Investigating the problem of IDS false alarms: An experimental study using Snort

G.C. Tjhai, M. Papadaki, S.M. Furnell, N.L. Clarke

Key words: Intrusion Detection System, False Alarm, Snort

1 Introduction

IDS can play a vital role in the overall security infrastructure, as one last defence against attacks after secure network architecture design, secure program design and firewalls [1]. Although IDS technology has become an essential part of corporate network architecture, the art of detecting intrusions is still far from perfect. A significant problem is that of false alarms, which correspond to legitimate activity that has been mistakenly classed as malicious by the IDS. Recognising the real alarms from the huge volume of alarms is a complicated and time-consuming task. Therefore, reducing false alarms is a serious problem in ensuring IDS efficiency and usability [2].

A common technique for reducing the false alarm rate is by performing a tuning procedure. This can be done by adapting the set of signatures to the specific environment and disabling the signatures that are not related to it [8], based on the fact that some vulnerabilities exist in a particular OS platform only. However, although this can offer a means of reducing the number of false alarms, the procedure can also increase the risk of missing noteworthy incidents. Therefore, the tuning process is actually a trade-off between reducing false alarms and maintaining the security level. This often leaves administrators with the difficulty of determining a proper balance between an ideal detection rate and the possibility of having false alarms. Furthermore, tuning requires a thorough examination of the environment by qualified IT personnel, and requires frequently updating to keep up with the flow of new vulnerabilities or threats discovered [26].

The authors are with the University of Plymouth, UK
e-mail: {gina.tjhai,maria.papadaki,s.furnell,n.clarke}@plymouth.ac.uk

Please use the following format when citing this chapter:

Tjhai, G.C., et al., 2008, in IFIP International Federation for Information Processing, Volume 278; *Proceedings of the IFIP TC 11 23rd International Information Security Conference*; Sushil Jajodia, Pierangela Samarati, Stelvio Cimato; (Boston: Springer), pp. 253–267.

This paper investigates the problem of false alarms based upon experiments involving the popular open source network IDS, Snort [7]. A number of potential issues are presented along with the analysis undertaken to evaluate the IDS performance on real network traffic. Section 2 critically reviews background information on the false alarm problem, and provides a critical analysis of existing research in the area. The methodology of the experiment is presented in section 3. Section 4 provides the findings from the private dataset, followed by conclusions in section 5.

2 Related work

The problem of false alarms has become a major concern in the use of IDS. The vast imbalance between actual and false alarms generated has undoubtedly undermined the performance of IDS [9]. For that reason, the main challenge of IDS development is now no longer focusing only upon its capability in correctly identifying real attacks but also on its ability to suppress the false alarms. This issue had been extensively explored and analysed by Axelsson [2] based on the base-rate fallacy phenomenon. At present, a solution to restrain the alarms is not close at hand, as numerous aspects (e.g. attack features) need to be considered as the prerequisites to develop a better alarm reduction technique [12]. Developing an alarms suppressing technique is a continuing process rather than an isolated, one-off action. The number of reported attacks (and the associated IDS signatures), increases each month, with the consequence that tuning becomes a requirement throughout the lifecycle of an IDS.

Similar to our research, an evaluation had been carried out by Brugger and Chow [4] to assess the performance of traditional IDS, Snort. This evaluation had been conducted using the baseline Defense Advanced Research Projects Agency (DARPA) dataset 1998 against a contemporary version of Snort. Although the use of DARPA dataset had been strongly criticised in IDS evaluation, it still serves as a benchmark by allowing the comparison of IDS tools with a common dataset [16]. This assessment was performed to appraise the usefulness of DARPA as an IDS evaluation dataset and the effectiveness of the Snort ruleset against the dataset. In order to analyse Snort's alarms, a perl matcher script was used to report the false negative and positive rates; thus generating the Receiver Operating Characteristic (ROC) curve for a given set of attacks. Given the six year time span between the ruleset and the creation of the dataset, it was expected that Snort could have effectively identified all attacks contained in the dataset. Conversely, what they found instead was the detection performance was very low and the system produced an unacceptably high rate of false positives, which rose above the 50% ROC's guess line rate. This might be due to the fact that Snort has a problem detecting attacks modelled by the DARPA dataset, which focused upon denial of service and probing activities [13]. In particular, Snort is alleged to commonly generate a high level of false alarms [17] and the alarm rate reported in this evaluation was not creditable enough to prove Snort's false positive performance in a real network, which

might be much worse or much better. Moreover, the other experiments took place a few years ago, which means that Snort's performance may have changed since then. In view of that, our research decided to assess the performance of Snort on a more realistic data, as an attempt to critically evaluate the false positive issue of the system.

3 Experiment Description

In order to further explore the problem of false alarms faced by current IDS technology, an experiment was conducted to analyse and evaluate IDS alerts generated by real network traffic. In common with the earlier research referenced in the previous section, Snort, was chosen as the main detector. The reason for utilising Snort was due to its openness and public availability. Moreover, an investigation involving such a commonly used IDS can give an insight into the extent of the false alarm problem in other IDS systems as well.

A number of criticisms had been raised over DARPA dataset, questioning the use of synthetic data to picture a real world network as well as the taxonomy used to categorise the exploits involved in the evaluation [15]. Owing to these issues, our experiments involved the evaluation of Snort on both DARPA [23] and private dataset. However, this paper only presents an experiment using a private dataset, which was collected at University of Plymouth. The data was collected on a public network (100-150 MB/s network) over a period of 40 days (starting from May 17th to June 25th), logging all traffic to and from the University's web server. This includes TCP (99.9%) and ICMP (0.1%) traffic. The traffic collection was conducted with a conventional network analysis tool, tcpdump, and it involved the collection of the full network packet, including the packet payload. Although storing the full packet information significantly increased the storage requirements for the experiment, it was important to maintain this information for the validation and analysis of IDS alarms. The collected payload data was then further processed by Snort IDS in Network Intrusion Detection (NIDS) mode. It should also be noted that traffic containing web pages with the potential of having sensitive / confidential information was excluded from the packet capture, in order to preserve the privacy of web users. This was accomplished by applying filters on the traffic, prior to it being captured by tcpdump. Ngrep was used for this purpose [18].

The first stage of the experiment was to run Snort in NIDS mode, in its default configuration. This means that no tuning whatsoever was conducted. The aim of this phase is to investigate the extent of the false alarm problem with Snort's default ruleset. The next phase of the experiment involved the analysis of the same traffic, after tuning had been performed on Snort. A number of techniques were applied for the tuning, including setting up the event thresholds and adjusting Snort's rules [19]. A necessary requirement for this was the manual validation and analysis of alerts produced by Snort in the first phase, and identification of signatures that are prone to false alarms. The analysis of IDS alerts was supervised by a certified intrusion

analyst, and the front-end tool Basic Analysis and Security Engine (BASE) was utilised to assist the intrusion analysis process [3].

The analysis of alerts was supervised by a GIAC Certified Intrusion Analyst [10]. Once the alerts were manually verified, the result was presented in a ROC diagram; a graphical plot of Snort alarm generation. In order to reveal a clear picture of the false alarm problem, a ROC plot is preferable. This type of graph can demonstrate the trade-off between the ability to identify correctly between true positives and the risk of raising too many false positives. Unfortunately, there were no true negative (number of benevolent activities passed) and false negative (number of real attacks missed) value known in this analysis since real network traffic was used as the input dataset. As an alternative, the plot diagram is drawn to represent the actual number of false and true alarms instead of their alarms rate. This diagram provides a simple graphical representation of the false alarm problem, thus enabling the analyzer to easily comprehend the trend of false alerts. By demonstrating the graphical plot of false positive versus true positive, this approach visibly explains the criticality of the false alarm issue. The alarm rate is calculated as follows:

$$\text{False Alarm Rate} = \frac{\text{False Alarm}}{\text{Total Alarm}} \times 100$$

$$\text{True Alarm Rate} = \frac{\text{True Alarm}}{\text{Total Alarm}} \times 100$$

4 Results

The lack of knowledge or awareness about the complexity of network by IDS technology has led to the generation of excessive amount of false alarms. Generally, there are three possible alert types raised by the system, namely true positives (alerts from real attacks), false positives (legitimate activities thought to be malicious) and irrelevant positive (alerts from unsuccessful attacks or attempts [12]. The last two alerts are the main concerns in this study.

This section presents the results of the experiment. Figure 1 depicts the ROC plot for the overall result, which represents the general detection performance of Snort IDS. In order to create a simpler illustrative graph, which facilitates the comprehension of Snort's detection ability, the false and true positives values are presented in a proportion of thousands. The number of false positives generated is presented per unit time (per day) for the X-scale, while true positives are portrayed for the Y-scale. This diagram also represents the random guess (known as non-discriminatory line), which gives a point along a diagonal line from the left bottom (0,0) to the top right corner (10,10). This diagonal line divides the space into two domains; namely good and bad classification. Ideally, a good detection system should yield a point above the line, meaning the number of real alerts (true positives) triggered should not be exceeded by the number of false positives generated.

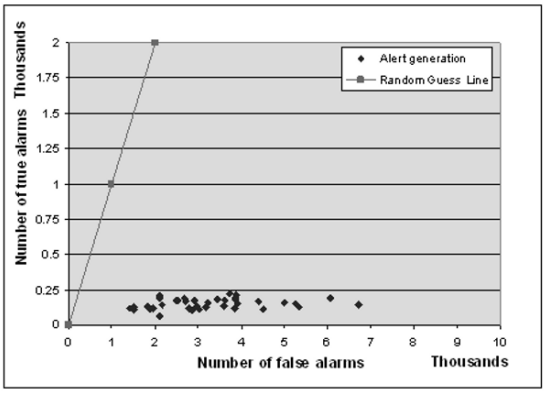

Fig. 1 Generation of alerts

Significantly, our research has also produced a similar result to that yielded in Brugger and Chow's evaluation. The number of false positives generated is massive. This indicates that the Snort's false positive performance on real network could be much worse than described in their evaluation.

This experiment focused on the analysis of false positive alarms, as opposed to other studies [14, 4], which were directed to explore the issue of false negatives. The main objective of this analysis is to merely provide a general view of the scale of false positives that may be generated by current IDS. The following subsections discuss this case in greater detail.

4.1 False Positives

A large volume of alerts, largely comprised of false alarms and irrelevant positives, drives the need to verify the validity of the alerts generated. Interestingly, apart from the false positives, our study reveals that some alerts were raised due to informational events, which merely occurred as a result of a network problem, not owing to the detection of real attacks. These types of alerts are known as irrelevant positives. Indeed, the unsuccessful attacks, or attempts that aim at an invincible target, might cause the system to generate such alarms.

Figure 2 provides a clear picture of the number of true and false alarms generated per day. In this context, it is obvious that the false alarms highly outnumbered the true alarms. Approximately 96% of alerts generated are asserted as false positives, while less than 1% of the total alerts are affirmed to be irrelevant positives. In order to make it simpler, irrelevant alarms are regarded as false positives alerts in this case since no immediate and crucial responses required from these events. By looking at the Snort alerts generated from the University's web server, most of the false positive alarms came from the category of web application activity. Table 1 shows

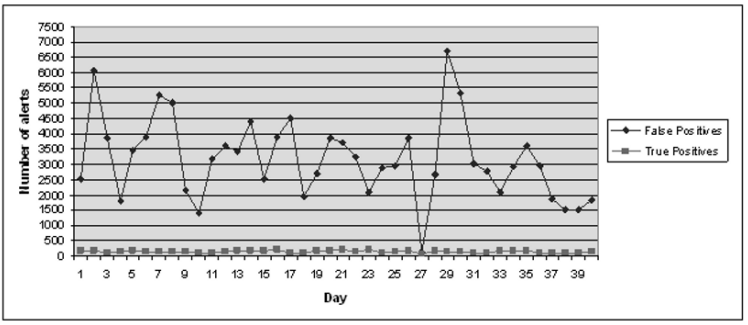

Fig. 2 Comparison between False Positive and True Positive alarms

a complete list of the Snort alerts triggered by the data. The first 3 alerts are the false positives alerts, which will be further investigated later in the subsubsections. The reason for focusing upon these alerts is due to the quantity generated, which is made up of more than 80% of total alerts raised by the system.

4.1.1 WEB-IIS view source via translate header

This event is categorized as web application activity, which targets the Microsoft IIS 5.0 source disclosure vulnerability [20]. Since Microsoft IIS has the capability of handling various advanced scriptable files such as ASP, ASA and HTR, the use of specialized header "Translate f" on HTTP GET request might force the web server to present the complete source code of the requested file to the client without being executed first by the scripting engine. In addition, this attack only works well if the trailing slash "/" is appended to the end of requested URL [5, 6].

Surprisingly, this single alert accounted for 59% of the total alerts. Therefore, approximately 1970 alerts were generated per day by this event. Although this event is deemed to be an attack that targets the Microsoft IIS source disclosure vulnerability, this could possibly be a false positive. Some applications, for example Web-based Distributed Authoring and Versioning (WebDAV) that make use of "Translate f" as a legitimate header, might cause this rule to generate an excessive amount of false alarms [25]. Moreover, WebDAV PROPFIND and OPTION methods also make use of this "Translate f" as a legitimate header to retrieve the information or properties of the resources identified by the Uniform Resource Identifier (URI) (nearly 96% of alerts generated by this event were not HTTP GET requests). Significantly, in this experiment, there is no alert generated by this signature, which required immediate action or indicated the occurrence of the real attack.

Table 1 Complete list of Snort alerts

No	Signatures	Total alerts
1	WEB-IIS view source via translate header	78865
2	WEB-MISC robots.txt access	30011
3	ICMP L3retriever Ping	10254
4	BARE BYTE UNICODE ENCODING	6392
5	POLICY Google Desktop activity	3258
6	SPYWARE-PUT Trackware funwebproducts mywebsearchtoolbar-funtools runtime detection	1873
7	ATTACK-RESPONSE 403 Forbidden	720
8	ICMP PING Cyberkit 2.2 Windows	651
9	DOUBLE DECODING ATTACK	504
10	ICMP Destination Unreachable Communication Administratively Prohibited	151
11	TCP Portsweep	124
12	SPYWARE-PUT Hijacker searchmiracle-elitebar runtime detection	80
13	WEB-MISC .DS_Store access	60
14	IIS UNICODE CODEPOINT ENCODING	49
15	WEBROOT DIRECTORY TRAVERSAL	35
16	SPYWARE-PUT Adware hotbar runtime detection - hotbar user-agent	27
17	WEB-IIS asp-dot attempt	26
18	TCP Portscan	19
19	SPYWARE-PUT Trackware alexa runtime detection	19
20	WEB-PHP IGeneric Free Shopping Cart page.php access	17
21	ICMP PING NMAP	17
22	ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited	13
23	WEB-CGI calendar access	11
24	MULTIMEDIA Quicktime User Agent Access	10
25	WEB-MISC intranet access	8
26	ICMP redirect host	8
27	ICMP PING speedera	7
28	SPYWARE-PUT Hijacker marketscore runtime detection	7
29	WARNING: ICMP Original IP Fragmented and Offset Not 0!	6
30	WEB-MISC WebDAV search access	5
31	WEB-FRONTPAGE /_vti_bin/access	5
32	Open Port	5
33	WEB-PHP remote include path	4
34	WEB-CGI formmail access	3
35	WEB-FRONTPAGE_vti_inf.html access	3
36	SPYWARE-PUT Trickler teomasearchbar runtime detection	2
37	WEB-PHP xmlrpc.php post attempt	2
38	WEB-CLIENT Microsoft wmf metafile access	2
39	WEB-MISC Domino webadmin.nsf access	2
40	OVERSIZE CHUNK ENCODING	2
41	ICMP Source Quench	2
42	WEB-PHP test.php access	2
43	WEB-PHP calendar.php access	1
44	WEB-PHP admin.php access	1

4.1.2 WEB-MISC robots.txt access

This event is raised when an attempt has been made to access robots.txt file directly [21]. Basically, robots.txt file is a file that is created to keep the web pages from being indexed by search engines. More to the point, this file provides a specific instruction and determines which part of a website a spider robot may visit. Interestingly, the problem is that the webmaster may detail sensitive and hidden directories or even the location of the secret files within the robots.txt file. This is considered extremely unsafe since this file is located in web server's document root directory, which can be freely retrieved by anyone.

Although this event is raised as the indicator of vulnerable information attack, there exists high possibility that all these alerts were raised due to legitimate activities from web robots or spiders. A spider is software that gathers information for search engines by crawling around the web indexing web pages and links in those pages. Robots.txt file is basically created to restrict the web spider from indexing pages that should not be indexed (e.g. submission pages or enquiry pages). As web indexing is regular and structurally repetitive, this activity tends to cause the IDS to trigger a superfluous amount of alerts. In this study, approximately 23% of total alerts (approximately 750 alarms per day) were accounted for by this web-misc activity. Given that all alerts generated from this event are owing to the activities of web spider, they are considered to be false positives. Significantly, this issue has apparently disclosed the drawback of Snort IDS in distinguishing legitimate activity from the malicious one; especially when it deals with the authorization or file permission.

4.1.3 ICMP L3retriever Ping

ICMP L3retriever Ping is an event that occurs when ICMP echo request is made from a host running L3Retriever scanner [22]. This type of ICMP echo request has a unique payload in the message, which significantly designates its distinctive characteristic. This traffic is considered to be an attempted reconnaissance since the attackers may use the ping command to obtain ICMP echo reply from a listening host. Surprisingly, in this analysis, quite a few alerts were generated from this event; contributing to 8% of the total alerts generated. This figure indicates that approximately 250 alerts were generated by this signature rule every day.

Considering the source IP address associated with these alerts, it is obviously clear that all ICMP requests were sent from the external hosts. Further investigation was conducted to critically analyse and discover if possible malicious events happened subsequent to the ICMP echo request. Surprisingly, there were no malevolent activities detected following the ICMP traffic. In addition, normal ICMP requests generated by Windows 2000 and Windows XP are also known to have similar payloads to the one generated by L3Retriever scanner [24]. Generally, this traffic is routine activities run by computer systems (especially Windows 2000 and XP systems) to communicate with their domain controllers or to perform network discov-

ery. In view of this issue and given that no suspicious output detected following these ICMP requests; these alerts were likely false positives.

4.2 Fine Tuning

False alarm for one system might not be an erroneous alert for other systems. For example, port scanning might be a malicious activity for normal users, but it is a legitimate activity if it is performed by a system administrator. Figure 3 shows an example of an event which triggered both false alarms and true alarms from the experiment. From the IDS's perspective, as long the activity's pattern match to the signature defined in the rule database, it is considered to be a malicious event. In view of this, fine tuning is exceptionally required to maintain the IDS's performance and enable the administrator to adapt the signature rule to the protected environment.

In order to optimize Snort's performance, fine tuning is necessary to reduce the number of alerts raised. Since only 3 signatures were tuned in this experiment, the false alarm rate accounted for 86.8% of total alarms after tuning was performed. Figure 4 depicts the ROC plots for the overall result after tuning was performed. Obviously, only less than two thousands alerts per alert type have been generated by Snort. In order to understand the effectiveness of fine tuning, the alarm rate between default and tuned Snort is presented in Figure 5. This figure does not seem particularly impressive but fine tuning did fare well on those signatures; reducing up to 90% of false alarms per signature, excluding WEB-MISC robots.txt access. The following subsections discuss tuning processes in more details.

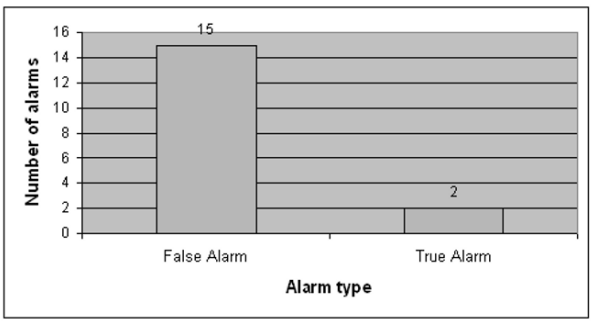

Fig. 3 "ICMP PING NMAP" event

4.2.1 WEB-IIS view source via translate header

Regarding the information disclosure vulnerability attack, Snort does not seem proficient enough to detect this type of event. The signature rule appears to be very loosely written, by searching for a particular string in the packet payload (in this case, "Translate: f"). Since the "Translate: f" is a valid header used in WebDAV application, as discussed previously, this rule tends to trigger a vast volume of alerts from the legitimate activities. Hence, tuning is needed to search for a more specific pattern of the attack signature.

As this attack is basically launched through HTTP GET request, searching for "GET" command in the content of analyzed packet can be a good start. Principally, this attack is performed by requesting a specific resource using HTTP GET command, followed by "Translate: f" as the header of HTTP request. In this case, a tuning can be performed by modifying the signature rule to:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"WEB-IIS view source via translate header";
flow:to_server,established; content:"GET|20|";content:
"Translate|3A| F"; distance:0; nocase; reference:arachnids,
305; reference:bugtraq,14764; reference:bugtraq,1578;
reference:cve,2000-0778; reference:nessus,10491;
classtype:web-application-activity; sid:1042; rev:13;)
```

The tuning process significantly reduced the number of alerts, with only 3463 generated by this rule as against 78865 alerts in the first case (i.e. without tuning). Significantly, this tuned rule had been proved to effectively reduce up to 95% of the initial false alarms from this event.

Although the tuning process had decreased the volume of alerts, there is still a possibility that those 5% alerts were false positives. Searching for GET command and the Translate f header is not effective enough to detect such attack. Putting trailing slash "/" at the end of requested URL to HTTP request for example could lead in the security bug [5]. Thus, matching the "/" pattern against the packet payload will be helpful. Unfortunately, this idea seems hardly possible to achieve. Snort does not have a specific rule option that can be used to match a specific pattern at a particular location.

As to Snort's signature, looking for an overly specific pattern of a particular attack may effectively reduce the false alarms; however, this method can highly increase the risk of missing its range. A skilful attacker can easily alter and abuse the vulnerability in various ways as an attempt to evade the IDS. This might lead to false negatives as a consequence.

4.2.2 WEB-MISC robots.txt access

Since accessing the robots.txt file is a legitimate request for Internet bots (web spiders), a subjective rule, which mainly focuses on the source IP addresses, is necessary to verify user authorization in accessing a certain file. This approach, however, seems to be hardly feasible to deploy. Of course, identifying all authorized hosts from their source IP addresses is impractical. There is an infinite number of IP addresses need to be discovered before the rule can be written. Indeed, lawfully allowing specific hosts to access certain file might increase the risk of having false negatives.

In this case, the only solution to suppress the number of false alarms generated is by using event thresholding [19]. As robots.txt access requests generate regular and repetitive traffic, a "limi" type of threshold command is the most suitable tuning in this case. Such a threshold configuration would be as follows:

```
threshold gen\_id 1, sig\_id 1852, type limit,
track by\_src, count 1, seconds 60
```

This rule logs the first event every 60 seconds, and ignores events for the rest of the time interval. The result showed that approximately 10% of false alarms had been effectively reduced. This indicates that only an insignificant number of false alarms can be reduced in this scenario. The frequency of fetching robots.txt files greatly depends on the web spider's policy. Hence, deploying event suppression and thresholding cannot effectively trim down the number of false alarms logged by the system. Additionally, suppressing the number of alerts generated can also create

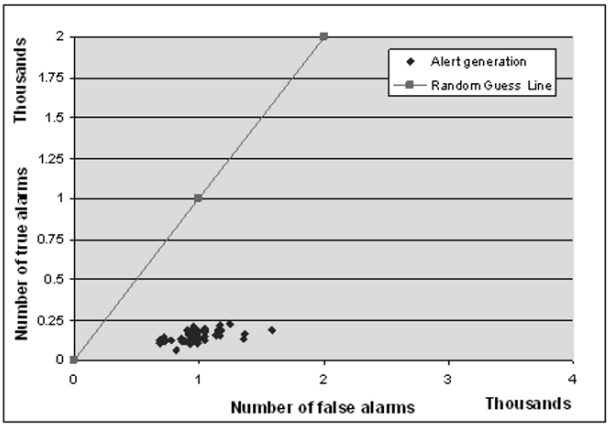

Fig. 4 Alerts generation after fine tuning

a possibility of ignoring or missing significant alerts. A malicious user can hide his/her action within the excessive number of alerts generated by using a spoofed address from web spider agent.

4.2.3 ICMP L3Retriever Ping

The only method that can be deployed to suppress the number of false positive triggered from this event is by applying event suppressing or thresholding command. Similar to the one applied to "WEB-MISC robots.txt access" signature, a threshold command is written to limit the number of alarms logged. Instead of using "limit" type of threshold command as previous signature, this rule utilized "both" type of command to log alerts once per time interval and ignore additional alerts generated during that period:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP
L3retriever Ping"; icode:0; itype:8; content:
"ABCDEFGHIJKLMNOPQRSTUVWABCDEFGHI"; depth:32; reference:
arachnids,311; classtype:attempted-recon; threshold: type
both, track by_src, count 3, seconds 60; sid:466; rev:5;)
```

Similar to the previous signature (robots.txt access), the threshold applied will not prevent the generation of false positives, but it will highly reduce the number of redundant false positives triggered. Importantly, the threshold is written to detect brisk ICMP echo requests by logging alerts once per 60 seconds after seeing 3 occurrences of this event.

The result showed that only 1143 alerts had been generated from this event in 40 days experiment data. This experiment has also proved that the event thresholding can successfully reduce up to 89% of the false alarms generated by this activity. Despite its ability in suppressing redundant alarms, the system is prone to missing stealthy ICMP requests (e.g. requests sent once every 60 seconds can be missed by the system).

5 Conclusions and Future Work

The issue of false positives has become a critical factor in determining the success of IDS technology. Not only must an IDS be accurate in detecting real attacks, but it must also have the ability to suppress the number of unnecessary alerts generated. The experiment presented in this paper has revealed a similar result to the work of Brugger and Chow [4]. Over a span of two years since their research was published, the issue of false positives remains a critical challenge for the current Snort IDS.

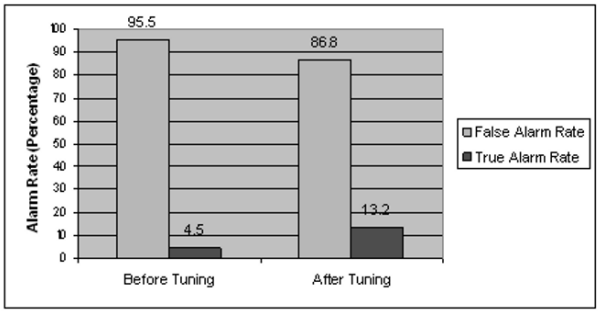

Fig. 5 Alarm rate before and after tuning

Obviously, Snort's performance does not look particularly remarkable as illustrated in Figure 1. The bottom right scattered plots demonstrate that the number of false positives largely overwhelms the number of true positives generated. Approximately 3,000 alerts had been generated per day, requiring manual verification to validate their legitimacy. Although the administrator can effectively distinguish the false and true positives from the alerts generated, the massive amount of false alarms triggered by one signature rule might cause the administrator to miss a malicious attack.

Principally, the overall effectiveness of Snort greatly hinges on the effectiveness of keyword spotting (i.e. matching the packet content to the signature rule). This has rendered the system prone to generating a superfluous number of false alerts. Interestingly, most of the rules looking for web traffic related attacks are loosely written and merely check for the presence of a particular string in the packet payload. This could trigger a large number of false alerts if a particular string is included in the content distributed by the web server. Hence, from this perspective, Snort is deemed not to be ideal enough to detect more complex attacks, which are not detectable by a pre-defined signature.

In view of these issues, an improvement is required to advance the performance of IDS technology. This involves developing an automatic alert verification, which no longer relies on human participation. Through this enhancement, it is expected that the number of false alarms can be substantially suppressed without increasing the possibility of false negatives. Also, a more intelligent system is required to help discover the logical relationship between alerts generated and to reveal the potential attack scenario; thus providing a better picture of the security issue to the system administrator. Given the complexity of systems and the ingenuity of attacks, an IDS will never be perfect, and there is still significant scope to enhance its performance.

Acknowledgements We want to thank Dr. Bogdan Ghita of University of Plymouth for his help in capturing the network traffic and for his support until the completion of this paper.

References

1. Allen J, Christie A, Fithen W, McHugh J, Pickel J, Stone E (2000) State of the Practice of Intrusion Detection Technologies. Available via Software Engineering Institute. http://www.sei.cmu.edu/publications/documents/99.reports/99tr028/99tr028abstract.html. Cited 9 January 2007
2. Axelsson S (2000) The Base-Rate Fallacy and the Difficulty of Intrusion Detection. ACM Transactions on Information and System Security 3(3), 186-205
3. BASE (2007) Basic Analysis and Security Engine (BASE) Project. Available via BASE Project. http://base.secureideas.net/. Cited 25 April 2007
4. Brugger ST, and Chow J (2005) An Assessment of the DARPA IDS Evaluation Dataset Using Snort. Available via UCDAVIS department of Computer Science. http://www.cs.ucdavis.edu/research/tech-reports/2007/CSE-2007-1.pdf. Cited 2 May 2007
5. Bugtraq (2007a) Microsoft IIS 5.0 "Translate: f" Source Disclosure Vulnerability. Available via Security Focus. http://www.securityfocus.com/bid/1578. Cited 9 June 2007
6. Bugtraq (2007b) Microsoft IIS WebDAV HTTP Request Source Code Disclosure Vulnerability. Available via Security Focus. http://www.securityfocus.com/bid/14764. Cited 9 June 2007
7. Caswell B and Roesch M (2004) Snort: The open source network intrusion detection system. Available via Snort. http://www.snort.org/. Cited 3 October 2007
8. Chapple M (2003) Evaluating and Tuning an Intrusion Detection System. Available online: SearchSecurity.com. http://searchsecurity.techtarget.com. Cited 1 November 2006
9. Chyssler T, Burschka S, Semling M, Lingvall T and Burbeck K (2004) Alarm Reduction and Correlation in Intrusion Detection Systems. Available via The Department of Computer and Information Science Linkopings Universitet. http://www.ida.liu.se/ rtslab/publications/2004/Chyssler04_DIMVA.pdf. Cited 15 June 2007
10. GCIA (2008) GIAC Certified Intrusion Analyst (GCIA). Available via Global Information Assurance Certification. http://www.giac.org/certifications/security/gcia.php. Cited 8 May 2007
11. Koziol J (2003) *Intrusion Detection with Snort*, 2Rev edition. Sams Publishing, United States of America
12. Kruegel C and Robertson W (2004) Alert Verification: Determining the Success of Intrusion Attempts, Proc. First Workshop the Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA 2004). Available via Department of Computer Science, University of California, Santa Barbara. http://www.cs.ucsb.edu/ wkr/publications/dimva04verification.pdf. Cited 19 May 2007
13. Lippmann RP, Haines JW, Fried DJ, Korba J and Das KJ (2000) The 1999 DARPA off-line intrusion detection evaluation. Computer Networks 34:579–595
14. Mahoney MV and Chan PK (2003) An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection. In Recent Advances in Intrusion Detection (RAID2003), Lecture Notes in Computer Science, Springer-Verlag 2820:220–237
15. McHugh J (2000) Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory. ACM Transactions on Information and System Security 3(4), 262-294
16. Mell P, Hu V, Lippmann R, Haines J and Zissman M (2003) An Overview of Issues in Testing Intrusion Detection Systems. NISTIR 7007. Available via National Institute of Standards and Technology. http://csrc.nist.gov/publications/nistir/nistir-7007.pdf. Cited 7 July 2007
17. Patton S, Yurcik W and Doss D (2001) An Archilles' Heel in Signature-Based IDS: Squealing False Positives in SNORT. Recent Advanced in Intrusion Detection (RAID), Univ. of California-Davis.

18. Ritter J (2006) Ngrep - network grep. Available via SourceForge.net. http://ngrep.sourceforge.net. Cited 30 June 2007
19. Snort (2007a) Event Thresholding. Available via Snort. http://www.snort.org/docs/snort_htmanuals/htmanual_2.4/node22.html. Cited 1 July 2007
20. Snort (2007b) WEB-IIS view source via translate header. Available via Snort. http://snort.org/pub-bin/sigs.cgi?sid=1042. Cited 9 June 2007
21. Snort (2007c) WEB-MISC robots.txt access. Available via Snort. http://www.snort.org/pub-bin/sigs.cgi?sid=1:1852. Cited 9 June 2007
22. Snort (2007d) ICMP L3retriever Ping. Available via Snort. http://www.snort.org/pub-bin/sigs.cgi?sid=1:466. Cited 13 June 2007
23. Tjhai GC, Papadaki M, Furnell SM and Clarke NL (2008) The problem of false alarms: Evaluation with Snort and DARPA 1999 Dataset. Submitted to TrustBus 2008, Turin, Italy, 1-5 September 2008
24. Web Server Talk (2005) L3Retriever false positives. Available via Web Server Talk. http://www.webservertalk.com/message893082.html. Cited 12 July 2007
25. WebDAV (2001) WebDAV Overview. Available via Sambar Server Documentation. http://www.sambar.com/syshelp/webdav.htm. Cited 20 June 2007
26. Zhou A, Blustein J, and Zincir-Heywood N (2004) Improving Intrusion Detection Systems Through Heuristic Evaluation. 17th Annual Canadian Conference on Electrical and Computer Engineering. http://users.cs.dal.ca/ jamie/pubs/PDF/Zhou+CCECE04.pdf. Cited 25 June 2007