International Journal of Computer Engineering Science (IJCES)
Volume 2 Issue 5 (May 2012) ISSN : 2250:3439
https://sites.google.com/site/ijcesjournal http://www.ijces.com/

An Analysis of Packet Fragmentation Attacks vs. Snort Intrusion Detection System

Tian Fu and Te-Shun Chou
Department of Technology Systems, East Carolina University
Greenville, NC, U.S.A.

Abstract. When Internet Protocol (IP) packets travel across networks, they must meet size requirements defined in the network's Maximum Transmission Unit (MTU). If the packet is larger than the defined MTU, then it must be divided into smaller pieces, which are known as fragments. Attackers can exploit this process for their own purposes by attacking the systems. Packet fragmentation attacks have caused problems for Intrusion Detection Systems (IDSs) for years. In this paper, Snort IDS was tested. VMware virtual machines were used as both the host and victim. Other tools were also implemented in order to generate attacks against the IDS. The experiment results show the performance of Snort IDS when it was being attacked, and the ability of Snort to detect attacks in different ways.

Keywords: Packet fragmentation, virtual machine, intrusion detection

1 Introduction

When IP packets travel across the network landscape, packets must conform to standard MTU sizes. If a packet exceeds the network's MTU, then the packet must be split into fractions. These fractioned packets are referred to as fragments. All information that travels in the network is stored in the IP head, which includes the fragment ID, fragment offset, fragment length and the more fragments flag. Fragment ID is the same as IP identification; fragment offset shows the location of the fragment when it is reassembled later; fragment length shows the length of this fragment; and the more fragments flag shows if it is the last fragment. Fogie introduced packets fragmentation and IP head in detail 1. When a large amount of data is being transported in a packet-based network, data will first be divided into small packets, and then those packets will be reassembled at the other end when it is received. Because of the IP head structure, in order to transmit in the network, every packet-based network needs to divide the data into their own MTU. For example, the Ethernet's MTU is 1,500 bytes, which means packets traveling over the Ethernet can't be larger than 1,500 bytes.

Attackers attempt to send malicious packets through the network by using packet fragmentation attacks. They try to avoid detection by the IDS installed in the victims. Through the process of fragmenting and reassembling, attackers have found many opportunities to exploit and attack the system. For example, attackers sent data that is larger than the network's MTU, while also using software that forces the system to not fragment the packets. This will cause the system to fail and crash. Although these kinds of attacks have caused problems for network security for years, this problem still continues to exist.

The objective of this paper is to test Snort IDS with packet fragmentation attacks in different parameters in order to seek the weaknesses of Snort IDS. VMware 2 virtual machines were used to establish the experiment environment, which included the host machine, the victim, and the test machine. Several tools were used to generate the packet fragmentation attacks which were then sent from the host to the victim. The tools used included: Scapy 3, Metasploit framework 4, Nmap 5, Tcpreplay 6, Tcpdump 7, and Wireshark 8. Attacks were generated from Scapy, Metasploit framework, and Nmap, the attacks were sent to test Snort IDS 9. Wireshark was used to monitor the packets on the victim. Simultaneously, these attack packets were captured and saved by using Tcpdump. The saved attacks were replayed by using Tcpreplay in different speeds and were sent to Snort to test it.

This paper is organized as follows. Section 2 presents the related works. We then demonstrate the experimental methodology, followed by a discussion of the experimental results. Finally, we conclude our work and discuss future work in the last section.

2 Related Works

Different types of networks use different protocols, so the MTU of each network varies. In order to transmit and be successfully reassembled, there are rules that each packet fragment must follow 10. The rules of packets fragment in networks include:

- All fragments must share a common fragment identification number.
- Each fragment must say what its place or offset is in the original un-fragmented packet.
- Each fragment must tell the length of the data carried in the fragment.
- Finally the fragment must know whether more fragments follow this one.

Packet fragmentation attacks can be used to cause damage to systems. Attacks have been used in multiple ways to attack systems and break networks. Attacks have been done by many people with different methods by using special software. Examples of packet fragmentation attacks can be found in many research and experiments. The basic theory of IP packet fragmentation had been studied by Garcia 11 and Anderson 12. Calculations of the packet fragmentations in different Ethernet versions such as 802.3 (v1), v2, 802.11,

802.5 were done in the research. Analysis of multiple networks' MTUs, and the varying amount of fragmentation that involved with the altered MTU sizes were calculated. Mathematical formulas of showing how to calculate the size of fragments in different networks were introduced. An example of attacking a network with packet fragmentation is shown in the research done by Cohen 13. Attackers use packet fragmentation attacks against firewalls and filters in order to break into the system without being detected by the IDS. The performance of firewalls and filters were tested and compared. Most of packets fragmentation attacks were detected by the firewalls and filters, but few of the attacks succeeded attacking the system.

And experiment of testing IDS by generating customized fragmentation attacks can be found in 14. Experiments were done by using virtual machines. The attacks were derived from Fragrouter, Metasploit Framework and Tcpdump for experiments. Three virtual machines were used in the experiments, one as a host, one as a victim, and another one as a test machine. The purpose of having a test machine was to send fragments to the victim with Fragrouter 15 in order to customize the attacks. Multiple kinds of attacks were sent to test the IDS, and one of the attacks generated by the tools was successful in entering the system without being detected by the Snort IDS.

Research of attackers using a variety of packet fragmentation attacks to attack the IDS can be found in 16. Many Packets fragmentation attacks were introduced, such as IP Fragmentation Attacks, Tiny Fragmentation Attacks, Tiny Fragmentation Attack Countermeasures, Overlapping Fragmentation Attack Countermeasures, as well as other attacks such as Teardrop Denial of Service (DoS) Attacks, Teardrop Attack Implementation and Teardrop Attack.

Because of the way fragments reassemble, packet fragmentation mechanisms lead to attacks that bypass many current Internet firewalls, they are called Packet Reassembly Attacks. An introduction of Packet Reassembly Attacks can be found in 17. All the datagrams are supposed to be fragmented into packets, leaving the header portion of the packet intact except for the modification of the fragmented packet byte. The number contained in the fragment's IP header indicates where it is supposed to be located in the datagram. In reassembly, the IP re-assembler creates a temporary packet with the fragmented part of the datagram in place and adds incoming fragments by placing their data fields at the specified offsets within the datagram being reassembled. Once the whole datagram is reassembled, it is processed as if it came in as a single packet.

Many common fragmentation attacks that attackers use to achieve different results were introduced in 18. Tiny Fragmentation Attacks, The Teardrop Attack, Overlapping Fragmentation Attacks, Ping O' Death Fragmentation Attacks were included. Tiny Fragmentation Attack is an attack that uses small fragments to force some of the TCP header information into the next fragment. This may produce situations where the TCP flags field is forced into the second fragment and filters the attempt to drop connection requests. It will be unable to test flags in the first octet thereby ignoring them in subsequent fragments. The Teardrop Attack is a DoS attack. The attacker uses an IP to

generate packet reassembly problems that will cause the target system to crash. The target computer system must overwrite data in other packets to re-assemble the packets. The attempt to re-assemble packets with overlapping data can cause the target computer system to crash. Overlapping fragmentation attacks are another variation on the teardrop attack that also uses overlapping fragments. This attack is not a DoS attack but is used in an attempt to bypass firewalls to gain access to the victim host. The Ping O' Death fragmentation attack is a DoS attack, which utilizes a ping system utility to create an IP packet, which then exceeds the maximum allowable size for an IP datagram of 65,535 bytes. This attack uses many small fragmented ICMP packets which, when reassembled at the destination, exceed the maximum allowable size for an IP datagram. This can cause the victim host to reboot, hang or even crash.

A serious flaw in a popular honeypot software suite that allows an attacker to easily identify the presence and scope of a deployed honeypot is described in 19. Experiments in this paper were based on a set of specially crafted packets which were able to elicit a response from a Honey-based honeypot. The flaws of honeypot software were analyzed and the ways attackers exploited it with these flaws were introduced.

3 The Experimental Methodology

3.1 Creation of a packet fragmentation experimental environment

This research started with the creation of a virtual network using the virtualization software VMware workstation 6.0. VMware allows users to install and configure multiple virtual machines that can run different operating systems on one physical machine. In order to carry on the packets fragmentation attacks experiments, three virtual machines were included in the network. One victim, one attacker, and one test machine were created in VMware workstation. Figure 1 shows the network architecture. The attacker generated the attacks and sent them to the victim, in order to test the Snort IDS installed on the victim. Test machine was used to record the packets sent in the network, in order to analyze and replay the packets.

Fig. 1. Networking Architecture

3.2 Tools and attacks used for the experiments

A variety of tools were installed and configured in three virtual machines. The victim was equipped with sniffing tool Wireshark and intrusion detection tool Snort IDS for recording the network traffic and testing the intrusion detection capability. Testing tool Metasploit framework and scanning tool Nmap were running on the attacker for exploiting the vulnerabilities of the victim. Attack packets were generated by Scapy, which was also installed in the attacker, Tcpdump was installed in the test machine, it was used to capture and save the packets sent in the network. Tcpreplay was also carried by the test machine, for replaying the collected network traffic.

Wireshark is an open source network protocol analyzer. It allows the user to sniff network traffic on network interfaces. For recording the network traffic for future analysis, the Wireshark with WinPcap was also installed in the victim. Each time the attacks were sent from Scapy, they were monitored with Wireshark. In the beginning of the experiments, probe and DoS attack packets were generated using Metasploit and Nmap, and the attack packets were saved in the attacker. By using Scapy, the packets were configured to generate fragmented packets. They were collected by Tcpdump, and Tcpreplay was used to play the collected traffic in different speeds.

Snort is one of the most widely used IDS software. It was created by Martin Roesch in 1998. Snort IDS can be used to generate network traffic, analyze traffic and detect attacks. The Snort V2.6.1.5, which was installed in the victim, provides packets fragment 3 signature rules. Fragment 3 is a target-based IP defragmentation module in Snort. It includes 13 defense rules, such as Teardrop attack alert, Short fragment, possible DoS attempt, Zero-byte fragment, etc.

The Metasploit framework is open source software for people to perform penetration testing, IDS signature development, and exploit research. Of its 320 exploits and 217 payloads, windows/vnc/ultravnc_client equipped with payload windows/shell_bind_tcp is chosen to exploit ultravnc_client buffer overflow DoS vulnerability of the victim machine. It is a client buffer overflow attack. The attacker exploits the vulnerability of a system that does not correctly perform a boundary check of a user's input data before copying it to a fixed length memory buffer. Once the vulnerability was found, the attacker can supply excess data into the insufficiently sized memory buffers and possibly corrupt the data and thus make the service crash. Furthermore, the attacker can add executable data into the stream and remotely activate it to gain unauthorized access when the buffer overflows.

SYN flood attack is one of the DoS attacks. It attacks the network by sending the TCP connection requests faster than the speed which the network could process. It will cause the high fake connections in the network and cause a DoS attack to the legitimate traffic.

Probe attacks are attacks to explore open vulnerabilities or weaknesses in a network. They aim to gather information on systems within a network in order to lead to access to targeted computers in the future. Among various types of Probe attacks, network port scanning is a common way to find out what resources are available on your network. In

the experiment, a free security scanner Nmap is used in the attacker for network ports exploration of target victim. It divides ports into six states: open, closed, filtered, unfiltered, open|filtered, or closed|filtered. These states give attackers an idea of services' statuses in the target computer system. If the connection to a port is successful, the port is listed as open, otherwise it is said to be closed.

Scapy is an open source packet manipulation tool written in Python. This tool combines the functions of scanning, tracerouting, probing, and attacking. In this research, the SYN flood attack was generated by using Scapy to modify the original packets' configurations.

Tcpdump is a program that uses command lines, which allows users of the program detect packets that were going through networks on which a computer is being used. Tcpdump was used to catch the packets in the network in order to replay it on Tcpreplay.

Tcpreplay is a tool that written by Aaron Turner for UNIX operating system. It is able to capture traffic to test a variety of network devices. By using Tcpreplay, collected DoS, Probe, and SYN flood attacks were replayed at different speed to the victim.

4 The Experiment Results

Three types of attacks were used in the research. Two trace files recorded by Tcpdump were launched in a variety of packet sizes and network transmission rates for testing the intrusion detection capability of Snort. One was *DoS* attack trace and the other was *Probe* attack trace file. The *DoS* attacks trace file was generated by Metasploit framework, and the *Probe* attack trace file was generated by using Nmap. *SYN flood* attacks were also used in experiments. It is launched and customized by using Scapy.

The experiments were divided into three parts. The first part was to test Snort using normal packet sizes at different network transmission rates. The second part was to test Snort using various fragmented packet sizes with constant network transmission rates. The third part was to test Snort using several fragmented packet sizes with varying network transmission rates.

4.1 Test Snort using normal packet size at different network transmission rate

Experiment 4.1.1. Replay *DoS* attack traffic once in normal network transmission rate
Experiment 4.1.2. Replay *DoS* attack traffic once in high network transmission rate
Experiment 4.1.3. Replay *DoS* attack traffic 100 times in high network transmission rate

In the first part of the research, packets size was set to 1,500 bytes, which is the natural size of Ethernet. Three experiments were done by sending DoS attack in different transmission rate to test Snort IDS. In experiment 4.1.1, the DoS attack was launched under normal network transmission rate, and this attack was replayed by experiment 4.1.2,

in higher network transmission rate. In the last experiment 4.1.3, this DoS attack was repeated replayed 100 times in highest speed that can be sent in the network.

There were 125 packets in the collected DoS attack that included both traffic directions from attacker to the victim and from the victim to the attacker. The attack generated 125 packets of which 123 were TCP and 2 were IP. Snort should have been able to receive traffic in both directions; all the TCP packets and IP packets should be able to be analyzed; each DoS attack that was received by Snort should be alerted by the system. The command used to replay the DoS attack and the Snort results in experiment 4.1.1 are shown on Figures 2 and 3, respectively. Table 1 summarizes the experimental results.

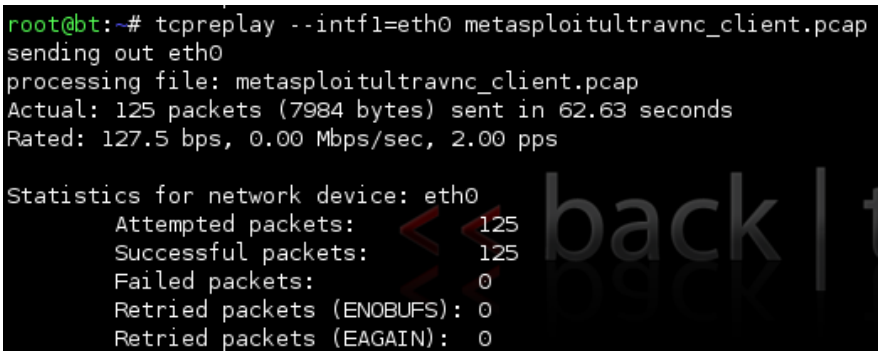

Fig. 2. Replay *DoS* Attack

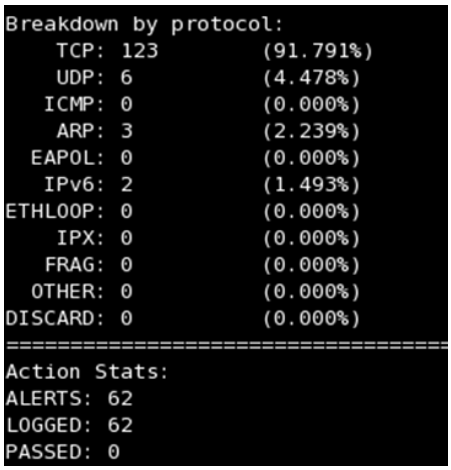

Fig. 3. Snort Alert Summary Table

Table 1. The experimental results of testing Snort with *DoS* attacks in different speeds

DoS attack	Replay traffic speed (bps)	# of packets sent by using Tcpreplay	# of packets received by Snort	# of alerts generated by Snort
Experiment 4.1.1	127.5	125	125	62
Experiment 4.1.2	348,362.9	125	99	50
Experiment 4.1.3	3,588,169.5	12,500	4,493	2,276

Traffic sent in the network included packets in both directions from the attacker to the victim and from the victim back to the attacker. Snort IDS should alert every DoS packet that was received, which should be 62 packets. Alerts were generated and stored in Snort. In experiment 4.1.1, packets were sent by the speed of 127.5bps. The result shows that 125 packets were detected, 62 alerts were generated by Snort. DoS attacks were detected with 100% detection rate under the normal network transmission rate.

In the second experiment, same DoS packets were replayed by Tcprelpay, and sent to the victim in the speed of 348,362.9bps. Instead of receiving 125 packets in the first experiment, Snort only reported 99 of them, which was only 80% of the packets sent in the network. It generated 50 alerts instead of 62, which was 80% of the alerts comparing to experiment 4.1.1. 20% of the DoS attacks were successfully sent to attack the victim. Snort IDS started to drop packets in higher transmission rate.

In experiment 4.1.3, DoS attack packets were repeated 100 times and sent to the victim in the speed of 3,588,169.5bps. The number of packets sent in this experiment was 12,500. The result of this experiment demonstrated that Snort could only receive and record 4,493 packets out of 12,500. The receiving rate of packets was 36%. Instead of alerting 6,200 DoS attacks, Snort could only generate 2,276 alerts, which is also 36% of the right amount of alerts. Snort dropped most of the packets in this experiment. More DoS attacks were successfully sent to attack the system even under the protection of Snort IDS.

In order to test the performance of Snort under other attacks, an additional group of test was completed by using probe attacks. The probe attack contained 2,010 packets, 1,000 of them were probe attack packets sent from the attacker to the victim, which should be detected and alerted by Snort. This additional test also contained three experiments, with the same idea of test Snort with normal speed, higher speed and repeated sending packets in the highest speed of the network. Table 2 summarizes the experimental results.

Experiment 4.1.4. Replay *Probe* attack traffic once in normal network transmission rate

Experiment 4.1.5. Replay *Probe* attack traffic once in high network transmission rate

Experiment 4.1.6. Replay *Probe* attack traffic 100 times in high network transmission rate

Table 2. The experimental results of testing Snort with *Probe* attack in different speeds

Probe attack	Replay traffic speed (bps)	# of packets sent by using Tcpreplay	# of packets received by Snort	# of alerts generated by Snort
Experiment 4.1.4	190.2	2,010	2,010	1,000
Experiment 4.1.5	1,951,345.6	2,010	230	112
Experiment 4.1.6	3,319,515.5	201,000	51,455	25,582

The result had shown that when the attacks were sent in a lower speed, which was 190.2bps, Snort IDS detected all 2,010 packets and generated 1,000 alerts, which is the correct number of packets contained in probe attacks file. Snort IDS could detect all the packets and alert all the attacks under the normal transmission rate. However, when the speed of sending the same probe packets reached 1,951,345.6bps, Snort dropped 88.6% of the packets. Instead of receiving 2,010 packets, Snort only received 230 packets. Instead of generating 1,000 alerts, there was only 112 alerts reported, which was only 11.2% of the amount which was generated under the normal speed. Furthermore, when the Probe attack traffic was sent repeatedly in the highest speed of the network, Snort dropped most of the packets. It received 51,455 instead of 201,000 packets sent from the attacker. Most of the alerts were also missing, 25,582 instead of 100,000 probe attack alerts were reported by Snort. 74.4% of the probe attacks successfully attacked the victim under the protection of Snort IDS.

4.2 Test Snort by sending packets in different fragment size

Experiment 4.2.1. *SYN flood* attack packets fragmented by the fragment size of 1,500 bytes
Experiment 4.2.2. *SYN flood* attack packets fragmented by the fragment size of 100 bytes
Experiment 4.2.3. *SYN flood* attack packets fragmented by the fragment size of 1000 bytes
Experiment 4.2.4. *SYN flood* attack packets fragmented by the fragment size of 10,000 bytes

The second part of the research tested the performance of Snort IDS under different fragmentation attacks. Four experiments were included in the second part of the research. Different packet fragment sizes were generated by the attacker and sent to the victim. SYN flood packets were used in all the experiments, Scapy installed in the attacker machine was used to generate and customize these SYN flood packets. Figure 4 shows an example of customized fragmented SYN flood packets generated by Scapy. Because the MTU of the Ethernet was 1500 bytes, the original SYN flood packets with 1500 bytes fragment size were sent to the victim in the first experiment. Then the fragment size of each packet varied from 100 bytes to 10,000 bytes in experiment 4.2.2 to 4.2.4. Table 3 shows the results of testing Snort IDS with different fragment sized SYN flood attacks.

```
>>> a=fragment(IP(dst="192.168.17.144",ttl=(1,10))/TCP(flags="S")/("x"*200000),f
ragsize=100)
>>> send(a)
```

Fig. 4. Customized Fragmented Packets Generated by Scapy

Table 3. Experimental results of testing Snort with *SYN flood* attack in different fragment sizes

SYN flood attack	Fragment size	# of packets sent	# of packets received by Snort	# of alerts generated by Snort
Experiment 4.2.1	1,500	1,360	1,360	98
Experiment 4.2.2	100	19,240	19,240	684
Experiment 4.2.3	1,000	2,010	2,010	101
Experiment 4.2.4	10,000	1,410	1,410	204

The original SYN flood attack packet was sent to test the system in the first experiment. When the packets were sent in the network, they will be fragmented according to the MTU of the Ethernet. The SYN flood attacks were fragmented into 1,360 fragmented IP packets, Snort received all of the packets, and detected 98 alerts which were from the rules of fragment 3.

Fragment size was changed to generate different packets fragmentation attacks, the SYN flood attacks were fragmented into the size of 100 bytes in the second experiment, 19,240 fragmented packets were generated by Scapy, all of which were detected and alerted by Snort. In experiment 3, the fragment size was changed from 100 bytes to 1,000 bytes. 2,010 packets were sent from the attacker. All of them were detected and reported by Snort. 101 alerts were generated by Fragment 3. In experiment 4, fragment size of the packets is changed from 1,000 to 10,000, which is higher than the MTU of the Ethernet. 1,410 fragmented packets were sent to attack the victim, all of which were detected by Snort IDS. None of the packets were dropped. This group of experiments showed that no matter how large or small the fragment sizes were Snort IDS can always detect all the attack packets.

4.3 Test Snort IDS by sending packets in various fragment size and in different speed

Experiment 4.3.1. Fragmented packets sent by normal speed
Experiment 4.3.2. Fragmented packets sent by high speed
Experiment 4.3.3. Fragmented packets sent 100 times at the same time by high speed

The final group of experiments combined the previous two groups of experiments. Three experiments were included in the last part of the research in order to test Snort IDS by sending customized fragmented packets in different network transmission rate.

Fragmented SYN flood attack packets were sent from the attacker to the victim in different speeds. The packets sent to Snort were fragmented into 1,362 packets. The fragmented packets were sent with the speed varied from 306,645.3bps in the first experiment to 68,405,072.5bps in experiment 4.3.3. Table 4 summarized the experimental results.

Table 4. The experimental results of testing Snort with fragmented in different speeds

SYN flood attack	Replay traffic speed (bps)	# of packets sent by using Tcpreplay	# of packets received by Snort	# of alerts generated by Snort
Experiment 4.3.1	306,645.3	1,362	1,362	98
Experiment 4.3.2	44,493,336.5	1,362	633	46
Experiment 4.3.3	68,405,072.5	136,200	90,414	8,733

Fragmented SYN flood packets were sent in the normal speed in the first experiment, which is 306,645.3bps. Snort detected 1,362 packets and 98 alerts, which is the same amount of packets sent from the attacker. Same amount of alerts as the second group experiments by Snort. All the attacks were detected and reported when fragmented packets were sent in the lower transmission rate. The network transmission rate increased from 306,645.3bps to 44,493,336.5bps in the second experiment. Instead of detect 1,362 SYN flood packets, Snort reported 633 packets, dropped 53.5% of the packets. The SYN flood traffic was sent repeatedly in the last experiment. 136,200 packets were sent to Snort, and 90,414 of them were detected. 8,733 alerts were also generated 8,733 instead of 9,800. Snort dropped 34% of the packets when the network transmission rate is high.

5 Conclusions and Future Work

In this paper, Snort IDS was tested with different types of attacks, transmission rates, and various packet fragment sizes. Different transmission rates and attacks were sent to test Snort in the first group of experiments. When the transmission rate exceeded the speed Snort IDS could handle, it could not detect all the packets transmitted in the network. Packets fragment size attacks were sent in the second experiment, and it showed that Snort could detect all the fragmented packets. The third group of experiments combined fragment packets and transmission rates, Snort failed to detect fragmentation packets in high speed attacks.

Snort IDS is not as strong as we thought. Although it can detect all the normal fragmented packets sent by Scapy, when the speed of sending packets is fast, and when we send a lot of packets at the same time, Snort IDS will drop most of the packets sent in the network. Conclusion can be made from all the experiments done in this research: there

are some chances attackers can use the software such as Scapy and Tcpreplay to attack the Snort IDS without being detected by Snort.

In the future, experiments about sending the packets fragmentation attacks to an actual computer with the powerful tools such as Scapy and the tools we used in these the experiment can be done to see if the computer system will react to these attacks.

References

1. Fogie, S., "Security Reference Guide," informIT, April, (2008)
2. VMware. http://www.vmware.com/, (Last December in May 2012).
3. Scapy. http://www.secdev.org/projects/scapy/, (Last browsed in May 2012).
4. Metasploit framework. http://www.metasploit.com/, (Last browsed in May 2012).
5. Nmap. http://nmap.org/, (Last browsed in May 2012).
6. Tcpreplay. http://tcpreplay.synfin.net/, (Last browsed in May 2012).
7. Tcpdump. http://www.tcpdump.org/, (Last browsed in May 2012).
8. Wireshark. http://www.wireshark.org/, (Last browsed in May 2012).
9. Snort. http://www.snort.org/, (Last browsed in May 2012).
10. Patel, A. B., Mokbel, M. F. and Zhao, S. "Fragmentation Attack on Wireless Network," School of Computer Science, University of Windsor, Canada. (2007)
11. Garcia, E., "IP Packet Fragmentation Tutorial," http://www.miislita.com/, November 2009.
12. Anderson, J., "An Analysis of Fragmentation Attacks," (2009)
13. Cohen, F. B., "TCP packet fragment attacks against firewalls and filters," http://all.net/ (Last browsed in May 2011).
14. Parker, D., "Packet fragmentation versus the IDS," Apr 19, (2007)
15. Fragrouter. http://monkey.org/~dugsong/fragroute/, (Last browsed in May 2012).
16. Techdoc, "Fragmentation Attacks," www.Bukisa.com, (2008)
17. Bittau, A., "The Fragmentation Attack in Practice," (2005)
18. Forte, D., "Fragmentation Attacks: Protection Tools and Techniques: Called "true preliminaries to denial-of-service", IP-Frags are a tough nut to crack for some firewalls and IDSs," Volume 2001, Issue 12, 1, pp.12-13, (2001)
19. Oberheide, J. and Karir, M., "Honeyd Detection via Packet Fragmentation," Networking Research and Development Merit Network Inc. (2006)