A NEW REDUCTION METHOD OF INTRUSION DETECTION FALSE ALERTS USING SNORT

Ben Soh, Daniel Bonello
Department of Compugter Science & Computer Engineering, La Trobe Ubiversity
Bundoora, VIC, Australia 3083

Keywords: Intrusion Detection, False Alerts, Snort

Abstract: In this paper, we propose a new approach to reducing the high levels of false positives encountered when deploying an intrusion detection system using Snort in a real live networking environment. We carry out an analysis of the effectiveness of such method in different networking environments. We conclude that the level of false positives is reduced considerably with the introduction of our implemented pass rules and that the rates at which false positives are generated become manageable.

1 INTRODUCTION

Snort (M. Roesch, 1998) is a lightweight network intrusion detection tool, capable of preforming real-time traffic analysis and packet logging on IP networks. It can preform protocol analysis, content searching/matching and can be used to detect a variety of attacks and probes.

Snort uses a flexible rules language which syntactically provides an effective and manageable means of directly configuring which types of network traffic should be either passed or collected. Snort also provides a modular plug-in architecture which is used by the detection engine along with real time alerting capabilities through syslog, a user specified file, UNIX sockets or WinPopup messages to Windows clients using Samba's "smbclient". In this case study, Snort version 2.0 was used which monitors TCP, ICMP and UDP traffic.

We used a large network dispersed over many geographical sites within our metropolitan suburbs. The network consists of approximately 25 servers and over 400 client nodes, and provides an excellent means of which to deploy and test the Snort intrusion detection system (IDS) (J. McHugh et al., 200) (Z. Yanchao et al., 2001).

Snort was deployed intentionally to monitor internal traffic and not external traffic. The organisation firewall is only permitted to pass through SMTP traffic on port 25 from any external source. Internally, the network experiences a range of packet types from TCP, ICMP and UDP packets to IPX, ARP and SNMP traffic.

The organisation uses many common software packages such as Lotus Notes R5, Microsoft Exchange, Novelle Netware Administrator, Windows NT 4.0 Client and Server, Citrix Clients and an email filtering system.

2 MOTIVATION

Upon setting up Snort to initiate as a service and rebooting, it seems initially that traffic was either being bypassed all together, or that no actual rule triggering traffic was experienced. The rule set had been verified and included in the Snort configuration file. Although it didn't take Snort long to begin alerting on packets which exhibited matches to rules defined in the rule set.

Initially, it is quite hard to know what to expect from the first batch of alerted rules. Is the system actually being attacked the reported number of times? This is hardly unlikely.

After an initial 34 hour running period, 2604 alerts were available for observation through the console. Effectively, this has provided motive for the purpose of research. Leaving the system in this state would prove unmanageable.

Soh B. and Bonello D. (2004).
A NEW REDUCTION METHOD OF INTRUSION DETECTION FALSE ALERTS USING SNORT.
In *Proceedings of the First International Conference on E-Business and Telecommunication Networks*, pages 46-53
DOI: 10.5220/0001397300460053

We aim to (i) benchmark (M. Ranum, 2001) the organisation's network to achieve a better understanding of the networking traffic experienced by the Snort sensor; (ii) propose and test (N. Puketza et al., 1996) a new reduction method and verify their effectiveness in reducing the false alerts to acceptable manageable levels.

3 A NEW FALSE ALERT REDUCTION METHOD

Common practice has to be adopted in order to effectively navigate through the initial list of generated alerts. This would prove useful in dealing with the many new and sometimes highly numerous alerts which would be encountered later.

We propose the following common, basic practices:

1. Verify the source IP address and subnet from which the packet has originated from. In some cases, the source port is also observed and verified, making sure traffic of this type is permitted through the given port on the source host.
2. Apply the same procedure as above to the destination section of the packet, making sure traffic of this type is permitted to a particular destination on a given port.
3. Acknowledge the packet type and its place in the networking environment.
4. Research the rule definition and what networking conditions trigger the particular alert.
5. Understand the rule and how the subsequent alert definition is triggered by a packet.
6. Provide an alert summary to justify why the alert has been triggered.
7. Understand the alert severity.
8. Understand the amount of times the alert was triggered.

By applying these steps to each rule and then effectively to groups of likened alerts, it is possible to systematically work through the ever growing list of alerts, with the main goal in mind not to miss an alert which may in fact report an actual happening of malicious activity.

To conclude each alert, we have to decide whether the trigger is a false positive or a false negative. As the system had just been deployed it was expected that many if not all alerts were false positives. By analysing each source in depth, we are able to provide evidence of rule triggering events, which are unintentional and hence, do not exhibit malicious intent.

From the alerts database, a common, manual reduction method is used, which involves locating the original Snort rule within the Snort rule set and applying the necessary changes to it. The method then re-deploys the edited rule back into the Snort rule set and enabling it for processing.
For example, the following alert:

alert udp $EXTERNAL_NET any -> $HOME_NET 67 (content:"|01|"; offset:0; depth:1; byte_test:1,>,6,2; classtype:misc-activity;)

can simply be changed to:

alert udp !192.168.1.32/32, 192.168.1.0/24 any -> $HOME_NET 67 (content:"|01|"; offset:0; depth:1; byte_test:1,>,6,2; classtype:misc-activity;)

Here, the original rule has been edited so that the whole subnet of 192.168.1.0/24 is monitored with exceptions to a single source IP address of 192.168.1.32 where the rule options do not apply. This effectively bypasses 192.168.1.32 from matching the rule options and prohibits the false positives from being generated from this source. This flexible rule syntax would then allow all rules within the rule set to be edited this way, reducing the total number of false positives without having to terminate network services and resources. However, in this method, manually searching through the large Snort rule set to find the rule that requires editing is extremely time consuming (Y Qiao et al., 2002).

In this paper, we propose a new reduction method, which involves creating new rules without having to locate and edit existing rules. These new rules, called pass rules, are created in a separate rule set which are then part of the larger Snort rule set. Snort requires more configuration work in order to utilize these new pass rules, as it would be pointless to trigger on a packet and then pass it. Snort would then be configured to execute the pass rule set first, after which, would carry out all other rule processing.

Using the previous example, passing UDP traffic from source 192.168.1.32 which matches the rule options involved creating the rule below:

pass udp 192.168.1.32/32 any -> $HOME_NET 67 (content:"|01|"; offset:0; depth:1; byte_test:1,>,6,2; classtype:misc-activity;)

which informs Snort to pass traffic matching the rule type and options.

Therefore it is quite easy to create pass rules when required and which are stored locally in a separate rule set making the overall larger Snort rule set more organised and hierarchically based on the rules processing priority.

4 PERFORMANCE EVALUATION

The main objective here is to test trial the implemented reduction method at different stages experienced by the network. The organisation's network is known to experience different traffic workload during two distinctive periods. The peak period is between 8:00am and 5:30pm weekdays, while the off-peak period is between 5:30pm to 8:00am weekdays and includes all weekends.

During these periods, traffic is analysed as to understand the types of packets monitored by the sensor. Once this has been established, pre-scheduled SYN, FYN and fragmented SYN attacks will be launched at different times during the periods. This will provide a means of validating the sensors ability to detect valid malicious behaviour. The experiments are carried out with the implemented pass rules enabled, and then again with the rules disabled. Doing this provides a direct means to observe the effectiveness of the rules implemented.

4.1 Test Case Considerations

The purposes of test cases are to:

- establish a baseline of traffic conditions experienced by the network during the off-peak period;
- confirm pre-scheduled attacks are detected during the experienced traffic conditions;
- understand the types of packets flowing through the subnet;
- compare between enabled and disabled pass rules results.

A test case consideration is not to benchmark the network by generating high traffic loads for the sensor because this would produce misleading results. High traffic loads may very well be the prelude of malicious activity or constitute an actual attack. Other points which we consider include:

- Only traffic from one subnet is to be considered when benchmarking.
- Snort is only capable of monitoring three types of packets : UDP, TCP and ICMP.
- Conditions used for testing are real environment conditions, which the IDS will be subjected to.

During execution of each test case, the sensor will not be used for any other purpose so as not to hinder benchmarking results.

4.2 Analysing Reduction Methods through Test Cases

As mentioned previously, the main objective of the test cases is to measure the effectiveness and need for reduction methods of false alarms on the network. To achieve this, the network is benchmarked at peak and off-peak periods where pre-scheduled attacks are launched at different intervals during these periods. This process is done once with pass rules enabled and then once again with pass rules disabled.

Knowing exactly how many alerts are triggered by each launched, pre-scheduled attack, allowed for any missed attacks during each scenario to be easily detectable. A complete listing of test cases used is included in Appendix A.

4.3 Test Case Results and Analysis

For all test cases, the results are recorded and reported. Each test case is analysed thoroughly and so is each recorded field. Once all test cases are successfully executed, the results are compared.

Because of space constraints, we give a sample result only.

Test Case 1 Results

Test Case 1 Off-Peak		
Execution Condition : Pass Rules Enabled		
Characteristic	**Value Measured**	**Comments**
Packet Count	443,924	
Average Packet/sec	7.751	
Average Packet Size (bytes)	165.550	
Average Bytes/sec	1283.154	
Bytes of Traffic	73,491,646	
Total Dropped Packets	0	
Execution Time (seconds)	57274.24	

Packet Types : All Ethernet Frames		
Packet Type	**Extended Packet Type**	**% of Total Packets**
Logical – Link Control		**44.16**
	Spanning Tree Protocol	6.43
	Internetwork Packet eXchange	37.22
	CISCO Discovery Protocol	0.21
	NETBIOS	0.03
	Datagram Delivery Protocol	0.26
	AppleTalk	0.01
	Address Resolution Protocol (ARP)	**5.73**
	DEC Spanning Tree Protocol	**12.90**
Internet Protocol (IP)		**4.35**
	User Datagram Protocol	4.27
	Internet Group Management	0.02
	Transmission Control Protocol	0.02
	Internet Control Message Protocol	0.04
Internetwork Packet eXchange (IPX)		**32.83**
	Service Advertisement Protocol	14.66
	IPX Routing Information Protocol	1.08
	NETBIOS over IPX	7.41
	Simple Network Management Protocol	4.91
	Name Management Protocol over IPX	2.97
	Sequence Packet Exchange	1.80
Data		0.03

Alerts Detected (Non Pre-Scheduled Attacks)					
Alert Type	**Alert Signature**	**Alert Severity**	**Trigger Cause**	**Time/Date of Alert & Total Number of Alerts**	**Alert Diagnosis**
ICMP	alert icmp $EXTERNAL_NET any -> $HOME_NET any (itype: 0; classtype:misc-activity;)	Medium	A host has replied to an ICMP Echo request with an ICMP Echo reply with an invalid or undefined ICMP code	Time :17:35:21pm Date : 16th Sep This alert was reported 1 time.	False Positive

Execution Condition : Pass Rules Disabled		
Characteristic	**Value Measured**	**Comments**
Packet Count	389099	Ethereal Statistics: Print Screen 6.3 Kerio Traffic Graph: Traffic Graph 6.3
Average Packet/sec	7.21	
Average Packet Size (bytes)	157.59	
Average Bytes/sec	1135.52	
Bytes of Traffic	61317293	
Total Dropped Packets	0	
Execution Time (seconds)	53999.29	

Packet Types : All Ethernet Frames		
Packet Type	**Extended Packet Type**	**% of Total Packets**
Logical – Link Control		**46.05**
	Spanning Tree Protocol	6.92
	Internetwork Packet eXchange	38.58
	CISCO Discovery Protocol	0.23
	NETBIOS	0.04
	Datagram Delivery Protocol	0.28
	AppleTalk	0
	Address Resolution Protocol (ARP)	**9.31**
	DEC Spanning Tree Protocol	**13.87**
Internet Protocol (IP)		**3.92**
	User Datagram Protocol	3.9
	Internet Group Management	0.01
	Transmission Control Protocol	0
	Internet Control Message Protocol	0.01
Internetwork Packet eXchange (IPX)		**26.83**
	Service Advertisement Protocol	13.73
	IPX Routing Information Protocol	0.94
	NETBIOS over IPX	3.41
	Simple Network Management Protocol	5.29

	Name Management Protocol over IPX	1.52
	Sequence Packet Exchange	1.94
Data		**0.02**

Alerts Detected (Non Pre-Scheduled Attacks)					
Alert Type	**Alert Signature**	**Alert Severity**	**Trigger Cause**	**Time/Date of Alert & Total Number of Alerts**	**Alert Diagnosis**
Not detailed as pass rules are disabled and many non-scheduled attacks are expected					

During the off-peak experiments, both cases PRE (pass rules enabled) and PRD (pass rules disabled) produced traffic conditions which are easily maintainable with a 100MBit connection, 0.01MBit/sec and 0.009MBit/sec respectively.

With the PRE, only 1 non-scheduled attack alert was reported in 70.09MB of monitored data compared to 120 alerts reported with the PRD in 58.48MB of monitored data for the off-peak period. This proves that it is crucial to have pass rules implemented on known false positives.

The percentage of packets actually monitored by the sensor is 4.33% for PRE and 3.92% for PRD of the total packets flowing passed the sensor in the given execution time. These low percentages have limited our experimental work. Future releases of Snort are to accommodate more packet types such as ARP and IPX.

Peak analysis stretches the test cases to their limits and effectively tests the limitations of the systems specifications and networking connections. Through the PRE cases, 0.02Mbit/sec traffic was experienced through 68.05MB of monitored data with no false positives. PRD shows 0.013MBit.sec with 41.83MB of monitored data and 1094 extra false positives. This amount is quite considerable when maintaining an IDS on a daily basis.

The experiment also shows a significant amount of missed alerts based on the pre-scheduled attacks. 27 fragmented packets did not generate alerts and were effectively missed by the sensor. Overall, a total of 216271 packets were dropped, indicative of the low monitored data result of 41.83MB. The total amount of dropped data is approximately 36.1MB, based on the average packet size calculated throughout the duration of the execution. This has provided additional evidence showing the importance of enabling the pass rules during peak network traffic.

The percentage of packets monitored with PRE and PRD is 4.94% and 18.5% respectively. The higher percentage of monitored packets for the peak PRD test case is simply a result of the type of network usage conducted by the users of the subnet at the time.

5 CONCLUSION AND FUTURE WORK

We can see that the level of false positives is reduced considerably with the introduction of the implemented pass rules. The rates at which false positives are generated become manageable. We find that in peak periods, the PRD results show false negatives and dropped packets. Yet under the same conditions with PRE, there are no reported false negatives or dropped packets. So this has proven yet again the importance of maintaining and reducing the high levels of false positives, using our proposed method.

Future work on the current Snort IDS setup includes rendering the sensor as an inaccessible network resource. This has many benefits and must be considered as attacks on the sensor itself render the whole network vulnerable to undetectable attacks. A successful attack on a sensor could see pass rules edited or the service stopped allowing the attacker to roam free. There are two known ways of rendering a sensor inaccessible, the first via registry edits and the second through physically limiting the capabilities of the network cable (wiring).

REFERENCES

M Roesch, “Snort: The Open Source Network Intrusion Detection System”, http://www.snort.org, December, 1998.

J McHugh et al, “Defending yourself: the role of intrusion detection systems”, IEEE Software, September/October 2000.

Z Yanchao et al, “An immunity-based model for network intrusion detection”, Proceedings of International Conference on Info-tech and Info-net, Beijing, 2001.

M Ranum, “Experiences benchmarking intrusion detection systems”, NFR Security Technical Publications, http://www.nfr.com, December, 2001.

N Puketza et al, “A methodology for testing intrusion detection systems”, IEEE Transactions on Software Engineering, October, 1996.

Y Qiao and X Weixin, “A network IDS with low false positive rate”, Proceedings of the Congress on Evolutionary Computation, 2002.

APPENDIX A

Test Case 1 : Off Peak

Purpose of Test	**Description**	**Testing Period**	**Launch Test Attacks**
Test network traffic characteristics during the off-peak period. Test ability to detect attacks during this network traffic environment.	Monitor the traffic passing through the monitored subnet and report on various traffic characteristics. Also review total alerts generated.	5:00pm to 8:00am. A 15 hour execution period which sees the network in its off peak state.	Launch Test Case 3.1 with enabled pass rules. Launch Test Case 3.2 with disabled pass rules.
Report On			
Characteristic	**Value Reported**	**Packet Type**	**% of Frames**
Packet Count			
Average Packet/sec			
Average Packet Size			
Average Bytes/sec			
Bytes of Traffic			
Total Dropped Packets			
Execution Time (sec)			
Alert Reports (Non Pre-Scheduled Attacks)			
Alert Type	**Alert Signature**	**Alert Severity**	
Trigger Cause	**Alert Diagnosis**	**Total Alerts**	
Comments			
• Establish a baseline of traffic conditions experienced by the network during off-peak periods • Confirm pre-scheduled attacks are detected during the experienced traffic conditions • Understand the types of packets flowing through the subnet • Comparisons between enabled and disabled pass rules			

Test Case 2 : Peak

Purpose of Test	**Description**	**Testing Period**	**Launch Test Attacks**
Test Network Traffic Characteristics during peak period. Test ability to detect attacks during this network traffic environment.	Monitor the traffic passing through the monitored subnet and report on various traffic characteristics. Also review total alerts generated.	8:00am to 5:00pm. A 9 hour testing period which monitors the network during peak usage.	Launch Test Case 3.3 with enabled pass rules. Launch Test Case 3.4 with disabled pass rules.

Report On			
Characteristic	**Value Reported**	**Packet Type**	**% of Total Frames**
Packet Count			
Average Packet/sec			
Average Packet Size			
Average Bytes/sec			
Bytes of Traffic			
Total Dropped Packets			
Execution Time (sec)			

Alert Reports (Non Pre-Scheduled Attacks)		
Alert Type	**Alert Signature**	**Alert Severity**
Trigger Cause	**Alert Diagnosis**	**Total Alerts**