apply the Lagrange multiplier method to to minimise the performance index $f(a_1, a_2, \ldots, a_K) = \sum_{k=1}^{K} a_k^2 \sigma_k^2$, under the restriction $g(a_1, a_2, \ldots, a_K) = \sum_{k=1}^{K} a_k - 1 = 0$. This is done by solving the equation $\nabla f = \lambda \nabla g$, where ∇f denotes the gradient of f. This is equivalent to the following sets of partial differential equations

$$\frac{\partial}{\partial_{a_k}} [f + \lambda g]_{a_k = \bar{a}_k} = 0,\ (k = 1, \ldots, K),$$

$$\frac{\partial}{\partial_{a_k}} \left[\sum_{l=1}^{K} a_l^2 \sigma_l^2 + \lambda (\sum_{l=1}^{K} a_l - 1) \right]_{a_k = \bar{a}_k} = 0,\ (k = 1, \ldots, K). \qquad \text{(A.1)}$$

When we take the derivatives we end up with the following set of lineare equations $2\bar{a}_k \sigma_k^2 + \lambda = 0$, with the solution $\bar{a}_k = -\frac{\lambda}{2\sigma_k^2}$, and $\lambda = \frac{-2}{\sum_{k=1}^{K} \frac{1}{\sigma_k^2}}$. This gives the optimal weights

$$\bar{a}_k = \frac{\frac{1}{\sigma_k^2}}{\sum_{k=1}^{K} \frac{1}{\sigma_k^2}}$$

and the minimum variance

$$Var[x] = \sum_{k=1}^{K} \left(\frac{\frac{1}{\sigma_k^2}}{\sum_{k=1}^{K} \frac{1}{\sigma_k^2}} \right)^2 \sigma_k^2 = \frac{1}{\sum_{k=1}^{K} \frac{1}{\sigma_k^2}}.$$

References

[ÅSH$^+$05] André Årnes, Karin Sallhammar, Kjetil Haslum, Tønnes Brekne, Marie Elisabeth Gaup Moe, and 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.

[ÅSHK06] André Årnes, Karin Sallhammar, Kjetil Haslum, and Svein Johan Knapskog. Real-time risk assessment with network sensors and hidden markov model. In *Proceedings of the 11th Nordic Workshop on Secure IT-systems (NORDSEC 2006)*, Oct 2006.

[ÅVVK06] André Årnes, Fredrik Valeur, Giovanni Vigna, and Richard A. Kemmerer. Using hidden markov models to evaluate the risk of intrusions. In *Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection, RAID 2006, Hamburg, Germany, September 20 – 22, 2006.*, September 2006.

[GK04] Ashish Gehani and Gershon Kedem. Rheostat: Real-time risk management. In *Recent Advances in Intrusion Detection: 7th International Symposium, RAID 2004, Sophia Antipolis, France, September 15-17, 2004. Proceedings*, pages 296–314. Springer, 2004.

[Rab90] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Readings in speech recognition*, pages 267–296, 1990.

[Ros03] Sheldon M. Ross. *Introduction to Probability Models.* Academic Press, New York, 8th edition, 2003.

[SGF02] Gary Stonebumer, Alice Goguen, and Alexis Feringa. Risk management guide for information technology systems, National Institute of Standards and Technology, special publication 800-30, 2002. http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf.

[Sta04] Standards Australia and Standards New Zealand. AS/NZS 4360: 2004 risk management, 2004.

Paper D

DIPS: A Framework for Distributed Intrusion Prediction and Prevention Using Hidden Markov Models and Online Fuzzy Risk Assessment

Kjetil Haslum, Ajith Abraham, and Svein Knapskog

In Proceedings of the Third International Symposium on
Information Assurance and Security

Manchester, United Kingdom. August 29-31, 2007

DIPS: A FRAMEWORK FOR DISTRIBUTED INTRUSION PREDICTION AND PREVENTION USING HIDDEN MARKOV MODELS AND ONLINE FUZZY RISK ASSESSMENT

Kjetil Haslum, Ajith Abraham, and Svein Knapskog
{haslum, ajith.abraham, knapskog}@q2s.ntnu.no
Centre for Quantifiable Quality of Service in Communication Systems
Norwegian University of Science and Technology
O.S. Bragstads plass 2E, N-7491 Trondheim, Norway

Abstract This paper proposes a Distributed Intrusion Prevention System (DIPS), which consists of several IPS over a large network (s), all of which communicate with each other or with a central server, that facilitates advanced network monitoring. A Hidden Markov Model is proposed for sensing intrusions in a distributed environment and to make a one step ahead prediction against possible serious intrusions. DIPS is activated based on the predicted threat level and risk assessment of the protected assets. Intrusions attempts are blocked based on (1) a serious attack that has already occurred (2) rate of packet flow (3) prediction of possible serious intrusions and (4) online risk assessment of the assets possibly available to the intruder. The focus of this paper is on the distributed monitoring of intrusion attempts, the one step ahead prediction of such attempts and online risk assessment using fuzzy inference systems. Preliminary experiment results indicate that the proposed framework is efficient for real time distributed intrusion monitoring and prevention.

1. Introduction

Firewalls are employed only at the network perimeter and they are not always effective against intrusion attempts. The average firewall is designed to filter detect and deny clearly suspicious traffic. Many attacks, intentional or otherwise, are launched from within an organisation. Intrusion detection systems may be effective at detecting suspicious activity, but do not provide protection against attacks. In Distributed IDS (DIDS), conventional intrusion detection system are embedded inside intelligent agents and are deployed over a network. In a distributed environment, IDS agents communicate with each other, or with a central server. By having these co-operative agents distributed across a network, incident analysts, network managers and security personnel are able to get a broader view of what is occurring on their network as a whole.

Distributed monitoring allows early detection of planned and coordinated attacks, thereby allowing the network managers to take preventive measures. In a DIDS, it is important to ensure that the individual IDS is light-weight and accurate. In the DIPS framework, code fragments developed using genetic programming models are embedded inside intelligent agents (IDS) to detect various types of attacks [AGMV07]. Individual IDS sensor node outputs are provided as inputs to the Hidden Markov Model (HMM).

The rest of the paper is organised as follows. Section 4 introduces key concepts of distributed intrusion prevention systems and the technical requirements to design such systems in practice. Section 3 deals with HMM followed by some experimental results in Section 4. Online risk assessment by fuzzy inference system is presented in Section 5 and some conclusions are provided towards the end.

2. Intrusion Prevention Systems (IPS)

Intrusion prevention systems are proactive defence mechanisms designed to detect malicious packets within normal network traffic, block the offending traffic automatically before it does any damage. Like IDS, IPS may be also classified as Host based IPS or Network based IPS. There are a number of challenges to the implementation of an IPS device in addition to those to be faced when deploying passive-mode IDS products. These challenges all stem from the fact that the IPS device is designed to work in-line, presenting a potential choke point and single point of failure. Some of these problems could be eliminated in a distributed intrusion prevention system, where there is no single point of control and the problems are tackled at its source of origin as much as possible. The main task of the IPS is to block a suspect traffic flow as soon as possible by immediately discarding suspect information packets. The suspicious traffic may also be re-routed for further forensic analysis etc. An IPS should have a maximum up time since it has the potential to close a vital network path and thus, inadvertently, cause a DoS condition. An IPS should be computationally light since it is essential that its impact on overall network performance is minimal and also achieve high packet processing rates. An IPS should minimize false positives since this can lead to a Denial of Service condition. IPS should be able to decide exactly which malicious traffic is blocked, provide a mechanism for alerts and have forensic analysis capabilities.

2.1 Distributed Intrusion Prevention Systems (DIPS)

DIPS are simply a superset of the conventional IPS implemented in a distributed environment. We consider IPS as an integrated IDS with many more functions as listed in Section 4. Due to the distributed nature of IPS, the implementation poses several challenges. IDS are embedded inside software mobile agents and placed in the network to be monitored. The individual IDS

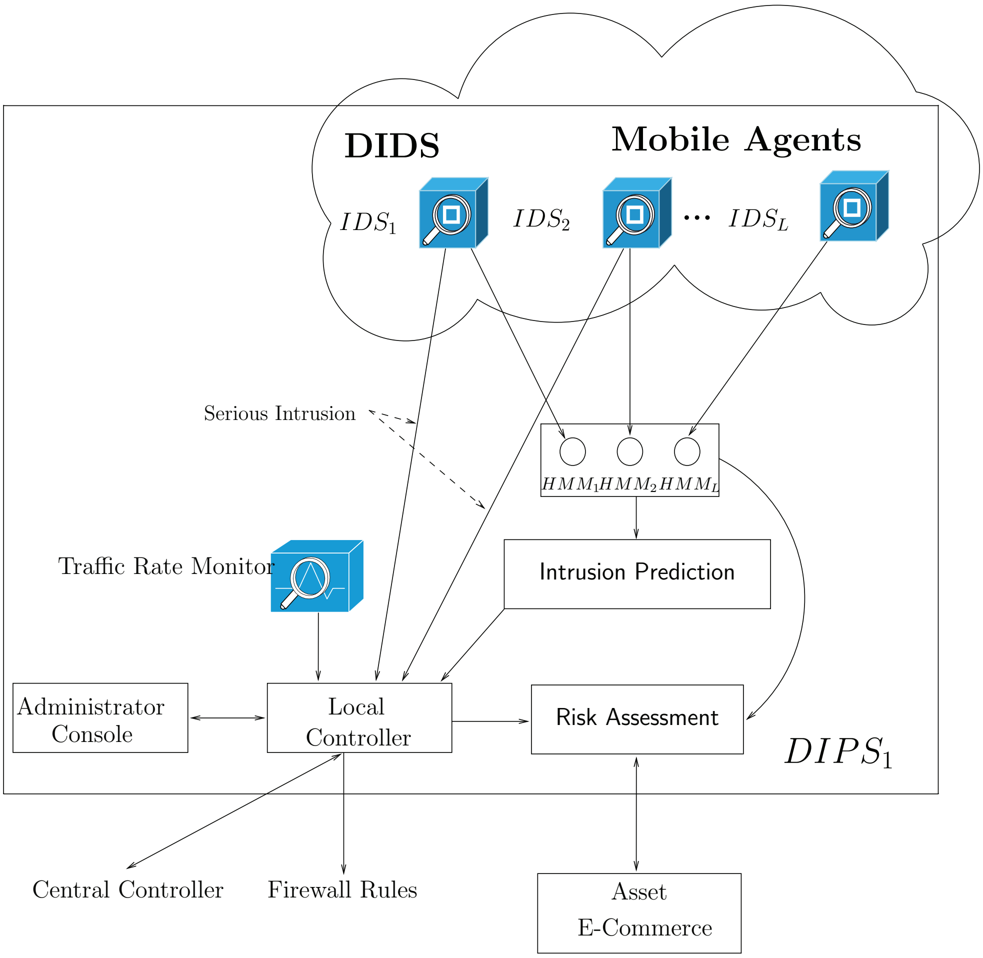


Figure 1. Architecture of a DIPS element.

may be configured to detect a single attack, or they may detect several types of attacks.

Figure 1 illustrates the basic architecture of a DIPS element, which is controlled by a local controller. In a large network, each DIPS element communicates/coordinates with other DIPS local controller and/or a central controller [AJTH07]. A HMM model processes the attack data information from the various mobile agent IDS sensors. IDS deployed are capable of detecting simple problems to serious denial of service type of attacks. Based on the nature of the detected attack, the following actions would be taken:

1 If the detected attack is simply a port scan or a probe, the HMM model will attempt to make a prediction of a possible future attack based on the current distributed attack pattern. Based on this prediction, the

central controller (or administrator) would take precautionary measures to prevent future attacks. The central controller would also make use of an online risk assessment of the assets subjected to this possible serious attack in the future.

2 If the detected attack is very serious, the central controller would take necessary actions to re-configure firewall rules or notify the administrator etc. Such serious attacks would bypass the HMM model.

3 At any time any abnormal traffic rate is noted by the monitor, then again the central controller would take necessary actions to re-configure firewall rules or notify the administrator etc.

In the DIPS framework, each network component may host one or many IDS. Since there will be a large number of flag generators, these must be extracted, summarised, analyzed, and condensed by a suitable architecture before arriving at a final conclusion. Very often, it is to be noted that the event information, which is detected by the IDS agents will follow a bottom up approach for analysis and the various command and control flows will follow a top-down approach. The physical location of IDS agents may be fixed or mobile so as to monitor certain parts of the network segments.

The co-operative intelligent agent network is one of the most important components of the DIDS [AJTH07]. Ideally these agents will be located on separate network segments, and very often geographically separated. Communication among the agents is done utilizing TCP/IP sockets. Agent modules running on host machines are capable of data analysis and to formulate adequate response actions and are very often implemented as read only and fragile. In the event of tampering or modification the agent reports to the server agent and automatically ends its life. Agents residing in the individual analyzer/controllers consist of modules responsible for agent regeneration, dispatch, updating and maintaining intrusion signatures and so on. These agents control the individual IDS agents for monitoring the network, manage all the communication and life cycle of the IDS agents and also updates the IDS agents with detection algorithms, response and trace mechanisms

3. Hidden Markov Modeling of DIPS

Gao et al. [GSW03] developed an HMM to predict attacks in the application layer and they claimed that the approach could be extended for network layer. Årnes et al. [ÅSH+05] used HMMs for real time risk assessment, but not directly for attack prediction as we proposed in this paper.

A HMM can be described as two stochastic processes; the hidden process $(x_t; t = 1, 2, \ldots)$ that representing the state of the system, and the observable process $(y_t; t = 1, 2, \ldots)$ representing the observations made by an IDS Agent. There will be no direct relation between the t index and time. t will be a sequence number for observations received from the IDS agents. The HMM model used in this paper is described as follows:

- A set of states $S = \{s_1, s_2, \ldots, s_N\}$ describing the possible states of the system. To simplify the notation of equations and algorithms we will use i instead of s_i. In this paper only four states are used; *Normal(N)* indicating no suspicious activity, *Intrusion Attempt (IA)* indicating suspicious activity against the network, e.g. probing, *Intrusion in Progress (IP)* indicating that one or more attacker have started an attack against the system, and *Successful Attack (SA)* indicating that one or more attacker have broken into the system. The state space used in this paper is simlar to the statespace used in [KL06].

- A set of observations $V = \{v_1, v_2, \ldots, v_M\}$. To simplify the notation we will use k instead of v_k. For this paper, we assume, that each IDS Agent only produces three different types of observations; *No suspicious activity (N)*, *Probing (P)* indicating suspicious activity against the network, and *Successful Attack (SA)* indicating that an IDS Agent have detected a successful attack.

- An initial distribution vector $\pi = \{\pi_i\},\ \pi_i = P(x_1 = i)$ describing the state of the system when the monitoring starts. We assume the system to be in the N-state when monitoring starts.

- A transition probability matrix $P = \{p_{ij}\},\ p_{ij} = P(x_t = j|x_{t-1} = i)$, describing the dynamics of the interaction between the intruder and the system.

- An observation probability matrix for each of the L IDS Agents $Q^l_k = \{q^l_i(k)\},\ q^l_i(k) = P(y^l_t = k|x_t = i)$, describing the quality or the trustworthiness of each IDS Agent.

The HMM model used in this paper models only integrity and confidentiality, and make no attempts to model availability. We believe that availability is best modeled separately. The Markov Model used in this paper is shown in Figure 1. States drawn as circles indicate secure states, and state drawn by a square indicates that damage has already happened. When using a discrete HMM to model the system, we can make the following assumptions; all information about the system is contained in the state of the system, observations are independent given the current state, and state occupation times are geometrically distributed.

Assume that we have a sequence of observations from each IDS agent $Y_t = \{Y^l_t\}_{l=1,\ldots,L}$, where $Y^l_t = y^l_1, \ldots, y^l_t$, and a model λ. For each of the L IDS Agents the probability of being in each of the N states is calculated only based on observations made by the corresponding IDS Agent $\gamma^l_t(i) = P(x_t = i|Y^l_t, \lambda)$. The computations required for updating the probability distribution γ^l_t can be found as Eq. 19 and Eq. 27 in [Rab90].

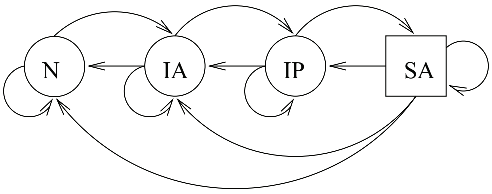


Figure 2. A Markov model modeling the security of a small network

The initial distribution π is used to initialize γ before the system starts to process observations from the IDS agents $\gamma_0^l = \pi, l = 1, \ldots, L$.

When an IDS Agent l detects suspicious activity in the network, it sends an observation y_t^l to the HMM, that updates γ_t^l.

After γ is updated the probability of being attacked (PA) is calculated based on the probability of being in the *IP* state. The PA can take on one of the three values; *Low*, *Medium* and *High*. A message with the current PA is sent to the Central Controller, to be presented for the administrator through the *Administrator Console* and for updating the *Intrusion frequency* as described in Section 5.

4. Experimental Results Using HMM

In order to demonstrate how HMMs can be used in DIPS, we have constructed a model of a small network as illustrated in Figure 3 and run some simulations to illustrate the proposed model. The example network consists of four different assets; a router, a public web server, a file server, and a database. Five IDS Agents denoted $IDS_1, \ldots, IDS_5$ are deployed in the network, and the observations are sent to the their corresponding HMM. We have generated a sequence of 32 observations for each of the five IDS agents $Y_1, \cdots, Y_5$. Figure 4 shows (from top to bottom); the hidden state X, observation sent from IDS Agent 1, and the probability of being in state Intrusion in Progress estimated by HMM 1 based on the observations from IDS Agent 1.

In the experiments, we have used four different states and three different observation symbols as described in Section 3. This is illustrated in Figure 1. For this illustration, we also assume that all the IDS Agents send the observations at fixed time intervals to the corresponding HMM. Even if there are no attacks or suspicious activities, the N observations will be sent.

The system is assumed to be in the *Normal* state when the IPS starts. This corresponds to the following initial distribution $\pi = (1, 0, 0, 0)$. We have used a supervised training method to estimate the transition probability matrix P and the observation probability matrix Q. By supervised training, we mean that the hidden state X, is used in the estimation of P and Q. The P matrix is estimated by counting the number of transitions, and Q is estimated by counting the number of emitted symbols for each state. All observations from

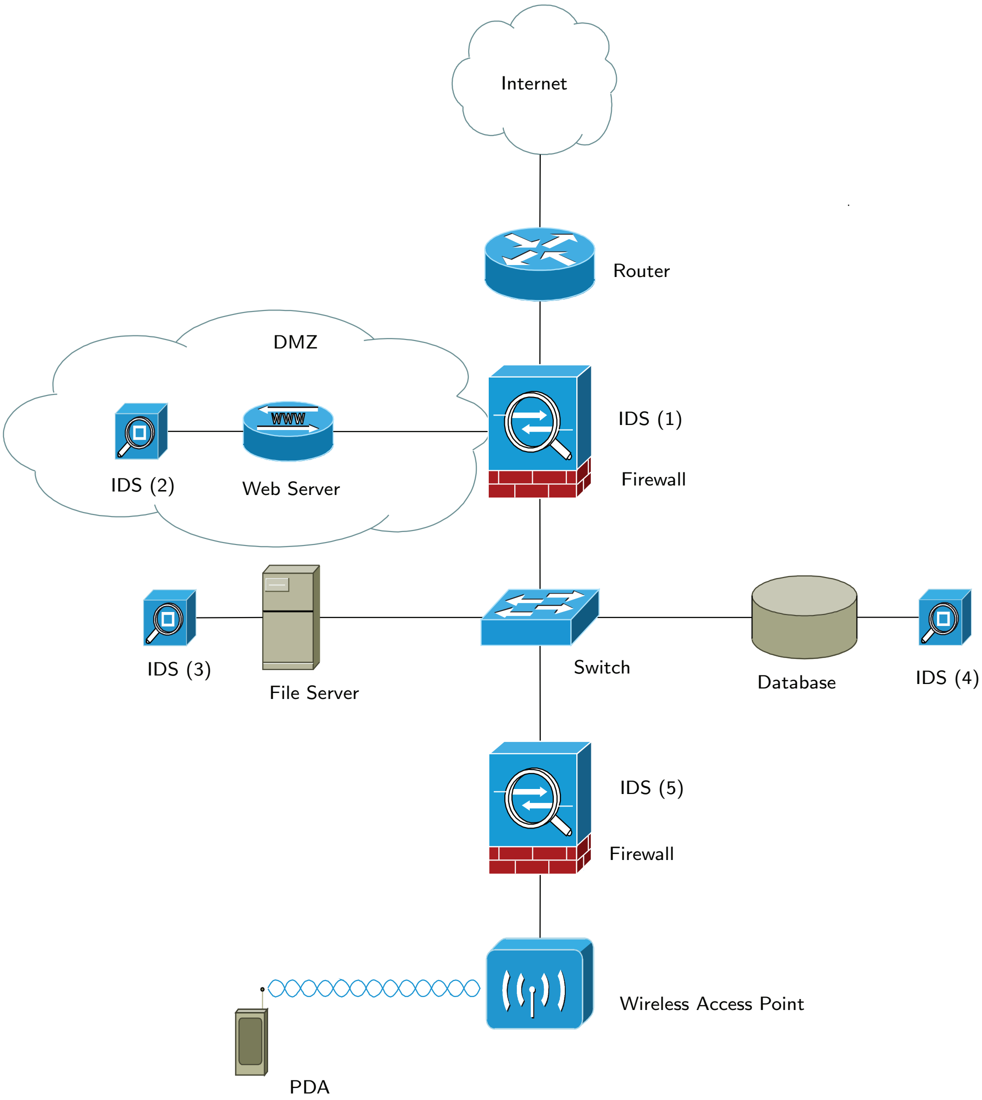


Figure 3. Example network showing assets and IDS agents

the five sensors were used to estimate one common Q, used in all the five HMMs. Events corresponds to the time when observations are received from the IDS agents. We assume that all observations are received at the same time, to make the figures more readable.

The upper graph in Figure 4 shows the hidden state used for the parameter estimation. The graph in the middle illustrates the output from the first IDS Agent, and the lower graph depicts the probability of being in the IP-state.

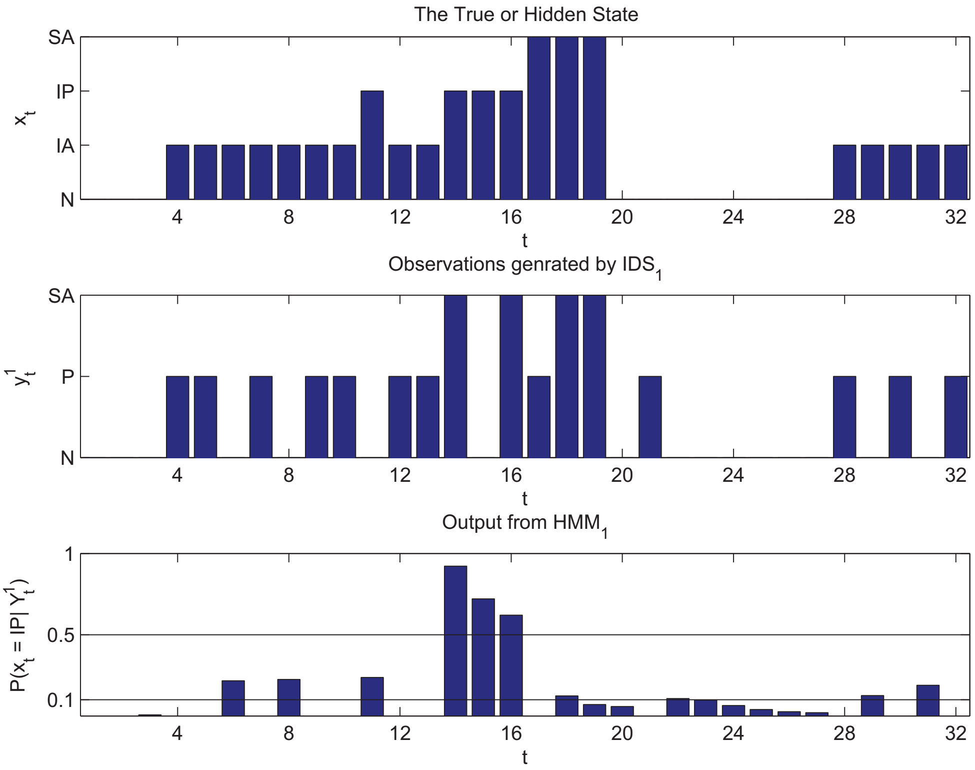
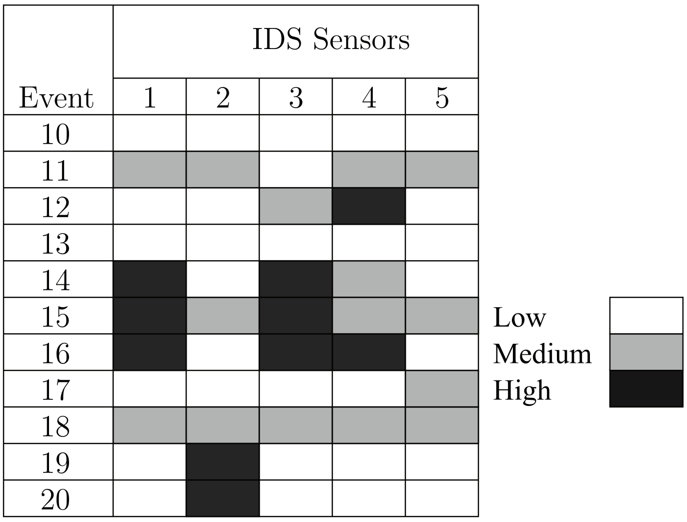


Figure 4. The hidden state, observations from the IDS and output from the HMM.

PA_l is estimated for each of the L IDS Agent based on the probability $\gamma_t^l(IP) = P(x_t = IP|y_t^l, \lambda_l)$ when the corresponding HMM is updated. The probability levels used to determine the PA_l is 0.1 and 0.5, also shown in Figure 4.

Figure 5 shows the AP from the five IDS Agents, which depicts how the output from the IPS may be visualized by the system administrator. It is observed that three of the five IDS agents reported higher risk of being attacked at $t = 16$, which is just one event before the first attack at $t = 17$.

High PA may lead to automated response like updating the fire wall rules or removal of users from this system. This kind of automated response should probably not always be based only on results from the HMM, but preferably also include some kind of automated forensic analysis based on traffic- and log-data stored in a database. High risk may trigger extended logging.



	IDS Sensors				
Event	1	2	3	4	5
10	□	□	□	□	□
11	▨	▨	□	▨	▨
12	□	□	▨	■	□
13	□	□	□	□	□
14	■	□	■	▨	□
15	■	▨	■	▨	▨
16	■	□	■	■	□
17	□	□	□	□	▨
18	▨	▨	▨	▨	▨
19	□	■	□	□	□
20	□	■	□	□	□

Low	□
Medium	▨
High	■

Figure 5. Estimated probability of being attacked, based on observations from the five IDS Agents.

5. Modeling Risk Assessment Using Hierarchical Fuzzy Inference System

Risk analysis is fundamentally all about establishing probabilities. In the DIPS framework, we model the risk analysis using *threat levels*, *vulnerability* and *asset value* [Jon06]. We consider that all components within a network scenario falls into one of these categories, and each has attributes, or derived factors, that contribute positively or negatively to risk.

Threat level is modeled as the frequency of attacks/intrusions, obtained from HMM predictions as described in Section 3, the probability that an intruder is being successful in overcoming protective controls and gains access to act against the organization or assets and the type and severity of attacks.

Vulnerability may be defined as the probability that an asset will be unable to resist the actions of an intruder. Vulnerability exists when this probability exceeds a given threshold. This may be because of weaknesses in software or hardware, missing software patches and so on. Vulnerability may be modeled as contemporary high threat capability and low system threat resistance.

Asset may be defined as any data, device, or other component of the environment that supports information-related activities, and which can be affected in a manner that result in loss. To determine asset loss could be one of the hardest tasks of analyzing risk. It is very difficult to put a precise value on the various types of assets, and there may be more than one value or liability characteristic. Complex relationships might exist between the different forms of loss and many factors determine loss magnitude. We model asset value/loss as cost, criticality, sensitivity and recovery.

The overall architecture for asset risk management is summarised in Figure 6.

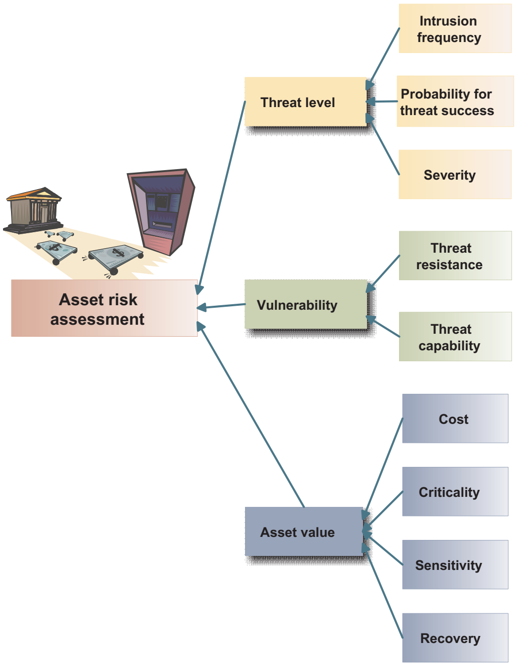


Figure 6. Generic structure of the risk assessment model.

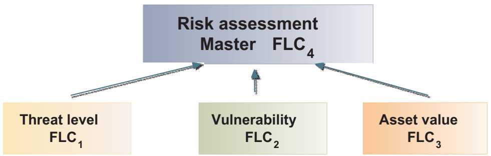
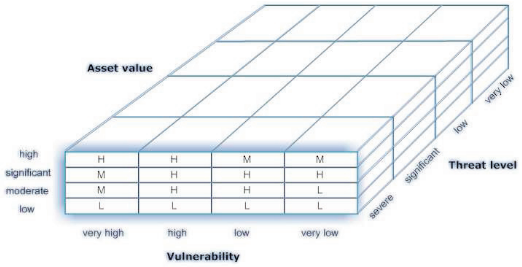


Figure 7. Hierarchical architecture of four fuzzy logic controllers.

5.1 Hierarchical Fuzzy Modeling of Risk Assessment

Most of the uncertainties in the risk assessment models are handled using statistical approaches. However, such methods cannot handle sources of imprecision that therefore may lead to uncertainty including scarce or incomplete data, measurement error, data obtained from expert judgment, or subjective interpretation of available information. In this paper we propose the use of fuzzy set theory to incorporate uncertainties into online risk assessment for



	very high	high	low	very low
high	H	H	M	M
significant	M	H	H	H
moderate	M	H	H	L
low	L	L	L	L

Figure 8. Fuzzy associative memory structure for FLC_4.

DIPS. Based on the form of available information, fuzzy set theory, probability theory, or a combination of both can be used to incorporate the uncertainty and variability of risk variables into various risk assessment models.

Zadeh [Zad65] introduced the concept of fuzzy logic to present vagueness in linguistics, and further implement and express human knowledge and inference capability in a natural way. We used an hierarchical fuzzy logic controller to asses the overall risk based on *threat level*, *vulnerability* and *asset value*. A fuzzy logic controller (FLC) is composed of a knowledge base, a fuzzification interface, an inference system and a defuzzification interface. The architecture of the hierarchical fuzzy logic controller for risk assessment is depicted in Figure 7. A FLC is assigned to make the inference from each of the three input variables *threat level*, *vulnerability* and *asset value* and a fourth FLC is assigned to make the overall inference for risk assessment. The Mamdani inference method [MA75] was used for all the four FLC's.

5.2 Fuzzy Modeling of Threat Level

Threat level is modeled as (1) frequency of attacks/intrusions (2) probability that intruder being successful in overcoming protective controls and (3) Type and severity of attack. A Fuzzy logic controller (FLC_1) observes three input variables and produce one output variable. Three triangular membership functions are assigned per input variable and three triangular membership functions are used for the output variable.

5.3 Fuzzy Modeling of Vulnerability

Vulnerability is modeled as (1) threat capability and (2) system threat resistance. Three triangular membership functions are assigned per input variable and three triangular membership functions are used for the output variable. A Fuzzy logic controller (FLC_2) observes two input variables and produces the output variable.

5.4 Fuzzy Modeling of Asset Value and Loss

Asset value/loss is modelled using four variables: (1) cost (2) criticality (3) sensitivity and (4) recovery. To minimize the number of rules only two triangular membership functions are assigned per input variable and four triangular membership functions are used for the output variable. Fuzzy logic controller (FLC_3) observes four input variables and produce one output variable.

5.5 Fuzzy Modeling of Risk

Figure 8 illustrates the fuzzy associate memory map linking *threat level*, *vulnerability* and *asset value* for overall risk assessment. Four membership functions are used to represent each of the three input variables. Triangular membership values are used as membership functions. A fuzzy *if-then* rule may be formulated as follows:

IF *threat level* is **SIGNIFICANT** and *vulnerability* is **VERY HIGH** and *asset value* is **HIGH** ***THEN*** *Risk* is **HIGH**.

All the 9 input variable values and the output variable (risk assessment) are scaled between 0-1. The fuzzy *if-then* rules were formulated based on expert knowledge of the network model and associated risks.

6. Conclusions

This paper proposes a distributed intrusion prevention system, which is activated based on the predicted threat level and risk assessment of the protected assets. We focus on the distributed monitoring of intrusion attempts, one step ahead prediction of such attempts, and online risk assessment using fuzzy inference systems. Preliminary experimental results indicate that the proposed framework is efficient for real time distributed intrusion monitoring and prevention.

Future work may include parameter estimation based on real data, better HMM models, more states, continuous models, Kalman filtering and integration with mobile agents in a real network. We also intend to use supervised learning schemes to optimise the quantity and quality of the fuzzy *if-then* rules to improve the online computational performance etc.

References

[AGMV07] A. Abraham, C. Grosan, and C. Martin-Vide. Evolutionary design of intrusion detection programs. *International Journal of Network Security*, 4(3):328–339, 2007.

[AJTH07] A. Abraham, R. Jain, J. Thomas, and S.Y. Han. D-scids: Distributed soft computing intrusion detection systems. *Journal of Network and Computer Applications, Elsevier Science*, 30(1):81–98, 2007.

[ÅSH+05] André Årnes, Karin Sallhammar, Kjetil Haslum, Tønnes Brekne, Marie Elisabeth Gaup Moe, and 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.

[GSW03] Fei Gao, Jizhou Sun, and Zunce Wei. The prediction role of hidden markov model in intrusion detection. In *IEEE CCECE 2003: Canadian Conference on Electrical and Computer Engineering*, volume 2, pages 893–896, 2003.

[Jon06] Jack Jones. An introduction to factor analysis of information risk (fair). *Norwich Journal of Information Assurance*, 2(1):67, 2006.

[KL06] Rahul Khanna and Huaping Liu. System approach to intrusion detection using hidden markov model. In *IWCMC '06: Proceeding of the 2006 international conference on Communications and mobile computing*, pages 349–354, New York, NY, USA, 2006. ACM Press.

[MA75] E.H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7(1):1–13, 1975.

[Rab90] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Readings in speech recognition*, pages 267–296, 1990.

[Zad65] L.A. Zadeh. Fuzzy sets. *Info. & Ctl.*, 8:338–353, 1965.

Paper E

Fuzzy Online Risk Assessment for Distributed Intrusion Prediction and Prevention Systems

Kjetil Haslum, Ajith Abraham, and Svein Knapskog

In Proceedings of EUROSIM/UKSIM.

Cambridge, England. April 1-3, 2008

FUZZY ONLINE RISK ASSESSMENT FOR DISTRIBUTED INTRUSION PREDICTION AND PREVENTION SYSTEMS

Kjetil Haslum, Ajith Abraham, and Svein Knapskog
Centre for Quantifiable Quality of Service in Communication Systems
Norwegian University of Science and Technology
O.S. Bragstads plass 2E, N-7491 Trondheim, Norway
{haslum, ajith.abraham, knapskog}@q2s.ntnu.no

Abstract A Distributed Intrusion Prediction and Prevention Systems (DIPPS) not only detects and prevents possible intrusions but also possesses the capability to predict possible intrusions in a distributed network. Based on the DIPS sensors, instead of merely preventing the attackers or blocking traffic, we propose a fuzzy logic based online risk assessment scheme. The key idea of DIPPS is to protect the network(s) linked to assets, which are considered to be very risky. To implement DIPPS we used a Distributed Intrusion Detection System (DIDS) with extended real time traffic surveillance and online risk assessment. To model and predict the next step of an attacker, we used a Hidden Markov Model (HMM) that captures the interaction between the attacker and the network. The interaction between various DIDS and integration of their output are achieved through a HMM. The novelty of this paper is the detailed development of Fuzzy Logic Controllers to estimate the various risk(s) that are dependent on several other variables based on the inputs from HMM modules and the DIDS agents. To develop the fuzzy risk expert system, *if-then* fuzzy rules were formulated based on interviews with security experts and network administrators. Preliminary results indicate that such a system is very practical for protecting assets which are prone to attacks or misuse, i.e. highly at risk.

1. Modelling of DIPPS

1.1 Introduction

Intrusion Prevention Systems (IPS) are proactive defense mechanisms designed to detect malicious packets embedded in normal network traffic and stop intrusions dead, blocking the offending traffic automatically before it does any damage rather than simply raising an alert as, or after, the malicious payload has been delivered. There are a number of challenges for the implementation of an IPS device that does not come across when deploying passive-mode Intrusion Detection System (IDS) products. These challenges all stem from the fact that the IPS device is designed to work in-line, presenting a potential choke point and single point of failure. Some of these problems could be eliminated in a distributed intrusion prevention system, where there is not a single point of control and the problems are tackled as close to its source of origin as possible. The main task of the IPS is to discard all suspect packets immediately and block the offending traffic flow as soon as possible. The suspicious traffic may be re-routed to honeynets or honeypots for further forensic analysis etc. An IPS should have a maximum up time since it has the potential to close a vital network path and thus, once again, causing a Self Denial of Service (SDoS) condition. IPS should be computationally light and also achieve high packet processing rates since it is essential that its impact on overall network performance is minimal. The IPS should minimize false positives since this can lead to a SDoS. The IPS should be able to decide exactly which malicious traffic is blocked and also provides a mechanism for alerts and forensic analysis capabilities. Rest of the article is organized as follows. Section 2 introduces DIPPS followed by HMM in Section 3. Fuzzy modeling is illustrated in Section 4 and experiment results are given in Section 5 followed by some Conclusions.

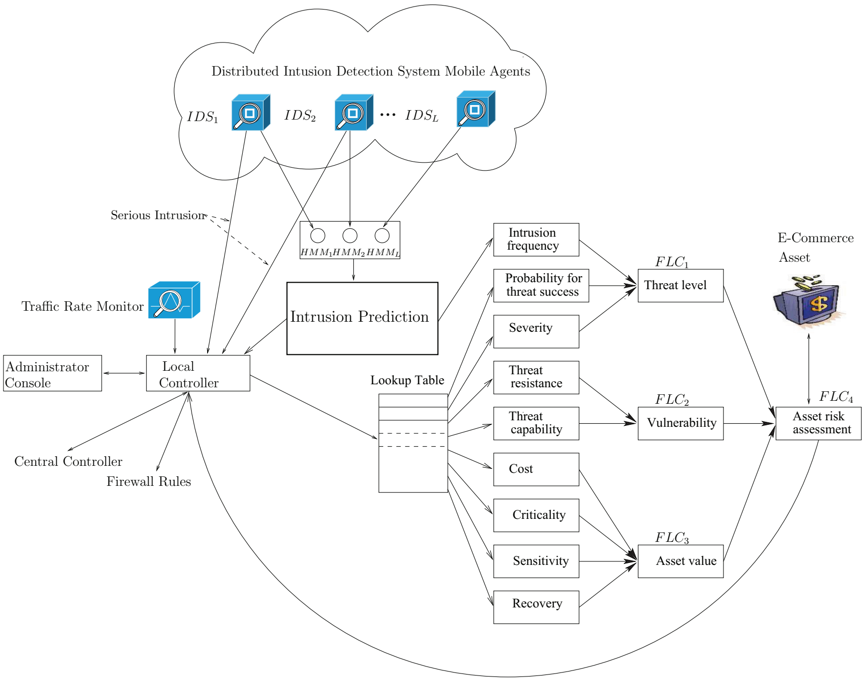


Figure 1. Architecture of a DIPPS element.

2. Distributed Intrusion Prediction and Prevention Systems (DIPPS)

DIPS are simply a superset of the conventional IPS implemented in a distributed environment. We consider IPS as an integrated IDS with many additional functions as listed in Section 1.1. Due to the distributed nature of IPS, the implementation poses several challenges. The IDSs are embedded inside software mobile agents and placed in the network to be monitored. An individual IDS may be configured to detect a single attack, or it may detect several types of attacks. Figure 1 illustrates the basic architecture of a DIPPS element, which is controlled by a local controller. In a large network, each DIPPS element communicates/coordinates with other DIPPS local controllers and/or a central controller. The HMM model processes the attack data information from the various mobile agent IDS sensors. IDS deployed are capable of detecting simple problems as well as serious denial of service type of attacks. Based on the nature of the detected attack, the following actions would be taken:

1 If the detected attack is simply a port scan or a probe, the HMM model attempts to make a prediction of a possible future attack based on the current distributed attack patterns. Based on this prediction, the central controller (or administrator) would take precautionary measures to prevent future attacks. The central controller would also make use of an online risk assessment of the assets subjected to this possible serious attack in the future.

2 If the detected attack is very serious, the central controller would take necessary actions to re-configure firewall rules or notify the administrator etc. Such serious attacks would bypass the HMM model.

3 At any time any abnormal traffic rate is noted by the monitor if a predetermined level is reached, the central controller may take necessary actions to re-configure firewall rules or notify the administrator etc.

In the DIPPS framework, each network component may host one or more IDS located in a distributed network. Since there will be a large number of flag generators, these must be abstracted, analyzed, and condensed by a suitable architecture before arriving at a final conclusion. Very often, it is to be noted that the event information, which is detected by the IDS agents will follow a bottom up approach for analysis and the various command and control flow will follow a top-down approach. The physical location of IDS agents may be fixed or mobile so as to monitor certain parts of the network segments.

The co-operative intelligent agent network is one of the most important components of the DIDS [AJTH07]. Ideally these agents will be located on separate network segments, and geographically separated. Communication among the agents is done utilizing TCP/IP sockets. Agent modules running on host machines are capable of data analysis and to formulate adequate response actions and may be implemented as read only and fragile. In the event of tampering or modification the agent reports to the server agent and automatically ends its life. Agents residing in the individual analyzer/controllers consist of modules responsible for agent regeneration, dispatch, updating and maintaining intrusion signatures and so on. These agents control the individual IDS agents for monitoring the network, manage all the communication and life cycle of the IDS agents and also update the IDS agents with detection algorithms as well as response and trace mechanisms.

3. Hidden Markov Model (HMM)

We model the interaction between the attackers and the system by a Markov model, and we assume the system to be in one of the following states; Normal (N) indicating that there is no ongoing suspicious activity, Intrusion Attempt (IA) indicating suspicious activity against the network, Intrusion in Progress (IP) indicating that one or more attacker have started an attack against the system, and Successfull Attack (SA) one or more attackers have already broken into the system. By using a Markov model, we assume that next state transition only depend on current state, this is known as the Markov assumption. To describe an IDS Agent we extend the Markov model to a Hidden Markov Model (HMM), by assuming that the alarms produced by the HMM Agent only depend on the state of the system. The word hidden indicates that the state of the system is not possible to observe, but only observations (output from the IDS Agents) that depend on the system state. Observations from the IDS Agents are used to estimate the system state distribution. One HMM model is used for each IDS Agent, and the state estimation is updated for each new observation from the IDS Agent. The state distribution is further used to estimate the intrusion frequency. The use of HMM to model the interaction between an attacker and a system is based on [HAK07, KL06], and [MVT02] explain how to model the interacton between an attacker and a system using a Markov model.

4. Why Fuzzy Modeling?

If the problem to be solved can be described mathematically and there exist techniques to solve the problem by using reasonable computational power and time, this method should be preferred. But for some real world problems no solution is known at all, and for these problems heuristic techniques may be the only practical solution. An heuristic method is not guaranteed to give the best solution, but often gives a satisfying solution. One way to make a heuristic solution is to use previous experience and some general rules, this is a very natural approach for humans.

Risk assessment is often done by human experts, because there is no exact and mathematical solution to the problem. Usually the human reasoning and perception process cannot be expressed precisely. Different people have different opinions about risk and the association of its dependent variables, and fuzzy logic provides an excellent framework to model this. The key idea is to capture knowledge or information from risk managers and security experts and to embed this vital knowledge in the form of *if-then* rules in a fuzzy inference system to automate the risk assessment.

4.1 Fuzzy Modeling of Risk

The difference between an ordinary crisp set and a fuzzy set is that elements of a fuzzy set have a degree of membership. An element $(x,\ \mu_A(x))$ of a fuzzy set A is therefore a pair where $\mu_A(x)$ is a membership function and represents the degree of membership for x in A. The x value is called a crisp input, to indicate that it is a number. The membership functions for intersections and unions of fuzzy sets are normally constructed using the T-norm and the T-conorm operators. The most frequently used T-norm operator is $T_{\min}(a,b) = \min(a,b)$ and the most frequently used T-conorm operator is $T_{\max}(a,b) = \max(a,b)$.

The first step in the fuzzy inference system is to fuzzify the inputs, that is using the membership functions to calculate the degree of membership in different fuzzy sets. Next step is to apply *if-then* rules. For a Mamdani fuzzy system the *if-then* rules are of the form

$$if\ x\ is\ A\ and\ y\ is\ B\ then\ z\ =\ C \tag{1}$$

where A, B, C are fuzzy sets, and the first part (between if and then) is called the antecedent and the last part (after then) is called the consequent. Usually the T-norm and T-conorm operators are used in the evaluation of the antecedents and consequences respectively. After the *if-then* rules have been applied the crisp output is calculated through a process called defuzzification. But the most widely used defuzzificaton technique is possibly the centroid of an area:

$$Z_{COA} = \frac{\int_Z \mu_A(z) z dz}{\int_Z \mu_A(z) dz} \tag{2}$$

A unit consisting of fuzzification, rule evaluation and defuzzification is called a Fuzzy logic controller (FLC). In this paper we use a hierarchical structure where output from one FLC is used as input to another FLC.

For the risk assessment, nine basic linguistic variables are used that are processed using three Fuzzy Logic Controllers ($FLC_1 - FLC_3$). The three FLC's represent *Threat Level*, *Vulnerability* and *Asset Value*, which are three derived linguistic variables. The derived linguistic variables are then combined using FLC_4 to compute the net *Asset Risk*. This forms a hierarchical fuzzy system as shown in Figure 1. In this research, we used a Mamdani fuzzy inference system.

Values for the input variables are estimated based on the information from the HMM module, the DIDS and the traffic rate monitor. To simplify the calculation of input values, we have used the same attack categories as proposed by MIT Lincoln Laboratory - DARPA IDS evaluation datasets IDS [Ken99]. The local controller uses information from the DIDS and the traffic rate monitor to predict which attack category the next attack will fit into.

The following sub-sections are strongly based on some of the principles of the FAIR method described in [Jon06].

4.2 Fuzzy Modeling of Threat Level

Threat level is modeled using three linguistic variables: *intrusion frequency*, *probability of threat success* and *severity*. Three Membership Functions (MF) are used for the three inputs and the output variable.

Intrusion frequency describes the intensity of attack against the asset that is subject to monitoring. To estimate the intrusion frequency we use the output from the HMM module and count how often the probability of being in state *intrusion in progress* exceeds a specific limit. Intrusion frequency is measured as attacks/unit time.

Probability for threat success is estimated based on output from the DIDS, and describes how likely it is that an attacker will mange to overcome the proactive controls. The actual values are in the range $0 - 1$ and is stored in a lookup table.

Severity describes the impact of an attack on the asset.

All input variables to FLC_1 have three different linguistic values *Low*, *Medium* and *High*. The output from FLC_1 is Threat Level, and Fig 2 illustrates the *if-then* rules implemented in FLC_1 as a fuzzy associative memory (FAM). Figure 3 shows the controll surface view of FLC_1 plotting *Threat Level* as a function of *Probability of Threat Success* and *Intrusion Frequency*.

4.3 Fuzzy modeling of Vulnerability

The Vulnerability is estimated in FLC_2. Vulnerability may be defined as the probability that an asset will be unable to resist the action of a threat agent [Jon06]. In this paper we model vulnerability as a derived variable from *Threat Resistance* and *Threat Capability*. Three MF are assigned to each of the two input variables and the output variable.

Threat resistance is the strength of the security measures compared to the forces the attacker might use. One example of threat resistance is password length.

Threat capability is the level of force an attacker is capable of applying against an asset.

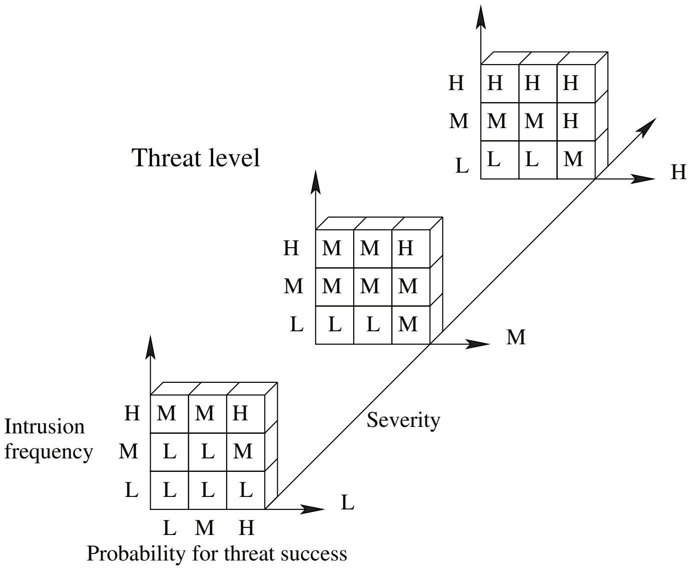


Figure 2. Sliced cube FAM representation of $\mathbf{FLC_1}$

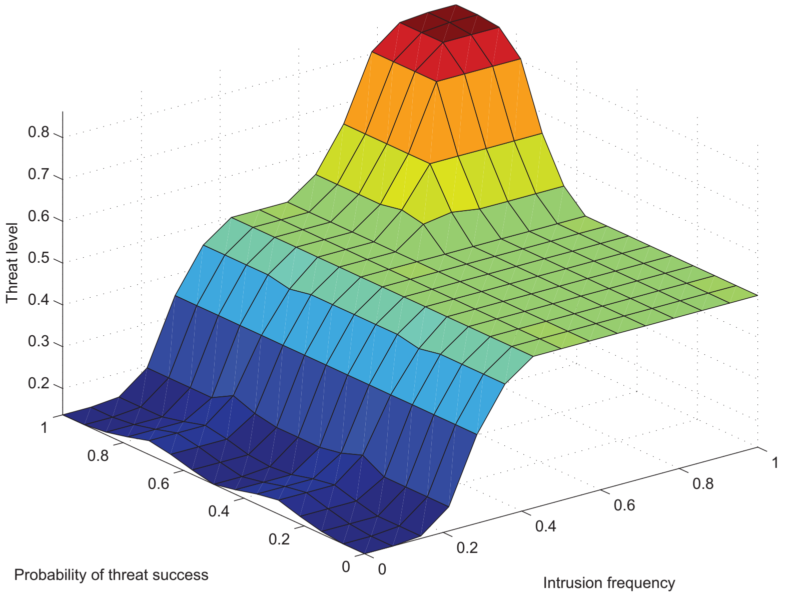
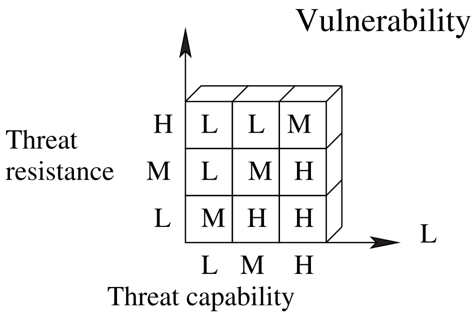


Figure 3. Controll Surface View of $\mathbf{FLC_1}$

The output variable from FLC_2 is the vulnerability and the *if-then* rules implemented in FLC_2 is depicted in Figure 4, and Figure 5 shows a control surface view of FLC_2.



Threat resistance	L	M	H
H	L	L	M
M	L	M	H
L	M	H	H

Figure 4. FAM representation of **FLC₂**

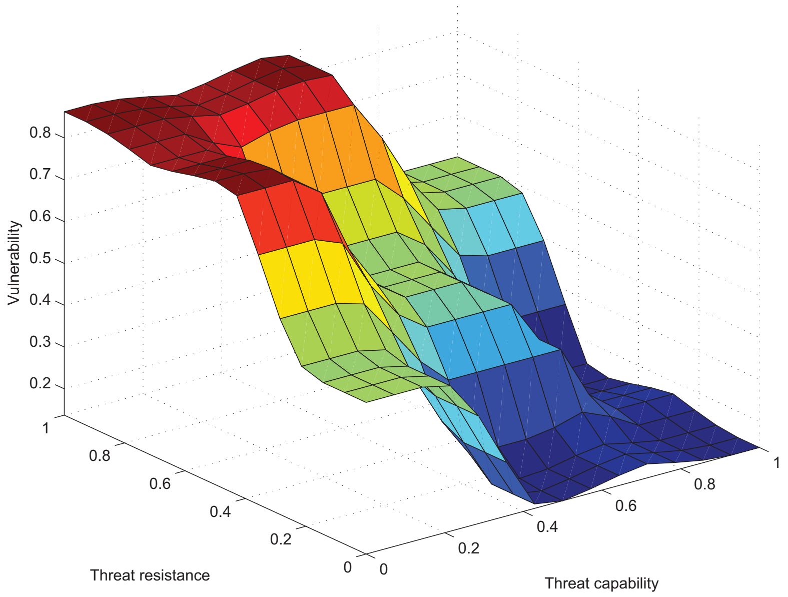


Figure 5. Control Surface View of **FLC₂**

4.4 Fuzzy modeling of Asset Value

The asset value is estimated in FLC_3 and is derived from three linguistic variables: *Cost*, *Criticality*, *Sensitivity* and *Recovery*. An asset is any data, device or other component that supports information-related activities, and which can be affected in a manner that result in loss. For all the input variables of FLC_3, we used only two MF (to reduce the number of *if-then* rules needed). Three MF are used for the output variable. The *if-then* rules implemented in FLC_3 is shown in Table 1. The first four columns represents the input linguistic values: *a* the *Cost*, *b* *Criticality*, *c* *Sensitivity* and *d* *Recovery*. The last column labeled e represents the output variable *Asset Value*. A control surface view of FLC_3 is shown in Figure 6.

Cost (a) Represents the cost associated with an asset that have been stolen or destroyed

Criticality (b) Mainly characterizes the impact on an organization's productivity. This attribute is related to integrity and availability.

Sensitivity (c) Impact of confidential information being disclosed.

Recovery (d) How fast the loss can be re-stored and the asset be back to normal again.

Table 1. Rule table for $\mathbf{FLC_3}$.

	Innput				Output
Rule	a	b	c	d	e
1	L	L	L	L	L
2	H	L	L	L	H
3	L	H	L	L	M
4	H	H	L	L	H
5	L	L	H	L	M
6	H	L	H	L	H
7	L	H	H	L	H
8	H	H	H	L	H
9	L	L	L	H	L
10	H	L	L	H	H
11	L	H	L	H	H
12	H	H	L	H	H
13	L	L	H	H	H
14	H	L	H	H	H
15	L	H	H	H	H
16	H	H	H	H	H

4.5 Fuzzy Modeling of Risk

The risk is estimated by FLC_4 and is based on the output from the three fuzzy logic controllers $FLC_1 - FLC_3$. For the input and output variables, three MF are used. The *if-then* rules used in FLC_4 is illustrated in Figure 7, and Figure 8 shows a control surface view of FLC_4.

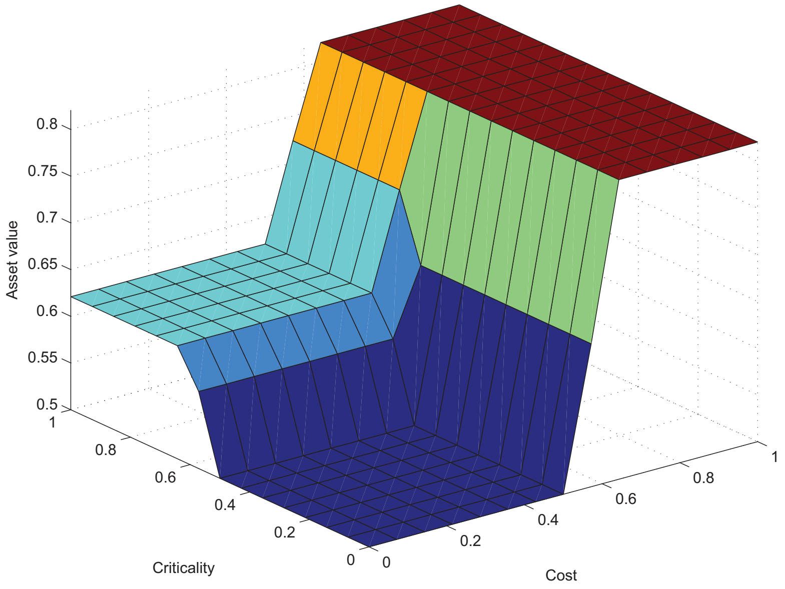


Figure 6. Surface plot of **FLC$_3$**

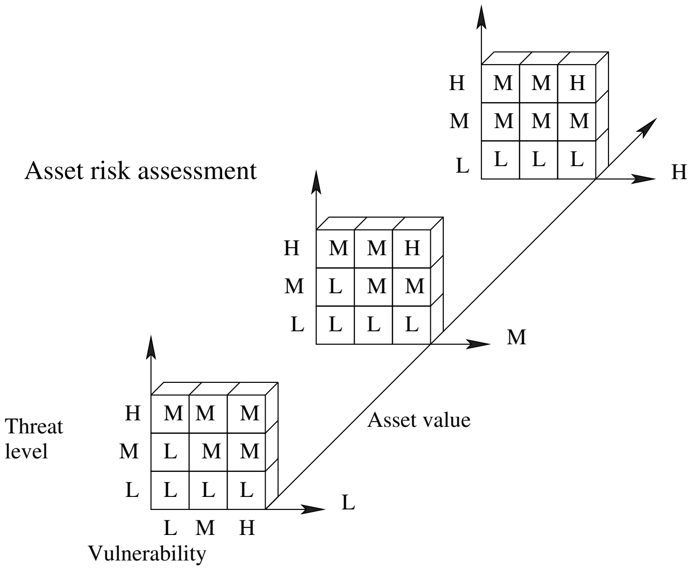


Figure 7. Sliced cube FAM representation of **FLC$_4$**

4.6 Lookup Table

All the input variables except the Intrusion Frequency is decided based on the information received from the DIDS agents about the the ongoing attack,

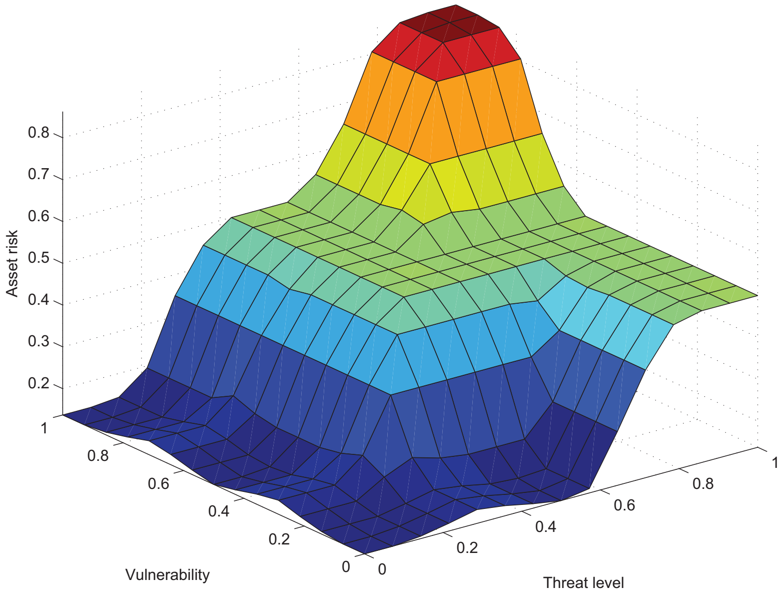


Figure 8. Surface plot of $\mathbf{FLC_4}$

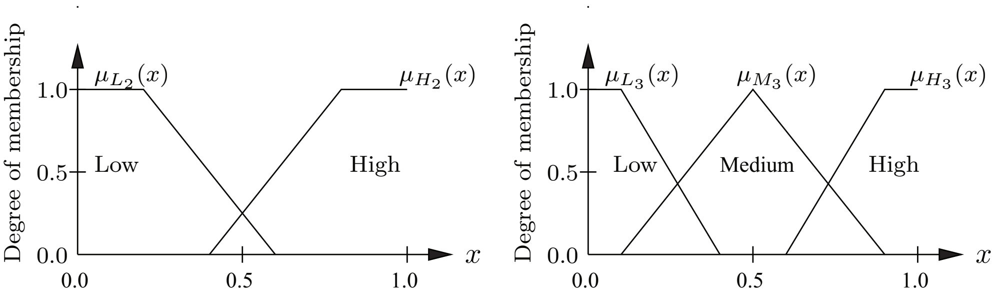
represented by a mapping $L(a) = y : A \to Y$ from attack types $a \in A = \{DoS, U2R, R2L, Pr\}$ to a parameter tuple $y \in Y = \mathbb{R}^8$. This function is implemented and referred to as a Lookup Table and contains parameters for different attack types. The information represents how vulnerable the system is to different attacks according to the value of different assets. These values have to be estimated by security experts.

For the risk assessment, two or three MF are proposed for each input variable, and three MF for the output variable. For two level input variables, we used the following two trapezoidal MF to define the *Low* and *High* linguistic values.

$$\begin{aligned} \mu_{L_2}(x) &= trap(x, -0.6, -0.2, 0.2, 0.6) \\ \mu_{H_2}(x) &= trap(x, 0.4, 0.8, 1.2, 1.6), \end{aligned} \tag{3}$$

For three level input and output variables, we propose two trapezoidal MF to define the *Low* and *High* linguistic values and a triangular MF to define *Medium* linguistic value (as illustrated below).

$$\begin{aligned} \mu_{L_3}(x) &= trap(x, -0.4, -0.1, 0.1, 0.4) \\ \mu_{M_3}(x) &= triang(x, 0.2, 0.5, 0.8) \\ \mu_{H_3}(x) &= trap(x, 0.6, 0.9, 1.0, 1.4). \end{aligned} \tag{4}$$



(a) Two level membership function

(b) Three level membership function

Figure 9. Membership functions

The membership functions used for input variables with three fuzzy sets are shown in Equation 5 and Figure 9(b).

All input variables are normalized and are members of the crisp set X defined as $X = \{x | 0 \leq x \leq 1,\ x \in \mathbb{R}\}$. The parameterized MF used are triangular Equation 5 and the trapezoidal Equation 6.

$$triang(x,a,b,c) = \begin{cases} 0 & x < a \\ (x-a)/(b-a) & a \leq x \leq b \\ (c-x)/(c-b) & b \leq x \leq c \\ 0 & x > c \end{cases} \tag{5}$$

$$trap(x,a,b,c,d) = \begin{cases} 0 & x < a \\ (x-a)/(b-a) & a \leq x \leq b \\ 1 & b \leq x \leq c \\ (d-x)/(d-c) & c \leq x \leq d \\ 0 & x > d \end{cases} \tag{6}$$

All fuzzy *if-then* rules were formulated based on expert knowledge.

5. Experiment Results

To illustrate the risk assessment, we have created two lookup chart as shown in Tables 2 and 3. Specific information about different attack categories is stored in a lookup table. All values in the lookup table is scaled within the range $0 - 1$. The attack category used for the risk assessment is based on inputs from the IDS agents and this value is used to assign values to eight of the nine input variables. Only the Intrusion Frequency is estimated based on the output from the HMM module.

Attacks are broadly divided into the following four categories: denial of service, remote to local, user to root and surveillance/probe.

A *denial of service* (DoS) attack is an attack where the attacker consume so much memory or CPU time that the legitimate users can not be served.

Table 2. Lookup Table1

	Attack Categories			
Variable	DoS	U2R	R2L	PR
Intrusion frequency	0.25	0.25	0.25	0.25
Pr threat success	0.90	0.70	0.70	0.10
Severity	0.40	0.90	0.90	0.30
Threat level	0.28	0.44	0.44	0.32
Threat resistance	0.10	0.60	0.90	0.20
Threat capabilit	0.50	0.85	0.80	0.10
Vulnerability	0.86	0.85	0.50	0.50
Cost	0.30	0.30	0.30	0.30
Criticality	0.70	0.70	0.70	0.10
Sensitivity	0.15	0.85	0.85	0.20
Recovery	0.40	0.85	0.70	0.15
Asset value	0.50	0.85	0.85	0.15
Asset risk	**0.34**	**0.50**	**0.50**	**0.40**

Table 3. Lookup Table 2

	Attack Categories			
Variable	DoS	U2R	R2L	PR
Intrusion frequency	0.25	0.25	0.25	0.25
Pr threat success	0.70	0.70	0.50	0.10
Severity	0.50	0.90	0.70	0.45
Threat level	0.32	0.44	0.32	0.28
Threat resistance	0.20	0.80	0.70	0.20
Threat capabilit	0.40	0.80	0.80	0.10
Vulnerability	0.85	0.50	0.63	0.50
Cost	0.40	0.40	0.50	0.30
Criticality	0.60	0.80	0.80	0.10
Sensitivity	0.20	0.80	0.70	0.10
Recovery	0.30	0.80	0.50	0.25
Asset value	0.50	0.84	0.82	0.15
Asset risk	**0.40**	**0.50**	**0.40**	**0.34**

Typical examples are Ping of Death, SYN Flood and Mailbomb. This attack is assumed to be easy to mount and indicated by high value for *Probability of Threat Success.* For DoS, the severity may be relatively low since it will not lead to much permanent damage. In most cases, the system may be restored to normal use once the attack is over.

An *User to root* (U2R) attack is an attack where an ordinary user in the system gain root access by exploiting some vulnerability in the system. Typical vulnerabilities that are exploited is buffer overflow and pure environment sanitation. A *remote to local* (R2L) attack is an attack where an attacker without an account on the computer tries to exploit some vulnerabilities to get access as an user of the computer. Possible attack strategies can be to exploit buffer overflows in network services software (imap, sendmail, apache). U2R and R2L categories are the most dangerous, since by gaining root access, the attacker could do almost everything with the system. Therefore a relatively high value for the *severity* is used in the above table. We assume the system to be well protected against U2R attacks indicated by relatively low *Probability of Threat Success*.

When an attacker uses some automated tools like Ipsweep, Nmap or Satan to gather information about the network and possible vulnerabilities we call it *probing*. This attack is assumed to be easy to mount and could pave way for further attacks.

Simulation results of the HMM is not reported in this paper due to space limitations, but the reader may consult [HAK07] for some preliminary results.

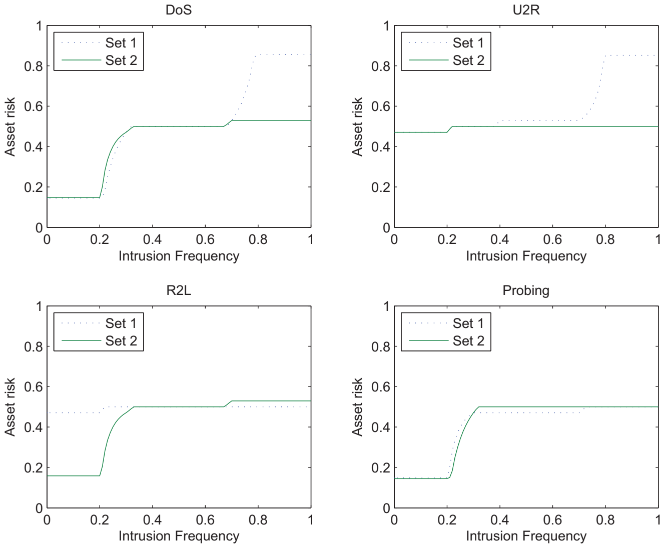


Figure 10. Parameter sensitivity for different attack categories

Figure 10 illustrates the asset risk values for different intrusion frequency variations (0-1). For the different parameter settings (Tables 2 and 3), as evident from Figure 10, the asset risk values show clear sensitivity for each attack category. This also illustrates that the proposed system is very adaptive for different attack categories under varying conditions.

6. Conclusions

This paper proposed a detailed implementation of a fuzzy logic based online risk assessment scheme, which could aid the functioning of a Distributed Intrusion Prediction and Prevention System (DIPPS) for protecting high risk assets. The implementation of the proposed scheme is very simple and the developed system is easy to interpret. Our discussions with security experts and preliminary empirical results indicate that such a system is very practical for protecting assets, which are prone to severe attacks or misuse.

In the current fuzzy risk expert system, fuzzy *if-then* rules were formulated based on expert knowledge. Our future research is targeted to develop adaptive fuzzy inference systems when some preliminary data or knowledge related to network risk is available. We also plan to investigate the use of different fuzzy inference methods.

References

[AJTH07] A. Abraham, R. Jain, J. Thomas, and S.Y. Han. D-scids: Distributed soft computing intrusion detection systems. *Journal of Network and Computer Applications, Elsevier Science*, 30(1):81–98, 2007.

[HAK07] Kjetil Haslum, Ajith Abraham, and Svein Knapskog. Dips: A framework for distributed intrusion prediction and prevention using hidden markov models and online fuzzy risk assessment. In *Third International Symposium on Information Assurance and Security, IEEE Computer Society press*, volume I, pages 183–188, 2007.

[Jon06] Jack Jones. An introduction to factor analysis of information risk (fair). *Norwich Journal of Information Assurance*, 2(1):67, 2006.

[Ken99] Kristopher Kendall. A database of computer attacks for the evaluation of intrusion detection systems. Master's thesis, MIT, USA, June 1999.

[KL06] Rahul Khanna and Huaping Liu. System approach to intrusion detection using hidden markov model. In *IWCMC '06: Proceeding of the 2006 international conference on Communications and mobile computing*, pages 349–354, New York, NY, USA, 2006. ACM Press.

[MVT02] B.B. Madan, K. Vaidyanathan, and K.S. Trivedi. Modeling and quantification of security attributes of software systems. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN'02)*, 2002.

Paper F

HiNFRA: Hierarchical Neuro-Fuzzy Learning for Online Risk Assessment

Kjetil Haslum, Ajith Abraham and Svein Knapskog

In Proceedings of the Second Asia International Conference on Modelling & Simulation

Kuala Lumpur, Malaysia. May 13-15, 2008

HINFRA: HIERARCHICAL NEURO-FUZZY LEARNING FOR ONLINE RISK ASSESSMENT

Kjetil Haslum, Ajith Abraham, and Svein Knapskog
{haslum, ajith.abraham, knapskog}@q2s.ntnu.no
Centre for Quantifiable Quality of Service in Communication Systems
Norwegian University of Science and Technology
O.S. Bragstads plass 2E, N-7491 Trondheim, Norway

Abstract Our previous research illustrated the design of fuzzy logic based online risk assessment for Distributed Intrusion Prediction and Prevention Systems (DIPPS) [HAK08]. Based on the DIPPS sensors, instead of merely preventing the attackers or blocking traffic, we propose a fuzzy logic based online risk assessment scheme. This paper propose a Hierarchical Neuro-Fuzzy online Risk Assessment (HiNFRA) model to aid the decision making process of a DIPPS. The fine tuning of fuzzy logic based risk assessment model is achieved using a neural network learning technique. Preliminary results indicate that the neural learning technique could improve the fuzzy controller performance and make the risk assessment model more robust.

1. Introduction

Intrusion prevention systems are proactive defense mechanisms designed to detect malicious packets embedded in normal network traffic and stop intrusions dead, blocking the offending traffic automatically before it does any damage rather than simply raising an alert as, or after, the malicious payload has been delivered. DIPPS are simply a superset of the conventional Intrusion Prevention System (IPS) implemented in a distributed environment. Basic architecture of a DIPPS element is depicted in Figure 1. We consider IPS as an integrated Intrusion Detection System (IDS) with many additional functions. Due to the distributed nature of IPS, the implementation poses several challenges. The IDSs are embedded inside software mobile agents and placed in the network to be monitored [AJTH07]. An individual IDS may be configured to detect a single attack, or it may detect several types of attacks.

In a large network, each DIPPS element communicates/coordinates with other DIPPS local controllers and/or a central controller. The Hidden Markov Model (HMM) [Rab90] model processes the attack data information from the various mobile agent IDS sensors [KL06]. Based on the nature of the detected attack, the following actions would be taken [HAK07]:

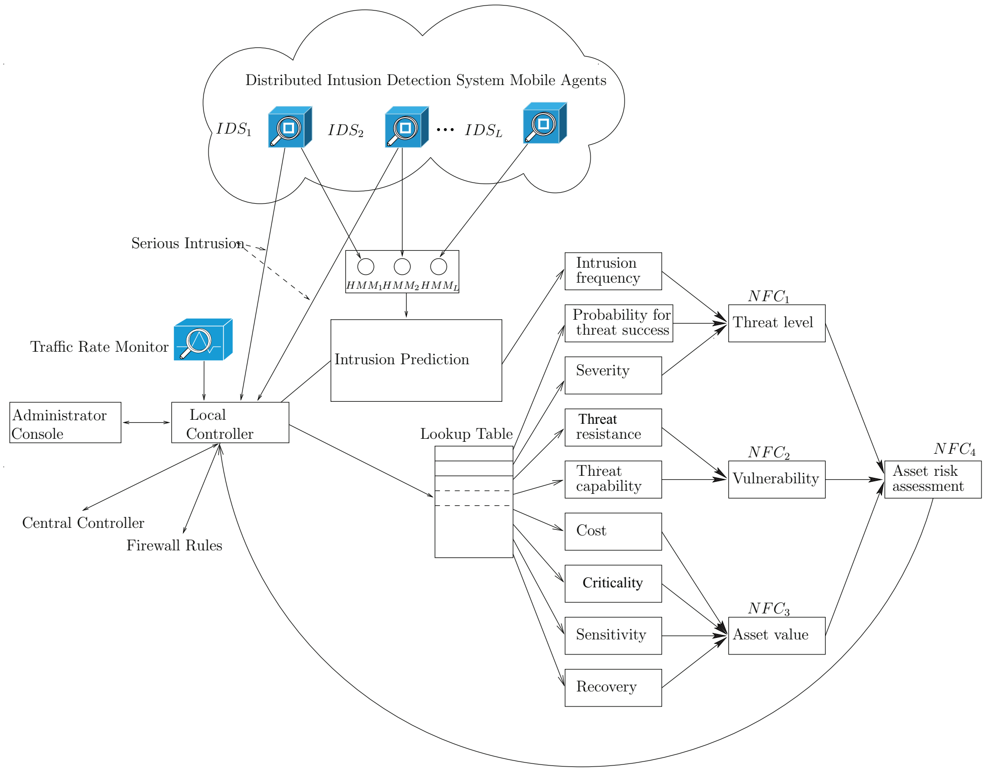


Figure 1. Architecture of a DIPPS element

1 If the detected attack is simply a port scan or a probe, the HMM model attempts to make a prediction of a possible future attack based on the current distributed attack patterns. Based on this prediction, the central controller (or administrator) would take precautionary measures to prevent future attacks. The central controller would also make use of an online risk assessment of the assets subjected to this possible serious attack in the future.

2 If the detected attack is very serious, the central controller would take necessary actions to re-configure firewall rules or notify the administrator etc. Such serious attacks would bypass the HMM model.

3 At any time any abnormal traffic rate is noted by the monitor if a predetermined level is reached, the central controller may take necessary actions to re-configure firewall rules or notify the administrator etc.

Risk assessment is often done by human experts, because there is no exact and mathematical solution to the problem. Usually the human reasoning and perception process cannot be expressed precisely. Different people have different opinions about risk and the association of its dependent variables, and fuzzy logic provides an excellent framework to model this [HAK08].

A Fuzzy Inference System (FIS) [Zad65] can utilize human expertise by storing its essential components in rule base and database, and perform fuzzy reasoning to infer the overall output value. The derivation of *if-then* rules and corresponding membership functions depends heavily on the *a priori* knowledge about the system under consideration. However there is no systematic way to transform experiences of knowledge of human experts to the knowledge base of a FIS. There is also a need for adaptability or some learning algorithms to produce outputs within the required error rate.

This paper is focused on the development of neural network learning techniques for the optimization of a fuzzy risk assessment system. The rest of this paper is organized as follows. Section 2 presents the proposed neuro-fuzzy risk assessment model. Experiment results are given in Section 3 followed by conclusions towards the end.

2. Neuro-Fuzzy Risk Assessment Model

2.1 Fuzzy modeling of risk

For risk assessment, nine basic linguistic variables are used that are processed using three Neuro-Fuzzy Controllers ($NFC_1 - NFC_3$). The three NFC's represent *Threat Level*, *Vulnerability* and *Asset Value*, which are three derived linguistic variables. Threat level is modeled using three linguistic variables: *intrusion frequency*, *probability of threat success* and *severity*. We model vulnerability as a derived variable from *threat resistance* and *threat capability*. The asset value is derived from three linguistic variables: *Cost*, *Criticality*, *Sensitivity* and *Recovery*. The derived linguistic variables are then combined using NFC_4 to compute the net *Asset Risk*. This forms a hierarchical fuzzy system as shown in Figure 2.

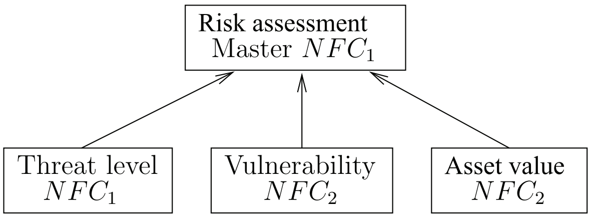


Figure 2. Hierarchical architecture of four fuzzy logic controllers

Values for the input variables are estimated based on the information from the HMM module, the DIDS and the traffic rate monitor. To simplify the calculation of input values, we have used the same attack categories as proposed by MIT Lincoln Laboratory - DARPA IDS evaluation datasets IDS [Ken99].

The local controller uses information from the DIDS and the traffic rate monitor to predict which attack category the next attack will fit into.

2.2 Fuzzy risk model optimization using neural learning

In an integrated model, neural network learning algorithms are used to determine the parameters of fuzzy inference systems. Integrated neuro-fuzzy systems share data structures and knowledge representations. A fuzzy inference system can utilize human expertise by storing its essential components in rule base and database, and perform fuzzy reasoning to infer the overall output value. The derivation of *if-then* rules and corresponding membership functions depends heavily on the *a priori* knowledge about the system under consideration. However there is no systematic way to transform experiences of knowledge of human experts to the knowledge base of a fuzzy inference system. There is also a need for adaptability or some learning algorithms to produce outputs within the required error rate. On the other hand, Artificial Neural Network (ANN) learning mechanism does not rely on human expertise. Due to the homogenous structure of ANN, it is hard to extract structured knowledge from either the weights or the configuration of the network. The weights of the neural network represent the coefficients of the hyper-plane that partition the input space into two regions with different output values. If we can visualize this hyper-plane structure from the training data then the subsequent learning procedures in a neural network can be reduced. However, in reality, the a priori knowledge is usually obtained from human experts, it is most appropriate to express the knowledge as a set of fuzzy if-then rules, and it is very difficult to encode into a neural network.

To a large extent, the drawbacks pertaining to these two approaches seem complementary. Therefore, it seems natural to consider building an integrated system combining the concepts of FIS and ANN modeling. A common way to apply a learning algorithm to a fuzzy system is to represent it in a special neural network like architecture. However the conventional neural network learning algorithms (gradient descent) cannot be applied directly to such a system as the functions used in the inference process are usually non differentiable. This problem can be tackled by using differentiable functions in the inference system or by not using the standard neural learning algorithm. In our simulation, we used the Adaptive Network Based Fuzzy Inference System (ANFIS) [Jan93]. ANFIS implements a Takagi Sugeno Kang (TSK) fuzzy inference system [Sug85] in which the conclusion of a fuzzy rule is constituted by a weighted linear combination of the crisp inputs rather than a fuzzy set.

For a first order TSK model, a common rule set with two fuzzy *if-then* rules is represented as follows:

Rule 1: If x is A_1 and y is B_1, then $f_1 = p_1x + q_1y + r_1$

Rule 2: If x is A_2 and y is B_2, then $f_2 = p_2x + q_2y + r_2$

where x and y are linguistic variables and A_1, $A_{2,}$, $B_{1,}$, B_2 are corresponding fuzzy sets and p_1 , q_1, $r_1$and p_2, q_2, r_2 ,are linear parameters.

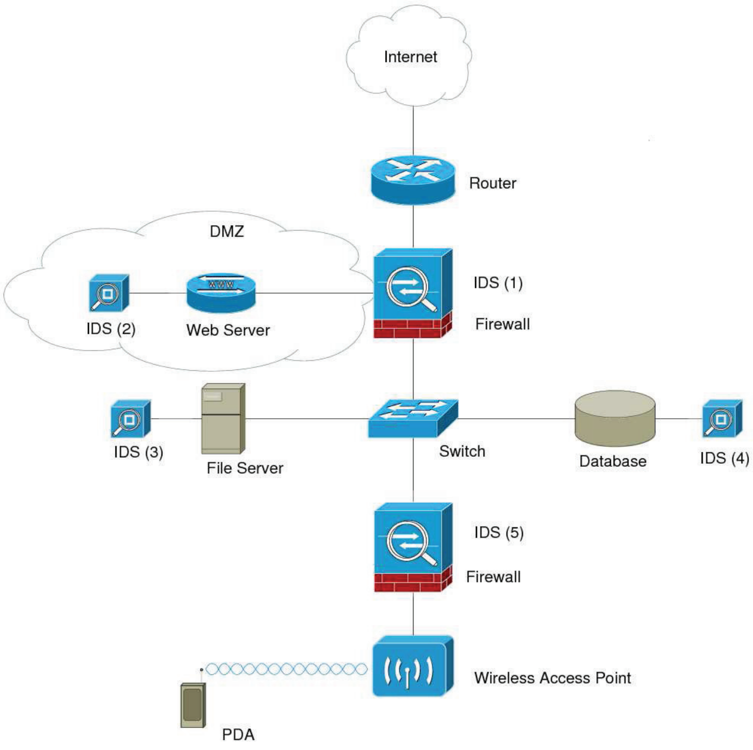


Figure 3. Example network showing assets and IDS agents

ANFIS makes use of a mixture of back propagation to learn the premise parameters and least mean square estimation to determine the consequent parameters. A step in the learning procedure has two parts: In the first part the input patterns are propagated, and the optimal conclusion parameters are estimated by an iterative least mean square procedure, while the antecedent parameters (membership functions) are assumed to be fixed for the current cycle through the training set. In the second part the patterns are propagated again, and in this epoch, back propagation is used to modify the antecedent parameters, while the conclusion parameters remain fixed. This procedure is then iterated.

3. Experiment Results

In order to illustrate the neuro-fuzzy risk assessment model, we constructed a small network model as illustrated in Figure 3. The sample network consists of four different assets; a router, a public web server, a file server, and a database. Five IDS Agents denoted by $IDS_1, \ldots, IDS_5$ are deployed in the

Table 1. Learning results for NFC1

MF		tri	gbell	gauss	psig
Data set 1	#MF	2	2	2	2
	Epochs	49	27	17	28
	Test error	0.0969	0.1159	0.1002	0.0881
	Train error	0.0203	0.0172	0.0192	0.0211
Data set 2	#MF	2	2	2	2
	Epochs	45	37	22	8
	Test error	0.1371	0.1500	0.1091	0.1732
	Train error	0.0220	0.0225	0.0228	0.0480
Data set 3	#MF	2	2	2	2
	Epochs	42	14	16	4
	Test error	0.1239	0.0694	0.0756	0.0513
	Train error	0.0251	0.0379	0.0379	0.0303

Table 2. Learning results for NFC2

MF		tri	gbell	gauss	psig
Data set 1	#MF	2	3	3	3
	Epochs	22	1	1	1
	Test error	0.0306	0.0392	0.0368	0.0444
	Train error	0.0400	0.0256	0.0252	0.0256
Data set 2	#MF	3	2	2	3
	Epochs	1	1	2	1
	Test error	0.0473	0.0443	0.0568	0.0528
	Train error	0.0214	0.0681	0.0294	0.0184
Data set 3	#MF	3	3	3	3
	Epochs	1	1	1	1
	Test error	0.0223	0.0350	0.0348	0.0376
	Train error	0.0308	0.0272	0.0267	0.0273

network, and the observations are sent to the their corresponding HMM. The attack category used for the risk assessment is based on inputs from the IDS agents and this value is used to assign values to eight of the nine input variables. Only the Intrusion Frequency is estimated based on the output from the HMM module.

Table 3. Learning results for NFC3

MF		tri	gbell	gauss	psig
	#MF	2	2	2	2
Data set 1	Epochs	1	8	1	22
	Test error	0.0840	0.0726	0.0686	0.0788
	Train error	0.0342	0.0536	0.0498	0.0557
	#MF	2	2	2	2
Data set 2	Epochs	1	1	1	68
	Test error	0.0627	0.1187	0.0665	0.1339
	Train error	0.0352	0.0552	0.0470	0.0290
	#MF	2	2	2	2
Data set 3	Epochs	1	20	14	30
	Test error	0.2268	0.2206	0.2644	0.1170
	Train error	0.0332	0.0344	0.0328	0.0423

Table 4. Learning results for NFC4

MF		tri	gbell	gauss	psig
	#MF	2	2	2	2
Data set 1	Epochs	2	7	11	4
	Test error	0.1691	0.1488	0.1555	0.1379
	Train error	0.0748	0.0829	0.0724	0.1031
	#MF	2	2	2	2
Data set 2	Epochs	51	17	15	19
	Testing error	0.1541	0.1557	0.1689	0.2250
	Training error	0.0649	0.0603	0.0600	0.0600
	#MF	2	2	2	2
Data set 3	Epochs	39	20	20	24
	Testing error	0.1396	0.1122	0.1148	0.1098
	Training error	0.0561	0.0508	0.0485	0.0558

3.1 Hierarchical neuro-fuzzy modeling

To avoid any bias in the learning process, we randomly sampled three sets of data from the master data set. 75% of the data was used for training and the remaining for test data. We implemented a Hierarchical Neuro-Fuzzy Risk Assessment (HiNFRA) model as illustrated in Figure 2. The performance of the four controllers for different membership functions for three different data sets are depicted in Tables 1, 2, 3 and 4. We used four different Membership Functions (MF): Triangular (tri), Gaussian bell (gbell), Gaussian (gauss) and product of two sigmoidal function (psig). As evident, depending on the

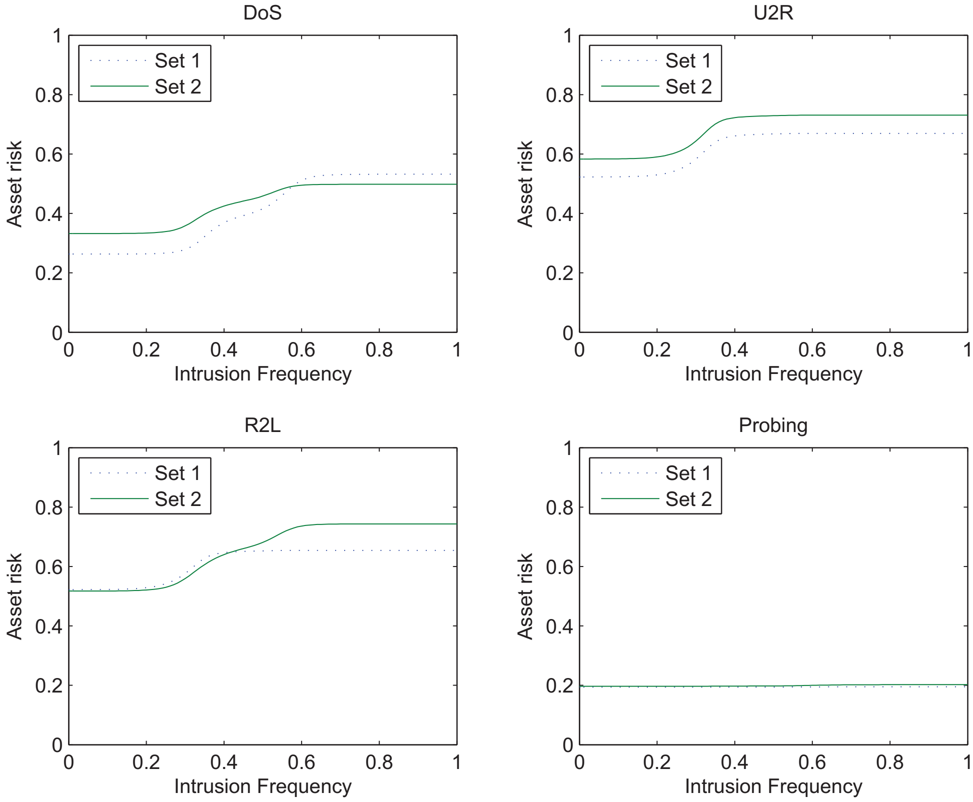


Figure 4. Parameter sensitivity for different attack categories

membership function and data set used, different controllers are using varying number of epochs. In some cases, the NFC was constructed after 1 learning epoch. The four NFC's were build individually and then connected as shown in Figure 2.

The developed HiNFRA model is then tested using two sets of data based on an attack situation. Specific information about different attack categories are stored in a lookup table. All values in the lookup table is scaled within the range $0 - 1$. The attack category used for the risk assessment is based on inputs from the IDS agents and this value is used to assign values to eight of the nine input variables. Only the Intrusion Frequency is estimated based on the output from the HMM module.

The two lookup tables and final results (asset risk) are illustrated in Tables 5 and 6.

Figure 4 illustrates the asset risk values for different intrusion frequency variations (0-1). For the different parameter settings (Tables 5 and 6), as evident from Figure 4, the asset risk values show clear sensitivity for each attack category. This also illustrates that the proposed system is very adaptive for different attack categories under varying conditions. Figures 5 - 8 illustrates the surface plots of the four developed controllers.

4. Conclusions

This paper presented a detailed implementation of the Hierarchical Neuro-Fuzzy online Risk Assessment (HiNFRA) model to aid the decision making process of DIPPS. The fine tuning of fuzzy logic based risk assessment model is achieved using neural network learning technique. Preliminary results indicate that neural learning techniques could improve the fuzzy controller performance and make the risk assessment model more robust. Compared to our previous model [HAK08], where the fuzzy *if-then* rules were formulated based on expert knowledge, the implementation of HiNFRA is more simple and adaptive.

Our future research is targeted to further develop and optimize fuzzy risk models using evolutionary algorithms.

Table 5. Lookup Table1

	Attack Categories			
Variable	DoS	U2R	R2L	PR
Intrusion frequency	0.25	0.25	0.25	0.25
Pr threat success	0.90	0.70	0.70	0.10
Severity	0.40	0.90	0.90	0.30
Threat level	0.38	0.52	0.52	0.29
Threat resistance	0.10	0.60	0.90	0.20
Threat capabilit	0.50	0.85	0.80	0.10
Vulnerability	0.46	0.36	0.15	0.10
Cost	0.30	0.30	0.30	0.30
Criticality	0.70	0.70	0.70	0.10
Sensitivity	0.15	0.85	0.85	0.20
Recovery	0.40	0.85	0.70	0.15
Asset value	0.33	0.52	0.52	0.24
Asset risk	**0.27**	**0.54**	**0.54**	**0.19**

Table 6. Lookup Table2

	Attack Categories			
Variable	DoS	U2R	R2L	PR
Intrusion frequency	0.25	0.25	0.25	0.25
Pr threat success	0.70	0.70	0.50	0.10
Severity	0.50	0.90	0.70	0.45
Threat level	0.42	0.52	0.44	0.31
Threat resistance	0.20	0.80	0.70	0.20
Threat capabilit	0.40	0.80	0.80	0.10
Vulnerability	0.33	0.20	0.26	0.10
Cost	0.40	0.40	0.50	0.30
Criticality	0.60	0.80	0.80	0.10
Sensitivity	0.20	0.80	0.70	0.10
Recovery	0.30	0.80	0.50	0.25
Asset value	0.39	0.59	0.60	0.24
Asset risk	**0.39**	**0.60**	**0.53**	**0.20**

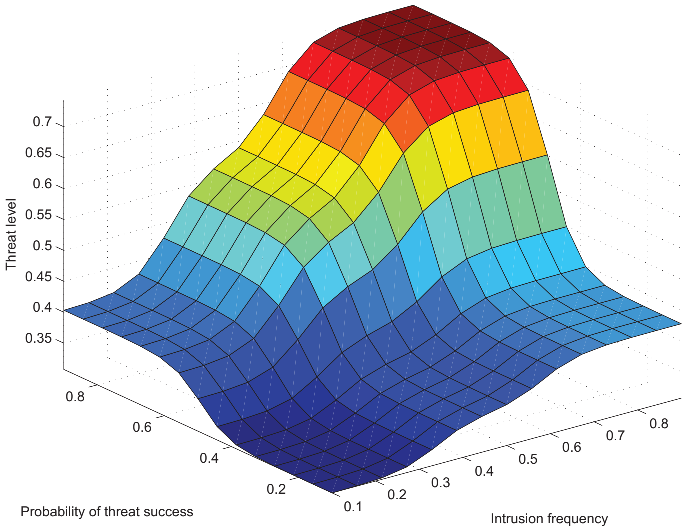


Figure 5. Control Surface View of NFC_1

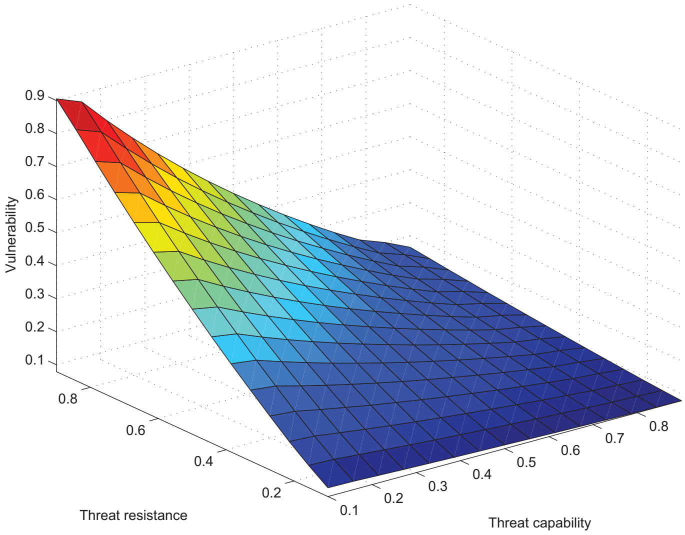


Figure 6. Control Surface View of NFC_2

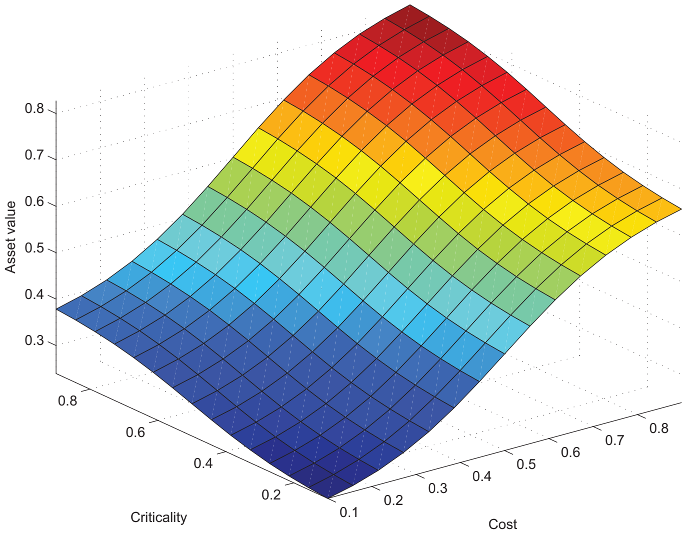


Figure 7. Control Surface View of NFC_3

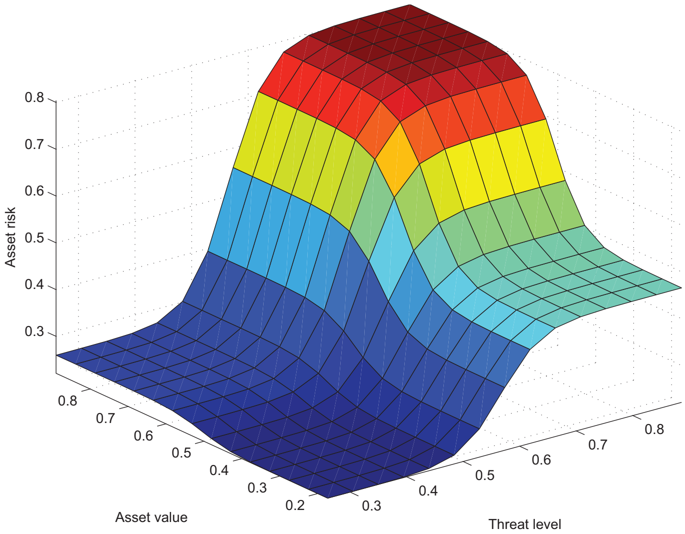


Figure 8. Control Surface View of NFC_4

References

[AJTH07] A. Abraham, R. Jain, J. Thomas, and S.Y. Han. D-scids: Distributed soft computing intrusion detection systems. *Journal of Network and Computer Applications, Elsevier Science*, 30(1):81–98, 2007.

[HAK07] Kjetil Haslum, Ajith Abraham, and Svein Knapskog. Dips: A framework for distributed intrusion prediction and prevention using hidden markov models and online fuzzy risk assessment. In *Third International Symposium on Information Assurance and Security, IEEE Computer Society press*, volume I, pages 183–188, 2007.

[HAK08] Kjetil Haslum, Ajith Abraham, and Svein Knapskog. Fuzzy online risk assessment for distributed intrusion prediction and prevention systems. In *Tenth International Conference on Modeling and Simulation, IEEE Computer Society press*, volume I, page in press, 2008.

[Jan93] J.-S.R. Jang. Anfis: adaptive-network-based fuzzy inference system. *Systems, Man and Cybernetics, IEEE Transactions on*, 23(3):665–685, May/Jun 1993.

[Ken99] Kristopher Kendall. A database of computer attacks for the evaluation of intrusion detection systems. Master's thesis, MIT, USA, June 1999.

[KL06] Rahul Khanna and Huaping Liu. System approach to intrusion detection using hidden markov model. In *IWCMC '06: Proceeding of the 2006 international conference on Communications and mobile computing*, pages 349–354, New York, NY, USA, 2006. ACM Press.

[Rab90] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Readings in speech recognition*, pages 267–296, 1990.

[Sug85] Michio Sugeno. *Industrial Applications of Fuzzy Control*. Elsevier Science Inc., New York, NY, USA, 1985.

[Zad65] L.A. Zadeh. Fuzzy sets. *Info. & Ctl.*, 8:338–353, 1965.

Paper G

Real-time Intrusion Prevention and Security Analysis of Networks using HMMs

Kjetil Haslum, Marie Elisabeth Gaup Moe, and Svein Knapskog

In Proceedings of the Fourth IEEE LCN Workshop on Network Security (WNS 2008)

Montreal, Canada. October 17, 2008

REAL-TIME INTRUSION PREVENTION AND SECURITY ANALYSIS OF NETWORKS USING HMMS

Kjetil Haslum, Marie Elisabeth Gaup Moe, and Svein Johan Knapskog
Centre for Quantifiable Quality of Service in Communication Systems
Norwegian University of Science and Technology
O.S. Bragstads plass 2E, N-7491 Trondheim, Norway
{haslum,marieeli,knapskog}@q2s.ntnu.no

Abstract In this paper we propose to use a hidden Markov model (HMM) to model sensors for an intrusion prevention system (IPS). Observations from different sensors are aggregated in the HMM and the intrusion frequency security metric is estimated. We use a Markov model that captures the interaction between the attacker and the network to model and predict the next step of an attacker. A new HMM is created and used for updating the estimated system state for each observation, based on the sensor trustworthiness and the time since last observation processed. Our objective is to calculate and maintain a state probability distribution that can be used for intrusion prediction and prevention. We show how our sensor model can be applied to an IPS architecture based on intrusion detection system (IDS) sensors, real-time traffic surveillance and online risk assessment. Our approach is illustrated by a small case study.

1. Introduction

In todays society most critical infrastructures depend on computer networks, which in many cases could be reachable for attackers. As a consequence the security of these critical systems, in addition to traditional quality of service (QoS) parameters like availability, reliability and performance, needs to meet certain requirements. In order to specify these requirements there is a need for the security attributes to be quantified. We also need to have methods for monitoring the security of the system, and means to detect security breaches and possibly prevent intrusions.

In this paper we will present an intrusion prevention system (IPS) architecture based on intrusion detection system (IDS) sensors with extended real-time traffic surveillance and online risk assessment. To model and predict the next step of an attacker, we use a Markov model that captures the interaction between the attacker and the network. The system security state is not always readily observable, due to stealthy attacks the system could be under attack even though there are no IDS alerts indicating a security breach. Since the true security state of the system is unknown, and we cannot expect the IDS sensors to detect all malicious activity at all times, we model each sensor using a hidden Markov model (HMM).

We also propose to use parameters of the Markov model as a basis for a quantification of the security of the network under surveillance. This work is an extension of the distributed IPS (DIPS) with online fuzzy risk assessment proposed in [HAK07], the focus of this paper is on a more extensive mathematical analysis of our model to derive quantitative measures of security that can be used as input to the fuzzy logic based risk assessment. We also demonstrate our approach with the help of an example case study. To simplify the presentation, we do not consider a *distributed* IPS in this paper, but the approach presented could be extended to a distributed setting with the use of distributed agents.

Previous work on Markov modeling of security includes Madan et al. [BB-MGPT04], that present a method for quantifying the security attributes of intrusion tolerant systems based on methods from the software dependability community. Their model differs from ours as they use a static Markov model of the system while we use a HMM to model the sensors. Khanna et al. [KL06] propose to build an IDS based on an HMM with multivariate Gaussian distributed observations, and dynamic re-estimation of parameters. In [KL07], Khanna et al. uses distributed HMM processing in combination with a proportional integral differential (PID) control engine to make a distributed IDS for ad hoc networks. Their approach is different from ours as they use continuous observations and are only modeling one sensor at a time.

Årnes et al. [ÅSH$^+$05] use HMMs for real-time risk assessment, some promising results on simulated and real-life data can be found in [ÅVVK06]. Sallhammar et al. [SHK07] describes a framework for combined security and dependability evaluation of computer networks, our approach differs as we use alerts from multiple IDS sensors, and do not poll the sensors, but instead update the system state estimate at the time when an alert is produced by one of the sensors, since alerts do not usually come at regular intervals.

The contribution of this paper is the method used for aggregating alerts from multiple sensors, the proposed sensor modeling technique that includes the time between observations and the integration of our model into an IPS architecture.

The remainder of this paper is organized as follows. Section 2 presents the system Markov model with a motivation of the stochastic modeling approach and the limitations and assumptions related to the model. The HMM approach for sensor modeling is presented in Section 3, where we discuss the observation process and the aggregation and translation of IDS alerts into observations in the HMM, and also derive the intrusion frequency metric.

The IPS architecture is presented in Section 4 and a small case study is given in Section 5. Finally, Section 6 gives some concluding remarks and discusses future work.

2. System Model

In this section we will explain our stochastic modeling approach of security. We use a first order continuous time Markov model (CTMM), where the states represent the security state of the system. Our objective is to calculate and maintain a state probability distribution that can be used for intrusion prediction and prevention.

2.1 Stochastic Modeling Approach

The *security state* of a system could be described by several aspects of the computer network and its environment. These aspects could for instance be the security condition of the network, if security updates and patches have been installed, how the security policies are implemented, what attack countermeasures are in place, the existence of intrusion detection systems and so on. In any modeling effort there is always a trade-off between model complexity and model precision, that limits how many aspects one can include in the state definition of the model. In our approach we seek to model the interaction between the system and the attackers by describing the security state of the system in terms of the different stages of a typical attack against a computer network. The system is in one of the states at any given time, the behavior of the system can be modeled as a *stochastic process* as it is probabilistic and described in terms of state transitions that are triggered by events which happen randomly according to a probability distribution.

Since the initial ideas on quantifying security with a stochastic modeling approach, presented by Littlewood et al. [LBF$^+$93], there has been some research effort, for instance the work done by Madan et al. [MVT02, WMT03, BB-MGPT04]. Markov models have traditionally been used to model and evaluate computer system dependability, Nicol et al. [NST04] argue that some of the modeling techniques used in the dependability community can be applied for security evaluation, they suggest that Markov reward models may be suitable to model security aspects of software systems. Jonsson and Olovsson [JO97] conducted an intrusion experiment where students attacked a real system under controlled conditions. Their experiments illustrated that an attack can be divided into different phases where time between security breaches is exponentially distributed. This supports our approach in this paper, where we use continuous time Markov models to model the interaction between attackers and the system they are attacking. Kaâniche et al. [KAN$^+$06] have performed an empirical analysis of data collected from thirty five honeypots located in twenty five countries. They found that the distribution of times between attacks is best described by a mixture distribution combining a Pareto

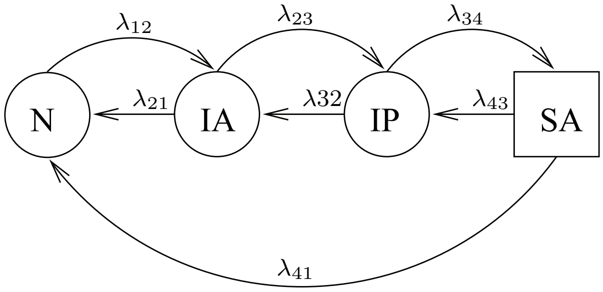


Figure 1. A Markov model describing the interaction between system and attackers

distribution and an exponential distribution. This is an argument for using a semi-Markov model to model the intrusion process, where state occupation times can have arbitrary distributions, and a hidden semi-Markov model (HSMM) to model a sensor. We have chosen to not use a HSMM, in order to keep the presentation as clear as possible. The reason for this is the complexity of statistical inference in HSMMs, but in our further work the use of different probability distributions in a HSMM will be considered.

2.2 Limitations and Assumptions

The security state of individual network elements is not modeled and it is assumed that if an attacker has control over one element then the whole system is compromised. This assumption is justified by the fact that when an attacker has broken into one network element, it is often easy to use this element as a platform for future attacks inside the network. We do not attempt to model the number of attackers that are attacking the system or have broken into the system, since it might be almost impossible to distinguish between attackers that operate independently and attackers that cooperate. Attacks are also often launched as distributed attacks using several compromised servers as stepping stones, thus attacks that come from different IP addresses might be originating from one or several attackers. When using a CTMM to model the system and an HMM to model the sensors, we make the following assumptions; all information about the system is contained in the state of the system, observations are independent given the current state, and state occupation times are exponentially distributed.

2.3 Markov Model

The Markov model (Λ, π) used in this paper is shown in Figure 1. States drawn as circles indicate secure states, states drawn by a square indicate a security breach, i.e. the system has been compromised and suffered some loss of integrity and/or confidentiality.

The state of the system is modeled as a stochastic process $\boldsymbol{x}(t)$, where t is a positive real number representing time since the system started operating. A CTMM has a set of states $S = \{s_1, s_2, \ldots, s_N\}$, an initial distribution $\pi = (\pi_i)$, describing the state of the system when monitoring starts, and a transition rate matrix $\Lambda = (\lambda_{ij})$, describing the dynamics of the system. To simplify the notation of equations and algorithms we will use i and j instead of s_i and s_j. The relation between system states and the transition rates is given by

$$\lambda_{ij} = \begin{cases} \lim_{dt \to 0} \dfrac{P(\boldsymbol{x}(t+dt) = j | \boldsymbol{x}(t) = i)}{dt} & \text{if } i \neq j \\ \sum_{j \neq i, j=1}^{N} -\lambda_{ij} & \text{if } i = j \end{cases} . \tag{1}$$

By using a CTMM we assume time between state changes or state occupation times to be negatively exponential distributed with mean $h_i = \frac{1}{\sum_j \lambda_{ij}}$.

The statespace used in this paper consists of the following four states:

1 Normal (N) indicates no directed suspicious activity against the network, except from the usual automated scannings which are considered as background noise

2 Intrusion attempt (IA) indicates that one or more attackers are scanning for vulnerabilities which the system is not protected against

3 Intrusion in progress (IP) indicates that one or more attackers have started an attack against the system. The system is still functioning correctly and no confidentiality or integrity breaches have occurred

4 Successful attack (SA) indicates that one or more attackers have broken into the system and may have full control over the system

We will now describe in more details the meaning of each transition probability and the state occupation time. To simplify the mathematical expressions the following notation is used: $\tilde{p}_a = 1 - p_a$, $\tilde{p}_r = 1 - p_r$, and $\tilde{p}_s = 1 - p_s$. The transition matrix $P = (p_{ij})$ is given by

$$P = \begin{array}{c c} & \begin{array}{cccc} N & IA & IP & SA \end{array} \\ \begin{array}{c} N \\ IA \\ IP \\ SA \end{array} & \begin{pmatrix} 0 & 1 & 0 & 0 \\ \tilde{p}_a & 0 & p_a & 0 \\ 0 & \tilde{p}_s & 0 & p_s \\ p_r & 0 & \tilde{p}_r & 0 \end{pmatrix} \end{array}, \tag{2}$$

where p_a denotes the probability that an attacker manually starts an active attack on the system, after first having investigated the system by some automatic tools to find security holes, p_s is the probability that the attack is successful, the attacker has become root and the system is completely compromised, and p_r is the probability that a system repair is successful, the attacker is shut out of the system and the security vulnerabilities that were exploited by the attacker have been removed.

The P matrix describes the probability distribution for next transition, and has no self transition because we model the state occupation times seperately. State occupation times are expressed by $H = (h_N \;\; h_{IA} \;\; h_{IP} \;\; h_{SA})$,

where h_N denotes the mean time before the system is targeted directly by an attacker based on vulnerabilities found by automated scanning, h_{IA} is the mean time between the discovery of an exploitable vulnerability until the active attack is launched, h_{IP} is the mean time an attacker uses to defeat the defense mechanisms in the system, and h_{SA} is the mean time the the system is not functioning correctly due to security attacks before the attack is discovered by the network administrator and repaired.

The relation between transition probability and transition rates is given by

$$\lambda_{ij} = \frac{p_{ij}}{h_i} \quad \text{if } i \neq j. \tag{3}$$

3. Sensor Model

In order to use Markov models for the detection and prediction of attacks, we need to process observations received from sensors. This is done by extending the CTMM to a Discrete time HMM by using an HMM to model the sensors that observe the interaction between an attacker and the system, see [Rab90] for an introduction to HMMs. Lets assume that the system has L sensors. Each sensor $\psi \in \{1, \ldots, L\}$ is modeled by a separate HMM (Λ, π, Q^{ψ}), that includes a common system model (Λ, π), but individual observation probability matrices Q^{ψ}, that are used to model specific properties for each sensor. The matrix Q^{ψ} describes the quality or the trustworthiness of sensor ψ that produced the observation, and $V = \{v_1, v_2, \ldots, v_M\}$ are the possible observations. To simplify the notation we use m instead of v_m.

3.1 Aggregation and Filtering of Alerts

Different sensors produce a large variation of alerts, that need to be translated into observations. In our model this translation is done by filtering alerts according to alert severity. The alert severity rating, together with the timestamps, are then used as observations in the HMM. To avoid dependencies between observations, we assume that there has been done some preprocessing of the alerts before they are translated to observations in our model. Alerts caused by the same event should be grouped together by some correlation or aggregation technique.

There has been a research focus lately on different types of automatic processing of alerts as a means to prioritize alerts, reduce the number of false IDS alerts and hence minimize the manual work in the handling of alerts. Aggregation and correlation techniques could be based on compression, filtering or thresholding. New alerts, so-called meta-alerts that refer to attack classes which gives a high level description of the attack, can be derived as a generalization or refinement of the correlated alerts [VVKK04]. Aggregation and correlation analysis [DW01] can be used to group together alerts based on certain criteria similar to database views. Alerts are then correlated by detecting duplicates and consequence alerts (alerts that are logically linked together according to pre-specified rules).

Data mining and clustering [Jul01] is another approach that is used for extracting the true alerts in a large data set of IDS output by clustering alerts according to their root cause. False positives can be reduced considerably, but at the cost of additional increase in false negatives for infrequent alerts.

In our approach the amount of alerts that needs to be manually handled by the system administrator is effectively minimized, as most of the alerts, like alerts for port scans or probing, are only used as input to the HMM. Only when the HMM output indicates that the system seems to be under attack will the most serious intrusion alerts produce output to the administrator console. These serious alerts will also be treated directly by the local controller, that will take necessary actions to prevent the attack by reconfiguration or termination of processes, according to prespecified rules.

3.2 The Observation Process

We have not made any assumptions about time between observations, and there is no direct relation between observations and state-changes. As a consequence the system could have made zero, one or more transitions during the time between to successive observations.

The times of state transitions are denoted $\hat{t}_l$, starting with $\hat{t}_0$, which is the time when the first state transition takes place. We assume the transition to take place a infinitesimal time after $\hat{t}_l$. Sojourn times are denoted $\hat{\delta}_l = \hat{t}_l - \hat{t}_{l-1}$, the state transitions are illustrated by the bullets in Figure 2.

The observation process is denoted $\boldsymbol{y}_k$, the time when the observation is produced is denoted t_k, and the sensor that has produced the observation is denoted ψ_k. Time between observation $k-1$ and observation k is denoted $\delta_k = t_k - t_{k-1}$, the observations are illustrated with the circles in Figure 2. The CTMM $\boldsymbol{x}(t)$ is representing the state of the system and is sampled each time an observation is produced, to form an inhomogenous discrete time Markov model (DTMM) $\boldsymbol{x}_k$, where $\boldsymbol{x}_k = \boldsymbol{x}(t_k)$.

The HMM consists of two stochastic processes; the hidden process $\boldsymbol{x}_k$, and an observable process $\boldsymbol{y}_k$ that depends on $\boldsymbol{x}_k$. The relation between $\boldsymbol{x}_k$ and $\boldsymbol{y}_k$ is described by a set of observation probability matrices $Q = \{Q^\psi\}$, one for each sensor. The observation probability matrix that corresponds to sensor ψ is denoted $Q^\psi = (q_i^\psi(m))$, where $q_i^\psi(m) = P(\boldsymbol{y}_k = m | \boldsymbol{x}_k = i, \psi_k = \psi)$, $q_i^\psi(m)$ is the probability that observation symbol m is observed, given that the system is in state i and the observation is prodced by sensor ψ.

Since observations are received at irregular intervals, the running transition probabilities $p_{ij}^k = P(x(t+\delta_k) = j | x(t) = i)$ depend on the time since last observation δ_k, and have to be calculated each time an observation is received

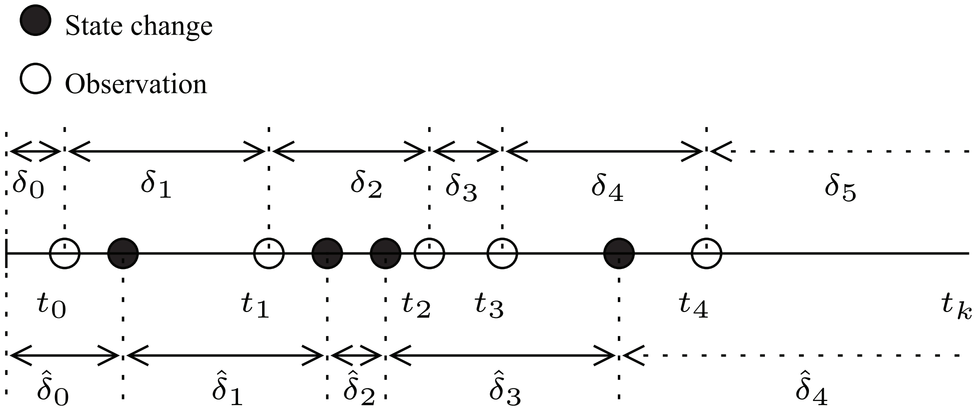


Figure 2. Observations and state changes

from one of the sensors. The running transition probability matrix $P_k = (p_{ij}^k)$ can be derived from Kolmogorov's equations [Ros03] as follows

$$P_k = e^{\Lambda \delta_k}. \tag{4}$$

For large state spaces this calculation can be quite expensive, but in our case the state space is small, and the calculations inexpensive.

3.3 Aggregating Observations and Finding the State Probabilities

The aggregation of observations from different sensors is done by having one common state distribution γ_k that is updated for each new observation received from one of the sensors. Algorithm 4 is used to update the current state distribution γ_{k-1}, based on the following inputs: k the observation index, $\gamma = \gamma_{k-1}$ current state distribution, $\psi = \psi_k$ the index of the sensor that has produced the current observation, $y = y_k$ the current observation, and $\delta = \delta_k$ the time between the current observation and last observation.

In addition to the dynamic variables listed above, the following parameters are assumed to be available for the algorithm: Λ the transition rates, π the initial state distribution, and Q the set of observation probability matrices.

The execution of Algorithm 4 proceeds as follows : first the transition probability matrix P_k is calculated based on time since last observation, then the observation probability matrix that corresponds to the sensor that has produced the observation is selected, finally the state distribution is updated. The algorithm is based on Equation 19 and Equation 27 in [Rab90], but use scaling as described in [Rab90] to avoid underflow. The novelty of this algorithm is the calculation of a new transition matrix P_k for each observation and the use of a separate observation matrix for each sensor.

This corresponds to using a new HMM for each observation, where the initial distribution is γ_{k-1}, the transition probability is P_k, and the observation probability matrix is Q^{ψ}. This means that this new HMM is updated only once based the observation received from sensor ψ, the observation probability matrix belonging to this specific sensor, and the time since last observation δ_k.

Algorithm 4 is based on dynamic programming and uses a set of temporary variables, during the processing of observation y_k the value stored in $\alpha(i)$ will represent the following probabilities $\alpha(i) = P(Y_k, \boldsymbol{x}_k = s_i)$, known as the forward variables. By using dynamic programming in the estimation of γ the complexity of an update is reduced from $O(2kN^k)$ (for a straight forward calculation) to $O(N^2)$.

Algorithm 4 Update state probability distribution

```
Require: $k, \gamma, \psi, y, \delta$
  $P_k \leftarrow e^{\Lambda\delta}$
  $Q \leftarrow Q^{\psi}$
  if $k = 1$ then
    for $i = 1$ to $N$ do
      $\alpha(i) \leftarrow q_i(y)\pi_i$
      $\gamma(i) \leftarrow \frac{q_i(y)\pi_i}{\sum_{j=1}^{N} q_j(y)\pi_j}$
    end for
  else
    for $i = 1$ to $N$ do
      $\alpha(i) \leftarrow q_i(y) \sum_{j=1}^{N} \gamma(j) p_{ji}^{k}$
    end for
    $\gamma \leftarrow \frac{\alpha}{\sum_{j=1}^{N} \alpha(j)}$
  end if
  return $\gamma$
```

3.4 Intrusion Frequency Metric

An interesting security measure which will be used in the risk assessment part of the proposed IPS is mean time to next security failure (MTNSF). This metric is estimated by using the state probability distribution γ and a modified version of the transition probability matrix P.

In order to do the MTNSF analysis we partition the statespace S into two subsets; S_G the set of good states, and S_F the set of failed states, such that $S = S_G \cup S_F$ and $\varnothing = S_G \cap S_F$. Then all failed states are made absorbing, by setting all transition probabilities out from the failed states to zero, letting all the good states become transient.

The new transition probability matrix is called $\hat{P} = (\hat{p}_{ij})$ where

$$\hat{p}_{ij} = \begin{cases} p_{ij} & \text{if } s_i \in S_G \\ 1 & \text{if } s_i \in S_F \wedge i = j \\ 0 & \text{if } s_i \in S_F \wedge i \neq j \end{cases} .$$

The idea is to estimate mean time to absorption given the distribution γ estimated by the HMM as initial distribution. We propose to use mean time to absorption as a security measure and call it mean time to next security failure (MTNSF).

The first step in the MTNSF analysis is to estimate the average number of times each of the transient states S_G is visited before the DTMC reaches one of the absorbing states S_F. This number is called w_i and can be estimated as follows

$$w_i = \gamma(i) + \sum_{j \in S_G} w_j \hat{p}_{ij}, \; s_i \in S_G. \tag{5}$$

More detail on the estimation of w_i can be found in [MVT02]. By applying Equation 5 on our model represented by the transition probability matrix in Equation 2, a set of linear equations are formed. By solving these equations, we get

$$\begin{aligned} w_1 &= \frac{\gamma(1)(1 + p_a\tilde{p}_s) + \gamma(2)\tilde{p}_a + \gamma(3)\tilde{p}_a\tilde{p}_s}{p_a(2 - p_s)} \\ w_2 &= \frac{\gamma(1) + \gamma(2) + \gamma(3)\tilde{p}_s}{p_a(2 - p_s)} \\ w_3 &= \frac{(\gamma(1) + \gamma(2) + \gamma(3))\tilde{p}_s}{(2 - p_s)} \end{aligned} \tag{6}$$

MTNSF can now be estimated by summing the product of w_i multiplied by mean state occupation time h_i for each of the transient states S_G, giving $MTNSF = \sum_{i \in S_G} w_i h_i$. The intrusion frequency f can now be estimated by inverting MTNSF,

$$f = (MTNSF)^{-1} = \frac{1}{\sum_{i \in S_G} w_i h_i}. \tag{7}$$

Based on observation y the distribution γ is estimated and then used to estimate f, which is the input to the online fuzzy risk assessment as described in [HAK07].

4. IPS Architecture

This section describes the IPS originally proposed in [HAK07] with more focus on specific types of IDS sensors.

The IPS architecture is illustrated in Figure 3.

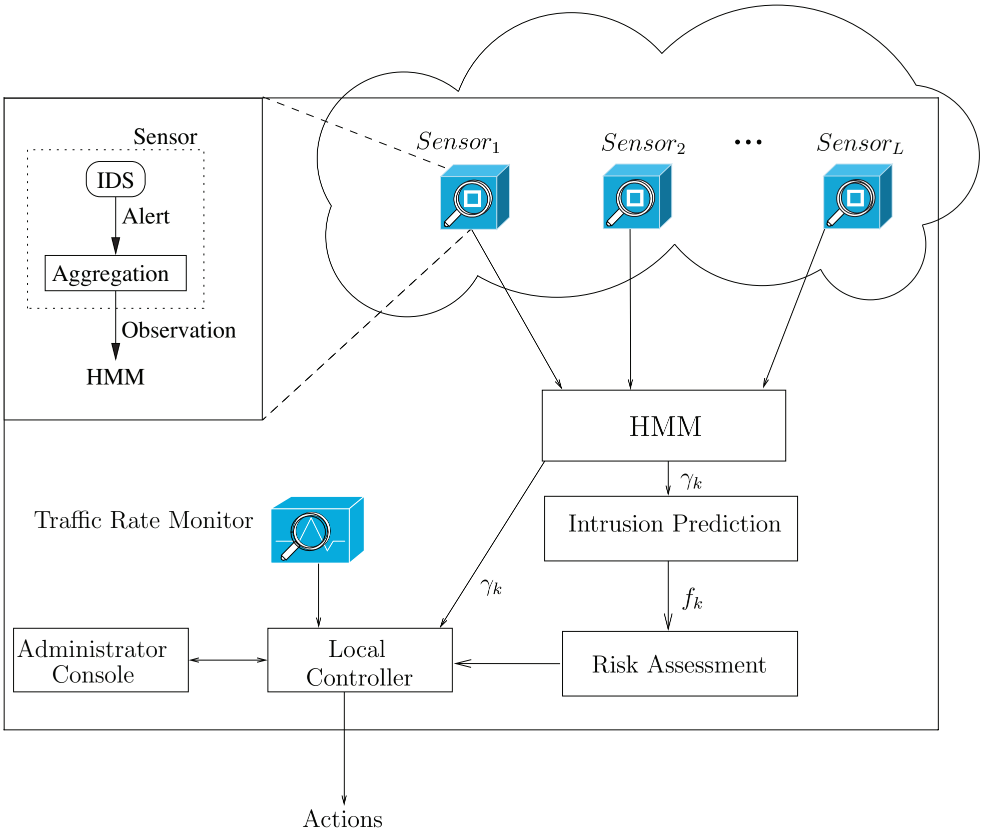


Figure 3. Architecture of the IPS

4.1 Alert Handling

The different sensors produce output in the form of alerts, this output is then transformed into observations that go into the HMM, as described in Section 3.1. Observations are assumed to be independent given the security state, this means that data used for producing one observation should not be used for the next observation. We assume that the format of the IDS output, after preprocessing, is given in the XML based IDMEF (Intrusion Detection Message Exchange Format) [DCF07], which is an emerging standard format for IDS messages proposed by the IETF.

4.2 Prediction and Prevention Mechanisms

For each observation the HMM will update the state probability distribution, as described in Section 3.3, and calculate the intrusion frequency metric as described in Section 3.4. The intrusion frequency will then be input to the risk assessment component that uses this value in the calculation of a risk value, as described in [HAK07]. The local controller receives all these values, if one or more of them reaches a predetermined threshold value, like for instance if the probability of being in the N state is less than 0.5, the most serious alerts will be output on the administrator console allowing for manual and automatic prevention mechanisms. The local controller will then take actions to prevent the attack by reconfiguration of firewalls or termination of user sessions or processes, as specified by predetermined rules. These rules are derived from the organization's security policies, one approach for doing this can be found in [DTBCC06].

4.3 Visualization

The traffic rate monitor gives additional information to the local controller about abnormal traffic rates, that can be a strong indication of malicious activity. After the processing of observations the HMM has an estimated probability distribution over the security states, this is visualized at the administrator console by graphs indicating the probability of being in a compromised state. In addition to this the output of the risk assessment is displayed as a risk value, together with the intrusion frequency and the traffic rates. If the system is likely to be in an attacked state according to the calculated state probability distribution, all serious alerts will also be displayed on the administrator console.

5. Case study

In this section we will illustrate our approach by a case study of a small computer network. We construct an attack scenario and show how the proposed measures can be calculated.

5.1 IDS Sensors

The specific IDS sensors employed in our case study architecture, illustrated in Figure 4, are network based IDS (NIDS), host based IDS (HIDS) and firewalls. Real-time detection and response is made possible by the combination of firewalls and IDS that give input to the local controller, which can take necessary actions to prevent attacks.

We need to have both NIDS and HIDS as they complement each other and both types are considered a requirement when monitoring a network for security intrusions. NIDS look for attack signatures in raw network packets, based on pattern, frequency or anomalies. HIDS monitor system specific logs, critical configuration files and other system components such as key executables. User and file access activity can be monitored by a HIDS in order to detect events not compliant with the organization's security policy, user specific properties like disk space utilization or abnormal user activity can also be detected.

In this small case study we will use observations from four different sensors in the network, a HIDS on the web server, a HIDS on the database server, a

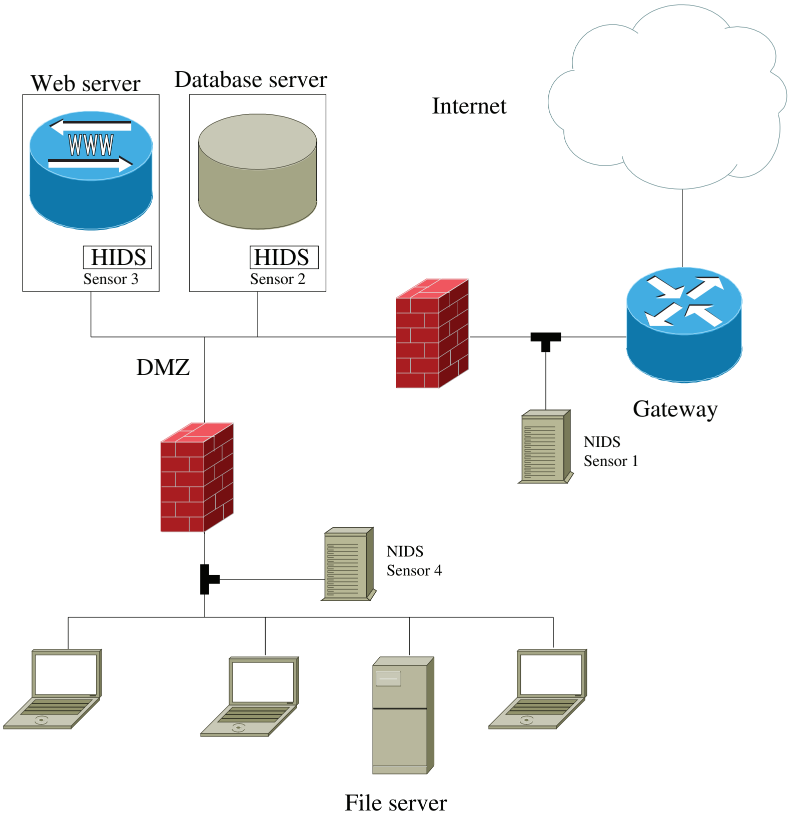


Figure 4. Network Architecture of the IPS case study

NIDS placed between the firewall and the production network, and a NIDS placed at the network perimeter. In a real working environment more sensors would of course be used, but for illustrative purposes we limit the number of sensors to get smaller matrices and a clearer presentation of the mathematical analysis.

The alerts are assumed to be in IDEMF format, and we filter the alerts according to the impact severity description, i.e. *info*, *low*, *medium* and *high*, which are the permitted values of the severity attribute of the impact class as described in [DCF07]. We do not consider the *info* type of alerts as observations in our model as these are not alerts per se, but only represent informational activity.

Intuitively the low severity observations will indicate the IA state, the medium severity observations indicate the IP state and the high severity observations indicate the SA state.

5.2 Parameters

The observation probability matrices model the confidence and trustworthiness of the sensors. A NIDS sensor located outside the firewall at the network perimeter will naturally produce a lot of alerts that could be considered as noise, while we expect a sensor inside the firewall to produce less and more correct alerts. We also assume that the web server is a more attractive target for attackers than the database, so the HIDS of the web server will produce more observations and more low severity alerts than the database HIDS. This is reflected in the Q matrices.

$$Q^1 = \begin{pmatrix} 0.80 & 0.19 & 0.01 \\ 0.80 & 0.19 & 0.01 \\ 0.80 & 0.15 & 0.05 \\ 0.90 & 0.05 & 0.05 \end{pmatrix}, Q^2 = \begin{pmatrix} 0.80 & 0.19 & 0.01 \\ 0.50 & 0.49 & 0.01 \\ 0.38 & 0.60 & 0.02 \\ 0.10 & 0.30 & 0.60 \end{pmatrix}$$

$$Q^3 = \begin{pmatrix} 0.80 & 0.19 & 0.01 \\ 0.60 & 0.39 & 0.01 \\ 0.45 & 0.50 & 0.05 \\ 0.10 & 0.35 & 0.55 \end{pmatrix}, Q^4 = \begin{pmatrix} 0.70 & 0.20 & 0.10 \\ 0.60 & 0.30 & 0.10 \\ 0.20 & 0.70 & 0.10 \\ 0.10 & 0.20 & 0.70 \end{pmatrix}$$

The transition probability matrix P and the state occupation times H have been estimated based on data from the high-interaction honeypot described in [ANK$^+$06]. In a real life setting we suggest that these parameters are estimated by security experts possibly based on data from a honeypot deployed inside the same network as the network that is to be monitored. Of course this honeypot needs to be safely separated from the production network by a honeywall. A honeywall should implement data control, data capture, data analysis and data collection. Data control is necessary to prevent the honeynet being used as a platform for attacking or harming other systems, and could be implemented by restricting the number of outgoing connections or by bandwidth limitations.

As the authors of [ANK$^+$06] point out, attacks on internet servers are often carried out in two steps. The preliminary step is the automated vulnerability scanners, that search for known vulnerabilities and typically perform brute force dictionary attacks, in our model this corresponds to the N state. When an attacking script discovers a vulnerability in the system a more directed investigation starts, for instance a password guessing attempt, this would correspond to the IA state. Several days after a successful password guessing the first step of the real intrusion takes place when a person logs on to the server using this password, this would correspond to the IP state. After having changed the password of the compromised user account, the attacker will now possibly proceed with trying to install some malware or tools to be used for either launching attacks on other networks or to take complete control of the server by becoming root. Depending on the skills of the attacker and the security of the server the attacker will succeed in becoming root, if this happens the server is completely compromised and we have entered the SA state in our model.

As can be seen from the estimated probabilities in the P matrix from Equation 2

$$P = \begin{pmatrix} 0.00 & 1.00 & 0.00 & 0.00 \\ 0.91 & 0.00 & 0.09 & 0.00 \\ 0.00 & 0.83 & 0.00 & 0.17 \\ 0.80 & 0.00 & 0.20 & 0.00 \end{pmatrix},$$

the probability that an attacker, after first performing a vulnerability scan, succeeds with an intrusion attempt is $p_a = 0.09$, this corresponds to the ratio of successful dictionary attacks observed in the honeypot study in [ANK$^+$06]. The probability of a successful attack $p_s = 0.17$ is the share of the attackers succeeding in becoming root after the initial user account intrusion. We then assume that once a successful attack has happened and been detected, the probability that all security vulnerabilities of the system have been removed to be $p_r = 0.80$, i.e. all compromised passwords on user accounts have been changed.

The mean state occupation times h_2 and h_3 in

$$H = \begin{pmatrix} 10.00 & 3.90 & 7.20 & 1.00 \end{pmatrix}$$

are based on Table 2 in [ANK$^+$06]. We have used the average duration in days between creation and first successful connection to the user accounts as h_2, and the average duration between first successful connection and first intrusion as h_3. The mean occupation time in the normal state is estimated to be ten days and we assume that a successful compromise will be discovered within one day.

5.3 Attack Scenario

We will now describe an attack scenario on the small computer network as shown in Figure 4. We have created observations based on a scenario where we assume an attack on the web server to take place on the sixth day after employment.

The transition rate matrix Λ is calculated by Equation 3 taking P and H as arguments. Algorithm 4 is then used on each of the 37 observations $y_1, \ldots, y_{37}$ shown in Figure 5(a), to estimate the state distributions $\gamma_1, \ldots, \gamma_{37}$ shown in Figure 6. For each state distribution γ_k the corresponding intrusion frequency f_k is estimated using Equation 6 and Equation 7.

The observations are denoted by L (low), M (medium) and H (high) and are illustrated in Figure 5(a). The NIDS at the network perimeter (sensor 1) triggers most of the alerts, as we consider this sensor to be the most noisy. The HIDS at the web server (sensor 3) is assumed to produce more correct alerts, so it is only triggering serious alerts at the time of intrusion. The database (sensor 2) and the NIDS inside the production network (sensor 4) are also assumed to not produce a lot of false alerts, so they only give some less serious observations from time to time in this scenario.

At startup we assume the system to be in the normal state N, as can be seen from Figure 6. But as observations indicating low severity alerts, for instance scanning alerts are received, the probability of being in the IA state increases. Since the trustworthiness of sensor 1 is modeled to be quite low in the Q^1 matrix and the state occupation time for state N is relatively long, the early observations indicating attacks from sensor 1 do not effect the state probability of the IA state considerably.

We can however see a clear indication of attack activity just before day 6, when the more trustworthy sensor 3 starts to contribute with observations, together with more observations of medium severity coming from sensor 1. Just before the attack takes place, the probability of the IP state also increases, and as the high severity alerts starts to come from sensor 3 the probability of being in state SA quickly rises.

The intrusion frequency illustrated in Figure 5(b) also starts to rise before the attack takes place, hence giving input to the risk assessment that indicates attack activity. The intrusion prevention mechanisms will be initiated once the risk value, intrusion frequency or the state probabilities reaches certain thresholds. For instance if the threshold for state IA was set to 0.4, we can see from Figure 6 that prevention mechanisms would be triggered just before day six.

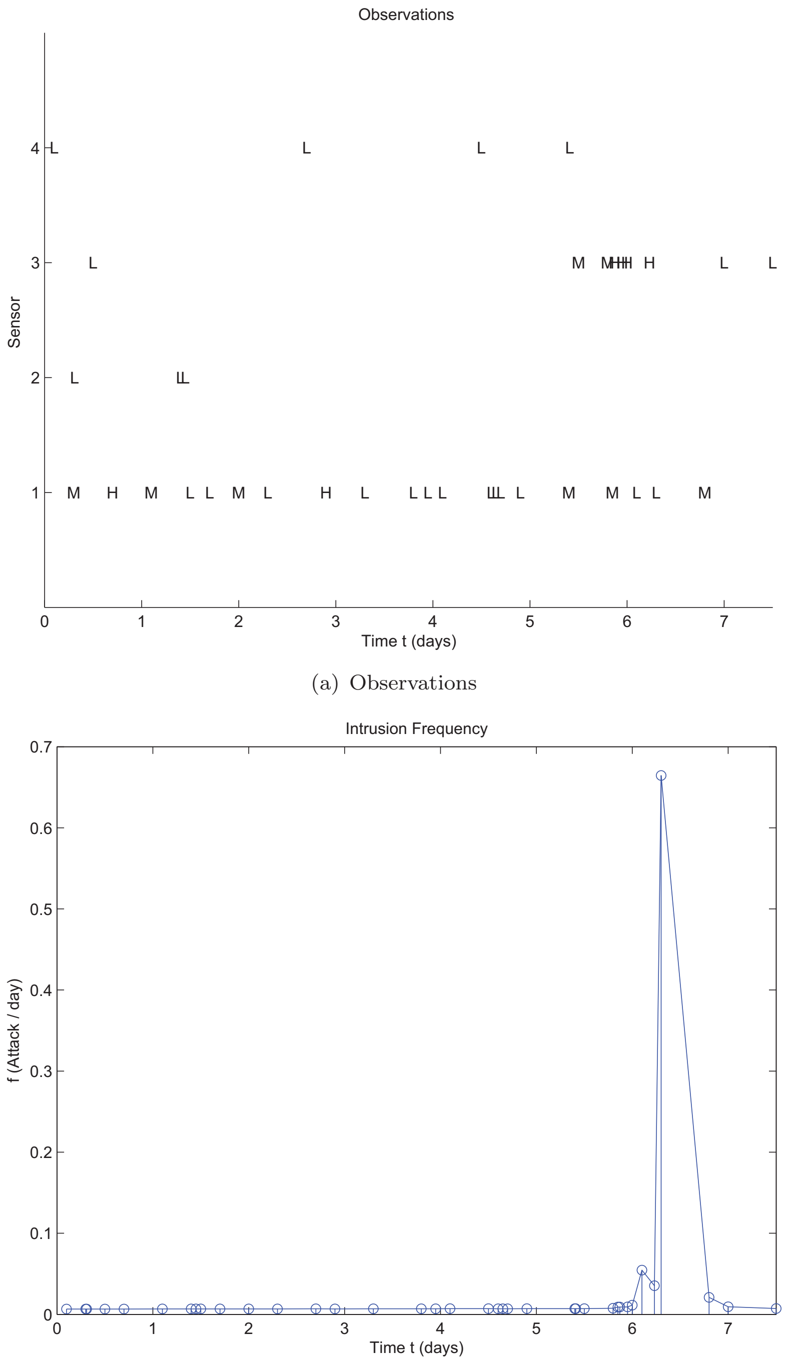
6. Conclusions and Future Work

In this paper we have described a framework for prediction and prevention of attacks against computer networks. Hidden Markov modeling of IDS sensors and security measures based on stochastic modeling techniques usually applied by the dependability community have been presented. The HMM approach effectively minimizes the amount of IDS alerts that needs to be handled manually by a network administrator. The concept of security states of the system facilitates the monitoring and visualization of network security. By using thresholds for the probability of being in a compromised state, attack countermeasures and prevention mechanisms can automatically be triggered before the network is entering a vulnerable state.

This work gives a theoretical basis for a future implementation of the IPS. Further security considerations concerning the secure communication of sensor information have not been treated in this paper, but will need to be considered in an actual implementation. The model needs to be tested in a real network setting, in order to do this parameters need to be estimated. We suggest that some parameters could be estimated based on data from a honeypot deployed inside the same network as the network that is to be monitored. Also input from security experts and experienced network administrators should be considered when deciding the initial model parameters. The network parameters should be updated on a regular basis, in particular whenever the network changes its security configurations as security updates and patches are installed. One method of reestimating the parameters could be by using a learning algorithm like the expectation-maximization or Baum-Welch method.

The aggregation of IDS alerts into the HMM should result in fewer false positives, but could potentially lead to an increase of false negatives, the amount of false negatives and positives would in our model depend on the parameters. These dependencies should be investigated further by applying our modeling technique on real life data sets.

In our future work we would like to extend the model to also include availability, by using a separate HMM with DoS alerts as observations. We will also consider increased granularity of the model by including more states, this will increase model complexity, so a trade-off analysis between the number of states and complexity of calculations will also be required. We also plan to do experiments with real life data sets to investigate the validity of the assumption about exponentially distributed state occupation times. The use of semi-Markov modeling to allow for other probability distributions will also be considered in our future work.



(a) Observations

(b) Intrusion frequency plotted against time

Figure 5. Observations and estimated intrusion frequency.

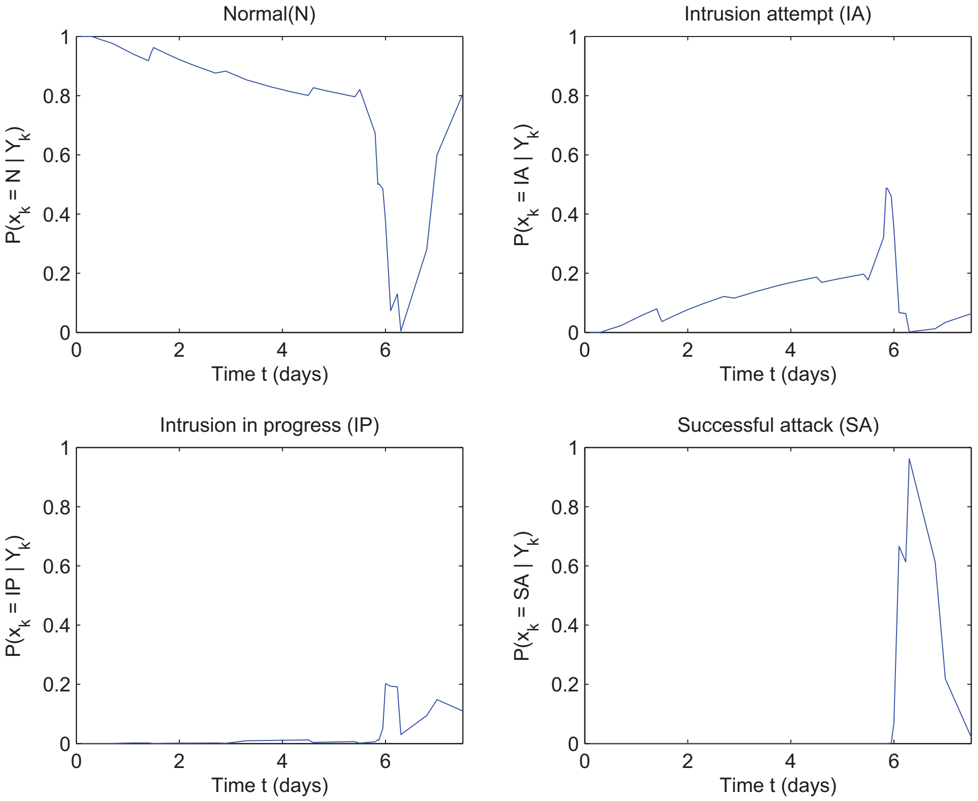


Figure 6. State probability distributions

References

[ANK$^+$06] Eric Alata, V Nicomette, M Kaâniche, Marc Dacier, and M Herrb. Lessons learned from the deployment of a high-interaction honeypot. In *EDCC'06, 6th European Dependable Computing Conference, October 18-20, 2006, Coimbra, Portugal*, Oct 2006.

[ÅSH$^+$05] André Årnes, Karin Sallhammar, Kjetil Haslum, Tønnes Brekne, Marie Elisabeth Gaup Moe, and 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.

[ÅVVK06] André Årnes, Fredrik Valeur, Giovanni Vigna, and Richard A. Kemmerer. Using hidden markov models to evaluate the risk of intrusions. In *Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection, RAID 2006, Hamburg, Germany, September 20 – 22, 2006.*, September 2006.

[BBMGPT04] 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.

[DCF07] H. Debar, D. Curry, and B. Feinstein. The intrusion detection message exchange format (IDMEF), 2007. IETF RFC 4765.

[DTBCC06] H. Debar, Y. Thomas, N. Boulahia-Cuppens, and F. Cuppens. Using contextual security policies for threat response. In *DIMVA '06: Proceedings of the 3th International Conference on Detection of Intrusions and Malware & Vulnerability Assessment, LNCS 4064*, pages 109–128. Springer-Verlag, 2006.

[DW01] Hervé Debar and Andreas Wespi. Aggregation and correlation of intrusion-detection alerts. In *RAID '00: Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection*, pages 85–103, London, UK, 2001. Springer-Verlag.

[HAK07] Kjetil Haslum, Ajith Abraham, and Svein Knapskog. Dips: A framework for distributed intrusion prediction and prevention using hidden markov models and online fuzzy risk assessment. In *Third International Symposium on Information Assurance and Security, IEEE Computer Society press*, volume I, pages 183–188, 2007.

[JO97] Erland Jonsson and Tomas Olovsson. A quantitative model of the security intrusion process based on attacker behavior. *IEEE Transactions on Software Engineering*, 23(4):235–245, 1997.

[Jul01] K. Julisch. Mining alarm clusters to improve alarm handling efficiency. In *ACSAC '01: Proceedings of the 17th Annual Computer Security Applications Conference*, page 12, Washington, DC, USA, 2001. IEEE Computer Society.

[KAN$^+$06] M Kaâniche, E Alata, V Nicomette, Yves Deswarte, and Marc Dacier. Empirical analysis and statistical modeling of attack processes based on honeypots. In *WEEDS 2006 - Workshop on empirical evaluation of dependability and security (in conjunction with the international conference on dependable systems and networks, DSN 2006), June 25 - 28, 2006, Philadelphia,USA*, Jun 2006.

[KL06] Rahul Khanna and Huaping Liu. System approach to intrusion detection using hidden markov model. In *IWCMC '06: Proceeding of the 2006 international conference on Communications and mobile computing*, pages 349–354, New York, NY, USA, 2006. ACM Press.

[KL07] Rahul Khanna and Huaping Liu. Distributed and control theoretic approach to intrusion detection. In *IWCMC '07: Proceedings of the 2007 international conference on Wireless communications and mobile computing*, pages 115–120, New York, NY, USA, 2007. ACM.

[LBF$^+$93] B. Littlewood, S. Brocklehurst, N. Fenton, P. Mellor, S. Page, D. Wright, J. Dobson, J. McDermid, and D. Gollmann. Towards operational measures of computer security. *Journal of Computer Security*, 2:211–229, 1993.

[MVT02] B.B. Madan, K. Vaidyanathan, and K.S. Trivedi. Modeling and quantification of security attributes of software systems. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN'02)*, 2002.

[NST04] David M. Nicol, William H. Sanders, and Kishor S. Trivedi. Model-based evaluation: From dependability to security. *IEEE Transactions on Dependable and Secure Computing*, 01(1):48–65, 2004.

[Rab90] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Readings in speech recognition*, pages 267–296, 1990.

[Ros03] Sheldon M. Ross. *Introduction to Probability Models*. Academic Press, New York, 8th edition, 2003.

[SHK07] Karin Sallhammar, Bjarne E. Helvik, and Svein J. Knapskog. A framework for predicting security and dependability measures in real-time. *International Journal of Computer Science and Network Security (IJCSNS)*, 7(3):169–183, 2007.

[VVKK04] Fredrik Valeur, Giovanni Vigna, Christopher Kruegel, and Richard A. Kemmerer. A comprehensive approach to intrusion detection alert correlation. *IEEE Transactions on Dependable and Secure Computing*, 1(3):146–169, 2004.

[WMT03] Dazhi Wang, Bharat B. Madan, and Kishor S. Trivedi. Security analysis of sitar intrusion tolerance system. In *SSRS '03: Proceedings of the 2003 ACM workshop on Survivable and self-regenerative systems*, pages 23–32, New York, NY, USA, 2003. ACM.

Bibliography

[AGMV07] A. Abraham, C. Grosan, and C. Martin-Vide. Evolutionary design of intrusion detection programs. *International Journal of Network Security*, 4(3):328–339, 2007.

[AJTH07] A. Abraham, R. Jain, J. Thomas, and S.Y. Han. D-scids: Distributed soft computing intrusion detection systems. *Journal of Network and Computer Applications, Elsevier Science*, 30(1):81–98, 2007.

[ALRL04] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 01(1):11–33, 2004.

[ANK$^+$06] Eric Alata, V Nicomette, M Kaâniche, Marc Dacier, and M Herrb. Lessons learned from the deployment of a high-interaction honeypot. In *EDCC'06, 6th European Dependable Computing Conference, October 18-20, 2006, Coimbra, Portugal*, Oct 2006.

[ÅSH$^+$05] André Årnes, Karin Sallhammar, Kjetil Haslum, Tønnes Brekne, Marie Elisabeth Gaup Moe, and 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.

[ÅSHK06] André Årnes, Karin Sallhammar, Kjetil Haslum, and Svein Johan Knapskog. Real-time risk assessment with network sensors and hidden markov model. In *Proceedings of the 11th Nordic Workshop on Secure IT-systems (NORDSEC 2006)*, Oct 2006.

[ÅVVK06] André Årnes, Fredrik Valeur, Giovanni Vigna, and Richard A. Kemmerer. Using hidden markov models to evaluate the risk of intrusions. In *Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection, RAID 2006, Hamburg, Germany, September 20 – 22, 2006.*, September 2006.

[BBMGPT04] 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.

[BGFI$^+$98] J. S. Balasubramaniyan, J. O. Garcia-Fernandez, D. Isacoff, E. Spafford, and D. Zamboni. An architecture for intrusion detection using autonomous agents. In *Proceedings of the 14th Annual Computer Security Applications Conference*, page 13. IEEE Computer Society, 1998.

[Cen90] National Computer Security Center. *Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria.* NSA/NCSC Rainbow Series. 1990.

[CHSP00] Curtis A. Carver Jr., John M.D. Hill, John R. Surdu, and Udo W. Pooch. A methodology for using intelligent agents to provide automated intrusion response. In *Proceedings of the IEEE Workshop on Information Assurance and Security*, 2000.

[CS96] Mark Crosbie and Gene Spafford. Defending a computer system using autonomous agents. In *In Proceedings of the 18th National Information Systems Security Conference*, 1996.

[DCF05] H. Debar, D. Curry, and B. Feinstein. Intrusion detection message exchange format (IDMEF) – Internet-Draft, 2005.

[DCF07] H. Debar, D. Curry, and B. Feinstein. The intrusion detection message exchange format (IDMEF), 2007. IETF RFC 4765.

[DDW99] Hervé Debar, Marc Dacier, and Andreas Wespi. Towards a taxonomy of intrusion-detection systems. *Computer Networks*, 31(8):805–822, 1999.

[Def83] Department Of Defense. *CSC-STD-001-83: Department of Defense Trusted Computer System Evaluation Criteria [DOD 5200.28].* Defense Department Rainbow Series. 1983.

[DTBCC06] H. Debar, Y. Thomas, N. Boulahia-Cuppens, and F. Cuppens. Using contextual security policies for threat response. In *DIMVA '06: Proceedings of the 3th International Conference on Detection of Intrusions and Malware & Vulnerability Assessment, LNCS 4064*, pages 109–128. Springer-Verlag, 2006.

[DW01] Hervé Debar and Andreas Wespi. Aggregation and correlation of intrusion-detection alerts. In *RAID '00: Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection*, pages 85–103, London, UK, 2001. Springer-Verlag.

[GGPW$^+$01] Fengmin Gong, Katerina Goseva-Popstojanova, Feiyi Wang, Rong Wang, Kalyanaraman Vaidyanathan, Kishor Trivedi, and Balamurugan Muthusamy. Characterizing intrusion tolerant systems using a state transition model. In *DARPA Information Survivability Conference and Exposition (DISCEX II)*, volume 2, 2001.

[GK04] Ashish Gehani and Gershon Kedem. Rheostat: Real-time risk management. In *Recent Advances in Intrusion Detection: 7th International Symposium, RAID 2004, Sophia Antipolis, France, September 15-17, 2004. Proceedings*, pages 296–314. Springer, 2004.

[GSW03] Fei Gao, Jizhou Sun, and Zunce Wei. The prediction role of hidden markov model in intrusion detection. In *IEEE CCECE 2003: Canadian Conference on Electrical and Computer Engineering*, volume 2, pages 893–896, 2003.

[HAK07] Kjetil Haslum, Ajith Abraham, and Svein Knapskog. Dips: A framework for distributed intrusion prediction and prevention using hidden markov models and online fuzzy risk assessment. In *Third International Symposium* *on Information Assurance and Security, IEEE Computer Society press*, volume I, pages 183–188, 2007.

[HAK08] Kjetil Haslum, Ajith Abraham, and Svein Knapskog. Fuzzy online risk assessment for distributed intrusion prediction and prevention systems. In *Tenth International Conference on Modeling and Simulation, IEEE Computer Society press*, volume I, page in press, 2008.

[HWH$^+$03] Guy Helmer, Johnny S. K. Wong, Vasant G. Honavar, Les Miller, and Yanxin Wang. Lightweight agents for intrusion detection. *J. Syst. Softw.*, 67(2):109–122, 2003.

[ISO05] ISO/IEC JTC 1, SC 27. *ISO/IEC 15408 Part 1-3:2005, Common Criteria for Information Technology Security Evaluation.* ISO, Geneva, Switzerland, 2005.

[Jan93] J.-S.R. Jang. Anfis: adaptive-network-based fuzzy inference system. *Systems, Man and Cybernetics, IEEE Transactions on*, 23(3):665–685, May/Jun 1993.

[JO97] Erland Jonsson and Tomas Olovsson. A quantitative model of the security intrusion process based on attacker behavior. *IEEE Transactions on Software Engineering*, 23(4):235–245, 1997.

[Jon06] Jack Jones. An introduction to factor analysis of information risk (fair). *Norwich Journal of Information Assurance*, 2(1):67, 2006.

[Jul01] K. Julisch. Mining alarm clusters to improve alarm handling efficiency. In *ACSAC '01: Proceedings of the 17th Annual Computer Security Applications Conference*, page 12, Washington, DC, USA, 2001. IEEE Computer Society.

[KAN$^+$06] M Kaâniche, E Alata, V Nicomette, Yves Deswarte, and Marc Dacier. Empirical analysis and statistical modeling of attack processes based on honeypots. In *WEEDS 2006 - Workshop on empirical evaluation of dependability and security (in conjunction with the international conference on dependable systems and networks, DSN 2006), June 25 - 28, 2006, Philadelphia,USA*, Jun 2006.

[Ken99] Kristopher Kendall. A database of computer attacks for the evaluation of intrusion detection systems. Master's thesis, MIT, USA, June 1999.

[KKM$^+$05] Christopher Krügel, Engin Kirda, Darren Mutz, William K. Robertson, and Giovanni Vigna. Polymorphic worm detection using structural information of executables. In Alfonso Valdes and Diego Zamboni, editors, *RAID*, volume 3858 of *Lecture Notes in Computer Science*, pages 207–226. Springer, 2005.

[KL06] Rahul Khanna and Huaping Liu. System approach to intrusion detection using hidden markov model. In *IWCMC '06: Proceeding of the 2006 international conference on Communications and mobile computing*, pages 349–354, New York, NY, USA, 2006. ACM Press.

[KL07] Rahul Khanna and Huaping Liu. Distributed and control theoretic approach to intrusion detection. In *IWCMC '07: Proceedings of the 2007* *international conference on Wireless communications and mobile computing*, pages 115–120, New York, NY, USA, 2007. ACM.

[Lab00] Lincoln Laboratory. Lincoln laboratory scenario (ddos) 1.0, 2000.

[LBF$^+$93] B. Littlewood, S. Brocklehurst, N. Fenton, P. Mellor, S. Page, D. Wright, J. Dobson, J. McDermid, and D. Gollmann. Towards operational measures of computer security. *Journal of Computer Security*, 2:211–229, 1993.

[MA75] E.H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7(1):1–13, 1975.

[Mil] John A. Miller. Jsim: A java-based simulation and animation environment. `http://chief.cs.uga.edu/~jam/jsim/`.

[MVT02] B.B. Madan, K. Vaidyanathan, and K.S. Trivedi. Modeling and quantification of security attributes of software systems. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN'02)*, 2002.

[NCSD09] Information Technology Laboratory NIST Computer Security Division. National vulnerability database version 2.2, 2009. http://nvd.nist.gov/.

[NST04] David M. Nicol, William H. Sanders, and Kishor S. Trivedi. Model-based evaluation: From dependability to security. *IEEE Transactions on Dependable and Secure Computing*, 01(1):48–65, 2004.

[OD99] Rodolphe Ortalo and Yves Deswarte. Experimenting with quantitative evaluation tools for monitoring operational security. *IEEE Transactions on Software Engineering*, 25:633–650, 1999.

[OMSH03] Dirk Ourston, Sara Matzner, William Stump, and Bryan Hopkins. Applications of hidden markov models to detecting multi-stage network attacks. In *Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS)*, 2003.

[PGR06] Roberto Perdisci, Giorgio Giacinto, and Fabio Roli. Alarm clustering for intrusion detection systems in computer networks. *Engineering Applications of Artificial Intelligence*, 19(4):429 – 438, 2006. Recent Advances in Data Mining.

[PN97] P. A. Porras and P. G. Neumann. EMERALD: Event monitoring enabling responses to anomalous live disturbances. In *Proc. 20th NIST-NCSC National Information Systems Security Conference*, pages 353–365, 1997.

[Rab90] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Readings in speech recognition*, pages 267–296, 1990.

[Ros03] Sheldon M. Ross. *Introduction to Probability Models.* Academic Press, New York, 8th edition, 2003.

[SBD$^+$91] Steven R. Snapp, James Brentano, Gihan V. Dias, Terrance L. Goan, L. Todd Heberlein, Che lin Ho, Karl N. Levitt, Biswanath Mukherjee, Stephen E. Smaha, Tim Grance, Daniel M. Teal, and Doug Mansur. DIDS (distributed intrusion detection system) - motivation, architecture, and an early prototype. In *Proceedings of the 14th National Computer Security Conference*, pages 167–176, Washington, DC, 1991.

[SCCC$^+$96] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle. GrIDS – A graph-based intrusion detection system for large networks. In *Proceedings of the 19th National Information Systems Security Conference*, 1996.

[SCS03] 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 2003.

[SGF02] Gary Stonebumer, Alice Goguen, and Alexis Feringa. Risk management guide for information technology systems, National Institute of Standards and Technology, special publication 800-30, 2002. http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf.

[SHJ$^+$02] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J.M. Wing. Automated generation and analysis of attack graphs. *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pages 273–284, 2002.

[SHK05] Karin Sallhammar, Bjarne E. Helvik, and Svein J. Knapskog. Incorporating attacker behavior in stochastic models of security. In Hamid R. Arabnia, editor, *Security and Management*, pages 79–85. CSREA Press, 2005.

[SHK06] Karin Sallhammar, Bjarne E. Helvik, and Svein J. Knapskog. A game-theoretic approach to stochastic security and dependability evaluation. In *DASC*, pages 61–68. IEEE Computer Society, 2006.

[SHK07] Karin Sallhammar, Bjarne E. Helvik, and Svein J. Knapskog. A framework for predicting security and dependability measures in real-time. *International Journal of Computer Science and Network Security (IJCSNS)*, 7(3):169–183, 2007.

[SKH05] Karin Sallhammar, Svein J. Knapskog, and Bjarne E. Helvik. Using stochastic game theory to compute the expected behavior of attackers. In *SAINT Workshops*, pages 102–105. IEEE Computer Society, 2005.

[Sta04] Standards Australia and Standards New Zealand. AS/NZS 4360: 2004 risk management, 2004.

[Sug85] Michio Sugeno. *Industrial Applications of Fuzzy Control*. Elsevier Science Inc., New York, NY, USA, 1985.

[Ven96] D.E. Ventzas. Kalman filters for dynamic position control of large scale systems. *Industrial Electronics, Control, and Instrumentation, 1996., Proceedings of the 1996 IEEE IECON 22nd International Conference on*, 2:647–652 vol.2, 5-10 Aug 1996.

[VKB01] Giovanni Vigna, Richard A. Kemmerer, and Per Blix. Designing a web of highly-configurable intrusion detection sensors. In *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID 2000)*, pages 69–84, London, UK, 2001. Springer-Verlag.

[VVK03] Giovanni Vigna, Fredrik Valeur, and Richard A. Kemmerer. Designing and implementing a family of intrusion detection systems. *SIGSOFT Softw. Eng. Notes*, 28(5):88–97, 2003.

[VVKK04] Fredrik Valeur, Giovanni Vigna, Christopher Kruegel, and Richard A. Kemmerer. A comprehensive approach to intrusion detection alert correlation. *IEEE Transactions on Dependable and Secure Computing*, 1(3):146–169, 2004.

[WFP99] C. Warrender, S. Forrest, and B. Pearlmutter. Detecting intrusions using system calls: Alternative data models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, 1999.

[WGZ04] Wei Wang, Xiao-Hong Guan, and Xiang-Liang Zhang. Modeling program behaviors by hidden markov models for intrusion detection. *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, 5:2830–2835 vol.5, Aug. 2004.

[WMT03] Dazhi Wang, Bharat B. Madan, and Kishor S. Trivedi. Security analysis of sitar intrusion tolerance system. In *SSRS '03: Proceedings of the 2003 ACM workshop on Survivable and self-regenerative systems*, pages 23–32, New York, NY, USA, 2003. ACM.

[WWT02] Wei Wei, Bing Wang, and Don Towsley. Continuous-time hidden markov models for network performance evaluation. *Performance Evaluation*, 49:129–146, 2002.

[Zad65] L.A. Zadeh. Fuzzy sets. *Info. & Ctl.*, 8:338–353, 1965.