SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks. (2010)
Published online in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/sec.186

Updating snort with a customized controller to thwart port scanning

Wassim El-Hajj[1*,†], Hazem Hajj[2], Zouheir Trabelsi[1] and Fadi Aloul[3]

[1]*College of Information Technology, UAE University, UAE*
[2]*Faculty of Engineering and Architecture, American University of Beirut, Lebanon*
[3]*Department of Computer Engineering, American University of Sharjah, UAE*

Summary

Wired and wireless networks are being attacked and hacked on continuous basis. One of the critical pieces of information the attacker needs to know is the open ports on the victim's machine, thus the attacker does what is called port scanning. Port scanning is considered one of the dangerous attacks that intrusion detection tries to detect. Snort, a famous network intrusion detection system (NIDS), detects a port scanning attack by combining and analyzing various traffic parameters. Because these parameters cannot be easily combined using a mathematical formula, fuzzy logic can be used to combine them; fuzzy logic can also reduce the number of false alarms. This paper presents a novel approach, based on fuzzy logic, to detect port scanning attacks. A fuzzy logic controller is designed and integrated with Snort in order to enhance the functionality of port scanning detection. Experiments are carried out in both wired and wireless networks. The results show that applying fuzzy logic adds to the accuracy of determining bad traffic. Moreover, it gives a level of degree for each type of port scanning attack. Copyright © 2010 John Wiley & Sons, Ltd.

KEY WORDS: intrusion detection system; fuzzy logic; port scanning; snort

1. INTRODUCTION AND RELATED WORK

Nowadays, using computers and computer networks in all communities all over the world has made computer network security an international priority. Because, it is not feasible to build a secure system with no vulnerabilities, intrusion detection becomes an important area of research.

An Intrusion Detection System (IDS) is an automated system designed to detect malicious attacks on computer systems through the Internet. The main aim of IDS is to protect the availability, confidentiality, and integrity of critical networked information systems by identifying preferably in real time, unauthorized use, misuse, and abuse of computer systems [1,2].

A typical IDS consists of three functional components [3]: an information source, an analysis

*Correspondence to: Wassim El-Hajj, College of Information Technology, UAE University, UAE.
†E-mail: welhajj@uaeu.ac.ae

engine and a decision maker. The information source provides a stream of event records. This component can also be considered as an event generator. The analysis engine finds signs of intrusions. There are two basic approaches used to detect intrusions: misuse detection and anomaly detection. A decision maker applies some rules on the outcomes of the analysis engine, and decides what reactions should be done based on the outcomes of the analysis engine [4].

As mentioned earlier, the analysis engine used two basic approaches: misuse detection and anomaly detection. Misuse detection attempts to recognize attacks that follow a certain intrusion pattern. Such patterns are stored in the form of signatures in the database. Whenever a certain pattern matches a signature in the database, an attack warning is issued. These patterns have been recognized and reported by experts, but these systems are vulnerable to attackers who use new patterns of behaviors that cannot be detected by the system. Anomaly detection, on the other hand can be identified by recording unusual behavior of operations. An anomaly is something out of the ordinary, e.g., abnormal network traffic which is actually caused by unknown attacks. An anomaly detection system models normal behavior and identifies a behavior as abnormal (or anomalous) if it is sufficiently different from known normal behaviors [5].

The main work of building an anomaly IDS is to build a classifier which can classify normal event data and intrusion event data from an original data set. In Reference [6], the authors presented an anomaly detection method by using a Hidden Markov Model to analyze the trace of system calls coming from a UNIX system. In Reference [7], the authors established an anomaly detection model that integrated the association rules and frequency episodes with fuzzy logic to produce patterns for intrusion detection. In Reference [8], the authors developed an anomaly IDS combining neural networks and fuzzy logic. In Reference [9], the authors applied genetic algorithms to optimize the membership function for mining fuzzy association rules.

Although the work presented in the above research work makes significant contributions, it still has some flaws. Some of the above research work uses artificial intelligence techniques on anomaly intrusion detection, but most of their methods depend on static input and are not integrated in practical IDSs such as Snort. So, the practicality of the suggested method cannot be tested in real life.

In this paper, we update Snort by integrating it with a customized Fuzzy Logic controller. We call the new system 'Fuzzy Based Snort (FB-Snort)'. The aim behind this merge is to better detect port scanning and to reduce the false negative and false positive alarms. Our choice for using Fuzzy Logic was based on two main reasons: (1) no clear boundaries exist between normal and abnormal events, (2) fuzzy logic rules help in smoothing the abrupt separation of normality and abnormality (anomaly).

Our strategy starts by finding the normal traffic from abnormal traffic using Snort. Then, we pass some chosen parameters (section IV) to the Fuzzy Logic controller to get one unique parameter. This parameter decides whether an attack exists or not. As a result, FB-Snort reduces the false positive and the false negative alarms.

This paper is organized as follows: Section 2 gives a background about Fuzzy Logic, Snort, and port scanning. It also presents the motivation behind suggesting FB-Snort. Section 3 explains FB-Snort architecture. Section 4 presents the fuzzy logic controller input parameters and their significance. Section 5 discusses the fuzzy logic controller. Section 6 presents the experimental results and section 7 concludes the paper and discusses future work.

2. BACKGROUND AND MOTIVATION

In this section, we present a brief background on Fuzzy Logic, Snort, and port scanning and present the motivation behind our work. Fuzzy logic, a widely deployed technology for developing sophisticated control systems [10–12], provides a simple way to get definite precise conclusion and solutions based on unclear, imprecise, ambiguous, or missing input information.

Figure 1 shows the steps that the fuzzy logic controller is composed of Reference [13]. The steps can be summarized as follows: (1) receiving of one or more input values representing the measurements of the parameters to be analyzed or aggregated. (2) Subjecting the input values to fuzzy If-Then rules. The rules can be expressed in plain language words, for example, if a person is tall, back-pain is high. (3) Averaging and weighting the resulting outputs from all the individual rules into one single output decision. (4) Defuzzification of the output to get a crisp value between 0 and 1.

In general, two major steps are needed to develop the fuzzy logic controller: (1) define membership functions

DOI: 10.1002/sec

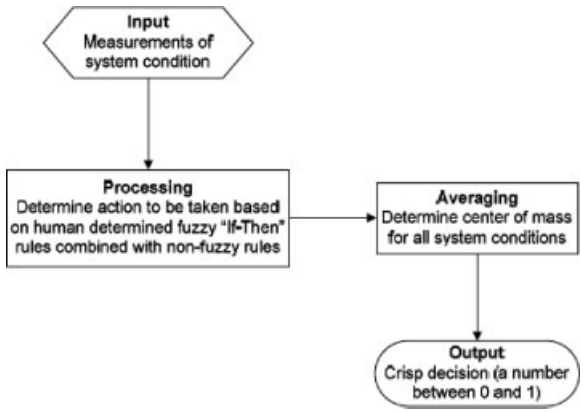

Fig. 1. Fuzzy Logic Controller Steps.

for each input/output parameter and (2) design the fuzzy rules. The membership function is a graphical representation of the magnitude of participation of each input. It associates a weighting with each of the inputs, define functional overlap between inputs, and determines an output response. The fuzzy logic rules use the input membership values as weighting factors to determine their influence on the output sets. In Section 5, we present the details about the controller that we integrated with Snort.

Snort is a signature-based NIDS that uses a combination of a flexible rule-driven language and preprocessors to analyze traffic [14]. The rule-driven language is used to create signatures and examine packets. However, the preprocessor code allows deeper examination to the packets that cannot be analyzed by the rules alone. Preprocessor can perform different tasks such as port scanning detection and web traffic normalization. It gives snort the power of looking at and manipulating stream, in contrast of looking to single packet at a time as rules.

One of the attacks that Snort detects is port scanning.

Port Scanning is one of the most popular reconnaissance techniques attackers use to discover services they can break into. All machines connected to a LAN network or Internet run many services that listen at well-known and not so well known ports. A port scan helps the attacker find which ports are available (i.e., what service might be listening to a port). Essentially, a port scan consists of sending a message to each port, one at a time. The kind of response received indicates whether the port is used and can therefore be probed further for weakness.

There are a number of different methods to perform the actual port scans. By setting different TCP flags or sending different types of TCP packets, the port scan can generate different results or locate open ports in different ways. A SYN scan will tell the port scanner which ports are listening and which are not, depending on the type of response generated. A FIN scan will generate a response from closed ports; but ports that are open and listening will not send a response, so the port scanner will be able to determine which ports are open and which are not.

Port scanning software, in its most basic state, simply sends out a request to connect to the target computer on each port sequentially and makes a note of which ports responded or seem open to more in-depth probing.

If the port scan is being done with malicious intent, the intruder would generally prefer to go undetected. Network security applications can be configured to alert administrators if they detect connection requests across a broad range of ports from a single host. To get around this, the intruder can do the port scan in strobe or stealth mode. Strobing limits the ports to a smaller target set rather than blanket scanning all 65 536 ports. Stealth scanning uses techniques such as slowing the scan. By scanning the ports over a much longer period of time you reduce the chance that the target will trigger an alert.

Snort [14] attempts to detect four kinds of port scanning: (1) Portscan is a one-to-one host scanning where one source scans multiple ports on the destination host. (2) Distributed Portscan is a many-to-one host scanning where multiple sources scan various ports on the destination host. (3) Decoy Portscan is also a many-to-one host scanning where multiple sources scan various ports on the destination host. It differs from Distributed Portscan in that it connects to a single port multiple times. (4) Port Sweep is a one-to-many host scanning where one host scans the ports of multiple destinations.

In Reference [15], the authors presented a method for detecting port scanning attacks using rule-based state diagram techniques. A set of rules corresponding with the appropriate thresholds was designed for intrusion decision. The parameters used in this work have static values, for example $\alpha = 1$ second and $\beta = 20$ packets. Many port scanning attacks occur within time more or less than 1 second, so this detection rule cannot detect such attack (scan). Also, some attacks send more than 20 packets to scan the victim. Therefore, assigning the number of attacking packets to 20 will not help in detecting port scanning, but on the other hand it will lead to false alarms.

In Reference [16], the authors present a system called Fuzzy Intrusion Recognition Engine (FIRE)

DOI: 10.1002/sec

that uses fuzzy Logic to detect malicious activities against computer networks. The major problem in this approach is that it fails to detect many kinds of port scanning attacks.

In Reference [17], the authors present an abnormal traffic control framework that detects slow port scan attacks using fuzzy rules and a stepwise policy. The major drawback in this work is that it only considers the number of packets in the decision making procedure; the same drawback that exists in most commercial IDS systems. Other important parameters are ignored, for instance, the inter arrival time between packets.

In Reference [18], the authors explored the possibility of integrating fuzzy logic with Data Mining methods using Genetic Algorithms for intrusion detection. Their work is very general and not tailored toward port scanning detection.

When an IDS detects an attack such as port scanning, it generates an alarm. Alarms can be categorized as follows depending on the attack and the occurrence of alarms:

(1) False positive alarm: The IDS will be triggered without any occurrence of a malicious attack.
(2) False negative alarm: The IDS sensors will not detect a malicious activity even though an attack exists.
(3) True positive alarm: The IDS sensors detect and report a malicious attack and an alarm is generated.
(4) True negative alarm: it is not an actual alarm but it is a state in which the IDS do not trigger an alarm for malicious activity within a network [1].

Reducing false negatives is given a very high priority, sometimes at the expense of higher rates of false positives because it is the most dangerous alarm in IDS. However, because of the nature of the signatures that IDS uses to detect malicious activity, it is almost impossible to completely reduce false positives and negatives without using new features to make the IDS more effective.

In our research, we discovered, through experiments, that Snort and some commercial IDS systems have several weaknesses (for example, they do not detect slow port scanning) that can be used by an attacker to exploit the system security. We conducted various types of port scanning attacks on two machines, one protected by Snort and the other by Juniper Netscreen. We performed the attacks using *Frameip* and *Advanced Port Scanner* tools. Table I clearly shows that both IDSs have their limitations. On the other hand, FB-Snort succeeded where the others failed. The '*wait*' variable indicates the time that *Frameip* waits before sending the next packet.

Table I. Successful port scanning attacks on Snort and Juniper IDSs (✓ means attack detected).

IDS system	Advanced port scanner	Frameip (wait = 0.6 microsecond)	Frameip (wait = 1 microsecond)
Snort	✓	✓	x
Juniper Netscreen 50	✓	x	x
FB-Snort	✓	✓	✓

The attack on Juniper Netscreen was successful because by default if a remote host scans 10 ports in 0.005 seconds (5000 microseconds), the device flags this as a port scan attack, and rejects all further packets from the remote source for the reminder of the specified timeout period (Figure 2). By changing the frequency of the sent packets, Juiper Netscreen can be fooled. Same reasoning applies to Snort signatures. We observed, that as the port scanning attack becomes slow (time between sent packets is large), Snort and other commercial IDS systems stop to produce any alarms. On the other hand, FB-Snort detects these attacks and produces an alert (Table I).

Based on our discussion so far, the approaches presented above to detect port scanning have some problems. We suggest a new approach (FB-Snort) that integrates a customized Fuzzy Logic controller with Snort in order to reduce Snort false negative and false positive alarms. FB-Snort takes some input from Snort and then decides whether an attack exists or not. The architecture of FB-Snort is discussed next.

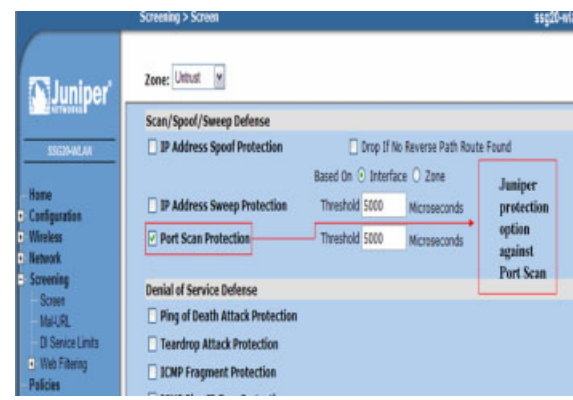

Fig. 2. Default settings for detecting port scanning in Juniper Netscreen.

DOI: 10.1002/sec

3. FB-SNORT ARCHITECTURE

Snort detects many kinds of attacks, but it gives many false positive and false negative alarms especially when detecting port scanning attacks. Furthermore, it does not show the levels of detected attacks. We designed Fuzzy-Based IDS (FB-Snort) to solve this problem. FB-Snort is a combination of snort and fuzzy logic. This combination will enhance the detection system within snort by reducing false alarms, and providing a system with levels of detected attacks. FB-Snort works within Snort and it is not a separate system. FB-Snort is supposed to improve on Snort by (1) adding levels to Snort alerts, (2) reducing the false positives and false negatives, and (3) generating the results efficiently. Actually, the results obtained by FB-Snort show more accurate results than Snort.

Figure 3 shows the architecture of FB-Snort in details. It describes the flow of information between snort and the fuzzy logic controller. As shown, the network traffic passes through many PC's which has snort sensors IDS; these sensors collect traffic for snort so they can be analyzed. Traffic data received from the sensors are stored in snort database. Then traffic passes through snort processor which is able to analyze packets, to get IP addresses, parameters and other value which help snort to alert. From all the parameters collected by Snort, we care about the parameters that will be inputted to the fuzzy logic controller. These parameters are: (1) average time between received packets by destination/victim (ART), (2) number of sent packets by source (NSP), and (3) number of received packets by destination/victim (NRP). These parameters are inputted to the fuzzy logic controller to calculate the attack level.

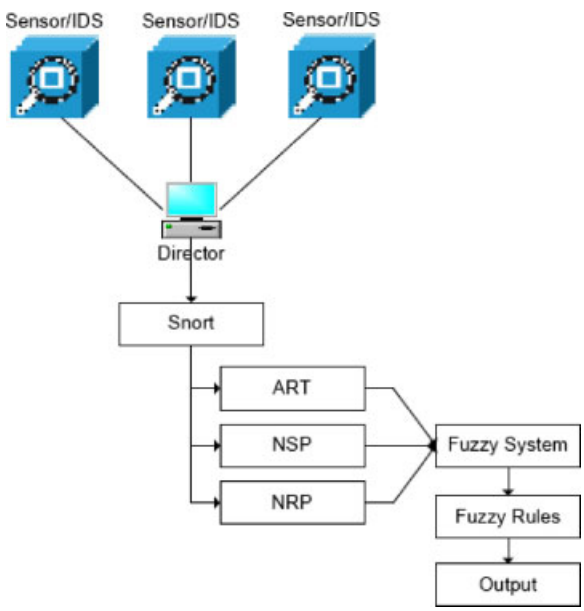

Fig. 3. FB-Snort architecture.

4. FUZZY LOGIC PARAMETERS

As mentioned above, we chose ART, NSP, and NRP as input to the controller in order to detect port scanning. While deciding on the parameters, we focused on the time when the packets were sent or received and their number. Time affects the detection process, because if time between received packets is too long, sources cannot be considered attackers with a high probability. On the other hand, if the time is short, it means that there might be an attacker trying to scan the ports of the system. Three different experiments were performed in order to obtain threshold values for the input parameters (training step). In these experiments, Advanced Port Scanner was used to perform various port scanning attacks. Also, *Commview* tool was used on the victim's host to monitor the parameter values.

In the experiments, we performed three kinds/levels of scanning (1) low scanning, (2) medium scanning, and (3) high scanning. The experiments depend on the number of hosts performing port scanning and the number of packets sent. The higher the number of sent packets, the higher the level of the scanning is. The different levels are used to assign ranges for port scanning attack. As a result, there are more chances to detect attacks. Sample results of the experiments are summarized in Table II. When two hosts did port scanning to one host, the time average between two received packets was 0.00226 second, the number of sent packets by two hosts was 1470 packets, and the number of received packets by victim was 3993 packets. Same explanation goes for medium and high scanning attacks.

Notice that the ART value when 'low scanning' is used is a large number (relatively); this means that the time between two received packets is large; therefore the level of attack is low. Similarly, the ART value when

Table II. Training the fuzzy logic parameters.

Parameters\level	Low	Med	High
1. ART	0.00226	0.0017	0.0013
2. NSP	1470	2007	2794
3. NRP	3993	7135	8526

DOI: 10.1002/sec

'high scanning' is used is a small number which means that there exists a high probability of having an attack because the time between two received packets is small. That is why the value of ART when 'low scanning' is used is bigger than the value of ART when 'high scanning' is used. Values in the 'medium scanning' column mean that the time between received packets is medium, so level of attack is medium.

The sole purpose of the experiment mentioned above is to tune and train the fuzzy logic parameters. The experiment uses two hosts trying to scan one host. It is worth mentioning that even if we conducted the experiment allowing more than two hosts to scan one host, the results (thresholds) obtained in Table II will not differ much.

Looking at NSP and NRP parameters, each level of attack reflects the number of sent packets by source and the number of received packets by destination. The larger the value of NSP and NRP, the higher the probability of having a port scanning attack. These three parameters are extracted from Snort and fed to the fuzzy logic controller. The controller then combines them in an intelligent way and produces a single number indicating the level of the attack. In the next section, we discuss the fuzzy logic controller.

5. FUZZY LOGIC CONTROLLER

As mentioned in Section 2, the fuzzy logic controller is composed of membership functions (for each input/output variable) and fuzzy rules. In this section we discuss these two components. The values of the parameters, taken from the experiments discussed in Section 4, were used to tune the fuzzy logic membership functions and to create the fuzzy logic rules. Three input parameters are used (ART, NSP, and NRP). For each input parameter, three trapezoidal membership functions were designed: Low, Med, and High. Figure 4 shows the three trapezoidal membership functions for the NRP parameter (this snapshot was taken from Matlab—the software we used to design the fuzzy logic controller). The output parameter also has three trapezoidal membership functions distributed in the range [0.0, 1.0].

After defining the input parameters, the fuzzy logic rules are designed and tested. These rules were written depending on the knowledge of detecting port scanning and the relationship between the parameters used to detect that attack (Figure 5). Out of the 20 rules we designed to detect port scanning, we discuss the following three rules:

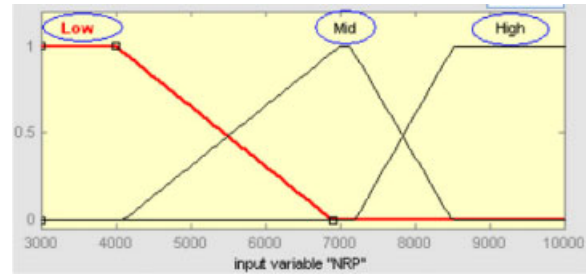

Fig. 4. NRP membership functions.

(1) *If (ART is low) and (NSP is med) and (NRP is high) then (output is high).* This rule presents an attack with a high accuracy because the time between the received packets is low (high attack) and the number of packets the victim received is high.
(2) *If (ART is high) and (NSP is med) and (NRP is low) then (output is low).* This rule presents an attack with a low accuracy because the time between the received packets is high (low attack) and the number of packets the victim received is low.
(3) *If (ART is med) and (NSP is high) and (NRP is med) then (output is med).* This rule presents an attack with a medium accuracy because the time between the received packets is medium and the number of packets the victim received is medium.

Once Snort captures the packets, we move to the Fuzzy-Logic controller to detect port scanning. This is done by entering the parameters' values which are gathered by Snort into the fuzzy system, and then the rules will be applied on them. The output of the system shows the level of the detected port scanning attack. The next section discusses the experimental results.

6. EXPERIMENTAL RESULTS

We installed and configured Snort on Windows environment. Snort can be run in various modes: Sniffer mode, Packet logger, and NIDS. In our system we run Snort as NIDS which is the most complex and complicated configuration mode in Snort. This mode allows Snort to analyze network traffic to be matched against a user defined rule set and performs several actions based upon what it discovers.

In our experiments, we consider a different number of hosts communicating with each other, where some hosts are attempting to port scan other hosts. We then use Snort and FB-Snort to analyze the traffic having

DOI: 10.1002/sec

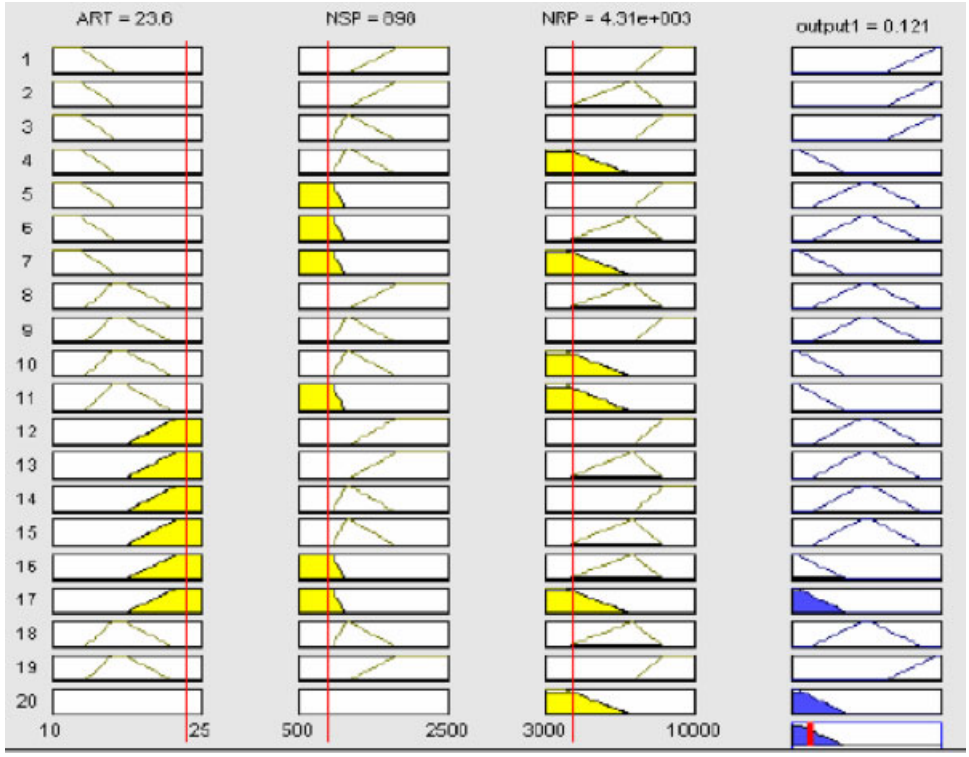

Fig. 5. The Fuzzy Logic rules used to detect port scanning. The inputs to the system are ART, NSP, and NRP.

in mind the following questions: Can FB-Snort detect whatever Snort detects? Does FB-Snort reduce the false alarms generated by Snort?

We start by using four hosts trying to ping and ftp each other. We place one host as a server and the others as clients. We then analyze the traffic (Table III). Our goal in this experiment is to study the traffic flow under normal conditions before introducing malicious nodes. The values presented in Table III are used to tune the fuzzy logic parameters.

After tuning the fuzzy logic parameters, we conduct different port scanning attacks. Both Snort and FB-Snort are used to detect the attacks. The tools used in our testing are *Snort*, *Snortsnarf*, advanced port scan, and *Commview*. We did two similar experiments with different number of attackers in order to differentiate between different levels of attack. In the first experiment, three hosts performed port scanning on a single host and in the second experiment, four hosts performed port scanning on a single host.

Table III. Parameter values under normal traffic.

Parameter	Ping (second)	FTP (second)
ART	0.17	0.059
NSP	20	173
NRP	92	330

In these experiments, both Snort and FB-Snort were able to detect the port scanning attacks. Table IV presents the parameter values passed to the fuzzy logic controller. When that attack was classified as medium, the values for ART, NSP, and NRP were 16.9, 1406, and 6544, respectively. When that attack was classified as high, the values for ART, NSP, and NRP were 15.6, 1925, and 8495, respectively. Definitely, in both scenarios, an attack was taking place. But, the 4-to-1 attack was more obvious and powerful. FB-Snort was able to detect these attacks and moreover it gave an idea on how powerful or severe the attack was. In the first attack (3-to-1), the output of fuzzy system was 0.41, and in the second attack, the output was 0.877 (Figure 6). The higher the number, the more severe the attack is. So, in these experiments FB-Snort outperforms Snort in the sense that it catches the attack and moreover it gives us an idea on how severe the attack is.

DOI: 10.1002/sec

Table IV. Values of the parameters gathered by snort and entered to the fuzzy logic controller.

Parameter\level	Medium attack (3-to-1)	High attack (4-to-1)
ART	16.9	15.6
NSP	1406	1925
NRP	6544	8495

Note that Figure 6 is not the final result that the system administrator will see. In fact, the system administrator will not see Figure 6 at all. Figure 6 displays a snapshot of the internal decision making procedure done by FB-Snort. It says that according to the collected input parameters, the degree of the attack is 0.41; i.e., based on our threshold it is an attack of a medium degree.

To show how FB-Snort outperforms Snort, we used *Frameip* tool to generate packets and send them to another host which is running both Snort and FB-Snort. Packets were sent with different frequency i.e., the waiting time between sending packets is changed in every experiments. By default, *Frameip* waits 1 microsecond before sending another packet. We varied this wait variable and checked the results. In many occasions, Snort considered normal traffic a port scanning attack (false positive alarm) where the traffic was completely normal. On the other hand, FB-Snort considered the traffic normal and did not issue any warning. The other way around was also true. Snort was not able to detect some port scanning attacks (Table I). As a result, we were able to achieve our aim which is an intelligent system that reduces false alarms. More testing need to be done, but the initial results look very promising.

7. CONCLUSION AND FUTURE WORK

As a summary we can say that anomaly based network intrusion detection is a complex process and we focus on one of these anomaly intrusions which is port scanning. The variety in the network data stream, the amount of data to be processed, and the subtle and ever-changing ways that attackers breach systems, all conspire to complicate the task. In one of our testing, when more than five attackers did port scanning at the

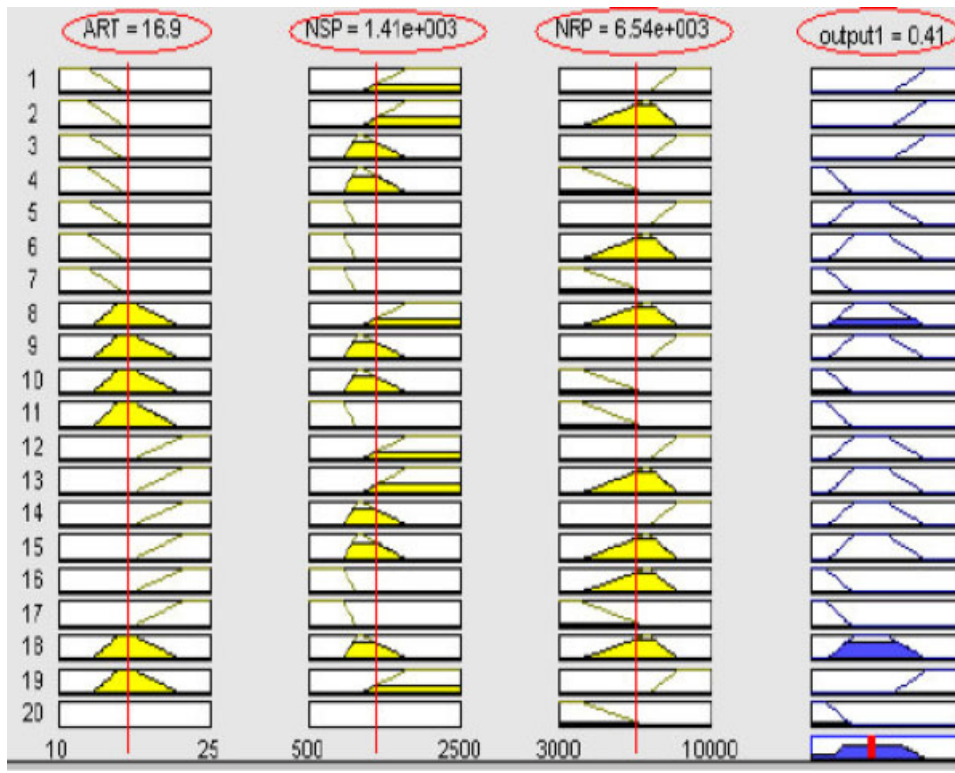

Fig. 6. The result FB-Snort produces when the parameter values presented in Table IV (medium attack) are inputted to the controller.

DOI: 10.1002/sec

same time, the victim's machine had a denial of service. Therefore, this attack is really a complicated attack and it can be the starter for different types of attack. While this research does not solve the problem of finding all network based attacks, the fuzzy intrusion detection (FB-Snort) holds promising results to be a high-level intrusion detection scheme. The use of a customized fuzzy logic controller enhances the capabilities of Snort to detect port scanning attacks. It also helps in reducing the false positive and negative alarms. The results show that the fuzzy system can be better combined with Snort to make Snort more intelligent and effective.

It is early to say that FB-Snort perfectly detects all sorts of port scanning attacks. In this work, we showed that FB-Snort outperforms Snort in many occasions. We still need to conduct comparison *versus* work in the literature. Once we are sure that the fuzzy logic controller helps in detecting all types of port scanning attacks with no false positives and negatives, we will go ahead and completely merge it with Snort, obtaining a final usable version of FB-Snort.

References

1. Mukherjee B, Todd Heberlein L, Levitt KN. Network intrusion detection. *IEEE Network* 1994; **8**(3): 26–41.
2. Denning DE. An intrusion-detection model. *IEEE Transactions on Software Engineering* 1987; **13**(2): 222–232.
3. Bace RG. Intrusion Detection, Macmillan Technical Publishing, Indianapolis, USA, 2000.
4. Yao JT, Zhao SL, Saxton LV. 2005; A study on fuzzy intrusion detection, In Data mining, intrusion detection, information assurance, and data networks security2005, (Orlando, March 28–29, 2005) Dasarathy B. (ed.). *Proceedings of the International Society for Optical Engineering*, Vol. 5812, pp. 23–30.
5. Kang Dae-Ki. Learning classifiers for misuse and anomaly detection using a bag of system calls representation. *Proceedings of the 6th IEEE Workshop on Information Assurance and Security United States Military Academy*, West Point, NY, 2005.
6. Qiao Y, Xin XW, Bin Y, Ge S. Anomaly intrusion detection method based on HMM. *Electronics Letters* 2002; **38**(13): 663--664.
7. Lee W, Stolfo SJ. Data mining approaches for intrusion detection. *7th USENIX Security Symposium*, 1998; pp. 79–94.
8. Mohajerani M, Moeini A, Kianie M. NFIDS: a neuro-fuzzy intrusion detection system. *Proceedings of the 10th IEEE International Conference on Electronics, Circuits and Systems*, 2003; pp. 348–351.
9. Wang WD, Bridges S. Genetic algorithm optimization of membership functions for mining fuzzy association rules, P*roceedings of the 7th International Conference on Fuzzy Theory & Technology*, Atlantic City, NJ, 2000; pp. 131–134.
10. Zdeh LA. Fuzzy sets. *Information and Control* 1965; **8**(3): 338--353 118.
11. Zadeh LA. Fuzzy algorithms. *Information and Control* 1968; **12**(2): 94–102.
12. Zadeh LA. Fuzzy logic and its application to approximate reasoning. *IFIP Congress* 1974; 591–594.
13. El-Hajj Wasim. A distributed hierarchical energy efficient scheme for large scale mobile wireless ad hoc networks. *PhD Thesis*, Western Michigan University, 2006.
14. Snort: www.snort.org
15. Kanlayasiri Urupoj. Surasak Sanguanpong and Wipa Jaratmanachot. A rule-based approach for port scanning detection. *23rd Electrical Engineering Conference (EECON 23)*, Chiangmai November, 2000.
16. Dickerson J, Juslin J, Koukousoula O, Dickerson J. Fuzzy intrusion detection. *Proceedings of the NAFIPS*, Vancouver, British Columbia. 2001; 3: 1506–1510.
17. Jaekwang Kim, Jee-Hyong Lee. A slow port scan attack detection mechanism based on fuzzy logic and a stepwise policy. *4th International Conference on Intelligent Environments 2008 (IET 2008) Seattle*, WA, 21–22 July 2008.
18. Dhanalakshmi Y, Ramesh Babu I. Intrusion detection using data mining along fuzzy logic and genetic algorithms. *International Journal of Computer Science and Network Security (IJCSNS)* 2008; **8**(2).

DOI: 10.1002/sec