

# XIDR: A Dynamic Framework Utilizing Cross-Layer Intrusion Detection for Effective Response Deployment

Igors Svecs, Tanmoy Sarkar, Samik Basu, Johnny S. Wong

*Department of Computer Science*

*Iowa State University*

*Email: {isvecs, tanmoy, sbasu, wong}@cs.iastate.edu*

**Abstract**—We present a complete intrusion detection and response framework named XIDR (Cross-layer Intrusion Detection and Response), which utilizes multi-source intrusion detection systems to enable cross-layer intrusion detection and cross-layer automated intrusion response system to deploy cost-effective and efficient preemptive responses. In this paper, we define the notion of cross-layer design which integrates features from various layers for detecting intrusions in wired environment, enables more fine grained detection technique and also helps us to reduce false positive and false negative rate. Moreover, cross-layer based approach for selecting and deploying response will help to deploy responses at various layers in the network. This approach will mitigate the impact of sophisticated attacks in the most efficient manner. The response selection will be preemptive as well as adaptive to the ongoing intrusion.

**Keywords**—Cross-layer Approach, Intrusion Detection, Intrusion Response

## I. INTRODUCTION

With the rapid increase of the Internet, networked computer systems are playing vital roles in our modern society. As people increasingly rely on the Internet, it has also made systems vulnerable to malicious attacks [1]. This leads to continued advancement in the state of the art of cyber security controls. However, similar to physical arms races, cyber attackers also continue to evolve in an attempt to remain effective at violating those controls. Therefore, modern attacks like coordinated attacks are becoming increasingly popular. Such attacks are difficult to detect and effectively defend from.

Early intrusion detection system (IDS) models were built to monitor activities of a single host [2], [3]. Later approaches accommodated multiple hosts of a network [4], [5]. More recent approaches [6], [7] pay more attention to large scale distributed attacks. An important research problem these systems address is how to detect coordinated attacks (e.g. Internet worm) over large distributed systems.

Even though significant progress has been achieved by the new systems [6], [7], additional research is needed to develop more practical approaches which will not only detect sophisticated attacks, such as distributed denial-of-service (DDoS), but also select and deploy a response preemptively with the selection being adaptive to the ongoing intrusion. In this paper we propose system named Cross-Layer Approach for Intrusion Detection and Response (XIDR), which emphasizes

detecting sophisticated attacks over a large scale environment and responds against such attacks to mitigate effects of the intrusion. The cross-layer approach will help us to detect signature-based and anomaly-based attacks with reduced false positive and false negative rates. This can be done using correlating alerts. Since we are using multiple detection sources for various layers, sources that inspect network layer properties of incoming packets could flag particular packets if they originate from known untrusted IP address ranges. This information could later be used by application-layer sources to issue alerts with greater confidence. Also, a cross-layer approach will help stop attacks by deploying a response at all necessary layers, with a minimal total cost across layers.

Our major contributions lie in the following:

- Multi-source intrusion detection systems (IDS) (e.g. host based, network based) providing cross layer based detection of malicious activity.
- Cross-layer automated intrusion response system (IRS) to deploy cost effective and efficient preemptive responses. Response selection will be preemptive as well as adaptive to the ongoing intrusion.
- Integration of IDS and IRS to develop a complete framework.
- Test-bed implementation for verification and performance evaluation.

The rest of the paper is organized as follows. Section II provides a brief survey of related work. Section III provides the motivation and overview of our work. Section IV will describe the design architecture of our test-bed in detail. Section V explains our simulation and performance results. Finally, we present conclusions and future work in section VI.

## II. RELATED WORK

Intrusion detection has been at the center of intense research for at least 20 years. It is repeatedly argued that layered architectures have served well for wired networks, but are not suitable for wireless networks. Therefore, cross layer design protocols have been introduced. This type of design exploits the dependence between protocol layers to obtain performance gains. Lee et al. [8] proposed a statistical anomaly detection at multiple layers and integration into a cross-layer defense system. The design of an IDS has been done according to six basic

components: *data gathering, local detection, security communication, cooperative detection, local response, and global response*. While Thamilrasu et al. [9] discussed the motivation and limitations of cross-layer techniques for security, applied to network-based intrusion detection, cross-layer approaches have been shown to outperform single layer detection both in detection rate and in reduced false positives [10], [11], [12]. In particular, previous work by our group [12] explored the application of intrusion detection to both the MAC layer and the network layer of the TCP/IP model and found that cross layer intrusion detection was more effective (6 to 10 percent) than network layer only detection.

While intrusion detection and response have been extensively studied, approaches that attempt to learn models based on data across different layers of interaction are relatively unexplored outside of the domain of wireless networks. In this work, we expand upon ideas employed in current intrusion detection and will use information from multiple layers for attack detection and automated real time response.

### III. BACKGROUND

#### A. Cross-Layer Intrusion Detection

For detecting and responding against sophisticated attacks, as opposed to focusing independent efforts on different layers of the TCP/IP stack, our aim is to integrate features from various layers for a more accurate detector with fewer false alarms. This allows detection of a broader range of attacks, e.g., those with characteristic signatures across multiple layers of interaction simultaneously. For example, the use of stolen SSH credentials may be undetectable from the application layer alone, but anomalies in the network layer (unusual source location) or the time of day (unusual human-computer interaction) could reveal such activity. Cross-layer intrusion detection also provides more information to the system administrator, permitting a greater degree of accuracy and more thorough cross-checking to reduce false positives.

#### B. Cross-Layer Intrusion Response

Nowadays, when malicious activity is identified, automated intrusion response is performed using cost sensitive response selection [13], [14], [15], [16], [17]. Our recent work [18] showed that the use of a novel metric for response cost assessment significantly reduced the impact of responding to intrusions using simulated data from the DARPA data set.

In order to effectively stop cross layer attacks with minimal cost, the responder must be able to determine the interaction layers that are being utilized by an attacker. This allows the response to be deployed at all necessary layers, with a minimal total cost across layers. Selecting an effective response first requires that the set of potentially effective responses can be constructed. Our focus starts with the analysis of attack behavior to determine the set of appropriate responses layers. This will not only allow the response selection mechanism to select the appropriate set of candidate responses to evaluate, but also to determine cases where sets of responses are more effective than single layer responses. For instance, two valid

responses to a brute force password attempt may be to block the IP addresses from which attempts are occurring (network layer), or to disable the user account (application layer). If the attempts are coming from a small set of IP addresses, the network layer response will prevent the attack while still allowing the legitimate user to login from their location. However, if the attack is widely distributed, it may be more cost effective to disable the user account until the cause of the targeted attack can be determined.

### IV. APPROACH

Figure 1 demonstrates an overview of our model. The basic components include multiple intrusion detection sources, data sources, automated response selection engine, and a collection of response deployment modules.

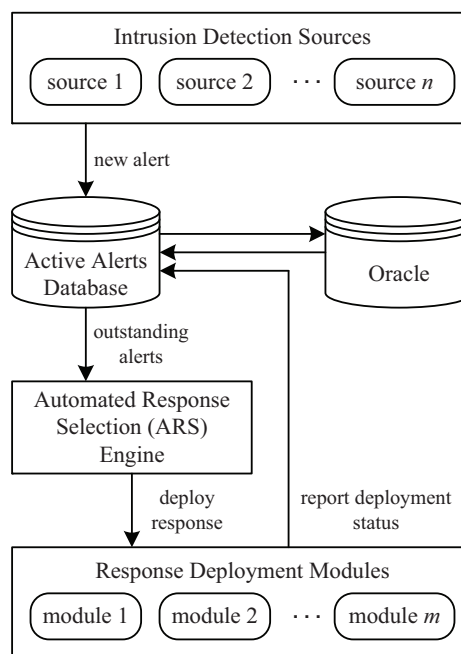


Fig. 1. Architecture diagram

#### A. Intrusion Detection Sources

A single intrusion detection source is often not enough in order to effectively detect coordinated attacks which exploit multiple layers of the TCP/IP model. While some intrusion detection sources, such as snort [19], are able to match incoming packets on multiple layers, they typically do not offer fine-grained detection mechanisms as applicable to a particular protocol. Moreover, generic pattern-matching algorithms may require comparatively high processing time if network traffic level is high, which leads to longer delays in servicing users' requests. Attempts to improve performance for detection of certain classes of intrusions have been made by projects such as fwsnort [20], which aim to reduce overhead by moving intrusion detection from user space applications into kernel space [21]. This is accomplished by translating snort rules into

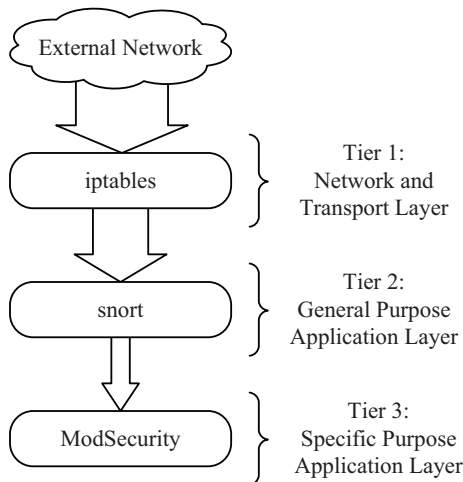


Fig. 2. Intrusion Detection Sources arranged in cascading manner. This approach allows application-layer sources running more sophisticated algorithms to process only a fraction of total traffic.

iptables rules. However, since the kernel often does not provide extended functionality needed for effective intrusion detection, such as regular expressions support, some rules cannot be moved into kernel space. This results in some types of attacks being undetected if only kernel-space facilities are used.

Our model addresses these concerns by utilizing multiple intrusion detection sources. This approach offers more granularity in packet inspection, which results in reduced false positive rate. Additionally, it improves performance by using cascading style detection (shown in Figure 2).

### B. Data Stores

Our model uses two data stores for short (active alert database) and long term (oracle) information storage. All current intrusion-related information is located in the active alerts database, which is regularly updated by intrusion detection sources as new alerts are generated. Selected properties that are associated with an alert include:

- *Detection source* that reported the alert, which often implies the layer or protocol in which the attack takes place.
- *Status* flag that indicates whether this alert has been responded to, and success status of the response if it is available.
- *Weight* indicating the probability that the alert corresponds to a real intrusion attempt, rather than being a false positive.

As mentioned before, each intrusion alert has a weight associated with it, which serves as a multiplier to the quantified damage potentially caused by the intrusion. Some factors may indicate that an alert generated by a detection source may be a false positive. For example, consider a scenario when XIDR implementation is employed to protect an e-commerce website. Information about current customers and hosts that they use to access the website may be stored in a database. It

is unlikely that an intrusion will originate from one of these hosts; therefore, an alert corresponding to one of these hosts that can possibly be generated by a detection source may be given less priority by decreasing its weight. On the other hand, we may want to increase the weight of alerts generated by the packets coming from known untrusted addresses. A local business serving a municipality may not expect international customers; hence, foreign IP address ranges may trigger an increase in the alert's weight. This information about IP ranges is stored in the oracle and consulted periodically by the service that updates alert records.

The oracle serves as a knowledge base for storing long-term information needed to augment alert record in short-term storage with additional information. Some records stored in the oracle include the following:

- *IP address ranges* used to modify the weight of the alerts based on the originating IP address.
- *Rule sets of intrusion detection sources.* Alerts reported by the sources in real time often lack extended information; thus, additional information, such as rule descriptions, may need to be extracted from the rule sets to facilitate mapping of source-specific identifiers to IDs used by the automated response selection engine.
- *Correlation data*, to be used with data mining techniques in order to aggregate multiple alerts from different sources into a composite intrusion.

### C. Automated Response Selection Engine

Our implementation uses an extension of the cost-based response selection engine proposed in [22]. We define a system as a collection of components listed below, together with confidentiality, integrity, and availability values associated with each component.

- *Resources:* components on the network to be protected from attacks. The notion of a resource is abstract, as it can identify a specific computer, a role performed by one or more machines (e.g. a web site provided by a server farm), a process running on a particular host, or any service in general. Therefore, the response selection engine can operate both within the scope of a single computer, and within the scope of an entire network.
- *Intrusions:* known attack vectors that can inflict damage to a set of resources. The intrusions specified for the response selection engine can either directly map to alerts generated by detection sources, or represent an intrusion generated by aggregating and analyzing multiple alerts.
- *Responses:* a set of actions to be taken in order to mitigate an intrusion. The least-cost *response* is selected when a known *intrusion* affects a protected *resource*.

The response selection engine is layer-agnostic, as all intrusions and responses reside in a flat space. Hence, the operational cost metric of each response is used to indicate the severity of a response. This way, lower-layer responses are assigned higher operational cost than those that operate on the application layer.

#### D. Response Deployment Modules

After the appropriate response is selected by the engine, a response deployment module that consists of custom scripts and runs as a service on each protected host is invoked. In order to ensure that unauthorized entities cannot send control commands to the hosts, we utilize public key infrastructure by encrypting all messages sent to the modules. One of the main advantages of cross-layer approach to response deployment is decrease of damage from potential false positives. If the cost of a radical response, such as blocking source address on network layer, is too high, a weaker response could be deployed on application layer that restricts actions that the client can do rather than completely blocking access to a resource. After a response has been deployed, success or failure code is being reported back to the active alerts database.

### V. EVALUATION

#### A. Attack Scenario

To evaluate the effectiveness of our approach, we considered a typical scenario in which an intrusion detection system might be used. In our scenario, a company provides a web site where orders may be placed by clients. To place an order, a client must first log in to the web site. On the back-end, the web server contacts a database that contains customers' orders and displays relevant information about the orders on the web site.

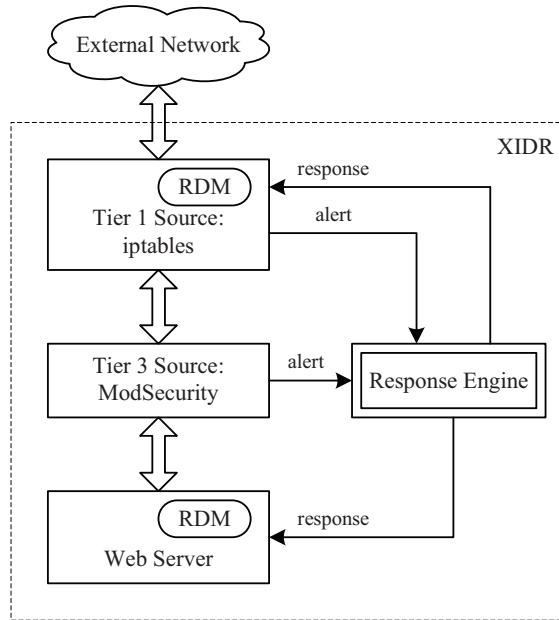


Fig. 3. Cross-layer IDS

Figure 3 shows a brief diagram of XIDR deployed in our test-bed environment. We can consider this framework a transparent proxy that resides between the external network and the web server. A web server is typically a resource which is responsible for providing multiple services to clients. An end user/client uses the external network to access such

services. To launch a web-based attack, the attacker will use the same external network to compromise the services hosted on the server. Therefore, it is reasonable to deploy our framework as a transparent layer between the web server and the external network. Whenever a network packet arrives carrying requests for the web server, it has to go through our iptables detection source to detect network layer intrusions. If an alert is generated by the source, a default response will be selected and deployed by XIDR. The framework will also store information like IP address and damage effect as a weight for future reference in our data sources. We refer to this information as *history*.

As mentioned previously, to protect resources like web servers from web-based attacks, such as SQL injection, packets arriving on TCP port 80 are required to be inspected; for that purpose, ModSecurity has been used as a detection source. Now, as we know, detecting SQL injection attacks in real time will give a high false positive rate and a high false negative rate; therefore, using ModSecurity or iptables alone will not be sufficient. Hence, before selecting and deploying response against the alerts generated by ModSecurity, our framework takes into account the history associated with the source IP address of each request that is stored in our data sources to measure the severity of the alert and then deploys the response at a particular layer using our cost-sensitive response selection model to curtail the effect of the intrusion. In particular, we consider that users had positive history when they successfully logged in to the web site, and users had negative history when they attempted to launch a port scan or had multiple successive failed attempts to authenticate.

#### B. Classification of Rules

Both approaches were tested against SQL injection attacks, which are a common threat to web application and web services. We used rule-based intrusion detection sources, such as snort and ModSecurity, to match requests against the rules. In our evaluation, the rules are broadly classified in two categories:

- 1) *Specific rules.* These rules are characterized by targeting a specific exploit or matching a specific keyword; therefore, the probability of false positives is low. However, only a small fraction of SQL injection attacks could be matched by these rules.
- 2) *Generic rules.* SQL injection strings could be easily modified to avoid most specific rules. For instance, most application-layer detection sources, such as snort or ModSecurity, include a regular expression of the form "' OR .\*=>1" that matches simple common injection strings. This rule could be easily avoided by forming a variation of this expression, e.g., "' OR 2>1". Therefore, generic rules could be employed to match strings containing single apostrophes or other indicators of SQL injection strings. Because of their generic nature, these patterns are likely to match a significant number of legitimate requests, leading to a high false positive rate.

Given the characteristics of the above types, the following alert-response maps have been established for both models:

- Triggering a specific rule leads to a strong response: blocking the source IP address.
- Triggering a generic rule leads to a weak response: logging the alert and allowing the request to proceed.

We based the evaluation of our approach on comparing the number of legitimate requests that are blocked when they trigger a specific rule (false positives) and the number of malicious requests that are allowed when a generic rule is triggered (false negatives). We assume that generic rules meet all necessary conditions to match any injection string.

### C. Cross-Layer Detection

In our implementation, we reduce the rate of errors by utilizing information gathered from other layers and protocols related to a request. This is done by augmenting the weight of each alert based on the history related to the originating host. We make the following assumptions:

- If the user’s previous history of interaction with our resources is *positive* and an alert is triggered with the user as the source, we *decrease the weight*, or significance, of this alert. Examples of positive history include successful authentication to the services that we provide (e.g. logging in to the web site), or extended period of activity without any alerts triggered.
- We *increase the weight* of each alert if the user has a *negative* history. Negative history includes attempts to probe network services by doing port scans, previously triggered alerts, or multiple unsuccessful attempts to authenticate. As most attackers are likely to gather preliminary information about the system by performing port scans, or attempt to gain access by other means (e.g. dictionary password attack), we can be more confident about issuing a stronger response to subsequent alerts.

Composing a comprehensive history of each host by inspecting single layer (in our scenario, HTTP) could be non-feasible. Looking at other layers and protocols provides us with additional information needed to classify the host.

### D. Simulation

Given the lack of availability of datasets for evaluating SQL injection attack, we simulated incoming traffic by developing a request simulator. The set of all requests is partitioned as shown in Figure 4, with particular requests of interest emphasized. An incoming request that is not triggered by any rule could either have positive or negative history associated with the source address. In case the request contains no previous event, we discard it as irrelevant in our evaluation. We also consider malicious packets that match a specific rule, and safe packets that match a generic rule irrelevant to measure, since the default response for these packets is appropriate.

Our model offers a reduction in false positive and false negative rate as compared to the single-layer approach in two cases. When a safe request with positive history triggers a

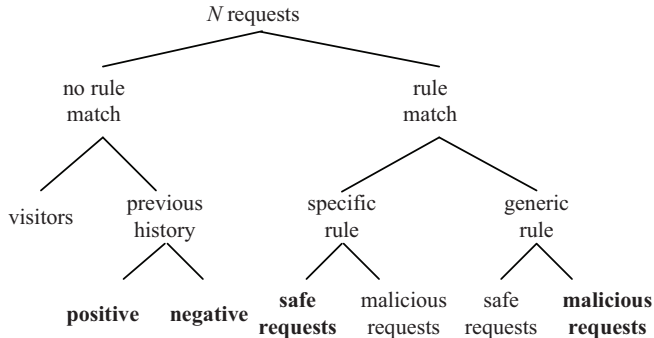


Fig. 4. Classification of incoming requests to a web server

specific rule, we select a weak response instead of denying access to a legitimate user. On the other hand, when a malicious packet with negative history triggers a generic rule, we block it rather than allowing it to proceed. The traditional single-layer approach would not be able to determine these requests; therefore, an incorrect default action would be taken.

### E. Results

We evaluated the effectiveness of our approach against the traditional use of single-layer intrusion detection systems while responding to SQL injection attacks by measuring the decrease in the fraction of false positives ( $R_{FP}$ ) and false negatives ( $R_{FN}$ ), defined as

$$R_{FP} = 1 - \frac{FP_{CL}}{FP_{SL}} \text{ and } R_{FN} = 1 - \frac{FN_{CL}}{FN_{SL}},$$

where  $FP_{CL}$  and  $FN_{CL}$  is the number of false positives and false negatives produced by cross-layer approach (XIDR), and  $FP_{SL}$  and  $FN_{SL}$  is the number of false positives and false negatives produced by single-layer IDS. We also denote the fraction of packets that have prior history by  $H$ .

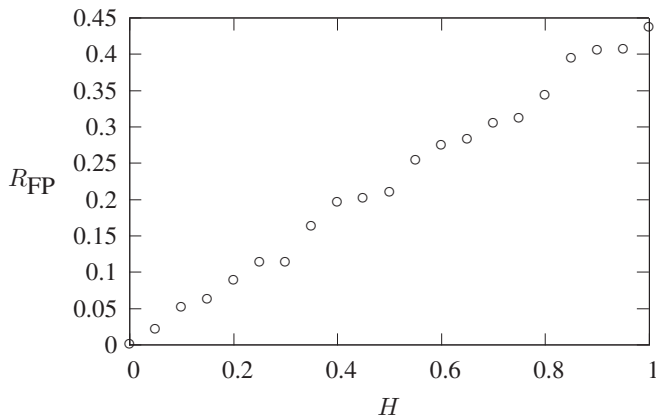


Fig. 5. Reduction in false positive rate

For our experimental evaluation, we have generated 100,000 data packets while varying the number of users from 5 to

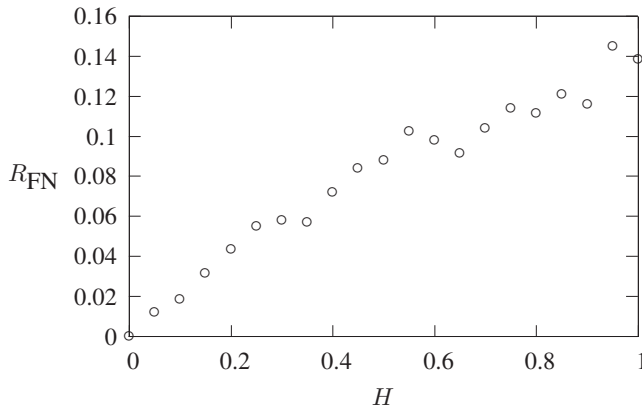


Fig. 6. Reduction in false negative rate

10. Figures 5 and 6 show our simulation results. When  $H = 0$ , our model behaves the same as the traditional single-layer approach. However, as we take advantage of cross-layer detection by inspecting other protocols and building history for each host, we increase the number of packets which trigger the correct response, despite having ineffective default response. From Figures 5 and 6 we can see that, while the history increases, the false positive rate can be reduced by approximately 44 percent and false negative rate by 14 percent. This shows the efficiency of cross-layer approach over the single layer approach in detecting intrusions and also helps us to deploy correct responses against those intrusions.

## VI. CONCLUSION AND FUTURE WORK

In this paper we have presented a framework that consists of multiple intrusion detection sources to utilize cross-layer based detection in a wired environment and deploy a response across various layers using a cost-sensitive response selection mechanism to minimize the cost of intrusion. In our future work, we intend to investigate the following:

- Event Correlation to investigate and identify layers of interaction between users (e.g. user to user, machine to machine, and machine to user). This technique will also help us to reduce the blow-up in our data sources due to the high volume of generated alerts.
- Extension of Cost-Sensitive Response Selection by response duration, response, and intrusion impact.
- A log of intrusion alerts and corresponding deployed responses could be used in the forensic analysis to identify the impact of intrusions and the effectiveness of deployed responses.

Our current work has shown a particular intrusion scenario. In the future, we will extend our approach to provide real-time detection and response in a distributed environment. The experiments we have conducted so far provide a solid foundation to implement a real-time detection and response framework.

## REFERENCES

- [1] A. Ghosh, J. Wanken, and F. Charron, "Detecting anomalous and unknown intrusions against programs," in *Proceedings of 14th Annual Computer Security Conference*, 1998, pp. 259–267.
- [2] S. Smaha, "Haystack: an intrusion detection system," in *Proceedings of the IEEE 4th Aerospace Computer Security Applications conference*, 1988, pp. 37–44.
- [3] H. Javitz and A. Valdez, "The sri ides statistical anomaly detector," in *Proceedings of the IEEE symposium on Security and Privacy*, 1991, pp. 316–326.
- [4] T. Heberlein and et al, "A network security monitor," in *Proceedings of the IEEE symposium on Security and Privacy*, 1990, pp. 296–304.
- [5] S. Snapp and e. a. J. Brentano, "Dids(distributed intrusion detection system)- motivation, architecture, and an early prototype," in *Proceedings of the 14th National Computer Security Conference*, 1991, pp. 167–176.
- [6] P. Neumann and P. Porras, "Emerald: Event monitoring enabling responses to anomalous live disturbances," in *Proceedings of National Information Systems Security Conference*, 1997, pp. 353–365.
- [7] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle, "Grids - a graph-based intrusion detection system for large networks," in *Proceedings of the National Information Systems Security Conference*, 1996, pp. 361–370.
- [8] Y. Zhang and W. Lee, "Intrusion detection in wireless ad-hoc networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking*, 2000, pp. 275–283.
- [9] G. Thamarasu and R. Sridhar, in *Exploring Cross-layer Techniques for Security Challenges and Opportunities in Wireless Networks*, 2007, pp. 275–283.
- [10] J. Parker, A. Patwardhan, and A. Joshi, "Cross-layer analysis for detecting wireless misbehaviour," in *IEEE Consumer Communications and Networking Conference Special Sessions*, January 2006, pp. 6–9.
- [11] G. Thamarasu, A. Balasubramanian, S. Mishra, and R. Sridhar, "A cross-layer based intrusion detection approach for wireless ad hoc networks," 2005, p. 861.
- [12] X. Wang, J. S. Wong, F. Stanley, and S. Basu, "Cross-layer based anomaly detection in wireless mesh networks," in *Proceedings of the International Symposium on Applications and the Internet*, 2009, pp. 9–15.
- [13] I. Balepin, S. Maltsev, J. Rowe, and K. Levitt, "Using specification-based intrusion detection for automated response," in *Proceedings of the International Symposium on Recent Advances in Intrusion Detection*. Springer, 2003, pp. 136–154.
- [14] B. Foo, Y.-S. Wu, Y.-C. Mao, S. Bagchi, and E. H. Spafford, "ADEPTS: Adaptive intrusion response using attack graphs in an e-commerce environment," in *Proceedings of the International Conference on Dependable Systems and Networks*, 2005, pp. 508–517.
- [15] M. Jahnke, C. Thul, and P. Martini, "Graph based metrics for intrusion response measures in computer networks," in *Proceedings of the IEEE Local Computer Networks*, 2007, pp. 1035–1042.
- [16] W. Lee, W. Fan, M. Miller, S. J. Stolfo, and E. Zadok, "Toward cost-sensitive modeling for intrusion detection and response," *J. Comput. Secur.*, vol. 10, no. 1-2, pp. 5–22, 2002.
- [17] T. Toth and C. Kruegel, "Evaluating the impact of automated intrusion response mechanisms," in *Proceedings of the Annual Computer Security Applications Conference*, 2002, pp. 301–310.
- [18] C. Strasburg, N. Stakhanova, S. Basu, and J. S. Wong, "Intrusion response cost assessment methodology," in *Proceedings of the ACM Symposium on Information, Computer and Communications Security*, 2009, pp. 388–391.
- [19] W. Metcalf and V. Julien, "Snort\_inline," World Wide Web electronic publication, 2009. [Online]. Available: <http://snort-inline.sourceforge.net/>
- [20] M. Rash, "fwsnort: Application layer ids/ips with iptables," World Wide Web electronic publication, 2009. [Online]. Available: <http://www.cipherdyne.org/fwsnort/>
- [21] —, *Linux firewalls*, 2007.
- [22] C. Strasburg, N. Stakhanova, S. Basu, and J. Wong, "A framework for cost sensitive assessment of intrusion response selections," in *Proceedings of IEEE Computer Software and Applications Conference, COMPSAC*, 2009, pp. 355–360.