AFRL-IF-RS-TR-2004-164
Final Technical Report
June 2004

HUMMER ENHANCEMENTS

University of Idaho

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. AON669 (00)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2004-164 has been reviewed and is approved for publication

APPROVED: /S/
W. JOHN MAXEY
Project Engineer

FOR THE DIRECTOR: /S/
WARREN H. DEBANY, Jr., Technical Advisor
Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	Jun 2004	Final Nov 02 – Dec 03

4. TITLE AND SUBTITLE

HUMMER ENHANCEMENTS

5. FUNDING NUMBERS

C - F30602-02-1-0165
PE - 62301E
PR - N669
TA - A6
WU - 10

6. AUTHOR(S)

Deborah Frincke, Hugh Pforsich, Jeff Harkins

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Idaho
PO Box 443020
Moscow, ID 83844

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/IFGB
525 Brooks Rd
Rome NY 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-2004-164

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: W. John Maxey, IFGB, 315-330-3617, maxeyw@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for Public Release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 Words)*

This document summarizes the work conducted by the Center of Secure and Dependable Systems at the University of Idaho related to DARPA-supported work from Nov 02 – Dec 03. This project was conducted to refine the Hummer Intrusion Detection System and to try new concepts in the system. These efforts involved a redesign of the Hummer architecture to support the suggested modifications. In this document, we present a summary of the changes incorporated into the Hummer Intrusion Detection System, along with the new concepts that were introduced into the system. Following this analysis of our work on Hummer is Dr. Hugh Pforsich and Dr. Jeffrey Harkins' assessment of their development of an analytic model for forensic analysis and relation of that model to a general predictive model for risk assessment and risk management. Their work was also supported by DARPA on the same grant, and used the Hummer prototype as their platform.

14. SUBJECT TERMS

Mobile Agents, Intrusion Detection Architecture, Forensic Analysis, Predictive Risk Analysis Model

15. NUMBER OF PAGES

63

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

Table of Contents

Table of Figures

Hummer

Task Agents

Hummer

Hummer is a collaborative distributed intrusion detection system developed at the Center for Secure and Dependable Systems at the University of Idaho. It employs a hierarchical system to gather and process data from distributed components. The Hummer intrusion detection system is composed of intrusion detectors as the base nodes. These nodes form the crux of a "neighborhood watch program." These "neighbors" keep a track of the system properties in their neighborhood, and communicate their reports to managers, which form the next layer of hierarchy. The managers communicate with peers, subordinates, and super managers. In most cases, a base node is related to one manager, but in special cases, systems can communicate with multiple managers as necessary. However, this technique is usually kept to a minimum, since managers can communicate with one another. They are generally special modules that can process information from the intrusion detection base nodes, and their intrusion detection capabilities are limited to the regions covered by the base nodes related to each individual manager.

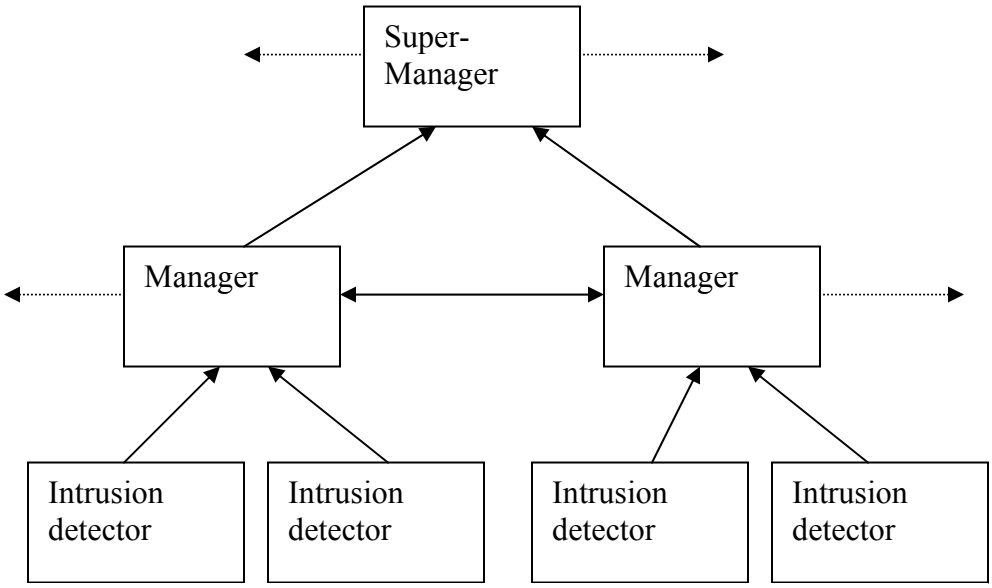


Figure 1: Figure showing hierarchy of components

Managers do have the capability to communicate with each other, which permits full comprehension of an intrusion scenario. However, this ability means that the managers need to be generalized and hence larger than if they were highly specialized. As a result, this hierarchical system incorporates the concept of superior managers receiving data from lower-level managers, which in turn gather additional information for further processing. These superior managers keep track of all information coming in from their underlings. The higher-level hierarchy managers have the same capabilities as a super manager and can also communicate with each other. The hierarchical system allows the system to incorporate as many multiple layers of super managers as necessary.

The operation of super level managers in the third hierarchy depends on the system complexity required. For instance, they could be employed in special intrusion detection cases that require multiple managers to process data in conjunction with one another. Super-managers distributed across the network can communicate with each other and keep each other updated about the attacks.

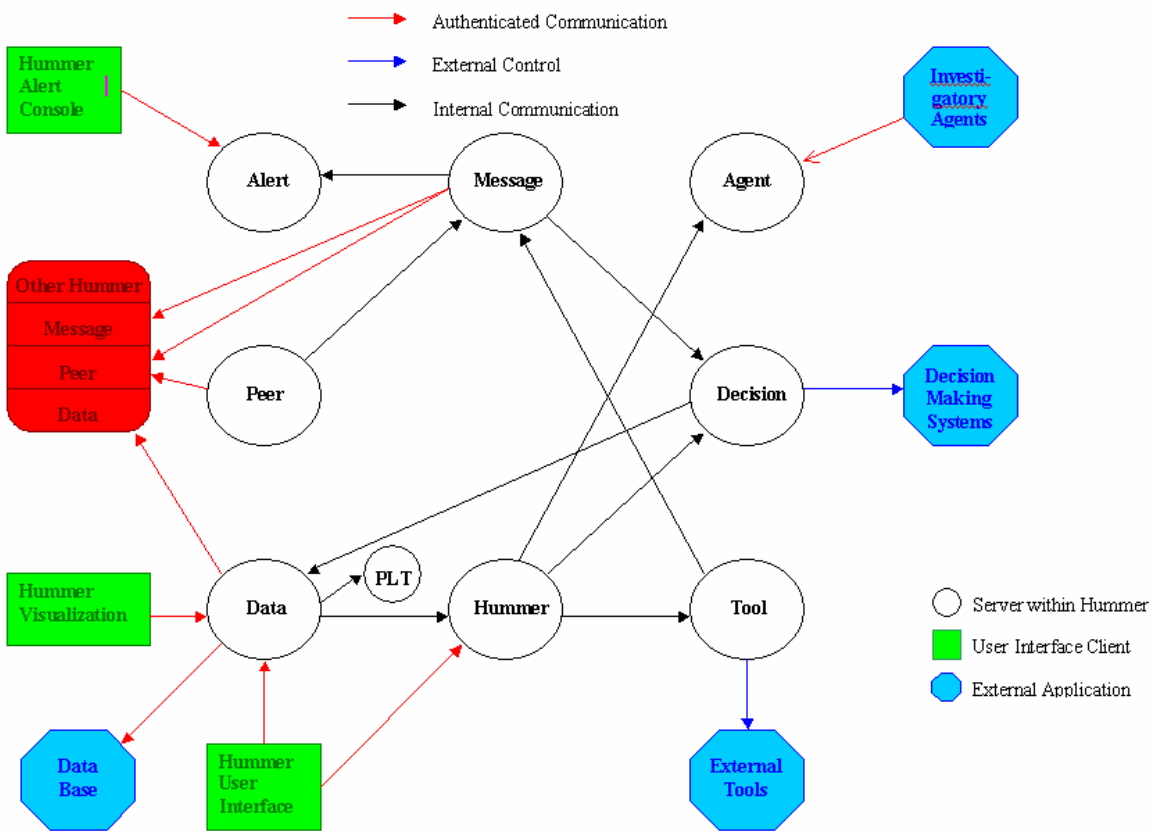


Figure 2: Earlier Hummer Data-flow Diagram

The figure above shows the data flow between the different components in the earlier versions of Hummer. In this context, some of the important components were the mobile agents, the DMS, Tools, servers (including the Hummer server) and the console. Given this framework, the DARPA project's primary effort between November 02 and December 03 was to extend the capabilities of the Hummer intrusion detection system. To this end, we proposed a number of possible enhancements to increase system efficiency and the flexibility of the configuration shown in figure 2. These proposed enhancements are listed below:

Incorporation of new capacities based on preliminary results

- Decision Tree capacity combined with mobile information-gathering agents to permit more precise control over data gathering and distributed responses to current conditions.
- Significantly expanded filtering system, including improvements in assessing level of risk associated with sharing a given data item and modifying behavior on that basis: for instance, through data sanitization.

Extending current capacity

- Significantly expanded implementation of the architectural concept "trust integrity cooperation" (TIC) model, and integration with both data collection and system response to threat (internal or external).
- Addition of more tools for data gathering and assessment, including those commercially available.

Research and Prototype

- Modification of the current decision-management system to permit more flexibility in terms of strategy combinations used.
- Incorporation of "forensic and audit friendly" aspects, allowing the system to be used more easily by auditors, law enforcement, and others to determine whether previous entity behaviors match system policy.

- Exploration and preliminary prototyping of how an "evolvable defense" strategy for organizing the defense of a large collection of cooperating systems might be implemented within the Hummer environment.

We extended the proposed enhancements to the Hummer intrusion detection system to achieve the following by the end of the research project:

Prototype

- We added a Decision Module to perform intrusion signature analysis to the existing DMS module.
- We added capabilities to the intrusion analyzing DM that permitted us to activate intrusion responses. This process was a crucial step towards achieving the evolving defenses goal.

Incorporation of new capacities

- In the event a Hummer node is compromised, the entire system could be compromised by the introduction of malicious information designed to change the event analysis. As such, we needed to determine how much to trust or believe communication with another Hummer. For this purpose, we added the BC module that keeps track of the "trust factor" with the entity in communication.
- The TIC concept was integrated into the BC module to keep track of the Believability levels of trust with a remote system/Hummer and the believability value of the outside data's integrity.
- Given that Hummer systems can communicate with each other across networks where data could be susceptible to interception, it is necessary to ensure that the data does not contain exploitable information (for instance, the type of operating system on a particular machine). For this purpose, we added a sanitization process to avoid information leaks.

Extending current capacity

- Given that the nodes were passing all messages generated, a reasonable methodology needed to be worked out which would reduce the information redundancy while maintaining the integrity of the data.
- Typical IDS tools create a large volume of messages. As Hummer is a message-passing system, the importance (or cost) of messages must be determined. For this purpose, we added a tool that keeps track of information flow.
- An intrusion response module was designed to coordinate responses for recognized computer attacks.

Term	Definition
BC	BC refers to Believability and Cost. Believability represents a trust value. The Cost value represents an estimate of resources to be spent processing a message.
CVE	Common Vulnerabilities and Exposures: A list of standardized names for publicly known vulnerabilities and other information security exposures.
DMS	Decision Management System. A grouping of decision modules that affect actions within the system.
DM	A Decision Module. These modules are pieces within the DMS, and are written in the language of choice for the DM.
PLT	Perceived Level of Threat. The likelihood that a Hummer will be attacked.
GUI	Graphical User Interface.

Information Gathering Refinement

Researcher: Stephen Tratz, Alex Oden.

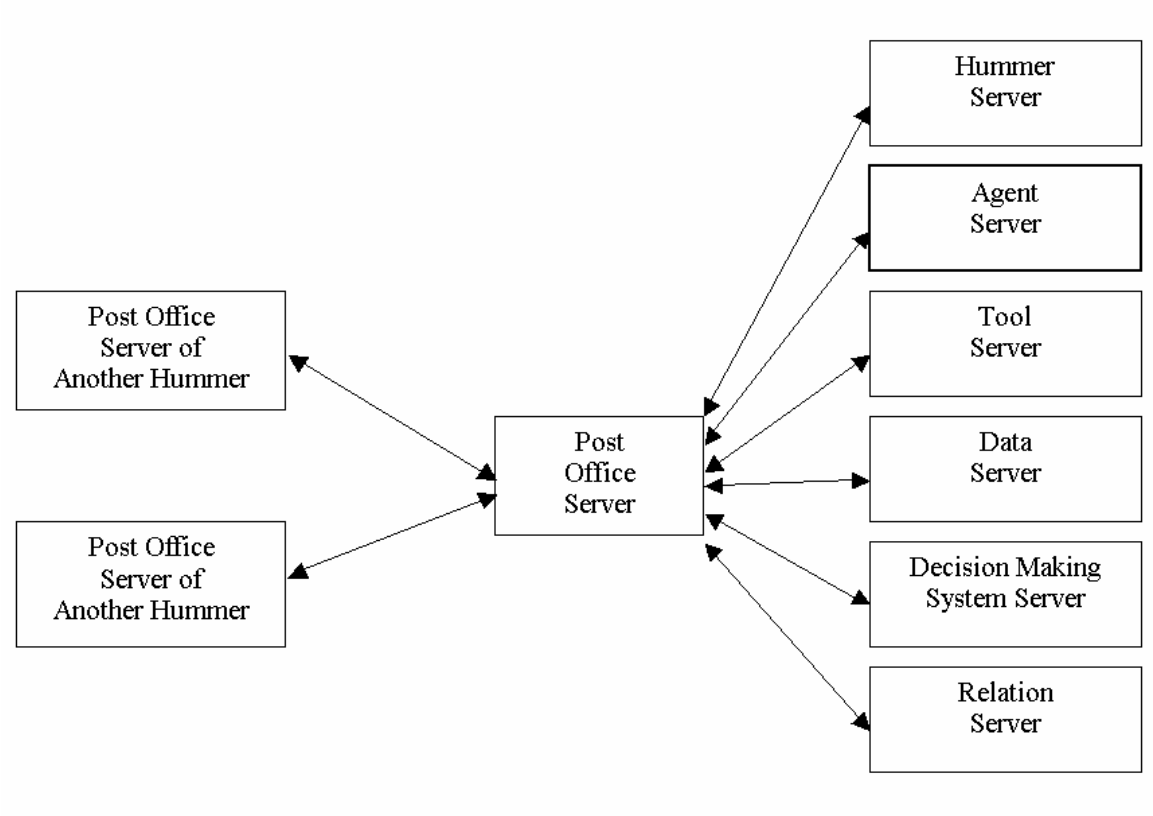


Figure 3: Revised Hummer Dataflow structure

To refine the messaging sub-system, we changed the Hummer architecture as shown in the above figure. We removed duplicate entries, thereby making the Post Office act as the center of the messaging system. This structure's primary advantage is that priority processing can now be established, with higher priority messages being dealt with before lower priority messages. For the purpose of this research, external messages were held with lower priority in comparison with local server messages. Previously, the post offices communicated with external Hummer systems. To form a connection, a Hummer registers with the post office of another Hummer, from which it receives messages. As an external Hummer system is not interested in the internal details of outside messages, we established a message filtering and message sanitization system. The Hummer Architecture Document has further details on this system.

Information Filtering

Belief Server

Researchers: Caleb, Al Carlson, Taylor.

This section deals with the Belief module and the research related to it. This module is an essential part of the main Hummer system as it controls the information flow between Hummers.

Introduction

While formal Belief Logic deals with the existence of belief, and attempts to describe what belief is, very little literature has been written to address the actual mechanics of belief. Most often, belief is expressed in the form of either First Order Predicate Logic (FOPL) or rule based sentences [Konolige1986]. Such forms do not readily lend themselves to precise definitions of matching belief to numbers [Lee1993]. Another form of belief is expressed simply in the Bigley Belief Measure. The Bigley Belief Measure uses an equation to determine the believability of information based on the path that the information traveled. This is merely one belief measure and only allows for the quantification of a single belief. Addressing the problem of making a quantified model of belief is multifaceted and not trivial in nature [Park1987].

To make it easier to understand and discuss the creation of a quantified system of belief, we will characterize systems in terms of the recipients, because their set of beliefs will be reflected in the final belief structure. Systems will be anthropomorphized and be referred to as "people." Individual beliefs can be applied to any object of interest. For purposes of this report, we will define objects of interest as other people (systems) and information.

The range of beliefs, changing beliefs, and the construction of a new belief are important to instantiating a quantified model of belief. Complicating the problem of quantification is that there is no verifiable standard for belief - individuals believe different things with

varying intensity. Each person sets their own belief level and belief criteria. In addition to the lack of a standard belief level, there is also the problem of creating and changing the level of belief. Affecting the beliefs of others is beyond the scope of any research yet located and is not addressed. In this discussion, the only beliefs that we can affect are the individual beliefs of the person in question.

In order to construct a belief, we must first define the foundations of belief. There are many aspects in creating and adjusting a belief level in an object. We need to quantify these factors and determine a method of combining them to create a level of believability. Quantifying belief begins with understanding the representation of belief logics in FOPL and rule-based sentences.

Belief Logic Representation

FOPL is a totally symbolic representation of an assertion or question. Key components of the language include symbols that carry the specific meaning of 'there exists a something such that' (∃) and 'for all of something there is' (∀) used to describe relationships. These are combined with functions, symbolic references, bindings, and a strict interpretation of order [Konilige1984; Konilige1986]. Konilige defines a belief being represented in FOPL as [S]φ, meaning that an agent S believes in an event, or object, represented by φ. An example is ∃x.[HP]M(x) which means that there exists some person x such that Hercule Poirot (HP) believes that x is the murderer (M(x)). Other than this definition, standard definitions of FOPL apply. This logic notation denotes the relationship between objects, but does not describe the mathematics to manipulate the objects, and is cumbersome to program. FOPL is best used as a universal notation and for documenting assertions in papers.

Rule-based sentences are Prolog-like in structure. Functions have names and definitions. Facts are shown as simple functors with arguments, while functions are series of functors that are temporally applied embodying the full conditions for the relationship. For example, brother(Ralph,Harry) indicates that Ralph and Harry are brothers. A more complex example would be:

```
inlaw(Mike,Susie) :-  brother(Mike,George),
                      married(George,Susie).
```

In this case, Mike and George must be brothers and George must have married Susie. While computationally efficient, this notation can become difficult to read when the number of variables is large or the relationship is complex.

Literature is widely available and comprehensive on both representation methods. Both representational methods are common and well understood. Each has attributes that recommend it in different situations. Printed versions of belief are better accomplished in FOPL while computational instantiation is better done in a rule-based form.

The Lone Computational Measure

The Bigley Belief Measure (BBM) is an example of what must be done in order to create a quantifiable model of belief. The BBM says that the belief in an object, or communication, degrades with the number of links it passes through. This type of degradation can be seen in the game of "telephone," where one person whispers a message to another individual, who then whispers it to another. The message thereby progresses in a linear fashion, but does not necessarily retain its meaning. A measure for the confidence in the reliable transmission of the message between individuals is expressed as a number between 0 and 1. The number of links traversed is given as n, and the resultant belief is calculated as:

$$C_n = B^n$$

Where B is the Bigley Belief Measure, C is the confidence in the final communication, and n is the number of links. BBM is interesting, but is limited in application. It can be used to assess the affect of transmission, but does not address the actual believability of the source of the communication.

Implementation

We constructed our present implementation of Hummer's belief system upon past research and infused it with our own theories about belief operations. The Hummer belief unit is essentially a graph with many internal links connecting nodes of information. Each node in the graph represents a system running Hummer, or a service running inside Hummer that has the capacity to communicate with the Belief system. The representational nodes contain values that store information such as the node name, confidence level, and event counter. The system calculates the confidence of a node by assessing the number of events that the node registers at a particular time. The user can set the value (or "worth") of each event, and thereby calculate the confidence rating through policy. A node's confidence within the system decays over time if no activity for the node has been recorded over a set interval. The user can set the decay time value used by the logarithmic decay function so that a finer level of control can be achieved for each deployment of the system.

The information collected and stored inside the belief graph can be used for a variety of purposes. Currently, when a node registers an event, it sends its confidence level to the response unit for evaluation. The true power of the Belief system arises through its ability to query the system for a benefit/detriment value. The system calculates benefit/detriment values using current node confidence levels and benefit/detriment information that the user enters into the system with policy settings. The benefit/detriment values can be intensive to calculate if the system is queried frequently,

but they give a valuable automatic evaluation of what actions the system deems beneficial or not.

Conclusion

A survey of the current literature reveals a small body of study available. While it addresses default values of belief and belief representation, none of the references deal directly with the calculation of how to set belief or change it in response to stimuli. Belief depends on a set of believability measures that are in turn directly related to constituent components of an agent. Default beliefs are those assigned to an unknown agent. They constitute a measure with which initial decisions must be made. Default values should be set with as much information as is possible on the new agent.

Creating a quantifiable model for belief that is tailored to the needs of the people using the beliefs requires some analysis. However, the most useful beliefs can be generated from careful selection of factors and keeping a history on those factors. The ability to combine the separate factors into one belief value requires that the factors be related and calls for an n-tuple. The scale of belief and changing belief should be made complex enough to be useful in as many situations as possible without making it difficult to implement. The process of calculating belief involves taking into account the history of each factor, which may be large or small, and may be computationally expensive. Consideration should be given to the optimization of these calculations and to careful selection of the factors to use.

Decisions can be made using one of three models, continuous believability measure, discrete measure using two values (Trust and Don't Trust), or a fuzzy implementation of as many values in a gradient as desired. Any of these models is usable, though the two-value discrete model does not allow enough flexibility in practice.

BBM is useful--though limited--in assessing the links between agents. It represents one of the few mathematical treatments of belief.

Experimentation is the only way to develop weighting measures to calculate belief. Though hooks need to be placed in any belief instantiation, only a few measures that give a reasonable approximation of belief should be implemented at this time.

References:

[Bacchus1988] Bacchus, Fahiem Ivor, Representing and Reasoning with Probabalistic Knowledge, PhD Dissertation, University of Alberta, 1988

[Cohen1984] Cohen, Paul Raymond, Hueristic Reasoning About Uncertainty: An Artificial Intelligence Approach, PhD Dissertation, Stanford University: 1984

[Collins1990] Collins, Glen Camp, Adaptive Reasoning for Belief – Function Based Diagnosis, PhD Dissertation, Vanderbilt University: 1990

[Darwiche1983] Darwiche, Adnan Youssef, A Symbolic Generalization of Probability Theory, PhD Dissertation, Stanford University: 1983

[Goodwin1991] Goodwin, Scott D,. Statistically Motivated Defaults, PhD Dissertation, University of Alberta: 1991

[Konolige1984] Koniloge, Kurt, A Deduction Model of Belief and Its Logics, PhD Dissertation, Stanford University: 1984

[Konolige1986] Konilige, Kurt, A Deduction Model of Belief, Morgan Kaufmann Publishers, Inc, Los Altos, CA: 1986

[Lee1993] Lee, Suk Kyoon, An AI Approach to Relational Data Models for Uncertain and Imprecise Information, PhD Dissertation, University of Iowa: 1993

[Park1987] Park, Jae Kuen, A Comparison of Reasoning Methods with Uncertainty for Expert Decision Support System, Doctoral Thesis, PhD Dissertation, Florida Institute of Technology: 1987

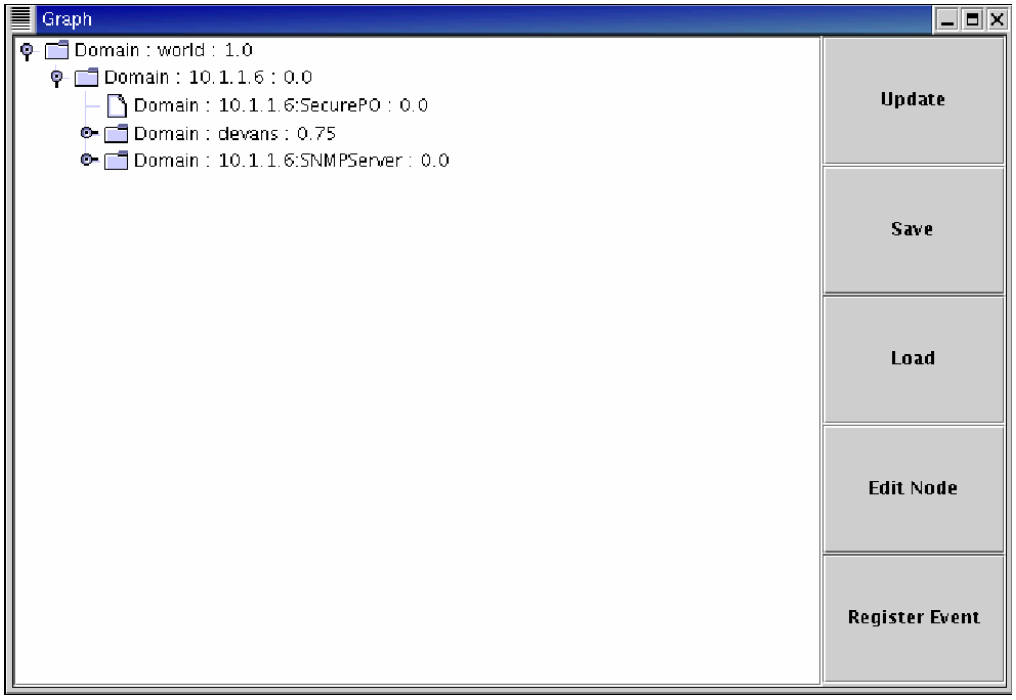


Figure 4: Graph before event

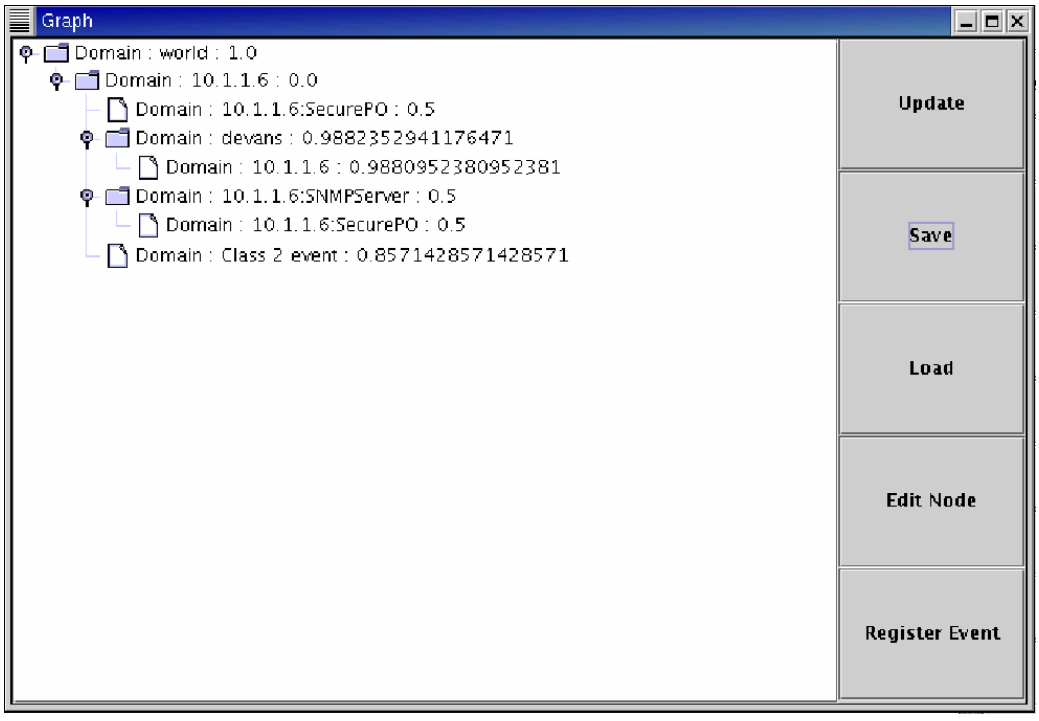


Figure 5: Graph after event

The above two figures present the change in the graph information upon the occurrence of an event. In the first figure, the graph depicts the initial belief values. Upon reception of an event, the values are modified as shown in the second figure.

Information Sanitization

For purposes already described in this document, we want the information outflow to be restricted and abstracted so as to reduce risk of broadcasting system-sensitive data. To this end, the sanitizer strips outgoing messages and thereby prevents the sharing of sensitive information to destinations that the belief server deems untrustworthy. For the purpose of this research, only alert messages have been stripped. Other messages need to retain all their fields in order to function properly, and therefore cannot be stripped. However, since the external Hummers are likely to be interested in the alert information, the role of the sanitizer suits our purposes. Modifications to policy determine the amount of stripping necessary (ranging from dropping the entire message to modifying one field). The Sanitizer also incorporates a message logger, which will log the headers of every external message as well as keep the entire message payload in binary form for the previous 50 messages that have passed through the system. This feature is useful in establishing the communication pattern for the concerned entity.

Decision Making System

Researchers: Darin Evans, Kevin Schafer, Alex Oden.

This section outlines the basic functionality, interface and working of a Decision Module (DM) that we designed to ”plug into” the Hummer system. DMs perform data analysis, reporting, sharing, and response duties within Hummer. They are essentially the “brains” of Hummer, deciding when incoming audit data and events correlate to an attack, what notifications should take place as a result, and if or when any specific responses should occur. DMs make use of a common interface API; therefore, it does not matter which programming language(s) may have been used to create the DM. In Hummer we have primarily implemented DMs with Java. However, we do have a Prolog-C interface DM

for intrusion analysis and response. A DM only needs to fulfill the data exchange interface requirements as outlined the DM Design guide Document [DMDesign].

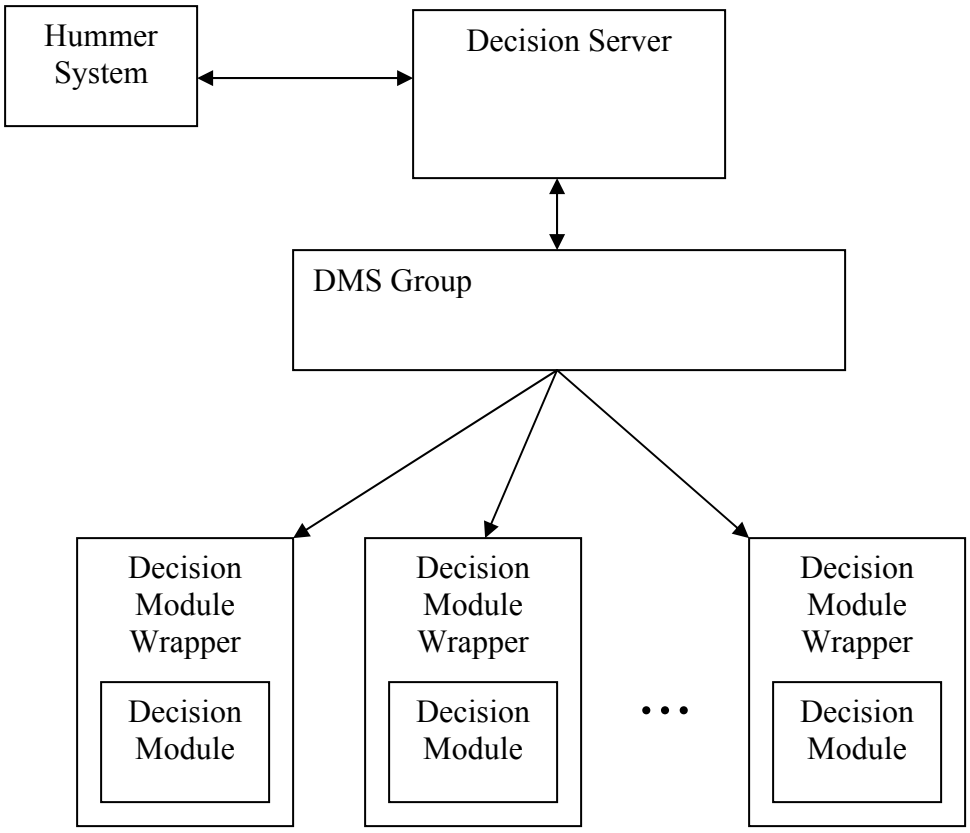


Figure 6: DMS Overview

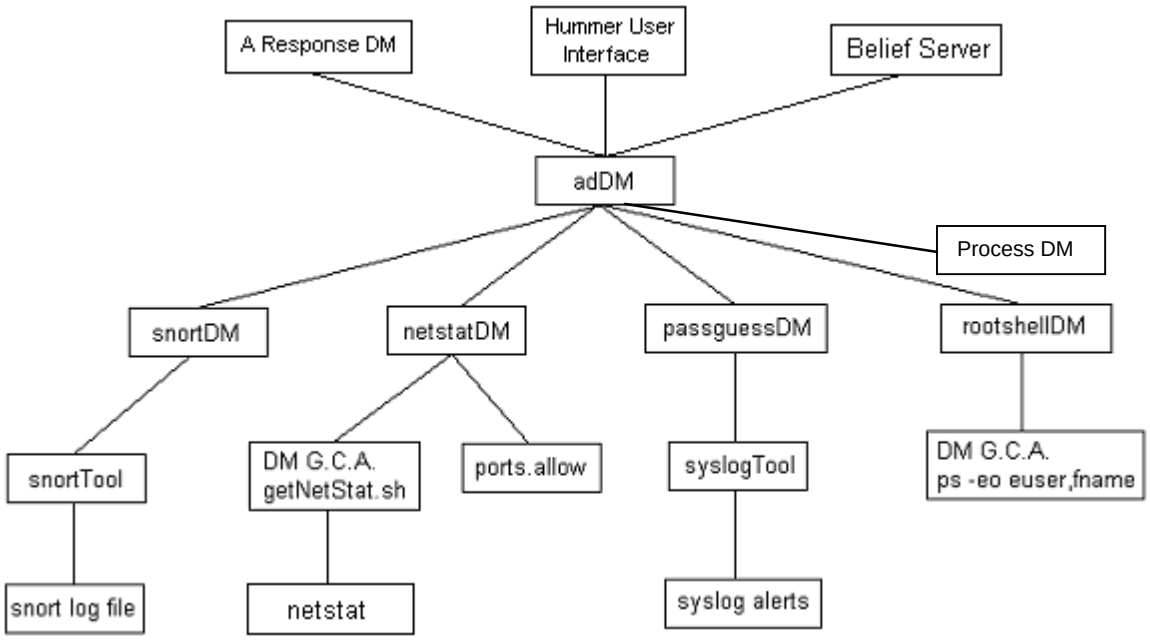


Figure 7: DMS employed in Hummer

The first figure in this section shows an overview of the decision making system, whose four components are the Decision server, DMS group, DM Wrapper, and the decision modules in which the *Decision Server* acts as the interface for the decision group with the rest of Hummer. The *DMS group* contains methods that operate on multiple decision modules at once. For evaluation purposes, we tested this feature but did not use it extensively. The decision group stores a list of known Decision Module configurations in the database. *DM Wrapper* is the module that directly interfaces with the decision module. The *Decision Module* creates the new process that the decision module runs in, handles delivering messages to the input of the decision module, and parses the resulting output from the decision module. Data received from the decision module is run through the DataParser object, and then delivered to Hummer as a message. A DM can be implemented in any language or as a script, as long as the DM can accept input and print output to the command line.

The second figure is a representation of the DMs employed in the current Hummer system.

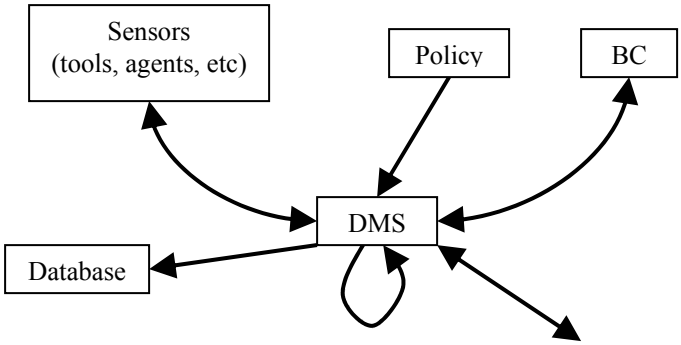


Figure 8: DMS interactions

In this section, we present a brief discussion of the DMs employed by the Hummer intrusion detection system. We present the flow of data between DMs and other parts of Hummer. Discussion on the configuration is available in the referenced documents

AdDM []

Input: received from low-level DMs: snortDM, netstatDM, passguessDM, and rootshellDM.

Output: messages to a response DM, the belief server, and the Hummer UserInterfaceAgent.

When adDM receives a message from one of the previously mentioned DMs, it distributes messages to three systems in Hummer: the Belief Server, Hummer UserInterfaceAgent, and to a response DM which performs actions within Hummer and the system it's running on (depending on which messages it receives). See Figure 1 for a diagram.

SnortDM []

Input: received from snort alert lists and the snort tool.

Output: message to a high-level DM. (adDM)

When an event occurs that snort recognizes, it writes a notice to the snort log file. The snort tool checks this log file and sends alert messages to snortDM, which is subscribed to the snort tool alert list. SnortDM handles the report. If there is a possible attack, snortDM sends a message through Hummer to the high-level DM it has been configured to report to (adDM).

Figure 9: Snort DM

NetstatDM []

Input: received from getNetStat.sh through the DM G.C.A. and the ports.allow file.

Output: message to a high-level DM. (adDM)

On timed intervals, a tool runs getNetStat.sh. getNetStat.sh uses the netstat command to gather information and returns the data to the DM G.C.A. The DM G.C.A attaches an IP address to the output and sends the report to netstatDM. NetstatDM compares the data in the message to data that it gets from the ports.allow file. The DM then handles the report and sends a message(s) through Hummer to a high-level DM if something is suspicious.

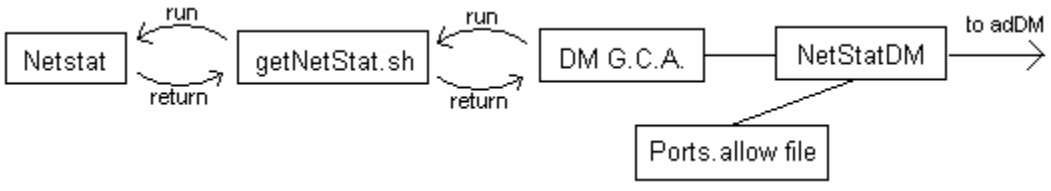


Figure 10: NetStat DM

PassguessDM []

Input: received from syslog alert lists that it is subscribed to.

Output: message to a high-level DM. (adDM)

When an event occurs, it is stored in the system logs. The syslog tool gathers the information from the system logs at "/var/log/messages" and then sends out an alert message, which is received by the DMs that are subscribed to the syslog alert list. PassguessDM receives the alert message, processes it and sends a message.

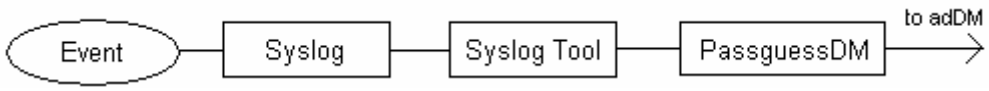


Figure 11: PassguessDM

RootshellDM []

Input: received from a shell command run by the DM G.C.A.

Output: message to a high-level DM. (adDM)

At timed intervals, a DM G.C.A. runs the command "ps –eo euser,fname" to gather process information. The DM G.C.A attaches an IP address to the information and sends the report to rootshellDM. If there is a possible attack, rootshellDM handles the report and sends a message(s) through Hummer to a high-level DM (adDM).

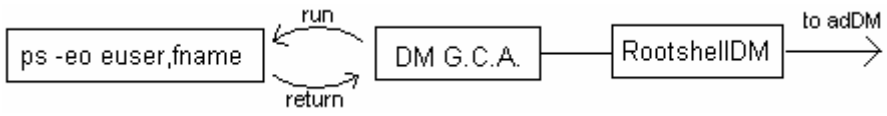


Figure 12: RootshellDM

Intrusion Response Module

Researchers: Timothy Meekhof, Donghui Yang, Kalyan Indoori.

Response DM

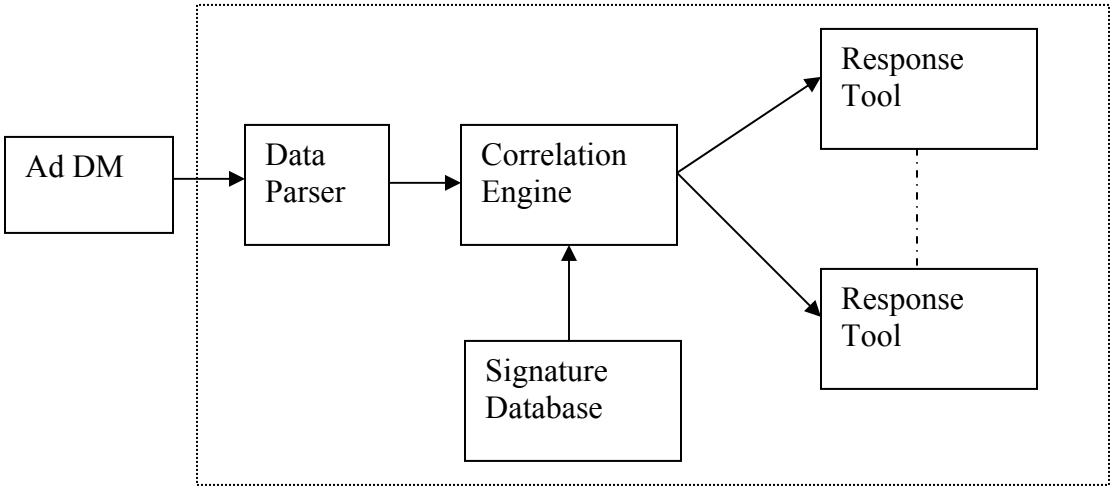


Figure 13: Response DM model

The intrusion response module is

Hummer Toolkit

We designed some new tools for the Hummer toolbase for use in addition to the existing set of tools.

Hummer Toolkit

The comprehensive list of tools can be classified into two categories:

1. Information gathering tools
2. Intrusion response tools

In the Information gathering section we include the following tools:

a. Snort tool: This tool interfaces the snort intrusion detection system with Hummer. It reads in the logs from snort, parses through the information for relevant messages, and passes the information.

b. Apache tool: This tool parses the apache system log to detect intrusions.

c. Syslog tool: This tool acts as an interface between Hummer and the Syslog log.
d. Login tool: This tool monitors intrusions such as doorknob diddling.
e. Process tool: This tool monitors the processes running in the system, and detects intrusions.
f. Tripwire tool: The tripwire tool is an interface with the tripwire application.
g. Windows login tool: This tool monitors windows logins to monitor failed login information.

Intrusion Response tools:

a. IPTables Tool: This tool makes use of the iptables firewall to control the connections.

VRRP Server

This server is responsible for communication with a vrrp server. It is useful when there are a series of identical systems, as VRRP allows the manager to switch seamlessly between the servers. This process enables the administrator to switch out malfunctioning machines while they are fixed, with no downtime.

IPTables Server

This server is responsible for communicating with IPTables on the Linux machines. This communication becomes useful when the system determines that an IP address or port must be stopped. In addition, communication with the pieces can be restored after a set time period has passed.

Hummer Communication Test

Researchers: Hummer Team

In this section we present Hummer communication test results.

Setup:

The test set-up was the simplest scenario, in order to minimize hidden variables and other unknown events, which would affect the actual maximum message threshold through and between Hummers.

Common system Configuration: Redhat Linux 7.2, 100Mbit switch, 850mhz Athlon, Located on the CSDS test bed.

Configuration 1: External Communication – One computer running the netstatDM, its agent and reporting to a dumpDM which is located on a second machine.

- Tests the external communication between Hummers.

Configuration 2: Internal Communication – One computer running the netstatDM, its agent and a dumpDM, which netstatDM reports to.

- Tests the internal communication inside of a single Hummer.

Results:

The graphs below represent the number of messages per second that are sent through or between Hummers. The x-axis on both graphs is the time in seconds, and the y-axis is the messages sent during per second. The graphs were then sent through a Bezier smoother in Gnuplot in order to convert the data points into lines.

External Communication

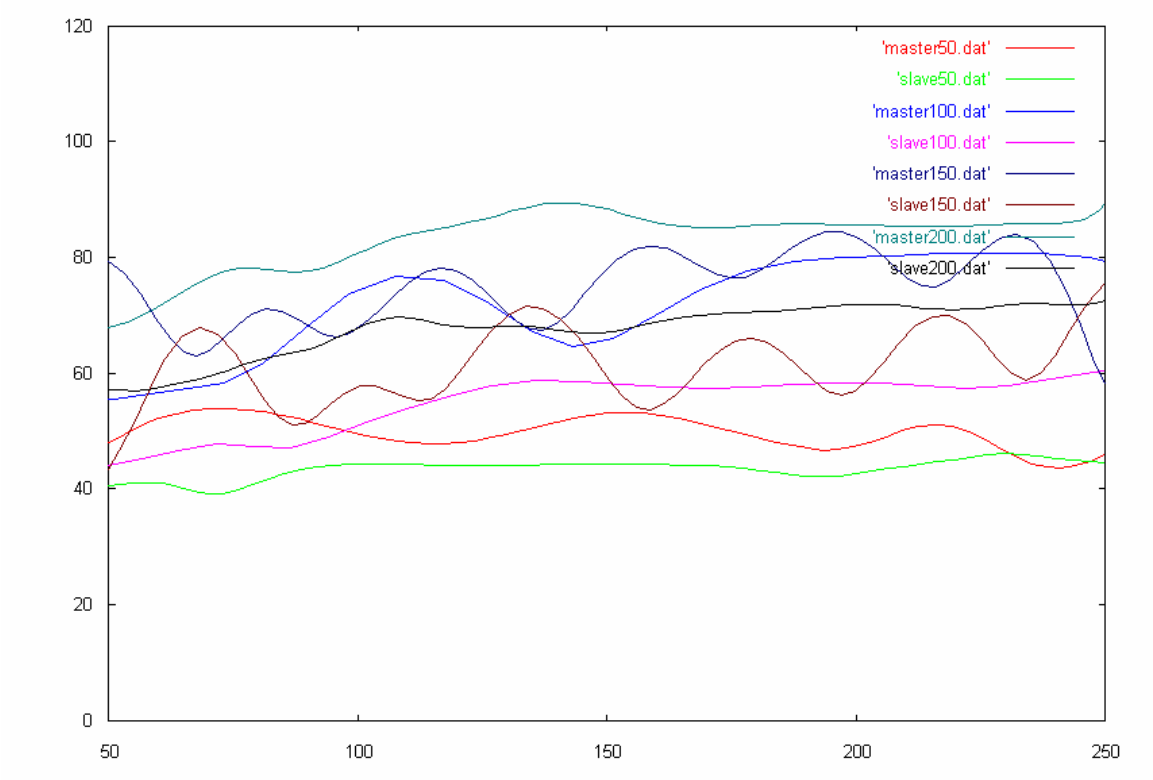


Figure 14: Average external messages per second

The external communication is shown above. The graph shows that as the messages accumulate in the netstatDM queue, a few more messages will be sent out each second. The data sets are defined beside the lines, so the master50.dat line is based on 50 open ports and logged from the computer hosting the DM. Therefore, the red line should be at 51 messages per second if all messages are getting through, 50 messages for open ports, and one for an agent message each second. Similarly, the green line is the log from the receiving machine where the output for the open ports is sent. This line should also be at 50. Conclusions made from this data indicate that a queue inside Hummer ensures that messages will pass the master's post office prior to reaching the slave's post office. The graph also demonstrates that the maximum number of messages between Hummers

seems to be slightly less than fifty. At 50 messages per second, there is a discrepancy between the number of messages sent and received, but it is rather small. In addition, the lines seem to deviate significantly from the messages that should be sent and the messages actually sent when the number increases.

Internal Communication

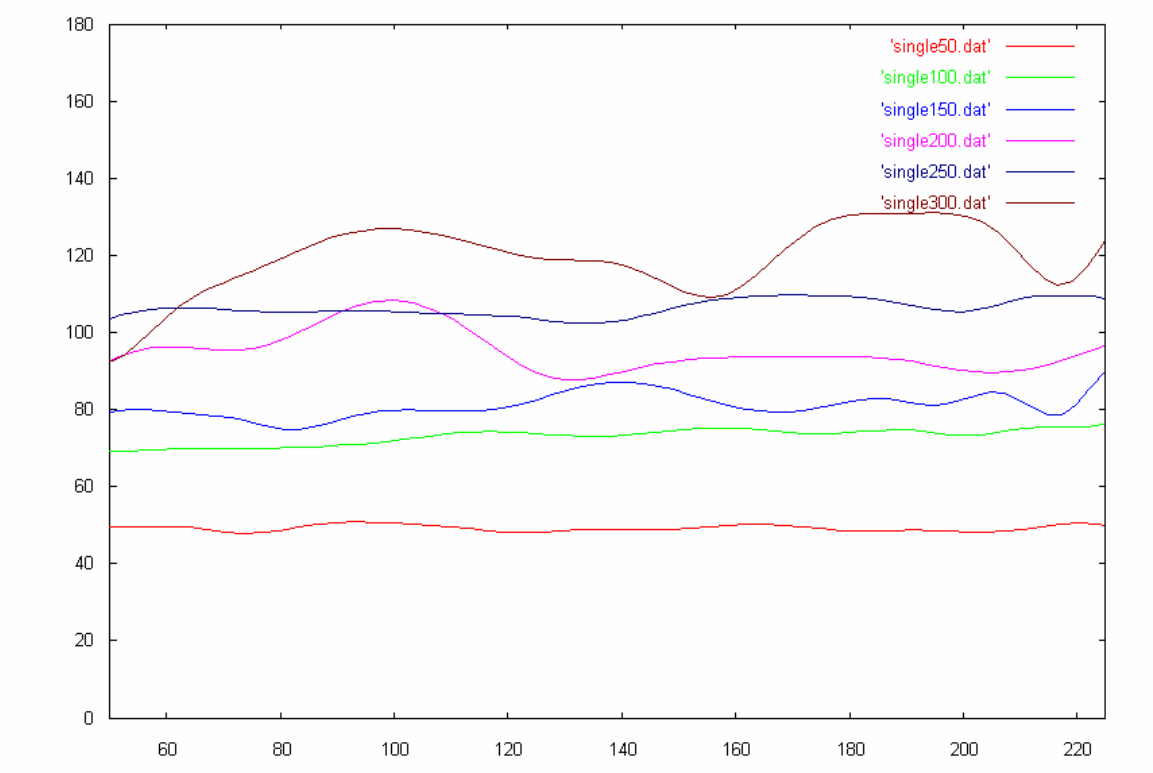


Figure 15: Average internal messages per second

The internal communication is shown above. The graph demonstrates that as the messages queued up in the netstatDM accumulate, a few more messages will be sent out each second. The data sets are defined beside the lines, so the single50.dat line is based on fifty open ports. The graph also shows that the maximum number of messages that can be sent inside an individual Hummer seems to be between fifty and one hundred. With fifty internal messages per second, all fifty of the messages seem to be sent each second. However, with one

hundred messages per second, only seventy or eighty messages are sent each second.

Issues considered & encountered:

During the testing process, there were certain variables that we needed to take into account. There are also testing results not represented in the output data.

- DMs are coded to sleep for intervals of 100ms, thereby limiting how often the DMs can get and report messages.
- The default heartbeat sent to a DM is once every 250ms.
- The DMS wrapper also has sleep times that primarily affect how long it should wait before checking for data to send and receive.
- The load average on the internal test was more of an issue than on the external test. On the internal test, the load average went above 1 for many of the larger test sizes.
- Queues for the Post office seem to allow it to accept messages for later processing.
- Normally, the message will not go through the post office to send a message between components within a single server in a Hummer. In this test, those messages went through the post office. The internal messaging capacity is most likely higher than seen in the graph because it does not have to start as many threads to send messages.

Fault Testing Hummer Communication

Loss of Hummer Servers in a Hierarchy:

Define loss?

If a Hummer server is lost before attacks and reported messages are made, there will not be any issues with bringing that server back into the hierarchy. It will remain fully functional.

However, once reports have been made and a Hummer server is lost, communication through the hierarchy cannot be re-instated without restarting all of the agents and DMs in the lost Hummer server and the Hummer server one link

down in the hierarchy. The latter machine (the subordinate) will report errors of broken sockets on the connections made with the lost server.

The results of restarting all of the agents and DMs on the previously lost machine may also result in response issues.

Results of Hummer Loss without reinstantiation:

The loss of a Hummer in the hierarchy breaks the link between the Hummers above and below the lost Hummer. None of the attacks on every Hummer server below the lost server in the hierarchy will be detected, and responses cannot be made by the high-level servers.

Hummer Functionality Test

This test is aimed to verify the proper functioning of all communication within Hummer. A Hummer message consists of envelopes containing a line of data that represents the message. These envelopes are processed by the Post office before they are sent to their respective destinations.

Given below are the communication lines to each module in the Hummer system. One such configuration can be seen in the following figure.

Agent – Receive lines of data from executed shell commands, external programs, searching the network, or by directly reading log files.

Tools – Receive lines of data directly from files, or commands such as Syslog, and Snort.

Belief – Receives input from Agents, the post office, the SNMP server, Response, and the adDM.

Decision – Receives lines of data sent from any Agents, Tools, dms, and subscription lists that they are configured to listen to, or that report to them.

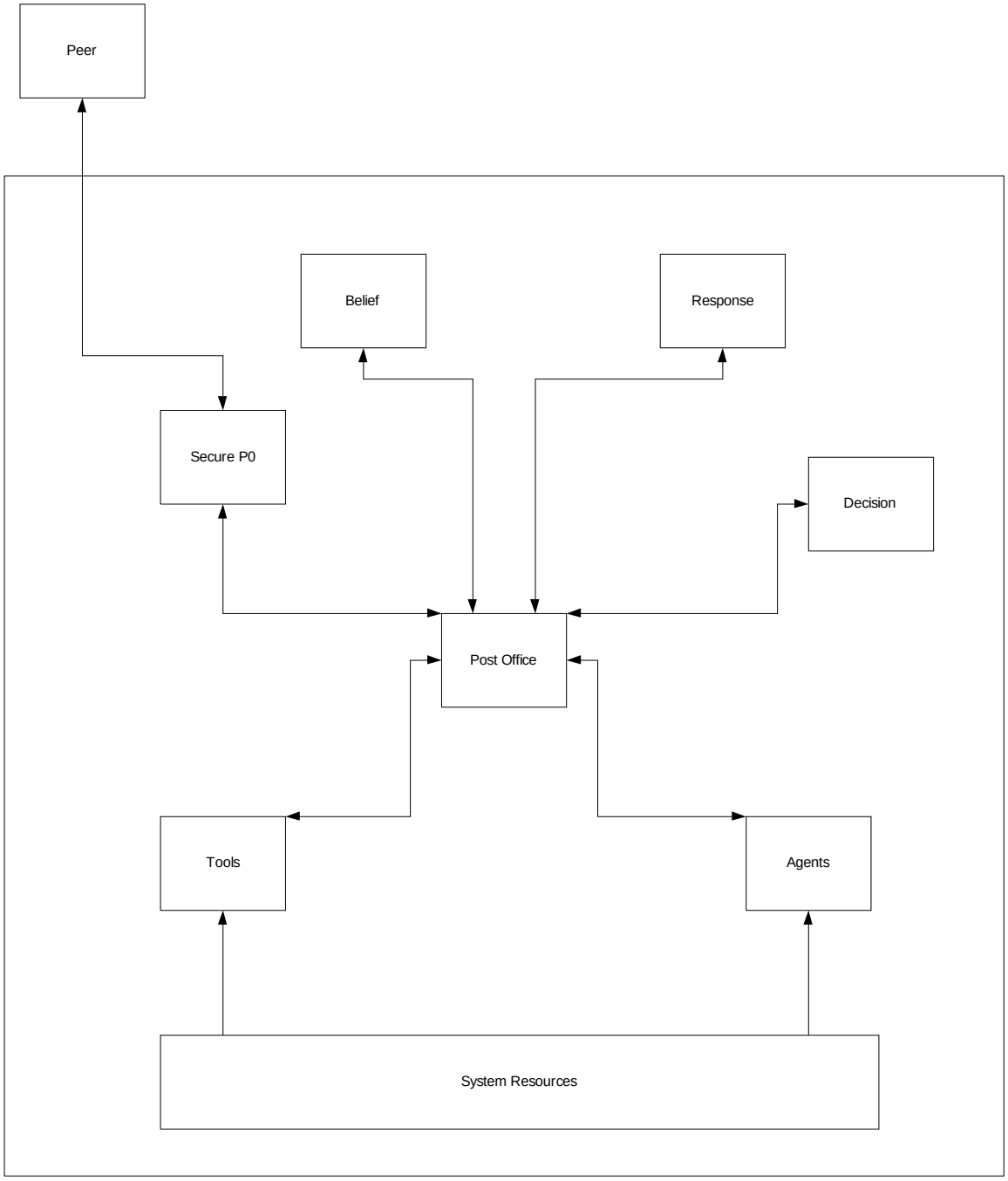


Figure 16: Hummer communication model

Response – Receives lines of data sent from Decision Modules and Belief, and sends responses to Agents, tools, VRRP, Belief, and IPTables.

Secure PO – Receives data from any of the above components that is for components within a different Hummer. It also receives any information that is sent from other Hummers to this machine and its components.

Post Office – Receives all messages from components and directs them to the destination component.

Test Results:

An attack suite is tested against the Hummer system on the CSDS Test suite system as discussed in the previous section. A brief overview of the test is presented below.

Communication within Hummer:

Use a file that gets a dump of the number of messages being sent in Hummer by specific servers.

Plan: count the number of messages that pass through the Post Office for each case.

Super Manager: runs adDM, Response, and Belief

Manager: runs DMs.

Subordinate: runs the needed Tools and Agents for the DMs running on the manager. The logical target for attacks against the system.

Normal data flow– (dependent on configuration)

- No attacks, just Hummer running on each of the systems in the hierarchy.

Threshold/Boundary data flow-

Super Manager - Supply enough attack messages so that Hummer stops responding.

Manager – Attack the manager until it stops responding.

Attack subordinates until manager stops responding.

Subordinate – Attack until the subordinate stops responding.

Attempt to determine the number of messages across the secure PO that will cause Hummer to DOS itself.

Communication Between Hummers:

Communication Techniques:

- Envelopes directed at components of other Hummer servers.
  - Envelopes containing messages
  - Envelopes containing agents
  - Envelopes containing envelopes
- Subscribing to lists on other Hummer servers.

Bibliography

[DMDesign] : DM Design Guide, Author: Evans, Darin.

Task Agents for Forensic and Attest Analysis

Jeffrey L. Harkins, PhD, CPA
Hugh Pforsich, PhD

University of Idaho

Research Assistants
Sean Diehl
Shawna Emery
Eric Lauer
Elise Shurtliff
Maria Stanfill

This research was funded by a grant from the Defense Advanced Research Projects Agency (DARPA) through the Center for Secure and Dependable Systems (CSDS) (an affiliated unit of the Microelectronic Research and Communications Institute (MRCI) at the University of Idaho.

Recent terrorist events coupled with a plethora of business and economic failures (Enron and Global Crossing) highlight the need for enhanced and cost-effective technological tools to assist with predicting, monitoring, assessing and analyzing the many threats and risks facing government and economic entity information systems. Intelligent task agents provide a vehicle for delivering and executing monitoring, assessment and analytical tasks. The purpose of this research proposal is to develop specific tasks for intelligent agents in forensic analysis, attest analysis and risk assessment. These tasks would map the strategies and behaviors of intelligent agents for gathering information, investigating events and activities and assessing data for compliance with known standards and controls.

Platform for initial development and testing of the tasked intelligent agents would be the DARPA-sponsored Hummer prototype. Long-term development would extend the application of the intelligent task agent across other platforms. Commercial application of this product is promising for law enforcement, defense applications, investigative activities and attestation (audit).

The purpose of the research was to develop an analytic model for forensic analysis and relate that model to a general predictive model for risk assessment and risk management.

We addressed two specific objectives:

1. Gather information, in real time, about a risk event for the purpose of improving our defense for a related risk event.

2. Gather information, in real time, that would lead to successful prosecution of perpetrator of the risk event.

A forensic examination is intended to:

1. Use specialized investigative skills in carrying out an inquiry conducted in such a manner that the outcome will have application to a court of law.
2. Examine evidence regarding an assertion to determine its correspondence to established criteria carried out in a manner suitable to a judicial process.

Our forensic model, based upon our generalized model for risk prediction, provides us with a prediction of "likely" risky events. This prediction provides a map of data points and other signals about that event, permitting the accumulation and storage of pertinent information that would be available for forensic and investigative analysis.

To test the parameters of the model, we selected a test bed built around an information management entity (school district or educational facility). We assumed that the test bed has implemented an intrusion detection system (IDS) that is intended to protect against two types of attacks:

- An attack on a master file of student grades
- A denial of service (DOS) attack that would undermine the integrity of an "outreach or distance-learning" program.

We constructed an intelligent "forensic agent" module which is alerted when one of the two risk events occurs. Because the forensic agent has a strategy for each "expected" type of attack, it has a map of tasks to be performed once an event is initiated. The tasks are twofold:

- Gather information, ***in real time***, about the event to confirm and improve our detection process
- Gather information, ***in real time***, that would be useful in identifying and prosecuting the attacker.

The actual operation of the "forensic agent" is to acknowledge the attack, obtain the code needed to go to work and begin to gather information according to the protocols established for the type of attack. Information obtained is stored at a secured system, encrypted and in conformity with Patriot Act standards and other applicable legal standards.

This project was conducted in the following steps:

- *Formulation of the risk assessment model*
- *Formulation of the forensic model*
- *Formulation of a forensic task agent*
- *Application of the forensic task agent model*

Formulation of the risk assessment model

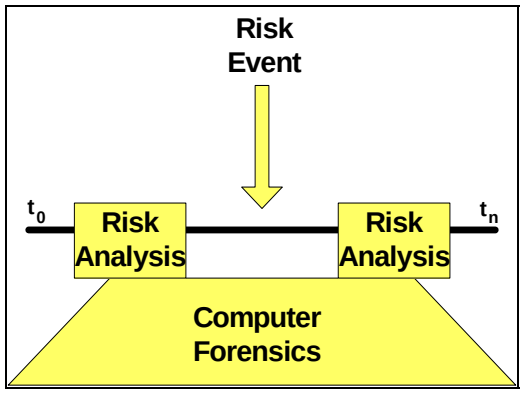
Since the tragedy of September 11, one of the largest complaints launched against the U.S. Government has been for its inability to predict the risk events with the information it had available. One reason for this circumstance lies in the traditional concept of risk analysis – the primary way to prevent a risk event is through an analysis of past risk events. Many individuals and organizations would prefer not to experience a risk event before being able to avert it. In order to address these concerns is to develop a risk analysis model that predicts risk events.

Definitions

Risk analysis is the ability to assess the likelihood of future attacks and future crimes. It is a preventative method used to predict who might commit a crime and what crime might be committed. ***Computer forensics analysis*** forms the converse of risk analysis as the collection of data from computers or computer systems to establish a link between a crime and its victim or a crime and its perpetrator. Computer forensics links the risk event with the perpetrator, the method and the motive for a risk event.

Typically, risk and forensic analysis are viewed as separate and distinct procedures. One occurs before a risk event, and the other takes place after. However, this research project views these two processes as inextricably linked. The relationship between risk and forensic analysis is illustrated in *Figure 1*. Along a timeline, risk analysis occurs both before and after a risk event occurs. The whole process, though, is supported by forensics. One objective of this project is to make this symbiotic relationship clear.

Figure 17: Timeline of Risk Analysis with Support from Computer Forensics



Risk Analysis

Current Practices

Risk analysis has typically been expressed in the form of a checklist or threat tree. Known risks are listed, safeguards for these risks are created, and the checklist is used to implement the safeguards. However, use of checklists and threat trees creates two significant problems: only known risks are accounted for and protected against, and the checklist or threat tree becomes a comprehensive map of system security weaknesses when in the possession of a perpetrator.

General Systems Theory

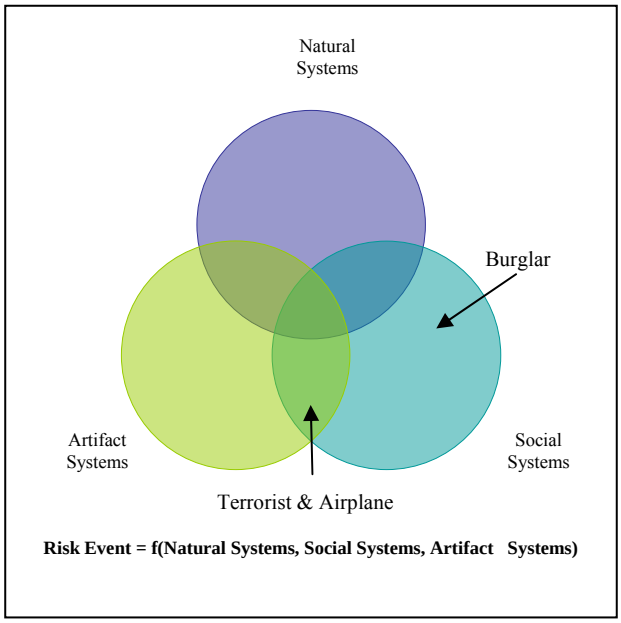
The solution to this use of checklists lies not in examining individual risk events, but in examining the source of risk. The method for accomplishing such an undertaking comes from an analysis and identification of systems. In the 1950's, ***Managerial Science*** published an article entitled "General Systems Theory: The Skeleton of Science." The article states all phenomena are composed of systems, and it describes a taxonomy of systems with nine hierarchical levels. If everything natural and artificial is a system, an analysis of systems becomes the key step to analyzing risk, as all sources of risk are systems as well.

Unfortunately, the taxonomy of General Systems Theory does not meet the needs of risk analysis. Therefore, Harkins and Pforsich created a new taxonomy of systems by classifying systems into one of three classes: ***natural systems***, ***artifact (artificial) systems***, and ***social systems***. The three system classes are completely exhaustive and mutually exclusive. Any identified system can be placed into one of these three classifications.

A New Perspective

A risk event, then, is a function of systems, or more specifically, a function of Natural, Artifact, and Social Systems (See *Figure 2*).

Figure 18: New Systems Taxonomy



The source of risk can either develop from a single system, such as a burglar, or from the combination of two or more systems, such as a terrorist and an airplane. This approach to risk analysis does raise the sizeable concern of infinite possible sources of risk and subsequent risk events. The model will address this concern. Filtering risk sources through the model would identify the most probable risk sources and the likely risk events generated by those sources.

The Basic Risk Analysis Model

The first step in understanding the relationship between risk events and risk analysis is to prepare a descriptive model of the occurrence of a risk event. There were two reasons for this: to aid in developing a base understanding of the relationship between risk events and the task of computer forensics, and to begin the research with a simple framework. The descriptive model forms a launching point to develop a risk event predictive model and a task agent for forensic analysis.

The Descriptive Model

The descriptive model developed outlines the occurrence of a risk from the inception of intent in the perpetrator's mind to the implementation of preventative measures once the investigation has concluded. It was developed using the REAL (**r**esources, **e**vents, **a**gents, **l**ocations) template described in ***Accounting, Information Technology, and Business Solutions*** by Hollander, et al. The REAL template organizes processes into events, and then identifies corresponding resources, agents, and locations. *Figure 3* illustrates the descriptive model for a risk event.

The REAL model portrays, descriptively, the strategically significant activities and essential characteristics associated with a risk event as well as the corresponding activities and characteristics associated with the forensic examination of the event. The descriptive model maps the underlying elements for the development of a predictive model for risk analysis and forensic activities. These elements are probably familiar to most mystery story fans: *who* was involved, *what* happened, *when* did it occur, *where* did the event occur, *how* did it happen and *why* was the event provoked.

The Predictive Model

The building blocks for the predictive model were derived from decision theory. Decision theory constitutes a method to model the behavior of a rational decision maker. While the situation may not appear so on the surface, it is assumed that perpetrators make rational decisions. Those intent on "breaking rules" are motivated toward some goal (or end payoff), e.g., monetary gain, destruction of or damage to an enemy. Consequently, their intent or actions (and their decisions) are rational. If this is not the case, the task of predicting criminal behavior becomes virtually impossible.

Decision theory maps a decision outcome over a decision maker's perception of possible states of the world, actions, perceived probabilities of those states occurring, and a belief system giving utility values to the decision outcomes. The base equation used in the predictive model is shown in *Figure 4*.

Figure 19: Descriptive Models of Risk Assessment and Computer Forensic Analysis

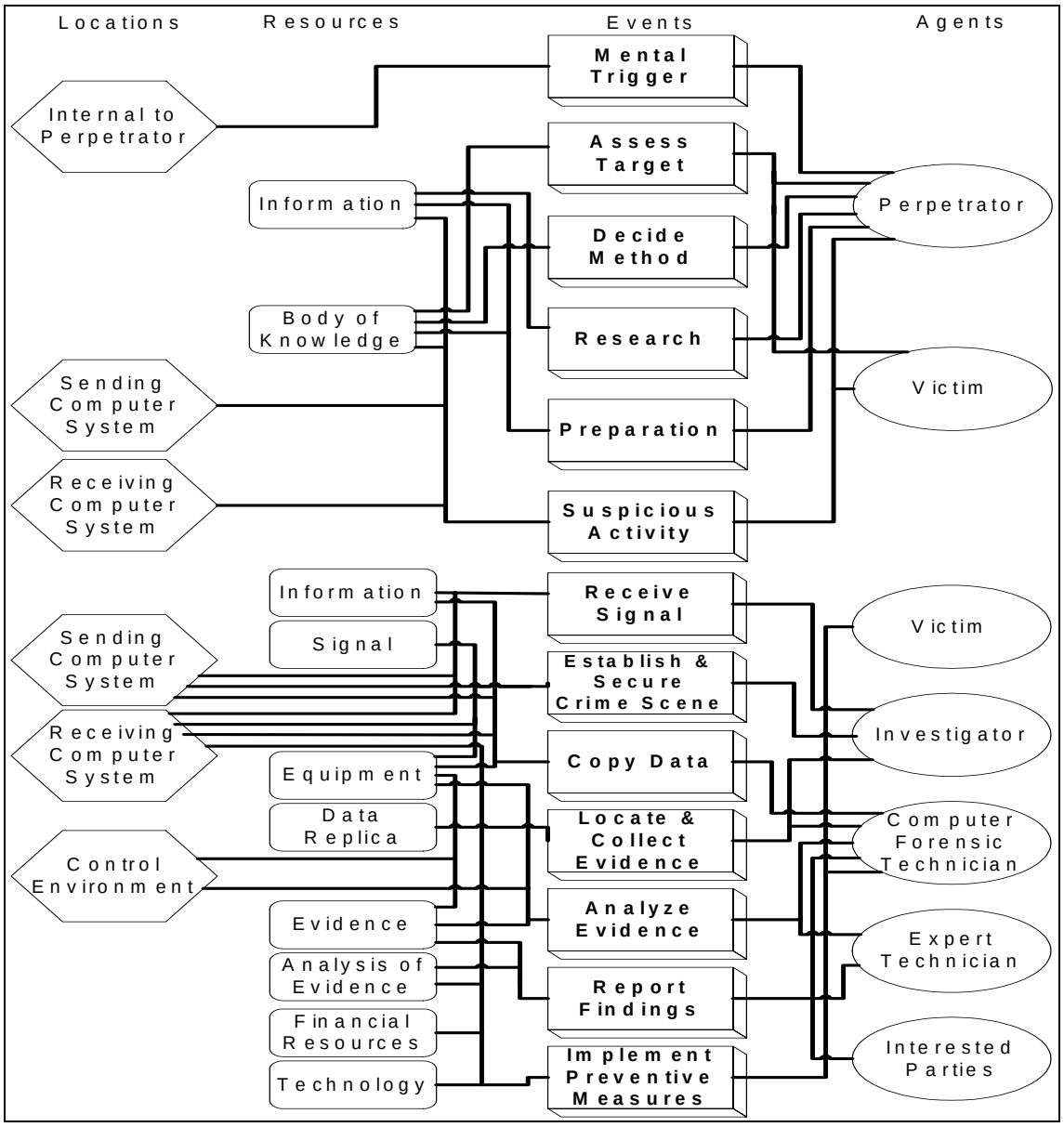


Figure 20: Decision Theory Equation and Terms

$$a^* = \max \left[\sum_{\substack{s \in S \\ a \in A \\ n \in N}} P(s \mid (B, n))\, U(B \mid (s, a)) \right]$$

Terms

- a^* = optimal action
- A = set of all perceived actions
- S = set of all perceived states
- P(s) = perceived probability of states occurring
- O = set of all possible outcomes (combinations of actions and states)
- U(o) = utility function, reward or disreward of outcome for decision maker
- B = knowledge, experience, & beliefs of decision maker
- N = set of all information systems available to decision maker

Utility

The concept of utility is critical to the implementation of the predictive model. Utility results from increases (or decreases) in well-being resulting from the outcome of a decision. For example, it could be the sense of well-being from owning another pair of shoes, the reward of a successful investment or the damage resulting from a terrorist attack. Utility is impacted by many different components including an individual's knowledge, beliefs, experiences, and desires.

The proper development of a perpetrator's utility function is critical to the predictive model's success. However, this piece of the model will also be one of the most difficult to create. Each decision maker has a utility function that maps a utility value to all outcomes of a decision. The difficulty lies in generating a utility function for an individual operating covertly, i.e. the perpetrator. The perpetrator's utility function will most likely be the most difficulty and the most powerful component of the predictive model. The more likely influences on behavior and knowledge will drive the decision maker's choices and include elements of culture, place of origin, knowledge, motivation, grievances and actions. These elements are driven by factors such as race, beliefs, religion, family, education, etc. The interlocking nature of these elements and the likely degree of control over those elements by an individual decision maker are presented in *Figure 5*.

Figure 21: Components of Utility

Culture	**Place of Origin**	**Knowledge**	**Motivation**	**<Grievances>**	**<Actions>**
Race Parents Religion Ethics Moralism	Race Parents Culture	Beliefs Experience Traditions	Education Religion Ethics Moralism	Desires	Tradition
		Socio-economic Status Social Framework Perceptions			
Less Control ----------Ultimate Control					

Information Systems

Information systems are tools employed by decision makers to improve their prediction of the likelihood of a state occurring. A classic example is the weather forecast. If one were to look out the window in the morning and see clear skies, the decision of bringing a raincoat to work would most likely not appear to be the optimal one. Yet, if the morning news reported a 90% chance of rain in the afternoon, the original perception of a clear day would be modified and the decision to bring a raincoat to work would now seem more promising.

In the predictive model, information systems play a similar role. These systems modify the perpetrator's perceptions of the likelihood of states occurring, such as the chances of success or being caught. When trying to prevent a risk event from occurring, information systems can serve a key function. The predictive model identifies which information systems provide the most benefit to the perpetrator with the least cost. If the information systems which the perpetrator uses are known, then perhaps they can be manipulated by those attempting to prevent the risk event.

The final element relating to information systems is ***information economics***. Acquiring information in any form has a cost. The cost can come in many different forms: time, money, energy, increased risk, etc. This cost must be incorporated into the predictive model in order to accurately portray the decision maker. Even if the information system provides highly reliable and relevant information, the cost of the information will help determine whether the benefits provided will lead the perpetrator to exploit the given system, use another system, or make a decision without additional information.

Applications of Risk Prediction Model

To demonstrate additional applications of the risk prediction model, student researchers were tasked to create additional applications. Four applications were developed and are included in the appendices. The four applications are:

Appendix 1: Employee sabotage in a software development company
Appendix 2: Poisoning soft drinks during the manufacturing process
Appendix 3: Fraudulent behavior by financial analysts
Appendix 4: Misstatement of financial information in publicly traded companies

The results of the development and application of a risk prediction model are promising. For the utilities, probabilities, modified probabilities, and costs of the information systems, the model produced an optimal decision, which may reflect the course of action an actual perpetrator would pursue. Thus, law enforcement personnel could place themselves "one step ahead" of the perpetrator instead of the classic "one step behind."

However, the model, both in its general form and in its specific applications, is not yet ready for practical use. As stated above, a standard method for defining a perpetrator's utility function and prior probabilities needs to be defined. Once these methods have been developed, the model should be tested against human decision makers to determine how accurately it mirrors a potential perpetrator's decision patterns.

Admittedly, these goals are easier stated than performed. The task of defining a means to determine a perpetrator's utility function faces many hurdles, given the difficulties in creating a comprehensive list of utility components (refer back to *Figure 5*). Difficulty also arises in different utility components being given different weights by each perpetrator. For example, religion may be weighted more heavily by a terrorist than by a financial analyst committing fraud. A similar hurdle arises in mapping a perpetrator's prior probabilities. If these methods can be defined, however, the usefulness and power of the model will be self-evident.

Formulation of the computer-based forensic model

Forensic science is the application of science to an investigative process. Most often, forensic techniques are employed to collect, analyze, and interpret evidence associated with a crime scene. If the evidence is to be used to prosecute a suspect, the forensic examination must meet the legal standards applicable for the type of evidence sought. General standards about the collection and custody of evidence are well-established in

Figure 22: Real Model Matrix

Event	Objective	Event Trigger	Risk	Notes
Receive signal	Inform of potential criminal activity	Improper activity in system	Receive false signal, failure to receive signal	Either from system administrator after crime occurs, internal sensors as crime is taking place or law enforcement personnel Multiple signals can be generated by the same system
Receive permission to take action	Ensure any evidence uncovered can be used in legal proceedings (probable cause not met	Receipt of signal	Receive permission from unauthorized agents, receive permission based on faulty information	Permission need only be obtained from one source
Isolate system from external environment	Protect integrity of evidence, prevent further damage to system	Receipt of permission	Damage of information, further erasure of footprints	Can occur during, or after, and attack on a system
Locate and extract pertinent evidence in system	Discovery of evidence to be used in legal proceedings and/or prevention of further system break ins	Isolation of system from external environment	Ignoring evidence, damaging original evidence as it is sought, damaging system, violating legal rules for seizure of evidence	Multiple traces of evidence can be found on each system Should be conducted by personnel familiar with forensic data extraction, not the "local computer expert"
Analyze data	Determine nature of attack, purpose of attack, identity of perpetrator, etc, for use in legal proceedings and/or prevent future attacks	Locating and extracting relevant data	Misinterpreting data, damaging original data during analysis	Should be conducted by personnel familiar with forensic data extraction
Report findings to proper parties	Deliver evidence to apprehend and/or prosecute the perpetrator or the systems administrator to prevent future attacks	Analysis of data	Communicating inaccurate data and/or recipient misunderstanding data	Reports can be sent to multiple parties

law as well as vetted in the courts. ***Technology, or computer forensics*** is a specialized area of forensic science. Computer forensics involves the search for and discovery of evidence stored in computer systems. Unique features of technology forensics include the use of forensic techniques in preventing criminal activity, the dual nature of computers in their role in facilitating a crime as well as providing documentation of the criminal activity, and the ability of the computer to maintain exact records of the digital processing events.

The objectives of the computer forensics process are to gather, ***in real time***, information (evidence) about a risk event that would be useful in:

1. improving defenses for a subsequent attack (a traditional application).
2. identifying and successfully prosecuting the risk event perpetrator(s).

A goal of this research module was to establish the feasibility of constructing a computer forensic task agent, constructed from the probabilistic predictions of a risk assessment model, that would gather information pertinent to the nature of (or type of) *predicted or expected attack*. In a computer-based risk event, if a forensic task agent is properly designed, it can gather critical information about the event in real time. In many computer-based risk events, critical elements of the event are only available while the attack is in process. If a computer forensic task agent is properly tasked, this critical information can be obtained, analyzed, and subsequently utilized to achieve the objectives of improving defenses for similar attacks as well as providing evidence that would lead to apprehension, prosecution and conviction of perpetrators.

A REAL model was constructed to facilitate an understanding of the computer forensic process and to assist in linking the forensic process to the risk prediction model. The REAL model matrix is presented in *Figure 6*. The REAL forensic model was presented earlier in *Figure 3*. Please refer back to *Figure 3*.

From either *Figure 6* or *Figure 3*, one can deduce that the forensic model is framed around an event caused by a perpetrator or perpetrators. Further, one can observe the "forensic process" (herein portrayed as a series of steps beginning with the securing of the crime scene and ending with report findings and implementing preventive measures) as systematically discovering the various systems (social, artifact and natural) that were a part of the risk event. The link to the risk assessment model is straightforward – with each involved system specifically and potentially identifiable through the forensic process while the involved systems were probabilistically determined in the risk assessment step.

Thus the construction and viability of the computer forensic task agent, if built from the results of a risk prediction model, are explicitly tied to the ability to predict which systems would be involved in risk schemes. The conclusion seems apparent – the more robust our risk prediction model, the more likely it is that a relevant forensic task agent can effectively unravel and document the mysteries of actual risk events.

Before delving into the construction of the forensic task agent, it is important to acknowledge the legal standards imposed on the gathering and storage of real-time electronic surveillance.

Legal Issues Governing Real-Time Electronic Surveillance and Data Collection

The development of a model for forensic analysis has two specific objectives: gathering information, in real time, about the event in order to improve defenses and collecting information, in real time, that would lead to successful prosecution of perpetrators. In accomplishing these objectives, the forensic model must stay within legal boundaries. The following is an examination of the legal guidelines and statutes governing electronic evidence and surveillance and there applicability to the forensic analysis model.

Real time electronic surveillance is governed by two federal statutes; the wiretap statute, Title III of the 1968 Omnibus Crime Control and Safe Streets Act, and the Pen/Trap statute, 18 U.S.C. §§ 3121-3127.

The wiretap statute prohibits the "interception" of "oral communications", "wire communications", and "electronic communications". It is important to understand the statutory definitions of these terms. Communications are considered "intercepted" if they are "acquired contemporaneously with transmission". In other words, the interception must happen at the same moment that transmission is occurring. This is almost impossible to do with electronic communications. In United States v. Steiger, 318 F.3d 1039, the court found "There is only a narrow window during which an E-mail interception may occur—the seconds or mili-seconds before which a newly composed message is saved to any temporary location following a send command."

Oral and wire communications are defined as communications that contain a human voice. "If a communication does not contain a genuine human voice, either alone or in a group conversation, then it cannot be a wire communication." See S. Rep. No. 99-541, at 12 (1986)

Electronic communications are defined as the "transfer of signs, signals, writing, images, sound, data, or intelligence of any nature….but does not include wire or oral transfer, tone-only paging, communication from a tracking device, or electronic funds transfer information stored by banks". In relation to the forensic analysis model, electronic communications will be key.

Title III, the wiretap statute, provides the statutory framework that governs real time electronic surveillance such as "keystroking" a hacker breaking into a computer system. It prohibits a non-participatory third party from intercepting private communications. However, Title III also provides seven statutory exceptions to this prohibition. It is not necessary to examine all seven of these exceptions, as only one of them is relevant to the forensic analysis model; the provider exception.

The provider exception of Title III reads: "Employees or agents of communications service providers may intercept and disclose communications to protect the providers' rights or property. For example, system administrators of computer networks generally may monitor hackers intruding into their networks and then disclose the fruits of monitoring to law enforcement without violating Title III." The key to this exception is understanding that the monitoring should be reasonable and should minimize the interception and disclosure of communications unrelated to the investigation. See United States v. Mullins, 992 F.2d 1478 (The need to monitor the misuse of a computer system justifies intercepting electronic communications and falls within the provider exception of Title III.) and United States v. Villanueva, 32 F. Supp. 2d, 639 (it shall not be unlawful…for an …employee or agent of a provider of wire or electronic communications, in the normal course of his duties….to intercept, disclose, or use that communication for the protection of the rights or property of the provider…)

The forensic agent module embedded within the intrusion detection system is a "software routine" which is activated when the event of interest occurs. Upon activation, the agent begins to gather information based upon the event protocols established for the type of attack. The agent's purpose in collecting relevant information and evidence is to better protect the system in the future and provide data that can be used as proof for the successful prosecution of any perpetrators. In performing its function, the forensic agent may trace connections and monitor system logs and running processes. In doing so, it falls within the parameters of a trap and trace device.

As mentioned previously, the Pen/Trap statute also governs electronic surveillance. This statute gives providers of wire or electronic communication service broad authority to use pen/trap devices on their own networks without a court order:

(1) Relating to the operation, maintenance, and testing of the service or to protect the rights or property of the providers or the users of their service.
(2) To record the fact that a wire or electronic communication was initiated or completed in order to protect the provider, any provider furnishing service toward completion of transmission, or users of the service, from fraudulent, unlawful, or abusive use of the service.
(3) Where the consent of the service user has been granted.

Pen registers and trap and trace devices are defined very broadly. A pen register is defined as "a device or process which records or decodes dialing, routing, addressing, or signaling information transmitted by an instrument or facility from which a wire or electronic communication is transmitted….". A trap and trace device is defined as "a device which captures the incoming electronic or other impulses which identify the source of a wire or electronic communication…". It is important to note that this statute also covers "software routines" as well as physical devices.

Title III and the pen/trap statute do not provide statutory suppression remedies for wrongfully intercepted electronic communications and evidence obtained in this manner

may result in suppression if guidelines have not been followed. However, the instances of electronic evidence being suppressed are few. The Sixth Circuit has fashioned a "clean hands" exception that permits the government to use any illegally intercepted communication so long as the government "played no part in the unlawful interception". A perfect example of this is United States v. Steiger, 318 F.3d. In this particular case, the Montgomery, Alabama Police Department (MPD) received an email from an "anonymous" source that identified a child molester. This source provided the defendant's IP address, pornographic pictures, checking account information, home address, and telephone information. The case was then turned over to the FBI who followed up and obtained a search warrant to gather further evidence. The "anonymous" source was from Turkey and had obtained the information through a "trap" and by illegally hacking into Steiger's computer. Steiger filed several motions to suppress, which were all denied. The courts found no reason to suppress because the "anonymous" source was not acting as an agent of the government nor did the source intercept communications in violation of the Wiretap Act.

By ensuring that the program parameters are within statutory requirements, evidence gathered by the forensic agent via electronic surveillance will not result in suppression.

To gather evidence that can be used in a court of law for successful prosecution, the forensic agent must follow federal guidelines. Understanding the definition of "relevant evidence" is the first step. Federal Rules of Evidence define it as "evidence having any tendency to make the existence of any fact that is of consequence to the determination of the action, more or less probable than it would be without the evidence."
In understandable terms, this means any evidence that makes it more or less likely an action took place and is attributable to a specific individual is "relevant evidence."

Because the evidence gathered by the forensic agent is electronic in form, the end report generated by the agent will be classified as computer records. Computer records fall into three distinct categories: computer-generated, computer-stored, or records that are a combination of the two. In defining them, the question to ask is whether a person or a machine created the records' content.

Computer-stored records are those that contain the input of human entry, such as a Word document. These records are admissible as evidence under the business records exception of Federal Rules of Evidence 803(6). This exception allows records made "in the course of regularly conducted activity" to be admissible if they are kept as a regular practice of the organization.

Computer- generated records are those that contain the output of a computer program, such as the log-in records from an Internet Service Provider. These records do not contain human "statements" or input; rather they contain the output of a program designed to process input following a defined algorithm.

The third type of computer records are those that are both computer-stored and computer-generated. An example is the results of a spreadsheet program that generates an output derived from data that is input by a person and calculated by the computer program. The forensic agent embedded within the intrusion detection system is activated when an attack occurs and runs according to defined algorithms. As a result, the program could be considered a part of the regularly conducted security activities of the system.

Once the forensic agent has gathered the relevant information, it will encrypt and store the information. The information can then be retrieved from the system to aid in prosecuting the offender. Because the information is the output of a program, it could be classified as computer-generated evidence. Either way, the information gathered by the forensic agent will be admissible if appropriate guidelines have been followed.

Some of the more important issues that can arise when seeking admission of computer records as evidence in a court of law include proving authenticity, and if the records contain any human statements, whether or not they are inadmissible as hearsay.

Computer-generated records generally result in a question of authenticity and computer-stored records must comply with the hearsay rule. For records that are both computer-stored and computer-generated, the hearsay issue and the question of authenticity need to be examined.

The courts generally allow computer-stored records to be admitted as business records under the business record exception of the Federal Rules of Evidence 803(6). If the record contain human statements, the court must establish that the statements were made in circumstances that tend to ensure their trustworthiness. See United States v. Cestnik, 36 F.3d 904, 909-10 (10th Cir. 1988). ("Computer records are admissible if (1) they are kept pursuant to a routine procedure designed to ensure their accuracy, (2) they are created for motives that tend to assure accuracy (e.g., not including those prepared for litigation), and (3) they are not themselves mere accumulations of hearsay") and United States v. Briscoe, 896 F.2d 1476, 1494 (7th Cir. 1990) (computer-stored records are admissible business records if they "are kept in the course of regularly conducted business activity, and [that it] was the regular practice of that business activity to make records, as shown by the testimony of the custodian or other qualified witness") (quoting United States v. Chappell, 698 F.2d 308, 311 (7th Cir. 1983)).

The key is establishing that the computer system from which the record came is maintained in the ordinary course of business, and that it is a regular practice of the business to rely upon the records for their accuracy.

The output that results from a computer's internal operations or from a specific program would not be classified as hearsay evidence. With a machine there is no possibility of a conscious misrepresentation. The possibility of false or inaccurate data only emerges if the machine is malfunctioning.

The question then becomes one of authenticity; was the computer program that generated the information (or in the case of the forensic agent module, *gathered* the information) functioning properly? Before computer records can be admitted, they must be shown to be authentic.

According to Federal Rule of Evidence 901(a) "The requirement of authentication or identification as a condition precedent to admissibility is satisfied by evidence sufficient to support a finding that the matter in question is what its proponent claims."

In most cases, the authenticity (reliability) of a computer program can established by showing that the users of the program regularly rely upon the results in the ordinary course of business. See United States v. Vela, 673 F.2d 90 (computer data compilations…should be treated as any other record of regularly conducted activity) and United States v. De Georgia, 420 F.2d 895 (…the circumstances of the making of the record are such that accuracy is not merely probable, but *highly* probable)

Authenticity can also be established through testimony that discloses what the program is designed to do and the precise instructions it has been given. United States v. Russo, 480 F.2d, 1229-1244 (Assuming that properly functioning computer equipment is used, once the reliability and trustworthiness of the information put into the computer has been established, the computer printouts should be received as evidence of the transactions covered by the input.)

The authenticity of computer records can be challenged. This usually occurs in one of three ways; questioning whether the records were altered, manipulated, or damaged after they were created, by questioning the identity of the author, or by challenging the reliability of the program that generated the records.

Because computer records can easily be altered, opposing parties often allege that computer records lack authenticity because they have been tampered with. A great example of this is United States v. Whitaker, 127 F.3d 601. Data retrieved from a drug dealer's computer using Microsoft Money was admitted as evidence against him. The authenticity of the records was testified to by the FBI agent that retrieved the records but the defendant's attorney appealed this on the grounds that the records had been tampered with. The court found that "absent specific evidence that tampering occurred, the mere possibility of tampering does not affect the authenticity of a computer record."
In other words, the party opposing the authenticity of computer records on these grounds must show that the computer records in question have been tampered with.

As mentioned before, computer records are sometimes challenged by questioning the identity of the author. Internet communications are a particular problem in this area because of the anonymity they provide. Circumstantial evidence usually provides the key to combating this particular challenge to the authenticity of computer records. In United States v. Simpson, 152 F.3d 1241 (10th Cir. 1998), prosecutors sought to show that the defendant had conversed with an undercover FBI agent in a chat room devoted to child pornography. The government offered a printout of an Internet chat between the agent

and an individual identified as "Stavron" and sought to show "Stavron" was the defendant. The court admitted the printout into evidence. On appeal, Simpson argued that "because the government could not identify that the statements attributed to him were in his handwriting or his voice," the printout had not been authenticated and should have been excluded. The Tenth Circuit rejected this argument, noting that considerable circumstantial evidence identified him as "Stavron".

Challenges to the authenticity of computer-generated records based on the reliability of the computer program can be overcome by (indicating that matters created according to a process or system can be authenticated with "[e] vidence describing a process or system used…and showing that the process or system produces an accurate result") Federal Rules of Evidence 901(b)(9). In other words, showing that the program is reliable and relied upon by the organization using it establishes the reliability of the computer program.

After ensuring that the forensic agent module, in performing its function, stays within the legal guidelines for conducting electronic surveillance and collecting electronic evidence and information, the final step will be forwarding that information on to law enforcement. The form that data takes brings up the issues of original versus copy and computer printouts as "summaries."

The best evidence rule states that the 'original' writing, recording, or photograph is required. Federal Rules state that "data stored in a computer or similar device, any printout or other output readable by sight, shown to reflect the data accurately, is an 'original'". The final printed report that shows the results of the forensic agent program will be considered an 'original'.

Federal Rule of Evidence 1006 allows summaries of voluminous evidence in the form of "a chart, summary, or calculation" subject to certain restrictions. Typically, a computer printout is not considered a summary that must comply with Federal Rule of Evidence 1006.

The forensic agent module will gather information related to the event in the most efficient and effective manner while remaining in compliance with the applicable Rules of Evidence, ensuring that the evidence gathered can be used to effectively prosecute the offenders. At the very least, the forensic agent will have achieved its programmed objectives if the information gathered provides "probable cause", allowing law enforcement to gain evidence for conviction through other means.

Formulation of the Forensic Task Agent

The concept of an intelligent forensic task agent is based on the premise that at the time that an attack has been launched against a computer system is an appropriate time to begin gathering evidence about the risk event. Thus, notification of an attack triggers or initializes the task agent to begin gathering information related to the attack. In

formulating the event sequence, it is assumed that an entity's intrusion detection system, upon determining that an attack is in progress, wakes up the forensic agent and the forensic process is initiated.

Forensic Process

In theory, the forensic process maps precisely to the elements established in the risk assessment model, e.g., *who* is involved in the attack, *when* is the attack, *where* is the attack, *how* is the attack being perpetrated, *what* resources are involved in the attack and *why* is the attack being launched. Thus, conceptually, the forensic agent would be tasked to gather information in real-time across the entire spectrum of attack elements. In most circumstances, it would be logical to assume that the why or motivation for the attack is not going to be determined until an analyst has studied the data gathered from the other elements of the attack. It is also reasonable to assume that as forensic task agents mature, they would be able to assess likely motivation and assist the intrusion detection system in redirecting resources that might be involved in the ultimate motivation for the attack.

The general model and major requirements for the forensic task agent are:

1. Determine the link interface between the intrusion detection system and the forensic task agent.
2. Determine the specific data elements to be collected by the task agent, prioritized as may be necessary.
3. Encrypt collected information and send to a write-once data store.
4. Assure that the process is transparent to user (perpetrator)

The requirements breakdown for a general model are:

1. IDS interface
    a. Create policy that includes the forensic task agent
    b. Create wrapper that launches forensic task agent
2. Data collection
    a. Organize fields with XML tags
    b. <when>
        i. <localtime> Get local system time
        ii. <remote time> Get outside source time
    c. <what>
        i. <syslog> Monitor syslog (/var/log/messages)
        ii. <snort> Monitor snort (var/log/snort/alert)
        iii. <ps> Monitor running processes (ps)
    d. <where>
        i. <hash> Get current hashes for important files
        ii. <mac> Get MAC (modify, access, create) information for important files
        iii. <nmap> Check open ports (nmap)

iv. <netstat> Check open connections (netstat)

e. <who>

i. <who> Check who is logged in

ii. <lastlog> Check who has logged in

iii. <traceroute> Trace open connections

f. Write out fields to data storage

3. Encrypt collected information and send to write-once data store
    a. Hash information
    b. Encrypt hash and information
    c. Write encrypted hash and information to data store
4. Make process transparent to user.

The basic program flowchart for the forensic task agent tool is presented in Figure 7 below.

Realistically, there are numerous constraints that will arise during an attack and it is reasonable to expect that, for a given attack, a hierarchy or protocol would be established to dictate the order in which data is collected.

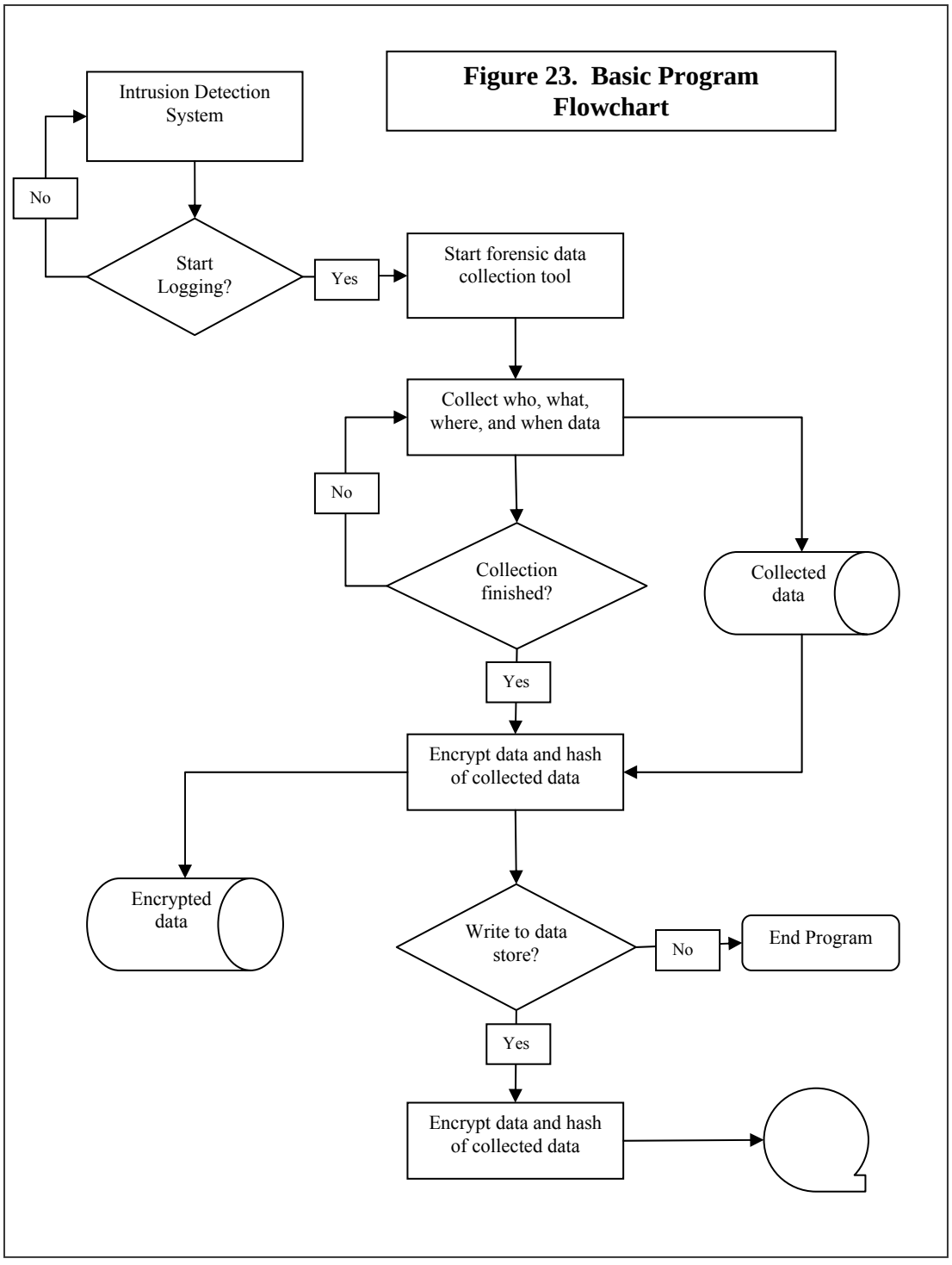


Figure 23. Basic Program Flowchart
Intrusion Detection System
No
Start Logging?
Yes
Start forensic data collection tool
Collect who, what, where, and when data
No
Collection finished?
Collected data
Yes
Encrypt data and hash of collected data
Encrypted data
Write to data store?
No
End Program
Yes
Encrypt data and hash of collected data

Application of the Forensic Task Agent Model

As proof of concept and to demonstrate the viability of the model, a simulation has been created for an information management organization (school system computer) which has predicted that they might be the target for two (2) types of computer systems attacks:

1. Attack on a master file of student grades
2. Denial of service attack that would undermine the integrity of the "outreach or distance learning" program.

The client (school system) has implemented an intrusion detection system (IDS) which is linked to a forensic task agent. The IDS alerts the task agent when one of the two predicted risk events occurs. In real time, the agent acknowledges the attack, initiates the code it needs to go to work and begins to gather information according to the protocols established for the type of attack. Information obtained is stored at a secure system, encrypted and in conformity with established legal standards of evidence. The attack scenarios are described below.

Attack Scenario 1 – Unauthorized Access and File Modification (0x333hate)

Target system:
Operating System: Redhat Linux 7.2 kernel version 2.4.7-10
Vulnerable Service(s): Samba 2.2

Explanation:

The Samba suite offers the ability for Linux to participate in Windows Style networking. The `nmbd` (name resolution and service announcement) program understands and can reply to NetBIOS over IP name service requests. This allows a Linux host to take part in the commonly used "Network Neighborhood" by responding to browsing requests and by allowing files sharing as if it were a Windows Client. The ability to operate as a WINS (Windows Internet Name Server) server is also built into the service, allowing Linux host to track other machines present in the workgroup and reply to queries submitted by those machines.

A buffer overflow exists in Samba (up to version 2.2.8 that can allow remote code to be executed. Since the Samba services are run as root, if an attacker can exploit the buffer overflow then they may successfully gain root access on the target system, as was documented in the initial Bugtraq report about this vulnerability on April 7, 2003. Subsequent to this warning, source code to exploit this service was developed and made available on the Packet Storm Security website. The 0x333hate program makes it incredibly easy to exploit this vulnerability.

Sample attack usage:

(1) Run nmap on target host (in this case a machine called *Bilbo*), looking for the netbios-ssn service:

```
# nmap Bilbo
        .
        .
        .
139/tcp      open       net bios-ssn
        .
        .
        .
```

(2) netbios-ssn service was found, so try to exploit it by running:

```
# ./0x333hate – t Bilbo
```

(3) Attacker now has root on *Bilbo* and creates an account with root privileges under the name *naughty*:

```
# echo 'naughty::0:0:/tmp:/bin/bash' >>
etc/passwd
# echo 'naughty::::::::' >> /etc/shadow
```

(4) The attacker logs out and re-logs in with the username *naughty* that has full access to *Bilbo*.

(5) Now, the attacker searches for and finds the grade file:

```
# find / –ipath '*grade*'
        .
        .
        .
/home/ugrads/erich/MasterGradeFile.txt
        .
        .
        .
```

(6) The attacker now changes the grade file and logs out.

References:

"CAN-2003-0201." Common Vulnerabilities and Exposures, 4 April 2003. http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0201 (25June 2003).

cOwboy. "0x333hate.c." Packet Storm Security. 29 April 2003. http://packetstormsecurity.nl/0304-exploits/0x333hate.c (1 July 2003).

Embrich, Mark. "0x333hate.c: Samba Remote Root Exploit."

http://www.giac.org/practical/GCIH/Mark_Embrich_GCIH.pdf (16 September 2003)

Attack Scenario 2 – Denial of Service (juno-z)

Target system:

Operating System: Redhat Linux 7.2 kernel version 2.4.7-10
Vulnerable Service(s): open TCP/IP services on local host

Explanation

To make a TCP connection with a system, a client must first issue a SYN request to the server it wishes to communicate with. In response to that SYN request, the server blindly sends back a SYN-ACK notice to the client, which tells the client it acknowledges its request for information. After the SYN-ACK is sent, the server will generally wait around 30-40 seconds for the client to reply with an ACK packet, which lets the server know that the client is now ready for transmission. After the ACK packet is received the connection is made and transmissions may now occur. One drawback exists with this protocol in that the server is left hanging for upwards of 40 seconds while waiting for the client to send its ACK packet. By crafting fake packets with random source IP information, a DoS attack may be launched against a server with open TCP ports.

If a packet with a spoofed source IP is sent to a server, then the server is going to return a SYN-ACK packet to that IP regardless of whether or not that IP actually exists. By flooding TCP/IP services with a multitude of spoofed packets, services may be denied to legitimate users because the resources necessary to respond to their requests may be used up waiting for ACK packets to return. If an extreme amount of packets are sent, the TCP/IP stack can potentially be overloaded and crashed. This is even more serious of a denial because it often requires a system reboot, thus interrupting more than just incoming TCP/IP services.

The juno-z program is one such SYN flooding utility that attempts to "nuke", or bring down services, or whole hosts by sending crafted SYN packets with arbitrary source IP addresses to targeted servers. Based on an earlier version, juno-z has the ability to send packets that appear to be from legitimate class A IP addresses that are running Windows systems, and can also create packets about four times as fast as the previous version.

Sample attack usage:

(1) Run nmap on targeted system (in this case bilbo) looking for open TCP ports:

```
# nmap bilbo
        .
        .
```

```
        .
21/tcp  open ftp
22/tcp  open ssh
```

(2) Open TCP ports exist, so launch attack (can target specific ports, or send to random ports. In this case we choose 21, with a delay of 0, and 16 threads of operation):

```
# ./juno-z bilbo 21 0 16
```

(3) It is likely that services are now unavailable, or possibly the whole system is dowm

References:

Feit, Sidnie, TCP/IP, McGraw-Hill, New York, 1997.

Sorceror. "juno-z.c." Packet Storm Security. 17 May 2001 http://packetstormsecurity.nl/DoS/juno-z.101f.c (16 September, 2003).

The demo versions for the two preceding attack scenarios are located in the attack directory on the accompanying CD.

***Instructions for working with the demo*:**

Welcome.

- **To collect data in an xml file, use the runcollection.sh tool.**
- **To visualize the data in a GUI, use the AuditVisualizer tool.**

- **Building instructions:**

  runcollection.sh -- none. Just a script to be run on a Unix machine.

  AuditVisualizer -- Unix:

  $ javac -classpath xerces.jar:. AuditVisualizer.java

  AuditVisualizer Windows:

  $ javac -classpath xerces.jar;. AuditVisualizer.java

  Notice the difference in using a ":" and a ";" above.

- **Running instructions:**

  runcollection.sh -- ./runcollection.sh -h gives instructions

  AuditVisualizer -- Unix:

  $ java -cp xerces.jar:. AuditVisualizer <log file>

  AuditVisualizer Windows:

  $ java -cp xerces.jar;. AuditVisualizer <log file>

  Again, notice the difference in using a ":" and a ";".

- **To decrypt the encrypted file use this command:**

  $ openssl enc -des -d -pass pass:password -in log.tar.crypt -out log.tar

- **To get a md5 checksum use this command:**

  $ openssl dgst <file you wish to get a checksum for>

- **To untar a file use this command:**

  $ tar xvf <file you wish to untar>