This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2009 proceedings

Policy-Based Security Configuration Management Application to Intrusion Detection and Prevention

Khalid Alsubhi*, Issam Aib*, Jérôme François‡ and Raouf Boutaba*
*David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada
‡MADYNES - INRIA Lorraine, CNRS, Nancy, France
email: kaalsubh@cs.uwaterloo.ca ; (iaib,rboutaba)@uwaterloo.ca; jerome.francois@loria.fr

***Abstract*—Intrusion Detection and/or Prevention Systems (IDPS) represent an important line of defense against the variety of attacks that can compromise the security and well functioning of an enterprise information system. IDPSes can be network or host-based and can collaborate in order to provide better detections of malicious traffic. Although several IDPS systems have been proposed, their appropriate configuration and control for effective detection and prevention of attacks has always been far from trivial. Another concern is related to the slowing down of system performance when maximum security is applied, hence the need to trade off between security enforcement levels and the performance and usability of an enterprise information system.**

In this paper we motivate the need for and present a policy-based framework for the configuration and control of the security enforcement mechanisms of an enterprise information system. The approach is based on dynamic adaptation of security measures based on the assessment of system vulnerability and threat prediction and provides several levels of attack containment. As an application, we have implemented a dynamic policy-based adaptation mechanism between the Snort signature-based IDPS and the light weight anomaly-based *FireCollaborator* IDS. Experiments conducted over the DARPA 2000 and 1999 intrusion detection evaluation datasets show the viability of our framework.

***Index Terms*—Security management policies, Security Configuration, Risk Management, Alert Management.**

I. Introduction

With the dominant use of networked information systems in modern enterprises, the management of their health and security becomes of vital importance. Security threats come into a variety of shapes and new attacks are crafted on a daily basis from a variety of sources and for different reasons ranging from competitor planned intrusions to amateur explorations.

Among other security enforcement tools and mechanisms, Intrusion Detection and/or Prevention Systems (IDPS) [1] represent an important line of defense against the variety of attacks that can compromise the security and well functioning of an enterprise information system. IDPSes can be signature-based or anomaly-based. Signature-based IDPSes, such as Snort [2] and Bro [3], are the most dominant and are based on the pre-knowledge of attack signatures which help identify malicious traffics from benign ones. Anomaly-based IDPSes work differently in that they learn about the normal behavior of a system and then raise alerts whenever an abnormal behavior is detected. IDPSes can be network or host-based and can collaborate into centralized or distributed clusters in order to provide better detections of malicious traffic across a distributed networked system.

Although many IDPS systems have been proposed, their appropriate configuration and control for effective detection and prevention of attacks has always been far from trivial [4]. Another concern is related to the significant slowing down of system performance when maximum security is applied [5], [6]; hence arises the need to tradeoff between security enforcement levels on one side and the performance and usability of an enterprise information system on the other.

In this paper we motivate the need for and present a framework for policy-based [7] configuration and control of the security enforcement mechanisms of an enterprise information system. The approach is based on the dynamic adaptation of security measures based on the assessment of system vulnerability and threat prediction and provides several levels of attack containment. As an application, we have implemented a dynamic policy-based adaptation mechanism between the Snort signature-based IDPS and the *FireCollaborator* [8] light-weight anomaly-based IDS [8]. Experiments conducted over the DARPA 2000 LLDOS 1.0 and 1999 intrusion detection evaluation datasets show the viability of our framework.

The paper proceeds with an overview of related work then follows by a presentation of the policy-based security management framework. In section IV-A, our *FireCollaborator* anomaly-based IDS is briefly presented. Section V presents the experiments we have conducted using dynamic adaptation policies, the *FireCollaborator*, and Snort. Section VI discusses the challenges related to policy-based security adaptation and finally section VII concludes the paper with an insight on future work.

II. Related Work

Using policies to drive security management has been mentioned in some previous work. However, none address the problem from the dynamic adaptation for the sake of balancing system performance and security. In addition, the adaptation mechanism in the provided use case is unique per see.

Authors of [9] seek to transform an IDS system into an IPS by proposing a policy management for firewall devices integrated with intrusion prevention capabilities. They propose an attack response matrix model which maps intrusion types

This research was partially supported by WCU (World Class University) program through the Korea Science and Engineering Foundation funded by the Ministry of Education, Science and Technology (Project No. R31-2008-000-10100-0) and partially supported by the Natural Science and Engineering Council of Canada (NSERC).

978-1-4244-3435-0/09/$25.00 ©2009 IEEE

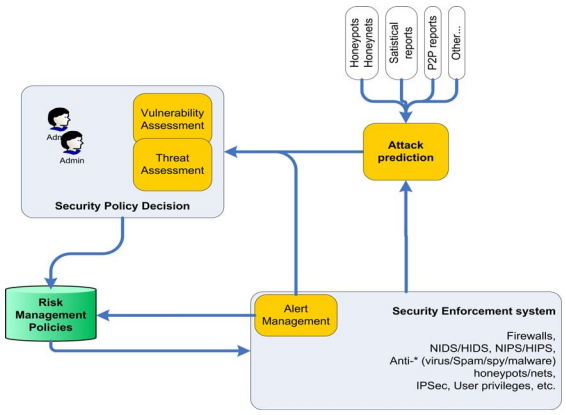


Fig. 1. Policy-based Risk Management Framework.

to traffic enforcement actions. Their proposal is however only at the design level and no concrete implementation or policy specifications have been provided. In addition, they do not consider the performance aspect but only how to transform an IDS into an IPS using policies. In [10], Tanachaiwiwat et. al. propose a method to represent the reports of different IDSes into a matrix. These metrics give the number of triggered alarms depending of the attack but also the number of false positives. This is followed by a risk assessment stage, where the possible cost of an attack and of the corresponding countermeasure is derived. Teo and Ahn [11] propose a policy framework called Chameleos-x which is design to enforce security policies on different kinds of equipment. A policy is entered in the console components and each device uses the enforcement monitor to enforce the policy. To be applicable, the policy should be adapted to the device through a translator component. Finally, the POSITIF project [12] provides an initial method of cooperation and integration between policy specification and anomaly detection systems (such as IDS). The desired behavior of the information security system (i.e. IDS) can be implicitly achieved by the policy specification. Their work has not yet been implemented into a real system.

III. Policy-based Security Management Framework

The framework we propose uses system administrator policies to balance the security measures employed by the enterprise information system. As depicted in figure 1, the adaptive management of the enterprise security is provided at the low level through the repository of risk management policies. This repository is maintained by the security administrator(s) based on previous expertise as well as the input from deployed threat prediction tools.

Threat prediction tools assess the degree of vulnerability of the enterprise system with regard to a range of potential attacks. They sense network traffic with the external world, combine that with information gathered from external threat measurement sources, and finally generate reports on the risks that are likely to threat the information system in the near future. Threat detection sources can be of different kinds, internal and external. Honeypots [13] represent an effective means to detect and learn about security threats. Statistical reports and peer security alerts from trusted parties represent another valuable source of information. For instance, in this paper we use the *FireCollaborator* [8] as an alert source for Distributed Denial of Service (DDoS) attacks. The *FireCollaborator* system forms rings of protection around a subscribed customer and lies outside the premises of the enterprise network.

The overall assessment of the system security and potential forthcoming threats is translated down as risk management policies (figure 1). These policies adapt the behavior of the underlying security measures accordingly. The translation is done by the threat assessment and security policy decision component. As it is not trivial to automatically assess and decide about which security policy to adopt, human administrators are expected to be involved in this process.

The underlying security enforcement system comprises all those measures, tools, and devices that collectively participate in implementing the overall security policy. This includes the dynamic (re)configuration of host-based and network-based intrusion detection and prevention systems (IDPSes), firewalls, proxies, application server security components, activity reporting and event analysis components, etc.

IV. FireCollaborator

A. Description

The *FireCollaborator* system [8] aims to detect DDoS attacks as far as possible from the victim and as close as possible to the attack source. It can be used both as an IDS or IPS. As depicted in figure 2, *FireCollaborator* instances can be distributed over ISP routers forming rings of protection around the customer.

A list of n rules $R_1, R_2, ...R_n$ describing packet patterns is input to the *FireCollaborator*. After that, simple metrics related to rule matching are maintained. One metric is the frequency f_i of a rule R_i (Eq. 1) during a predefined detection window dw. It is computed by dividing the number of packets matching this rule F_i by the total number of packets.

Another metric is the entropy (Eq. 2) which is an indicator of the diversity of traffic. For instance, the entropy is 1 when the distribution is uniform.

The last metric is the relative entropy metric (Eq. 4) which permits to compare easily if the current distribution f is close to the profile f' (higher than a certain threshold ω). In the first step of the detection mechanism, *FireCollaborator* determines whether the traffic has changed before continuing, otherwise it indicates that there is no attack because it would have be detected before. Therefore, the detection process continues only if the relative entropy is higher than a threshold ω.

$$f_i = \frac{F_i}{\sum_{j=1}^{n} F_j} \quad (1)$$

$$H = -E[log f_i] = -\sum_{i=1}^{n} f(i) log_n(f_i) \text{ , (entropy)} \quad (2)$$

$$\psi_i = log \frac{f_i}{f_i'} \quad (3)$$

$$K(f, f') = \sum_{i=1}^{n} f_i \ \psi_i \text{ , (relative entropy)} \quad (4)$$

$$\frac{f_i(t)}{f_i(profile)} > 1 + \gamma, \ 0 \leq \gamma \leq 1, f_i(t) > \epsilon \quad (5)$$

$$Score_i = f_i \times b_k \text{ , (confidence score)} \quad (6)$$

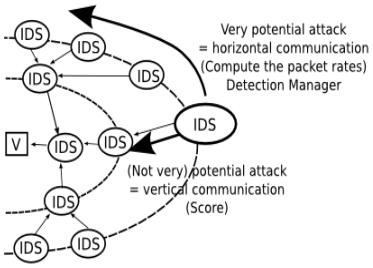


Fig. 2. Sample simulation topology

Based on the use of the decision table I, a confidence score for each pre-selected rule is derived (Eqs. 5, 6.) Basically, a high frequency means a possible DDoS attack. However, this detection rule cannot be used because the frequency of rule depends on the other one. Thus *FireCollaborator* considers the entropy also. More detail about the decision table are given in [8].

After that, the current score, the previous score affected by an ageing factor a, and the score from upstream *FireCollaborator* instances are combined in order to obtain the effective score of a rule. If the score is higher than a certain threshold τ, a horizontal communication (figure 2) between the *FireCollaborator* instances on the ring structure is launched to compute the overall data rate and compare it with the capacity of the client. In the other case, the decision is delegated by sending the score to the next upstream *FireCollaborator* instance and so on [8]. Here we did not considered a ring architecture and we used *FireCollaborator* on a single host instead.

B. Complexity

Although the *FireCollaborator* have many stages during the detection process, the operations are simple. For example, we can count mathematical operations in the case of all stages are needed. Considering that the frequency and entropy are computed only at the end of the detection window dw, *FireCollaborator* has just to count the number of packets for each rules. Assuming p_{dw} is the number of packets during dw, there are p_{dw} operations to increment the counter. To compute the frequencies at the end, *FireCollaborator* has to sum all counters (*i.e.*, n operations where n is the number of rules.) Then we have the following steps:

- Compute the frequency of each rule (Eq. 1): n operations
- Compute the entropy (Eq. 2): $2n$ operations
- Compute the relative entropy (Eq. 4): $2n$ operations + 1 comparison
- Extract suspect n_2 rules (Eq. 5): 1 operations
- Examine extracted rules (Table I): n_2 + 1
- Compute the score for the case 1,2,3 of the decision table (n_3 rules) (Eq. 6): n_3 operations
- Compute the confidence level: $2n_3$ operations

TABLE I
FireCollaborator DECISION TABLE

Case	Entropy	Frequency	Conclusion	Score factor
1	High	High	Potential	b_1
2	Low	High	Medium threat	b_2
3	High	Low	Potential later	b_3
4	Low	Low	No potential	$b_4 = 0$

The main question is about the values of n_2 and n_3. We have $n \geq n_2 \geq n_3$ because few rules are kept at each stage for further analysis. It can be formalized by using an exponential distribution *i.e.*, $n_2 = n \times p(\frac{f_i(t)}{f_i(profile)} > 1 + \gamma) = e^{-\lambda(1+\gamma)}$ where γ is the parameter of the exponential law. Similarly, n_3 can be calculated. As a result, *FireCollaborator* has $O(n)$ complexity even if all the rules are selected *i.e.*, $n = n_2 = n_3$.

C. Parameters tuning

FireCollaborator has many parameters. Currently, the parameters are fixed by hand thanks to the previous experience of [8] and in order to detect the majority of the attacks. Normally, *FireCollaborator* should be run with a training set with corresponding to the environment where it is deployed. Furthermore, another could be to use the complexity evaluation to use the complexity formulas before to determine the fitted parameter value to have an acceptable value of operations. In fact, this is directly correlated to the number of triggered alerts by *FireCollaborator* which can be used also for determining the parameters during a training stage.

V. Experiments and Results

A. Settings

In order to test the validity of policy-based adaptation of security configuration while keeping the use case simple, we focus on DDoS attacks and define adaptation policies that help adapt Snort security level based on alert inputs from the *FireCollaborator* and from Snort itself. The policies are described in table II.

Snorts rules (signatures) are classified into classes. Each class contains a number of rules that are related to a known attack type. For example, the FTP class contains all rules related to attacks on FTP servers. Snort allows the enabling/disabling of rule classes (categories) or individual rules through a set of configuration files. Snort (v 2.8.0.2) has a set of 51 categories.

For the sake of our experiments, we defined three levels of detection capability for Snort. The *medium* detection level contains rules that are enabled by default when deploying Snort. The total of number of rules in this detection level is 8722 categorized into 32 libraries. The *minimum* detection level is defined by choosing the minimal set of rules that detect essential attack types. This level includes 11 libraries containing 4152 rules. The third level of detection is the *maximum* level, which includes all the libraries and has 9205 rules categorized into 51 libraries. As it is not trivial, section VI elaborates on the issue of choosing the appropriate set of categories for each detection level.

B. Scenarios and Policies

We conduct experiments using different Snort detection capabilities to scan the DARPA 2000 LLDoS 1.0 [14] and 1999 [15] datasets, which last for three hours and one week respectively. The policies used for dynamic adaptation as shown in table II.

As predicted in our previous work, relying only on a single instance of the *FireCollaborator* may entail many false alerts. In this experiment we use Snort as a confirming tool

TABLE II
POLICIES FOR DYNAMIC DETECTION CAPABILITIES

P_1	on FireCol alert a do increase security by one level when a.score = medium
P_2	on FireCol alert a do increase security by two levels when a.score = high
P_3	on Snort alert a do increase security by one level when a.score = high
P_4	on low security threat do switch to performance mode.
P_5	on DDoS attack do enforce DDoS mitigation, notify remote mirror servers, increase backup-policy level.

of the validity of a *FireCollaborator* alert. In our case, the *FireCollaborator* is deployed at a single location close to the victim.

We introduce in this paper a confidence level between zero and one associated to each *FireCollaborator* alert. The maximal score of a *FireCollaborator* alert is the maximal frequency multiplied by the maximal factor b_1. When the detection is triggered, this maximal score is affected by the ageing factor a. So we obtain:

$$max_{score} = max_{frequency} \times b_1 \times a \tag{7}$$

Considering the most aggressive attack with a constant frequency of 1 during all detection windows, the maximal possible score is:

$$max_{score}(0) = b_1 \tag{8}$$

$$max_{score}(i) = (max_{score}(i-1) \times a) + b_1 \tag{9}$$

When i tends to ∞, this term tends to

$$m = \frac{b_1}{1-a} \tag{10}$$

Hence, the confidence level of the score s is:

$$confidence = \frac{s-\tau}{m-\tau} \tag{11}$$

The dynamic security-level adaptation policies are listed in table II. For the purpose of this evaluation, the inputs to them are alerts from either the *FireCollaborator* or Snort. Whenever the *FireCollaborator* generates an alert with high confidence ($> 60\%$), Snort detection is upgraded by two levels (policy p_2). If the *FireCollaborator* generates an alert with a medium confidence ($> 40\%$), the detection level of Snort is increased by one level (policy p_1). Low confidence *FireCollaborator* alerts are ignored ($\leq 40\%$). For Snort, we only consider high-priority alerts by augmenting the detection level by one (policy p_3). Furthermore, we define a *caution window* cw for the medium and maximum detection levels. This window represents the period where Snort should stay in a particular detection level before it switches down to a lower level in case no abnormal behavior has been observed (policy p_4).

C. Results

Figure 3 represents the running time of Snort with three different detection levels (minimum, medium, and maximum) over a number of datasets. Each group of bars corresponds to a particular dataset and each bar represents the detection level

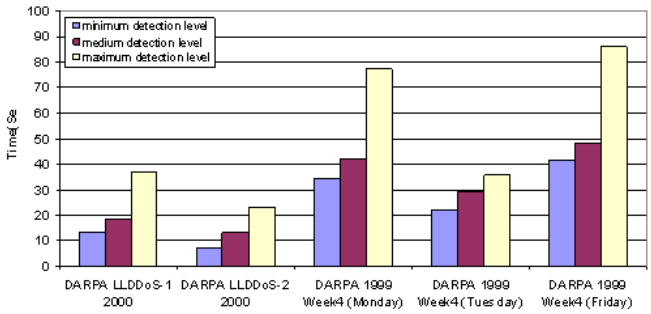


Fig. 3. Snort performance over multiple datasets using different Detection Levels

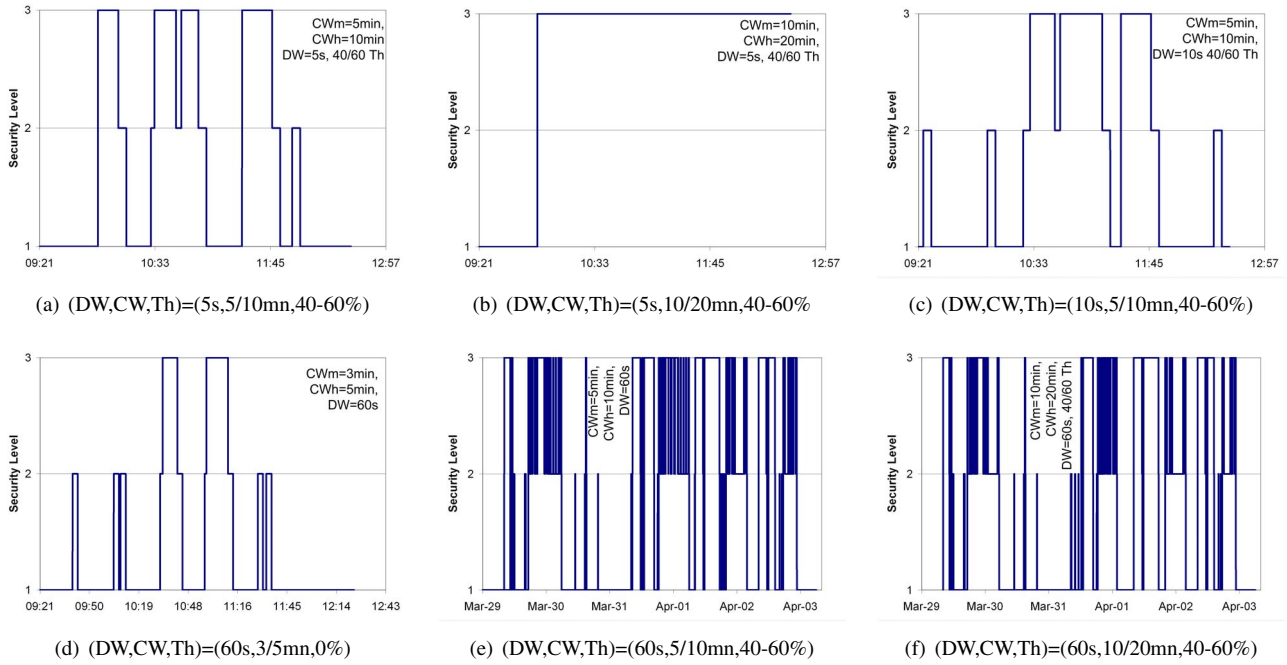
used in Snort to scan the dataset. Although, the medium and maximum detection levels of Snort include nearly the same number of rules (the medium level has 95% of all rules), Snort performance was different for these levels. This is because the default-disabled rule sets are computationally expensive and timely consuming. Since these results are conducted by scanning the tcpdump file of the datasets, Snort did not drop any packet due to unrestricted time limit. However, this may not be the case in a real time environment with high-rate links (i.e. gigabit) where some packets may be dropped to keep up with the rate. Dropping packets has the undesired outcome of the possibility of missing some attacks.

In the first experiment, we use the DARPA 2000 LLDOS 1.0 dataset [14] which contains traffic collected from two sensors installed in the DMZ and the Inside parts of the evaluation network. We focus our analysis on the inside sensor. The dataset features a series of attacks carried out over multiple sessions. These sessions start with a scanning the network in order to launch a DDOS attack against an off-site server. The sessions are grouped into five phases. First, the attacker starts by scanning the network (IPsweep) looking for any live IP addresses. Then, it looks for the sadmind daemon of live IP addresses. The attacker exploits the sadmind vulnerability in order to install an mstream Trojan, which is required for launching the DDoS attack. Finally, the DDoS attack is launched against an off-site server in case the attack is successful.

Using the *FireCollaborator*, and as shown in table III, the DDoS victim is identified to be 131.84.1.31. In this experiment the *FireCollaborator* was run with a detection window of one minute. The *FireCollaborator* manages to detect the attack with a high confidence level (67%). There are some other false positive alerts. However, most of them got

TABLE III
FireCollaborator ALERTS FOR THE DARPA 2000 DATASET ($dw = 60s$)

Time	Destination	Confidence
09:21:36	172.16.112.50	0.04
10:04:36	172.16.112.50	0.14
10:08:36	172.16.112.50	0.24
10:31:36	172.16.113.204	0.29
10:57:36	172.16.112.50	0.15
10:58:36	172.16.112.50	0.30
11:04:36	172.16.113.105	0.59
11:28:36	131.84.1.31	0.67
11:33:36	172.16.112.50	0.05



(a) (DW,CW,Th)=(5s,5/10mn,40-60%) (b) (DW,CW,Th)=(5s,10/20mn,40-60% (c) (DW,CW,Th)=(10s,5/10mn,40-60%)

(d) (DW,CW,Th)=(60s,3/5mn,0%) (e) (DW,CW,Th)=(60s,5/10mn,40-60%) (f) (DW,CW,Th)=(60s,10/20mn,40-60%)

Fig. 4. Selected Results of Dynamic Snort Detection level adaptation for DARPA 2000 and 1999 Datasets

filtered out using the 40-60% confidence thresholds (policies p_1 and p_2) and only one false alert for host 172.16.113.105 remained. This alert may be due to a traffic burst to which the *FireCollaborator* is sensitive.

The detection window size is an important parameter of the *FireCollaborator*. Big values of it will imply that some DDoS attacks will be detected too late, while very small values may results in increased false positives due to traffic profile fluctuation. Figure 5 shows the time and overhead in terms of number of operations induced by several detection window sizes for the same dataset. In all cases however, the overhead induced by the *FireCollaborator* is much less than the one due to the execution of Snort due to the difference of detection specialization. The delay and the number of operations are greatly reduced because there are more detection window slots. However, when the detection window size decreases, the traffic is more variable.

Therefore we have adapted the parameters to have always the same number of alerts but they can have a higher confidence.

Figure 4 shows the impact of dynamic security level adaptation policies on the behavior of Snort when scanning the

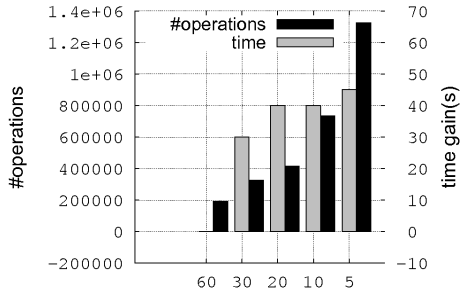


Fig. 5. Impact of Detection window size on the *FireCollaborator* overhead. Time reduction in seconds comparing with a 60 seconds detection window and number of mathematical operations

DARPA dataset with different caution and detection window sizes. Figure 4(a) shows the result for a caution window of $cw_m = 5$ and $cw_x = 10$ minutes for the medium and maximum security levels respectively. The first alert that changed the security level of Snort comes from the *FireCollaborator* at 09:57 with a confidence value of 0.8 (policy p_2). It is worth noting that the first phase of the attacks (IPsweep) did not change Snort detection level because the severity of all the alerts generated by Snort for this phase was medium. However, for all the other attack phases, except the forth phase which needs a host-based IDS to be detected, Snort was in its maximum detection level. Overall, Snort was running at maximum detection 28.66% of the time, at medium detection 13.16%. This implies that it was running at minimum detection 58.17% of the time while it still managed to detect the attack phases. In terms of the time necessary to analyze the dataset, Snort took 28% less time than it did if ran at maximum detection. This is good if we know that the DARPA 2000 dataset is only three hours long and contains a five phases attack.

In figure 4(c), we use the same caution window size as in 4(a) but with a larger window detection size. This results in having Snort run in maximum detection level in only three of the four detectable attack phases. In figure 4(d), we use smaller caution windows and a large window detection size of one minute, which results in reducing the overall time for the maximum and the medium detection levels and having Snort run at medium detection and not maximum detection during the DDoS attack.

Figure 4(b) represents the detection behavior of Snort with a large caution window of 10 minutes for the medium and 20 minutes for the maximum security levels. This results in Snort running in maximum detection level since the occurrence of the first high-level alert. This is because the high-level alerts coming either from Snort or the *FireCollaborator* make it so that the cw_x caution window never reaches an end. Finally, figures 4(e) and 4(f) show the dynamic security levels

when using the DARPA 1999 dataset. The results are quite interesting as in the many DDoS attacks figuring in this long dataset (one week), all those which last for more than around five seconds manage to get captured by the *FireCollaborator* and hence reported to Snort which adapt its security level accordingly.

VI. Discussion

The promises of policy-based dynamic adaptation of a security system so as to reduce security enforcement overhead while preserving good system performance are faced with the problem of defining the appropriate definition and configuration of policies in order to avoid an excessive exposure to real threats. In our experiments, configuration parameters, such as the caution windows of adaptation policies, the window detection size of the *FireCollaborator*, and score thresholds are not trivial to set. For the *FireCollaborator*, reducing the detection window size helps in the early detection of DDoS attacks but has the disadvantage of increased overhead and number of false alerts. However, in all cases, the overhead due to the *FireCollaborator* is always much lower than that entailed by Snort. In addition, depending on resource availability and attack prediction, the confidence thresholds may be lowered or increased by the system administrator. Using artificial intelligent techniques, such as neural networks or fuzzy logic, can be helpful in this configuration problem.

An additional concern which affects the accuracy of the detection is the selection of categories (set of rules) for each detection level as well as the number/type of security levels to define. The selection of categories may vary form an environment to another. For instance, the maximum detection level for protecting the web server of a company should include not only all the rules which detect web server specific attacks, but also those related to potential preliminary steps of these attacks, such as scanning. A possible solution for choosing the categories for each detection level can be based on common attack graphs [16] where the early steps of the attacks are included in the minimum detection level. However, the attacker may learn about the use of the caution window-based behavior and delay between attack steps accordingly. This can be countered by using a dynamic caution window size.

Another important point is related to the inherent support of dynamic adaptation by existing security mechanisms and tools. In this work, Snort was a hurdle in the sense that it does not yet support dynamic adaptation. We recur to the use of virtual machines or double Snorting. However, these techniques have the main disadvantage of loosing detection state. The ideal solution would be to improving Snort source code to support the dynamic loading and unloading of rules/categories without the loss of detection state.

VII. Conclusion

In this paper we presented a policy-based framework for adaptive risk management of an enterprise information system. Our proposal is intended to provide policy-based dynamic adaptation and reconfiguration mechanisms of the deployed levels of security measures. These policies are intended to define and dynamically maintain the right balance between effective security on one hand and system usability and performance on the other.

The proof of concept has been carried out using dynamic adaptation experiments between the light-weight anomaly-based *FireCollaborator* IDS and the more advanced signature-based Snort IDPS and this over known datasets. The experiments showed how good attack detection can still be feasible along with a low resource overhead when the right set of rule categories and configuration parameters are enabled. The performance gains in terms of resource utilization as well as the ability to detect security threats and respond to them at the right time provided a proof of concept at least for the *FireCollaborator*/Snort adaptation use case. Ongoing work is considering the investigation of attack graphs, attack statistical relationships, as well as learning mechanisms so as to define appropriate adaptation policies and security levels, in addition to conducting experiments on other IDPS systems such as Bro.

References

[1] K. Scarfone and P. Mell, "Guide to intrusion detection and prevention systems (idps)," *National Institute of Standards and Technology (NIST)*, no. CSRC special publication SP 800-94, Feb 2007.

[2] M. Roesch, "Snort - lightweight intrusion detection for networks," in *LISA '99: Proceedings of the 13th USENIX conference on System administration*. Berkeley, CA, USA: USENIX, 1999, pp. 229–238.

[3] V. Paxson, "Bro: a system for detecting network intruders in real-time," in *SSYM'98: Proceedings of the 7th conference on USENIX Security Symposium, 1998*. Berkeley, CA, USA: USENIX, 1998, pp. 3–3.

[4] H. Debar and et. al., "Towards a taxonomy of intrusion-detection systems," *COMPUT. NETWORKS*, vol. 31, no. 8, pp. 805–822, 1999.

[5] J. Cabrera, J. Gosar, W. Lee, and R. Mehra, "On the statistical distribution of processing times in network intrusion detection," *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 1, 2004.

[6] L. Schaelicke, T. Slabach, B. Moore, and C. Freeland, "Characterizing the Performance of Network Intrusion Detection Sensors," *Recent Advances in Intrusion Detection: 6th International Symposium, Raid 2003, Pittsburgh, Pa, Usa, September 8-10, 2003: Proceedings*, 2003.

[7] R. Boutaba and I. Aib, "Policy-based Management: A Historical Perspective," *Journal of Network and System Management*, vol. 15, no. 4, pp. 447–480, 2007.

[8] J. François, A. El Atawy, E. Al Shaer, and R. Boutaba, "A collaborative approach for proactive detection of distributed denial of service attacks," in *IEEE Workshop on Monitoring, Attack Detection and Mitigation - MonAM2007*, Toulouse France, 11 2007.

[9] Y. Chen, Y. Yang, and I. WatchGuard Technologies, "Policy management for network-based intrusion detection and prevention," in *Network Operations and Management Symposium, 2004. NOMS 2004. IEEE/IFIP*, 2004, pp. 219–239.

[10] S. Tanachaiwiwat, K. Hwang, and Y. Chen, "Adaptive Intrusion Response to Minimize Risk over Multiple Network Attacks," *ACM Trans on Information and System Security, August*, vol. 19, 2002.

[11] L. Teo and G.-J. Ahn, "Managing heterogeneous network environments using an extensible policy framework," in *ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security*. New York, NY, USA: ACM, 2007, pp. 362–364.

[12] C. Basile, A. Lioy, G. M. Perez, F. J. G. Clemente, and A. F. G. Skarmeta, "Positif: A policy-based security management system," in *POLICY '07: Proceedings of the Eighth IEEE International Workshop on Policies for Distributed Systems and Networks*. Washington, DC, USA: IEEE Computer Society, 2007, p. 280.

[13] N. Provos, "A virtual honeypot framework," in *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*. Berkeley, CA, USA: USENIX, 2004, pp. 1–1.

[14] M. L. Lab, "2000 DARPA intrusion detection scenario specific datasets."

[15] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 DARPA off-line intrusion detection evaluation," *Computer Networks*, vol. 34, no. 4, pp. 579–595, 2000.

[16] P. Ning, Y. Cui, D. Reeves, and D. Xu, "Techniques and tools for analyzing intrusion alerts," *ACM Transactions on Information and System Security (TISSEC)*, vol. 7, no. 2, pp. 274–318, 2004.