# A Survey of Intrusion Detection Systems

DOUGLAS J. BROWN, BILL SUCKOW, and TIANQIU WANG
*Department of Computer Science, University of California, San Diego*
*San Diego, CA 92093, USA*

## 1 Introduction

There should be no question that one of the most pervasive technology trends in modern computing is an increasing reliance on network connectivity and inter-host communication. Along with the tremendous opportunities for information and resource sharing that this entails comes a heightened need for information security, as computing resources are both more vulnerable and more heavily depended upon than before.

One subset of information security research that has been the subject of much attention in recent years is that of intrusion detection systems. The National Institute of Standards and Technology classifies intrusion detection as "the process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions, defined as attempts to compromise the confidentiality, integrity, availability, or to bypass the security mechanisms of a computer or network."[1] This definition captures the essence of intrusion detection but fails to address the methods by which Intrusion Detection Systems (IDS's) automate this process. The concepts of *false positive* and *false negative* are essential to this classification process. False positives are those sequences of innocuous events that an IDS erroneously classifies as intrusive, while false negatives refer to intrusion attempts that an IDS fails to report: the reduction of both false positives and false negatives is a critical objective in intrusion detection.

Modern IDS's are extremely diverse in the techniques they employ to gather and analyze data. Most rely, however, on a common architecture for their structure: a detection module gathers data that may contain evidence of intrusions, an analysis engine processes this data to identify intrusive activity, and a response component reports intrusions. As the response mechanisms tend to be dictated by site-specific policy rather than science, this paper will not discuss this feature of IDS's in much detail. Section 2 discusses the primary sources for intrusion detection while Section 3 examines current approaches to the problem of intrusion data analysis. Section 4 describes a number of issues that represent the current state of research in intrusion detection; Section 5 is devoted to addressing the state of the art in evaluating these systems. Opportunities for future research are presented in Section 6 and concluding remarks are located in Section 7.

## 2 Information Sources

Virtually all modern intrusion detection systems monitor either host computers or network links to capture intrusion-relevant data. Each of these data sources offers a unique set of opportunities as well as challenges for an IDS.

### 2.1 Host Intrusion Detection

Host intrusion detection refers to the class of intrusion detection systems that reside on and monitor an individual host machine. There are a number of system characteristics that a host intrusion detection system (HIDS) can make use of in collecting data including:

**File System** - changes to a host's file system can be indicative of the activities that are conducted on that host. In particular, changes to sensitive or seldom-modified portions of the file system and irregular patterns of file system access can provide clues in discovering attacks.

**Network Events** - an IDS can intercept all network communications after they have been processed by the network stack before they are passed on to user-level processes. This approach has the advantage of viewing the data exactly as it will be seen by the end process, but it is important to note that it will is useless in detecting attacks that are launched by a user with terminal access or attacks on the network stack itself.

**System Calls** - with some modification of the host's kernel, an IDS can be positioned in such a way as to observe all of the system calls that are made. This can provide the

IDS with very rich data indicating the behavior of a program.

A critical decision in any HIDS is therefore choosing the appropriate system characteristics to monitor. This decision involves a number of tradeoffs including the content of the data that is monitored, the volume of data that is captured, and the extent to which the IDS may modify the operating system of the host machine.

### 2.2 Network Intrusion Detection

A network intrusion detection system (NIDS) monitors the packets that traverse a given network link. Such a system operates by placing the network interface into promiscuous mode, affording it the advantage of being able to monitor an entire network while not divulging its existence to potential attackers. Because the packets that a NIDS is monitoring are not actually addressed to the host the NIDS resides on, the system is also impervious to an entire class of attacks such as the "ping-of-death" attack that can disable a host without ever triggering a HIDS. A NIDS is obviously of little value in detecting attacks that are launched on a host through an interface other than the network.

Network data has a variety of characteristics that are available for a NIDS to monitor: most operate by examining the IP and transport layer headers of individual packets, the content of these packets, or some combination thereof. Regardless of which characteristics a system chooses to monitor, however, the positioning of a NIDS fundamentally presents a number of challenges to its correct operation.

On a heterogeneous network, a NIDS generally does not possess intimate knowledge of all of the hosts on the network and is incapable of determining how a host may interpret packets with ambiguous characteristics. Without explicit knowledge of a host system's protocol implementation, a NIDS is impotent in determining how a sequence of packets will affect that host if different implementations interpret the same sequence of packets in different ways.[4]

A savvy attacker can exploit this property by sending packets that are designed to confuse a NIDS. Such attacks are referred to as insertion and evasion attacks based on whether they insert additional information into a packet stream that a NIDS will see and the target host will ignore or if they evade detection by forging data in such a way that a NIDS cannot completely analyze a packet stream.

Protocol ambiguities can also present a problem to a NIDS in the form of crud. Crud appears in a network stream from a variety of sources including erroneous network implementations, faulty network links, and network pathologies that have no connection to intrusion attempts.[10] If a NIDS performs insufficient analysis on a stream containing crud, it can generate false positives by incorrectly identifying this crud as being intrusive. While a NIDS therefore is in a very convenient position whereby it has complete access to all packets traversing a network link, its perspicacity is challenged due to ambiguities in network data and its limited perspective of host system implementations and network topology.

## 3 Analysis Techniques

Once intrusion detection data has been gleaned, an IDS uses its analysis engine to process this data in order to identify intrusions. Modern systems primarily employ two techniques to perform this analysis: misuse detection and anomaly detection.

### 3.1 Misuse Detection

The essence of misuse detection centers around using an expert system to identify intrusions based on a predetermined knowledge base. As a result, misuse systems are capable of attaining high levels of accuracy in identifying even very subtle intrusions that are represented in their expert knowledge base; similarly, if this expert knowledge base is crafted carefully, misuse systems produce a minimal number of false positives.[1]

A less fortunate ramification of this architecture results from the fact that a misuse detection system is incapable of detecting intrusions that are not represented in its knowledge base. Subtle variations of known attacks may also evade analysis if a misuse system is not properly constructed. Therefore, the efficacy of the system relies heavily on the thorough and correct construction of this knowledge base, a task that traditionally requires human domain experts.

### 3.2 Anomaly Detection

Anomaly detection is concerned with identifying events that appear to be anomalous with respect to normal system behavior. A wide va-

riety of techniques including statistical modeling, neural networks, and hidden Markov models have been explored as different ways to approach the anomaly detection problem. Each of these anomaly-based approaches fundamentally relies upon the same principles: anomalous activity is indicative of an attempted attack and the correct set of characteristics can sufficiently differentiate anomalies from normal system usage. Developing an anomaly detection system therefore involves first establishing a baseline model that represents normal system behavior and against which anomalous events can be distinguished. The system then analyzes an event by considering it within this model and classifying it as anomalous or normal based on whether it falls within a certain threshold of the range of normal behavior.

The most appealing feature of anomaly detection systems is their ability to identify new and previously unseen attacks. Because the process of establishing a baseline model of normal behavior is usually automated, anomaly systems also do not require expert knowledge of computer attacks. This approach is not without its handicaps, however, as anomaly detection may fail to detect even attacks that are very well-known and understood if these attacks do not differ significantly from what the system establishes to be normal behavior. Anomaly based systems are also prone to higher numbers of false positives, as all anomalous events are assumed to be intrusive although in reality a variety of other factors can produce behavior that appears anomalous (e.g., implementation errors).[1]

# 4 Intrusion Detection Issues

Individual systems take differing approaches to the problem of intrusion detection. There exist, however, a number of common issues that plague the range of detection strategies. This section examines a number of these issues and some of the ways in which researchers have attempted to ameliorate them.

## 4.1 Deriving an Expert Rule Set

As previously mentioned, one drawback of misuse detection systems is their reliance on an expert rule set that traditionally must be constructed by a human domain expert. This rule set is therefore expensive to produce and susceptible to human error.

Snort, a real-time NIDS, addresses this conundrum by minimizing the effort required to develop new attack rules. In Snort, each attack rule is a single line of text that specifies exactly which characteristics of a packet are to be examined and what values these characteristics must equal in order to trigger the rule.[11] This approach to the problem is helpful, but is limited in its ability to detect variations of codified attacks and does not resolve the issue of requiring a human expert to devise a knowledge base for the IDS.

A technique that allows for the automated construction of attack rules would therefore be tremendously valuable. The architecture for an artificial immune system (ARTIS) that Hofmeyr and Forrest have proposed takes a bold step in this direction. Based on a model of the human immune system, ARTIS is concerned with developing a set of lymphocytes (analogous to attack rules) that can detect pathogens (intrusive activity) in a system. In an attempt to closely model the way this is performed in the human immune system, ARTIS continuously generates lymphocytes with random characteristic values and deploys these to detectors that are replicated throughout the system. Over time, lymphocytes randomly die off and are replaced; when a particular lymphocyte correctly binds to a particular event sequence in the system, it is reinforced and its probability of being replaced is reduced.[5] In this fashion, those lymphocytes that are most effective in detecting misuse become more prevalent in the system.

This design is advantageous in that it eliminates much of the dependency on expert human knowledge in developing a misuse detection system. In reality, however, ARTIS does require human intervention in distinguishing the lymphocytes that bind correctly from those that do not in order to reduce the number of false positives that the system produces. Because lymphocytes are constructed at random, there is also a high probability that well-known attacks may not be detected at all if the characteristics needed to identify these attacks are never generated. ARTIS achieves one of the benefits of anomaly systems in the ability to detect new attacks without relying on a human-encoded expert knowledge base. It does this, however, at the expense of some of the advantage of traditional misuse systems, namely it is less effective at detecting known attacks and is more likely to produce false positives.

Another approach to the problem of deriving

expert system rule bases uses anomaly detection to identify new attacks which could then be codified into rules that could be used in a misuse system. This differs from ARTIS primarily in the sense that new attack rules are determined based on perceived anomalies rather than from randomly generated data. One of the critical challenges of such a system would be the proper encoding of an attack into an appropriate attack rule with the proper characteristics. To date, there has been little exploration of this problem space.

## 4.2 Detecting Attack Variations

Detecting subtle variations of known attacks presents a sizeable challenge to many systems that rely upon misuse detection. Because of the way intrusion signatures must be codified into an expert knowledge base, it can be very difficult for misuse systems to identify attacks that may originate from more than one source, vary in the means by which they are conducted, or are protracted over long periods of time.

State transition analysis (STAT) is one technique that addresses this issue. In such a system, attacks are represented by a state transition diagram: the start state represents a pristine system, intermediate states represent changes to the system that occur during an attack, and the final state represents a system compromise. For all of the codified attacks that exist in an IDS's rule base, the system retains internal data indicating which states of the attack have been reached. Because this data is global, it is independent of who causes a state change, the way in which a new state is reached, and the time scale on which state transitions occur. Through this means, countless variations of a single attack can still be detected because the system monitors system state changes that are symptomatic of an intrusion attempt rather than monitoring the actions that cause those state changes.[6]

This advantage is not gained, however, without considerable expense. Because the occurrence of state transitions in an attack may not be a simple linear progression, the system must maintain internal data for every intermediate state of an attack that has ever been reached. This can lead to an explosion in the amount of data that the system must maintain as the number of attacks and attack states monitored becomes large, a property that an attacker could exploit. Codifying attacks as state transition diagrams also complicates the process of developing the system's expert knowledge base: if a critical attack state is omitted from the diagram, false positives result when state progressions resembling an attack occur; if a non-critical state is inserted into an attack's diagram, false negatives occur when variations of the attack transpire that do not instantiate this non-critical state.

A similar tactic to the problem of detecting attack variations makes use of Colored Petri Nets to monitor codified event sequences. Colored Petri Nets allow for attacks to be encoded as graphs in such a way that the progression of attack steps is more flexible than the rigid order imposed by state transition analysis. Guards are used to define the situations in which attack signatures are matched rather than maintaining data indicating all current and previous states.[7] This helps to achieve many of the benefits of using the STAT approach without making the system vulnerable to the state-consuming attacks.

Unfortunately, Colored Petri Nets do not present a viable option for practical intrusion detection because the process of state unification and matching in this model is prohibitively compute-expensive. The complexity of this matching is exponential on the number of states represented by the system, while partial order matching requires super-exponential time.[7] Although an IDS based on Colored Petri Nets would not be vulnerable to a state-consuming attack, it would be very vulnerable to a CPU-consuming attack. The efficacy of such a system would therefore be rapidly eroded in a production environment in which an attacker could render the IDS useless by overwhelming the analysis engine.

## 4.3 Training Behavioral Models

Anomaly systems universally suffer from the problem of how to correctly construct a baseline model of behavior that is sufficient for complete and correct operation of the system. Any successful means of training the system must expose the system to the full range of normal behavior in order to minimize false positives as well as avoid exposing the system to properties of anomalous activity that may desensitize the IDS to attacks.

Depending on the design of the anomaly system, systems can be trained with just normal data or with two sets of data that are correctly identified as normal and intrusive. In presenting intrusive data that is going to be used to train the

system, it is important that this data represent a range of anomalies so that the trained system is not incapable of identifying certain classes of attack.

Ghosh et al. have experimented with using randomly generated events to represent anomalous behavior in order to train the neural networks that provide the analysis for their IDS. In empirical tests, those networks that were trained with randomly generated anomaly data consistently out-performed those that did not receive this training by reducing the number of false negatives in the system (none of the networks produced any false positives).[3] While these results are very promising, they suggest that a likely reason that the neural networks trained with random data performed well is because the normal data set with which they were trained defined a narrow range of behavior that very closely resembled the normal data that the system was tested against. Were this not the case, the system could reasonably have been expected to generate at least a small number of false positives as some of the randomly generated events trained as anomalies would fall into the range of normal activity. It is further unclear how well a system trained over random anomalous data would perform in correctly identifying actual attacks, as the attack data against which this system was tested also consisted of randomly generated events.

A more ideal solution to the training problem would be one that allows for an anomaly system to be correctly trained over noisy data, ie. data that contains an assortment of both normal and anomalous behavior. This would allow for the system to be effectively trained in a production environment without relying on hypothetical data sets representing normal and anomalous behavior.

Eleazar Eskin has developed a process that uses learned probability distributions to train an anomaly system over noisy data. This technique uses machine learning to create a probability distribution of the training data and then applies a statistical test to identify anomalies.[2] Interestingly, Eskin's technique requires no domain-specific knowledge. It does, however, operate on three assumptions about the training data: normal data can be effectively modeled using a probability distribution; anomalous events differ significantly enough from normal events that they can be identified; and the number of anomalous events is small compared to the number of normal events. Furthermore, before the system is trained, one must define a value $\lambda$ indicating the percentage of the training data that is expected to be anomalous. Because this data is noisy and not artificially constructed, choosing the value of $\lambda$ to best ensure correct operation of the system is very difficult. This model is additionally limited by its assumption that normal data is distributed normally across noisy data: in actuality, it is likely that intrusion attempts are temporally clustered.

Regardless of the means by which one is trained, there is also the issue of evolving normal behavior with which anomaly systems must contend. One option is that a system be retrained periodically with new training data that represents current normal behavior. This, however, increases the dependency of the system on training and underscores the inherent difficulties in developing sufficient data or an appropriate technique for this task.

Adaptive anomaly systems have been suggested as a solution that would allow for systems to evolve their normal behavior models gradually as normal behavior evolves. Lane and Brodley's machine learning system is an example of a system that makes use of this technique. This system maintains a finite-sized dictionary of normal event sequences and uses a least-recently-used (LRU) policy to replace seldom-occurring sequences with new ones that are determined to be normal.[8] It is important to note that systems that use adaptive training techniques face the problem of preventing an attacker from gradually training the system over time to accept a range of anomalous behavior as normal. Resolving this difficulty remains an open challenge.

### 4.4 Attack Against the IDS

While the purpose of an intrusion detection system is to detect attacks against a host or set of hosts, an ironic consequence of its existence is that the IDS itself may draw attack from an attacker seeking to disable the IDS. It is critical that the design of a system be performed within the framework that the IDS itself be resistant to and tolerant of attack attempts designed to obstruct its ability to correctly detect intrusions.

One class of such attacks referred to as "crash attacks" attempts to disable an IDS by causing it to fault or to run out of some critical resource. Assuming that it is infeasible to totally prevent these attacks, the goal of an IDS in the face of such an attack is therefore to minimize

the extent to which the attacker is successful in disabling the IDS.

Bro, a real-time system for detecting network intrusions, provides two mechanisms for maximizing operation in the face of crash attacks. First, Bro maintains a "watchdog" timer that expires on a configurable interval and checks to see if the system is still analyzing the same event that it was when the previous timer expired. If this is the case, the system assumes that it is in a processing jam and terminates the monitor process so that the system can continue with the next event. Second, Bro is launched from a script that can recognize if the system ever crashes, in which case it launches tcpdump in order to gather the data that Bro would be gathering had it not crashed. This data can then be analyzed at a later time or by another system.[10]

While these facilities are a certain improvement over a system that has no means of providing failure recovery, they are certainly not without weakness. Provided that an attacker has a means of engaging an IDS in a processing jam, it would be a relatively easy matter to simply inject a series of such events into the data stream in order to keep the system occupied while an attack on a monitored host is launched. Although the watchdog timer ensures that Bro never becomes permanently disabled by a single series of events, it cannot prevent a determined attacker from creating a quantum of time during which an attack can proceed undetected. Bro's second provision suffers similarly: although the script does ensure that any intrusion that follows Bro's crash will be recorded, it provides no means of analyzing this record. Additionally, because tcpdump is launched immediately after the IDS crashes, it cannot determine the sequence of events that caused the crash to occur.

Another class of attacks seeks simply to inject a large quantity of spurious data into a monitor's event stream in order to distract an IDS while an attack on a monitored host takes place. Such attacks can be particularly lethal when launched against a NIDS that is already under the onus of monitoring and performing analysis on data for a large number of hosts. In the face of attacks like these, Vern Paxson suggests that such systems perform "triage" against incoming flows: if the system detects that it is nearing exhaustion, it can shed load by discarding state for monitored flows that do not appear to be making progress. This suggestion operates under the assumption that an attacker is less likely to have complicity from hosts on both sides of the monitor, therefore making it difficult for the attacker to fake a large number of active connections.[4]

The idea of triage is a precarious one. On the one hand, the load that is shed during triage can allow for an IDS to operate continuously, helping to maximize the coverage of the system and prevent an attacker from denying service to the system by overwhelming it with illegitimate data. On the other hand, when a system enters a mode of triage, it is in essence performing denial of service on itself. The efficacy of a triage mechanism therefore hinges on the system's ability to properly determine which data it can safely ignore and which data it cannot: if the system were able to make this distinction perfectly, however, there would never be any need to examine the irrelevant data. As with the survival facilities that Bro employs, the adoption of triage involves a tradeoff between correctness and performance, but is certainly preferable to the absence of such mechanisms which may sacrifice both.

# 5 Results

This paper has presented a veritable cornucopia of intrusion detection systems and discussed the relative merits of each, but has not addressed the issue of quantifiable results. The matter of performance metrics is another extremely challenging issue in intrusion detection and one that does not lend itself well to simple empirical evaluation. In evaluating intrusion detection systems, the three most important qualities that need to be measured are completeness, correctness, and performance.[4]

The current state of the art in intrusion detection restricts measurement of new systems to tests over incomplete data sets and micro-benchmarks that can test a narrowly defined component of the system. Presently, a number of anomaly-based systems are tested over contrived data sets in order to determine how well the system classifies anomalies. This evaluation is limited by the quality of the data set that the system is measured against: constructing data sets that are both realistic and comprehensive is an extremely hard and open problem. Examples of micro-benchmarks include stress tests to expose the maximum rate of events that an IDS can withstand before it begins to experience exhaustion or running the system in a production environment to determine the speed with which

it can be expected to perform under typical load. Tests such as these can give a good indication as to the computational boundaries of an IDS but are very limited in the degree to which they can quantify completeness and correctness.

A number of ideas for the establishment of security metrics have been proposed. For instance, "pretty good assurance" seeks to provide a process by which claims about the security properties of systems can be clearly stated and accompanied by evidence that substantiates these claims. As formal proof of correctness in the intrusion detection domain is exceptionally challenging and expensive, pretty good assurance presents a way in which systems can be measured that allows fuzzy decisions, trade-offs, and priorities as long as these properties are accompanied by appropriate assurance arguments.[12]

Bennet Yee has suggested another metric in which the strength of a system is measured by the work factor required for an attacker to penetrate the system's defenses. Such a measure must take into consideration the amount of work required to discover a vulnerability, engineer a means to exploit this weakness, and execute an attack on the system.[13] Although such a measurement inherently involves a good deal of approximation and guesswork, the concept of work factor yields great promise in providing an acceptable benchmark against which intrusion detection systems could be compared.

# 6 Future Research

The study of intrusion detection systems is quite young relative to many other areas of systems research and it stands to reason that this topic offers a number of opportunities for future exploration. In discussing some of the problematic issues that confront IDS designers, this paper has touched upon a number of open questions related to intrusion detection including:

- Can anomaly detection systems be used to generate attack rules for misuse detection systems?
- In what ways can variations of known attacks be detected by a misuse system without exposing the IDS to resource-consuming attacks?
- Can an anomaly system that adaptively modifies its model of normal behavior over time be protected from being training by attackers to accept intrusions as normal behavior?
- Is it possible for triage mechanisms to provide an IDS with the ability to shed load without diminishing its efficacy or its coverage?
- How can the completeness, correctness, and performance of intrusion detection systems be measured in order to facilitate relative comparison and absolute evaluation of these systems?

In addition to these issues, there are a number of unresolved issues regarding the scope of analysis that an IDS performs and the interoperability of intrusion detection systems. Most intrusion detection efforts today focus on providing analysis for a relatively localized target: either a single host or a collection of hosts joined by a network. A system that operates with a more global scope may be capable of detecting distributed attacks or those that affect an entire enclave. Development of such a system would be a valuable contribution to the study of intrusion detection.

There have recently been a number of efforts including the Common Intrusion Detection Format (CIDF) and the IETF standardization effort motivated towards providing interoperability among intrusion detection systems. Such frameworks can provide a means by which differing analysis and data collection techniques can be aggregated within a single system, improving both the coverage and redundancy of the system. An increasing number of intrusion detection systems such as EMERALD are beginning to make use of this idea, although it will likely be some time before a standard framework finds its way into widespread use.[9] More research towards synthesizing the commonalities of intrusion detection systems and the most efficient format for inter-IDS communication is still needed.

# 7 Conclusions

Since the study of intrusion detection began to gain momentum in the security community roughly ten years ago, a number of diverse ideas have emerged for confronting this problem. Intrusion detection systems vary in the sources they use to obtain data and in the specific techniques they employ to analyze this data. Most systems today classify data either by misuse detection or

anomaly detection: each approach has its relative merits and is accompanied by a set of limitations.

It is likely not realistic to expect that an intrusion detection system be capable of correctly classifying every event that occurs on a given system. Perfect detection, like perfect security, is simply not an attainable goal given the complexity and rapid evolution of modern systems. An IDS can, however, strive to "raise the bar" for attackers by reducing the efficacy of large classes of attacks and increasing the work factor required to achieve a system compromise. The coordinated deployment of multiple intrusion detection systems promises to allow greater confidence in the results of and to improve the coverage of intrusion detection, making this a critical component of any comprehensive security architecture.

# References


1. Rebecca Bace and Peter Mell. "NIST Special Publication on Intrusion Detection Systems," 16 August 2001.

2. Eleazar Eskin. "Anomaly Detection over Noisy Data Using Learned Probability Distributions," *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000)*, Palo Alto, California, July 2000.

3. Anup K. Ghosh, James Wanken, and Frank Charron. "Detecting Anomalous and Unknown Intrusions Against Programs," *Annual Computer Security Applications Conference (ACSAC'98)*, Scottsdale, Arizona, 7-11 December 1998.

4. Mark Handley, Vern Paxson, and Christian Kreibich. "Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics," *10th USENIX Security Symposium*, Washington, D.C., 13-17 August 2001.

5. Steven A. Hofmeyr and S. Forrest. "Architecture for an Artificial Immune System," *Evolutionary Computation Journal*, 2000.

6. Koral Ilgun, Richard A. Kemmerer, and Phillip A. Porras. "A State Transition Analysis Tool for Intrusion Detection," *IEEE Transactions on Software Engineering*, 1995.

7. Sandeep Kumar and Eugene H. Spafford. "A Pattern Matching Model for Misuse Intrusion Detection," *Proceedings of the 17th National Computer Security Conference*, pp. 11-21, October 1994.

8. Terran Lane and Carla E. Brodley. "An Application of Machine Learning to Anomaly Detection," *20th Annual National Information Systems Security Conference*, 1, pp. 366-380, 1997.

9. Peter G. Neumann and Phillip A. Porras. "Experience with EMERALD to Date," *1st USENIX Workshop on Intrusion Detection and Network Monitoring*, Santa Clara, California, 11-12 April 1999.

10. Vern Paxson. "Bro: A System for Detecting Network Intruders in Real-Time," *Computer Networks*, 31, pp. 2435-2463, Dec. 1999.

11. Martin Roesch. "Snort - Lightweight Intrusion Detection for Networks," *13th USENIX Systems Administration Conference - LISA '99*, Seattle, Washington, 7-12 November 1999.

12. Jeffrey R. Williams, Marv Schaefer, and Douglas J. Landoll. "Pretty Good Assurance," *New Security Paradigms Workshop*, September 1995.

13. Bennet S. Yee. "Security Metrology and the Monty Hall Problem," *Workshop on Information Security System Rating and Ranking*, May 2001.