Constructing Context-based Non-Critical Alarm Filter in Intrusion Detection

Yuxin Meng
Department of Computer Science
City University of Hong Kong
Hong Kong SAR, China
ymeng8@student.cityu.edu.hk

Wenjuan Li
Computer Science Division
Zhaoqing Foreign Language College
Guangdong, China
wenjuan.anastatia@gmail.com

***Abstract*—Currently, intrusion detection systems (IDSs) are being widely deployed in various computer networks aiming to detect all kinds of attacks. But the major problem is that a large amount of alarms are generated during their detection and most of them are non-critical alarms. This issue greatly increases the analysis workload and reduces the effectiveness of an IDS. We argue that this bottleneck stems primarily from the lack of contextual information to the intrusion detection systems. To mitigate this issue, we propose an architecture of context-based non-critical alarm filter to help filter out these non-critical alarms. In particular, our alarm filter consists of an *indexing component* to link input alarms to corresponding contextual information, an *analysis engine* aims to filter out non-critical alarms according to contextual information and a *monitor engine* to update index values. In the evaluation part, we explored the initial effectiveness of our non-critical alarm filter in a deployed network environment. The experimental results show that our alarm filter is promising and effective in filtering out non-critical alarms.**

***Keywords*-Intrusion detection; Network security; Non-critical alarm filter; Context-based system**

I. INTRODUCTION

Computer systems have become vulnerable to attacks with the rapid development of networks. According to the latest released *Internet Security Threat Report* from Symantec [7], the trend of malware attacks is increasing significantly. More than 286 million unique variants of malware were discovered and up to 6,253 new vulnerabilities were recorded.

To address these network threats, intrusion detection systems (IDSs) [1, 2] have been widely deployed into different kinds of organizations (e.g., assurance company) aiming to safeguard computer security and network environment. The security administrators can rely on them to detect and identify attacks and prevent future uses of known exploits and vulnerabilities. Moreover, the use of intrusion detection systems is a powerful complementary solution to firewall technology through defending against attacks and suspicious network traffic that are missed by the firewall.

In general, there are two traditional types of intrusion detection systems according to their detection techniques. One is the *signature-based intrusion detection systems* (also called *rule-based* IDS), which are mainly based on attack signatures to detect various attacks and threats. This kind of detection systems has to maintain a signature database and keep updating it to the latest version periodically. The signatures are usually extracted from the previously detected malicious network packets, therefore, the signature-based IDS can only identify known attacks. Take Snort [6, 12] as an example, this lightweight rule-based network intrusion detection system detects attacks by monitoring and analyzing network packets (e.g., UDP, TCP, IP). The common Snort rule format is shown as follows:

Action-type protocol-type Source-ip Source-port -> Destination-ip Destination-port (content:"attack signature"; msg:"attack msg";)

The other type of IDSs is called *anomaly-based intrusion detection systems*. Compared to the signature-based IDS, the anomaly-based IDS has the capability of identifying novel attacks. In reality, the anomaly-based approach will pre-build a normal profile to model the normal network traffic by training relevant systems with machine learning algorithms. During the detection, this approach aims to detect deviations through comparing current events with the normal profile. In actual deployment, the signature-based approach is more prevalent than the anomaly-based method in that the false alarm rate of anomaly-based systems is significantly higher than the signature-based detection systems since it is very hard to build a good normal profile in most cases [3].

Problem. Although the IDSs are proven to be effective in detecting network attacks, their generated large number of alarms greatly increase the analysis workload for a security officer. What is worse, most of these alarms are false alarms or non-critical alarms [8, 9]. The false alarm rate (or *non-critical alarm rate*) is a major limiting factor in encumbering the high performance of an IDS [5]. This issue primarily stems from the fact that current IDS detects not only the intrusions, but also unsuccessful attack attempts. Whereas it is hard for an IDS to decide the situation of an attack attempt (whether it is a successful attack or not), it has to report all detected attack attempts to security officers with the purpose of reducing security risk [4]. In this case, it is a big challenge for a security officer to analyze the information of real attacks without discarding all non-critical alarms since these non-critical alarms can greatly make negative effects on the final analysis results.

To more explicitly illustrate this problem in this paper, we

 ISBN: 978-1-61208-201-1

learn from the previous research work [10] and provide the definition of non-critical alarms as below:

Definition of non-critical alarms. A non-critical alarm is either not related to a malicious activity or not related to a successful attack. In other words, a non-critical alarm is either a false positive or a non-relevant positive.

Contributions. To improve the performance of IDSs by filtering out the non-critical alarms, we advocate that making the IDSs be aware of the contextual information of their deployed contexts is a promising method to achieve the improvement [10, 11]. In this paper, therefore, we develop and construct a context-based non-critical alarm filter to help filter out the non-critical alarms aiming to improve the effectiveness of IDSs and reduce the workload of a security officer. In particular, our proposed context-based non-critical alarm filter consists of three major components: namely, an indexing component, an analysis engine and a monitor component. For the indexing component, its main function is to link the input alarms to corresponding look-up tables which consist of contextual information. The analysis engine aims to compare the contextual information with the input alarms to determine whether the input alarms are critical or not. The monitor component is responsible for recording alarm information and updating the index values in the indexing component periodically. In terms of the indexed contextual information, our alarm filter can be adaptive to the specific network contexts.

To explore the feasibility and effectiveness of our context-based non-critical alarm filter, we implemented and evaluated this alarm filter under an established network environment with Snort. During the experiment, we converted all original Snort alarms to the type of *contextual alarms* (see Section IV). Then our alarm filter analyzed the *contextual alarms* by comparing to the stored contextual information and finally output the critical alarms. The initial experimental results show that our alarm filter is encouraging and effective in our network settings.

The rest of this paper is organized as follows: in Section II, we describe research papers that relate to false alarm reduction such as alert verification, alert correlation and machine learning based methods; Section III illustrates the architecture of our context-based non-critical alarm filter and gives an in-depth description of each component; Section IV presents our experimental methodology and shows the experimental results; limitations and future work are presented in Section V; finally, Section VI states our conclusion.

II. Related Work

A variety of solutions have been proposed aiming to reduce the number of non-critical alarms in intrusion detection. These efforts fall roughly into two general folders: *indirect reduction* such as *IDS signature enhancement*; and *direct reduction* including *alert correlation*, *alert verification* and kinds of *machine learning based approaches*.

For the signature-based IDSs, the false alarm rate (or *non-critical alarm rate*) depends heavily on the capability of their signatures. Therefore, *signature enhancement* is regarded as a promising approach to control the false alarm rate. Sommer and Paxson [14] proposed and designed a type of *contextual signature* as an improvement for the string-based signature matching. They then developed a signature engine for Bro as follows: low-level context by using regular expressions and high-level context by taking advantage of the semantic information from Bro's protocol analysis. In addition, Cost *et al.* [13] proposed *Vigilante*, a new approach to create a signature for the execution path of worms under an end-to-end environment. Followed by above work, Brumley *et al.* [15] improved *Vigilante* and provided the definition of *vulnerability signature* that is a representation for the set of inputs to satisfy a specific vulnerability condition. In the evaluation, they showed that this new type of signature achieved an improvement over existing signatures.

To directly reduce the non-critical alarms, the common approaches are *alert correlation*, *alert verification* and building *alarm filters* by using *machine learning algorithms*.

Debar and Wespi [16] proposed an aggregation and correlation algorithm to manage IDS alarms and relate these alarms together in order to output a condensed view of the reported security issue, and they also designed an aggregation and correlation component to handle alarms which were generated by probes. Then, Cuppens and Miege [17] introduced an architecture of CRIM, a cooperative module for IDSs to manage, cluster, merge and correlate alarms. The function of this module is to recognize alarms and create a new alarm from various alarms. The method of *alert verification* is to help determine whether an attack is successful or not, which is regarded as a pre-processing step for alert correlation in achieving good correlation results [19]. Gagnon *et al.* [10] evaluated the feasibility of using target configuration (i.e., operating system and applications) as contextual information for identifying non-critical alarms. Several other work (e.g., [21], [20], [32]) further showed that the alert verification improved the quality of alerts by effectively verifying the results of intrusion attempts and enhanced the performance of alert correlation.

Another widely used method in filtering out non-critical alarms is constructing an alarm filter through using machine learning algorithms. Pietraszek [31] described an adaptive alert classifier based on an analyst's feedback to help reduce false positives by using machine learning techniques. This classifier could discard alerts in terms of their classification confidence to reduce the workload of an analyst. Law and Kwok [18] proposed a method to decrease the number of false alarms by using a KNN classifier (*k-nearest-neighbor classifier*). Alharbt and Imai [23] illustrated an algorithm by using continuous and discontinuous sequential patterns to detect abnormal alarms. Davenport *et al.* [26] implemented a support vector machine to control false alarms.

 ISBN: 978-1-61208-201-1

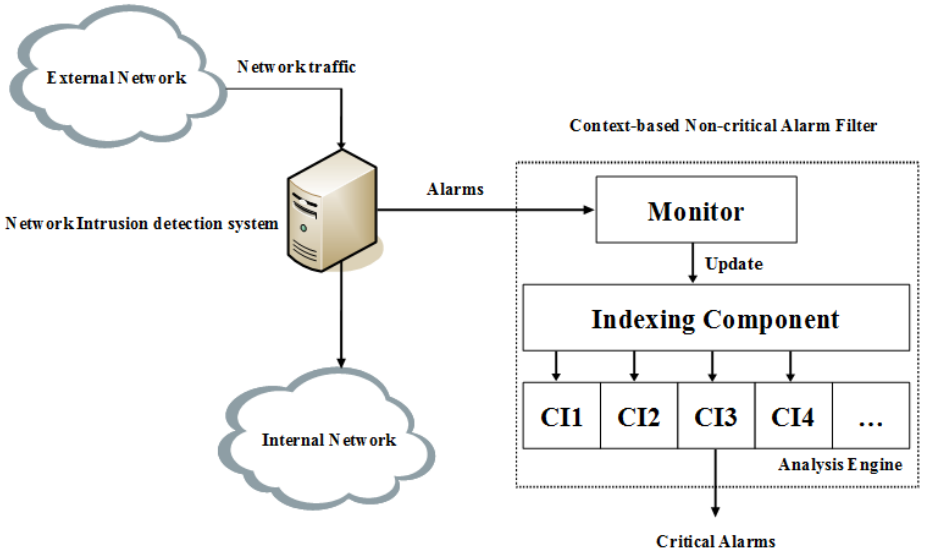

Figure 1. The architecture of context-based non-critical alarm filter with the deployment of network intrusion detection system.

Our approach is related to alert verification that aims to filter out non-critical alarms by considering the contextual information. The previous work (e.g., [10], [20]) was mainly explored the feasibility of alert verification while our work towards constructing a non-critical alarm filter. We acknowledge that our work is based on the previous results that it is appropriate and feasible to combine contextual information such as OS information and applications with IDS alarms. Differently, our work further develop this method in practice and towards automating this approach.

III. Context-based Non-critical Alarm Filter

In this section, we illustrate the architecture of our proposed context-based non-critical alarm filter. The alarm filter mainly contains three components: an indexing component, an analysis engine and a monitor component. A high-level diagram with these major components of this architecture is demonstrated in Fig. 1.

As illustrated in Fig. 1, the network intrusion detection system is deployed between an external network and an internal network in detecting network attacks by examining network packets. Its generated alarms are forwarded into our context-based non-critical alarm filter. There are three main components in the alarm filter: a monitor, an indexing component and an analysis engine. First of all, the *monitor component* records both *source IP address* and *destination port number* of an input alarm into a database and updates the indexing component periodically. Then the NIDS alarms are all forwarded to the *indexing component* in searching for the contextual information according to their index values. Finally, the *analysis engine* specifically compares the input alarms with relevant contextual information to identify whether the input alarms are critical or not. In addition, the non-critical alarms can be discarded or stored in another database for back-up and future analysis.

In the next three subsections, we give an in-depth description of these three major components respectively.

A. Monitor Component

The main task of the monitor is to collect statistical data and to update the index values in the indexing component. In this work, we use *source IP address* and *destination port number* as the index values. The construction of the monitor is shown in Fig. 2.

According to Fig. 2, when an alarm arrives, the monitor component first records its source IP address and destination port number into a database. The database is responsible for storing the source IP addresses and destination port numbers for all input alarms and updating the index values in the indexing component periodically. After recording the index values, then the input alarms are forwarded to the indexing component without any modification.

Based on the above descriptions, the database (as shown in Fig. 2) mainly contains two items: *source IP address* and *destination port number*. The major operations of the database are described as below.

- If the source IP address of the input alarm is brand new, then the database will create a new value corresponding to this IP address under the *item of source IP address*. For the new source IP address, the database can directly record its destination port number in the *item of destination port number*.
- While if the source IP address has been logged before, then the database will record the new *updating date* for this IP address. For the logged source IP address, the database has to check the destination port number.

 ISBN: 978-1-61208-201-1

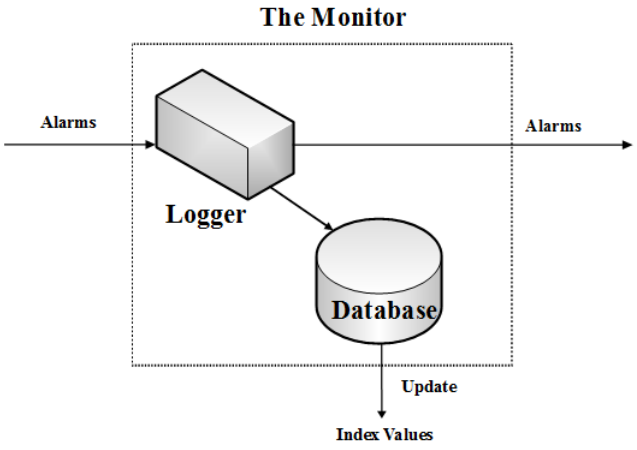

Figure 2. The construction of the monitor component in our alarm filter.

- If the destination port number has not been logged, then the database will create a new value corresponding to this destination port number under the *item of the destination port number*
- If the the destination port number has been logged, then the database will record the new *updating date* for this destination port number.

According to the specific *updating date* (which is determined by an administrator), the database can delete a source IP address if it is too dated to reduce the list length of the source IP address. Therefore, the database can be up-to-date and be adaptive to the alarm changes in real environment.

B. Indexing Component

The purpose of the indexing component[1] is to category incoming alarms based on their IP addresses and port numbers. There are two index items: *source IP address* and *destination port number*. Our scheme maintains a *complete CI database* that stores all available contextual information. In this component, the contextual information can be indexed in terms of recorded alarms' IP addresses and port numbers. For example, if an alarm was matched in the previous comparison, then its source IP address and destination port number will be recorded and be linked to that contextual information.

When alarms pass through the monitor and arrive at this component, the component will first check the *source IP addresses* of the input alarms in terms of the look-up table. If a match is identified, then the component will look for the *item of destination port number* in the look-up table and try to find another match. If a matched *destination port number* is also detected, then the relevant alarms will be forwarded to the analysis engine based on the above two items.

Otherwise, if a dis-match is either identified in the *item of source IP address* or in the *item of destination port number*, the relevant alarms will be regarded as *fresh alarms* (which have not been logged before) and have to be compared with the *complete CI database*. After the comparisons, the *source IP address* and *destination port number* of these *fresh alarms* will be recorded.

[1]In fact, this component can be incorporated into the analysis engine in real deployment, but it provides the key connection between the monitor and the analysis engine. Due to its importance, we consider it as a major component in the architecture of our alarm filter.

C. Analysis Engine

The analysis engine aims to compare the input alarms with relevant contextual information. The contextual information is the key element to our alarm filter, we mainly consider two major types: *Networking features* and *Target configuration*.

- *Networking features* consist of many different kinds of network information such as network topology, protocol specifications. These features can reflect the characteristics of distinct network environments.
- *Target configuration* usually refers to the information obtained from operating systems or applications. The information is used to help determine whether a target system is vulnerable to a given attack. For example, a Windows-based virus or worm cannot be running under a Linux system.

The basic information of known exploits can be extracted from various vulnerability databases such as Security Focus [29], National Vulnerability Database [24], Common Vulnerabilities and Exposures [27], Open Source Vulnerability Database [28]. What is more, the use of scanners [22] is an alternative if the information is not available in these vulnerability databases.

In this work, we use the *operating system (OS)* and *application (APP)* as the contextual information in the analysis engine. The evaluation steps are listed as follows:

1) If the target OS is marked as non-vulnerable to this exploit, then relevant alarms are non-critical alarms.
2) If the target APP is marked as non-vulnerable to this exploit, then relevant alarms are non-critical alarms.
3) If the target OS is marked as vulnerable to this exploit, then relevant alarms are *potential* critical alarms.
4) If the target APP is marked as vulnerable to this exploit, then relevant alarms are *potential* critical alarms.
5) If 3) and 4) are both fulfilled, then relevant alarms are regarded as critical alarms.

IV. Evaluation

In this section, we constructed an experimental network environment by using Snort (version 2.9.0.5) [6], Wireshark [30] and packet generator [25] to evaluate the performance of our context-based non-critical alarm filter. The network environment is illustrated in Fig. 3.

As shown in Fig. 3, the network traffic goes through from the source network (Internet) to the target network (Internal

 ISBN: 978-1-61208-201-1

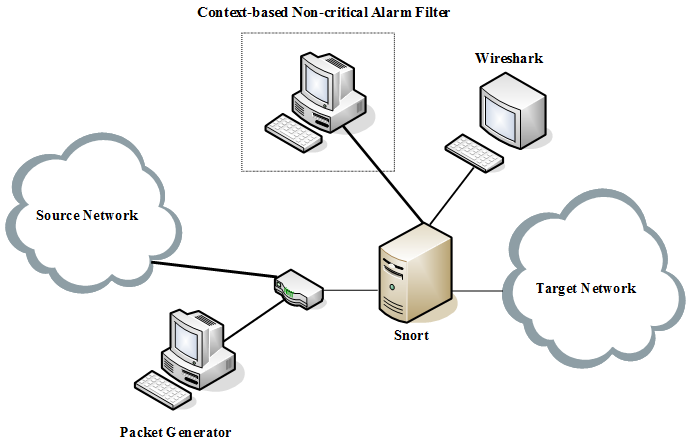

Figure 3. The deployment of experimental environment.

network). The Snort is deployed between these two networks aiming to identify attacks by checking the network packets. Our context-based non-critical alarm filter is deployed close to the Snort and all the Snort alarms will be forwarded to our alarm filter. The Wireshark is used to record the network packets and provide statistical data for analyzing experimental results.

A. Experimental Methodology

In the experiment, we use the Snort alarms as the basis for our evaluation but these original alarms do not contain any contextual information. In this case, we develop the type of *contextual alarms* by adding the *application and OS information* to the original Snort alarms before forwarding them to the analysis engine. The conversion is completed by an additional module which is designed to convert the original Snort alarms to *contextual alarms*.

By means of the concept of *contextual alarms*, the evaluation process is as below. First of all, the target network will communicate with the source network (i.e., using QQ, MSN and browsing the internet). The Wireshark will monitor and record all the network packets. Then, we used the packet generator to simulate some malicious packets by modifying the contents of three packet types: ICMP, TCP and UDP according to Snort rule database such as *icmp.rules*, *telnet.rules* and *scan.rules*.

Through sending out malicious packets, Snort can generate a number of alarms (including both real alarms and non-critical alarms) by examining the network packets. All the generated alarms will be forwarded into our alarm filter and converted to the *contextual alarms* by adding the information of target applications and OS. In the analysis engine, all *contexture alarms* will be compared with relevant contextual information followed by the evaluation steps (see Section III). The outputs of our alarm filter are critical alarms.

B. Evaluation Results

By understanding the experimental methodology, we give several examples of the *contextual alarms* in Table I. There are totally 8 items as follows: *packet type*, *source IP address*, *destination IP address*, *source port number*, *destination port number*, *alarm description*, *application information (APP)* and *OS information*.

In Table I, the first *contextual alarm* is related to TCP packets that come from the source IP "197.218.177.1" to the destination IP "172.16.114.15", the source port number is 20 and the destination port number is 80. The application information of this alarm is "IE application" and the target OS is "Windows". The description of this alarm is "ATTACK-RESPONSES 403 Forbidden". The other two contextual alarms are similar to the first one.

Table I
SEVERAL EXAMPLES OF CONTEXTUAL ALARMS.

Packet type	TCP	TCP	ICMP
Sour. IP	197.218.177.1	197.218.176.1	197.218.177.15
Dest. IP	172.16.114.15	172.16.114.15	172.16.112.2
Sour. Port	20	80	–
Dest. Port	80	4000	–
Description	ATTACK-RESPONSES 403 Forbidden	ATTACK-RESPONSES 403 Forbidden	ICMP Echo Reply
APP	IE	QQ	–
OS	Windows	Linux	Windows

In addition, Table II gives some examples of the contextual information in the analysis engine.

Table II
SEVERAL EXAMPLES OF CONTEXTUAL INFORMATION IN THE ANALYSIS ENGINE.

Sour. IP	197.218.177.1	197.218.176.1	197.218.177.1
Dest. IP	172.16.114.15	172.16.114.24	172.16.112.2
Packet Type	TCP	TCP	ICMP
Dest. Port	20	80	21
Description	ATTACK-RESPONSES 403 Forbidden	ATTACK-RESPONSES 403 Forbidden	ICMP Echo Reply
APP	IE	QQ	–
OS	Windows	Linux	Windows

The contextual information in the analysis engine contains 7 items: *source IP address*, *destination port number*, *destination IP address*, *packet type*, *alarm description*, *application information (APP)* and *OS information*.

Following by the experimental methodology, we conduct the experiment and show the initial experimental results in Fig. 4.

Analysis: As shown in Fig. 4, our alarm filter greatly reduces the number of Snort alarms in our deployed network environment. For instance, in the 10th hour of our experiment, the number of Snort alarms is decreased by 48.6% after adding the contextual information. The filtration accuracy of non-critical alarms is nearly 95% in terms of

 ISBN: 978-1-61208-201-1

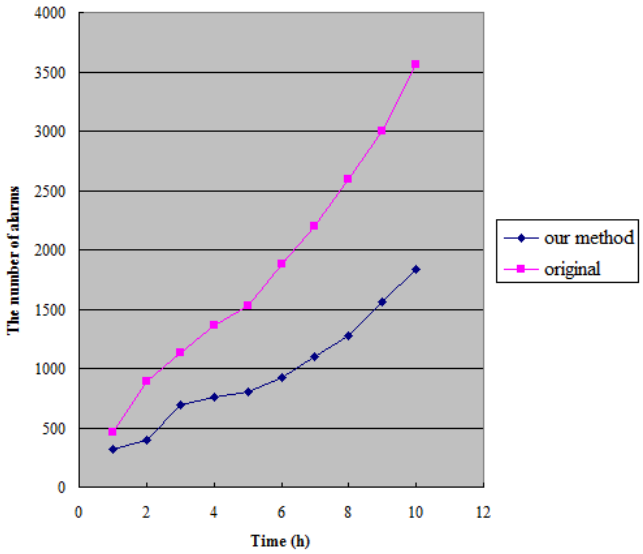

Figure 4. The alarm filtration results with regard to our context-based non-critical alarm filter in the experiment.

the packet records logged by the Wireshark. The reason for some missed non-critical alarms is that there is no relevant contextual information stored in the analysis engine or the contextual information is not found in some *contextual alarms* (i.e., the third contextual alarm in Table I, the target APP is not recognized).

In this experiment, the false filtration rate (FFR)[2] of our alarm filter is 0 in that we only filter out the alarms which are regarded as non-critical alarms according to APP and OS information. While the negative filtration rate (NFR)[3] of our alarm filter is about 10% in this experiment since we regard input alarms as critical alarms by default if the contextual information is not stored. On the whole, the initial experimental results show that our context-based non-critical alarm filter is encouraging in the network environment.

V. Limitations and Future Work

This is an early work on constructing non-critical alarm filter. Based on our initial experimental results, there are some issues that we can improve in the future work.

- *Anomaly-based detection system*. In our current work, we only investigate the performance of our scheme on Snort alarms (which are generated from a signature-based IDS). For the anomaly-based detection system, we leave it as an open problem for our future work to investigate the performance of our scheme on the alarms from the anomaly-based detection systems.
- *Contextual information*. In this paper, our work uses OS operating system and application types as the contextual information. However, we acknowledge that it is hard to identify the information accurately in some cases. We consider it as an open problem for our future work, to explore other types of contextual information.

Therefore, our future work could include considering and combining more applicable contextual information into our alarm filter to help better filter out the non-critical alarms, or constructing a more powerful alarm type to facilitate our alarm comparisons. In addition, the future work could also include comparing our scheme with other research work in this domain or deploying our alarm filter in a real network environment to further explore its performance.

[2]FFR: A critical alarm is regarded as a non-critical alarm.
[3]NFR: A non-critical alarm is regarded as a critical alarm.

VI. Conclusion

The large number of non-critical alarms is a big problem with regard to IDSs. In this paper, we propose a context-based non-critical alarm filter to help filter out these non-critical alarms. In particular, our proposed alarm filter consists of three main components: an indexing component, an analysis engine and a monitor component. The indexing component uses the source IP address and the destination port number as the index values to link the input alarms to corresponding contextual information. Then, the analysis engine compares the contextual information with the contextual alarms which are converted from the original alarms. The monitor component is used to update the index values in the indexing component. In the experiment, we explore the performance of the alarm filter under a constructed network environment. The initial experimental results show that our alarm filter is encouraging and effective to reduce the non-critical alarms in our network deployment and lighten the analysis burden for security analysts.

References

[1] V. Paxson, "Bro: A System for Detecting Network Intruders in Real-Time," *Computer Networks* 31(23-24), pp. 2435-2463, 1999.
[2] K. Scarfone and P. Mell, Guide to Intrusion Detection and Prevention Systems (IDPS), NIST Special Publication 800-94, Feb 2007.
[3] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," *In IEEE Symposium on Security and Privacy*, pp. 305-316, 2010.
[4] T.H. Ptacek and T.N. Newsham, "Insertion, evation, and denial of service: Eluding network intrusion detection," *Technical Report*, Secure Networks, January 1998.
[5] S. Axelsson, "The base-rate fallacy and the difficulty of intrusion detection," *ACM Transactions on Information and System Security*, pp. 186-205, August 2000.
[6] Snort - The Open Source Network Intrusion Detection System. http://www.snort.org/. (Accessed on November 2011)
[7] Symantec Corp., Internet Security Threat Report, Vol. 16. http://www.symantec.com/business/threatreport/index.jsp
[8] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," *ACM Transactions on Information System Security*, pp. 262-294, 2000.

 ISBN: 978-1-61208-201-1

[9] R.P. Lippmann, D.J. Fried, I. Graf, J.W. Haines, K.R. Kendall, D. Mcclung, D. Weber, S.E. Webster, D. Wyschogrod, R.K. Cunningham, and M.A. Zissman, "Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation," *In DARPA Information Survivability Conference and Exposition*, pp. 12-16, 2000.

[10] F. Gagnon, F. Massicotte, and B. Esfandiari, "Using Contextual Information for IDS Alarm Classification," *In Proceedings of the 6th Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA'09)*, pp. 147-156, 2009.

[11] R. Lippmann, S. Webster, and D. Stetson, "The Effect of Identifying Vulnerabilities and Patching Software on the Utility of Network Intrusion Detection," *In Proceedings of Recent Advances in Intrusion Detection*, pp. 307-326, 2002.

[12] M. Roesch, "Snort-lightweight intrusion detection for networks," *In Large Installation System Administration Conference*, pp. 229-238, 1999.

[13] M. Cost, J. Crowcroft, M. Castro, A. Rowstron, L. Zhou, L. Zhang, and P. Barham, "Vigilante: End-to-end containment of Internet worms," *In Proceedings of 20th ACM Symposium on Operating System Principles (SOSP)*, pp. 133-147, 2005.

[14] R. Sommer and V. Paxson, "Enhancing Byte-Level Network Intrusion Detection Signatures with Context," *In Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS 2003)*, pp. 262-271, 2003.

[15] D. Brumley, J. Newsome, D. Song, H. Wang, and S. Jha, "Towards automatic generation of vulnerability based signatures," *In IEEE Symposium on Security and Privacy*, pp. 2-16, May 2006.

[16] H. Debar and A. Wespi, "Aggregation and correlation of intrusion-detection alerts," *In Proceedings of Recent Advances in Intrusion Detection (RAID)*, pp. 85-103, October 2001.

[17] F. Cuppens and A. Miege, "Alert correlation in a cooperative intrusion detection framework," *In Proceedings of IEEE Symposium on Security and Privacy*, pp. 202-215, May 2002.

[18] K.H. Law and L.F. Kwok, "IDS False Alarm Filtering Using KNN Classifier," *In Proceedings of the International Workshop on Information Security Applications*, pp. 114-121, 2005.

[19] P. Ning and D. Xu, "Learning Attack Strategies from Intrusion Alert," *In Proceedings of the ACM Conference on Computer and Communications Security*, pp. 200-209, October 2003.

[20] J. Zhou, A.J. Carlson, and M. Bishop, "Verify Results of Network Intrusion Alerts Using Light-weight Protocol Analysis," *In Annual Computer Security Applications Conference*, pp. 117-126, December 2005.

[21] C. Kruegel and W. Robertson, "Alert verification: Determining the success of intrusion attempts," *In Proceedings of the Workshop on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA)*, pp. 25-38, July 2004.

[22] Nessus Vulnerability Scanner, http://www.nessus.org/

[23] A. Alharbt and H. Imai, "IDS False Alarm Reduction Using Continuous and Discontinuous Patterns," *In Proceedings of International conference on Applied Cryptography and Network Security*, pp. 192-205, 2005.

[24] National Vulnerability Database (NVD), http://nvd.nist.gov

[25] Colasoft Packet Builder: Packet Generator. (Accessed on July 2011) http://www.colasoft.com/packet_builder/

[26] M.A. Davenport, R.G. Baraniuk, and C.D. Scott, "Controlling false alarms with support vector machines," *In Proceedings of the International Conference on Acoustics, Speech and Signal (ICASSP)*, pp. 589-592, May 2006.

[27] Common Vulnerabilities and Exposures (CVE), (Accessed on November 2011) http:// cve.mitre.org

[28] Open Source Vulnerability Database (OSVDB), (Accessed on November 2011) http:// osvdb.org/

[29] Security Focus, http://www.securityfocus.com/. (Accessed on November 2011)

[30] Wireshark, http://www.wireshark.org. (Accessed on November 2011)

[31] T. Pietraszek, "Using adaptive alert classification to reduce false positives in intrusion detection," *In Proceedings of Recent Advances in Intrusion Detection (RAID)*, pp. 102-124, 2004.

[32] C. Mu, H. Huang, and S. Tian, "Intrusion Detection Alert Verification Based on Multi-level Fuzzy Comprehensive Evaluation," *In Proceedings of International Conference on Computational Intelligence and Security (CIS)*, pp. 9-16, Lecture Notes in Computer Science, 2005.

 ISBN: 978-1-61208-201-1