2010

# Evidence Fusion using D-S Theory: utilizing a progressively evolving reliability factor in wireless networks

Aqila Dissanayake
*University of Windsor*

# EVIDENCE FUSION USING D-S THEORY: UTILIZING A PROGRESSIVELY EVOLVING RELIABILITY FACTOR IN WIRELESS NETWORKS

By

Aqila Dissanayake

A Thesis
Submitted to the Faculty of Graduate Studies
Through the School of Computer Science
In Partial Fulfillment of the Requirements for
The Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada

2009

**EVIDENCE FUSION USING D-S THEORY: UTILIZING A PROGRESSIVELY EVOLVING RELIABILITY FACTOR IN WIRELESS NETWORKS**

By

Aqila Dissanayake

APPROVED BY:

_____

Dr. D. Kao
Odette School of Business

_____

Dr. R. Kent
School of Computer Science

_____

Dr. Akshai Aggarwal, Advisor
School of Computer Science

_____

Dr. Luis Rueda, Chair of Defense
School of Computer Science

December 17th, 2009

## AUTHOR'S DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

# ABSTRACT

The Dempster-Shafer (D-S) theory provides a method to combine evidence from multiple nodes to estimate the likelihood of an intrusion. The theory's rule of combination gives a numerical method to fuse multiple pieces of information to derive a conclusion. But, D-S theory has its shortcomings when used in situations where evidence has significant conflict. Though the observers may have different values of uncertainty in the observed data, D-S theory considers the observers to be equally trustworthy. This thesis introduces a new method of combination based on D-S theory and Consensus method, that takes into consideration the reliability of evidence used in data fusion. The new method's results have been compared against three other methods of evidence fusion to objectively analyze how they perform under Denial of Service attacks and Xmas tree scan attacks.



**Keywords:** Dempster-Shafer, Theory of Evidence, Intrusion Detection, Multi Sensor Data Fusion, Consensus Operator

# DEDICATION

I dedicate this thesis to my parents.

# ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Dr. Akshai Aggarwal, for guiding me throughout the three years I've been a student at the University of Windsor. Without his help and direction I would not have been able to complete my thesis.

I would like to thank my committee members, Dr. Robert Kent and Dr. Diana Kao, for their valuable comments and suggestions towards my thesis. I would also like to thank Dr. Luis Rueda for being the chair in the examination committee.

I would like to thank my parents for encouraging me and supporting me to reach higher goals in life. Also, I would like to thank numerous friends who helped me in many ways to conduct experiments and research by providing me equipment, ideas, and other things. Special thanks go to Dinusha, Himani, and Debashis, for helping me numerous times to conduct experiments. Last but not least I would like to thank Lakmini for being there for me and inspiring me to complete my studies.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# 1 INTRODUCTION

The rapid growth of the Internet and its related network infrastructure is changing computing as we know it. Especially in the last decade we have seen the greatest leap in wireless technology which has given new meaning to networking. Unlike old days where one needed a physical connection to connect to the Internet, these days one can connect to the Internet using a wireless network connection through a computer or even one's mobile phone device. Unfortunately, along with the facility of wireless connectivity have come risks of malicious intrusions. Though intrusion detection in wired networks is a well explored subject, intrusion detection in wireless networks is yet to be studied to the same extent. Timely detection of intrusions in wireless networks and appropriate responses remain extremely important areas of research. A security breach can cause mission critical systems to be unavailable to end-users causing millions of dollars worth of damage. If the next generation of Internet and network technology is to operate successfully, it will require a set of tools to analyze wireless networks and detect and prevent intrusions. A large distributed network, particularly with facilities of wireless connectivity, would need multiple sensors to be able to catch intrusions of all types. Alerts may be generated by the sensors or the associated Intrusion Detection Systems. These alerts provide evidence, which has to be used for generating reliable information about intrusions, even though it is known that none of the sensors or intrusion detection systems are fully reliable.

According to the Wikipedia, intrusion detection is the act of detecting actions that attempt to compromise the confidentiality, integrity or availability of a resource. Intrusion detection is a difficult process that requires security practitioners to have a deep understanding of networks and their functionality. Finding an accurate attack signature is extremely challenging even if we know the network is under attack. This is because the signature needs to be narrow enough to differentiate between normal legitimate traffic and attack traffic. Good intrusion detection is completely dependent on this property. If the attack signature is not accurate it will cause "False Positives" and "False Negatives". If the intrusion detection system gives too many false positives, that would mean that the security practitioner who is responsible for checking the alerts and tracing them would waste a lot of time on false positives. On the other hand, if the intrusion detection system does not give an alert when there is an actual attack that would be bad as this means that the security practitioner is unaware that his system is under attack. Though, some intrusion detection systems automate some aspects of the process, the intervention of the security practitioner is very much required to complete the process of good intrusion detection. Ideally, the goal of a good intrusion detection system is to lower the false positive rate and the false negative rate.

At present, completely preventing intrusions and other unauthorized actions appear unrealistic. Due to the rapid growth of the Internet and the vast array of possibilities it has opened up more and more systems that have become the target of intruders. So, it has become critical to detect these intrusions in a timely manner and carry out necessary preventative

measures to track down the attackers and discourage future attacks. There are many intrusion detection systems (IDS) and intrusion prevention systems (IPS) in the market today which facilitate in identifying intrusions and taking necessary preventative measures. New techniques are developed every year to make these IDS/IPS systems work more accurately and efficiently. One of these new techniques is to combine evidence gathered through multiple systems or access points to arrive at a more accurate result. The theory associated with this new technique is known as the Dempster-Shafer Theory of Evidence. The Dempster-Shafer theory is also known as D-S theory of evidence.

Research on intrusion detection has been going on for more than two decades. However research on intrusion detection using the D-S theory of evidence only started after the year 2000. Since then researchers have published around twenty papers that try to improve the idea of data fusion in intrusion detection using the Dempster-Shafer theory.

The National Technical University of Athens (NTUA) has been one of the main universities that has been conducting research on intrusion detection using the D-S theory. Three of the leading researchers in this field are also from NTUA. Vasilis Maglaris and Basil Maglaris of the NTUA have both published two papers on multi sensor data fusion for Denial of Service (DoS) detection using the D-S theory of evidence. Christos Siaterlis of the NTUA is the only researcher so far to publish three papers on intrusion detection using the D-S theory. Researchers from the Florida International University (FIU) have also been involved in research related to D-S theory and intrusion detection. Two of their researchers, Te-Shun Chou and Kang K. Yen have also published two papers each in the area. No other researcher in this field has published more than one paper.

# 2 DEFINITIONS AND RELATED CONCEPTS

## 2.1 Dempster Shafer Theory

The Theory of Evidence is a branch of mathematics that is concerned with using evidence to calculate the probability of an event. The Dempster-Shafer theory (D-S theory) is a theory of evidence used to combine separate pieces of evidence to calculate the probability of an event. The Dempster-Shafer theory was introduced in the 1960's by Arthur Dempster [1968] and developed in the 1970's by Glenn Shafer [1976]. According to Glen Shafer the D-S theory is a generalization of the Bayesian theory of subjective probability.

The Dempster-Shafer theory can be viewed as a method for reasoning under epistemic uncertainty. A major advantage of the Dempster-Shafer theory in an intrusion detection environment is its ability to combine evidence provided by different observers. These observers could even be completely different and located remotely from each other. Each observer could provide its own perceived state to a central server which will combine the evidence to determine

the final state of the network. The most important part of this theory is Dempster's rule of combination which combines evidence from two or more sources to form inferences.

In the Following sub-section we shall describe some of the important definitions that are needed to understand this thesis.

### 2.1.1 The Frame of Discernment (Θ)

A complete (exhaustive) set describing all of the sets in the hypothesis space. Generally, the frame is denoted as Θ. The elements in the frame must be mutually exclusive. If the number of the elements in the set is n, then the power set (set of all subsets of (Θ) will have $2^n$ elements.

### 2.1.2 BPA (Basic Probability Assignment)

The theory of evidence assigns a belief mass to each subset of the power set. It is a positive number between 0 and 1. It exists in the form of a probability value.

If Θ is the frame of discernment, then a function,

$m: 2^{\Theta} \rightarrow [0, 1]$ is called a BPA, whenever

$m(\varnothing) = 0$, and

$m(A) \geq 0, \forall A \subseteq \Theta$

$\Sigma\, m(A) = 1$ and

$A \subseteq \Theta$

### 2.1.3 Belief (Bel)

Given a frame of discernment Θ and a body of empirical evidence $\{m(B_1), m(B_2), m(B_3)\ldots.\}$, the belief committed to $A \varepsilon \Theta$ is

$Bel(A) = \Sigma\, m(Bi)$

$B \subseteq A$

Also, $Bel(\Theta) = 1$

### 2.1.4 Plausibility Function (Pl)

The plausibility (Pl) is the sum of all the masses of the sets B that intersect the set of interest A:

$$Pl(A) = \Sigma\, m(Bi)\ ,\ \ B \mid B \cap A \neq \varnothing$$

### 2.1.5 Belief Range

The interval [Bel (A), Pl (A)] is called the belief range. Plausibility (Pl) and Belief (Bel) are related as follows

$$Pl(A) = 1 - Bel(\bar{A})$$

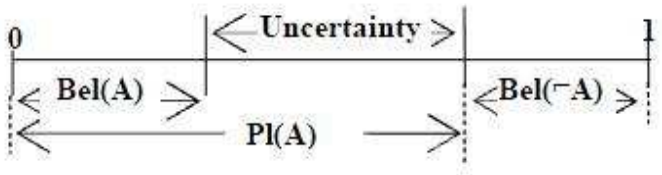


**Figure 2-1. Relationship between Plausibility and Belief**

### 2.1.6 Dempster's Combination Rule

The combination called the joint mass ($m_{12}$) is calculated from the two sets of masses $m_1$ and $m_2$.

$$m_{12}(A) = \frac{B \cap C = A,\ \Sigma\, m_1(B)\, m_2(C)}{1 - [B \cap C = \varnothing,\ \Sigma\, m_1(B)\, m_2(C)]} \qquad m_{12}(A) \neq \varnothing$$

- $m_{12}$ (A) = Combined belief of the hypothesis A
- $m_1$ (B) = Belief committed to B as seen by the first observer
- $m_2$ (C) = Belief committed to C as seen by the second observer

In this equation $[B \cap C = \varnothing, \Sigma\, m1(B)\, m2(C)]$ part in the denominator is known as K.

$$K = [B \cap C = \varnothing, \Sigma\, m1(B)\, m2(C)]$$

K represents basic probability mass associated with conflict. K is calculated by summing the products of the BPA's of all sets where the intersection is null.

## 2.2 CONSENSUS OF OPINIONS AND RELATED DEFINITIONS

The definitions in this sub section are derived from the paper "A logic for uncertain probabilities" by Dr. Audun Jøsang [Jøsang, 2000] and "The Consensus Operator for Combining Beliefs" [Jøsang, 2002].

### 2.2.1 Disbelief Function

Let $\Theta$ be a frame of discernment, and let $m_\Theta$ be a Belief Mass Assignment (BMA) (same as BPA or Basic Probability Assignment) on $\Theta$. Then the disbelief function corresponding with $m_\Theta$ is the function d: $2^\Theta \rightarrow [0, 1]$ defined by:

$$d(x) = \Sigma\, m_\Theta(y), \qquad x,y \in 2^\Theta$$

$$y \cap x = \varnothing$$

### 2.2.2 Uncertainty Function

Let $\Theta$ be a frame of discernment, and let $m_\Theta$ be a Belief Mass Assignment (BMA) on $\Theta$. Then the uncertainty function corresponding with $m_\Theta$ is the function u: $2^\Theta \rightarrow [0, 1]$ defined by:

$$u(x) = \Sigma\, m_\Theta(y), \qquad x,y \in 2^\Theta$$

$$y \cap x \neq \varnothing$$

Total uncertainty can be expressed by assigning all the belief mass to $\Theta$. The belief function corresponding to this situation is called the vacuous belief function. A BMA with zero belief mass assigned to $\Theta$ is called a dogmatic BMA.

### 2.2.3 Relative Atomicity

Let $\Theta$ be a frame of discernment, and let $x,y \in 2^\Theta$. Then for $y \neq \varnothing$ the relative atomicity of x to y is the function a : $2^\Theta \rightarrow [0, 1]$ defined by:

$$a(x/y) = |x \cap y| / |y| , \quad x,y \in 2^\Theta, y \neq \varnothing$$

It can be observed that $x \cap y = \varnothing \rightarrow a(x/y) = 0$ and that $y \subseteq x \rightarrow a(x/y) = 1$. In all other cases the relative atomicity will be a value between 0 and 1.

The relative atomicity of an atomic state to its frame of discernment, denoted by $a(x/\Theta)$, can simply be written as $a(x)$. If nothing else is specified, the relative atomicity of a state then refers to the frame of discernment.

### 2.2.4 Opinion

Let $\Theta$ be a binary frame of discernment with 2 atomic states x and $\neg x$, and let $m_\Theta$ be a BMA on $\Theta$ where b(x), d(x), u(x), and a(x) represent the belief, disbelief, uncertainty, and relative atomicity functions on x in $2^\Theta$ respectively. Then the opinion about x, denoted by $w_x$, is the tuple defined by:

$$w_x \equiv (b(x), d(x), u(x), a(x))$$

### 2.2.5 Consensus

The consensus opinion of two opinions is an opinion that reflects both opinions in a fair and equal way.

Let $W_x^A = (b_x^A, d_x^A, u_x^A, a_x^A)$ and $W_x^B = (b_x^B, d_x^B, u_x^B, a_x^B)$ be opinions respectively held by agents A and B about the same proposition x. Let $W_x^{A,B} = (b_x^{A,B}, d_x^{A,B}, u_x^{A,B}, a_x^{A,B})$ be the opinion such that, $K = u_x^A + u_x^B - u_x^A u_x^B$. When $u_x^A, u_x^B \rightarrow 0$, the relative dogmatism between $W_x^A$ and $W_x^B$ is defined by $\gamma$ so that $\gamma = u_x^B / u_x^A$. Then $W_x^{A,B}$ is called the consensus between $W_x^A$ and $W_x^B$, representing an imaginary agent [A, B]'s opinion about x, as if she represented both A and B.

**For $K \neq 0$**

(1) $b_x^{A,B} = (b_x^A u_x^B + b_x^B u_x^A) / K$
(2) $d_x^{A,B} = (d_x^A u_x^B + d_x^B u_x^A) / K$
(3) $u_x^{A,B} = (u_x^A u_x^B)/K$
(4) $a_x^{A,B} = a_x^B u_x^A + a_x^A u_x^B - (a_x^A + a_x^B) u_x^A u_x^B / u_x^A + u_x^B - 2u_x^A u_x^B$

- $a_x^{A,B} = (a_x^A + a_x^B)/2$ when $u_x^A, u_x^B = 1$

- $b_x^A = r^A/r^A+s^A+2$
- $d_x^A = s^A/r^A+s^A+2$ where $u \neq 0$
- $u_x^A = 2/r^A+s^A+2$

The parameters **r** represents the amount of evidence supporting the actual event and the parameters **s** represents the amount of evidence supporting its negation.

**For K = 0**

(1) $b_x^{A,B} = \gamma\, b_x^{A} + b_x^{B} / \gamma + 1$
(2) $d_x^{A,B} = \gamma\, d_x^{A} + d_x^{B} / \gamma + 1$
(3) $u_x^{A,B} = 0$
(4) $a_x^{A,B} = \gamma\, a_x^{A} + a_x^{B} / \gamma + 1$

## 2.3 Denial of Service (DoS) Attack

According to the Wikipedia, "a denial-of-service attack (DoS attack) or distributed denial-of-service attack (DDoS attack) is an attempt to make a computer resource unavailable to its intended users." One of the common methods of attack involves saturating the target (victim) machine with communication requests that it cannot process legitimate requests or it responds so slowly as to render itself unavailable [Denial of Service Attack, Wikipedia]. In wireless networks, to perform a Denial of Service Attack, one requires a high-powered network interface card (NIC) [Denial of Service Attack, Wikipedia]. In our research we utilized a laptop computer equipped with a Wireless N, NIC to perform all DoS attacks.

According to [Denial of Service Attack, Wikipedia], the United States Computer Emergency Response Team defines symptoms of denial-of-service attacks to include:

1. Unusually slow network performance
2. Unavailability of a particular web site
3. Inability to access any web site
4. Dramatic increase in the number of spam emails received

In our research we extensively use packet flooding DoS attacks that saturate the victim with packets. This reduces the available bandwidth in the network to do legitimate work there by restricting a computer resource or making it unavailable for legitimate tasks.

## 2.4 Transmission Control Protocol (TCP)

"The Transmission Control Protocol (TCP) is one of the core protocols of the Internet protocol suite" [Transmission Control Protocol, Wikipedia]. "TCP is intended for use as a highly reliable host-to-host protocol between hosts in packet-switched computer communication networks, and in interconnected systems of such networks" [RFC 793]. "TCP is a connection-oriented, end-to-end reliable protocol designed to fit into a layered hierarchy of protocols which support multi-network applications" [RFC 793]. Figure 5-2 shows a TCP header. As one can see from this, 8 bits are allocated for flags. TCP based scan techniques set these flags to different values or combination of values in order to do the scanning. According to the Wikipedia [TCP] these flags are

1. CWR – Congestion Window Reduced (CWR) flag is set by the sending host to indicate that it received a TCP segment with the ECE flag set
2. ECE (ECN-Echo) – indicate that the TCP peer is ECN capable during 3-way handshake
3. URG – indicates that the URGent pointer field is significant
4. ACK – indicates that the ACKnowledgment field is significant
5. PSH – Push function
6. RST – Reset the connection
7. SYN – Synchronize sequence numbers
8. FIN – No more data from sender

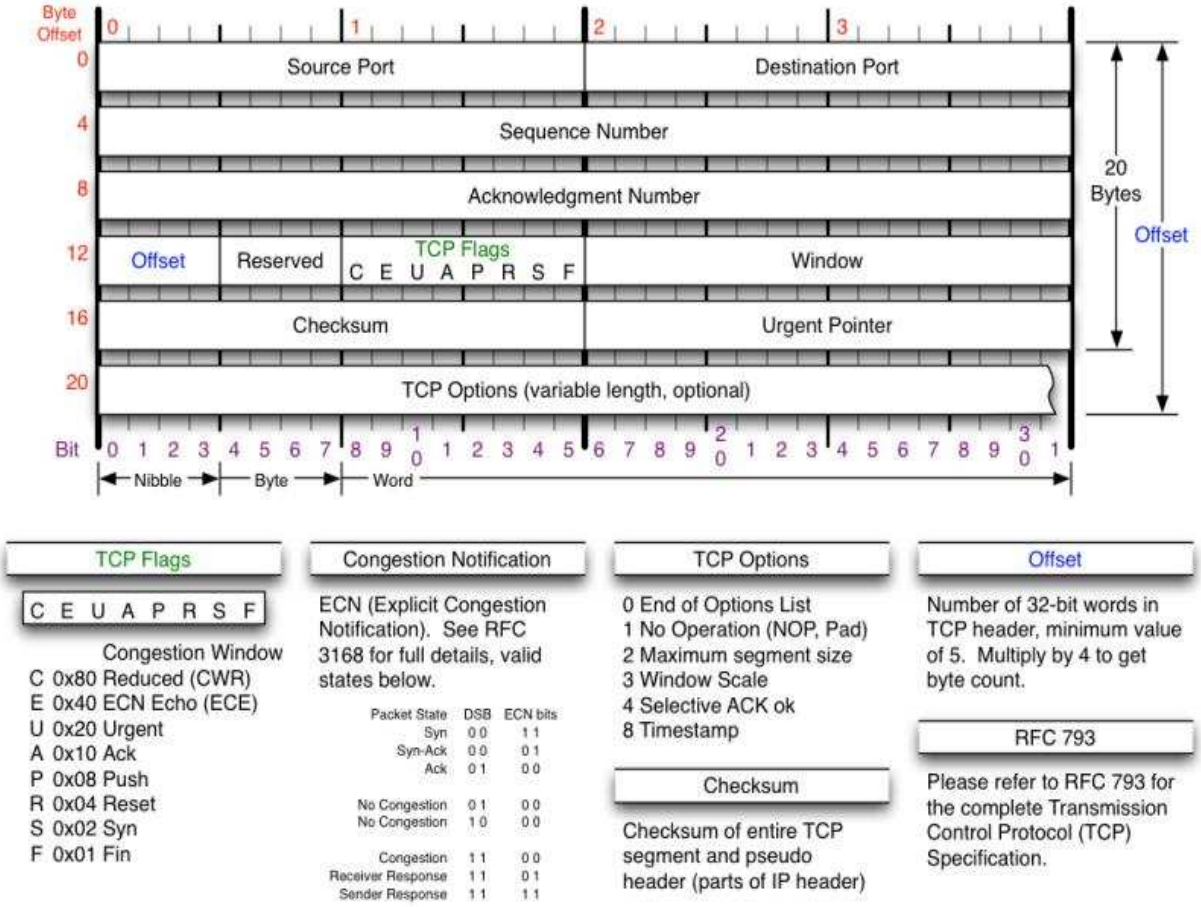


**Figure 2-2. TCP Header**

Adapted from http://nmap.org/book/images/hdr/MJB-TCP-Header-800x564.png

## 2.5 Port Scanning

"The most common type of network probe is probably the port scan" [TEO, 2000]. "A port scan is a method used by intruders to discover the services running on a target machine" [TEO, 2000]. By simply checking whether a given port is opened or closed, an attacker can determine whether

to attack that machine on that specific port or not. “For example, if the intruder finds that port 143 (the IMAP port) is open; she may proceed to find out what version of IMAP is running on the target machine. If the version is vulnerable, she may be able to gain super user access to the machine using an exploit” [TEO, 2000].

Port scanning can be conducted in many ways. The most well known port scanning techniques are listed below [Port Scanning Techniques, Insecure.org]

1. TCP connect scan
2. TCP SYN scan
3. TCP FIN scan
4. TCP null scan
5. TCP window scan
6. TCP ACK scan
7. TCP Maimon scan
8. Xmas tree scan
9. UDP scan
10. IP protocol scan
11. FTP bounce scan
12. Idle scan

### 2.5.1 Xmas Tree Scan

The Xmas tree scan exploits a subtle loophole in the TCP RFC to differentiate between open and closed ports [Port scanning techniques, Insecure.org]. According to Insecure.org “If the [destination] port state is CLOSED, an incoming segment not containing a RST causes a RST to be sent in response”. According to Nmap.org, when scanning systems compliant with the TCP RFC text, any packet not containing SYN, RST, or ACK bits will result in a returned RST if the port is closed and no response at all if the port is open. According to Nmap.org as long as none of those bits are included, any combination of the other three (FIN, PSH, and URG) are ok. Nmap exploits this with the Xmas tree scan.

In a Xmas tree scan, if a RST packet is received, the port is considered closed. This is illustrated by the diagram below.

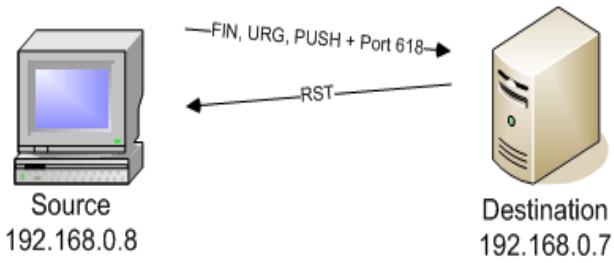


**Figure 2-3. Closed Port in a Xmas Scan**

Adapted from http://www.networkuptime.com/nmap/page3-5.shtml

A no response means it is open or filtered. The port is marked filtered if an ICMP unreachable error (type 3, code 1, 2, 3, 9, 10, or 13) is received. This scenario of not receiving a response is displayed below.

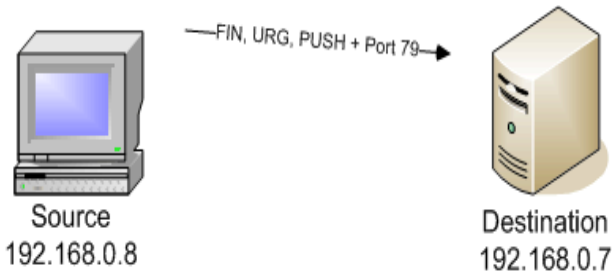


**Figure 2-4. Open or Filtered port in a Xmas Scan**

Adapted from http://www.networkuptime.com/nmap/page3-5.shtml

According to the Wikipedia a key advantage of these scan types is that they can sneak through certain stateless firewalls. That makes the Xmas tree scan stealthier than a regular SYN scan. Xmas tree packets are not commonly present in networks and indicate a high probability of network reconnaissance activities [Christmas tree packet, Wikipedia]. Luckily though, intrusion detection products and advanced firewalls can be configured to detect these types of reconnaissance scans. Snort intrusion detection system will alert on a Xmas tree scan which was tested in our research work.

Since there are systems that do not follow RFC 793, some systems send RST responses to the probes regardless of whether the port is open or not [Port scanning techniques, Insecure.org]. This will result in all ports being labeled as closed. This behavior is shown by Microsoft Windows and many Cisco devices [Port scanning techniques, Insecure.org]. However, this scan will work against most UNIX based system [Port scanning techniques, Insecure.org]. Also, these scans can't distinguish open ports from certain filtered ones, leaving one with the response open or filtered [Port scanning techniques, Insecure.org].

### 2.5.2 Packet Design for the Xmas Tree Scan

We shall demonstrate the packet design for a Xmas tree scan using CommView. As explained earlier, a TCP packet contains certain flags which should be activated to do a Xmas tree scan. A Xmas tree scan sends a TCP packet to a remote device with the URG, PUSH, and FIN flags set [Xmas tree scan, Networkuptime]. “This is called a Xmas tree scan because of the alternating bits turned on and off in the flag byte (00101001), much like the lights of a Christmas tree” [Xmas tree scan, Networkuptime]. Using CommView we set the TCP flags to match the

value 00101001 which corresponds to 29 in hexa-decimal. The flag structure will be displayed in the following way in CommView.

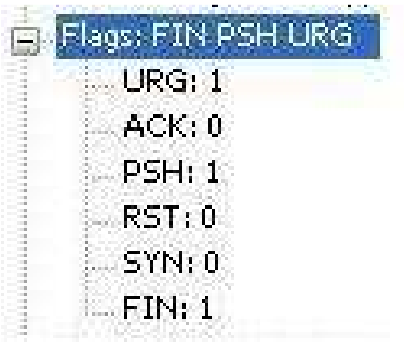


**Figure 2-5. Xmas Tree Packet with FIN PSH URG Flags Set**

# 3 EQUIPMENT AND SOFTWARE USED IN THE EXPERIMENTS

## 3.1 Aruba AP-70 Sensors

According to Aruba Networks AP-70 datasheet, the "Aruba AP-70 is a dual-radio indoor wireless access point capable of supporting functions including WLAN access, air monitoring/wireless intrusion detection and prevention, and secure enterprise mesh across the 2.4-2.5 GHz and 5 GHz RF spectrums." In our research two Aruba AP-70 sensors were used to do intrusion detection through the RF protect intrusion detection system.

**Figure 3-1. Aruba AP-70 Sensor**

## 3.2 CommView

According to the CommView website, CommView is a network monitor and analyzer that provides a picture of the traffic flowing through a PC or LAN segment. In our research we will be using CommView to construct TCP, UDP and ICMP packets for various attacks and create Denial of Service (DoS) attacks using its built in packet flooder. For creating TCP packets, CommView provides a hex editor that provides the facility of setting each flag of a TCP packet to the desired value. Further, CommView provides an interface to flood packets to a network at a maximum rate of 5000 packets per second. Given below is a screen shot of CommView's hex editor. It also shows its packet flooding interface with parameters such as "packets per second, packet size".

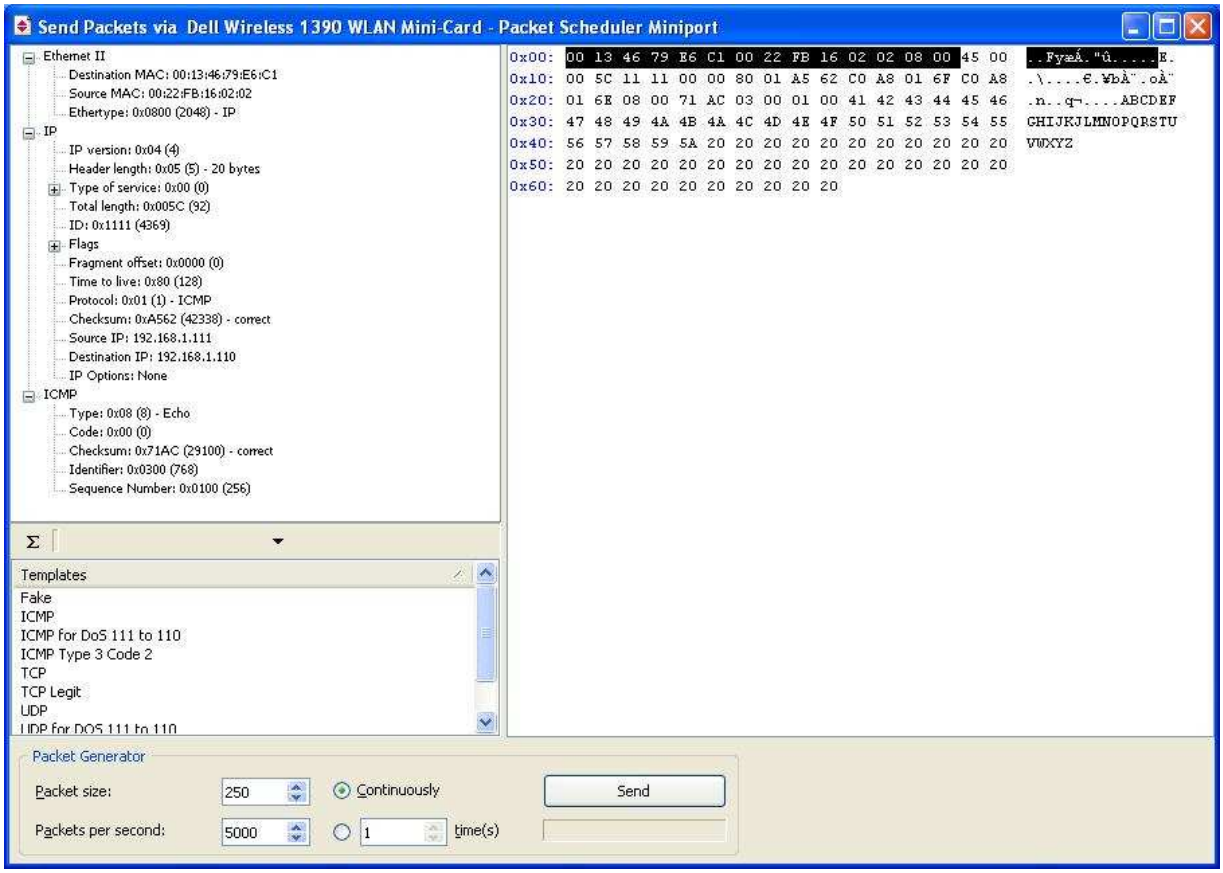


**Figure 3-2. CommView Hex Editor**

## 3.3 Nmap/Zenmap

According to Nmap.org, Nmap or "Network Mapper" "is a free and open source utility for network exploration or security auditing." "Nmap uses raw IP packets in to determine what hosts are available on the network, what services (application name and version) those hosts are

offering, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and other characteristics" [Nmap]. It was designed to scan large networks, but can work to scan single hosts also." Recently Nmap added an advanced GUI interface, called Zenmap. We have used Zenmap in our experiments to conduct Xmas tree attacks.

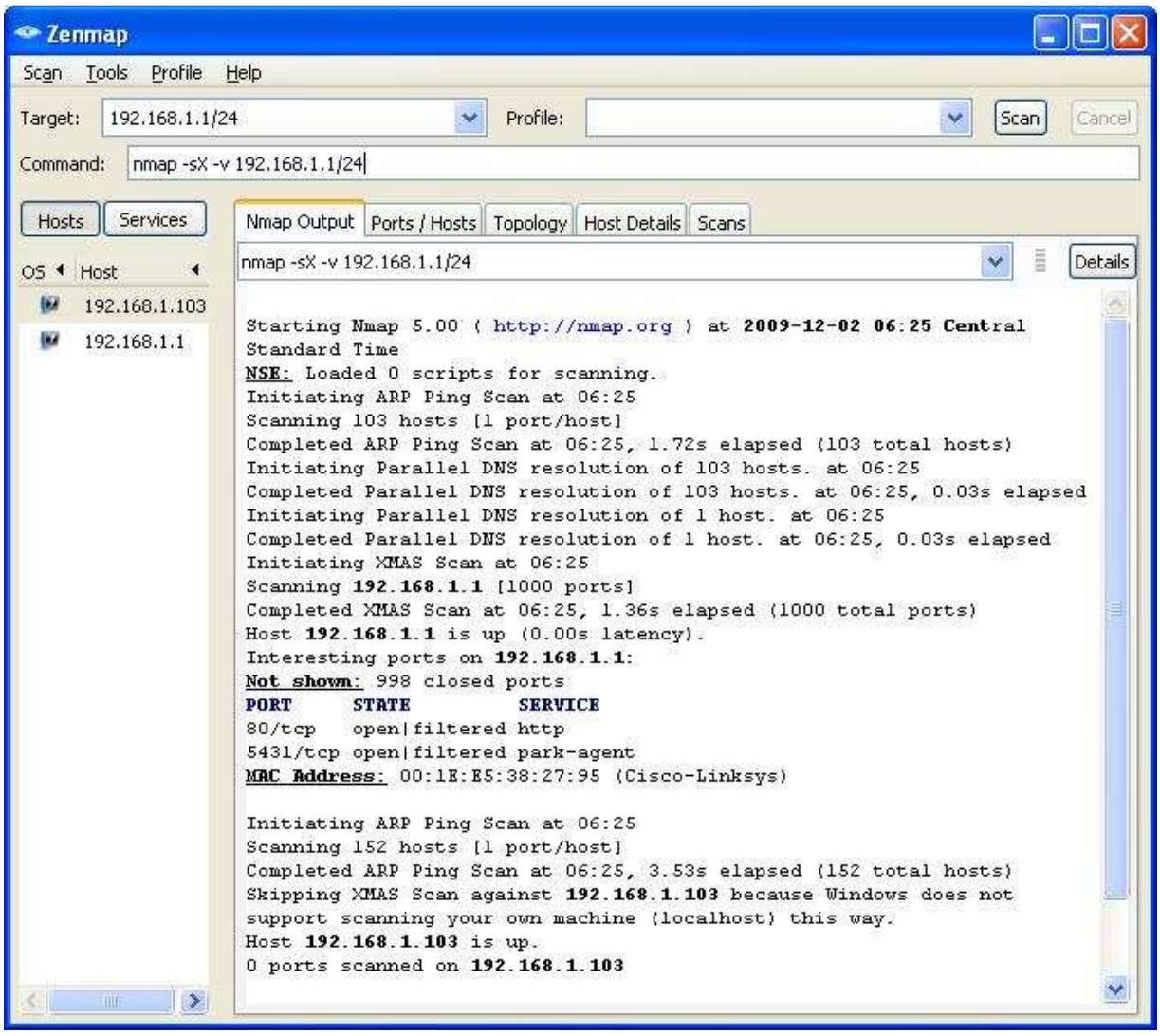


**Table 3-1 - Zenmap doing an Xmas Tree Scan**

## 3.4 Snort

According to Snort.org, "Snort is an open source network intrusion prevention and detection system (IDS/IPS) developed by Sourcefire." Currently it combines the benefits of signature, protocol and anomaly-based inspection and is a widely deployed IDS/IPS technology. In our research we will use a computer installed with Snort to do intrusion detection.

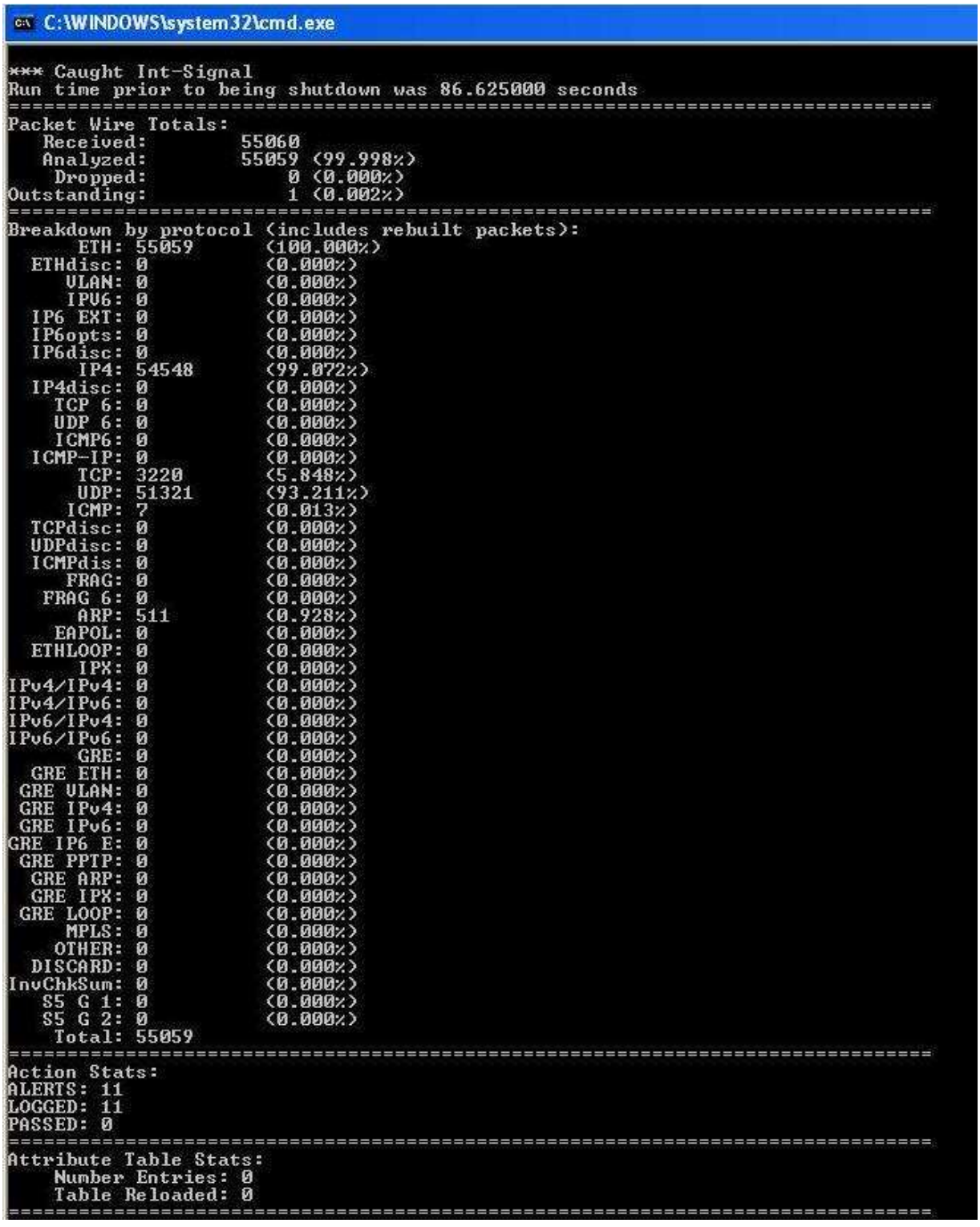


**Figure 3-3. Snort IDS used in Intrusion Detection**

## 3.5 RF Protect

RF protect is a proprietary wireless intrusion detection and prevention (WIDP) system developed by Aruba Networks. In our research we use RF protect to conduct intrusion detection and collect alert data related to attacks. The AP-70 sensors send their alerts to RF Protect's central database where it analyzes the data.

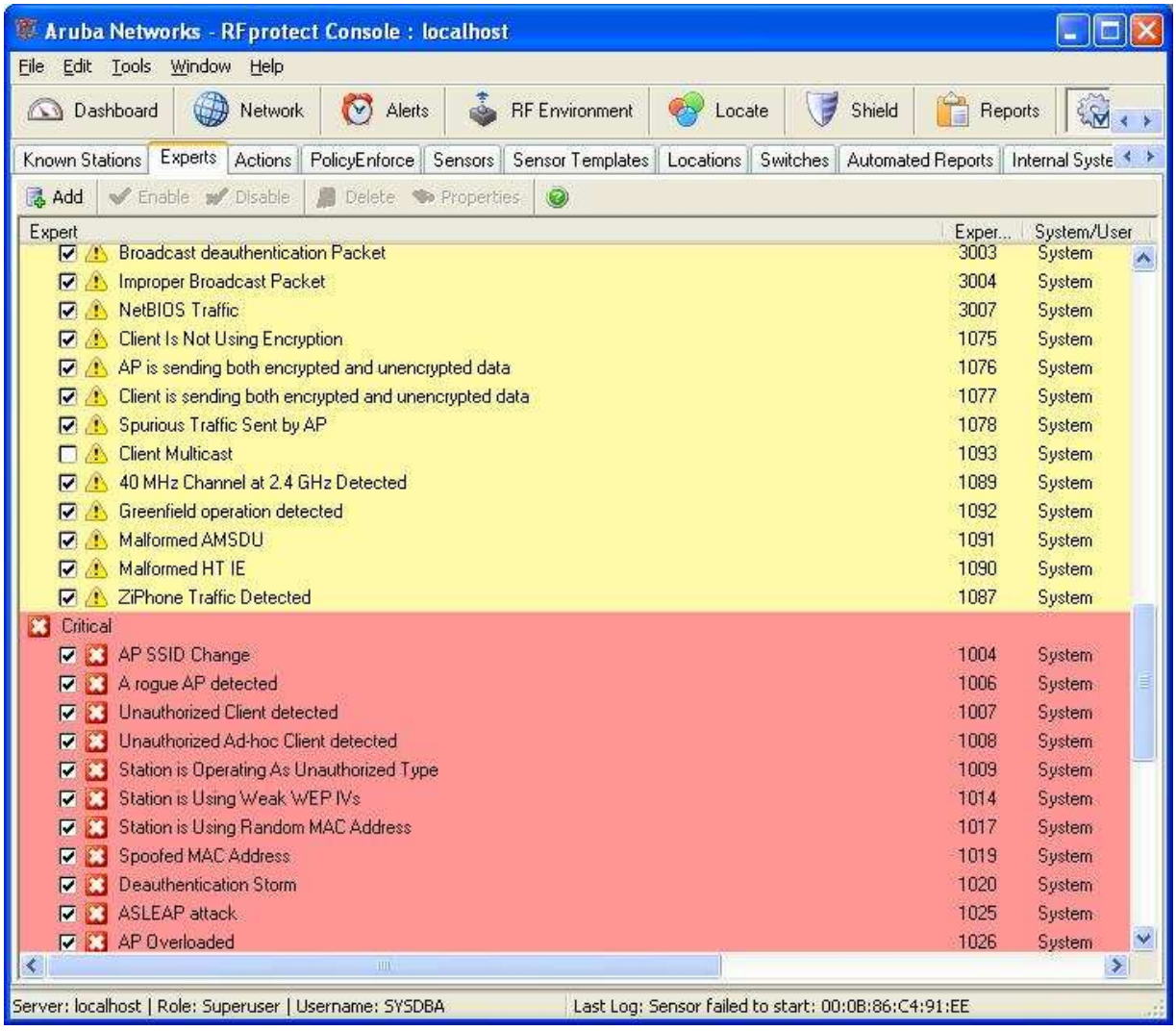


**Figure 3-4. RF Protect Console**

## 3.6 Wireshark

Wireshark is a network protocol analyzer. According to the Wireshark website [Wireshark], it has the following useful features

1. Live capture and offline analysis
2. "Live data can be read from Ethernet, IEEE 802.11, PPP/HDLC, ATM, Bluetooth, USB, Token Ring, Frame Relay, FDDI, and others"

3. “Decryption support for many protocols, including IPSec, ISAKMP, Kerberos, SNMPv3, SSL/TLS, WEP, and WPA/WPA2 “
4. “Coloring rules can be applied to the packet list for quick, intuitive analysis “

We used two Wireshark sensors to gather evidence on wireless networks in our experiments.

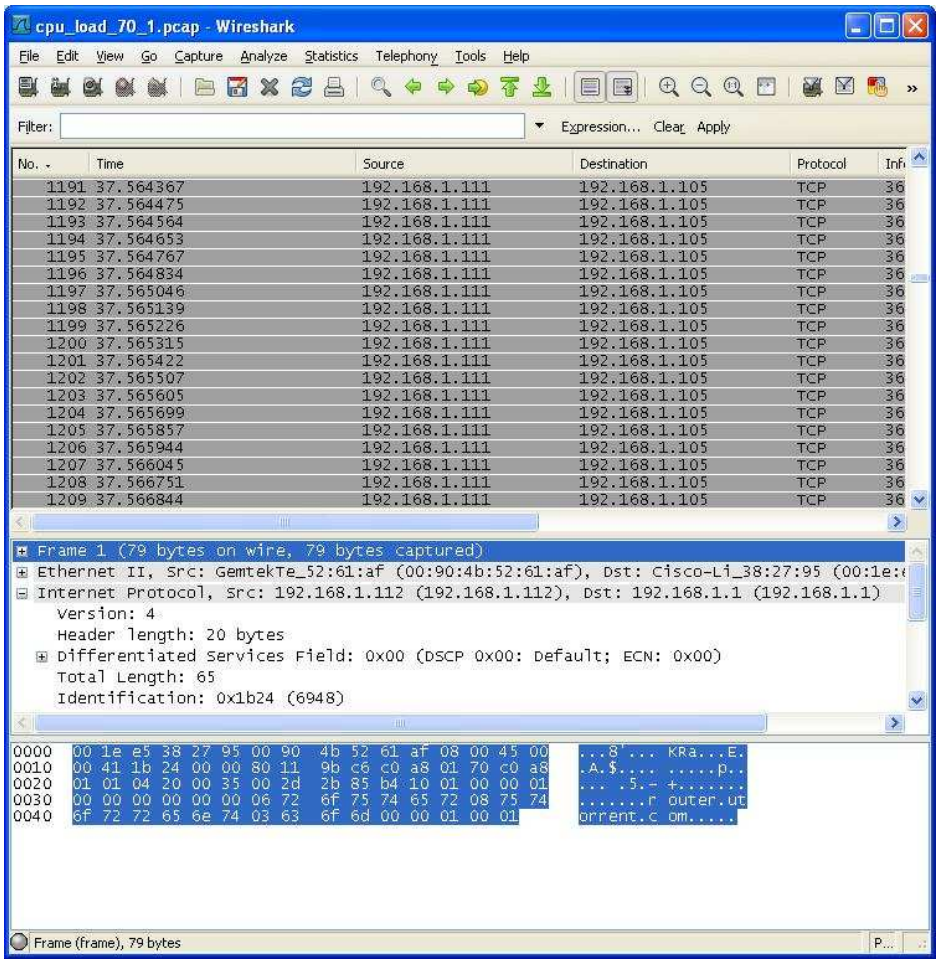


**Figure 3-5. Wireshark at work**

## 3.7 Loadcontrol

According to http://www.codeproject.com/KB/cpp/CPU_Load_Control.aspx?msg=1915014 this is a program that demonstrates how to retrieve the current CPU load percentage and set it using a high priority thread control loop.

The program provides an interface to set the current CPU load as a percentage. Once it is set the CPU load increases to the correct load. According to the author of the program, “It only fully works on single core systems.” We found this to be true and found a way to make it run on both cores. The way it is done is, first one needs to launch two instances of the program. Then

one instance should be run on the first core and the second instance on the second core of the system. Then both threads should be given real time priority. A normal process on windows would have priority set to normal by default. The screen shots below will make the process clearer.

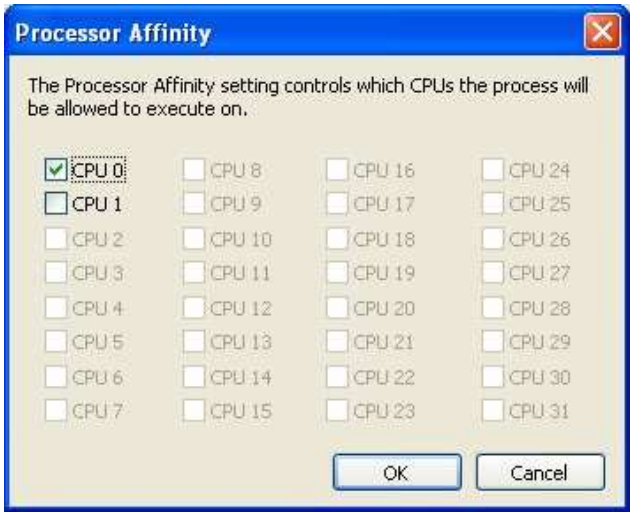


**Figure 3-6. Setting Processor Affinity 1**

First thread should be run on the core CPU 0

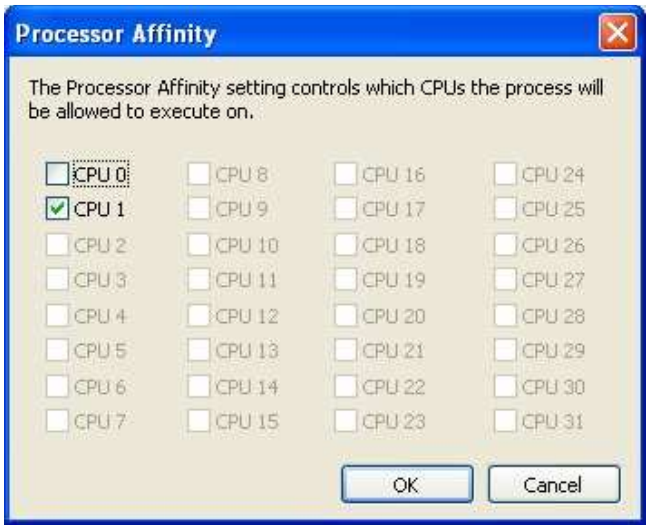


**Figure 3-7. Setting Processor Affinity 2**

Second thread should be run on the core CPU 1

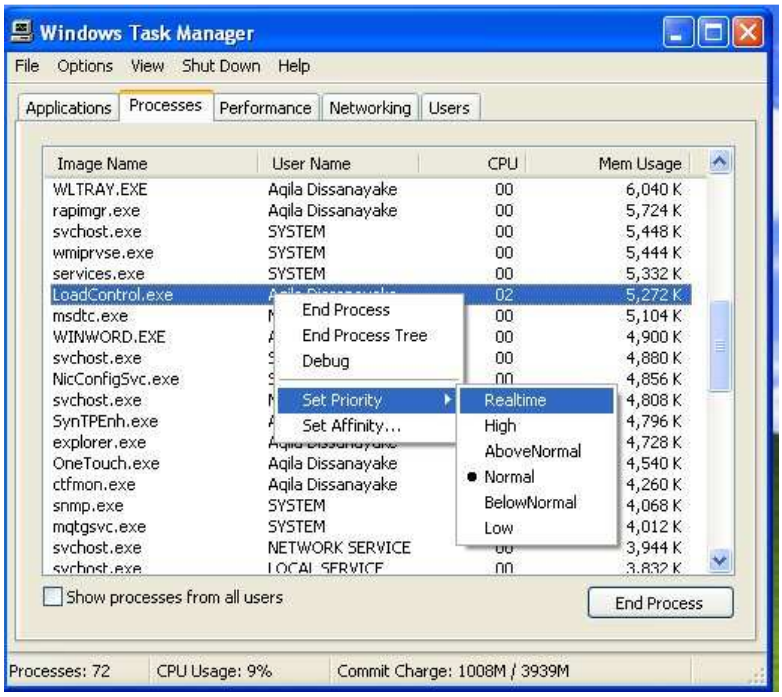


**Figure 3-8. Setting Processor Priority to Realtime**

Both LoadControl processes should be given Realtime priority through the task manager.


**Figure 3-9. Load Control Software at Work**

When we used CPU Load Control, we found that even when we use this tool to apply the same load to both the cores, the load on the two cores would differ greatly and the average load would not be equal to the specified value. Moreover the CPU load goes on varying. When we start Wireshark on a machine and another machine is used for sending the packets, which Wireshark would capture and display, the CPU load on the two cores would come closer to the specified value. Since we could not find any other tool, we used this tool for loading the three machines, which were running Wireshark and Snort. This, as will be explained later in this thesis, helped create for our experimentation, tools, with variable reliability. We took the specified value of the load as the CPU load for our experimental reading.

# 4 SURVEY OF DEMPSTER-SHAFER THEORY IN DATA FUSION

## 4.1 Theory of Evidence and Dempster-Shafer Theory in Data Fusion

According to Siaterlis and Maglaris [2004] “data fusion is a process performed on multisource data towards detection, association, correlation, estimation and combination of several data streams into one with a higher level of abstraction and greater meaningfulness.” According to the authors, the process of collecting information from multiple and possibly heterogeneous sources and combining them leads to more descriptive, intuitive and meaningful results. According to Bass [2000], multi sensor data fusion is a relatively new discipline that is used to combine data from multiple and diverse sensors and sources in order to make inferences about events, activities and situations. Bass [2000] states that this process can be compared to the human cognitive process where the brain fuses sensory information from various sensory organs to evaluate situations, make decisions and to direct specific actions. Bass [2000] and Siaterlis and Maglaris [2004 and 2005] give several examples of systems that use data fusion in the real world. Bass [2000] claims data fusion is widely used in military applications such as battlefield surveillance and tactical situation assessment and in commercial applications such as robotics, manufacturing, remote sensing, and medical diagnosis. Siaterlis and Maglaris [2004 and 2005] provide military systems for threat assessment and weather forecast systems as examples of such systems currently in use today.

The Theory of Evidence is a branch of mathematics that is concerned with using evidence to calculate the probability of an event. The Dempster-Shafer theory (D-S theory) is a theory of evidence used to combine separate pieces of evidence to calculate the probability of an event. According to Chen and Aickelin [2006], the Dempster-Shafer theory was introduced in the 1960’s by Arthur Dempster and developed in the 1970’s by Glenn Shafer. They view the theory as a mechanism for reasoning under epistemic uncertainty. According to Sentz [2002], epistemic uncertainty is “the type of uncertainty which results from the lack of knowledge about a system and is a property of the analysts performing the analysis.” Sentz [2002] also states that epistemic uncertainty is also known as, Subjective uncertainty, Type B uncertainty, Reducible uncertainty, State of Knowledge uncertainty, and Ignorance. Chen and Aickelin [2006] also stated that the part of the D-S theory which is of direct relevance to anomaly detection is the Dempster’s rule of combination. According to Siaterlis et al. [2003] D-S theory can be considered as an extension of Bayesian inference. According to Shafer [2002] “the Dempster-Shafer theory is based on two ideas: the idea of obtaining degrees of belief for one question from subjective probabilities for a related question, and Dempster's rule for combining such degrees of belief when they are based on independent items of evidence.”

According to Chen and Aickelin [2006], the Dempster-Shafer theory is a combination of a theory of evidence and probable reasoning, to deduce a belief that an event has occurred. They stated that the D-S theory updates and combines individual beliefs to give a belief of an event

occurring in the system as a whole. According to Chen and Venkataramanan [2005], in previous approaches data has been combined using simplistic combination techniques such as averaging or voting. They further stated that a distributed intrusion detection system combines data from multiple nodes to estimate the likelihood of an attack, yet fails to take into consideration that the observing nodes might be compromised. Dempster-Shafer theory takes this uncertainty into account when making the calculations.

## 4.2 Data Used in Experiments

One of the important parts of an experiment is to determine what kind of data should be used in the experiment. Should the data be generated to create an original dataset or should the research make use of the already generated data? When it came to Dempster-Shafer data fusion, the same questions arose. In the research conducted so far, most researchers decided to use an existing dataset while some preferred to generate their own data.

The *Defense Advanced Research Projects Agency* (DARPA) DDoS intrusion detection evaluation datasets are a popular choice among many intrusion detection system (IDS) testers. It is no different when it came to testing the Dempster-Shafer IDS models. Yu and Frincke [2005] used the DARPA 2000 DDoS intrusion detection evaluation dataset to test their model. Chou et al. [2007 and 2008] used the DARPA KDD99 intrusion detection evaluation dataset. The KDD99 dataset can be found at http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. The 1998 DARPA intrusion detection evaluation data set was used by Katar [2006] for his experiments.

According to Chou et al. [2007], the DARPA KDD99 data set is made up of a large number of network traffic connections and each connection is represented with 41 features. Further, each connection had a label of either normal or the attack type. They stated that the data set contained 39 attack types which fall into four main categories. They are, Denial of Service (DoS), Probe, User to Root (U2R), and Remote to Local (R2L). The authors have reduced the size of the original data set by removing duplicated connections. They further modified the data set by replacing features represented by symbolic values and class labels by numeric values. Also, they normalized values of each feature to between 0 and 1 in order to offer equal importance among features.

Chen and Aickelin [2006] used the Wisconsin Breast cancer dataset and the Iris data set [Asuncion and Newman 2007] of the University of California, Irvine (UCI) machine learning repository for their research. Some authors chose to generate their own data for the attacks and background traffic. For example, Siaterlis et al. [2003] used background traffic generated from more than 4000 computers in the National Technical University of Athens (NTUA) for their experiment.

## 4.3 Frame of Discernment

When using Dempster-Shafer's theory of evidence, defining the frame of discernment is of great importance. Most of the authors referred in this survey did not explicitly mention their frame of discernment. Some of them did not mention a frame of discernment at all. It can be argued that this is a major weakness of those particular papers.

Wang et al. [2004] defined their frame of discernment to be Stealthy Probe [Paulauskas and Garsva 2006], DDoS [Rogers 2004], Worm [http://en.wikipedia.org/wiki/Computer_worm], LUR (Local to User, User to Root) [Paulauskas and Garsva 2006], and Unknown. According to the authors, 'Unknown' is defined into the frame of discernment because abrupt increases of network traffic could be a result of a DDoS or a worm spreading or LUR or a Probe attack. The authors argue that in this situation, the host agent information will help to make the final decision as to what attack it was. Siaterlis et al. [2003] and Siaterlis and Maglaris [2004 and 2005] defined their frame of discernment to be

1. Normal
2. SYN-flood [http://en.wikipedia.org/wiki/SYN_flood]
3. UDP-flood [http://en.wikipedia.org/wiki/UDP_flood_attack]
4. ICMP-flood [http://en.wikipedia.org/wiki/Ping_flood]

According to the authors, these states are based on a flooding attack categorization of the DDoS tools [Mirkovic et al. 2001] that were in use at the time they wrote their paper. Hu et al. [2006] defined their frame of discernment to be normal, TCP, UDP, and ICMP. Hu et al. [2006] were concerned with flooding attacks in their research. Chatzigiannakis et al. [2007] defined four states for the network. They are Normal, SYN-attack, ICMP-flood, and UDP-flood. These states are quite similar to what Siaterlis and Maglaris [2004 and 2005] defined for their frame of discernment. Siaterlis and Maglaris [2004] and Chatzigiannakis et al. [2007] conducted their research at the National Technical University of Athens (NTUA).

## 4.4 Application of D-S in Anomaly Detection

Anomaly detection systems work by trying to identify anomalies in an environment. In other words an anomaly detection system looks for what is not normal in order to detect whether an attack has occurred. According to Chen and Aickelin [2006] the problem with this approach is that user behavior changes over time and previously unseen behavior occurs for legitimate reasons which leads to generation of false positives in the system. The authors say that this can lead to a sufficiently large number of false positives forcing the administrator to ignore the alerts or disable the system.

According to Katar [2006], a majority of intrusion detection systems, based on detection of anomalies, adopt a single algorithm either for modeling normal behavior patterns and/or attack signatures which ensures a lower detection rate and increases false negative rate.

### 4.4.1 Experiments of Yu and Frincke

Yu and Frincke [2005] state that modern intrusion detection systems often use alerts from different sources to determine how to respond to an attack. According to the authors, alerts from different sources should not be treated equally. They argue that information provided by remote sensors and analyzers should be considered less trustworthy than that provided by local sensors and analyzers. They also state that identical sensors and analyzers installed at different locations may have different detection capabilities because the raw events captured by these sensors are different. Further, different kinds of sensors and analyzers which detect the same type of attack may do so with a different level of accuracy.

In their research the authors addressed the fact that all observers cannot be trusted equally and a given observer may have different effectiveness in identifying individual misuse types by extending the D-S theory to incorporate a weighted view of evidence. In other words, the authors proposed to improve and assess alert accuracy by incorporating a weight component to each observer to reflect how much trust they place on each observer. For this purpose they proposed a modified D-S combination rule. The new D-S combination rule has exponents as weights for each observer. This new theory is called the Extended Dempster-Shafer Theory. The new combination rule is given below.

$$m_{12}(A) = \frac{B \cap C = A,\ \Sigma\ [m_1(B)]^{w_1}[m_2(C)]^{w_2}}{1 - [B \cap C = \varnothing,\ \Sigma\ [m_1(B)]^{w_1}[m_2(C)]^{w_2}]}$$

- $m_{12}$ (A) = Combined belief of the hypothesis A
- $m_1$ (B) = Belief committed to B as seen by the first observer
- $m_2$ (C) = Belief committed to C as seen by the second observer
- Where $w_i$ is the weight for the $i^{th}$ observer. When $w_1 = w_2 = 1$, is reduced to the basic D-S combining rule.

According to the authors, in their system they estimated the weights based on the Maximum Entropy principle [Berger et al. 1996; Rosenfeld 1996) and the Minimum Mean Square Error (MMSE) criteria.

Yu and Frincke [2005] performed experiments using two DARPA 2000 DDoS intrusion detection evaluation data sets. According to the authors, both datasets include network data from both the demilitarized zone (DMZ) and the inside part of the evaluation network. They stated that they used RealSecure Network Sensor 6.0 with maximum coverage policy in their experiments.

The authors stated that the extended D-S further increases the detection rate while keeping false positive rate low. They also pointed out that when using the basic D-S combination algorithm, the detection rate decreases relatively to the extended D-S. According to them, the extended D-S algorithm provides 30% more accuracy. Also, they have compared their method with Hidden Colored Petri Net (HCPN) based alert analysis component. They stated "Our initial evaluations on the DARPA IDS evaluation data set show that our alert fusion algorithm can improve alert quality over those from Hidden Colored Petri-Net (HCPN) based alert correlation components installed at the demilitarized zone (DMZ) and inside network sites. Due to alert confidence fusion in our example, the detection rate rises from75% to 93.8%, without adversely affecting the false positive rate" Yu and Frincke [2005].

The authors claim that their "alert confidence fusion model can potentially resolve contradictory information reported by different analyzers, and further improve the detection rate and reduce the false positive rate." They state that their approach has the ability to quantify relative confidence in different alerts.

### 4.4.2 Experiments of Chen and Aickelin

Chen and Aickelin [2006] have constructed a Dempster-Shafer based anomaly detection system using the Java 2 platform. First they use the Wisconsin Breast Cancer Dataset (WBCD) to perform their experiment. According to the authors, the WBCD is used for two reasons. One reason is that they can compare the performance of other algorithms to their approach. The other is to "investigate if it is possible to achieve good results by combining multiple features using D-S, without excessive manual intervention or domain knowledge based parameter tuning." Secondly, Chen and Aickelin [2006] have used the Iris plant dataset [Asuncion and Newman 2007] for their experiments. According to the authors the Iris dataset was chosen because it contains fewer features and more classes than the WBCD. By using this they can confirm whether D-S can work on problems with fewer features and more classes. Thirdly, they conducted an experiment using an e-mail dataset which was created using a week's worth of e-mails (90 e-mails) from a user's sent box with outgoing e-mails (42 e-mails) sent by a computer infected with the netsky-d worm. The aim of the experiment was to detect the 42 infected e-mails. They used D-S to combine features of the e-mails to detect the worm infected e-mails.

Their anomaly detection system utilizes a training process to derive thresholds from the training data, and classifies an event as normal or abnormal. According to Chen and Aickelin [2006], the basic probability assignment (BPA) functions are made based on these thresholds to assign mass values. In their experiment, first they process data from various sources and send them to corresponding BPA functions. Then, mass values for each hypothesis are generated by these functions which are then sent to the D-S combination component. The D-S combination component combines all mass values using Dempster's rule of combination and generates the overall mass values for each hypothesis.

The authors claimed that their experimental results show that they were able to successfully classify a standard dataset by combining multiple features for WBCD using the D-S method. According to them, the experimental results with the Iris dataset [Asuncion and Newman 2007] show that D-S can be used for problems with more than two classes, with fewer features. They also claimed experiments with the e-mail dataset show that D-S method works successfully for anomaly detection by combining beliefs from multiple sources.

The authors claimed that combining features using D-S improves accuracy. Also, they claimed that a few badly chosen features do not negatively influence the results, as long as most chosen features are suitable. Therefore they stated that D-S is ideal for solving real-world intrusion detection problems. Also, they claimed that the results of the Iris dataset prove that D-S can be used for problems with more than two classes, with fewer features. By successfully detecting e-mail worms through experiments, they claimed that the D-S method works successfully for anomaly detection by combining multiple sources.

The authors concluded that based on their results, D-S can be a good method for network security problems with multiple features (various data sources) and two or more classes. They also stated that the initial feature selection influences overall performance as with any other classification algorithm. Further, D-S approach works in cases where some feature values are missing which they say is very likely to happen in real world network security scenarios. Chen and Aickelin stated “our continuing aim is to find out how D-S based algorithms can be used more effectively for the purpose of anomaly detection within the domain of network security.”

### 4.4.3 Experiments of Chatzigiannakis et al

Chatzigiannakis et al [2007] conducted their experiments at the National Technical University of Athens (NTUA). They addressed the problem of discovering anomalies in a large-scale network based on the data fusion of heterogeneous monitors. The authors built their work partially on the data fusion algorithms presented by Hall [1992].

They monitored the link between National Technical University of Athens (NTUA) and the Greek Research and Technology Network (GRNET) which connects the university with the Internet. The authors claim that this link has an average traffic of 700-800 Mbits/sec and that it contains a rich network traffic mix that consists of standard web traffic, mail, FTP and p2p traffic.

According to the authors, two anomaly detection techniques, namely Dempster-Shafer and Multi-Metric-Multi-Link (M3L), were evaluated and compared under various attack scenarios. The authors performed a SYN-attack from GRNET using the TFN2K DoS tool on the target which was in the NTUA network. The attack was done by sending IP spoofed TCP SYN packets. According to the authors ICMP-flood and UDP-flood attacks were injected manually into the network traces of the collected data.

The D-S algorithm correctly detected an ICMP flood when attack packets correspond to 5% of the background traffic. For a SYN attack, when attack packets correspond to 2% of background traffic, the D-S algorithm erroneously concluded that the network is normal. However, their research showed that when attack packets correspond to 20% of background traffic, the D-S algorithms correctly detects the SYN attack state. When attack packets correspond to 20% of total traffic in an ICMP flood attack, the M3L algorithm fails to detect the attack. According to the authors M3L fails to detect the attack because the selection of metrics is inappropriate (metrics utilized are uncorrelated) so the algorithm fails to create a precise model of the network. For a SYN attack which consists of packets corresponding to 2% of background traffic, the M3L algorithm correctly detects the attack.

According to the authors, the differences in the performance of the algorithms lie in the correlation of the metrics used. They stated that the D-S theory of evidence performs well on the detection of attacks that can be sensed by uncorrelated metrics. The explanation they give for this is that it is because the D-S theory requires the evidence originating from different sensors to be independent. According to the authors, M3L requires the metrics fed into the fusion algorithm present some degree of correlation. “The method models traffic patterns and interrelations by extracting the eigenvectors from the correlation matrix of a sample data set. If there is no correlation among the utilized metrics then the model is not efficient.” The authors stated that “Metrics such as TCP SYN packets, TCP FIN packets, TCP in flows and TCP out flows are highly correlated and should be utilized in M3L, whereas the combination of UDP in/out packets, ICMP in/out packets, TCP in/out packets are uncorrelated and should be used in D-S.” According to the authors, “attacks that involve alteration in the percentage of UDP packets in traffic composition such as UDP flooding are better detected by the D-S method.” Further, “attacks such as SYN attacks, worms spreading, port scanning which affect the proportion of correlated metrics such as TCP in/out, SYN/FIN packets and TCP in/out flows are better detected with M3L.” Also, the authors derive an important result from their study and numerical results. That is, the conditions under which the two algorithms operate efficiently are complementary, and therefore could be used effectively in an integrated way to detect a wide range of possible attacks.

Chatzigiannakis et al. [2007] studied the problem of discovering anomalies in a large-scale network based on the data fusion of heterogeneous monitors. They studied two different anomaly detection techniques, one based on the D-S theory of evidence and the other based on Principal Component Analysis. They evaluated the two algorithms via emulation and simulation, and the numerical results showed that the conditions under which they operate efficiently are complementary. So, they came to the conclusion that they should be used effectively in an integrated way to detect a wide range of attacks. Also, they claimed timely and proactive detection of network anomalies is a prerequisite for the operational and functional effectiveness of secure networks because of the explosive growth of the global Internet and electronic

commerce infrastructures. They further claimed without well designed tools for the management of future networks, it will be hard to dynamically and reliably identify network anomalies.

The major contributions of the papers discussed in this section are summarized below.

| **Year** | **Paper** | **Major Contribution** |
|---|---|---|
| 2005 | Alert confidence fusion in intrusion detection systems with extended Dempster-Shafer theory.<br>[Yu and Frincke] | Extended the D-S theory to incorporate weights to different observers to reflect that every observer cannot be trusted equally. By doing so they gave birth to the Extended D-S theory.<br>Showed how to improve and assess alert accuracy by incorporating an algorithm based on the exponentially weighted Dempster-Shafer theory of Evidence. This was the first time the extended D-S was used in intrusion detection.<br>Showed through experiments that extended D-S is 30% more accurate when it comes to detection accuracy than the basic D-S. |
| 2006 | Dempster-Shafer for Anomaly Detection.<br>[Chen and Aickelin] | Showed by experiments that one is able to successfully classify a standard dataset by combining multiple features for the WBCD (Wisconsin Breast Cancer Dataset) using the D-S method.<br>Showed through experiments with the Iris dataset that D-S can be used for problems with more than two classes, with fewer features.<br>Showed through experiments with the e-mail dataset that D-S method works successfully for anomaly detection by combining beliefs from multiple sources. |
| 2007 | Data fusion algorithms for network anomaly detection: classification and evaluation.<br>[Chatzigiannakis et al] | Compared two anomaly detection techniques, Dempster-Shafer and Multi-Metric-Link (M3L) under various attack scenarios.<br>Showed that M3L fails to detect attacks whose metrics utilized are uncorrelated which cause the algorithm not to |

| | | |
|---|---|---|
| | | create a precise model of the network.<br><br>Showed that D-S theory of evidence performs well on the detection of attacks that can be sensed by uncorrelated metrics.<br><br>Showed that the conditions under which the two algorithms operate efficiently are complementary, which makes it better to use them in an integrated environment. |

## 4.5 Application of D-S to Detect DoS and DDoS Attacks

### 4.5.1 Experiments of Siaterlis et al. of the NTUA

Various experiments have been conducted to study the use of D-S theory to detect DoS and DDoS attacks. Some of the major research in this area has taken place at the National Technical University of Athens (NTUA). Siaterlis et al [2003], Siaterlis and Maglaris [2004] and Chatzigiannakis et al [2007] have conducted their experiments related to DoS attacks and D-S theory at the NTUA. Vasilis Maglaris and Basil Maglaris of the NTUA have both published two papers on multi sensor data fusion for Denial of Service (DoS) detection using the D-S theory of evidence. Christos Siaterlis of the NTUA is the only researcher so far to publish 3 papers on intrusion detection using the D-S theory.

Siaterlis et al. [2003], addressed the problem of detecting distributed denial of service attacks (DDoS) “on high bandwidth links that can sustain the flooded packets without severe congestion.” According to the authors, DDoS attacks have been the focus of the research community in the last few years but still remain an open problem. They stated that many DDoS prevention techniques like Ingress and RPF filtering have been proposed in the literature and implemented by router vendors but they were not able to lessen the problem. The authors say that when they refer to DDoS, they refer to packet flooding attacks not logical DoS attacks that exploit application vulnerabilities. Also, they do not require the attackers to be truly distributed in the network topology in their DoS attacks. Their research consists of developing a framework for DDoS detection engine using the Dempster-Shafer’s “Theory of Evidence”.

According to Siaterlis and Maglaris [2004] and Siaterlis and Maglaris [2005] “The Internet” can be compared to an essential utility such as electricity or telephone access. They say that even a short downtime of the Internet could cause grave financial damages. According to them DDoS is one of the main reasons for Internet cutoffs. Siaterlis and Maglaris provide several examples to prove their reasoning including a DDoS attack against one of the largest anti-spam black-list companies, and another DDoS against the “Al-Jazeera” news network and another against the root name servers. According to them, in a DoS attack the bandwidth, near the victim,

has already been consumed. Therefore, techniques such as firewall filtering, rate limiting, route blackholes, are not effective countermeasures for such an attack. They argue that IP traceback, IP pushback, are ineffective (to move the countermeasure near the source of the attack) because automated large scale cooperation is difficult in a diverse networked world like the Internet. Other techniques such as Ingress filtering, RPF filtering, are only helpful to discourage the attacker because they make the traceback easier. They argue that the only reliable solution to DoS mitigation is to have a solid DoS detection mechanism. According to the authors, the custom detection methods that are being used by network engineers are weak as they utilize thresholds on single metrics. Therefore, they utilize a data fusion algorithm based on the "Theory of Evidence" to combine output of several sensors to detect attempted DoS attacks.

#### 4.5.1.1 Data Fusion Architecture

According to Siaterlis et al. [2003] a data fusion architecture consists of the following main stages.

1. Data Collection – Data is detected and collected through various sensors

2. Data Alignment & Association – Since data is collected through various sensors, they may exhibit differences in time, space or measurement. These will be aligned properly at this stage.

3. State Estimation – A data fusion algorithm estimates the state based on the knowledge gathered through sensors.

4. Attribute classification & Identification - In this phase the different targets and events that are being monitored are identified.

5. Situation Assessment – Based on the results of the previous 2 stages, the overall status of the system is determined. This is the highest level of information fusion.

#### 4.5.1.2 Data Fusion Models

According to Siaterlis et al. there are 3 kinds of data fusion models.

1. Physical Models

2. Parametric Classification
    a. Bayesian Inference Model
    b. Dempster Shaffer Theory of Evidence
    c. Adaptive Neural Nets
    d. Voting Methods

3. Cognitive Algorithms
    a. Expert Systems

b. Fuzzy set theory

An example of a physical model is the Kalman filter which provides a solution to minimize the mean square error between the true state of the system and the estimate of the state. Siaterlis et al. states that the Kalman filter requires the knowledge of the state transition matrix and the measurements are corrupted by white zero mean noise with known covariance matrix. They also state network behavior has not yet been successfully modeled; therefore such a system's usability is questionable. Siaterlis et al. state that adaptive neural nets have been used in the context of intrusion detection but it requires training data that will be representative of the normal traffic data, which is extremely hard to gather or generate. Voting is one of the simplest and intuitive methods for fusion models. According to Siaterlis et al. each sensor's data serves as a vote in a democracy where the fused declaration is the declaration of the majority. They state that this method is useful when a priori statistics are not known. According to Siaterlis et al. there are also variation of the voting system that include, weighted voting systems and use of intermediate decision on a decision tree. The authors stated that the underlying theory in expert systems is "First Order Logic". The drawback with first order logic is that it cannot model the whole spectrum between belief and disbelief in a statement but uses a plain true or false approach instead. They state that fuzzy logic has many common elements with Theory of evidence.

After reviewing all the above mentioned methods they have concluded that Dempster-Shafer's theory of evidence needs further investigation. They stated that "there is a clear need to utilize information from multiple heterogeneous sources with different sensitivity, reliability and false alarm rates." The authors considered the Dempster-Shafer approach as an extension of the Bayesian inference.

#### 4.5.1.3 DDoS Detection Engine and the Early Research

Siaterlis et al. [2003] built a prototype for a DDoS detection engine that uses the Dempster-Shafer theory of Evidence for their experiment. According to them this "might aid network administrators to monitor their network more efficiently and with small set up cost." They evaluated the D-S detection engine prototype in the National Technical University of Athens (NTUA). According to the authors, related experiments were carried out over several days during regular business hours with background traffic generated from more than 4000 computers in the campus. The authors hosted the victim inside the campus network while the attacker was outside the campus network. The attacker was connected to a fast Ethernet interface to simulate the aggregation of traffic from several attacking hosts. The authors claimed that their DDoS detection engine can maintain a low false positive alarm rate with a reasonable effort from the network administrator.

The authors stated that their architecture is made up of several distributed and collaborating sensors which share their beliefs about the network's true state. By the true state of

the network, they mean whether the network is under attack or not. The authors view the "network as a system with stochastic behavior without assuming any underlying functional model." The attempt to determine the unknown system state is based on knowledge reported by sensors that may have acquired their evidence based on totally different criteria. They have implemented a system that fuses the knowledge collected from the reports of various sensors, in order to infer the state of the monitored network. Their architecture consists of the following two sensor types.

(1) A preprocessor plug-in for Snort – They collected data from incoming and outgoing TCP, TCP SYN, TCP FIN, UDP, ICMP packet rates and their corresponding share of the link utilization.
(2) A SNMP data collector and analyzer stored the data in round robin databases using the RRDtool. They calculated the bytes/sec, packets/sec ratios and active flow numbers based on Cisco's Netflow [22].

#### 4.5.1.4 Sensor Functionality

The authors stated that the sensors can express beliefs about the network state after the right configuration and fine tuning and that their detection principle differs from many of the existing detection techniques which are focused on a single metric.

They have introduced a sensor that monitor the number of active flows seen by a router. A flow is defined as a unique set of 5 characteristics that include (protocol, source IP, source port, destination IP, and destination port). According to the authors, in the presence of a spoofed attack the number of active flows rises. Further, in the presence of a flooding attack the number of transports that are not completed (with TCP FIN or RST) is high. These flows fill up the cache without being removed gracefully. The reason for including this metric, the authors state is that it will give a good indication of a spoofed attack even though it cannot give an insight into the exact attack type. The sensor in this case states its belief in the hypothesis, H = {SYN-flood, UDP-flood, ICMP-flood}.

The authors have built basic probability assignments (BPA's) that match measured values to beliefs about the true system state. They have defined their frame of discernment to be Θ = {Normal, TCP SYN Attack, UDP BWDTH Attack, and ICMP BDWDTH Attack}. The authors have defined a way to assign BPA's in the following manner. For example, if the sensor measures a high value for the ratio

$$\frac{\text{Incoming UDP bytes/sec}}{\text{Outgoing UDP bytes/sec}}$$

Then the sensor states its increased belief in a UDP attack state. The sensors then take the following steps to calculate the BPA's

1. Assigns a value m(H) ε [0,1]
2. Assigns a value to the set ¬H, to express the refuting evidence of the hypothesis H, so m(¬H) ε [0,1].
3. Assigns a value to the set Θ to express ignorance of the sensor and the possibility that it might be erroneous. m(Θ) ε [0, 1].

According to the authors, m(H) + m(¬H) + m(Θ) = 1. The sensors calculate the corresponding BPA's and transfer the data to the fusion node which has the DS inference engine. The DS inference engine calculates the belief intervals for each member of the frame of discernment. These belief intervals are then graphically represented and the interpretations of the results are left to the network admin.

#### 4.5.1.5 Later Research

In 2004, Siaterlis and Maglaris published another paper based on intrusion detection using D-S theory of evidence. The work done is continued from where they left off in Siaterlis and Maglaris [2003]. In this work, they have changed the frame of discernment from Θ = {Normal, TCP SYN Attack, UDP BWDTH Attack, and ICMP BDWDTH Attack} to Θ = {Normal, SYN flood, UDP flood, and ICMP flood}.

In 2005, Christos Siaterlis published another paper with Vasilis Maglaris that extended the work from Siaterlis and Maglaris [2004]. According to them, the 2005 paper discusses how to automate the process of tuning the sensors while taking advantage of expert knowledge. Also, they discussed the combination of different metrics to enhance detection performance compared to the use of a single metric. Further they compared the D-S approach with the use of an Artificial Neural Network (ANN) when it comes to data fusion.

Unlike in the previous two papers, Siaterlis and Maglaris [2005] go into much more detail as to how their system operates. They stated their customized Netflow collector gathers flows that are exported by the router and calculates the number of flows with lifetime shorter than 10ms according to the flow generation rate. According to the authors, though this metric does not give an indication of exact attack type, it is a good indication of spoofed or a highly distributed attack.

The authors stated that in the early stages of their work, the sensors were required to be manually configured to express beliefs about the network state by translating the measurements to basic probability assignments (BPA). Later on they have used a supervised learning approach and inserted a neural network at the sensor level to ease the administrator from having to configure the sensor manually. They have further given a formula they used in the neural node to calculate the BPA's automatically. It is given below.

- $m(H) = x$ , if $m(\Theta) + x < 1$

- m(H) = 1 – m(Θ), otherwise

  and

- m( ¬ H) = 1-x-m(Θ), if m(Θ) + x < 1
- m( ¬ H) = 0, otherwise

  Where x is the sensor output.

These BPA's are then transferred to the D-S engine. The D-S engine then fuses the information using the Dempster's rule of combination to calculate the belief intervals for each member of the frame of discernment. Then, the attacks are detected by the output of the belief of individual attack states. As an example they state,

- UDP Alert = true, if Bel({UDP-flood}) > 0.5; and
- UDP Alert = false, otherwise

The authors have compared their data fusion approach to Artificial Neural Network (ANN) data fusion approach. They stated "If we feed the detection metrics directly into an ANN, like the feed-forward Multi Layer Perceptron (MLP) network, we can teach it to classify the network state in elements of the same set {NORMAL, SYN-flood, UDP-flood, ICMP-flood}." They have used the Levenberg-Marquardt back propagation algorithm for training because of its speed. They have performed many tests which included changing the number of neurons in the hidden layer. Their results have indicated that compared to ANN, D-S produces fewer false positives. Also, they stated that apart from the above comparison, in the D-S system they can incorporate human expertise which is an added advantage. What they meant by this was that they can define which attack states each sensor is sensitive to using their expertise.

Siaterlis and Maglaris [2005] stated that implementing their ideas into an operational network could be a task of significant difficulty, but it may offer many advantages if done successfully. The advantages include,

1. Sensors can provide both supportive and refuting evidence of an attack. Therefore, different sensors can lower or raise the combined belief of an attack state.
2. Each sensor can contribute information at its own level of detail. This enables the use of metrics such as CPU utilization of routers that are not specific to attack type.
3. No need to assume the probability of the network being in a specific state. Just need to express the belief that an observed event supports a state.
4. Multiple data sources can be used to increase the confidence in the estimation.
5. Can incorporate knowledge from sensors that are based on different detection algorithms.
6. Can activate detection algorithms on demand to refine the beliefs.

They also point out that knowledge based systems can only be as good as their source from which they acquire knowledge. Also, they state that their system cannot handle multiple simultaneous attacks because mutual exclusivity of system states is assumed.

### 4.5.2 Experiments of Hu et al.

According to the Hu et al. [2006], when it comes to implementing network security management, multi-sensor data fusion faces a lot of problems. For example, there is no appropriate physical model to describe a network. They stated that the state transition matrix for a network is hard to acquire and a network's behavior has not yet been successfully modeled. Also, they state that a physical model such as the Kalman Filter is limited in use and using it to predict traffic is a tradeoff between accuracy and efficiency. Cognitive algorithms have good adaptability but need a lot of training data, which they state is hard to capture in a real network. So, in their experiments they have used the D-S theory of evidence to make uncertainty inferences because it does not require state transition matrices or training data.

According to the authors, an improved detection engine has been introduced in this paper. They also introduce "Detection Uncertainty" to describe the fuzzy problem which cannot be avoided in the detection and merges identity inference and intrusion detection. They defined detection uncertainty to be the sum of subject uncertainty and objective uncertainty.

Detection Uncertainty = Subjective Uncertainty + Objective Uncertainty

The uncertainty that arises because of the selected detection metrics and sensor specific techniques was defined as Subjective Uncertainty. The uncertainty that arises because of the experimental environment was defined as Objective Uncertainty. According to the authors once the sensors are completely setup, Subjective Uncertainty will not change. Also, they stated that because of the difference of detection techniques and metrics, different sensors have different Subjective Uncertainties.

According to the authors, the experiments were carried out in a small scale LAN. They have used LibPcap based sensors to poll the network and assign appropriate mass/belief values to the current state of the network. LibPcap is a system-independent interface for user-level packet capture. It can be downloaded from http://sourceforge.net/projects/libpcap/.

The authors stated that they put more emphasize on the accuracy of the simulation than doing it on real time. Therefore, they have conducted an off-line simulation. They used a MySQL database to store the data (evidence) captured through sensors. MySQL is a popular open source database which can be downloaded from http://www.mysql.com/. An ICMP flooding attack was used to attack the victim. The authors utilized two sensors in the simulation to sample and assign probabilities to the current state of the network.

The authors stated that the experimental results showed the combination of evidence has improved the detection accuracy. Also, they stated that “the assignment of basic probability assignments after combination is much more accurate and makes the discernment range smaller.” According to the authors, the independence of experimental environment reduces some interference of background flow, and guarantees the effect of the experiment. They have admitted that this is not the case in reality.

The authors claimed the next generation network management systems and intrusion detection systems will be "Cyberspace Situational Awareness" systems that will support multi-sensor data fusion. They further claimed that the D-S theory can be successfully used to identify and detect cyberspace intrusions and locate the risks through multi sensor data fusion.

The major contributions of the papers discussed in this section are summarized below.

| **Year** | **Paper** | **Major Contribution** |
|---|---|---|
| 2003 | A novel approach for a distributed denial of service detection engine<br><br>[Siaterlis et al.] | Built a prototype for a DDoS detection engine that uses the Dempster-Shafer theory of Evidence for their experiment<br><br>The authors claim that their DDoS detection engine can maintain a low false positive alarm rate with a reasonable effort from the network administrator. |
| 2004 | Towards multisensor data fusion for DoS detection<br><br>[Siaterlis and Maglaris] | Showed through experiments that even if one sensor fails to detect an outgoing attack, combined knowledge gathered from other sensors indicate the increased belief on an attack state clearly. |
| 2005 | One step ahead to multisensor data fusion for DDoS detection<br><br>[Siaterlis and Maglaris] | Discusses how to automate the sensor tuning process by taking advantage of expert knowledge.<br><br>Discussed the combination of different metrics to enhance detection performance compared to the use of a single metric. |

| | | |
|---|---|---|
| | | Compared the D-S approach with the use of an Artificial Neural Network (ANN) when it comes to data fusion.<br><br>Showed by experiments that compared to ANN, D-S produces fewer false positives. |
| 2006 | Intrusion Detection Engine Based on Dempster-Shafer's Theory of Evidence. | Showed by experiments that the assignment of basic probability assignments after combination is much more accurate and makes the discernment range smaller. |

## 4.6 Advantages of Using D-S Theory

The research reviewed in this chapter has shown that the use of the D-S theory has certain advantages.

According to Siaterlis et al. [2003], and Siaterlis and Maglaris [2004 and 2005], the D-S approach has significant advantages over the Bayesian approach. They stated that in contrast to the Bayesian approach where one can only assign probabilities to single elements of the frame of discernment (Θ), the D-S theory can assign probabilities to the states (elements) of the power set of Θ. Another advantage according to the authors is that D-S theory calculates the probability of the evidence supporting a hypothesis rather than calculating the probability of the hypothesis itself unlike the traditional probabilistic approach. Also, they say that D-S theory has a definite advantage in a vague and unknown environment.

According to Chen and Venkataramanan [2005] the D-S theory of evidence provides a mathematical way to combine evidence from multiple observers without the need to know about a priori or conditional probabilities as in the Bayesian approach.

According to Chen and Aickelin [2006], D-S theory is very well suited for anomaly detection because it does not require any priori knowledge. Another advantage of D-S according to Chen and Aickelin is that it can express a value of ignorance, giving information on the uncertainty of a situation. They stated that Bayesian inference requires a priori knowledge and does not allow allocating probability to ignorance. So, the authors stated that, in their opinion, Bayesian approach is not always suitable for anomaly detection because prior knowledge may not always be available. Especially, when the aim of anomaly detection is to discover previously unseen attacks, then a system that relies on existing knowledge cannot be used.

According to Chatzigiannakis et al. [2007] the D-S theory of evidence has a clear advantage in an unknown environment when compared to inference processes like first order logic that assume complete and consistent knowledge. They also stated that the D-S theory has

an advantage when compared with probability theory which requires knowledge in terms of probability distributions.

### 4.7 Disadvantages of Using D-S Theory

The research reviewed in this chapter has also shown that the use of D-S theory has certain disadvantages associated with it. They are mentioned below.

According to Siaterlis et al. [2003], Siaterlis and Maglaris [2004 and 2005], and Chatzigiannakis et al. [2007] the main disadvantage of the D-S theory is the assumption it makes saying that the evidence is statistically independent from each other. Since sources of information are often linked with some sort of dependence in real life situations, this assumption does not always hold true. Also, in Siaterlis et al. [2003] framework, they pointed out that the system's inability to detect multiple simultaneous attacks was because D-S theory assumed a mutually exclusive set of system states.

According to Chen and Aickelin [2006], D-S has two major problems associated with it. One they say is the computational complexity associated with D-S. The other is the conflicting beliefs management. According to them the computational complexity of D-S increases exponentially with the number of elements in the frame of discernment ($\Theta$). If there are n elements in $\Theta$, there will be up to $2^n$-1 focal elements for the mass function. Further the combination of two mass functions needs the computation of up to $2^n$ intersections.

## 5 WEAKNESSES OF THE EXISTING METHODS

Our research is mainly based on the work of Siaterlis and Maglaris [2004 and 2005] and Yu and Frincke [2005]. Therefore, we will explain the weaknesses we saw in their methods in this section.

### 5.1 Weakness' of the Method Used by Siaterlis and Maglaris

Siaterlis and Maglaris used two sensor types in their research.

(1) Snort preprocessor plug-in

(2) Netflow collector

Though they used two sensor types, they treated all sensor data as equally reliable. Even If the sensors were identical their detection accuracy could depend on the environment. If the sensors are different, there is a good chance their detection capabilities could differ even in the same environment. This is more clearly explained in the following section.

### 5.2 Reliability of Evidence in Fusion

The D-S theory of evidence considers all evidence to be of the same importance and reliability. According to Yu and Frincke [2005] in a distributed environment this does not hold true. One of the reasons is that remote sensors and analyzers are considered less trustworthy than local sensors [Yu and Frincke, 2005]. Further, if the sensors used are different it is not wise to assume they all behave the same way. It is obvious that different sensors would have different detecting capabilities for different attacks. Assuming all sensors behave the same way could lead to incorrect conclusions after combination. We demonstrate this through an example. For simplicity, we shall use evidence from just two sensors only.

Example with two sensors:

| | | | Sensor 1 | | |
|---|---|---|---|---|---|
| Sensor2 | Attack | | Mac Spoof | DoS | Xmas Tree |
| | | Mass (M) | 0.99 | .01 | 0 |
| | Mac Spoof (MS) | 0 | 0 | 0 | 0 |
| | DoS (D) | 0.01 | 0.0099 | 0.0001 | 0 |
| | Xmas Tree (XT) | 0.99 | 0.9801 | 0.0099 | 0 |

1. To calculate the combined basic probability assignment (BPA) for a particular cell, multiply the mass values from the relevant column and row.

For example, to calculate the combined BPA for the cell that has the red value

M(MS) * M(MS) = 0.99 * 0 = 0

According to the Dempster 's Combination Rule

$$m_{12}(A) = \frac{B \cap C = A,\ \Sigma\, m_1(B)\, m_2(C)}{1 - [B \cap C = \varnothing,\ \Sigma\, m_1(B)\, m_2(C)]} \qquad m_{12}(A) \neq \varnothing$$

The combination called the joint mass ($m_{12}$) is calculated from the two sets of masses $m_1$ and $m_2$. $m_1(B)$ and $m_2(C)$ are evidence supporting hypothesis B and C respectively as observed by the two observers. In our example Sensor 1 and Sensor 2 are our observers. Both have two sets of masses corresponding to each attack. Sensor 1 has a mass set ($m_1$) and sensor 2 has a mass set ($m_2$). In this equation $[B \cap C = \varnothing,\ \Sigma\, m_1(B)\, m_2(C)]$ part in the denominator is known as K.

$$K = [B \cap C = \varnothing,\ \Sigma\, m_1(B)\, m_2(C)]$$

K represents basic probability mass associated with conflict. K is calculated by summing the products of the BPA's of all sets where the intersection is null. In the above example to calculate the combined probability that the attack is a DoS attack, first we need to calculate K. To calculate K we need to sum up three cells that contribute to conflict represented by empty intersections.

K = (0.99) (0.01) + (0.99) (0.01) + (0.99) (0.99) = 0.9999

The only non-zero value where the intersection is non-empty that yields a DoS attack has the value 0.0001. Therefore, using the Dempster's combination rule we can calculate the joint mass, which is

$m_{12}$ (DoS) = (0.01) (0.01) / (1 – 0.9999) = 1

Though there is highly conflicting evidence, the basic probability assignment for a DoS attack is 1. This corresponds to Bel (DoS) = 1, which means it was a DoS attack with no uncertainty. The reason for this is the normalization of the masses to exclude those associated with conflict. This is one of the weaknesses of D-S theory of evidence. It gives incorrect results when used in circumstances where there is significant conflict. But, in cases where we know for certain that evidence is less reliable, by assigning a reliability factor to the evidence we can improve the accuracy of our combination.

For example, consider the same example, but with a reliability factor taken into consideration. Assume that Sensor 1 has a reliability of 25% and Sensor 2 has reliability of 95%. Now do the same calculation to find the combined probability for a DoS attack.

| | | Sensor 1 | | | |
|---|---|---|---|---|---|
| Sensor2 | Attack | | Mac Spoof | DoS | Xmas Tree |
| | | Mass (M) | 0.99 * 0.25 = 0.2475 | .01 * 0.25 = 0.0025 | 0 * 0.25 = 0 |
| | Mac Spoof (MS) | 0 * 0.95 = 0 | 0 | 0 | 0 |
| | DoS (D) | 0.01 * 0.95 = 0.0095 | 0.00235125 | 0.00002375 | 0 |
| | Xmas Tree (XT) | 0.99 * 0.95 = 0.9405 | 0.23277375 | 0.00235125 | 0 |

K = 0.00235125 + 0.00235125 + 0.23277375= 0.23747625

$m_{12}$ (DoS) = 0.00002375 / (1 – 0.23747625) = 0.00002375 / 0.76252375 = 0.00003114657

As one can see, now the probability of a DoS attack is almost zero. This is one of the cases where we assigned 25% accuracy to one sensor and 95% accuracy to the 2nd sensor. By doing so, we essentially negated the unreliability of that sensor in the combining process. The D-S

theory does not provide this functionality; therefore this could be an improvement to increase the accuracy of intrusion detection.

## 5.3 Why Do We Need to Consider Reliability of Evidence?

Dempster's rule requires all belief sources to be reliable [Josang, 2002]. Also Josang [2002] stated that "The consensus operator does not make any assumption about reliability of the belief sources, but does of course not escape the 'garbage in, garbage out' principle." Wrong information is essentially evidence that is not accurate. More often than not, inaccurate evidence is given by unreliable sources.

In a perfect world one might find perfect or totally reliable observers. There are a variety of network sensors that can be used to monitor the wireless networks today. At a high level, they might perform the same task of "monitoring radio frequency signals". They might even see the same data flowing through the air. Of course it is still possible, that some are more powerful than others, which results in the ability of powerful sensors seeing more data in a wide range. Moreover it is likely that the way they process their data is very different. As data analysis is done through some code, and this code may have been written by totally different individuals, it is likely, they used different methods to achieve the same task. Therefore, each sensor is unique. Hence some sensors might be better at detecting certain attacks better than the others. In other words, if we use two different sensors to detect the same attack they may do so with different values of accuracy for the given attack. Some sensors might not detect a certain type of attack at all. As an example for this from our research, the AP-70 sensors combined with RF protect does not detect a Xmas tree scan while Snort detects it. Completely ignoring a sensor's capability in a distributed intrusion detection environment cannot make intrusion detection better. By taking into consideration that a sensor has a certain reliability value for a given attack will increase the likelihood of detecting the given attack.

In our research, to simulate the reliability of a sensor we use CPU load of a sensor as a factor that alters reliability in a controlled way. This is done in this research, so that we can show how reliability affects a sensor's detection capability in a controlled environment. More details of how the actual experiment was carried out will be provided in a later chapter.

## 5.4 Weakness' of the Method Used by Yu and Frincke

Yu and Frincke conducted their research using two RealSecure network sensors. One of the goals of their research is to show that even identical sensors could have different detection capabilities depending on their location. They introduced an exponentially weighted D-S theory to combine evidence which takes into account different locations sensors are located and their accuracy based on the location. However, Yu and Frincke only used one sensor type to conduct their research. In our view, to fully utilize the power of the D-S theory one needs to use a diverse set of sensors and combine the evidence. This way, one can detect a varied set of attacks that is very hard to detect with a single type of sensor.

We feel that our experiments have uncovered a weakness in the exponentially weighted D-S combination rule of Yu and Frincke. We shall show this error through the following

example. Yu and Frincke's exponentially weighted D-S combination rule is given below

$$m_{12}(A) = \frac{B \cap C = A,\ \Sigma\ [m_1(B)]^{w1}[m_2(C)]^{w2}}{1 - [B \cap C = \varnothing,\ \Sigma\ [m_1(B)]^{w1}[m_2(C)]^{w2}]}$$

The weakness shows up when $w_1$ and $w_2$ have values close to zero. For example, raising 0.95 to the power of 0.1 would result in 0.9948. As a matter of fact, raising any value from (0 to 1) to a positive exponent that is less than one (greater than zero) would result in a value that is greater than the original value. In certain cases, this could result in getting negative values for an attack state. This will have an effect in the calculation of K in the combination process that will cause it to be greater than one. This will cause the denominator of the equation to be a negative number, which causes the eventual combined value to be negative. When combining multiple sensors this negative value will adversely affect the end result.

We shall show the example calculations for the exponentially weighted D-S combination rule, when more than two sensors are used in the experiment. In the first step, the observations from the first two sensors are combined with Yu and Frincke's Extended D-S fusion. In the second step the results of the first step is combined with the evidence from the third sensor. In the third step, the results of the second step are combined with the evidence from the fourth sensor. In the fourth step the results of the third step are combined with the fifth sensor.

Yu and Frincke

Step 1

| Sensor | AP70-1 | Attack | | Xmas | ¬ Xmas | U |
|---|---|---|---|---|---|---|
| AP70-2 | Attack | | Reliability | 1 | 1 | 1 |
| | | Mass | | 0 | 0.333333 | 0.666667 |
| | Xmas | 0 | 1 | 0 | 0 | 0 |
| | ¬ Xmas | 0.5 | 1 | 0 | 0.166667 | 0.333333 |
| | U | 0.5 | 1 | 0 | 0.166667 | 0.333333 |

Step 2

| Sensor | AP70-Combined | Attack | | Xmas | ¬ Xmas | U |
|---|---|---|---|---|---|---|
| AP70-2 | Attack | | Reliability | 1 | 1 | 1 |
| | | Mass | | 0 | 0.666667 | 0.333333 |
| | Xmas | 0.535714 | 0.8458 | 0 | 0.393224 | 0.196612 |
| | ¬ Xmas | 0.392857 | 0.8458 | 0 | 0.302491 | 0.151245 |
| | U | 0.071429 | 0.8458 | 0 | 0.071534 | 0.035767 |

Step 3

| Sensor | Step 2 Combined Result | Attack | | Xmas | ¬ Xmas | U |
|---|---|---|---|---|---|---|
| AP70-2 | Attack | | Reliability | 1 | 1 | 1 |
| | | Mass | | 0.324028 | 0.865675 | 0.058946 |
| | Xmas | 0.769231 | 0.8457 | 0.25955 | 0.693415 | 0.047216 |
| | ¬ Xmas | 0.192308 | 0.8457 | 0.080363 | 0.214699 | 0.014619 |
| | U | 0.038462 | 0.8457 | 0.020603 | 0.055044 | 0.003748 |

Step 4

| Sensor | Step 3 Combined Result | Attack | | Xmas | ¬ Xmas | U |
|---|---|---|---|---|---|---|
| AP70-2 | Attack | | Reliability | 1 | 1 | 1 |
| | | Mass | | 1.447118 | 1.257011 | 0.016568 |
| | Xmas | 0.769231 | 0.8439 | 1.159704 | 1.007354 | 0.013278 |
| | ¬ Xmas | 0.192308 | 0.8439 | 0.359972 | 0.312683 | 0.004121 |
| | U | 0.038462 | 0.8439 | 0.092557 | 0.080398 | 0.00106 |

| Step 1 | | Step 2 | | Step 3 | | Step 4 | |
|---|---|---|---|---|---|---|---|
| K = | 0 | K = | 0.393224 | K = | 0.773778 | K = | 1.367326 |
| $m_{12}$(Xmas) | 0 | $m_{12}$(Xmas) | 0.324028 | $m_{12}$(Xmas) | 1.447118 | $m_{12}$(Xmas) | -3.44527 |
| $m_{12}$(¬Xmas) | 0.666667 | $m_{12}$(¬Xmas) | 0.865675 | $m_{12}$(¬Xmas) | 1.257011 | $m_{12}$(¬Xmas) | -1.081332 |
| M(U)= | 0.333333 | M(U)= | 0.058946 | M(U)= | 0.016568 | M(U)= | -0.002885 |

As one can see from the above example, combination through Yu and Frincke's method in certain cases results in negative numbers for probability values. This is clearly noticeable when there are more than two sensors involved in the combination. Also, they do not explicitly define what happens in the special case when both the weight and the mass is zero. This special case would require zero be raised to zero power. This could have a different meaning, depending on the context, according to the exponentiation article in Wikipedia that discusses "Zero to the zero power".

# 6 OUR RESEARCH AND PROPOSED IMPROVEMENTS

In our research we have addressed the weaknesses of the existing methods and tried to improve them. The proposed method will make additional adjustments the way probabilities are calculated to make intrusion detection more accurate. Also, five network sensors that fall into three diverse categories will be used to conduct intrusion detection. This will give us the capability to cover a broader spectrum of attacks in the research.

## 6.1 Intrusion Detection in Wireless Networks

In our research we have chosen to apply the D-S theory of evidence to combine information about intrusion detection from multiple sensors in wireless networks.

These days, wireless networks have become common in work places and homes. As wireless networks are being widely used, it is important that we take necessary precautions against intrusions into these networks. Though wired networks still exist, more often than not they co-exist with wireless networks. In wireless networks, a sensor may be able to access only a sub-set of the total number of packets. Hence the reliability of conclusions, reached by each sender may not be the same. Hence the application of D-S theory to detect intrusions in wireless networks may be of great value. However to the best of our knowledge, the D-S theory has not yet been applied to conduct intrusion detection in wireless networks.

## 6.2 Increased Number of Sensors

A major feature of the D-S theory of evidence is its ability to combine evidence in a distributed intrusion detection environment. Further it is able to combine evidence provided by multiple sensors. We found through our studies that researchers tend to utilize only two sensors which is the minimum required for data fusion. As increased number of sensors could give more evidence, in essence more knowledge about a network in question, we have utilized five sensors in our research. Data fusion involving more than two sensors is achieved by generalizing the Dempster's combination rule by iteration. For example, if we have already fused two sensors S1 and S2, we can take that result and treat it as a result from a single sensor and fuse it with sensor S3. By extension, we are able to combine any number of sensors.

## 6.3 Sensor Diversity Increased

One of the major advantages of the D-S theory of evidence is that it can combine evidence from a heterogeneous pool of observers. Unfortunately research in the field indicates this potential advantage has not been fully utilized. Siaterlis and Maglaris used two sensor types. Yu and Frincke only used one sensor type. Our study of the past material indicated that more than 95% of other researchers in the field also conducted research using only a single sensor type.

Our research examines the effects of having more than one intrusion detection system, and more than one sensor type in an IDS environment. Today, any large distributed network has to cater to both wired as well as wireless end-terminals. In such an environment, a single ID system may not be an effective solution to guard against potential attackers. Multiple ID systems and wireless and wired sensor systems generate alerts. A concatenation of all alerts can lead to a larger number of false positives. However, if we understand the environment and the characteristics of the sensors and the IDSs, the fusion of alerts from a variety of sources can reduce the number of false positives and false negatives and increase the number of true positives.

Our research includes the following sensors and IDSs to do intrusion detection.

1. Aruba AP-70 Sensors (2 sensors) and RF Protect distributed IDS
2. Snort sensor (1) using the Snort IDS
3. Wireshark sensors (2)

## 6.4 Progressively Evolving Reliability Factor (PERF)

To take into account the reliability of evidence we introduce a reliability coefficient known as $R_i$. Its value lies between zero and one. One corresponds to totally reliable evidence from the sensor or IDS for the type of attack, under consideration, and zero corresponds to totally unreliable.

$$R_i \mid 0 \leq R_i \leq 1$$

Instead of taking reliability for a sensor it can be broken down into reliability for each attack for a given sensor. This is because a sensor may detect a certain attack at certain reliability and another attack at a different reliability. For example, in our research we noticed that AP-70 sensors do not detect a Xmas tree scan. Hence instead of giving the AP-70 sensor with a reliability value, it could be differentiated down into each attack. We, therefore, assign a reliability value for a specific attack for a specific sensor.

In an environment of a very high load of normal packets, a computing system, having the sensor or the IDS, may be overwhelmed by the large processing required by the packets and it may not be able to process the attack packets. On the other hand, when the load of normal packets is less, it may be able to recognize the attack packets fast. To take into account such

situations, in our system, this coefficient ($R_i$) need not stay fixed throughout the intrusion detection period. In a real intrusion detection environment, this would mean updating reliability values periodically to reflect newly discovered evidence and the accuracy of the sensors detecting a given attack. If an attack was discovered (true positive) and one of the sensors failed to detect it, the security practitioner can lower the reliability value for the attack in question in the particular sensor. How much the value is decreased will be determined by the security practitioner according to available data. In other words, once the security practitioner determines that a sensor has given a false positive or a false negative she can decrease the sensor's reliability coefficient for a given attack by a certain factor which could be calculated by determining its current observed correctness for the given attack state. Likewise she can increase the reliability coefficient by a certain factor when it gives a true positive for a given attack state up to a maximum value of one corresponding to totally reliable. Therefore, instead of using a constant value, we can use a variable ($R_i$) that evolves during the experiment progressively.

In our research, this will be simulated by varying the CPU load of a sensor. It will be shown in a later section that when the CPU load is increased sensor reliability decreases and vice versa.

## 6.5 New Method of Basic Probability Assignment (Mass Calculation)

Our research of the field indicated that there is no well defined way to calculate belief masses in the Dempster Shafer theory. Also, there is no well defined way to calculate the uncertainty or the unknown state of a system. Most researchers do not define how the masses were calculated and just state the values of the masses. We adapted a method used by Jøsang [2000] to calculate the belief masses. Using this method, we shall show how the masses are calculated. For this example, assume that we are considering a Xmas tree attack. So our frame of discernment consists of (Xmas, ¬Xmas, Unknown). So, first we need to calculate the masses of these states for each sensor before we do the fusion. In our research we have done the mass calculation the following way,

- M (Xmas) = r / r + s +2
- M (¬Xmas) = s / r + s + 2
- M (Unknown) = 2 / r + s + 2

} When (U)ncertainty ≠ 0

r – Positive evidence, s – Negative evidence

Jøsang [2000] justifies this way of calculation in his paper "A logic for uncertain probabilities". There are a couple of reasons why this method is suitable to be used in our research. Using the Xmas attack as an example, we shall explain the reasons:

1. M (Xmas) needs to increase when r (positive evidence) increases
2. M (¬Xmas) needs to increase when s (negative evidence) increases

3. M (Unknown) needs to decrease when r + s (the amount of positive and negative evidence, or in other words total available evidence) increases

This makes perfect sense because as the amount of evidence increases, we get to know more and more, eliminating the need for mass to be assigned to uncertainty. Mathematically speaking for uncertainty to be zero it will require that we have an infinite amount of evidence. These can be graphically represented as follows.

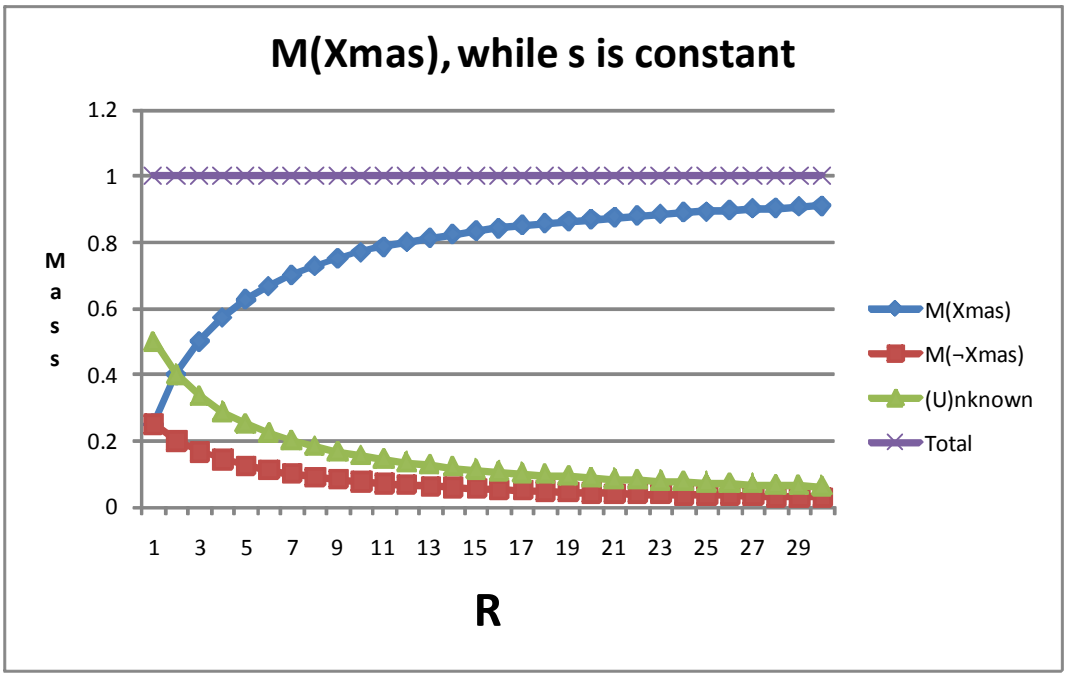


**Figure 6-1. M(Xmas) is an increasing Function of R**

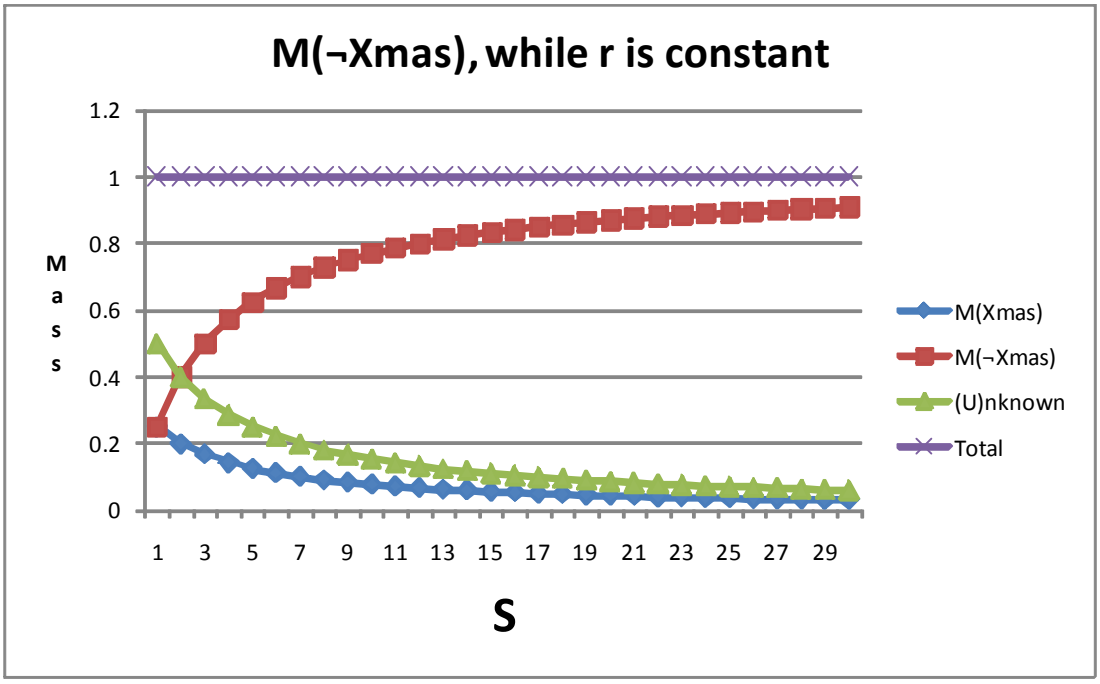


**Figure 6-2. M (¬Xmas) is an increasing Function of S**

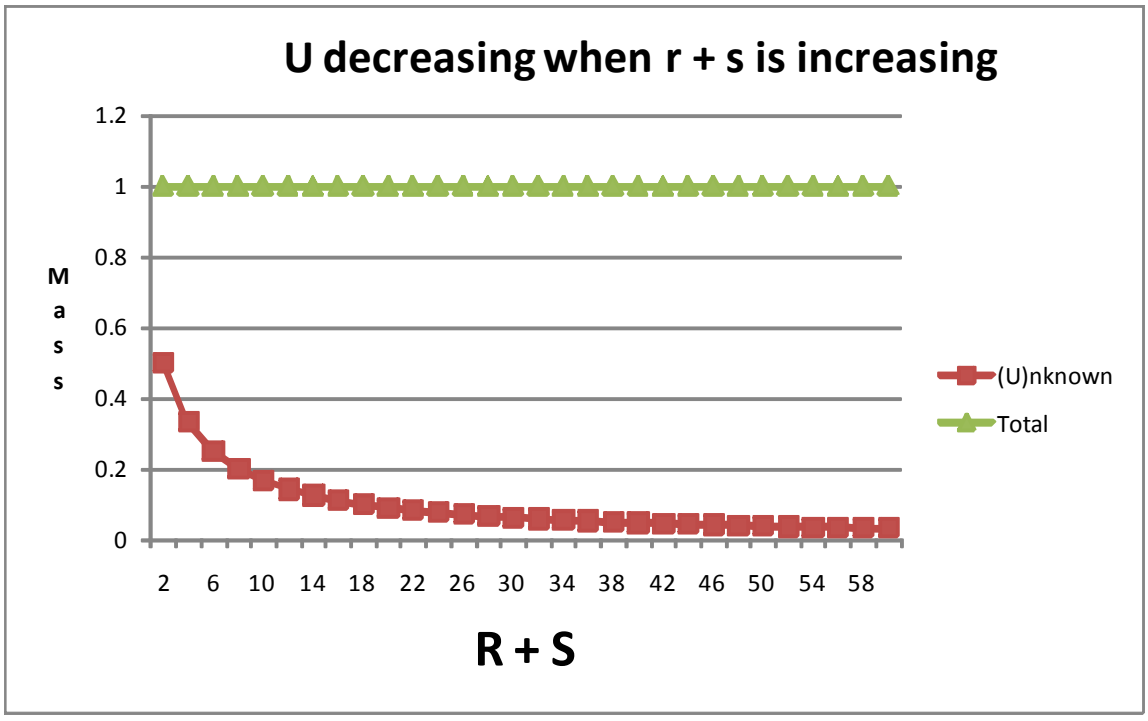


**Figure 6-3. U is a decreasing Function of (R + S)**

## 6.6 New Rule of Combination

In our research we addressed some weaknesses of Siaterlis and Maglaris and Yu and Frincke. Siaterlis and Maglaris considered all sensors in their research environment to be equally reliable for detecting denial of service attacks. Actually, this is assumed by the Dempster's combination rule because it considers all evidence to be of the same importance. In a distributed environment this does not hold true as explained earlier.

$$m_{12}(A) = \frac{B \cap C = A,\ \Sigma\ R_1\ m_1(B)\ R_2\ m_2(C)}{1 - [B \cap C = \varnothing,\ \Sigma\ R_1\ m_1(B)\ R_2\ m_2(C)]}$$

Where,

- $m_{12}$ (A) = Combined belief of the hypothesis A
- $m_1$(B) = Belief committed to hypothesis B as seen by the first sensor
- $m_2$(C) = Belief committed to hypothesis C as seen by the second sensor
- $R_1$ = Reliability of Sensor 1 (Progressively evolving values)
- $R_2$ = Reliability of Sensor 2 (Progressively evolving values)

As shown earlier, Yu and Frincke's exponential combination rule can run into potential problems (negative values) when combining evidence. Wu et al. [2002] also proposed a combination rule that uses weights proportionally for a purpose other than intrusion detection. However their method is different from ours as their method does not produce combination weights that add up to 1 (def. 5.1.2). Our method of combination which uses PERF D-S combination rule will guarantee there will be no negative values. Hence every mass value will be between zero and one [0, 1], multiplying it by a positive reliability factor which is also between zero and one [0, 1], will always result in a positive value that is less than or equal to one. Hence no negative values are possible for a given attack state. Therefore, the calculated K value is guaranteed to be a positive number less than one. Also, it will ensure that combination weights will add up to 1 as in definition 5.1.2. How it achieves the combined weights to be 1 will be shown in a later section.

We shall show the PERF D-S combination rule in practice with the same values we used to show the error in Yu and Frincke's method.

PERF D-S COMBINATION

Step 1

| Sensor | AP70-1 | Attack | | Xmas | ¬ Xmas | U |
|---|---|---|---|---|---|---|
| AP70-2 | Attack | | Reliability | 1 | 1 | 1 |
| | | Mass | | 0 | 0.333333 | 0.666667 |
| | Xmas | 0 | 1 | 0 | 0 | 0 |
| | ¬ Xmas | 0.5 | 1 | 0 | 0.166667 | 0.333333 |
| | U | 0.5 | 1 | 0 | 0.166667 | 0.333333 |

Step 2

| Sensor | AP70-Combined | Attack | | Xmas | ¬ Xmas | U |
|---|---|---|---|---|---|---|
| Snort | Attack | | Reliability | 1 | 1 | 1 |
| | | Mass | | 0 | 0.666667 | 0.333333 |
| | Xmas | 0.5357143 | 0.8458 | 0 | 0.302071 | 0.151036 |
| | ¬ Xmas | 0.3928571 | 0.8458 | 0 | 0.221519 | 0.11076 |
| | U | 0.0714286 | 0.8458 | 0 | 0.040276 | 0.020138 |

Step 3

| Sensor | Step 2 Combined Result | Attack | | Xmas | ¬ Xmas | U |
|---|---|---|---|---|---|---|
| Wireshark | Attack | | Reliability | 1 | 1 | 1 |
| | | Mass | | 0.216406 | 0.533801 | 0.249794 |
| | Xmas | 0.7692308 | 0.8457 | 0.14078 | 0.347258 | 0.1625 |
| | ¬ Xmas | 0.1923077 | 0.8457 | 0.035195 | 0.086814 | 0.040625 |
| | U | 0.0384615 | 0.8457 | 0.007039 | 0.017363 | 0.008125 |

Step 4

| Sensor | Step 3 Combined Result | Attack | | Xmas | ¬ Xmas | U |
|---|---|---|---|---|---|---|
| Wireshark | Attack | | Reliability | 1 | 1 | 1 |
| | | Mass | | 0.502504 | 0.23448 | 0.263016 |
| | Xmas | 0.7692308 | 0.8439 | 0.326202 | 0.152214 | 0.170738 |
| | ¬ Xmas | 0.1923077 | 0.8439 | 0.081551 | 0.038053 | 0.042685 |
| | U | 0.0384615 | 0.8439 | 0.01631 | 0.007611 | 0.008537 |

| Step 1 | | Step 2 | | Step 3 | | Step 4 | |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| K = | 0 | K = | 0.302071 | K = | 0.382453 | K = | 0.233764 |
| $m_{12}$(Xmas) | 0 | $m_{12}$(Xmas) | 0.216406 | $m_{12}$(Xmas) | 0.502504 | $m_{12}$(Xmas) | 0.669833 |
| $m_{12}$(¬Xmas) | 0.666667 | $m_{12}$(¬Xmas) | 0.533801 | $m_{12}$(¬Xmas) | 0.23448 | $m_{12}$(¬Xmas) | 0.115302 |
| M(U)= | 0.333333 | M(U)= | 0.249794 | M(U)= | 0.263016 | M(U)= | 0.214865 |

As one can see, the results produced for the same values as in Yu and Frincke's method are within the valid range of probability values. We shall provide fusion results, based on measured values of r and s in a later section.

# 7 PERF D-S CALCULATION

In this section we shall show how Progressively Evolving Reliability Factor (PERF) D-S calculations are carried out.

## 7.1 Evidence Gathering

We begin with the process of evidence gathering by mounting two types of attacks by using a test-bed in the laboratory.

1. DoS Attack
2. Xmas Tree Scan

More details about these attacks can be found in sections 5.3 and 5.6. First we carried out DoS attacks and gathered the required evidence from them. The DoS attacks were carried out using the CommView tool, described in section 6.2, to flood the victim with packets. Once the attack finishes, we record the evidence from all the five sensors. More specifically, we take from each sensor anything that supports the believed attack state or anything that refutes the attack state. In this research we denote,

- r = positive evidence
- s = negative evidence

This is quite similar to Jøsang [2000]. For any positive evidence, r will be incremented and for any negative evidence s will be incremented. By how much r or s will be incremented will depend on many factors. We defined these guidelines for our research as in the following tables, since no such guidelines exist in the literature. We found that by using the guidelines, the belief mass, in each case, provided a value, which when combined with others, generated the correct results for the cases of an attack for both the attacks.

| **RF Protect Evidence (DoS)** | **r** | **s** |
|---|---|---|

| | | |
|---|---|---|
| **For every totally favorable alert** | 5 points | |
| **For every totally unfavorable alert** | | 5 points |
| **For every partially favorable alert** | 1 points | |
| **For every partially unfavorable alert** | | 1 points |
| | | |
| | | |

**Table 7-1. Point Assignment for RF Protect Evidence in a DoS attack**

| **Snort Evidence (DoS)** | **r** | **s** |
|---|---|---|
| **For every totally favorable alert** | 5 points | |
| **For every totally unfavorable alert** | | 5 points |
| **For every partially favorable alert** | 1 points | |
| **For every partially unfavorable alert** | | 1 points |
| **For every 4000 packets captured in a DoS attack** | 1 point | |
| | | |
| | | |

**Table 7-2. Point Assignment for Snort in a DoS attack**

| **Wireshark Evidence (DoS)** | **r** | **s** |
|---|---|---|
| | | |
| **For every 4000 packets captured in a DoS attack** | 1 point | |
| **For other packets (packets that are used to attack the victim is not considered in this case) indicating no attack state, consider the amount of packets and assign a subjective value to S – the reason being if we find packets that we don't expect to be there, that shows the network could possibly be in another non-attack state** | | 1 point |
| **For every two packets indicating an attack state, such as ICMP Destination Unreachable (Wireshark filter for finding this is icmp.type == 3 && icmp.code == 3)** | 1 point | |

**Table 7-3. Point Assignment for Wireshark in a DoS attack**

| **RF Protect Evidence (Xmas)** | **r** | **s** |
|---|---|---|
| | | |
| **For every totally favorable alert** | 5 points | |
| **For every totally unfavorable alert** | | 5 points |
| **For every partially favorable alert** | 1 points | |
| **For every partially unfavorable alert** | | 1 points |
| | | |
| | | |

**Table 7-4. Point Assignment for RF Protect Evidence in a Xmas attack**

| Snort Evidence (Xmas) | r | s |
|---|---|---|
| **For every totally favorable alert** | 5 points | |
| **For every totally unfavorable alert** | | 5 points |
| **For every partially favorable alert** | 1 points | |
| **For every partially unfavorable alert** | | 1 points |
| **For every 4000 packets captured indicating no attack state** | | 1 point |
| | | |
| | | |

**Table 7-5. Point Assignment for Snort in a Xmas attack**

| Wireshark Evidence (Xmas) | r | s |
|---|---|---|
| | | |
| **For every 50 Xmas packets captured**<br>**Wireshark Filter used to select Xmas packets is**<br>**(tcp.flags.urg == 1 && tcp.flags.push == 1 && tcp.flags.fin == 1)** | 1 point | |
| **For every 4000 packets captured indicating no attack state** | | 1 point |
| **An Xmas scan is preceded by scanning the network for available hosts (who has) using the ARP protocol, if this phase is present in the evidence (stage before Xmas packets are sent)** | 2 points | |
| | | |

**Table 7-6. Point Assignment for Wireshark in a Xmas attack**

For a higher bandwidth network, that has high traffic for a very long period, one may take the number of packets in each of the above cases as a larger value. In our test-bed, we chose to increment 's' by 1 point for 4000 packets indicating no attack state.

## 7.2 Belief Mass Calculation and Combination

Once the evidence is gathered and points assigned accordingly, we can calculate the belief masses for each state. In our research the belief mass was calculated in the following way. The data given below is for a single sensor that gathered data in a Xmas tree scan.

| r | s | M(Xmas) | M(¬Xmas) | (U)nknown |
|---|---|---|---|---|
| 4 | 1 | 0.5714286 | 0.1428571 | 0.285714286 |

$$M(Xmas) = r/r+s+2$$

$$M(\neg Xmas) = s/r+s+2$$

$$U = 2/r+s+2$$

When $U \neq 0$

Also, it is required that

M (Xmas) + M (¬Xmas) + U = 1

We adapted this method of calculation from Jøsang [2000]. The Dempster-Shafer theory does not explicitly define a way to calculate basic probability assignment (BPA). We shall show a complete fusion example through PERF D-S below. In this example, we assume that all sensors are totally reliable and hence their reliability is 1. We shall later show the same example with the same evidence but with different reliability values.

First the evidence gathered from 5 sensors is analyzed and belief masses are calculated as follows.

| $r^A$ | $s^A$ | M (Xmas) | M (¬Xmas) | (U)nknown |
|---|---|---|---|---|
| 0 | 4 | 0 | 0.6666667 | 0.333333333 |
| $r^B$ | $s^B$ | M (Xmas) | M (¬Xmas) | (U)nknown |
| 0 | 4 | 0 | 0.6666667 | 0.333333333 |

**Table 7-7. Evidence from AP-70 Sensors**

| $r^D$ | $s^D$ | M(Xmas) | M(¬Xmas) | (U)nknown |
|---|---|---|---|---|
| 15 | 5 | 0.6818182 | 0.2272727 | 0.090909091 |

**Table 7-8. Evidence from Snort Sensor**

| $r^D$ | $s^D$ | M (Xmas) | M (¬Xmas) | (U)nknown |
|---|---|---|---|---|
| 22 | 4 | 0.7857143 | 0.1428571 | 0.071428571 |

**Table 7-9. Evidence from Wireshark Sensor (1)**

| $r^D$ | $s^D$ | M (Xmas) | M (¬Xmas) | (U)nknown |
|---|---|---|---|---|
| 22 | 4 | 0.7857143 | 0.1428571 | 0.071428571 |

**Table 7-10. Evidence from Wireshark Sensor (2)**

After belief mass assignment, we do the combination of these masses to form a more informed decision. Since there are 5 sensors, we need to perform 4 fusions. First data from two AP-70 sensors are fused; the result of that combination is then fused with evidence from the Snort sensor. That result would be fused with Wireshark sensor 1 and the resulting values would be fused with Wireshark sensor 2. The steps are shown in table 10-11.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | PERF D-S COMBINATION | | | | | |
| Step 1 | | | | | | | |
| | Sensor | AP70-1 | Attack | | Xmas | ¬ Xmas | U |
| | AP70-2 | Attack | | Reliability | 1 | 1 | 1 |
| | | | Mass | | 0 | 0.6666667 | 0.333333 |
| | | Xmas | 0 | 1 | 0 | 0 | 0 |
| | | ¬ Xmas | 0.666667 | 1 | 0 | 0.4444444 | 0.222222 |
| | | U | 0.333333 | 1 | 0 | 0.2222222 | 0.111111 |
| | | | | | | | |
| Step 2 | | | | | | | |
| | Sensor | AP70-Combined | Attack | | Xmas | ¬ Xmas | U |
| | Snort | Attack | | Reliability | 1 | 1 | 1 |
| | | | Mass | | 0 | 0.8888889 | 0.111111 |
| | | Xmas | 0.681818 | 1 | 0 | 0.6060606 | 0.075758 |
| | | ¬ Xmas | 0.227273 | 1 | 0 | 0.2020202 | 0.025253 |
| | | U | 0.090909 | 1 | 0 | 0.0808081 | 0.010101 |
| | | | | | | | |
| Step 3 | | | | | | | |
| | Sensor | Step 2 Combined Result | Attack | | Xmas | ¬ Xmas | U |
| | Wireshark AD | Attack | | Reliability | 1 | 1 | 1 |
| | | | Mass | | 0.192308 | 0.7820513 | 0.025641 |
| | | Xmas | 0.785714 | 1 | 0.151099 | 0.6144689 | 0.020147 |
| | | ¬ Xmas | 0.142857 | 1 | 0.027473 | 0.1117216 | 0.003663 |
| | | U | 0.071429 | 1 | 0.013736 | 0.0558608 | 0.001832 |
| | | | | | | | |
| Step 4 | | | | | | | |
| | Sensor | Step 3 Combined Result | Attack | | Xmas | ¬ Xmas | U |
| | Wireshark Dul | Attack | | Reliability | 1 | 1 | 1 |
| | | | Mass | | 0.516624 | 0.4782609 | 0.005115 |
| | | Xmas | 0.785714 | 1 | 0.405919 | 0.3757764 | 0.004019 |
| | | ¬ Xmas | 0.142857 | 1 | 0.073803 | 0.068323 | 0.000731 |
| | | U | 0.071429 | 1 | 0.036902 | 0.0341615 | 0.000365 |

**Table 7-11. Example: PERF D-S Combination for a Xmas Tree Scan with Sensors, assumed to be Totally Reliable**

| Step 1 | |
|---|---|
| | |
| K = | 0 |
| $m_{12}$(Xmas) | 0 |
| $m_{12}$(¬Xmas) | 0.888888889 |
| M(U)= | 0.111111111 |

| Step 2 | |
|---|---|
| | |
| K = | 0.606060606 |
| $m_{12}$(Xmas) | 0.192307692 |
| $m_{12}$(¬Xmas) | 0.782051282 |
| M(U)= | 0.025641026 |

| Step 3 | |
|---|---|
| | |
| K = | 0.641941392 |
| $m_{12}$(Xmas) | 0.516624041 |
| $m_{12}$(¬Xmas) | 0.47826087 |
| M(U)= | 0.00511509 |

| Step 4 | |
|---|---|
| | |
| K = | 0.449579832 |
| $m_{12}$(Xmas) | 0.811815466 |
| $m_{12}$(¬Xmas) | 0.187520743 |
| M(U)= | 0.00066379 |

The final result is the mass assigned to the state M (Xmas) indicating the likelihood of a Xmas tree scan which is 0.8118 or 81.18%. The mass assigned to the state M(¬Xmas) is 0.1875 or 18.75%. The value of uncertainty is 0.0006637 or 0.066%.

Now we consider the same example but with different reliabilities for the sensors. By using reliability we can negate the effect of evidence that has a larger likelihood of being false. If we know that evidence from a certain sensor/s is erroneous or incorrect to a certain degree, we can reduce the effect of that evidence by using the reliability of that sensor. In the real world, reliability of a sensor can be calculated through historical data available and other knowledge available about the sensor's performance. Any manufacturer of IDS systems will specify the kind of attacks the IDS detects. Depending on that knowledge one can adjust the reliability of that sensor for a certain attack. We shall give an example that uses the same data as the earlier example but with different reliabilities for the sensors. First, the modified reliabilities are given for the 5 sensors as below:

| Reliability | Value |
|---|---|
| AP-70 - 1 | 1 |
| AP-70 - 2 | 1 |
| Snort | 0.95 |
| Wireshark AD | 0.97 |
| Wireshark Dul | 0.93 |

Now, what we expect from the results is M(Xmas) state which is less than 81.18%. The reason being, 3 of the sensors that provided support to the Xmas state have reliability values that are less than the first example. Also the AP-70 sensors have unmodified reliability values in this example. Hence their negative evidence will not make a difference as compared to last example. Now, let's take a look at the combination of evidence with the same evidence as before.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | PERF D-S COMBINATION | | | | | |
| Step 1 | | | | | | | |
| | Sensor | AP70-1 | Attack | | Xmas | ¬ Xmas | U |
| | AP70-2 | Attack | | Reliability | 1 | 1 | 1 |
| | | | Mass | | 0 | 0.6666667 | 0.333333 |
| | | Xmas | 0 | 1 | 0 | 0 | 0 |
| | | ¬ Xmas | 0.666667 | 1 | 0 | 0.4444444 | 0.222222 |
| | | U | 0.333333 | 1 | 0 | 0.2222222 | 0.111111 |
| | | | | | | | |
| Step 2 | | | | | | | |
| | Sensor | AP70-Combined | Attack | | Xmas | ¬ Xmas | U |
| | Snort | Attack | | Reliability | 1 | 1 | 1 |
| | | | Mass | | 0 | 0.8888889 | 0.111111 |
| | | Xmas | 0.681818 | 0.95 | 0 | 0.5757576 | 0.07197 |
| | | ¬ Xmas | 0.227273 | 0.95 | 0 | 0.1919192 | 0.02399 |
| | | U | 0.090909 | 0.95 | 0 | 0.0767677 | 0.009596 |
| | | | | | | | |
| Step 3 | | | | | | | |
| | Sensor | Step 2 Combined Result | Attack | | Xmas | ¬ Xmas | U |
| | Wireshark AD | Attack | | Reliability | 1 | 1 | 1 |
| | | | Mass | | 0.169643 | 0.689881 | 0.140476 |
| | | Xmas | 0.785714 | 0.97 | 0.129292 | 0.5257878 | 0.107063 |
| | | ¬ Xmas | 0.142857 | 0.97 | 0.023508 | 0.0955978 | 0.019466 |
| | | U | 0.071429 | 0.97 | 0.011754 | 0.0477989 | 0.009733 |
| | | | | | | | |
| Step 4 | | | | | | | |
| | Sensor | Step 3 Combined Result | Attack | | Xmas | ¬ Xmas | U |
| | Wireshark Dul | Attack | | Reliability | 1 | 1 | 1 |
| | | | Mass | | 0.550491 | 0.3613513 | 0.088158 |
| | | Xmas | 0.785714 | 0.93 | 0.402252 | 0.2640446 | 0.064418 |
| | | ¬ Xmas | 0.142857 | 0.93 | 0.073137 | 0.0480081 | 0.011712 |
| | | U | 0.071429 | 0.93 | 0.036568 | 0.0240041 | 0.005856 |

**Table 7-12. Example PERF D-S Combination for a Xmas tree scan with 3 sensors having reduced reliability**

| Step 1 | | Step 2 | | Step 3 | |
|---|---|---|---|---|---|
| | | | | | |
| K = | 0 | K = | 0.575757576 | K = | 0.549295493 |
| $m_{12}$(Xmas) | 0 | $m_{12}$(Xmas) | 0.169642857 | $m_{12}$(Xmas) | 0.550491153 |
| $m_{12}$(¬Xmas) | 0.888888889 | $m_{12}$(¬Xmas) | 0.689880952 | $m_{12}$(¬Xmas) | 0.361351324 |
| M(U)= | 0.111111111 | M(U)= | 0.14047619 | M(U)= | 0.088157524 |

| Step 4 | |
|---|---|
| | |
| K = | 0.337181256 |
| $m_{12}$(Xmas) | 0.75923932 |
| $m_{12}$(¬Xmas) | 0.12631585 |
| M(U)= | 0.11444483 |

The value of M (Xmas) = 75.92% which is less than what we got with total reliability which was 81.18%. This is what we expected by lowering the reliability of the sensors. One of the shortcomings of using a reliability factor is that the total of $m_{12}$(Xmas) + $m_{12}$(¬Xmas) + U does not equal one after combination. This is because reliability is usually less than 1 for sensors. This is the case in every combination when you consider reliability. In the regular Dempster-Shafer theory, combined mass is calculated as follows.

$m_{12}(Xmas) = [(m_1(Xmas).m_2(Xmas) + m_1(Xmas).m_2(U) + m_2(Xmas).m_1(U))] / 1 - K$

$m_{12}(¬Xmas) = (m_1(¬Xmas).m_2(¬Xmas) + m_1(¬Xmas).m_2(U) + m_2(¬Xmas).m_1(U)) / / 1 - K$

$m_{12}(U) = m_1(U).m_2(U) / / 1 - K$

$K = (m_1(Xmas).m_2(¬Xmas) + m_1(¬Xmas).m_2(Xmas))$

In our method (PERF), we do the calculations as follows.

$m_{12}(Xmas) = [(R_1.m_1(Xmas).\ R_2.m_2(Xmas) + R_1.m_1(Xmas).\ R_2.m_2(U) + R_2.m_2(Xmas).\ R_1.m_1(U))] / (1 – K)$

$m_{12}(¬Xmas) = (R_1.m_1(¬Xmas).\ R_2.m_2(¬Xmas) + R_1.m_1(¬Xmas).\ R_2.m_2(U) + R_2.m_2(¬Xmas).\ R_1.m_1(U)) / / (1 – K)$

$m_{12}(U) = 1 – [m_{12}(Xmas) + m_{12}(¬Xmas)]$

$K = (R_1.m_1(Xmas).\ R_2.m_2(\neg Xmas) + R_1.m_1(\neg Xmas).\ R_2.m_2(Xmas))$

When reliability is 1 (totally reliable) for all sensors, the two sets of formulae become the same.

**Dempster-Shafer**

- $m_{12}(U) = m_1(U).m_2(U) / / (1 - K)$

**PERF**

- $m_{12}(U) = 1 - [m_{12}(Xmas) + m_{12}(\neg Xmas)]$

But, when taking reliability into consideration, they are not equal. In other words PERF's

$m_{12}(Xmas) + m_{12}(\neg Xmas) + U \neq 1$ when reliability is less than 1

In PERF, we select the value of $m_{12}(U)$ in such a way that the total will add up to 1. This way is logical, because when reliability is taken into consideration, what we lose from the actual mass can be attributed to uncertainty. Yu and Frincke's method does not address this fact in their method. Hence, in their method,

$m_{12}(Xmas) + m_{12}(\neg Xmas) + U \neq 1$ when reliability is less than 1

This is one of the reasons, the Yu and Frincke's method starts showing a weakness in some cases.

# 8 EXPERIMENTS AND ANALYSIS

All the experiments were conducted using 5 wireless sensors and one attacker and one victim. A summary of the sensors and computers and wireless routers used are given below with the specifications of the CPU and random access memory (RAM).

| Sensor/Computer | CPU | RAM |
|---|---|---|
| Linksys WRT310N Router | | |
| Attacker | Core 2 Duo 2.0 Ghz | 4GB DDR2 |
| Victim | Pentium D | 2GB DDR2 |
| Snort | Core 2 Duo 1.6 Ghz | 2GB DDR2 |
| Wireshark AD | Core 2 Duo 1.6 Ghz | 2GB DDR2 |
| Wireshark Dul | Pentium 4 2.8 Ghz | 512 MB DDR |
| RF Protect | Pentium 4 | 256 MB RDRAM |
| AP-70 - 1 (Connects to RF Protect) | | |
| AP-70 - 2 (Connects to RF Protect) | | |

**Table 8-1. Computers, Router and Sensors used in the Experiment**

The sensor setup and computer setup used in the experiment are given in the following diagram.

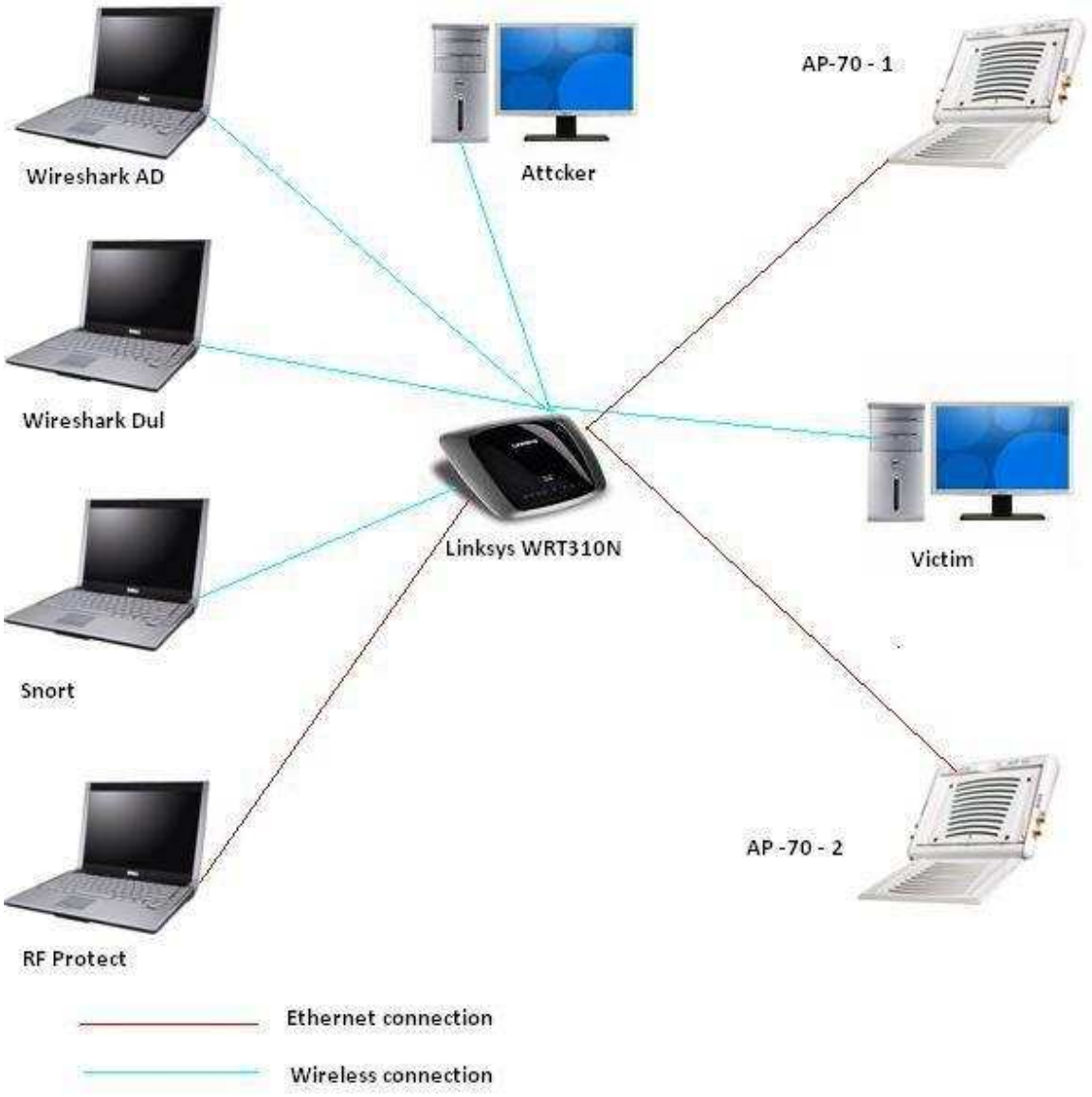


**Figure 8-1. Setup of the test-bed (Sensors, Computers and Router)**

## 8.1 Reliability of Sensors and CPU Utilization in DoS Experiments

Since PERF adds the idea of reliability in fusion, we needed sensors, for which the reliability could be adjusted. In the experimental setting, we needed a way to simulate the reliability of a sensor. Therefore we introduced CPU load as a parameter. In this section we shall show that when the CPU load increases the reliability of a software sensor, decreases. Utilizing that fact, we conducted several experiments with various CPU loads.

In this thesis, to obtain a variable reliability factor for the sensors, we define the reliability of a sensor as a function of the CPU load of the sensor. In other words by modifying the CPU load of a sensor, the reliability of a sensor will be increased or decreased. We conducted several experiments. The end results of the experiments proved that when the CPU load is increased the reliability of a sensor decreases. In the experimental setting, we were able to successfully modify the CPU load of the Wireshark sensors and the Snort sensor. The AP-70 sensor's reliability was left untouched as we did not find a practical way of loading the AP-70 sensors with CPU loading software that can be measured in an experimental setting. The AP-70's are hardware devices specializing in wireless monitoring whereas Wireshark and Snort are software devices, installed on actual computers that monitored the airwaves in promiscuous mode.

The metric used to measure the reliability of a sensor was the number of packets a sensor captured under a certain CPU load. By sending constant sized packets at a constant speed throughout the experiment, one can observe how many packets each sensor captures. Conducting the same experiment at varying CPU loads in the sensors, one can observe how the CPU load affects the packet capturing ratio of the sensor. Using this one can determine how reliable a sensor is under different CPU loads.

Several packet flooding experiments were conducted to observe the reliability of a sensor with varying CPU loads. In this section only the average values obtained from 30 experiments with different CPU loads are provided.

First we carried out the experiment without modifying the CPU load. The CPU load, in this case, is the normal CPU load when just the sensor software is running with the operating system, with no other load. Then CPU load was increased gradually up to 90% of the CPU power utilized at the end. CPU load was varied from 30% to 50%, 70% and 90% of the total CPU power.

In each of the five cases, the experiments consisted of sending a flood of packets. The same experiment was repeated 6 times and the average value was selected as the reliability of the sensor. The attacker fired the packets at 5000 packets per second at the victim while the sensors were sniffing the network in promiscuous mode.

The results below give the averages of these 30 experiments at different CPU utilization levels.

| CPU Utilization - Snort AD | CPU Utilization - Wireshark Dul | CPU Utilization - Wireshark AD | Packet Rate per second | Packet Size in Bytes | AVG # of Captured Packets AD Snort | AVG Percentage Captured from Total - Snort AD | AVG # of Captured Packets Dul Wireshark | AVG Percentage Captured from Total - Wireshark Dul | AVG # of Captured Packets AD Wireshark | AVG Percentage Captured from Total - Wireshark AD | AVG Total Packets Sent |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | |
| No Extra Load | No Extra Load | No Extra Load | 5000 | 250 | 9596 | 22.86% | 9579 | 22.82% | 9593 | 22.86% | 41967 |
| | | | | | | | | | | | |
| 30% | 30% | 30% | 5000 | 250 | 9449 | 21.94% | 9407 | 21.84% | 9459 | 21.96% | 43067 |
| | | | | | | | | | | | |
| 50% | 50% | 50% | 5000 | 250 | 9269 | 21.28% | 9180 | 21.08% | 9269 | 21.28% | 43550 |
| | | | | | | | | | | | |
| 70% | 70% | 70% | 5000 | 250 | 8848 | 20.22% | 8728 | 19.94% | 8848 | 20.22% | 43767 |
| | | | | | | | | | | | |
| 90% | 90% | 90% | 5000 | 250 | 6676 | 15.56% | 6564 | 15.30% | 6672 | 15.55% | 42900 |
| | | | | | | | | | | | |
| | | | | | | | | | | | |

**Table 8-2 Reliability Values (Averages) from 30 experiments**

The above table displays the average percentage of packets captured at various CPU utilizations during high speed packet floods of 5000 packets per second. The average was taken from a total of 30 experiments (6 each at a given CPU utilization). The data are for 2 Wireshark sensors and one Snort sensor sniffing the network in promiscuous mode.

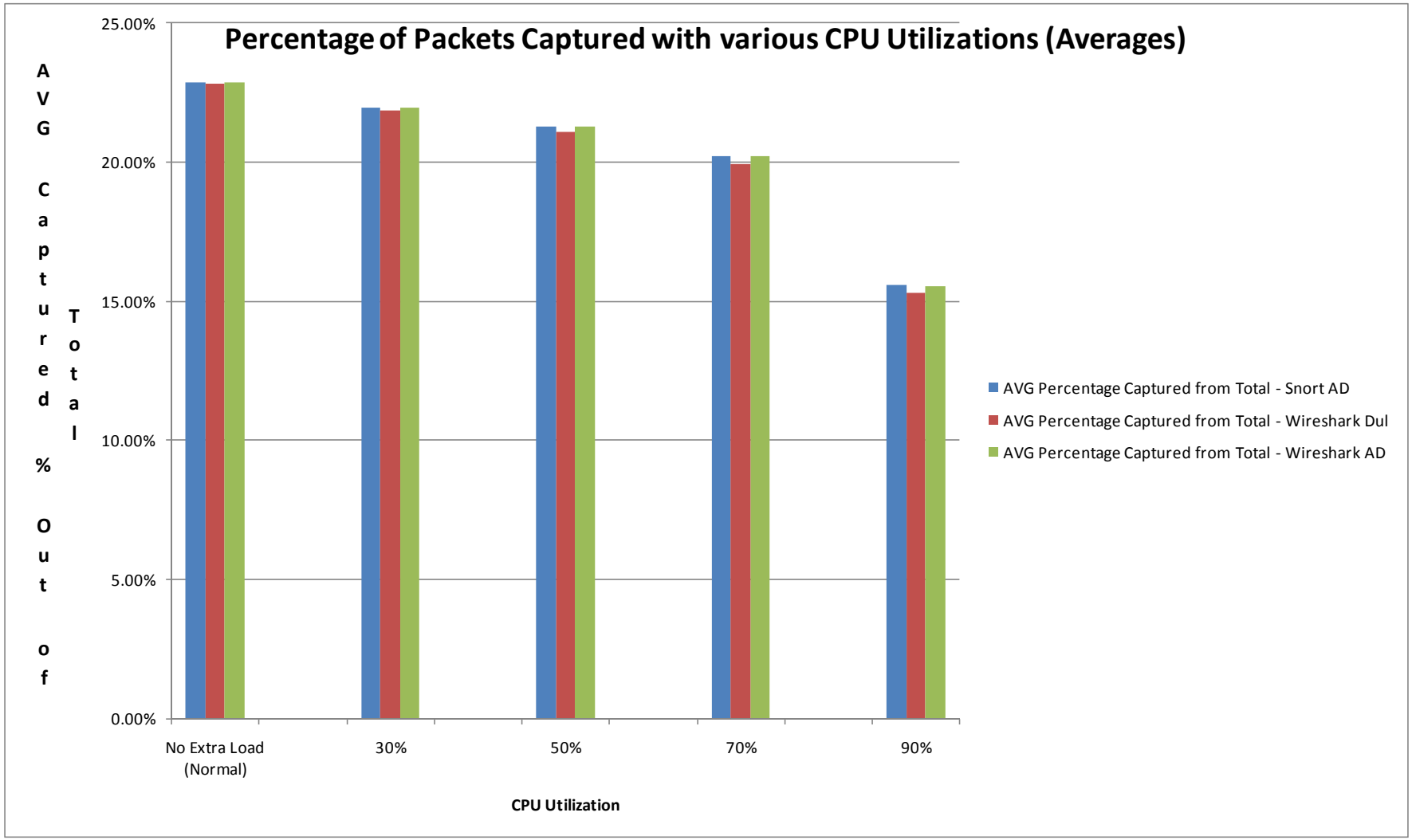


**Figure 8-2. Average Percentage of packets captured at various CPU Utilizations**

| AVG % Packet Difference from First | Relative Reliability of Snort AD | Wireshark Dul, AVG % of Packet Difference from First | Relative Reliability of Wireshark Dul | AVG % of Packet Difference from First | Relative Reliability of Wireshark AD |
|---|---|---|---|---|---|
| | | | | | |
| 0.00% | 100% | 0.00% | 100% | 0.00% | 100% |
| | | | | | |
| -0.92% | 99.08% | -0.98% | 99.02% | -0.90% | 99.10% |
| | | | | | |
| -1.58% | 98.42% | -1.75% | 98.25% | -1.58% | 98.42% |
| | | | | | |
| -2.65% | 97.35% | -2.88% | 97.12% | -2.64% | 97.36% |
| | | | | | |
| -7.30% | 92.70% | -7.52% | 92.48% | -7.31% | 92.69% |
| | | | | | |
| | | | | | |

**Table 8-3. Average Relative Reliability of Sensors**

Instead of assigning absolute reliability values to the sensors, we calculated them on a relative basis. Therefore, at the start without any CPU loading software, the sensors were given a total (100%) reliability. As the experiment progressed, different readings were taken at varying CPU loads. First, when there is no extra load on the CPU, we calculate the captured packet percentage by

- Captured Packet Percentage = Number of captured packets / Number of sent packets

We do this procedure for every experiment at varying CPU loads. Then, at each experiment we calculate the percentage difference from the beginning of the experiment. Through this procedure we calculate the relative reliability of a sensor at a given CPU load.

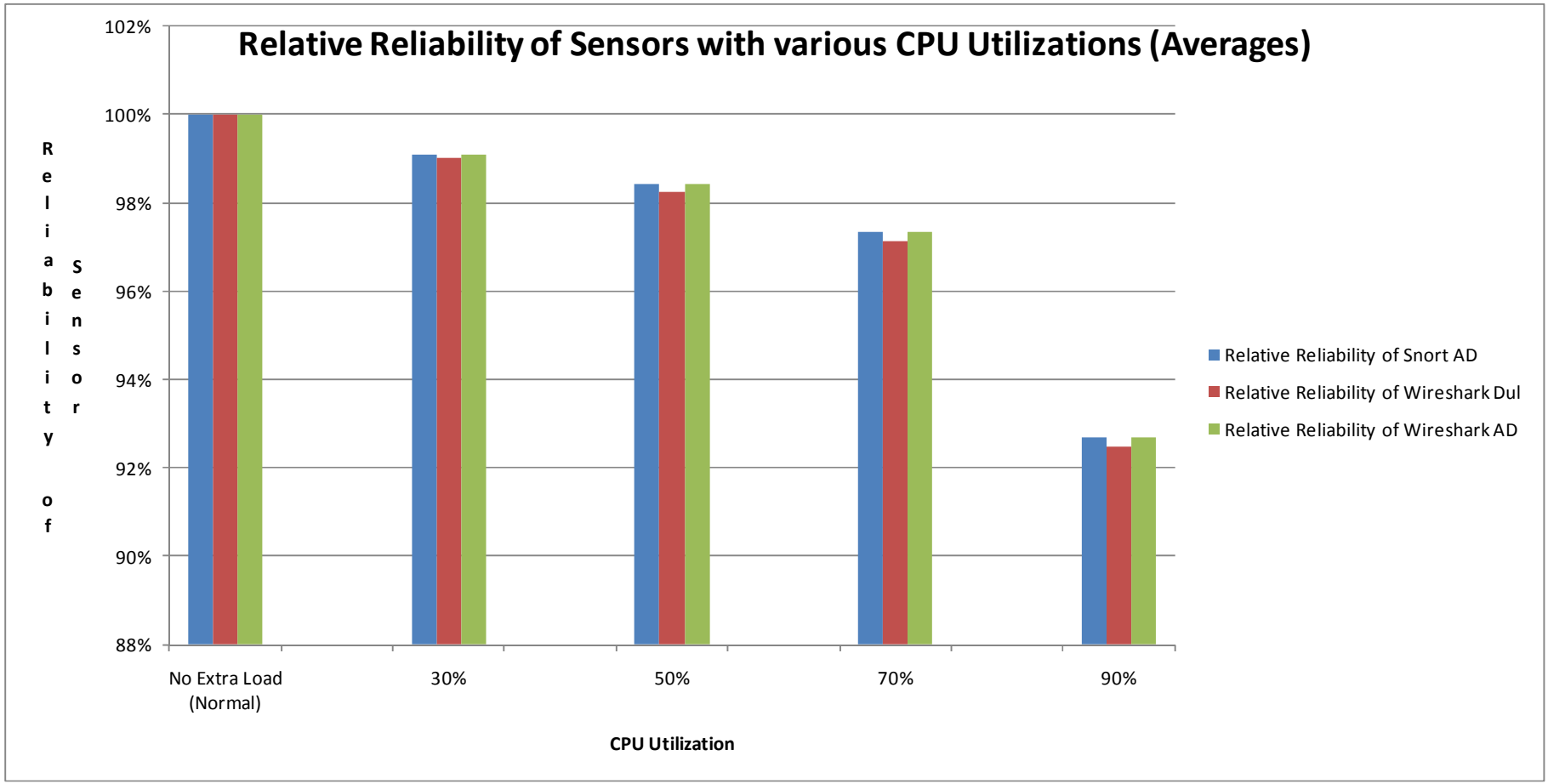


**Figure 8-3. Average Relative Reliability of Sensors with increasing CPU Utilizations**

The experimental results clearly showed that a sensor captures fewer packets when the CPU load is increased. Therefore, by increasing the CPU utilization of a sensor, the reliability of a specific sensor can be decreased by a certain factor. The speed of 5000 packets per second was chosen because it was the maximum allowed by the CommView software, which we have used to flood packets.

## 8.2 Reliability of Sensors and CPU Utilization in Xmas Tree Experiments

To test the reliability of sensors under a Xmas tree attack, we conducted another set of experiments. In this experiment, the attacker conducts a series of Xmas tree scans on the network using the Zenmap network scanner under varying CPU loads. More details about Xmas tree attacks and Zenmap can be found in section 5.6. First the attacker does the Xmas scan with no CPU loading software. Also, the attacker fires packets at 500 packets per second toward the victim, so that the Xmas scan becomes less obvious. As a Xmas does not use a large number of packets for a total scan, by having the extra packet stream, it creates a real network environment by having other packets flowing through the network. The reliability calculations in the Xmas scans are done in a manner similar to that for DoS attack. We conducted 5 Xmas tree scans at Normal, 30%, 50%, 70%, 90% CPU loads and repeated it for 5 times for a total of 25 Xmas tree scans and calculated the average reliability of a sensor during a Xmas tree scan. The results were as follows.

| CPU Utilization - Snort AD | CPU Utilization - Wireshark Dul | CPU Utilization - Wireshark AD | Packet Rate per second | Packet Size in Bytes | AVG # of Captured Packets AD Snort | AVG Percentage Captured from Total - Snort AD | AVG # of Captured Packets Dul Wireshark | AVG Percentage Captured from Total - Wireshark Dul | AVG # of Captured Packets AD Wireshark | AVG Percentage Captured from Total - Wireshark AD | AVG CommView Packets Sent | AVG Zenmap Packets Sent | AVG Total Packets Sent |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| No Extra Load | No Extra Load | No Extra Load | 500 | 250 | 55470 | 83.14% | 55409 | 83.04% | 55473 | 83.14% | 60100 | 6622 | 66722 |
| 30% | 30% | 30% | 500 | 250 | 52705 | 79.92% | 52635 | 79.81% | 52707 | 79.92% | 59320 | 6629 | 65949 |
| 50% | 50% | 50% | 500 | 250 | 46751 | 76.17% | 46689 | 76.06% | 46754 | 76.17% | 54800 | 6581 | 61381 |
| 70% | 70% | 70% | 500 | 250 | 41796 | 70.92% | 41722 | 70.80% | 41795 | 70.92% | 52360 | 6571 | 58931 |
| 90% | 90% | 90% | 500 | 250 | 40775 | 67.71% | 40608 | 67.43% | 40776 | 67.71% | 53600 | 6618 | 60218 |

**Table 8-4. Average packets captured at various CPU Utilizations in Xmas tree scans**

The above table displays the average percentage of packets captured at various CPU utilizations during Xmas tree scans. The average was taken from a total of 25 experiments (5 each at a given CPU utilization). The data are for 2 Wireshark sensors and one Snort sensor sniffing the network in promiscuous mode.

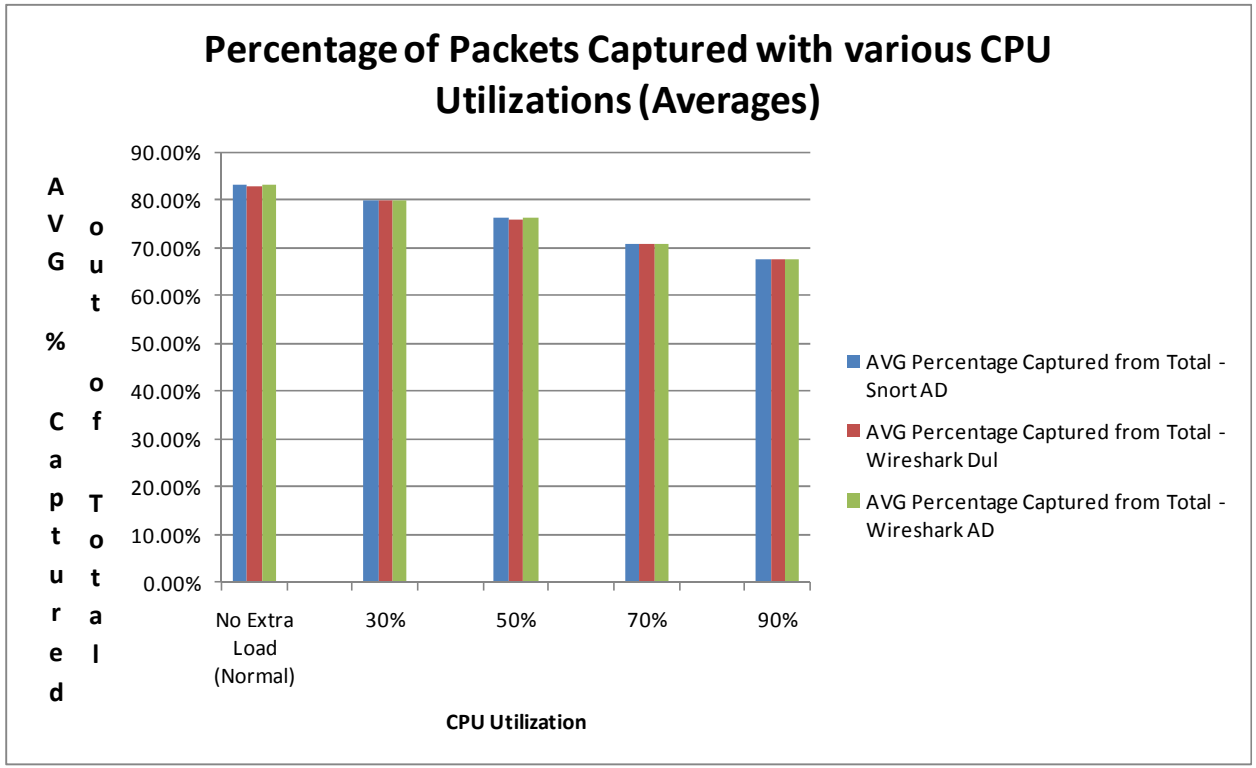


**Figure 8-4. Average Percentage of packets captured with various CPU Utilizations in 25 Xmas tree scans**

| Snort AD, AVG % Packet Difference from First | Relative Reliability of Snort AD | Wireshark Dul, AVG % of Packet Difference from First | Relative Reliability of Wireshark Dul | Wireshark AD, AVG % of Packet Difference from First | Relative Reliability of Wireshark AD |
|---|---|---|---|---|---|
| | | | | | |
| 0.00% | 100% | 0.00% | 100% | 0.00% | 100% |
| | | | | | |
| -3.22% | 96.78% | -3.23% | 96.77% | -3.22% | 96.78% |
| | | | | | |
| -6.97% | 93.03% | -6.98% | 93.02% | -6.97% | 93.03% |
| | | | | | |
| -12.21% | 87.79% | -12.25% | 87.75% | -12.22% | 87.78% |
| | | | | | |
| -15.42% | 84.58% | -15.61% | 84.39% | -15.43% | 84.57% |
| | | | | | |

**Table 8-5. Relative reliabilities with varying CPU Utilizations in Xmas tree scans**

Instead of assigning absolute reliability values to the sensors, we calculated them on a relative basis. Therefore, at the start without any CPU loading software, the sensors were given a total (100%) reliability. As the experiment progressed, different readings were taken at varying CPU loads. First, when there is no extra load on the CPU, we calculated the captured packet percentage by

- Captured Packet Percentage = Number of captured packets / (Number of sent packets generated by Commview + packets, sent by Zenmap)

Notice, that in this calculation we add both Commview packets and Zenmap packets to the total, unlike last time where we only added Commview packets. We do this procedure for every experiment at varying CPU loads. Then, at each experiment we calculate the percentage difference from the beginning of the experiment. Through this procedure we calculate the relative reliability of a sensor at a given CPU load during Xmas tree attacks.

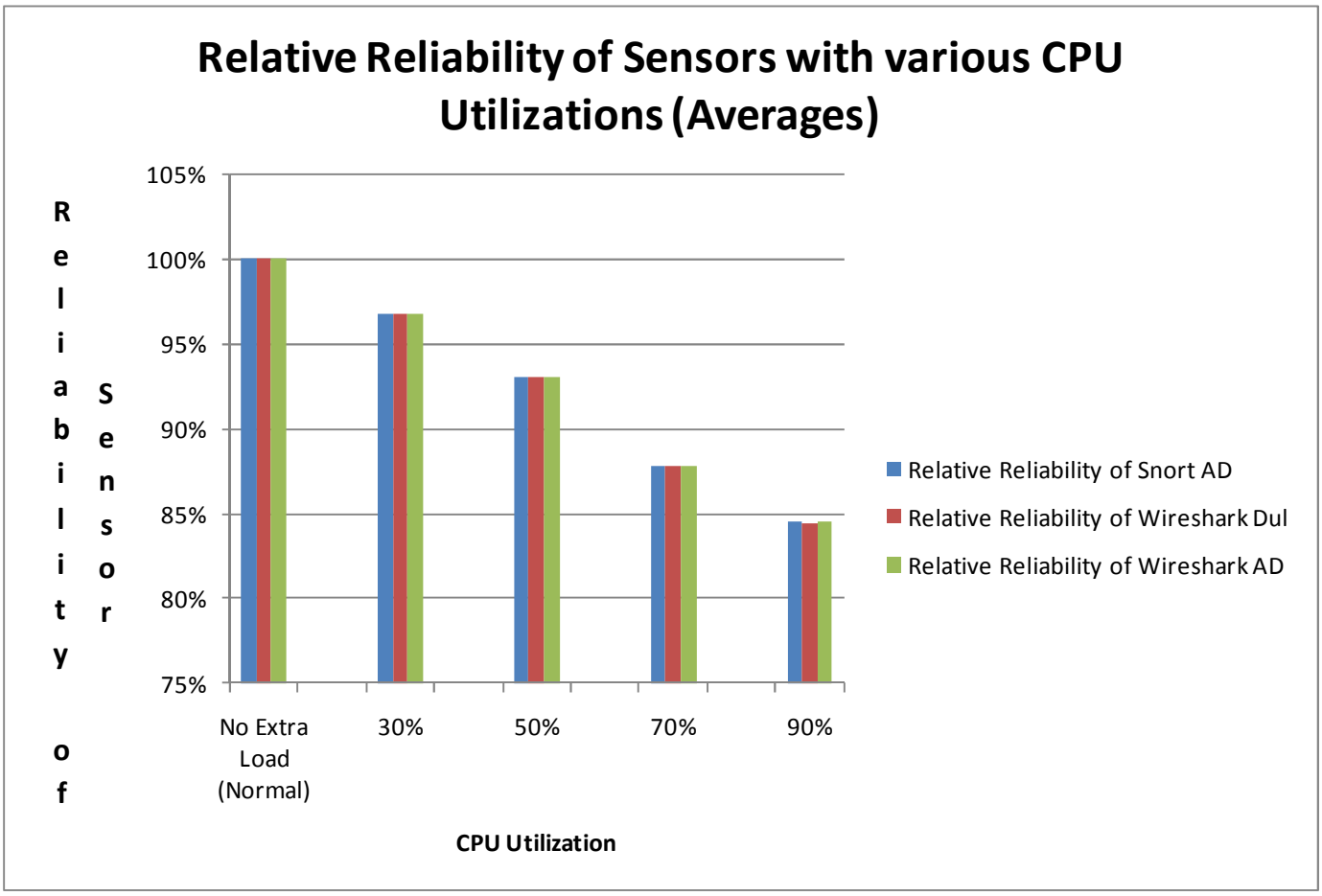


**Figure 8-5. Average Relative Reliability of Sensors with Various CPU Utilizations in 25 Xmas Tree Scans**

The experimental results clearly showed that a sensor captures fewer packets when the CPU load is increased during a Xmas tree attack. But, the actual percentage of packets captured during Xmas tree attacks were greater than the percentage of packets captured at the same CPU load in DoS attacks. The reason being that we only send packets at 500 packets per second during the Xmas tree attack while during the DoS attack we sent packets at 5000 packets per second. However the relative reliability of sensors during Xmas tree attacks decreased compared with DoS attacks. A possible reason for this could be because our Xmas tree scans lasted longer than DoS attacks.

## 8.3 DoS Attack Experiments and Analysis

The experimental setup for DoS attacks consisted of 2 AP-70 sensors, 2 Wireshark sensors, 1 Snort sensor, an attacker machine and a victim machine. The attacker flooded the victim with

high speed packet streams for a continuous period using 5000 packets per second (maximum allowed by CommView) packet streams using CommView. The Experimental setup is given below.

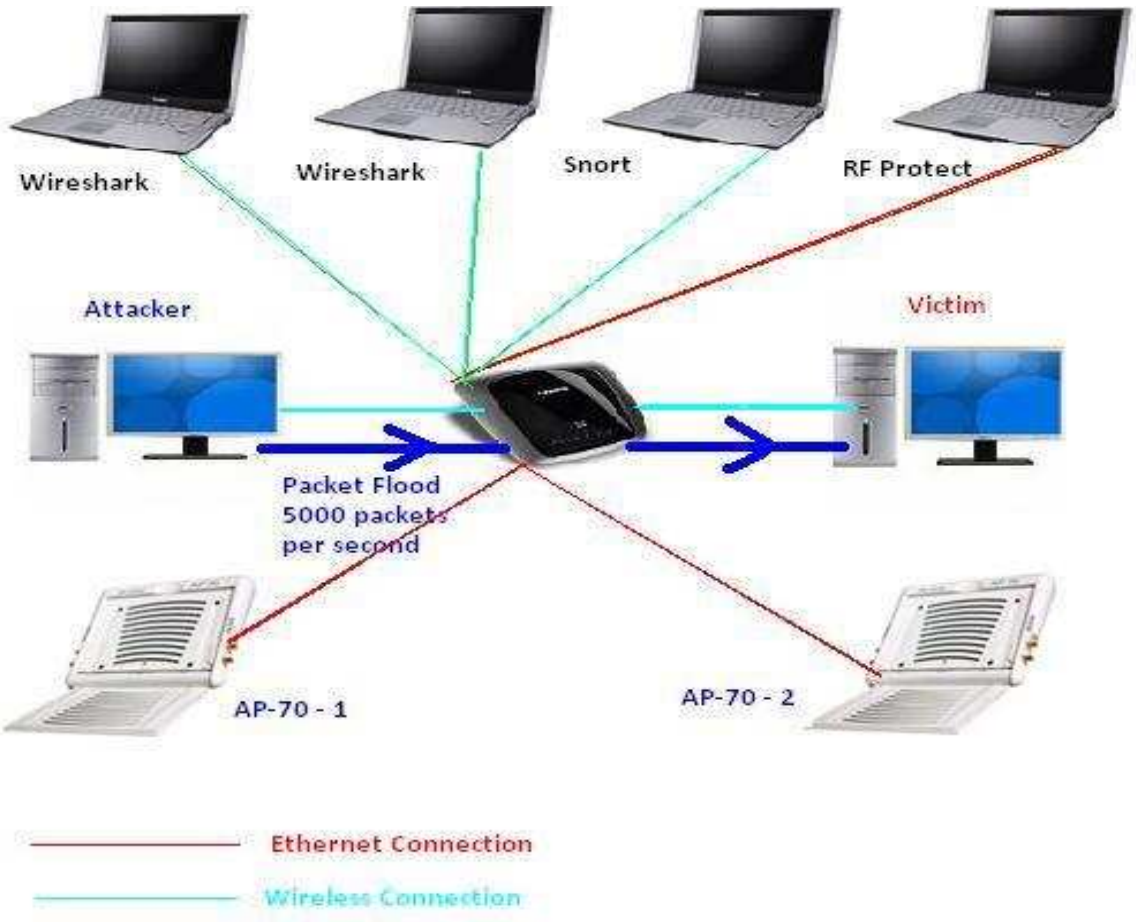


**Figure 8-6. DoS Attack Setup**

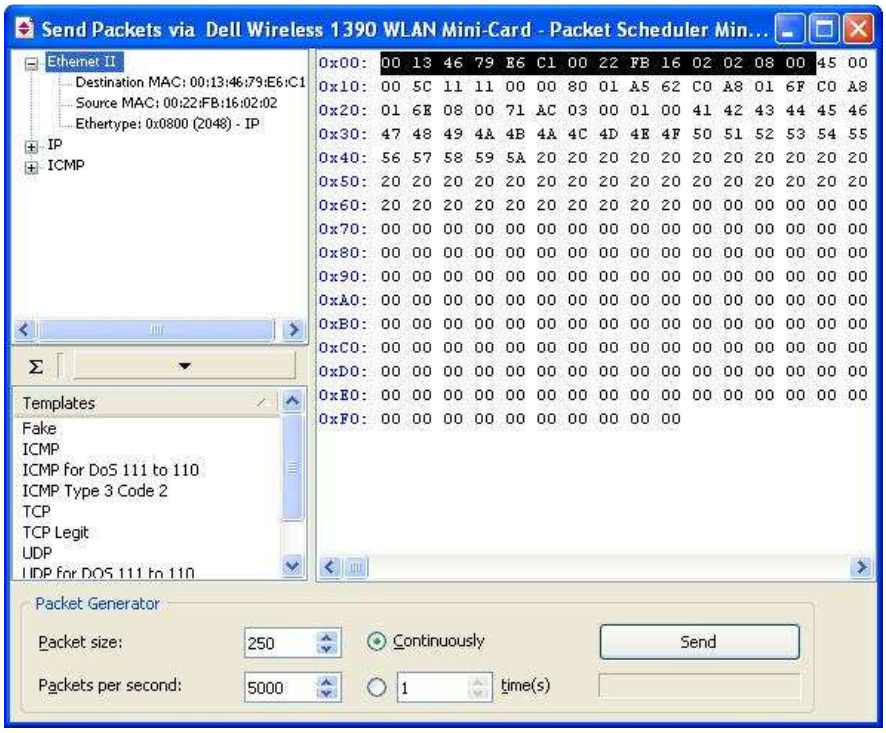


**Figure 8-7. CommView Software Configured to Flood Packets at 5000 Packets per Second**

30 Denial of Service (DoS) attacks were conducted at CPU utilizations, (normal, 30%, 50%, 70%, and 90%). After each attack, evidence was gathered. This evidence was used for basic probability assignment and evidence fusion. We used 4 different methods of evidence combination. Namely they were,

1. PERF D-S (Our method)
2. Consensus (by Dr. Audun Jøsang)
3. Yu and Frincke's method
4. Dempster-Shafer Theory

After gathering data and calculating basic probability assignments, they were fused as described earlier. First we fuse two sensors and use the result as one from a single sensor and fuse that with the next sensor. We get the final combination result after fusing the 5 sensors. The results at each reliability level have been calculated. Since 4 methods of fusion are used, we compared them. Given below are the results of the experiments. The detailed calculations of fusion and basic probability mass calculation are given in the technical report 09-027, School of Computer Science, University of Windsor.

| Comparative Results for $m_{12}$(DoS) | | | | |
|---|---|---|---|---|
| CPU Load | PERF | Consensus | Yu and Frincke | Regular D-S |
| | | | | |
| Normal | 96.65% | 76.00% | 96.65% | 96.65% |
| 30% | 95.39% | 75.86% | 100.69% | 97.12% |
| 50% | 93.14% | 73.08% | 103.26% | 96.19% |
| 70% | 89.26% | 71.43% | 110.85% | 96.06% |
| 90% | 83.96% | 74.07% | 135.23% | 96.60% |
| | | | | |
| Normal | 97.43% | 78.57% | 97.43% | 97.43% |
| 30% | 94.01% | 73.08% | 99.38% | 95.74% |
| 50% | 93.52% | 73.33% | 104.49% | 96.91% |
| 70% | 92.01% | 76.00% | 108.91% | 97.38% |
| 90% | 81.30% | 70.37% | 138.04% | 94.89% |
| | | | | |
| Normal | 95.50% | 73.91% | 95.50% | 95.50% |
| 30% | 95.81% | 78.26% | 99.93% | 97.42% |
| 50% | 93.90% | 75.00% | 104.99% | 97.29% |
| 70% | 87.79% | 69.57% | 105.73% | 93.27% |
| 90% | 81.30% | 70.37% | 138.04% | 94.89% |
| | | | | |
| Normal | 98.54% | 79.31% | 98.54% | 98.54% |
| 30% | 92.70% | 73.91% | 97.84% | 94.36% |
| 50% | 91.72% | 70.97% | 102.73% | 94.82% |
| 70% | 88.68% | 70.37% | 110.59% | 95.43% |
| 90% | 81.91% | 71.43% | 137.06% | 95.59% |
| | | | | |
| Normal | 96.64% | 78.26% | 96.64% | 96.64% |
| 30% | 94.45% | 73.33% | 100.02% | 96.20% |
| 50% | 91.68% | 71.43% | 102.11% | 94.75% |
| 70% | 88.85% | 70.37% | 106.99% | 94.20% |
| 90% | 79.41% | 69.57% | 131.67% | 92.49% |
| | | | | |
| Normal | 99.13% | 84.62% | 99.13% | 99.13% |
| 30% | 92.68% | 73.91% | 97.49% | 94.36% |
| 50% | 92.13% | 72.00% | 102.61% | 95.42% |
| 70% | 90.18% | 73.08% | 109.52% | 96.91% |
| 90% | 84.12% | 78.26% | 133.00% | 97.66% |
| | | | | |

**Table 8-6. Comparative Results of the 4 Combination Methods in 30 DoS Attacks**

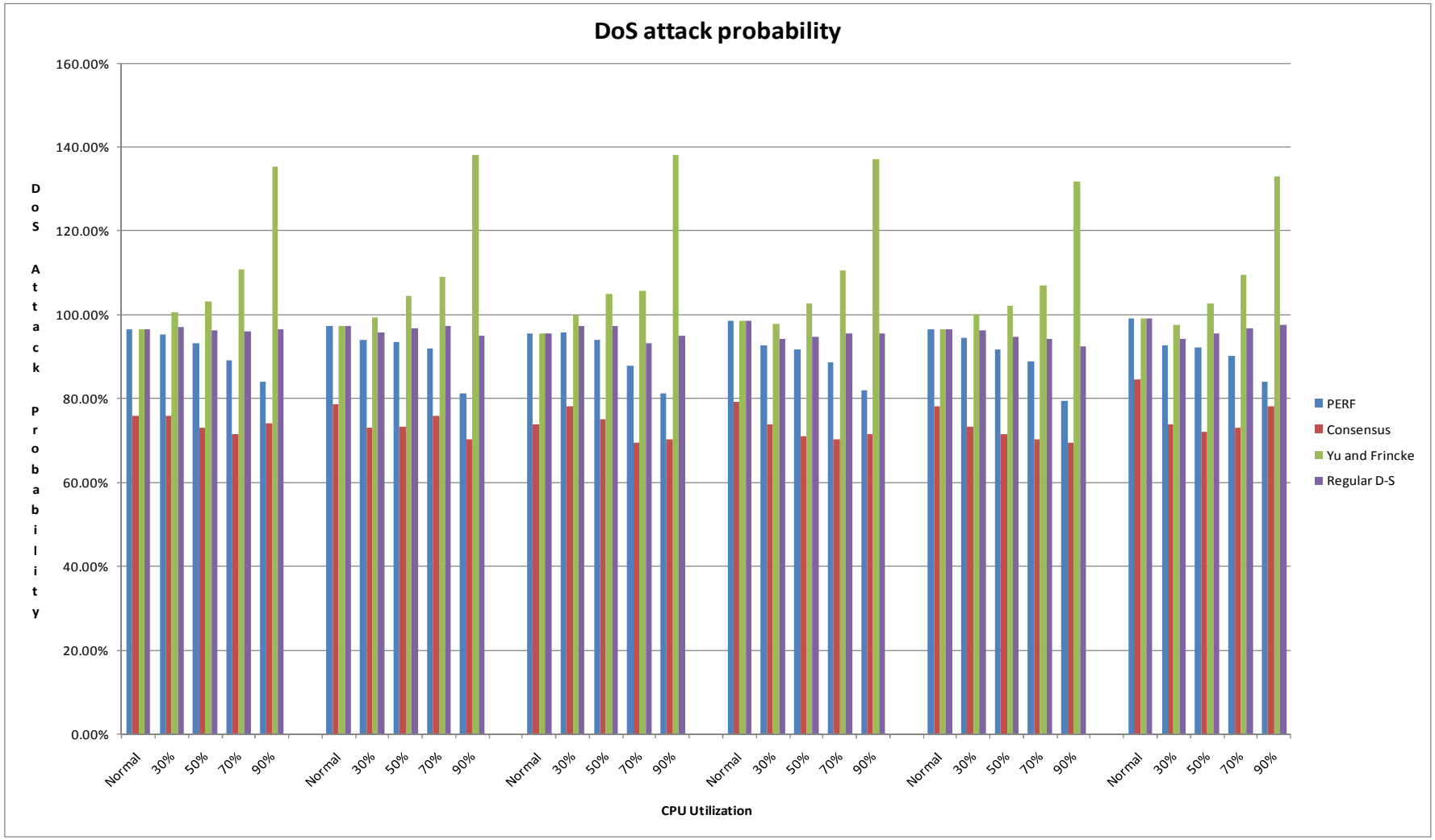


**Figure 8-8. DoS Attack Probability After Combination for 30 DoS Attacks**

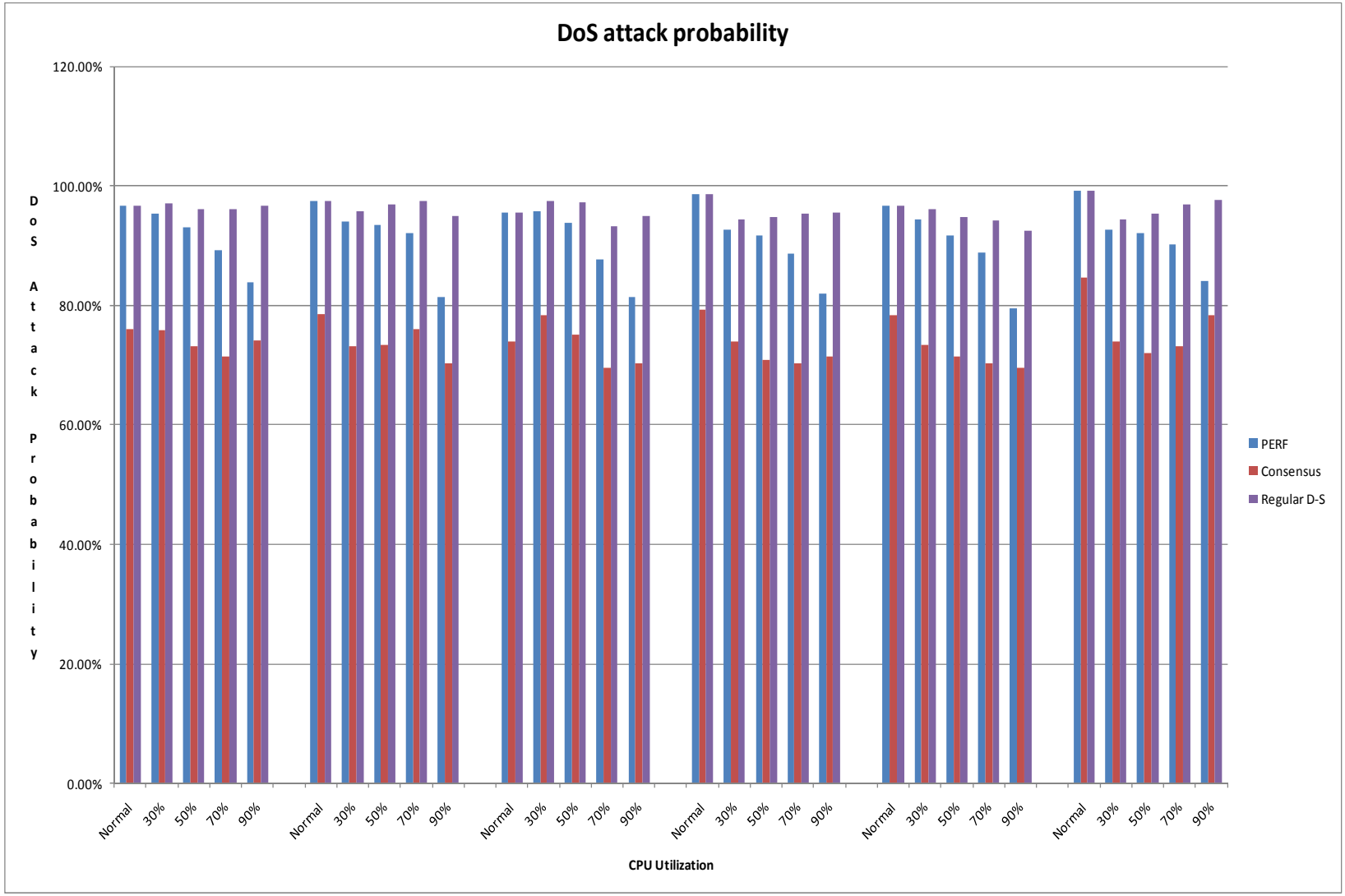


**Figure 8-9. DoS Attack Probability After Combination Excluding Yu and Frincke's method for 30 DoS Attacks**

It was quite evident from the results that Yu and Frincke's exponential combination rule did not fare well with 5 sensors. After 2 combinations, Yu and Frincke's exponential method started giving probability values exceeding 100% for the attack state. Yu and Frincke's method seemed to have a maximizing effect of the actual situation.

As for Consensus method, it seemed to provide reasonable results with the given evidence. In our DoS attack scenario since we decreased the reliability of the sensors gradually there seemed to be a gradual decrease of evidence gathered, that affected the outcome of the final result. Especially, in a DoS attack scenario we paid special attention to the number of packets received at various CPU Utilization levels. When the CPU utilization is increased we receive slightly less number of packets as compared with the case of lower CPU utilization stage. Hence we detected a slight decrease in the combination results in successive stages of the experiment. The Consensus method does not have a way to account for the decrease in reliability of sensors. Therefore, the decrease in Consensus was small compared to that in PERF.

Dempster-Shafer also lacked the capability to take into account the reliability of sensors. Hence the values of fusion, obtained by using D-S method, provided slightly higher estimates for the attack states compared with the values obtained through Consensus and PERF. D-S combination showed a slight decrease but the decrease was smaller than that for Consensus and PERF. This is because D-S theory always tries to give high positive results when the evidence is positive (and extremely negative results under negative evidence).

PERF showed results as expected. Thus we found that the feature of considering reliability of sensors in PERF gives better results. Moreover this eliminates the problem encountered by Yu and Frincke's exponential D-S, as explained in section 8.4. All the 3 methods of combination except Yu and Frincke's method provided probability values that are within the valid range of 0% to 100%.

| Comparative Results for $m_{12}$(DoS) | | | | |
|---|---|---|---|---|
| CPU Load | AVG PERF | AVG Consensus | AVG Yu and Frincke | AVG Regular D-S |
| | | | | |
| Normal | 97.31% | 78.45% | 97.31% | 97.31% |
| 30% | 94.17% | 74.73% | 99.22% | 95.87% |
| 50% | 92.68% | 72.63% | 103.37% | 95.90% |
| 70% | 89.46% | 71.80% | 108.76% | 95.54% |
| 90% | 82.00% | 72.34% | 135.51% | 95.35% |
| | | | | |

**Table 8-7. Average Results from the Combination of Evidence from the 30 DoS attacks**

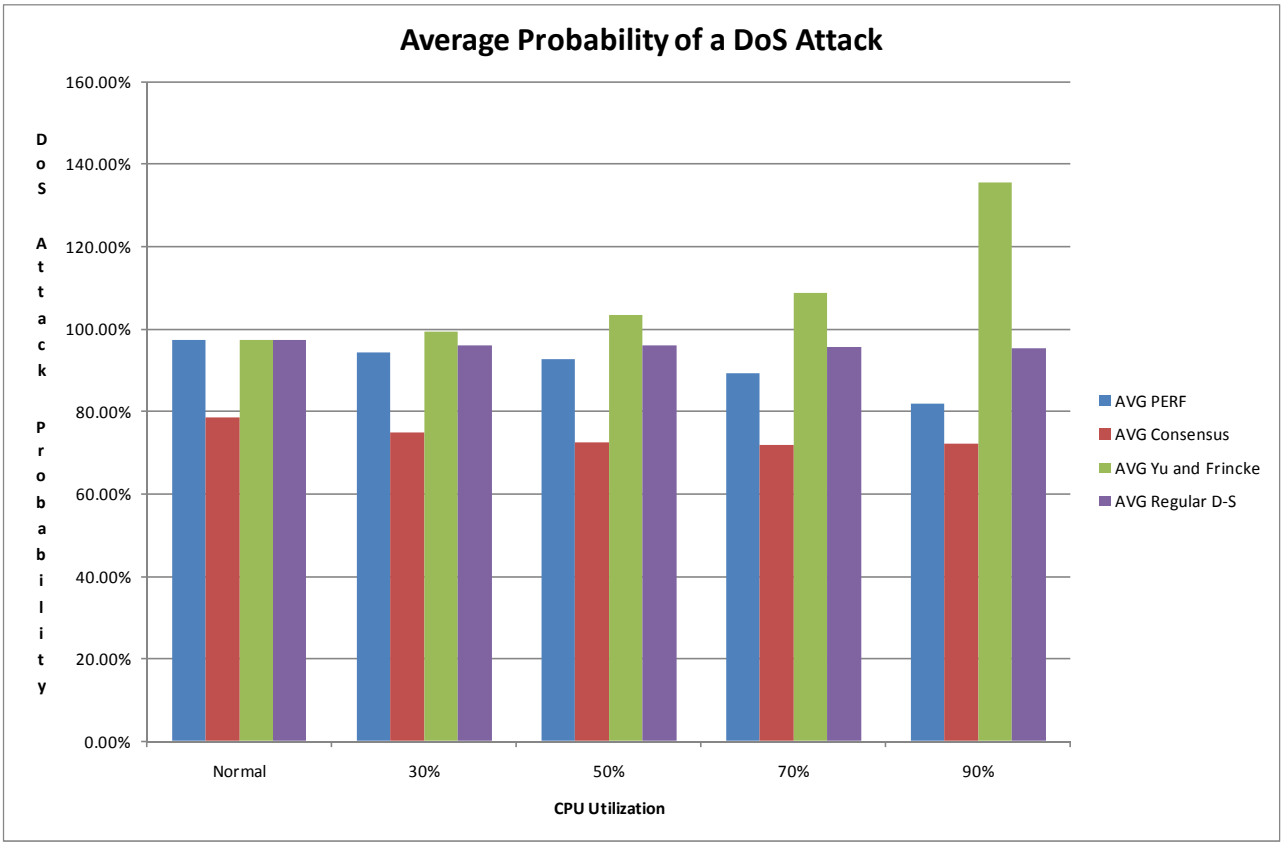


**Figure 8-10. Average Probability of 30 DoS attacks at various CPU Utilizations**

As can be seen from the above graphs, it is clear that all the methods except Yu and Frincke's method, on an average, had a decreasing value of probability of attack as the CPU load was increased. PERF had a slightly bigger decrease due to its consideration of reliability of evidence in fusion. Also, unlike the other methods, when the reliability of sensors decreased, the uncertainty in the combined state or $m_{12}$(U) increased. This is clearly shown from the following data.

| Comparative Results for $m_{12}(U)$ in DoS Attacks | | | | |
|---|---|---|---|---|
| CPU Load | PERF D-S | Consensus | Yu and Frincke | Regular D-S |
| | | | | |
| Normal | 0.66% | 8.00% | 0.66% | 0.66% |
| 30% | 1.92% | 6.90% | 0.40% | 0.38% |
| 50% | 3.33% | 7.69% | 0.63% | 0.56% |
| 70% | 5.61% | 7.14% | 0.53% | 0.44% |
| 90% | 11.95% | 7.41% | 0.80% | 0.50% |
| | | | | |
| Normal | 0.51% | 7.14% | 0.51% | 0.51% |
| 30% | 2.13% | 7.69% | 0.59% | 0.56% |
| 50% | 3.29% | 6.67% | 0.38% | 0.34% |
| 70% | 5.19% | 8.00% | 0.60% | 0.52% |
| 90% | 12.85% | 7.41% | 0.81% | 0.50% |
| | | | | |
| Normal | 1.00% | 8.70% | 1.00% | 1.00% |
| 30% | 2.25% | 8.70% | 0.80% | 0.76% |
| 50% | 3.32% | 7.14% | 0.44% | 0.40% |
| 70% | 5.86% | 8.70% | 1.18% | 1.00% |
| 90% | 12.85% | 7.41% | 0.81% | 0.50% |
| | | | | |
| Normal | 0.29% | 6.90% | 0.29% | 0.29% |
| 30% | 2.66% | 8.70% | 1.18% | 1.11% |
| 50% | 3.27% | 6.45% | 0.43% | 0.38% |
| 70% | 5.68% | 7.41% | 0.61% | 0.51% |
| 90% | 12.72% | 7.14% | 0.70% | 0.44% |
| | | | | |
| Normal | 1.00% | 8.70% | 1.00% | 1.00% |
| 30% | 1.96% | 6.67% | 0.40% | 0.38% |
| 50% | 3.38% | 7.14% | 0.58% | 0.52% |
| 70% | 5.47% | 7.41% | 0.68% | 0.57% |
| 90% | 13.35% | 8.70% | 1.58% | 0.99% |
| | | | | |
| Normal | 0.39% | 7.69% | 0.39% | 0.39% |
| 30% | 2.67% | 8.70% | 1.17% | 1.11% |
| 50% | 3.61% | 8.00% | 0.75% | 0.68% |
| 70% | 5.61% | 7.69% | 0.60% | 0.52% |
| 90% | 12.44% | 8.70% | 1.05% | 0.69% |
| | | | | |

**Table 8-8. Comparative Results for m12(Uncertainty) During 30 DoS Attacks**

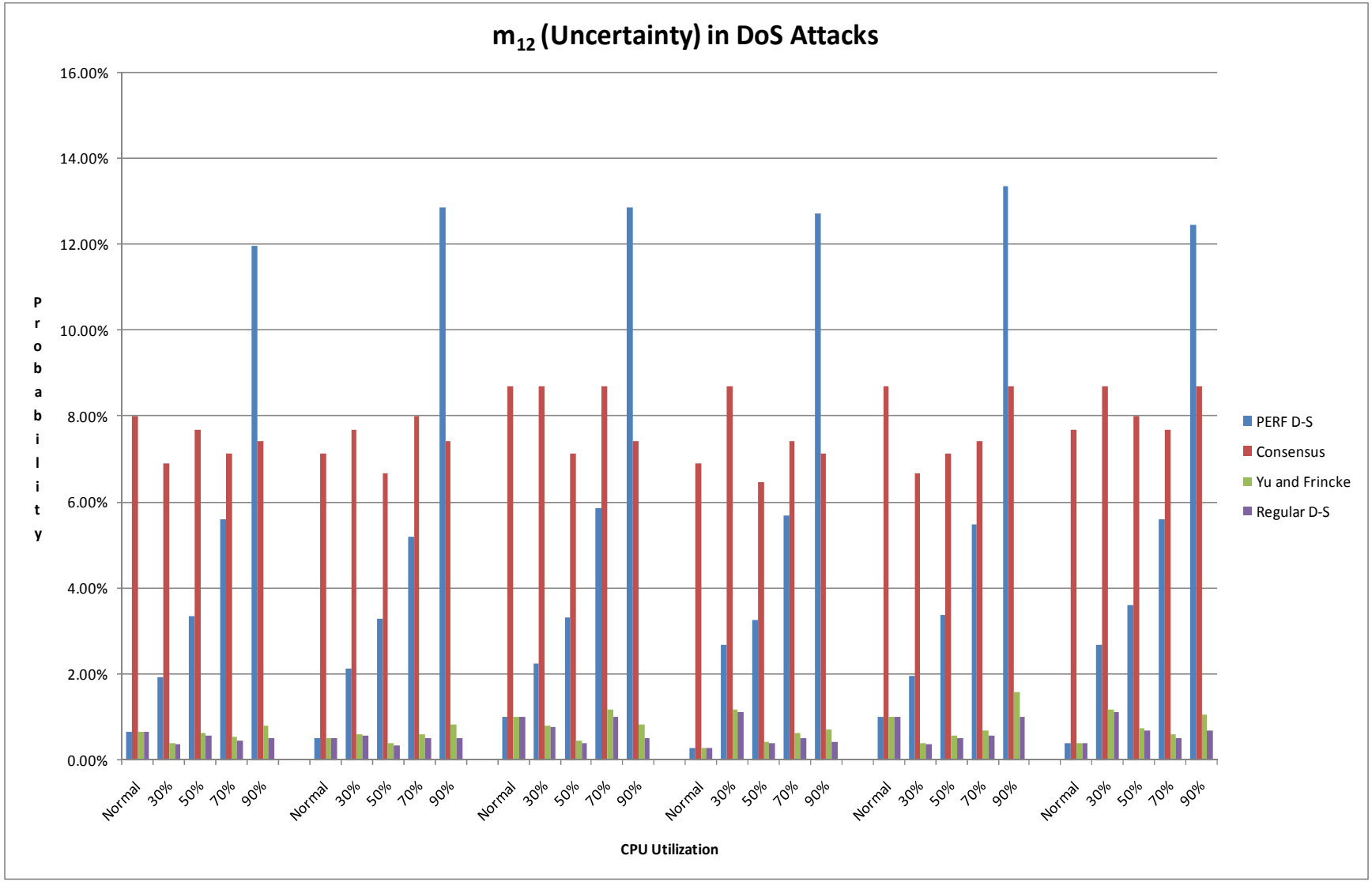


**Figure 8-11. . Combined Uncertainty in 30 DoS Attacks**

| Averages of Combined Uncertainties | | | | |
|---|---|---|---|---|
| CPU Load | AVG PERF | AVG Consensus | AVG Yu and Frincke | AVG Regular D-S |
| | | | | |
| Normal | 0.64% | 7.85% | 0.64% | 0.64% |
| 30% | 2.27% | 7.89% | 0.76% | 0.72% |
| 50% | 3.37% | 7.18% | 0.53% | 0.48% |
| 70% | 5.57% | 7.72% | 0.70% | 0.59% |
| 90% | 12.69% | 7.79% | 0.96% | 0.61% |
| | | | | |

**Table 8-9. Averages of Combined Uncertainties in 30 DoS Attacks**

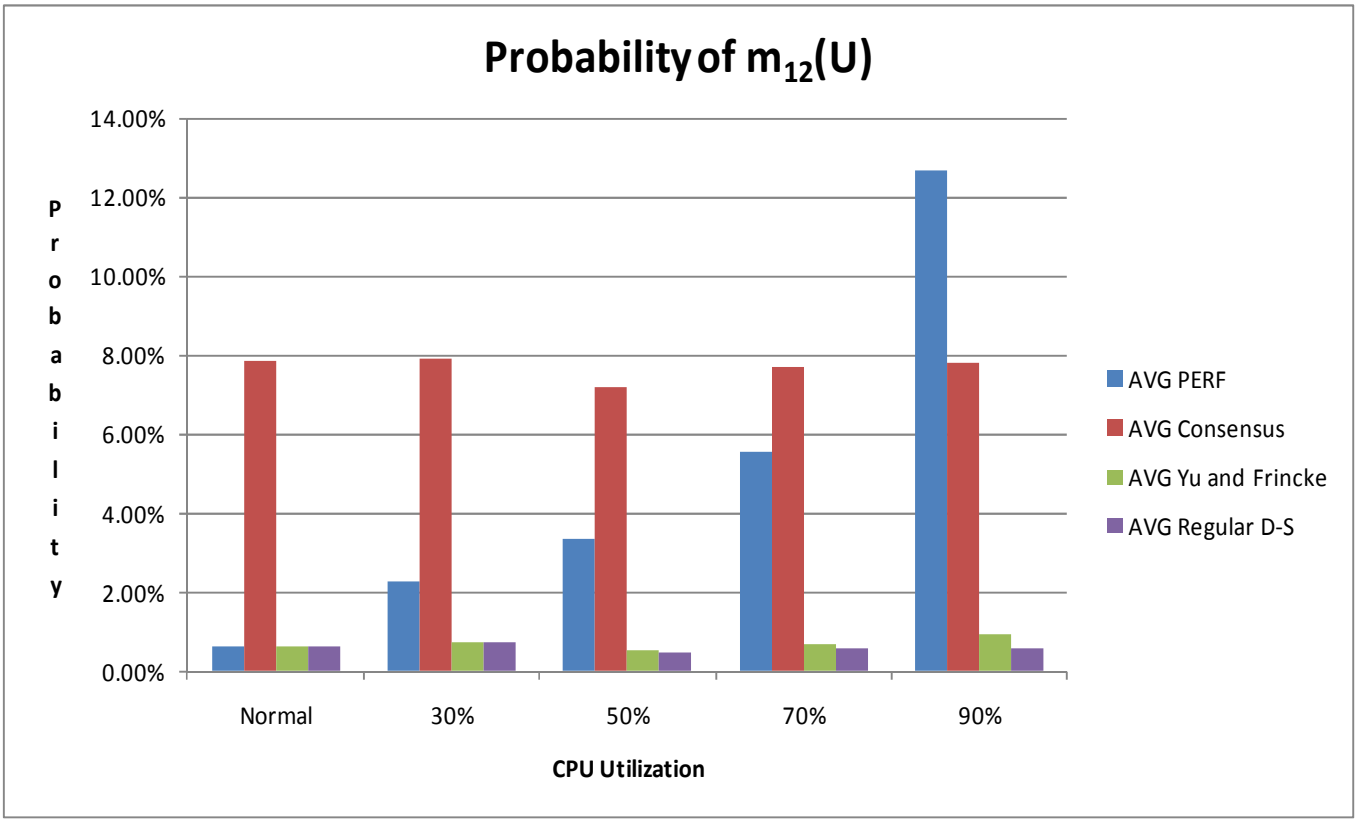


**Figure 8-12. Average Combined Uncertainty with CPU Utilization for 30 DoS Attacks**

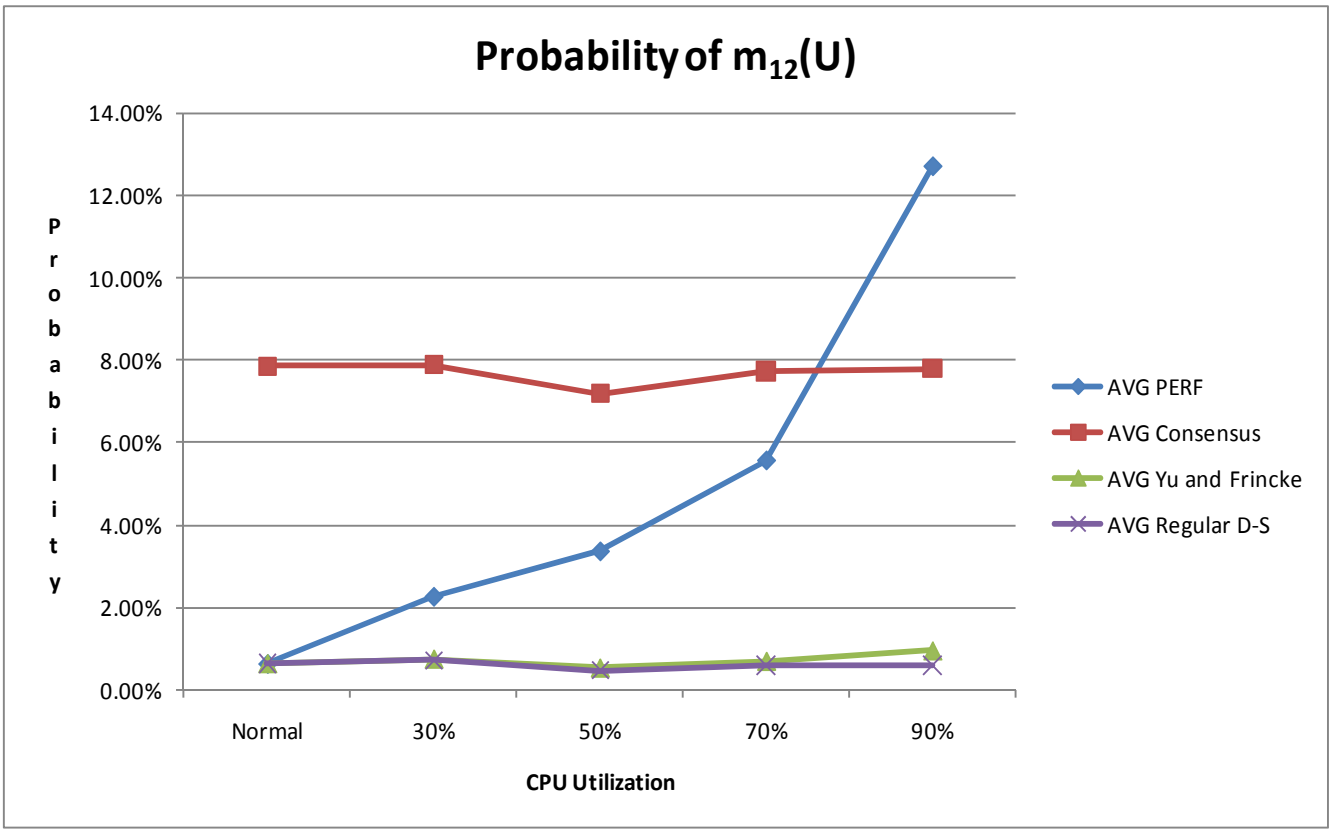


**Figure 8-13. Average Combined Uncertainty with CPU Utilization for 30 DoS Attacks**

Experimental results clearly indicated that when the reliability decreases, the PERF combination assigns more mass to the unknown state. Although, other methods showed decreases of uncertainty for the combined state it was not consistent with the given reliability. Consensus and Dempster-Shafer do not have any way of handling reliability. The results given by PERF are fairer because when the reliability of evidence decreases, the uncertainty of evidence should increase. This in turn results in having a higher value for the combined uncertainty of the sensors. No other method successfully handled uncertainty with decreasing reliability.

## 8.4 Xmas Tree Scan Experiments and Analysis

The experimental setup for Xmas tree scans were similar to that of DoS attacks. The Xmas scans were conducted using Zenmap (graphical user interface of Nmap). Also, packets were sent to the victim at 500 packets per second using CommView. If we conducted Xmas scans without background traffic, it would be too easy to detect. Further, these extra packets would mean that the network sensors have more work to do and catch the normal packets and the Xmas packets. This decreases the chance of capturing the Xmas packets. Moreover this normal packet stream

would act as negative evidence, indicating some state other that the Xmas attack state in evidence fusion.

A total of 30 Xmas scans were conducted during the experiment with CPU utilizations varying from (normal, 30%, 50%, 70%, and 90%). The gathered evidence was used to calculate the basic probability assignments as before. As in the DoS research we used 4 different methods of evidence combination to fuse the gathered evidence. The Xmas tree scans were conducted as described earlier in the definitions section. The results are compared side by side to see which method performs the best. Given below are the results of the experiments.

| Comparative Results for $m_{12}$(Xmas) | | | | |
|---|---|---|---|---|
| CPU Load | PERF | Consensus | Yu and Frincke | Regular D-S |
| | | | | |
| Normal | 90.39% | 74.05% | 90.39% | 90.39% |
| 30% | 80.47% | 72.48% | 115.66% | 84.97% |
| 50% | 75.59% | 71.43% | 174.72% | 84.92% |
| 70% | 73.40% | 74.81% | 723.44% | 89.46% |
| 90% | 69.39% | 74.24% | -728.75% | 88.68% |
| | | | | |
| Normal | 90.59% | 74.03% | 90.59% | 90.59% |
| 30% | 78.01% | 72.85% | 128.45% | 80.82% |
| 50% | 75.68% | 72.39% | 213.02% | 83.86% |
| 70% | 72.22% | 73.48% | 505.04% | 88.18% |
| 90% | 66.98% | 72.52% | -344.53% | 84.22% |
| | | | | |
| Normal | 89.14% | 73.68% | 89.14% | 89.14% |
| 30% | 85.21% | 74.48% | 114.08% | 90.40% |
| 50% | 79.43% | 73.97% | 164.44% | 89.74% |
| 70% | 70.76% | 73.08% | 4460.71% | 85.26% |
| 90% | 68.45% | 73.95% | -1191.03% | 87.07% |
| | | | | |
| Normal | 93.63% | 74.84% | 93.63% | 93.63% |
| 30% | 87.04% | 74.17% | 110.96% | 92.49% |
| 50% | 77.90% | 73.83% | 193.48% | 87.25% |
| 70% | 71.96% | 73.57% | 519.37% | 87.92% |
| 90% | 69.44% | 74.82% | -879.86% | 89.15% |
| | | | | |
| Normal | 95.49% | 75.00% | 95.49% | 95.49% |
| 30% | 88.28% | 75.00% | 110.85% | 93.79% |
| 50% | 79.93% | 75.34% | 187.37% | 89.64% |
| 70% | 73.83% | 75.18% | 428.74% | 90.24% |
| 90% | 71.56% | 75.91% | 978.27% | 92.33% |
| | | | | |
| Normal | 92.19% | 74.84% | 92.19% | 92.19% |
| 30% | 85.64% | 74.52% | 113.51% | 90.97% |
| 50% | 81.55% | 75.66% | 155.04% | 92.35% |
| 70% | 71.15% | 73.24% | 1206.43% | 86.60% |
| 90% | 66.93% | 72.66% | -385.41% | 84.64% |
| | | | | |

**Table 8-10. Comparative Results of the 4 Combination Methods in 30 Xmas Tree Scans**

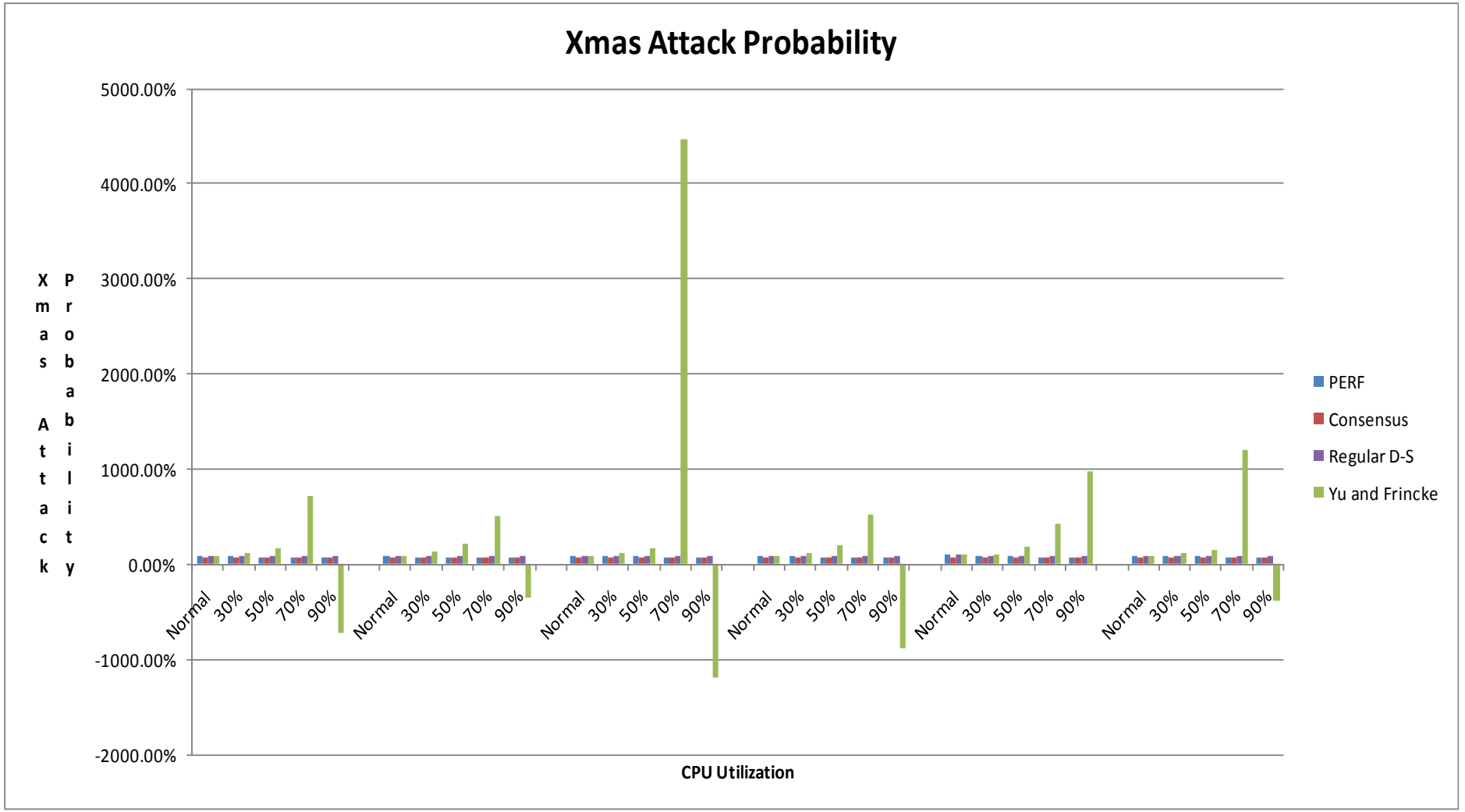


**Figure 8-14. Comparative Results of the 4 Combination Methods in 30 Xmas Tree Scans**

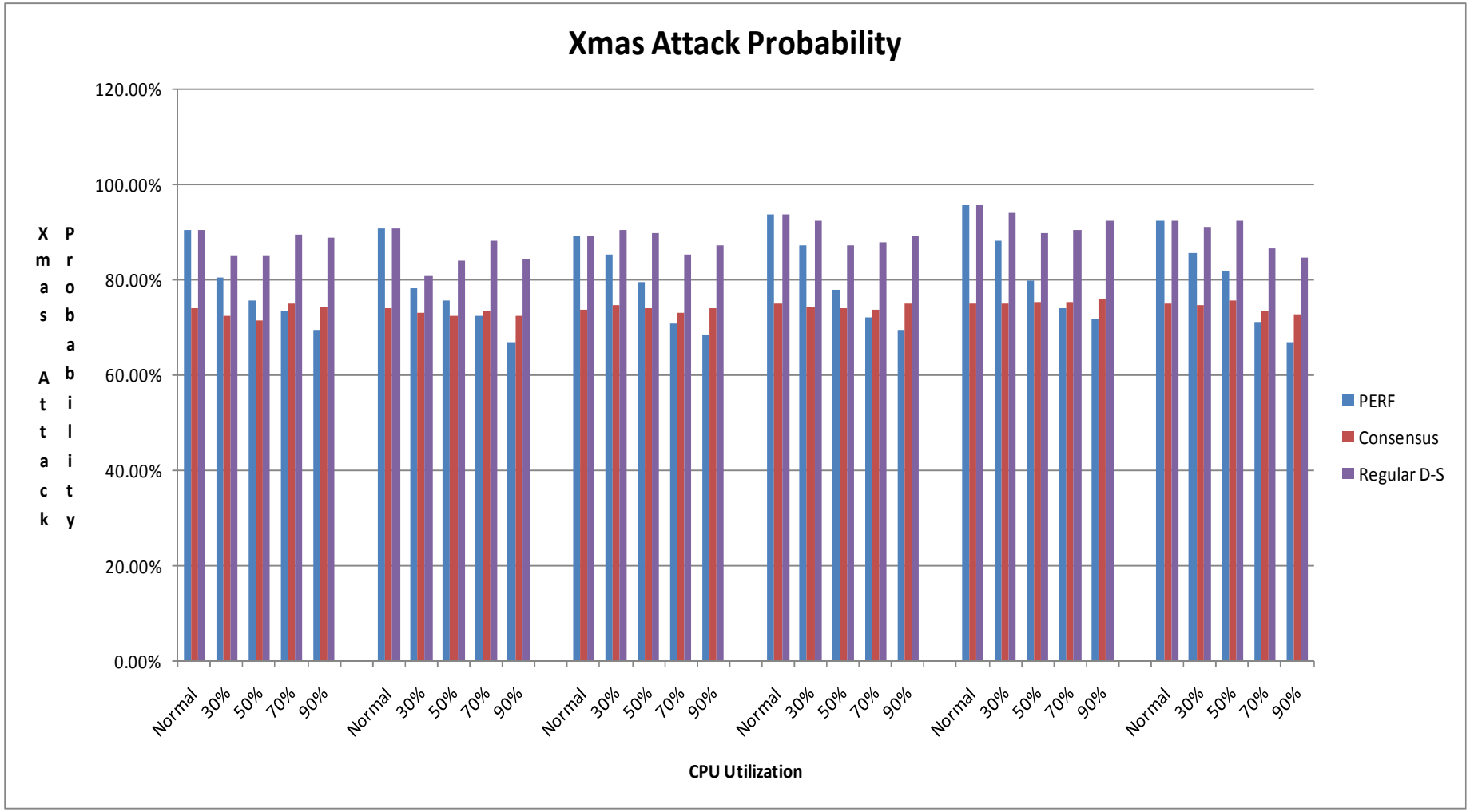


**Figure 8-15. Comparative Results of the Combination Methods in 30 Xmas Tree Scans excluding Yu and Frincke's method**

As in the DoS attack experiments Yu and Frincke's exponential combination rule did not fare well in an environment with 5 sensors. The only stage where it gave results that are in the correct range is when the reliability of sensors were taken as 1 or totally reliable. In all other stages of CPU utilizations (30%, 50%, 70%, 90%) Yu and Frincke's exponential method gave probability values exceeding 100% for the attack state or values below 0% (negative values). In the actual combination Yu and Frincke's method gave values within the correct probability range until the $2^{nd}$ or $3^{rd}$ stage of the fusion (3 or 4 sensors fused). But, the last two stages of fusion always gave results that are not within the valid range of probabilities when reliability was used as an exponent.

As for the Consensus method, it seemed to provide reasonable results with the given evidence. The average results from the combination results indicated that the combined probability values for the attack varied about 2% points during the 5 different CPU utilization levels. The little variation can be explained as follows. During the Xmas tree scan using Zenmap, we also flooded the victim with UDP packets at 500 packets per second. In a Xmas tree scan we considered Xmas tree packets as positive evidence or evidence supporting a Xmas scan state in the network. But, the UDP packets were considered as negative evidence refuting the Xmas state. When the CPU utilization was increased, less number of Xmas packets as well as UDP packets were captured. Or in other words it caused a decrease of both the positive and negative evidence. Hence the results obtained by Consensus method did not change by more than 2% points. Consensus method did not have a way to account for the decrease in reliability in sensors. Therefore, the decreases in Consensus' combined results were small compared to PERF.

Dempster-Shafer fusion provided slightly higher estimates for the attack states compared with Consensus and PERF. This is because D-S theory always tries to give high positive results when the evidence is positive (and extremely negative results under negative evidence). It also lacked the capability to take into account the reliability of sensors. D-S combination showed a decrease for the probability of the attack state when the reliability of sensors decreased. But, this decrease was small, about 1% point for each CPU Utilization level.

PERF combination results indicated that the probability of a Xmas attack decreased with increasing CPU utilization. This is what the result should be, as the better results were obtained since PERF had a way to take care of reliability of sensors in a multi sensor environment. Moreover it did not encounter problems faced by Yu and Frincke's exponential D-S. Also, unlike the other methods, when the reliability of sensors decreased, the uncertainty in the combined state or $m_{12}(U)$ increased. This is clearly shown from the following data.

| Comparative Results for $m_{12}(U)$ in Xmas Tree Scans | | | | |
|---|---|---|---|---|
| CPU Load | PERF D-S | Consensus | Yu and Frinck | Regular D-S |
| | | | | |
| Normal | 0.01% | 1.27% | 0.01% | 0.01% |
| 30% | 5.42% | 1.34% | 0.03% | 0.01% |
| 50% | 10.88% | 1.50% | 0.08% | 0.02% |
| 70% | 16.98% | 1.53% | 0.40% | 0.02% |
| 90% | 20.97% | 1.52% | -0.51% | 0.02% |
| | | | | |
| Normal | 0.01% | 1.30% | 0.01% | 0.01% |
| 30% | 5.73% | 1.32% | 0.03% | 0.01% |
| 50% | 10.99% | 1.49% | 0.09% | 0.02% |
| 70% | 17.17% | 1.52% | 0.32% | 0.02% |
| 90% | 21.49% | 1.53% | -0.29% | 0.02% |
| | | | | |
| Normal | 0.01% | 1.32% | 0.01% | 0.01% |
| 30% | 5.03% | 1.38% | 0.03% | 0.02% |
| 50% | 10.33% | 1.37% | 0.05% | 0.02% |
| 70% | 17.49% | 1.54% | 2.94% | 0.02% |
| 90% | 21.21% | 1.68% | -1.38% | 0.03% |
| | | | | |
| Normal | 0.01% | 1.26% | 0.01% | 0.01% |
| 30% | 4.84% | 1.32% | 0.02% | 0.01% |
| 50% | 10.59% | 1.34% | 0.06% | 0.01% |
| 70% | 17.18% | 1.43% | 0.29% | 0.02% |
| 90% | 20.90% | 1.44% | -0.55% | 0.02% |
| | | | | |
| Normal | 0.01% | 1.28% | 0.01% | 0.01% |
| 30% | 4.73% | 1.28% | 0.02% | 0.01% |
| 50% | 10.32% | 1.37% | 0.05% | 0.01% |
| 70% | 16.84% | 1.46% | 0.24% | 0.02% |
| 90% | 20.40% | 1.46% | 0.62% | 0.02% |
| | | | | |
| Normal | 0.01% | 1.26% | 0.01% | 0.01% |
| 30% | 4.97% | 1.27% | 0.02% | 0.01% |
| 50% | 9.99% | 1.32% | 0.04% | 0.01% |
| 70% | 17.34% | 1.41% | 0.59% | 0.02% |
| 90% | 21.48% | 1.56% | -0.34% | 0.02% |
| | | | | |

**Table 8-11. Comparative Results for $m_{12}(U)$ During 30 Xmas Tree Scans**

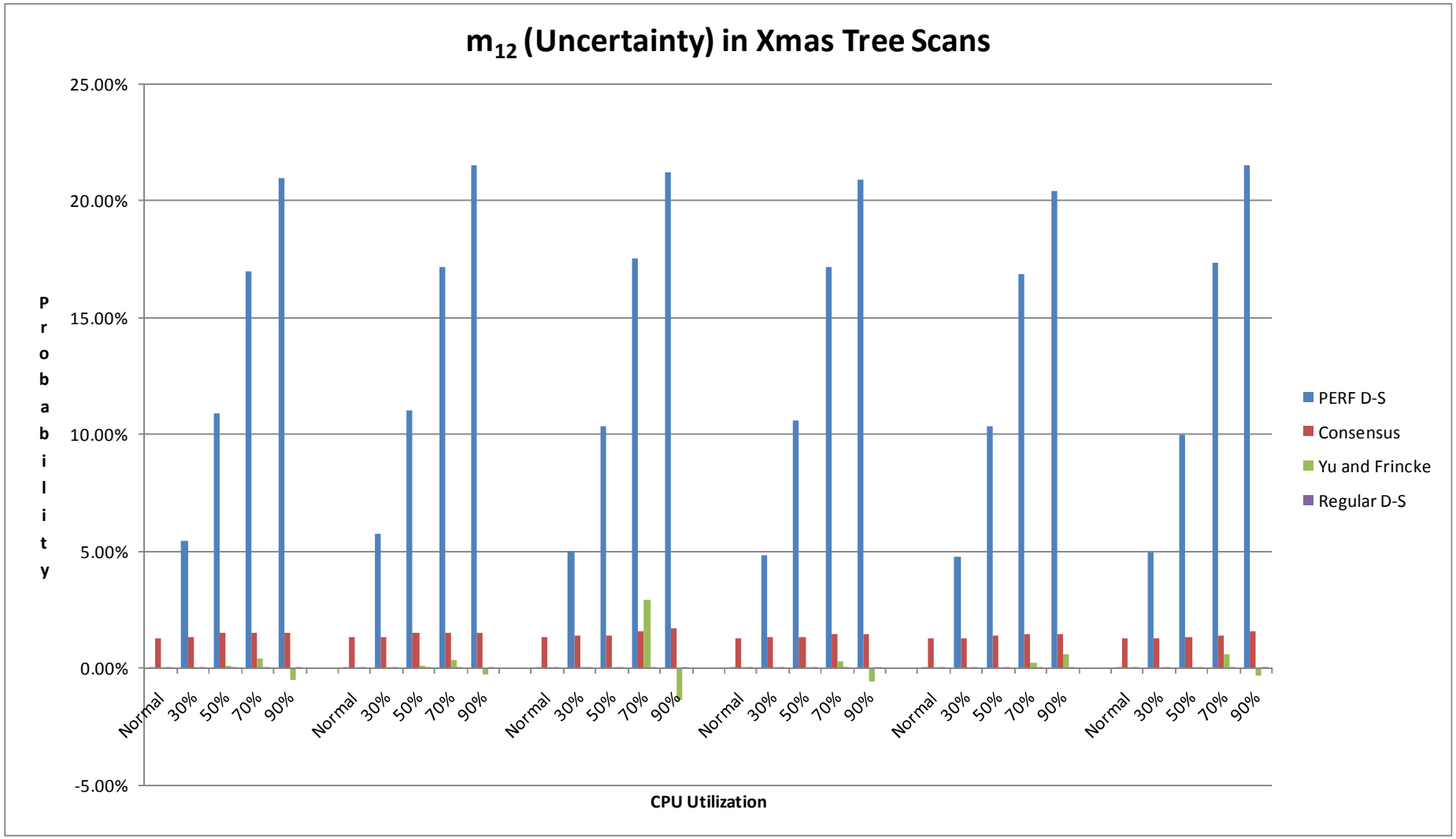


**Figure 8-16. - Combined Uncertainty in 30 Xmas Tree Scans**

| Averages of Combined Uncertainties | | | | |
|---|---|---|---|---|
| CPU Load | AVG PERF | AVG Consensu | AVG Yu and Fri | AVG Regular D-S |
| | | | | |
| Normal | 0.01% | 1.28% | 0.01% | 0.01% |
| 30% | 5.12% | 1.32% | 0.02% | 0.01% |
| 50% | 10.52% | 1.40% | 0.06% | 0.02% |
| 70% | 17.17% | 1.48% | 0.80% | 0.02% |
| 90% | 21.08% | 1.53% | -0.41% | 0.02% |
| | | | | |

**Table 8-12. Averages of Combined Uncertainties in 30 Xmas Tree Scans**

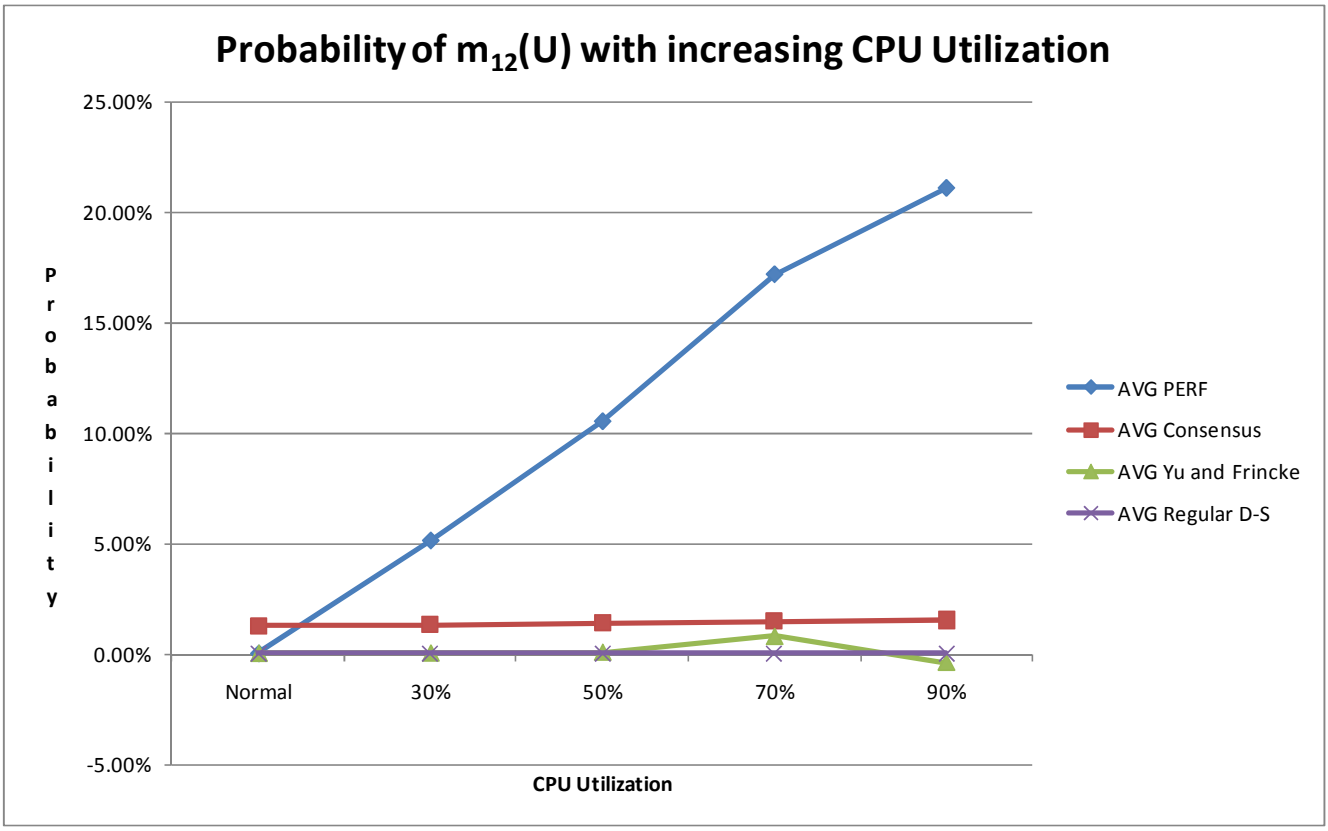


**Figure 8-17. Average Combined Uncertainty with CPU Utilization for 30 Xmas Tree Scans**

This is what should happen in reality. When the reliability of sensors decreases, simply decreasing the attack state and increasing the not attack state is not enough. Instead PERF after calculating the attack state, increases the combined uncertainty that was caused by unreliable data. This makes sense because by providing us with unreliable data, the sensors increase the data related to uncertainty.

All the 3 methods of combination, except Yu and Frincke's method, provided probability values that are within the valid range of 0% to 100% for the $m_{12}$(Xmas) state. Also, the following data clearly indicates that the PERF combination worked out a lower probability of attack state when reliability of sensors decreased.

| Comparative Result for $m_{12}$(Xmas) | | | | |
|---|---|---|---|---|
| CPU Load | AVG PERF | AVG Consensus | AVG Yu and Frincke | AVG Regular D-S |
| | | | | |
| Normal | 91.90% | 74.41% | 91.90% | 91.90% |
| 30% | 84.11% | 73.92% | 115.59% | 88.91% |
| 50% | 78.34% | 73.77% | 181.35% | 87.96% |
| 70% | 72.22% | 73.89% | 1307.29% | 87.94% |
| 90% | 68.79% | 74.02% | -425.22% | 87.68% |
| | | | | |

**Table 8-13. Average Probabilities for $m_{12}$(Xmas) from the Combination of Evidence from 30 Xmas Tree Scans**

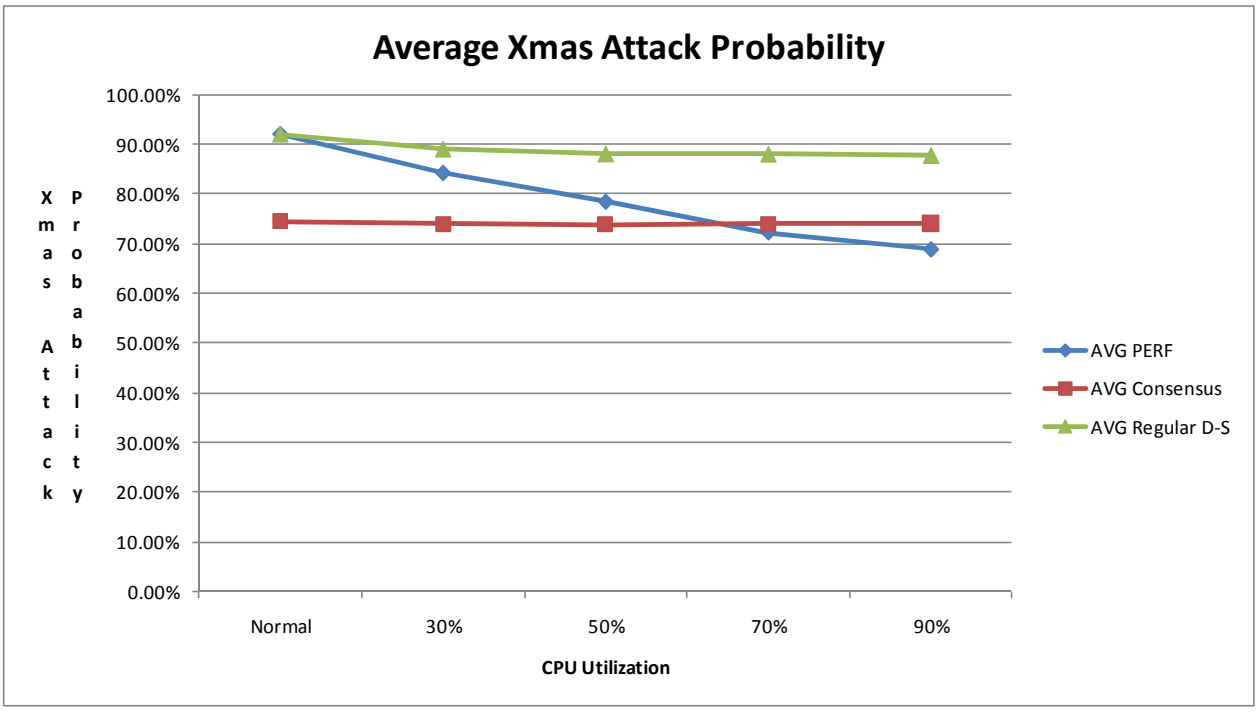


**Figure 8-18. Average Probability of 30 Xmas Attacks at Various CPU Utilizations Excluding Yu and Frincke**

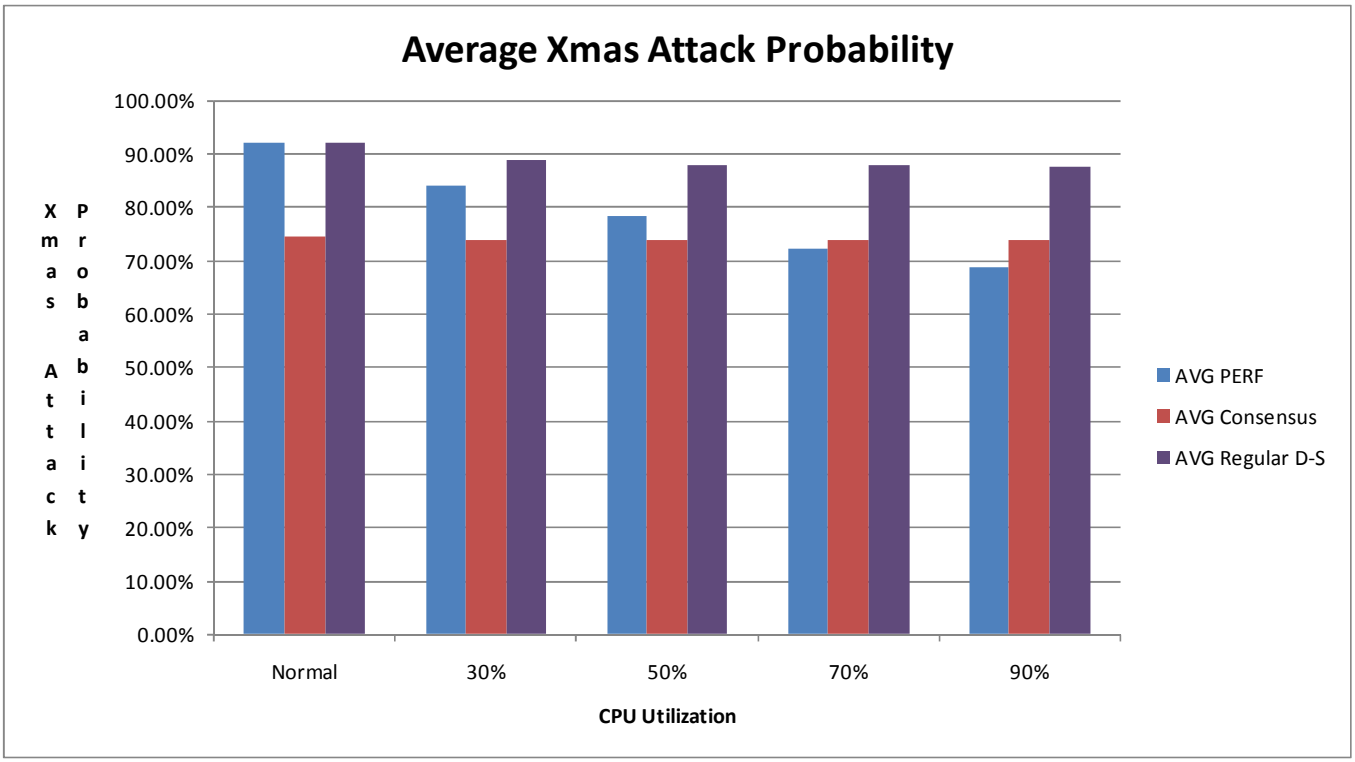


**Figure 8-19. Average Probability of 30 Xmas attacks at various CPU Utilizations Excluding Yu and Frincke**

# 9 CONCLUSION

In this thesis we have introduced a new evidence combination method called PERF D-S that takes the reliability of evidence into consideration. Further we adapted a technique used by Jøsang [2000] to calculate the basic probability assignments in a logical way. We conducted over a hundred experiments in a wireless network to mount two different types of attacks. Intrusion detection in wireless networks and fusing evidence based on the Dempster-Shafer theory, to the best of our knowledge, has not been studied by any other researcher. Moreover other researchers had only utilized a couple of sensors. We have used five sensors, belonging to three different categories. The diversity of the sensors helped to increase the number of attacks that can be covered by intrusion detection in a multi-sensor data fusion environment. By using two attack types, we showed that even when a certain type of sensor fails to detect an attack another sensor will detect it and give an alert, thereby improving the range of attacks covered by the IDS. In a wireless environment, using a diverse set of sensors should increase the accuracy of intrusion detection. Moreover Yu and Frincke's method showed inconsistent results, when the number of sensors increased beyond two.

In an environment where multiple sensors that are different from each other are utilized, the evidence from all sensors should not be treated equally. In our research we introduced reliability of evidence as a factor. In any large network, intrusion detection sensors will have inherent reliabilities, depending on what kind of attacks they can detect and which ones they cannot detect. Every IDS has a particular average value of False Positive value and a False Negative value. Moreover the performance of IDSs and sensors can become less reliable in the presence of high traffic. The reliability factor that we have introduced should be modulated depending on historical performance of the sensor. If we know that a sensor performed at an 80% accuracy level for a certain attack over the previous attack, we can utilize that factor when fusing evidence.

We compared the results of our method with the results of Consensus Operator, the Dempster-Shafer Theory, Yu and Frincke's exponential method. Yu and Frincke's exponential combination rule produced negative probability values and probability values exceeding 100% when more than two sensors were fused. Using reliability as an exponent seemed to be the cause of the problem. Our way of using a progressive reliability factor took care of the problem. Combined with the way of calculating basic probability assignments using Jøsang [2000] method, uncertainty decreases as the amount of evidence increases, if PERF is used. In other words, when PERF is used and when the amount of positive or negative evidence increases, uncertainty decreases.

The results of combining using PERF seemed to be encouraging for the attacks we covered in our research. Since both Dempster-Shafer and Consensus did not have a way of handling reliability, our method was the only one that showed a decrease that is consistent with the given reliability in the combined state of the attacks when reliability was decreased.

The open area of further research is to test PERF's performance under more attack scenarios. Tests of PERF in production wireless networks can provide confidence for using PERF more widely.

# BIBLIOGRAPHY


1. JØSANG, A. 2000. A Logic for uncertain probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, Vol. 9, No. 3 (June 2001)*
2. JØSANG, A. 2002. The Consensus Operator for Combining Beliefs. *Artificial Intelligence Journal* 142(1–2), 157–170.
3. AL-ANI, A., DERICHE. M. 2002. A New Technique for Combining Multiple Classifiers using The Dempster- Shafer Theory of Evidence. *Journal of Artificial Intelligence Research, 17*, 333-361.
4. Aruba AP-70. http://www.arubanetworks.com/products/access-points/ap-70.php.
5. ASLANDOGAN, Y. A., MAHAJANI, G. A., TAYLOR, S. 2004. Evidence Combination in Medical Data Mining. *IEEE International Conference on Information Technology. Coding and Computing*, LasVegas, NV, April 2004, 465 – 469.
6. ASUNCION, A. AND NEWMAN, D.J. 2007. UCI Machine Learning Repository. http://www.ics.uci.edu/~mlearn/MLRepository.html. Irvine, CA, University of California, School of Information and Computer Science.
7. BARLOW, J., THROWER, W. 2000. TFN2K – An analysis. http://packetstormsecurity.com/distributed/TFN2k_Analysis-1.3.txt.
8. BASS, T. 1999. Sensor data fusion for next generation distributed intrusion detection systems. *The proceedings of 1999 IRIS National Symposium on Sensor and Data Fusion*, 24-27.
9. BASS, T., 2000. Intrusion detection systems and multisensory data fusion. *Communications of the ACM*, Vol. 43, No. 4, 99–105.
10. Berger, A., Pietra, S.D., Pietra, V. D. 1996. A Maximum Entropy Approach to Natural Language Processing, Computational Linguistics. 22(1), 39–71.
11. BERK. V., GRAY, R., BAKOS, G. 2003. Using sensor networks and data fusion for early detection of active worms. *Proceeding of 2003 SPIE Aerosense conference*, Orlando, FL.
12. BEZDEK, J. C. 1981. *Pattern recognition with objective function algorithm*s. Plenum Press, New York.
13. Blake, C. L., Merz, C. J. 1998. UCI repository of machine learning databases.
14. BURROUGHS, D. J., WILSON, L. F., Cybenko, G. V. 2002. Analysis of distributed intrusion detection systems using Bayesian methods. *Performance, Computing, and Communications Conference*, 2002, 329 – 334.
15. CHATZIGIANNAKIS, V., ANDROULIDAKIS, G., PELECHRINIS, K., PAPAVASSILIOU, S., MAGLARIS, V. 2007. Data fusion algorithms for network anomaly detection: classification and evaluation. *Proceedings of the Third International Conference on Networking and Services*, 50 - 51.
16. CHEN, T.M., VENKATARAMANAN, V. 2005. Dempster-Shafer theory for intrusion detection in ad hoc networks. *Internet Computing, IEEE*, 9(6), 35 – 41.
17. CHEN, Q., AICKELIN, U. 2006. Dempster-Shafer for Anomaly Detection. *In Proceedings of the International Conference on Data Mining (DMIN 2006)*, Las Vegas, USA, 232-238.

18. CHOU, T. 2007. Ensemble fuzzy belief intrusion detection design. PhD thesis, Florida international university, Florida, USA.
19. CHOU, T., YEN, K.K., PISSINOU, N., MAKKI, K. 2007. Fuzzy Belief Reasoning for Intrusion Detection Design. *Intelligent Information Hiding and Multimedia Signal Processing*, 2, 621–624.
20. CHOU, T., YEN, K. K., LUO, J. 2008. Network intrusion detection design using feature selection of soft computing paradigms. *International Journal of Computational Intelligence*, 4(3).
21. Christmas tree packet. http://en.wikipedia.org/wiki/Christmas_tree_packet
22. CISCO. Netflow. http://www.cisco.com/go/netflow.
23. CommView. http://www.tamos.com/products/commview/
24. Computer port. Wikipedia. http://en.wikipedia.org/wiki/Computer_port_%28software%29
25. DEMPSTER, A. P., 1968. A generalization of Bayesian Inference. *Journal of the Royal Statistical Society*, Series B, 30, 205-247.
26. DENEOUX, T. 1995. A k-nearest neighbor classification rule based on Dempster-Shafer theory. *IEEE transactions on systems, man and cybernetics*, 25(5), 804-813.
27. DENIAL OF SERVICE ATTACK. Wikipedia. http://en.wikipedia.org/wiki/Denial-of-service_attack
28. DITTRICH, D. 1999. The Stacheldraht distributed denial of service attack tool. http://staff.washington.edu/dittrich/misc/stacheldraht.analysis.
29. DUNN, J. C. 1973. A fuzzy relative of the ISODATA process and its use in detecting compact well separated clusters. *Journal of Cybernetics*, 3, 32-57.
30. EXPONENTIATION. Wikipedia. http://en.wikipedia.org/wiki/Exponentiation#Zero_to_the_zero_power
31. FIORETTI, G., 2001. A mathematical theory of evidence for G. L. S. Shackle. *International Centre for Economic Research Working Papers*, 3(2001).
32. FIX, E., HODGES, J. L. 1951. Discriminatory analysis: Nonparametric discrimination: Consistency properties. *Report number 4, Project number* 21-49-004, USAF school of aviation medicine, Randolph Field, Texas.
33. Hagan, M. T., Menhaj, M. 1994. Training feedforward networks with the Marquardt algorithm, *IEEE Transactions on Neural Networks.* 5(6), 989–993.
34. HALL, D. 1992. *Mathematical Techniques in Multisensor Data Fusion*. Artech House, Norwood, Massachussets.
35. HALL, D. L., Linas, J. 1997. An introduction to multi-sensor data fusion. *The proceedings of the IEEE*, 85(1), 6-23.
36. HU, W., LI, J., GAO, Q. 2006. Intrusion Detection Engine Based on Dempster-Shafer's Theory of Evidence. *Communications, Circuits and Systems Proceedings, 2006 International Conference*, 3, 1627-1631.
37. KATAR, C. 2006. Combining multiple techniques for intrusion detection.*IJCSNS International Journal of Computer Science and Network Security*, 6(2B).
39. KDD99 archive: The Fifth International Conference on Knowledge Discovery and Data Mining. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.
40. KELLER, M., GRAY, M. R., GIVENS JR, J. A. 1985. A fuzzy k-nearest neighbor algorithm. *Transaction on systems, Man and Cybernetics.* vol. SMC-15(4), 580-585.

41. KOHLAS, J., MONNEY, P. A. 1994. Theory of evidence - a survey of its mathematical foundations, applications and computational anaylsis. *ZOR- Mathematical Methods of Operations Research*, 39, 35–68.
42. Llinas, J., Waltz, E. 1990. *Multisensor data fusion*. Artech House, Norwood, Massachusetts.
43. LoadControl. Demonstration to control the amount of CPU load used by your computer. http://www.codeproject.com/KB/cpp/CPU_Load_Control.aspx?msg=1915014
44. MATETI, P. 2001. Port scanning. http://www.cs.wright.edu/~pmateti/Courses/499/Probing/
45. MIRKOVIC, J., MARTIN, J., REIHER, P., 2001. A taxonomy of DDoS attacks and DDoS defense mechanisms. *Technical report 020018*. Computer Science Dept., University of California, Los Angeles.
46. MIRKOVIC, J., PRIER, G., REIHER, P. 2002. Attacking DDoS at the source. *In Proceedings of ICNP*, Paris, France *2002*, 312–321.
47. MURPHY, C. K. 2000. Combining belief functions when evidence conflicts. *Decision support systems*, 2000, 1-9.
48. PAULAUSKAS, N., GARSVA, E. 2006. Computer System Attack Classification. Electronics and Electrical Engineering. – Kaunas: Technology, 2006. 2(66), 84–87.
49. Port numbers. 2009. http://www.iana.org/assignments/port-numbers
50. Port scanning techniques. Insecure.org. http://insecure.org/nmap/man/man-port-scanning-techniques.html
51. PRESS, W. H., FLANNERY, B. P., TEUKOLSKY, A., VETTERLING, W. T. 1988. *Numerical recipes in C: The art of scientific computing*, Cambridge University Press.
52. QUINLAN, J. R. 1993. *C4.5: Programs for machine learning*. Morgan Kaufmann.
53. Registered port. "http://en.wikipedia.org/wiki/Registered_port"
54. RF PROTECT. http://www.arubanetworks.com/products/rf_protect.php
55. RFC 793. http://www.faqs.org/rfcs/rfc793.html
56. ROGERS, L. 2004. What is a Distributed Denial of Service (DDoS) Attack and What Can I Do About It? http://www.cert.org/homeusers/ddos.html.
57. Rosenfeld, R. 1996. A Maximum Entropy Approach to Adaptive Statistical Language Modeling, Computer, Speech, and Language. 10.
58. RUTA, D., GABRYS, B. 2000. An Overview of Classifier Fusion Methods. *Computing and Information Systems*, 7, 1-10.
59. SENTZ, K. 2002. *Combination of evidence in Dempster-Shafer theory*. Binghamton University Press, Binghamton.
60. SHAFER, G., 2002. Dempster-Shafer Theory.
61. SHAFER, G., 1976. *A mathematical theory of evidence*. Princeton University Press, Princeton, NJ.
62. SIATERLIS, C., MAGLARIS, B., RORIS, P. 2003. A novel approach for a distributed denial of service detection engine.
63. SIATERLIS, C., MAGLARIS, B. 2004. Towards multisensor data fusion for DoS detection. *Proceedings of the 2004 ACM symposium on Applied computing*. 439 – 446.
64. SIATERLIS, C., MAGLARIS, V. 2005. One step ahead to multisensor data fusion for DdoS detection. *Journal of Computer Security*, Vol. 13 2005, 779–806.
65. SMARANDACHE, F., DEZERT, J. 2006. An Introduction to the DSm Theory for the Combination of Paradoxical, Uncertain, and Imprecise Sources of Information. Presented

at 13th International Congress of Cybernetics and Systems, Maribor, Slovenia, July 6-10, 2005.
66. SNORT. The open source network intrusion detection system. http://www.snort.org.
67. TEO, L. 2000. Network Probes Explained: Understanding Port Scans and Ping Sweeps, Linux Journal. http://www.linuxjournal.com/article/4234
68. Transmission Control Protocol. http://en.wikipedia.org/wiki/Transmission_Control_Protocol
69. VENKATARAMANAN, V. 2005. Models for evidence analysis for intrusion detection in ad-Hoc networks. *M.S thesis*, Southern Methodist University, Dallas, TX, USA.
70. WANG, Y., YANG, H., WANG, X., ZHANG, R. 2004. Distributed intrusion detection system based on data fusion method. *Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on*, vol. 5, 4331–4334.
71. WU, H., SIEGEL, M., STIEFELHAGEN, R. 2002. Sensor fusion using Dempster-Shafer theory. *In Proceedings of IEEE instrumentation and measurement technology conference.* Anchorage, AK, USA.
72. Xmas tree scan. Networkuptime. http://www.networkuptime.com/nmap/page3-5.shtml
73. XU, L., KRZYZAK, A., SUEN, C. Y. 1992. Methods of Combining Multiple Classifiers and their Applications to Handwriting Recognition. *IEEE Trans.* SMC 22 418-435.
74. YAGER, R. R. 1987. On the Dempster–Shafer framework and new combination rules. *Information Sciences*, 1987, 93-138.
75. YU, D., FRINCKE, D. 2004. A Novel Framework for Alert Correlation and Understanding. *International Conference on Applied Cryptography and Network Security (ACNS) 2004.* Springer's LNCS series, 3089, 452-466.
76. YU, D., FRINCKE, D. 2005. Alert confidence fusion in intrusion detection systems with extended Dempster-Shafer theory. *ACM-SE 43: Proceedings of the 43rd annual southeast regional conference.* 2, 142 – 147.
77. YU, D. 2006. A novel alert correlation and confidence fusion framework in ids. *PhD thesis*, University of Idaho, Idaho, USA.
78. YU, L., LIU, H. 2003. Feature selection for high-dimensional data: a fast correlation-based filter solution. *In proceedings of the twentieth international conference o*
79. Well known SCTP, TCP and UDP ports, 0 through 999. http://www.networksorcery.com/enp/protocol/ip/ports00000.htm
80. WIRESHARK. http://www.wireshark.org/about.html

# APPENDIX A

## A.1. E-mail Communication with Aruba Networks about Configuring AP-70 Sensors

On Fri, 13 Feb 2009 13:41:52 -0800 "Senthil Kumaran" wrote:
> Hi Aqila,
>
> Thank you for calling Aruba Networks Technical Support.
>
>
> The ticket number for your reference is T-48725 and you are speaking to
> Engineer Sindhu Kizhakeel who is CC'ed on this e-mail. By the way, if
> you prefer a certain type of communication method over another, i.e.,
> email over phone, etc, please let us know as well.
>
>
> Also for all controller and/or Aruba OS related queries, please be
> prepared to provide the controllers' tar.logs as these are most commonly
> used in troubleshooting a high percentage of all issues - your assigned
> engineer will confirm this as a requirement. For instructions on
> extracting logs from! the controller, please visit our Knowledge Base
> website for Answer ID 44
> > dp.php?p_faqid=44> .
>
>
> For better service regarding this ticket, please include the ticket
> number in all further communication.
>
> Regards,
> --
> Senthil Kumaran
> Customer Support Executive | Global Support Center
> Aruba Networks Inc.

**From:** "Sindhu Kizhakeel" <skizhakeel@arubanetworks.com>

**Subject:** RE: Need assitance the provisioning AP / Ticket # T-48725 **Date:** Fri, 13 Feb 2009 15:42:45 -0800

**To:** "Dissanayake Aqila" <dissanaa@uwindsor.ca>

**Cc:** "Bhavik Kiri" <bhavik@arubanetworks.com>, "Bipin Babu" <bipin@arubanetworks.com>, "Joshua Simon" <jsimon@arubanetworks.com>, "Preethi Devarajan" <pdevarajan@arubanetworks.com>, "Jagan Smile" <jsmile@arubanetworks.com>

Hi Aqila,

As informed earlier we need to use a splitter cable and go to the boot prompt of the ap.

Apboot>purge

Apboot>save

The above mentioned steps is to remove the parameters like ip address which is already configured on the ap.Once this is done please follow the steps below

Apboot>setenv ipaddr 192.168.1.10

Apboot>setenv netmask 255.255.255.0

Apboot>setenv gatewayip 192.168.1.50

Apboot>setenv master 172.160.10.10

Apboot>setenv serverip 172.160.10.10

Apboot>save

Apboot>Reset

Apboot>boot.

The ip address that I have specified here is just for an example you can add accordingly. Important thing that you have note here is the master and the server ip that has to be entered must be the RF protect server's ip address.

Kindly let me know if you have any further queries.

Regards,

Sindhu Kizhakeel

Network engineer

Customer Advocacy Team

Aruba Networks Inc.

**From:** Dissanayake Aqila [mailto:dissanaa@uwindsor.ca]
**Sent:** Tuesday, March 03, 2009 3:32 PM
**To:** Sindhu Kizhakeel
**Subject:** Re: T-48725

Hi Sindhu,

We still haven't been able to connect to the AP-70. We have ordered an Aruba Serial Breakout Adapter, and is awaiting it's arrival. Once it arrives, we'll try again and let you know. Thanks,

Aqila.

On Tue, 3 Mar 2009 15:25:14 -0800 "Sindhu Kizhakeel" wrote:
> Hi Aqila,

**From:** Sindhu Kizhakeel
**Sent:** Friday, March 06, 2009 4:48 PM
**To:** 'Dissanayake Aqila'
**Cc:** Sriram Subramanian (Support); Preethi Devarajan; Ravi Kumar Gollapudi
**Subject:** RE: T-48725

Hi Aqila,

As per the conversation with my colleague Sriram iam sending the documents. Please do let us know if you have any further queries.

Regards,

Sindhu Kizhakeel

Network engineer

Customer Advocacy Team

Aruba Networks Inc.

| | |
|---|---|
| **From:** | "Sindhu Kizhakeel" <skizhakeel@arubanetworks.com> |
| **Subject:** | RE: T-48725 |
| **Date:** | Fri, 6 Mar 2009 16:56:21 -0800 |
| **To:** | "Dissanayake Aqila" <dissanaa@uwindsor.ca> |
| **Cc:** | "Sriram Subramanian \(Support\)" <srirams@arubanetworks.com>, "Preethi Devarajan" <pdevarajan@arubanetworks.com>, "Ravi Kumar Gollapudi" <rkgollapudi@arubanetworks.com> |

Hi Aqila,

Here is another document. Please log into to https://iris.arubanetworks.com and enter the code mentioned below. You will get the file. since it is above 12 mb I am not able to send the email and hence giving the access code below.

96a03a7f7601e288d5b75da2ff6f2cb9

Do let me know if you find any difficulties in accessing the file.

Regards,

Sindhu Kizhakeel

Network engineer

Customer Advocacy Team

Aruba Networks Inc.

| | |
|---|---|
| **From:** | "Sriram Subramanian \(Support\)" <srirams@arubanetworks.com> |
| **Subject:** | RE: T-48725 |
| **Date:** | Fri, 6 Mar 2009 19:24:32 -0800 |
| **To:** | "Dissanayake Aqila" <dissanaa@uwindsor.ca> |
| **Cc:** | "Preethi Devarajan" <pdevarajan@arubanetworks.com>, "Ravi Kumar Gollapudi" <rkgollapudi@arubanetworks.com> |

Aqila ,

Please let me know if you have any issues on the firmware upgrade or installation on the RF Protect server .

Thanks

Sriram Subramanian

Network Engineer, Customer Advocacy Team

Aruba Networks Inc.

**From:** Dissanayake Aqila [mailto:dissanaa@uwindsor.ca]
**Sent:** Wednesday, March 11, 2009 7:35 AM
**To:** Sindhu Kizhakeel
**Subject:** Re: T-48725

Hi Sindhu,

After downloading and installing the software, I'm still unable to connect to the AP-70. I e-mailed the following message to Sriram on the 7th of march but haven't got a reply yet. If you could look into that it'll be great. Thanks.

Aqila.

Hi Sriram,

After installing the new server version I get the following error when trying to start up the RF protect client. "Database version does not match this console. Please use the correct version of the console. The client version we have is 5.0.6 of RF Protect. Is it possible to obtain the newer version of client? Please let me know. Thank you.

| | |
|---|---|
| **From:** | "Sindhu Kizhakeel" <skizhakeel@arubanetworks.com> |
| **Subject:** | RE: T-48725 |
| **Date:** | Thu, 12 Mar 2009 12:59:49 -0700 |
| **To:** | "Dissanayake Aqila" <dissanaa@uwindsor.ca> |
| **Cc:** | "Sriram Subramanian \(Support\)" <srirams@arubanetworks.com>, "Preethi Devarajan" <pdevarajan@arubanetworks.com>, "Ravi Kumar Gollapudi" <rkgollapudi@arubanetworks.com> |

Hi Aqila,

Please find the access code for installing the 6.7 code version on the Rf protect client. Please log on to https://iris.arubanetworks.com and enter the access code

3affc750267ad93b739917f9364ffd67

and you can get the file. Install the software on the console and please let me know how it goes.

Regards,

Sindhu Kizhakeel

Network engineer

Customer Advocacy Team

Aruba Networks Inc.

**From:** Dissanayake Aqila [mailto:dissanaa@uwindsor.ca]
**Sent:** Friday, March 13, 2009 3:31 PM
**To:** Sindhu Kizhakeel
**Subject:** Re: T-48725

Hi Sindhu,

I was able to download the software successfully. I will let you know whether we are able to connect to the sensors with the new software. Thank you for your help.

Aqila.

**From:** Dissanayake Aqila [mailto:dissanaa@uwindsor.ca]
**Sent:** Sunday, March 15, 2009 3:29 PM
**To:** Sindhu Kizhakeel
**Subject:** Re: T-48725

Hi Sindhu,

We are still having problems configuring the AP-70 sensors. We have installed the new software (both client and server). The problem is when we try to run the commands to configure the AP-70 through Hyperterminal, the sensor doesn't restart properly and errors out. I was wondering whether there's a way for you to remotely assist us in this configuration. (Initial Configuration). We can successfully connect to the boot> prompt in the AP, but once we set the RF Protect Servers IP address in the AP and restart, it errors out. Let me know what we can do about this. Thank you.

Aqila.

On Mon, 16 Mar 2009 12:47:35 -0700 "Sindhu Kizhakeel" wrote:
> Hi Aqila,
>
>
>
> Kindly send me the error message you are getting on the screen or please
> send the screen shot of the error message.
>
>
>
> Regards,
>
> Sindhu Kizhakeel
>
> Network engineer
>
> Customer Advocacy Team
>
> Aruba Networks Inc.

**From:** Dissanayake Aqila [mailto:dissanaa@uwindsor.ca]
**Sent:** Monday, March 16, 2009 3:19 PM
**To:** Sindhu Kizhakeel
**Subject:** Re: T-48725

Hi Sindhu,

I have attached 2 screenshots of the error.

Thanks,

Aqila.

Hi Aqila,

Please give me your convenient time so that I can call you

Regards,

Sindhu Kizhakeel

Network engineer

Customer Advocacy Team

Aruba Networks Inc.

**From:** Dissanayake Aqila [mailto:dissanaa@uwindsor.ca]
**Sent:** Tuesday, March 17, 2009 1:33 PM
**To:** Sindhu Kizhakeel
**Subject:** Re: T-48725

Hi Sindhu,

How about 2 p.m central time on Thursday?

Thanks,

Aqila.

On Tue, 17 Mar 2009 12:36:12 -0700 "Sindhu Kizhakeel" wrote:
> Hi Aqila,
>
>
>
> Let me give you a call on Thursday then. Please let me know your
> convenient time so that I can call you accordingly.
>
>
>
> Regards,
>
> Sindhu Kizhakeel
>

**From:** "Sindhu Kizhakeel" <skizhakeel@arubanetworks.com>

**Subject:** RE: T-48725 **Date:** Tue, 17 Mar 2009 13:46:50 -0700

**To:** "Dissanayake Aqila" <dissanaa@uwindsor.ca>

Aqila,

Sure,

I will give you a call at that time. Meanwhile you can try this command and let me know if there is any change.

Apboot>tftp boot

Apboot>setenv master <ip address of master >

Apboot>setenv serverip <ip address of the server>

Apboot>save

Apboot>reset

Apboot>boot

Regards,

Sindhu Kizhakeel

Network engineer

Customer Advocacy Team

Aruba Networks Inc.

**From:** "Sindhu Kizhakeel" <skizhakeel@arubanetworks.com>

**Subject:** RE: T-48725

**Date:** Mon, 23 Mar 2009 16:33:03 -0700

**To:** "Dissanayake Aqila" dissanaa@uwindsor.ca

Hi Aqila,

Try the Earlier command that I had given you and if that shows the same error message then please try this and please let me know the status after you try that.

Apboot>setenv ipaddr <ipaddress>

Apboot>setenv netmask <mask>

Apboot>setenv gatewayip <gateway ip>

Apboot>setenv serverip<system ip>

Apboot>setenv master <master ip address>

Apboot>save

Apboot>reset

Apboot>boot

Now try pinging the tftp server from the ap boot prompt and please check if you are able to ping.If you are able to ping Then try the below command and please let me know if there is any change

Apboot>setenv bootcmd tftpboot

Apboot>save

Apboot>boot

Regards

Sindhu

**From:** Dissanayake Aqila [mailto:dissanaa@uwindsor.ca]
**Sent:** Tuesday, March 24, 2009 10:58 AM
**To:** Sindhu Kizhakeel
**Subject:** Re: T-48725

Hi Sindhu,

I will try and out the new commands and let you know. (I will most probably try them out on coming friday). I have a couple of questions regarding the commands.

1.) When you say "Now try pinging the tftp server from the ap boot prompt " do you mean to ping the computer where RF Protect Server is installed from the boot prompt of the sensor?

2.) In the command sequence

Apboot>setenv ipaddr
>
> Apboot>setenv netmask
>
> Apboot>setenv gatewayip
>
> Apboot>setenv serverip
>
> Apboot>setenv master
>
> Apboot>save
>
> Apboot>reset
>
> Apboot>boot

After the save step, you say to reset and then boot, when I type reset this will actually boot the sensor, typing boot again will boot the sensor again. Is this correct? Or does reset s! erve a different purpose? Let me know.

Thanks,

Aqila.

| | |
|---|---|
| **From:** | "Sindhu Kizhakeel" <skizhakeel@arubanetworks.com> |
| **Subject:** | RE: T-48725 |
| **Date:** | Tue, 24 Mar 2009 11:04:48 -0700 |
| **To:** | "Dissanayake Aqila" <dissanaa@uwindsor.ca> |

Aqila,

You are correct.

1.Ping the computer where rf protect server is installed from the boot prompt of the sensor.

2.If Reset command is booting the sensor then that serves the purpose. You can just run the command ‘reset’.

Do Let me know once you try these steps.

Regards

Sindhu

**From:** Dissanayake Aqila [mailto:dissanaa@uwindsor.ca]
**Sent:** Friday, March 27, 2009 3:27 PM
**To:** Sindhu Kizhakeel
**Subject:** Re: T-48725

Hi Sindhu,

Now we are able to successfully connect to both sensors through RF Protect. We didn't do anything different. I basically connected the AP-70 sensors directly to the router (after they were configured to point to the RF Protect Server). No TFTP Boot was required, and when we tried tftp boot it failed. Thanks your help with this matter. One thing I noticed was that the sensors act differently each time we boot them up. Somehow now RF Protect can detect them, and that's good.

Aqila.

**From:** "Sindhu Kizhakeel" skizhakeel@arubanetworks.com

**Subject:** RE: T-48725 **Date:** Fri, 27 Mar 2009 16:50:58 -0700

**To:** "Dissanayake Aqila" dissanaa@uwindsor.ca

Oh that's nice to know Aqila. May I have the status of this ticket as closed since everything is working now. please let me know

Regards,

Sindhu Kizhakeel

Network engineer

Customer Advocacy Team

Aruba Networks Inc.

# VITA AUCTORIS

| | | |
|---|---|---|
| NAME | : | Aqila Nandaka Charith Dissanayake |
| PLACE OF BIRTH | : | Colombo, Sri-Lanka |
| YEAR OF BIRTH | : | 1981 |
| EDUCATION | : | Bachelor of Science, 2005. University of Wisconsin-Superior, USA<br><br>Master of Science, 2009. School of Computer Science, University of Windsor, ON, Canada |