Experience with EMERALD to Date

Peter G. Neumann and Phillip A. Porras
Computer Science Laboratory
SRI International, Menlo Park CA 94025-3493
Neumann@CSL.sri.com and Porras@CSL.sri.com
1-650-859-2375 and 1-650-859-3232

1st USENIX Workshop on Intrusion Detection and Network Monitoring
Santa Clara, California, 11-12 April 1999

Abstract

After summarizing the EMERALD architecture and the evolutionary process from which EMERALD has evolved, this paper focuses on our experience to date in designing, implementing, and applying EMERALD to various types of anomalies and misuse. The discussion addresses the fundamental importance of good software engineering practice and the importance of the system architecture – in attaining detectability, interoperability, general applicability, and future evolvability. It also considers the importance of correlation among distributed and hierarchical instances of EMERALD, and needs for additional detection and analysis components.

1. Introduction

EMERALD (Event Monitoring Enabling Responses to Anomalous Live Disturbances) [6, 8, 9] is an environment for anomaly and misuse detection and subsequent analysis of the behavior of systems and networks. EMERALD is being developed under DARPA/ITO Contract number F30602-96-C-0294 and applied under DARPA/ISO Contract number F30602-98-C-0059. EMERALD has farsighted goals for real-time detection, analysis, and response for a broad range of threats other than just security.

Anomaly detection involves the recognition of deviations from expected normal behavior, whereas misuse detection involves the detection of various types of misuse. The term "intrusion detection" is often used to encompass both, but unfortunately suggests only the *detection of intrusions* rather than the broader scope of EMERALD.

2. EMERALD

EMERALD targets both external and internal threat agents that attempt to misuse system or network resources. It is an advanced highly software-engineered environment that combines signature-based and statistical analysis components with a resolver that interprets analysis results, all of which can be used iteratively and hierarchically. Its modules are designed to be independently useful, dynamically deployable, easily configurable, reusable, and broadly interoperable. Its design scales well to very large enterprises. The objectives include achieving innovative analytic abilities, rapid integration into current network environments, and much greater flexibility of surveillance whenever network configurations change.

EMERALD employs a building-block architectural strategy using independently tunable distributed surveillance monitors that can detect and respond to malicious activity on local targets, and can interoperate to form an analysis hierarchy. The basic architectural structure is shown in Figure 1. The figure shows the three main types of existing analysis units (profiler engines, signature engines, and resolver) surrounding the target-specific resource objects. It also shows the possible integration of third-party modules, including inputs derived from other sources, and outputs sent to other analysis platforms or administrators and emergency response centers. This architecture is explained in the following text.

A key aspect of this approach is the introduction of EMERALD monitors. An EMERALD monitor is dynamically deployed within an administrative domain to provide localized real-time analysis of infrastructure (e.g., routers or gateways) and ser-

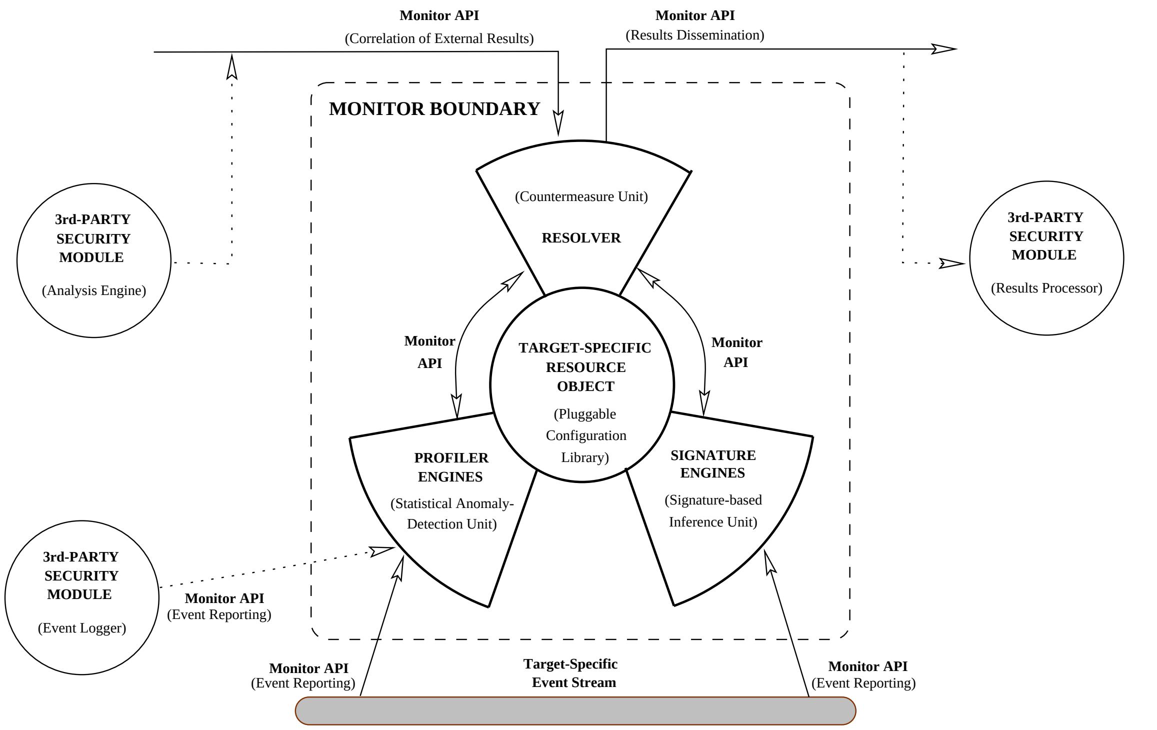

Figure 1: The Generic EMERALD Monitor Architecture

vice (privileged subsystems with network interfaces). An EMERALD monitor may interact with its environment passively (reading activity logs or network packets) or actively (via probing that supplements normal event gathering). As monitors produce analytical results, they are able to disseminate these results asynchronously to other client monitors. Client monitors may operate at the domain layer, correlating results from service-layer monitors, or at the enterprise layer, correlating results produced across domains.

Under the EMERALD framework, a layered analysis hierarchy may be formed to support the recognition of more global threats to interdomain connectivity, including coordinated attempts to infiltrate or destroy connectivity across an enterprise.

Equally important, EMERALD does not require the adoption of this analysis hierarchy. Monitors themselves stand alone as self-contained analysis modules, with a well-defined interface for sharing and receiving event data and analytical results among other third-party security services. An EMERALD monitor is capable of performing both signature analysis and statistical profile-based anomaly detection on a target event stream. In addition, each monitor includes an instance of the EMERALD resolver, a countermeasure decision engine capable of fusing the alerts from its associated analysis engines and invoking response handlers to counter malicious activity. The statistical subsystem tracks subject activity via one of four types of statistical variables called measures: categorical (e.g., discrete types), continuous (e.g., numerical quantities), traffic intensity (e.g., volume over time), and event distribution (e.g., a meta-measure of other measures) [9]. EMERALD's signature analysis subsystem employs a variant of the P-BEST (Production-Based Expert System Tool) expert system [6] that allows administrators to instantiate a rule set customized to detect known "problem activity" occurring on the analysis target. Results from both the statistical and signature engines are then forwarded to the monitor's resolver – which acts as the coordinator of the monitor's external reporting system and the implementor of the monitor's response policy.

Fundamental to EMERALD's design is the abstraction of analysis semantics from the monitor's code base. Under the EMERALD monitor architecture, all analysis-target-specific information is contained within each resource object, specifying items from a pluggable configuration library. The resource object encapsulates all the analysis semantics necessary to instantiate a single service monitor, which can then be distributed to an appropriate observation point in the network. Resource-object elements customize the monitor for the analysis target, containing data and methods, such as the event collection methods, analytical module parameters, valid response methods, response policy, and subscription list of external modules with which the monitor exchanges alarm information. This enables a spectrum of configurations from lightweight distributed monitors to heavy-duty centralized analysis platforms.

In a given environment, service monitors may be independently distributed to analyze the activity of multiple network services (e.g., FTP, SMTP, HTTP) or network element (router, firewall). Resource objects are being developed for each analysis target. As each EMERALD monitor is deployed to its target, it is instantiated with an appropriate resource object (e.g., an FTP resource object for FTP monitoring, and a BSM resource object for BSM Solaris kernel analysis). The monitor code base itself is analysis target-independent. As EMERALD monitors are redeployed from one target to another, the only thing that is modified is the content of the resource object.

See the paper by Lindqvist and Porras [6] for discussion of the analysis of FTP (which currently exists for SunOS, FreeBSD, and Linux) and BSM (on Solaris). In particular, that paper gives specific examples of rules for failed authentication, buffer overflows, and SYN flooding attacks.

Resource objects lend themselves to the key project objectives of reusability and fast integration to new environments. The project is developing a library populated with resource objects that have been built to analyze various service and network elements. Installers of EMERALD will be given our monitor code base, which they do not have to touch. They can then download appropriate resource objects associated with their analysis targets, modify them as desired, and instantiate the monitors with the downloaded resource objects.

The project is also working toward new techniques in alarm correlation and management of analytic services. The concept of composable surveillance will allow EMERALD to aggregate analyses from independent monitors in an effort to isolate commonalities or trends in alarm sequences that may indicate a more global threat. Such aggregate analyses are classified under four general categories: commonality detection, multiperspective reinforcement, alarm interrelationships, and sequential trends.

Briefly, commonality detection involves the search for common alarm indicators produced across independent event analyses. In such cases, the results from one monitor's analyses may occur under a threshold that warrants individual response, but in combination with results from other monitors may warrant a global response. This approach can address low-rate distributed attacks and cooperative attacks, as well as widespread contamination effects. Multiperspective analysis refers to efforts to independently analyze the same target from multiple perspectives (e.g., an analysis of a Web server's audit logs in conjunction with Web network traffic). Alarm interelationships refer to EMERALD's ability to have a monitor model an interrelationship (cause and effect) between the occurrence of alarms across independent analysis targets. For example, an alarm regarding activity observed on one host or domain may give rise to a warning indicator for a different threat against a second host or domain. Last, sequential trends in alarms seek to detect patterns in alarms raised within or across domains. These patterns of aggressive activity may warrant a more global response to counteract than can be achieved by a local service monitor.

The EMERALD project represents an effort to combine research from distributed high-volume event correlation with over a decade of intrusion-detection research and engineering experience. It represents a comprehensive attempt to develop an architecture that inherits well-developed analytical techniques for detecting intrusions, and casts them in a framework that is highly reusable, interoperable, and scalable in large network infrastructures. Its inherent generality and flexibility in terms of what is being monitored and how the analytical tools can be customized for the task suggest that EMERALD can be readily extended for monitoring other forms of malicious and nonmalicious "problem activities" within a variety of closed and networked environments.

3. Experience Gained

This section summarizes our experience in the EMERALD development thus far.

Earlier Experience

EMERALD has drawn on our earlier experience in developing and using IDES (Intrusion Detection Expert System [7]) and its successor NIDES (Next-Generation IDES [1, 2, 3, 4]. Particularly for those people who are not aware of our earlier work, we summarize a few conclusions.

- From IDES, we attained considerable flexibility and runtime efficiency in the use of P-BEST [7]), which we have now adapted into EMERALD's pluggable analysis-engine framework as a self-sufficient component. The P-BEST approach proved to be very useful, and rules are relatively easy to write. P-BEST was adapted by Alan Whitehurst from its previous incarnation in MIDAS [10]. IDES also gave us the second generation of our statistical algorithms, begun in 1983 in an earlier project [5].
- From the NIDES development [1], several observations influenced the EMERALD effort. (1) Much of the available audit data (e.g., from C2 Unix and BSM) was not naturally well suited for our analytical purposes, and different sources of data would have been desirable. Greater abstraction would have been useful. (2) Although we did experiment with some higher-level audit data (from database management systems in relatively closed environments), attempting to detect misuse was less fruitful because the security policies of the DBMSs generally permitted what was closer to acceptable behavior. (3) We recognized that the NIDES statistical detection system as then configured would not scale well to distributed and networked environments, for two reasons. First, the measures needed to be treated in their entirety, rather than subsetted – as would be desirable for lightweight instances. Second, the results were not in a form that could be used recursively at a higher-layer instance. (4) We recognized the importance of the administrator interface, and observed that its complexities are unavoidable if flexibility in detection and response is required. However, we initially spent too much effort on developing our own GUI tools, until we decided to rely on some newly developed generic tools. In retrospect, we believe we would have progressed faster if we had had more emphasis on software engineering and on in-house applications.
- From the NIDES Safeguard effort [2], we observed that profiling functionality proved to be more effective than profiling individual users. That approach resulted in far fewer profiles, each of which tended to be much more stable. The resulting false-positive and false-negative rates were reduced considerably. We concluded that statistical analyses could be very effective in dealing with systems and subsystems such as servers and routers. (As a consequence, EMERALD subsequently broadened the statistics algorithms to improve handling of network protocols, by having a master profile of client usage against which a single service can be compared. For example, anonymous FTP sessions can simultaneously be profiled against the master profile for anonymous sessions.)

These observations have had a significant impact on the EMERALD architecture and its implementation, particularly in moving to a distributed and networked target environment.

EMERALD Experience

The underlying generic analysis-engine infrastructure uniformly wraps the signature analysis, statistical engine, resolver, and any future engines we might wish to integrate. The infrastructure provides the common EMERALD API, event-queue management, error-reporting services, secondary storage management (primarily for the statistical component), and internal configuration control. The statistical and P-BEST components are integrated as libraries. The infrastructure was assembled first for the EMERALD statistics component (estat), but proved its generality when we attempt to integrate P-BEST as the EMERALD expert system (eXpert). The integration of P-BEST inference engines required some linkage code to bind with the underlying EMERALD libraries, and is now automatically generated as part of the compilation process.

After more than two years developing EMERALD, our experience thus far is summarized as follows.

- Generality of approach. We have attempted to solve some difficult problems rather generally, and have typically avoided optimizing our approach to any domain-specific assumptions. In particular, the decoupling of generic and target-specific concepts simplifies reusability of components and extensibility, and enhances integration with other data sources, analysis engines, and response capabilities. The hierarchically iterative nature permits analyses with broader scope across networks and distributed systems. Although the advantages of such a farsighted approach may not be evident until EMERALD is more widely used and extended to new application areas, we firmly believe that this approach can be very instructive to us and to other groups, from the perspective of research and development potential – and can have major long-term advantages. (Platform-specific optimizations are of course possible, if they are deemed necessary.)

- Software engineering. We believe that our strong emphasis on good software engineering practice in EMERALD has already had substantial payoffs, particularly in enabling us to rapidly incorporate different analytic engines into the generic framework. (The modularization and integration of the P-BEST expert system component is discussed below.) This emphasis clearly improves the general evolvability of the system, and also has significant benefits with respect to interoperability – within EMERALD, with independently developed analysis engines, with analysis data from arbitrary sources, and in terms of the distribution of analysis results. The software-engineering emphasis also helps facilitate the iterative use of EMERALD analytic engines by making the layered instances of the system symmetric. These benefits remain to be demonstrated explicitly with extensive and well-documented experiments, but our expectations are very high. A fuller justification of the extent to which this software engineering approach is actually paying off requires a more detailed description of the architecture, which is beyond the scope of this workshop paper; however, such a description is high on our priority list for the future.

- Scope of applicability. We believe that our attention to software engineering simplifies the broadening of EMERALD's domains of applicability – for example, detecting, analyzing, and responding to potential threats to survivability, reliability, fault tolerance, and network management stability. There is nothing intrinsic in the EMERALD architecture and implementation that would limit its applicability. The application to requirements other than security is basically a matter of writing or modifying the relevant resource objects and configuring the system appropriately, and is not expected to require major changes to the existing analysis infrastructure.

- Relative merits of various paradigms. It should be no surprise to those in the intrusion-detection community that signature-based analysis is good at detecting and identifying well-defined known scenarios, but very limited in detecting hitherto unknown attacks (except for those that happen to trigger existing rules serendipitously). On the other hand, statistical profile-based analysis can be effective in detecting unknown attacks and providing early warnings on strangely deviant behaviors; however, the statistical approach does not naturally contribute to an automated identification and diagnosis of the nature of an attack or other type of deviation that it has never identified before. Although inferences can be drawn about the nature of an anomaly, based on the statistical measures that were triggered, further reasoning is typically necessary to identify the nature of the anomaly – for example, is it an attack in progress, or a serious threat to system survivability.

  Precisely because it is aimed at detecting potentially unforeseen threats rather than very specific scenarios that can be easily detected by signature-based analyses, the statistical component can be expected to turn up false positives. In the EMERALD framework, this is not necessarily a problem. We believe it is much more effective for the resolver to discard statistical anomalies that it deems nonserious rather than try to reduce the false positives in the statistical component itself (which requires greater knowledge of the potential threats – which is what can otherwise be avoided). Furthermore, once new attacks and threats are identified, it is desirable to add new rules to the expert-system rule base.

  Overall, we believe that each type of analysis (such as the expert system, the statistical component, the resolver, or any additional analysis engines) will have its own areas of greatest effectiveness, but that no one paradigm can cover all types of threats. Therefore we endorse a pluralistic approach. Inference and reasoning engines, Bayesian analysis, and other paradigms may also be applicable to detection, identification, and resolution of the nature of anomalies and attacks.

- Local, hierarchical, and distributed correlation. One of the most far-reaching observations relates to the importance of being able to correlate local results from different target platforms at the same or different layers of abstraction, and also to correlate results relating to different aspects of system behavior. The inherent layered iterative nature of the EMERALD architecture is significant in this respect, because the same analytic component can be used at different layers of abstraction. We are just now beginning to conduct some experiments to demonstrate the power of this approach. In so doing, we are extending the existing EMERALD resolver to interpret the results of different analytic engines and to recommend responses appropriate to the specific layer of abstraction. Further analytic engines may also be required at various layers of abstraction, such as some reasoning tools.

- Importance of further research, prototype development, and experimentation. EMERALD continues to explore advanced concepts, as did IDES and NIDES. Although most of the necessary analysis infrastructure is now in place, R&D advances are still required for EMERALD relating to inference necessary to enhance correlation in the analysis of and response to coordinated attacks and interdependent anomalies in distributed environments, and in generalizations of applicability beyond security. These are ongoing efforts.

- Interoperability. The Common Intrusion Detection Format (CIDF) and the ongoing IETF standardization effort are important. Both are expected to increase the interoperability within and among different analysis and response systems. EMERALD is very much in line with these efforts, and compatibility is not expected to be a problem. CIDF interface definitions are based on an architectural decomposition that is aligned closely to that of EMERALD's monitor design. In particular, EMERALD's target-specific event-generation components are equivalent in function to CIDF E-boxes; EMERALD's statistical and signature analysis engines are equivalent in function to CIDF A-boxes; EMERALD's resolver is equivalent in function to a CIDF R-box. In hierarchical composition, an EMERALD service layer monitor is capable of passing alerts to a domain monitor. The service layer monitor can operate as a CIDF E-box, and the domain monitor can operate as a CIDF A-box. CIDF working documents are available online (`seclab.csl.ucdavis.edu/cidf`).

EMERALD's Expert System

With respect specifically to the integration of P-BEST into EMERALD [6], our experience has strongly reinforced our conceptual framework.

- The software engineering quality of the EMERALD monitor architecture was put to a test when a summer visitor previously unfamiliar with the system joined us to integrate the signature analysis engine into the generic monitor framework. The statistical anomaly detection engine had been developed in concert with the EMERALD API, and the NIDES expert-system-based signature engine was the first additional component to use the API. The revision and integration procedure went very rapidly (about a man-week), and minor problems that were discovered and solved were due to constraints in the expert-system tool rather than in the EMERALD API. This supports our claim that the EMERALD API is well suited for integration of various kinds of third-party modules into the monitor architecture. Although this is not an exciting *gotcha,* it was important to the development effort.

- The data-driven nature of the EMERALD monitors makes the intermonitor and intramonitor message-passing a central function of the API. The programmer is provided with a set of abstract data types, including a set of methods to handle messages and fields within messages. An example of a powerful feature of the EMERALD message format is the possibility of defining a message field as an array of message fields. This allows the programmer to effectively encapsulate one EMERALD message inside another. In the signature-analysis engine, this capability is used to include the original event record(s) in every alert message sent to the resolver, in addition to the information provided by the triggered rules. This also allows a hierarchy of analysis units (including resolvers) to be able to pass along any or all information produced earlier.

- The generality of the API with respect to the abstract data types is also reflected by the ease with which we were able to write a code-generation utility for the interface code that connects the expert system to the monitor. This utility is used when redirecting the signature-analysis engine to a completely new event stream, using the information in the resource object to fit the engine to the analysis target. The purpose of the utility is to relieve the creator of a resource object from the inner workings of the monitor. The API design made it easy to isolate the target-dependent code and let it be machine-generated.

4. Conclusions

Overall, the progress to date in developing and using EMERALD has been very promising. However, considerable further effort is needed to demonstrate the effectiveness of the software engineering approach and the power of the analytic capabilities.

- The software engineering practice used in EMERALD's modular design and the attention devoted to well-defined interfaces and information hiding in the sense of David Parnas have proven very valuable in EMERALD's development thus far, and will be even more valuable to the ability to interoperate with components developed elsewhere, to its long-term evolvability, and to subsequent generalizations of EMERALD beyond security applications to address human safety, enterprise survivability, reliability, real-time performance, and other critical attributes.
- Hierarchical and distributed correlation is necessary in analyzing highly distributed environments, because of the inability to recognize global patterns from isolated local events. However, additional analysis techniques are likely to be required.
- The iterative nature of EMERALD instantiations will enable lightweight detection components to specialize in particular areas of concern, for different event spaces and at different layers of abstraction.

A few general conclusions are also noted in an attempt to put the EMERALD experience in perspective.

- Commercial intrusion-detection systems have concentrated mostly on string matching and other forms of signature identification to detect classes of outsider attacks. To date, primarily the easy parts of the problem have been addressed by the commercial community.
- Research advances in the community at large seem to have slowed, along with the increased emphasis on detecting known types of outsider attacks. Detecting, identifying, and responding to hitherto unknown attacks and anomalies remain as very challenging problems, including highly coordinated attacks, subtle forms of misuse by insiders, and anomalous network behavior resulting from malfunctions and outages. Providing global rather than local analysis is still a very important research area that is relatively uncharted. Generalizations beyond known security attacks are also challenging.

Further Information

See `http://www.csl.sri.com/intrusion.html` for background and online versions of papers and reports [2, 4, 6, 8, 9]. See also Web pages for Porras and Neumann (`www.csl.sri.com/users/porras/` and `www.csl.sri.com/users/neumann/`).

Acknowledgments

We are indebted to Martin Fong, Ulf Lindqvist, Keith Skinner, and Al Valdes, all of whom have contributed significantly to the EMERALD development. We are also grateful to those people whose work on the development of IDES and NIDES has influenced EMERALD, including quite notably Teresa Lunt. Charles Antonelli made some very helpful suggestions on the paper, and served as our official workshop shepherd.

References

[1] D. Anderson, T. Frivold, and A. Valdes. Next-generation Intrusion-Detection Expert System (NIDES). Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, SRI-CSL-95-07, May 1995.

[2] D. Anderson, T. Lunt, H. Javitz, A. Tamaru, and A. Valdes. Safeguard final report: Detecting unusual program behavior using the NIDES statistical component. Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, 2 December 1993.

[3] R. Jagannathan, T.F. Lunt, D. Anderson, C. Dodd, F. Gilham, C. Jalali, H.S. Javitz, P.G. Neumann, A. Tamaru, and A. Valdes. System

Design Document: Next-generation Intrusion-Detection Expert System (NIDES). Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, 9 March 1993.

[4] H.S. Javitz and A. Valdes. The NIDES statistical component description and justification. Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, March 1994.

[5] H.S. Javitz, A. Valdes, D.E. Denning, and P.G. Neumann. Analytical techniques development for a statistical intrusion-detection system (SIDS) based on accounting records. Technical report, SRI International, Menlo Park, California, July 1986.

[6] U. Lindqvist and P.A. Porras. Detecting computer and network misuse through the Production-Based Expert System Toolset (P-BEST). In *Proceedings of the 1999 Symposium on Security and Privacy*, Oakland, California, May 1999. IEEE Computer Society.

[7] T.F. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, C. Jalali, P.G. Neumann, H.S. Javitz, and A. Valdes. A Real-Time Intrusion-Detection Expert System (IDES). Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, 28 February 1992.

[8] P.A. Porras and P.G. Neumann. EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. In *Proceedings of the Nineteenth National Computer Security Conference*, pages 353–365, Baltimore, Maryland, 22-25 October 1997. NIST/NCSC.

[9] P.A. Porras and A. Valdes. Live traffic analysis of TCP/IP gateways. In *Proceedings of the Symposium on Network and Distributed System Security*. Internet Society, March 1998.

[10] M.M. Sebring, E. Shellhouse, M.E. Hanna, and R.A. Whitehurst. Expert system in intrusion detection: A case study. In *Eleventh National Computer Security Conference*, Baltimore, Maryland, October 1988.