CHID: CONDITIONAL HYBRID INTRUSION DETECTION SYSTEM FOR REDUCING FALSE POSITIVES AND RESOURCE CONSUMPTION ON MALICOUS DATASETS

HASHEM MOHAMMED ALAIDAROS

DOCTOR OF PHILOSOPHY
UNIVERSITI UTARA MALAYSIA
2017

Awang Had Salleh
Graduate School
of Arts And Sciences

Universiti Utara Malaysia

PERAKUAN KERJA TESIS / DISERTASI
(Certification of thesis / dissertation)

Kami, yang bertandatangan, memperakukan bahawa
(We, the undersigned, certify that)

HASHEM MOHAMMED ABDULRAHMAN AL-AIDAROS

calon untuk Ijazah PhD
(candidate for the degree of)

telah mengemukakan tesis / disertasi yang bertajuk:
(has presented his/her thesis / dissertation of the following title):

"CHID: CONDITIONAL HYBRID INSTRUSION DETECTION SYSTEM FOR REDUCING FALSE POSITIVES AND RESOURCE CONSUMPTION ON MALICOUS DATASETS"

seperti yang tercatat di muka surat tajuk dan kulit tesis / disertasi.
(as it appears on the title page and front cover of the thesis / dissertation).

Bahawa tesis/disertasi tersebut boleh diterima dari segi bentuk serta kandungan dan meliputi bidang ilmu dengan memuaskan, sebagaimana yang ditunjukkan oleh calon dalam ujian lisan yang diadakan pada : *27 November 2016.*
That the said thesis/dissertation is acceptable in form and content and displays a satisfactory knowledge of the field of study as demonstrated by the candidate through an oral examination held on: November 27, 2016.

Pengerusi Viva: *(Chairman for VIVA)*	Assoc. Prof. Dr. Wan Rozaini Sheik Osman	Tandatangan *(Signature)* [illegible]
Pemeriksa Luar: *(External Examiner)*	Prof. Dr. Omar Zakaria	Tandatangan *(Signature)* [illegible]
Pemeriksa Dalam: *(Internal Examiner)*	Dr. Mohd Nizam Omar	Tandatangan *(Signature)* [illegible]
Nama Penyelia/Penyelia-penyelia: *(Name of Supervisor/Supervisors)*	Dr. Massudi Mahmuddin	Tandatangan *(Signature)* [illegible]

Tarikh:
(Date) *November 27, 2016*

Permission to Use

Abstrak

Memeriksa paket untuk mengesan pencerobohan berhadapan cabaran apabila berlakunya jumlah trafik rangkaian yang tinggi. Proses pengesanan berdasarkan paket bagi setiap muat beban pada wayar mengurangkan prestasi sistem pengesanan pencerobohan rangkaian (NIDS). Isu ini memerlukan kepada satu pengenalan NIDS berasaskan aliran untuk mengurangkan jumlah data yang akan diproses dengan memeriksa agregat maklumat dari paket yang berkaitan. Walau bagaimanapun, pengesanan berdasarkan aliran masih mengalami penjanaan amaran positif palsu kerana input data yang tidak lengkap. Kajian ini mencadangkan Pengesanan Pencerobohan Hibrid Bersyarat (CHID) dengan mencantumkan pengesanan berasaskan aliran dengan pengesanan berasaskan paket. Tambahan lagi, ia juga bertujuan untuk memperbaiki penggunaan sumber pendekatan pengesanan berasaskan paket. CHID menggunakan algoritma penilaian ciri pembalut atribut yang menandakan aliran hasad untuk analisis selanjutnya oleh pengesanan berasaskan paket. Pendekatan Rangka Kerja Input telah digunakan untuk mencetus aliran paket diantara pengesanan berasaskan paket dan berasaskan aliran. Eksperimen tapak ujiterkawal telah dijalankan untuk menilai prestasi mekanisme pengesanan CHID menggunakan set data yang diperolehi daripada pada kadar trafik yang berbeza. Hasil penilaian didapati CHID memperoleh peningkatan prestasi yang ketara dari segi penggunaan sumber dan kadar paket susut, berbanding pelaksanaan pengesanan berasaskan paket lalai. Pada kelajuan 200 Mbps, CHID dalam senario IRC-bot, boleh mengurangkan 50.6% dari penggunaan memori dan menyusut 18.1% penggunaan CPU tanpa paket susut. Pendekatan CHID boleh mengurangkan kadar positif palsu berdasarkan pengesanan berasaskan aliran dan mengurangkan penggunaan sumber pengesanan berasaskan paket disamping memelihara ketepatan pengesanan. Pendekatan CHID boleh dianggap sebagai sistem generik untuk diaplikasikan untuk sistem pemantauan pengesanan pencerobohan.

Kata Kunci: Pengesanan berasaskan aliran, Pengesanan berasaskan paket, Bro-NIDS, Rangka kerja input.

Abstract

Inspecting packets to detect intrusions faces challenges when coping with a high volume of network traffic. Packet-based detection processes every payload on the wire, which degrades the performance of network intrusion detection system (NIDS). This issue requires an introduction of a flow-based NIDS that reduces the amount of data to be processed by examining aggregated information of related packets. However, flow-based detection still suffers from the generation of the false positive alerts due to incomplete data input. This study proposed a Conditional Hybrid Intrusion Detection (CHID) by combining the flow-based with packet-based detection. In addition, it is also aimed to improve the resource consumption of the packet-based detection approach. CHID applied attribute wrapper features evaluation algorithms that marked malicious flows for further analysis by the packet-based detection. Input Framework approach was employed for triggering packet flows between the packet-based and flow-based detections. A controlled testbed experiment was conducted to evaluate the performance of detection mechanism's CHID using datasets obtained from on different traffic rates. The result of the evaluation showed that CHID gains a significant performance improvement in terms of resource consumption and packet drop rate, compared to the default packet-based detection implementation. At a 200 Mbps, CHID in IRC-bot scenario, can reduce 50.6% of memory usage and decreases 18.1% of the CPU utilization without packets drop. CHID approach can mitigate the false positive rate of flow-based detection and reduce the resource consumption of packet-based detection while preserving detection accuracy. CHID approach can be considered as generic system to be applied for monitoring of intrusion detection systems.

Keywords: Flow-based detection, Packet-based detection, Input Framework approach.

Acknowledgement

First of all, I would like to express my sincere and deep gratitude to my supervisor Dr. Massudi Mahmuddin who provided considerable and invaluable insights and comments to help me on this journey. Without his patient support, enlightened guidance, it is impossible for me to complete and enhance the quality of my work.

I would like to thank the InterNetWork Lab team for their co-operation, kindness, and sharing useful discussions for my research.

Thank you goes to Dr. Shakeel Habeeb, the dean of Prince Sultan College (PSCJ), Al-Faisal University, for providing me the testbed resources for my research.

Special thanks go to Johanna Amann and Robin Sommer, the developers of the Bro-IDS. Their supports with the many problems encountered during experiments were of great help.

Finally, my heartiest gratitude goes to my beloved family, in particular, my parents my wife without their love, support and encouragement, it would not be possible for me end this journey.

Table of Contents

List of Tables

List of Figures

List of Appendices

List of Abbreviations

API	Application Programming Interface
BPF	Berkeley Packet Filtering
Broccoli	Bro Client Communications Library
C&C	Command and Control
CHID	Conditional Hybrid Intrusion Detection
CTU	Czech Technical University
DARPA	Defence Advanced Research Project Agency
DPI	Deep Packet Inspection
DoS	Denial of Service
FL	FLow-based-detection
FPA	Front Payload Aggregation
FPR	False Positive Rate
HIDS	Host-based Intrusion Detection System
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
IF	Input Framework
IP	Internet Protocol
IPFIX	IP Flow Information Export
IPS	Intrusion Prevention System
IRC	Internet Relay Chat
ISOT	Information Security and Object Technology

ISP	Internet Service Provider
KDD	Knowledge Discovery in Dataset
LAN	Local Area Networks
NAT	Network Address Translation
NIDS	Network Intrusion Detection System
OSI	Open Systems Interconnection
RP	Received Packets
P2P	Peer to Peer
PCAP	Packet Capturing
PH	Packet-based in Hybrid
PO	Packet-based Only
PSCJ	Prince Sultan College Jeddah
PYL	Payload
SQL	Structured Query Language
SSH	Secure Shell
TCP	Transmission Control Protocol
TPR	True Positive Rate
UDP	User Datagram Protocol
VoIP	Voice over Internet Protocol
WAN	Wide Area Networks

List of Publications

[1] H. Alaidaros and M. Mahmuddin, “Conditional hybrid approach for intrusion detection,” *Research Journal Information Technology*, vol. 8, pp. 55-65, 2016.

[2] H. Alaidaros, and M. Mahmuddin, “Flow-based Approach on Bro Intrusion Detection,” *in Advancement on Information Technology International Conference*, 2015

[2] H. Alaidaros, M. Mahmuddin, A. Al Mazari, “From Packet-based Towards Hybrid Packet-based and Flow-based Monitoring for Efficient Intrusion Detection: An Overview” *in 1st Taibah University International Conference on Computing and Information Technology (ICCIT12)*, 2012

[4] H. Alaidaros, M. Mahmuddin, A. Al Mazari, “An Overview of Flow-based and Packet-based Intrusion Detection Performance in High Speed Network,” *in 12th International Arab Conference on Information Technology (ACIT12)*, 2011.

CHAPTER ONE
INTRODUCTION

1.1 Background

The number of Internet clients and services is growing more and more [1]. New Internet applications give users benefits for either their businesses or future life. The Internet is a powerful medium that has changed how people communicate and do businesses with the partners. These universal applications let companies achieve things that never been imagined before.

In addition to growing of the Internet users, networks become bigger and bigger. Although the Internet gives users' bright life and good businesses, it also has its unknown dark face. Since many new Internet services, devices, and hosts are developing, the number of vulnerabilities either in user smartphones, computers or servers is also increasing [2]. The more computers connected to the Internet the more possibility that the attacks take place. Many security gaps are exposed and misused by attacks. Unfortunately, attacks are growing with the Internet almost in parallel, and the race between them is continuing.

The number and the damage cost by those attacks are rising continuously. The security threats can exploit all types of the network, including LAN-based clusters, intranet, large-scale computational grids, and peer-to-peer service networks. These threats also exploit all exposed protocols and operating systems (OS) threatening different kinds of their applications such as database and web servers. Considering the damage cost originated from the attacks, it is important to detect an attack as soon as possible. The

REFERENCES

[1] Internet World Stats, "Internet Growth Statistics," 2016, [Online; accessed 8-Dec-2015]. [Online]. Available: http://www.internetworldstats.com/emarketing.htm

[2] J. Nazario and J. Kristoff, "Internet Infrastructure Security," *IEEE Security & Privacy*, vol. 10, pp. 24-25, 2012.

[3] E. Amoroso, *Intrusion Detection: An Introduction to Internet Surveillance, Correlation, and Response*: New Jersey, 1999.

[4] G. Khalil, "Open Source IDS High Performance Shootout," *SANS Institute InfoSec Reading Room*, 2015.

[5] S. S. Silva, R. M. Silva, R. C. Pinto, and R. M. Salles, "Botnets: A survey," *Computer Networks*, vol. 57, pp. 378-403, 2013.

[6] H. Debar, M. Dacier, and A. Wespi, "Towards a Taxonomy of Intrusion Detection Systems," *Computer Networks*, vol. 31, pp. 805-822, 1999.

[7] N. Weng, L. Vespa, and B. Soewito, "Deep Packet Pre-filtering and Finite State Encoding for Adaptive Intrusion Detection System," *Computer Networks*, vol. 55, pp. 1648-1661, 2011.

[8] R. Koch, "Towards Next-generation Intrusion Detection," in *2011 3rd International Conference on Cyber Conflict*, 2011, pp. 1-18.

[9] M. Golling, R. Hofstede, and R. Koch, "Towards Multi-layered Intrusion Detection in High Speed Networks," in *6th International Conference On Cyber Conflict (CyCon 2014)*, 2014, pp. 191-206.

[10] J. Svoboda, "Network Traffic Analysis with Deep Packet Inspection Method," Master thesis, Faculty of Informatics, Masaryk University, Brno, 2014.

[11] M. Nor, "Malware Detection Using IP Flow Level Attributes," *Journal of Theoretical and Applied Information Technology*, vol. 57, 2013.

[12] H. Dreger, A. Feldmann, V. Paxson, and R. Sommer, "Operational Experiences with High-volume Network Intrusion Detection," in *11th ACM conference on Computer and Communications Security*, 2004, pp. 2-11.

[13] A. Papadogiannakis, M. Polychronakis, and E. P. Markatos, "Improving the Accuracy of Network Intrusion Detection Systems under Load using Selective Packet Discarding," in *Proceedings of the Third European Workshop on System Security*, 2010, pp. 15-21.

[14] I. Sourdis, D. N. Pnevmatikatos, and S. Vassiliadis, "Scalable Multigigabit Pattern Matching for Packet Inspection," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, pp. 156-166, 2008.

[15] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion Detection System: A Comprehensive Review," *Journal of Network and Computer Applications*, vol. 36, pp. 16-24, 2013.

[16] R. Hofstede, V. Bartoš, A. Sperotto, and A. Pras, "Towards Real-time Intrusion Detection for NetFlow and IPFIX," in *Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013)*, 2013, pp. 227-234.

[17] Y. Abuadlla, G. Kvascev, S. Gajin, and Z. Jovanovic, "Flow-based Anomaly Intrusion Detection System using Two Neural Network Stages," *Comput. Sci. Inf. Syst.*, vol. 11, pp. 601-622, 2014.

[18] J. Zhang, R. Perdisci, W. Lee, X. Luo, and U. Sarfraz, "Building a Scalable System for Stealthy P2P-Botnet Detection," *IEEE Transactions on Information Forensics and Security*, vol. 9, pp. 27-38, 2014.

[19] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An Overview of IP Flow-based Intrusion Detection," *IEEE Communications Surveys & Tutorials*, vol. 12, pp. 343-356, 2010.

[20] T. Hyslip and J. Pittman, "A Survey of Botnet Detection Techniques by Command and Control Infrastructure," *Journal of Digital Forensics, Security and Law*, vol. 10, pp. 7-26, 2015.

[21] L. Sheng, L. Zhiming, H. Jin, D. Gaoming, and H. Wen, "A Distributed Botnet Detecting Approach Based on Traffic Flow Analysis," in *Second International Conference on Instrumentation, Measurement, Computer, Communication and Control (IMCCC)*, 2012, pp. 124-128.

[22] T. Limmer and F. Dressler, "Flow-based Front Payload Aggregation," in *IEEE LCN*, 2009, pp. 1102-1109.

[23] F. Hensel, "Flow-based and Packet level-based Intrusion Detection as Complementary Concepts," High Diploma Thesis, Department of Informatics, University of Zurich, Zurich, Switzerland, 2008.

[24] S. Khattak, N. R. Ramay, K. R. Khan, A. A. Syed, and S. A. Khayam, "A Taxonomy of Botnet Behavior, Detection, and Defense," *IEEE Communications Surveys & Tutorials*, vol. 16, pp. 898-924, 2014.

[25] S. Soltani, S. A. H. Seno, M. Nezhadkamali, and R. Budiarto, "A Survey on Real World Botnets and Detection Mechanisms," *International Journal of Information and Network Security*, vol. 3, p. 116, 2014.

[26] Z. Zhu, G. Lu, Y. Chen, Z. J. Fu, P. Roberts, and K. Han, "Botnet Research Survey," in *32nd Annual IEEE International Computer Software and Applications Conference*, 2008, pp. 967-972.

[27] S. Abt and H. Baier, "Towards Efficient and Privacy-Preserving Network-Based Botnet Detection Using Netflow Data," in *Internation Network Conference*, 2012, pp. 37-50.

[28] V. M. Igure and R. D. Williams, "Taxonomies of Attacks and Vulnerabilities in Computer Systems," *IEEE Communications Surveys & Tutorials*, vol. 10, pp. 6-19, 2008.

[29] Symantec Corp, "Internet Security Threat Report," 2016, [Online; accessed 4-Feb-2016]. [Online]. Available: https://www.symantec.com/security-center/threat-report

[30] S. X. Wu and W. Banzhaf, "The Use of Computational Intelligence in Intrusion Detection Systems: A Review," *Applied Soft Computing*, vol. 10, pp. 1-35, 2010.

[31] Snort IDS, "Snort," 2012, [Online; accessed 8-May-2013]. [Online]. Available: www.snort.org

[32] J. GERBER,"Suricata: A Next Generation IDS/IPS Engine," 2010, [Online; accessed 4-May-2014]. [Online]. Available: https://suricata-ids.org/

[33] P. Mehra, "A Brief Study and Comparison of Snort and Bro Open Source Network Intrusion Detection Systems," *International Journal of Advanced Research in Computer and Communication Engineering,* vol. 1, pp. 383-386, 2012.

[34] J. Beale, A. R. Baker, and J. Esler, *Snort: IDS and IPS toolkit*: Syngress, 2007.

[35] Bro, "Bro IDS," 2012, [Online; accessed 5-June-2013]. [Online]. Available: www.bro.org

[36] B. Morin and L. Mé, "Intrusion Detection and Virology: an Analysis of Differences, Similarities and Complementariness," *Journal in Computer Virology,* vol. 3, pp. 39-49, 2007.

[37] R. R. Singh, N. Gupta, and S. Kumar, "To Reduce the False Alarm in Intrusion Detection System Using Self Organizing Map," *International Journal of Soft Computing and Engineering (IJSCE),* vol. 1, pp. 27-32, 2011.

[38] K. Wang, & Stolfo, S. J. , "Anomalous payload-based Network Intrusion Detection," *Recent Advances in Intrusion Detection,* p. 19, 2004.

[39] M. Mahoney and P. Chan, "Learning Non-stationary Models of Normal Network Traffic for Detecting Novel Attacks," in *8th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, 2002, pp. 376–385.

[40] M.-S. Kim, H.-J. Kong, S.-C. Hong, S.-H. Chung, and J. W. Hong, "A Flow-based Method for Abnormal Network Traffic Detection," in *Network Operations and Management Symposium*, 2004, pp. 599-612.

[41] S. M. Hussein, F. H. M. Ali, and Z. Kasiran, "Evaluation Effectiveness of Hybrid IDS using Snort with Naïve Bayes to Detect Attacks," in *Second International Conference on Digital Information and Communication Technology and it's Applications (DICTAP)*, 2012, pp. 256-260.

[42] Z. M. Fadlullah, T. Taleb, A. V. Vasilakos, M. Guizani, and N. Kato, "DTRAB: Combating Against Attacks on Encrypted Protocols through Traffic-feature Analysis," *IEEE/ACM Transactions on Networking (TON),* vol. 18, pp. 1234-1247, 2010.

[43] K.-K. Tseng, J. Lo, Y. Liu, S.-H. Chang, M. Merabti, F. Ng, CK, *et al.*, "A Feasibility Study of Stateful Automaton Packet Inspection for Streaming Application Detection Systems," *Enterprise Information Systems,* pp. 1-20, 2016.

[44] H. Dreger, A. Feldmann, V. Paxson, and R. Sommer, "Predicting the Resource Consumption of Network Intrusion Detection Systems," in *International Workshop on Recent Advances in Intrusion Detection*, 2008, pp. 135-154.

[45] F. Fusco and L. Deri, "High Speed Network Traffic Analysis with Commodity Multi-core Systems," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet measurement*, 2010, pp. 218-224.

[46] J. Morgan, "Streaming Network Traffic Analysis Using Active Learning," Master thesis, Department of Computer Science, Dalhousie University, Halifax, Nova Scotia, 2015.

[47] M. Pihelgas, "A Comparative Analysis of Open-Source Intrusion Detection Systems," Master thesis, Departement of Computer Science, Tallinn University of Technology, Tallinn, 2012.

[48] J. Korenek and P. Kobiersky, "Intrusion Detection System Intended for Multigigabit Networks," in *2007 IEEE Design and Diagnostics of Electronic Circuits and Systems*, 2007, pp. 1-4.

[49] L. Braun, A. Didebulidze, N. Kammenhuber, and G. Carle, "Comparing and Improving Current Packet Capturing Solutions based on Commodity Hardware," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, 2010, pp. 206-217.

[50] P. Lambruschini, M. Raggio, R. Bajpai, and A. Sharma, "Efficient Implementation of Packet Pre-filtering for Scalable Analysis of IP Traffic on High-speed Lines," in *20th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2012, pp. 1-5.

[51] D. Ficara, G. Antichi, A. Di Pietro, S. Giordano, G. Procissi, and F. Vitucci, "Sampling Techniques to Accelerate Pattern Matching in Network Intrusion Detection Systems," in *IEEE International Conference on Communications (ICC)*, 2010, pp. 1-5.

[52] G. Jacob, P. M. Comparetti, M. Neugschwandtner, C. Kruegel, and G. Vigna, "A Static Packer-agnostic Filter to Detect Similar Malware Samples," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2012, pp. 102-122.

[53] Q. Zhao, J. Xu, and A. Kumar, "Detection of Super Sources and Destinations in High-speed Networks: Algorithms, Analysis and Evaluation," *IEEE Journal on Selected Areas in Communications*, vol. 24, pp. 1840-1852, 2006.

[54] F. Haddadi, J. Morgan, E. Gomes Filho, and A. N. Zincir-Heywood, "Botnet Behaviour Analysis using IP Flows: with HTTP Filters using Classifiers," in *Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on*, 2014, pp. 7-12.

[55] C.-H. Lin and S.-C. Chang, "Efficient Pattern Matching Algorithm for Memory Architecture," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, pp. 33-41, 2011.

[56] N. Weaver, V. Paxson, and J. M. Gonzalez, "The Shunt: an FPGA-based Accelerator for Network Intrusion Prevention," in *Proceedings of the 2007 ACM/SIGDA 15th International Symposium on Field Programmable Gate Arrays*, 2007, pp. 199-206.

[57] G. Munz and G. Carle, "Real-time Analysis of Flow Data for Network Attack Detection," in *2007 10th IFIP/IEEE International Symposium on Integrated Network Management*, 2007, pp. 100-108.

[58] A. Karim, R. B. Salleh, M. Shiraz, S. A. A. Shah, I. Awan, and N. B. Anuar, "Botnet Detection Techniques: Review, Future Trends, and Issues," *Journal of Zhejiang University SCIENCE*, vol. 15, pp. 943-983, 2014.

[59] P. Porras, H. Saidi, and V. Yegneswaran, "A Multi-perspective Analysis of the Storm (Peacomm) Worm," Computer Science Laboratory, Tech. Rep., 2007

[60] G. Sinclair, C. Nunnery, and B. B. Kang, "The Waledac Protocol: The How and Why," in *4th International Conference on Malicious and Unwanted Software (MALWARE)*, 2009, pp. 69-77.

[61] D. Andriesse and H. Bos, "An Analysis of the Zeus Peer-to-Peer Protocol," 2013. [Online]. Available: http://www.few.vu.nl/~dae400/papers/zeus-tech-report-2013.pdf

[62] W. Zilong, W. Jinsong, H. Wenyi, and X. Chengyi, "The Detection of IRC Botnet based on Abnormal Behavior," in *2010 Second International Conference on Multimedia and Information Technology*, 2010.

[63] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "BotMiner: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection," in *USENIX Security Symposium*, 2008, pp. 139-154.

[64] G. Gu, J. Zhang, and W. Lee, "BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic," 2008.

[65] J. Goebel and T. Holz, "Rishi: Identify Bot Contaminated Hosts by IRC Nickname Evaluation," *HotBots*, vol. 7, pp. 8-8, 2007.

[66] T.-F. Yen and M. K. Reiter, "Traffic Aggregation for Malware Detection," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2008, pp. 207-227.

[67] G. Gu, P. A. Porras, V. Yegneswaran, M. W. Fong, and W. Lee, "Bothunter: Detecting Malware Infection through IDS-Driven Dialog Correlation," in *Usenix Security*, 2007, pp. 1-16.

[68] P. Wurzinger, L. Bilge, T. Holz, J. Goebel, C. Kruegel, and E. Kirda, "Automatically Generating Models for Botnet Detection," in *European Symposium on Research in Computer Security*, 2009, pp. 232-249.

[69] D. H. Kim, T. Lee, J. Kang, H. Jeong, and H. P. In, "Adaptive Pattern Mining Model for Early Detection of Botnet Propagation Scale," *Security and Communication Networks*, vol. 5, pp. 917-927, 2012.

[70] S. García, A. Zunino, and M. Campo, "Botnet Behavior Detection using Network Synchronism," *Privacy, Intrusion Detection and Response: Technologies for Protecting Networks*, pp. 122-144, 2011.

[71] G. Jian, K. Zheng, Y. Yang, and X. Niu, "An Evaluation Model of Botnet based on Peer to Peer," in *Fourth International Conference on Computational Intelligence and Communication Networks (CICN)*, 2012, pp. 925-929.

[72] L. Dan, L. Yichao, H. Yue, and L. Zongwen, "A P2P-Botnet Detection Model and Algorithms based on Network Streams Analysis," in *International Conference on Future Information Technology and Management Engineering (FITME)*, 2010, pp. 55-58.

[73] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, *et al.*, "Flow Monitoring Explained: From Packet Capture to Data Analysis with NetFlow and IPFIX," *IEEE Communications Surveys & Tutorials*, vol. 16, pp. 2037-2064, 2014.

[74] B. Claise, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information," RFC 5101, 2008. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5101.txt

[75] C. Estan, K. Keys, D. Moore, and G. Varghese, "Building a Better NetFlow," *ACM SIGCOMM Computer Communication Review*, vol. 34, p. 245, 2004.

[76] U. Banerjee, A. Vashishtha, and M. Saxena, "Evaluation of the Capabilities of WireShark as a Tool for Intrusion Detection," *International Journal of Computer Applications*, vol. 6, 2010.

[77] L. MartinGarcia,"TcpDump and Libpcap," 2012, [Online; accessed 9-July-2012]. [Online]. Available: http://www.tcpdump.org

[78] V. Kumaran, "Event Stream Database based Architecture to Detect Network Intrusion," in *Proceedings of the 7th ACM International Conference on Distributed Event-based Systems*, 2013, pp. 241-248.
[79] V. Carela-Español, P. Barlet-Ros, A. Cabellos-Aparicio, and J. Solé-Pareta, "Analysis of the Impact of Sampling on NetFlow Traffic Classification," *Computer Networks*, vol. 55, pp. 1083-1099, 2011.
[80] N. Duffield, "Sampling for Passive Internet Measurement: A Review," *Statistical Science,* pp. 472-498, 2004.
[81] T. Zseby, T. Hirsch, and B. Claise, "Packet Sampling for Flow Accounting: Challenges and Limitations," in *International Conference on Passive and Active Network Measurement*, 2008, pp. 61-71.
[82] D. Brauckhoff, M. May, and B. Plattner, "Flow-level Anomaly Detection-Blessing or Curse," in *IEEE INFOCOM Conference*, 2007.
[83] J. David and C. Thomas, "DDoS Attack Detection using Fast Entropy Approach on Flow-based Network Traffic," *Procedia Computer Science,* vol. 50, pp. 30-36, 2015.
[84] S. Yu, W. Zhou, W. Jia, S. Guo, Y. Xiang, and F. Tang, "Discriminating DDoS Attacks from Flash Crowds using Flow Correlation Coefficient," *IEEE Transactions on Parallel and Distributed Systems,* vol. 23, pp. 1073-1080, 2012.
[85] S. A. Abdulla, S. Ramadass, A. Altaher, and A. A. Nassiri, "Setting a Worm Attack Warning by Using Machine Learning to Classify Netflow Data," *International Journal of Computer Applications,* vol. 36, pp. 49-56, 2011.
[86] F. Dressler, W. Jaegers, and R. German, "Flow-based Worm Detection using Correlated Honeypot Logs," in *Communication in Distributed Systems Conference*, 2007, pp. 1-6.
[87] L. Hellemons, L. Hendriks, R. Hofstede, A. Sperotto, R. Sadre, and A. Pras, "SSHCure: a Flow-based SSH Intrusion Detection System," in *IFIP International Conference on Autonomous Infrastructure, Management and Security*, 2012, pp. 86-97.
[88] M. Vizváry and J. Vykopal, "Flow-based Detection of RDP Brute-force Attacks," in *Proceedings of 7th International Conference on Security and Protection of Information (SPI 2013),* 2013.
[89] P. Amini, R. Azmi, and M. Araghizadeh, "Botnet Detection using NetFlow and Clustering," *Advances in Computer Science: an International Journal,* vol. 3, pp. 139-149, 2014.
[90] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, *et al.*, "Botnet Detection based on Traffic Behavior Analysis and Flow Intervals," *Computers & Security,* vol. 39, pp. 2-16, 2013.
[91] J. François, S. Wang, and T. Engel, "BotTrack: Tracking Botnets using NetFlow and PageRank," in *International Conference on Research in Networking*, 2011, pp. 1-14.
[92] M. Stevanovic and J. M. Pedersen, "Machine Learning for Identifying Botnet Network Traffic," *Journal of Aalborg University*, 2013.
[93] S. Ganapathy, K. Kulothungan, S. Muthurajkumar, M. Vijayalakshmi, P. Yogesh, and A. Kannan, "Intelligent Feature Selection and Classification

Techniques for Intrusion Detection in Networks: A Survey," *EURASIP Journal on Wireless Communications and Networking*, vol. 2013, p. 271, 2013.
[94] M. Stevanovic and J. M. Pedersen, "An Efficient Flow-based Botnet Detection using Supervised Machine Learning," in *International Conference on Computing, Networking and Communications (ICNC)*, 2014, pp. 797-801.
[95] N. Bhargava, G. Sharma, R. Bhargava, and M. Mathuria, "Decision Tree Analysis on J48 Algorithm for Data Mining," *Proceedings of International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, 2013.
[96] M. N. Anyanwu and S. G. Shiva, "Comparative Analysis of Serial Decision Tree Classification Algorithms," *International Journal of Computer Science and Security*, vol. 3, pp. 230-240, 2009.
[97] A. Liaw and M. Wiener, "Classification and Regression by Random Forest," *R news*, vol. 2, pp. 18-22, 2002.
[98] A. Nogueira, P. Salvador, and F. Blessa, "A Botnet Detection System based on Neural Networks," in *Fifth International Conference on Digital Telecommunications (ICDT)*, 2010, pp. 57-62.
[99] S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, *et al.*, "Detecting P2P Botnets through Network Behavior Analysis and Machine Learning," in *Ninth Annual International Conference on Privacy, Security and Trust (PST)*, 2011, pp. 174-180.
[100] S. Ting, W. Ip, and A. H. Tsang, "Is Naive Bayes a Good Classifier for Document Classification," *International Journal of Software Engineering and Its Applications*, vol. 5, pp. 37-46, 2011.
[101] D. Miller,"Softflowd: A Software Netflow Probe," 2012, [Online; accessed 7-June-2013]. [Online]. Available: http://www.mindrot.org/projects/softflowd/
[102] F. Tegeler, X. Fu, G. Vigna, and C. Kruegel, "Botfinder: Finding Bots in Network Traffic without Deep Packet Inspection," in *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, 2012, pp. 349-360.
[103] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel, "Disclosure: Detecting Botnet Command and Control Servers through Large-scale Netflow Analysis," in *Proceedings of the 28th Annual Computer Security Applications Conference*, 2012, pp. 129-138.
[104] U. Wijesinghe, U. Tupakula, and V. Varadharajan, "An Enhanced Model for Network Flow Based Botnet Detection," in *Proceedings of the 38th Australasian Computer Science Conference (ACSC 2015)*, 2015, p. 30.
[105] G. Schaffrath, & B. Stiller, , "Conceptual Integration of Flow-based and Packet-based Network Intrusion Detection," *Resilient Networks and Services*, pp. 190-194, 2008.
[106] J. Steinberger, L. Schehlmann, S. Abt, and H. Baier, "Anomaly Detection and Mitigation at Internet Scale: A survey," in *IFIP International Conference on Autonomous Infrastructure, Management and Security*, 2013, pp. 49-60.
[107] M. A. Mehmood, A. Feldmann, S. Uhlig, and W. Willinger, "We Are All Treated Equal, Aren't We?—Flow-level Performance as a Function of Flow Size," in *Networking Conference, 2014 IFIP*, 2014, pp. 1-9.

[108] G. F. Guo, "The Study of the Ontology and Context Verification Based Intrusion Detection Model," in *Applied Mechanics and Materials*, 2014, pp. 3338-3341.

[109] U. Shankar and V. Paxson, "Active Mapping: Resisting NIDS Evasion without Altering Traffic," in *Proceedings Symposium on Security and Privacy*, 2003, pp. 44-61.

[110] F. Valeur, G. Vigna, C. Kruegel, and R. A. Kemmerer, "Comprehensive Approach to Intrusion Detection Alert Correlation," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, pp. 146-169, 2004.

[111] M. Sourour, B. Adel, and A. Tarek, "Environmental Awareness Intrusion Detection and Prevention System Toward Reducing False Positives and False Negatives," in *IEEE Symposium on Computational Intelligence in Cyber Security*, 2009, pp. 107-114.

[112] G. S. Kumar and C. Sirisha, "Robust Preprocessing and Random Forests Technique for Network Probe Anomaly Detection," *International Journal of Soft Computing and Engineering (IJSCE) ISSN*, pp. 2231-2307, 2012.

[113] D. G. Bhatti and P. Virparia, "Data Preprocessing for Reducing False Positive Rate in Intrusion Detection," *International Journal of Computer Applications*, vol. 57, 2012.

[114] D. G. Bhatti, P. Virparia, and B. Patel, "Conceptual Framework for Soft Computing based Intrusion Detection to Reduce False Positive Rate," *International Journal of Computer Applications*, vol. 44, pp. 1-3, 2012.

[115] G. P. Spathoulas and S. K. Katsikas, "Using a Fuzzy Inference System to Reduce False Positives in Intrusion Detection," in *2009 16th International Conference on Systems, Signals and Image Processing*, 2009, pp. 1-4.

[116] T. Pietraszek and A. Tanner, "Data Mining and Machine Learning—Towards Reducing False Positives in Intrusion Detection," *Information Security Technical Report*, vol. 10, pp. 169-183, 2005.

[117] D. Bolzoni, B. Crispo, and S. Etalle, "ATLANTIDES: An Architecture for Alert Verification in Network Intrusion Detection Systems," in *LISA*, 2007, pp. 1-12.

[118] T. Kaur, "A Hybrid approach using Signature and Anomaly Detection to Detect Network Intrusions," Ph.D. thesis, Thapar Univeristy Patiala, 2013.

[119] G. Gu, M. Sharif, X. Qin, D. Dagon, W. Lee, and G. Riley, "Worm Detection, Early Warning and Response based on Local Victim Information," in *20th Annual Computer Security Applications Conference*, 2004, pp. 136-145.

[120] K. Wang, G. Cretu, and S. J. Stolfo, "Anomalous Payload-based Worm Detection and Signature Generation," in *International Workshop on Recent Advances in Intrusion Detection*, 2005, pp. 227-246.

[121] A. D. Todd, R. A. Raines, R. O. Baldwin, B. E. Mullins, and S. K. Rogers, "Alert Verification Evasion through Server Response Forging," in *International Workshop on Recent Advances in Intrusion Detection*, 2007, pp. 256-275.

[122] M. A. Aydın, A. H. Zaim, and K. G. Ceylan, "A Hybrid Intrusion Detection System Design for Computer Network Security," *Computers & Electrical Engineering*, vol. 35, pp. 517-526, 2009.

[123] E. Tombini, H. Debar, L. Mé, and M. Ducassé, "A Serial Combination of Anomaly and Misuse IDSes Applied to HTTP Traffic," in *20th Computer Security Applications Conference* 2004, pp. 428-437.

[124] Y.-X. Ding, M. Xiao, and A.-W. Liu, "Research and Implementation on Snort-based Hybrid Intrusion Detection System," in *2009 International Conference on Machine Learning and Cybernetics*, 2009, pp. 1414-1418.

[125] K. Hwang, M. Cai, Y. Chen, and M. Qin, "Hybrid Intrusion Detection with Weighted Signature Generation over Anomalous Internet Episodes," *IEEE Transactions on Dependable and Secure Computing*, vol. 4, pp. 41-55, 2007.

[126] J. Yang, X. Chen, X. Xiang, and J. Wan, "HIDS-DT: An Effective Hybrid Intrusion Detection System Based on Decision Tree," in *International Conference on Communications and Mobile Computing (CMC)*, 2010, pp. 70-75.

[127] J. Zhang and M. Zulkernine, "A Hybrid Network Intrusion Detection Technique using Random Forests," in *First International Conference on Availability, Reliability and Security (ARES'06)*, 2006, p. 8 pp.

[128] S. M. Hussein, F. H. M. Ali, and Z. Kasiran, "Evaluation effectiveness of Hybrid IDS using Snort with Naïve Bayes to Detect Attacks," in *Second International Conference on Digital Information and Communication Technology and it's Applications*, 2012, pp. 256-260.

[129] D. J. Day, D. A. Flores, and H. S. Lallie, "CONDOR: A Hybrid IDS to Offer Improved Intrusion Detection," in *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, 2012, pp. 931-936.

[130] V. Jacobson and S. McCanne, "libpcap: Packet Capture Library," *Lawrence Berkeley Laboratory, Berkeley, CA*, 2009.

[131] C. Kreibich and R. Sommer, "Policy-controlled Event Management for Distributed Intrusion Detection," in *25th IEEE International Conference on Distributed Computing Systems Workshops*, 2005, pp. 385-391.

[132] B. Amann, R. Sommer, A. Sharma, and S. Hall, "A Lone Wolf No More: Supporting Network Intrusion Detection with Real-time Intelligence," in *International Workshop on Recent Advances in Intrusion Detection*, 2012, pp. 314-333.

[133] L. Deri,"PF_Ring Packet Capture," 2011, [Online; accessed 4-May-2013]. [Online]. Available: http://www.ntop.org

[134] J. Stebelton,"Berkeley Packet Filters – The Basics," 2014, [Online; accessed 5-May-2013]. [Online]. Available: http://www.infosecwriters.com/text_resources/pdf/JStebelton_BPF.pdf

[135] L. Deri and N. Spa, "nProbe: An Open Source Netflow Probe for Gigabit Networks," in *TERENA Networking Conference*, 2003.

[136] S. Astashonok,"fprobe: a NetFlow Probe," 2007, [Online; accessed 25-October-2013]. [Online]. Available: http://fprobe.sourceforge.net/

[137] P. B. Ruthven, "Contextual Profiling of Homogeneous User Groups for Masquerade Detection," Master Thesis, Department of Computer Science and Media Technology, Gjøvik University, Norway, 2014.

[138] Logging Framework, "Bro 2.4.1 documentation Framework," [Online; accessed 19-Dec-2013]. [Online]. Available: https://www.bro.org/sphinx/frameworks/logging.html#streams

[139] R. G. Sargent, "Verification and Validation of Simulation Models," *Journal of Simulation*, vol. 7, pp. 12-24, 2013.

[140] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research," *Journal of Management Information Systems*, vol. 24, pp. 45-77, 2007.

[141] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An Empirical Comparison of Botnet Detection Methods," *Computers & Security*, vol. 45, pp. 100-123, 2014.

[142] C. Rossow, C. J. Dietrich, C. Grier, C. Kreibich, V. Paxson, N. Pohlmann, *et al.*, "Prudent Practices for Designing Malware Experiments: Status Quo and outlook," in *2012 IEEE Symposium on Security and Privacy*, 2012, pp. 65-79.

[143] M. Tavallaee, N. Stakhanova, and A. A. Ghorbani, "Toward Credible Evaluation of Anomaly-based Intrusion Detection Methods," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, pp. 516-524, 2010.

[144] A. Papadogiannakis, D. Antoniades, M. Polychronakis, and E. P. Markatos, "Improving the performance of passive network monitoring applications using locality buffering," in *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2007. MASCOTS'07. 15th International Symposium on*, 2007, pp. 151-157.

[145] F. Schneider and J. Wallerich, "Performance evaluation of packet capturing systems for high-speed networks," in *Proceedings of the 2005 ACM conference on Emerging network experiment and technology*, 2005, pp. 284-285.

[146] J. Corsini, "Analysis and Evaluation of Network Intrusion Detection Methods to Uncover Data Theft," Napier University, 2009.

[147] A. Turner and M. Bing,"TcpReplay," 2011, [Online; accessed 9-Dec-2012]. [Online]. Available: https://sourceforge.net/projects/tcpreplay/

[148] A. Folkerts, G. Portokalidis, and H. Bos, "Multi-tier Intrusion Detection by Means of Replayable Virtual Machines," Technical Report IR-CS-47, VU University2008

[149] A. Yeow,"Bit-Twist: Libpcap-based Ethernet Packet Generator," 2016, [Online; accessed 19-Jan-2016]. [Online]. Available: http://bittwist.sourceforge.net/

[150] S. Forge,"TOMAHAWK," [Online; accessed 10-December-2016]. [Online]. Available: http://tomahawk.sourceforge.net

[151] S. C. Smith, K. W. Wong, I. Hammell, J. Robert, and C. J. Mateo, "An Experimental Exploration of the Impact of Network-level Packet Loss on Network Intrusion Detection," DTIC Document, 2015

[152] J. W. Haines, R. P. Lippmann, D. J. Fried, M. Zissman, and E. Tran, "1999 DARPA Intrusion Detection Evaluation: Design and Procedures," 2001.

[153] N. Nwanze, S.-i. Kim, and D. H. Summerville, "Payload Modeling for Network Intrusion Detection Systems," in *MILCOM 2009-2009 IEEE Military Communications Conference*, 2009, pp. 1-7.

[154] C. Thomas, V. Sharma, and N. Balakrishnan, "Usefulness of DARPA Dataset for Intrusion Detection System Evaluation," in *SPIE Defense and Security Symposium*, 2008, pp. 69730G-69730G-8.

[155] H. Om and A. Kundu, "A Hybrid System for Reducing the False Alarm Rate of Anomaly Intrusion Detection System," in *1st International Conference on Recent Advances in Information Technology (RAIT)*, 2012, pp. 131-136.

[156] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward Developing a Systematic Approach to Generate Benchmark Datasets for Intrusion Detection," *Computers & Security*, vol. 31, pp. 357-374, 2012.

[157] J. O. Nehinbe, "A Simple Method for Improving Intrusion Detections in Corporate Networks," in *International Conference on Information Security and Digital Forensics*, 2009, pp. 111-122.

[158] S. Tricaud,"French Honeynet Chapter Status Report," 2011, [Online; accessed 20-May-2013]. [Online]. Available: http://www.honeynet.org/chapters/france

[159] G. Szabó, D. Orincsay, S. Malomsoky, and I. Szabó, "On the Validation of Traffic Classification Algorithms," in *International Conference on Passive and Active Network Measurement*, 2008, pp. 72-81.

[160] Lawrence Berkeley National Laboratory, "Enterprise Tracing Project," 2005, [Online; accessed 8-July-2014]. [Online]. Available: http://www.icir.org/enterprise-tracing/

[161] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips, "The Bittorrent P2P File-Sharing System: Measurements and Analysis," in *International Workshop on Peer-to-Peer Systems*, 2005, pp. 205-216.

[162] O. E. Elejla, A. B. Jantan, and A. A. Ahmed, "Three Layers Approach For Network Scanning Detection," *Journal of Theoretical & Applied Information Technology*, vol. 70, 2014.

[163] G. Kumar, "Evaluation metrics for intrusion detection systems-a study," *International Journal of Computer Science and Mobile Applications, II*, vol. 11, 2014.

[164] D. Smallwood and A. Vance, "Intrusion Analysis with Deep Packet Inspection: Increasing Efficiency of Packet Based Investigations," in *Cloud and Service Computing (CSC), 2011 International Conference on*, 2011, pp. 342-347.

[165] A. Bremler-Barr, Y. Harchol, D. Hay, and Y. Koral, "Deep Packet Inspection As a Service," in *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, 2014, pp. 271-282.

[166] M. M. Masud, T. Al-khateeb, L. Khan, B. Thuraisingham, and K. W. Hamlen, "Flow-based Identification of Botnet Traffic by Mining Multiple Log Files," in *First International Conference on Distributed Framework and Applications*, 2008, pp. 200-206.

[167] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software: An Update," *ACM SIGKDD Explorations Newsletter*, vol. 11, pp. 10-18, 2009.

[168] R. A. Rodríguez-Gómez, G. Maciá-Fernández, and P. García-Teodoro, "Survey and Taxonomy of Botnet Research Through Life-cycle," *ACM Computing Surveys (CSUR)*, vol. 45, p. 45, 2013.

[169] X. Ma, X. Guan, J. Tao, Q. Zheng, Y. Guo, L. Liu, *et al.*, "A Novel IRC Botnet Detection Method Based on Packet Size Sequence," in *IEEE International Conference on Communications (ICC)*, 2010, pp. 1-5.

[170] S. Garg, A. K. Sarje, and S. K. Peddoju, "Improved Detection of P2P Botnets Through Network Behavior Analysis," in *International Conference on Security in Computer Networks and Distributed Systems*, 2014, pp. 334-345.

[171] H. R. Zeidanloo and A. B. A. Manaf, "Botnet Detection by Monitoring Similar Communication Patterns," 2010. [Online]. Available: http://arxiv.org/abs/1004.1232

[172] G. Stringhini, T. Holz, B. Stone-Gross, C. Kruegel, and G. Vigna, "BOTMAGNIFIER: Locating Spambots on the Internet," in *USENIX Security Symposium*, 2011, pp. 1-32.

[173] G. Vliek, "Detecting Spam Machines, A Netflow-data Based Approach," Master thesis, Faculty of Electrical Engineering, University of Twente, 2009.

[174] Y. Li, D. Gruenbacher, and C. Scoglio, "Reward Only Is Not Enough: Evaluating and Improving the Fairness Policy of the P2P File Sharing Network eMule/eDonkey," *Peer-to-Peer Networking and Applications*, vol. 5, pp. 40-57, 2012.

[175] D. Garant and W. Lu, "Mining Botnet Behaviors on the Large-Scale Web Application Community," in *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*, 2013, pp. 185-190.

[176] E. B. Beigi, H. H. Jazi, N. Stakhanova, and A. A. Ghorbani, "Towards Effective Feature Selection in Machine Learning-based Botnet Detection Approaches," in *IEEE Conference on Communications and Network Security (CNS)*, 2014, pp. 247-255.

[177] W. T. Strayer, D. Lapsely, R. Walsh, and C. Livadas, "Botnet Detection based on Network Behavior," *Botnet Detection*, pp. 1-24, 2008.

[178] A. I. Madbouly, A. M. Gody, and T. M. Barakat, "Relevant Feature Selection Model Using Data Mining for Intrusion Detection System," *International Journal of Engineering Trends and Technology (IJETT)*, 2014.

[179] P. Sangkatsanee, N. Wattanapongsakorn, and C. Charnsripinyo, "Practical Real-time Intrusion Detection Using Machine Learning Approaches," *Computer Communications*, vol. 34, pp. 2227-2235, 2011.

[180] P. Narang, J. M. Reddy, and C. Hota, "Feature Selection for Detection of Peer-to-Peer Botnet Traffic," in *Proceedings of the 6th ACM India Computing Convention*, 2013, p. 16.

[181] J. V. Gomes, P. R. Inácio, M. Pereira, M. M. Freire, and P. P. Monteiro, "Detection and Classification of Peer-to-peer Traffic: A survey," *ACM Computing Surveys*, vol. 45, p. 30, 2013.

[182] F. Giroire, J. Chandrashekar, N. Taft, E. Schooler, and D. Papagiannaki, "Exploiting Temporal Persistence to Detect Covert Botnet Channels," in *International Workshop on Recent Advances in Intrusion Detection*, 2009, pp. 326-345.

[183] A. Sperotto, G. Vliek, R. Sadre, and A. Pras, "Detecting Spam at the Network Level," in *Meeting of the European Network of Universities and Companies in Information and Communication Engineering*, 2009, pp. 208-216.

[184] H. Weststrate, "Botnet Detection using Netflow Information," in *10th Twente Student Conference on IT, 23rd January*, 2009.

[185] Y. Liu, "Data Streaming Algorithms for Rapid Cyber Attack Detection," Ph.D. thesis, Department of Computer Engineering, Iowa State University, Ames, Iowa, 2013.

[186] H. Ma, S. Tan, and Z. He, "The Research of P2P Recognition Technology," in *Software Engineering and Service Science (ICSESS), 2014 5th IEEE International Conference on*, 2014, pp. 601-604.

[187] R. Keralapura, A. Nucci, and C.-N. Chuah, "A Novel Self-learning Architecture for P2P Traffic Classification in High Speed Ntworks," *Computer Networks*, vol. 54, pp. 1055-1068, 2010.

[188] M. Agnihotri,"DeepEnd Research: Library of Malware Traffic Patterns," 2013, [Online; accessed 9-May-2014]. [Online]. Available: http://www.deependresearch.org/2013/04/library-of-malware-traffic-patterns.html

[189] S. Stover, D. Dittrich, J. Hernandez, and S. Dietrich, "Analysis of the Storm and Nugache Trojans: P2P is here," *USENIX Login*, vol. 32, pp. 18-27, 2007.

[190] Bro IDS, "Signature framework — Bro 2.4.1 documentation," 2012, [Online; accessed 6-Nov-2013]. [Online]. Available: https://www.bro.org/sphinx/frameworks/signatures.html

[191] J. Amann, S. Hall, and R. Sommer, "Count Me In: Viable Distributed Summary Statistics for Securing High-Speed Networks," in *International Workshop on Recent Advances in Intrusion Detection*, 2014, pp. 320-340.

[192] M. Jonkman,"Emerging Bro Threats," 2008, [Online; accessed 30-June-2012]. [Online]. Available: http://doc.emergingthreats.net/bin/view/Main/EmergingBro

[193] M. Jonkman,"Storm Worm Emerging Threats," 2007, [Online; accessed 4-April-2013]. [Online]. Available: http://doc.emergingthreats.net/bin/view/Main/StormWorm

[194] M. Tavallaee, "An Adaptive Hybrid Intrusion Detection System," Ph.D. thesis, University of New Brunswick, 2011.

[195] G. Maier, R. Sommer, H. Dreger, A. Feldmann, V. Paxson, and F. Schneider, "Enriching Network Security Analysis with Time Travel," in *ACM SIGCOMM Computer Communication Review*, 2008, pp. 183-194.

[196] B. AsSadhan, J. M. Moura, D. Lapsley, C. Jones, and W. T. Strayer, "Detecting Botnets using Command and Control Traffic," in *Eighth IEEE International Symposium on Network Computing and Applications* 2009, pp. 156-162.

[197] PacketFilter, "Packet Filter in Bro," 2013, [Online; accessed 20-June-2013]. [Online]. Available: https://www.bro.org/sphinx/scripts/base/frameworks/packet-filter/main.bro.html

[198] G. Carle, F. Dressler, R. A. Kemmerer, H. Koenig, C. Kruegel, and P. Laskov, "Network attack detection and defense–Manifesto of the Dagstuhl Perspective Workshop, March 2nd–6th, 2008," *Computer Science-Research and Development*, vol. 23, pp. 15-25, 2009.

[199] G. Münz, N. Weber, and G. Carle, "Signature Detection in Sampled Packets," in *Workshop on Monitoring, Attack Detection and Mitigation (MonAM 2007), Toulouse, France*, 2007.
[200] R. Sommer,"Bro Cluster Architecture — Bro 2.4.1 Documentation," 2013, [Online; accessed 24-Jan-2015]. [Online]. Available: https://www.bro.org/sphinx/cluster/index.html
[201] E. Alparslan, A. Karahoca, and D. Karahoca, "BotNet Detection: Enhancing Analysis by Using Data Mining Techniques," *INTECH Open Access Publisher*, 2012.
[202] Bro IDS, "Policy Stats," 2008, [Online; accessed 7-Dec-2013]. [Online]. Available: https://www.bro.org/sphinx/scripts/policy/misc/stats.bro.html
[203] R. Love, "Kernel Korner: CPU Affinity," *Linux Journal*, vol. 2003, p. 8, 2003.
[204] Open BL, "Abuse Reporting and Blacklisting," 2014, [Online; accessed 4-July-2014]. [Online]. Available: https://www.openbl.org
[205] Black List, "URL Blacklist," 2013, [Online; accessed 2-May-2015]. [Online]. Available: http://urlblacklist.com/
[206] S. Hansman and R. Hunt, "A Taxonomy of Network and Computer Attacks," *Computers & Security*, vol. 24, pp. 31-43, 2005.
[207] K. Labib, "Computer Security and Intrusion Detection," *Crossroads*, vol. 11, pp. 2-2, 2004.
[208] Y. Gao, Z. Li, and Y. Chen, "A DoS Resilient Flow-level Intrusion Detection Approach for High-speed Networks," in *26th IEEE International Conference on Distributed Computing Systems (ICDCS'06)*, 2006, pp. 39-39.
[209] T. Diibendorfer and B. Plattner, "Host Behaviour based Early Detection of Worm Outbreaks in Internet Backbones," in *14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE'05)*, 2005, pp. 166-171.
[210] A. Sperotto, R. Sadre, P.-T. de Boer, and A. Pras, "Hidden Markov Model modeling of SSH Brute-force Attacks," in *International Workshop on Distributed Systems: Operations and Management*, 2009, pp. 164-176.

Appendix A
Attack Classification

In the literature, many attack classifications and taxonomies have been presented and surveyed. However, not all the taxonomies that outlined in the literature provide the same classification. Some studies classify the attack based on their goals, results, and tools [28] and others classify the attacks based on the network type [206]. The highest priority attacks are those who have a critical impact on the computer system. In this appendix, the following major types of attacks are described.

A.1 Denial of Service (DoS)

The main objective of DoS attacks is to deny a legitimate user from using or accessing his/her system in a normal mode. It often disturbs the service of a computer, a server, or a network. Thus it is impossible to use its resources. This kind of attack is frequent on the Internet. There are three types of Dos attacks: host based, network based, and distributed based.

Host-based DoS attacks: This attack targets a vulnerability in the operating system, application software, CPU, and memory. The main aim of this attack is to crash the host. It also works by exploiting the implementation of network protocols

Network based DoS attacks: Network resources are targeted in this attack by flooding the network with packets to disrupt legitimate use. In this case, the bandwidth is overwhelmed with packets so that there is no left bandwidth for the legitimate users.

TCP floods, ICMP floods, and UDP floods are the most network based DoS attack identified that stream their packets to the target.

Distributed based DoS or DDoS attacks: This attack use a large number of attacked computers to direct coordinated DoS attack against target or targets.

A.2 Information Gathering and Scanning

These attacks try to gather information about the system for further attacks. No actual attack is launched on the computer and the network; they are, however, sniffed, scanned, and probed. A packet sniffer is a simple tool to gather information about computer and network by listening to every packet at a particular point in a network. In conventional packet sniffer, the attacker set the Ethernet card into promiscuous mode so that the card accepts and read all traffic packets in the network, even when a packet is not addressed to this network card. MAC address, IP addresses, and running services for a particular host can be obtained using sniffer tools.

A.3 Malicious Software

Malware includes Worms, Virus and Trojan horse, are malicious programs that are inserted into a host to corrupt a system, deny access to a service. The worm runs random code on the victim’s host and installs copies of itself in the memory, which infects other hosts on the network. It leads to network congestion, delay, and loss packets. A virus is a program that is attached to another program to run a particular harmful function on the victim’s computer. The virus needs the user interaction to run it and propagate to other files or hosts. However, the spread of worms is extremely faster than the virus. A Trajan horse is a program looks like a useful application, but in fact, performs unwanted actions such as controlling the victim's host remotely using backdoor installation.

A.4 IP Spoofing

This kind of attack is functioning on networks and TPC/IP protocols. Network spoofing is used when the attacker pretends himself as a legitimate user by spoofing who they are. Session Hijacking is the most popular attack in this kind of attack. The attacker usually takes over a session between two hosts and then cuts one of these hosts to be replaced by him. Session Hijacking usually operates at TCP layer and is used to take over sessions of services such as FTP and Telnet. TCP session hijacking also takes advantage of using IP spoofing and TCP sequence number. To make this attack easy to the attacker, the attacker has to guess the TCP sequence number of the session that is attempted to be hijacked by capturing and analysing the packets travelling between the two victims. After the attackers manage to get the sequence number, they spoof their IP address to be matched with one of the victim hosts and then send a TCP packet to the other host with the hijacked sequence number. When the other host accepts the packet and verifies the sequence number which is correct, this host starts to reply to the attacker and continue the hijacked session.

Other types of attacks may include:

- **Physical attacks:** The aim of this attack is to damage the computer hardware and network devices**.**

- **Buffer overflows:** This attack overflow the process's buffer of the victim's system to damage the process.

- **Password attacks:** This attack involves when the attacker is attempting to guess a password of a protected host. Password dictionary and brute force are the main example of this attack.

- **Botnet attack:** This attack was discussed in Chapter 4.

The following steps explain the nature and the methodology of the computer attacks [207]:

1. Reconnaissance: This step involves the process when the attacker collects information about its victim, including the network infrastructure, before launching its attack.

2. Scanning: In this stage, the attacker starts to look for vulnerabilities and holes by scanning the victim's system. Towards the end, the attacker can obtain precious information such as network topology, IP addresses of live hosts, open port numbers, and security devices rules.

3. Getting Access: This step takes place when the attacker attempt to gain access either using the operating system and application attacks if the attacker is a legitimate user, or using the network if the attacker is an outsider.

4. Retaining Access: After the attacker gained access to the compromised host, he/she has to maintain this access. Trajan horse and Backdoors are the famous techniques to perform this step.

5. Hiding Imprint: When the attackers have achieved what they want, they should not leave any track on the system. Backdoor and RootKit are among techniques that help the attacker to modify system logs and build hidden channel for data transmission.

Appendix B
NIDS Requirements

There are many requirements for efficient NIDS mentioned in the literature [6]. The main two requirements that attracted researchers currently are scalability and detection accuracy.

- **Scalability:** NIDS should operate in large volume networks without resource consumption. This happens when all potential packets and traffic are analysed without packet loss. Thus, detection analysis should be performed smoothly in a large data network as well as with increase traffic and network's size. Also, the data amount to be processed by detection methods should be as small as possible. Note that the term "potential packet" is used instead of "incoming packet", this is because potential packets are extracted after sampling processes as will be discussed later.

- **Detecting accuracy or detection rate**: beside all potential packets should be processed correctly; detection methods have to make the right decision, not to decide falsely. To achieve this requirement, the true-positive rate should be high while fewer false positive and negative rate.

Other requirements of NIDS may include:

- **Detecting unknown attacks:** novel intrusion should be detected

- **Detecting encrypted traffic:** encrypted payloads should be readable and analysed for intrusion detection.

- **Early detection:** intrusion should be detected as soon as possible

- **Large data storage:** all potential signatures, profiles, alerts, and reports should be stored for long-term and further usage.

- **NIDS security:** NIDS should be secured enough against attackers who direct attacks into the NIDS itself.

- **Events correlation:** For distributed attacks, NIDS should correlate single attack event with other resources such as firewall, routers or other NIDS for detection.

- **IPv6 compatibility:** NIDS should support IPv4 and IPv6

- **Success attacks identification:** NIDS should differentiate between successful and unsuccessful attack so that the operator should take a proper action against them.

- **Privacy:** NIDS should not violate privacy regulation of users by inspecting private information both in payload and header of the packets.

- **Attack classification:** After detection, NIDS should also identify and classify attacks. Each attack has to be labelled and be under a category for further analysis and measurements.

Appendix C
Attacks Detectable by Flow-based Approach

This appendix presents the attacks that are detectable by flow-based NIDS and how the current research community handles its limitation.

DoS Attack

Gao, et al. [208] proposed and implemented a DoS resilient High-speed Flow-level Intrusion Detection system, HiFIND. The authors developed a prototype that accepts flows exported from a Netflow router in real time. Their approach handles the problem of DoS using flow aggregation accounted in data stream called a sketch. A sketch is a hash table in one-dimension appropriated for quick storage of information. Sketch counts incidences of an event and studies how the traffic behaves over a period of time using statistics. It stores values that help an anomaly-based engine to trigger alarms based on a statistical forecast. So an abnormal deviation from this forecast values is detected as an intrusion. SYN flooding attack is one of DoS attacks that can be used by sketch to detect this type of attack with the following steps:

- The sketch stores and calculates the difference between the number of SYN packets and the number of SYN/ACK packets of each flow.

- If this difference is not within the normal range, a DoS SYN flooding attack is detected.

This approach can be implemented with relying on packet headers only instead of flows but, however; data reduction which is provided by flows cannot be achieved.

Zhao, et al. [53] proposed and designed data streaming algorithms that can detect super sources and super destinations attacks. Super source happens when a source or a host has a unusual number of outgoing connection (fan-out) within specified period. An example of the super source is port scanning that searches for vulnerable services among different hosts. Super destination is considered when a destination or a host receive abnormal number of incoming connection attempts within a small time interval (fan-in). Distributed Denial of Service (DDoS) attack is an example of super destination when a large number of hosts flood flows to a single destination. Data streaming algorithms used in their work is to identify flows that have an unusual number of connection after filtering part of the traffic. Unlike [208], the algorithms of used in [53] is based on two dimension hash tables. To reduce the amount of data to be processed, they perform flow sampling algorithm, hence improving the speed of the process. Since not all the flows are processed, data reduction may compromise the accuracy. The authors solve this problem by combining the power of data streaming and sampling.

Kim, et al. [40] presented a detecting method for detecting abnormal network traffic by analysing the traffic based on flows only. They use the term “traffic pattern” to express different types of DoS attacks. A traffic pattern is a signature that describes the number of flows, number of packets per flow, the size of flow, the size of packets, and the total bandwidth occupied during the session. The authors use these patterns to differentiate between instances when detecting scanning or flooding attacks. For example, during scanning or SYN flooding attack, since the attacker makes many connection attempts, this pattern can be detected because of:

- a large number of flows generated since the attacker sends many packets to the victim,
- a small number of packets per flow,
- moreover, the small size of the packet as the attacker sends small SYN packets.

The authors also managed to detect ICMP and UDP flooding attack. These attacks have dynamic traffic patterns since it depends on the number of packets and hosts used in these attacks. However, these attacks can be detected since they create large bandwidth consumption and a high number of packets. Their approach can detect traffic of different attacks with a similar traffic pattern by identifying their metrics and then formalizing them into one detection function. However, certain attacks cannot be observed using their method since Kim, et al. focused on detecting DoS and DDoS attacks only. Since they used static threshold values of their parameters in the detection function, their method cannot be suitable for every network condition. So, the adaptive threshold for various network environments is required.

Munz and Carle [57] proposed a general system for DoS flow-based detection named "TOPAS" (Traffic flOw Packet Analysis System). This system operates as a flow collector from multiple sources. It receives data to be analysed in real-time. The authors develop TOPAS so that it supports different kinds of DoS detection modules and it is publicly available. These modules are including SYN flood detection, Web Server overloading module using HTTP request, and traceback module that identifies the entry points of attack packet with spoofed source IP address. These modules can be adjusted by the network administrator to increase the detection opportunities and accuracy. An example of this is adjusting the number of SYN and SYN/ACK packets in case of SYN flooding detection module. Although the authors state that TOPAS can also analyse packet-base data, their approach does not support the combination of packet-based and flow-based to reduce the false alarms.

Worms

Worm mechanism such as Code Red usually has two stages: victim discovery and transfer code. In discovery stage, the worm surveys the network to find vulnerable holes in the systems while in transfer stage, the worm starts to spread the code to the systems. Unfortunately, the second stage cannot be detected using the flow-based system since the code is injected in the payload which is not analysed by the flow-based. Thus only the first stage of worm behaviour can be analysed and detected using flow-based approach. Some attributes on the hosts when worms infect them are used to detect worms attack. Such attributes include the number of connections, ratio of outgoing to incoming traffic, and response way. However, some researchers deal with worm detection the same way when dealing with scanning detection since they have some common characteristics. DoS detection methods achieved by [53, 208] can be used to detect the worm.

Diibendorfer and Plattner [209] proposed a near real-time method for outbreak worm detection in high-speed networks using flow-based approach. The method is based on examination the behaviour and the number of incomings and outgoing connection of the host. For detection method, the authors used the host behaviour and characteristics to classify hosts into three classes: traffic class, connector class, and responder class. Only suspicious hosts belong to these classes.

Hosts are classified as traffic class when the amount of traffic sent from the host is more than received. An example of this is the worms send out exploit code or when the worm spread in email attachments. Hosts that initiate an abnormal high number of outgoing connections are classified under connector class. Such class happens when hosts scan others. Responder class involves when a host holds bidirectional connections such as TCP connection. An example of this class is when the host responds to TCP handshake initiation or scan during a worm outbreak. In their approach, overlapping within these classes is possible, meaning that a host can be belonging to more than one class.

Figure C.1 illustrate this overlap. Worm outbreak attack can be detected by tracking the cardinality of each class of an entire network periodically. Thus, any unexpected or sudden changes in the cardinality of one or more classes are detected as worm outbreak. The authors validate their method by tracing archived flow-level of recent Internet emails and by tracing fast spreading worms such as Blaster.

Abdulla, et al. [85] proposed a worm warning system using IP flow and machine learning approach. The authors consider the case that when a host is infected by an email worm or scanning, an unusual amount of traffic is initiated. This traffic is not relied on DNS. They classify flow-based records using Support Vector Machine (SVM) to extract features that belong to worm attacks. For training SVM, the features are gathered into a set of patterns. The authors propose a structure that consists of three modules: data collecting, data sampling, and classifier.

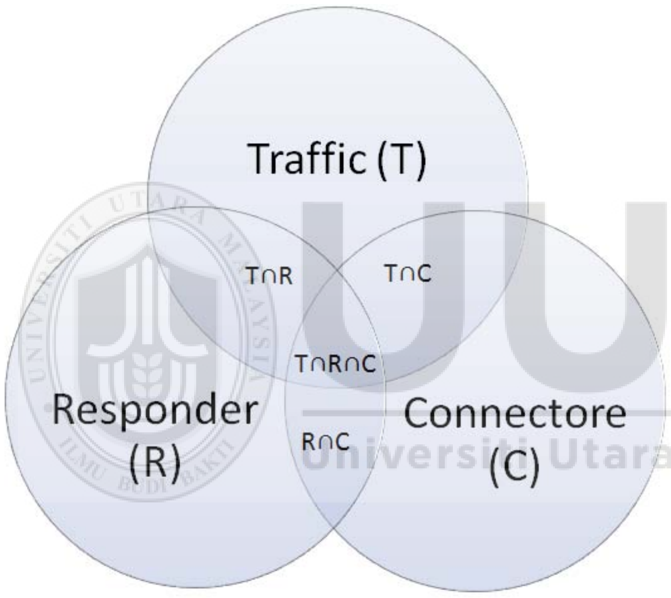


Figure C.1. Classes of Host Behaviour for Worm Detection

The first module collects the raw traffic and extracts the flow record information and stores them into a database. The authors address the problem of dealing with a large amount of flow data by creating the data sampling module. The classifier module classifies the sampled traffic into a worm and benign flow. The SVM was trained by the following scanning worms: CodeRed, Slammer, Doomjuice, and Witty. For email worms, it was trained by sobig, Netsky, Storm, MyDoom, and Conficker.

SSH

Secure SHell (SSH) is a communication protocol that allows a user to have full control over a host's resources remotely. Thus, hosts with SSH-enabled are unfortunately targeted by intrusions. Sperotto, et al. [210] have studied and analysed the flow traffic during SSH. They extract the flow data that is suspected to be malicious traffic. The authors then develop a model which presents the flow characteristics when SSH intrusion takes place. Although their model can detect these attacks, however, the possibility of this model to be in practice is still unknown. Based on their work, Hellemons (2012) develop an algorithm to test the practical applicability of the SSH intrusion model. The algorithm uses the processed flow data to construct attack metadata in the form of properties. Hellemons answered the question: "Can SSH intrusion attacks be detected and analysed in practice by using only flow data?" affirmatively. This method reduces the need for deep packet inspection system, allowing for more scalable NIDS solution.

Appendix D
Main Bro Log Files

D.1 *Connection.log*

Bro generates this log during run time. It consists of the complete connection log of incoming and outgoing traffic. Table D.1 shows the fields of the *connection.log* file.

Table D.1

Fields Description of Connection.log file

Field	Type	Description
ts	time	Timestamp
uid	string	Unique ID of Connection
id.orig_h	addr	Originating endpoint’s IP address (AKA ORIG)
id.orig_p	port	Originating endpoint’s TCP/UDP port (or ICMP code)
id.resp_h	addr	Responding endpoint’s IP address (AKA RESP)
id.resp_p	port	Responding endpoint’s TCP/UDP port (or ICMP code)
proto	transport _proto	Transport layer protocol of connection
service	string	Dynamically detected application protocol, if any
duration	interval	Time of last packet seen – time of first packet seen
orig_bytes	count	Originator payload bytes; from sequence numbers if TCP
resp_bytes	count	Responder payload bytes; from sequence numbers if TCP
conn_state	string	Connection state
local_orig	bool	If conn originated locally T; if remotely F. If Site::local_nets empty, always unset.
missed_bytes	count	Number of missing bytes in content gaps
history	string	Connection state history
orig_pkts	count	Number of ORIG packets
orig_ip_bytes	count	Number of ORIG IP bytes
resp_pkts	count	Number of RESP packets
resp_ip_bytes	count	Number of RESP IP bytes (via IP total_length header field)
tunnel_parents	set	If tunneled, connection UID of encapsulating parent (s)
orig_cc	string	ORIG GeoIP Country Code
resp_cc	string	RESP GeoIP Country Code

D.2 *Signatures.log*

This is log is generated when content matching occurs. Bro raises an event with the alert named. This log also contains the payload content which triggers this event. Table D.2 shows each field with its description for this log.

Table D.2

Fields Description of Signatures.log file

Field	Type	Description
ts	time	Timestamp of match
src_addr	addr	Host triggering the signature match event
src_port	port	Host port on which the match occurred
dst_addr	addr	Host which was sent the matching payload
dst_port	port	Port which was sent the matching payload
note	string	Notice associated with the signature event
sig_id	string	Name of the signature that matched
event_msg	string	More descriptive message of the event
sub_msg	string	Extracted payload data or extra message
sig_count	count	Number of sigs
host_count	count	Number of hosts

The following log text is a sample of *Signatuers.log* generated from PH when CTU-52 dataset is used. It shows three infected IRC-bot were detected: `147.32.84.165, 147.32.84.191, and 147.32.84.192`

```
#separator \x09
#set_separator ,
#empty_field    (empty)
#unset_field    -
#path   signatures
#open   2015-08-01-08-13-34
#fields ts      uid     src_addr        src_port        dst_addr        dst_port
        note    sig_id  event_msg       sub_msg sig_count       host_count
#types  time    string  addr    port    addr    port    enum    string  string  string
        count   count
1313675274.978894       CoX6Zn4wnPAUOfTuOk      147.32.84.165   1027    74.125.232.201  80
        Signatures::Sensitive_Signature         ircattack_client        147.32.84.165:
signature match         GET /service/check2?appid=%7B430FD4D0-B729-4F61-AA34-
```

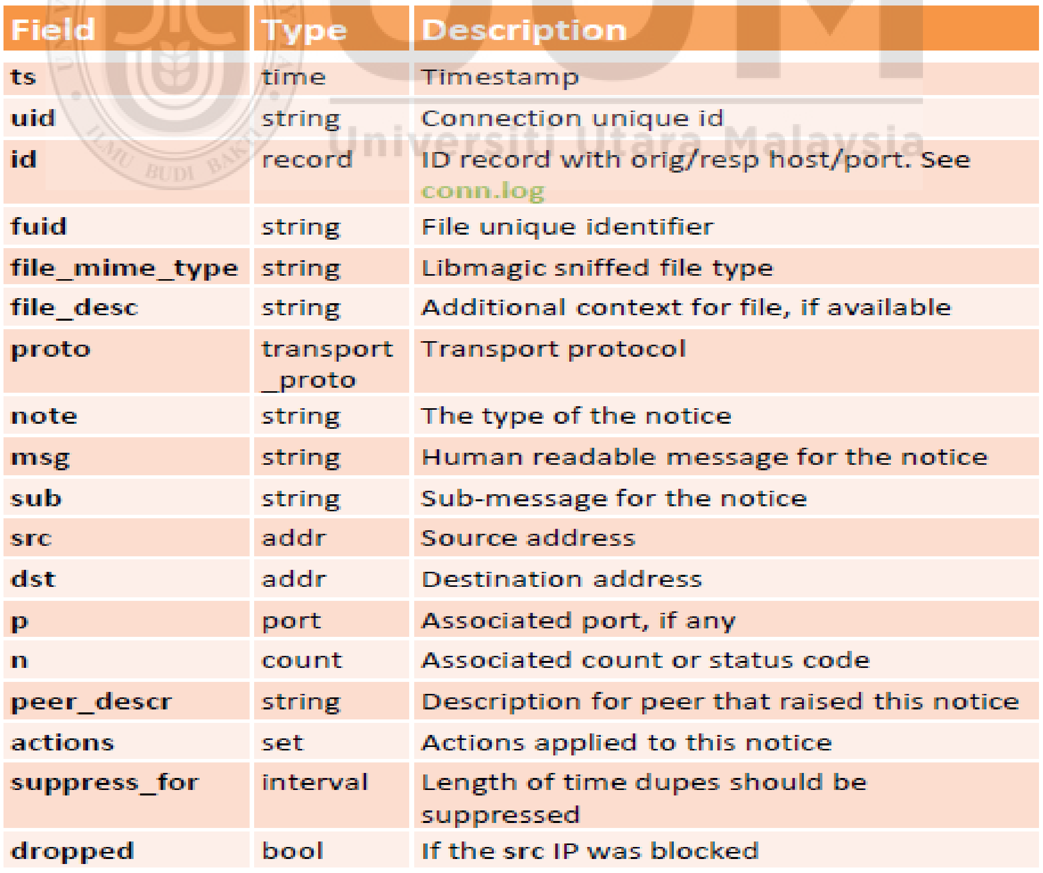
```
91526481799D%7D&appversion=1.3.21.65&applang=&machine=0&version=1.3.21.65&osversion=5
.1...   -       -
1313675281.195719      CxIZuw1HkEATGTlkL6     147.32.84.191  1027    74.125.232.200 80
       Signatures::Sensitive_Signature       ircattack_client      147.32.84.191:
signature match        GET /service/check2?appid=%7B430FD4D0-B729-4F61-AA34-
91526481799D%7D&appversion=1.3.21.65&applang=&machine=0&version=1.3.21.65&osversion=5
.1...   -       -
1313675284.530430      CPfunv1ZCWV1ZnfWBj     147.32.84.192  1027    74.125.232.199 80
       Signatures::Sensitive_Signature       ircattack_client      147.32.84.192:
signature match        GET /service/check2?appid=%7B430FD4D0-B729-4F61-AA34-
91526481799D%7D&appversion=1.3.21.65&applang=&machine=0&version=1.3.21.65&osversion=5
.1...   -       -
#close  2015-08-01-08-13-48
```

D.3 *Notice.log*

Bro also generates this log at runtime. In this log, it contains activities that Bro recognizes as interesting or bad. Table D.3 shows the filed description of this log.

Table D.3

Fields Description of Notice.log file



Field	Type	Description
ts	time	Timestamp
uid	string	Connection unique id
id	record	ID record with orig/resp host/port. See conn.log
fuid	string	File unique identifier
file_mime_type	string	Libmagic sniffed file type
file_desc	string	Additional context for file, if available
proto	transport _proto	Transport protocol
note	string	The type of the notice
msg	string	Human readable message for the notice
sub	string	Sub-message for the notice
src	addr	Source address
dst	addr	Destination address
p	port	Associated port, if any
n	count	Associated count or status code
peer_descr	string	Description for peer that raised this notice
actions	set	Actions applied to this notice
suppress_for	interval	Length of time dupes should be suppressed
dropped	bool	If the src IP was blocked

Appendix E

Resource Consumptions Results

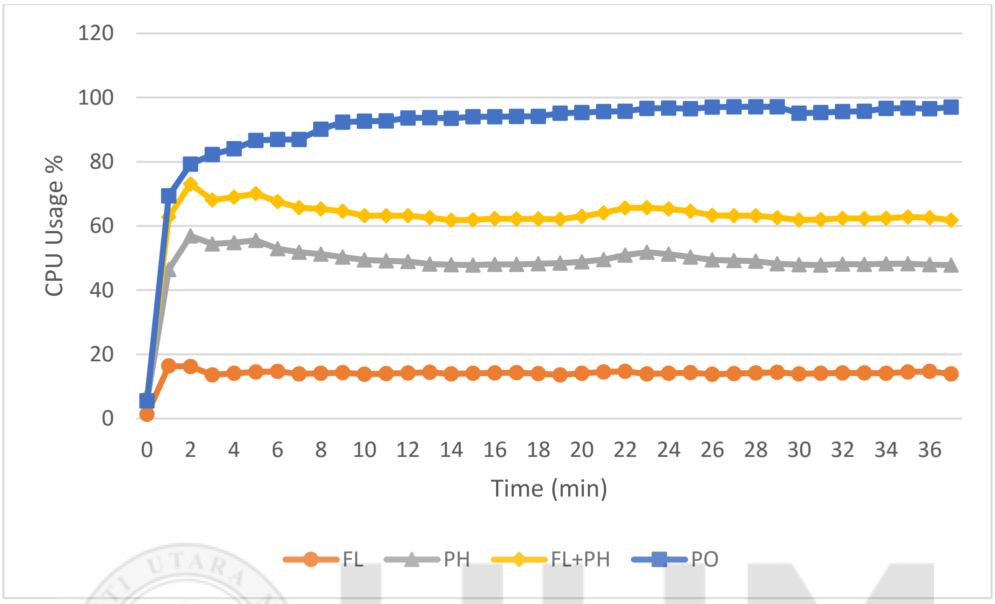


Figure E.1. CPU Usage over Time at 100 Mbps – P2P-bot

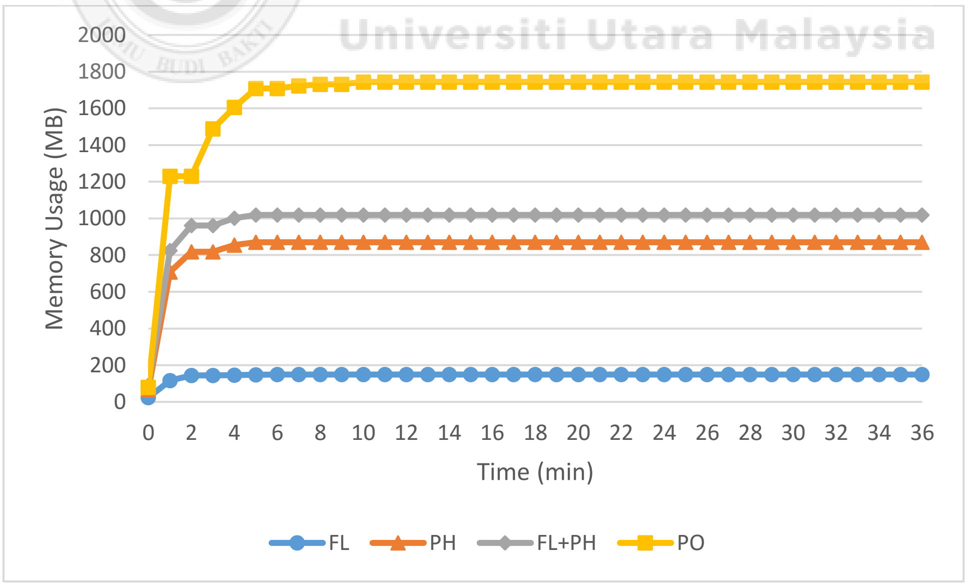


Figure E.2. Memory Usage over Time at 100 Mbps – P2P-bot

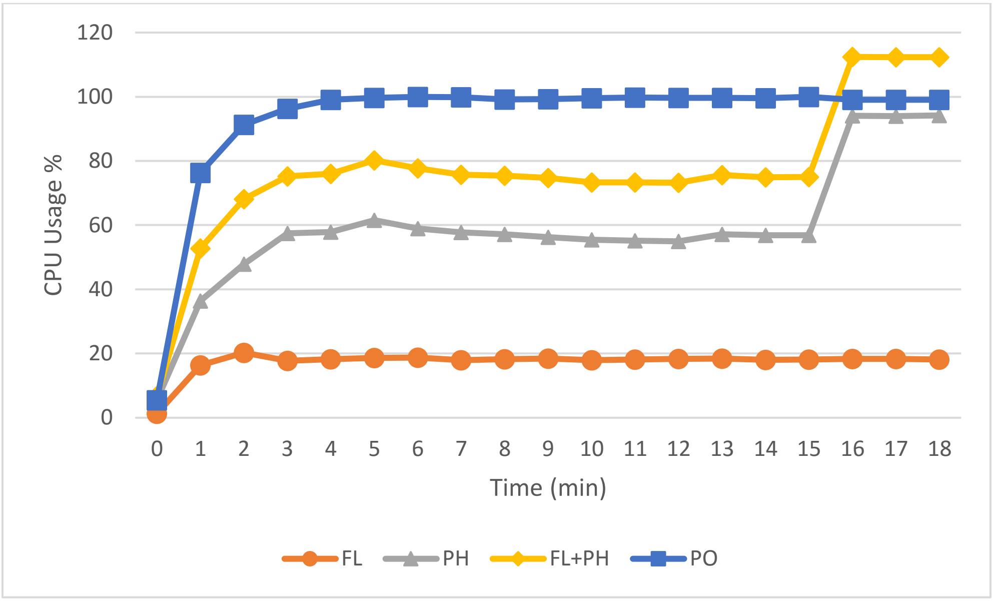


Figure E.3. CPU Usage over Time at 200 Mbps- P2P-bot

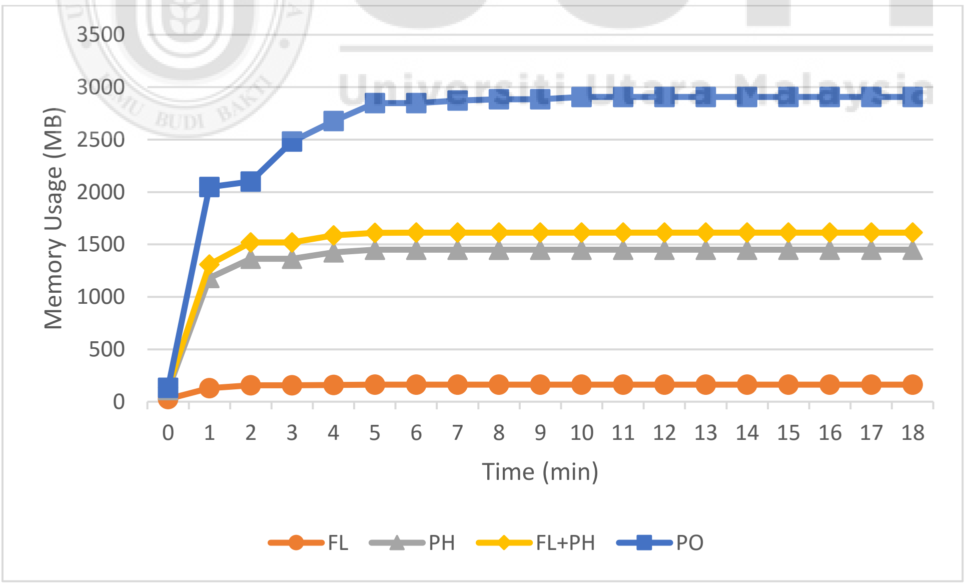


Figure E.4. Memory Usage over Time at 200 Mbps – P2P-bot

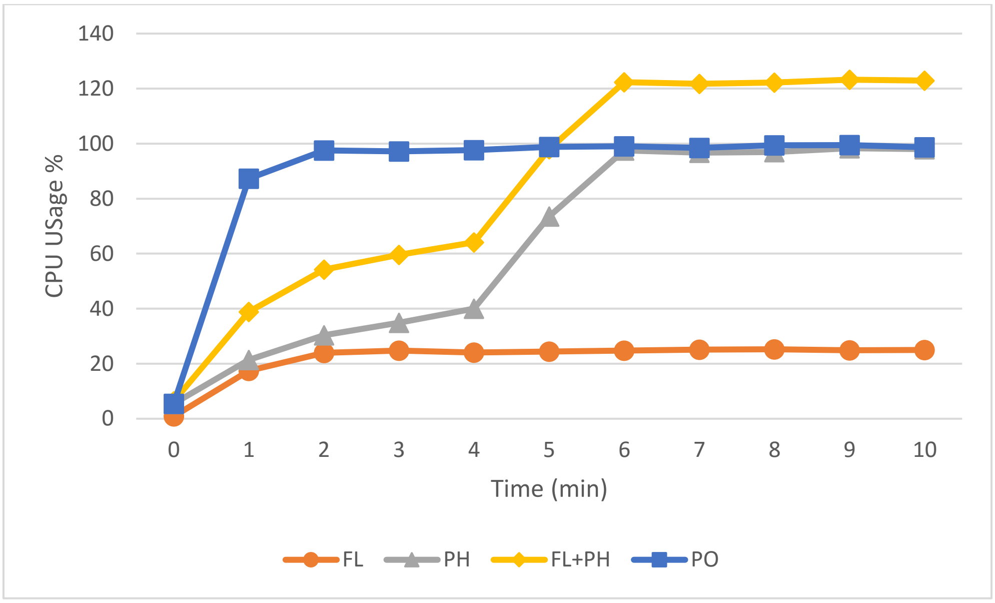


Figure E.5. CPU Usage over Time at 500 Mbps – P2P-bot

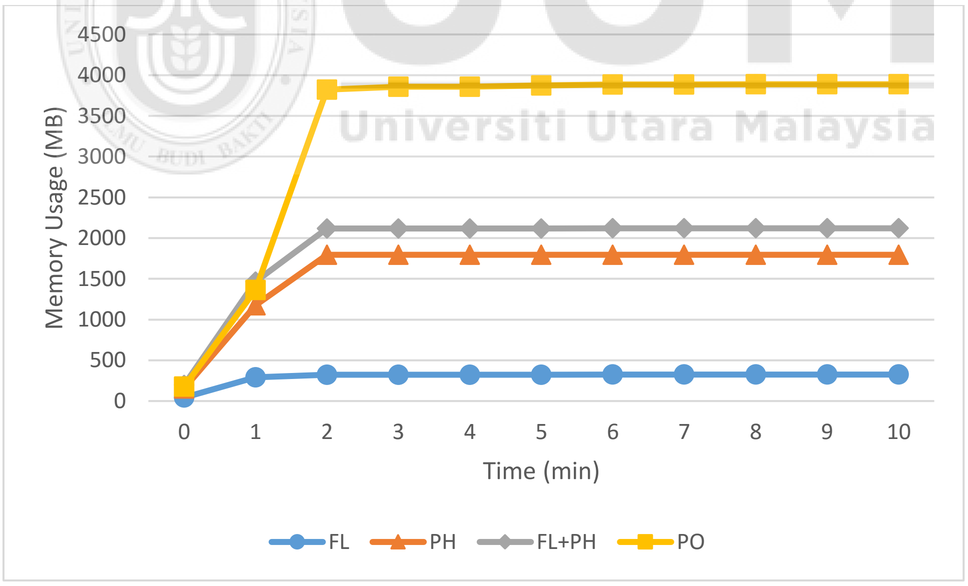


Figure E.6. Memory Usage over Time at 500 Mbps – P2P-bot

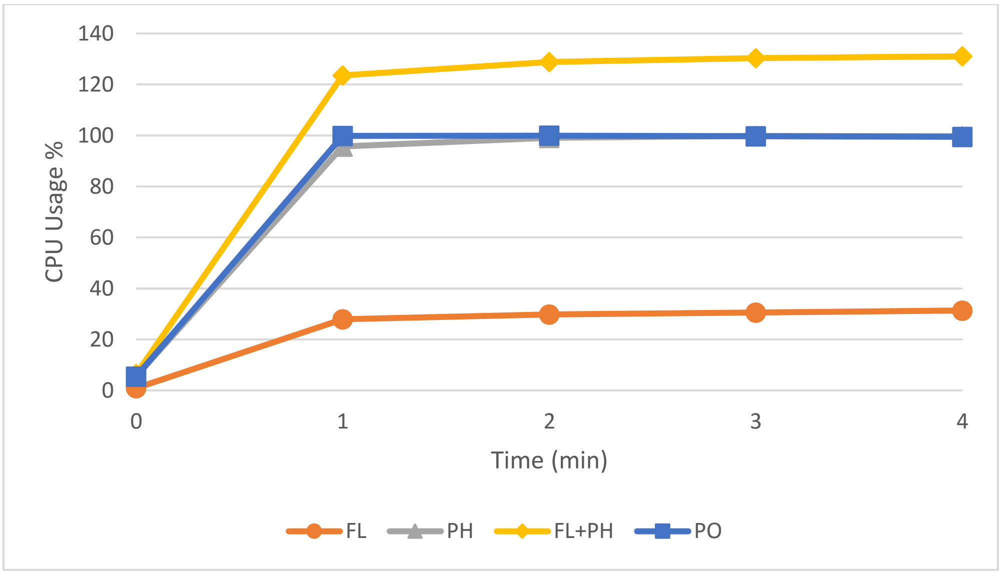


Figure E.7. CPU Usage over Time at 1000 Mbps – P2P-bot

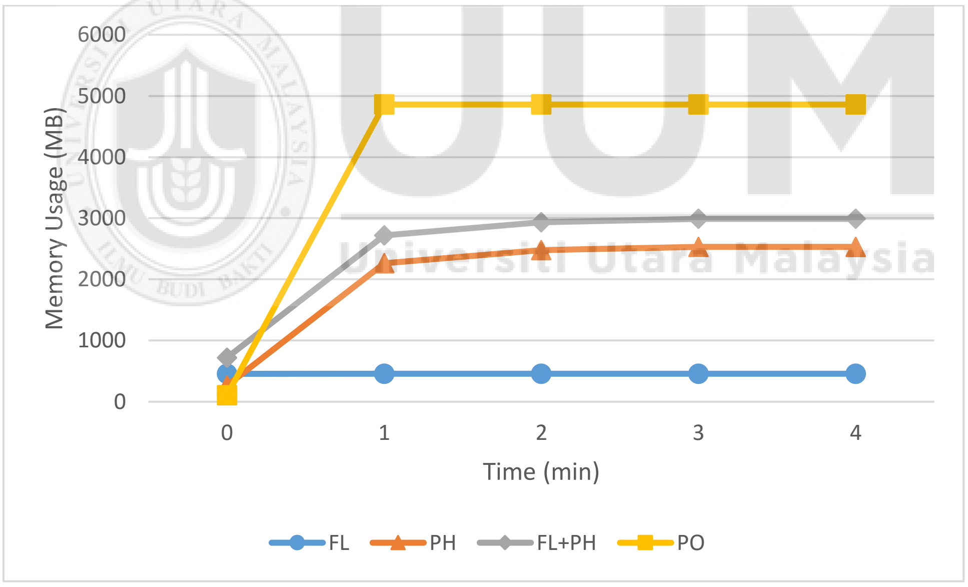


Figure E.8. Memory Usage over Time at 1000 Mbps – P2P-bot

Appendix F

Samples of Detection Code

F.1 Bro SumStats Mechanism Code for Packet-based Spam Identifications

```
@load base/frameworks/sumstats
## Networks that are considered "local":
 const private_address_space: set[subnet] = {
                10.0.0.0/8,
                192.168.0.0/16,
                172.16.0.0/12,
                147.32.84.0/8,
                100.64.0.0/10,
                127.0.0.0/8,
                [fe80::]/10,
                [::1]/128,
 } &redef;
const local_nets: set[subnet] &redef;
global spam_detect = open_log_file("spamhosts") &redef;

event connection_attempt(c: connection)
{
        # Make an observation!
        # This observation is about the host attempting the connection.
        if(c$id$resp_p == 25/tcp) {
                SumStats::observe("SMTP conn",
                          SumStats::Key($host=c$id$orig_h),
                       SumStats::Observation($num=1));
        }
        if(c$id$orig_p == 25/tcp) {
                SumStats::observe("SMTP conn",
                          SumStats::Key($host=c$id$resp_h),
                       SumStats::Observation($num=1));
        }
}

event connection_established(c: connection)
{
# Make an observation!
# Each established connection counts as one so the observation is always 1.
        if(c$id$resp_p == 25/tcp) {
                SumStats::observe("SMTP conn",
                          SumStats::Key($host=c$id$orig_h),
                       SumStats::Observation($num=1));
        }
        if(c$id$orig_p == 25/tcp) {
                SumStats::observe("SMTP conn",
                          SumStats::Key($host=c$id$resp_h),
                       SumStats::Observation($num=1));
        }
}

event bro_done()
{
}

event bro_init()
{
        Log::disable_stream(Conn::LOG);
        # The reducer attaches to the "SMTP conn" observation stream
        # and uses the summing calculation on the observations.
        local r1 = SumStats::Reducer($stream="SMTP conn",
                       $apply=set(SumStats::SUM));
        # Create the final sumstat.
        # $threshold_val.  The actual threshold itself is provided with
        # $threshold.
        # Another callback is provided for when a key crosses the
        # threshold.
        SumStats::create([$name = " Detecting spam activities",
                          $epoch = 10sec,
                          $reducers = set(r1),
                          # Provide a threshold.
                          $threshold = 10.0,
                          # Provide a callback to calculate a value from
                          # the result
                          # to check against the threshold field.
                          $threshold_val(key: SumStats::Key, result:
SumStats::Result) =
                                {
                                return result["SMTP conn"]$sum;
                                },
                          # Provide a callback for when a key crosses
                          # the threshold.
                         $threshold_crossed(key: SumStats::Key, result:
SumStats::Result) =
                          {
                       if (key$host in private_address_space) {
                                print fmt("%s attempted %.0f or more connections",
                                   key$host, result["SMTP conn"]$sum);
                                print spam_detect, fmt(
                                   "%s attempted %.0f or more connections",
                                   key$host, result["SMTP conn"]$sum);
                       }
        }]);
}
```

F.2 Bro PH Code for IRC-bot Detection

```
@load base/frameworks/notice
@load base/frameworks/signatures/main
@load base/protocols/irc
@load policy/misc/stats
@load-sigs ./ircattack.sig
@load base/frameworks/packet-filter

redef capture_filters = { ["filter_table"] = "" };
global print_logs = open_log_file ("print_log") &redef ;
global filter : string = "";

#To read a file into a Bro table, two record types have to be defined:
# This record contains the types and names of the columns that should constitute the
table keys.
#Our key record only contains the host IP
type Idx: record {
        ip: addr;
};

#This record contains the types and names of the columns that should constitute the
table values.
type Val: record {
        comment: string;
};
# Create an empty table that should contain the suspicious data
global suspicious: table[addr] of Val = table();

event update_filter ()
{
local ns = net_stats();
local filter_counter : count = 0;
local pre_filter : string = "host 100.101.102.103";

# 2) convert suspicious table into filter format string
```

```
for ( ip in suspicious )
{
pre_filter += fmt (" or host %s " , ip ) ;
++ filter_counter;
}
print "pre_filter is";
print pre_filter;

# 3) packet filter framework read the filters
if ( pre_filter != filter )
{
print " Filter has beed altered";
print " Perform Recompiling Filter";
captured_filter [filter_table] = pre_filter ;
}
else
{
print " Filter has not beed altered";
}
filter = copy ( pre_filter ) ;
print print_logs , " number of susp hosts marked ; hosts in filter";
print print_logs , fmt (" %s; %s", |suspicious| , filter_counter);
# to update the capture_filter from suspicious, but not to update the suspicious
itself (since Reread is there)
schedule 10 sec { update_filter () };
flush_all () ;
}

event bro_init() &priority = 5
        {
#1) transfer + update flow suspicious ips into suspicious table
Input::add_table([$source="/home/hashem-bro/b-irc/flowirc/suspicious_file.log",
        $name="suspicious", $idx=Idx, $val=Val, $destination=suspicious,
$mode=Input::REREAD]);
        Input::remove("suspicious");
                schedule 5 sec { update_filter () };
set_buf(detailed_log, F);
        set_buf(bot_log, F);
        }

global checkflag = 0;
global ircbotdetect = open_log_file("ircbot_packet_hosts") &redef;
global p_at_in : count = 0;
global p_es_in : count = 0;

module IrcBot;
export {
        global detailed_log = open_log_file("irc.detailed") &redef;
        global bot_log = open_log_file("irc-bots") &redef;
        global summary_interval = 1 min &redef;
        global detailed_logging = T &redef;
        global content_dir = "irc-bots" &redef;
        global bot_nicks =
                /^\[([^\]]+\|)+[0-9]{2,}]/              # [DEU|XP|L|00]
                | /^\[[^ ]+\]([^ ]+\|)+([0-9a-zA-Z-]+)/         # [0]CHN|3436036
[DEU][1]3G-QE
                | /^DCOM[0-9]+$/                        # DCOM7845
                | /^\{[A-Z]+\}-[0-9]+/                  # {XP}-5021040
                | /^\[[0-9]+-[A-Z0-9]+\][a-z]+/                 # [0058-X2]wpbnlgwf
                | /^\[[a-zA-Z0-9]\]-[a-zA-Z0-9]+$/      # [SD]-743056826
                | /^[a-z]+[A-Z]+-[0-9]{5,}$/
                | /^[A-Z]{3}-[0-9]{4}/                  # ITD-1119
                ;
        global bot_cmds =
                /(^| *)[.?#!][^
]{0,5}(scan|ndcass|download|cvar\.|execute|update|dcom|asc|scanall) /
                | /(^| +\]\[ +)\* (ipscan|wormride)/
                | /(^| *)asn1/
                ;
        global skip_msgs =
                /.*AUTH .*/
                | /.*\*\*\* Your host is .*/
                | /.*\*\*\* If you are having problems connecting .*/
                ;
        redef enum Notice::Type += {
                IrcBotServerFound,
                IrcBotClientFound,
        };
        type channel: record {
name: string;
passwords: set[string];
topic: string &default="";
topic_history: vector of string;
        };
        type bot_client: record {
host: addr;
p: port;
nick: string &default="";
user: string &default="";
realname: string &default="";
channels: table[string] of channel;
servers: set[addr] &optional;
first_seen: time;
last_seen: time;
        };
        type bot_server: record {
host: addr;
p: set[port];
clients: table[addr] of bot_client;
global_users: string &default="";
passwords: set[string];
channels: table[string] of channel;
first_seen: time;
last_seen: time;
        };
        type bot_conn: record {
client: bot_client;
server: bot_server;
conn: connection;
fd: file;
ircx: bool &default=F;
        };
# We keep three sets of clients/servers:
#  (1) tables containing all IRC clients/servers
#  (2) sets containing potential bot hosts
#  (3) sets containing confirmend bot hosts
#
# Hosts are confirmed when a connection is established between
# potential bot hosts.
#
# FIXME: (1) should really be moved into the general IRC script.
        global expire_server:
                function(t: table[addr] of bot_server, idx: addr): interval;
        global expire_client:
                function(t: table[addr] of bot_client, idx: addr): interval;
        global servers: table[addr] of bot_server &write_expire=24 hrs
                &expire_func=expire_server &persistent;
        global clients: table[addr] of bot_client &write_expire=24 hrs
                &expire_func=expire_client &persistent;
        global potential_bot_clients: set[addr] &persistent;
        global potential_bot_servers: set[addr] &persistent;
        global confirmed_bot_clients: set[addr] &persistent;
        global confirmed_bot_servers: set[addr] &persistent;
# All IRC connections.
        global conns: table[conn_id] of bot_conn &persistent;
# Connections between confirmed hosts.
        global bot_conns: set[conn_id] &persistent;
# Helper functions for readable output.
        global strset_to_str: function(s: set[string]) : string;
        global portset_to_str: function(s: set[port]) : string;
        global addrset_to_str: function(s: set[addr]) : string;
}
function strset_to_str(s: set[string]) : string
{
        if ( |s| == 0 )
                return "<none>";
        local r = "";
        for ( i in s )
        {
                if ( r != "" )
                        r = cat(r, ",");
                r = cat(r, fmt("\"%s\"", i));
        }
        return r;
}
function portset_to_str(s: set[port]) : string
{
        if ( |s| == 0 )
                return "<none>";
        local r = "";
        for ( i in s )
        {
                if ( r != "" )
                        r = cat(r, ",");
                r = cat(r, fmt("%d", i));
        }
        return r;
}
function addrset_to_str(s: set[addr]) : string
{
        if ( |s| == 0 )
                return "<none>";
        local r = "";
        for ( i in s )
        {
                if ( r != "" )
                        r = cat(r, ",");
                r = cat(r, fmt("%s", i));
        }
        return r;
}
function fmt_time(t: time) : string
{
        return strftime("%y-%m-%d-%H-%M-%S", t);
}
event print_bot_state()
{
        local bot_summary_log = open_log_file("irc-bots.summary");
        disable_print_hook(bot_summary_log);
        print bot_summary_log, "---------------------------";
        print bot_summary_log, strftime("%y-%m-%d-%H-%M-%S", network_time());
        print bot_summary_log, "---------------------------";
        print bot_summary_log;
        print bot_summary_log, "Known servers";
        for ( h in confirmed_bot_servers )
        {
                local s = servers[h];
                print bot_summary_log,
                      fmt("    %s %s - clients: %d ports %s password(s) %s last-seen
%s first-seen %s global-users %s",
                                         "L",
                                        s$host, |s$clients|, portset_to_str(s$p),
                                        strset_to_str(s$passwords),
                                        fmt_time(s$last_seen), fmt_time(s$first_seen),
                                        s$global_users);
                for ( name in s$channels )
                {
                        local ch = s$channels[name];
                        print bot_summary_log,
                              fmt("        channel %s: topic \"%s\", password(s) %s",
                                         ch$name, ch$topic,
                                         strset_to_str(ch$passwords));
                }
        }
        print bot_summary_log, "\nKnown clients";
        for ( h in confirmed_bot_clients )
        {
                local c = clients[h];
                print bot_summary_log,
                      fmt("    %s %s - server(s) %s user %s nick %s realname %s last-
seen %s first-seen %s",
                                      "L", h,
                                     addrset_to_str(c$servers),
                                     c$user, c$nick, c$realname,
                                     fmt_time(c$last_seen), fmt_time(c$first_seen));
        }
        close(bot_summary_log);

        if ( summary_interval != 0 secs )
                schedule summary_interval { print_bot_state() };
}
function do_log_force(c: connection, msg: string)
{
        local id = c$id;
        print bot_log, fmt("%.6f %s:%d > %s:%d %s %s",
                        network_time(), id$orig_h, id$orig_p,
                        id$resp_h, id$resp_p, c$addl, msg);
}
function do_log(c: connection, msg: string)
{
        if ( c$id !in bot_conns )
                return;

        do_log_force(c, msg);
}
function log_msg(c: connection, cmd: string, prefix: string, msg: string)
{
        if ( skip_msgs in msg )
                return;
        do_log(c, fmt("MSG command=%s prefix=%s msg=\"%s\"", cmd, prefix, msg));
}
function update_timestamps(c: connection) : bot_conn
{
        local conn = conns[c$id];
        conn$client$last_seen = network_time();
        conn$server$last_seen = network_time();
# To prevent the set of entries from premature expiration,
# we need to make a write access (can't use read_expire as we
# iterate over the entries on a regular basis).
        clients[c$id$orig_h] = conn$client;
        servers[c$id$resp_h] = conn$server;
        return conn;
}
function add_server(c: connection) : bot_server
{
        local s_h = c$id$resp_h;
        if ( s_h in servers )
                return servers[s_h];
        local empty_table1: table[addr] of bot_client;
        local empty_table2: table[string] of channel;
        local empty_set: set[string];
        local empty_set2: set[port];
        local server = [$host=s_h, $p=empty_set2, $clients=empty_table1,
              $channels=empty_table2, $passwords=empty_set,
              $first_seen=network_time(), $last_seen=network_time()];
        servers[s_h] = server;
        return server;
}
function add_client(c: connection) : bot_client
{
        local c_h = c$id$orig_h;
        if ( c_h in clients )
                return clients[c_h];
        local empty_table: table[string] of channel;
        local empty_set: set[addr];
        local client = [$host=c_h, $p=c$id$resp_p, $servers=empty_set,
              $channels=empty_table, $first_seen=network_time(),
              $last_seen=network_time()];
        clients[c_h] = client;
        return client;
}
function check_bot_conn(c: connection)
{
        if ( c$id in bot_conns )
                return;
        local client = c$id$orig_h;
        local server = c$id$resp_h;
        if ( client !in potential_bot_clients || server !in potential_bot_servers )
                return;
# New confirmed bot_conn.
        add bot_conns[c$id];
        if ( server !in confirmed_bot_servers )
        {
                NOTICE([$note=IrcBotServerFound, $src=server, $p=c$id$resp_p, $conn=c,
                                $msg=fmt("ircbot server found: %s:%d", server,
$p=c$id$resp_p)]);
                add confirmed_bot_servers[server];
        }
        if ( client !in confirmed_bot_clients )
        {
                NOTICE([$note=IrcBotClientFound, $src=client, $p=c$id$orig_p, $conn=c,
                                $msg=fmt("ircbot client found: %s:%d", client,
$p=c$id$orig_p)]);
                add confirmed_bot_clients[client];
        }
}
function get_conn(c: connection) : bot_conn
{
        local conn: bot_conn;
        if ( c$id in conns )
        {
                check_bot_conn(c);
                return update_timestamps(c);
        }
        local c_h = c$id$orig_h;
        local s_h = c$id$resp_h;
        local client : bot_client;
        local server : bot_server;
        if ( c_h in clients )
                client = clients[c_h];
        else
                client = add_client(c);
        if ( s_h in servers )
                server = servers[s_h];
        else
                server = add_server(c);
        server$clients[c_h] = client;
        add server$p[c$id$resp_p];
        add client$servers[s_h];
        conn$server = server;
        conn$client = client;
        conn$conn = c;
        conns[c$id] = conn;
        update_timestamps(c);
        return conn;
}
function expire_server(t: table[addr] of bot_server, idx: addr): interval
{
        local server = t[idx];
        for ( c in server$clients )
        {
                local client = server$clients[c];
                delete client$servers[idx];
        }
        delete potential_bot_servers[idx];
        delete confirmed_bot_servers[idx];
        return 0secs;
}
function expire_client(t: table[addr] of bot_client, idx: addr): interval
{
        local client = t[idx];
        for ( s in client$servers )
                if ( s in servers )
                        delete servers[s]$clients[idx];
        delete potential_bot_clients[idx];
        delete confirmed_bot_clients[idx];
        return 0secs;
}
function remove_connection(c: connection)
{
        local conn = conns[c$id];
        delete conns[c$id];
        delete bot_conns[c$id];
}
event connection_state_remove(c: connection)
{
        if ( c$id !in conns )
                return;
        remove_connection(c);
}
event irc_client(c: connection, is_orig: bool, prefix: string, data: string)
{
        if ( detailed_logging )
                print detailed_log, fmt("%.6f %s > (%s) %s", network_time(),
id_string(c$id), prefix, data);
        local conn = get_conn(c);
        if ( data == /^ *[iI][rR][cC][xX] *$/ )
                conn$ircx = T;
}
event irc_server(c: connection, is_orig: bool, prefix: string, data: string)
{
        if ( detailed_logging )
                print detailed_log, fmt("%.6f %s < (%s) %s", network_time(),
id_string(c$id), prefix, data);
        local conn = get_conn(c);
}
event irc_user_message(c: connection, is_orig: bool, user: string, host: string,
server: string, real_name: string)
{
        local conn = get_conn(c);
        conn$client$user = user;
        conn$client$realname = real_name;
        do_log(c, fmt("USER user=%s host=%s server=%s real_name=%s", user, host,
server, real_name));
}
function get_channel(conn: bot_conn, channel: string) : channel
{
        if ( channel in conn$server$channels )
                return conn$server$channels[channel];
        else
        {
                local empty_set: set[string];
                local empty_vec: vector of string;
                local ch = [$name=channel, $passwords=empty_set,
$topic_history=empty_vec];
                conn$server$channels[ch$name] = ch;
                return ch;
        }
}
event irc_join_message(c: connection, is_orig: bool, info_list: irc_join_list)
{
        local conn = get_conn(c);
        for ( i in info_list )
        {
                local ch = get_channel(conn, i$channel);
                if ( i$password != "" )
                        add ch$passwords[i$password];
                conn$client$channels[ch$name] = ch;
                do_log(c, fmt("JOIN channel=%s password=%s", i$channel, i$password));
        }
}
global urls: set[string] &read_expire = 7 days &persistent;
```

```
event http_request(c: connection, method: string, original_URI: string,
		unescaped_URI: string, version: string)
{
	if ( original_URI in urls )
		do_log_force(c, fmt("Request for URL %s", original_URI));
}
event irc_channel_topic(c: connection, is_orig: bool, channel: string, topic: string)
{
	if ( bot_cmds in topic )
	{
		do_log_force(c, fmt("Matching TOPIC %s", topic));
		add potential_bot_servers[c$id$resp_h];
	}
	local conn = get_conn(c);
	local ch = get_channel(conn, channel);
	ch$topic_history[|ch$topic_history| + 1] = ch$topic;
	ch$topic = topic;
	if ( c$id in bot_conns )
	{
		do_log(c, fmt("TOPIC channel=%s topic=\"%s\"", channel, topic));
		local s = split(topic, / /);
		for ( i in s )
		{
			local w = s[i];
			if ( w == /[a-zA-Z]+:\/\/.*/ )
			{
				add urls[w];
				do_log(c, fmt("URL channel=%s url=\"%s\"",
							channel, w));
			}
		}
	}
}
event irc_nick_message(c: connection, is_orig: bool, who: string, newnick: string)
{
	if ( bot_nicks in newnick )
	{
		do_log_force(c, fmt("Matching NICK %s", newnick));
		add potential_bot_clients[c$id$orig_h];
	}
	local conn = get_conn(c);
	conn$client$nick = newnick;
	do_log(c, fmt("NICK who=%s nick=%s", who, newnick));
}
event irc_password_message(c: connection, is_orig: bool, password: string)
{
	local conn = get_conn(c);
	add conn$server$passwords[password];
	do_log(c, fmt("PASS password=%s", password));
}
event irc_privmsg_message(c: connection, is_orig: bool, source: string, target: 
string,
		message: string)
{
	log_msg(c, "privmsg", source, fmt("->%s %s", target, message));
}
event irc_notice_message(c: connection, is_orig: bool, source: string,
		target: string, message: string)
{
	log_msg(c, "notice", source, fmt("->%s %s", target, message));
}
event irc_global_users(c: connection, is_orig: bool, prefix: string, msg: string)
{
	local conn = get_conn(c);
# Better would be to parse the message to extract the counts.
	conn$server$global_users = msg;
	log_msg(c, "globalusers", prefix, msg);
}

event Input::end_of_data(name: string, source: string) {
for(ip in suspicious) {
		#print ip;
```

```
        }
}

event bro_done()
{
}
event bro_init() &priority = -5
{
        if ( summary_interval != 0 secs )
                schedule summary_interval { print_bot_state() };
Log::disable_stream(Conn::LOG);
Log::disable_stream(HTTP::LOG);
Log::disable_stream(Files::LOG);
}
```

F.3 Sample of Snort Rules for Botnet Detection

```
alert udp $HOME_NET 1024:65535 -> $EXTERNAL_NET 1024:65535 (msg:"E7[rb] BOTHUNTER
Storm(Peacomm) Peer Coordination Event [SEARCH RESULT]"; content:"|E311|"; depth:5;
rawbytes; pcre:"/[0-9]+\.mpg\;size\=[0-9]+/x"; rawbytes; classtype:bad-unknown;
sid:9910013; rev:99;)

alert udp $HOME_NET 1024:65535 -> $EXTERNAL_NET 1024:65535 (msg:"E7[rb] BOTHUNTER
Storm Worm Peer Coordination Event [PUBLISH]"; content:"|E313|"; depth:5; rawbytes;
pcre:"/[0-9]+\.mpg\;size\=[0-9]+/x"; rawbytes; classtype:bad-unknown; sid:9910011;
rev:99;)
```