AN ATTACKER INTENTION DISCOVERY LAYER FOR INTRUSION DETECTION SYSTEMS USING HIDDEN MARKOV MODELS

By

JORDAN KIPROP KOSKEI

Bachelor of Technology

Moi University

Eldoret, Kenya

2008

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 2011

AN ATTACKER INTENTION DISCOVERY

LAYER FOR INTRUSION DETECTION

SYSTEMS USING HIDDEN

MARKOV MODELS

Thesis Approved:

Dr. Johnson Thomas

Thesis Adviser

Dr. Subhash Kak

Dr. Michel Toulouse

Dr. Mark E. Payton

Dean of the Graduate College

AKNOWLEDGEMENTS

I would like to express my gratitude to my advisor Dr. Thomas for his valuable guidance and contributions during the entire research work. I am also extremely grateful to Dr. Toulouse and Dr. Kak for serving on my committee and giving invaluable advice especially at the proposal stage. I would also like to express my sincere appreciation to my brothers Titus, Leo and Brian, and parents, Apton and Elizabeth Goren for their unending love and support, financial or otherwise, without which it would be impossible to come this far.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

1. INTRODUCTION

Computer and network security are built on 3 pillars [1]: - Data confidentiality, Integrity and availability. In computer security, networks intrusion is the act of users accessing the network without authorization or authorized users with legitimate access abusing their privileges. Generally, intrusions cause loss of integrity, loss of confidentiality and denial of services (loss of availability) [2].

Intrusion detection is defined as the problem of identifying the intruders as defined above. The most widely deployed intrusion detection systems (IDS) such as Snort [3] can only detect intrusions. However every intruder always has an intention in mind. Intentions in this context are the high level goals of the intruder. While Intrusion detection deals with the problem of identifying intruders, intention discovery is the ability to uncover the intruder's high level goals. Attacker intention discovery provides critical benefits both when the attack is underway and in the post attack phase. When the attack is underway, discovering the attackers current and future intentions aids in deploying more effective counter measures mechanisms such as blocking off some ports on a server. This is important to minimize damage and possibly fend off the attacker. During the post attack period, discovering the full set of the attacker's intentions can ensure more precise disaster recovery efforts and in the hardening of the security systems among others.

In this thesis, we present a novel solution to the problem of discovering a network attacker's intention. We model a known attack scenario as a Hidden Markov Model (HMM)

with variables being among others output from existing IDS such as Snort. As a statistical tool widely used in temporal pattern recognition, the HMM, upon receiving initial training input, can be used to dynamically discover an intruder's current set of intentions and the future intention based on the alerts from the IDS. As explained later, the HMM has a number of advantages over other approaches, but our key motivation in using it is the fact that prior knowledge can be incorporated into the model at design time.

To validate and evaluate the performance characteristics of our method we ran Snort, the most widely used IDS against raw packet data previously collected from a simulated attack scenario. The results indicate that our method can discover an attacker's intention. Further, the results from our experiments provided us with important performance characteristics information.

1.1 Scope of the thesis

In this thesis, we explain the need for attacker intention discovery system and demonstrate how we can implement such systems using our method which involves modeling using HMM and decoding using the Viterbi algorithm, an algorithm which has also been widely applied in other fields such as bio-sequencing, voice recognition and image recognition. We undertake experimentation to validate our method and evaluate performance.

1.2 Thesis organization

Chapter 2 provides the literature review. Here, we review work in intrusion detection systems and focus on the work done on the subject of this thesis, intention discovery in IDS, and the advantages of our approach. Various related concepts are introduced and discussed. Chapter 3 on "Intrusion intention discovery" describes in detail our network intruder plan discovery

system. We illustrate how we can model attacks using HMM. We further discuss its integration into a typical intrusion detection system. In chapter 4, "Experimental setup and simulation", we validate our work. We model our system using the 2000 DARPA Intrusion Detection Scenario-Specific data set to show its effectiveness. Chapter 5 gives a summary of the findings of this work and provides suggestions for future work.

Chapter 2

Literature Review

2. LITERATURE REVIEW

This chapter goes into depth describing previous work related to network security, intrusion detection in general and more specifically intruder intent discovery. The similarities and differences between these papers and the work of this thesis are explained. The purpose of this analysis is to better establish the specific problem that is being addressed by this thesis, how others have solved it and our approach.

2.1 Computer networks and security

Computer networks have become a reality of modern living. One of the largest physical networks is the internet which serves as a platform for millions of networks running on it such as banking networks, social networks, online TV etc. At the heart of any computer infrastructure is security which Pfleeger [4] defines as the protection of the system against threats to confidentiality, integrity and availability. By confidentiality, we mean that network resources can only be accessed by authorized parties. By integrity we mean that network resources can only be modified by authorized parties where modification includes creation, deletion and changing among other operations. By availability we mean that the network resources are accessible to authorized entities in the right format at appropriate times. A resource as described here refers to the main categories of system resources including hardware, software and data. Each of these resources face different types of threats as illustrated in Fig 2.1.

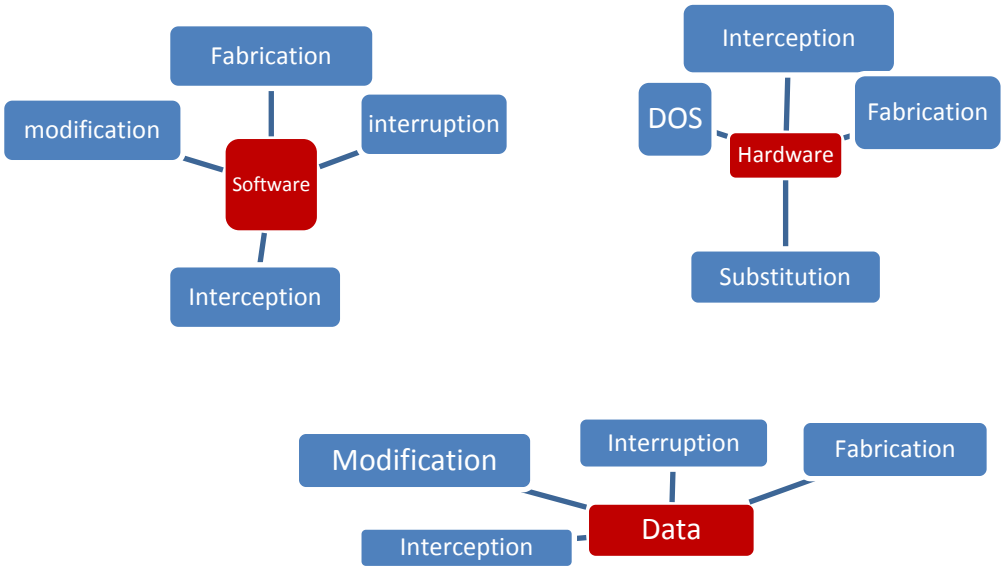


Fig 2.1: Threats that software, hardware and data resources face

With the rapid growth of the internet and internet based services, security has become a serious concern on the internet. Elliot and Fowell [5] point out the fact that actual and perceived security concerns, in particular, are large barriers preventing a more rapid uptake of internet based services.

Pfleeger and Pfleeger [4] further points out 3 fundamental characteristics of computer networks that make them inherently insecure:-

(i) Lack of physical proximity among users.

(ii) Use of insecure shared media.

(iii) Anonymity of users.

The sum effect of these 3 characteristics is that intruders feel very safe while undertaking their activities. Skilled cyber criminals may not leave any footprint behind and hence make their identification near impossible.

2.2 Intrusions

Heady et al [30] describes an intrusion as any set of actions that attempt to compromise the integrity, confidentiality and availability of resources. The problem of intrusion is significant since it leads to loss of confidentiality and/or integrity and/or availability of network resources.

Sundaram [29] classifies intrusions into 6 main types:-

(1) Attempted break-ins, which are detected by typical behavior profiles or violations of security constraints.

(2) Masquerade attacks, which are detected by atypical behavior profiles or violations of security constraints.

(3) Penetration of the security control system, which are detected by monitoring for specific patterns of activity.

(4) Leakage, which is detected by atypical use of system resources.

(5) Denial of service, which is detected by atypical use of system resources.

(6) Malicious use, which is detected by atypical behavior profiles, violations of security constraints, or use of special privileges.

Anderson [6] classified intruders into two types, the external intruders who are unauthorized users of the machines they attack, and internal intruders, who have permission to access the system, but not some portions of it. He further divides internal intruders into intruders who masquerade as another user, those with legitimate access to sensitive data, and the most dangerous type, the clandestine intruders who have the power to turn off their own audit control**.**

2.3 Intrusion detection

Intrusion detection is the monitoring of a network for the purpose of identifying malicious or suspicious events. Intrusion detection is implemented in intrusion detection devices which are essentially sensors that raise an alarm when something specific happen. There are two types of intrusion detection, misuse-based detection and anomaly-based detection.

2.3.1 Misuse/Signature based intrusion detection

A misuse-based detection system is equipped with a database that contains a number of signatures about known attacks [6]. The audit data collected by the IDS is compared with the content of the database and, if a match is found, an alert is generated. Events that do not match any of the attack models are considered as a part of legitimate activities.

James P. Anderson [6] published a study outlining ways to improve computer security auditing and surveillance at customer sites. The original idea behind automated IDS is often credited to him for his paper on “How to use accounting audit files to detect unauthorized access”. This IDS study paved the way for misuse detection for mainframe systems and later for general computing. In his design, the first task was to define what threats existed. Before designing a IDS, it was necessary to understand the types of threats and attacks that could be mounted against computers systems and how to recognized them in an audit data. Fig 2.2shows the typical components of a misuse IDS as proposed by Anderson.

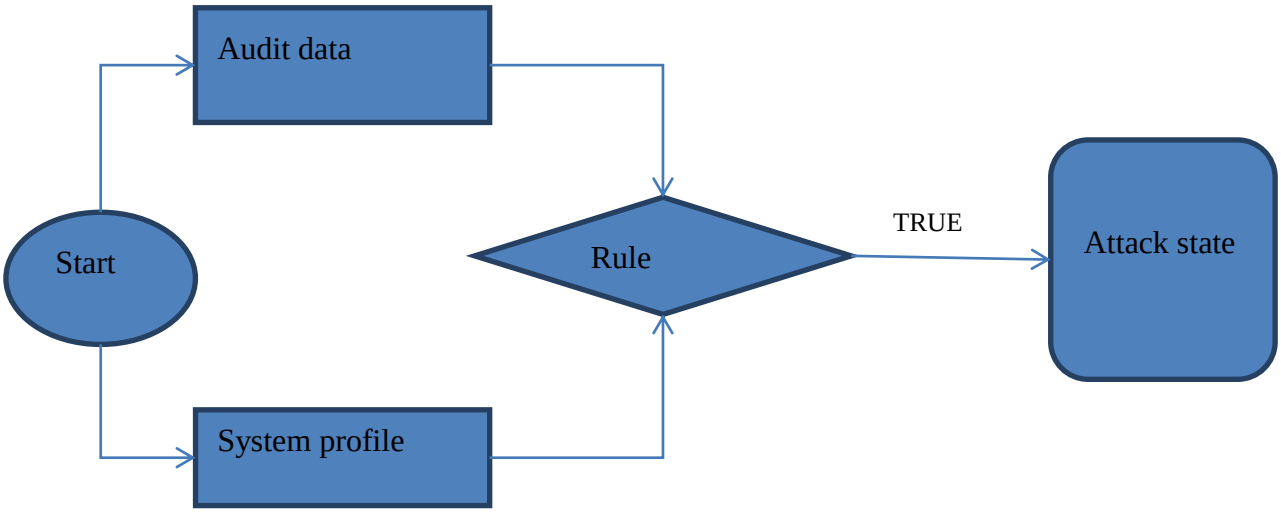


Fig 2.2: Components of a misuse IDS

The main advantage of misuse-based systems is that they usually produce very few false positives. The main disadvantage with misuse based intrusion detection system is that there is a lag between the new threat discovery and the new Signature being applied in IDS for detecting the threat. During this lag time the IDS will be unable to identify the threat.

2.3.2 Anomaly based intrusion detection

Anomaly-based detection is a behavior-based detection method [7]. It is based on the assumption that all anomalous activities are malicious and all the attacks are subset of anomaly activities. By building a model of the normal behavior of the system, then it looks for anomalous activities that do not conform to the established model.

Dorothy Denning [7] developed the first model of real anomaly based IDS. It was named Intrusion Detection Expert System (IDES). It was initially a rule based expert system trained to detect known malicious activity. The core idea behind this is that intrusion behaviors involve abnormal usage of the system. In this approach, models of normal behavior are developed and verified against the current usage and significant deviation from normal usage is

flagged as abnormal usage. Fig. 2.3 below illustrates the typical components of the IDS proposed by Denning. It is also commonly known as the anomaly based IDS.

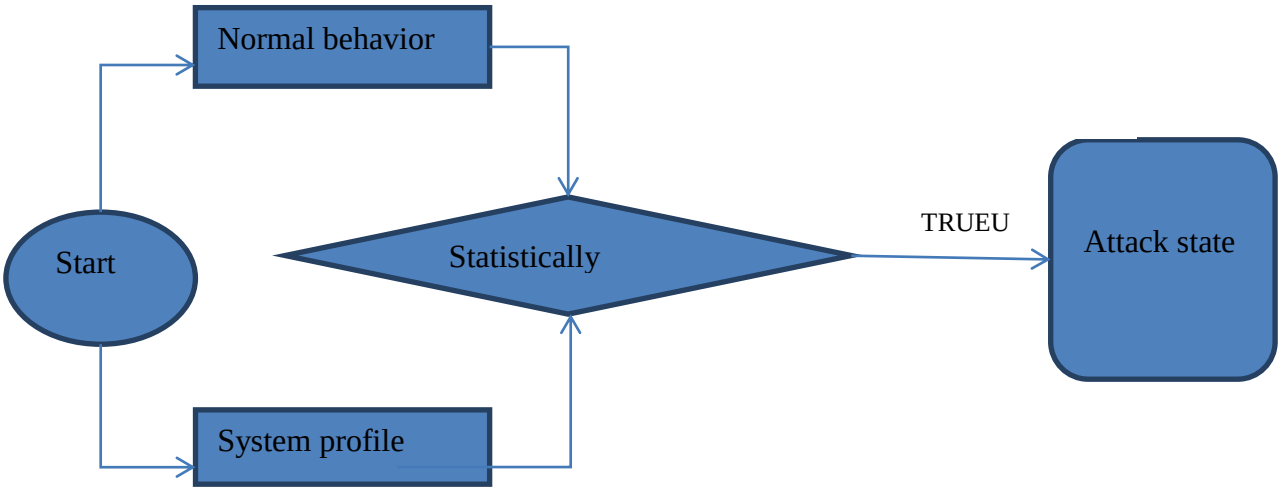


Fig 2.3: Components of an anomaly IDS

One of the key advantages of anomaly based IDS is that it has the capacity to detect previously unknown attacks. The main drawback is the fact that it generally has a high false alarm rate since detecting a deviation may not necessarily imply a false alarm.

Forrest et al. [10] developed IDS with similar behavior to the natural immune systems of living things. They provided a simple and practical way to detect anomalous behavior by analyzing short sequences of system calls which may generate a stable signature for some normal system behaviors. In other words, an abnormal activity could be detected from the system call sequences when an attack occurs. This approach incorporated two phases. The first phase involved collecting traces of normal behaviors and building a database to characterize normal patterns from the observed system calls. In the second phase, newly observed system call sequences were matched against the normal pattern of the system behaviors.

Lane and Brodley [11] developed an approach to distinguish the behavior of normal users and masquerading or illegal users by comparing a current behavioral sequence to historical users' profiles.

2.4 Intention discovery

Conceived by Schmidt et al [12], intention discovery is the process of identifying an entity's plans or overall intentions. By observing actions of the entity, the observing agent assimilates the entities actions as progressive phases of a plan and translates these actions into some kind of understanding of an entities plan. While there may be multiple ways to attack a typical system, it usually requires tools and actions to be applied in a specific logical order to launch the sequence of effective attacks to achieve the particular goal. This logical partial order is what reveals the short and long term intentions of the attacking entity. For example IP address spoofing is a common attack where IP packets with forged addresses are created with the purpose of concealing the identity of the true sender or impersonating another computer system. It typically involves the following steps:-

(1) <Selecting a target host (the victim)>

(2) <Identify a host that the target "trust">

(3) <Disable the trusted host, sampled the target's TCP sequence>

(4) <The trusted host is impersonated and the ISN forged.>

(5) <Connection attempt to a service that only requires address-based authentication.>

(6) <If successfully connected, executes a simple command to leave a backdoor.>

From the standpoint of IDS such as Snort, all that we can see are a sequence of alerts which don't necessarily give us information about what the intruder is attempting to achieve. For example Snort IDS returns the alert "*WEB-IIS BDIR.HTR ACCESS*" when an attempt is

made to access the "*bdir.htr*" file in the Windows Internet Information Server (IIS) system. Although we know that this attack has the potential of disclosing the directory structure of a vulnerable IIS we cannot know what the attacker is trying to achieve. However, from overall past experience, it is possible to know the series of alerts that typically lead to a certain type of attack. For example when there is the "*WEB-IIS BDIR.HTR ACCESS*" alert, there is a probability that the intruder's intention is to execute remotely a previously installed malicious script to perform a distributed denial of service attack (DDOS). However, this depends also on the previous intruder activities and previous alerts since this may be part of a long chain of attacks activities aimed at achieving an overall plan.

Cohen et al [13] distinguish between two kinds of intent discovery, keyhole and intended intent discovery. In keyhole discovery, the recognizer system is simply watching normal actions of an agent. The agent does not care or is not aware that their actions are being observed. They are simply engaging in the task. In intended discovery, the agent is cooperative; its actions are done with the intent that they be understood. This may result in the agent performing the action in a particular or stylized way in an effort to assist the recognizer in the task. Intended plan discovery arises, for example, in cooperative problem-solving and in understanding indirect speech acts. In these scenarios, discovering the intentions of the agent may allow us to provide assistance or respond appropriately

While intent discovery has applications in a wide array of fields including military simulations [14], human-robot interaction systems [15], intelligent prosthetic devices [16] .Geib and Goldman [17] first introduced it into the field of intrusion detection.

Geib and Goldman [17] argued that intent discovery must be a central component of future intrusion detection systems since with the current approaches, the role of the IDS has

been reduced to that of the post mortem analysis. They further give general approaches towards intent discovery.

While most of the work has been centered on building IDS, very little has been done in intruder intention discovery. Feng et al [18] developed intrusion plan recognition method for predicting the anomaly events and the intentions of possible intruders to a computer system based on the observation of system call sequences. In their implementation, the system calls served as the source of alerts. First, the goals of different sequences of system calls were classified into two states normal and anomalous. With the clearly defined normal (e.g. issuing normal commands) or abnormal goals (e.g. launching local or remote buffer overflow to get higher privileges), they built a dynamic Bayesian network whose structure changed with time as new nodes are added with new incoming calls.

Wu et al [19] proposed an approach to recognizing intentions based on attack path analysis using Bayesian rules and a path generation algorithm. In their approach they use a path generation algorithm to build a graph of the network configurations. In a specific network, an attacker's intentions could then be given at first according to the host vulnerabilities and network topology or the protective focuses. Through their algorithms, all probabilities of an attacker's intentions can be listed, and then more attention is paid to intentions with higher probabilities. Qin and Lee [19] developed an approach of using causal networks in the recognition of an attackers plan. In their approach they apply graph techniques to correlate isolated attack scenarios and identify their relationship. Based on the high-level correlation results, they further apply probabilistic reasoning techniques to recognize the attack plans, evaluate the likelihood of potential attack steps and predict upcoming attacks. Cuppens et al [21] suggest a model to discover intrusion scenarios and malicious plans in a network. The model does not follow previous proposals that require to explicitly specification of a library of

intrusion scenarios. Instead, their approach is based on specification of elementary attacks and intrusion objectives. They then show how to derive correlation relations between two attacks or between an attack and an intrusion plan.

2.5 Motivation and uniqueness of our approach

Our work is also inspired by the current reality that IDS such as Snort, which is the most widely used IDS only emit raw alerts and cannot decipher the intruders' high level intentions for recovery and pre-emptive counter measures. In current implementations, Snort and other popular IDS allows rules to be set so that when a certain alert is sounded, certain counter measures may be deployed. This approach is disadvantageous in that the counter-action is based on very limited information since the counter action if a function of the current alert. With our approach, the counteraction is a function of a big alert stream and is thus more effective. Our goal is that in the future, developers can build an intelligent intention discovery layer over IDSs using approaches such as the one that we describe here.

While our approach is unique since no one has approached the problem using Hidden markov models, we could make some comparisons to other related works by others. Unlike the system by Feng et al [19] which is host based and whose system depended on input data from operating system calls (meaning that it may not recognize attacks against other hosts in a network), in our approach, input for the intent recognition engine is obtained directly from the network IDS. This has the advantage of being easy to integrate and a large network can be protected from one point by a single intention discovery engine. Further, their approach categorized intentions into "normal" and "abnormal". As it is in our proposed system, it is more useful to classify intentions more definitively to ensure more precise counter actions. While the system by Wu et al [19] may have some similarities in approach since they apply a related

algorithm, the Bayesian algorithm, their approach depends on an attacker not backtracking in his action. Our approach is very flexible and will handle any attacker's behavior. Further, unlike Wu et al's [19] approach whose dependence on the network configurations can result in scalability issues our approach is very scalable since all that is required is the training of the HMM engine. Further, approaches using Bayesian networks such as those by Qin and Lee [20] and Cuppens et al [21] once implemented may be susceptible to Bayes poisoning and exploit best associated with email spam filters based on the Bayesian theorem.

In our approach, we chose to use HMM for a number of reasons. Unlike other possible approaches such as using data mining tools and techniques to achieve the same, our algorithms would entail minimal computational overheads and ease of implementation .This is due to the fact that a data mining approach would entail use of complex relational databases while in using HMM and Viterbi, all that is required is an implementation and simple data storage with no need for complex data management. Another advantage of HMM is the fact that it allows a modeler to easily incorporate prior knowledge about the network in order to constrain the training process using algorithms such as Baum-welch. Unlike approaches involving Bayesian/Causal networks where reliability is dependent on the probability values of one point, the root node, due to its cause-effect nature, in HMM overall reliability is not dependent on the reliability at a single point As such, a HMM based system would be more resistant to excessive shift in its performance characteristic due to shift of an input variable at one point. The graphical nature of HMM and the Viterbi algorithms clearly displays the links between different components in the architecture. This ensures that the design can be studied and shared with ease.

Chapter 3

Intrusion Intention Discovery

3. INTRUSION PLAN DISCOVERY

In this chapter, we begin by describing the architecture that incorporates our proposition. We then explain the model of our intrusion plan discovery engine. In particular, we explain how we incorporate the HMM in the ordinary IDS and then use a decoding algorithm to help discover an intruders plans for a given set of alerts logged by the resident IDS.

3.1 The architecture

Fig 3.1 is a block diagram illustrating how our intruder plan discovery mechanism would fit into an existing network equipped with an IDS.

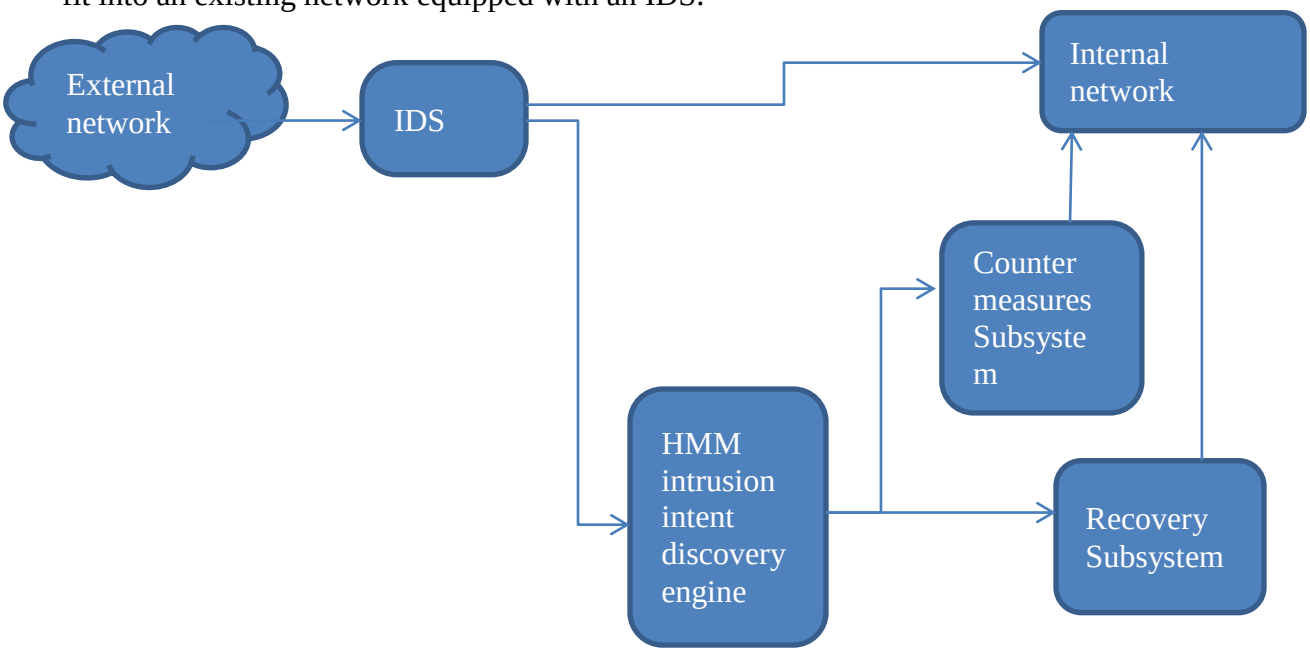


Fig 3.1: Architecture of network with IDS and intent discovery mechanism

The HMM intruder plan discovery engine essentially consists of a HMM model built based on data of previous intruder activities for a given set of alerts and a decoding algorithm. Alerts are pulled from the IDS in batches. An alert is an alarm sounded by the IDS that indicates that some form of malicious operation has been detected just in the same way as an anti-virus program may alert a user that a file contains a virus or a worm. Using a decoding algorithm such as Viterbi, the HMM intruder plan discovery engine outputs the intruders plans for the set of alerts. Further, there is an output of the most probable future intention of the intruder. These two sets of outputs are used for recovery and attack preemption respectively. Refer to fig 3.2

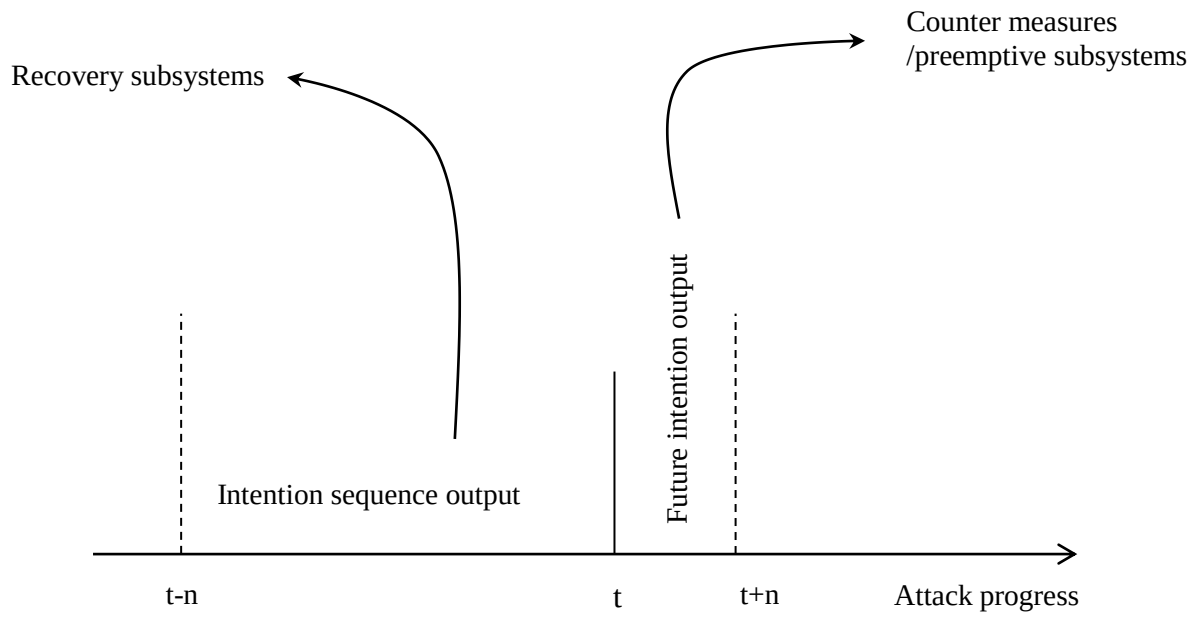


Fig 3.2: Application of information from the intention discovery engine

As illustrated in Fig 3.2, at any present time *t*, the set of alerts generated between *t-n* and *t* would generate the intention sequence output. Since this represents intruders actions that have already happened, the information about his sequence of intentions can best be used in the recovery of the system. Hence the intention sequence output is fed to the recovery subsystem.

The recovery subsystem is a broad term that refers to any system or tools that help restore the system state or provide intelligence to the administrators to help mitigate the attack. Obviously, some of the effects of the attack will at this stage be visible. For example, a process may be found to be running or files will have been deleted. Having more precise information about the attackers sequence of intentions would aid in better recovery effort.

The time $t+n$ in Fig 3.2 represent the future. The next intruder plan is inferred from the output of the last intention at time t since the current intention and the next intention are related through probabilities in our HMM. This intelligence about the future intention provides the system with preemptive capacity. Based on this future expected intention, proper counter-measures can be adopted such as closing a port or killing certain vulnerable processes.

3.2 The Hidden Markov Model

In this thesis, we use the hidden markov model to model the relationship between the intruder states representing the sequence of intentions and the sequence of alerts from the IDS. We then use a related algorithm, the Viterbi algorithm to reconstruct the sequence of intentions of an intruder based on the sequence of alerts. Hidden Markov Models are an extension of the theory of discrete Markov chains. Rabiner [8] presents a tutorial on HMMs with applications to speech recognition.

Formally, the hidden markov model is composed of:-

(a) Set of states: $S = \{S_1, S_2 \ldots . S_n\}$

(b) A process moves from one state to another generating a sequence of states: $\{S_{i1}, S_{i2}, \ldots\ldots, S_{ik} \ldots\}$

(c) Markov chain property: probability of each subsequent state depends only on the previous state: $P(S_{ik}|S_{il}, S_{i2}, \ldots., S_{ik-1}) = P(S_{ik}|S_{ik-1})$

(d) States are not visible, but each state randomly generates one of M observations (or visible states) $V=\{V_1,V_3....V_m\}\}$

(e) Further, the following probabilities have to be specified: matrix of transition probabilities $A=[a_{ij}]$, $a_{ij}= P[s_i \mid s_j]$, matrix of observation/emission probabilities $B=[b_i[v_m]]$, $b_i[v_m] = P[v_m \mid s_i]$ and a vector of initial probabilities $\pi=[\pi_i]$, $\pi_i = P[s_i]$.

To better illustrate the concept, Fig 3.3 depicts a typical HMM clearly showing the various transition probabilities and states.

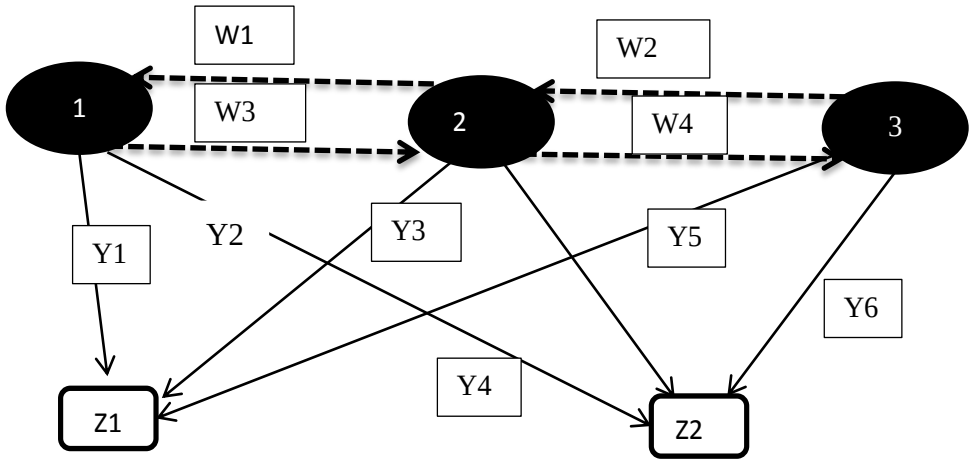


Fig 3.3: Example of HMM

In the HMM in Fig 3.3:-

(a) 1, 2 and 3 are the hidden states of the system.

(b) Z1 and Z2 are the observations.

(c) W1, W2, W3 and W4 are the transition probabilities i.e P[1|2]=W3, P[2|1]=W1 etc

(d) Y1 through Y6 are output probabilities also called observation probabilities i.e P[Z1|1]=Y1,P[Z2|3]=Y6 etc

(e) An initial probabilities may be P[1]=0.4,P[2]=0.3 and P[3]=0.3.

There are three canonical problems associated with HMMs. Solutions to these problems are what makes HMMs useful. The problems are:-

(1) The Evaluation problem.

Given the HMM $M = [A, B, \pi]$ and the observation sequence $V = v_1 v_2 \ldots v_K$, calculate the probability that model M has generated sequence V.

(2) The Decoding problem.

Given the *HMM* $M = [A, B, \pi]$ and the observation sequence $V = (v_1 v_2 \ldots v_K)$ calculate the most likely sequence of hidden states S_i that produced this observation sequence *V.* In this thesis, we use the Viterbi algorithm [9] solve this problem. The outputs of the decoding process are the actual user intentions for the set of alerts.

(3) Learning problem.

Given some training observation sequences $V = v_1 v_2 \ldots v_K$ and general structure of HMM (numbers of hidden and visible states), determine HMM parameters $M = [A, B, \pi]$ that best fit training data. In this thesis, this is the first problem that we tackle in modeling our IDS to discover users plans.

Consider a scenario where to undertake some attack, the intruder must do the following:

(a) Sweep the IP's

(b) Do a Sadmind Ping

(c) Issue commands that exploit Sadmind

(d) Install DDOS software

(e) Launch a Distributed denial of service attack (DDOS).

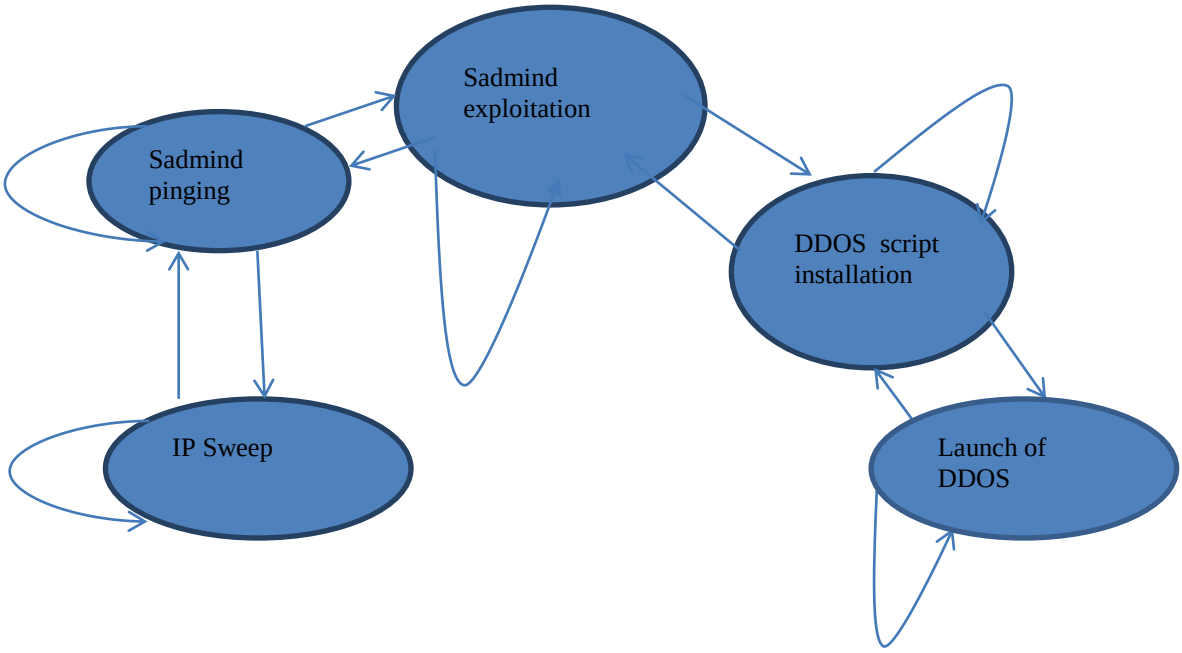


Fig 3.4: State representation of an intruder's plan

Fig 3.4 illustrates the intruder's plans as a state diagram. The arcs at each state represent the probability of a users plan changing from the current one to the next one, previous one or current one.For example, when the intruder is <<*Sadmind Pinging*>> his next action may be to <<*Sadmind exploit*>>. However, there is a possibility that he may also decide to <<*IP Sweeping*>>. This may be necessitated by an array of factors including, procedural requirements, a target restarting or to confuse a potential target. The characteristic of this attack progression profile is a good candidate for a hidden markov model. The intruders sequence of intention over time may be viewed as being a chain of states .These states represent the hidden state of the HMM. The states are hidden since we cannot tell what the intruder is trying to do unless we look at the sequence of alerts as raised by the IDS. The alerts represent the observable sequences of the HMM.

At each stage of an attack, certain alerts will sound that indicate that an intruder is intending to accomplish something. As such there is a probabilistic relationship between the alerts and the intruder's intentions. With the intruder's intentions comprising the sequence of hidden states and the alerts being the observables, the probabilistic relationship between the intruder's intentions and the alerts comprise the emission probability. Fig 3.4 is reproduced in fig 3.5 with alerts included. The dashed arrows represent the emission probabilities.

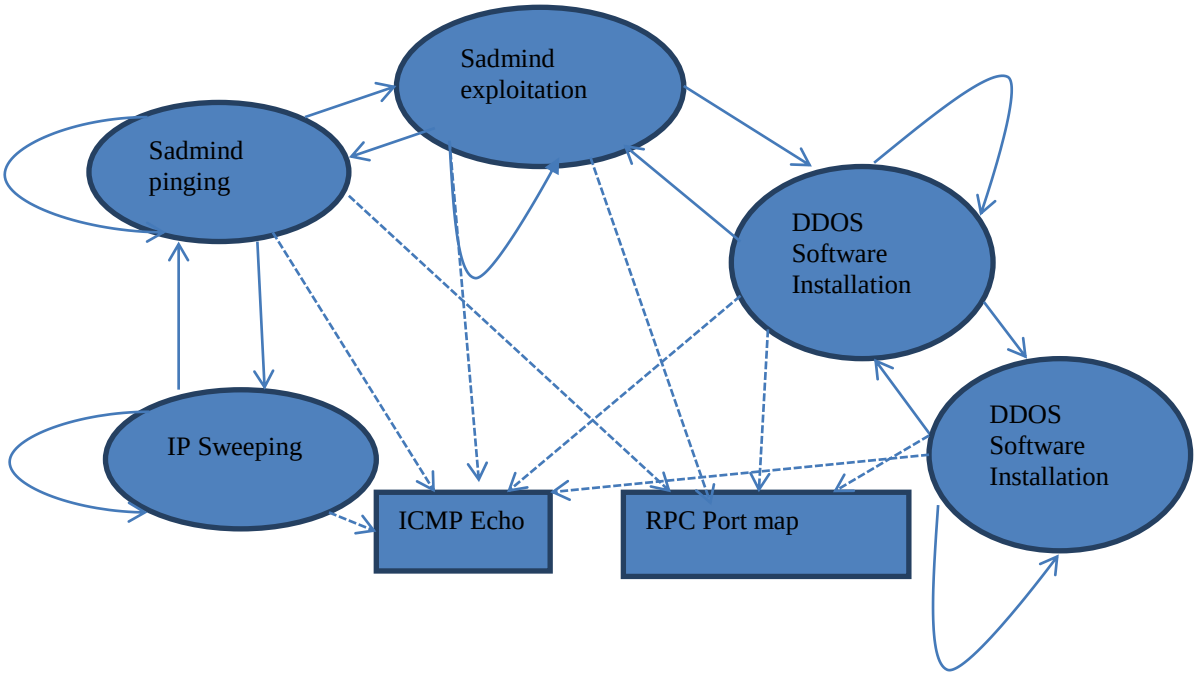


Fig 3.5: HMM Model of an attack

3.3 Training

Training is the task of estimating the parameters such as emission and transition probabilities in a way that best accounts for the data and the scenario being modeled. For example if fig. 3.5 above, training would replace all the arcs with probabilities. The required input for the task of training any HMM is a database of sample behavior. Training is critical as it

determines the accuracy and performance of the model.

In practice, for small datasets, training can be done manually by estimating the various parameters based on past experience. The Baum-Welch algorithm [22] may also be used for the training. It can compute maximum likelihood estimates and posterior mode estimates for the parameters (transition and emission probabilities) of an HMM, when given only emissions as training data. To train the model, the inputs to the training algorithm will be the hidden states (possible intentions of the intruder), the sequence of observables (the alerts) and the typical network behavior if necessary.

3.4 The decoding problem

Once we have modeled the previously understood attack scenario as described in section 3.2 we can discover a sequence of intruder's intention based only on the set of sequence of alerts. This is the decoding problem.

In this thesis, we solve the decoding problem using the Viterbi algorithm. Viterbi [9] proposed the Viterbi algorithm as a method of decoding convolutional codes. Since that time, it has been recognized as an attractive solution to a variety of estimation problems. The algorithm can be summarized as follows:-

Suppose we are given a HMM with states y, initial probabilities π_i of being in state i and transition probabilities $a_{i,j}$ of transitioning from state i to state j. Say we observe outputs $\{x_O \ldots\ldots\ldots x_T\}$. The state sequence $\{y_0 \ldots\ldots y_t\}$ most likely to have produced the observations is given by the recurrence relations:

$$V_{0,k} = P(x_o|k).\pi_k$$

$$V_{t,k} = P(x_t|k).max_{y \epsilon Y}(a_{y,k}, V_{t-1,y})$$

Here $V_{t,k}$ is the probability of the most probable state sequence responsible for the first $t + 1$ observations (we add one because indexing started at 0) that has k as its final state. The Viterbi path can be retrieved by saving back pointers which remember which state $\boldsymbol{y}$ was used in the second equation. Let $Ptr[k,t]$ be the function that returns the value of y used to compute $V_{t,k}$ if $t > 0$, or k if $t = 0$. Then:

$$y_T = argmax_{y \epsilon Y}(V_{T,y})$$

$$y_{t-1} = Ptr(y_t, t)$$

The algorithm makes a number of assumptions.

(1) First, both the observed events and hidden events must be in a sequence. This sequence often corresponds to time.

(2) Second, these two sequences need to be aligned, and an instance of an observed event needs to correspond to exactly one instance of a hidden event.

(3) Third, computing the most likely hidden sequence up to a certain point t must depend only on the observed event at point t, and the most likely sequence at point $t - 1$.

Upon running the Viterbi algorithm implementation we get the set of most probable set of intruder intentions for the given set of alerts. Fig 3.6 illustrates the idea.

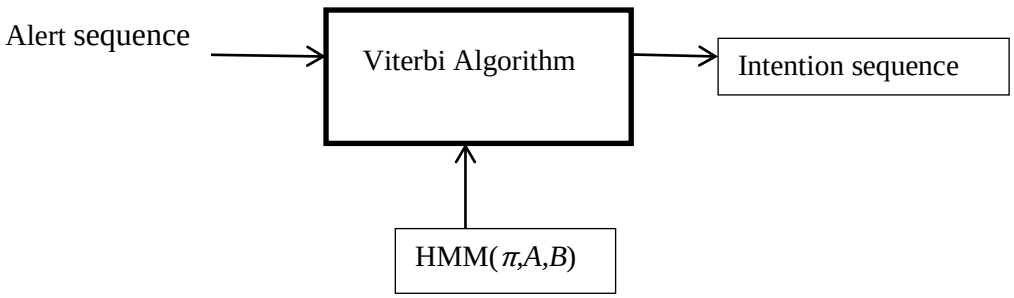


Fig 3.6: Viterbi solution to the decoding problem

Chapter 4

Experimental Setup and Simulation

4. EXPERIMENTAL SETUP AND SIMULATION

In this chapter we set up a simulation experiment to validate our work.

The general methodology to our experiment is:

(1) Run the Snort IDS to obtain the set of alerts that constitute the observable sequence.

(2) Build a HMM model based on the alerts and our background knowledge of the attackers intentions. The model is built using data from and IDS sensor in the demilitarized zone (DMZ) of the network since it is exposed to more traffic. The demilitarized zone is a physical or logical sub network that contains and exposes an organization's external services to a larger untrusted network such as the internet. While a single model is enough for experimental purposes, in a real implementation, more than one model may be required, built using data from different locations of the network which are exposed to different kinds of traffic depending on the security layers protecting them. In such a case, their output may be averaged or a voting algorithm implemented.

(3) Run the Viterbi algorithm the model and a sample sequence of alerts to discover the intentions (Hidden state) for the given alerts sequence.

(4) Compare the output from the algorithm and the expected results for validation purposes.

(5) Repeat step 3 again with different sizes of inputs to observe its performance.

First, we will discuss the tools we used in the experiment and highlight the background of the simulation data. We will then describe the experiment.

4.1 Tools

In our experiment, we used analyzed packets logged by the Tcpdump [23] , an open source powerful command-line packet analyzer for network traffic, captured during a simulated attack as undertaken by the MIT Lincoln lab in 2000. The logs are the raw traffic captured by Tcpdump and do not provide us with any information of the alerts raised as the attack progressed. However, the background information accompanying the data explains what the intruder is trying to achieve and the phases of his attack.

By using the log files, we simulate the events around the time of the attack. We use Snort running in IDS mode to detect the intrusions. Snort is an open source network intrusion prevention and detection system (IDS/IPS) originally developed by Martin Roesch [3]. Snort combines the benefits of signature and anomaly-based inspection, and is the most widely deployed IDS/IPS technology worldwide. Snort is a command line application. Fig 4.1 illustrates a screenshot of Snort as used in the experiment.

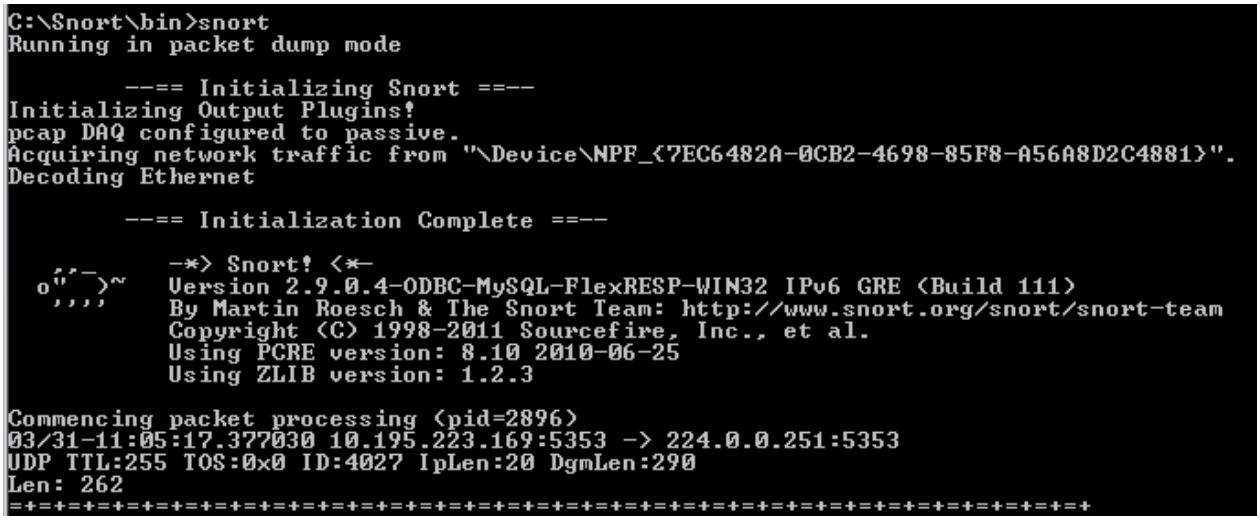


Fig 4.1: Screenshot of the Snort command line

For Snort to be able to detect attacks, it needs rules. Snort rules are a library of statements that enable Snort to catch malicious traffic and sound alarms. Good rules ensure minimal false

alarms. The powerful nature and popularity of the Snort detection system is attributed to the rules which are shared in public and constantly improved by Snort users and network administrators [24]. In our experiment we downloaded the latest rules from the snort website *(http://www.snort.org*) released on 23 Feb, 2011 and configured Snort to use the new rules.

Output from log is ordinarily logged to a file hence for ease of analysis, we configured snort to save the alerts to the local MySQL [25] database server. Support for MySQL is inbuilt in Snort.

To view the alerts from Snort we downloaded and installed the Base Analysis and Security Engine (BASE) [26]. BASE is an open source web interface to perform analysis of intrusions that Snort has detected on a network as logged on the MySQL database.

Fig. 4.2 illustrates a screenshot from BASE showing the various parameters and options as captured during the experiment.

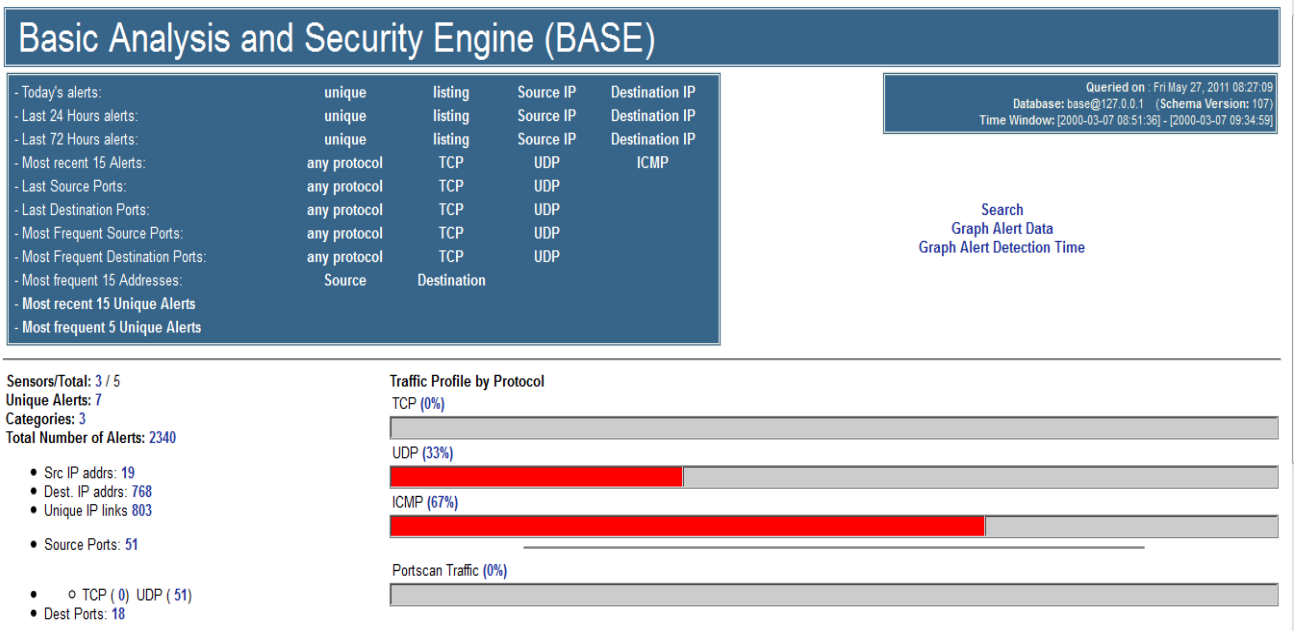


Fig 4.2: Screenshot of the BASE application dashboard

We used a C# application originally written by Paulo Costa Carvalho [27] that implements the Viterbi algorithm to solve the decoding problem. All experiments were undertaken on a Windows based Acer Aspire 5741Z machine with a 64 bit windows 7 operating system , 4 gigabytes of RAM and a dual core 1.87GHz Intel Pentium P6000 processor.

4.2 The Intrusion data

The Information Systems Technology Group (IST) of MIT Lincoln Laboratory, under the Defense Advanced Research Projects Agency (DARPA ITO) and Air Force Research Laboratory (AFRL/SNHS) sponsorship collected and distributed their first standard corpora for evaluation of computer network intrusion detection systems in 1998 through 2000. The datasets generated are of interest to researchers working on the general problem of workstation and network intrusion detection. According to them evaluation was designed to be simple, to focus on core technology issues, and to encourage the widest possible participation by eliminating security and privacy concerns, and by providing data types that were used commonly by the majority of intrusion detection systems.

Joshi and Phoha [28] among other researchers have also used this data set successfully in their work related to network intrusion. We will use the 2000 DARPA Intrusion Detection Scenario-Specific LLDOS1.0 dataset. This data set consists of data logged from two different locations in the network: - at the demilitarized zone and inside the network. The DMZ is a physical or logical subnet that contains and exposes an organization's external services to a larger untrusted network. The purpose of a DMZ is to add an additional layer of security to an organization's local area network (LAN)

4.2.1 The scenario

The premise of the attack is that a relatively novice adversary seeks to show his/her prowess by using a scripted attack to break into a variety of hosts around the Internet, install the components necessary to run a Distributed Denial of Service, and then launch a DDOS at a US government site. As a part of the attack the adversary uses the Solaris sadmind exploit, a well-known Remote-To-Root attack to successfully gain root access to three Solaris hosts at Eyrie Air Force Base. These attacks succeed due to the relatively poor security model applied at the AFB, many services, including the dangerous "*sunrpc*" service, are proxied through the base's firewall from outside to inside. The attacker is using the *Mstream* DDOS tool, one of the less sophisticated DDOS tools. It does not make use of encryption and does not offer as wide a range of attack options as other tools, such as *TribeFloodNetwork* or *Trinoo*. An *Mstream* "server", the software that actually generates and sends the *DDOS* attack packets, is installed on each of the three victim hosts, while an *Mstream* "*master*", the software that provides a user-interface and controls the "*servers*" is installed on one of the victims. The attack scenario is carried out over multiple networks and phases.

The attacker has 5 intentions:-

- *IPsweep* of the AFB from a remote site
- Probe of live IP's to look for the sadmind daemon running on Solaris hosts
- Breakins via the *sadmind* vulnerability, both successful and unsuccessful on those hosts
- Installation of the trojan *mstream* DDoS software on three hosts at the AFB
- Launching the DDoS

The graph of fig 4.3 [31] visualizes the progression of the attack.

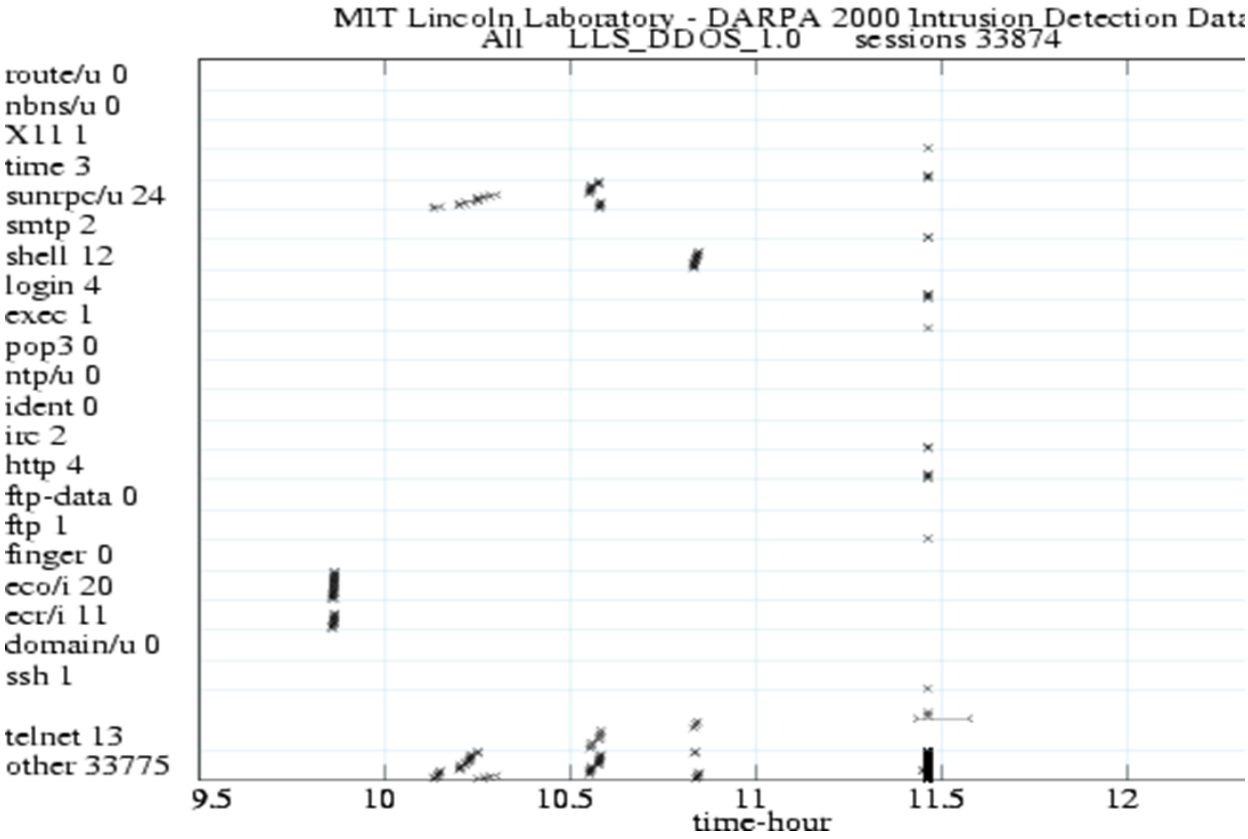


Fig 4.3: Graphical illustration of the progression of the attack

In the graph of fig 4.3 the X-axis is the time of day, in hours, in Eastern Standard Time. The Y-axis shows the TCP and UDP services on which attack traffic might flow.

4.2.2 The intruder's intentions

To achieve his overall plan of launching the DDOS the intruder had 5 plans.

- *IP Sweeping* –The intruder performed scripted *IPsweeps* of multiple class C subnets on the Air Force Base. The following networks were swept from address 1 to *254: 172.16.115.0/24, 172.16.114.0/24, 172.16.113.0/24, 172.16.112.0/24.* The attacker sent *ICMP echo-requests* in this sweep and listened for *ICMP echo-replies* to determine which hosts are "*up*".

- *Sadmind pinging*- The hosts discovered in the previous phase were probed to determine which hosts were running the "*sadmind*" remote administration tool. This told the intruder which hosts might be vulnerable to the exploit that he/she has. Each host was probed, by the script, using the "*ping*" option. The ping option made a *rpc* request to the host in question, asked what TCP port number to connect to for the *sadmind* service, and then connected to the port number supplied to test to see if the daemon was listening.
- *Sadmind exploitation*: The attacker then tried to break into the hosts found to be running the *Sadmin* services. The intruder attempts the *Sadmind* remote to root several times on each host. With each attempt, the intruder attempted to execute one command, as root, on the remote system. The attacker needed to also execute two commands however, one to "*cat*" an entry onto the victim's /etc/passwd file and one to "cat" an entry onto the victim's /*etc*/shadow file. The new root user's name was *'hacker2'* and *hacker2's* home directory was set to be /*tmp*. Thus, there were 6 exploit attempts on each potential victim host. To test whether or not a break-in was successful, the attack script attempted a login, via telnet, as hacker2, after each set of two break-in attempts.
- *DDOS Software Installation*- Entering this phase, the intruder had built a list of those hosts on which it had successfully installed the 'hacker2' user. These were *mill* (172.16.115.20), *pascal* (172.16.112.50), and *locke* (172.16.112.10). For each host on this list, the intruder performed a telnet login, made a directory on the victim called "/*tmp/.mstream*/" and used rcp to copy the server-sol binary into the new directory. This is the mstream server software. The attacker also installed a "*.rhosts*" file for themselves in */tmp*, so that they could *rsh* in to startup the binary programs. On the first victim on the list, the intruder also installed the "master-sol" software, which is the *mstream master.* After installing the software on each

host, the intruder used *rsh* to startup first the master, and then the servers. As they came up, each server "registered" with the master that it was alive. The master wrote out a database of live servers to a file called *"/tmp/.sr*".

- *Launch of DDOS* – Finally, the intruder launched a DDOS. This was done via a TELNET login to the victim the "*master*" installed previously was running. This was done on port 6723 where the "*master*" was listening. The intruder supplied the command "*mstream*" with an IP address as a parameter. This caused a DDOS attack of 5 seconds duration against the supplied IP address launched by the 3 serves simultaneously. The intruder then logged out.

4.3 The alerts

In this section of the experiment, we wanted to know what kind of alerts the IDS sensor would raise. We already know the users progressive intentions throughout the attack period as provided in the background information that came with the Tcpdump files. Further, we match the alerts with the intentions of the intruder.

4.3.1 Generation of the alerts

The alerts in were Fig 4.5 were obtained from IDS sensor in DMZ of the network. We simulated this by running Snort with the source of packets being the Lincoln Lab Tcpdump file of the DMZ. Fig 4.4 shows the details of our simulated IDS sensor.

Sensor Details	Sensor Address	Interface	Filter
	JORDAN:DMZ3	DMZ3	*None*

Fig 4.4: Details of Simulated IDS sensor at the DMZ

Sensor Alerts		
Alert name	Number of alerts	Protocol
ICMP Echo Reply	36(2%)	ICMP
RPC portmap Solaris sadmin port query udp request	320(13%)	UDP
RPC portmap sadmind request UDP	320(13%)	UDP
ICMP Destination Unreachable Port Unreachable	2(0%)	ICMP
RPC sadmind query with root credentials attempt UDP	64(3%)	UDP
RSERVICES rsh root	32(1%)	TCP
ICMP PING	1534(65%)	ICMP
	Total: 2372(100%)	

Fig 4.5: Alerts from IDS on the network DMZ

The alerts in Fig 4.7 were obtained from the IDS sensor inside the network. As with the previous case we simulated this by running Snort with the source of packets being the Lincoln Lab TcpDump of the inside zone of the network. Table 4.3 shows the details of the IDS sensor.

Sensor Details	Sensor Address	Interface	Filter
	JORDAN:in3	in3	*none*

Fig 4.6: Details of Simulated IDS on the inside network

Sensor Alerts		
Alert name	Number of alerts	Protocol
ICMP PING	40(6%)	ICMP
ICMP Echo Reply	40(6%)	ICMP
RPC portmap sadmind request UDP	180(27%)	UDP
RPC portmap Solaris sadmin port query udp request	180(27%)	UDP
ICMP Destination Unreachable Port Unreachable	144(22%)	ICMP
RPC sadmind query with root credentials attempt UDP	28(4%)	UDP
RPC portmap Solaris sadmin port query udp portmapper sadmin port query attempt	28(4%)	UDP
RSERVICES rsh root	16(2%)	TCP
	Total: 656(100%)	

Fig 4.7: Alerts from the IDS on the inside network

The higher number of alerts from the DMZ section of the network may be attributed to the fact that the DMZ is exposed to more potentially hostile traffic compared to the internal network. Hence, we chose to use alerts from the DMZ in our model. In total there were 8 types of alerts in our experiments.

4.3.2 Description of the alerts.

Each of the alerts points to the fact that the intruder has some intention. We give a brief description of the alerts below. We have labeled each of the alerts as shown in the first column in Fig 4.8.

Label	Alert	Brief description
1	ICMP PING	The ICMP echo request is used by the ping command to elicit an ICMP echo reply from a listening live host.
2	RPC portmap sadmind request UDP	Request to discover the port sadmind is running from
3	RSERVICES rsh root	Connection to the rsh daemon as root
4	RPC portmap Solaris sadmin port query udp portmapper sadmin port query attempt	Query for sadmind host port
5	ICMP Echo Reply	ICMP echo reply from a listening live host elicited in response to a ICMP echo request.
6	RPC portmap Solaris sadmin port query udp request	Request to know if host is running Sadmind
7	RPC sadmind query with root credentials attempt UDP	This indicates that the RPC query for the sadmind service has been made with the credentials of the root user supplied
8	ICMP Destination Unreachable Port Unreachable	Indicates that someone or something tried to connect to a port on a system that was not available

Fig 4.8: Description of the alerts

4.3.3 The alert sequence

Section 4.2.1 provided the statistics of the different types of alert sequences and their quantities. In this section, we list the sequence with which the alerts occurred. While 3,028 alerts were logged by the IDS, we only needed a small portion to build our HMM model. We obtained DMZ alerts whose target was host "*mill*" with IP address 172.16.115.20 .All alerts as identified by Snort indicate the source address and the destination address (see Appendix A). For host 172.16.115.20, 61 alerts were returned by Snort for attacks directed at it at the DMZ. The first 18 alerts as they were displayed by BASE are shown in figure 4.9 ordered in descending orders with the first alert being "ICMP PING". (See Appendix B and C for the full sequence).

#35-(28-12)	[arachNIDS] [snort] RPC portmap sadmind request UDP
#36-(28-13)	[snort] RPC sadmind query with root credentials attempt UDP
#37-(28-14)	[snort] RPC sadmind query with root credentials attempt UDP
#38-(28-16)	[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp portmapper sadmin port query attempt
#39-(28-15)	[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp portmapper sadmin port query attempt
#40-(28-1)	[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request
#41-(28-2)	[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request
#42-(28-3)	[arachNIDS] [snort] RPC portmap sadmind request UDP
#43-(28-4)	[arachNIDS] [snort] RPC portmap sadmind request UDP
#44-(28-5)	[snort] RPC sadmind query with root credentials attempt UDP
#45-(28-7)	[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp portmapper sadmin port query attempt
#46-(28-8)	[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp portmapper sadmin port query attempt
#47-(28-6)	[snort] RPC sadmind query with root credentials attempt UDP
#48-(27-52)	[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request
#49-(27-50)	[arachNIDS] [snort] RPC portmap sadmind request UDP
#50-(27-49)	[arachNIDS] [snort] RPC portmap sadmind request UDP
#51-(27-51)	[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request
#52-(26-41)	[snort] ICMP PING
#53-(26-42)	[snort] ICMP PING

Fig 4.9: Sequence of Snort alerts for host "*mill*"

4.4 The Hidden Markov Model

The aim of creating a model is to establish the probabilistic relationship between the sequence of alerts (the observables) and the intention states (hidden states). The model design is critical to the performance of the intention discovery system. In this section, we use the alerts from section 4.3.1, and background information about the attack scenario to estimate parameters for the HMM. We compute the parameters, A, B, and π, A, and B respectively as described in section 1.5 . As explained in section 3.2 the set of states $S=\{S_1,S_2,S_3,S_4....S_n\}$ correspond to the intentions of the intruder in the 5 phases of the attack. These are:-

S={*<<IP Sweep to determine which hosts are up>>,<< Probing to look for Sadmind daemons running on target> >,<< Attempting breaking via the sadmind vulnerability> >,<< Installing the Trojan mstream DDOS script> >,<< launching DDOS> >*}

For clarity, we shall label the states as shown below with the letters corresponding to the respective intentions as shown in Fig 4.11

Intention state	Label
IP Sweep to determine which hosts are up	U
Probing to look for Sadmind daemons running on target	V
Attempting breaking via the sadmind vulnerability	W
Installing the Trojan mstream DDOS script	X
launching DDOS	Y

Fig 4.10: List of intention states

Hence our hidden states are S= {*U, V, W, X, Y*}.

The observable sequences are the alerts (see Fig 4.8), hence V= {*1, 2, 3, 4, 5, 6, 7, 8*}.

The next step is estimating the parameters A (state transition probability), B (emission probability) and π (initial probability) for both attacks aimed at "*pascal*" and "*mill*"

(a)Parameter estimation for attack on "mill" (172.16.115.20)

As a method to estimate parameters A, B and π we pick a sequence of the 61 DMZ IDS alerts with destination address being host "*mill*" (172.16.115.20) . The labeled alert sequence was:-

(1,1,4,4,2,2,2,4,4,7,7,6,6,2,4,4,2,2,6,6,2,2,4,7,7,6,6,6,2,4,7,7,6,6,2,4,4,7,7,4,4,2,4,7,7,6,6,2,4,4,7,7,4,4,2,2,4,7,7,6,6,2,2,3,3,3,3,3,3,3,3) (See appendix B). The sequences are numbered as per the labeling scheme in table 4.5.

Similarly, The labeled state transition pattern is:- {$UUVVVV(W)^{48}$XXXXXXXX }

From the transition pattern starting at intention U we have 1 UU, 1UV, 3 VV, 1VW , 47WW and 1 WX. This is summarized in fig 4.12

Intention Sequence	**Count**	**Probability**
Transitions from U		
UU	1	1/2
UV	1	1/2
Transitions from V		
VV	3	3/4
VW	1	1/4
Transitions from W		
WW	47	47/48
WX	1	1/48
Transitions from X		
XX	8	8/8

Fig 4.11: Intention states transition probability for *A*

Hence the estimated transition probability matrix *A* is:

$$A = \begin{Bmatrix} 1/2 & 1/2 & 0 & 0 \\ 0 & 3/4 & 1/4 & 0 \\ 0 & 0 & 47/48 & 1/48 \\ 0 & 0 & 0 & 1 \end{Bmatrix}$$

Similarly from our alert sequence there are when in state *U*, we have 2 type *1* alerts. *V* intention states with with 10 type 2 alerts, 0 type *4* alerts, 7 type *6* alerts and 0 type *7* alerts. There are also 16 *W* intention states with 0 type *2* alerts, 8 type *4* alerts , 0 type *6* alerts and 8 type *7* alerts. This is summarised in fig 4.13

Alert type	Alert count when in intention state U	Alert count when in intention state V	Alert count when in intention state W	Alert count when in intention state X	Probability of alert when in state u	Probability of alert when in state V	Probability of alert when in state w	Probability of alert when in state x
1	2	0	0	0	2/2	0	0	0
2	0	2	13	0	0	2/4	13/48	0
3	0	0	0	8	0	0	0	8/8
4	0	2	13		0	2/4	13/48	0
6	0	0	11	0	0	0	11/8	0
7	0	0	11	0	0	0	11/48	0

Fig 4.12: Emission probability for *B*

From Fig 4.13 the estimated emission probability matrix *B* is:-

$$B=\begin{Bmatrix} 2/2 & 0 & 0 & 0 & 0 & \\ 0 & 2/4 & 0 & 0 & 2/4 & 0 \\ 0/48 & 13/48 & 0/48 & 13/48 & 11/48 & 11/48 \\ 0 & 0 & 8/8 & 0 & 0 & 0 \end{Bmatrix}$$

To estimate π, we assumed that the intruder can have any of the intentions (*U,V,W,X*) at the start. Hence $\pi = \{0.25, 0.25, 0.25, 0.25\}$.

4.5 Validation and analysis of the results

In section 4.4 we developed the HMM *HMM_mill*= (π, *A*, *B*) and by defining the parameters π, *A, and B.* Since the intruders intentions during an attack will be characterized by the IDS alerts, the problem is to solve the decoding problem as described in section 3.3 and obtain the corresponding intention sequence for a given sequence of alerts.

In our experiment, we used our *HMM_mill* parameters and a sequence of alerts we seek to decode as input to Carvalo's C# Viterbi algorithm implementation. The outputs were the corresponding intention states and the probabilities with which they occur.

For the sequence of labeled alerts: *<1,1, 2,2> HMM_mill* returned the intention sequence <*UUVV*> with a Viterbi path probability of 87.891/10^^4 . Comparing the output state sequence <UUVV> with what we expect for the given alerts, we see that our method predicted with 100% accuracy. The Viterbi path is a path of transitions of intention states with the greatest probability. This path is best visualized using the trellis diagram as illustrated in Fig 4.14.

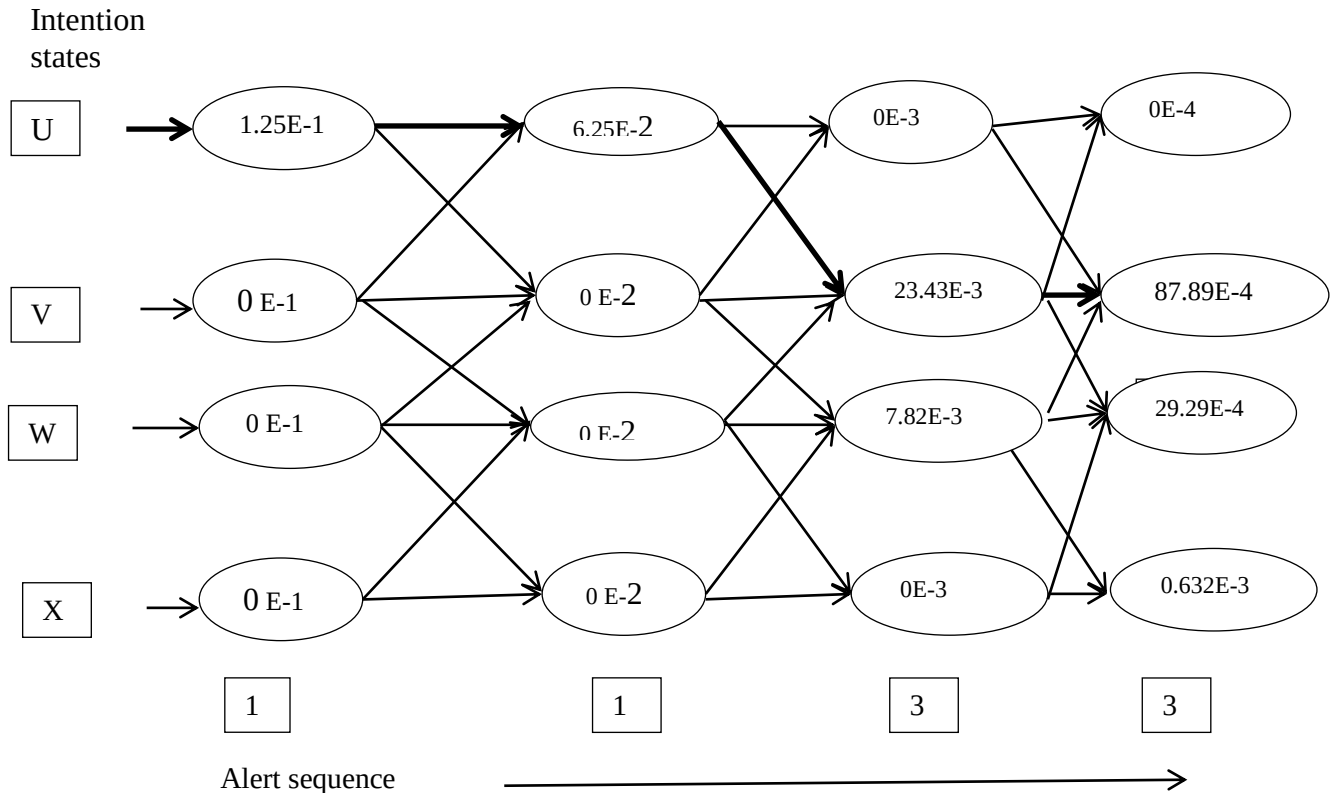


Fig 4.13: Trellis diagram of the Viterbi path for HMM_mill

As it can be seen in the trellis diagram in Fig 4.14, the intention sequence <*UUVV*> has the highest probability path that can be generated for the for the set of alert sequence <*1,1,2,2*>.

Based on our transition probability matrix A in section 4.4 the most likely future intention would be *V* with a probability of 0.75. Note that the output results are labeled and both the input alert sequences and output intention states decoded represent their respective states as per the labeling scheme in Fig 4.8 and Fig 4.11 respectively. For example for the preceding experiment involving alerts *<1,1, 2,2>*, the results implies that, if the network IDS detected the alert sequence (*<ICMP ping>, <ICMP ping>, <RPC portmap sadmind request UDP>,<RPC portmap sadmind request UDP)* the intruders sequence of intentions were (*<IP Sweep to determine which hosts are up >,<IP Sweep to determine which hosts are up >,<prob to look for sadmind daemons running>,<prob to look for sadmind daemons running>*) with a probability of 87.891/10^^4 and that the intruder is most likely planning to continue probing to looking for sadmind daemons running. The probability of his predicted most likely future intention is 0.75

4.5.1 Performance characteristics

In this section, we analyze the performance characteristics of our proposed system. Our first metric is the false discovery rate. The false discovery rate is the percentage of false discoveries for a set of decoded intention states. To do this, we decoded random sequences of alerts derived from the alerts generated in section 4.3. We then compared the output intention states with the actual intention states associated with those alerts as specified in the background information of the data. The results are tabulated if Fig 4.15

Sequence	Alerts	Output intention sequence	Expected Output intention sequence	False discoveries	Viterbi path probability	Future predicted intention(s) and probability
1	1,1	<UU>	<UU>	0%	$6.25x10^{\wedge -1}$	(U,0.25), (V,0.25)
2	1,1,4	<UUV>	<UUV>	0%	$23.44/10^{\wedge -3}$	(V,0.75),(W,0.25)
3	1,1,4,4	<UUVV>	<UUVV>	0%	$87.891/10^{\wedge 4}$	(V,0.75),(W,0.25)
4	4,4,2,2,2,4	<VVVVVV>	<VVVVWW>	33%	$695.22x10^{\wedge -6}$.	(V,0.75),(W,0.25)
5	2,7,3,3	<VWXX>	<WWXX>	25%	$4.712x10^{\wedge -4}$.	(X,1)
6	4,7,7,6,6,2,4,4	<VWWWWWWW>	<WWWWWWWW>	12.5%	$4.712x10^{\wedge -8}$.	(W,47/48),(X,1/48)

Fig 4.14: Summary of results for experiment 1 and 2

In all the above cases our system was able to discover the intruder's intentions. However, there were some false discoveries in some of the test sequences. We found that the false discovery rate was random and not tied to the input alert sequence size that we are decoding. This is illustrated in Fig 4.15 where the largest false discovery rate (33% from sequence 4) was not obtained from the largest sequence size while the smaller false discovery rates(0%-25%) were not necessarily obtained from the smaller sequence size.

An important performance characteristic of our proposed system is how the output intention states path probability varies with the input alert size. This is an important characteristic since in the design of a real system, an optimum size of input alert sequence to be decoded must be found. To test this characteristic, we analyzed and graphed the Viterbi path probabilities as tabulated in column 6 of Fig 4.15. The results were graphed as shown on Fig 4.16.

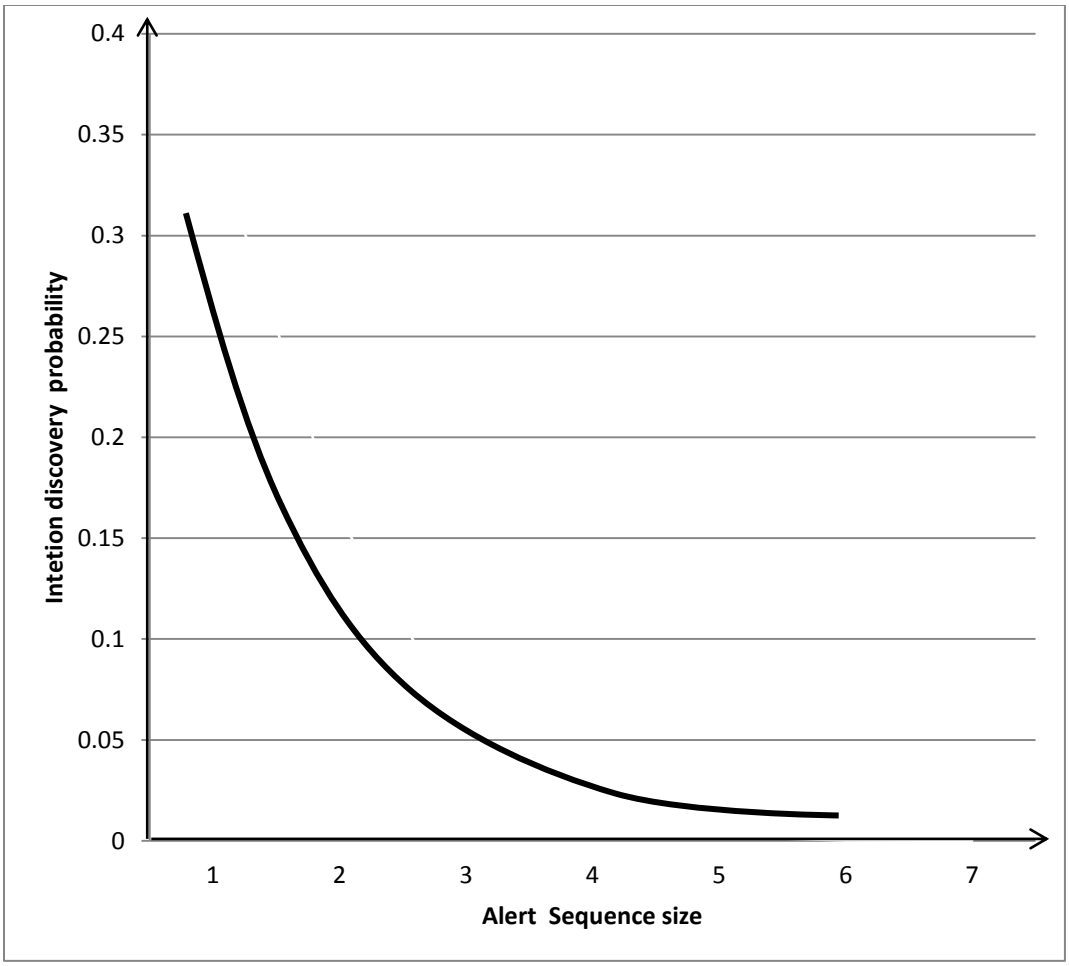


Fig 4.15: Graph of intention discovery probability vs alert sequence size

The results indicate that the bigger the input alert size we attempt to decode the lesser the intention discovery probability. In simple terms it means that for a bigger input alert sequence, the discovered intention sequence has a lower certainty. This is expected since the evaluation for the intention sequence involves multiplication of probabilities at each stage. Implementers of a real system will have a tradeoff between the input alert sequence they are attempting to decode and the level of certainty they expect from their results.

A significant drawback of our system is that its intention discovery capability is also tied to the accuracy of the underlying IDS. If the IDS is unable to return alerts for a given attack, or returns false alerts, then the discovered outputs may not be accurate. In our experiment, the Snort

IDS did not return any alerts for the last intention of the intruder (installation and execution of the DDOS script). Consequently, our model could not capture this stage of the attack since we had no data to train for this stage of the attack. As such, our system cannot discover previously unknown user intentions. This behavior is analogous to signature based IDS or antivirus software.

Chapter 5

Conclusions

5. CONCLUSIONS

In this thesis, we have outlined the idea of intruder intention discovery in IDS. We have demonstrated that using known information about an attacker's behavior, we can create HMM models and later decode the alerts from an IDS to discover the intruders high level intentions for the given alerts and predict the future intention. This system constitutes the intrusion intention discovery layer that is the subject of this thesis. Information about the intruder's intentions can be used both for recovery activities and for preemptive purposes in the case of the future intention predicted.

To validate our proposition we ran a simulation of a network environment under attack using Snort as a sensor and used the data to build a HMM model. We used a sample of the Snort alerts as input to a Viterbi decoder. The results of the experiments indicate that:

(1) Our proposition can discover intruders set of intentions for the given alerts already collected and from this, we can infer his future intention.

(2) For some input alerts, there may be false discoveries .We found that the occurrence of false discoveries was random and not tied to input size.

(3) The proposed system is sensitive to variation in the input size of the alerts. With a larger alert input size we seek to decode, the Viterbi path probability decreases. This means that although we still get the intention states, we have lower certainty. Hence, there is a tradeoff between the accuracy of the predicted intention states and the size of raw alerts used to discover the intruder's intention.

5.1 Future work

One of the biggest challenges that may be encountered in the implementation of this system is the training of the HMM model. HMMs are notoriously sensitive to changes in variables assigned to during training and since this determines the accuracy of output of the whole system, work should be done on an optimum training algorithm.

As seen in the results of our experiments in section 4, when a larger alert sequence is used as input to the Viterbi decoder, the output result has a lower probability. As such, work may have to be done on ways of finding an optimum alert sequence size that probably matches other factors such as threat.

If this system is to be deployed on a large scale, it may probably require the efforts of network administrators involved in network security worldwide preferably in an open source initiative. This is because the background information that is crucial in building the HMM model is collected from the networks themselves. This is the very approach that has been adopted in building the library of Snort signature libraries that enable the Snort IDS to function. Similarly, an architecture to enable building and submission of HMM models to a central database by individual network security personnel may be necessary. Any user will then be able to download files containing these already trained models and integrate the intention discovery capability to the network.

REFERENCES

[1] Rick Lehtinen, Deborah Russell, G. T. Gangemi, *Computer Security Basics,* 2nd edition, O'Reilly Media, 2006.

[2] Biswanath Mukherjee, L. Todd Heberlein, and Karl N. Levitt, "Network intrusion detection", *IEEE Network*, 8[3]:26-41, May/June 1994.

[3] *Snort*, http://www.snort.org, last accessed 4/1/2011.

[4] R. P. Goldman, W. Heimerdinger, and S. A. Harp, "Information modeling for intrusion report aggregation", *DARPA Information Survivability Conference and Exposition (DISCEX II)*, June 2001.

[5] S Elliot and S Fowell, "Expectations versus reality: a snapshot of consumer experiences with Internet retailing", *International Journal of Information Management*, vol. 20, pp. 323-336, 2000.

[6] J.P Anderson, "Computer Security Threat Monitoring and Surveillance", *Technical report*, James P Anderson Co., Fort Washington, Pennsylvania, April 1980.

[7] Dorothy Denning, "An intrusion detection model", *IEEE transactions on Software Engineering*, vol. SE-13, no. 2, pp. 222-232, February 1987.

[8] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition", *Proceedings of the IEEE*, vol 77, no 2, pp. 257-285, February 1989.

[9] J Viterbi, "Error bounds for convolutional codes and an asymptotically Optimum decoding algorithm," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 260-269, Apr. 1967.

[10] S. Forrest, S.A Hofmeyr, A. Somayaji, T.A Longstaff, "A sense of self for Unix processes", *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, Los Alamitos, CA. IEEE Computer Society Press, pp. 120-129, 1996

[11] T Lane, C.E Brodley, “Temporal sequence learning and data reduction for anomaly detection”, *Proceedings of Fifth ACM Transactions on Information and System Security*, vol. 2, no. 3, August 1999.

[12] Shrijit S. Joshi , Vir V. Phoha, “Investigating hidden Markov models capabilities in anomaly detection” , *Proceedings of the 43rd annual Southeast regional conference*, Kennesaw, Georgia March 18-20, 2005.

[13] R. Cohen, F. Song, B. Spencer, and P. van Beek, “Exploiting Temporal and Novel Information from the User in Plan Recognition”, *User Modeling and User-Adapted Interaction* 1(2), 125-148, 1991.

[14] C. Heinze, S. Goss, and A. Pearce, "Plan Recognition in Military Simulation: Incorporating Machine Learning with Intelligent Agents", *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI99) Agent Workshop on Team Behaviour and Plan Recognition*, Stockholm, pp. 53-63, 1999.

[15] A. G. Hofmann and B. C. Williams, “Intent recognition for human-robot interaction”, *Interaction Challenges for Intelligent Assistants*, pp. 60–61, 2007

[16] H. Herr, A. Wilkenfeld, “User-Adaptive Control of a Magneto rheological Prosthetic Knee”, *Industrial Robot: An International Journal*, Vol. 30, pp. 42-55, 2003

[17] W. Geib Christopher, Goldman Robert, “Plan recognition in intrusion detection systems”, *DARPA Information Survivability Conference and Exposition (DISCEX),* Vol. 1, pp. 46-55, 2001

[18] Li Feng, Xiaohong Guan , Sangang Guo, Yan Gao, Peini Liu , “Predicting the intrusion intentions by observing system call sequences”, *Elsevier Computer and Security,* Vol. 23, pp. 241-252, 2004

[19] Peng Wu, Yao Shuping, Chen Junhua, "Recognizing Intrusive Intention and Assessing Threat Based on Attack Path Analysis", *Multimedia Information Networking and Security International Conference*, vol.2, no., pp. 450-453, 18-20 Nov. 2009.

[20] Xinzhou Qin , Wenke Lee, "Attack Plan Recognition and Prediction Using Causal Networks" , *Proceedings of the 20th Annual Computer Security Applications Conference*, pp.370-379, December, 2004.

[21] Cuppens, F. Autrel, A. Mi`ege, and S. Benferhat, "Recognizing malicious intention in an intrusion detection process", *Proceeding of the Second International Conference on Hybrid Intelligent Systems*, Santiago, Chile, December 2002.

[22] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains", *Ann. Math. Statist*. Vol. 41, No. 1, pp. 164–171, 1970.

[23] *TCPdump*, http://www.tcpdump.org/, last Accessed: 4/1/2011

[24] Jaeius Lapitan , *A brief introduction to snort rules* , http://www.secanalyst.org/?p=98, last Accessed: 4/1/2011.

[25] *MySql Database*, http://www.mysql.com, last Accessed: 4/1/2011.

[26] *BASE System*, http://base.secureideas.net/, last Accessed: 4/1/2011.

[27] *Paulo Costa Carvalo*, http://pcarvalho.com/ , last Accessed: 4/1/2011.

[28] *Shrijit S. Joshi, Vir V. Phoha*, "Investigating hidden Markov models capabilities in anomaly detection", *Proceedings of the 43rd annual Southeast regional conference*, 2005,

[29] A Sundaram, "An Introduction to intrusion detection", *ACM Cross Roads*, Vol 2. No. 4, April 1996.

[30] R. Heady, G. Luger, A. Maccabe, and M. Servilla, "The architecture of a network level intrusion detection system*", Technical report*, Computer Science Department, University of New Mexico, August 1990.

[31] *Lincoln Laboratory Scenario (DDoS) 1.0* , http://www.ll.mit.edu/mission/ communications/ist/corpora/ideval/data/2000/LLS_DDOS_1.0.html, last Accessed: 4/1/2011.

APPENDICES

APPENDIX A: ALERTS AS DISPLAYED BY BASE

ID	< Signature >	< Timestamp >	< Source Address >	< Dest. Address >	< Layer 4 Proto >
#0-(18-224)	[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp portmapper sadmin port query attempt	2000-03-07 09:34:46	202.77.162.213:695	172.16.112.10:60548	UDP
#1-(18-217)	[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request	2000-03-07 09:34:46	202.77.162.213:694	172.16.112.10:111	UDP
#2-(18-218)	[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request	2000-03-07 09:34:46	202.77.162.213:694	172.16.112.10:111	UDP

APPENDIX B: ALERTS AT THE DMZ FOR HOST “MILL”

< ALERT >	< TIMESTAMP >	< Source Address >
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp portmapper sadmin port query attempt	3/7/2000 9:33	202.77.162.213:673
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request	3/7/2000 9:33	202.77.162.213:672
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request	3/7/2000 9:33	202.77.162.213:672
[arachNIDS] [snort] RPC portmap sadmind request UDP	3/7/2000 9:33	202.77.162.213:672
[arachNIDS] [snort] RPC portmap sadmind request UDP	3/7/2000 9:33	202.77.162.213:672
[snort] RPC sadmind query with root credentials attempt UDP	3/7/2000 9:33	202.77.162.213:673
[snort] RPC sadmind query with root credentials attempt UDP	3/7/2000 9:33	202.77.162.213:673
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp portmapper sadmin port query attempt	3/7/2000 9:33	202.77.162.213:673
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request	3/7/2000 9:33	202.77.162.213:671
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request	3/7/2000 9:33	202.77.162.213:671
[arachNIDS] [snort] RPC portmap sadmind request UDP	3/7/2000 9:33	202.77.162.213:671
[arachNIDS] [snort] RPC portmap sadmind request UDP	3/7/2000 9:33	202.77.162.213:671
[snort] RPC sadmind query with root credentials attempt UDP	3/7/2000 9:33	202.77.162.213:672
[snort] RPC sadmind query with root credentials attempt UDP	3/7/2000 9:33	202.77.162.213:672
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp portmapper sadmin port query attempt	3/7/2000 9:33	202.77.162.213:672
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp portmapper sadmin port query attempt	3/7/2000 9:33	202.77.162.213:672
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request	3/7/2000 9:33	202.77.162.213:670
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request	3/7/2000 9:33	202.77.162.213:670
[arachNIDS] [snort] RPC portmap sadmind request UDP	3/7/2000 9:33	202.77.162.213:670
[arachNIDS] [snort] RPC portmap sadmind request UDP	3/7/2000 9:33	202.77.162.213:670

[snort] RPC sadmind query with root credentials attempt UDP	3/7/2000 9:33	202.77.162.213:671
[snort] RPC sadmind query with root credentials attempt UDP	3/7/2000 9:33	202.77.162.213:671
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp portmapper sadmin port query attempt	3/7/2000 9:33	202.77.162.213:671
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp portmapper sadmin port query attempt	3/7/2000 9:33	202.77.162.213:671
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp portmapper sadmin port query attempt	3/7/2000 9:33	202.77.162.213:670
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp portmapper sadmin port query attempt	3/7/2000 9:33	202.77.162.213:670
[snort] RPC sadmind query with root credentials attempt UDP	3/7/2000 9:33	202.77.162.213:670
[snort] RPC sadmind query with root credentials attempt UDP	3/7/2000 9:33	202.77.162.213:670
[arachNIDS] [snort] RPC portmap sadmind request UDP	3/7/2000 9:33	202.77.162.213:669
[arachNIDS] [snort] RPC portmap sadmind request UDP	3/7/2000 9:33	202.77.162.213:669
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request	3/7/2000 9:33	202.77.162.213:669
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request	3/7/2000 9:33	202.77.162.213:669
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request	3/7/2000 9:33	202.77.162.213:668
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request	3/7/2000 9:33	202.77.162.213:668
[arachNIDS] [snort] RPC portmap sadmind request UDP	3/7/2000 9:33	202.77.162.213:668
[arachNIDS] [snort] RPC portmap sadmind request UDP	3/7/2000 9:33	202.77.162.213:668
[snort] RPC sadmind query with root credentials attempt UDP	3/7/2000 9:33	202.77.162.213:669
[snort] RPC sadmind query with root credentials attempt UDP	3/7/2000 9:33	202.77.162.213:669
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp portmapper sadmin port query attempt	3/7/2000 9:33	202.77.162.213:669
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp portmapper sadmin port query attempt	3/7/2000 9:33	202.77.162.213:669
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request	3/7/2000 9:33	202.77.162.213:667
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request	3/7/2000 9:33	202.77.162.213:667
[arachNIDS] [snort] RPC portmap sadmind request UDP	3/7/2000 9:33	202.77.162.213:667
[arachNIDS] [snort] RPC portmap sadmind request UDP	3/7/2000 9:33	202.77.162.213:667

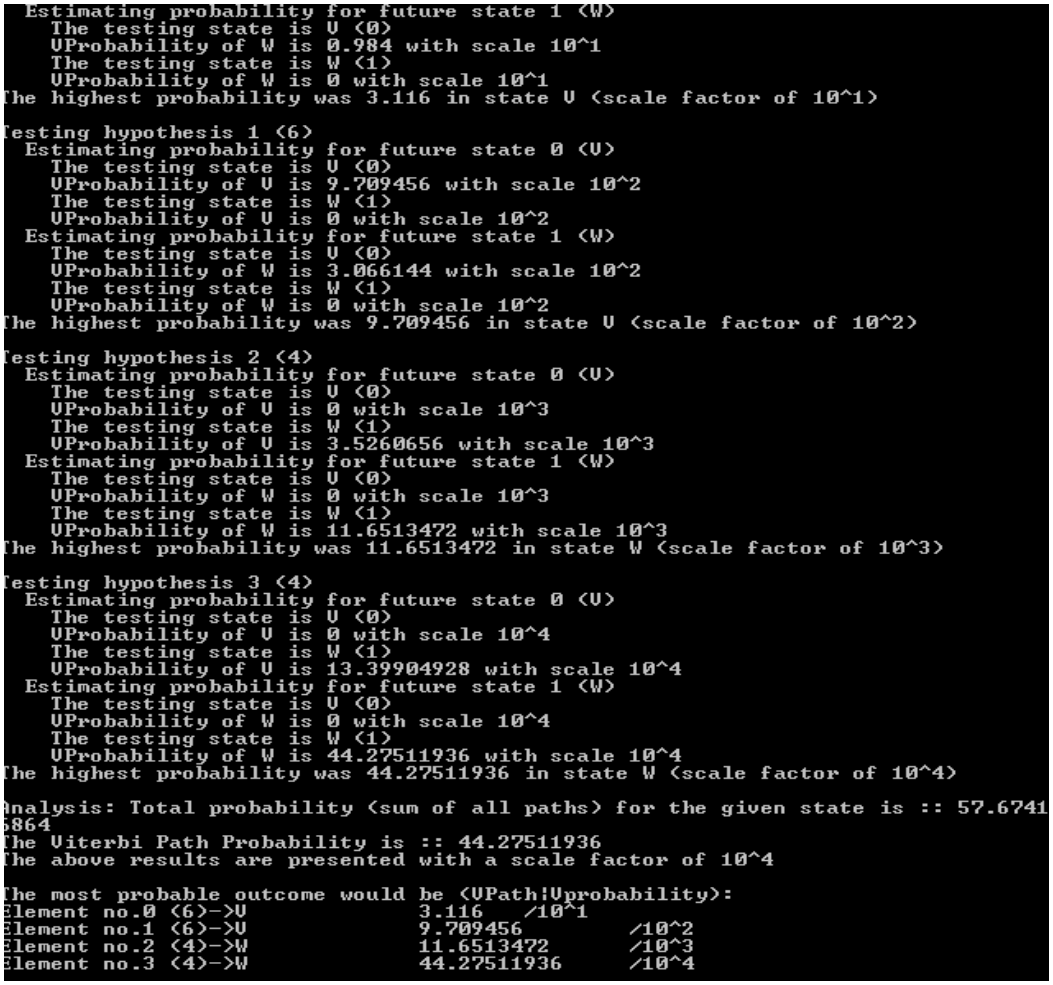
[snort] RPC sadmind query with root credentials attempt UDP	3/7/2000 9:33	202.77.162.213:668
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp portmapper sadmin port query attempt	3/7/2000 9:33	202.77.162.213:668
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp portmapper sadmin port query attempt	3/7/2000 9:33	202.77.162.213:668
[snort] RPC sadmind query with root credentials attempt UDP	3/7/2000 9:33	202.77.162.213:668
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request	3/7/2000 9:08	202.77.162.213:646
[arachNIDS] [snort] RPC portmap sadmind request UDP	3/7/2000 9:08	202.77.162.213:646
[arachNIDS] [snort] RPC portmap sadmind request UDP	3/7/2000 9:08	202.77.162.213:646
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request	3/7/2000 9:08	202.77.162.213:646
[snort] ICMP PING	3/7/2000 8:51	202.77.162.213
[snort] ICMP PING	3/7/2000 8:51	202.77.162.213

APPENDIX C: ALERTS AT THE INSIDE OF NETWORK FOR "*PASCAL*"

< Alerts >	< Timestamp >	< Source Address >
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp portmapper sadmin port query attempt	3/7/2000 9:34	202.77.162.213:60619
[arachNIDS] [snort] RPC portmap sadmind request UDP	3/7/2000 9:34	202.77.162.213:60617
[arachNIDS] [snort] RPC portmap sadmind request UDP	3/7/2000 9:34	202.77.162.213:60617
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request	3/7/2000 9:34	202.77.162.213:60617
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request	3/7/2000 9:34	202.77.162.213:60617
[snort] RPC sadmind query with root credentials attempt UDP	3/7/2000 9:34	202.77.162.213:60619
[snort] RPC sadmind query with root credentials attempt UDP	3/7/2000 9:34	202.77.162.213:60619
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp portmapper sadmin port query attempt	3/7/2000 9:34	202.77.162.213:60619
[arachNIDS] [snort] RPC portmap sadmind request UDP	3/7/2000 9:34	202.77.162.213:60603
[arachNIDS] [snort] RPC portmap sadmind request UDP	3/7/2000 9:34	202.77.162.213:60603
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request	3/7/2000 9:34	202.77.162.213:60603
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request	3/7/2000 9:34	202.77.162.213:60603
[snort] RPC sadmind query with root credentials attempt UDP	3/7/2000 9:34	202.77.162.213:60605
[snort] RPC sadmind query with root credentials attempt UDP	3/7/2000 9:34	202.77.162.213:60605
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp portmapper sadmin port query attempt	3/7/2000 9:34	202.77.162.213:60605
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp portmapper sadmin port query attempt	3/7/2000 9:34	202.77.162.213:60605
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp portmapper sadmin port query attempt	3/7/2000 9:34	202.77.162.213:60578
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp portmapper sadmin port query attempt	3/7/2000 9:34	202.77.162.213:60578
[snort] RPC sadmind query with root credentials attempt UDP	3/7/2000 9:34	202.77.162.213:60578
[snort] RPC sadmind query with root credentials attempt UDP	3/7/2000 9:34	202.77.162.213:60578
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request	3/7/2000 9:34	202.77.162.213:60576

[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request	3/7/2000 9:34	202.77.162.213:60576
[arachNIDS] [snort] RPC portmap sadmind request UDP	3/7/2000 9:34	202.77.162.213:60576
[arachNIDS] [snort] RPC portmap sadmind request UDP	3/7/2000 9:34	202.77.162.213:60576
[arachNIDS] [snort] RPC portmap sadmind request UDP	3/7/2000 9:34	202.77.162.213:60567
[arachNIDS] [snort] RPC portmap sadmind request UDP	3/7/2000 9:34	202.77.162.213:60567
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request	3/7/2000 9:34	202.77.162.213:60567
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request	3/7/2000 9:34	202.77.162.213:60567
[snort] RPC sadmind query with root credentials attempt UDP	3/7/2000 9:34	202.77.162.213:60569
[snort] RPC sadmind query with root credentials attempt UDP	3/7/2000 9:34	202.77.162.213:60569
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp portmapper sadmin port query attempt	3/7/2000 9:34	202.77.162.213:60569
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp portmapper sadmin port query attempt	3/7/2000 9:34	202.77.162.213:60569
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request	3/7/2000 9:15	202.77.162.213:56260
[cve] [icat] [bugtraq] [snort] RPC portmap Solaris sadmin port query udp request	3/7/2000 9:15	202.77.162.213:56260
[arachNIDS] [snort] RPC portmap sadmind request UDP	3/7/2000 9:15	202.77.162.213:56260
[arachNIDS] [snort] RPC portmap sadmind request UDP	3/7/2000 9:15	202.77.162.213:56260
[snort] ICMP PING	3/7/2000 8:51	202.77.162.213
[snort] ICMP PING	3/7/2000 8:51	202.77.162.213

APPENDIX D: OUTPUT SCREENSHOT FROM THE VITERBI DECODER



VITA

Jordan Kiprop Koskei

Candidate for the Degree of

Master of Science

Thesis: AN ATTACKER INTENTION DISCOVERY LAYER FOR INTRUSION DETECTION SYSTEMS USING HIDDEN MARKOV MODELS.

Major Field: Computer Science

Biographical:

Education: Graduated from Sunshine Secondary School, Nairobi, Kenya in November 2001; received a Bachelor of Technology degree in Computer Engineering from Moi University, Eldoret, Kenya in October 2008. Completed the requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University in July, 2011.

Experience: Lead Developer, Information Convergence Technologies, 2008; Network Administrator and Developer, The Daily O'Collegian from 2010 to July 2011.

Name: Jordan Kiprop Koskei Date of Degree: July, 2011

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: AN ATTACKER INTENTION DISCOVERY LAYER FOR INTRUSION DETECTION SYSTEMS USING HIDDEN MARKOV MODELS.

Pages in Study: 58 Candidate for the Degree of Master of Science

Major Field: Computer Science

Scope and Method of Study: Currently deployed intrusion detection systems (IDS) have no capacity to discover attacker high level intentions. Understanding an intruder's intention greatly enhances network security as it allows deployment of more accurate pre-emptive counter-measures and better disaster recovery. In this thesis, we propose a system where we model a known attack scenario using Hidden Markov Models and use alerts from an IDS later to discover an attackers set of intentions for a given set of alerts.

Findings and Conclusions: Experiments conducted shows that while the system can discover the attacker's intention, it is not always 100% accurate. There is an associated false error rate. The occurrence of false discoveries is purely random and not linked to other factors such as input size. The larger the input alert sequence size we seek to decode, the lower probability of the output intention sequence. More work needs to be done to develop optimum training algorithms for better models.

ADVISER'S APPROVAL: Dr. Johnson Thomas