# Investigating the Evasion-Resilience of Network Intrusion Detection Systems

J.A.Ytreberg, M. Papadaki
University of Plymouth, Plymouth, United Kingdom
maria.papadaki@plymouth.ac.uk

**Abstract:** Network Intrusion Detection Systems provide an extra security precaution by detecting attacks that have bypassed the firewall. Knowledge-based intrusion detection systems rely upon rules to trigger alerts, mainly based upon the occurrence of certain keywords. However, attackers can send evading attack packets that will try to avoid detection by the IDS, and tools can be obtained to automate such attacks. A crucial question is therefore the extent to which modern IDS are resilient to evasion attempts of this type. This paper presents the results of experiments conducted using the Nikto evasion tool against the Snort IDS, with the aim of assessing Snort's alerting capabilities when mutated attack packets were sent to a web server. It was found that Snort alerted for about half of the attack packets. In addition, some weaknesses were identified in Snort's ability to detecting certain evasion attacks, which can be solved by creating customized rules. As a result of these findings, the paper also discusses a new detection method, based upon the division of large request strings into smaller ones, analyzing each of them against the rules. The total danger level of these combined strings could decide if the IDS would alert for the request.

**Keywords:** NIDS, Evasion, Snort, Nikto.

## 1. Introduction

The growing dependence upon information technology and networked systems highlights the need for advanced security countermeasures. As a result, a number of security technologies and tools have been employed to combat the problem of attacks and protect system resources. However, these are not completely foolproof and despite the efforts to improve their effectiveness, attackers still manage to penetrate them and compromise the security of systems. Network Intrusion Detection Systems (NIDS) aim to alert for the occurrence of attacks that have managed to bypass existing security controls, such as firewalls. Therefore, NIDSs are often thought of as a second line of security defense (Anderson, 1980).

Knowledge-based NIDS rely upon rules to trigger alerts, mainly based upon the occurrence of certain keywords. However, the task of detecting intrusions is by far trivial, and a common difficulty facing NIDSs is the problem of accurately detecting intrusions without significantly degrading the performance of the NIDS. For example, if the NIDS is too sensitive, it would alert for too many packets, including legitimate activity (false positives). On the other hand, if it is configured to be less sensitive there is a greater chance of allowing attacks to bypass the NIDS unnoticed (false negatives). Therefore, it is important to maintain a balance between the number of false positives and false negatives, also called Crossover Error Rate (CER) (Chapple, 2003); however, this is often very difficult to achieve. According to Sodiya et al. (2004) IDS systems still produce too many false positives and false negatives and lack the efficiency sorely needed.

Evasion techniques are ways of mutating attacking packets, luring the NIDS not to trigger an alert, simply because it thinks the packets are legitimate traffic. Seminal work in the area of IDS evasion was the work by Ptacek and Newsham (1998), who demonstrated the main techniques on how IDS systems can be evaded. Several attacking tools have utilized evading techniques since then, often managing to bypass intrusion detection successfully. The aim of this research is to investigate the susceptibility of modern NIDS to IDS evasion techniques, and assess whether their detection ability has improved over the recent years in that respect. For that reason, a series of experiments using Snort IDS, one of the most popular IDS on the market (Siddhart, 2005), has been conducted. It is important to note that only knowledge-based NIDSs have been considered in this research; the detection problems of other IDS categories, such as anomaly-based ones, are out of scope of this work. The IDS evasion techniques have been launched by the attacking tool Nikto. Although there are more recent tools than Nikto, the reason it was used for the experiments was due to its wide range of IDS evasion techniques (Cirt.net, 2006). The research experiments attempted to answer three questions:

- How Snort IDS responds to evasion attacks
- How Snort functions when its processor is fully engaged
- How Snort rules can be improved to strengthen its detection capability

This paper presents some of the prior work in the area of evaluating and improving IDS detection, followed by the experiments design and results. Finally the findings from this work are analyzed and discussed, leading to future work and conclusions about this work.

## 2. Related work

Barber conducted a technical assessment of the main IDS products in 2001, and found that IDS applications analyzing protocols and packet behaviour were more efficient than applications utilizing pattern matching techniques. The research also discovered that if network load exceeds 35%, the NIDS performance can suffer and start dropping packets. If the load at the Network Interface Card (NIC) on the NIDS increases significantly, packets will start evading it, simply because it does not have the capacity to analyze them all (Barber, 2001).

Another more recent effort to assess the behaviour of NIDSs, and especially Snort, was conducted by Broucek and Turner (2004). This research used Snort to observe the hit rate, and false-positives generated when monitoring a web server over a two month period. It was found that Snort was still a subject to a number of false-positives, even after the use of fine-tuned rules. Additionally, the information provided in the Snort logs was not sufficient to trace attackers, once attacks were discovered. Only IP addresses were available for inspection. Snort also had severe difficulties in analyzing encrypted communication, especially tracing packets back to the attackers.

On a different note, another interesting effort which attempts to improve the detection capability of IDSs, uses a keyword selection method. The intention is to make the IDS smarter by counting keywords and calculating the risk of the attack probability. This technique improved the hit rate of an IDS by 80%, without increasing the false alarms (Lippmann and Cunningham, 2000).

It is evident that the detection capabilities of NIDS are far from perfect. The aim of this study is to complement existing work by looking into more recent versions of IDS and attacking software.

## 3. Design of experiments

The experiments utilized three essential components: a web server, an attacking computer running Nikto, and the NIDS installed on a monitoring computer. All these devices are connected to each other via a hub, enabling the NIDS to monitor all traffic between the attacking computer and the web server. The NIDS was also configured to have a NULL IP. This was because of two important factors:

- The NIDS should stay hidden to prevent attackers from noticing it
- The NIDS should not be able to send packets through the monitoring NIC, only receive.

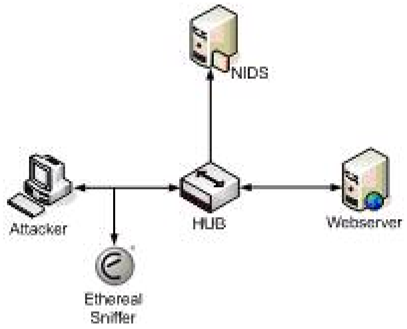

**Figure 1:** Network topology of experiment and traffic flow

Figure 1 illustrates the network packet flow and the placement of the devices. The Ethereal sniffer is installed on the attacker computer to analyse detected traffic. It is important to note that the NIDS only receives packets; it is not allowed to send any. The main reason for that was to ensure that it does not interfere with the network traffic (Kistler, 2003). The attacking PC uses the network tool Nikto, with its built-in evasion techniques. The NIDS consists of a laptop installed with Snort IDS version 2.4.5. The research accomplished four different experiments, each one with dissimilar goals, test setup and configuration:

**Table 1:** Description of experiments

| Test Description | Test Relevance |
|---|---|
| *Test1* is an experiment where focus is upon Snort's detection abilities when submitted with mutated evasion packets. | Snort's performance can be compared to other IDS |
| *Test2* is dedicated to finding out how Snort reacts when the CPU runs at maximum capacity (Snort recommends at least 300 MHz and 128 RAM for a low level traffic network) (Bull, 2002). | How would Snort perform if installed on a busy network or overloaded client? |
| *Test3* is designed to find how new modified signatures affect the results. This experiment is much like Test1, with the exception of the adding of the new signatures. | Research upon writing own signatures and configuration options in Snort. |
| *Test4* combines more evasion techniques to each attack packets using a method in Nikto which allows several evasion techniques at once. The goal of this experiment is to see if the research modified signatures still alert for complex packets involving several evasion techniques combined. | To what degree does several evasion techniques combined together affect the attacks, and the newly created signatures. |

These experiments were conducted in the same manner to produce the most accurate results. Where there are unexpected results, the tests were run several times to produce more stabile results. This was to eliminate incidents with extreme results only occurring once.

## 4. Results

### 4.1 Results for Test1 – Snort's detection engine efficiency

Test1 is designed to find the efficiency of Snort when it comes to the different available evasion techniques that Nikto-1.35 offers. Nikto offers nine (9) evasion techniques, each of which were tested one by one, to find what kind of techniques create more problems for the Snort sensor.

It was found that Snort alerted differently but not randomly. There where several similarities in the alerts on all evasion techniques, but also similarities in the attacks that were not detected. Specifically, the number of total attacks sent from Nikto varied by just a few packets per experiment, while the Snort detection varied much more frequently. Snort normally alerted for about 50% of the total evasion attack packets that where on the wire. The exceptions occurred when Nikto used its evasion technique 4 "prepend long random string to request". Snort actually alerted for more than the total evasion packets sent on this occasion (see Figure 2). Technique 4 prepends long random strings to request, making all packets have an unordinary long string in the GET request that is sent to the web server. This method causes the NIDS to react more often compared to the other techniques, because of the fact that more packets look dangerous based on the fact that they all have these long stings.

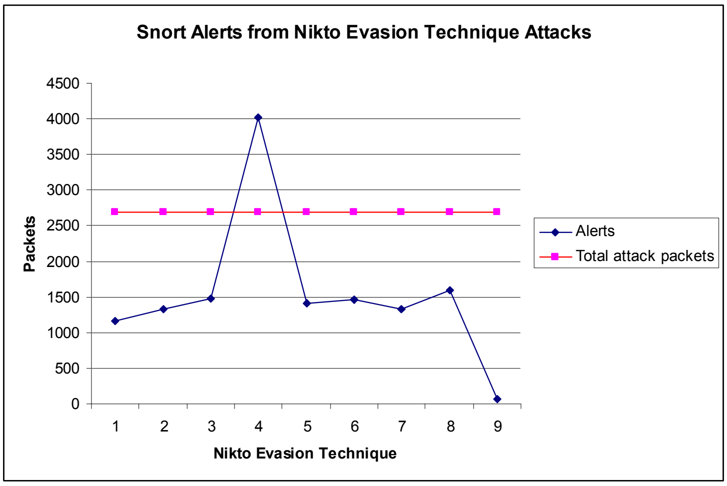

**Figure 2:** Snort's alerts compared to the total evasion attack packets sent

Figure 2 also shows a relative small alert figure when Nikto used its evasion technique nine (9) "session splicing". This number is to be taken lightly, because of an incomplete experiment when testing this method. This technique split up the attacking string in many small fragments that have to be reassembled at both the NIDS and the web server before processing. As a result, this attack takes many hours, and the research had to abort the logging after a time period. Also, it generates significantly larger amount of traffic, which was too time consuming and complex to analyse. However, it is worth pointing out that a NIDS would have similar difficulties in analysing attacks with this technique especially in a busy network.

## 4.2 Results for Test2 – Snort Detection under extreme conditions

This experiment was designed to stress the NIDS, forcing it to make mistakes while analyzing packets. It was conducted by running Snort in verbose mode, which results in the NIDS displaying all packets received on its interface. This can have a significant effect on the CPU and memory consumption on the NIDS. The goal was to see how Snort reacted when it had less processor capacity and memory than needed. As seen on Figure 3, Snort started dropping packets after a few minutes of the experiment.

The preliminary results clearly indicated that the NIDS did not manage to process as many packets as it received. The NIDS used 100% of its processor, focusing only on handling the Snort application. While Nikto managed to send about 1000 packets per 15 seconds, Snort had problems analyzing those packets. Snort dropped around 50% of the packets when face upon the evasion attacks, with the exception of evasion technique nine, which was not the focus of this work.

Finally, it should be noted that some tests have been performed several times to minimize the risk of wrong numbers and statistics. Where unusual numbers have appeared, more tests have been performed to analyze if this was a one time incident or a continuous event.

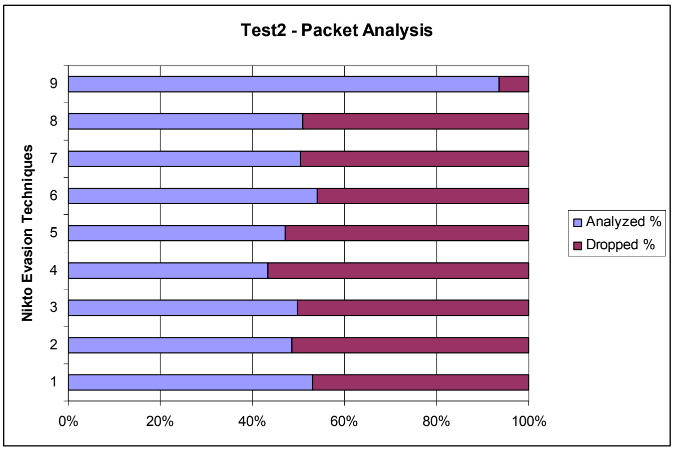

**Figure 3:** Snort's alerts compared to the total evasion attack packets sent

## 4.3 Results for Test3 – Enhanced Signature Testing

Based on the results from Tests 1 and 2, the next stage of the research was to improve Snort rules and compare their effectiveness to existing ones. Therefore the aim of Test3 is to test the improved rules and compare them against the results from Test1 to find if the alert rate has improved.

### *4.3.1 Signature improvements*

After analysing the NIDS alerts from Test1, it was found that evasion technique two, *self-reference directory,* was in greater need for further improvement, due to its severity and ease of exploitation. The first step was the analysis and improvement of all passwd attacks in Test1, mainly due to the severity of these attacks.

Figure 4 illustrates a few etc/passwd stings that evaded the NIDS:

```
GET /./forum-ra.asp?n=/./etc/./passwd HTTP/1.0

GET /./cgi-bin/./netsonar;cat\t/./etc/./passwd|?data=Download HTTP/1.0
```

**Figure 4:** Examples of etc/passwd attacks

As seen above, these attack strings use a similar string format. Both of them use the string "/./" instead of "/", which may look as a different string to "/", but will actually run in the same way on a target system. The reason for Snort not detecting these kinds of signatures is simply because it is not looking for all these variations of attack strings. Snort normally detects packets containing the string "etc/passwd", but when this string is modified, the NIDS cannot detect it, even if the target web server will effectively perceive it the same way as the "etc/passwd".

In order to enhance the etc/passwd rule, the existing signature was modified to alert when the separate strings "etc", "/./" and "passwd" are used in the same request. The modified rule section is depicted in Figure 5.

```
content:"etc"; nocase; content:"/./"; content:"passwd"; nocase;
```

**Figure 5:** Enhanced "etc/passwd" rule section

In the next step, rules attempting to get access to key files or directories "usr/bin", "etc/hosts" , and "boot.ini" were modified in a similar manner. Examples of resulting rules are depicted in Figure 6.

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-CGI self reference
directory on usr bin"; flow:to_server,established; content:"/./"; content:"usr"; nocase;
content:"bin"; nocase; reference:jay_usr_bin,2006-1806; classtype:web-application-attack;
sid:3333332; rev:2;)

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-CGI hosts self reference
directory attempt"; flow:to_server,established; content:"etc"; nocase; content:"/./";
content:"hosts"; nocase; reference:jay_etc_hosts,2006-1206; classtype:web-application-
attack; sid:3333333; rev:2;)

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-CGI hosts directory
traversal attempt"; flow:to_server,established; content:"/./"; content:"boot.ini"; nocase;
reference:jay_/./_boot.ini,2006-1806; classtype:web-application-attack; sid:3333334;
rev:1;)
```

**Figure 6:** Examples of enhanced rules

All the modified rules are usually alerted by Snort, except when self-reference evasion techniques are being used. The aim of introducing the enhanced rules was to improve Snort's detection, without introducing false positives at the same time.

### *4.3.2 Test Results – With enhanced signatures*

After applying the enhanced signatures in the Snort configuration, Test3 aimed to perform another series of experiments, to find if these new signatures had improved Snort's detection. The exact same attacks were performed to make the results comparable with Test1.

The results (see Figure 7) showed us that the new signatures increased Snort's hit rate on all evasion methods, except method one and four (method nine was not included in this test). As the chart shows, we have an increase of alerts on evasion method two of about 250. This large increase is mostly because of enhancing self-reference directory in combination with "etc/passwd", "etc/hosts" and "usr/bin". The rest of the evasion techniques get a small increase as well, but obviously not as significant as technique 2. Nikto evasion technique one and four show a decrease of one hit and increase of two hits. These two results should be ignored as they are not a product of the research enhanced signatures, but because of result and packet fluctuations.

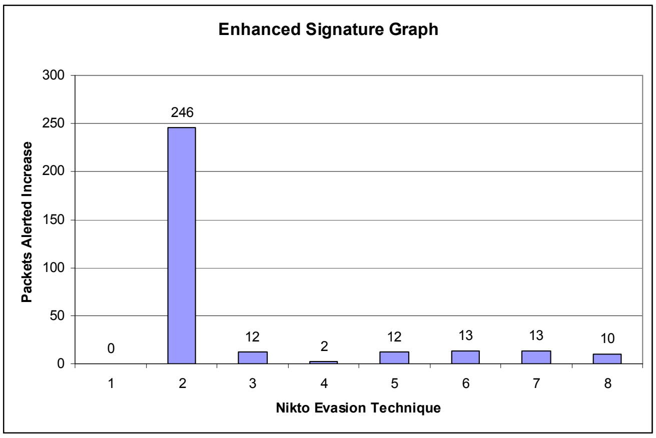

**Figure 7:** Alerts increase with new signatures

## 4.4 Results for Test4 – Combining Nikto Attacks

Test4 aims to test the difference that the new enhanced rules can have when Nikto combines more attacks together for a more complex attack. As examples this experiment has combined evasion method two and three together, and five combined with six. These results will hopefully show that the enhanced signatures are functionally even when more methods are combined together.

As Figure 8 illustrates, the enhanced rules have a large impact in instances when evasion method two is combined with others. This test proved that the research's newly created signatures work even when Nikto mutates its attack packets by using several techniques per packet. Specifically, there is a steep increase when Nikto combines method two and three, and a smaller increase when Nikto uses evasion method five (5) and six (6).

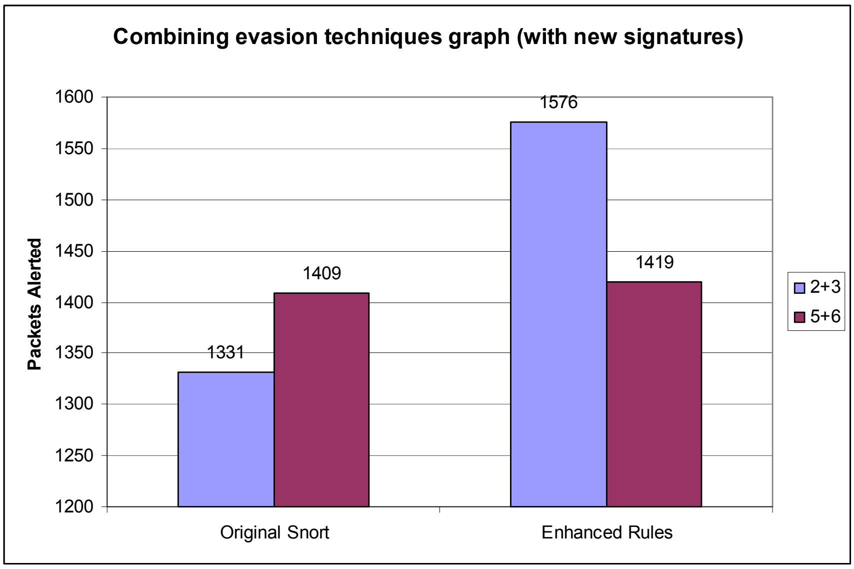

**Figure 8:** Packet increase alerts with new signatures

## 5. Discussion

The reason for Test1 - evasion technique four (4) triggering the high number of alerts, is because of two factors. Firstly, all packets look almost identical sending around 700 bytes of random strings before the actual request string in the end. This leads to Snort triggering off an alert per packet. Secondly, Snort will also alert on packets with malicious requests in the end, meaning each attack packet can get several alerts. This leads to the high number of alerts. When using evasion technique nine (9) the low number of alerts is simply because the experiments never ended. Session splicing requires many hours to complete, and the research had to abort after about five hours. Parallels can be drawn however, towards a NIDS on a busy network encountering session splicing attacks. If the NIDS is quite busy, it will not have the ability waiting for all session packets for reassembly, thus the packets will evade the NIDS.

If the NIDS is installed on a computer that is occupied with other duties as well as the intrusion detection system, it can be subject to dropping packets. As Test2 show the NIDS will drop packets if its resources are fully engaged. The most ideal placement of a NIDS would be on a separate node in a network, with no other applications and a processor of at least 1 GHz and memory of preferably 512 kb or more. With these resources the NIDS have the ability to withstand a high load on the network.

The area of which the research had focus on creating new signatures was towards Nikto's evasion technique two "add directory self-reference /./". These new signatures improved Snort's hit rate greatly when Nikto used this technique, and also when this technique was combined with most of the other evasion techniques. The command "nocase" were added to all signatures to prevent evasion techniques evading the signatures by using random casing. The problem occurred when Nikto used its evasion technique one (1) "random URI-encoding". The new signatures did not alert when this technique was used, or any other in combination with this one. This is why Figure 8 shows no increase in alerts using evasion technique one. All the other evasion techniques had an improved hit rate when using the new signatures (except session splicing technique, which was not tested adequately.

## 6. Future work

The work done by Lippmann and Cunningham in 2000 was quite novel, trying to make the IDS smarter. The authors believe that a similar approach, where Snort analyzes the whole packet in a different way, instead of scanning the requests for incidents, is needed. A possible solution that is worth investigating further is to divide the request content into smaller strings, and then to analyze each string a danger level classification (see example in Table 2). If the total level of danger for the entire request exceeds a limit, an alert would be raised. This method would eliminate any incidents where one attack packet gets several alerts, but most importantly it would make the Snort IDS more dynamic. This approach is more dynamic in the way it divides and conquers any mutated packets trying to bypass its systems.

**Table 2:** Content separation and danger classification

| Content | Danger level (1-10) |
|---|---|
| /./ | 4 |
| cgi-bin | 2 |
| /./ | 4 |
| passwd | 8 |
| Total: | 18 |

Another advantage of changing Snort's detection capabilities is the fact that when an attack packet is alerted in Snort now, a log is created naming the alert with details. The problem lies with the incidents where one packet triggers off more than one alerts, creating several alerts for only one packet. If an administrator is to analyze the logs, he has to find all alerts for the specific packet, for then to improve the signatures. With this proposed change in Snort's detection engine using danger level classifications, Snort would analyze the whole request, dividing it into smaller strings, finding the danger level of each small string, and finally alerting or not. If the string is alerted, the log would contain more useful information for the administrator, like what incidents triggered off the most danger levels and how the format of the attack looked like.

A significant problem with this approach would be where to actually separate each of the strings. Another problem would be how to design an application able to handle this method of separation and analyzing at the same time.  In any case, the application and rules would need a lot of testing and modifications on the rules and danger levels.

## 7. Conclusions

The presented work investigated Snort's capabilities at detecting IDS evasion attacks. It was found that IDS evasion techniques are still an effective method of bypassing detection, and that Snort has not improved enough to detect old, and widely available, IDS evasion techniques. Snort without any special customized configuration and new rules detected around 50% of evasion-only packets sent from Nikto. Also, another parameter that needs to be considered when trying to improve Snort 's detection capability is its loading. When Snort NIDS CPU was overloaded it started dropping packets, but it was consistent in all experiments around this. It analysed as many packets as it could, until further packets were suddenly dropped.

Finally, it was demonstrated that the proposed signatures could enhance Snort's detection capability. The limitations of this work, is the fact that the enhanced signatures will need some more work to be fully operational. The problem that they do not alert when Nikto uses evasion method one is a problem. Also, it is important to extend this work further and investigate session splicing techniques fully.

## References

Anderson, J.P. (1980), "Computer Threat Monitoring and Surveillance", (In Anderson, J.P. Technical report, Fort Washington, Pennsylvania)

Barber, R. (2001), "The Evolution of Intrusion Detection Systems – The Next Step", Computers & Security, Vol. 20, pp 132-145

Broucek, V., Turner, P. (2004), "Intrusion Detection: Issues and Challenges in Evidence Acquisition", Internal Review of Law Computers & Technology, Vol. 18 No.2, pp 149-164

Bull, J. (2002), "Snort's Place in a Windows 2000 Environment", www.snort.org/docs/snort-win2k.htm (Accessed 19 December 2006)

Chapple, M. (2003), "Evaluation and tuning an intrusion-detection system", searchsecurity.techtarget.com/tip/1,289483,sid14_gci918619,00.html?track=IDSLG, (Accessed 4 January 2007)

Cirt.net (2006) "Nikto-1.35", http://www.cirt.net/code/nikto.shtml (Accessed: 18-Dec-06)

Graham, I. (2006), "Achieving Zero-loss Mutli-gigabit IDS – Results from Testing Snort® on Endace Accelerate Multi-CPU Platforms", www.touchbriefings.com/pdf/2259/graham.pdf (Accessed 3 January 2007)

Kistler, U. (2003), "Snort IDScenter 1.1 manual", www.engagesecurity.com/docs/idscenter/ (Accessed 30 December 2006)

Lippmann, R.P., Cunningham, R.K. (2000), "Improving intrusion detection performance using keyword selection and neural networks", Computer Networks, Vol. 34, pp 597-603

Ptacek, T.H., Newsham, T.N. (1998) "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection", http://www.snort.org/docs/idspaper/ (Accessed: 25-May-06).

Rain Forest Puppy - rfp (1999), "A look at whisker's anti-IDS tactics", www.ussrback.com/docs/papers/IDS/whiskerids.html, (Accessed 30 December 2006)

Siddharth, S. (2005), "Evadion NIDS, revisited", www.securityfocus.com/infocus/1852, (Accessed 19 December 2006)

Sodiya, A.S, Longe, H.O.D and Akinwale, A.T. (2004), "A new two-tiered strategy to intrusion detection", Information Management & Computer Security, Vol. 12, No. 1, pp 27-44.