Efficient Assessment and Evaluation for Websites Vulnerabilities Using SNORT

Mohammad Dabbour, Izzat Alsmadi and Emad Alsukhni

Yarmouk University, Jordan

mohmmad.dabbour@ gmail.com, ialsmadi@yu.edu.jo, ealsukhni@yu.edu.jo

Abstract

An endless number of methods or ways exists to access illegally a web server or a website. The task of defending a system (e.g. network, server, website, etc.) is complex and challenging. SNORT is one of the popular open source tools that can be used to detect and possibly prevent illegal access and attacks for networks and websites. However, this largely depends on the way SNORT rules are designed and implemented. In this paper, we investigated in details several examples of SNORT rules and how they can be tuned to improve websites protection. We demonstrated practical methods to design and implement those methods in such ways that can show to security personnel how effectively can SNORT rules be used. Continuous experiments are conducted to evaluate and optimized the proposed rules. Results showed their ability to prevent tested network attacks. Each network should try to find the best set of rules that can detect and prevent most network attacks while at the same time cause minimal impact on network performance.

Keywords: *Network security, vulnerability, Intrusion detection systems, SNORT, vulnerability assessment, rule-based detection*

1. Introduction

Websites face an enormous amount of possible attacks through the Internet. Attackers may try to access a particular website for one of several possible reasons. The major reason behind such attacks includes trying to retrieve sensitive data for identity theft purposes. Websites can also be accessed for spam purposes. Spammers try to inject their links or codes in websites to get higher traffic or popularity and hence be more visible by users and search engines. Such market goal may also include trying to spy on users, their machines or websites and their search behavior in order to develop guided advertisements or marketing campaigns. Websites and machines can be also accessed by friends, relatives or lovers looking for personal sensitive information. They may be also accessed by disgruntled employee or ex-employee looking for a revenge for employer. Political or international crime reasons can also be a factor in attacking websites. Finally, some individuals may try to access websites to be popular among their rivals or to only use their skills and long available time.

Attackers search thoroughly those websites for possible vulnerabilities or weaknesses. They will then use such weaknesses to attack or access websites from. The game between attackers and defenders is an ongoing struggle and mission where defenders need always to be educated, proactive and ready. They feel happy that they are doing a good job in protecting their systems. On the other hand, attackers have less stress from some perspectives where they can choose their attack time and need only to know one vulnerability to access successfully a website. On the other hand, and since they are breaking the laws in doing so, they need always to be on the run and develop tools to hide their correct identity or they will be caught and prosecuted.

SNORT (www.SNORT.org) is a popular open source network intrusion prevention and detection software or system (IDS/IPS) developed by Sourcefire [1]. The functions of this application can be categorized into two main ones:

- IDS. SNORT is capable to list to incoming and outgoing packets. Based on user defined rules, intrusions or attacks can be detected.
- IPS. This is where rules can be used to prevent and stop attacks. SNORT can take act actively in permitting or preventing specific types of packets inwards or outwards.

In general rules can be defined to take one of several actions

1. Drop Packets in this rule are going to be dropped and prevented to continue its mission whether into the system or outward.
2. Log In such case, SNORT is only asked to log the information about this particular packet.
3. Allow In this case, packets are going to be allowed without any further action.
4. Alert In alert case, users are going to receive a warning or alert if conditions in a rule defined in this category are achieved.

Attackers continuously improve their methods of attacking. Their goal is to possibly avoid all detection techniques and to successfully intrude a website or system. On the other hand, a major challenge for intrusion detection tools is to compromise between correctly detecting intrusions and at the same time correctly detecting normal or regular packets and permit them. In the next section, we will first introduce the Receiver Operation Characteristics (ROC) or confusion matrix. This is a method used to evaluate prediction quality or accuracy.

Figure 1 shows the four attributes of the confusion matrix that combines between the prediction outcome and the actual values.

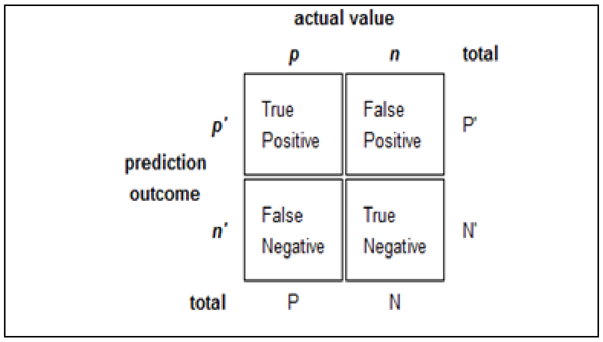

Figure 1. Confusion Matrix

In intrusion detection problem, True Positive (TP) means that the IDS will accurately detect an attack. True Negative (TN) means that the IDS will also permit regular traffic and detect it as regular, not attack. For an IDS to detect all intrusions correctly, TP and TN values should be 100 %. Their complements are false positive (FP), and false negative (FN) respectively [2].

Focusing only on one aspect and ignoring the rest may give wrong impressions about the spam detection abilities. For example, an IDS tool may have a high TP value (e.g. 95%)

which can be seen as a very reliable tool. However, this can be accompanied with a high FN value as well; which means that the tool is not good in detecting correct and incorrect cases. On the other hand, a tool that detects all normal traffic correctly (i.e. high TN), may also have high false alarms (i.e. high FP).

2. Related Works

In this section, we will survey some papers related to the subject of this paper: using SNORT rules for detecting and prevent network attacks.

Ibrahim and Lazim 2011 paper [3] focused on the issue of performance in Intrusion Detection Systems' (IDS) or Network IDS (NIDS) search in Ubicom network processor. In security in generation and in IDSs in particular, there are two criteria that are always in contradiction or conflict with each other. Those are security and performance. If we want to ensure a very high level of security, we have to implement a large number of rules each can test one vulnerability or security problem. However, such large number of rules will cause a delay in the network traffic. Authors' paper focused more on the hardware and their performance effects. Authors also discussed several examples of searching algorithms that can be used in IDSs such as: string matching, Native, Boyer Moore, etc. algorithms. They evaluated also the impact of SNORT rules patterns as well as payload lengths.

Kurundkar, *et al.*, 2012 paper [4] evaluated in general using SNORT as a NIDS with a case study applied on Linux Cento. In general, SNORT has two main modes: passive or sniffing in which the tool is a pure listener and active where rules can be defined and SNORT can take action based on those rules.

Riad, *et al.*, 2012 paper [5] discussed using Jquery application with SNORT to visualize Intrusion detection. Visualization can b extremely helpful in such cases since the amount of ongoing and outgoing data is huge. For a human who monitors such huge traffic observing or picking a problem can unlikely occur. Mixing visualization with roles can improve such ability such like using cameras in the airport with real time searching system looking for possible suspects in a huge database.

Gandhiand and Srivatsa paper [6] discussed also using SNORT for detecting and preventing network attacks. There are several types and classes of network attacks. In addition, detecting those attacks can be based on one of several methods such as: signature based, rule-based, etc.

Rani and Singh paper [7] evaluated performance issues with SNORT IDS. As expected, their small study showed that performance is dependent on the type of attack. This is since some network attacks are larger in size and/or need more time to process and judge.

Voznak and Safarik paper 2012 [8] discussed the vulnerability of SIP to Denial of Servoce (DoS) attacks. The evaluation approach combined SNORT with other system components and aimed at evaluating the system efficiency in detecting attacks. Efficiency is calculated based on some metrics such as: memory, CPU, and bandwidth usage or depletion. Applications such as NIDS are usually consume a significant amount of computer or network resources such as those mentioned earlier.

Mcareavey, *et al.*, paper 2011 [9] evaluated heuristically, inconsistency in results based on SNORT rules detection. They evaluates regular expression based rules written in SNORT. Rules based on SNORT can be defined in very simple or complex mechanisms. They can be as simple as specifying a specific port or IP address to alert when packets are coming or going from. They can be also very complex to include a description of a very complex pattern that may occur over several packets, many hours or days, etc. Problems arise specially when some of those rules contradict with each other. For example, the rules: log 212.33.233.44:8070 and drop 212.33.233.44:8070, because for the same data coming to, or going from, port 8070 two different decisions are supposed to be made, drop packet command will prevent those packets from completing their journey while the log command is supposed to archive them (In this specific scenario, log then drop maybe an option in some scenarios). There are several challenges associated with IDS tasks. The first challenge is to compromise between security and performance. The second challenge is to eliminate harmful packets to a maximum level and at the same time permit normal or regular packets with the least amount of irritation or hassle. The real challenge occurs as those 3 of 4 challenges compete at the same resources which means that with no doubt improving one aspect positively, will impact negatively the other ones.

Song, *et al.*, paper [10] tried to compare data from SNORT IDS and Honypot reports. Honypot systems are tools or applications that try to set traps for attacks to fall in. The paper collected a dataset for SNORT data with some important features such as: time, attack type, priority, source and destination addresses, source and destination ports, and IP protocol. They applied Support Vector Machine (SVM) algorithm for training and learning of the dataset packets classification with one of two possible classes (attack or normal). Results showed that results' integration from the two sources (i.e. Honeypot and IDS can significantly improve performance or accuracy.

Ashoor and Gore paper 2012 [11] focused on IDS misuse detection using multi agents architecture. Such multi agent combinational effort is expected to improve accuracy when information related to possible attacks come from different sources.

Beg, *et al.*, 2010 conducted a survey paper on IDSs [12]. The paper also discussed some of the Artificial Intelligent AI techniques that can be used to improve IDS detection and prevention accuracy. The paper also discussed some of the anomaly based techniques for detection such as: statistical models, machine learning, and knowledge based. The paper discussed also IDSs for High Performance Computing (HPC) environments where challenges can be higher due to the larger bandwidth and greater traffic speed.

Kumar, *et al.*, paper 2011 [13] discussed also IDS in HPC environment. The paper presented the evaluation for two pattern matching techniques to be used in the IDS environment. In one category, pattern matching techniques can be broadly divided into software and hardware based techniques. The paper discussed some of the detailed components required in hardware based pattern matching for IDS.

Victor, *et al.*, paper 2010 [14] discussed some of the proposed solutions to minimize false alarms in IDS detection. False alarms means that the IDS suggested for a packet or data to be harmful while in reality it is or was not. As mentioned earlier it is a challenge for IDSs to compromise between detecting all threats successfully and at the same time permitting all normal packets without eliminating any of them.

Fugate paper [15] discussed the issue of scalability in relation with performance in IDSs. The idea is that when IDSs rule-sets are small, system accuracy and performance can be acceptable. However, most of the proposed systems will have problem in real scenarios where significantly a large number of rule-sets is required for system protection.

3. Goals and Approaches

We proposed several examples of SNORT rules that can be used to detect some types of network attacks. We then evaluated those rules and their effectiveness on DVWA (http://www.dvwa.co.uk/). This is a vulnerable web application that is used for testing in several research papers. In all examples, we used the symbols [] to show the enclosed commands sent to SNORT.

3.1. Network Attacks

The goal of the following experiments is to introduce and explain several SNORT rules and then evaluate their ability to detect or sop network attacks. Focus will be on the following types of attacks: SQL injection, XSS, and command execution attacks

- **SQL Injection Attacks**

In this kind of attacks the attacker tries to execute an SQL query on a web application that contains this vulnerability. These queries could be used to retrieve data from the database or about the database server. The attacker by using SQL injection tries to get some sensitive information (e.g. getting a password from the website or create new users accounts or delete some users). This can eventually help attacker to compromise the web application or even the web server.

- **XSS (cross side scripting) Attack**

In this attack, the attacker tries to execute a Java script code on the website or to include HTML code inside the website source code. This code enables the attacker to start a possible phishing process or to perform some other possible attacks such as: man in the middle (MiM) attack.

- **Command Execution**

In this kind of attacks the attacker tries to execute operating system commands which may enable him/her to compromise the server and control the system.

After defining and analyzing attacks in those three types we will write some rules that will try to prevent these attacks depending on the nature of the attack.

3.2. Rules to Detect Network Attacks

- **Rule to Prevent SQL Injection**

For this type of attacks, we developed two rules to detect and prevent this kind of attacks.

1. The first rule tries to deal with an attack that contains single quotation (‘). In this type of attacks, the attacker uses this symbol to close the SQL condition parameter and start another one which is the attack itself. Following are some examples of this attack that we tested on DVWA:

[‘ or 1=1;#%27%20%6f%72%20%31%3d-%31%3b%23]

[‘union select user_id, password from users where user_id=10 or 1=1;#]

A more complex attack of this type can occur where the attacker tries using the like C programming language multi lines’ comments to make the attack signature unpredictable. We tried also this type and through the second rule, such attack was successfully detected.

[/*********/’/***********/or/******************/1=1/**********/;/*********/#]

[%2f%2a%2a%2a%2a%2a%2a%2a%2a%2a%2a%2a%2a%2a%2f%e2%80%99]

2. The attacker can finish the original, normal, query and start a new query which is the attack. This can be achieved through using the semicolon

[;select f_name,l_name from users where user_id=10;#]

3. Not all websites can be injectable by previous described attacks as most of the websites use the escape function which kills those attacks by changing the meaning of the meta characters. More than 65% of websites use those techniques (i.e. escape functions) to avoid SQL injection. However, this may increase the level of web application security but not make it uninjectable.

By the following query we can bypass this kind of escaping attacks.

[1 union select password,user_id from users where user_id=10 or 1=1;#]

Following is the first SNORT rule that can be possibly preferred to be used only on detection mode and not on prevention mode. This rule is written as the following:

```
[alert $EXTERNAL_NET any → $HOME_NET  $HTTP_PORT (msg:”[SQL Injection attack has
been detected--1]”; flow:to_server,established; pcre:”(((\?id=)(\?id\%3D))(\w*)((\')|(\%27)))/ix”;
classtype:web-application-attack; sid:10000015;rev:5; )]
```

2. The Second Rule:

```
[drop $EXTERNAL_NET any → $HOME_NET  $HTTP_PORT (msg:”[SQL Injection attack has
been detected--2]”;
flow:to_server,established;
pcre:”(((\?id=)|(\?id\%3D)).{0,}(\%3b)|(\;).{0,}((#)|(\%23)))/ix”;
classtype:web-application-attack; sid:10000016;rev:5; )]
```

The second rule is more reliable than the first one in its ability to detect such attacks with the least possible False Positive (FP) occurrences.

- **Rule to Prevent XSS Attacks**

Cross-site scripting (XSS) is a type of computer security vulnerability typically found in Web applications (such as web browsers through breaches of browser security) that enables attackers to inject client-side scripts into Web pages viewed by other users. A cross-site scripting vulnerability may be used by attackers to bypass access controls such as: the same origin policy. Cross-site scripting carried out on websites accounted for roughly 80.5% of all security vulnerabilities documented by Symantec as of 2007.

In this type of attacks, the attacker inserts on the vulnerable Web application a malicious script. This script can be converted to its equivalent hexadecimal code.

This attack usually starts with a script tag that is considered as a signature for this attack type.

This attack includes HTML insertion which are injected to the web site page and sometimes asking the user to log in by a malicious form.

We have evaluated several types of XSS attacks. For example:

[<script>alert, this is a testing attack.

If the page accepted this attack then it is vulnerable!")</script>]

From the two SNORT rules that we defined for XSS attacks, this attack was detected by the first rule but not in the second rule.

The same attack can be converted to hexadecimal code and applied:

```
[%3c%73%63%72%69%70%74%3e%61%6c%65%72%74%28%e2%80%9c%54%68%69%73%20%69%73%20%61%20
%74%65%73%74%69%6e%67%20%41%74%74%61%63%6b%20%69%66%20%74%68%65%20%70%61%67%65%20%61
%63%63%65%70%74%65%64%20%74%68%69%73%20%61%74%74%61%63%6b%20%74%68%65%6e%20%69%74%20
%69%73%20%76%75%6c%6e%65%72%61%62%6c%65%21%e2%80%9d%29%3c%2f%73%63%72%69%70%74%3e]
```

The second example of XSS attack is injecting the page with html tags.

```
[<a href="http://goToMysite.com"> This is a test text </a>]
```

Based on the two rules that we defined for this attack type, we can see that the first rule can detect attacks that does not contain hexadecimal values inside the script while the second rule can detect attacks that contains hexadecimal. We then combined both rules together to produce a new rule that can detect both kinds of signatures. The first rule was able to detect the first described XSS attack and the second rule was able to describe the second attack.

XSS rule 1:

```
[alert $EXTERNAL_NET any → $HOME_NET  $HTTP_PORT (msg:"This is a test text";
flow:to_server,established;
pcre:"(((\%3e\%3c)|(\%3c))(\w*)([.]*|([\n]*.[\n]*)|([\r]*.[\r]*)(\w*)(\%3e))/ix";
classtype:web-application-attack; sid:10000016;rev:5; )]
```

XSS rule 2:

```
[alert $EXTERNAL_NET any → $HOME_NET  $HTTP_PORT (msg:"This is a test text";
flow:to_server,established;
pcre:"(((\%3e\%3c)|(\%3c))((\%[0-9a-f]+)+)(\%3e))/ix";
classtype:web-application-attack; sid:10000016;rev:5; )]
```

XSS third combined rule:

```
[alert $EXTERNAL_NET any → $HOME_NET  $HTTP_PORT (msg:"This is a test text";
flow:to_server,established;
pcre:""(((\%3e\%3c)|(\%3c))((\w*)([.]*|([\n]*.[\n]*)|([\r]*.[\r]*)(\w*))|((\%[0-9a-f]+)+)(\%3e))/ix";
classtype:web-application-attack; sid:10000016;rev:5; )]
```

As we can see XSS rules defined depend on the "<" and ">" signature.

After the experimental analysis using DVWA experimental web application, we can discover that this kind of attacks can happen on pages that dynamically show some values on the pages.

We found out also that even though the SQL injection attacks are usually related to XSS attacks where the rules may need to be combined to guarantee protection against those two attacks together. Customizing a rule for only XSS attack may still be vulnerable to SQL injection attacks.

However, the tags “<, or “>” should be prevented from input fields as first no name (e.g. person, animal, country, etc.) will contain them and yet adding them to an input field can be part of XSS or SQL injection attack. Based on the way, we defined rules 1 and 3 above, they will alert or prevent such type in this case.

- **Rules for Command Execution Attacks**

In the DVWA vulnerable web application, to test command execution attacks, we tried to execute system commands. The web application asks the user to insert an IP address to ping the host holding the IP address. This can be implemented as the following in order to execute such command: Notice the three used symbols (;, | and &).

```
[133.212.23.34;cmd]
[133.212.23.34|cmd]
[133.212.23.34&cmd]
[;cmd]
[|cmd]
[&cmd]
[X;cmd]
[X|cmd]
[X&cmd]
```

X is any character or sum of characters. If we increased the security level to medium we can bypass it by adding additional meta characters. We will however, demonstrated rules to prevent such intrusions.

If the attacker tried to use the equivalent hexadecimal codes, the rule will also prevent it.

- **Command execution detection and prevention rule**

We evaluated several versions of the rule below and came up to the optimized version written below. Tests showed that it can detect and prevent all evaluated command execution possible attacks.

```
[drop $EXTERNAL_NET any → $HOME_NET  $HTTP_PORT
(msg:”[Command Excution Attack]”;
flow:to_server,established;content:”ip”;
pcre:”
((ip=((\w*)|(\d*)|(\D*)|(.*))((\%3b)|(\;)((\w*)|((\d*)|(\D*)|(.*))|((\%7C)|(\|))|((\%26)|(&)[^submit])(\w+)
[&submit]))/ix”;classtype:web-application-attack;
sid:10000016;rev:5; )]
```

4. Conclusion

Achieving a high level of security in the network Internet globally open environment can be challenging. There are several types of network attacks that can harm web applications and servers. In this paper, we focused on three types of attacks: SQL injection, XSS and command execution.

Using SNORT IDS system, and DVWA vulnerable web application for testing, we defined several examples to simulate those types of attacks. We then wrote and evaluated SNORT rules that can detect and prevent such attacks. Those rules were optimized based on several frequent repeated processes.

We believe however that this is a continuous effort for finding the smallest number and at the same time most effective set of rules that can be included in the IDS to achieve a high reliable security with minimal impact on performance.

References

[1] Sourcefire Incorporation, http://www.sourcefire.com/about-us.
[2] C. Castillo, D. Donato, A. Gionis, V. Murdock and F. Silvestri, "Know your neighbors: Web spam detection using the Web topology", Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, **(2007)**, pp. 423-430.
[3] Q. Ibrahim and S. Lazim, "Applying an Efficient Searching Algorithm for Intrusion Detection on Ubicom Network Processor", International Arab Journal of e-Technology, vol. 2, no. 2, **(2011)** June.
[4] G. D. Kurundkar, N. A. Naik and S. D. Khamitkar, "Network Intrusion Detection using SNORT", International Journal of Engineering Research and Applications (IJERA), vol. 2, Issue 2, **(2012)** March-April, pp. 1288-1296.
[5] A. El - Din Riad, I. Elhenawy, A. Hassan and N. Awadallah, "Using Jquery with SNORT to Visualize Intrusion", IJCSI International Journal of Computer Science Issues, vol. 9, Issue 1, no. 3, **(2012)** January.
[6] M. Gandhiand and S. K. Srivatsa, "Detecting and preventing attacks using network intrusion detection systems", International Journal of Computer Science and Security, vol. 2, Issue 1, **(2008)**, pp. 49-60.
[7] S. Rani and V. Singh, "SNORT, an open source network security tool for intrusion detection in campus network environment", International Journal of Computer Technology and Electronics Engineering (IJCTEE), vol. 2, Issue 1, **(2012)** February.
[8] M. Voznak and J. Safarik, "DoS Attacks Targeting SIP Server and Improvements of Robustness", International Journal of Mathematics and Computers in Simulation, vol. 6, Issue 1, **(2012)**.
[9] K. Mcareavey, W. Liu, P. Miller and K. Mu, "Measuring Inconsistency in a Network Intrusion Detection Rule Set Based on SNORT", International Journal of Semantic Computing, vol. 5, no. 3, **(2011)**, pp. 281-322.
[10] J. Song, H. Takakura, Y. Okabe and Y. Kwon, "Correlation Analysis Between Honeypot Data and IDS Alerts Using One-class SVM", Intrusion Detection Systems, In Tech, **(2011)** March, pp. 173-193.
[11] A. Ashoor and S. Gore, "Misuse Detection System based- SNORTrules: JESS Using Multiagents", International Journal of Scientific & Technology Research, vol. 1, Issue 3, **(2012)** April.
[12] S. Beg, U. Naru, M. Ashraf and S. Mohsin, "Feasibility of Intrusion Detection System with High Performance Computing: A Survey", International Journal for Advances in Computer Science, vol. 1, Issue 1, **(2010)** December.
[13] P. Kumar, V. Bhaskar and B. R. Reddy, "Multi-Gigabit Pattern for Data in Network Security", International Journal of Computer & communication Technology, vol. 2, Issue VIII, **(2011)**.
[14] G. J. Victor, M. S. Rao and V. CH. Venkaiah, "Intrusion Detection Systems - Analysis and Containment of False Positives Alerts", International Journal of Computer Applications (0975 – 8887), vol. 5, no. 8, **(2010)** August.
[15] S. J. Fugate, "These go to eleven: Cranking up the knobs on IDS scaling performance", The University of New Mexico Department of Computer Science, 8th Student Conference, **(2004)** April 24, pp. 120-125.

Authors

Mohammad Dabbour

Mohammad Dabbour is a recent bachelor degree graduate in computer information systems from Yarmouk University in Jordan with distinguished skills and grades. He currently works at Yahoo, Jordan. His research interests are largely in security networks, and applications.

Izzat M. Alsmadi

Izzat Alsmadi is an associate professor in the department of computer information systems at Yarmouk University in Jordan. He obtained his Ph.D. degree in software engineering from NDSU (USA), his second master in software engineering from NDSU (USA) and his first master in CIS from University of Phoenix (USA). He had a B.sc degree in telecommunication engineering from Mutah University in Jordan. He has several published books, journals and conference articles largely in software engineering and information retrieval fields.

Emad Alsukhni

Emad Alsukhni is an assistant professor in the department of computer information systems at Yarmouk University in Jordan. He got his B. Sc. and master degree in computer information systems from Yarmouk University and PhD degree from Ottawa University in Canada in 2010. His research interests focus in distributed systems and evaluation.