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

Issam Aib, Khalid Alsubhi, Jérôme François, and Raouf Boutaba
David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada
iaib@uwaterloo.ca, kaalsubh@cs.uwaterloo.ca, jerome.francois@loria.fr, rboutaba@uwaterloo.ca.
Technical Report CS-2008-24

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 *FireCol* IDS. Experiments conducted over the DARPA 2000 and 1999 intrusion detection evaluation datasets show the viability of our framework[1].

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) [13] 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 [12] and Bro [10], 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.

[1]This manuscript was originally submitted to IEEE Globecom 2008.

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 [5]. Another concern is related to the significant slowing down of system performance when maximum security is applied [3], [14]; 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 [2] 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 *FireCol* [6] light-weight anomaly-based IDS [6]. 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, our *FireCol* anomaly-based IDS is briefly presented. Section V presents the experiments we have conducted using dynamic adaptation policies, the *FireCol*, 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 [4] 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 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 [15], 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 [16] propose a policy framework called Chameleos-x which is design to enforce security policies on different kinds of equipments. 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 [1] 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.

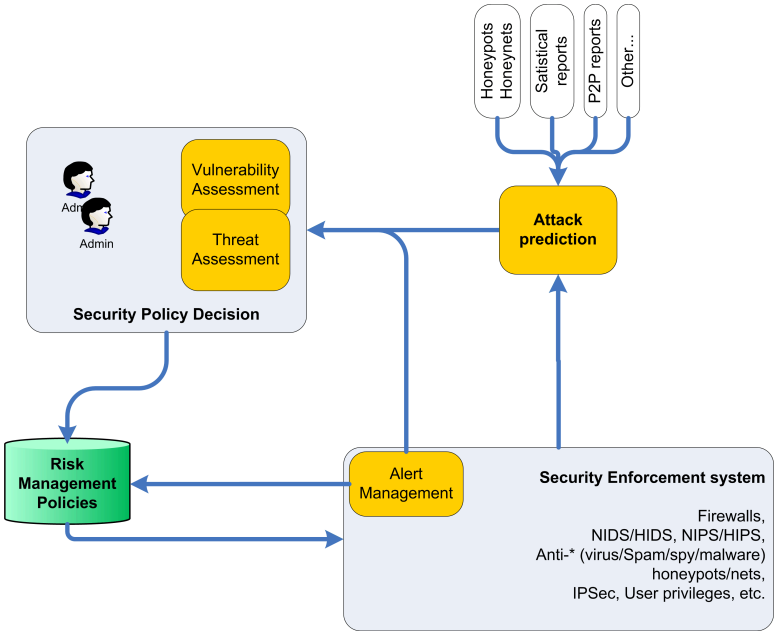


Fig. 1. Policy-based Risk Management Framework

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 [11] 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 *FireCol* [6] as an alert source for Distributed Denial of Service (DDoS) attacks. The *FireCol* 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 IDPSes, firewalls, proxies, application server security components, activity reporting and event analysis components, etc.

IV. The *FireCol* system

The *FireCol* [6] is a light weight system we developed for the detection of DDoS attacks as far as possible from the victim and as close to the attack source as possible. It can be used both as an IDS or IPS. As depicted in figure 2, *FireCol* instances can be distributed over ISP routers form rings of protection around the customer. A list of rules R_i describing packet patterns is input to the *FireCol*. After that, simple metrics related to rule matching are maintained. The metrics have a linear computational complexity which makes the *FireCol* light to run. One metric

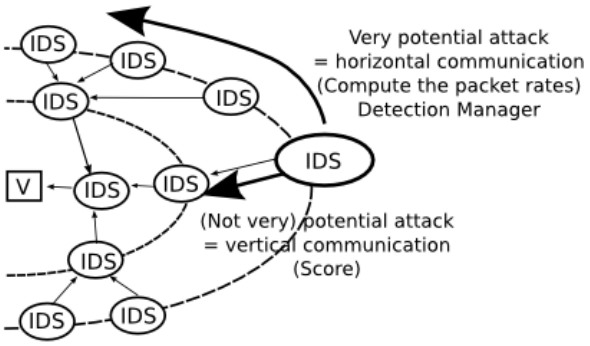


Fig. 2. Sample *FireCol* simulation topology

is related to the frequency f_i (eqn. 2) of the matching of rule R_i during a predefined detection window dw.

At each new detection window, the *FireCol* checks if f (eq. 1) is close to the last updated profile distribution. This is done using the entropy (eqn. 3) and relative entropy metrics (eqn. 5). In case the deviation is high, rules whose frequencies deviate much from profile (eqn. 6) are further investigated for potential DDoS attack based on the use of the decision table I. The output of the decision table is then used to derive a confidence score for the rule.

After that, the current score, the previous score affected by an ageing factor, and the score from upstream *FireCol* 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 *FireCol* 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 *FireCol* instance and so on [6].

$$f = (f_1, \ldots, f_i, \ldots, f_n) \text{ , (profile tuple)} \tag{1}$$

$$f_i = \frac{F_i}{\sum_{j=1}^{n} F_j} \tag{2}$$

$$H = -E[log f_i] = -\sum_{i=1}^{n} f_i log_n(f_i) \text{ , (entropy)} \tag{3}$$

$$\psi_i = log \frac{f_i}{f'_i} \tag{4}$$

$$K(f, f') = \sum_{i=1}^{n} f_i \ \psi_i \text{ , (relative entropy)} \tag{5}$$

$$\frac{f_i(t)}{f_i(profile)} > 1 + \gamma, \ 0 \leq \gamma \leq 1, f_i(t) > \epsilon \tag{6}$$

$$Score_i = f_i \times b_k \text{ , (confidence score)} \tag{7}$$

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

TABLE I
THE DECISION TABLE

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

from the *FireCol* and from Snort itself. The policies are described in table II. The *FireCol* has the advantage of being able to detect DDoS attacks as close as possible to attack sources with a light resource overhead.

Snorts rules (signatures) come into different 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. The current version of Snort (v 2.8.0.2) has a set of 51 categories. These categories can be further grouped together based on attack types. For instance, web attacks are covered by seven libraries, namely web-cgi, web-coldfusion, web-iis, web-frontpage, web-misc, web-client, and web-php.

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 2070 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 [7] and 1999 [8] datasets, which last for three hours and one week respectively. The policies used for dynamic adaptation as shown in table II.

Using the maximum detection ability (enabling all the rule sets), Snort processes the DARPA dataset in nearly 20 seconds (for both DMZ and Inside) and generates almost 4000 alerts. Furthermore, we use the default Snort configuration rule sets, which enable 32 (out of 56) of the rules and disables the others, such as policy, chat, and virus. This rule sets configuration reduced both the processing time and the number of generated alerts. Additionally, we configure the rule sets of Snort to detect DDoS by involving only the rules that responsible for detecting any steps on the DDoS attack and we successfully detected all attack phases of this type. However, customizing Snort to involve only the rules sets that accountable for detecting the Web attack results in missing all the attacks of the DARPA dataset.

As predicted in our previous work, relying only on the *FireCol* may entail many false alerts. In this experiment we use Snort as a confirming tool of the validity of a *FireCol* alert. In our case, the *FireCol* 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 *FireCol* alert. The maximal score of a *FireCol* alert is the maximal frequency multiplied by the maximal factor b_1. When the detection

TABLE II
POLICIES FOR DYNAMIC DETECTION CAPABILITIES

$\mathbf{p_1}$	**on** FireCol alert a **do** increase security by one level **when** a.score = medium
$\mathbf{P_2}$	**on** FireCol alert a **do** increase security by two levels **when** a.score = high
$\mathbf{P_3}$	**on** Snort alert a **do** increase security by one level **when** a.score = high
$\mathbf{p_4}$	**on** low security threat **do** switch to performance mode.
$\mathbf{P_5}$	**on** DDoS attack **do** enforce DDoS mitigation, notify remote mirror servers, increase backup-policy level.

is triggered, this maximal score is affected by the ageing factor a. So we obtain:

$$max_{score} = max_{frequency} \times b_1 \times a \tag{8}$$

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

$$max_{score}(0) = b_1 \tag{9}$$

$$max_{score}(i) = (max_{score}(i-1) \times a) + b_1 \tag{10}$$

When i tends to ∞, this term tends to

$$m = \frac{b_1}{1-a} \tag{11}$$

Hence, the confidence level of the score s is:

$$confidence = \frac{s-\tau}{m-\tau} \tag{12}$$

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 *FireCol* or Snort. Whenever the *FireCol* generates an alert with high confidence ($> 60\%$), Snort detection is upgraded by two levels (policy p_2). If the *FireCol* generates an alert with a medium confidence ($> 40\%$), the detection level of Snort is increased by one level (policy p_1). Low confidence *FireCol* 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). For instance, the detection level of Snort will remain in the maximum level if the *FireCol* or Snort still generating high confidence or high priority alerts and the detection level will drop down by one level if no high confidence or high priority alert is raised during the caution window that specified for the maximum detection level.

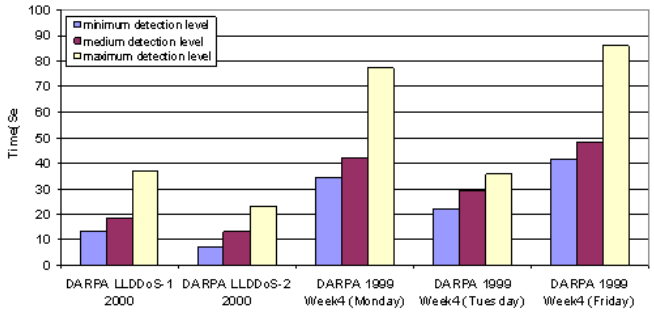


Fig. 3. Snort performance over multiple datasets using different Detection Levels

C. Results

The following experiment shows how different risk management policies can significantly affect the detection capabilities of Snort. It shows also the ability and the effectiveness of emphasizing some kind of attack according to the security administrator requirements which are based on the prediction mechanism or on assistance from other security components. For example, if the security administrator is expecting the network to be under a DoS attack, then Snort needs to be dynamically tuned for maximum DoS detection and so on. The actual difficulties ahead are related to how to effectively predict the appropriate risk management policy and be able to enforce it into the relevant components of the security enforcement components.

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 that is used in Snort to scan the dataset. 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 [7] 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. As depicted in figure 4, the sessions are grouped into five phases with the duration for each phase included. 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 *FireCol*, and as shown in table III, the DDoS victim is identified to be 131.84.1.31. In this experiment

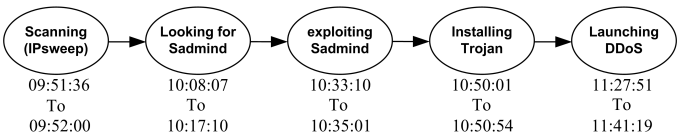


Fig. 4. Attack phases of the DARPA 2000 LLDOS 1.0 dataset

TABLE III
FireCol 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

the *FireCol* was run with a detection window of one minute. The *FireCol* manages to detect the attack with a high confidence level (67%). There are some other false positive alerts. However, most of them got 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 *FireCol* is sensitive.

The detection window size is an important parameter of the *FireCol*. 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 *FireCol* is much less than the one due to the execution of Snort due to the difference of detection specialization.

We define three detection levels in Snort which are controlled by predefined policies which support the dynamic behavior of Snort detection mechnisim.

We applied the proposed technique by scanning DARPA 2000 LLDOS 1.0 dataset (Inside part). As stated above, three detection levels are defined to be used when scanning the DARPA LLDOS 1.0 dataset. Snort starts scanning the dataset with the minimum detection level and updates its detection level based on predefined policies.

Figure 6 shows the impact of dynamic security level adaptation policies on the behavior of Snort when scanning the DARPA dataset with different caution and detection window sizes. Figure 6(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 *FireCol* 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 (figure 4), 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

γ	0.3	a	0.5	τ	0.4
b_1	1	b_2	0.65	b_3	0.8

TABLE IV
FireCol PARAMETERS

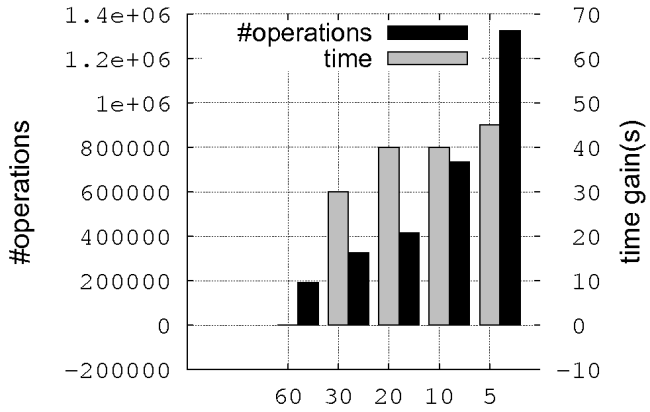


Fig. 5. Impact of Detection window size on the *FireCol* overhead

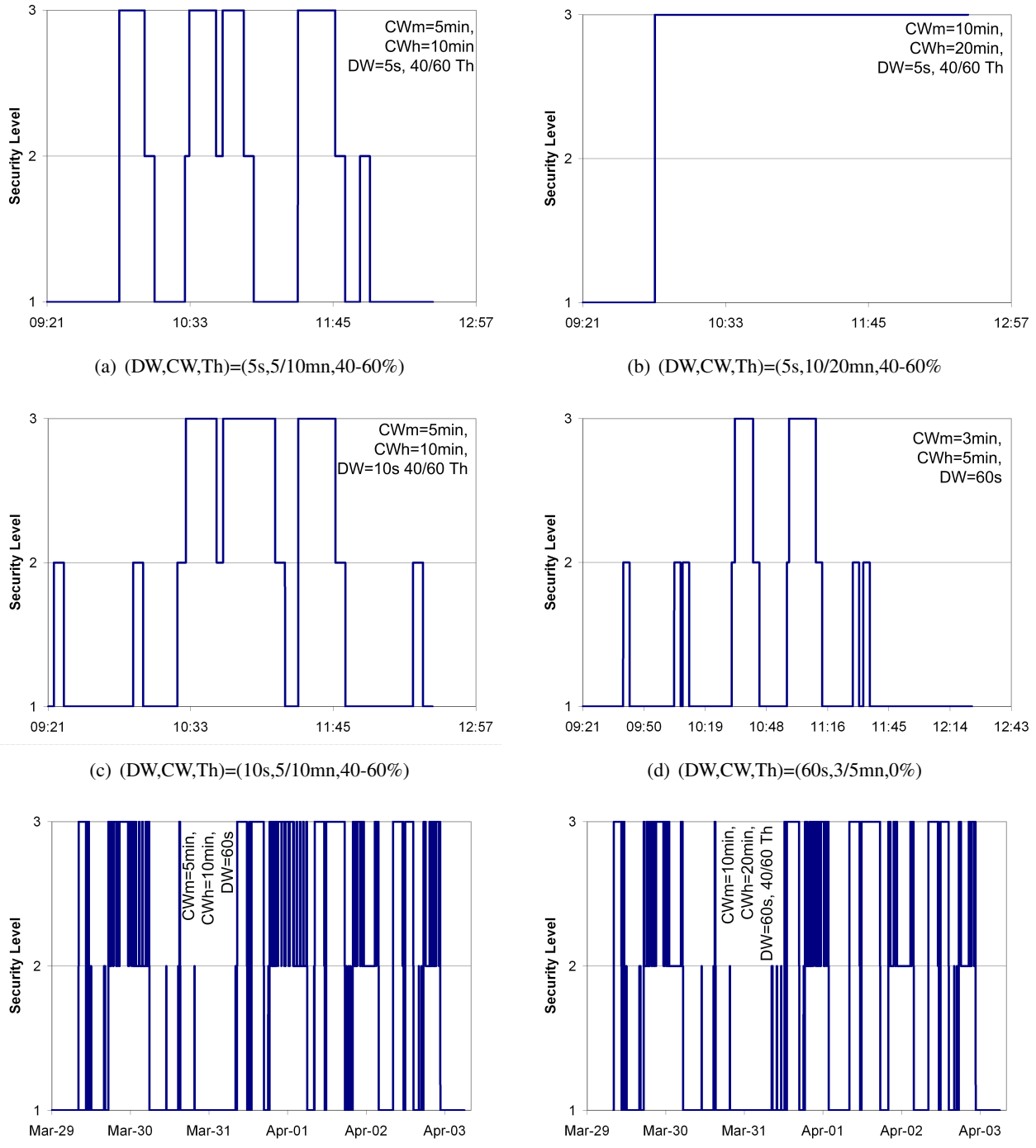
implies that it was running at minimum detection 58.17% of the time while it still managed to detect the attack phases. In terms of the time necessary to analyze the dataset, Snort took 28% less time than it did if ran at maximum detection. This is good if we know that the DARPA 2000 dataset is only three hours long and contains a five phases attack.

In figure 6(c), we use the same caution window size as in 6(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 6(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 6(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 *FireCol* make it so that the cw_x caution window never reaches an end. Finally, figures 6(e) and 6(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 *FireCol* 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 *FireCol*, and score thresholds are not trivial to set. For the *FireCol*, 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 *FireCol* 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.



(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. 6. Selected Results of Dynamic Snort Detection level adaptation for DARPA 2000 and 1999 Datasets

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 [9] 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 *FireCol* 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 *FireCol*/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] Cataldo Basile, Antonio Lioy, Gregorio Martinez Perez, Felix J. Garcia Clemente, and Antonio F. Gomez 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*, page 280, Washington, DC, USA, 2007. IEEE Computer Society.

[2] Raouf Boutaba and Issam Aib. Policy-based Management: A Historical Perspective. *Journal of Network and System Management*, 15(4):447–480, 2007.

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

[4] Y.M. Chen, Y. Yang, and INC WatchGuard Technologies. Policy management for network-based intrusion detection and prevention. In *Network Operations and Management Symposium, 2004. NOMS 2004. IEEE/IFIP*, pages 219–239, 2004.

[5] H. Debar, M. Dacier, and A. Wespi. Towards a taxonomy of intrusion-detection systems. *COMPUT. NETWORKS*, 31(8):805–822, 1999.

[6] Jérôme François, Adel El Atawy, Ehab Al Shaer, and Raouf 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.

[7] MIT Lincoln Lab. 2000 DARPA intrusion detection scenario specific datasets, 2000.

[8] Richard Lippmann, Joshua W. Haines, David J. Fried, Jonathan Korba, and Kumar Das. The 1999 DARPA off-line intrusion detection evaluation. *Computer Networks*, 34(4):579–595, 2000.

[9] P. Ning, Y. Cui, D.S. Reeves, and D. Xu. Techniques and tools for analyzing intrusion alerts. *ACM Transactions on Information and System Security (TISSEC)*, 7(2):274–318, 2004.

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

[11] Niels Provos. A virtual honeypot framework. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, pages 1–1, Berkeley, CA, USA, 2004. USENIX.

[12] Martin Roesch. Snort - lightweight intrusion detection for networks. In *LISA '99: Proceedings of the 13th USENIX conference on System administration*, pages 229–238, Berkeley, CA, USA, 1999. USENIX.

[13] Karen Scarfone and Peter Mell. Guide to intrusion detection and prevention systems (idps). Technical Report CSRC special publication SP 800-94, National Institute of Standards and Technology (NIST), Feb 2007.

[14] 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.

[15] 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*, 19, 2002.

[16] Lawrence Teo and Gail-Joon 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*, pages 362–364, New York, NY, USA, 2007. ACM.