NAVAL
POSTGRADUATE
SCHOOL

MONTEREY, CALIFORNIA

THESIS

MALWEBID–AUTODETECTION AND IDENTIFICATION OF MALICIOUS WEB HOSTS THROUGH LIVE TRAFFIC ANALYSIS

by

Tony Nichols

March 2013

Thesis Co-Advisors: Robert Beverly
Geoffrey Xie

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704–0188*

1. REPORT DATE *(DD–MM–YYYY)* 1–4–2013	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* 2012-06-01 – 2013-03-15

4. TITLE AND SUBTITLE MalWebID–Autodetection and Identification of Malicious Web Hosts Through Live Traffic Analysis	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Tony Nichols	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Department of the Navy	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

The views expressed in this work are those of the author and do not reflect official policy or position of the Department of Defense or the U.S. Government.

14. ABSTRACT

This thesis investigates the ability for recently devised packet-level Transmission Control Protocols (TCP) transport classifiers to discover abusive traffic flows, especially those not found via traditional methods, e.g., signatures and real-time blocklists. Transport classification is designed to identify hosts considered to be part of abusive infrastructure without deep packet inspection. A particular focus is to understand the applicability of such methods to live, real-world network traffic obtained from the Naval Postgraduate School campus enterprise network. This research evaluates both how consistent and how complimentary transport traffic classification is with known blocklists. In particular, the system has a 97.8% average accuracy with respect to blocklist ground-truth, while also correctly identifying 94% of flows to abusive hosts unknown to the blocklists as verified through manual sampling.

15. SUBJECT TERMS

Network Security, Malicious Activity, Abusive Infrastructure

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(include area code)*
Unclassified	Unclassified	Unclassified	UU	81	

THIS PAGE INTENTIONALLY LEFT BLANK

MALWEBID–AUTODETECTION AND IDENTIFICATION OF MALICIOUS WEB HOSTS THROUGH LIVE TRAFFIC ANALYSIS

Tony Nichols
Lieutenant Commander, United States Navy
B.S., University of Phoenix

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 2013

Author:	Tony Nichols
Approved by:	Robert Beverly Thesis Co-Advisor
	Geoffrey Xie Thesis Co-Advisor
	Peter J. Denning Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis investigates the ability for recently devised packet-level Transmission Control Protocols (TCP) transport classifiers to discover abusive traffic flows, especially those not found via traditional methods, e.g., signatures and real-time blocklists. Transport classification is designed to identify hosts considered to be part of abusive infrastructure without deep packet inspection. A particular focus is to understand the applicability of such methods to live, real-world network traffic obtained from the Naval Postgraduate School campus enterprise network. This research evaluates both how consistent and how complimentary transport traffic classification is with known blocklists. In particular, the system has a 97.8% average accuracy with respect to blocklist ground-truth, while also correctly identifying 94% of flows to abusive hosts unknown to the blocklists as verified through manual sampling.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

1 Introduction **1**
1.1 Significance of Work to the DoD 2
1.2 Scope 3
1.3 Goals 3
1.4 Results 4
1.5 Thesis Organization 4

2 Background **7**
2.1 Prior Work 7
2.2 Machine Learning / Supervised Classifiers 12
2.3 Classification Techniques 13
2.4 Real-time Blackhole List and Alexa 16

3 Methodology **19**
3.1 Overview 19
3.2 Packet Captures 22
3.3 Programmatic Functions 23
3.4 RBS and Alexa 27
3.5 Flow Feature Labeling 28

4 Results **29**
4.1 Testing 29
4.2 Testing Sets 30
4.3 Hybrid Testing 40
4.4 Sampling Analysis 41

4.5 Classification Examples. 49

5 Conclusion and Future Work **55**

5.1 Conclusion 55

5.2 Key Results 56

5.3 Future Work 57

5.4 Summary 59

List of Figures

Figure 3.1 MalWebID Flow Process. 21

Figure 3.2 Example Traffic Features Extracted by TTAD. 24

Figure 4.1 MalWebID Programmatic Flow for Capture 1. 32

Figure 4.2 MalWebID Programmatic Flow for Capture 2. 35

Figure 4.3 MalWebID Programmatic Flow for Capture 3. 37

Figure 4.4 MalWebID Programmatic Flow for Capture 4. 38

Figure 4.5 MalWebID Programmatic Flow for Hybrid Test. 40

Figure 4.6 Flow Labeling Conflicts 48

Figure 4.7 Test 1 - Conflict Percentages 52

Figure 4.8 Test 2 - Conflict Percentages 52

Figure 4.9 Test 3 - Conflict Percentages 53

Figure 4.10 Test 4 - Conflict Percentages 53

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

Table 2.1 Supervised Learning Feature Model 13

Table 3.1 MalWebID Program Process 19

Table 3.2 Packet Captures 22

Table 3.3 Transport Traffic Analysis Daemon (TTAD) Transport Traffic Features . 24

Table 3.4 Transmission Control Protocol (TCP) Header Information 25

Table 3.5 Fetcher Processed Uniform Resource Locators (URLs) 26

Table 3.6 Processed Capture Details 27

Table 4.1 Composition of *labeled.csv* and *unknown.csv* Files 31

Table 4.2 Test One - Prediction Accuracy 33

Table 4.3 Test One - Confusion Matrix 33

Table 4.4 Test Two - Prediction Accuracy 34

Table 4.5 Test Two - Confusion Matrix 34

Table 4.6 Test Three - Prediction Accuracy 36

Table 4.7 Test Three - Confusion Matrix 36

Table 4.8 Test Four - Prediction Accuracy 39

Table 4.9 Test Four - Confusion Matrix 39

Table 4.10 Hybrid Test (1 and 2) - Prediction Accuracy 41

Table 4.11 Hybrid Test - Confusion Matrix 41

Table 4.12 Fetcher - Manual Sample Confusion Matrix 43
Table 4.13 Fetcher - Manual Sample Results 43
Table 4.14 Test 1 - Manual Sample Confusion Matrix 43
Table 4.15 Test 1 - Manual Sample Results 43
Table 4.16 Test 2 - Manual Sample Confusion Matrix 44
Table 4.17 Test 2 - Manual Sample Results 44
Table 4.18 Hybrid Test - Manual Sample Confusion Matrix 45
Table 4.19 Hybrid Test - Manual Sample Results 45
Table 4.20 Test 3 - Manual Sample Confusion Matrix 45
Table 4.21 Test 3 - Manual Sample Results 46
Table 4.22 Test 4 - Manual Sample Confusion Matrix 46
Table 4.23 Test 4 - Manual Sample Results 46
Table 4.24 Conflict Percentages 49

List of Acronyms and Abbreviations

ACC	Accuracy
CA	Classification Accuracy
CDF	Cumulative Distribution Function
CIO	Chief Information Officer
C&C	Command and Control
CV	Cross Validation
DBL	Domain Block List
DNS	Domain Name Service
DNSBL	Domain Name System Block Lists
DoD	Department of Defense
DON	Department of Navy
DOS	Denial of Service
FN	False Negative
FP	False Positive
IG	Information Gain
IP	Internet Protocol
MAP	Maximum A-posteriori Probability
MWID	Malicious Website Identification
NPS	Naval Postgraduate School
PBL	Policy Block List
PII	Personally Identifiable Information
PPV	Positive Predictive Value
RBL	Real-time Blackhole List
SENS	Sensitivity

SORBS	Spam and Open Relay Blocking System
SBL	Spamhaus Block List
SMTP	Simple Mail Transfer Protocol
STC	Spatial-Temporal Correlation
TCP	Transmission Control Protocol
TN	True Negative
TP	True Positive
TTAD	Transport Traffic Analysis Daemon
URL	Uniform Resource Locator
XBL	Exploits Block List

Acknowledgements

First and foremost, I must thank my loving wife, Michelle, who has sacrificed the most these past two years. Her unwavering patience, understanding, and unconditional support were crucial to me making it through this program and finishing the studies for my degree. Words can never begin to describe my gratitude for everything she has done and continues to do for me.

I would like to express my sincere gratitude and appreciation to my thesis advisors, Dr. Robert Beverly and Dr. Geoffrey Xie. Your expertise and exceptional level of knowledge are unparalleled! I can not begin to thank you both enough for your constant encouragement, patience, keen technical insight, and the superb guidance you provided me toward making this thesis possible. Thank you!

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1: Introduction

In today's advanced digital age of high-tech sophisticated and integrated networking systems, one consistent peril that most organizations and common computer users face is "abusive" infrastructure. Despite the concerted efforts of industry and academia, the threats posed by abusive sites remain a constant problem. This thesis explores a technique to discern hosts that support abusive web infrastructure by discovering differentiating characteristics of their traffic flows. In particular, TCP-level flow statistics can reveal hosts that are both congested and poorly provisioned–features that this work shows are frequently associated with abusive systems. Importantly, the approach in this thesis is privacy preserving, allowing deployment in new domains, while being complimentary to existing abusive traffic detection methods.

Abusive infrastructure on the network consists of systems that function subversively to assist and/or advance malicious or illegal activities in various forms. Abusive infrastructure can include, but is not limited to, web sites that cater to, sponsor, or actively participate in supporting various forms of malicious content. Thus, abusive infrastructure commonly includes botnets and complicit service providers who may produce abusive traffic that includes email spam, malware, scam hosting, denial of service, scanners, etc. Underscoring the impact of abusive traffic, in 2010, more than 89% of all email messages on the Internet were attributed to spam. Furthermore, about 88% of these spam messages were sent with the help of botnets [1].

Much time and effort continues to be expended toward methods of limiting or negating the impact of malicious type activities that occur throughout the Internet. Existing methods, some of which are covered in Chapter 2, have been developed and used to deter abusive traffic. Unfortunately, many deployed techniques are often circumvented by adversaries who quickly adapt by evading detection/blocking methods, or by developing new schemes designed to go undetected. Thus, despite significant advances in technology and methodology, malicious activities, and resulting abusive traffic, remain prevalent. Not only is abusive traffic a constant source of frustration, it can have potentially more serious consequences when used to subvert security policies and carry out malicious acts aimed at harming systems or the data they store.

Being able to identify abusive traffic, especially traffic not previously identified, at an early stage to block its ingress into critical information systems infrastructure, and have the ability to block egress from a specific system or user that is bound for an abusive infrastructure, is the primary goal of this research.

1.1 Significance of Work to the DoD

Within the Department of Defense (DoD), unsolicited bulk email, for example, spam, phishing, or spoofing, is a constant source frustration and economic burden. According to the Department of Navy (DON) Chief Information Officer (CIO), the protection of vital DON networks and information, given the popularity of malware and cyber terrorism, continues to be a challenge. In his report in 2009, the CIO goes on to indicate that each month the DON blocks approximately 9 million spam messages and detects more than 1,200 intrusion attempts, while blocking an average of 60 viruses monthly before they can infect the Navy and Marine Corps network [2].

The particular burden that is placed upon Naval organizations comes in the form of wasted man hours, which is the time, effort and money spent to eliminate this type of email. The time spent toward remediation efforts of systems that have become infected, as a result of the execution of a malicious link within the unsolicited email or the visit to a malicious web site, can vary greatly depending on the particular infection and method of deployment. This is even more so a challenge for those units, for example, the Navy, where limited bandwidth and satellite connectivity is the primary means of connectivity for communications and networking, both tactical and non-tactical. Spam ties up crucial limited bandwidth and network resources, as spam adds to the total cost of network operations. Spam has and continues to be disruptive in that it can contribute to network mail server crashes or Denial of Service (DOS).

Navy information systems that become compromised as the result of malware or malicious activities can lead to anything from the minor inconvenience associated with a temporary loss of service to a major catastrophe with systems being impacted for long periods of time. Like many organizations, the Navy relies on these information systems to a great extent and cannot adequately perform many of its mission sets when critical information systems are rendered inoperable.

Though Navy information systems, like other armed services and government agencies, have networks internal to the specific organization, they are still connected to the Internet. This in turn opens up vulnerabilities that any malicious entity might attempt to exploit. Even the common user offers up a level of vulnerability through the websites he may visit or a particular spam email that happen to have arrived in their inbox. All of these threaten to compromise the integrity, availability, and accessibility of our networks and critical information systems.

In serious cases involving compromised systems, not only can the system be rendered unavailable, it is also possible that the malicious activities would involve data exfiltration and loss of official use only or Personally Identifiable Information (PII).

The impact of abusive infrastructure is not confined to spam. Compromised hosts may also contribute to scam hosting infrastructure, including web and Domain Name Service (DNS) capabilities, or may directly participate in harvesting sensitive data or sending attack traffic.

1.2 Scope

In our research, we will concentrate on conducting transport analysis on real-world active network traffic and apply recently devised packet-level TCP transport classifiers to discover abusive traffic flows, especially those not found via traditional methods, e.g., blocklists. Through transport analysis methods, we will gain a better understanding of the real-world performance of our classifier, improve upon both the classifier and fine tune existing methods, and seek to develop new analysis techniques to identify and expose malicious web hosts.

Malicious Website Identification (MWID) or MalWebID is our collection of tools, programs, and processes that have been developed to identify abusive web sites and infrastructure as part of this thesis.

In particular, focusing on and understanding the practical, real-world performance of using transport traffic features to identify malicious web sites found in live traffic streams, we aim to introduce a novel approach to identifying abusive infrastructure. Our approach and use of MWID's technique may in fact help redefine how abusive infrastructure is identified and help pave the way to better methods of blocking these types of sites.

1.3 Goals

Leveraging the research previously performed in the papers outlined in Chapter 2 and understanding the different approaches taken in identifying and combating malicious content, our research will demonstrate a novel technique that utilizes the features of network traffic for detecting and classifying abusive infrastructure. Our goals are to explore the following:

- Determine if it is possible to accurately detect and identify a malicious web host in real-time, via an automated means, even when the malicious host is currently unknown or does not currently exist on any known “black list” or “block list”, which are both discussed in Chapters 2 and 3.

- Introduce the MWID tool set and demonstrate its capabilities as a fast and efficient means of collecting and analyzing live network traffic to accurately detect and identify malicious or abusive infrastructures, all while preserving privacy.
- Determine if our technique of transport traffic analysis can provide a mechanism to help aid existing host-based security systems and provide an added layer to improve protection.
- Determine if new techniques or classification algorithms are required in order to produce the overall Classification Accuracy (CA) to demonstrate the effectiveness of our approach at correctly identifying and classifying malicious hosts.

1.4 Results

By conducting transport analysis on real-world “active” network traffic and applying the use of recently devised packet-level TCP transport classifiers to discover abusive traffic flows, we are able to provide the following contributions through the use of MWID:

1. present a fast and efficient (passive) tool that is able to run on a variety of systems, primarily connected or installed at the core of a network, that will help augment existing network security infrastructure by detecting abusive traffic and infrastructure.
2. then demonstrate the effective means of detecting abusive traffic through the statistical analysis of key transport layer traffic features.
3. provide detailed analysis of our findings while working with live data captures, specifically an overall classification accuracy that exceeds 95 percent.
4. offer recommendations for future work.

This tool represents a step toward combating malicious and abusive infrastructure. With the proper integration within a network’s security architecture, such as that of the DoD, MWID may provide a viable means to help reduce the threat posed by many types of abusive infrastructure.

1.5 Thesis Organization

The remaining sections of this thesis are broken down into the following key chapters:

- Chapter 2 provides background information to tie in the scope and context of the problem. We discuss previous research performed in the area of detecting and identifying abusive infrastructure and perform a brief comparison of our work to that the previous work.

- Chapter 3 provides a synopsis of our experiment methodology, detailing the steps that were conducted during the research. Specifically:
  – Phase I: Data Collection and Development
  – Phase II: Testing and Refinement of Classifier
  – Phase III: Evaluate Live Captures
  – Phase IV: Test and Reevaluate Data Sets
- Chapter 4 provides a review and overall assessment of the results found during the research and experiments.
- Chapter 5 provides a summary of conclusions and outline future work of importance in stopping abusive infrastructure.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2: Background

Over the past few years there has been a dizzying growth in the use of the Internet, primarily in the area of of web services, including cloud computing, blogs, video services, and popular social media sites. In December 2000, there were near 361 million Internet users and by the end of December 2012, there were over 2.4 billion users, which equates to a 566% increase in Internet users [3].

Unfortunately, systems connected to the Internet are constantly in danger and run the risk of exposure to malicious acts, which include phishing, malware, and scams, all of which pose a reoccurring threat.

While seemingly able to manifest in a variety of different forms, attacks can cause problems ranging from a simple inconvenience to something more severe, such as extracting sensitive personal data or the permanent deletion of data. Considering the negative impact that malicious activity has and continues to have on our networks and information systems, a great deal of work has taken place within academia related to detecting abusive infrastructure and thwarting its ability to adversely impact critical computing infrastructure.

2.1 Prior Work

Considering the enormous efforts that have been expended toward combating abusive infrastructure, significant progress has been made in the field, to include traffic content monitoring, detecting and documenting abusive networks, and identifying abusive communications patterns. Specific research that has had the greatest influence on this thesis are in the area of auto-learning of Simple Mail Transfer Protocol (SMTP) TCP transport-layer features for spam [4], transport traffic analysis of abusive infrastructure [5], exploiting transport-level characteristics of spam [6], and BotMiner [7], and more recently BotFinder [8]. Some key areas of interest are summarized here.

2.1.1 Transport Traffic Analysis

Recent work by Nolan on transport traffic analysis of abusive infrastructure [5] investigated a passive analysis technique to identify discriminating features of abusive traffic, using per-packet

TCP header and timing features, to identify congestion, flow-control, and other key flow characteristics. Because abusive infrastructure is often poorly connected, or connected to residential networks, these characteristics were used to identify links that are typically associated with that of an abusive infrastructure. The passive approach shown in his research did not rely on either content inspection, Command and Control (C&C) signatures, or that of sender reputation, thus maintaining data privacy and facilitating wider potential deployment scenarios. Further, the technique is complementary to other methods. The technique showed promise, in that he achieved a classification accuracy up to 94% in detecting abusive infrastructure, with only a 3% false positive rate. Unfortunately, the results were limited in that they were inclusive only of traffic captured in an experimental environment, and used Alexa as the source of legitimate websites, rather than the true mix of traffic an enterprise or organization might experience.

This thesis builds upon Nolan's work, where we perform live network traffic analysis with the goal of identifying not only previously identified malicious hosts, which account for those that have been identified and are currently on a blacklist, but more significantly identifying unknown malicious hosts, those that have not been seen or previously documented, i.e., truly abusive hosts that do not appear in any current blacklists.

Blacklists, also known as blocklists, are simply a list that consist of either Internet Protocol (IP) addresses, host names, or a combination of both and have been determined by a variety of means to be malicious in nature. i.e., IP addresses that are identified as generating spam can be blacklisted and are blocked from being able to deliver email. In most circumstances, when an IP address or host name is blacklisted, no notice is sent to the user or owner.

2.1.2 Auto-learning

Kakavelakis *et al.* introduces auto-learning using the transport-layer features, by developing an application that performs transport traffic analysis aimed at the spam reception problem. The auto-learning aspect is the process of building up the classification model based on exemplar e-mail messages whose scores exceed certain threshold values [4]. They take their training data, which consists of data that is clearly spam and clearly ham, and use the associated flow-features to train Spamflow. Working off of the premise that for a spam campaign to be financially viable, the host server must maximize its output of spam traffic. This in turn means that the host machines produce an exorbitant amount of network traffic, which can then be identified and disassociated from normal network traffic. Here, they design SpamFlow [6] as a plugin to SpamAssassin, a well-known learning-based spam filter used to identify spam signatures.

SpamFlow is a unique tool (analyzer) that listened on an interface and captured flows of data, to then build features for each flow of traffic seen. For spammers, the properties of the feature properties of the transport-layer prove to be different than that of normal user traffic and able to be identified. Once these features are gathered and processed using an auto-learning classifier, they were able to accurately identify and distinguish spam traffic from that of general use traffic, with a greater than 95% accuracy rate.

This thesis uses the method performed by Kakavelakis *et al.* and Nolan, but as applied to the problem of specifically identifying abusive infrastructure. Using statistical traffic signal characterization as a discriminator for spam has proven to be effective. We believe that by capturing specific traffic features from each tcp flow and conducting statistical analysis on those features, previously labeled as good or bad, will show to be an effective tool at differentiating between non-malicious and malicious and traffic, independent of content or location.

Building upon this previous work and recently developed packet-level TCP transport classifiers, our research involves methods applied to live network traffic to discover not only "known" abusive traffic flows, i.e., those in blacklists, but also those not found via traditional methods, i.e., hosts not in existing blacklists. To do this, we utilize a newly revised version of SpamFlow, now known as TTAD, which is a crucial tool used in this research and necessary for successfully generating the feature vectors that are used for classifying traffic as either good or abusive/malicious (bad). TTAD avoids performing content analysis, reputation analysis, and maintains privacy of data, allowing for a deployment of the tool near the network core.

2.1.3 Malware Detection Systems

Tegeler *et al.* presented BOTFINDER, which is a newly devised malware detection mechanism that is primarily based off of statistical network flow analysis [8]. In their implementation of a unsupervised auto-learning tool, they perform a training phase on predefined known data sets about bots. The BOTFINDER tool then creates models, which are based on the statistical features found in the C&C communication. The C&C path is simply the channel by which malicious software components are coordinated, primarily by a single source known as a Botmaster. Through their analysis, they show that BOTFINDER is able to detect malware infections with detection rates near 80%, based on traffic pattern analysis with no deep packet inspection.

There are some similarities to the work Tegeler *et al.* performed and the research performed as part of this thesis. Of key note are the use of traffic features, which we rely on for our flow labeling, and the ability to perform the statistical analysis without the need for deep packet inspection. This is another key advantage in our research, as we introduce the capability for a fast and efficient means of collecting and analyzing live network traffic to accurately detect and identify malicious or abusive infrastructures, all while preserving privacy (no deep packet inspection necessary.)

2.1.4 Image Analysis

In the research conducted by Andersen *et al.* we are introduced to a technique called Spamscatter [9], which is used to characterize scam hosts and identify the infrastructure that supports spam. The method proposed in this thesis would be able to identify the hosts of the scam sites. These hosts may be serving or hosting multiple scams, being used to advertise spam, or serve as a spam type server and prevent delivery.

In our approach, we gather specific traffic features from each HTTP request. We then perform a statistical analysis to apply discriminators to the transport layer features, which aid in identifying malicious or abusive infrastructure. Our approach, unlike Spamscatter, will not involve any type of image analysis, and will rely on key traffic features to help identify the type of traffic.

2.1.5 Communications Patterns

Using clustering analysis of communication patterns [10] Gu *et al.* created a system that captures communication structure from network traffic, specifically dealing with bots. They created BotSniffer, which uses Spatial-Temporal Correlation (STC) techniques that provide a means of identifying bots without any prior knowledge of that bot or a botnet.

BotSniffer consists of two separate components, one of which is the monitor engine and the second is the STC.

Combined with IDS like functionality, they use clustering of communication patterns to differentiate between that of good hosts and malicious hosts. The monitor engine was placed at the network edge or perimeter and performs several key functions. It would detect and log any suspicious connection C&C protocol and response behaviors, which are forwarded to the STC for group anaylsis. The STC engine processes two types of communication techniques (IRC and HTTP) into specialize algorithms to perform there communications checks.

This research does not look at the communications structure of malicious activity or that of abusive infrastruction and is focused on the properties of the traffic streams, ignoring the application-layer content and IP addresses to identifying abusive infrastructure. The important difference enables our proposed technique to operate, without complete knowledge of the data or message, either in the core or at the edge of the network.

2.1.6 Behavioral Analysis

RB-Seeker [11], introduced by Hu *et al.* is a tool that can detect compromised computers, which are being used as a redirection or proxy server for a botnet, with a very high accuracy rate. The RB-Seeker tool performs analysis on the behavioral attributes that are extracted from DNS queries of the redirected domains. While this research does not look at any behavioral attributes, prior work by Nolan [5] showed that many websites, not just abusive ones, use redirection in order to control traffic flow, suggesting that such techniques alone may not be currently viable to identify bots. Similar to RB-Seeker, this research looks at finding discriminatory attributes of redirection, however; instead of cataloging DNS queries, we are concerned with transport traffic features for each HTTP request.

2.1.7 Online Learning

In the research by Ma *et al.* they look at online learning techniques for detecting malicious infrastructure by using features that are associated with URLs [12]. The problem they had to overcome was the fact that URL features had no context or content. They looked at two different feature sets, Lexical and Host-based, which contain URL information that is easy to obtain. The Lexical features involve the appearance of the URL, where the address is not in a format that is generally associated with that of a normal URL (i.e., www.google.com).

Their research involved a variety of features, including black-lists, heuristics, domain name registration, host properties and lexical properties. Looking at a few of these key aspects, we see that the black-lists they used were some of the same that we use in our research (SORBS, Spamhaus, etc.). They used WHOIS to gather information on the sites registrar, registrant, and dates. The host-based features they looked at provided information concerning the host web site, such as where the malicious code is being hosted, who might own it, and how it is being managed. The lexical features are simply tokens in the URL hostname plus the path, the length of the URL and number of dots in the notation.

The describe their URL classification system as one that extracts URL source data, processes it through a feature collection routine (extracting key features) and processing the combined URL and features through their classifier, similar to the method we use to process our traffic flows.

However, a key difference in the research by Ma *et al.* and our research, is that they are looking at features of URLs and attempting to determine, based off of the features, whether the URL is from an abusive site or not. Our approach is different in that, we extract 21 feature vectors from the flows of traffic, process the flow host names and IP addresses separately to determine labeling (good, bad or unknown), merge the two pieces of information together to process via our learner/classifier model. We then demonstrate the ability to actively identify malicious hosts, which have not previously been identified.

2.1.8 Signatures and Characteristics

Xie *et al.* looked at signatures and characteristics [13] which involved a spam signature generation by URL analysis and the development of a system, AutoRE, to identify and characterize botnets automatically, which is based on mining spam URL data embedded in emails. This thesis research is more involved than what Xie *et al.* was trying to accomplish with AutoRE. We are looking at the behavior of network traffic and avoid content analysis, whereby we preserve privacy. AutoRE only looked at the URL of a domain and disregards all other important evidence that was discovered in the traffic flow.

By leveraging the research performed in the above papers and through better understanding of the prior approaches and vantage points taken towards identifying and combating abusive infrastructure, our approach outlined in this thesis will be unique and stand apart.

2.2 Machine Learning / Supervised Classifiers

Of the variety of different methods that have been devised to detect abusive infrastructure, our research is based on supervised learning. In supervised machine learning, an algorithm is sought that can learn from external data instances and is able to form a hypothesis, followed by prediction about additional data that has no current label or classification [14]. This means that by building a model of our labeled "known" data, our constructed classifier is then used to assign labels to future "unknown" test data. If our data has known labels then the learning is called supervised and conversely, in unsupervised learning the data has no labels. See table 2.1 for an example of supervised learning.

We use extracted transport traffic features as input to supervised learning algorithms (train-then-test method). Specifically, Naïve Bayes and Tree Learner algorithms are used as our trainers, processing our labeled data and then used to predict the labels for our set of unknown data types. The complete methodology is outlined in chapter 3.

Table 2.1: Supervised Learning Feature Model

Flow	**Feature1**	**Feature2**	**Feature$_n$**	**Classified**
Data 1	xx	xx	xx	GOOD
Data 2	xx	xx	xx	BAD
Data 3	xx	xx	xx	GOOD
Data$_n$	xx	xx	xx	GOOD

2.3 Classification Techniques

We use two well-known supervised learning algorithms (classifiers) to process the data which was extracted from the processed packet captures, Naïve Bayes and Decision Tree, which are employed through the use third-party software [15].

2.3.1 Naïve Bayes

The Naïve Bayes classifier, often called the Bayesian classifier, is one that is probabilistic, which generally has a short computational time for training [14], and estimates conditional probabilities from our training data, then uses that to label any new instances of data.

The naïve assumption made by Bayes is that each feature is conditionally independent of the other features, given a particular class variable. Bayes theorem is defined as [16]:

$$P(C = c_k|\overrightarrow{F} = \overrightarrow{f}) = \frac{P(\overrightarrow{F} = \overrightarrow{f}|C = c_k)P(C = c_k)}{P(\overrightarrow{F} = \overrightarrow{f})}$$

Label definitions:

- C = class variable
- c_k = variable (“GOOD” or “BAD”) labeled for each individual flow
- $\overrightarrow{F}$ = feature vector
- $\overrightarrow{f}$ = flow vector
- $f_1, ..., f_n$ = attribute values

Next, we must apply the independence assumption with the following:

$$P(\overrightarrow{F} = \overrightarrow{f}|C = c_k) = \prod_i P(\overrightarrow{F_i} = \overrightarrow{f_i}|C = c_k)$$

Applying a decision rule [17], which is the Maximum A-posteriori Probability (MAP), we have:

$$c = classify(f_1, f_2, ..., f_n)$$

$$= \underset{k=(BAD,GOOD)}{\operatorname{argmax}} (P(C = c_k | \overrightarrow{F} = \overrightarrow{f}))$$

$$= \underset{k=(BAD,GOOD)}{\operatorname{argmax}} \left(\frac{P(\overrightarrow{F} = \overrightarrow{f} | C = c_k) P(C = c_k)}{P(\overrightarrow{F} = \overrightarrow{f})} \right)$$

$$= \underset{k=(BAD,GOOD)}{\operatorname{argmax}} \left(P(C = c_k) \prod_i P(\overrightarrow{F} = \overrightarrow{f} | C = c_k) \right)$$

Here, the following applies:

- $P(C = c)$ = prior probability
  – ratio of examples belonging in a class (c) to the total examples
- F_i = number of vectors
- f_i = vector value
- c_k = particular class – conditional probability is the ratio of vectors with a value belonging to a particular class to the total vectors belonging to a particular class

When dealing with continuous values, we must assume a normal distribution [18], which we defined as:

$$P(\overrightarrow{F} = \overrightarrow{f} | C = c_k) = g(\chi_i; \mu_i, c_k, \sigma_i, c_k)$$

$$g(\chi; \mu_X, \sigma_X) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

In simplistic terms, it assigns the most likely classification to unlabeled data, based on the data's distinct features or characteristics. Despite often violating the conditional independence rule, naïve Bayes generally performs quite well under a variety of problem sets.

Key advantages of naïve Bayes:

- Handles quantitative and discrete data
- Handles missing values by ignoring the instance
- Fast and space efficient
- Not sensitive to irrelevant features

Disadvantages of naïve Bayes:

- If conditional probability is zero
- Assumes independence of features

Unfortunately, as we discovered during our testing, naïve Bayes did not perform as well as the Decision Tree algorithm, often labeling data incorrectly, which may have been due to the existence of multiple sets of the same data, resulting in numerous inconsistently labeled traffic flows.

2.3.2 Decision Tree

The Decision Tree is a predictive model that maps observations about a particular piece of data to conclusions about the data's value. Using the traffic features for each flow of data in the *labeled.csv* file, the Decision Tree creates a model that predicts the value of our data based on those features. The manner in which a decision tree algorithm works makes it a kind of greedy algorithm, because it uses a top-down recursive manner to determine the tree structure [19].

Decision trees uses information gain in order to build a tree from a list of classified examples, evaluating each feature in the feature vector as follows:

- selects the best feature as the root
- creates a descendant node for each value of a different feature
- repeats recursively for each node
- ends when:
  - vectors of a current node are of the same class
  - no features remain

Using a decision tree, we are able to classify flows that have unknown attribute values by estimating the probability of the various possible results. For evaluating features and in order to select the best feature, decision trees generally use attribute Information Gain (IG) [18]. With a set *S* of training examples, the IG of features *A* is:

$$IG(A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where:

$$Entropy(S) = -\sum_{i=1}^{c} p_i \log_2 p_i$$

Consider a set of attribute *A* values, where:

- S_v = subset of *S*
- p_i = ratio of the number of examples belonging to class *i* to the total number of examples provided.

The goal is to simply maximize the IG of a selected attribute, done by minimizing the entropy of S_v. However, IG unfortunately selects attributes with a large set of values and has to be overcome. This can be accomplished by utilizing the information-gain ratio [20]. The following formula is offered:

$$GR(A) = \frac{IG(A)}{IV(A)}$$

where:

$$IV(A) = -\sum_{i=1}^{|A|} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

For the testing we perform, the Decision Tree performed best, and was our main choice of classifier algorithm.

2.4 Real-time Blackhole List and Alexa

In this section we discuss the individual Real-time Blackhole Lists (RBLs) and the Alexa Top 1 Million List, which we use in our labeling process in Chapter 3.

2.4.1 Spamhaus

The Spamhaus website provides us with several “realtime” spam-blocking databases that are utilized in industry to combat spam sent via the Internet [21]. The key databases that we use in this research are the Spamhaus Block List (SBL), Exploits Block List (XBL), Domain Block List (DBL) and the Policy Block List (PBL).

Using Spamhaus, we process all of the flows from our captures, to include the fetcher data. Any host names discovered currently listed on a Spamhaus blacklist was labeled as “BAD.” Conversely, any host names that are not found on a blacklist but found on Alexa are labeled as “GOOD.” As previously mentioned, in the event that we encounter a conflict where a host name is listed on an RBL as malicious and the same host name appears on the Alexa top 1 million list, it is defaulted to malicious or “BAD.”

2.4.2 Spam and Open Relay Blocking System (SORBS)

The SORBS site provides us with Domain Name System Block Lists (DNSBL), which are used throughout industry to help identify and block spam email, to help prevent phishing attacks, as-well-as combat other types of malicious email [22]. The lists, which is over 12 million hosts, are maintained by IP address, to include those that are dynamically allocated, and can include any email server that is suspected sending spam, has been hacked, has been hijacked, or suspected of having any Trojan Horse infection.

2.4.3 ZEN

ZEN, a product under Spamhaus, combines all of the Spamhaus IP-based DNSBLs into a single all encompassing comprehensive blocklist [21], containing the SBL, acSBLCSS, XBL and PBL blocklists. This simply allows for greater efficiency and faster database querying for host resolution.

2.4.4 Malware Domain

As the name suggests, the site provides a list of malicious domains that can be utilized to help determine whether domain caters to malicious or abusive traffic [23]. The list of malicious hosts was downloaded and referenced locally when performing our research, and checked weekly for updates. The list would range in size, but generally contains approximately 1200 malicious host names. Only during capture 2 were we able to resolve a single malicious host name from Malware Domain.

2.4.5 PhishTank

PhishTank is a site dedicated to providing information about phishing that occurs throughout the Internet [24]. Phishing is any fraudulent attempt to steal another individuals personal identifiable information. Like that of the malware domain list, we downloaded and referenced the PhishTank list of malicious hosts locally during our research. The list was checked weekly for updates. Generally, there were approximately 12000 hosts listed as malicious.

2.4.6 Alexa

We utilize the the Alexa Top 1 Million database as a foundation to measure and build a list of non-malicious sites, for comparison in our research, though we understand that it is not a certainty that all sites listed in Alexa are in fact non-malicious. It provides an important ground truth as to hosts we consider non-malicious.

CHAPTER 3: Methodology

Malicious Website Identification (MWID) leverages recently devised packet-level TCP transport classifiers to discover abusive traffic flows in live traffic, especially those not found via traditional methods (e.g., RBLs). This chapter details the methodology employed in this thesis to implement and test MWID. We first summarize the operation of MWID, then describe the four real-world packet captures analyzed. Next, we explain the basic operation of the transport-layer feature extraction and classification engine, as well as the use of a web site "fetcher" program that, when combined with an operational honeypot, ensures that the packet captures contain known abusive flows in addition to the user-generated traffic. Finally, we describe the use of multiple white and block lists for obtaining other sources of ground-truth, which we will use as a basis to understand the performance of MWID.

3.1 Overview

This section provides a high-level overview of the operation of the MWID system. Table 3.1 summarizes the various programs used to analyze the data, while figure 3.1 illustrates the overall collection, processing and analysis work flow implemented by MWID.

Table 3.1: MalWebID Program Process

Program	**Input**	**Output**	**Description**
ttad	pcap.list	ttad.out	Extract per-flow traffic features from *.pcap*
mwid_pcap	pcap.list	hosts.out	Extract per-flow HTTP host from *.pcap*
mwid_label	hosts.out	flows.out	Label flows with RBL host and IP lookups
mwid_merge	flows.out ttad.out	labeled.csv unknown.csv	Merge flow and feature data
mwid_class	labeled.csv unknown.csv	reclassed.csv	Classify unknown flows using RBL-based training model

1. Using `tcpdump`, we capture multiple *libpcap* formatted traffic traces, several of which exceed 24 hours in duration (§3.2):
    - As part of several captures, we utilized our external honeypot to provide URLs that are presumed to be malicious, for the purpose of injecting them into the capture. In this way, we ensure that the flows contain traffic from malicious hosts and allows for better classification performance testing.

- Process each capture with `ttad` to accumulate per-flow traffic characteristics and features (table 3.3)

  – *ttad.out* file is created
- Process multiple .pcap files with `mwid_pcap.py` to extract per-flow TCP header, HTTP application-layer host information, IP addresses, and TCP port numbers (table 3.4)

  – *hosts.out* file is created, which includes all .pcap data

2. Perform data labeling:
   - Process the hosts.out file using `mwid_label.py` to determine proper classification or label for each host flow. The labeling is accomplished by looking each host name and associated destination IP address up on key RBL list sites (§2.4). For a list of non-malicious sites, we utilize the Alexa Top 1 million list. Sites (by IP or host name) discovered on an RBL are labeled as "BAD" and sites listed on Alexa are labeled as "GOOD;" otherwise the flow is marked as "UNKNOWN" (§2.1).

     – *flows.out* file is created
3. Perform data merge:
   - Process the flows.out and ttad.out using `mwid_merge.py` to combine per-flow traffic characteristics with that of the associated TCP header information

     – *labeled.csv* file is created, which contains "GOOD" and "BAD" traffic flows

     – *unknown.csv* file is created, which contains only "UNKNOWN" traffic flows
4. Classification:
   - The *labeled.csv* file used as training data
   - The *unknown.csv* file used as testing data
   - Both files are processed using `mwid_class.py`, which invokes third-party software [15] to perform statistical analysis on our training data to build a model. The learned model then determines the label for each unknown data flow

     – *reclassed.csv* file is created, where all unknown flows of data are labeled
5. Sampling:
   - Given the reclassification results contained in *reclassed.csv*, manual sampling is performed on randomly selected flows, with an even split between flows labeled "GOOD" and "BAD." A human uses a standard web browser to visit the sites and determine whether it is in fact malicious or non-malicious. This necessary and time critical step aids provides ground-truth to better assess the accuracy MWID. As a majority of malicious sites are short-lived, we ensure that sampling occurs soon

after obtaining data so that sites are visited while operational.

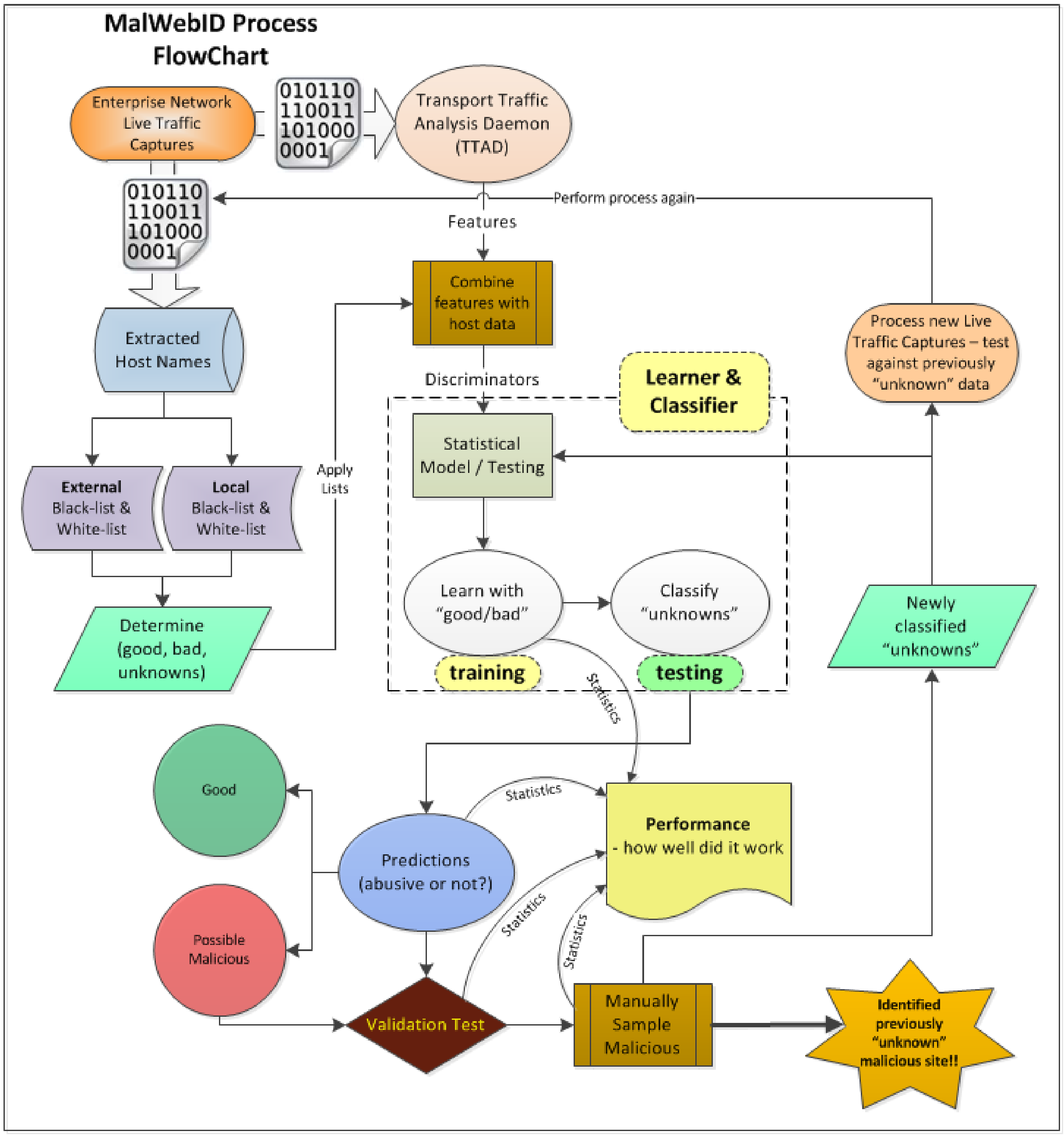

MalWebID Process FlowChart
Enterprise Network Live Traffic Captures
010110 110011 101000 0001
Transport Traffic Analysis Daemon (TTAD)
Perform process again
Features
Combine features with host data
Extracted Host Names
Process new Live Traffic Captures – test against previously "unknown" data
Discriminators
Learner & Classifier
Statistical Model / Testing
External Black-list & White-list
Local Black-list & White-list
Apply Lists
Learn with "good/bad"
Classify "unknowns"
Newly classified "unknowns"
Determine (good, bad, unknowns)
training
testing
Statistics
Good
Statistics
Performance - how well did it work
Predictions (abusive or not?)
Statistics
Statistics
Possible Malicious
Validation Test
Manually Sample Malicious
Identified previously "unknown" malicious site!!

Figure 3.1: MalWebID Flow Process.

3.2 Packet Captures

We performed several live network packet captures, using a network tap on the Naval Postgraduate School (NPS) enterprise network, over a period of seven months. This connection was able to capture all inbound and outbound network traffic for the NPS Computer Science and Operations Research Departments, but not the school as a whole. Three of our captures include traffic generated by an automated program, the "fetcher," that automatically pulls content from web sites discovered by a honeypot. A honeypot is simply a computer system that has been set up to appear as though it is part of a normal network, where it could contain some information or data that would be useful to an intruder or attacker. However, the honeypot is actually an isolated and monitored computer, used in this case for research to provide URL information from sites that visit it.

We performed the captures on different days, with the requirement that captures were performed during peak times, where we knew that a majority of users are accessing the network (i.e., weekdays vice weekends). Table 3.2 summarizes the captured data analyzed in this thesis. Despite the capture timeframe, we found that some captures that ran longer, (i.e., 46hrs), did not capture as much data as a capture that ran for a shorter period of time. This can be attributed to several reasons, specifically the number of users and amount of utilization (traffic on the network) during those particular time frames obviously play a critical role in that regard.

Table 3.2: Packet Captures

Capture Number	**Time Period**	**Capture Duration**	**Kilo Packets**	**Mega Bytes**	**Description**
Capture 1	20Sep12	10hrs	6475.6	4779.6	NPS users + injected URLs
Capture 2	28-29Nov12	46hrs	608.5	6805.9	NPS users only
Capture 3	07Mar13	24hrs	1650.2	3969.8	NPS users + injected URLs
Capture 4	08Mar13	24hrs	650.3	1985.2	NPS users + injected URLs

3.2.1 Capture 1

The first series of captures, conducted on 20SEP2012, incorporated the use of the Fetcher program created by Nolan [5], which is used to "fetch" web addresses (hosts) that have been extracted from our external honeypot. See figure 4.1 for the overall flow process.

Once we initiate a packet capture session, we simultaneously use Fetcher to actively fetch or visit the URLs that were acquired from the honeypot, which we consider to be malicious in nature. In this test, we fetch 2,983 malicious URLs, the flows of which are mixed with the normal user traffic being captured at NPS. Though the honeypot URLs are initially considered malicious, we further manually remove hostnames that are actually non-malicious (which can occur when spammers utilize real URLs within spam messages). Table 3.5 provides details associated with the URLs injected by Fetcher.

Considering that a majority of the traffic outbound from the NPS network is not destined for abusive infrastructure, simply due to the nature of a military environment at the institution and the expectations levied on those using the network resources, it was necessary to introduce a variety of suspected malicious URLs to perform our analysis on. We utilize Fetcher for this task. During our capture process, Fetcher initiates port 80 calls to each of the hosts previously discovered from our honeypot, thus allowing that flow to be present in the capture. By utilizing Fetcher, we have a much greater certainty of encountering malicious content than without. This approach resulted in over 36GB of data to be used for our first series of tests. Additional details regarding Fetcher and the data processed are discussed in section 3.3.3.

3.2.2 Capture 2

In our second set of packet captures, which ran for 46 hours (28-29Nov2012), we acquired over 33GB of data, spanning 92 saved *.pcap* files containing 608,409 hosts names and 2,419,532 individual flows of traffic. See figure 4.2 for the overall process flow.

For these captures, we did not want to introduce any additional URLs from our honeypot. The decision to capture only NPS traffic was in order to evaluate our methods and have a baseline test where we process only school traffic. This allows us analyze how well our classification process works in our testing phase, when there is generally not a large amount of malicious traffic.

Captures 3 and 4 will again introduce Fetcher injected URLs into the the capture process and the data is provided in table 3.2.

3.3 Programmatic Functions

Here we detail the processes performed to extract flows from our packet captures and the programmatic functions of MWID's specialized sub-routines (programs) that analyze and classify the traffic flows. Table 3.1 is provided as a quick reference for the programs used and functionality of each.

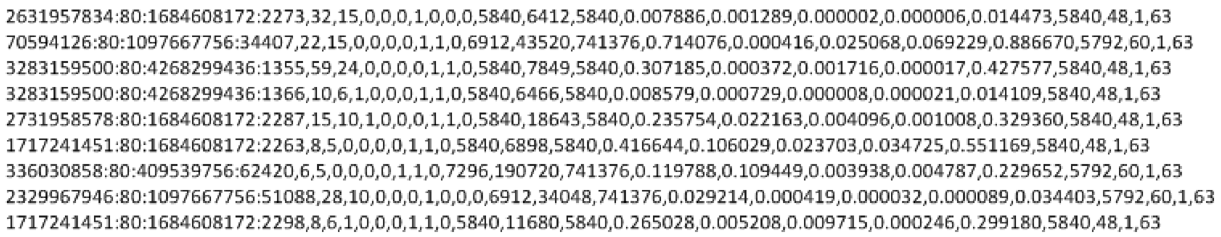
3.3.1 Traffic Features

Once we have obtained our packet captures, we must then process each *.pcap* with our TTAD program, which provides per-flow traffic characteristics and features. See table 3.3 for a detailed list of the specific traffic features we utilize. Once the *.pcap* files are processed, we accumulated distinct flows of traffic and output to *ttad.out*.

Table 3.3: TTAD Transport Traffic Features

Feature	**Description**
pktsin	Packets in from source and MTA
pktsout	Packets out to source and MTA
rxmtin	Retransmissions in from source and MTA
rxmtout	Retransmissions out to source and MTA
rstsin	TCP reset segments from source and MTA
rstsout	TCP reset segments to source and MTA
finsin	TCP segments with FIN bit set from source and MTA
finsout	TCP segments with FIN bit set, to source and MTA
zrwn	Number of times the receiver window went to zero
mrwn	Minimum receiver window over flow life
avgrwn	Average receiver window over flow life
initrcwn	Initial receiver window
idle	Maximum flow idle time
3whs	RTT of the TCP three-way handshake
jvar	Inter-packet arrival variance
rttv	Per-segment RTT variance
tdur	Flow duration (seconds)
wss	TCP SYN window size
syn	TCP SYN packet size
idf	Do not fragment the IP bit
ttl	Arriving IP time to live

An example output of the 21 distinct features that TTAD produces is provided in figure 3.2. All values are comma separated.



```
2631957834:80:1684608172:2273,32,15,0,0,0,1,0,0,0,5840,6412,5840,0.007886,0.001289,0.000002,0.000006,0.014473,5840,48,1,63
70594126:80:1097667756:34407,22,15,0,0,0,0,1,1,0,6912,43520,741376,0.714076,0.000416,0.025068,0.069229,0.886670,5792,60,1,63
3283159500:80:4268299436:1355,59,24,0,0,0,0,1,1,0,5840,7849,5840,0.307185,0.000372,0.001716,0.000017,0.427577,5840,48,1,63
3283159500:80:4268299436:1366,10,6,1,0,0,0,1,1,0,5840,6466,5840,0.008579,0.000729,0.000008,0.000021,0.014109,5840,48,1,63
2731958578:80:1684608172:2287,15,10,1,0,0,0,1,1,0,5840,18643,5840,0.235754,0.022163,0.004096,0.001008,0.329360,5840,48,1,63
1717241451:80:1684608172:2263,8,5,0,0,0,0,1,1,0,5840,6898,5840,0.416644,0.106029,0.023703,0.034725,0.551169,5840,48,1,63
336030858:80:409539756:62420,6,5,0,0,0,0,1,1,0,7296,190720,741376,0.119788,0.109449,0.003938,0.004787,0.229652,5792,60,1,63
2329967946:80:1097667756:51088,28,10,0,0,0,1,0,0,0,6912,34048,741376,0.029214,0.000419,0.000032,0.000089,0.034403,5792,60,1,63
1717241451:80:1684608172:2298,8,6,1,0,0,0,1,1,0,5840,11680,5840,0.265028,0.005208,0.009715,0.000246,0.299180,5840,48,1,63
```

Figure 3.2: Example Traffic Features Extracted by TTAD.

The first two sets of numbers provided by TTAD (i.e., 1705268294:80:2358812342:5256) represent the 32 bit integer format of the destination and source IP addresses and associated port numbers. The remaining numbers that make up the distinct characteristics for a particular flow follow the order of items as listed in table 3.3.

3.3.2 TCP Headers

Once we have processed the *.pcap* files to gather per-flow traffic features using `ttad`, we then process the same *.pcap* files using our `mwid_pcap` program. `mwid_pcap` reads in each flow of data from the *.pcap* file and extracts key header information, to include the destination and source IP addresses, destination and source ports, and HTTP-host name. In general, `mwid_pcap` utilizes a Python programming library known as dpkt, which is an ethernet packet decoding module that decodes single network packets and allows access to individual segments of the packet (i.e., destination IP, source IP, port numbers, host name, etc.).

Once we have processed all of the flows and obtained the necessary information in each packet, the information is output in a specific format to a *hosts.out* file. The output format is key in how we later process this data §3.3.3. An example of the output data is shown in table 3.4.

Table 3.4: TCP Header Information

Host Name	Source IP	Destination IP	Src Port	Destination Port
www.pandora.com	172.20.xxx.xxx	208.85.40.80	60057	80
www.amazon.com	172.20.xxx.xxx	72.21.194.212	51204	80
....	....	....	....	....

3.3.3 Host Labeling

We now take the *hosts.out* file produced by `mwid_pcap` and process it using our `mwid_label` program. The purpose of the program is to determine the appropriate label for each flow, which is “GOOD,” “BAD” or “UNKNOWN.” To determine the proper labeling, the program reads each individual line of the input file, takes the hostname and destination IP and performs lookups against several well-known online RBL sites and local databases. Specifically, the host names are processed using Spamhaus [21], Phishtank [24] and Malware Domain [23] RBLs.

Not only are we using multiple block-list sites to check each host name in our list, but we also perform an additional check of each flow by using the destination IP and process each using a corresponding RBL catering to IP address searches. These RBLs are SORBS [22] and ZEN, which are offerings from Spamhaus. By doing so, this gives us a better confidence level as to the appropriate classification label for each flow.

Additionally, the hosts are checked against the Alexa top 1 million sites list [25] for a baseline of presumably good web hosts. For the remainder of this paper, the terms "host names" and "hosts" are synonymous, as well as "block-list" and "black-list," which will be used interchangeably. Additional details concerning the RBLs and Alexa are discussed in §2.4.

Once we have processed all of the traffic flows, each flow is appended with two additional pieces of data, the name of the RBL reporting the site or IP as malicious, or Alexa if the site is non-malicious, followed by the appropriate label. A sample of the label flow is:

```
redditenhancementsuite.com,172.20.106.150,96.126.125.231,54256,80,ALEXA,GOOD
www.jamaicaobserver.com,172.20.109.160,208.88.246.123,55035,80,ALEXA,GOOD
tnt.doctorflig.ru,172.20.109.183,84.22.127.36,49317,80,SPAMHAUS,BAD
www.1.dat-e-baseonline.com,172.20.110.125,66.186.15.226,50658,80,,UNKNOWN
biwqrqr.mathdoctor.ru,172.20.109.183,84.22.127.36,54057,80,SORBS,BAD
```

As is evident in the above sample output, each flow has the appended with the name of the RBL or Alexa, as appropriate, and the label for that flow. The host name *tnt.doctorflig.ru*, for example was found on the Spamhaus RBL and is labeled as "BAD." The hosts that were found listed on Alexa are subsequently labeled as "GOOD." Any host name not found on a black-list or not found listed on Alexa are simply labeled as "UNKNOWN."

In table 3.5, we provide a summary of the preprocessed URLs that we use with Fetcher as injects during our packet captures routine.

Table 3.5: Fetcher Processed URLs

Test URLs	Unique Flows	Good	Bad	Unknown	Alexa	Conflicts
Test 1	1,179	57	278	844	61	4
Test 2	** Fetcher not used for test 2 **					
Test 3	3,766	36	3,195	535	38	2
Test 4	2,971	16	1,870	1,085	20	4

One interesting observation from our data labeling is the appearance of conflicts during our tests. These are where we find a host or IP address on both a black-list and on the Alexa list. In such cases, we default to labeling the flow as "BAD." We will discuss these more in-depth in Chapter 4.

Table 3.6 provides details of each packet capture session we processed and the overall results.

Table 3.6: Processed Capture Details

Captures	Unique Flows	Good	Bad	Unknown	Alexa	Conflicts
Capture 1	21,083	1,243	5,762	14,078	1,246	3
Capture 2	9,029	1,648	22	7,359	1,656	8
Capture 3	8,468	1,751	1,095	6,178	1,099	4
Capture 4	6,322	377	2,948	2,997	382	5

In both tables, we refer to the number of flows used as "unique," which is not indicative of the total number of flows reviewed, but simply refers to flows that are not duplicated. By duplicated, we mean that when we encounter two flows with the same host name. We only count and use one of the host names and associated IP addresses to do the search on an RBL or Alexa.

Fetcher Data

As discussed in section 3.2.1, Fetcher was utilized to inject flows of traffic into our capture process to help increase the number of malicious hosts encountered. Prior to performing the actual captures, we "preprocessed" a list of URLs extracted from our honeypot using `mwid_label`. Those results are included in Table 3.5 This was in order to give us a better insight and ground truth into the number of actual known "bad" hosts that were presently identified via blocklists as it relates to the honeypot data as a whole. We expect most flows recovered from our honeypot to be malicious and by determining whether the flows do in fact currently reside on a blacklist or how our methods identify them is key. Thus, these results are important as we will use them as an additional tool in our validation phase to help determine how accurately our classifier performs at labeling our data.

3.4 RBS and Alexa

This section is provided in order to help clarify some of the data in table 3.6. It is important to understand that `mwid_label` takes in list of data flows from *hosts.out*, where the data is divided into separate categories: host names and destination IP addresses associated with each host name.

While all host names are processed through Spamhaus, Malware Domain and Phishtank, the number of IP addresses processed through SORBS and ZEN will vary and will depend on certain variables encountered as the data is processed. Since multiple IP addresses can be associated with one host name, there will be a difference in the number of hosts that are processed as opposed to the number of IP addresses processed. Additionally, because the IP addresses are resolved by SORBS and ZEN separately, the numbers associated with each will generally not match, as some IPs would not resolve or timeout.

Aside from Alexa, which simply lists the total number of addresses found during a particular session, the RBLs will have listed the total of “BAD” sites that were found, “GOOD” sites that were found as they appeared on Alexa and not an RBL, and the “UNKNOWNS,” which did not meet the good or bad criteria.

Additional details concerning RBLs and Alexa can be found in Chapter 2 2.4.

3.5 Flow Feature Labeling

We now take our two output files, *flows.out* which has our labeled host names and *ttad.out* which contains all of the flows produced by `ttad`, and process them both using our `mwid_merge` program. This program takes the input files and labels the traffic flows by simply comparing each flow from *ttad.out* with that of the same tuple keys in our *flows.out* file.

Referring to the features in figure 3.2, when TTAD saves each flow to *ttad.out*, the IP addresses are in a 32 bit format along with the port numbers and associated flow features. `mwid_merge` will take each flow and strip the tuple of IP addresses and associated port number, convert the IP addresses into a dotted-decimal notation, and append the new tuple to the head of the flow. The tuple now consists of a destination IP address, destination port number, source IP address, and a source port number (i.e., 172.0.0.2:80:180.1.1.4:2413), and are the key distinguishing characteristic for a particular flow of traffic features. We then read each line item of data from *flows.out* and use the associated tuples to compare to the tuples in each line of the *ttad.out* data. When performing this comparison, if the two match - meaning the tuples are the same, the flow features are appended with the appropriate label. All flow features that are labeled “GOOD” and “BAD” are saved to an output file *labeled.csv*, which will be utilized in out testing phase as our training data. Those flows that are labeled as “UNKNOWN” are saved to a separate output file *unknown.csv* and will be our testing data during the testing phase.

CHAPTER 4: Results

In this chapter, we cover our testing method and results from processing all of the data sets collected and analyzed from the four real-world packet captures. We first discuss the classifier and methods used to classify our data. We then discuss the testing process and describe the results of each individual test we conducted, to include a unique "Hybrid" test that trains and tests on different captures. These initial analyses reveal how well MWID performs relative to the RBLs. We continue by discussing the manual sampling technique that was performed on flows *unknown* to the RBLs, which was used to obtain ground-truth labels and evaluate our larger hypothesis that MWID is complimentary to existing techniques. Finally, we conclude with a summary of concerning conflicts and present a method of resolution.

4.1 Testing

Using `mwid_class`, which invokes the use of a third-party software suite called Orange [15], we perform statistical analysis on our data while training the classifier. The *labeled.csv* file is used as our training data and once the file is processed and the learning is complete, we then input the *unknown.csv* file as our testing data for classification. During the statistical analysis process, we use two well-known supervised learning algorithms (classifiers) to process our data, Naïve Bayes and Decision Tree, which are both invoked using Orange.

4.1.1 Learner/Classifier

During the classification process, we perform a series of analyses to determine the accuracy of MWID on our real-world capture data. There are two modes or process levels that the learner/classifier will process in. First is the training mode, where the learner is "learning or training" on the input data. Second is the testing mode, where the classifier actually "classifies or tests" the input data, based off of the model built from training data. The following training and testing processes occur:

1. Train and test on the (same data set).
    - Basic analysis that simply evaluates the performance of the learned model on itself as a sanity check.
2. 10-fold (10x) Cross Validation (CV) (same data set).

   - Evaluates the performance of MWID relative to the RBLs and Alexa ground-truth.
   - Breaks the data into 10 sets of size n/10.
   - Trains on 9 data-sets and test on 1.
   - Repeats the process 10 times.
   - Takes the mean accuracy of all the processes.
3. Train and test on separate data.
   - Use MWID to classify flows *unknown* to the RBLs or Alexa. Requires manual sampling to obtain ground-truth and evaluate performance.
   - "GOOD" and "BAD" labeled data used as training data.
   - "UNKNOWN" labeled data is used as testing data.

Performance metrics that are reported include the CA, precision or Positive Predictive Value (PPV), recall or Sensitivity (SENS), and the F1 Score, which is the measure of our test's overall accuracy, considering both the PPV and SENS.

The metric values are derived from the following formulas:

- SENS (recall) = $TP/(TP + FN)$
- PPV (precision) = $TP/(TP + FP)$
- Accuracy (ACC) (accuracy) = $(TP + TN)/(TP + TN + FP + FN)$
- F1 score = $2 * (PPV * SENS)/(PPV + SENS)$

We define the following:

- True Positive (TP) – a "GOOD" flow correctly labeled as a "GOOD"
- True Negative (TN) – a "BAD" flow correctly labeled as "BAD"
- False Positive (FP) – a "BAD" flow incorrectly labeled as a "GOOD"
- False Negative (FN) – a "GOOD" flow incorrectly labeled as a "BAD"

4.2 Testing Sets

In each of the different testings we conduct, we wanted to test several key hypotheses. First, show that traffic features and characteristics associated with flows of traffic can in fact be used to classify data accurately. Second and most importantly, positively classify previously unknown data flows as malicious, given no prior knowledge of the flow, and where the host or IP address has not been previously identified by other means or found on an RBL. In this process, we performed four separate tests on different packet captures and include 3 additional unique tests to further meet our objectives. Table 4.1 summarizes the data we are use for training and testing of our classifier, along with its complexion.

Table 4.1: Composition of *labeled.csv* and *unknown.csv* Files

Data Set	**File Name**	**Total Flows**	**Good**	**Bad**
Capture 1	labeled1.csv	61843	55877	5966
	unknown1.csv	287730	–	–
Capture 2	labeled2.csv	28984	28745	239
	unknown2.csv	357544	–	–
Capture 3	labeled3.csv	15356	13632	1724
	unknown3.csv	176601	–	–
Capture 4	labeled4.csv	7815	4003	3812
	unknown4.csv	41303	–	–

In a spin-off from how our original tests were conducted, and based off of the overall results from all tests, we find it interesting to experiment with mixed data among the test sets.

Specifically, we take the *reclassed.csv* file from one test and use it as our training set for the *unknown.csv* data from one of the other tests. We cover such hybrid testing in §4.3.

4.2.1 Test 1

In test one, working with first set of captures, we perform the basic routine of training our classifier on the labeled data contained in the *labeled1.csv* file, which contains 61,843 distinct flows of traffic. There are 55,877 flows that are labeled as "GOOD" (90%), as they appeared on the Alexa list and not found on the RBLs and 5,966 that are labeled "BAD" (10%) where they were in fact found on a RBL. The testing methodology that MWID performs for test 1 is provided in figure 4.1.

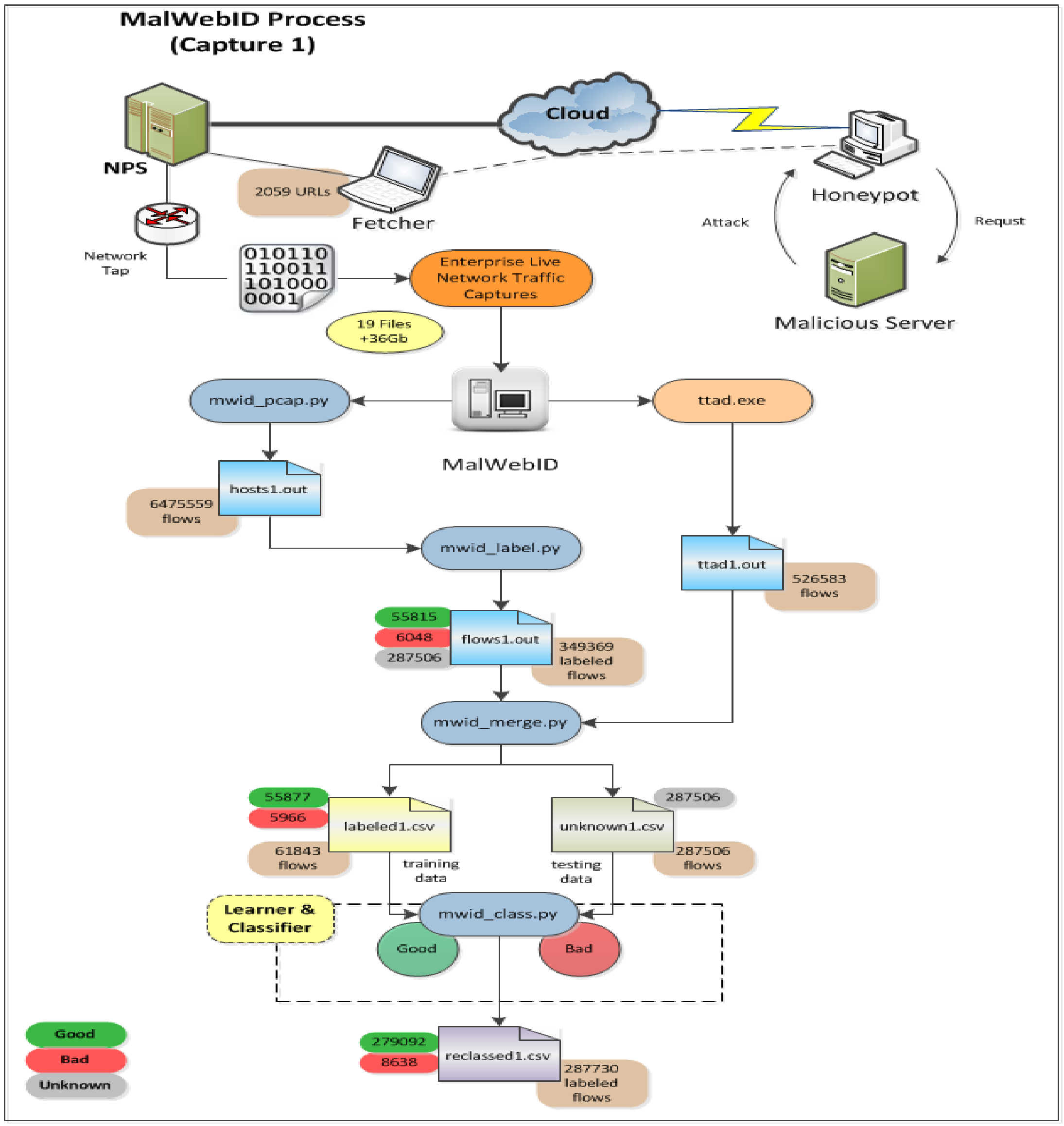


Figure 4.1: MalWebID Programmatic Flow for Capture 1.

Table 4.2: Test One - Prediction Accuracy

Learner	CA	PPV	SENS	F1
Bayes	94.3%	99.1%	94.6%	96.8%
Tree	99.3%	99.7%	99.6%	99.6%

Table 4.3: Test One - Confusion Matrix

True Positive 55645	**False Negative** 232
False Positive 188	**True Negative** 5778

Using our classifiers, we perform a series of analysis to determine the accuracy of our classifiers. In the basic analysis, training and testing on the same data set, Bayes recorded an accuracy of 93.8% while the Decision Tree logically performed better, attaining a 100% CA.

In the follow-on test, involving 10x CV, Bayes shows a slight improvement with a 94.3% CA while Decision Tree drops slightly to a 99.3% CA. See table 4.2 for test 1 specifics on CA, PPV, SENS, and the F1 Score.

We provide a confusion matrix, table 4.3, to demonstrate how well MWID performed using labeled data to train and test, prior to actually testing on the unknown data for classification. While we expect that this test to do well overall, we perform it to ensure that the learner/classifier model works well on itself.

The accuracies of both classifiers show promise and are both initially used in our follow-on classification of the 'UNKNOWN' data. Here, we are working with 287,730 "UNKNOWN" flows of traffic. We begin by processing the data using our classifier program `mwid_class`, where we first train the classifier using the labeled flows in the *labeled1.csv* file. Once complete, we then use the classifier to test on the flows of "UNKNOWN" traffic in the *unknown1.csv* file. At this point, all of the unknown flows are reclassified and labeled, with the results output to the *reclassed1.csv* file.

During this process, we discover that the naïve Bayes algorithm does not perform as well as expected when attempting to classify the "UNKNOWN" data set. We made the decision to remove the algorithm from our testing and to proceed with using only the Decision Tree algorithm.

Once the classification process is finished, all of the previous 287,730 unknown flows of data are labeled and output to *reclassed1.csv*. There are 8,638 flows that are labeled as "BAD" (3.0%)

Table 4.4: Test Two - Prediction Accuracy

Learner	CA	PPV	SENS	F1
Tree	98.6%	99.4%	99.3%	99.3%

Table 4.5: Test Two - Confusion Matrix

True Positive 28534	**False Negative** 211
False Positive 184	**True Negative** 55

and 279,092 labeled as "GOOD" (97.0%). In section 4.4.1 we perform the statistical analysis on our reclassified data to determine the accuracy, how well we did, at classifying the unknown data.

4.2.2 Test 2

In our second test, working with the second set of packet captures, we again perform the same testing scheme as performed in test 1, training our classifier on the new labeled data in the *labeled2.csv* file, which contains 28,984 distinct flows of traffic. There are 28,745 that are labeled as "GOOD" (99.2%) and 239 that are labeled "BAD" (8.0%). The testing methodology that MWID performs for test 2 is provided in figure 4.2.

The number of "BAD" flows found here is actually surprising, given that no Fetcher URLs were injected for this test set. As previously mentioned, given the environment of a military institution where we perform our packet captures, it is less likely that abusive sites are visited on a regular basis. Using our classifier, we again perform an accuracy analysis by training and testing on the same data set, to look at how well the classifier performs. Absent Bayes, the Decision Tree again performed well, attaining a 100% CA in all anaylsys and 98.6% CA with the 10x CV test. The overall results can be seen in table 4.4.

In table 4.5 we provide a confusion matrix from our second test, which is the the result of training and testing on the same set of data. This step was performed, as with all other tests, to understand how well our classifier is performing before testing on the actual "UNKNOWN" data.

Now, only using the Decision Tree as our classifier for the follow-on classification of the "UNKNOWN" data, we're working with 357,544 "UNKNOWN" flows of traffic. We first train our classifier using the labeled flows in the *labeled2.csv* file. Once complete, we then process

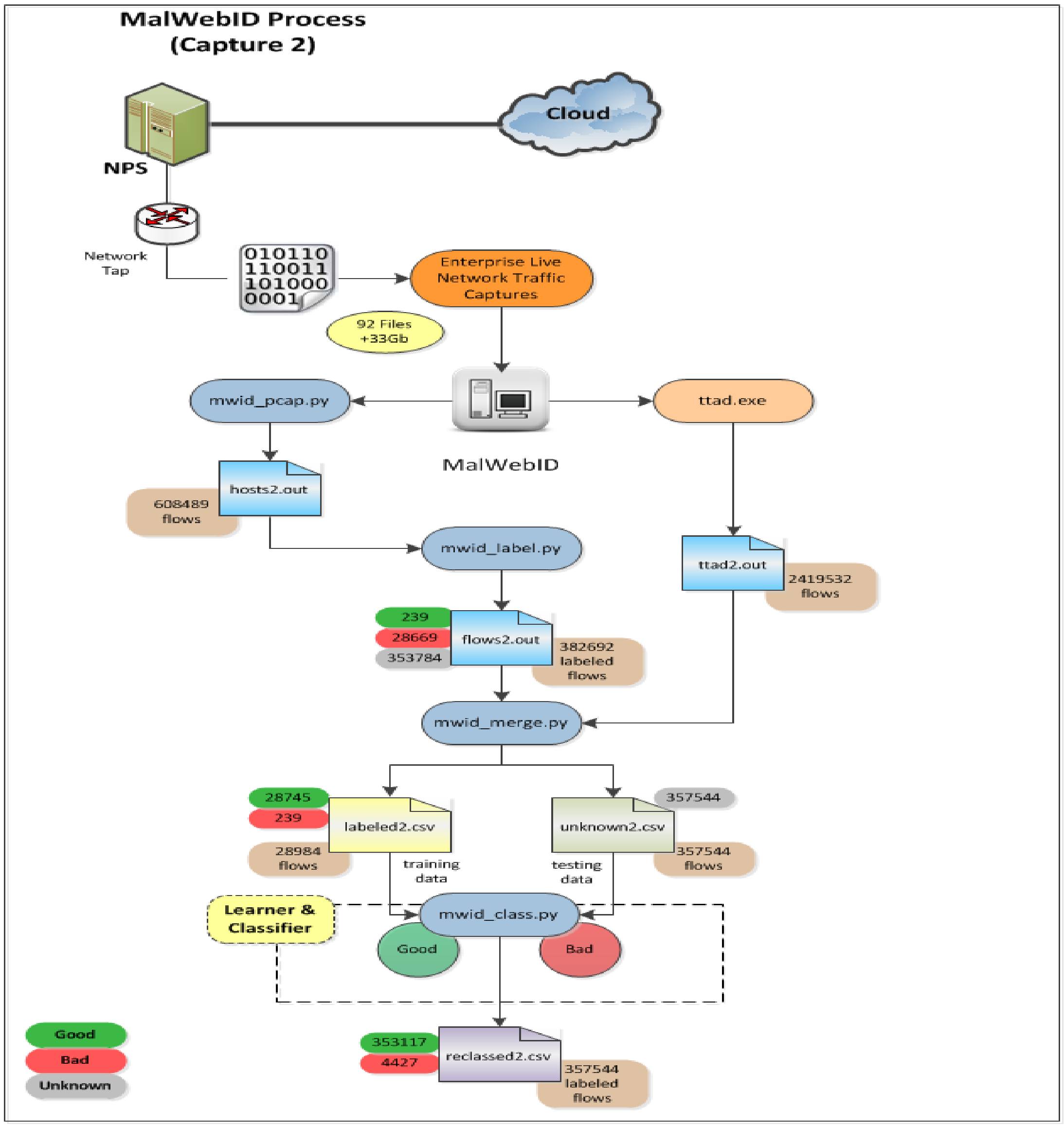


Figure 4.2: MalWebID Programmatic Flow for Capture 2.

our *unknown2.csv* as the test data. The results are output to the *reclassed2.csv* file. Of the 357,544 reclassified flows of data, there are 4,427 flows that are labeled as "BAD" (1.0%) and 353,117 labeled as "GOOD" (99.0%) In section 4.4.2 we perform the sampling analysis on our reclassified data to determine accuracy.

4.2.3 Test 3

In our third test case, we are working with the third set of packet captures that contain Fetcher injected flows of traffic. These fetcher flows were obtained from our honeypot just prior to the

Table 4.6: Test Three - Prediction Accuracy

Learner	CA	PPV	SENS	F1
Tree	97.0%	98.4%	98.2%	98.3%

Table 4.7: Test Three - Confusion Matrix

True Positive 13383	**False Negative** 249
False Positive 217	**True Negative** 1507

test, as to help ensure that they were fresh with respect to the host URLs and associated IPs being valid. As with most of the URLs taken from the honeypot, they will go stale or become invalid after a few days. This means that the site is no longer available, which we discovered can mean the site has been taken offline, suspended (i.e., violation of terms, non-payment, etc.), or no longer resolves to a usable site (i.e., blank screen). Additionally, considering the sites do not remain valid for a long period of time, in most cases, our assumption is the faster we can perform the tests using fresh data, we increase the odds of the malicious hosts not being documented on a black-list. The testing methodology that MWID performs for test 3 is provided in figure 4.3.

Moving on to our test, we again perform the same testing scheme as before, training our classifier on the new labeled data in the *labeled3.csv* file, which contains 15,356 distinct flows of traffic, of which there were 13,632 that were labeled as “GOOD” (89.0%) and 1,724 that were labeled “BAD” (11.0%)

Using our classifier, we again perform an accuracy analysis by training and testing on the same data set. The Decision Tree continued to perform well in most of the test with a 100% CA. It dropped slightly in this round of testing when performing a 10x CV, with and overall CA of 97%. The overall results can be seen in table 4.6.

In table 4.7 we provide a confusion matrix from our third series of tests, prior to training on the unknown data.

Of the 176,601 flows that are reclassified, there are 4,393 flows that are labeled as “BAD” (2.0%) and 172,208 labeled as “GOOD” (98.0%). In section 4.4.4 we perform the statistical analysis on our reclassified data to determine the classifiers accuracy.

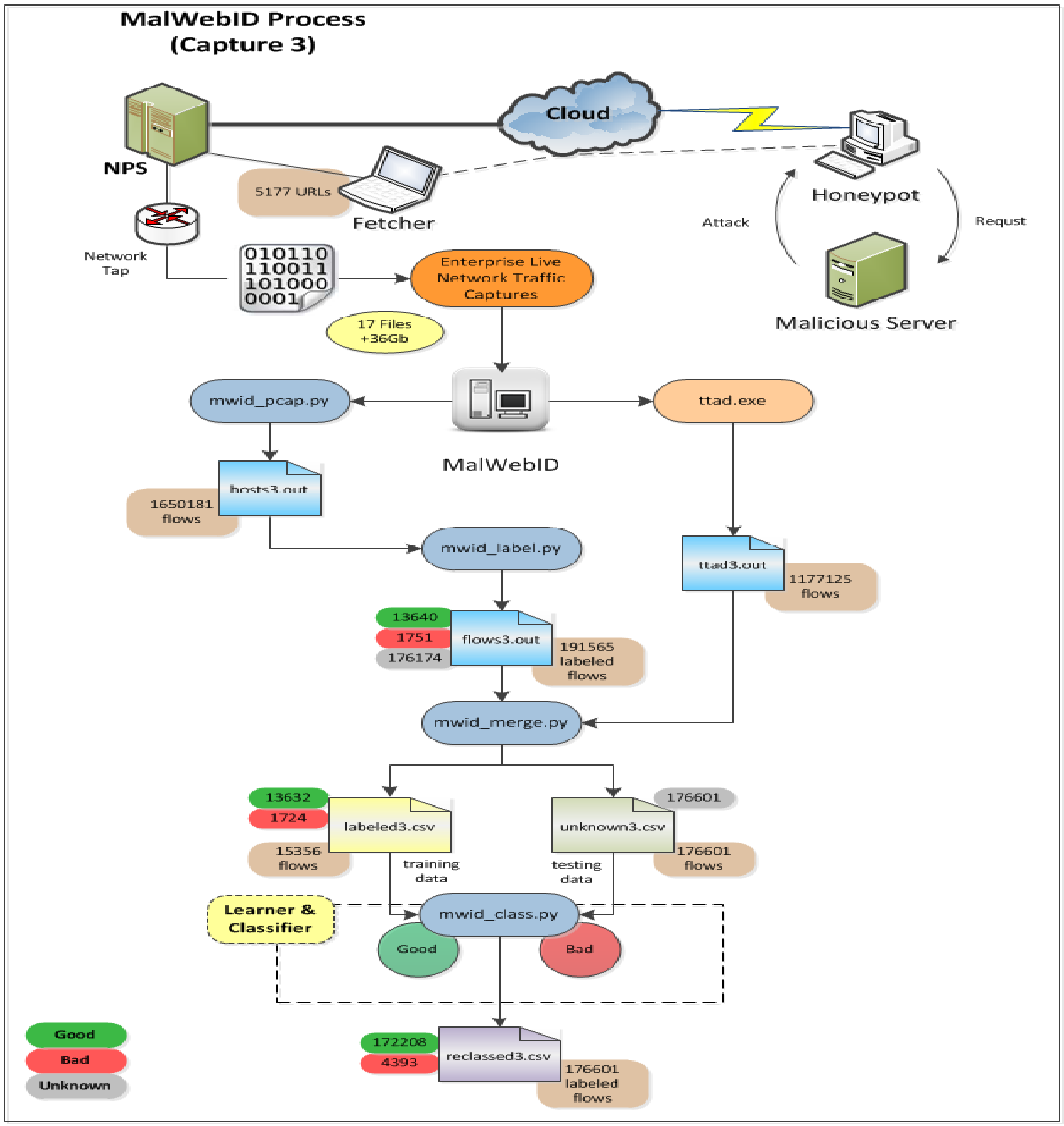


Figure 4.3: MalWebID Programmatic Flow for Capture 3.

4.2.4 Test 4

In our fourth and final test case, we are working with the fourth set of packet captures that contain new Fetcher injected flows of traffic. Like in test 3, these new flows were obtained from our honeypot just prior to the test to help ensure freshness, as described above. The testing methodology that MWID performs for test 4 is provided in figure 4.4.

Performing the same testing scheme as before, we train our classifier on the new labeled data in the *labeled4.csv* file, which contains 7,815 distinct flows of traffic, of which there were 4,003

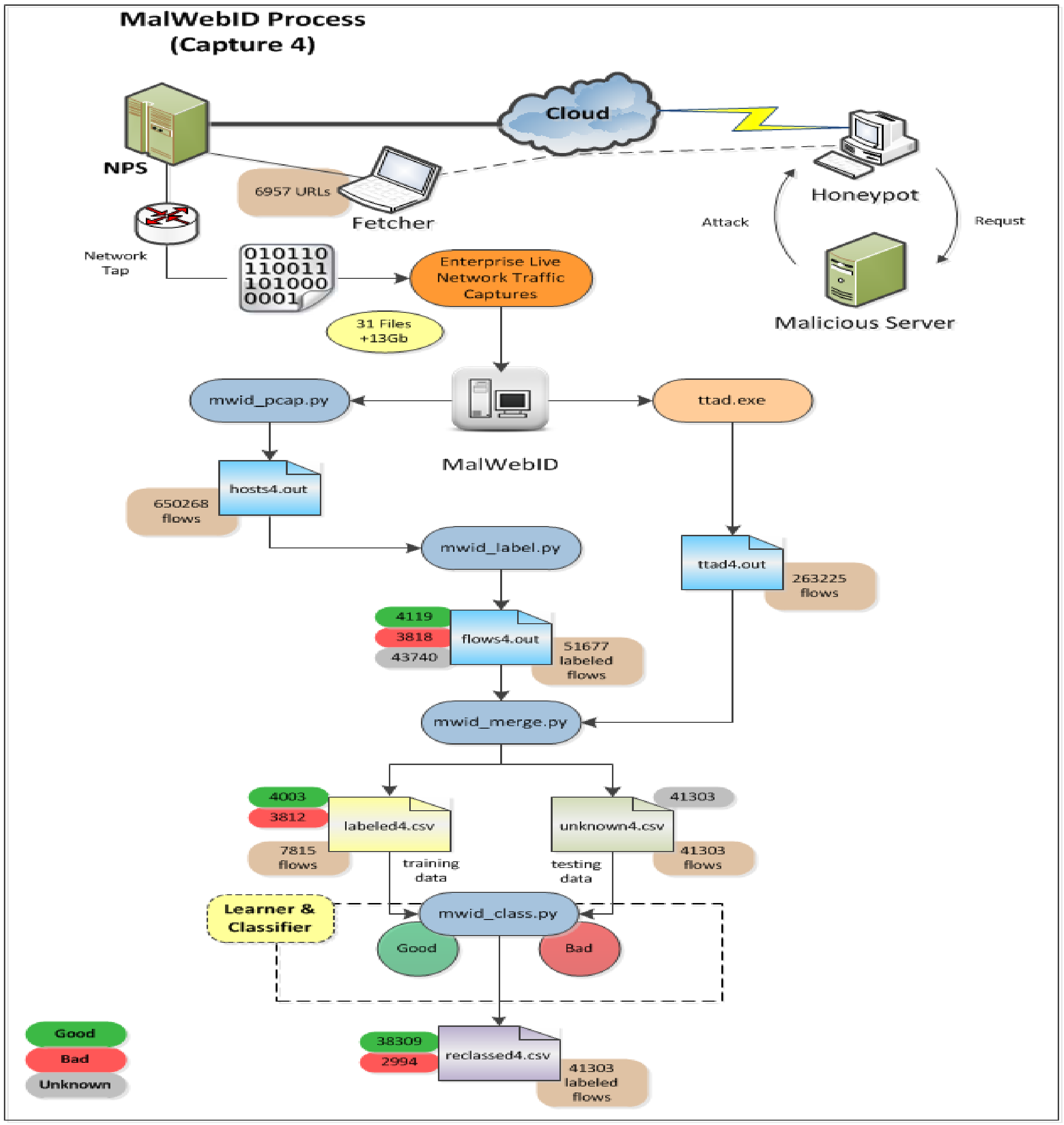


Figure 4.4: MalWebID Programmatic Flow for Capture 4.

that were labeled as “GOOD” (51.0%) and 3,812 that were labeled “BAD” (49.0%).

Using our classifier, we again perform an accuracy analysis by training and testing on the same data set. We found the same exact CA with our classifier as seen in test 3, but a very slight change in the overall CA, which was 96.9%. These results can be seen in table 4.8.

Again, we provide a confusion matrix in table 4.9 from our third series of tests, prior to actual testing on the unknown data.

Table 4.8: Test Four - Prediction Accuracy

Learner	**CA**	**PPV**	**SENS**	**F1**
Tree	96.9%	97.0%	96.9%	96.9%

Table 4.9: Test Four - Confusion Matrix

True Positive 3877	**False Negative** 126
False Positive 120	**True Negative** 3692

Of the 41,303 flows that are reclassified, there are 2,994 flows that are labeled as “BAD” (7.0%) and 38,309 labeled as “GOOD” (93.0%). In section 4.4.5 we perform the statistical analysis on our reclassified data to determine the classifiers accuracy.

4.3 Hybrid Testing

Having completed the initial round of testing, we were curious as to what would happen if we were to combine different aspects of our tests together and if that would effect change with respect to the outcome of classifying our data. To this end, we specifically looked at taking the *reclassed#.csv* data from one test series and using it as our training data for our classifier, while using an *unknown#.csv* from a separate test to perform our classification test on. Figure 4.5 shows the hybrid testings scheme and data process flow.

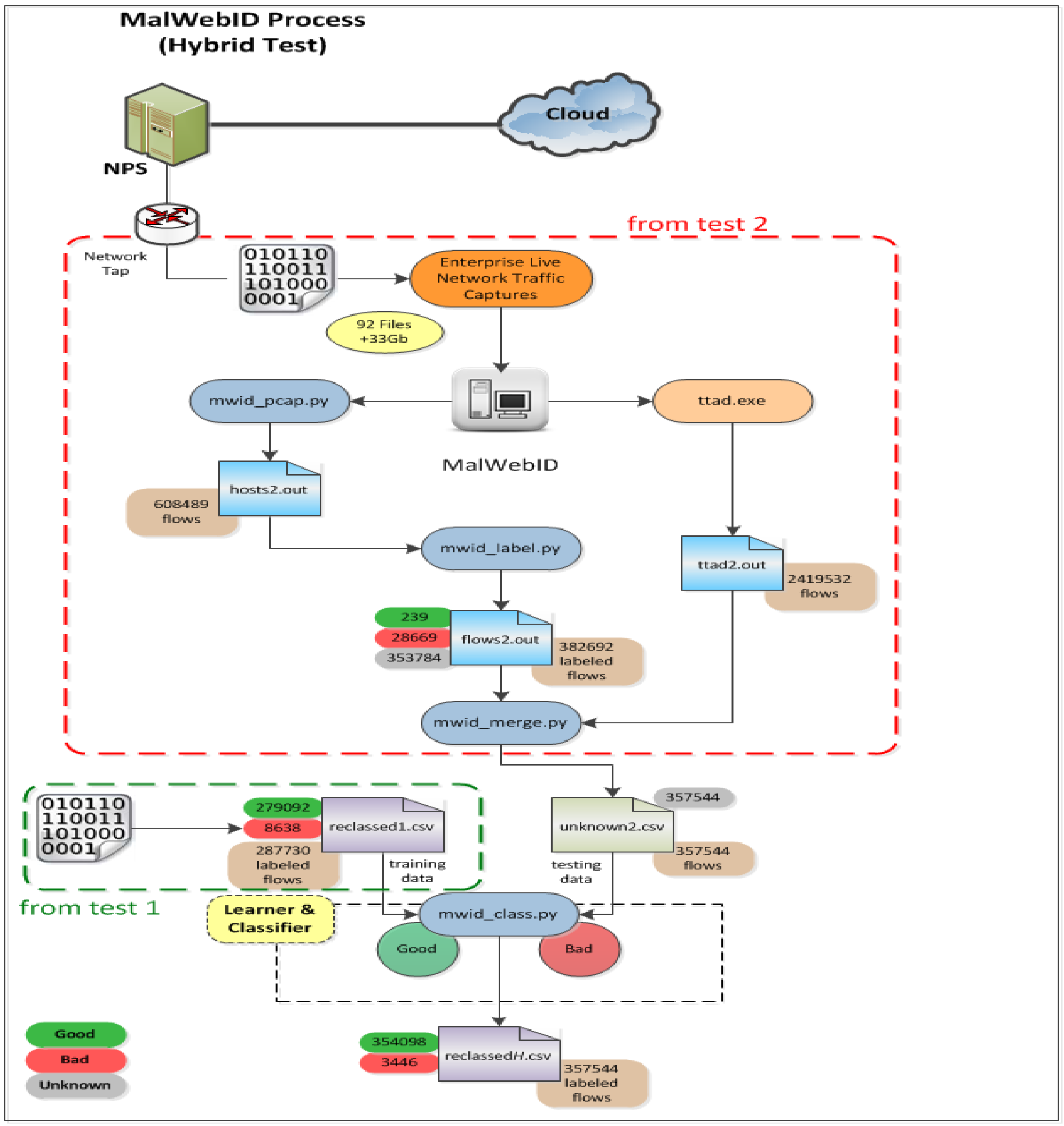


Figure 4.5: MalWebID Programmatic Flow for Hybrid Test.

Table 4.10: Hybrid Test (1 and 2) - Prediction Accuracy

Learner	CA	PPV	SENS	F1
Tree	97.3%	98.7%	98.5%	98.6%

Table 4.11: Hybrid Test - Confusion Matrix

True Positive 278452	**False Negative** 640
False Positive 559	**True Negative** 8079

We decided on utilizing our first two series of tests for this experiment, primarily because of the length of time between the packet captures and because of one test having Fetcher injected traffic while the other did not. The *reclassed1.csv* data set from test 1, containing 287,730 flows, is used as our training set. Of these flows, 279,092 were labeled as "GOOD" (97.0%) and 8,638 were labeled as "BAD" (3.0%). From test 2, we use the *unknown2.csv* data for our test set, which contained 357,544 "UNKNOWN" flows.

Once processed, we noted that aside from the normal testing criteria that yielded a 100% CA from our classifier, the level of accuracy shown after a 10x CV test was impressive, rendering a 99.6% overall CA. Table 4.10 refers.

The confusion matrix is provided in table 4.11, which too shows how well our classifier performed, using *reclassed1.csv* data to train, prior to testing on the *unknown2.csv* data.

4.4 Sampling Analysis

Given our results from each test, we now focus on the reclassified data in each of our *reclassed.csv* files, where the previous "UNKNOWN" traffic is now labeled either "GOOD" or "BAD." A key focal point in our analysis is to determine how well our classifier performed in predicting the correct label for a flow of traffic, based off of the statistical analysis performed on all of the individual traffic flow features.

To assess performance, we perform manual inspection on a random set of hosts from both the data in each *reclassed.csv* files from each test set and of the host lists that was obtained from our honeypot. This manual inspection gives us an accurate accountability as to the performance of our classifier. We provide a confusion matrix for each manual inspection result and indicated the overall ACC, PPV or precision, SENS or recall and provide the F1 score (harmonic mean).

When we conduct a manual sampling of the data, we physically visit each website to make the final determination as to the actual type of infrastructure present. Determination was centered around the following criteria when visiting websites:

- Malicious (BAD)
  - resolve to a malicious site (i.e., Porn, Rx/Pharmaceutical, illegal activity, etc.)
  - propagate or contain viruses, spyware, or other harmful programs, participate in spamming or any type of illegal activity
  - reported as an attack site by search engine (i.e., Google)
  - multiple redirections to alternate sites (i.e., spam sites)
  - suspended sites (by domain administrator), generally spam related
- Non-Malicious (Good)
  - resolve to a functional, non-threatening, well-behaved website that does not exhibit the behaviors or indications as mentioned above

4.4.1 Test 1 - Manual Inspection

Prior to performing the manual inspection on the test 1 data, we performed a manual inspection of the "pre-processed" URLs that were injected into the test 1 captures.

Fetcher Analysis - Test 1

We utilized Fetcher to inject 2,983 hosts, taken from our honeypot, into our packet capture routine. Of these hosts, we pre-processed them and know in advance that 1,804 (60.0%) of the hosts were duplicates and thus removed. Of the 1,179 remaining, 278 (23%) exist on a black-list and marked as malicious, 57 (5.0%) appear on Alexa's list and marked as non-malicious, and we are left with 844 (72.0%) unknown, which we assume are malicious for the purpose of our analysis. Looking at the labeled data from test 1, we compare the labels with the list of malicious URLs from the honeypot. We identified 1,417 flows in the list that matched.

Of the 1,417 flows, 872 are labeled as "GOOD" (62.0%) and 545 are labeled "BAD" (38.0%). Performing a manual inspection of each host labeled as "BAD," we found that 510 were in fact malicious and 35 were non-malicious sites. Considering the size of those flows labeled at "GOOD," we take 100 randomly selected flows and perform a manual inspection of each. 94 were found to be non-malicious while 6 were found to be malicious. The confusion matrix in table 4.12 is provided for reference, with performance metrics given in table 4.13.

Table 4.12: Fetcher - Manual Sample Confusion Matrix

True Positive 94	**False Negative** 35
False Positive 6	**True Negative** 510

Table 4.13: Fetcher - Manual Sample Results

ACC	PPV	SENS	F1
93.6%	94.0%	72.8%	82.1%

Table 4.14: Test 1 - Manual Sample Confusion Matrix

True Positive 90	**False Negative** 9
False Positive 10	**True Negative** 91

Table 4.15: Test 1 - Manual Sample Results

ACC	PPV	SENS	F1
90.5%	90.0%	91.0%	90.5%

Test 1 Analysis

Looking at the *reclassed1.csv* file, of the 287,730 "UNKNOWN" flows that were processed and labeled, 8,638 were labeled as "BAD" (3.0%)with the remaining 279,092 labeled as "GOOD" (97.0%). Focusing on those flows that were labeled as "BAD," we take a random sample of 200 for manual inspection: 100 "GOOD" and 100 "BAD" flows. Of 100 "GOOD" flows, there were 10 found to be improperly labeled, as they resolved to malicious sites. Of the 100 "BAD" flows, there were 9 found to be non-malicious and improperly marked. Of the 91 flows that were truly malicious, all 91 were found on the host file we used originating from our honeypot.

The confusion matrix in table 4.14 is provided for reference. From the manual inspection of the 200 flows, the results are seen in table 4.15.

Here we are already showing positive results toward proving that our method of discriminating live traffic data is capable of detecting previously unknown malicious hosts, solely based off of the features gathered from each individual traffic flow.

Table 4.16: Test 2 - Manual Sample Confusion Matrix

True Positive 50	**False Negative** 49
False Positive 0	**True Negative** 1

Table 4.17: Test 2 - Manual Sample Results

ACC	PPV	SENS	F1
51.0%	50.5%	100%	67.11%

4.4.2 Test 2 - Manual Inspection

In test 2, we have 357,544 "UNKNOWNS" that were reclassified, 4,427 "BAD" (1.0%) and 353,117 "GOOD" (99.0%). Because there are no Fetcher injects for this test, we take and manually sample 100 random hosts from the *reclassed2.csv* file, 50 "GOOD" and 50 "BAD." Of all 100 sampled hosts, there were 1 malicious sites discovered and was properly labeled as "BAD." All 50 "GOOD" were in fact non-malicious, and 49 of those hosts labeled as "BAD" were in fact non-malicious.

Overall, the results are not very surprising, considering it is indicative of the environment at NPS and that of a military installation, where it is less likely to encounter an enormous number of abusive web sites. However, this outcome is very good, considering an environment where the overwhelming majority of network traffic is non-malicious, with no Fetcher injected URLs, we were able to positively identify and properly classify a malicious site that has not been reported on any of the RBLs that we are using! Table 4.16 and 4.17 provides our results.

4.4.3 Hybrid - (Test 1 & Test 2) Manual Inspection

Looking at the data collected during the hybrid test, we were anxious to see what would be the effect on the actual classification of our data. Figure 4.5 shows the hybrid testing method we used.

The outcome has 3,446 flows classified as "BAD" and 354,098 as "GOOD." We take and manually sample 200 random hosts from the new *reclassed2H.csv* file, 100 "GOOD" and 100 "BAD." Of all 200 sampled hosts, there were no malicious sites discovered, which again given that we are using the test 2 data with no Fetcher injects, is understandable. The interesting aspect here is the change in the number of hosts that were labeled as "BAD" in this test, as opposed to the original test 2 *reclassed2.csv* output of 4,427 "BAD," which decreased by 981.

Table 4.18: Hybrid Test - Manual Sample Confusion Matrix

True Positive 100	**False Negative** 100
False Positive 0	**True Negative** 0

Table 4.19: Hybrid Test - Manual Sample Results

ACC	**PPV**	**SENS**	**F1**
100%	100%	50.0%	66.7%

Table 4.20: Test 3 - Manual Sample Confusion Matrix

True Positive 636	**False Negative** 124
False Positive 2	**True Negative** 8

This was an interesting discovery, as it clearly demonstrates that by training the classifier on data flows that truly are malicious, given the distinct feature vectors of those flows, allows the classifier to predict with greater accuracy when labeling the unknown data formats. See Table 4.18 and table 4.19 for the results.

4.4.4 Test 3 - Manual Inspection

In test 3, we again utilized Fetcher to inject 5,177 malicious hosts from our honeypot into our packet capture routine. After pre-processing, 1,411 (27.0%) of the hosts were duplicates and thus removed. Of the 3,766 remaining, 3,195 (85%) exist on a black-list and marked as malicious, 36 (1.0%) appear on Alexa's list and marked as non-malicious, and we are left with 535 (14%) unknown, which default to malicious.

Analyzing the newly labeled data from test 3, we performed a comparison of the labeled data with that of the malicious host list that we obtained from our honeypot. We identified 770 flows that matched and are marked for manual inspection. Of the 770 flows, 638 are labeled as "GOOD" (83.0%) and 132 are labeled "BAD" (17%).

After manually sampling all 770 flows, we find that 8 flows that are marked as "BAD" are in fact truly malicious. 124 actual non-malicious site are improperly labeled as "BAD," 636 sites are properly labeled as "GOOD" and finally there are 2 "BAD" that were incorrectly labeled as "GOOD." Table 4.20 refers.

Table 4.21: Test 3 - Manual Sample Results

ACC	PPV	SENS	F1
83.6%	99.7%	83.7%	91.0%

Table 4.22: Test 4 - Manual Sample Confusion Matrix

True Positive 1042	**False Negative** 486
False Positive 3	**True Negative** 108

Table 4.23: Test 4 - Manual Sample Results

ACC	PPV	SENS	F1
69.3%	99.7%	68.2%	81.0%

We've now discovered and correctly labeled 8 malicious sites that were previously unknown to any black-list. From the manual inspection of the 770 flows, the results are seen in table 4.21

4.4.5 Test 4 - Manual Inspection

In final test 4, we utilize Fetcher to inject 6,957 fresh malicious hosts from our honeypot into our packet capture routine. After pre-processing, 3,986 (57.0%) of the hosts were duplicates and thus removed. Of the 2,971 remaining, 1,870 (62.5%) exist on a black-list and marked as malicious, 16 (.5%) appear on Alexa's list and marked as non-malicious, and we are left with 1,085 (37.0%) unknown, which default to malicious.

Analyzing the newly labeled data from test 4, we perform a comparison of the labeled data with that of the malicious host list that we obtained from our honeypot. We identified 1,660 flows that matched and are marked for manual inspection. Of the 1,660 flows, 1,066 are labeled as "GOOD" (64.0%) and 594 are labeled "BAD" (36.0%).

After manually sampling all 1,660 flows, we find that 108 flows that are marked as "BAD" are in fact truly malicious. 486 good flows are improperly labeled as "BAD," 1,063 flows are properly labeled as "GOOD" and finally there are 3 "BAD" flows that were incorrectly labeled as "GOOD." Table 4.22 refers. From the manual inspection of the 1,660 flows, the results are seen in table 4.23

This was an extremely encouraging result, considering we were able to positively identify 108 previously unknown malicious hosts, none of which were currently found on any of the RBLs that we were searching.

4.4.6 Conflicts

There are two type of conflicts that must be addressed that were discovered during our research.

List Conflicts

As shown in Chapter 3, we discovered the presents of conflicts as a result of hosts or IP addresses appearing on two different lists that we are using to determine appropriate labels for our data. Specifically, the host or IP resides on both a black-list and on the Alexa list.

Of the 20 conflicts discovered over the course of our experiments, all were found to be non-malicious in nature.

- 3 conflicts discovered in test 1
- 8 conflicts discovered in test 2
- 4 conflicts discovered in test 3
- 5 conflicts discovered in test 4

This is a great example of how black-lists in general can be inaccurate at times, considering the manner by which they are updated and managed. In these particular instances, the Alexa list of our presumed "GOOD" sites was indeed accurate. This is also a perfect example of where a tool such as MWID would be very useful.

Reclassified Data Conflicts

Another understood conflict that we discovered was with the results of the reclassification of our "UNKNOWN" data sets. The conflict occurs when we have multiple traffic flows that resolve to the same host name, but are differentiated by another factor, such as the source port number or destination IP address. For example:

```
0.gravatar.com  72.21.91.121:80:172.20.105.136:55432  BAD
0.gravatar.com  72.21.91.121:80:172.20.106.125:1986  GOOD
latesummergifts.ru  180.70.9.31:80:172.20.109.183:45972  BAD
latesummergifts.ru  180.70.9.31:80:172.20.109.183:32857  GOOD
```

In this example, you can see that there are two hosts that are labeled both "GOOD" and "BAD." The differences can be associated with the source IP addresses of *0.gravatar.com* and the source

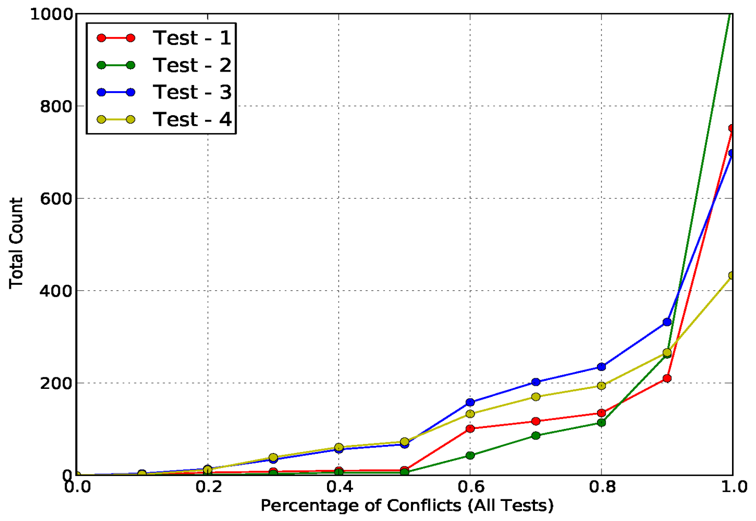


Figure 4.6: Flow Labeling Conflicts

port numbers for *latesummergifts.ru*. The reason that two different flows with the same host name and IP addresses can result in a labeling conflict would primarily be because of the features that were associated with each of those flows. One set of features will be the statistical threshold for being classified as a "GOOD" while the next will not. The differences in the features can be attributed to many different variables (i.e., network congestion, difference in time of day, difference in location, etc.).

Considering these types of conflicts in labeling, we build a Cumulative Distribution Function (CDF), provided in figure 4.6, to help with determining where we would like to implement a threshold for what is considered good or bad flows. A summary of the conflicts are detailed in table 4.24. More specifically, the percentages represent a threshold and the numbers under each of the test areas represent the number of "GOOD" that fall below that particular percentage mark.

For example, looking at table 4.24 and test 3, if we wanted to know the number of flows where "GOOD" falls below the 50% mark, it would be 67 flows. Further examples can be seen below, looking at a small subset of flows from test 3, the number of flows labeled as "GOOD" are below the 50% threshold and the "BAD" is greater than 50%. In this case, if we were placing our threshold on all the data at 50%, then all of these flows would receive a final label of "BAD."

Table 4.24: Conflict Percentages

Percentage Level	Test 1 Conflicts	Test 2 Conflicts	Test 3 Conflicts	Test 4 Conflicts
100 %	752	1034	698	433
90 %	210	262	332	266
80 %	135	114	235	194
70 %	117	86	202	170
60 %	101	43	158	133
50 %	11	6	67	73
40 %	10	6	56	61
30 %	8	3	34	39
20 %	6	1	14	12
10 %	1	0	4	2

```
Bad =   14 (0.609), Good =    9 (0.391)  -  3cm.kz
Bad =    2 (0.667), Good =    1 (0.333)  -  action.metaffiliation.com
Bad =    5 (0.625), Good =    3 (0.375)  -  argus.nul.usb.ve
Bad =    7 (0.700), Good =    3 (0.300)  -  baec.eu
Bad =    8 (0.571), Good =    6 (0.429)  -  bestzm.ru
```

Figures 4.7, 4.8, 4.9 and 4.10 are provided and display the conflict numbers above 50% for each test.

4.5 Classification Examples

In this section we provide examples of our classification results, by category of TP, TN, FP, FN and provide an additional sample of FN URLs that were of interest, considering they are not only known non-malicious, but are .mil sites. Due to space limitation, we limit these examples to only a small subset of the overall data examined.

- TP examples – "GOOD" correctly classified as "GOOD."
  - facebook.com
  - google.com
  - echoenabled.com
  - pandora.com
  - weather.gov
  - dailymail.co.uk

  - ytimg.com
  - newegg.com
  - pressdisplay.com
  - mediaplex.com
- TN examples – “BAD” correctly classified as “BAD.”
  - acf.mediccella.ru
  - acg.nimdoctor.ru
  - acx.doctorghos.ru
  - awoxiyn.giigagbu.getbigg.ru
  - azpj.doctorpes.ru
  - azs.pinkdoctor.ru
  - azskxg.doctornode.ru
  - aztecpk.medicfles.ru
  - azzuukh.swanmedic.ru
  - mydrugstorerx.com
- FP examples – “BAD” incorrectly classified as “GOOD.”
  - focydu.igynu.buyheragift.ru
  - fohe.donuisai.summergifts.ru
  - cyoxeuq.dotdte.summer2012watches.ru
  - exatiav.fbewytywi.rolexsummersale.ru
  - i.viwmwyr.tk
  - itemqualitytop.ru
  - mbynd.co
  - mcal.ru
  - mqtwzd.pochta.com
  - opo.co
- FN examples – “GOOD” incorrectly classified as “BAD.”
  - a1128.g.akamai.net
  - edt02.net
  - espncdn.com
  - fluidra.com
  - idmel.e-bamboo.fr
  - lh3.googleusercontent.com
  - mapbox.com

  - pinterest.com
  - twimg.com
  - wunderground.com
- Of Interest, the following (.mil) URLs were classified as “BAD,” but are in fact non-malicious. This is an example where a host that often experiences considerable periods of traffic congestion will negatively influence our testing method.
  - crl.disa.mil
  - iase.disa.mil
  - doni.daps.dla.mil
  - mail.dodiis.mil
  - dma.mil
  - defensetravel.dod.mil
  - public.navy.mil
  - sldcada.disa.mil

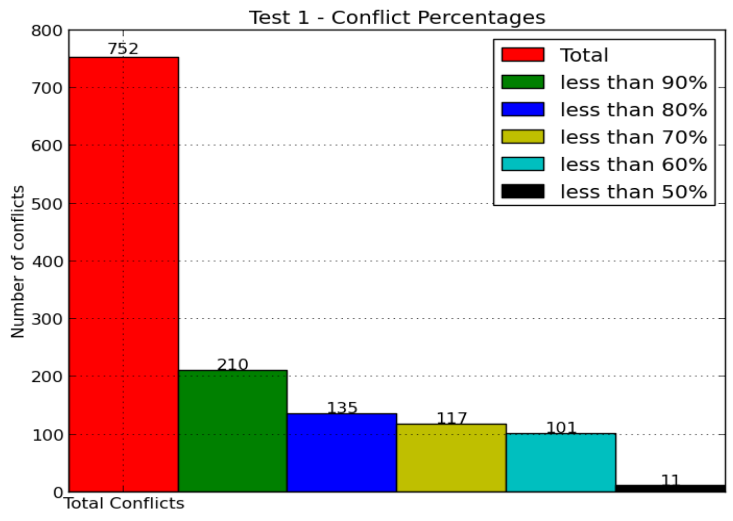


Figure 4.7: Test 1 - Conflict Percentages

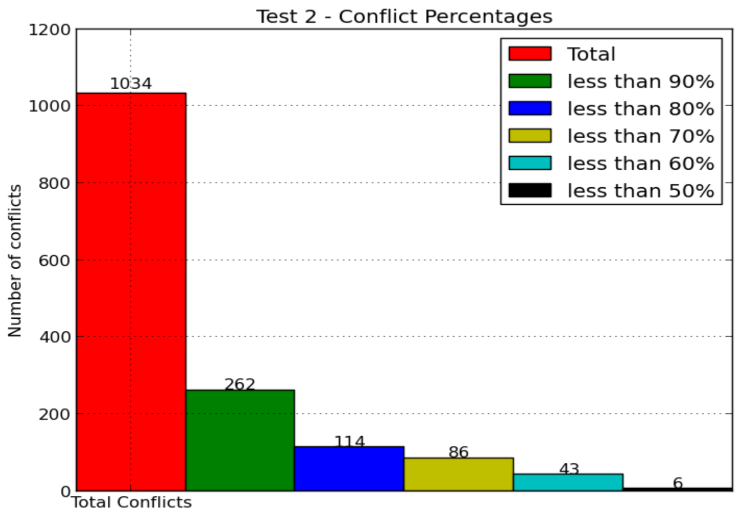


Figure 4.8: Test 2 - Conflict Percentages

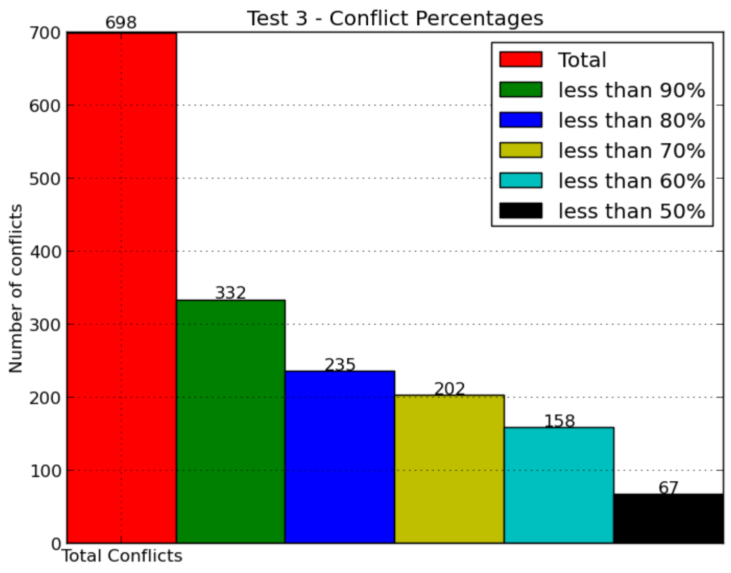


Figure 4.9: Test 3 - Conflict Percentages

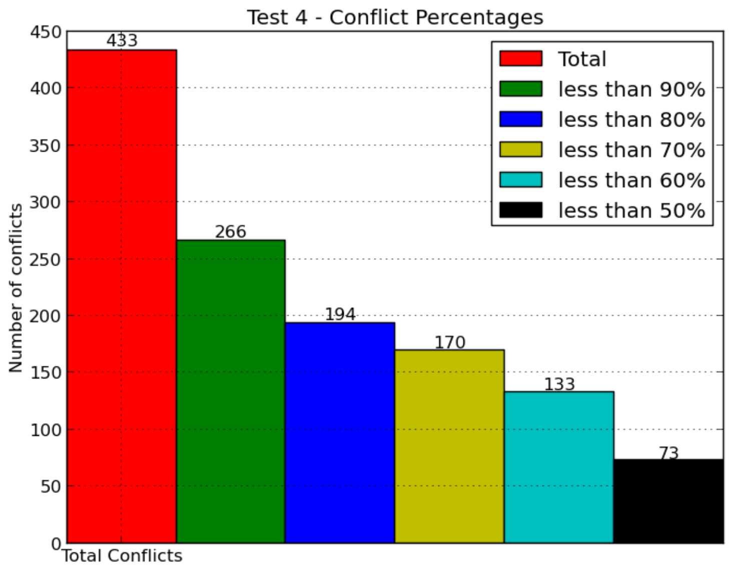


Figure 4.10: Test 4 - Conflict Percentages

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5: Conclusion and Future Work

This section discusses the overall findings of our research and offers suggestions on areas of future work to advance the state-of-the-art in identifying malicious and abusive infrastructure.

5.1 Conclusion

The goal of our research was to investigate the ability of packet-level TCP transport classifiers to discover abusive traffic flows, primarily those that are not found via traditional methods (e.g., RBL, signature based systems, etc.). Thus, we are most interested in discovering traffic to and from abusive infrastructure, especially where the abusive infrastructure has not been previously discovered or documented.

Our approach in testing this hypothesis involved a passive method of traffic analysis, MWID, a tool set that does not rely on packet inspection for content, C&C signatures, or sender reputation to discriminate traffic type. MWID leverages prior work in using fine-grained traffic features to identify discriminating features of communications involving abusive hosts. Of key importance was understanding the applicability of these methods to *live* network traffic, which we were able to obtain from the Naval Postgraduate School campus enterprise network. In total, we analyze four separate periods of real-world traffic.

We show the ability to discriminate between traffic flows that are abusive or malicious from those that are non-malicious, without the need for deep packet inspection, thus fully preserving the privacy of the data.

In contrast to the most recent related prior work [5], which used synthetically gathered data flows that were known malicious and non-malicious, we analyze “unknown” data flows–flows whose provenance is unknown to existing systems. Specifically, these unknown data flows are not listed in RBLs or whitelists, and therefore we do not have an immediate basis or knowledge of the nature of the host or the infrastructure supporting it. We use our system to predict the label of each unknown traffic flow, after training on a set of known flows. This provides for a more realistic approach at identifying and helping defend against not only known hosts and abusive infrastructure, but infrastructure which is of even greater threat: unknown abusive systems. Our hope is that this work brings transport-layer classifiers a step closer to operational deployment.

5.2 Key Results

Our results demonstrate that by utilizing per-packet TCP header information, distinct feature vectors of each flow, and supervised learning techniques, we are able to identify not only known abusive traffic, but successfully identify previously unknown forms of abusive traffic. The test data analyzed encompasses:

- over 119GB of data
- over 9 million flows of traffic
- over 92 million traffic features
- over 2 trillion bytes of information
- 15,117 Fetcher [5] injected URLs were extracted from our honeypot

The key findings of our experiment and results are encouraging:

- MWID's functionality, and use of RBLs in combination with Alexa to appropriately label our initial data sets, was accurate, with only a < (.001%) occurrence of a conflict, where the host could be classified as either malicious or non-malicious (§3.2).
- Our classifier averaged 97.8% CA when training and testing on flows known to the RBLs and Alexa (i.e., those with "GOOD" and "BAD" labels) (table 4.2).
- Fetcher Manual Inspection in conjunction with the Test 1 - Manual Inspection (§4.4.1) demonstrated that our method and approach with MWID in utilizing per-flow traffic features was successful in identifying *previously unknown* abusive infrastructure. With the Fetcher URLs, we positively identify and correctly classify 510 of 545 (94%) of the URLs from our honeypot that are unknown to the RBLs, resulting in a CA of 93.6%. Further, we show that in Test 1 (§4.4.1), among a random sample of 100 previously unknown flows labeled as abusive by MWID, 91 are truly abusive sites as verified through manual inspection. Thus, on unknown flows, MWID empirically provides an overall CA of 90.5%.
- In Test 2 (§4.2.2), with no Fetcher injected URLs, we discover 239 of 28,984 (8.0%) flows that exist on RBLs and are known "BAD." This was a surprising result, considering the military environment of NPS, and we were able to capture and classify the malicious host traffic on the network. Encouragingly, we successfully classified 55 of the 239 as abusive. While this only represented 24% of the overall number, the results help prove our hypothesis.
- In Test 2 (§4.4.2), our results are especially encouraging. Here we discovered and prop-

erly classified an “UNKNOWN” malicious site that was not injected as part of the Fetcher operation and was not found on any of the RBLs. We further verified this URL to be an abusive site through manual inspection.

- Results from tests 3 (§4.4.4) and 4 (§4.4.5) again confirmed our ability to positively identify previously unknown malicious traffic. We correctly classify 8 and 108 malicious hosts that were unknown to the RBLs respectively. We again verify the results through manual inspection to provide ground-truth.

5.3 Future Work

In this section we outline future work that will be beneficial to furthering the research and help bolster the ability to detect abusive infrastructure.

5.3.1 Algorithms

In our tests we initially started by using two learning algorithms, naïve Bayes and Decision Tree. However, after initial testing, we discovered that the decision tree performed much better than naïve Bayes and resulted in a much higher CA, to include a very low FP rate when training and testing on known labeled data.

First, further work is required to better understand the poor performance of naïve Bayes. Second, there exist many other machine learning algorithms. We recommend further experimentation with these to better understand the feasible classification performance.

5.3.2 Malicious URLs

In Chapter 4 we discussed a concern whereby malicious URLs that we extracted from our honeypot could become outdated, which may indicate the site is taken off-line, suspended (i.e., violation of terms, non-payment, etc.), or no longer resolves to a usable site (i.e., blank screen), etc.. While we did not attempt to determine a “life expectancy” of these URLs in this research, we did encounter some that would expire in as little as 3 days.

The fact that such URLs are short-lived prompted the need to manually sample the sites as quickly as possible. As future work, we plan to characterize the time between a malicious host being initially discovered by MWID and the time it takes to be added to any of the RBLs (if ever).

Whether abusive flows unknown to the RBLs are later included by them is important to understanding the full utility of MWID. Another avenue of research would include MWID having the ability to not only query the RBLs and Alexa in "real-time", but to report malicious sites once manual verification is complete.

5.3.3 Other Improvements

Other suggested pieces of future work include:

- Expand the number of sources used to collect data sets
  – by increasing the range of sources, this would give a more diverse data set to train and test with.
- Expand the packet captures to cover a week, vice smaller 24 hour increments
  – would allow us to continuously inject URLs as they are recorded by the honeypot, thereby giving a larger set of abusive hosts to train on.
- Expand on the number and type of RBLs that are used for determining malicious hosts.
  – while we employed the use of the more well known RBLs, it may be worthwhile to utilize a larger number of lists for labeling purposes.
- Implement the use of a whitelist, which is a list that would contain all verified non-malicious hosts.
  – these could help reduce the number of FNs, as the sites are known in advance, where currently sites such as .gov and .mil are being misclassified.

5.3.4 Deployment

Considering all of the work and analysis performed by MWID on live enterprise traffic, we recommend that a prototype model be designed, tested, and implemented. With the implementation phase taking place in a lab or virtual setting during testing, a matrix can be gathered to determine how well MWID performs and record any impact the system may have in an integrated environment or network.

With a prototype, more details can be gathered and used to determine the feasibility of a fully functional and deployable system. Of these details, key network characteristics, properties, and policy issues should be addressed. Essentially, this would help direct how MWID might be deployed into an existing or future network design.

Foreseeing MWID as a viable security option capable of running in parallel with existing Navy network security infrastructure, future work can offer recommendations and steps necessary to make this tool readily available for deployment on Naval systems.

5.4 Summary

To understand the practical, real-world performance of using transport traffic features to identify malicious web sites found in live traffic flows, we presented a novel approach using our MWID system as a viable option for identifying such sites.

We were able to successfully prove several key hypotheses. First, we were able to show that traffic features and characteristics associated with each flow of network traffic can in fact be used to classify data. Second, looking strictly at our manual inspection analysis, we positively identify and properly classify 718 previously unknown malicious hosts, given no prior knowledge of the flow and where the host and IP addresses were not identified by the RBLs.

MWID shows promise as a tool for raising the CA and lowering the overall FP rate in systems that seek to identify and block abusive or malicious traffic. Our hope is that this work serves as a step forward in bringing transport-based classifiers to production fruition.

THIS PAGE INTENTIONALLY LEFT BLANK

REFERENCES

[1] B. Stone-Gross, T. Holz, G. Stringhini, and G. Vigna, “The underground economy of spam: a botmaster’s perspective of coordinating large-scale spam campaigns,” in *Proceedings of the 4th USENIX conference on Large-scale exploits and emergent threats*, ser. LEET’11. Berkeley, CA, USA: USENIX Association, 2011, pp. 4–4. [Online]. Available: http://dl.acm.org/citation.cfm?id=1972441.1972447

[2] R. J. Carey, “Message from the DON CIO,” *CHIPS*, pp. 5–7, January 2009, http://www.doncio.navy.mil/chips/Issue.aspx?ID=13.

[3] M. M. Group, “Internet world stats,” 2013, http://www.internetworldstats.com/stats.htm.

[4] G. Kakavelakis, R. Beverly, and J. Young, “Auto-learning of SMTP TCP transport-layer features for spam and abusive message detection,” in *Proceedings of the 25th USENIX Large Installation Systems Administration Conference (LISA)*, Dec 2011. [Online]. Available: http://www.rbeverly.net/research/papers/autolearn-lisa11.html

[5] L. Nolan, “Transport traffic analysis for abusive infrastructure characterization,” Master’s thesis, Naval Postgraduate School, September 2012.

[6] R. Beverly and K. Sollins, “Exploiting transport-level characteristics of spam,” in *Proceedings of the Fifth Conference on Email and Anti-Spam (CEAS)*, August 2008. [Online]. Available: http://www.rbeverly.net/research/papers/spamflow-ceas08.html

[7] G. Gu, R. Perdisci, J. Zhang, and W. Lee, “Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection,” in *USENIX Security Symposium*, 2008, pp. 139–154.

[8] F. Tegeler, X. Fu, G. Vigna, and C. Kruegel, “Botfinder: finding bots in network traffic without deep packet inspection,” in *CoNEXT*, 2012, pp. 349–360.

[9] D. S. Anderson, C. Fleizach, S. Savage, and G. M. Voelker, “Spamscatter: characterizing internet scam hosting infrastructure,” in *Proceedings of 16th USENIX Security Symposium*. Berkeley, CA, USA: USENIX Association, 2007, pp. 10:1–10:14. [Online]. Available: http://dl.acm.org/citation.cfm?id=1362903.1362913

[10] G. Gu, J. Zhang, and W. Lee, “BotSniffer: Detecting botnet command and control channels in network traffic,” in *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS’08)*, February 2008.

[11] X. Hu, M. Knysz, and K. G. Shin, “Rb-seeker: Auto-detection of redirection botnets,” in *Proceedings of 16th Annual Network and Distributed System Security Symposium*, 2009.

[12] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, “Identifying suspicious urls: an application of large-scale online learning,” in *ICML*, 2009.

[13] Y. Xie, F. Yu, K. Achan, R. Panigrahy, G. Hulten, and I. Osipkov, "Spamming botnets: signatures and characteristics," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 171–182, 2008.

[14] S. B. Kotsiantis, "Supervised machine learning: A review of classification techniques," *Informatica (Slovenia)*, vol. 31, no. 3, pp. 249–268, 2007.

[15] T. Curk, J. Demšar, Q. Xu, G. Leban, U. Petrovič, I. Bratko, G. Shaulsky, and B. Zupan, "Microarray data mining with visual programming," *Bioinformatics*, vol. 21, pp. 396–398, Feb. 2005. [Online]. Available: http://bioinformatics.oxfordjournals.org/content/21/3/396.full.pdf

[16] I. Androutsopoulos, K. Koutsias, V. Chandrinos, G. Paliouras, and C. Spyropoulos, "An evaluation of naive bayesian anti-spam filtering," in *Proceeding of the Workshop of Machine Learning in the New Information Age*, 2000, pp. 9–17.

[17] E. Blanzieri and A. Bryl, "A survey of learning-based technique of email spam filtering," in *Artifical Intelligence Review*, vol. 29, March 1993, pp. 63–92.

[18] S. Russell and P. Norvig, in *Artificial Intelligence A Modern Approach.* Pearson Education Inc, March 2010, pp. 495–503.

[19] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, ser. The Morgan Kaufmann Series In Data Management Systems. Elsevier Books, Oxford, 2001. [Online]. Available: http://books.google.co.in/books?id=6hkR_ixby08C

[20] J. R. Quinlan, "Induction of decision trees," in *Machine Learning 1.* Kluwer Academic Publishers, 1986, pp. 81–106.

[21] "The Spamhaus Project," 2013, http://www.spamhaus.org/.

[22] "Spam and Open Relay Blocking System (SORBS)," 2013, http://www.sorbs.net/.

[23] "Malware Domain List," 2013, http://www.malwaredomainlist.com/.

[24] "Phish Tank," 2013, http://www.phishtank.com/.

[25] Alexa, "Top 1,000,000 sites," 2013, http://www.alexa.com/topsites/.

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia
2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California
3. Dr. Peter Denning
   Naval Postgraduate School
   Monterey, California
4. Dr. Robert Beverly
   Naval Postgraduate School
   Monterey, California
5. Dr. Geoffrey Xie
   Naval Postgraduate School
   Monterey, California
6. LCDR Tony Nichols
   Naval Postgraduate School
   Monterey, California