



# **NAVAL POSTGRADUATE SCHOOL**

**MONTEREY, CALIFORNIA**

## **THESIS**

### **DISTRIBUTED DEPLOYMENT OF THERMINATORS IN THE NETWORK**

by

Cheng Kah Wai

December 2004

Thesis Advisor:  
Second Reader:

John C. McEachen  
Su Wen

**Approved for public release; distribution is unlimited.**

THIS PAGE INTENTIONALLY LEFT BLANK

<b>REPORT DOCUMENTATION PAGE</b>			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
<b>1. AGENCY USE ONLY (Leave blank)</b>		<b>2. REPORT DATE</b> December 2004	<b>3. REPORT TYPE AND DATES COVERED</b> Master's Thesis	
<b>4. TITLE AND SUBTITLE:</b> Distributed Deployment of Therminators in the Network			<b>5. FUNDING NUMBERS</b>	
<b>6. AUTHOR(S)</b> Cheng Kah Wai				
<b>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</b> Naval Postgraduate School Monterey, CA 93943-5000			<b>8. PERFORMING ORGANIZATION REPORT NUMBER</b>	
<b>9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)</b> N/A			<b>10. SPONSORING/MONITORING AGENCY REPORT NUMBER</b>	
<b>11. SUPPLEMENTARY NOTES</b> The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
<b>12a. DISTRIBUTION / AVAILABILITY STATEMENT</b> Approved for public release; distribution is unlimited.			<b>12b. DISTRIBUTION CODE</b>	
<b>13. ABSTRACT (maximum 200 words)</b>  <p>The idea of deploying a distributed network intrusion system using Therminator is explored in this thesis. There are many advantages in having a distributed system compared to a standalone network intrusion system. The underlying principle of Therminator is modeling network traffic on conversation exchange models. Using Zippo, a new implementation of Therminator, the experimental setup consisted of multiple sensors reporting individual findings to a central server for aggregated analysis. Different scenarios of network attacks and intrusions were planned to investigate the effectiveness of the distributed system. The network attacks were taken from the M.I.T Lincoln Lab 1999 Data Sets. The distributed system was subjected to different combinations of network attacks in various parts of the network. The results were then analyzed to understand the behavior of the distributed system in response to the different attacks. In general, the distributed system detected all attacks under each scenario. Some surprising observations also indicated attack responses occurring in unanticipated scenarios. These results are subject to further investigation.</p>				
<b>14. SUBJECT TERMS</b> Distributed, Network Intrusion System, Therminator, Zippo, Lincoln Lab Data			<b>15. NUMBER OF PAGES</b>  107	
			<b>16. PRICE CODE</b>	
<b>17. SECURITY CLASSIFICATION OF REPORT</b> Unclassified	<b>18. SECURITY CLASSIFICATION OF THIS PAGE</b> Unclassified	<b>19. SECURITY CLASSIFICATION OF ABSTRACT</b> Unclassified	<b>20. LIMITATION OF ABSTRACT</b> UL	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)  
Prescribed by ANSI Std. Z39-18

THIS PAGE INTENTIONALLY LEFT BLANK

**Approved for public release; distribution is unlimited.**

**DISTRIBUTED DEPLOYMENT OF THERMINATORS IN THE NETWORK**

**Cheng Kah Wai  
Defence Science & Technology Agency Singapore  
B.E., Nanyang Technological University, 1999**

**Submitted in partial fulfillment of the  
requirements for the degree of**

**MASTER OF SCIENCE IN COMPUTER SCIENCE**

from the

**NAVAL POSTGRADUATE SCHOOL  
December 2004**

Author: Cheng Kah Wai

Approved by: John C. McEachen  
Thesis Advisor

Su Wen  
Second Reader

Peter J. Denning  
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

## **ABSTRACT**

The idea of deploying a distributed network intrusion system using Terminator is explored in this thesis. There are many advantages in having a distributed system compared to a standalone network intrusion system. The underlying principle of Terminator is modeling network traffic on conversation exchange models. Using Zippo, a new implementation of Terminator, the experimental setup consisted of multiple sensors reporting individual findings to a central server for aggregated analysis. Different scenarios of network attacks and intrusions were planned to investigate the effectiveness of the distributed system. The network attacks were taken from the M.I.T Lincoln Lab 1999 Data Sets. The distributed system was subjected to different combinations of network attacks in various parts of the network. The results were then analyzed to understand the behavior of the distributed system in response to the different attacks. In general, the distributed system detected all attacks under each scenario. Some surprising observations also indicated attack responses occurring in unanticipated scenarios. These results are subject to further investigation.

THIS PAGE INTENTIONALLY LEFT BLANK



# TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	INTRUSION DETECTION SYSTEMS .....	1
B.	THERMINATOR.....	1
C.	ZIPPO.....	1
D.	THESIS RESEARCH .....	2
E.	THESIS ORGANIZATION .....	2
II.	INTRUSION DETECTION SYSTEMS.....	3
A.	GENERAL .....	3
B.	STRUCTURE OF IDS .....	4
C.	CLASSIFICATION OF IDS .....	5
D.	DISTRIBUTED IDS (DIDS) .....	7
E.	SUMMARY .....	9
III.	THERMINATOR AND ZIPPO .....	11
A.	GENERAL.....	11
B.	CONVERSATION EXCHANGE MODEL .....	11
C.	THERMINATOR.....	13
1.	Architecture of Therminator .....	13
2.	Limitations of Therminator .....	15
D.	ZIPPO – THE NEW THERMINATOR.....	15
1.	Architecture of Zippo .....	15
2.	Zippo Application .....	16
a.	<i>Sensor Component.....</i>	<i>17</i>
b.	<i>Core Component.....</i>	<i>18</i>
c.	<i>GUI Component.....</i>	<i>18</i>
3.	Zippo Control Center (ZCC) .....	18
a.	<i>Configuration of Core Component .....</i>	<i>18</i>
b.	<i>Definition of PID Instances.....</i>	<i>20</i>
c.	<i>Sensor Listener.....</i>	<i>21</i>
d.	<i>Thermal Canyon and Thermal Tower Displays .....</i>	<i>21</i>
E.	SUMMARY .....	23
IV.	NETWORK AND SYSTEM SETUPS .....	25
A.	NETWORK SETUP .....	25
B.	NETWORK ATTACK REPLAY.....	26
1.	Mailbomb Attack .....	27
2.	Smurf Attack .....	27
3.	Apache2 Attack.....	27
C.	SYSTEM SETUP.....	28
1.	Core Component.....	28
2.	Sensor Component.....	29
3.	GUI Component .....	29

D.	DEFINED PID INSTANCES .....	29
1.	SMTP PID Instance .....	30
2.	ICMP PID Instance .....	31
3.	HTTP PID Instance .....	32
E.	CONFIGURATION PARAMETERS FOR CORE .....	33
F.	SUMMARY .....	35
V.	EXPERIMENT AND ANALYSIS I – SINGLE SENSOR.....	37
A.	EXPERIMENT SETUP .....	37
B.	MAILBOMB ATTACK.....	37
C.	SMURF ATTACK.....	40
D.	APACHE2 ATTACK.....	41
E.	SUMMARY .....	43
VI.	EXPERIMENT AND ANALYSIS II – DUAL SENSORS WITH THE SAME ATTACK .....	45
A.	EXPERIMENT SETUP .....	45
B.	MAILBOMB ATTACK.....	46
C.	SMURF ATTACK.....	49
D.	APACHE2 ATTACK.....	51
E.	ANALYSIS OF RESULTS.....	54
F.	SUMMARY .....	55
VII.	EXPERIMENT AND ANALYSIS III – DUAL SENSORS WITH DIFFERENT ATTACKS.....	57
A.	EXPERIMENT SETUP .....	57
B.	SMURF AND MAILBOMB ATTACKS .....	58
1.	SMTP PID Instance .....	58
2.	ICMP PID Instance .....	62
C.	SMURF AND APACHE2 ATTACKS.....	67
1.	ICMP PID Instance .....	67
2.	HTTP PID Instance .....	71
D.	MAILBOMB AND APACHE2 ATTACKS.....	75
1.	SMTP PID Instance .....	75
2.	HTTP PID Instance.....	79
E.	ANALYSIS OF RESULTS.....	82
F.	SUMMARY .....	83
VIII.	REPORT SUMMARY AND FUTURE RESEARCH .....	85
A.	REPORT SUMMARY .....	85
B.	FUTURE RESEARCH.....	86
	LIST OF REFERENCES.....	87
	INITIAL DISTRIBUTION LIST .....	89

## LIST OF FIGURES

Figure 1.	Interaction of CIDF components in an IDS. ....	5
Figure 2.	Classification of IDS. ....	5
Figure 3.	A decision tree with four defined buckets. ....	12
Figure 4.	Therminator System .....	13
Figure 5.	Interaction between MVC components.....	16
Figure 6.	Distributed IDS using Zippo.....	17
Figure 7.	Examples of the a) Thermal Canyon. And b) Thermal Tower.....	19
Figure 8.	A PID instance named SMTP.....	20
Figure 9.	Thermal Canyon .....	22
Figure 10.	Thermal Tower .....	22
Figure 11.	Thermalate data .....	23
Figure 12.	Network Topology of Experiment Setup .....	25
Figure 13.	DARPA Simulation Network (Source:[8]).....	26
Figure 14.	SMTP PID Instance.....	30
Figure 15.	ICMP PID Instance.....	31
Figure 16.	HTTP PID Instance.....	32
Figure 17.	Thermal Canyon Displays for Mailbomb attack with (a) Number of Time Slices = 120, Slide Length = 1, (b) Number of Time Slices = 60, Slide Length = 2, (c) Number of Time Slices = 200, Slide Length = 2, (d) Number of Time Slices = 100, Slide Length = 2.....	35
Figure 18.	Experiment Setup for a Single Sensor.....	37
Figure 19.	(a)Thermal Canyon and (b)Thermal Tower for SMTP PID Instance during the Mailbomb Attack. ....	38
Figure 20.	Patterns of Conversation Exchanges during the Mailbomb Attack. ....	39
Figure 21.	(a)Thermal Canyon and (b)Thermal Tower for ICMP PID Instance during the Smurf Attack. ....	40
Figure 22.	(a)Thermal Canyon and (b)Thermal Tower for HTTP PID Instance during the Apache2 Attack. ....	41
Figure 23.	Patterns of Conversation Exchanges during the Apache2 Attack. ....	42
Figure 24.	Experiment Setup for Dual Sensors with the Same Attack.....	45
Figure 25.	Thermal Canyon and Tower Displays for (a) and (c) Single Sensor, (b) and (d) Dual Sensors during a Mailbomb attack. ....	48
Figure 26.	Thermal Canyon and Tower Displays for (a) and (c) Single Sensor, (b) and (d) Dual Sensors with Smurf attacks. ....	51
Figure 27.	Thermal Canyon and Tower Displays for (a) and (c) Single Sensor, (b) and (d) Dual Sensors with Apache2 attacks. ....	54
Figure 28.	Experiment Setup for Dual Sensors with Different Network Attacks...	57
Figure 29.	Thermal Canyons and Thermal Tower for SMTP PID Instance during (a), (b) Mailbomb attack; (c), (d) Smurf attack and (e), (f) combined Mailbomb and Smurf attacks.....	61

Figure 30.	Thermalate contributing to the canyon peak of the SMTP PID Instance during the combined Mailbomb and Smurf attacks. ....	62
Figure 31.	Thermalate Contributing to the Canyon Floor of the SMTP PID Instance during the combined Mailbomb and Smurf Attacks. ....	62
Figure 32.	Thermal Canyons and Thermal Tower for ICMP PID Instance during (a), (b) Smurf attack; (c), (d) Mailbomb attack and (e), (f) combined Mailbomb and Smurf attacks. ....	66
Figure 33.	Thermalate Contributing to the Canyon Peak of the ICMP PID Instance during the combined Mailbomb and Smurf attacks. ....	67
Figure 34.	Thermal Canyons and Thermal Tower for ICMP PID Instance during (a), (b) Smurf attack; (c), (d) Apache2 attack and (e), (f) combined Smurf and Apache2 attacks. ....	71
Figure 35.	Thermal Canyons and Thermal Tower for HTTP PID Instance during (a), (b) Apache2 attack; (c), (d) Smurf attack and (e), (f) combined Apache2 and Smurf attacks. ....	74
Figure 36.	Thermal Canyons and Thermal Tower for SMTP PID Instance during (a), (b) Mailbomb attack; (c), (d) Apache2 attack and (e), (f) combined Mailbomb and Apache2 attacks. ....	78
Figure 37.	Thermal Canyons and Thermal Tower for HTTP PID Instance during (a), (b) Apache2 attack; (c), (d) Mailbomb attack and (e), (f) combined Apache2 and Mailbomb attacks. ....	82

## LIST OF TABLES

Table 1.	Denotation of Buckets for SMTP PID Instance. Buckets 0 and 4 are not applicable (N.A.) because they represent classifications that cannot occur (i.e. – a port number 25, 110, 113 or 161 that is at the same time not 25, 110, 113, or 161).....	31
Table 2.	Denotation of Buckets for ICMP PID Instance.....	32
Table 3.	Denotation of Buckets for HTTP PID Instance. Similar to table 1, buckets 0 and 4 represent classifications that cannot occur.....	33
Table 4.	Flows of Ball Transfer between Defined Buckets of SMTP PID Instance during the Mailbomb Attack.....	39
Table 5.	Flows of Ball Transfer between Defined Buckets during the Apache2 Attack. ....	43
Table 6.	State Changes with a Single Network Packet from A to B.....	55
Table 7.	State Changes with Duplicate Network Packets from Node A to B. ...	55
Table 8.	Summary of Responses to attacks. ....	82

THIS PAGE INTENTIONALLY LEFT BLANK

## **ACKNOWLEDGMENTS**

I would like to thank Dr John McEachen and Dr Su Wen for their kind guidance and support, and also Dr John Zachary and Junling for their help in the installation of Zippo. Last but not least, I would like to thank my husband, Kim Pin, for his unwavering support and sacrifices made during the past year.

THIS PAGE INTENTIONALLY LEFT BLANK



## **ACRONYMS AND ABBREVIATIONS**

DARPA	Defense Advanced Research Projects Agency
DIDS	Distributed Intrusion Detection System
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
HTTP	Hypertext Transfer Protocol
PID	Patternless Intrusion Detection
SMTP	Simple Mail Transfer Protocol
TCP	Transmission Control Protocol
ZCC	Zippo Control Center

THIS PAGE INTENTIONALLY LEFT BLANK

## **I. INTRODUCTION**

### **A. INTRUSION DETECTION SYSTEMS**

Intrusion Detection Systems (IDS) have gained increasing importance in ensuring the overall security of organizations. They act as an additional layer of security to the organization's perimeter defense, which usually, is implemented using firewalls. Firewalls are effective in preventing unauthorized entry into the organization's network. However, firewalls cannot detect unauthorized behavior that is present in network traffic they allow to go through.

The role of detecting anomalous behavior is performed by IDS, which try to identify and report attacks and security incidents [1]. There are two categories of IDS: network-based IDS and host-based IDS [2]. Network-based IDS monitor and analyze network traffic in the network segments where they are installed. Host-based IDS monitor and analyze network traffic that goes in and out of specific hosts.

### **B. THERMINATOR**

The Therminator is a network-based IDS created by Stephen Donald and Robert McMillen [3], using a mathematical model developed by Dr David Ford [4]. In the model, buckets represent classifications of network nodes and balls represent IP (Internet Protocol) packets. Conversations between network nodes are modeled by the movement of balls between buckets. Network intrusions result in anomalies in the conversation exchange between nodes. Therminator provides a graphical means of displaying the intrusions in real time.

### **C. ZIPPO**

ZIPPO is a newer version of Therminator, developed by the University of South Carolina (USC). Written in a mix of Java and C++ languages, ZIPPO provides a more user-friendly interface and consists of three components: the sensor, the core and the graphical user interface (GUI). With a modular design,

ZIPPO offers better robustness and portability. ZIPPO was the first version of Terminator to aggregate input from multiple sensors into one display. This new aspect of ZIPPO will be the focus of thesis.

#### **D. THESIS RESEARCH**

The area of interest in this research is implementing, analyzing and validating a distributed IDS using ZIPPO. This is achieved by distributing the sensors on different network segments to monitor the traffic in different parts of the network. The sensors relay the information to the core component, which converts it into a pre-defined format before analyzing it and displaying it on the GUI.

This research will be accomplished by analyzing the response of the distributed IDS to network attacks under a variety of conditions. In particular, the Smurf, Mailbomb and Apache2 attacks extracted from the MIT Lincoln Lab IDS datasets will be used to generate a distinctive response [\[5\]](#).

#### **E. THESIS ORGANIZATION**

Chapter II looks into the different types of IDS currently available, and compares central and distributed IDS. It also explains the conversation exchange model in more depth.

Chapter III discusses the architecture of ZIPPO, while Chapter IV explains the experiment setup and methodology.

Chapters V to VII present and analyze the results that were obtained from the experiments. Chapter VIII concludes the report with a summary of the findings obtained, and discusses the possible areas of future research.

## II. INTRUSION DETECTION SYSTEMS

### A. GENERAL

The aim of IDS is to detect, and possibly prevent an on-going network attack on one or more systems. Network attacks can generally be classified as passive or active. Passive attacks gain access to systems without compromising any resources, while active attacks will result in unauthorized changes to the resources. These attacks can be executed either from outside the organization network (often via the Internet), or from inside the network by employees, trusted users, etc.

The types of network attacks which IDS can identify are [\[6\]](#):

- i. **Unauthorized Access to Resources.** The intruder attempts to gain access to systems using password cracking, Trojan horses, interceptions of TCP sessions, spoofing, etc. He may also stealthily probe for information using port scans, IP scans, operating system fingerprinting, etc.
- ii. **Unauthorized Modification of Resources.** The intruder may make unauthorized configuration changes to systems and network resources, alter or delete information on systems after gaining unauthorized access to them.
- iii. **Denial of Service (DoS).** A targeted system is flooded with unnecessary information to block out legitimate traffic and deny services, e.g. ping and mail flood. The system can also be compromised by exploiting its vulnerabilities, e.g. buffer overflow.

In examining the network traffic or systems for anomalous behavior, an IDS may not be entirely correct in their diagnoses. False positives are generated when IDS wrongly identify particular events as intrusions or attacks. False negatives occur when IDS overlook an attack and let it go through without sounding any alarms.

## B. STRUCTURE OF IDS

Many different types of IDS are available in the market, be it commercially or academically developed. Each has its own design and implementation, but in general, IDS are made up of similar components as defined by the Common Intrusion Detection Framework (CIDF) [\[7\]](#). These include:

- i. **Event generators (E-Boxes).** These are sensors in the networks which monitor network traffic and activities, and generate events accordingly. These events are then passed to the other parts of the IDS.
- ii. **Event Analyzers (A-Boxes).** These boxes process and analyze data that is sent from the E-boxes. The processed information may be summarized, statistically profiled, or correlated to obtain further information.
- iii. **Event Databases (D-Boxes).** The huge amount of data that may be produced by E-boxes and A-boxes are stored in D-boxes, which can be assessed anytime by system administrators.
- iv. **Response Units (R-Boxes).** These provide countermeasures to attacks, such as killing processes, resetting connections, altering file permissions, etc.

Figure 1 shows how these boxes interact with each other in an intrusion detection system.

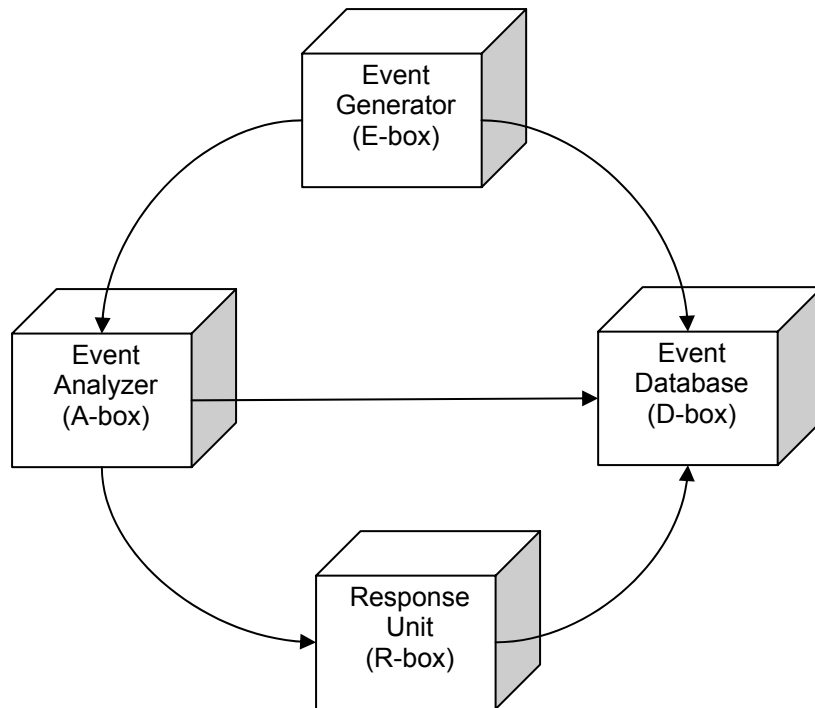


Figure 1. Interaction of CIDF components in an IDS.

### C. CLASSIFICATION OF IDS

There are many ways to classify IDS [8], as shown in Figure 2.

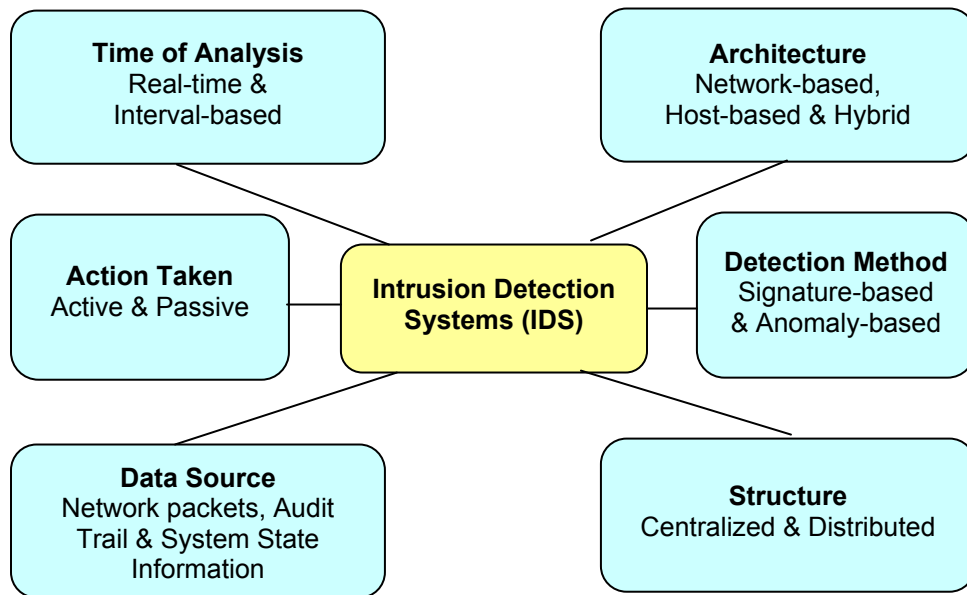


Figure 2. Classification of IDS.

- i. **Architecture. Network-based IDS (NIDS)** monitor and analyze inbound and outbound network traffic on the network segments where they are installed. Each data packet is examined in real time for signs of intrusion. Though NIDS may be effective in monitoring an entire, large network with only a few well-placed nodes, they are unable to analyze encrypted data. **Host-based IDS (HIDS)** analyze inbound and outbound network traffic of specific systems which they are installed on, check the integrity of system files and look out for suspicious process activities. Most importantly, they can examine encrypted traffic, data, storage and activities. However HIDS do consume a lot of resources, e.g. processing time, storage, memory, etc, on the systems they operate on. **Hybrid IDS** combine HIDS technology with the ability to monitor inbound and outbound network traffic of specific hosts using NIDS technology.
- ii. **Detection Method. Signature-based IDS** analyze network traffic or activities based on matches for patterns of events known to specific attacks. This type of IDS is effective against known attacks. One major disadvantage is that signature databases must constantly be updated, otherwise new attacks will not be detected. **Anomaly-based IDS** compare network traffic or activities against normal usage profiles. By creating baselines of normal behavior, these IDS can observe when current behavior differs from normal behavior, and determine if attacks are underway. The advantage is that both known and new attacks may be detected easily. However, as normal behavior can change easily, these IDS may set off false alarms when there is normal network traffic which contributes to deviations.
- iii. **Structure. Centralized IDS** operate standalone, with centralized applications physically integrated within a box, while **Distributed IDS** consist of multiple IDS over a large network, all of which



communicate with each other. More information on distributed IDS will be provided in Section D.

- iv. **Data Source.** The data which IDS analyze can either be raw **network packets**, event logs and alarms from **audit trail**, or **system state information**.
- v. **Action Taken. Active IDS** respond to attacks proactively once they are detected, e.g. log out potential intruders, block services or even reset a TCP connection on behalf of the victim. They attempt to stop the attacks before more damage is caused. **Passive IDS** only generate alerts and log network packets when attacks are detected.
- vi. **Time of Analysis.** IDS which process data in **real-time**, monitor the data constantly and analyze current information to sense possible attack attempts. This is useful in stopping attacks that are underway. **Interval-based** IDS monitor periodic feeds of information such as event logs, system alarms and other system information from audit trail. The advantage is events can be correlated to detect an attack trend, especially if the attacker tries to masquerade attack attempts over an extended period of time.

#### **D. DISTRIBUTED IDS (DIDS)**

The biggest shortcoming in centralized, standalone IDS is that they are built on a single physical entity, which is responsible for both collecting and analyzing data. This can impose severe limitations on efficiency and the system resources, especially when a high volume of data needs to be processed. DIDS can overcome this shortcoming, by performing distributed data collection and possibly preprocessing, depending on the design of the system.

DIDS consist of multiple sensors deployed in different areas of a large network, all of which report to a central server that aggregates the information and processes it. The sensors should ideally be deployed on separate network segments and geographical locations.

These are many benefits which DIDS offer as compared to the centralized, standalone IDS:

- i. **Efficiency.** As explained earlier, higher efficiency can be achieved with distributed data collection and preprocessing.
- ii. **Holistic View of the Entire Network.** By placing sensors in strategic locations in different parts of the network, the DIDS can provide a holistic view of the entire network, which will allow administrators and analysts to make better judgments.
- iii. **Early Detection.** By placing sensors in different geographical locations, attack patterns may be detected across the entire organization network. It may be possible to have early detections of coordinated attacks on the organization. This allows network and system administrators to secure targeted systems and arrest intruders at the different entry points of the network.
- iv. **Scalability and Configurability.** Sensors can be easily added to or removed from the network. As changes to the network topology and configurations are common, especially in large organization networks, this allows more flexibility for the network and system administrators.
- v. **Reliability.** There is no single point of failure as different parts of the network are monitored by different sensors. Even if a sensor is brought down by a network attack, the information will have been sent to the centralized host for alerting and forensic analysis.
- vi. **Extensibility.** New features and changes can be made to the sensors one at a time, without much interruption to the monitoring of the network.

## **E. SUMMARY**

This chapter explained the underlying structure of IDS, and gave an overview of the types of network attacks which can be detected. There are several methods of classifying IDS, such as by their architecture, method of detection and structure, etc. When the structure of the IDS is used for classification, IDS can either be centralized or distributed. Distributed IDS offer many advantages over centralized IDS.

The next chapter will explore Therminator, an IDS that is developed by Naval Postgraduate School, and Zippo, that is a new implementation of Therminator, in greater details.

THIS PAGE INTENTIONALLY LEFT BLANK

### **III. THERMINATOR AND ZIPPO**

#### **A. GENERAL**

Therminator is a network-based IDS that was originally developed at Naval Postgraduate School. It is based on a conversation exchange model, and uses statistical mechanics and thermodynamic principles to detect anomalous behavior in a monitored network. Many tests were conducted on Therminator, which has been proven to be successful in detecting different types of network intrusions.

In spite of Therminator's success, there were several shortcomings in the program. A research team from University of Southern Carolina was contracted to address these inadequacies by developing a new version of the program, which they codenamed Zippo. New capabilities were also added to this new version of Therminator.

#### **B. CONVERSATION EXCHANGE MODEL**

The underlying concept of Therminator and Zippo is based on the conversation exchange model, which is used to model network traffic. It defines a conversation exchange as an exchange of information between two conversation groups. These conversation groups may represent network nodes, protocols or the tasks which network nodes perform (e.g. client or server). This model uses buckets to represent conversation groups and balls to represent the information that is exchanged between the conversation groups.

Network traffic analysis is based on decision trees that have buckets as leaf nodes. At the beginning of the analysis, each bucket starts off with an initial number of balls. These balls are dynamically moved around in accordance with conversation exchanges that are modeled on information extracted from the network traffic. For instance, the buckets in Figure 3 represent four network nodes. A conversation exchange of  $n$  network packets between nodes A and B will result in the movement of  $n$  balls from bucket  $B_A$  to bucket  $B_B$ . However, the

number of balls in each bucket cannot decreased below a minimum level or increased beyond a maximum level, as pre-defined in the decision tree.

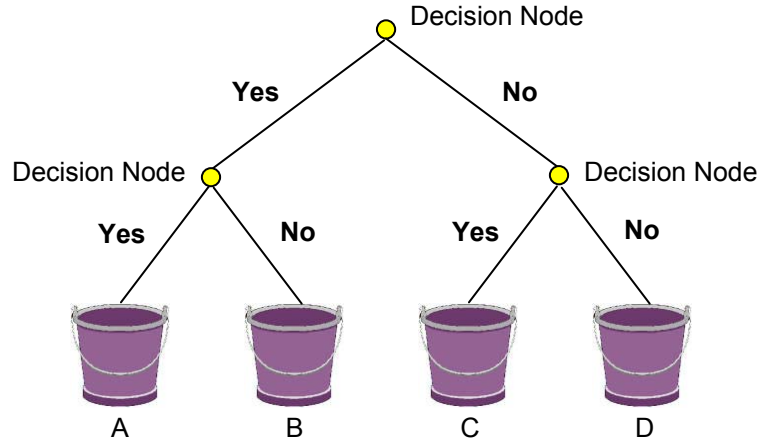


Figure 3. A decision tree with four defined buckets.

During the network traffic analysis, the number of balls in each bucket is constantly varying. A network state is the combination of the number of balls in each bucket at any given time. The state space covers the entire range of possible number of states,  $N$ , which is determined by the formula [9]:

$$N = \binom{M+K-1}{M-1} \quad (3.1)$$

where  $M$  is the number of buckets, and  $K$  is the total number of balls in the system. A state space walk traces all the states that were visited during a given period of time.

The average number of balls in the buckets can be represented in real time on a 3-D graphical display, known as thermal tower. Information about the network states visited and the number of occurrences can also be accumulated and plotted on a 3-D graphical display, known as a thermal canyon. When there are unusually high counts of certain states, or when there are a large number of states that are usually not visited, it can be an indication of anomalous network

activity. Thermodynamic principles of energy, entropy and temperature can be applied to the thermal canyon, which reveal more information about the network health.

## C. THERMINATOR

### 1. Architecture of Therminator

Therminator was developed using C programming language, and comprises of 3 components: the Sucker, the Patternless Intrusion Detection System (PID) and the GUI, as illustrated in Figure 4.

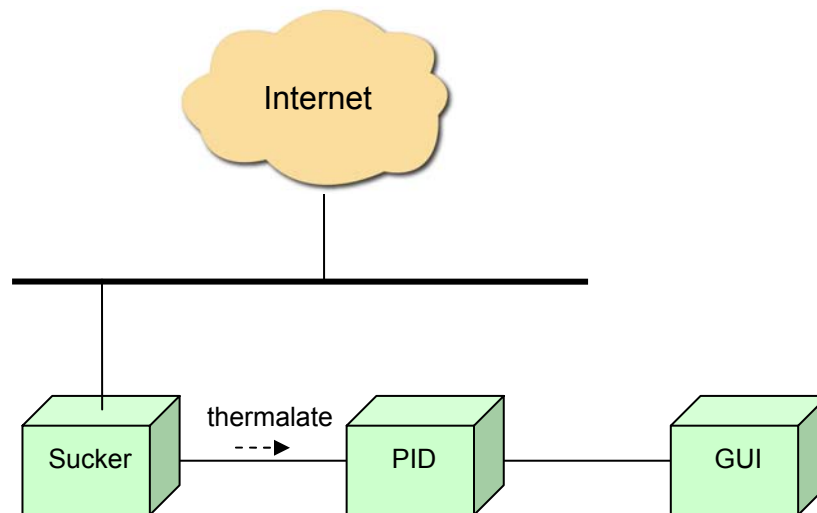


Figure 4. Therminator System

The Sucker, also known as a sensor, is basically a sniffer which captures or “sucks” every IP packet off the network. Sucker captures the network packets using libpcap and analyzes the protocols using a custom packet decoder.

Sucker extracts essential components of the decoded information, including IP header length, TCP header length and protocol used. It also analyses the information and sets the values of defined variables FLOW, MATCH, BROADCAST and NEW FRIEND. The BROADCAST bit is set if the destination IP address in the packet is a broadcast address. The FLOW field indicates if the packet is traveling in or out of the monitored network. The MATCH

bit is set if the packet is a response to a stimulus, after a state inspection. Lastly, the NEW FRIEND bit indicates if a visiting host is new to the monitored network. These pieces of information are then put together in a new data format, known as thermalate.

Thermalate is sent to the PID at regular intervals through a TCP socket connection, where it undergoes thermodynamic analysis based on pre-defined configuration files. A typical configuration file defines the decision tree, and the initial, minimum and maximum number of balls in the buckets. After the thermodynamic analysis, the data is stored and processed according to the SLIDELENGTH (SL) and WINDOWLENGTH (WL) parameters also specified in the configuration file.

The SL defines the single display time interval on the thermal canyon and thermal tower, while the WL is the period of time which the data is averaged over. Another parameter, the smoothing ratio (SR) [\[10\]](#), is derived from the following formula:

$$SR = \frac{WL}{SL} \quad (3.2)$$

The SL, WL and SR are important parameters that affect how the data is represented in the thermal canyon. Inappropriate settings can result in false positives or missed intrusions. Marinovich and Walch [\[11\]](#) showed in their investigations that small values for SL and WL could result in missed intrusions, while large SL and WL values presented late information.

The GUI allows different configuration settings for the plotting of the thermal canyon and the thermal tower. It retrieves the output data files from the PID and presents the graphic displays using the XRT/3d 3.0 widget, formerly from KL Group Inc.



## **2. Limitations of Therminator**

Therminator has proven to be highly successful, as it accurately picked up signs of intrusions and network attacks in the many tests that were conducted. There are, however, limitations to the program which are briefly described below:

- a. The configurations of the decision tree and the bucket space cannot be changed on the “fly”. The program has to be stopped and restarted to effect the new changes.
- b. The graphics for the thermal canyons and towers are created using a commercial graphical widget, the XRT/3d 3.0. The license for this widget is limited to a Solaris 8 system; it cannot be run on any other platforms.
- c. The PID is programmed to receive data only from one sensor. Therminator is not capable of supporting a DIDS architecture, whereby multiple sensors report to a central server and information is aggregated for analysis.

## **D. ZIPPO – THE NEW THERMINATOR**

When Zippo was being developed, the goal was to build a software system which is robust, high performing, portable and scalable, and able to overcome the limitations of Therminator.

### **1. Architecture of Zippo**

ZIPPO is written in a mix of Java and C++ languages, and is based on a Model-View-Controller (MVC) software architecture, which separates the application into three distinct components: data model, user interface and control logic [\[12\]](#).

The Model object is the core of the system, which implements the state space model and executes the complex operations of the underlying thermodynamics model. It takes in information about its data elements from the Controller, and passes computed information to the View object for display.

The View object takes the responsibility of presenting data from the Controller and the Model in a combination of graphics and text, (e.g. the Zippo Control Center GUI and the 3-D displays of the thermal canyons and towers.)

The Controller mainly translates interactions with the View object into actions to be performed by the Model. It is responsible for the configuration of the application, extraction and conversion of the data from network packets into thermalate, and storage of thermalate for later analysis. The Controller implements the sensor and the core functions of the application, which will be explained in the next section. Figure 5 shows the interaction between the MVC components.

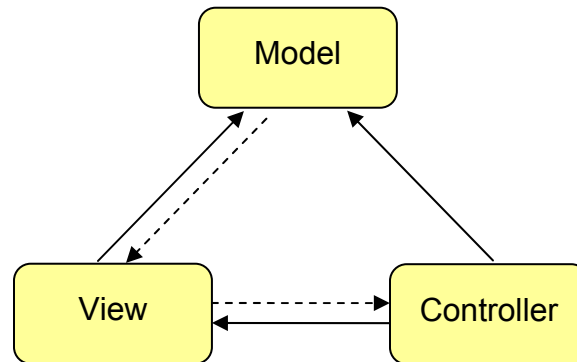


Figure 5. Interaction between MVC components.

## 2. Zippo Application

The Zippo application consists of three components: the sensor, the core and the GUI. The MVC architecture of Zippo allows these components to reside on separate host machines. Most importantly, the core is able to receive data from multiple sensors. This demonstrates that Zippo is capable of supporting a DIDS architecture, which is the focal point of this thesis. In addition, there can be multiple GUI displays for each core instance. Figure 6 shows how a DIDS architecture can be achieved using Zippo.

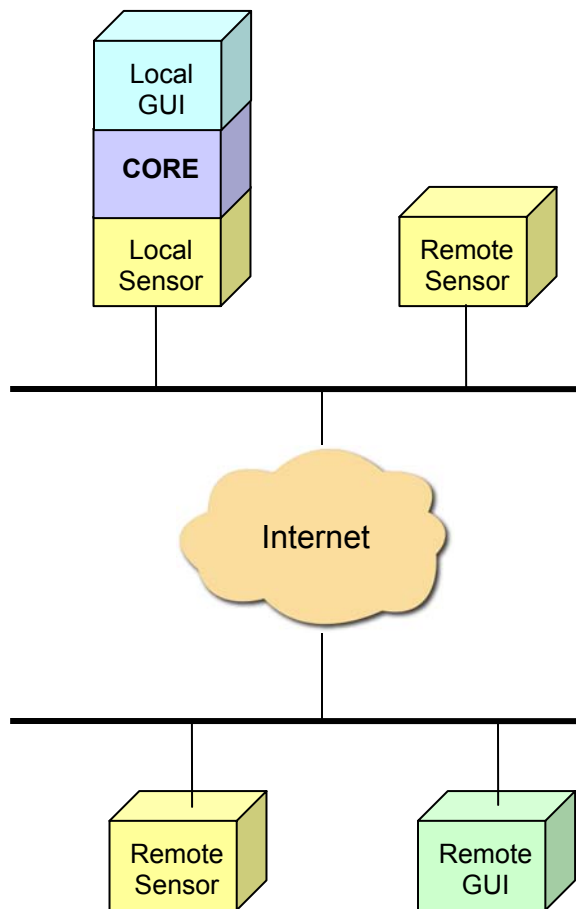


Figure 6. Distributed IDS using Zippo.

**a. Sensor Component**

The sensor component can be installed locally (on the same host machine as the core component) or remotely (on a different host machine as the core component). A local sensor and a remote sensor perform the same functions. The only difference between them is that the former is coded in C++ language, while the latter is coded in Java language for tighter coupling to the core component.

The sensor captures network packets from a live network using libpcap. It then converts the captured data into thermalate before sending it to the core component for processing. The sensor component has a configuration file in

which the IP address of the machine hosting the core component and the communicating TCP port number have to be specified.

As an added measure of security, communications between the sensor and core components is encrypted via OpenSSL. OpenSSL is an open source implementation of Secure Sockets Layer (SSL) and Transport Layer Security (TLS). The installation of Zippo comes with the default private and public keys for the sensor and core components and a self-signed Certificate Authority (CA). Users of Zippo can implement their own Public Key Infrastructure (PKI) in the system if it is desired.

***b. Core Component***

The core component is written in Java language, and hence is OS independent. However, the machine that hosts the core component is required to install Java 2 SDK. The core component aggregates thermalate that is received from the various sensors, and performs the thermodynamic analysis. The results from the complex computations are then sent to the GUI component for display.

***c. GUI Component***

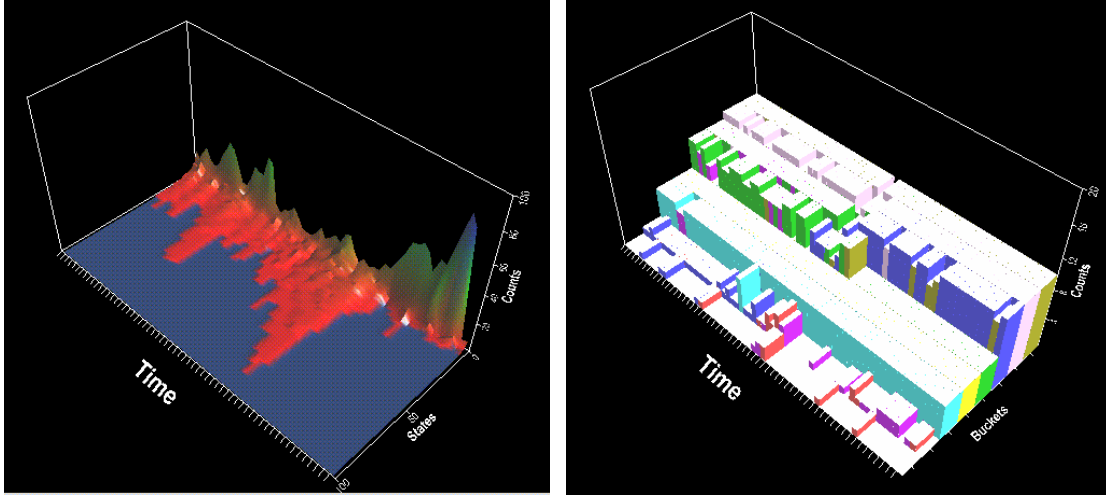
The GUI component provides interaction between the user and Zippo through Zippo Control Center (ZCC), which allows the user to administer the configuration of the sensor and the core components, and to display results on the thermal canyons and thermal towers. The GUI component needs to be connected to the core process by specifying the IP address of the machine hosting the core component, and the TCP socket to connect through.

**3. Zippo Control Center (ZCC)**

ZCC provides an intuitive and user-friendly interface for users to configure the application according to their needs. The key components of ZCC are described in this section:

***a. Configuration of Core Component***

The parameters which can be configured for the core component are: number of states, number of time slices, slide length and Smoothing Factor (SF). These parameters affect the displays on the thermal canyon and thermal tower.



(a)

(b)

Figure 7. Examples of the a) Thermal Canyon. And b) Thermal Tower

Figure 7 shows a thermal canyon and a thermal tower, with the following parameters: number of states = 100; number of time slices = 60; slide length = 2; and SF = 0.7. The specification of the number of states sets the limit on the states scale on the thermal canyon, which is 100 in this case. The number of time slices sets the length of the time display on both the thermal canyon and thermal tower. The slide length is defined in seconds, and indicates the time width of each time slice. Hence in this example, the length of the time display is 120s (the number of time slices multiplied by the slide length).

Zippo uses an exponential weighted averaging mechanism to process and display state information on the thermal canyon. The SF is a variable used in the following formula:

$$AverageValue = ActualValue \times (1 - SF) + LastAverageValue \times SF$$

The SF can take any values between zero and one. Different values of SF affect the display of the results on the thermal canyon. Larger values of SF will produce “smoother” thermal canyons.

### ***b. Definition of PID Instances***

The definition of a PID instance starts with the building of a decision tree for the thermodynamic analysis of thermalate and the corresponding bucket space. The PID instance can be given a meaningful name to reflect its characteristic. For example, it can be named SMTP if it is to be used to analyze thermalate for SMTP-based attacks, as shown in Figure 7. Starting from the top of the decision tree, a decision node can be based on any of these parameters: IP List, Port List, New Friend, Matched, Protocol List. The buckets at the bottom of the decision tree need to be defined with the ball parameters, and can be assigned with different colors, which will be reflected in the thermal towers display.

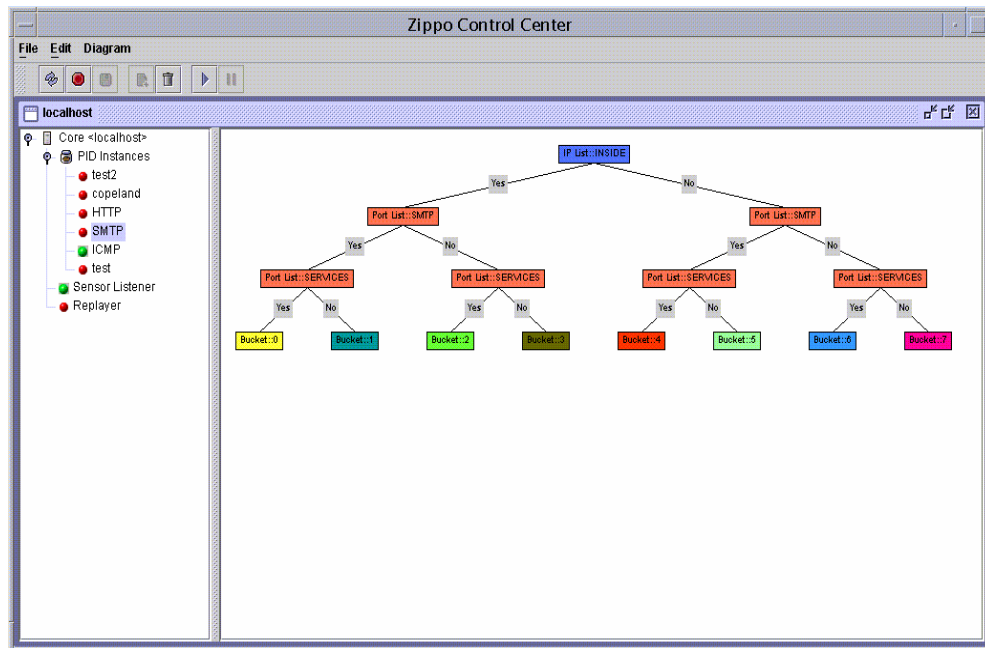


Figure 8. A PID instance named SMTP.

A PID instance functions as a service, which can be started and stopped at any point in time. This facilitates making instantaneous changes to the PID instance, without any interruptions to the core process. Multiple PID

instances can be created and started at the same time. Each instance will process and analyze the same set of thermalate and interpret the results accordingly.

**c. *Sensor Listener***

The Sensor Listener is also a start/stop service, which controls local and remote sensors. When the service is started, the IP addresses of the connected sensors will be displayed. Several parameters of the Sensor Listener can be configured including the key store password. The key store password is a shared password between the sensor and the core components, which allows the sensor component to authenticate with the core component.

**d. *Thermal Canyon and Thermal Tower Displays***

For each PID instance that is started, a thermal canyon and a thermal tower can be generated to display the results. Hence, on the same monitor display there can be multiple thermal canyons to detect different forms of network attacks (e.g. Smurf and mailbomb attacks.)

The thermal canyon displays the state information in real-time, as shown in Figure 9. Each point on the z-axis shows a unique state that was visited, while the number of occurrences of each visited state is plotted along the y-axis.

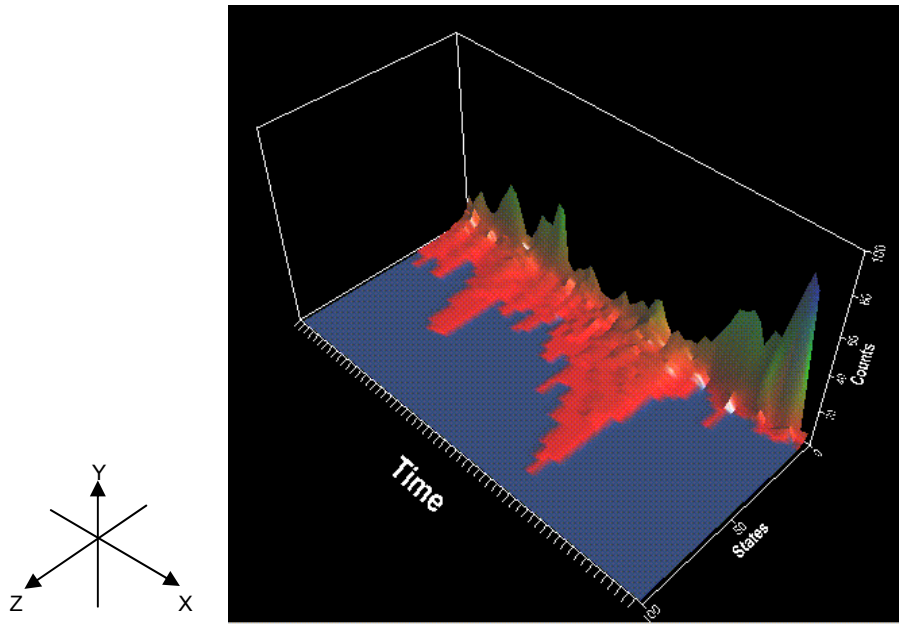


Figure 9. Thermal Canyon

The thermal tower shows the average number of balls in each bucket in real-time. Each bucket is represented by a different color, as defined in the PID instance. During each time slice, the buckets are sorted along the z-axis in decreasing order of the number of balls, as shown in Figure 10.

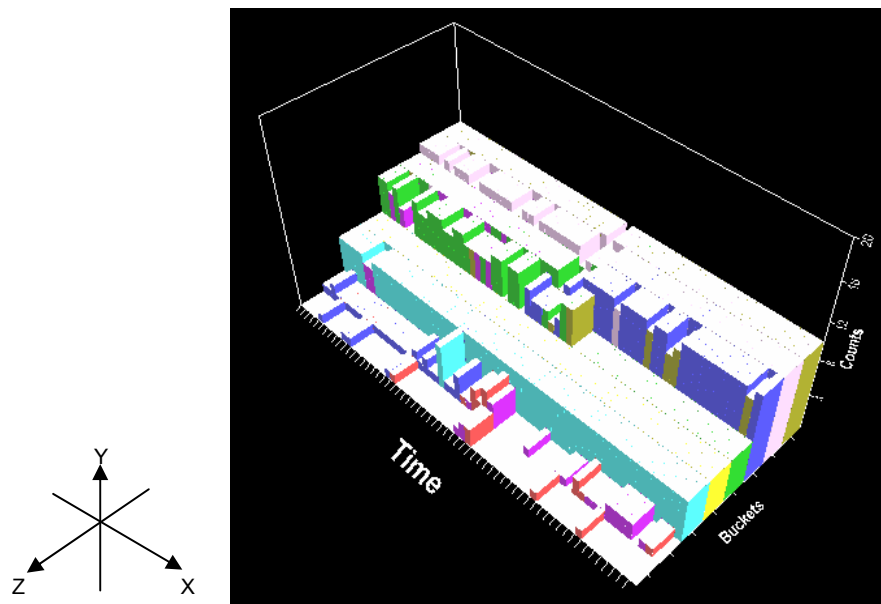


Figure 10. Thermal Tower



Both the thermal canyon and thermal tower displays can be rotated in any angle, and zoomed in for a close-up look. The scale on the y-axis for both displays is variable and adjusts according to the highest value that is shown at the time of interest. The user can also click on a point of interest on the thermal canyon or thermal tower to display the thermalate data for more detailed information, as shown in Figure 11.

Thermalate Information										
Thermalates										
Timestamp: Tue Sep 07 16:21:55 PDT 2004										
Bucket State: [5] [1] [10] [0] [5] [9] [10] [9]										
Total Packets: 19										
Src	SrcPort/ Type	Dst	DstPort/ Code	Flow	Match	New Friend	Fragmented	Protocol	Pkt Count	
172.16.113.204	21307	195.115.218.108	23	OUT_OUT	false	false	false	TCP	7	
195.115.218.108	23	172.16.113.204	21307	OUT_OUT	false	false	false	TCP	3	
172.16.114.50	25	194.27.251.21	2127	OUT_OUT	false	false	false	TCP	1	
172.16.114.207	3327	196.37.75.158	23	OUT_OUT	false	false	false	TCP	1	
172.16.114.207	30933	196.37.75.158	23	OUT_OUT	false	false	false	TCP	2	
196.37.75.158	23	172.16.114.207	30933	OUT_OUT	false	false	false	TCP	1	
196.227.33.189	13580	172.16.114.168	25	OUT_OUT	false	false	false	TCP	3	
172.16.114.168	25	196.227.33.189	13580	OUT_OUT	false	false	false	TCP	1	

Figure 11. Thermalate data

## E. SUMMARY

This chapter reviewed the conversation exchange model, which is the underlying concept of Terminator and Zippo, and looked into the implementation details of each. Compared to Terminator, Zippo is much more user-friendly and has the added feature of receiving and aggregating information from multiple sensors. The next chapter will describe how this feature is analyzed and the setup of the experiments.

THIS PAGE INTENTIONALLY LEFT BLANK

## IV. NETWORK AND SYSTEM SETUPS

### A. NETWORK SETUP

The focus of the thesis is to implement and validate a DIDS using Zippo. Figure 12 shows the network topology of the experiment setup. The network attack replay workstation has two network interface cards, which separately send out pre-recorded network traffic containing both normal user traffic and simulated network attacks. The hubs receive and broadcast the network packets, which are then picked up by sensors A and B.

Sensor A and the Zippo core component reside on Network A, while sensor B resides on Network B. This demonstrates that a remote sensor is capable of routing thermalate to the core component. The two sensors sniff every network packet on their respective network segments and produce thermalate that is sent to the Zippo core. Upon processing and analysis, the Zippo core generates thermal canyons and thermal towers that reflect the state of the networks. These results are verified and validated.

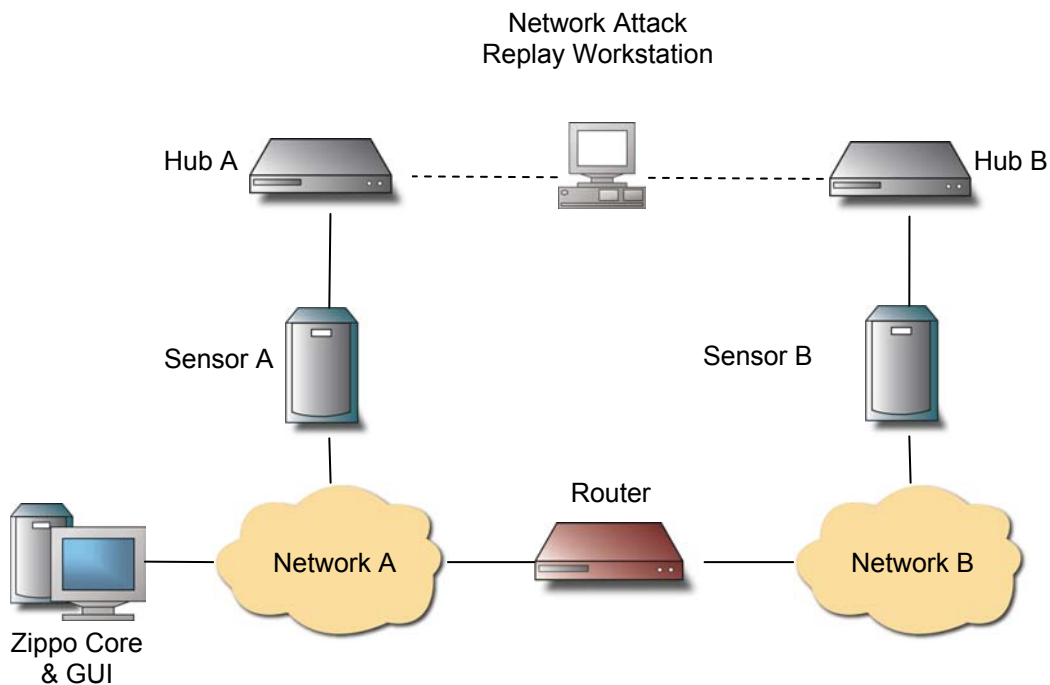


Figure 12. Network Topology of Experiment Setup

In particular, three scenarios are tested:

- i. Sensor A alone detects a network attack.
- ii. Sensors A and B detect the same network attack concurrently.
- iii. Sensors A and B detect different network attacks concurrently.

These experiments are explained in more detail in later chapters.

## B. NETWORK ATTACK REPLAY

In their thesis research, Marinovich and Walch [11] made use of the 1999 DARPA Intrusion Detection Evaluation Data Set from the M.I.T Lincoln Laboratory for their experiments. Figure 13 shows the diagram of the simulation network that was set up by Lincoln Laboratory. The monitored or “inside” network consisted of machines with IP prefixes 172.16.112.\* - 172.16.118.\*.

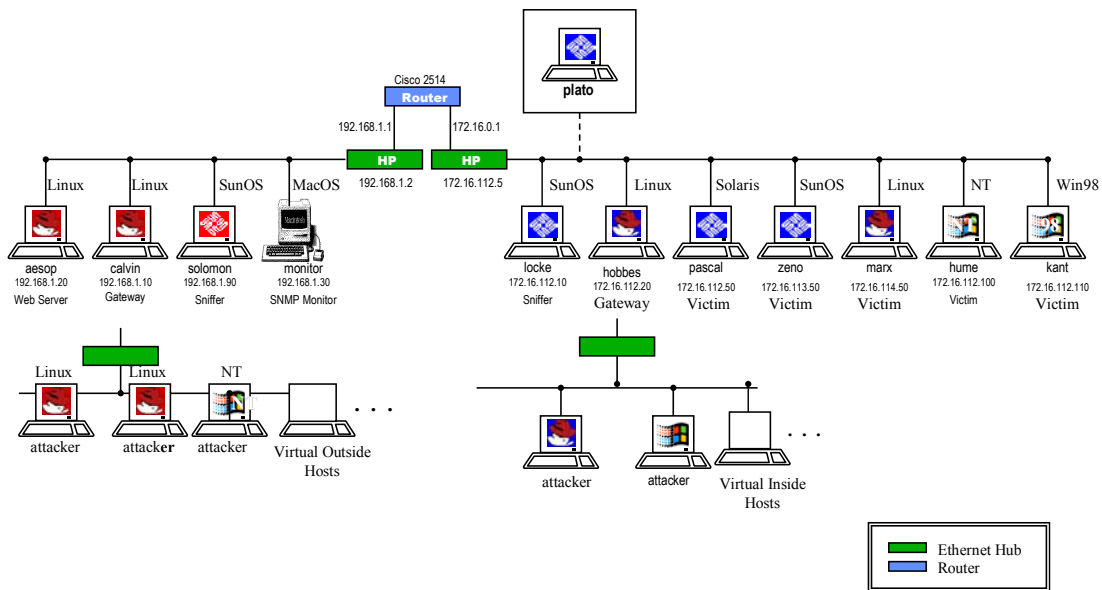


Figure 13. DARPA Simulation Network (Source:[8]).

The data set contains recorded network traffic, which includes scripted attacks conducted from both inside and outside of the monitored network. Marinovich and Walch [11] downloaded the data set and further edited them to obtain *tcpdump* files containing specific attacks.

In this thesis, three of the *tcpdump* files are used to simulate Mailbomb, Smurf and Apache2 attacks, which are all denial-of-service attacks. The purpose of choosing these attacks is to test the ability of Zippo in the analysis of SMTP, ICMP and HTTP traffic.

### **1. Mailbomb Attack**

The Mailbomb attack is contained in the file named *42155148.tcpdump*. In this attack, thousands of machines flood the email server with e-mail messages in an attempt to overwhelm the server, which has the IP address 172.16.114.50. These malicious network packets are either authentications to the email server using TCP port 113 or actual mail transfer using TCP port 25.

The file *42155148.tcpdump* contains 19 minutes worth of network traffic, with the Mailbomb attack being launched five minutes from the start of the packet stream. The attack lasts for about 10 minutes.

### **2. Smurf Attack**

The Smurf attack uses spoofed broadcast ping messages to flood a target system. The attacker sends a large amount of ICMP ping packets to IP broadcast addresses, and spoofs the source address by masquerading as the victim. The result is that every machine on the broadcast network will send an ICMP reply packet to the victim, thus overwhelming the victim.

The file *41213446.tcpdump* contains the Smurf attack, which has 10 minutes worth of recorded network traffic in total. The Smurf attack starts after 5 minutes of the network traffic replay, and lasts only for 15s. The IP address of the victimized machine is 172.16.112.50, and the attack occurs when multiple computers bombard the victim with ICMP reply packets.

### **3. Apache2 Attack**

In the Apache2 attack, the victim, with IP address 172.16.114.50, is a web server that is subjected to numerous concurrent HTTP requests (at TCP port 80) from various attack machines.

The attack is contained in the file *51140100.tcpdump* which reproduces 21 minutes worth of network traffic. The Apache2 attack appears 5 minutes from the start of the network traffic replay and lasts for 11 minutes.

## **C. SYSTEM SETUP**

The Zippo core component is installed on a Sun Microsystems Sun Blade 2000 Workstation, which runs on dual Ultra-Sparc III processors and 4 Gigabytes of RAM in a Solaris 9 operating environment. The GUI component is co-located with the core component. Two Dell Systems servers running on Red Hat Linux 8.0 operating system are installed with the Zippo sensor component and are named sensors A and B. The network attack replay workstation is a Dell Dimension 4100 running on Red Hat Linux 8.0 operating system, which has been pre-installed with the program, TCPReplay, and has two network interface cards.

All three components of Zippo – the core, sensor and GUI, are packaged in a single jar file (*zippo.jar*). The file contains Java class files for the core and the GUI, as well as the source codes for all the components. The installation of each component is simple, which only requires the files to be unpackaged in a chosen directory, using the command:

```
#jar xvf zippo.jar
```

### **1. Core Component**

The core component is installed in the */export/home/usc/zippo* directory on the Sun Blade 2000. As Zippo Core is written in Java code, it requires Java Run Time in its execution. The Sun Blade 2000 has Java 2 SDK 1.4.2 installed for this purpose. The commands to set the environment and start the core component are:

```
#ZIPPO_BASE=/export/home/usc/zippo
```

```
#export ZIPPO_BASE
```

```
#./start_core
```

## **2. Sensor Component**

The sensor components are installed in the */usr/zippo* directory on both Dell Systems servers. The */etc* subdirectory contains the configuration file (sensor.conf), where the IP address of the machine hosting the core component and the socket for communications have to be specified. The commands to set the environment and start the sensor component are:

```
#ZIPPO_BASE=/usr/zippo  
  
#export ZIPPO_BASE  
  
#./sensor
```

## **3. GUI Component**

Like the core component, the GUI component requires Java 2 SDK 1.4.2 or above to function. In addition, Java 3D has to be installed to display the 3-D graphics of the Thermal Canyon and Thermal Tower. The commands to set the environment and start the GUI component are:

```
#ZIPPO_BASE=/usr/zippo  
  
#export ZIPPO_BASE  
  
#./start_gui
```

This command brings up the ZCC, which is the main interface of the program. ZCC can be executed on multiple machines at the same time. Multiple instances of ZCC can also be run concurrently on the same machine.

## **D. DEFINED PID INSTANCES**

Different PID instances are created for the detection of SMTP, ICMP and HTTP attacks. The decision trees and bucket spaces are constructed based on reference [\[11\]](#), and the PID instances are named SMTP, ICMP and HTTP respectively. The top-level decision node of all three PID instances separate network packets according to their origin (i.e. if they originate from the monitored network.)

All three PID instances are multi-tier decision trees with eight buckets each. Each bucket is initialized with 5 balls, and can have a minimum of 0 balls and a maximum of 10 balls.

### 1. SMTP PID Instance

The SMTP PID instance is a 3-tier decision tree, as shown in Figure 14. The lower level decision nodes further differentiate the packets by their ports. Table 1 explains what each bucket denotes.

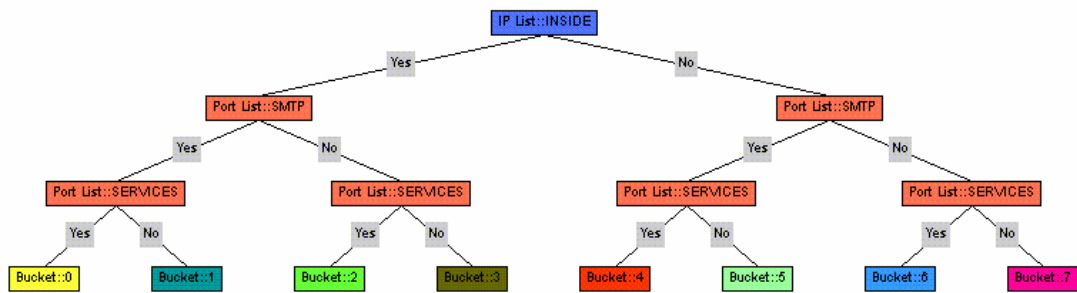


Figure 14. SMTP PID Instance.



Bucket No.	Classification
0	N.A.
1	Insider IP address with TCP ports no. 25, 110, 113 or 161
2	Insider IP address with TCP port no. lower than 1024, excluding 25, 110, 113 and 161.
3	Insider IP address that does not have TCP port no. lower than 1024.
4	N.A.
5	Outsider IP address with TCP ports no. 25, 110, 113 or 161.
6	Outsider IP address with TCP port no. lower than 1024, excluding 25, 110, 113 and 161.
7	Outsider IP address that does not have TCP port no. lower than 1024.

Table 1. Denotation of Buckets for SMTP PID Instance. Buckets 0 and 4 are not applicable (N.A.) because they represent classifications that cannot occur (i.e. – a port number 25, 110, 113 or 161 that is at the same time not 25, 110, 113, or 161).

## 2. ICMP PID Instance

The ICMP PID instance is a 4-tier decision tree, which classifies network traffic according to ICMP packet types: error, request and reply (see Figure 15). Table 2 explains what each bucket represents.

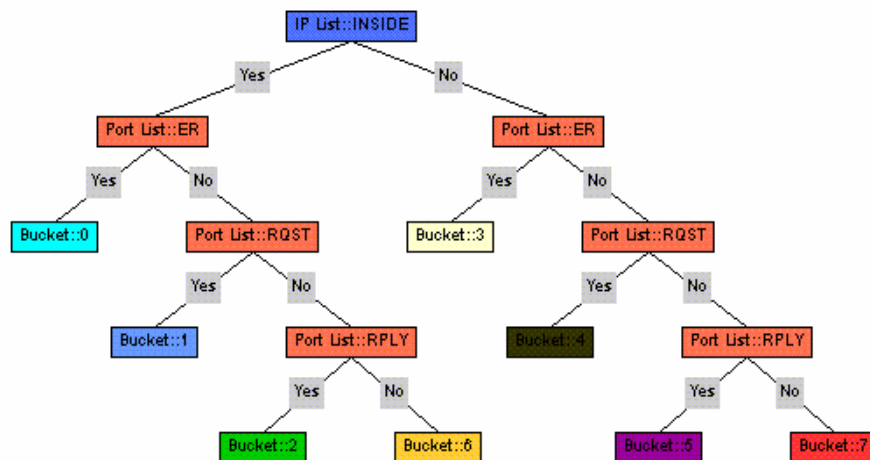


Figure 15. ICMP PID Instance.

Bucket No.	Classification
0	Insider IP address with ICMP type 3, 4, 5, 11 or 12.
1	Insider IP address with ICMP type 8 or 17.
2	Insider IP address with ICMP type 0 or 18.
3	Outsider IP address with ICMP type 3, 4, 5, 11 or 12.
4	Outsider IP address with ICMP type 8 or 17.
5	Outsider IP address with ICMP type 0 or 18.
6	Insider IP address that does not have ICMP type 0, 3, 4, 5, 8, 11, 12, 17 or 18.
7	Outsider IP address that does not have ICMP type 0, 3, 4, 5, 8, 11, 12, 17 or 18.

Table 2. Denotation of Buckets for ICMP PID Instance.

### 3. HTTP PID Instance

The HTTP PID instance is also a 3-tier decision tree, with buckets that are designed to sort traffic into HTTP and non-HTTP traffic, as shown in Figure 16. Table 3 explains the denotation of each bucket.

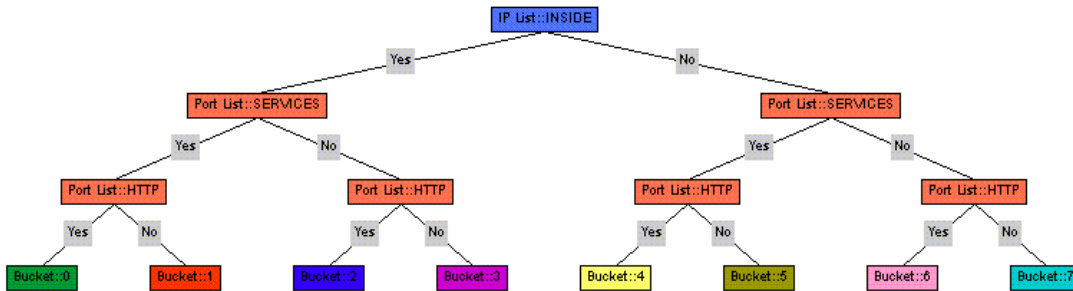


Figure 16. HTTP PID Instance.

Bucket No.	Classification
0	N.A
1	Insider IP address with TCP port no. lower than 1024, excluding 80 and 443.
2	Insider IP address with TCP port no. 80 or 443.
3	Insider IP address that does not TCP port no. lower than 1024.
4	N.A
5	Outsider IP address with TCP port no. lower than 1024, excluding 80 and 443.
6	Outsider IP address with TCP port no. 80 or 443.
7	Outsider IP address that does not TCP port no. lower than 1024.

Table 3. Denotation of Buckets for HTTP PID Instance. Similar to table 1, buckets 0 and 4 represent classifications that cannot occur.

## E. CONFIGURATION PARAMETERS FOR CORE

Experiments were conducted varying the parameters for configuration of the core to investigate the effect on the thermal canyon displays. In this side experiment, sensor A is activated to listen for network intrusions, while the SMTP PID instance was started on Zippo core to analyze the data received from sensor A. From the ZCC, the core was configured with a SF fixed at 0.7 and the number of displayed states at 100. The number of time slices and the slide length were varied.

The Mailbomb attack was replayed several times under different configuration settings and the corresponding thermal canyon displays were collected, as shown in Figure 17. All four diagrams show the presence of the Mailbomb attack, as evident from the sudden increase of the number of visited states.

Figures 17(a) and (b) show two thermal canyons with the same time length display of 120s, but with a different number of time slices and slide length.

The slide length parameter affects the computation of the number of states and the number of counts. Figure 17(b) shows a thermal canyon with a higher number of states and a higher number of counts, which enables an administrator to discern the attack profile more easily.

Figures 17(c) and (d) show two thermal canyons with time length displays of 400 secs and 200 secs respectively. The canyons appear more “spiky” compared to Figure 17(b), as more data is packed onto the same display space.

Comparing all four thermal canyons, Figure 17(b) gives the best graphical representation of the same network attack, which shows a relatively “smooth” canyon with a discernible attack profile. Hence the optimum setting for the core component parameters is: number of time slices = 60, slide length = 2, SF = 0.7. This setting is repeated for all subsequent tests.

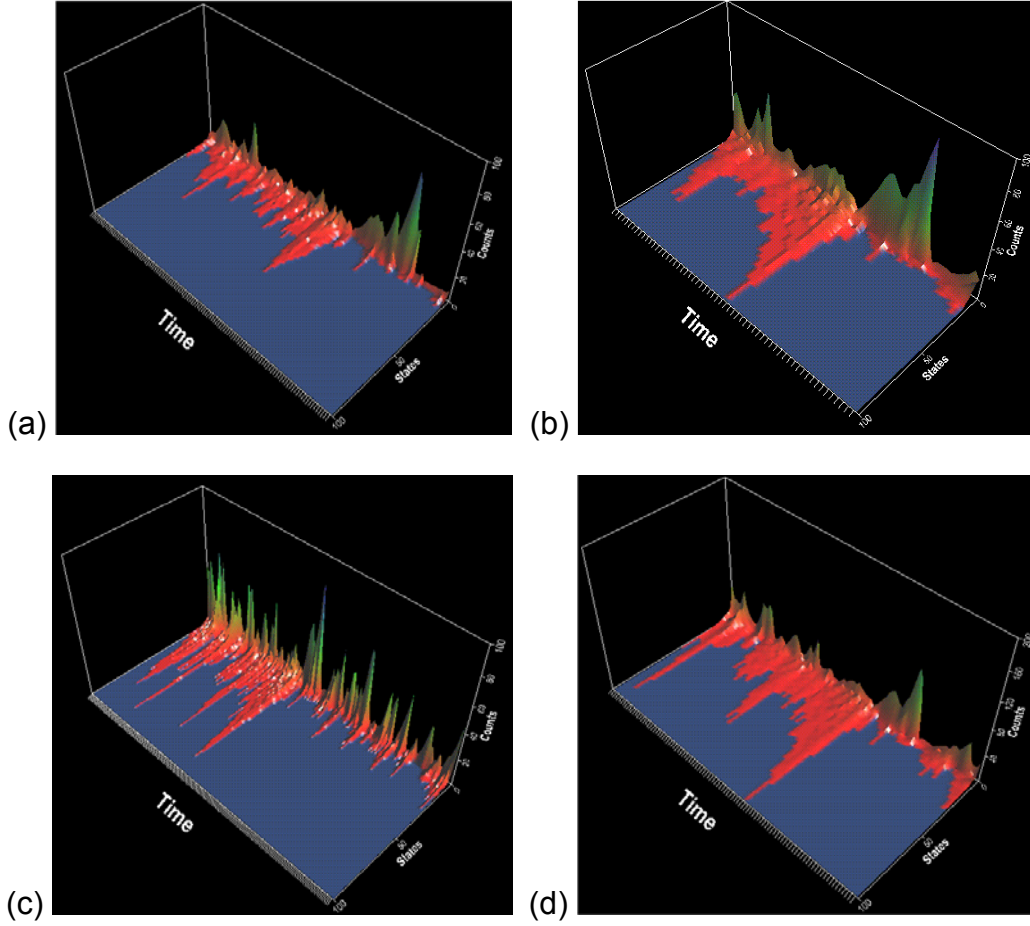


Figure 17. Thermal Canyon Displays for Mailbomb attack with (a) Number of Time Slices = 120, Slide Length = 1, (b) Number of Time Slices = 60, Slide Length = 2, (c) Number of Time Slices = 200, Slide Length = 2, (d) Number of Time Slices = 100, Slide Length = 2.

## F. SUMMARY

This chapter gave an overview of the network and system setups, and explained how two Zippo sensors are set up in the experiment network to simulate a DIDS architecture. The Zippo core is configured with different PID instances to detect Mailbomb, Smurf and Apache2 attacks. These attacks are taken from M.I.T. Lincoln Laboratory data set.

In the next chapter, the Zippo system is tested in the detection of Mailbomb, Smurf and Apache2 attacks.

THIS PAGE INTENTIONALLY LEFT BLANK

## V. EXPERIMENT AND ANALYSIS I – SINGLE SENSOR

### A. EXPERIMENT SETUP

The purpose of this experiment is to test the effectiveness of Zippo in the detection of some known attacks. The three network attacks, the Mailbomb, Smurf and Apache2 attacks are separately sent from the network attack replay workstation to hub A. Sensor A sniffs the packets broadcast by hub A, converts the necessary information into thermalate and sends it to the Zippo core component.

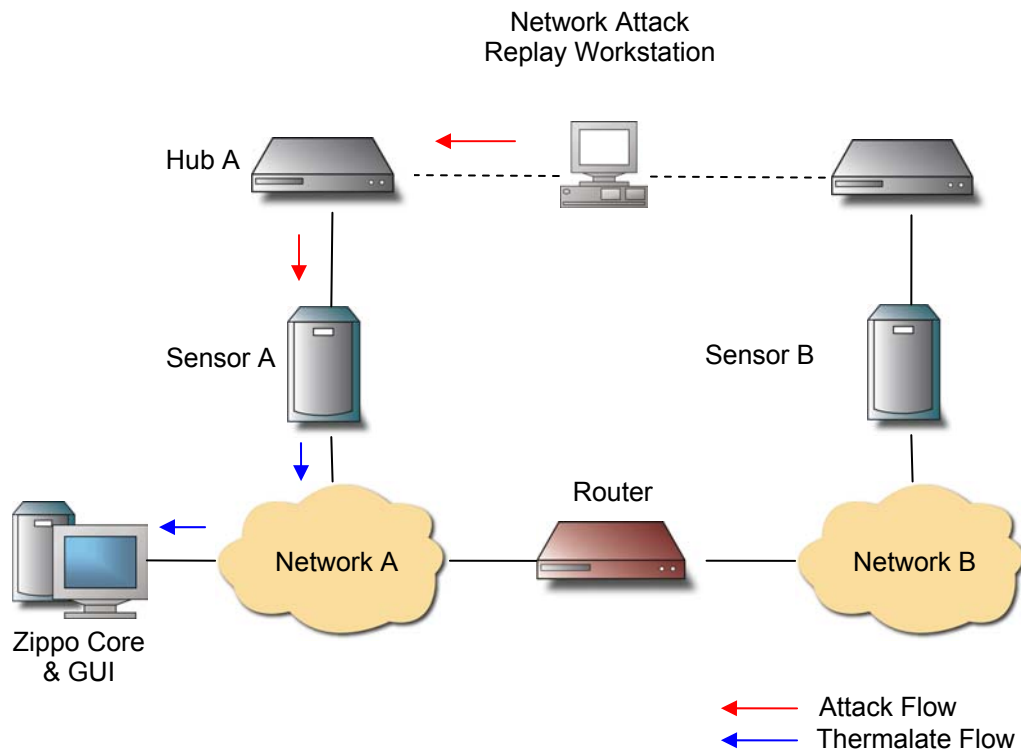


Figure 18. Experiment Setup for a Single Sensor.

### B. MAILBOMB ATTACK

Figures 19(a) and (b) show the thermal canyon and thermal tower displays generated from the SMTP PID instance, when the file *42155148.tcpcdump* containing the Mailbomb attack is replayed.

During the attack, there is a sudden increase in the ball transfer activity between the buckets, as seen from the thermal tower. This translates to more bucket states, which affirms with the sudden increase in visited states on the thermal canyon display. The terrain during the attack is rather flat, as there are relatively low counts of the bucket states. This contrasts with the few bucket states (but with high counts for each state) during the period of normal traffic.

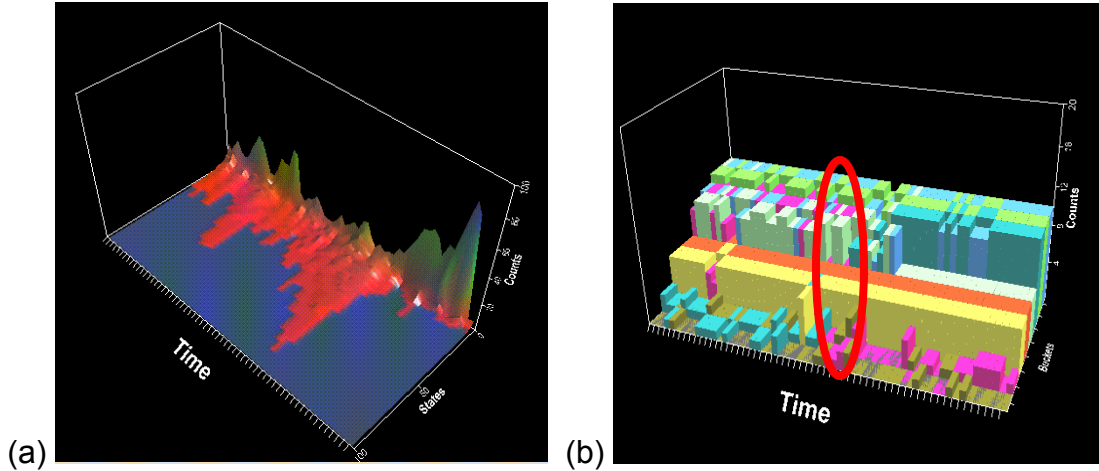


Figure 19. (a)Thermal Canyon and (b)Thermal Tower for SMTP PID Instance during the Mailbomb Attack.

Ethereal is used to further examine the conversation exchanges between the attacker and the victim during the attack. Figure 20 shows the observed patterns of the conversation exchanges and Table 4 explains the flow of ball transfer between the defined buckets of the SMTP PID instance.



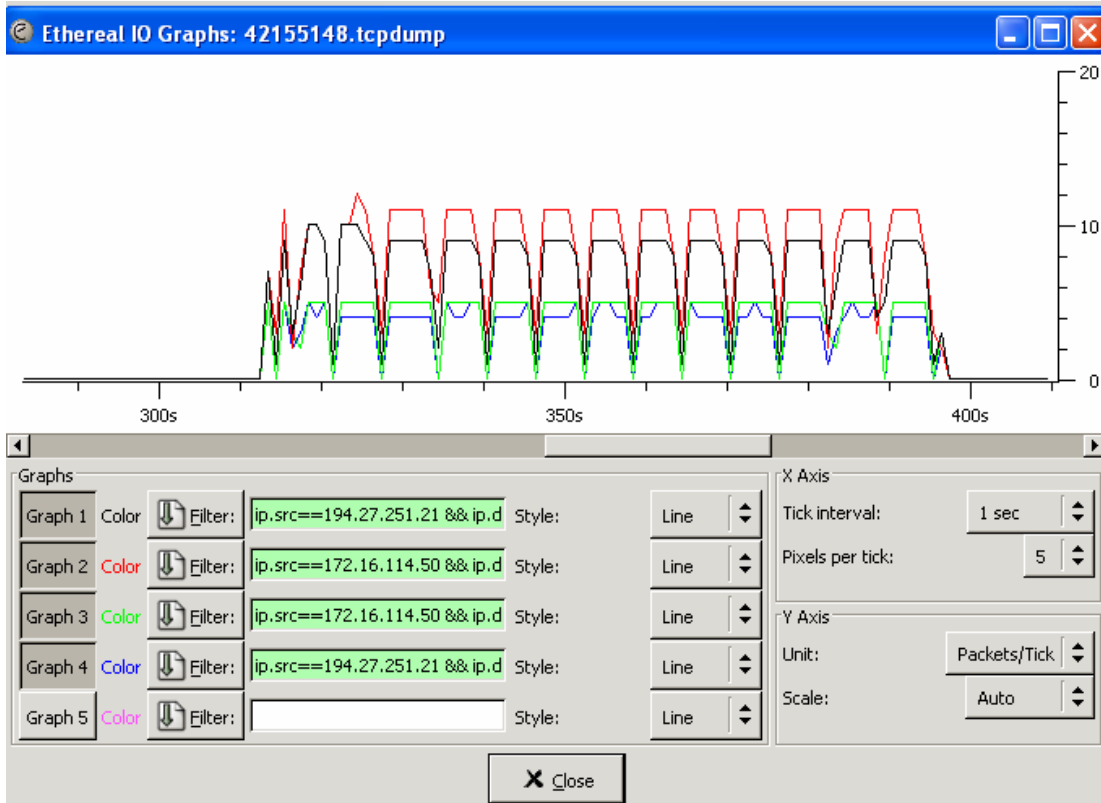


Figure 20. Patterns of Conversation Exchanges during the Mailbomb Attack.

Source Address	Source Port	Destination Address	Destination Port	Line Color	Ball Transfer From
194.27.251.21	Ephemeral	172.16.114.50	25	—	Bucket 7 to 1
172.16.114.50	25	194.27.251.21	Ephemeral	—	Bucket 1 to 7
172.16.114.50	Ephemeral	194.27.251.21	113	—	Bucket 3 to 5
194.27.251.21	113	172.16.114.50	Ephemeral	—	Bucket 5 to 3

Table 4. Flows of Ball Transfer between Defined Buckets of SMTP PID Instance during the Mailbomb Attack.

As seen from Figure 20, there are more SMTP port 25 packets originating from 172.16.114.50 to 194.27.251.21, than the other way round. Thus more balls are transferred from Bucket 7 (maroon-colored bar) to Bucket 1 (turquoise-

colored bar), resulting in fewer balls in Bucket 7 compared to Bucket 1. Similarly, there are more ball transfers from Bucket 3 (lime green) to Bucket 5 (light green) than the other way round. The thermal tower shows fewer balls in Bucket 3 compared to Bucket 5.

This is dramatically reflected in the thermal canyon because several more state combinations are being visited due to the activity in bucket 7 and bucket 5. Prior to the attack there was very little activity in bucket 7. The attack induces large variations in the bucket flow – indicated by the turbulence in the thermal towers – that cause an increase in state entropy and is reflected by the run out in the thermal canyon.

### C. SMURF ATTACK

The file *41213446.tcpdump* which contains the Smurf attack produces the thermal canyon and thermal tower in Figures 20(a) and (b) when it is replayed and the ICMP PID instance is activated.

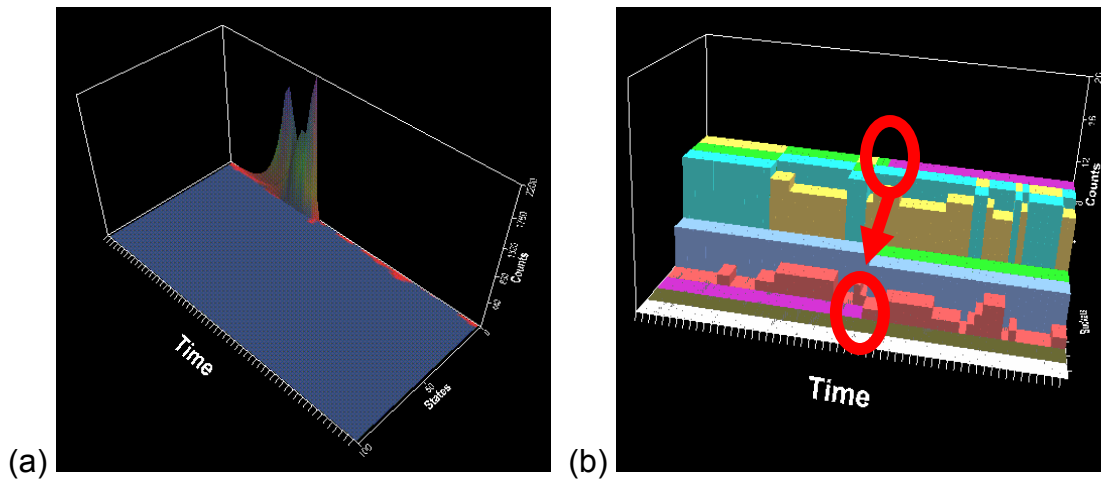


Figure 21. (a)Thermal Canyon and (b)Thermal Tower for ICMP PID Instance during the Smurf Attack.

The most indicative sign of an anomaly in the behavior of the network is the sudden peak on the thermal canyon, jumping from virtually 0 to 2200 counts. During the attack, the ICMP reply packets (with type 0) from the various attackers

to the victim translate to ball transfers from Bucket 5 to 2. This is reflected in the thermal tower as indicated by the red circles, where Bucket 5 (maroon-colored bar) is empty during the attack, while Bucket 2 (green-colored bar) is filled with the maximum number of balls.

Due to the characteristic of the Smurf attack, ball transfer between the buckets is only one way, i.e. from Bucket 5 to 2. After the buckets hit the boundary conditions, they remain at this state throughout the duration of the attack due to the large number of malicious packets sent from the attackers to the victim. This results in few bucket states, but with very high counts.

#### D. APACHE2 ATTACK

Figures 22(a) and (b) show a snapshot of the thermal canyon and thermal tower displays during an Apache2 attack when the file *51140100.tcpdump* is replayed.

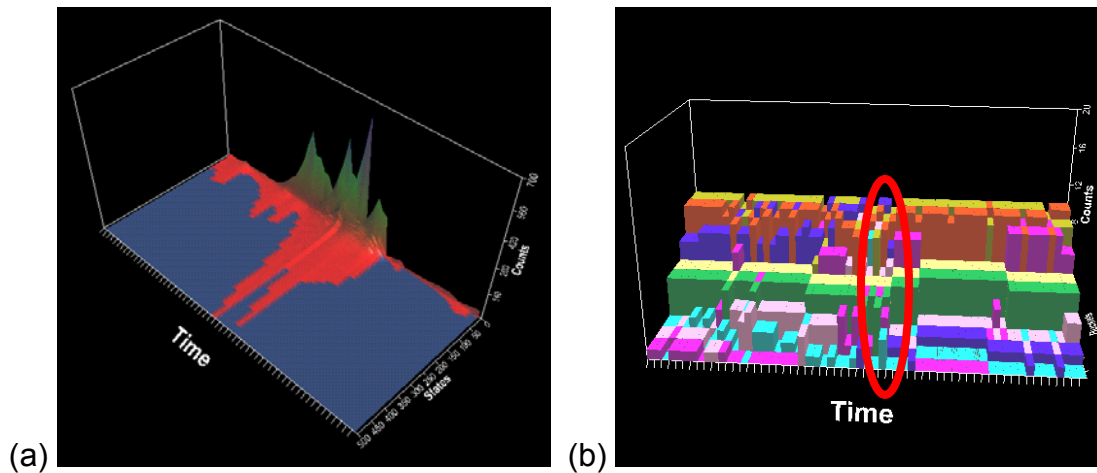


Figure 22. (a)Thermal Canyon and (b)Thermal Tower for HTTP PID Instance during the Apache2 Attack.

The sudden increases in both the number of bucket states and the counts of bucket states on the thermal canyon are matched by the increased variations in ball transfer activities as shown on the thermal tower.

Figure 23 shows the pattern of conversation exchanges between the attacker and the victim using Ethereal, and Table 5 explains the flow of transfer between the buckets. As there are more packets originating from the attacker to the victim than the other way, the number of balls in Bucket 7 (turquoise-colored bar on the thermal tower) is less than Bucket 2 (blue-colored bar). Additionally, since a large number of packets are being sent to the non-HTTP ports, there is high variability in the thermal towers plot and a significant run out in the thermal canyon plot.

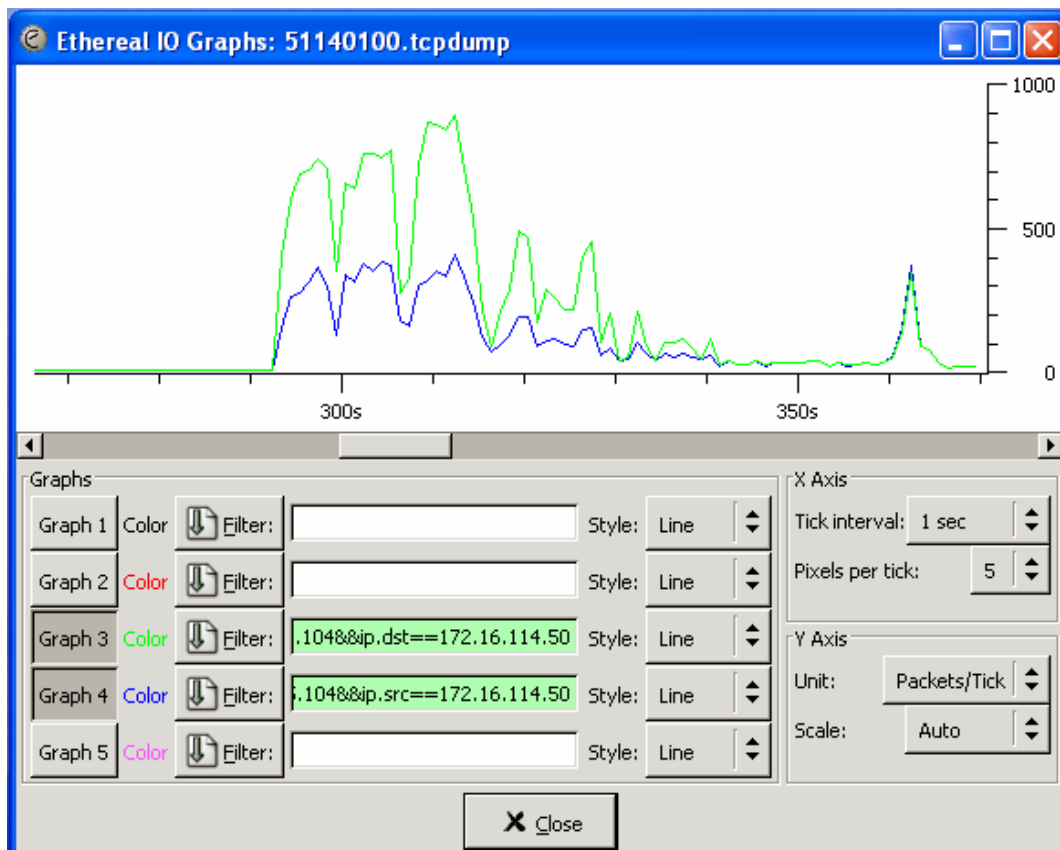


Figure 23. Patterns of Conversation Exchanges during the Apache2 Attack.



Source Address	Source Port	Destination Address	Destination Port	Line Color	Ball Transfer From
152.169.215.104	Ephemeral	172.16.114.50	113		Bucket 7 to 2
172.16.114.50	113	152.169.215.104	Ephemeral		Bucket 2 to 7

Table 5. Flows of Ball Transfer between Defined Buckets during the Apache2 Attack.

## E. SUMMARY

The results above show that Zippo is effective in the analysis of SMTP, ICMP and HTTP traffic, as demonstrated in the successful detection of Mailbomb, Smurf and Apache2 attacks. Indications of anomalies in the network behavior are most obvious when the thermal canyon shows either a sudden increase in the number of states visited or a sharp increase in the number of occurrences of visited states over a prolonged period of time.

The next chapter describes experiments that investigate the response of Zippo when two sensors detect the same type of attacks.

THIS PAGE INTENTIONALLY LEFT BLANK

## VI. EXPERIMENT AND ANALYSIS II – DUAL SENSORS WITH THE SAME ATTACK

### A. EXPERIMENT SETUP

In this experiment, the same network attack is sent out of the two network interfaces of the network attack replay workstation at the same time. The purpose is to investigate the response of the Zippo when two sensors detect the same network attack at the same time. Figure 24 shows the experiment setup. Both sensors A and B sniff the same packets and produce the same set of thermalate which is routed to the Zippo core. The Zippo core then performs an aggregated analysis on the thermalate received from both sensors.

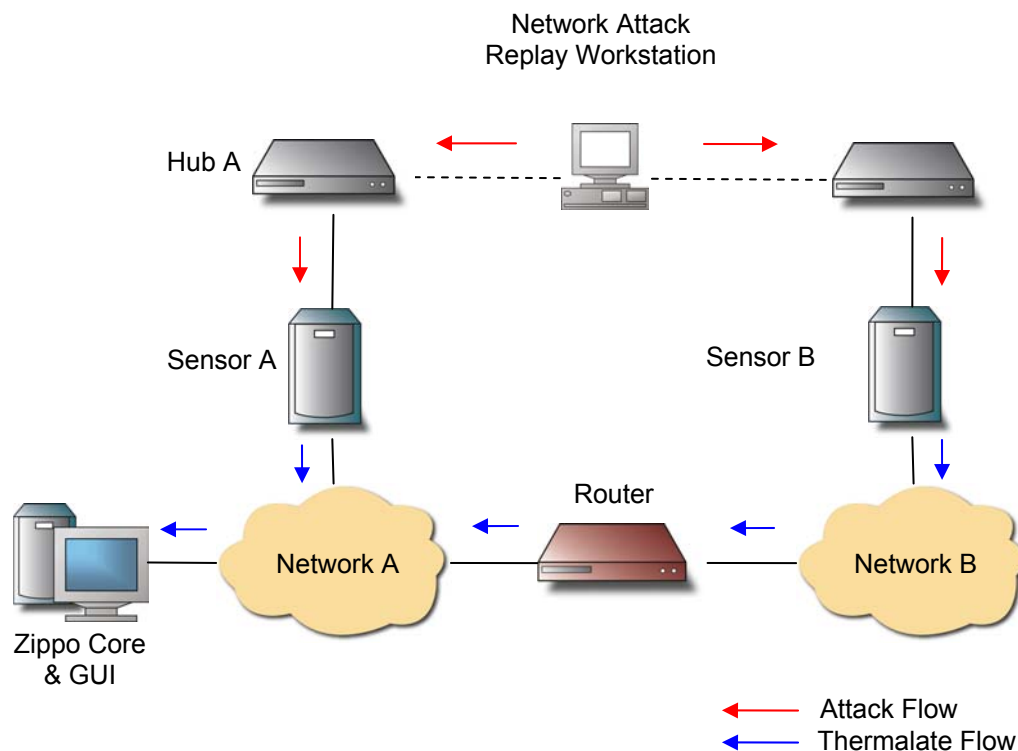
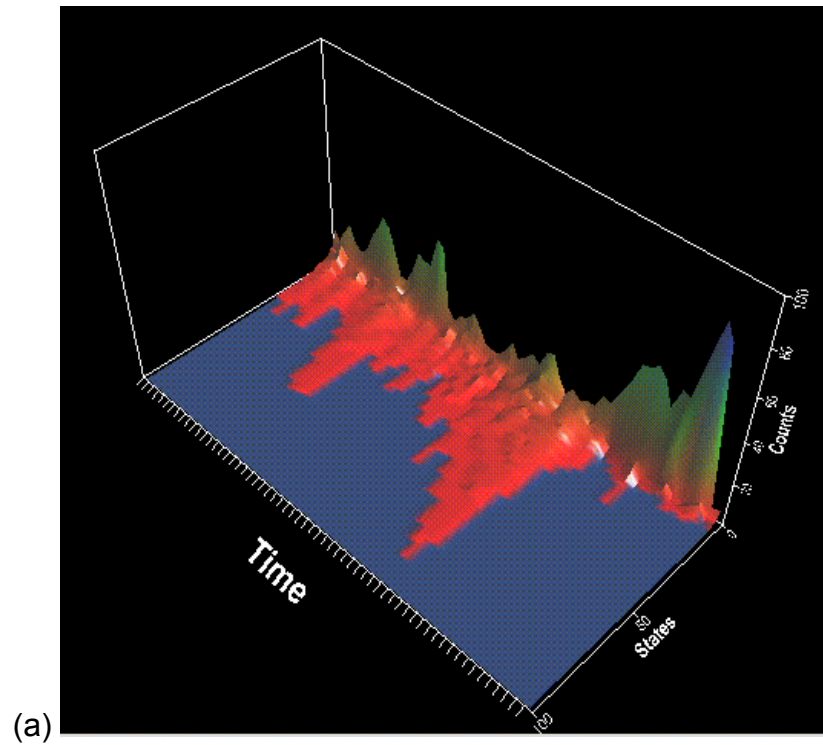


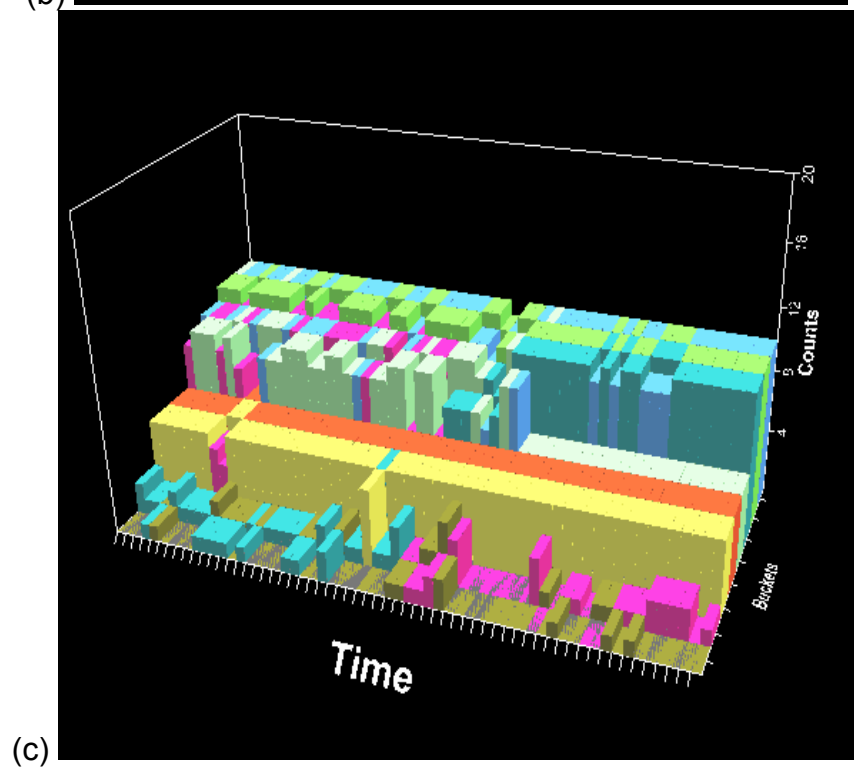
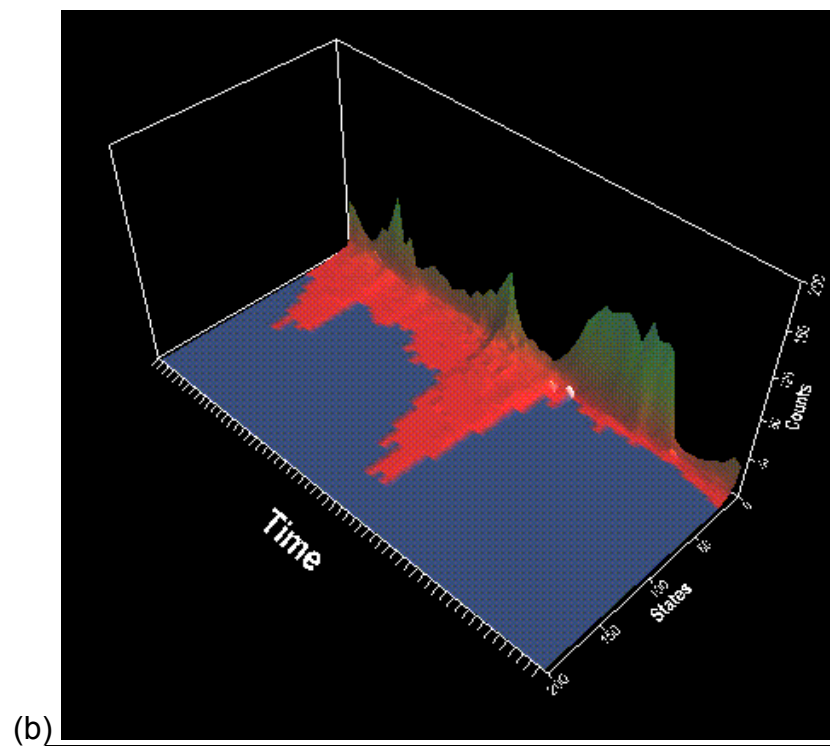
Figure 24. Experiment Setup for Dual Sensors with the Same Attack.

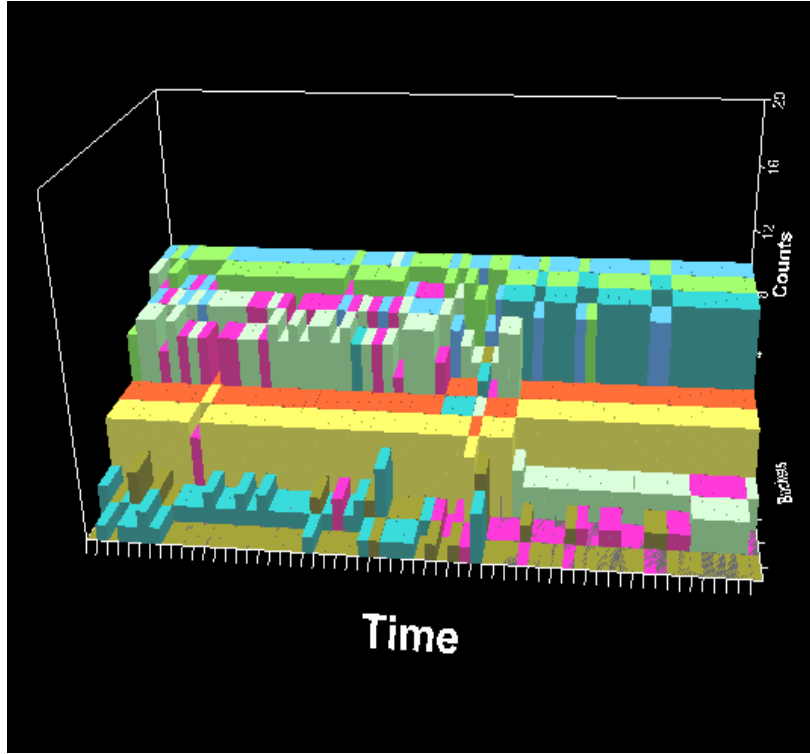
## B. MAILBOMB ATTACK

The thermal canyon and tower on Figures 25(a) and (c) have been repeated from Chapter V for ease of comparison, which are produced from the detection of a Mailbomb attack by sensor A alone.









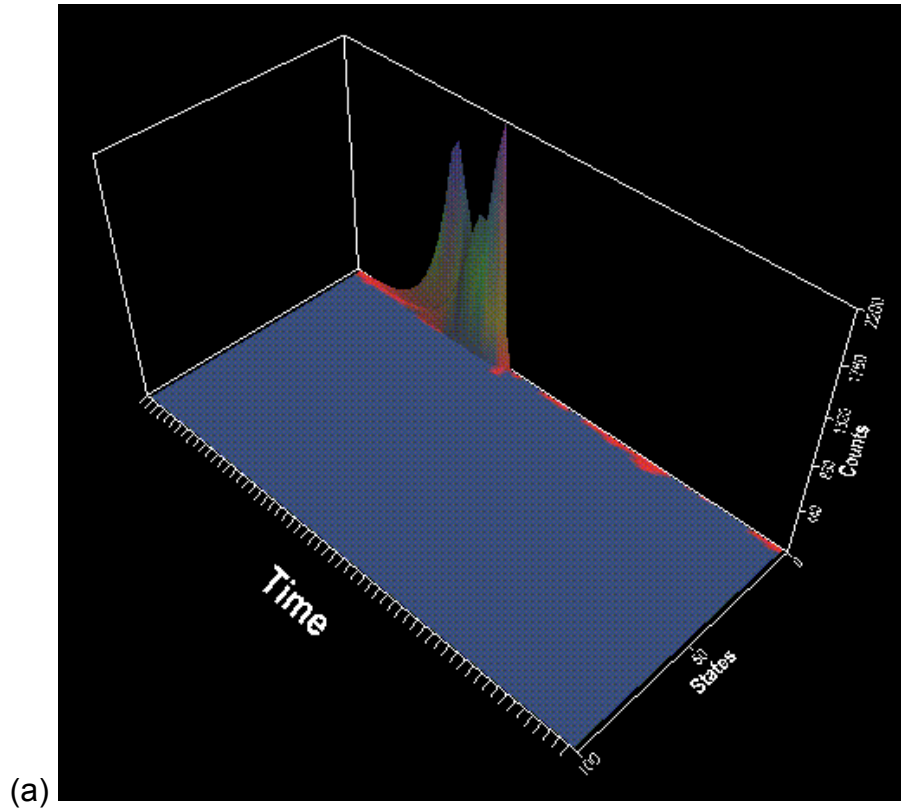
(d)  
Figure 25. Thermal Canyon and Tower Displays for (a) and (c) Single Sensor, (b) and (d) Dual Sensors during a Mailbomb attack.

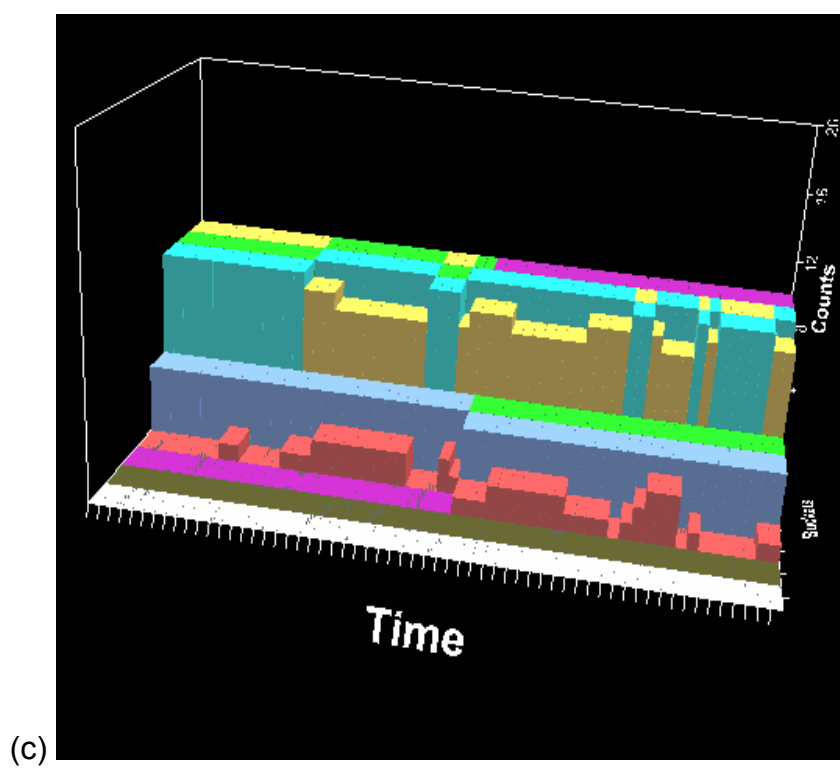
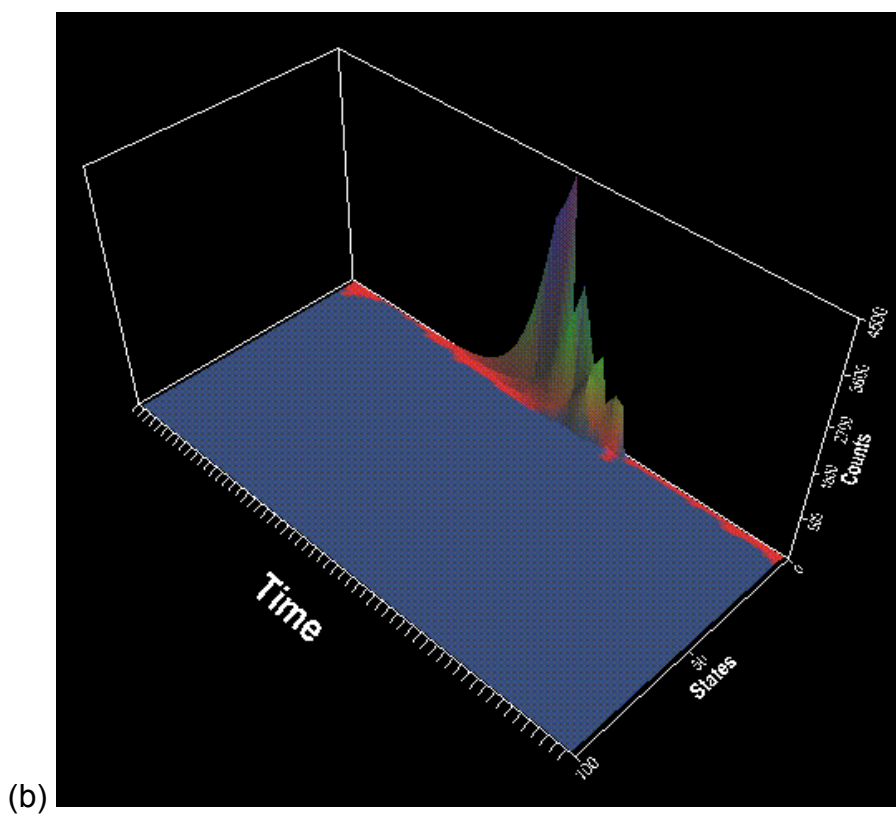
Figures 25(b) and (d) show the results of the aggregated analysis when both sensors A and B detect the Mailbomb attack at the same time. This is achieved by having the network attack reply workstation replay two instances of *42155148.tcpdump* at the same time. The Zippo core receives two sets of thermalate with the same information, and as the thermal tower in Figure 25(d) shows, the ball transfers between the buckets have increased compared to Figure 25(c).

Comparing the thermal canyons, it can be observed the shapes of both canyons are very similar. The difference between the two canyons is that the number of bucket states and the counts for each bucket state have doubled in Figure 25(b). For instance, the highest peak on the canyon floor in Figure 25(a) indicates 80 visited states. The corresponding peak in Figure 25(b) is double that of Figure 25(a), at approximately 160 states.

### C. SMURF ATTACK

The thermal canyon and tower in Figures 26 (a) and (c) reflect a Smurf attack detected by sensor A alone, while Figures 26(b) and (d) show the response when the same Smurf attack is detected by both sensors A and B.





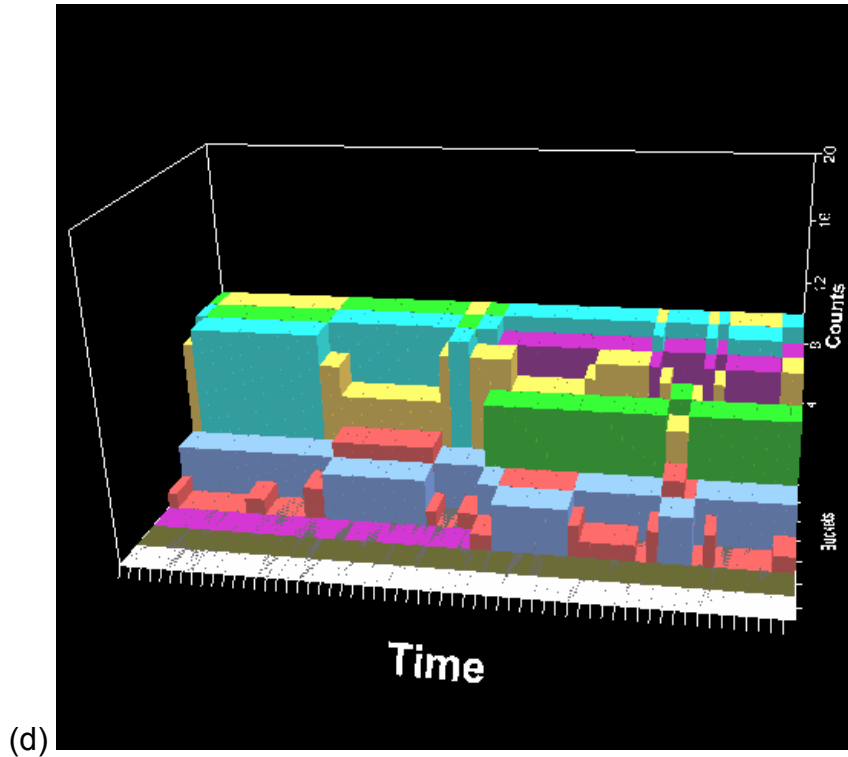


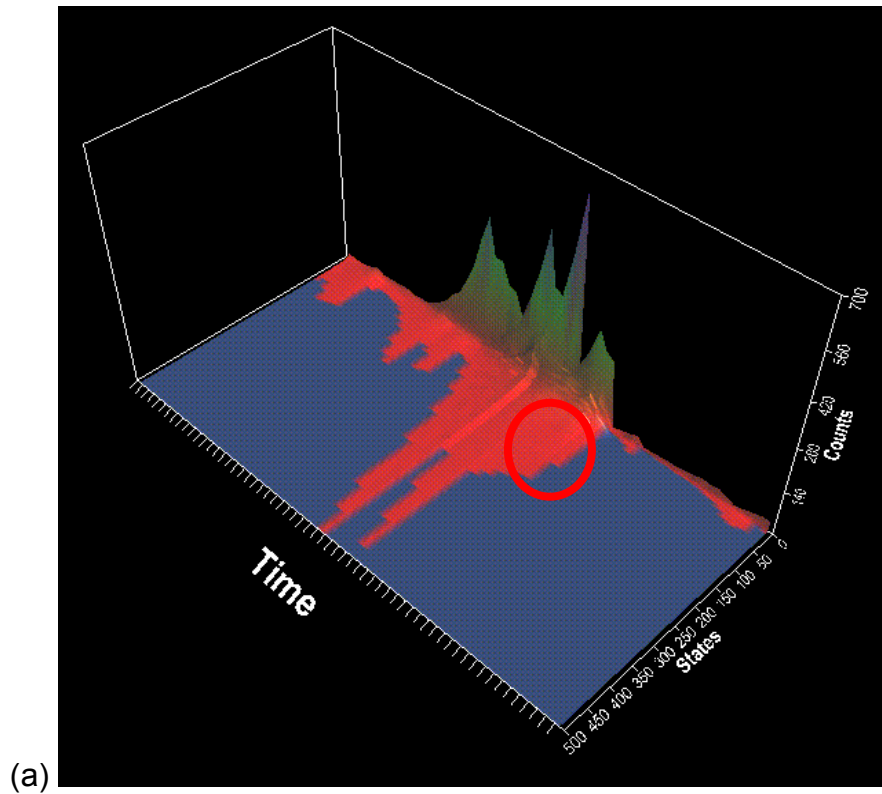
Figure 26. Thermal Canyon and Tower Displays for (a) and (c) Single Sensor, (b) and (d) Dual Sensors with Smurf attacks.

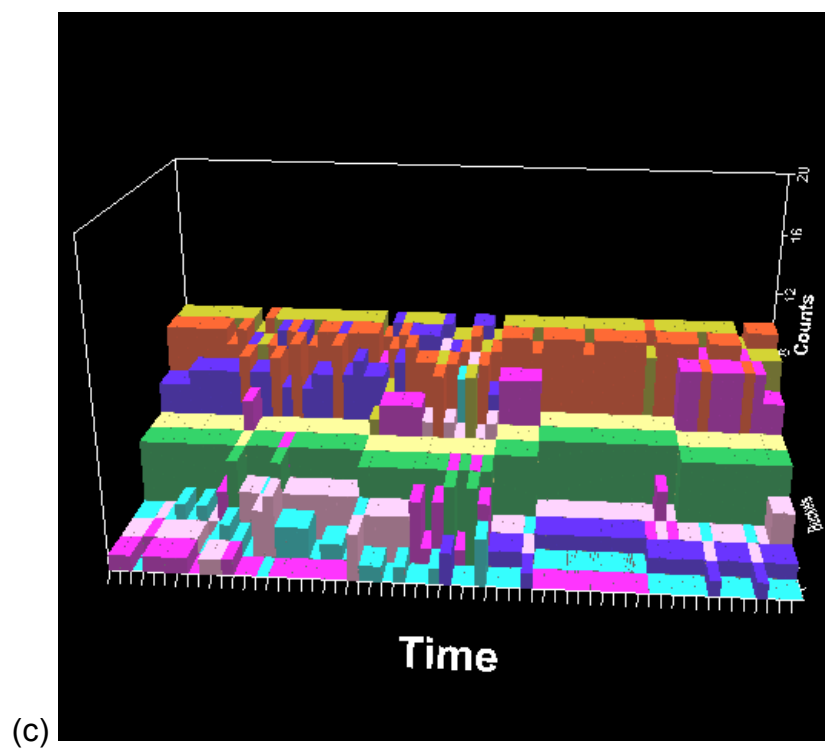
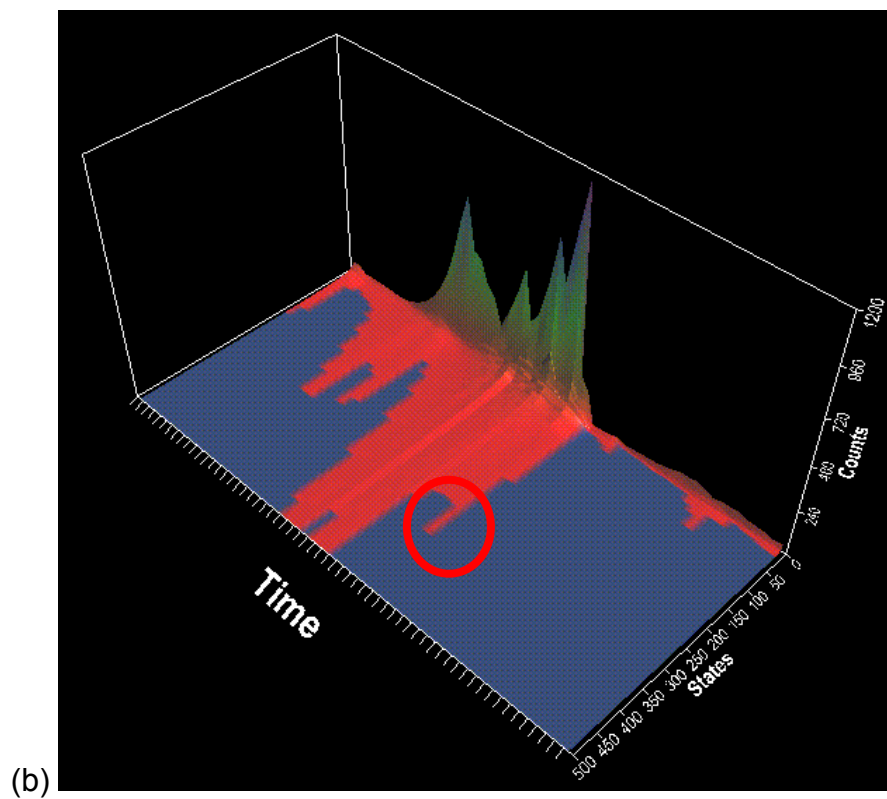
It is easily observed from the thermal towers, that ball movement between buckets increased considerably when two attacks are launched (see Figure 26(d)). Although there are few bucket states on both thermal canyons, Figure 26(b) shows more spreading of redness on the thermal canyon floor, indicating more bucket states compared to Figure 26(a). As there are double the number of malicious packets, the peak number of counts on the thermal canyon in Figure 26(b) is also doubled.

#### D. APACHE2 ATTACK

Figures 27(a) and (c) show the results from the detection of an Apache2 attack by sensor A. When both sensors detect the same attack, a similar thermal canyon landscape is produced. Figures 27(b) shows a thermal canyon that appears to be an enlarged version of Figure 27(a), with a much wider and longer canyon floor, and higher peaks. The highest peak in Figure 27(a) is about 600

counts, while the highest peak in Figure 27(b) is almost 1200, double that of Figure 27(a). Comparing the two red circles in Figures 27(a) and (b), the number of visited states in Figure 27(b) at that point is 300, which is approximately double that of Figure 27(a). This corresponds with more rapid changes in the number of balls in the buckets of the thermal tower in Figure 27(d).





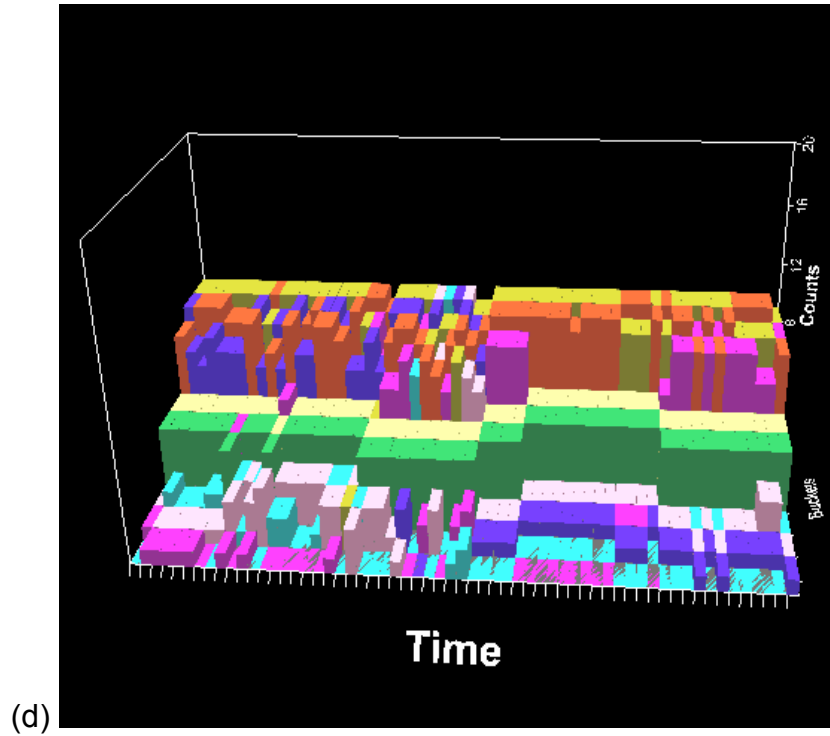


Figure 27. Thermal Canyon and Tower Displays for (a) and (c) Single Sensor, (b) and (d) Dual Sensors with Apache2 attacks.

## E. ANALYSIS OF RESULTS

The results from all three attacks show that when two Zippo sensors detect the same attack concurrently, the thermal canyon produced has the same landscape as when only one sensor detects the attack. However, as there is double the quantity of thermalate produced, the number of bucket states and the counts of each bucket state are increased proportionately.

To use a simple illustration, consider two buckets  $B_A$  and  $B_B$  which represent Nodes A and B respectively. Let there be two balls in each bucket. Suppose A sends a network packet to B, and B replies to A. The state changes, corresponding to the number of balls in the two buckets are as shown in Table 6. Excluding the initial state (2,2), there is one new state, which is (1,3) with only one count.



Event	No. of Balls in B <sub>A</sub>	No. of Balls in B <sub>B</sub>
Initial Condition	2	2
A sends packet to B	1	3
B replies	2	2

Table 6. State Changes with a Single Network Packet from A to B.

If each packet is duplicated, the state changes will be as follows (see Table 7). Excluding the initial state condition (2,2), there are two new states: (1,3) and (0,4). The number of counts for state (1,3) in particular is 2.

Event	No. of Balls in B <sub>A</sub>	No. of Balls in B <sub>B</sub>
Initial Condition	2	2
A sends packet to B	1	3
A sends packet to B (duplicate)	0	4
B replies	1	3
B replies (duplicate)	2	2

Table 7. State Changes with Duplicate Network Packets from Node A to B.

This simple example shows how the number of states visited and the number of counts of the visited states can be doubled with double the network packets.

## F. SUMMARY

This chapter investigates the response of the Zippo core component when both its sensors detect the same attack. The conclusion is that, as the Zippo core receives double the amount of thermalate, the thermal canyon and thermal tower display a proportionate increase in the bucket states and state counts.

The next chapter discusses the experiment conducted to investigate the response of Zippo to concurrent different attacks.

## VII. EXPERIMENT AND ANALYSIS III – DUAL SENSORS WITH DIFFERENT ATTACKS

### A. EXPERIMENT SETUP

The purpose of this experiment is to examine the ability of Zippo to deal with different and concurrent network attacks. Figure 28 shows the network attack replay workstation launching two different network attacks at the same time. Sensors A and B detect the different attacks, and send the thermalate to Zippo core. Zippo core performs an aggregated analysis and the results are presented in the following sections. The thermal canyons and thermal towers produced in this experiment are also compared and contrasted with results obtained from earlier experiments.

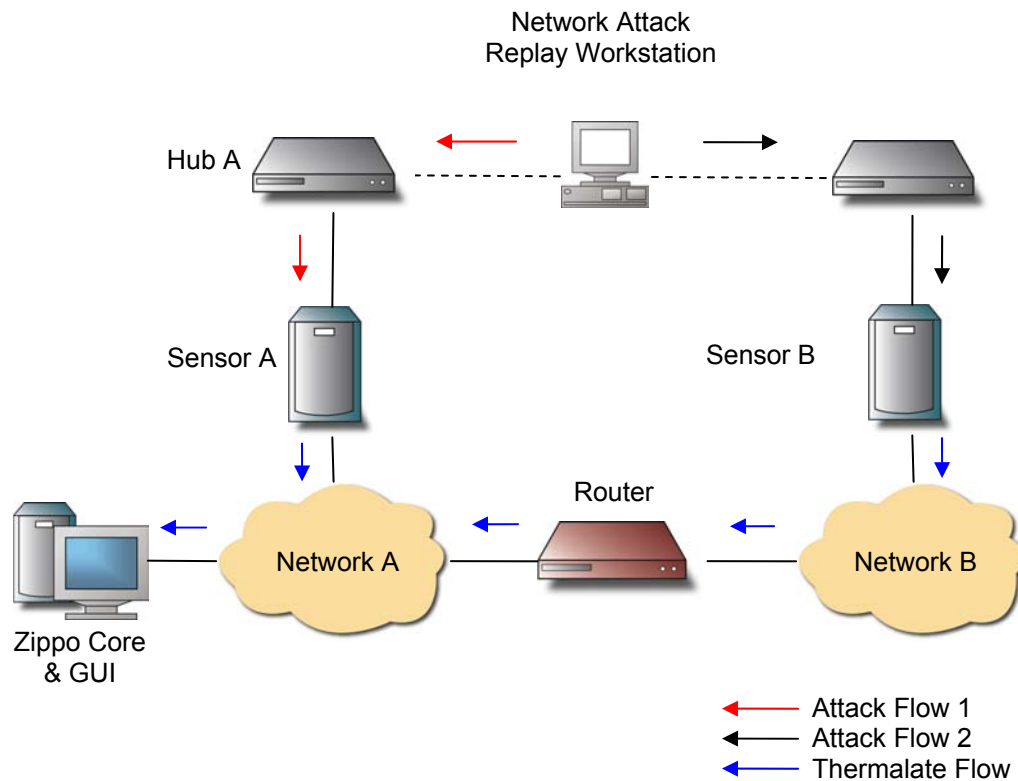


Figure 28. Experiment Setup for Dual Sensors with Different Network Attacks.

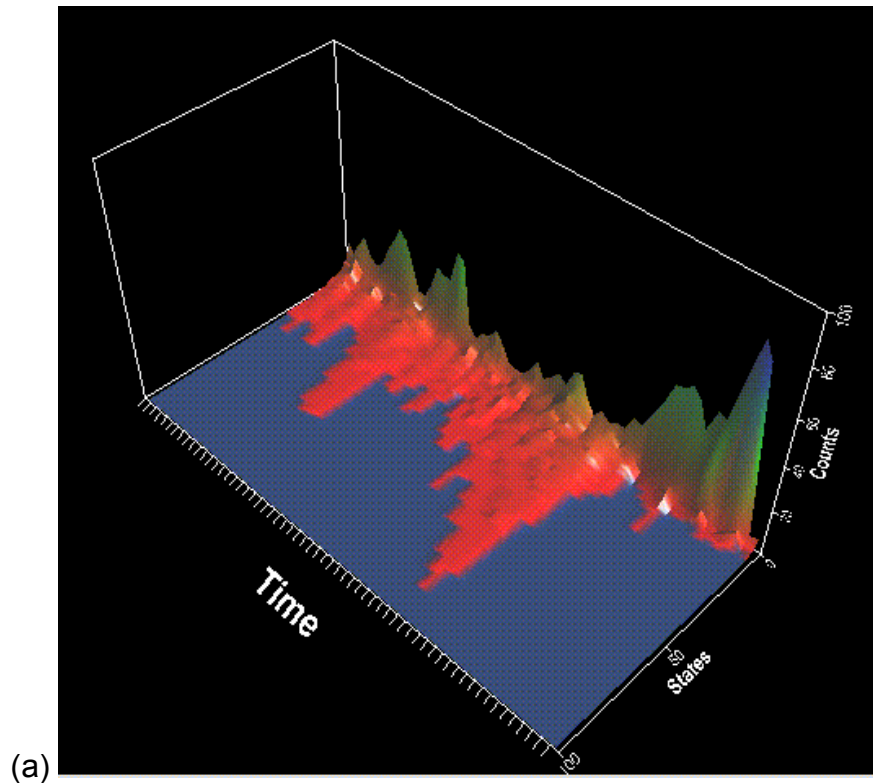
## B. SMURF AND MAILBOMB ATTACKS

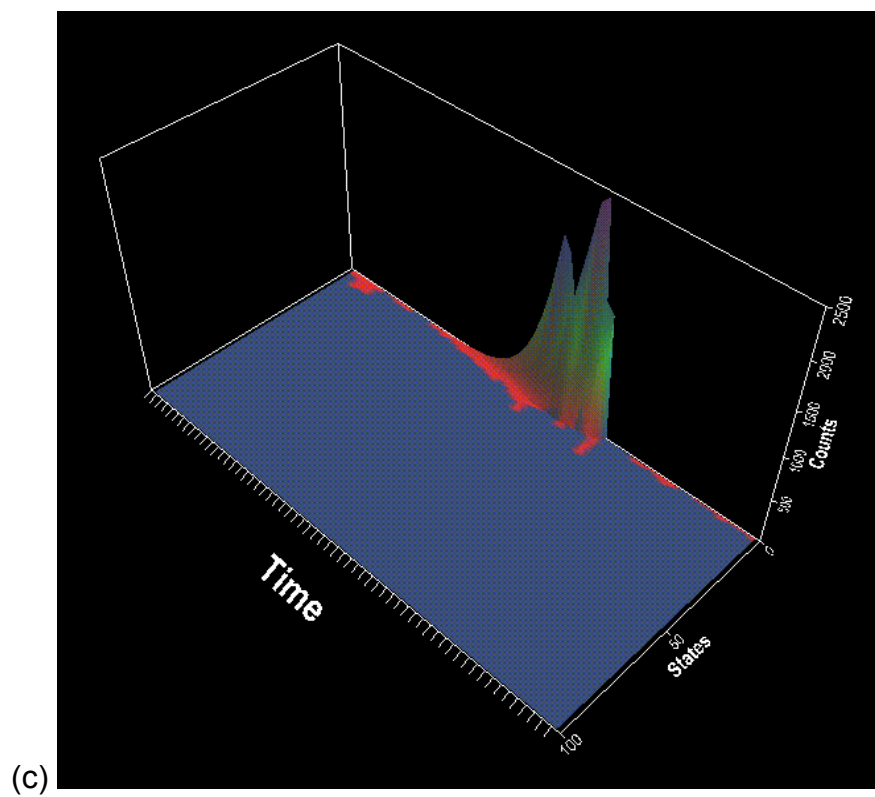
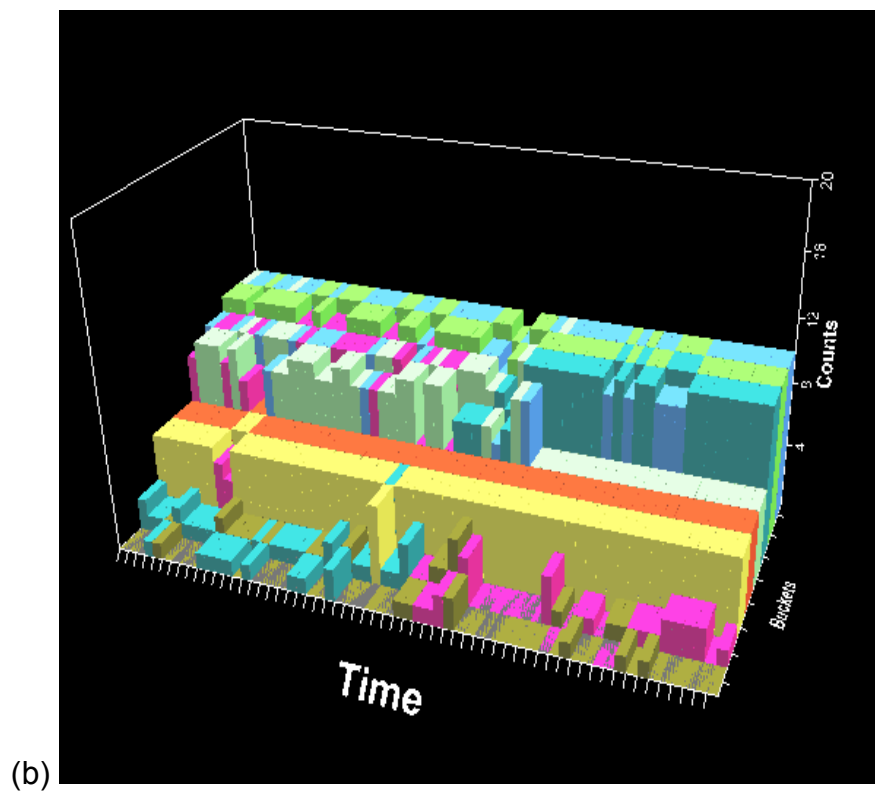
In this section, the network attack replay workstation replays both Smurf and Mailbomb attacks at the same time. The SMTP and ICMP PID instances are activated on the Zippo core to perform an aggregated analysis on the thermalate sent from sensors A and B.

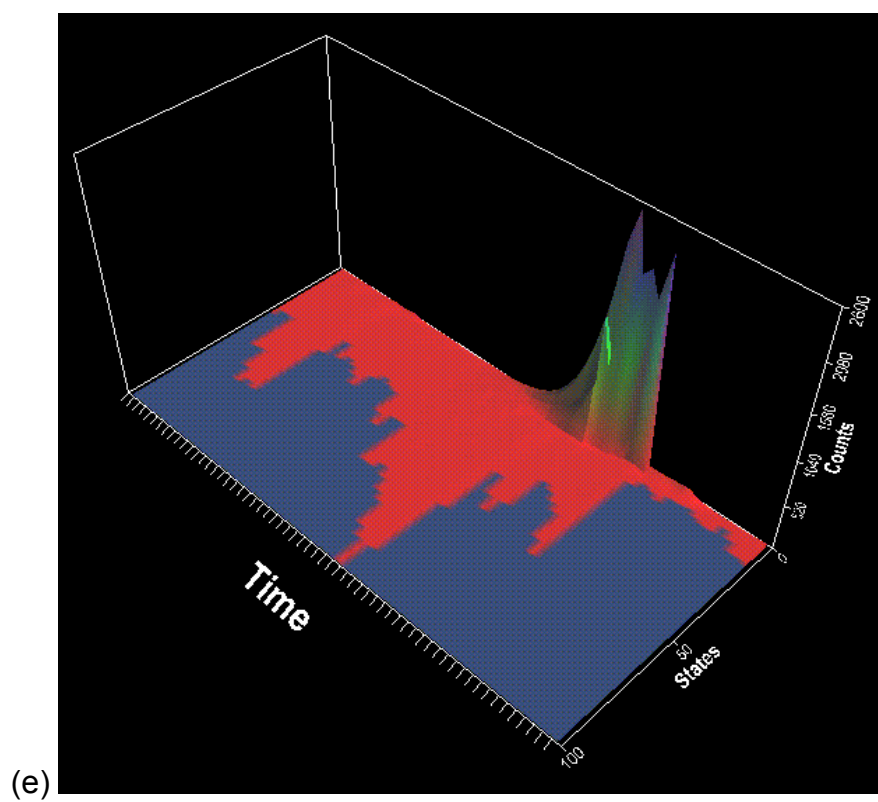
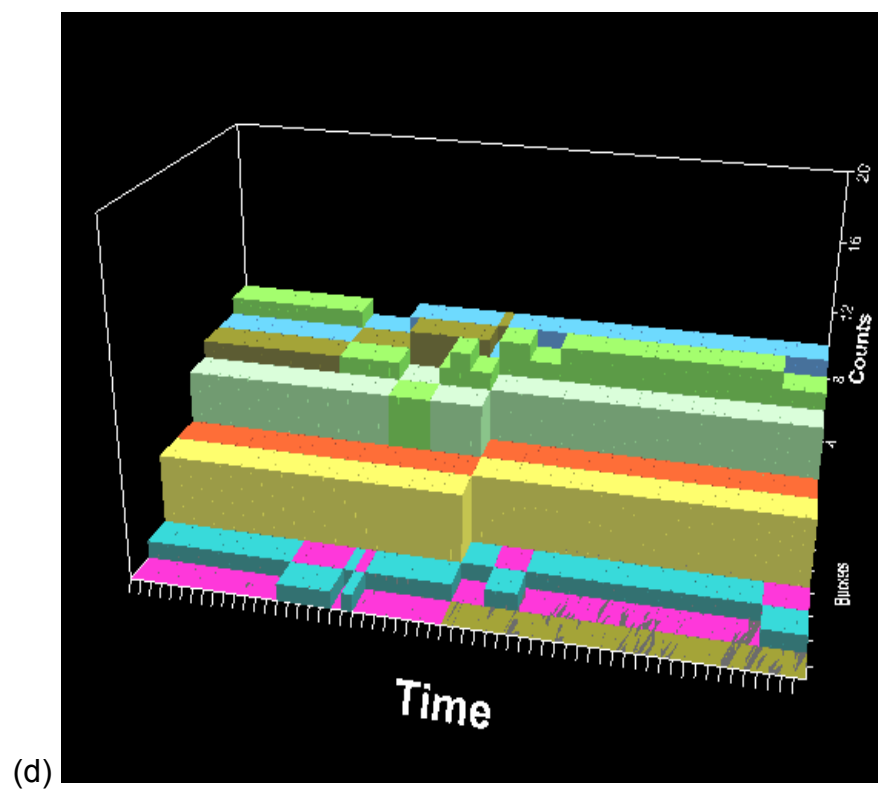
### 1. SMTP PID Instance

Figures 29(a) to (f) show the different thermal canyon and thermal tower displays for the SMTP PID instance when the Mailbomb attack, the Smurf attack and the combined Mailbomb and Smurf attacks are launched respectively.

Looking at Figures 29(c) and (d), during the Smurf attack, the large number of ICMP reply packets from the attackers to the victim result in ball transfers from Bucket 7 (represented by the maroon-coloured bar on the thermal tower) to Bucket 3 (represented by the dark green-colored bar). There are few ball exchanges between the other buckets, which is why there are few bucket states on the thermal canyon.







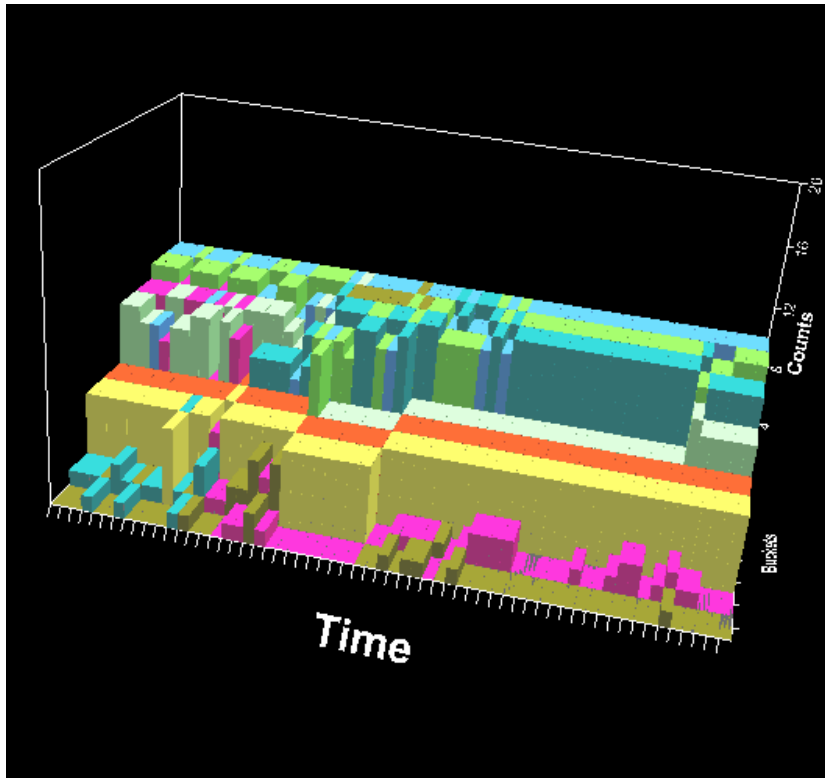


Figure 29. Thermal Canyons and Thermal Tower for SMTP PID Instance during (a), (b) Mailbomb attack; (c), (d) Smurf attack and (e), (f) combined Mailbomb and Smurf attacks.

The thermal canyon in Figure 29(e) appears to be a superposition of the thermal canyons in Figure 29(a) and (c). The wide spreading of bucket states on the canyon floor are caused by the Mailbomb attack, while the huge peaks against the vertical plane are due to the Smurf attack. This is verified by the two sets of thermalate shown in Figures 31 and 32, which are obtained by clicking on the canyon peak and the canyon floor respectively.

Thermalate Information										
Thermalates										
Timestamp: Wed Sep 08 16:06:16 PDT 2004										
Bucket State: [5] [10] [8] [10] [5] [5] [10] [0]										
Total Packets: 2552										
Src	SrcPort/ Type	Dst	DstPort/ Code	Flow	Match	New Friend	Fragmented	Protocol	Pkt Count	
118.204.35.141	0	172.16.112.50	0	OUT_OUT	false	false	false	ICMP	19	
118.204.35.119	0	172.16.112.50	0	OUT_OUT	false	false	false	ICMP	12	
21.15.139.72	0	172.16.112.50	0	OUT_OUT	false	false	false	ICMP	12	
118.204.35.58	0	172.16.112.50	0	OUT_OUT	false	false	false	ICMP	12	
118.204.35.246	0	172.16.112.50	0	OUT_OUT	false	false	false	ICMP	15	
6.238.105.36	0	172.16.112.50	0	OUT_OUT	false	false	false	ICMP	12	
242.99.186.214	0	172.16.112.50	0	OUT_OUT	false	false	false	ICMP	17	

Figure 30. Thermalate contributing to the canyon peak of the SMTP PID Instance during the combined Mailbomb and Smurf attacks.

Thermalate Information										
Thermalates										
Timestamp: Wed Sep 08 16:06:36 PDT 2004										
Bucket State: [5] [4] [10] [1] [5] [9] [10] [3]										
Total Packets: 18										
Src	SrcPort/ Type	Dst	DstPort/ Code	Flow	Match	New Friend	Fragmented	Protocol	Pkt Count	
194.27.251.21	113	172.16.114.50	5555	OUT_OUT	false	false	false	TCP	4	
172.16.114.50	5555	194.27.251.21	113	OUT_OUT	false	false	false	TCP	3	
172.16.114.50	25	194.27.251.21	2122	OUT_OUT	false	false	false	TCP	6	
194.27.251.21	2122	172.16.114.50	25	OUT_OUT	false	false	false	TCP	5	

Figure 31. Thermalate Contributing to the Canyon Floor of the SMTP PID Instance during the combined Mailbomb and Smurf Attacks.

## 2. ICMP PID Instance

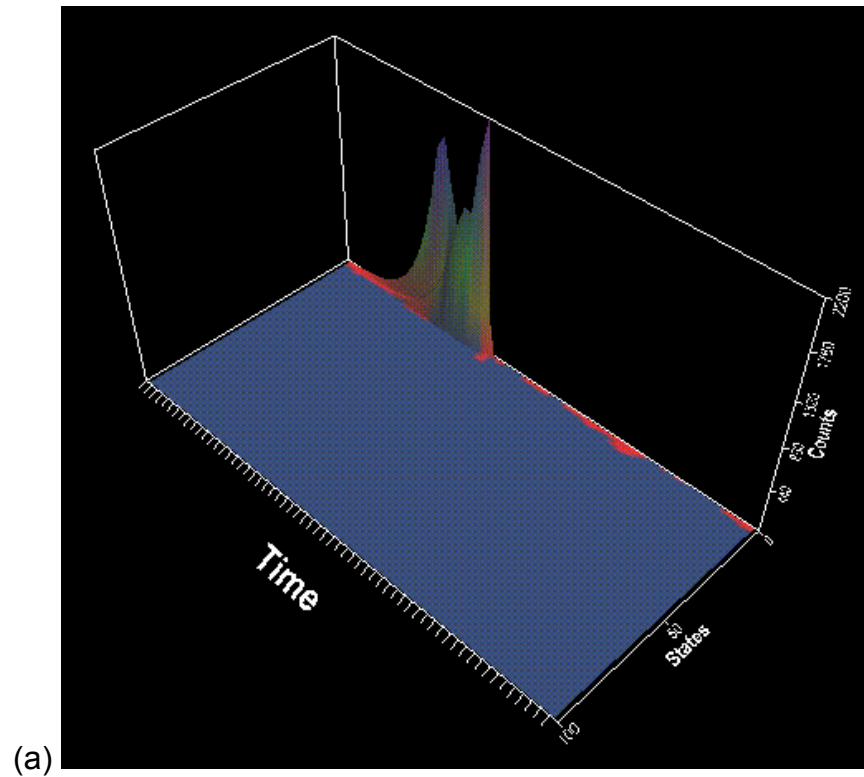
The thermal canyon and thermal tower displays in Figures 32(a) to (f) are produced by the ICMP PID instance when the Smurf attack, the Mailbomb attack and the combined Mailbomb and Smurf attacks are launched respectively.

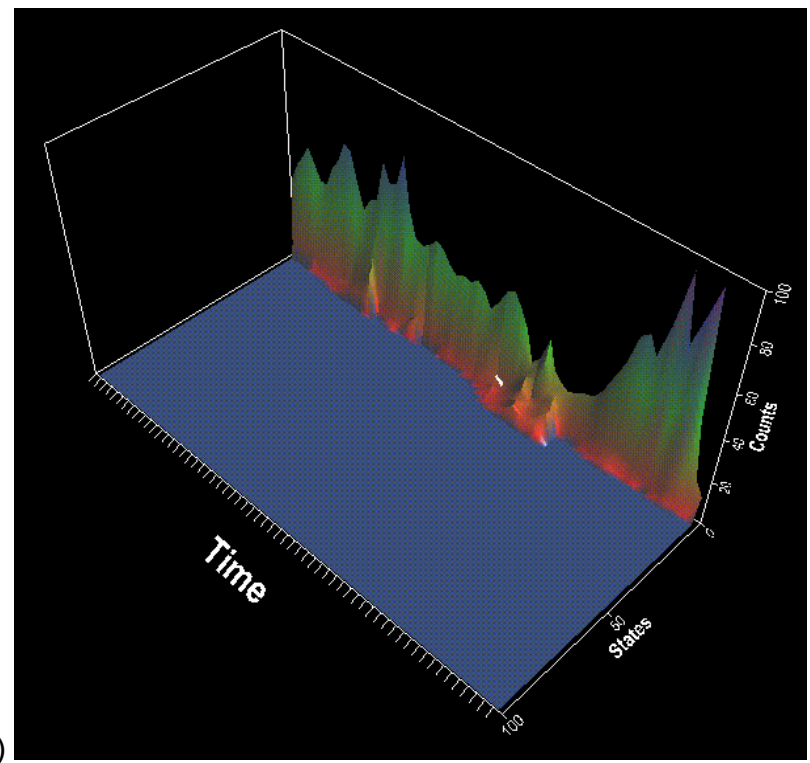
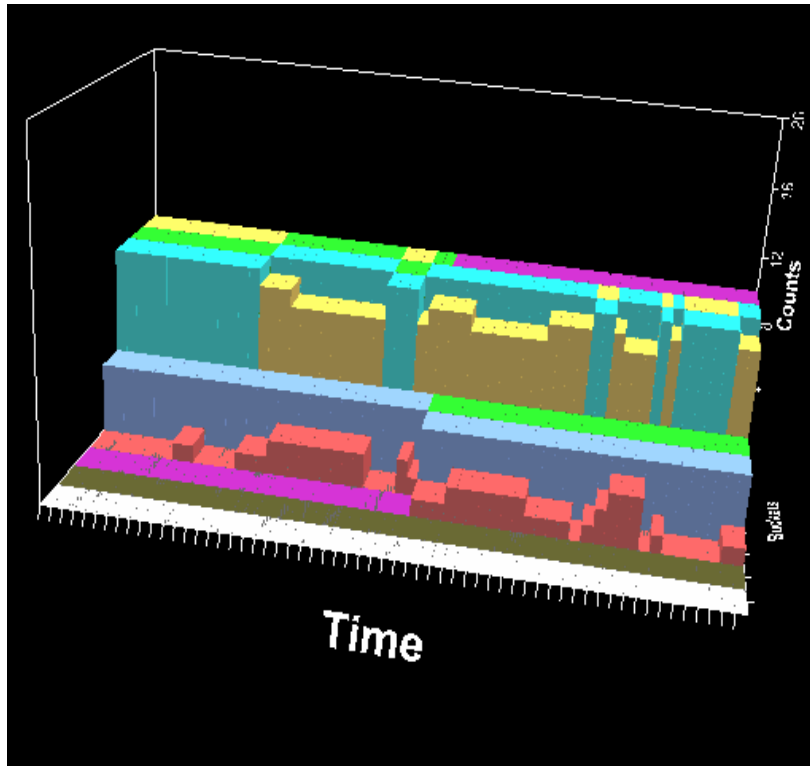
During the Mailbomb attack, the thermal canyon in Figure 32(c) does not display any obvious signs of anomaly. The network traffic during the attack

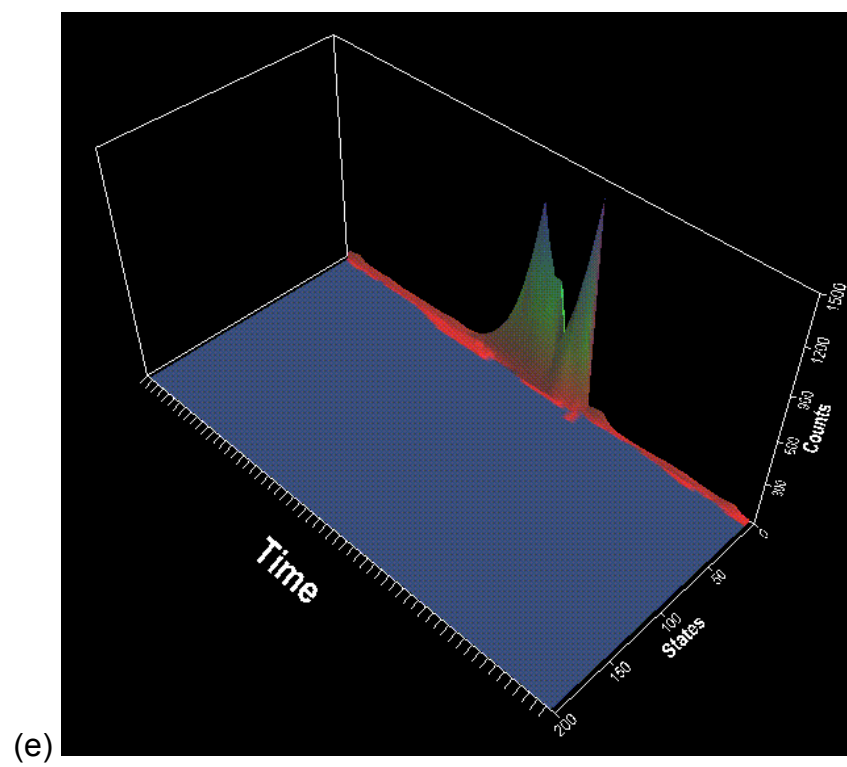
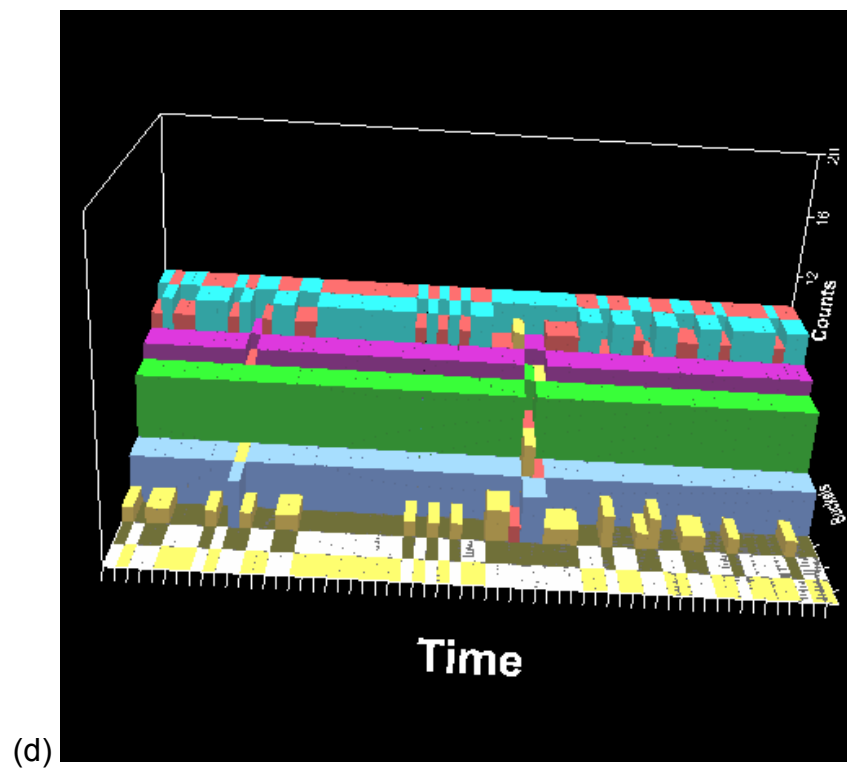


results in ball exchanges between Buckets 6 (represented by the yellow-colored bar) and 7 (represented by the red-colored bar).

The scale on the y-axis automatically scales to accommodate the highest value. It should be noted that the scale on Figures 32(a) and (c) are very different.







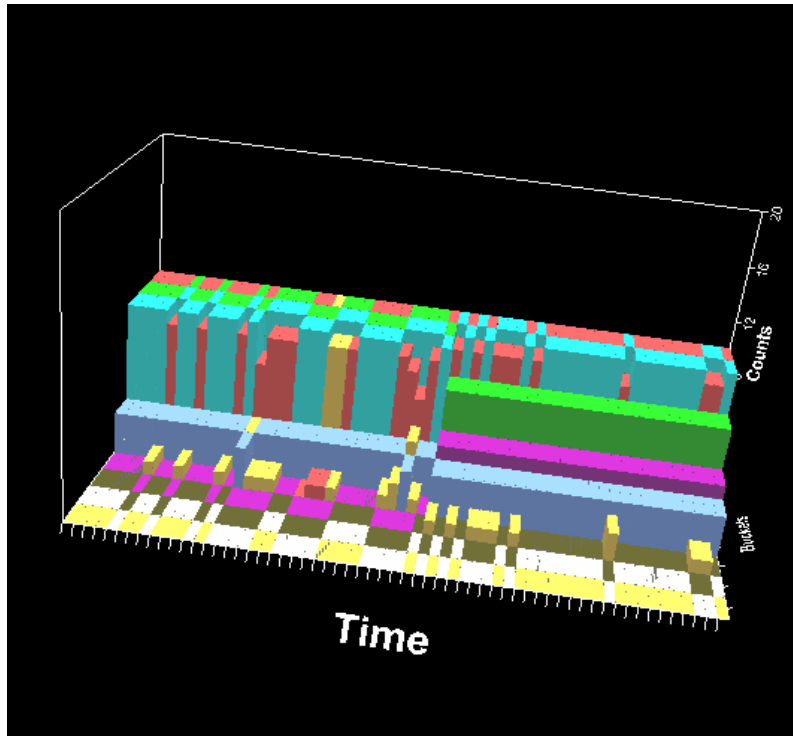


Figure 32. Thermal Canyons and Thermal Tower for ICMP PID Instance during (a), (b) Smurf attack; (c), (d) Mailbomb attack and (e), (f) combined Mailbomb and Smurf attacks.

The thermal canyon in Figure 32(e) has a landscape that is similar to Figure 32(a). The thermalate contributing to the canyon peak in Figure 32(e) is shown in Figure 33, which consists mostly of ICMP packets. The Smurf attack has completely overshadowed the Mailbomb attack.

Thermalate Information										
Thermalates										
Timestamp: Wed Sep 08 15:32:45 PDT 2004										
Bucket State: [10] [5] [10] [0] [0] [0] [0] [10]										
Total Packets: 1253										
Src	SrcPort/ Type	Dst	DstPort/ Code	Flow	Match	New Friend	Fragmented	Protocol	Pkt Count	
172.16.113.204	21307	195.115.218.108	23	OUT_OUT	false	false	false	TCP	14	
195.115.218.108	23	172.16.113.204	21307	OUT_OUT	false	false	false	TCP	7	
21.15.139.71	0	172.16.112.50	0	OUT_OUT	false	false	false	ICMP	4	
21.15.139.63	0	172.16.112.50	0	OUT_OUT	false	false	false	ICMP	6	
118.204.35.119	0	172.16.112.50	0	OUT_OUT	false	false	false	ICMP	6	
6.238.105.98	0	172.16.112.50	0	OUT_OUT	false	false	false	ICMP	4	
21.15.139.184	0	172.16.112.50	0	OUT_OUT	false	false	false	ICMP	7	
242.99.186.175	0	172.16.112.50	0	OUT_OUT	false	false	false	ICMP	5	
242.99.186.31	0	172.16.112.50	0	OUT_OUT	false	false	false	ICMP	6	
242.99.186.16	0	172.16.112.50	0	OUT_OUT	false	false	false	ICMP	5	
6.238.105.171	0	172.16.112.50	0	OUT_OUT	false	false	false	ICMP	9	
21.15.139.72	0	172.16.112.50	0	OUT_OUT	false	false	false	ICMP	3	
118.204.35.173	0	172.16.112.50	0	OUT_OUT	false	false	false	ICMP	7	
6.238.105.196	0	172.16.112.50	0	OUT_OUT	false	false	false	ICMP	7	

Figure 33. Thermalate Contributing to the Canyon Peak of the ICMP PID Instance during the combined Mailbomb and Smurf attacks.

### C. SMURF AND APACHE2 ATTACKS

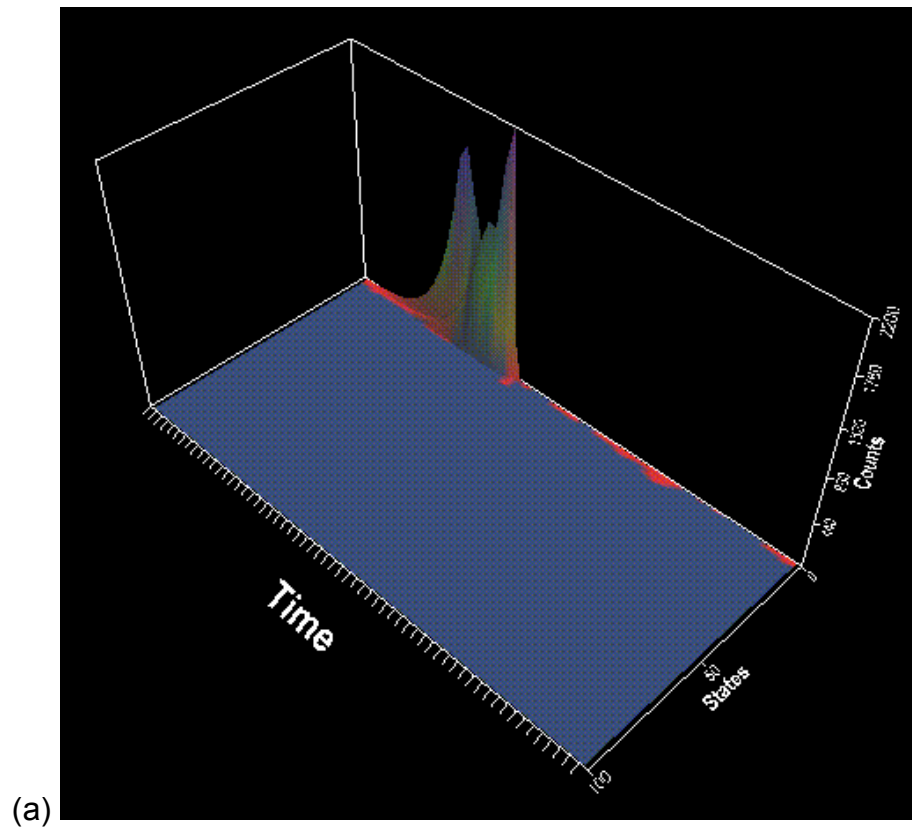
In this section, the network attack replay workstation replays the Smurf and the Apache2 attacks concurrently. The HTTP and ICMP PID instances are activated on the Zippo core to analyze the aggregated thermalate sent from sensors A and B.

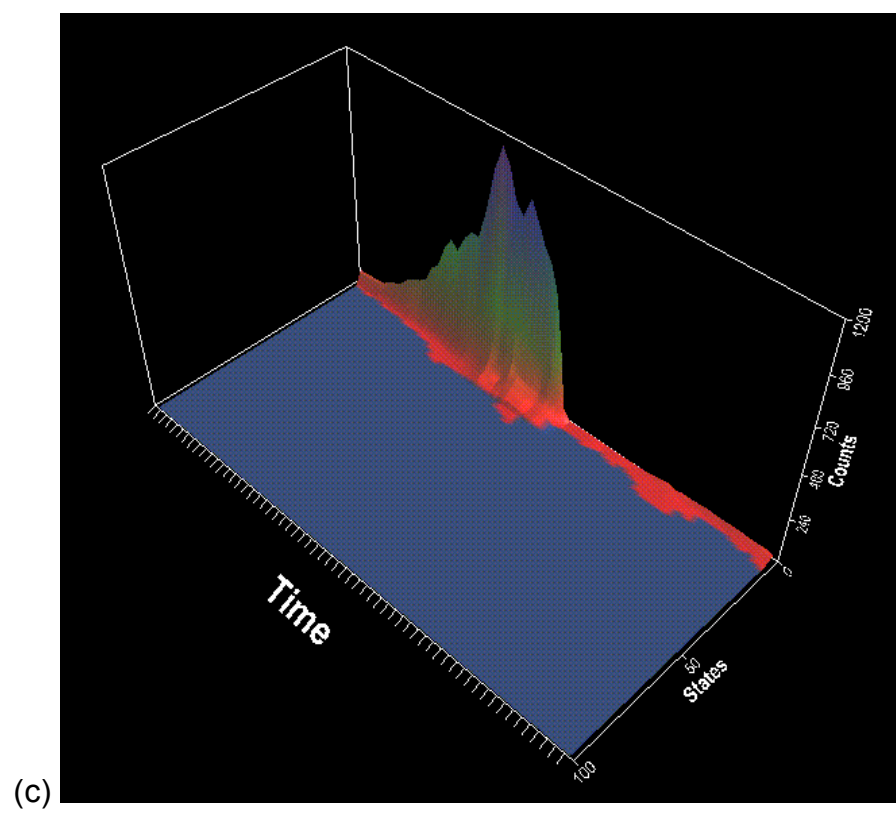
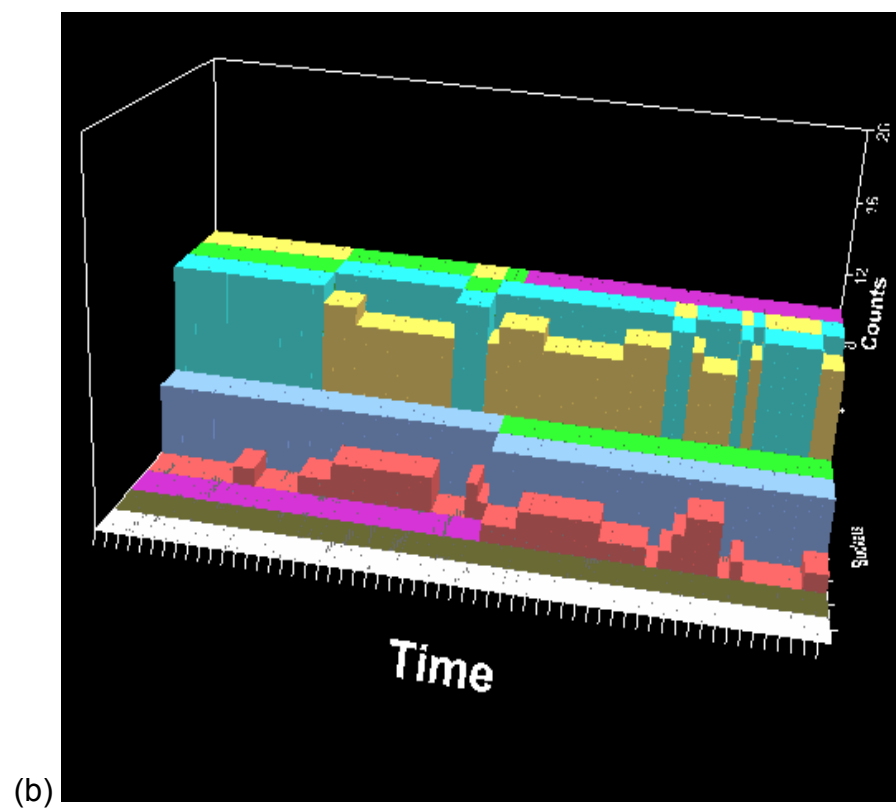
#### 1. ICMP PID Instance

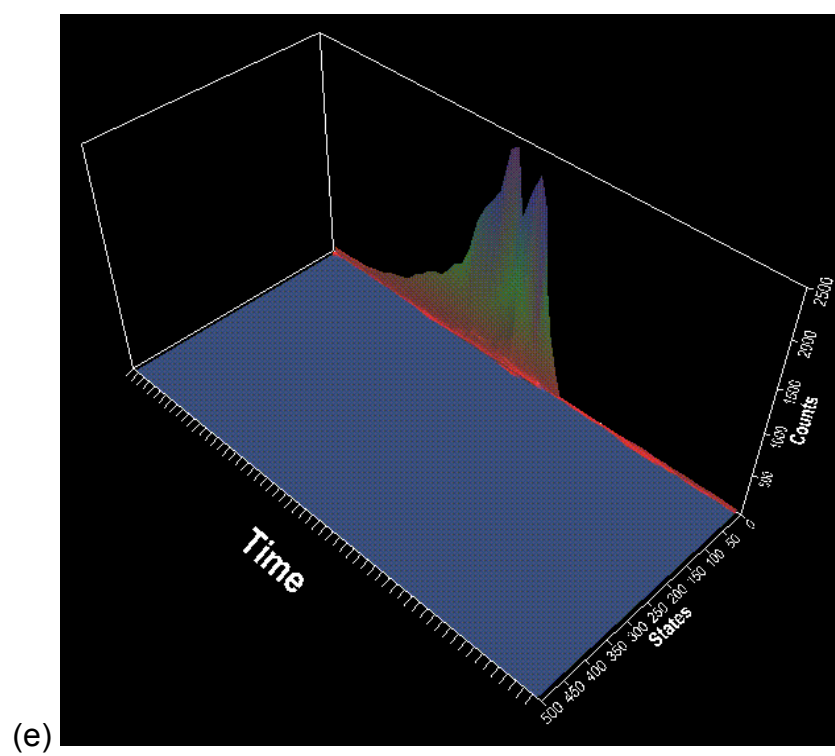
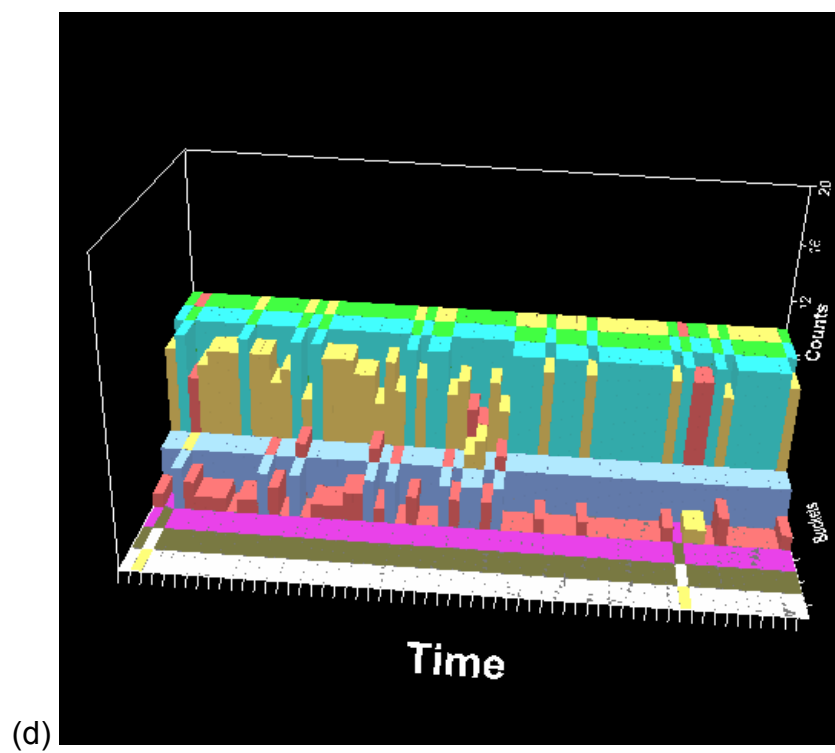
Figure 34(a) to (f) compare the thermal canyon and thermal tower displays of the ICMP PID Instance to a Smurf attack, an Apache2 attack and combined Smurf and Apache2 attacks respectively.

The Apache2 attack results in ball transfers between Buckets 6 (yellow-coloured bar) and 7 (red-colored bar), as shown in Figure 34(d). The peaks on the thermal canyon clearly indicate an anomalous situation.

During the combined Smurf and Apache2 attacks, the peaks on the thermal canyon are much higher, due to the large combined volume of traffic. Balls are transferred mainly between Buckets 5 and 2, and Buckets 6 and 7. It is difficult to tell from the thermal canyon alone that there are two attacks going on, unless the details about the network packets are obtained from the thermalate.









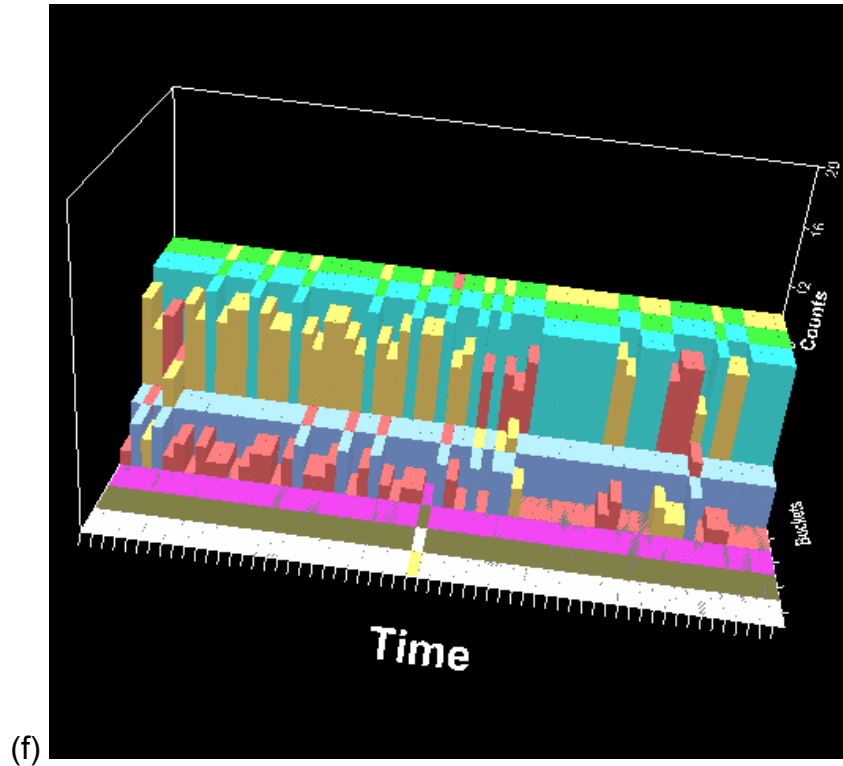
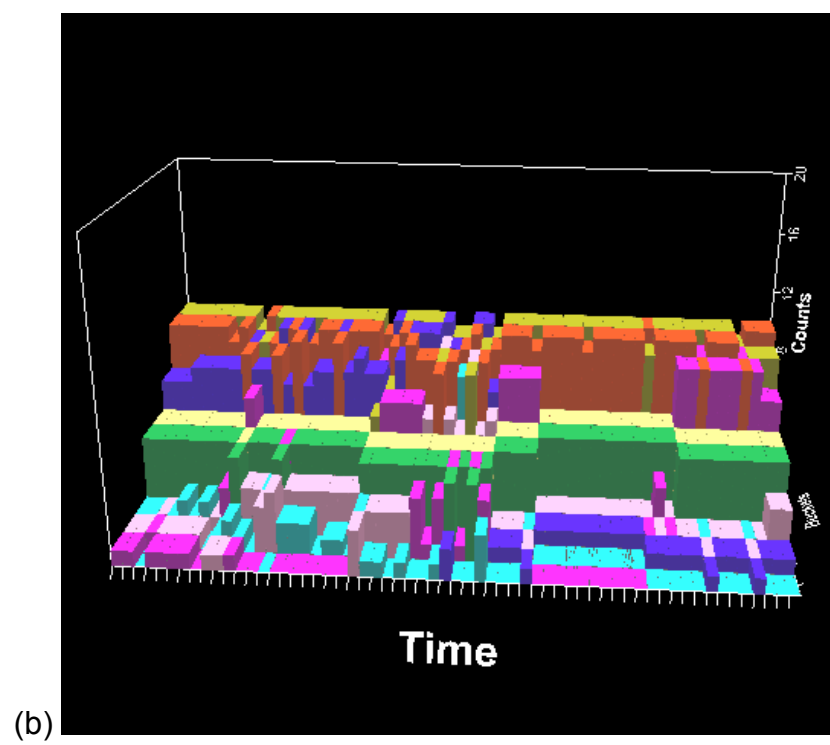
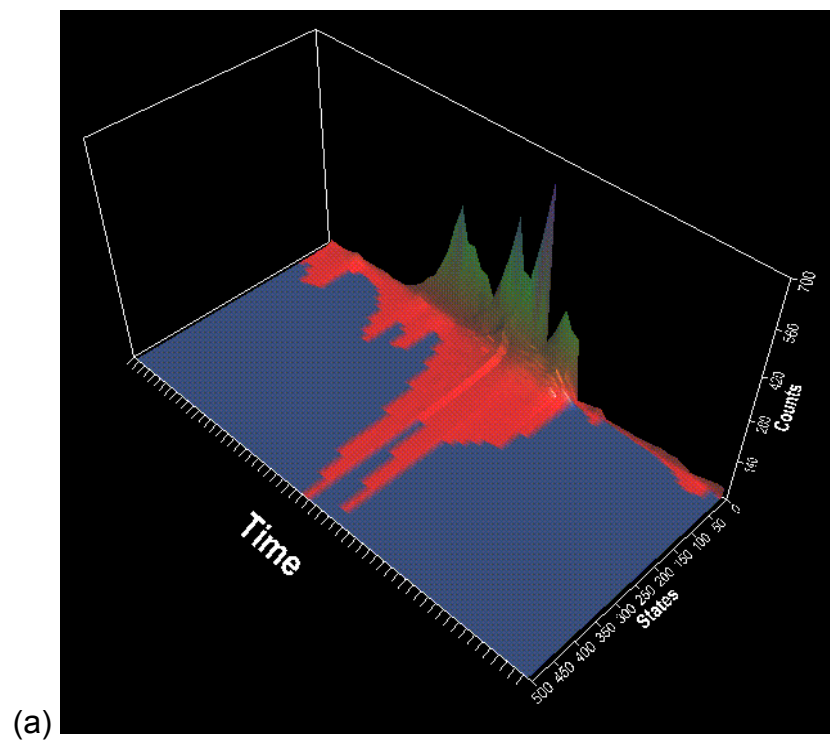


Figure 34. Thermal Canyons and Thermal Tower for ICMP PID Instance during (a), (b) Smurf attack; (c), (d) Apache2 attack and (e), (f) combined Smurf and Apache2 attacks.

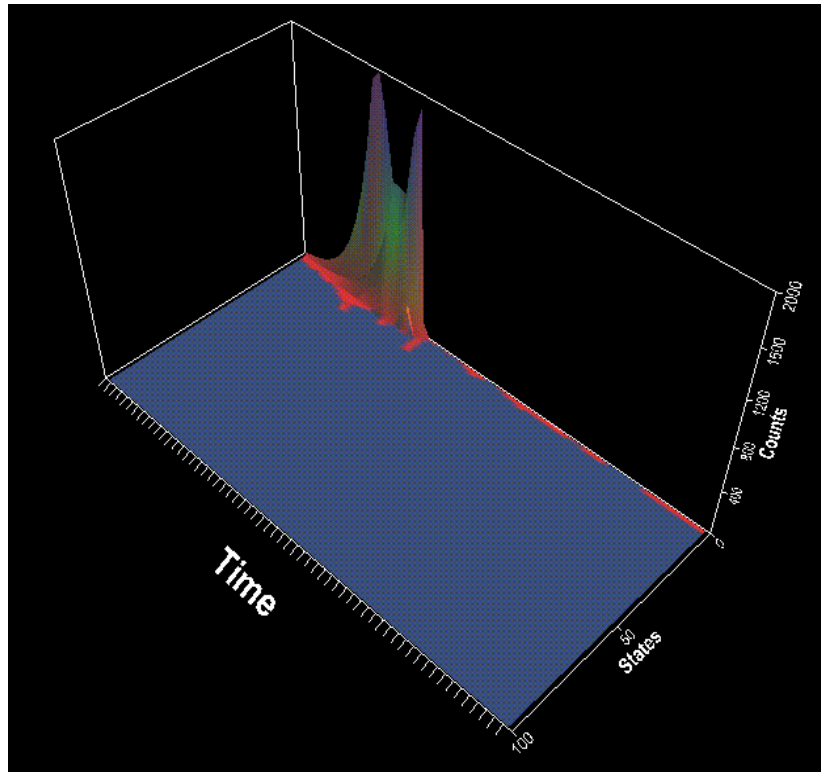
## 2. HTTP PID Instance

Figures 35(a) to (f) show different thermal canyons and tower displays of the HTTP PID Instance, corresponding to the detection of Apache2 attack, Smurf attack and a combination of the two attacks.

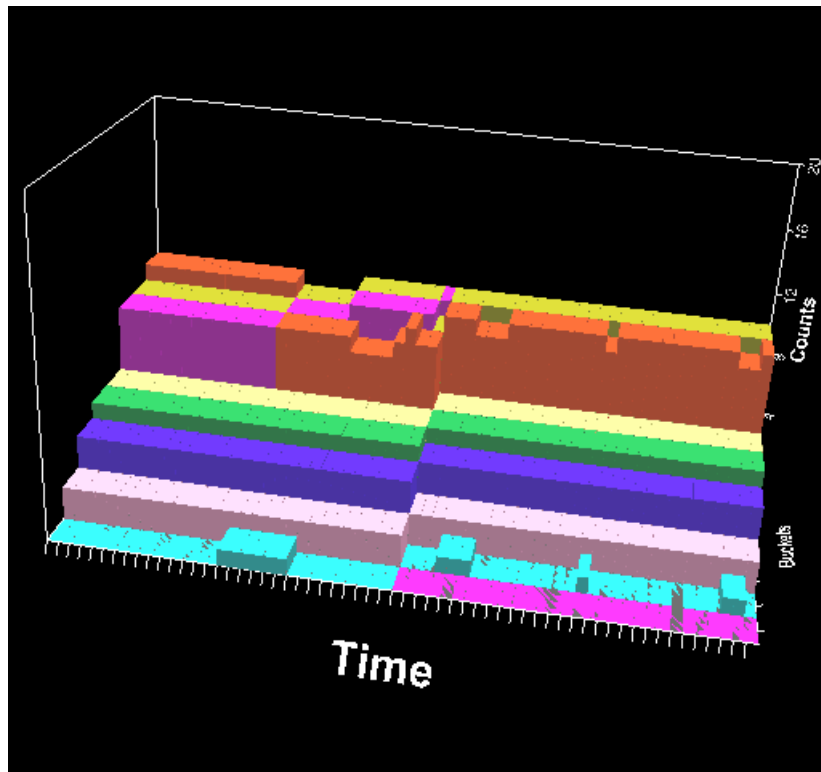
The Smurf attack results in high peaks on the thermal canyon, as shown in Figure 35(c). The high counts of the states visited are due to massive ball transfers from Bucket 7 (turquoise-colored bar) to 3 (maroon-colored bar).



(c)



(b)



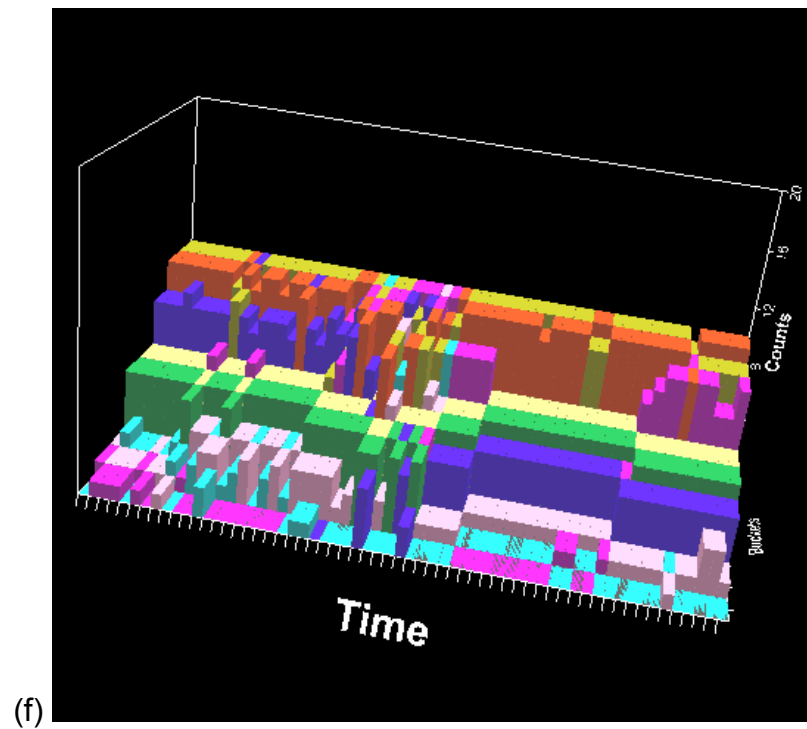
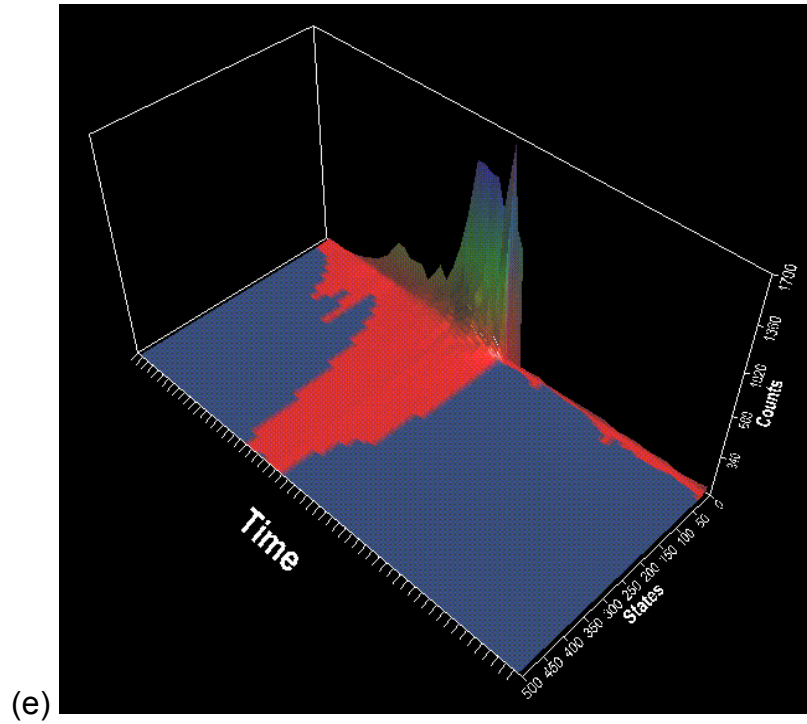


Figure 35. Thermal Canyons and Thermal Tower for HTTP PID Instance during (a), (b) Apache2 attack; (c), (d) Smurf attack and (e), (f) combined Apache2 and Smurf attacks.

The combined Smurf and Apache2 attacks produce a thermal canyon that is a combination of their individual thermal canyons. Balls are transferred between Buckets 2, 3 and 7. This results in a thermal canyon that has more bucket states than Figure 35(a), and lower peaks than Figure 35(c).

#### **D. MAILBOMB AND APACHE2 ATTACKS**

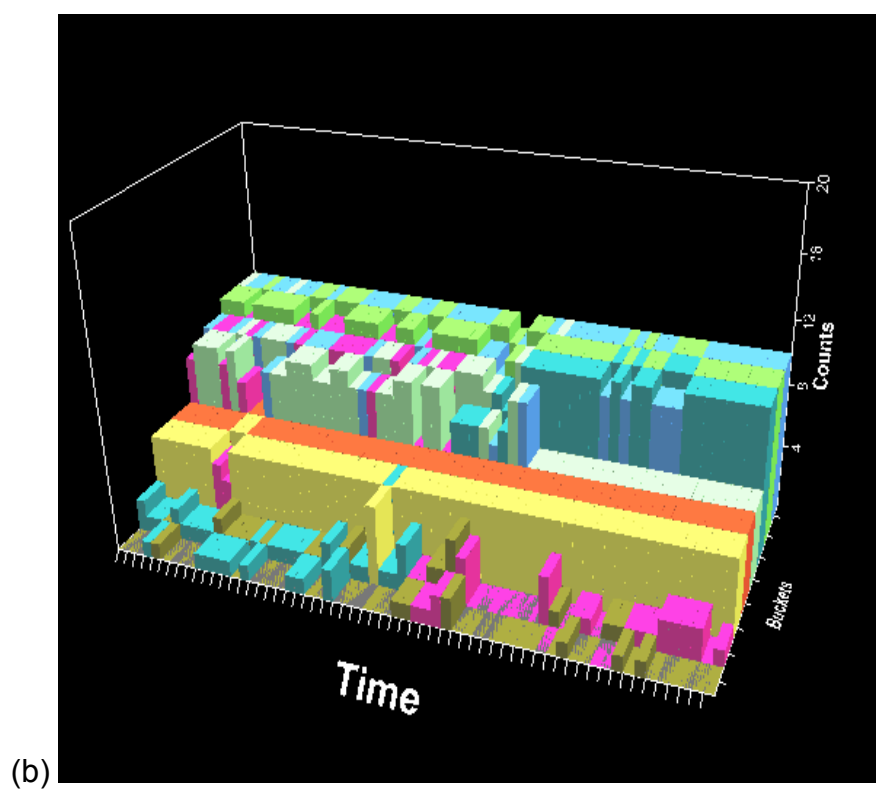
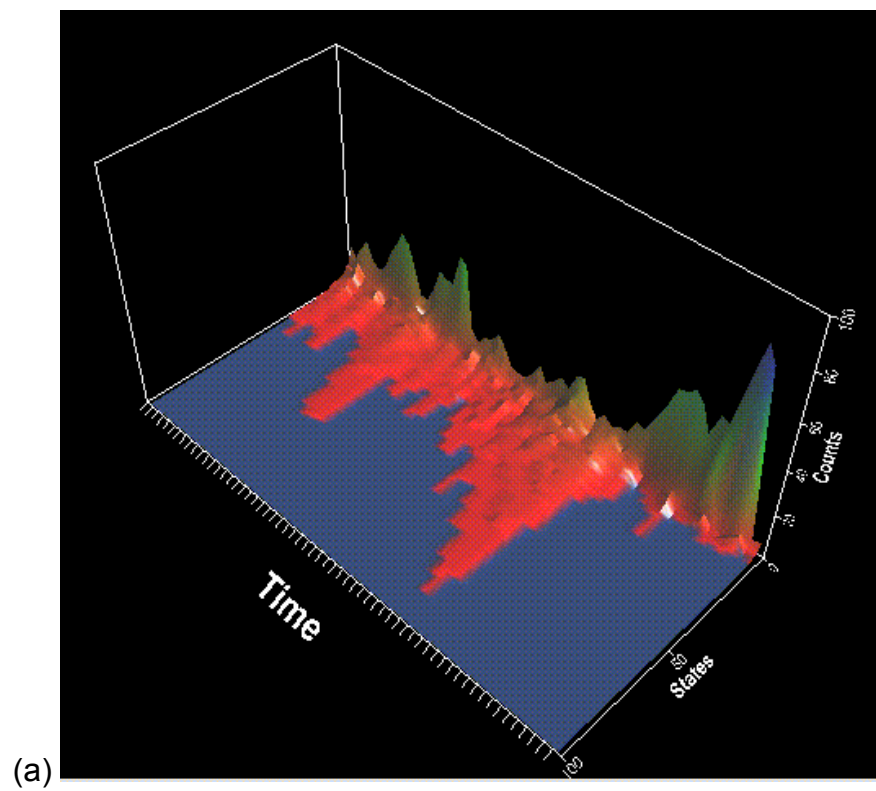
In this section, the network attack replay workstation replays the Mailbomb and the Apache2 attacks concurrently. The SMTP and HTTP PID instances are activated on the Zippo core to analyze the aggregated thermalate sent from sensors A and B.

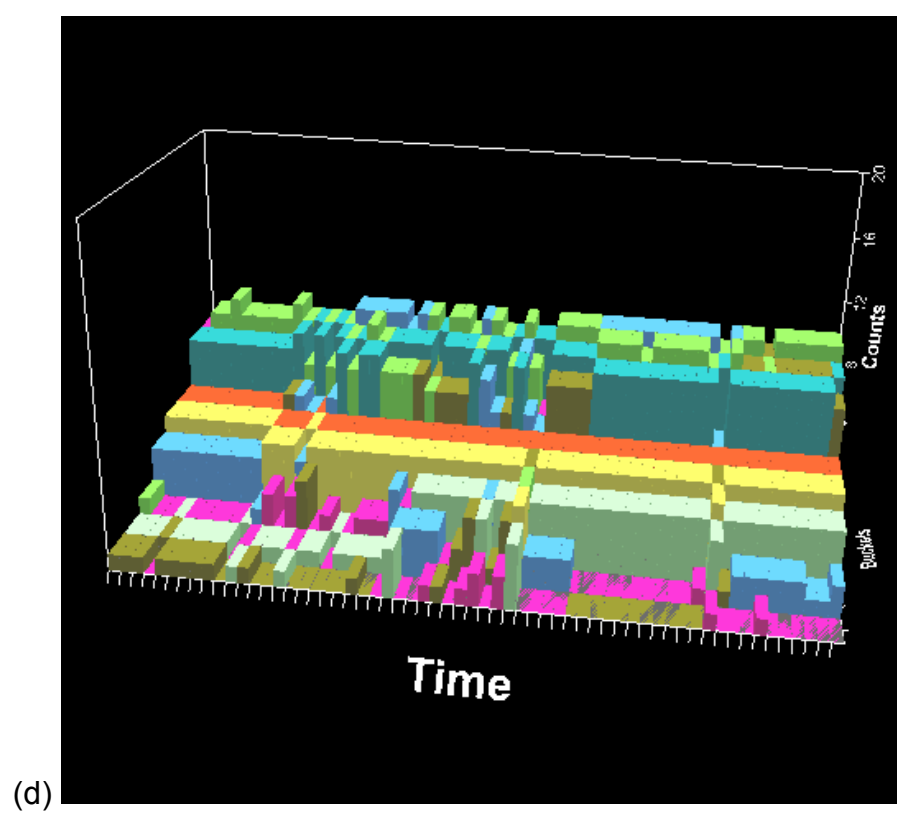
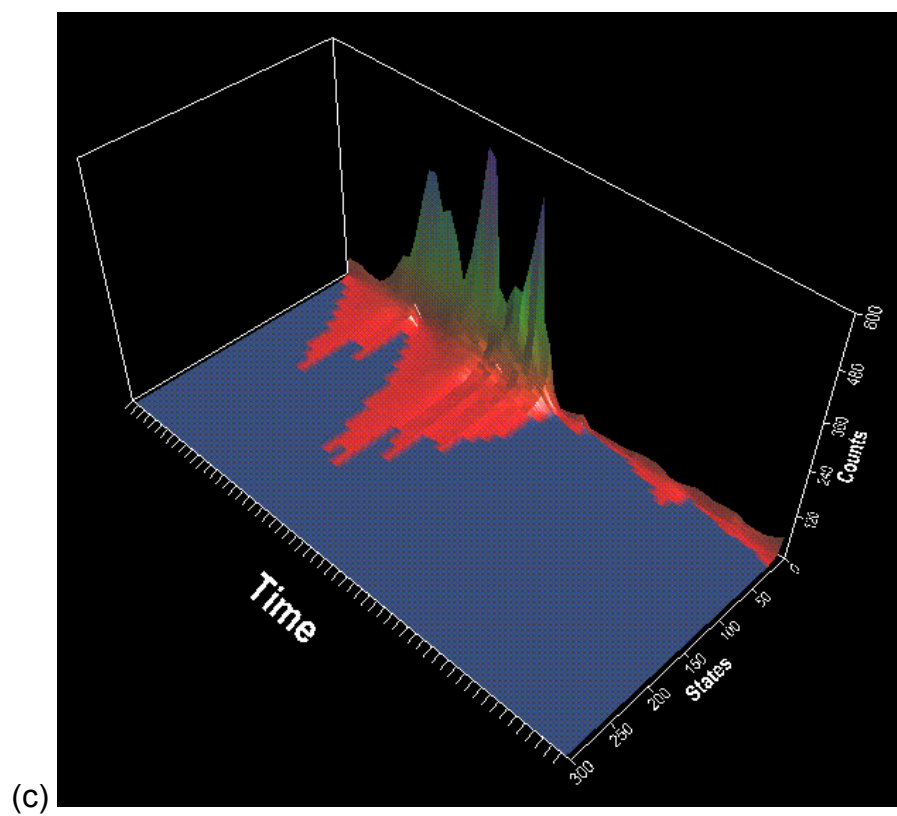
##### **1. SMTP PID Instance**

The SMTP PID instance produce different thermal canyon and tower displays, in consequence to the detection of Mailbomb, Apache2 and a combination of the two attacks, as shown in Figures 36(a) to (f).

The Apache2 attack causes balls to be moved between Buckets 7 and 2, which result in an increase in the bucket states, and high peaks on the thermal canyon, as shown in Figure 36(c).

The combined attacks result in ball transfers between Buckets 1, 2, 3, 5 and 7. The many permutations of the number of balls in each bucket lead to wide spreading of the bucket states, but narrower peaks on the thermal canyon.





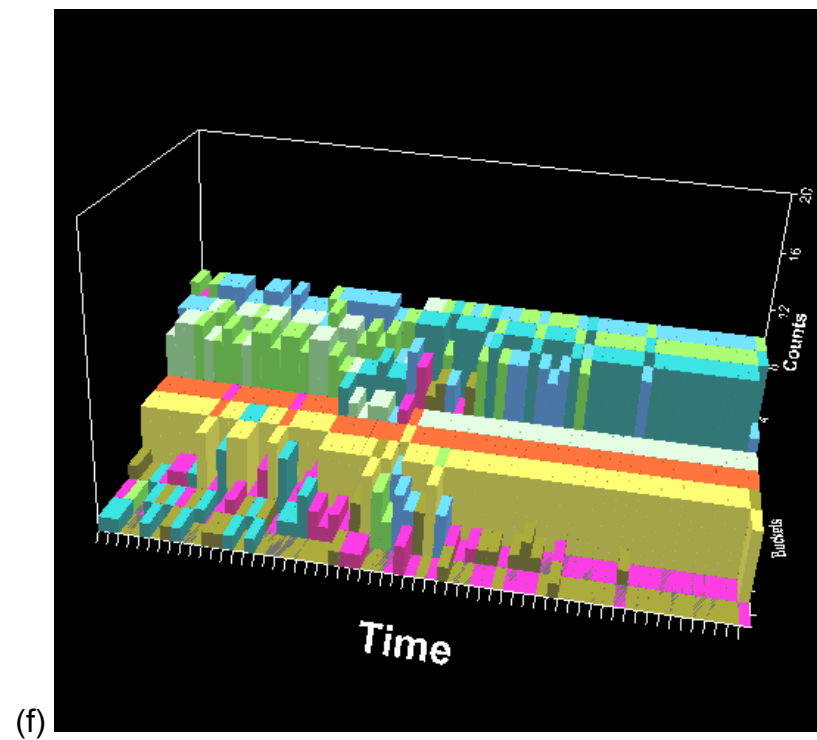
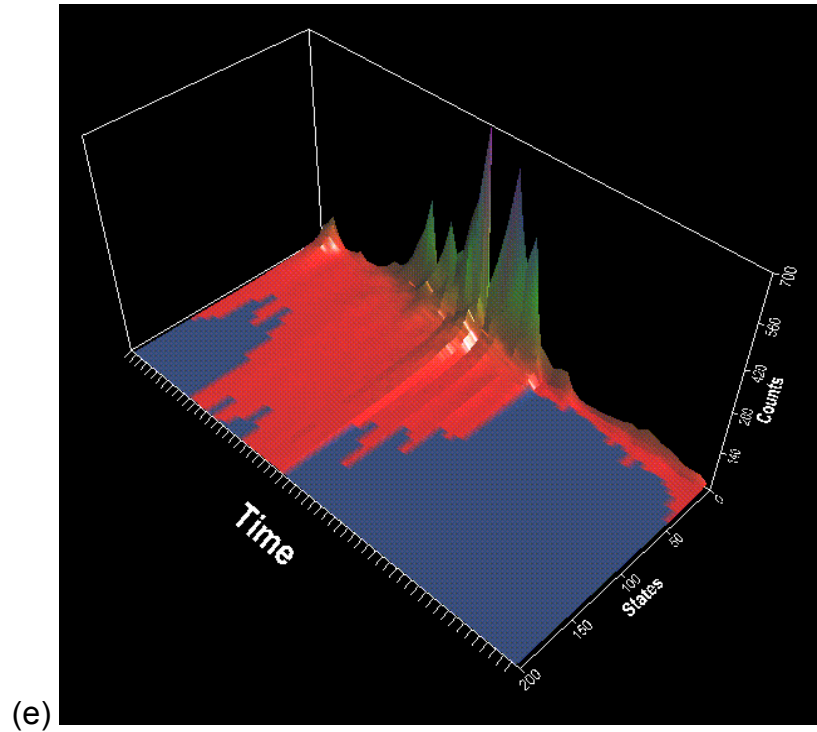


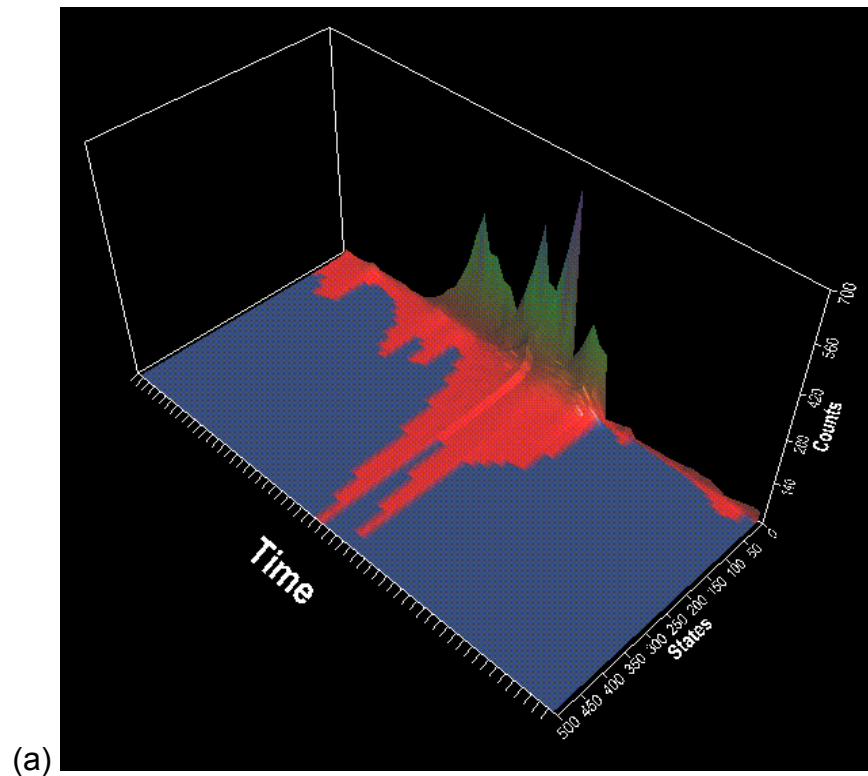
Figure 36. Thermal Canyons and Thermal Tower for SMTP PID Instance during (a), (b) Mailbomb attack; (c), (d) Apache2 attack and (e), (f) combined Mailbomb and Apache2 attacks.

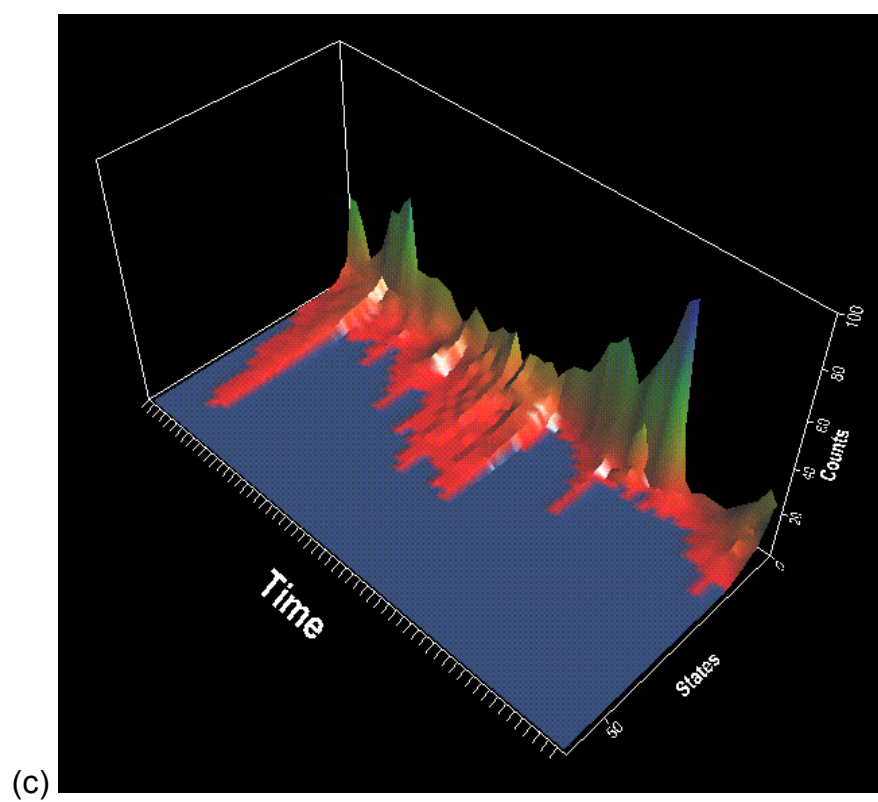
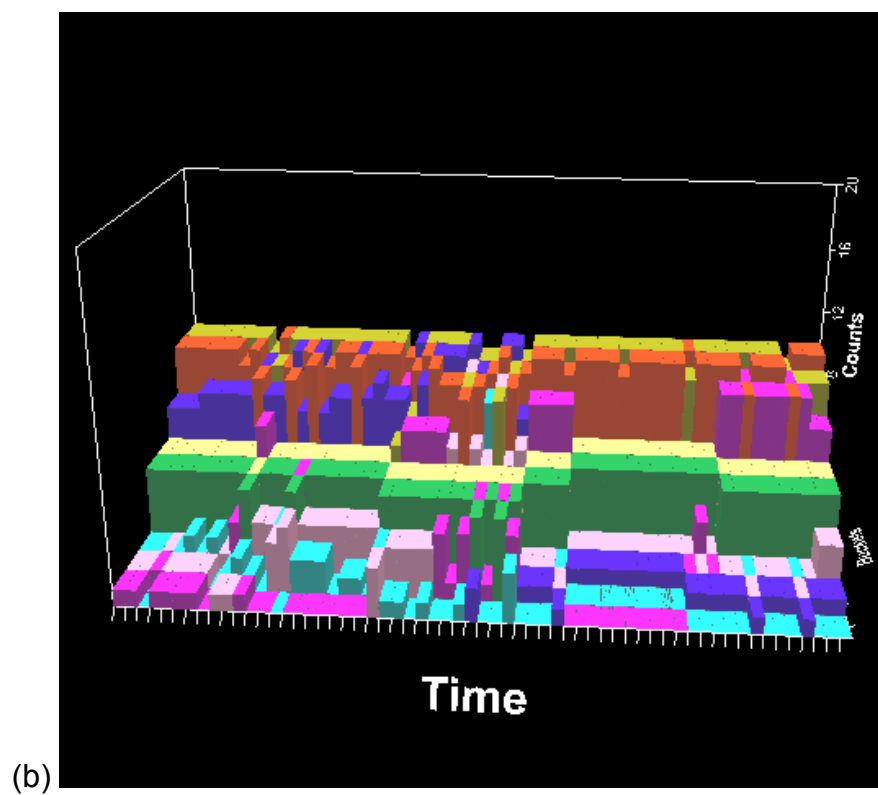


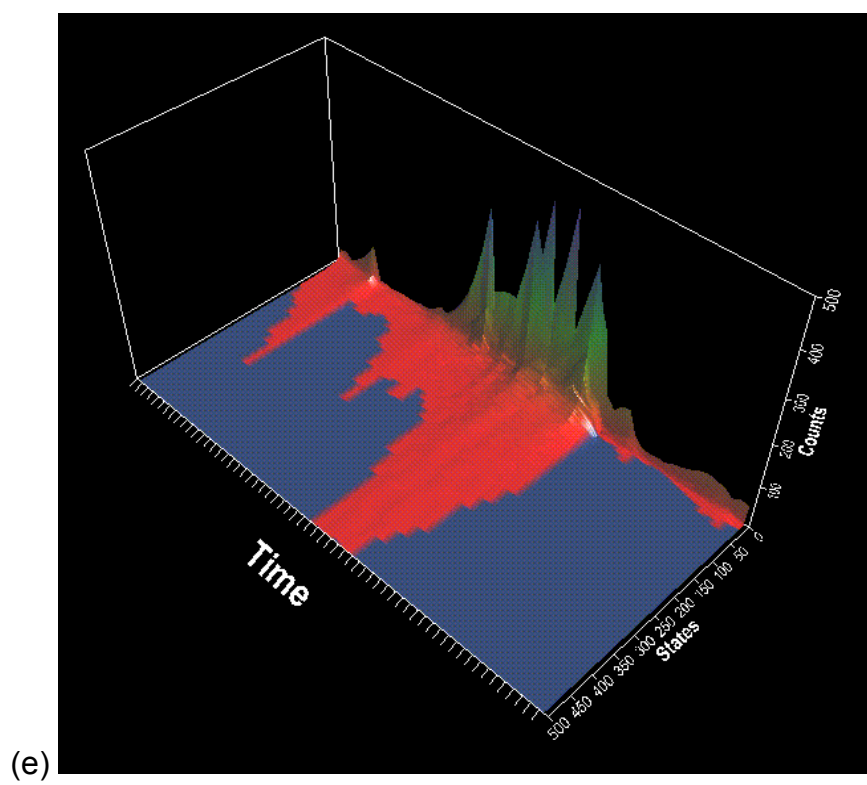
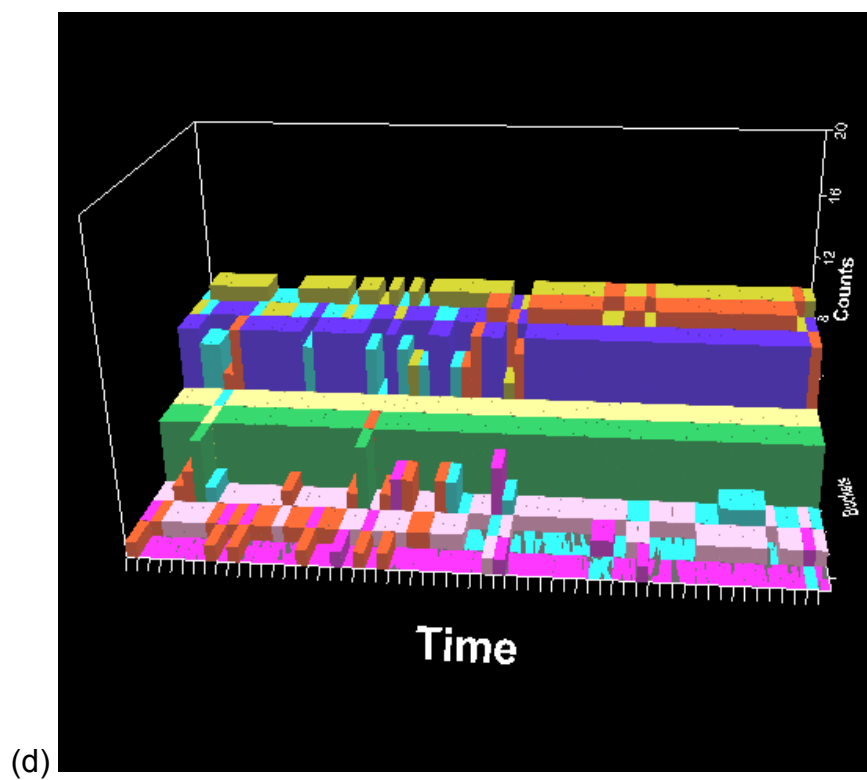
## 2. HTTP PID Instance

Figures 37(a) to (f) show the response of the HTTP PID instance to Apache2, Mailbomb and combined Apache2 and Mailbomb attacks. The Mailbomb attack results in the transfer of balls between Buckets 1, 3, 5 and 7 and leads to an increase in the bucket states.

The combined attacks result in a thermal canyon (see Figure 37(e)) that is a combination of the thermal canyons in Figure 37(a) and (c).







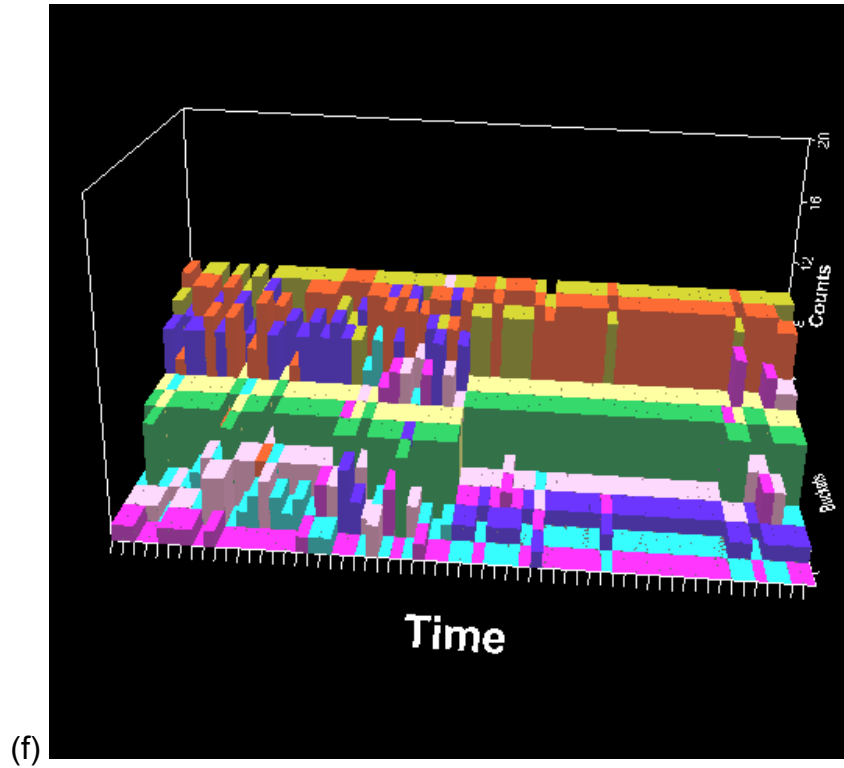


Figure 37. Thermal Canyons and Thermal Tower for HTTP PID Instance during (a), (b) Apache2 attack; (c), (d) Mailbomb attack and (e), (f) combined Apache2 and Mailbomb attacks.

## E. ANALYSIS OF RESULTS

The above results show the responses of the SMTP, ICMP and HTTP PID instances to different attacks. Table 8 shows a summary of the results – a ‘Y’ indicates that the PID instance gave an indication of the anomalous behavior on the corresponding thermal canyon, and an ‘N’ indicates otherwise.

PID Instance	Smurf Attack	Mailbomb Attack	Apache2 Attack
ICMP	Y (4 states, 7200 counts)	N (15 states, 90 counts)	Y (10 states, 1200 counts)
SMTP	Y (10 states, 2500 counts)	Y (90 states, 50 counts)	Y (200 states, 550 counts)
HTTP	Y (10 states, 2000 counts)	N (50 states, 90 counts)	Y (150 states, 700 counts)

Table 8. Summary of Responses to attacks.

With the exception of the Mailbomb attack with the ICMP and HTTP PID instances, all other PID instances yielded a response, even when they are subjected to attacks which they are not designed to detect. The Mailbomb attack in the HTTP PID instance could be considered a borderline detection response. The response was certainly strongest in PID instances designed to services being monitored. To correctly identify the attack, the thermalate has to be examined in order to understand the nature of the network packets.

## **F. SUMMARY**

This chapter presents the findings of the experiment to investigate the response of Zippo to concurrent attacks of differing nature. The results show that each PID instance is able to detect the attack that it is designed to look out for, even in the presence of another attack. The downside, however, is that most of the PID instances yield a response even to attacks which they not designed to detect.

The next chapter summarizes the findings from this thesis and discusses possible areas for future research.

THIS PAGE INTENTIONALLY LEFT BLANK

## **VIII. REPORT SUMMARY AND FUTURE RESEARCH**

### **A. REPORT SUMMARY**

The main objective of this thesis is to implement a DIDS using Terminator. Zippo, a newer version of Terminator, is tried and tested instead. The modular software design of Zippo facilitates the task of deploying a DIDS, as the sensing, core and GUI components can function separately on different host operating systems.

The experimental network was setup for the deployment of a distributed Zippo system. It consisted of two remote sensors which report to the Zippo core. The GUI, which displays the thermal canyons and thermal towers, was co-located with the Zippo core. A separate machine was set up to replay pre-recorded network traffic containing various network attacks.

The first experiment on the Zippo system was to test the response to Mailbomb, Smurf and Apache2 attacks. In order to correctly identify the attacks, the Zippo system had to be configured to analyze and differentiate SMTP, ICMP and HTTP traffic from other traffic. This was achieved with the construction of three decision trees, named as SMTP, ICMP and HTTP PID instances. When the attacks were launched, the thermal canyons and thermal towers of the corresponding PID instances gave correct responses.

The next experiment was to investigate the response of the Zippo system when both remote sensors were subjected to the same network traffic containing the same attack. The Zippo core had to perform aggregated analyses of the thermalate received from both sensors. The results showed that in general, the thermal canyons produced similar landscapes as when only one sensor detected the attack. But due to the double volume of traffic, the number of bucket states and the counts of the bucket states were doubled.

The last experiment was to investigate the response of the Zippo system when the remote sensors were subjected to different network traffic containing different attacks at the same time. The results showed that the PID instances

could still identify the attacks which they were designed for, even in the presence of other attacks. However, it led to the discovery that the PID instances could also respond to attacks which they were not meant to detect.

Overall, the objective of the thesis research has been met. A DIDS has been successfully implemented using Zippo. Interesting results have been obtained and discussed from the various experiments conducted on the distributed Zippo system.

## **B. FUTURE RESEARCH**

In Chapter III, the configuration parameters of the Zippo core were briefly experimented with. More work could be carried out in this area, to find out the most optimum set of parameters to use in order to obtain the most optimum response. In particular, the SL and SF could be varied to investigate the effect on the thermal canyon displays.

The existence of borderline response emphasizes the need for robust detection techniques. The area of attack detection is not addressed in the thesis but holds significant potential for future research.

Another area to research is the optimum design of the PID instances to identify only the type of attack which it is designed to detect. This ensures better accuracy in the detection of attacks, instead of having every PID instance respond to an attack.



## LIST OF REFERENCES

- [1] Dorothy E. Denning, "An Intrusion-Detection Model," *Proc. Of IEEE Symposium on Security and Privacy*, pp. 118 131, 1986.
- [2] Stephen Northcutt and Judy Novak, *Network Intrusion Detection*, 3<sup>rd</sup> Edition, Sams, August 2002.
- [3] Donald, Stephen D. and McMillen, Robert V., "Therminator 2: Developing a Real Time Thermodynamic Based Patternless Intrusion Detection System," Master's Thesis, Naval Postgraduate School, Monterey, California, September 2001.
- [4] John C. McEachen, Stephen D. Donald, Robert V. McMillen, David K. Ford, "Using Thermodynamics to Model Network Conversation Flux for Intrusion Detection", IEEE MILCOM 2002, Los Angeles, CA, October 2002.
- [5] Massachusetts Institute of Technology, Lincoln Laboratory, DARPA Intrusion Detection Evaluation, <http://www.ll.mit.edu/IST/ideval/index.html>, last accessed 14 September 2004.
- [6] Przemyslaw Kazienko and Piotr Dorosz, "Intrusion Detection Systems (IDS) Part I – (network intrusions; attack symptoms; IDS tasks; and IDS architecture)," [http://www.windowsecurity.com/articles/Intrusion\\_Detection\\_Systems\\_IDS\\_Part\\_I\\_network\\_intrusions\\_attack\\_symptoms\\_IDS\\_tasks\\_and\\_IDS\\_architecture.html](http://www.windowsecurity.com/articles/Intrusion_Detection_Systems_IDS_Part_I_network_intrusions_attack_symptoms_IDS_tasks_and_IDS_architecture.html), last accessed 2 October 2004.
- [7] Phil Porras, Dan Schnackenberg, Stuart Staniford-Chen, Maureen Stillman and Felix Wu, "The Common Intrusion Detection Framework Architecture," <http://www.isi.edu/gost/cidf/drafts/architecture.txt>, last accessed 2 October 2004.
- [8] Przemyslaw Kazienko and Piotr Dorosz, "Intrusion Detection Systems (IDS) Part 2 – Classification; methods; techniques," <http://www.windowsecurity.com/articles/IDS-Part2-Classification-methods-techniques.html>, last accessed 2 October 2004.
- [9] John Zachary, John McEachen, Dan Ettlich, "Conversation Exchange Dynamics for Real-Time Network Monitoring and Anomaly Detection," *Second IEEE International Information Assurance Workshop*, April 2004.

- [10] Daniel W. Ettlich, "Terminator: Configuring the Underlying Statistical Mechanics Model", M.S. Thesis, Department of Electrical and Computer Engineering, Naval Postgraduate School, Monterey, California, December 2003.
- [11] John Marinovich and Stefan Walch, "Analysis of Initial and Boundary Conditions in Terminator Conversation Exchange Dynamics," Master's Thesis, Naval Postgraduate School, Monterey, California, March 2004.
- [12] John Zachary, "Zippo: A Robust and Portable Network Anomaly Detection System," May 2004.

## INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center  
Ft. Belvoir, Virginia
2. Dudley Knox Library  
Naval Postgraduate School  
Monterey, California
3. Dr. John C. McEachen  
Naval Postgraduate School  
Monterey, California
4. Dr. Su Wen  
Naval Postgraduate School  
Monterey, California