A Simple Method for Improving Intrusion Detections in Corporate Networks

Joshua Ojo Nehinbe

School of Computer Science and Electronic Engineering System
University of Essex, Colchester, UK
jnehin@essex.ac.uk

Abstract. Intrusion redundancies are fundamental flaws of all intrusion detection systems. Over the years, these are frequently exploited by stealthy attackers to conceal network attacks because it is fundamentally difficult to discern false alerts from true positives in a massive dataset. Consequently, attacks that are concealed in massive datasets often go undetected. Accordingly, the jobs of system administrators and the return on investment on network intrusion detectors are often threatened. Therefore, this paper presents clustering method that we have designed to lessen these problems. We have broadly evaluated our method on six datasets that comprised of synthetic and realistic attacks. Alerts of each dataset were clustered into equivalent and unique alerts and a cluster of unique alerts was eventually synthesized from them. The results that we have obtained have indicated how system administrators could achieve substantial reduction of redundancies in corporate networks.

Keywords: Redundancy, probing attacks, correlation, aggregation, equivalent alerts and unique alerts.

1 Introduction

Intrusion redundancies that are classical flaws of [1], [17], [18], [15], [19] traditional intrusion detection systems often pose serious threats to both the system administrators and continuous usage of intrusion detection systems. In reality, there are three critical issues that are associated with intrusion redundancy. Firstly, how to achieve reasonable reduction in the proportion of redundancy that is present in a collection of alerts so that security breaches on the networks are not overestimated is a difficult task. The second problem is how to accurately discern true positives from unrealistic attacks that are erroneously reported together so that countermeasures are not implemented against legitimate events. Also, timeliness in responding to attacks is another critical issue that relates to intrusion redundancy.

Generally, the origin of these problems is traced to the point at which network detector decides which network packet is suspicious or which packet is a normal event. Basically, each network detectors has collections of rules that contain signatures, patterns or characteristics of what should be classified as security violations. These rules are used to validate incoming traffics by comparison to identify matches that indicate intrusions. Unfortunately a detector treats each event as a new occurrence and thus assigns a unique sequence identity to each of them irrespective of whether it is repeated within the same timestamp. Hence, the system administrators [9], [7], [11], [14], [3], [19] are overwhelmed with repeated notices whenever there are continuous migration of repeated packets across the networks. Now, the classification problems that could not be adequately tackled by the detector are invariably transferred to the system administrators. However, alerts that need urgent attentions are manually analyzed. Also more time and efforts are spent [9], [22] to ascertain the correctness of each alert and to ensure that preventive actions are not taken against legitimate network activities. Furthermore, this process becomes extremely cumbersome especially if there are very few novel attacks that are purposely buried in massive alerts.

One of the effective approaches to lessen these problems is to configure the detector and the detection rules to [16] suppress some proportions of alerts that will be reported at a specified frequency [9]. The implementation can also be directed towards specific addresses (source and destination), protocols, etc. Similarly, detection rules can also be prioritized to completely deactivate nuisance rules or rules with low priorities but then, these methods would only be feasible for a detector that has such functionalities.

Although alerts suppression techniques can significantly reduce some proportions of redundancies in some datasets [16] but their flaws often outnumber their benefits. For instance, they cannot significantly reduce redundancies that are caused by equivalent events that occur in different time windows. Also, suppression methods are vulnerable to high rate of false negatives especially when a target machine is attacked with probing attacks that are below the threshold that has been designed for suppressing alerts. For instance, a packet of ping attack is just enough to evade detections. For these reasons, alerts suppression methods frequently underestimate security breaches on the networks. Furthermore, alerts suppression methods have limited capacity to cluster long sequence of attributes. Hence, they cannot substantiate variability in redundancies using different alerts' attributes. Apart from the trouble in reconfiguring tons of detection rules to suppress alerts, fundamentally, the efficacies of alerts suppression methods are limited to a few selection criteria. Accordingly, these flaws have necessitated the implementation of intrusion detectors in default modes while reduction of the redundancies that are simultaneously generated are central research issues in a recent time [18].

Furthermore, the capability of adapting some methods such as [23] intrusion correlation and aggregation to solve these problems have been fully established [19] in recent publications. Intrusion correlation is the process of finding fundamental relationships that connect two or more alerts together through in-depth analysis of audit logs while intrusion aggregation is a correlation technique to succinctly reduce intrusion redundancy. Essentially, both techniques should enhance prompt post-intrusion reviews so that appropriate countermeasures that would foil computer attacks can be promptly achieved. Nevertheless, the existing methods have three inherent limitations. They are unable to substantiate the complexity in removing all redundancies in a set of evaluative datasets. Also, they were not objectively evaluated and they were not design to model system administrators that were saddled with these important responsibilities in realistic networks. Consequently, these three core issues were explored in this paper to solve the aforementioned problems. Our idea was that attacks on realistic and experimental networks could only be thoroughly investigated using offline analyses of synthetic and realistic trace files. We also premised that the results obtained could be used to preempt future attacks, establish root causes of most computer attacks and to demonstrate feasible countermeasures that would reduce the problems of intrusion redundancy and other related issues in realistic computer networks.

Therefore, we deployed Snort intrusion detector on a segment of our Local Area Network (LAN) that enabled it to sniff simulated and realistic attacks. The alerts produced in each case were processed with an automated clustering technique that was designed to model system administrators that were saddled with the aforementioned challenges in corporate networks.

One of the significant contributions of this paper was our ability to reduce redundancy with a simple clustering technique. Also, series of experiments that we have conducted with wide-range of selection criteria have been able to substantiate the variability of redundancies across realistic and synthetic datasets.

The remainder of this paper is organized as follows. Section 2 discusses related works that have been done to reduce redundancy. Section 3 gives fundamentals of intrusion detections with particular reference to Snort. Section 4 gives an overview of evaluative datasets and description of our method to reduce redundancy. Section 5 gives the results of various experimentations that we have conducted while section 6 gives conclusion of our work and future research direction.

2 Related Work

Extensive researches that have been conducted on log analysis [8], [13], [23] cannot be fully elaborated in this paper due to space limitation. However, these methods were not directly designed to solve the problems of intrusion redundancy. Instead, intrusion redundancies are solved with the problems of false positives.

Algorithm that uses [7] implicit rules is used to eliminate alerts that have the same consequence as redundant alerts and the rules generate a new attack thread each time a new attack is discovered in the audit log. Furthermore, this idea is further extended to design expert systems that transform attributes of alerts into expert rules [12], [14]. In this approach, each incoming alert is validated against the expert rules to detect similar patterns of attacks while a deviation is taken as a false positive. A substantial difference in the two approaches that we have reviewed was the mode of updating the rules. While a group adopted an automatic update of the expert rules [12], the second group manually updates its rule engine [14]. Nevertheless, the major problem with rule-based methods is that their performances depend on the ability of the rules to correctly identify attacks.

Furthermore, queue graph is implemented to [20] derive the patterns of multistep attacks that occur at different timestamp. The algorithm searches for the most recent alerts of each type that prepare for future alerts and correlates them on the basis of timestamp. To the best of our knowledge, this model will mismatch multistep attacks that do not feasibly prepare for future attacks. Also interpretation of queue graphs become complex as the quantity of novel attacks in a dataset increases.

Besides, the outcome of earlier attacks (consequences) and the conditions that enable them to succeed (prerequisites) [4] on a network have been aggregated to reduce redundancy. Similarly, we have noticed that this method usually mismatches probing attacks such as version and operating system attacks that have roughly the same consequence and prerequisites.

Bayes probabilistic [2] theorem is adapted to aggregate log entries using a correlation algorithm that takes synthetic alert as input data and returns series of values from 0 to 1. Another statistical method that [22] has a statistical-based correlation algorithm is implemented on the premise of Granger Causality Test. Similar alerts are aggregated using their corresponding pre-conditions and the impacts they have on the networking infrastructure[22]. However, statistical methods are unable to aggregate alerts that do [23] not form statistical relationships and they are unable to process complex alerts [14].

Clustering technique that analyzes [8] audit log and construct attack scenarios to show attack graphs has been implemented to lessen the proportion of redundant false positives in an audit log. In this method, causality matrix is introduced to automatically extract attack scenarios using small number of rules and the model clusters alerts on the bases of attack, addresses and time of occurrence. Apart from its low ability to reduce redundancy, the model was not extensively evaluated.

Similar alerts that are commonly reported by all the detectors in a network have been adopted [21] to isolate redundancy. Alerts are prioritized and the impact of each attack on the network is determined. Thus, redundancy is expunged as alerts with low priority while non-redundant alerts are converted to Intrusion Detection Message Exchange Format (IDMEF) for further processing. However, high false positives and manual evaluation are some of the major flaws of this technique.

A formalized approach [13] is proposed to eliminate redundancy and some fundamentals challenges of intrusion detection technology based on a detailed understanding of the resources on the networks. Basically, inconsistency in the alerts of any of the detectors whenever they are clustered together [21] is presumably taken as redundancy. However, this method often underestimates the topological effects, the locations of the detectors on the networks and the detection capabilities of each detector on the discrepancies of their results. Practically, a formalized method is unable to handle attacks that elude detections. We have also noticed that formalized approach is complex and it was not implemented. Hence it was not objectively evaluated to ascertain its efficacy.

The performances of [3] three different classifiers in reducing redundancies have been substantiated in a model that uses Naive Bayesian (NB), k-nearest neighbors (K-NN) and Neural Network (NN) classification methods to process alerts from multiple intrusion detectors. All alerts are classified on the basis of their severity and each classifier eliminates redundancy with alerts that have low severity details. Nonetheless, ability to correctly determine interesting alerts, eliminate the problems of discrepancy that are associated with multiple detectors and capability to correctly map heterogeneous alerts together are some of the factors that militate against the efficacy of this method.

3 Network Intrusion Detections

Snort is a rule-base expert system that uses precedence rules to process network packets during intrusion detections. We have observed that the rules can monitor specified addresses, protocols and intrusive signatures or characteristics in every network packet and report any suspicious pattern as a security violation.

Hence, Snort validates each network packet that migrates across its sensor against its rules and coded actions are invoked upon detection of event that matches any of the rules. Routinely, each suspicious event is further validated against the alert rules and if it is successful, it is subsequently (fig 1 below) validated against the log rules. At this point, an alert would be raised to notify the system administrators of the presence of such suspicious event on the network and simultaneously, the event will be logged into two audit repositories (log and database). These repositories are used as contingency approach and further for post-intrusive review. However, if a packet fails alert rules, Snort subsequently reject the event by invoking its pass rules. Logically, the more the detection rules are unable to differentiate repeated events from new activities on a network, the more the proportion of redundancy in the audit repositories.

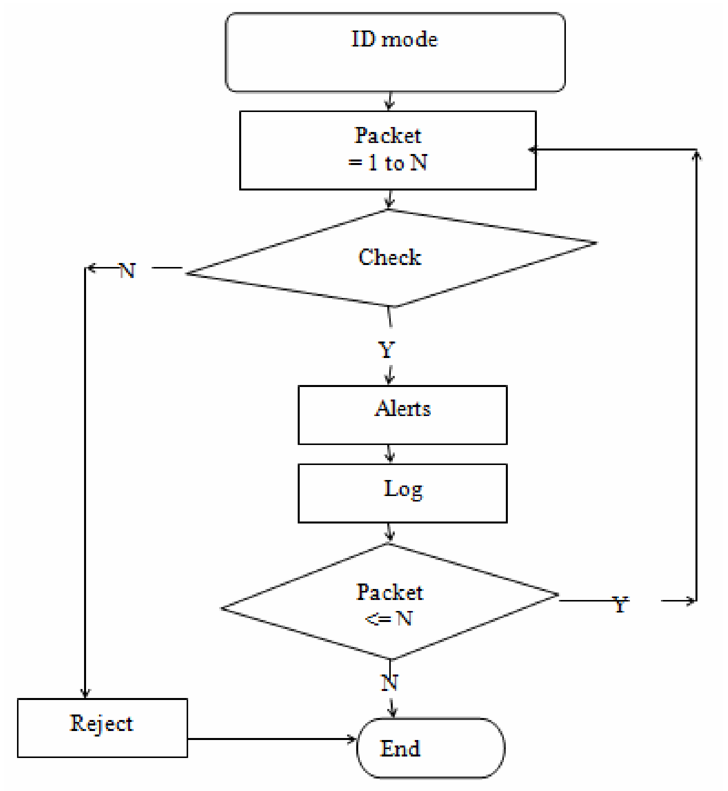

Fig. 1. Intrusion Detection (ID) mode

Selection criteria to reduce redundancy
The alerts of each audit repository are usually the same but in a different format. The alerts in the audit log are in printer text formats (prn) while those of the database are in relational formats.

```
[1] [**] [116:150:1] (snort decoder) Bad Traffic Loopback IP [**]
[2] [Priority: 3]
[3] 04/16-21:06:15.785691 127.93.72.86:23348 -> 131.84.1.31:4692
[4] TCP TTL:255 TOS:0x8 ID:14030 IpLen:20 DgmLen:40 DF
[5] ***A**** Seq: 0x7BE9C279  Ack: 0x0  Win: 0x4000  TcpLen: 20
[6] [**] [116:150:1] (snort decoder) Bad Traffic Loopback IP [**]
[7] [Priority: 3]
[8] 04/16-21:06:15.794429 127.192.221.148:23406 -> 131.84.1.31:21551
[9] TCP TTL: 255 TOS:0x8 ID:14088 IpLen:20 DgmLen:40 DF
[10]***A**** Seq: 0x7BE9C2B3 Ack: 0x0 Win: 0x4000  TcpLen: 20
```

Fig. 2. Darpa-2 alerts from audit log

Basically, each alert is described by series of attributes (fig 2 above) such as source address or attacking machine (source-IP and or source-ports), destination address or machine that is attacked (target IP and or target ports), Protocol, Timestamp, Time To Live (TTL), Ip length (IpLen), Type of Service (TOS), Ipflag, Check sum, description, ID, Generator's identity, etc. However, throughout this paper, we only adopted the *Source* and *Destination IP addresses, Protocol, Timestamp, TTL, IpLen, TOS and Ipflag*. We assumed that eight attributes would be sufficient enough to identify possible ways to lessen the proportion of redundancies in the audit logs without necessarily underestimate the attacks that they signify. The Time-to-Live (TTL) is the lifetime of a network packet on a network before it is discarded or returned to sender [10]. Hence, the TTL value of each packet gradually decreases [10] as the packet migrates towards its destination and the packet gets discarded at any point its TTL value reaches zero. The Type-of-Service (TOS) value determines the order [10] of precedence that network devices should treat each packet on the network. The Ipflag is used to classify each packet into fragmented and non-fragmented packets. The IpLen denotes the [10] length of individual Internet Protocol (IP) header while the protocols are message standard conventions [10]. Nevertheless, extensive discussions of these attributes are seen in [10] and [16].

4 Datasets and Methodology

The six categories of datasets that we have used to evaluate our method were chosen to cover realistic and synthetic attacks. Four of the datasets were standard evaluative datasets while the other two were from experimental and realistic networks. Each of the datasets was sniffed with Snort in intrusion detection and default modes to generate alerts that were automatically analyzed. We labeled a trace file of probing attacks that were simulated on our experimental network as TEST-DATA. The probing attacks contained Ping of death (POD), port scan, UDP, Xmas tree, SYN stealth scan, versions and O/S attacks. The attacks started on 2009-04-28 at 12:45:45 hour and lasted for 2009-04-29 at 15:21:38 hour. We define probing attacks as attacks that seek for vulnerabilities in target machines while non-probing attacks exploit vulnerabilities that are present in target systems. The UNI-DATA dataset was a trace file that was extracted from real-time monitoring of a segment of the university's networks. The dataset contained truncated traffics that were launched between 2009-06-21 at

14:41:55 hour and 2009-06-22 at 15 11:38:05 hour. Additionally, we extracted DARPA2000 [6] datasets (LLDoS 1.0 or first scenario) and LLDOS 2.0.2 (second scenario) and labeled them as DARPA-1 and DARPA-2 respectively. Both datasets were examples of DDoS attacks only that attacks in DARPA-1 were launched by a novice attacker within 5s on 2000-03-07 at 16:27:51 hour of the same day and stopped at 16:27:56 hour while DARPA-2 was launched by experienced attackers on 2000-04-16 at 21:06:15 hour for about 8s.

The fifth and sixth datasets were traces of DEFCON-8 and DEFCON-10 datasets [5]. DEFCON-8 dataset was port scan and fragmented packets that initiated buffer overflow attacks. The attacks were launched on 2000-07-28 at 16:39:03 hour through 2000-07-30 at 02:11:15 hour while DEFCON-10 dataset was bad packet, attacks on administrative privilege, FTP attacks via telnet protocol, ports scan and port sweeps and they were launched from 2002-08-03 at 00:57:17 hour to 2002-08-04 at 20:56:00.

A new taxonomy of alerts

There are numerous classifications of computer attacks or alerts [1] in recent literatures but they were not adaptable to our method of reducing the aforementioned problems. Hence, we reclassified them on the premise that alerts of each dataset can be perfectly categorized into unique and equivalent groups. Hence, we defined alerts that have closely related attributes as equivalent alerts while an alert that occurred once in the audit repository was regarded as unique alert. We also noticed some variability in these classifications that are determined by the attributes that we have considered in each case. For example, 2 alerts that are shown in fig (2) above could be classified as equivalent and they could as well be classified as unique alerts. For instance, they could be aggregated into a cluster of 2 unique alerts if they are aggregated by source-ip, ID, timestamp or a sequence of <Source-ip, ID, Timestamp>. Contrarily, the alerts can be transformed to form 1 unique alert if they are aggregated by their destination-ip and or TTL, TOS, IpLen, or a sequence of < Destination-ip, TTL, TOS, IpLen>.

Clustering-based analysis Method

We designed a simple clustering technique that used selection criteria to automatically reclassify alerts and eventually reduced the proportion of redundancy in each of the datasets to substantial quantity. Our technique was divided into *alerts-filtering* and *alerts-aggregation* stages as shown in fig (3) below.

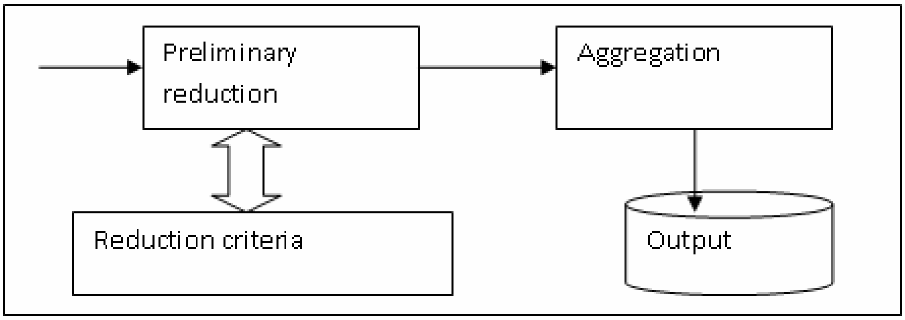

Fig. 3. Redundancy reduction process

Both processes were further implemented in C++ language to cluster alerts of the datasets. The reduction criteria were a collection of user-defined criteria with two flags. The first input flag only used Timestamp as selection criteria and at each of the time intervals of t=5mins, t=10mins, t=20mins, t=1hr and t=2hrs to cluster each group of the input data while the second input flag only used each of the selection criteria.

The input data to *alerts-filtering* stage was a sequence of audit alerts of each dataset together with an input flag while the final out of the entire model was sequence of unique alerts. For each dataset, *alerts-filtering* carried out preliminary reduction of the alerts using each of the user-defined reduction criteria and an input first flag. In this stage, all kinds of alerts in the dataset were identified and were clustered into sequences of unique and equivalent alerts. Also, the outputs were further passed to the next phase. In the second phase, a*lerts-aggregation* carried out intrusion aggregation. This phase processed all input data from previous phase and generated a cluster that contained sequences of unique alerts. This phase also totaled all unique alerts that were synthesized and the final output data were warehoused in an output repository. Hence, the system administrators then execute countermeasures to forestall attacks the alerts signify. In addition, the entire processes were repeated for all other datasets using the second flag and the results obtained in each experiment before and after we have applied our method are described in section (5) below.

To the best of our knowledge, our method was a simplified clustering method that automatically modeled system administrators that were saddled with the responsibilities of reducing intrusion redundancy in realistic world. Moreover, extensive investigations of the variability of these problems across six different datasets that represented realistic environment that we carried out were another uniqueness of our technique.

5 Results of Experiments

The experiments aimed to reduce redundancy and to substantiate the variations in the proportions of redundancies across several datasets. Also, Snort generated 67,469 alerts on TEST-DATA, 16,087 on UNI-DATA, 834 on DARPA-1, 816 on DARPA-2, 917,221 on DEFCON-8 and 5,372 alerts on DEFCON-10. In addition, fig 4 below indicates distributions of alerts when we clustered each dataset at respective time intervals while figures 5-8 indicate results that substantiated variability of redundancy reductions and clustering with the second flag (i.e. different selection criteria). The results in fig 4 also substantiated the evidence that all the datasets were reduced as the analyses time interval increased from 5minutes to 2 hours while in fig 6, the plotted points from left-hand side to the right-hand side represented TEST-DATA, DARPA-1, DARPA-2, DEFCON-8, DEFCON-10 and UNI-DATA respectively.

Also, the results in figs 5-7 have established the fact that attacks were bunched together within a very short time frame in DARPA-1 and DARPA-2 and occasionally in DEFCON-8 and TEST-DATA. These results have also indicated that attacks in the TEST-DATA were launched by 12 source machines against 8 targets while that of UNI-DATA involved 20 sources of attacks against 5 destination machines. Also, DARPA-1 and DARPA-2 attacks were respectively launched from 265 and 408 sources against a target. DEFCON-8 was launched from 75 machines against 126 destinations while DEFOCN-10 was launched from 21 sources against 27 targets.

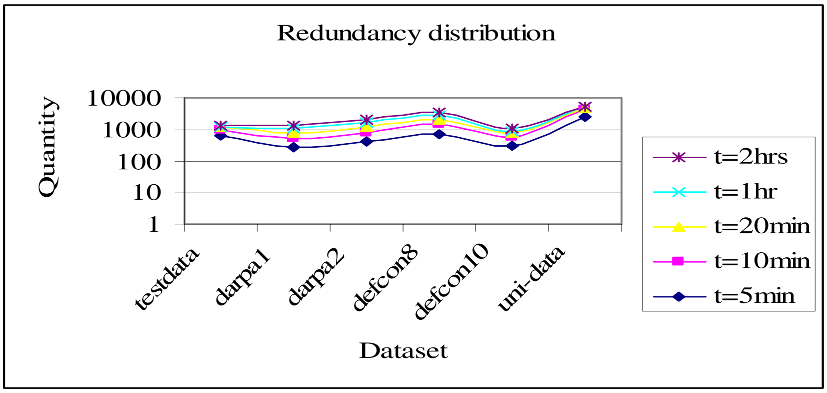

Fig. 4. <Time intervals>

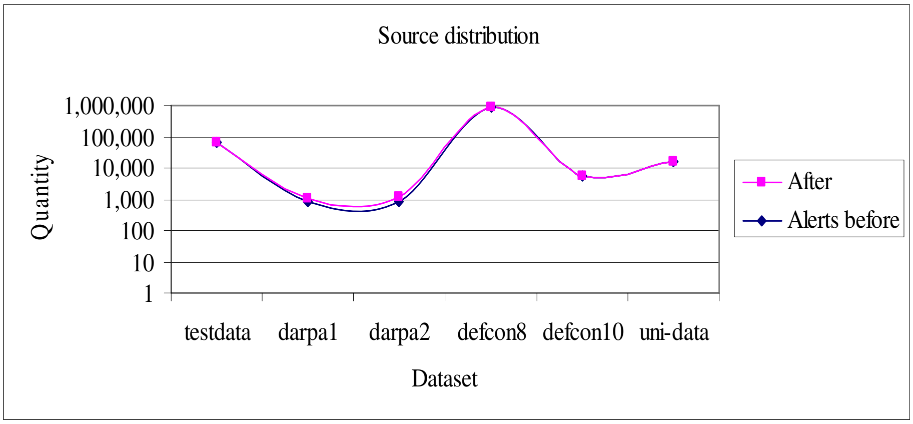

Fig. 5. <source-ip address>

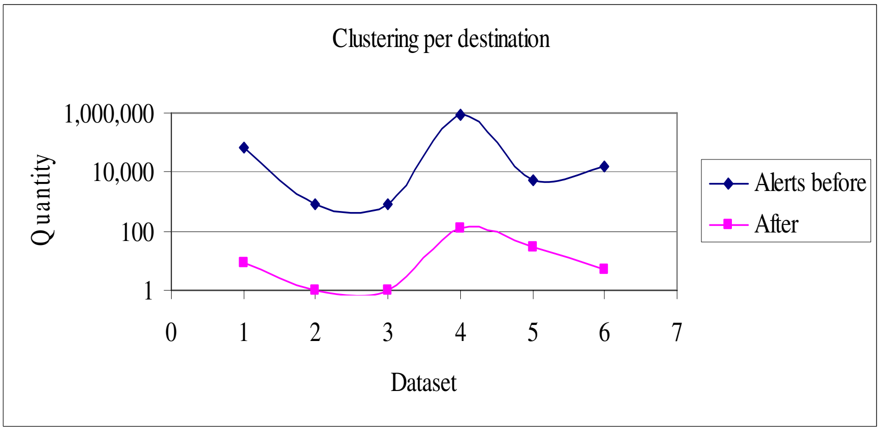

Fig. 6. < destination-ip address>

The results in fig 8 below have also indicated that attacks in DARPA-1 and DARPA-2 were launched with the same protocol while DEFCON-8 and TEST-DATA were launched with 3 protocols.

Effects of long sequential patterns on redundancies are shown in fig 8. TEST-DATA, UNI-DATA and DEFCON-10 datasets were near to linear reduction pattern, collapsed reduction was illustrated by DEFCON-8 dataset while DARPA-1 and DARPA-2 datasets collectively exhibited a reduction pattern that was intermediate between the previous two patterns.

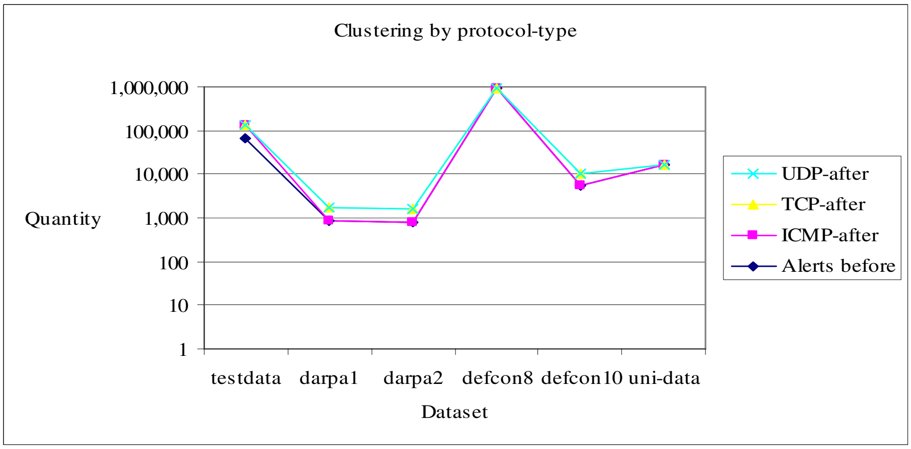

Fig. 7. < protocol-type>

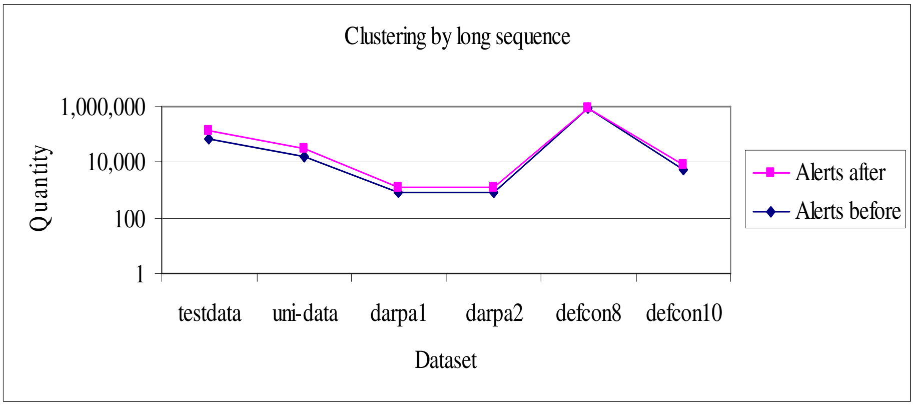

Fig. 8. < Source, destination, protocol, timestamp, TTL, IpLen, TOS and flag>

Also, we have observed that there was no selection criterion that was able to totally eliminate redundancies in all the dataset at the same time. In addition, the proportions of reductions of the datasets in each experiment were not the same.

6 Conclusion and Future Research Work

Intrusion redundancies are classical problems that are difficult to completely eradicate in corporate networks. As a result, some system administrators erroneously ignore them completely because of the complexity of separating true positives from tons of false positives. Since attackers often exploit these weaknesses to achieve malicious motives, we have critically investigated these problems across six kinds of datasets that covered probing and none-probing attacks that are commonly encountered on corporate networks. We have also demonstrated our automated clustering method that modeled system administrators that are saddled with the roles of logs analyses in realistic networks with the view to lessen these problems. Also substantiated was the difficulty in achieving total elimination of these problems in realistic networks using collection of filtering criteria. These experiments have also confirmed the variability of these problems across different categories of datasets and of course, our experiments have demonstrated that redundancies and attacks have some behavioural patterns.

Though our method has the tendency to have concurrently reduced some other fundamental problems of intrusion detection technology, however, we were unsure about the efficacy of such results. Also, we were not surely convinced that our results were not influenced by the kinds of attacks that were present in the datasets. Therefore, we plan to explore these issues in our future experiments.

References

1. Aleksandar, L., Vipin, K., Jaidep, S.: Intrusion detection: A survey. Computer Science Department, University of Minnesota (2005)
2. Alfonso, V., Keith, S.: Probabilistic alert correlation. In: Lee, W., Mé, L., Wespi, A. (eds.) RAID 2001. LNCS, vol. 2212, pp. 54–68. Springer, Heidelberg (2001)
3. Chyssler, T., Burschka, S., Semling, M., Lingvall, T., Burbeck, K.: Alarm Reduction and Correlation in Intrusion Detection Systems, Department of Computer Science, Linkoping University, Sweden (2004)
4. Cuppens, F., Miege, A.: Alert correlation in cooperative intrusion detection framework. In: Proceedings of IEEE symposium on security and privacy (2002)
5. Capture The Flag Contest-Defcon datasets (2009), `http://cctf.shmoo.com/data/`
6. DARPA.: Intrusion Detection Scenario Specific Data Sets (2009), `http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/2000data.html`
7. Debar, H., Wespi, A.: Aggregation and correlation of intrusion detection alerts. In: Proceedings of international symposium on recent advances in intrusion detection, Davis, CA, pp. 85–103 (2001)
8. Fatima, L.S., Mezrioui, A.: Improving the quality of alerts with correlation in intrusion detection. International Journal of Computer Science and Network Security 7(12) (2007)
9. Hartsein, B.: Intrusion Detection Likelihood: A Risk-Based Approach SANS Institute (2008)
10. Internet Protocol: Internetworking Technology overview (1999), `cisco.com/en/US/docs/../technology/handbook/Internet-Protocols.pdf` (2009)

11. Jan, N.Y., Lin, S.C., Tseng, S.S., Lin, N.P.: A decision support system for constructing an alert classification model. Journals of Expert Systems with Applications (February 2009)
12. Kabiri, P., Ghorbani, A.A.: A Rule-Based Temporal Alert Correlation System. International Journal of Network Security 5(1), 66–72 (2007)
13. Morin, B., Me, L., Debar, H., Ducass, M.: M2D2: A formal data model for IDS alerts correlation. In: Wespi, A., Vigna, G., Deri, L. (eds.) RAID 2002. LNCS, vol. 2516, pp. 115–137. Springer, Heidelberg (2002)
14. Ning, P., Cui, Y., Reeves, D.S.: Constructing Attack Scenarios through correlation of alerts, department of computer science, NC state University, USA (2002)
15. Paxson, V.: Considerations and Pitfalls for Conducting Intrusion Detection Research, International Computer Science Institute and Lawrence Berkeley National Laboratory Berkeley, California USA (2007)
16. Roesch, M.: Introduction to Snort, A lightweight Intrusion-Detection-System (2009), `http://www.seren.net/documentation/unix%20utilities/Snort.pdf`
17. Sadoddin, R., Ghorbani, A.: Network Security Laboratory, University of New Brunswick, Fredericton, Canada (2006)
18. Scarfone, K., Mell, P.: Guide to Intrusion Detection and Prevention Systems (IDPS), Recommendations of the National Institute of Standards and Technology, Special Publication 800-94, Technology Administration, Department of Commerce, USA (2007)
19. Urko, Z., Roberto, U.: Intrusion Detection Alarm Correlation: A Survey, Computer Science Department, Mondragon University, Gipuzkoa Spain (2004)
20. Wang, L., Liu, A., Jajodia, S.: Using attack graph for correlating, hypothesizing, and predicting intrusion alerts. Science Direct, pp. 2917–2933. Elsevier, Amsterdam (2006)
21. Valeur, F., Vigna, G., Kruegel, C., Kemmerer, R.A.: A Comprehensive approach to Intrusion Detection Alert Correlation. IEEE Transactions on Dependable and Secure Computing 1(3) (2004)
22. Xinzhou, Q., Wenke, L.: Discovering Novel Attack Strategies from INFOSEC Alerts, College of Computing Georgia Institute of Technology, Atlanta, GA 30332, USA (2004)
23. Yusof, R., Sulamat, S.R., Sahib, S.: Intrusion Alert Correlation Technique Analysis for Heterogeneous Log. International Journal of Computer Science and Network Security 8(9) (September 2008)