International Journal of Hybrid Information Technology
Vol. 7, No. 6 (2014), pp. 395-410
http://dx.doi.org/10.14257/ijhit.2014.7.6.34

An Approach to Verify, Identify and Prioritize IDS Alerts

Tu Hoang Nguyen[1,2], JiaWei Luo*[1] and Humphrey Waita Njogu[3]

[1]*College of Information Science and Engineering, Hunan University, Changsha, 410082, P.R. China*
[2]*Centre for Informatics and Foreign Language, Hanoi University of Industry, Hanoi, Vietnam*
[3]*Kenya Institute for Public Policy Research and Analysis (KIPPRA), Nairobi, Kenya*
hoangtu8081@yahoo.com, luojiawei@hnu.edu.cn, hnjogu@yahoo.com

Abstract

Lack of effective alert management technique to verify, identify and prioritize alerts is a well-known problem that severely degrades the worthiness of Intrusion Detection Systems (IDSs). IDSs often appear problematic because of triggering huge number of non-interesting alerts which diminish the value and urgency of interesting alerts. An average commercial IDS reports tens of thousands alerts per day. Analysts rarely look at the voluminous alerts until a sign is reported by other security means because it is laborious and challenging task to identify interesting alerts. Alerts evaluated in this manner are often unverified, mis-prioritized, misinterpreted, ignored, misclassified, delayed and are given undue attention. So far none of the current alert management techniques appear to be effective.

In this paper, we present our approach to verify, identify and prioritize alerts based on post processing of alerts. Central to our approach is the computation of new alert metrics in order to further describe and understand interestingness of alerts. We synergized Alert Verification and Alert Prioritization techniques to build an effective alert management technique. Our approach gives superior results when compared to other alert management techniques.

Keywords: *Improving Alert Quality; New Alert Metrics; Alert Classification; Intrusion Detection Systems*

1. Introduction

IDSs provide an extra layer of defense to computer networks. IDSs look for evidence of malicious behaviors by inspecting network activities and should there be manifestation of a possible security violation, IDSs generate alerts or alarms. IDSs generate alerts to notify analysts in order to make the necessary actions.

Unfortunately, all types of IDSs, including both signature and the anomaly based IDSs, have not lived up to their promise because they face critical challenges that degrade their effectiveness. They generate overwhelming amounts of alerts, some reporting thousands of alerts daily [1-3]. These alerts are unsorted, unverified, noisy and dirty. And most of the alerts are false alerts resulting from non-existing intrusions thereby diminishing the value and urgency of interesting alerts. Evaluating and analyzing raw alerts manually is error prone, resource consuming (time, skills, memory and storage) and challenging laborious task. Due to this, analysts rarely look at these alerts until a break in has been detected by other equally entrusted means such as firewalls thus making IDS more of a tool used after a break in has been reported. IDSs appear problematic and their usefulness is disregarded by many analysts. In addition, IDSs are nowadays vulnerable to *Alert Flooding* attacks where IDSs produce high

ISSN: 1738-9968 IJHIT

number of alerts with no valid reasons [4-5]. Alert flooding attack presents another challenge to IDSs that is not likely to be solved anytime soon.

According to Pietraszek and Axel [2], identifying and separating alerts has always been challenged because of the following reasons: Owing to the harsh real time requirements and high speed networks, IDSs hardly analyze the contexts of all activities to determine the presence of intrusions.

- Many IDSs use general signatures that hardly capture all variations of known attacks hence difficult to differentiate legitimate activities from the illegitimate ones. Writing specific detection signatures is often very difficult.
- Events that are normal in certain environments may be malicious in others.
- Other reasons are: Poor alerts and events correlation rules; Absence of holistic view of security environment; Availability of attacking tools to fool the IDS to generate huge number of alerts.

The present alert management techniques are ineffective and offer little help to analysts because of the nature of alerts. We identified alert classification as one of the most practical approaches to deliver quality alerts to analysts. It assigns related alerts into predefined classes according to alerts contents. We studied alert classification techniques contained in the works of [2-3, 5-8] and however noted some weaknesses where alerts are misclassified, unverified, misinterpreted, ignored, delayed and mis-prioritized. We made the following key observations after carefully studying several alert classification methods:

i. Alerts are often misinterpreted, misclassified or ignored and given undue attention because alerts features provides basic and elementary information that inadequately describe alerts in terms of their interestingness (Alerts are often processed according to their contents). Additional information describing alerts in a better way is needed. In our approach, we use new alert metrics to define alerts by their relevance, severity, frequency and source confidence.
ii. Analysts who review the alerts regularly are in better positions to identify and tell the sources (sensors) that are well known for generating high numbers of true alerts as well as sources (sensors) that are well known for generating high numbers of false alerts. This is possible because alerts are usually linked to their sources (sensors). Thus, the confidence of sensors generating alerts is deemed important when processing the alerts in our approach.
iii. Raw alerts are a mixture of all sorts of alerts: unverified, true and false alerts. The nature of alerts often makes analysts loose confidence in them. The validity and interestingness of an alert is not fully reflected by the values explicitly contained on its attributes. There is need to verify the validity of alerts in order to link them to known vulnerabilities. For example, an alert reporting intrusion on Microsoft Internet Explorer should be raised in a window based environment and not in Linux environment.
iv. The present Alert classifiers only label alerts either as true or false and do not perform other equally important tasks such as prioritize the alerts. In our approach, we perform alert prioritization when classifying alerts.

To build an effective alert management technique, we synergized alert verification and alert history. Alert verification technique verifies the validity of the raised alerts and improves the quality of alerts using vulnerability assessment data by reducing the unnecessary noisy alerts. The Alert History keeps profiles of the most frequent interesting alerts and non-interesting alerts. We used the Alert Classification Tree to build the predefined classes of our alert classifier. Our approach verifies, identify and prioritize alerts thus delivering quality alerts for analysts to quickly spot the urgent and interesting alerts. We conducted experiments with the purpose of demonstrating the effectiveness of the proposed approach.

This paper is organized as follows: Section 2 describes basic concepts of alert management. In Section 3, we propose a method to verify, classify and prioritize alerts. Section 4 presents the empirical results and discussion. Section 5 concludes this paper.

2. Basic Concepts of Alert Management

2.1. Alert Structure

IDSs produce an alert or alarm immediately after detecting a malicious activity. Usually, an alert contains several features including: Sensor Id; Alert Id; Date and Time the alert occurred; Rule generating the alert; Source IP Address; Destination IP Address; Source Port; Destination Port; Severity and Classification. Below is a sample alert from Snort.

Sept 22 00:12:39 2013 mysensor1 [auth.alert] snort[8317]: [1:2003:12] SQL Worm propagation attempt [Classification: Misc Attack] [Priority: 2]: {UDP} 202.163.16.209:1298 -> 192.168.1. 5:1434

2.2. Nature of Raw Alerts

IDSs indiscriminately generate voluminous number of different alerts with a slight deviation from normal event or behaviour. Indeed, owing to the difficulty of having specific signatures, IDSs generate voluminous alerts. These raw alerts are a mixture of non-redundant, similar, unrelated, relevant, non-relevant, true alerts, false alerts, frequent, non-frequent, interesting, non-interesting, severe and non-severe alerts.

2.3. Overview of Alert Management Approach

IDSs produce an alert immediately after detecting a malicious activity. These alerts according to Pietraszek and Axel [2] are passed to analysts to process and later identify the true alerts (see Figure 1).Usually, the analysts use their knowledge to distinguish true alerts from false alerts, a task that could be frustrating and time consuming when dealing with huge volumes of alerts. After identifying the true alerts, the analysts may investigate the intrusion, identify and fix the problem causing the alert, and or modify IDS signatures to filter out the non-interesting alerts. True alerts are always mixed with redundant, irrelevant and false alerts.

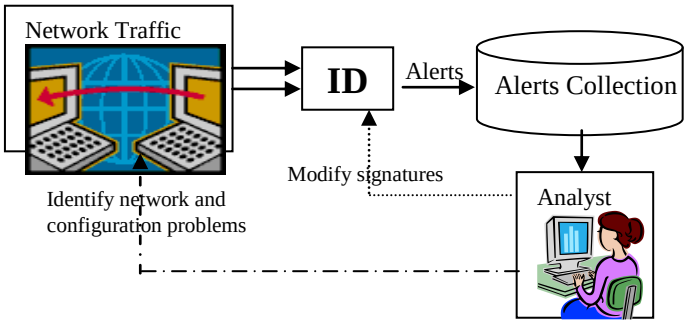

Figure 1. Overview of Alert Management [13]

3. Proposed Approach

To tackle the earlier discussed problems associated with the current alert management techniques, our approach verify alerts; compute new alert metrics; classify alerts and prioritize alerts. Our framework of how raw alerts from IDSs are verified, identified and prioritized is shown in Figure 2.

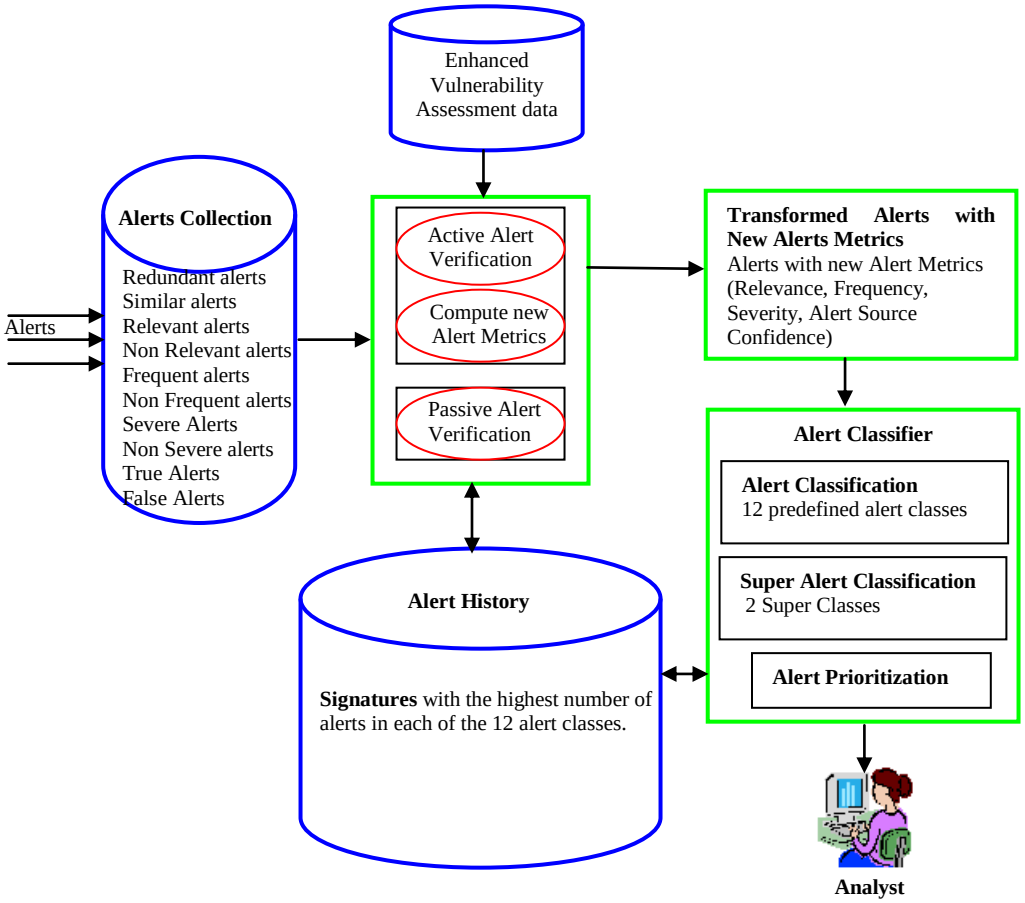

Figure 2. Alert Processing Framework

Our Approach is Described Here Below:

i. IDS detects an intrusion and issues an alert
ii. Alerts are received and stored in Alert Collection (in IDMEF format). Important features of alerts are extracted.
iii. Extracted alert information is compared with most frequent interesting alerts and non-interesting alerts stored in Alert History. If a match is found, then it means the alert has been verified by passive alert verification and the alert is placed in one of the 12 classes of Alert Classifier and prioritized. If a match is not found, the alert extracted information is verified by active alert verification.
iv. Alert extracted information is compared with Enhanced Vulnerability Assessment data in order to compute new alerts metrics: Relevance; Severity; Frequency; and

Alert Source Confidence. Alerts showing to show no similarities with assessment data are deleted.

v. The verified alerts are classified into one of the 12 predefined classes of Alert Classifier.
vi. Alerts contained in 12 classes are further classified into 2 super classes : Interesting Alerts Super Class and Non Interesting Alerts Super Class.
vii. Alerts contained in the super classes are ranked according to their alert class priorities.There are four categories of Alert Priorities: Highest Priority for Ideal Interesting Alert; Medium High Priority for Partial Interesting Alerts; Medium Low Priority for Partial Non Interesting Alerts; and Lowest Priority for Ideal Non Interesting Alerts.
viii. Alert history data is updated with necessary changes about the most frequent interesting alerts and non-interesting alerts.

Analysts are able to view alerts in terms of alert interestingness and priorities base on option of correcting the Alert History data along with Alert Classifier in case any alert misclassification is identified and can correct any network configurations if detected.

3.1. Alert Collection

It receives and stores raw alerts (IDMEF format) in an alert relational database for further analysis. The raw alerts are a mixture of redundant, similar, relevant, irrelevant, true alerts, false alerts, frequent, non-frequent, severe and non-severe alerts. Important alert features are extracted for further analysis.

3.2. Alert Verification

It is advocated as an important tool to reduce noise in the alert system [9-10]. Noisy alerts usually degrade the quality of IDS performance. This process differentiates alerts resulting from successful from those of failed intrusion attempts. It is argued as the only surest way to rule out whether an intrusion is real or not. An accurate alert verification eliminates alerts representing failed intrusions thereby reducing alert load and ensures that only quality alerts are being dealt with. We used two types of alert verification.

1) Active Alert Verification using Enhanced Vulnerability Assessment data

Alerts undergo active alert verification using vulnerability assessment data if they are not reflected in the alert history. Vulnerability assessment is the use vulnerability scanner, a proactive technique, to determine the security loop holes and potential vulnerabilities in a network [9, 11]. We prepared Enhanced Vulnerability Assessment data by integrating vulnerability assessment data together with network resource data gathered from our network in order to have a comprehensive picture of all vulnerabilities existing in the network. It contained: vulnerable services, protocol and applications.

The main purpose of active alert verification is to know the alerts worthy for further processing. It is indisputable fact that the basic alert features such as source IP, source port and timestamp provide low level, inadequate, and unreliable alert information that do not fully describe alerts thereby making them prone to being misclassified, misinterpreted, ignored or delayed. Improving the quality of alerts is the most practical way of addressing issues of alert management. Our approach computes the new alert metrics as alerts undergo alert verification process to fully describe alerts in terms of their relevancy, frequency, alert source confidence and severity. Analysts can use their knowledge to occasionally verify the new alerts metrics. The four metrics are computed together starting with Alert Relevance,

followed by Alert Severity, Alert Frequency and Alert Source Confidence. The metrics are discussed here below:

Alert Relevance: This metric measures the interestingness, importance and soundness of an alert. Alert relevance is either High or Low meaning an alert with High relevance value are regarded as Relevant while those with Low relevance value are regarded as Non Relevant. We compute the alert relevance by measuring the degree of similarity between alert data and Enhanced Vulnerability Assessment data since both data contain similar data types as alerts undergo alert verification process.

Alert Severity: this metric measure the degree of undesirable effect associated with alert's intrusion. It measures how severe an intrusion is. The alert severity is either High or Low meaning an alert with High severity can cause a higher degree of damage and is regarded as Severe Alert while the one with Low Severity is regarded as Non Severe.

Alert Frequency: It counts the number of alerts (with common features) occurring in a given period of time because an attack can trigger numerous similar alerts. An alert sharing features with others previously alerts belonging to a particular group whose count exceeds a dynamic frequency threshold is regarded to have high frequency hence Frequent Alert while an alert belonging and sharing features with a particular group whose count is below a dynamic frequency threshold is regarded to have low frequency hence Non Frequent Alert.

Alert Source Confidence: This metric is based on alert observations made by analysts on individual IDS sources (sensors) generating alerts. Analysts who review the alerts regularly can tell the sensors well known for generating high numbers of interesting alerts as well those sensors generating high numbers of non-interesting alerts. This is possible because the alerts are usually linked to their sensors by sensor Ids. This metric measures the confidence of sensors to output alerts of what they are known for in the recent past. To get better confidence value, the sensors must use the latest signatures which are frequently updated because well-tuned sensors have a high probability of capturing real intrusions thus low number of non-interesting alerts. This metric is most useful in networks with multiple IDS sensors and with their knowledge, analysts occasionally verify this metric.

2) Passive Alert Verification using Alert History

Alert History keeps profiles of the frequent alerts (interesting alerts and non-interesting alerts) that are reported in the recent past. Alert history is useful in two ways:

- Alerts are passively verified if they are reflected in Alert History. Alert History keeps profiles of the frequent alerts that are reported in the recent past thereby saving the resources associated with active alert verification process. Alert History ensures not all alerts are subjected to the unnecessary active alert verification process that degrades IDSs performance.
- Assist in the classification of similar alerts.

The reason why we keep Alert History is based on observations made by Julisch [12] that: Large group of alerts have a common root cause. Most alerts are triggered by only a few signatures and if a signature has triggered many alerts over longer periods of time, it is also likely to do so in the near future generating many similar alerts with same features. Vaarandi [1] in his work concludes that more than 85% of alerts are produced by 10 most prolific signatures.

3.3. Alert Classification Tree

We constructed an Alert Classification Tree (see Figure 3) based on new alert metrics previously discussed. This tree forms the basis for predefining classes for Alert Classifier. The tree is easily converted to classification rules. The tree handles high dimensional data

with ease and represents alert information in an intuitive form which human can easily assimilate. Alert classification tree is fast and simple to use because a path of alert can be traced from the root to a leaf node within the tree. When constructing the tree, we followed a top-down approach starting with attribute that is deemed to be of great significance. We used attribution selection measures to determine the order of importance of our metrics. Alert Relevance is considered to be of greatest significance. Alert Severity is the second most important metric. Alert Frequency and Alert Source Confidence are chosen in that order. We chose to prune and exclude Alert Severity in Non Relevant Alerts in our Alert classification tree because it is deemed unnecessary and has little significance.

3.4. Alert Classifier

We developed an Alert Classifier based on Alert Tree. The Alert Tree helps to deduce and build 12 predefined alert classes for alert classification. In order to improve the results of classifier, the Alert History simplifies classification of similar reported alerts in the recent past accordingly. Each alert can only belong to one of the 12 classes. We derived the following classification rules from the Alert Classification Tree for the 12 predefined classes:
Rule 1: If Relevance=High AND Severity=High AND Frequency=High and Alert Source Confidence =High THEN Alert belongs to **Class 1**.
Rule 2: If Relevance=High AND Severity=High AND Frequency=High and Alert Source Confidence =Low THEN Alert belongs to **Class 2**.
Rule 3: If Relevance=High AND Severity=High AND Frequency=Low and Alert Source Confidence =High THEN Alert belongs to **Class 3**.
Rule 4: If Relevance=High AND Severity=High AND Frequency=Low and Alert Source Confidence =Low THEN Alert belongs to **Class 4**.
Rule 5: If Relevance=High AND Severity=Low AND Frequency=High and Alert Source Confidence =High THEN Alert belongs to **Class 5**.
Rule 6: If Relevance=High AND Severity=Low AND Frequency=High and Alert Source Confidence =Low THEN Alert belongs to **Class 6**.
Rule 7: If Relevance=High AND Severity=Low AND Frequency=Low and Alert Source Confidence =High THEN Alert belongs to **Class 7**.
Rule 8: If Relevance=High AND Severity=Low AND Frequency=Low and Alert Source Confidence =Low THEN Alert belongs to **Class 8**.
Rule 9: If Relevance=Low AND Frequency=High and Alert Source Confidence =High THEN Alert belongs to **Class 9**.
Rule 10: If Relevance=Low AND Frequency=High and Alert Source Confidence =Low THEN Alert belongs to **Class 10**.
Rule 11: If Relevance=Low AND Frequency=Low and Alert Source Confidence =High THEN Alert belongs to **Class 11**.
Rule 12: If Relevance=Low AND Frequency=Low and Alert Source Confidence =Low THEN Alert belongs to **Class 12**.

To determine the interestingness of alert (classes), we identified some unique characteristics distinguishing interesting alerts from non-interesting alerts. The key component of an interesting alert is that it must be Relevant meaning all relevant alerts are interesting alerts. While the key component of non-interesting alert is that it must be Non Relevant meaning Non Relevant alerts. In our approach, we regard all Non Interesting Alerts as Non Severe.

3.5. Supper Classification

Alerts contained in the above twelve classes are further sorted out and classified into two super classes according to their interestingness. The super classes are: Interesting Alert super class and Non Interesting Alert super class.

1) Interesting Alerts Super Class

We treat all alert classes containing Relevant Alerts as Interesting Alert Classes. Thus all classes with Relevant Alerts are Interesting Alert Classes hence form Interesting Alerts Super Class. The Interesting Alerts Super Class contains alerts drawn from eight classes namely: Class 1; Class 2; Class 3; Class 4; Class 5; Class 6; Class 7; and Class 8.

2) Non Interesting Alerts Super Class

We treat all alert classes containing Non Relevant Alerts as Non Interesting Alert Classes. Thus all classes with Non Relevant Alerts are Non Interesting classes hence form Non Interesting Super Class. The Non Interesting Alerts Super Class contains alerts drawn from four classes namely: Class 9; Class 10; Class 11; Class 12.

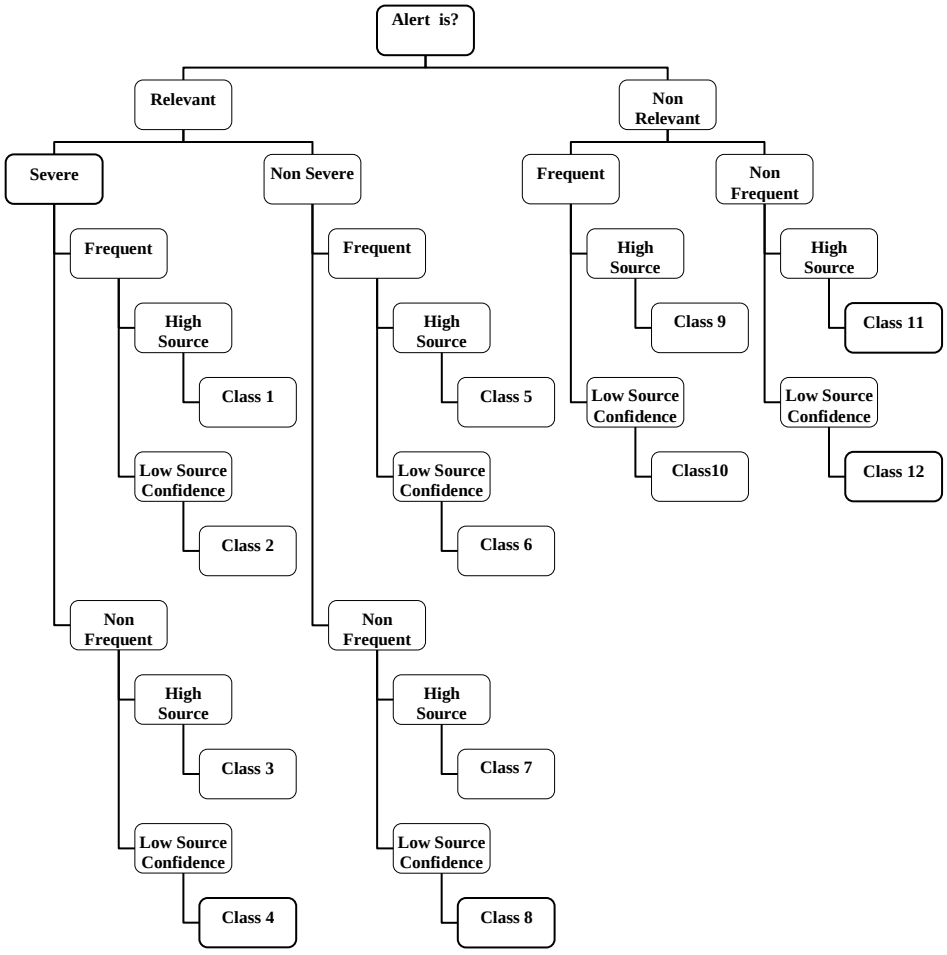

Figure 3. Alert Tree Diagram

3.6. Alerts Prioritization

It helps analysts to quickly identify alerts and act on alerts requiring immediate attention. Our approach prioritises alerts into 4 categories: Highest Priority for Ideal Interesting Alerts;

Medium High Priority for Partial Interesting Alerts; Medium Low Priority for Partial Non Interesting Alerts; and Lowest Priority for Ideal Non Interesting Alerts.

Prioritizing the Interesting Alerts

Our approach separates interesting alerts contained in the Interesting Alerts classes into 2 categories: Ideal Interesting Alerts and Partial Interesting Alerts. Alert Classifier treat all relevant alerts as interesting alerts, however, it should be noted that not all interesting alerts are ideal interesting alerts. With no doubt any alert that is Severe, Relevant and with high source confidence qualifies to be an ideal interesting alert. Ideal Interesting Alerts have the highest Priority and alerts come from 2 alert classes: Class 1 and Class 3 (both classes have relevant, severe and high source confidence alerts). Partial Interesting Alerts have the Medium High Priority and alerts come from 6 alert classes: Class 2, Class 4, Class 5, Class 6, Class 7 and Class 8 (At least these classes have relevant alerts).

Prioritizing the Non Interesting Alerts

Our approach separates non interesting alerts into 2 categories: Ideal Non Interesting Alerts and Partial Non Interesting Alerts. Alert Classifier treat all non-relevant alerts as non-interesting alerts, however, it should be noted that not all non-interesting alerts are ideal non interesting alerts. Ideal Non Interesting Alerts has Lowest Priority and alerts come from 2 alert classes: Class 9 and Class 11 (Non Relevant, High Source Confidence). Partial Non Interesting Alerts has Medium Low Priority and alerts come from 2 alert classes: Class 10 and Class 12.

4. Experiment

Currently there are neither standard performance evaluation tests nor datasets designed for methods proposed for verifying, identifying and prioritizing alerts. Therefore, it is extremely difficult to compare our results with other similar proposed approaches.

4.1 Overview of experiment setup

We conducted experiments to generate alert datasets. We used one attacking machine, 4 target machines installed with Snort sensors and 1 server. We introduced a variety of known vulnerabilities such as using old versions of software on the target machines so that the corresponding signatures could detect attacks directed on vulnerabilities. A wide range of attacks is performed against the target machines by the attacker. The attacks are both severe and trivial attacks targeted to exploit known vulnerabilities in order to generate comparable alerts in terms of severity, relevance, frequency and source confidence *e.g.*, same attack targeting all machines to generate similar alerts needed in computing alert frequency. The attacks are run severally and alerts are categorised into four groups. We installed different versions of Snort and different signature updates into target machines as attacks were performed to produce alerts for better alerts evaluation results.

We chose Snort 2.8.5.2 as our network IDS because it is robust, flexible and open source to generate alert set in for our experiments. We used Nessus 4.2.1 vulnerability scanner to generate vulnerability data in our network because of its high quality vulnerability and the ease in integration into Snort. We prepared Enhanced Vulnerability Assessment data that integrates Nessus's Vulnerability Assessment data and with Network Resource data (an asset database prepared to list all important resources in the network) in order to link local network resources to vulnerabilities existing in our network.

MySql receives and stores alerts from four sensors (Alert Collection Server); keeps Alert History; keeps alert sensor data; and Enhanced Vulnerability Assessment data. We also used

MySql to keep the profiles of alert sources performance in the recent past of sensors. WEKA tools were used to perform alert classification.

Our system is trained for over two weeks on how to verify, classify and prioritize alerts by dividing the datasets into training dataset and testing dataset. The training dataset represented 30% of total input alerts while testing dataset represented 70% of total input alerts. The training and testing datasets is replayed over and again to produce the desired results meaning that our system should ideally not decrease the detection rate but should lower the false positive rate. This also involved keeping profiles of each sensor for a week tracking the maximum and minimum number of alerts as well as the ratio of interesting alerts to non-interesting alerts.

4.2. Processing Alert

We tested the usefulness of the new alert metrics and we were surprised to see how they recorded an impressive effect on separating alerts. We employed each alert metric individually to evaluate their performance. We decided to settle on the four alert metrics to represent alert in terms of relevance, severity, frequency and source confidence thus giving optimal results. We kept on reviewing the parameters of alert metrics such as frequency thresholds to ensure the metrics truly represented the alerts.

To know importance of alert source confidence, we compared alerts produced from different Snort versions and with different updates. We observed that the Snort versions and the frequency of updating sensor's signatures influence the production of alerts. Group 1 has the lowest number of alerts because alerts were generated by 4 sensors using versions 2.8.5.2 of Snort (see Table 1).

Table 1. Alerts Collected and Verified

Alerts /Group	Group 1	Group 2	Group 3	Group 4
Alerts Collected (Unverified)	21564	28945	33855	53452
Number of alerts verified by passive alert verification (Alert History)	4313	5789	6675	4278
Number of alerts verified by active alert verification (Enhanced Vulnerability Assessment data)	3795	5570	5301	9187
Number of Alerts deleted during alert verification	13456	17586	21879	39987

Group 2 has alerts generated when three sensors used the lastest versions of Snort and one sensor with older version. Group 3 contains alerts from two sensors using latest versions of Snort while the other two used older versions of Snorts having different signature updates. Group 4 contains alerts, generated by alert flooding attacks leading to high number of alerts. The unusual number of alerts reported by in Group 4 is detected because of sensor profile thus notifying the analysts in time.

The goal of any alert management system is to have high detection rate and low false positive rate. To achieve low positive rate, we reduced the alerts through alert verification using Enhanced Vulnerability Assessment data and Alert History data. To ensure our vulnerability data was representing the actual vulnerabilities in the network, we manually verified whether the Enhanced Vulnerability Assessment data contained the already known

vulnerabilities and confirmed that 99% of known vulnerabilities were reflected. This ensured that we were not running at risks of ignoring or deleting important alerts whose contents is not reflected in the vulnerability data. With alert verification process, unnecessary alerts were dropped thereby reducing the alert load thus enabling us to focus on important alerts. At least 35% of input alerts were deleted meaning our classifier alert load is reduced significantly making it more efficient (refer to Table 1).

To take advantage of previously mined knowledge, our approach opted to use Alert History data to verify alerts passively. From the results shown above at least 20% of input alerts was verified automatically (alerts are classified directly like other similar previously mined alerts) using passive alert verification meaning the classifier alert load is reduced by 20%. This means our classifier does not need to classify and prioritize alerts already reflected in the alert history hence saving the resources such as memory, CPU and time. This is one the reasons that made our approach to perform the tasks of alert verification, classification and prioritization within a very short time as compared to other approaches.

We classified alerts without verifying alerts and noted that it took longer time to classify alerts and even much longer to identify the urgent alerts. When we manually reviewed the alerts classified this way, we noted numerous cases where alerts were misclassified, ignored, delayed or misinterpreted and given undue attention. From the test conducted on Group 2 alert set, we noted that 507 cases of alerts were misclassified meaning some interesting alerts were classified as non-interesting alerts and non-interesting alerts were classified as interesting alerts, 212 cases of alerts were delayed meaning that all these cases were severe and urgent alerts hence they were not given the due attention needed. 303 alerts were misinterpreted because their alert data could not tell much about them. These alerts were actually severe alerts but were interpreted as non-severe alerts. We also noted that 592 alerts were ignored meaning that they appeared as non-interesting alerts but actually were interesting alerts.

Still when dealing with alerts that were not verified, we took 7 hours to manually identify the interesting alerts in 5,000 alerts which represent 10% of total alerts from Group 4. This means it could have taken us at least 70 hours to manually identify interesting alerts from a total of 53,452 alerts. It then took us another four hours to identify the most urgent alerts from the interesting alerts (representing 10% of interesting alerts). This means we could have taken at least forty hours to identify the urgent interesting alerts. Therefore, it could have taken us at least 110 hours to manually identify the interesting alerts and later identify urgent alerts from 53,452 alerts.

We used our approach to verify and classify alerts are distributed in the twelve alert classes as shown in Figure 4.

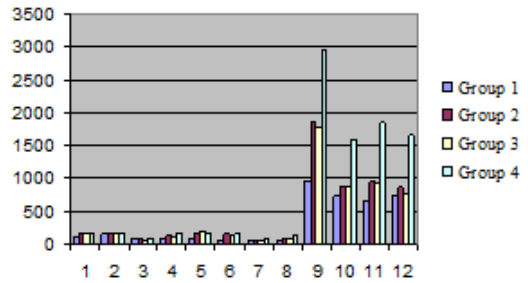

Figure 4. Distribution of Alerts in 12 Alert Classes for Simulated Data

Alerts contained in the 12 predefined alert classes were further classified into the 2 super classes.

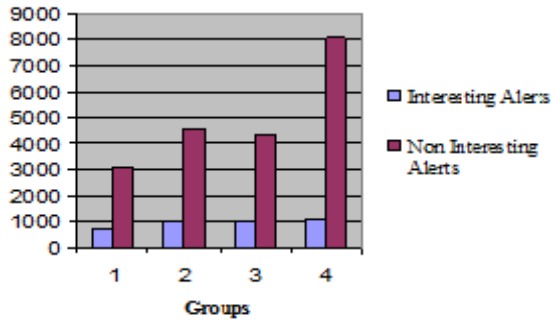

Figure 5. Interesting Alerts and Non Interesting Alerts for Simulated Data

From the results see Figure 5 we noted that interesting alerts represented about 10% of alerts while 90% were non interesting alerts. This confirms Vaarandi observation [1] that false alerts are the majority of alerts and constitutes of 90% of processed alerts. Our method separates non interesting alerts enabling analysts to focus more on interesting alerts (10% of processed alerts). Our approach eliminated by 98% of cases of misclassification, delay, misinterpretation witnessed when alerts were not verified and were manually classified.

To test the importance of alert prioritization, we sorted the classified alerts manually. Most classification just label alerts as true or false but do not provide any further information. It could have taken us at least 40 hours to identify urgent alerts when alerts were just classified and not prioritized. But when alerts were prioritized it took less than a minute to find urgent and interesting alerts from the classified alerts.

As seen in Figure 6, alerts with the highest priority are the least. These alerts represent what analysts must focus on urgently. Alerts with the medium high priority are second least from highest priority alerts. They are second important alerts and analysts should focus on these alerts too. Alert with medium low priority may either be deleted or not. Alert with lowest priority should be treated with least concern and should be deleted immediately. We manually reviewed the prioritized alerts and found 98% of alerts were prioritized correctly.

From alert collection, we observed that raw alerts are a mixture of all sorts of alerts that is unverified, frequent, severe, non-severe, true and false alerts. When we classified the alerts without computing new alert metrics and with no alert verification, we took us at least 110 hours to get the urgent and interesting alerts. We conclude that the nature of alerts is likely to make analysts lose confidence in alerts. The reason behind this is that the validity and interestingness of an alert is not fully reflected by the values explicitly contained on its attributes hence the need to verify the alerts in order to link them to known vulnerabilities.

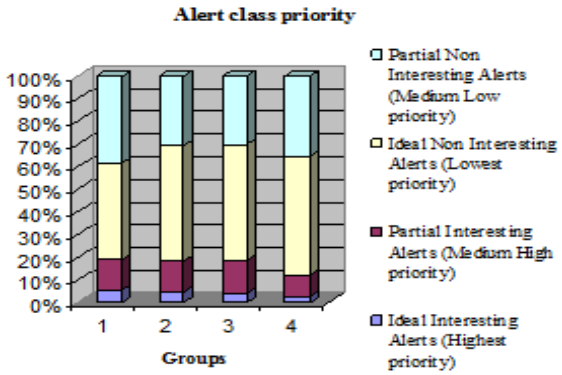

Figure 6. Alert Class Priority for Simulated Data

To validate the simulated data, we decided to use a real world dataset generated by Snort IDS in our medium network. The network connects to the Internet through firewalls and to the rest of the corporate intranet. The 4 Snort sensors recorded traffic between the Internet and the Intranet and triggered alerts for a period of two weeks. We kept on manipulating different versions of Snort and different signature updates installed. We recorded 285,647 and 295,487 alerts for week 1 and week 2 respectively. When alerts were verified, a much similar pattern was observed. 20% of alerts were verified passively. Similar to simulated data, at least 50% of alerts were deleted during alert verification. Similar patterns of how classified and prioritize alerts were distributed in simulated data were seen in real world real data.
(see Figure7 and Figure 8).

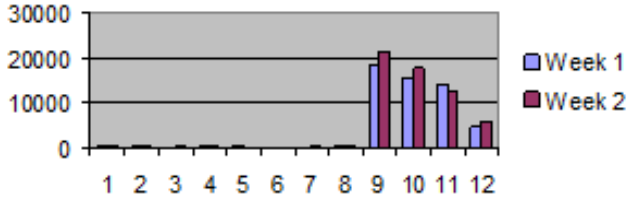

Figure 7. Distribution of Alerts in 12 Alert Classes for Real World Data

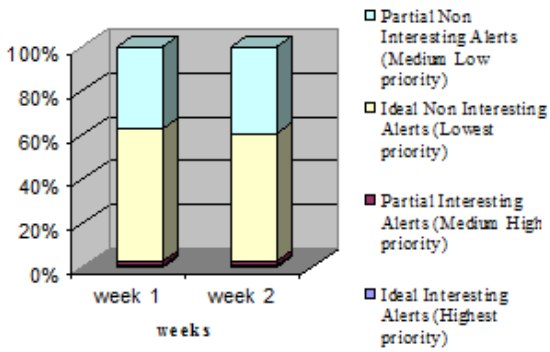

Figure 8. Alert Class Priority for Real World Data

When compared with other approaches [2-3, 5-8], our approach appears to be superior because we recorded high accuracy rate of 90 %, reduced alerts load by over 70% and lower

false positive rate. The present proposed alert classifiers only label alerts either as true or false and do not perform other equally important tasks such as prioritize the alerts and verifying alerts. Our approach presents a key advantage of carrying two tasks at once, that is, alert verification and computing new alert metrics hence more efficient. Our approach, takes relatively shorter time to verify, classify and prioritize alerts. In our approach, we perform alert prioritization when classifying alerts. Our approach combines respective advantages of major alert management techniques by synergizing alert classification with alert verification, alert history and alert prioritization.

5. Conclusion

In this paper, we have presented approach to verify, identify and prioritize alerts. Central to this approach is the computation of new alert metrics to further understand alerts in terms of their relevancy, severity, frequency and their source confidence. Based on new alert metrics, we developed a useful system to help analysts in identifying alerts quickly according to their priority and interestingness. We conducted experiments to demonstrate the effectiveness of the proposed approach and its performance is superior than other alert management techniques.

As future work we are planning to evaluate the performance of our approach using DARPA dataset because our dataset may not be representative for all real world datasets. Other future work is to automatically build a comprehensive Enhanced Vulnerability Assessment data that is updated on hourly basis to ensure alerts are tagged with correct new metrics that truly describe alerts. We also want to improve on how to verify the new alert metrics in future.

Acknowledgement

This work is supported by Hunan Provincial Natural Science Foundation of China (Grant No. 13JJ2017).

References

[1] R. Vaarandi, "Real-time Classification of IDS Alerts with Data Mining Techniques". Proceedings of Military Communication Conference, MILCOM, **(2009)**, pp. 1-7
[2] T. Pietraszek and A. Tanner, "Data mining and machine learning -Towards reducing false positives in intrusion detection", Information Security Technical Report **(2005)**, pp.169-183
[3] S. Lee, B. Chung, H. Kim, Y. Li, C. Part and H. Yoon ,"Real-time analysis of intrusion detection alerts via correlation", Computer & Security, vol. 25, no. 3, **(2006)**, pp.169-183
[4] P. Georgios and K. Sokratis, "Reducing False Positives in Intrusion Detection System", Computer and Security, vol. 29 , **(2010)**, pp. 35-44
[5] T. Chyssler, S. Burschka, M. Semling, T. Lingval and K. Burbeck, "Alarm Reduction and Correlation in Intrusion Detection Systems", Detection of Intrusions and Malware & Vulnerability Assessment, GI SIG SIDAR Workshop (DIMVA 2004), **(2004).**
[6] T. Subbulakshmi, G. Mathew and S. Shalinie, "Real Time Classification and Clustering of IDS Alerts using Machine Learning Algorithm", International Journal of Artificial & Application , **(2010)**, vol. 1, no.1.
[7] X. Liu, D. Xiao and X. Peng, "Towards a Collaborative and Systematic Approach to Alert Verification, "Journal of Software, vol. 3, no. 9, **(2008)**, pp. 77-84.
[8] B. Najwa and B. Belaton, "Towards Implementing Intrusion Detection Alert Quality Framework", Proceedings of the first International Conference on Distributed Framework for Multimedia Applications, **(2005)**
[9] C. Kruegel, W. Roberstson and G. Vigna, "Using Alert Verification to Identify Successful Intrusion Attempts", PIK 27, vol. 4, **(2004)**
[10] M. Colajanni, D. Gozzi and M. Marchetti, "Selective alerts for run time protection of distributed systems", **(2008)**

[11] R. Perdisci, G. Giacinto and F. Roli, "Alert Clustering for intrusion detection systems in computer networks", Engineering Application of Artificial Intelligence", vol.19, **(2006)**, pp. 429-438
[12] K. Julisch, "Clustering Intrusion Detection Alerts to Support Root Cause Analysis", ACM Transactions on Information and System Security, vol.6, no.4, **(2003)**, pp. 443-471
[13] N. HoangTu, L. Jiawei and N. HumphreyWaita., "Improving the management of IDS alerts", International Journal of Security and Its Applications, vol.8, no.3, **(2014)**, pp. 394-405

Authors

Tu Hoang Nguyen, he received his M.Sc in Computer Science from College of Information Science and Engineering, Hunan University, Changsha, Republic of China. After completing his Master's program, he got the distinguished student scholarship to for a PhD program in Computer Science at Hunan University. His research interests include Intrusion detection and prevention, vulnerability analysis, network security, bioinformatics, and data mining.

Jiawei Luo, she is a full professor and vice dean at the College of Information Science and Engineering, Hunan University, Changsha, Republic of China. She holds Ph.D., M.Sc. and B.Sc. degrees in Computer Science. Her research interests include data mining, network security and bio informatics. She has a vast experience in implementing national projects on Bioinformatics. She has authored many research articles in leading international journals.

Humphrey Waita Njogu, he is currently an IT Security Expert working in a government agency in Kenya. He received his Ph.D. and M.Sc in Computer Science (Security) from Hunan University, China. He received his B.Sc. degrees in Information Science from Moi University, Kenya. He holds several IT professional certifications in Cisco, Oracle, Comptia, Linux and Microsoft. His research interests include most aspects of IT security, with an emphasis on network security, Intrusion detection and prevention, vulnerability analysis, data mining, Green Computing, Cloud computing security and Social media security. He has a vast experience in implementing several IT security projects. He has authored many IT security research articles in leading ISI/SCI top indexed international journals and conferences.