ARL-TR-9964 • Sep 2024

The Lossy Intelligent Network Traffic Squeezer (LINTS)

by Sidney C Smith

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-9964 ● SEP 2024

The Lossy Intelligent Network Traffic Squeezer (LINTS)

by Sidney C Smith
DEVCOM Army Research Laboratory

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
Sep 2024	Technical Report	August 2016	March 2020

4. TITLE AND SUBTITLE

The Lossy Intelligent Network Traffic Squeezer (LINTS)

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**

6. AUTHOR(S)

Sidney C Smith

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

DEVCOM Army Research Laboratory
ATTN: FCDD-RLA-ND
Aberdeen Proving Ground, MD 21005-5066

8. PERFORMING ORGANIZATION REPORT NUMBER

ARL-TR-9964

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

primary author's email: <sidney.c.smith24.civ@army.mil>.

14. ABSTRACT

This report documents the DEVCOM Army Research Laboratory Lossy Intelligent Network Traffic Squeezer (LINTS). LINTS is a refinement of the previous Netcomp tool. This report will compare the efficiency and effectiveness of LINTS against the Netcomp. LINTS implements several different compression strategies to compress the network traffic which must be transmitted from the network intrusion detection sensor and the central analysis servers in a network intrusion detection application. This report will examine the effects of compressing several different data sets of network traffic with each of these strategies. As a final test two data sets will be divided into two pieces. One piece will be use to adjust the LINTS parameters for a blind test with the other pieces of the data set.

15. SUBJECT TERMS

Network Intrusion Detection; Lossy Compression; N-gram; Bloom filter; Tainted Flow; entropy; LINTS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UU	181

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
Sidney C Smith	410-278-6235

STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures v

List of Tables ix

Acknowledgments xiv

1. Introduction 1

2. Background 2

3. Approach 5

3.1 Lossy Intelligent Network Traffic Squeezer (LINTS) 6

3.1.1 Compression Strategies 6

3.1.2 Optimization 8

3.2 Data Sets 9

3.2.1 ARL 2016 9

3.2.2 ARL 2017 9

3.2.3 CIC-IDS 2017 9

3.2.4 Cyber Defense Exercise (CDX) 2009 10

3.2.5 ISCX-IDS 2012 10

3.2.6 MACCDC 2010 11

3.2.7 MACCDC 2011 11

3.2.8 MACCDC 2012 11

3.2.9 UNSW-NB15 12

3.3 Rule Sets 12

3.4 Preparation 13

3.4.1 Data Set Database 13

3.4.2 Bloom Filters 14

3.5 Experimental Control 15

4. Results 16

4.1 Comparison against Netcomp 17

4.1.1 Information Security Centre of Excellence (ISCX) 2012 Data Set 17
4.1.2 Canadian Institute for Cybersecurity Intrusion Detection System (CIC-IDS) 2017 18
4.1.3 Netcomp vs. LINTS Observations 20
4.2 LINTS Trials 21
4.2.1 Compress Only TCP 21
4.2.2 Entropy 23
4.2.3 Capped by Packets 30
4.2.4 Capped by Bytes 37
4.2.5 Tainted Flow Capped by Packets 44
4.2.6 Tainted Flow Capped by Bytes 51
4.2.7 Tainted Flow Capped by Packets with Entropy 58
4.2.8 Tainted Flow Capped by Bytes with Entropy 65
4.2.9 LINTS Trial Observations 72
4.3 LINTS Compression Based upon Lessons Learned 72
4.3.1 UNSW-NB15 February 72
4.3.2 ARL 2017 73

5. Conclusion 73

6. References 75

Appendix A. Online Manual Pages 78

Appendix B. Shell Scripts 87

Appendix C. Raw Results 93

List of Symbols, Abbreviations, and Acronyms 164

List of Figures

Fig. 1 Distributed network intrusion detection system ……1

Fig. 2 Proposed Kelly compressor ……2

Fig. 3 Network traffic composition ……4

Fig. 4 Netcomp compression using the ISCX 15 June 2012 data set and the RegAug2013 rule set …… 18

Fig. 5 LINTS compression using the ISCX 15 June 2012 data set and the RegAug2013 rule set …… 19

Fig. 6 Netcomp compression using the Canadian Institute for Cybersecurity (CIC) 5 July 2018 data set and the RegSep2018 rule set …… 19

Fig. 7 LINTS compression using day 3 of the CIC-IDS 2017 data set and the RegSep2018 rule set …… 20

Fig. 8 LINTS compression using the entropy strategy on the CDX 2009 gator-usama010 data set as analyzed by the Circa2009 rule set …… 25

Fig. 9 LINTS compression using the entropy strategy on the CDX 2009 gator-usama020 data set as analyzed by the Circa2009 rule set …… 25

Fig. 10 LINTS compression using the entropy strategy on the Mid-Atlantic Collegiate Cyber Defense Competition (MACCDC) 2010 testing data set as analyzed by the Circa 2009 rule set …… 26

Fig. 11 LINTS compression using the entropy strategy on the MACCDC 2011 testing data set as analyzed by the RegAug2013 rule set …… 26

Fig. 12 LINTS compression using the entropy strategy on the MACCDC 2012 testing data set as analyzed by the RegAug2013 rule set …… 27

Fig. 13 LINTS compression using the entropy strategy on the ISCX 2012 testing data set as analyzed by the RegAug2013 rule set …… 27

Fig. 14 LINTS compression using the entropy strategy on the UNSW-NB 2015 January data set as analyzed by the RegSep2018 rule set …… 28

Fig. 15 LINTS compression using the entropy strategy on the ARL2016 data set as analyzed by the RegSep2018 rule set …… 28

Fig. 16 LINTS compression using the entropy strategy on the CIC 2017 testing data set as analyzed by the RegJul2018 rule set …… 29

Fig. 17 LINTS compression using the capped-by-packets strategy on the CDX 2009 gator-usama010 data set as analyzed by the Circa2009 rule set . 32

Fig. 18 LINTS compression using the capped-by-packets strategy on the CDX 2009 gator-usama020 data set as analyzed by the Circa2009 rule set . 32

Fig. 19 LINTS compression using the capped-by-packets strategy on the MACCDC 2010 data set as analyzed by the Circa 2009 rule set 33

Fig. 20 LINTS compression using the capped-by-packets strategy on the MACCDC 2011 data set as analyzed by the RegAug2013 rule set 33

Fig. 21 LINTS compression using the capped-by-packets strategy on the MACCDC 2012 data set as analyzed by the RegAug2013 rule set 34

Fig. 22 LINTS compression using the capped-by-packets strategy on the ISCX 2012 testing data set as analyzed by the RegAug2013 rule set 34

Fig. 23 LINTS compression using the capped-by-packets strategy on the UNSW-NB 2015 January data set as analyzed by the RegSep2018 rule set 35

Fig. 24 LINTS compression using the capped-by-packets strategy on the ARL 2016 data set as analyzed by the RegSep2018 rule set 35

Fig. 25 LINTS compression using the capped-by-packets strategy on the CIC 2017 testing data set as analyzed by the RegJul2018 rule set 36

Fig. 26 LINTS compression using the capped-by-bytes strategy on the CDX 2009 gator-usama010 data set as analyzed by the Circa2009 rule set . 39

Fig. 27 LINTS compression using the capped-by-bytes strategy on the CDX 2009 gator-usama020 data set as analyzed by the Circa2009 rule set . 39

Fig. 28 LINTS compression using the capped-by-bytes strategy on the MACCDC 2010 data set as analyzed by the Circa2009 rule set 40

Fig. 29 LINTS compression using the capped-by-bytes strategy on the MACCDC 2011 data set as analyzed by the RegAug2013 rule set 40

Fig. 30 LINTS compression using the capped-by-bytes strategy on the MACCDC 2012 data set as analyzed by the RegAug2013 rule set 41

Fig. 31 LINTS compression using the capped-by-bytes strategy on the ISCX 2012 testing data set as analyzed by the RegJul2018 rule set 41

Fig. 32 LINTS compression using the capped-by-bytes strategy on the UNSW-NB 2015 January data set as analyzed by the RegSep2018 rule set 42

Fig. 33 LINTS compression using the capped-by-bytes strategy on the Army Research Laboratory (ARL) 2016 data set as analyzed by the RegJul2018 rule set 42

Fig. 34 LINTS compression using the capped-by-bytes strategy on the CIC 2017 testing data set as analyzed by the RegJul2018 rule set 43

Fig. 35 LINTS compression using the tainted flow capped-by-packets strategy on the CDX 2009 gator-usama010 data set as analyzed by the Circa2009 rule set 46

Fig. 36 LINTS compression using the tainted flow capped-by-packets strategy on the CDX 2009 gator-usama020 data set as analyzed by the Circa2009 rule set 46

Fig. 37 LINTS compression using the tainted flow capped-by-packets strategy on the MACCDC 2010 data set as analyzed by the Circa 2009 rule set 47

Fig. 38 LINTS compression using the tainted flow capped-by-packets strategy on the MACCDC 2011 data set as analyzed by the RegAug2013 rule set 47

Fig. 39 LINTS compression using the tainted flow capped-by-packets strategy on the MACCDC 2012 data set as analyzed by the RegAug2013 rule set 48

Fig. 40 LINTS compression using the tainted flow capped-by-packets strategy on the ISCX 2012 testing data set as analyzed by the RegAug2013 rule set 48

Fig. 41 LINTS compression using the tainted flow capped-by-bytes strategy on the UNSW-NB 2015 January data set as analyzed by the RegSep2018 rule set 49

Fig. 42 LINTS compression using the tainted flow capped-by-packets strategy on the ARL 2016 data set as analyzed by the RegJul2018 rule set 49

Fig. 43 LINTS compression using the tainted flow capped-by-packets strategy on the CIC 2017 testing data set as analyzed by the RegJul2018 rule set 50

Fig. 44 LINTS compression using the tainted flow capped-by-bytes strategy on the CDX 2009 gator-usama010 data set as analyzed by the Circa2009 rule set 53

Fig. 45 LINTS compression using the tainted flow capped-by-bytes strategy on the CDX 2009 gator-usama020 data set as analyzed by the Circa2009 rule set 53

Fig. 46 LINTS compression using the tainted flow capped-by-bytes strategy on the MACCDC 2010 data set as analyzed by the Circa 2009 rule set .. 54

Fig. 47 LINTS compression using the tainted flow capped-by-bytes strategy on the MACCDC 2011 data set as analyzed by the RegAug2013 rule set 54

Fig. 48 LINTS compression using the tainted flow capped-by-bytes strategy on the MACCDC 2012 data set as analyzed by the RegAug2013 rule set 55

Fig. 49 LINTS compression using the tainted flow capped-by-bytes strategy on the ISCX 2012 testing data set as analyzed by the RegJul2018 rule set 55

Fig. 50 LINTS compression using the tainted flow capped-by-bytes strategy on the UNSW-NB 2015 January data set as analyzed by the RegSep2018 rule set 56

Fig. 51 LINTS compression using the tainted flow capped-by-bytes strategy on the ARL 2016 data set as analyzed by the RegJul2018 rule set 56

Fig. 52 LINTS compression using the tainted flow capped-by-bytes strategy on the CIC 2017 testing data set as analyzed by the RegJul2018 rule set 57

Fig. 53 LINTS compression using the tainted flow capped-by-packets with entropy strategy on the CDX 2009 gator-usama010 data set as analyzed by the Circa2009 rule set 60

Fig. 54 LINTS compression using the tainted flow capped-by-packets with entropy strategy on the CDX 2009 gator-usama020 data set as analyzed by the Circa2009 rule set 60

Fig. 55 LINTS compression using the tainted flow capped-by-packets with entropy strategy on the MACCDC 2010 data set as analyzed by the Circa2009 rule set 61

Fig. 56 LINTS compression using the tainted flow capped-by-packets with entropy strategy on the MACCDC 2011 data set as analyzed by the RegAug2013 rule set 61

Fig. 57 LINTS compression using the tainted flow capped-by-packets with entropy strategy on the MACCDC 2012 data set as analyzed by the RegAug2013 rule set 62

Fig. 58 LINTS compression using the tainted flow capped-by-packets with entropy strategy on the ISCX 2012 testing data set as analyzed by the RegAug2013 rule set 62

Fig. 59 LINTS compression using the tainted flow capped-by-packets with entropy strategy on the UNSW-NB 2015 January data set as analyzed by the RegSep2018 rule set 63

Fig. 60 LINTS compression using the tainted flow capped-by-packets strategy with entropy on the ARL 2016 data set as analyzed by the RegSep2018 rule set 63

Fig. 61 LINTS compression using the tainted flow capped-by-packets with entropy strategy on the CIC 2017 testing data set as analyzed by the RegJul2018 rule set 64

Fig. 62 LINTS compression using the tainted flow capped-by-bytes with entropy strategy on the CDX 2009 gator-usama010 data set as analyzed by the Circa2009 rule set 67

Fig. 63 LINTS compression using the tainted flow capped-by-bytes with entropy strategy on the CDX 2009 gator-usama020 data set as analyzed by the Circa2009 rule set 67

Fig. 64 LINTS compression using the tainted flow capped-by-bytes with entropy strategy on the MACCDC 2010 data set as analyzed by the Circa2009 rule set 68

Fig. 65 LINTS compression using the tainted flow capped-by-bytes with entropy strategy on the MACCDC 2011 data set as analyzed by the RegAug2013 rule set 68

Fig. 66 LINTS compression using the tainted flow capped-by-bytes with entropy strategy on the MACCDC 2012 data set as analyzed by the RegAug2013 rule set 69

Fig. 67 LINTS compression using the tainted flow capped-by-bytes with entropy strategy on the ISCX 2012 testing data set as analyzed by the RegAug2013 rule set 69

Fig. 68 LINTS compression using the tainted flow capped-by-bytes with entropy strategy on the UNSW-NB 2015 January data set as analyzed by the RegSep2018 rule set 70

Fig. 69 LINTS compression using the tainted flow capped-by-bytes with entropy strategy on the ARL 2016 data set as analyzed by the RegSep2018 rule set 70

Fig. 70 LINTS compression using the tainted flow capped-by-bytes with entropy strategy on the CIC 2017 testing data set as analyzed by the RegJul2018 rule set 71

List of Tables

Table 1 Results from the Compress only TCP strategy..... 23

Table 2 Results from the entropy strategy..... 30

Table 3 Results from the capped-by-packets strategy..... 37

Table 4 Results from the capped-by-bytes strategy 43

Table 5 Results from the tainted flow capped-by-packets strategy 51

Table 6 Results from the tainted flow capped-by-bytes strategy..... 58

Table 7 Results from the tainted flow capped-by-packets with entropy strategy..... 65

Table 8 Results from the tainted flow capped-by-bytes with entropy strategy 72

Table C-1 Results from Netcomp compression using the ISCX 15 June 2012 data set and the RegAug2013 rule set 94

Table C-2 Results from LINTS compression using the ISCX 15 Jun 2012 data set and the RegAug2013 rule set 95

Table C-3 Results from Netcomp compression using the CIC 5 July 2017 data set and the RegSep2018 rule set 96

Table C-4 Results from LINTS compression using day 3 of the CIC-IDS data set and the RegSep2018 rule set 97

Table C-5 Results using the LINTS entropy strategy to compress the CDX 2009 usama-gator010 data set and the Circa 2009 rule set 98

Table C-6 Results using the LINTS entropy strategy to compress the CDX 2009 usama-gator020 data set and the Circa 2009 rule set 100

Table C-7 Results using the LINTS entropy strategy to compress the "' 2010 data set and the Circa 2009 rule set 102

Table C-8 Results using the LINTS entropy strategy to compress the MACCDC 2011 data set and the RegAug2013 rule set 104

Table C-9 Results using the LINTS entropy strategy to compress the MACCDC 2012 data set and the RegAug2013 rule set 106

Table C-10 Results using the LINTS entropy strategy to compress the ISCX 2012 testing data set and the RegAug2013 rule set 108

Table C-11 Results using the LINTS entropy strategy to compress the UNSW-NB 2015 January data set and the RegSep2018 rule set 110

Table C-12 Results using the LINTS entropy strategy to compress the CIC 2017 testing data set and the registered Snort rule set from July 2018 ... 112

Table C-13 Results using the LINTS capped-by-packets strategy to compress the CDX 2009 usama-gator010 data set and the Circa2009 rule set 114

Table C-14 Results using the LINTS capped-by-packets strategy to compress the CDX 2009 usama-gator020 data set and the Circa2009 rule set 115

Table C-15 Results using the LINTS capped-by-packets strategy to compress the MACCDC 2012 data set and the RegAug2013 rule set 116

Table C-16 Results using the LINTS capped-by-packets strategy to compress the ISCX 2012 testing data set and the RegAug2013 rule set 117

Table C-17 Results using the LINTS capped-by-packets strategy to compress the UNSW-NB 2015 January data set and the RegSep2018 rule set ... 118

Table C-18 Results using the LINTS capped-by-packets strategy to compress the CIC 2017 testing data set and the RegJul2018 rule 124

Table C-19 Results using the LINTS capped-by-bytes strategy to compress the CDX 2009 usama-gator010 data set and the Circa 2009 rule set … 125

Table C-20 Results using the LINTS capped-by-bytes strategy to compress the CDX 2009 usama-gator020 data set and the Circa 2009 rule set … 126

Table C-21 Results using the LINTS capped-by-bytes strategy to compress the MACCDC 2012 data set and the RegAug2013 rule set …… 127

Table C-22 Results using the LINTS capped-by-bytes strategy to compress the ISCX 2012 testing data set and the RegAug2013 rule set …… 128

Table C-23 Results using the LINTS capped-by-bytes strategy to compress the UNSW-NB 2015 January data set and the RegSep2018 rule set … 129

Table C-24 Results using the LINTS capped-by-bytes strategy to compress the CIC 2017 testing data set and the RegJul2018 rule …… 130

Table C-25 Results using the LINTS tainted flow capped-by-packets strategy to compress the CDX 2009 usama-gator010 data set and the Circa 2009 rule set …… 131

Table C-26 Results using the LINTS tainted flow capped-by-packets strategy to compress the CDX 2009 usama-gator020 data set and the Circa 2009 rule set …… 131

Table C-27 Results using the LINTS tainted flow capped-by-packets strategy to compress the MACCDC 2012 data set and the RegAug2013 rule set …… 133

Table C-28 Results using the LINTS tainted flow capped-by-packets strategy to compress the ISCX 2012 testing data set and the RegAug2013 rule set …… 134

Table C-29 Results using the LINTS tainted flow capped-by-packets strategy to compress the UNSW-NB 2015 January data set and the RegSep2018 rule set …… 135

Table C-30 Results using the LINTS tainted flow capped-by-packets strategy to compress the CICCIC 2017 testing data set and the RegJul2018 rule …… 140

Table C-31 Results using the LINTS tainted flow capped-by-bytes strategy to compress the CDX 2009 usama-gator010 data set and the Circa 2009 rule set …… 141

Table C-32 Results using the LINTS tainted flow capped-by-bytes strategy to compress the CDX 2009 usama-gator020 data set and the Circa 2009 rule set …… 141

Table C-33 Results using the LINTS tainted flow capped-by-bytes strategy to compress the MACCDC 2012 data set and the RegAug2013 rule set 143

Table C-34 Results using the LINTS tainted flow capped-by-bytes strategy to compress the ISCX 2012 testing data set and the RegAug2013 rule set 144

Table C-35 Results using the LINTS tainted flow capped-by-bytes strategy to compress the UNSW-NB 2015 January data set and the RegSep2018 rule set 145

Table C-36 Results using the LINTS tainted flow capped-by-bytes strategy to compress the CIC 2017 testing data set and the RegJul2018 rule .. 145

Table C-37 Results using the LINTS tainted flow capped by packets with entropy strategy to compress the CDX 2009 usama-gator010 data set and the Circa 2009 rule set 147

Table C-38 Results using the LINTS tainted flow capped by packets with strategy to compress the CDX 2009 usama-gator020 data set and the Circa 2009 rule set 148

Table C-39 Results using the LINTS tainted flow capped by packets with entropy strategy to compress the MACCDC 2012 data set and the RegAug2013 rule set 149

Table C-40 Results using the LINTS tainted flow capped by packets with entropy strategy to compress the ISCX 2012 testing data set and the RegAug2013 rule set 150

Table C-41 Results using the LINTS tainted flow capped by packets with entropy strategy to compress the UNSW-NB 2015 January data set and the RegSep2018 rule set 151

Table C-42 Results using the LINTS tainted flow capped by packets with entropy strategy to compress the CIC 2017 testing data set and the RegJul2018 rule 152

Table C-43 Results using the LINTS tainted flow capped by bytes with entropy strategy to compress the CDX 2009 usama-gator010 data set and the Circa 2009 rule set 153

Table C-44 Results using the LINTS tainted flow capped by bytes with entropy strategy to compress the CDX 2009 usama-gator020 data set and the Circa 2009 rule set 153

Table C-45 Results using the LINTS tainted flow capped by bytes with entropy strategy to compress the MACCDC 2012 data set and the RegAug2013 rule set 155

Table C-46 Results using the LINTS tainted flow capped by bytes with entropy strategy to compress the ISCX 2012 testing data set and the RegAug2013 rule set 156

Table C-47 Results using the LINTS tainted flow capped by bytes with entropy strategy to compress the UNSW-NB 2015 January data set and the RegSep2018 rule set 157

Table C-48 Results using the LINTS tainted flow capped by bytes with entropy strategy to compress the CIC 2017 testing data set and the RegJul2018 rule 162

Acknowledgments

I would like to thank my supervisors at the DEVCOM Army Research Laboratory, Messrs Curtis Brandon Arnold and Jerry Allan Clarke, for their support in this effort. In addition, I would like to thank my colleagues on the Product Integration and Test Team, especially Messrs Ralph Paul Ritchey, Travis William Parker, Carlos J Mateo, Kin Wai Wong, and Stephen R Neyens. I would also like to thank Dr Lisa M Marvel who started me on this path by suggesting that I consider using the Kelly criterion. Furthermore, I would like to thank my advisor, Dr Robert J Hammell II.

1. Introduction

A distributed network intrusion detection system (NIDS) allows a relatively small number of highly trained analysts to monitor a much larger number of sites; however, it requires information to be transmitted from the remote sensors to a central analysis system (CAS), as pictured in Fig. 1. Unless an expensive dedicated NIDS network is employed, this transmission must use the same channels that the site uses to conduct their daily business. This makes it important to reduce the amount of information transmitted back to the CAS to minimize the impact that the NIDS has on daily operations.

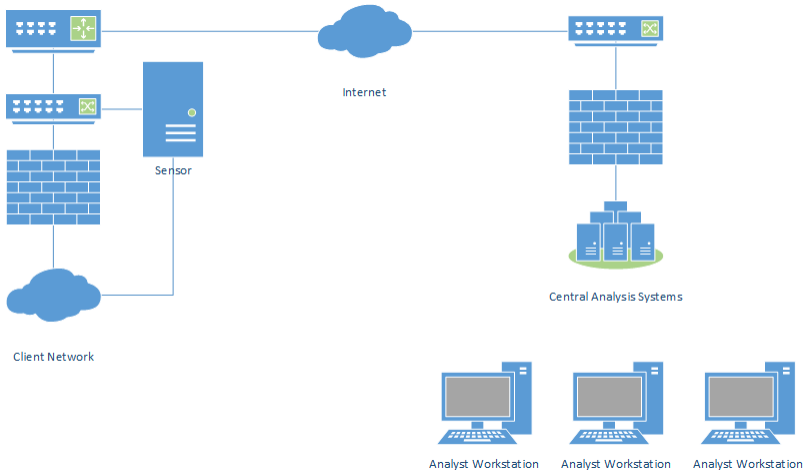

Fig. 1 Distributed network intrusion detection system

Smith and Hammell[1] proposed that it should be possible to create a lossy compression tool using anomaly detection techniques to rate each session and a modification of the Kelly criterion[2] to select how much traffic from each session to return, as seen in Fig. 2.

The Lossy Intelligent Network Traffic Squeezer (LINTS) is based upon that proposal[1] and the lessons learned from their research.[3–8] Lossy Intelligent Network Traffic Squeezer (LINTS) implements Smith and Hammell's tainted flows strategy[8] to compress network traffic without impacting the ability of the NIDS to detect and analyze malicious activity. Based upon Smith and Hammell's findings that malicious network flows will manifest their maliciousness early,[6] the tainted flows strategy adds the ability to determine if a flow is malicious by collecting malicious

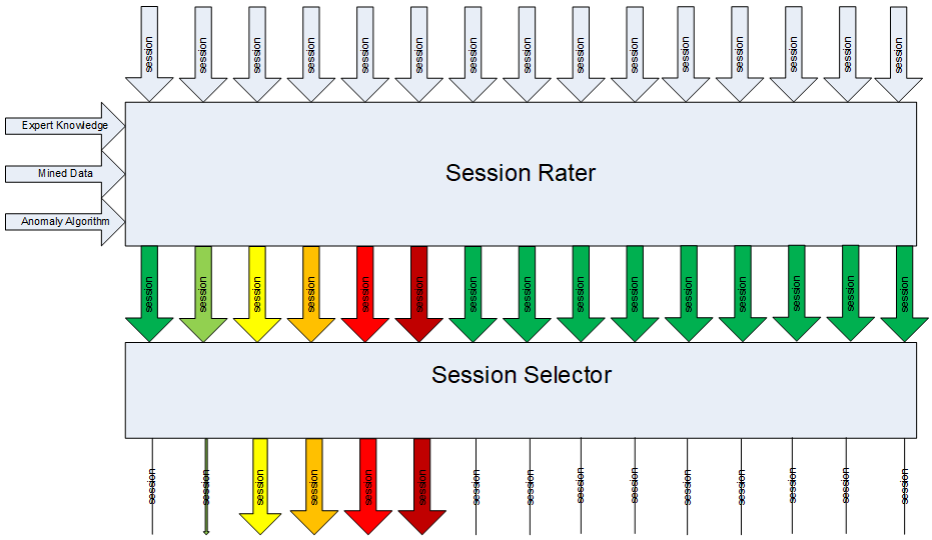

Fig. 2 Proposed Kelly compressor

N-grams and hashing them into a Bloom filter.[9] Packets are then broken into N-grams, which are hashed and compared to the Bloom filter. If the hash of an N-gram matches one in the malicious Bloom filter, the flow is tainted and all packets from that flow will be transmitted to the CAS. If no N-grams match, transmission of the flow to the CAS will stop after a user-defined threshold in either packets or bytes.

The remainder of this report is organized into the following sections: Section 2 provides background, Section 3 outlines the approach chosen to address this problem, Section 4 presents the results, and Section 5 provides a conclusion and discussion of future work.

2. Background

One popular strategy for implementing a distributed NIDS is to do all of the intrusion detection on the sensor and send only alerts or logs to the CAS.[10,11] A second strategy might be to use lossless compression to reduce the size of the data returned to the CAS. A third strategy is to implement some form of lossy compression algorithm to send back relevant portions of traffic.

There are three problems with the first strategy. The first is that it has the potential to overburden the sensor's central processing unit (CPU) and introduce packet loss. Smith et al. discovered that the impact of packet loss can sometimes be quite severe

for even small rates of packet loss.[12,13] The second problem is that the alerts by themselves often do not contain enough information to determine whether the attack was successful. The third problem is that these systems are most often implemented with signature-based intrusion detection engines. Signature-based systems may be tuned to produce few false positives; however, they are ineffective at detecting zero-day and advanced persistent threats.[14]

The problem with the second strategy is that lossless compression alone is not capable of reducing the amount of traffic enough. Using GNU Zip to compress the 2009 Cyber Defense Exercise[15] data set provides a compression ratio of 2:1.[3] Compression ratios of better than 10:1 are required to minimize the impact of NIDS on day-to-day operations.

The third strategy is to use lossy compression to provide a solution. Network traffic may be considered to be composed of sessions that span spectrums from known to unknown and malicious to benign, as illustrated in Fig. 3. Quadrant III, the known malicious quadrant, is the domain of intrusion prevention systems as described by Ierace et al.[16] This research is most interested in quadrant II, the unknown malicious quadrant, because that is the quadrant where evidence of zero-day and advanced persistent threat attacks will be found. In 2004, Long described the Interrogator Intrusion Detection System Architecture.[17] In this architecture, remotely deployed sensors collect network traffic and transmit a subset of the traffic to the analysis level. The Interrogator employs "a dynamic network traffic selection algorithm called Snapper."[17] Long and Morgan describe how they used data mining to discover known benign traffic that they excluded from the data transmitted back to the analysis servers.[18]

At the heart of Snapper's compression is a strategy called short snapping. This strategy captures portions of packets that are considered more likely to be malicious. Smith and Hammell discovered a problem with short snapping where detectors like Snort[10] will not analyze incomplete packets.[5] This is not a problem in the Interrogator Architecture[17] because the detector is executed on the sensor and the compressed data is reviewed by custom tools and an analyst. The resources required to analyze traffic are directly proportional to the number of rules in the rule set. Smith and Hammell discovered that even a small amount of packet loss can have a severe impact on the number of alerts lost.[13,19] Given these considerations, it is advantageous

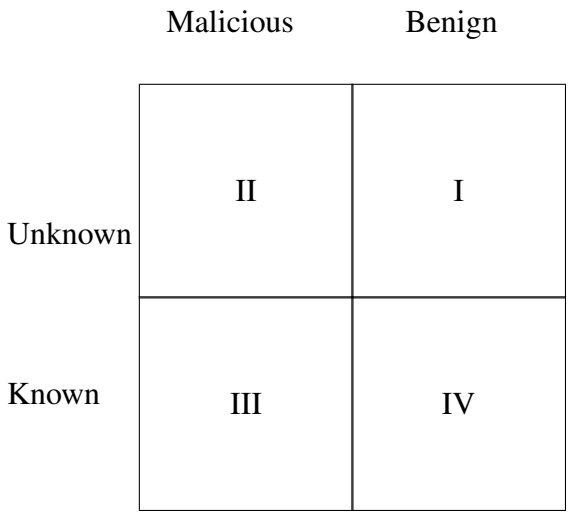

Fig. 3 Network traffic composition

to reduce the number of Snort rules deployed on the sensor, and analyze the compressed data on the CAS using Snort with an expanded rule set. The short snapping compression strategy would prevent this.

Smith and Hammell, in their work on compressing network traffic based upon flow features,[6] discovered that malicious flows manifested themselves early; however, some malicious flows remained malicious deep into the flow. In this work, the flow features are combined with a network intrusion detection algorithm to taint flows. This algorithm needs to be very lightweight. Anagram[20] used N-grams and Bloom filters[9] to differentiate benign and malicious traffic. Its approach was to load one Bloom filter with hashes of N-grams of known benign network traffic payloads and another Bloom filter with hashes of N-grams of known malicious network traffic payloads. ELIDe used N-grams but a different hashing method for detecting malicious traffic.[21] FAST-D,[22] inspired by Anagram and ELIDe, also makes use of N-grams and Bloom filters.

The tainted flow strategy was tested using a tool called Netcomp.[7,8] Netcomp implemented the entropy strategy,[3,23] the magnitude strategies,[5] the flow features strategies,[6] the benign Bloom filter strategy,[8] the malicious Bloom filter strategy,[8] and the tainted flow features strategy.[8] The results of these strategies were normalized

to a number between zero and one. These normalized results were then multiplied by a user-provided strategy-specific factor. These products were then summed to produce an overall threshold that was compared against a user-provided threshold. This is illustrated in Eq. 1 where R is the cumulative score, E is the normalized entropy score, F_e is the user-provided entropy factor, M is the normalized magnitude score, F_m is the user-provided magnitude factor, B is the normalized malicious or bad Bloom filter score, F_b is the user-provided malicious Bloom filter factor, G is the normalized benign or good Bloom filter score, F_g is the user-provided benign Bloom filter factor, T is the normalized tainted flow score, and F_t is the user-provided flow factor. Equation 1 is the Session Rater, and Eq. 2, where R is the cumulative score computed from Eq. 1 and t is a user-provided threshold, is the Session Selector from Fig. 2. If the result of Eq. 2 is true, then the packet is included in the compress network stream; otherwise, the packet is compressed out of the network stream.

$$R = F_e E + F_m M + F_b B + F_g G + F_t T \quad . \tag{1}$$

$$R >= t \quad . \tag{2}$$

Netcomp proved more successfull at compressing network traffic than lossless compression, and combining these strategies was more successfull than any single strategy[8]; however, there was still room for improvement.

3. Approach

LINTS builds upon and refines Netcomp.[7,8] This section addresses the improvements to Netcomp implemented in LINTS. Also covered are the cybersecurity data sets used to test both Netcomp and LINTS; the Snort[10] rule set used to test Netcomp and LINTS; the preparation required to perform the trials; and the experimental controls used to conduct the trials.

3.1 LINTS

In practice, the session selector described in Eq. 2 did not work as well as expected. LINTS reworks the way that the strategy thresholds are calculated and combined. Tracking network flows can be resource intensive; LINTS includes several improvements to optimize network flow tracking.

3.1.1 Compression Strategies

LINTS implements several compression strategies, and each of these strategies produces a score between –1 and 1. For each packet, LINTS iterates over each strategy in the order each threshold is defined until a strategy returns a score less than zero or a score greater than or equal to the user-provided threshold. Packets with a score in any strategy less than zero are known to be benign and are immediately compressed out of the network stream. Scores greater than or equal to the user-provided threshold are considered to be malicious and immediately retained in the network stream. If the strategies are exhausted, then the packet is assumed to be benign and is compressed out of the network stream.

3.1.1.1 Benign Berkeley Packet Filter

The benign Berkeley Packet Filter (BPF) strategy allows the user to define a BPF for known benign traffic. Typically this would include traffic between the sensor and the CAS. If the packet matches this filter, the score is –1; otherwise, it is zero.

3.1.1.2 Malicious Berkeley Packet Filter

The malicious BPF strategy allows the user to define a BPF for known malicious traffic. If the packet matches this filter, the score is 1; otherwise, the score is zero.

3.1.1.3 Compress Only Transmission Control Protocol (TCP) Traffic

The bulk of network traffic today employs Transmission Control Protocol (TCP). TCP is the only protocol that has flows and sessions. Some flow-tracking software attempts to apply flows and sessions to User Datagram Procotol (UDP) traffic; however, this is problematic since UDP is a connectionless protocol. Also, UDP contains most Domain Name Server (DNS) traffic, and Internet Control Message Protocol (ICMP) contains network control traffic, both of which are highly valuable for network intrusion detection. This strategy keeps every package that is not using the TCP and allows TCP packets to be futher examined for compression. If the packet employs TCP, then the score is zero; otherwise, the score is 1.

3.1.1.4 Entropy

The entropy[23] score does not really reflect a continuum; rather, payloads with very high entropy scores are either compressed or encrypted. There is value in excluding compressed data from the network stream because of the difficulty uncompressing on the CAS. If files must be carved out of the network stream, then this can best be done on the sensor. Furthermore, a hash of the file is often all that needs to be transmitted to the CAS. Tools like Bro[11] will easily perform this function. There is value in excluding encrypted data from the network stream because of the difficulty decrypting it on the CAS. Due to the random nature of high-entropy data, a false positive hit on a malicious N-gram is more likely. LINTS implements the entropy strategy by computing the entropy of the payload of a packet and comparing that to a user-provided threshold. If the computed entropy is less than the threshold, the score is zero. The score is –1 if the computed entropy value is greater than or equal to the threshold.

3.1.1.5 Capped

TCP flows or sessions may be capped either by packets or bytes. A flow is one side of the communication, and a session is both sides of the conversation. When the number of packets or bytes in a flow or session is less than the threshold, the score is 1. When the number of packets or bytes in a flow is greater than or equal to the threshold, the score is zero.

3.1.1.6 Tainted Flow

The tainted flow strategy tracks TCP flows counting the number of packets or bytes. If the current packet is below the user-provided threshold in either packets or bytes, the payload is split into N-grams and these N-grams are tested against a malicious Bloom filter. If malicious N-grams are found in the payload, then this flow is tainted. If the flow is below the threshold or the flow is tainted, the score is 1; otherwise, the score is zero. This strategy also has the option of tainting the other half of the conversation if a malicious N-gram is found.

3.1.1.7 Keep

The keep strategy always returns 1. It may be used with strategies like the entropy and benign BPF that only identify known benign packets. If used, any packet not previously determined benign is included in the network stream.

3.1.2 Optimization

In addition to reworking the way the various strategies' scores are computed and combined, efforts were made to optimize the code for efficiency. The flow-tracking code was rewritten to improve efficiency. The magnitude strategy from Netcomp was reworked to provide limits for how much the packet payload is analyzed for entropy and malicious N-grams.

3.1.2.1 Flow Tracking

The Netcomp program employed a C++ map data structure to store the active flows. This data structure is based upon a balanced binary tree and allows searching the active flow container in $O(n \log_2(n))$ time. LINTS employs a C++ unordered map data structure to store the active flows. This data structure is based upon a hash table and allows searching the active flow container in $O(1)$ time.

It is necessary to prune the active flows container of stale flows that have not properly terminated. This is especially true when the time to search the active flows container depends upon the number of active flows. The Netcomp program has a periodic time-out function. This function would scan the active flows container for flows that had not received a packet in a user-provided span of time. This operation could be very time consuming. It is also problematic because there is little consensus on how long a connected flow should go without receiving a packet before it is considered stale. Further analysis revealed that the real problem was not stale connected flows, but stale unconnected flows. LINTS solves this problem by creating an unconnected flows container in a map where the key is the timestamp of the first packet. Flows are removed from the unconnected container when they are connected. Stale unconnected flows can be purged by searching the unconnected container, which is much smaller than the active flows container. Furthermore, since a map is sorted by the key, the iteration may stop once a timestamp inside the time-out value has been reached.

LINTS also implements a deleted flows container that uses a map where the timestamp of the last packet is the key. Since TCP packets are not guaranteed to arrive in the order in which they were sent, sometimes a packet may arrive after the flow has been terminated. The deleted list allows these packets to be captured as part of the recently terminated flow until a delete time-out has been reached and the flow is completely removed. Furthermore, since a map is sorted by the key, the iteration

may stop once a timestamp inside the time-out value has been reached.

3.1.2.2 Magnitude

After flow tracking, the next time-consuming operations are computing the entropy of a packet and comparing all of the N-grams of a packet. The magnitude setting allows the user to instruct LINTS to only process a given number of bytes either for entropy calculation or to search for malicious N-grams.

3.2 Data Sets

Each data set is described in the following sections. More detail for some of the datasets may be found in Smith.[7]

The ARL 2016 and CIC-IDS 2017 data sets were used to test the performance of LINTS compared to Netcomp. Fragments of larger data sets were used because the resource requirements of Netcomp precluded using the entire data sets.

The UNSW-NB15 February and ARL 2017 data sets were used for the blind compression tests. The UNSW-NB15 February data set was chosen because it was similiar but not identical to the UNSW-NB15 January data set. The ARL 2017 data set was chosen because it is the second half of the only real-world data set available.

3.2.1 ARL 2016

The ARL 2016 data set was captured from a US Army Combat Capabilities Development Command (DEVCOM) Army Research Laboratory (ARL) sensor attached to the Defense Research Engineering Network (DREN); it is 2 TB containing 3 days of real-world actual traffic captured on 29 and 30 December 2016.

3.2.2 ARL 2017

The ARL 2017 data set was captured from an ARL sensor attached to the DREN; it is 1.6 TB containing 3 days of real-world actual traffic captured on 3–5 January 2017.

3.2.3 CIC-IDS 2017

In 2017 the Canadian Institute for Cybersecurity (Canadian Institute for Cybersecurity (CIC)) at the University of New Brunswick (UNB) created the Canadian Institute for Cybersecurity Intrusion Detection System (CIC-IDS) 2017 data set. The website https://www.unb.ca/cic/datasets/ids-2017.html includes details about

the data sets and download instructions. This data set, built using the approach described in Sharafaldin et al.,[24] includes web-based, brute force, denial of service (DOS), distributed denial of service (DDOS), infiltration, heartbleed, botnet, and scan attacks.

It contains 5 days of network traffic. The first day contains no malicious traffic and is the training portion. The next 4 days contain malicious traffic and have been combined into the testing portion, which contains 39 GB of network traffic.

3.2.4 Cyber Defense Exercise (CDX) 2009

In 2009, the National Security Agency/Central Security Service (NSA/CSS) conducted an exercise pitting teams from the military academies of the United States and Canada against teams of professional network specialists to see who best defended their network. Data from this exercise were captured and used by Sangster et al. in his efforts to generate labeled data sets.[15] Two network traffic sensors were employed in the exercise: gator-usama010 and gator-usama020. The data captured from the gator-usama010 sensor contains 701 MB and a little over 103 h of network traffic. The data captured from the gator-usama020 sensor contains 23 GB and a little over 72 h of network traffic. The Cyber Defense Exercise (Cyber Defense Exercise (CDX)) data from 2009 is available from https://drive.google.com/open?id=0B0u9Tg7udaAXaUFHRFpQWjR0dW8.

3.2.5 ISCX-IDS 2012

In 2012, the Information Security Centre of Excellence (Information Security Centre of Excellence (ISCX)) at UNB created the ISCX-IDS data set for intrusion detection research.[25] This synthetically labeled data set contains full network capture files. It also contains the HyperText Transfer Protocol (HTTP), Simple Mail Transport Protocol (SMTP), Secure Shell (SSH), Internet Message Access Protocol (IMAP), Post Office Protocol 3 (POP3), and File Transfer Protocol (FTP) network protocols. More details about the data set, including download instructions, may be found at https://www.unb.ca/cic/datasets/ids.html.

The ISCX-IDS data set contains 6 days of network traffic. The first day contains no malicious traffic and is the training portion. The next 5 days contain malicious traffic and have been combined into the testing portion, which contains 51 GB and a little over 144 h of network traffic.

3.2.6 MACCDC 2010

The Mid-Atlantic Collegiate Cyber Defense Competition (MACCDC) conducted a contest in 2010 at the SAIC Conference Center where teams from several universities competed against each other in a cybersecurity competition. During this competition, the teams were required to keep their networks operational while being attacked by several different teams. The scenario for this competition has the student teams working in the emergency operations for a city about to host a major convention. This data set contains 35 GB and a little over 76 h of network traffic. Details about the competition may be found at https://maccdc.org/maccdc-2010/ and the data set may be retrieved from https://www.netresec.com/?page=MACCDC.

3.2.7 MACCDC 2011

The MACCDC conducted a contest in 2011 at the Johns Hopkins University Applied Physics Laboratory where teams from several universities competed against each other in a cybersecurity competition. During this competition the teams were required to keep their networks operational while being attacked by several different teams. The scenario for this competition has the student teams working for a fictious national electric cooperative with operations spread across the country. This data set contains 31 GB and a little over 45 h of network traffic. Details about the competition may be found at https://maccdc.org/maccdc-2011/ and the data set may be retrieved from https://www.netresec.com/?page=MACCDC.

3.2.8 MACCDC 2012

The MACCDC conducted a contest in 2012 at the Johns Hopkins University Applied Physics Laboratory where teams from several universities competed against each other in a cybersecurity competition. During this competition, the teams were required to keep their networks operational while being attacked by several different teams. The scenario for this competition has the student teams working for a fictious hospital with eight regional hospitals. This data set contains 17 GB and a little over 32 h of network traffic. Details about the competition may be found at https://maccdc.org/maccdc-2012/ and the data set may be retrieved from https://www.netresec.com/?page=MACCDC.

3.2.9 UNSW-NB15

Researchers at the University of New South Wales (UNSW) created this data set using the Ixia Perfect Storm tool in the Australian Centre for Cyber Security (ACCS) cyber range laboratory.[26,27] This is a labelled data set containing full network captures. Information about this data set and download instructions may be found at htt ps://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Data sets/.

This data set contains a portion captured in January 2015 and another portion captured in February 2015. The January portion contains 49 GB and a little over 12 h of network traffic. The February portion contains 46 GB and a little over 12 h of network traffic.

3.3 Rule Sets

The initial rule set was the registered rule set downloaded from www.snort.org in August 2013. This rule set was effective with the traffic from the CDX 2009 data set, but worked poorly with the Defense Advanced Research Projects Agency (DARPA) 1998 data set. The more rules that are tested by Snort, the more computing resources Snort requires to complete the analysis. These resources may climb to the point where Snort is unable to keep up with the network traffic causing packet loss. Therefore, the registered rule set from 2013 has most of the rules commented out. In order to analyze older data sets, it was necessary to tailor the rule set to ensure that rules appropriate to the time period are active. Several rule sets were used in this research: Circa2009 is the registered rule set from 2013 with rules appropriate for 2009 activated, RegAug2013 is the registered rule set as downloaded from the Snort website in August 2013, RegJul2018 is the registered rule set as downloaded from the Snort website in July 2018, and RegSep2018 is the registered rule set as downloaded from the Snort website in September 2018.

These rule sets were collected and tailored for and shared with the ARL Packet Loss Study (PLS). In this study, traffic was replayed at increasing multiples of the original speed to induce packet loss. Rules that relied on events per second were problematic because they would fire at increasing rates the faster the traffic was replayed. To avoid these false positives, time-based rules were removed from the active rule set.

3.4 Preparation

The trials conducted for this experiment have been significantly automated using the ARL Cybersecurity Research Data Set Infrastructure (CSRDSI). One of the key components of the CSRDSI is the Data Set Database (DSDB). In order to prepare for the trial, entries must be made in the DSDB. Before LINTS can use the tainted flows strategy, a Bloom filter of malicious N-grams must be constructed.

3.4.1 Data Set Database

The DSDB consists of three tables: the data sets table, the rule sets table, and the alerts table. The data sets table contains relevant information about each data set required to automate trials. The rule sets table contains relevant information about each rule set required to automate trials. The alerts table is a join table of the data set and rule set tables.

3.4.1.1 Data Sets Table

The data sets table contains the following information about each data set: the name, the short name, the location, the number of packets, the duration in seconds, the size in bytes, and the minimum number of seconds required to replay each data set. The name of the data set is chosen to be concise yet recognizable by people familiar with network intrusion data sets (e.g., the name of the CDX 2009 gator-usama010 data set[15] is *cdx2009usama010*). These names usually match the name of the packet capture (PCAP) file where the network stream is stored. The short name of the data set is chosen to be as compact as possible and still remain unique (e.g., the short name of the CDX 2009 gator-usama010 data set is *cdx09u010*). The location is the path to the network capture file in PCAP format relative to the base of the CSRDSI. The *pcapinfo*[7] tool may be used to obtain the packet count, duration, and byte count. The minimum number of seconds required to replay the data set is usually obtained by collecting the duration of running *tcpreplay*[28] at top speed.

3.4.1.2 Rule Sets Table

The rule sets table contains the full name, the short name, and the location of each rule set. The name of the rule set is chosen to be as compact as possible and yet still be recognizable by people familiar with network intrusion rule sets (e.g., the name of the registered Snort rule set downloaded in August 2013 is *RegAug2013*). These names usually match the name of the directory where the rule set is stored. The short name of the rule set is chosen to be as concise as possible and still remain

unique (e.g., the short name of the registered Snort rule set downloaded in August 2013 is *raug13*). The location is the path to the Snort configuration file relative to the base of the CSRDSI.

3.4.1.3 Alerts Table

The alerts table contains the name of the data set, the name of the rule set, and the number of alerts generated when the data set is analyzed by the rule set. The number of alerts is obtained by conducting a control run of Snort against the uncompressed data set and capturing the alerts.

3.4.2 Bloom Filters

There are many ways the malicious N-grams may be captured, hashed, and added to a Bloom filter. Ideally, samples of malicious activity would be collected and used to tune the NIDS rule sets. These samples could then be used to generate the Bloom filter. However, creating and tuning NIDS rule sets are outside the scope of this effort.

For this effort, the malicious Bloom filters were created from the malicious traffic contained in the data sets. Each data set was scrubbed for time warps. (The necessity and procedure for doing so is described in Section 3.4.2.1. The data sets were analyzed by Snort. The Snort alerts were used to separate the malicious traffic from the benign traffic. A Bloom filter was created from the N-grams contained in the benign traffic. Since malicious traffic often contains a considerable amount of benign N-grams, the N-grams extracted from the malicious traffic were first filtered through the benign Bloom filter before being added to the malicious Bloom filter.

3.4.2.1 Time Warps

Since the *pcapurge*[7,8] tools are used to separate the benign from the malicious traffic in uses timestamps, it is important that all of the packets in the network stream be in order. A time warp occurs when the PCAP time stamps in a network stream are out of order. This is especially common in network streams that have been created by combining several smaller network streams. The *pcaptsplit* and *pcaptmerge*[7,8] tools were created to remove time warps from network capture files.

3.4.2.2 Snort Analysis

Each data set was analyzed with Snort using the appropriate rule set. Snort was configured to generate its rules into a comma-separated value (CSV) file. It was

also configured to include the year in each alert.

3.4.2.3 Separate Malicious from Benign Traffic

The *pcappurge* tool was used to separate the malicious traffic from the benign traffic based upon the Snort analysis. The *pcappurge* tool may be configured to collect malicious packets, flows, or sessions where a flow is unidirectional and a session is bidirectional. For these trials, *pcappurge* was configured to collect malicious sessions.

3.4.2.4 Generate Bloom Filter

The *mkbf*[7,8] was used to generate a benign Bloom filter from the benign traffic separated from the network stream using the *pcappurge* tool. The *mkbf* tool supports an arbitrary N-gram size. An N-gram size of 5 was used for these trials. The *mkbf* tool supports several hash functions. The MurMurHash[29] was used for these trials. The *mkbf* tool was then used to create a malicious Bloom filter from the malicious traffic separated from the network stream using the *pcapurge* tool. The N-gram extracted from the malicious traffic was filtered by the Bloom filter created from the benign traffic to remove benign N-grams from the malicious Bloom filter.

3.5 Experimental Control

The experimentation needed to support the work required that several data sets be repeatedly compressed by LINTS under different configurations and analyzed by Snort[30] against a given rule set reporting the number of alerts detected. Collecting all of this data in a timely manner required that the process be highly automated. The root of each experimental direction tree was a directory given the same name as the short name of the data set. The second level contains a directory given the same name as the short name of the rule set. The third level contains a directory for each trial. The fourth level contains a directory for each run. The fifth level contains a directory for each iteration.

Since the first trial uses the compress only TCP strategy, which contains only one run, the second trial will be used as an example. The second trial employs the entropy strategy, and the third level is named *trial2*. Typically, there is only one run contained in a directory, named *run1*. Inside the *run1* directory are the files *lints.conf*, *runlints.conf*, and *sequence.conf*. The *lints.conf* file contains the configuration information for the LINTS. The online manual page for LINTS, which

documents all of the configuration items, is found in Appendix A. This includes the strategies to be used, constant thresholds, whether the flows are to be capped by packets or bytes, and the location of the malicious Bloom filter. The *runlints.conf* file is read by the *runit.sh* script and is typically used to set the independent variable. The code for this script is found in Appendix B. The *sequence.conf* script is used to configure the sequence program to provide values for the independent variable. The *run_lint_exp* program reads the values generated by the *sequence* program. The online manual page for this program, which documents all of the configuration items, is found in Appendix A. The program creates a directory for each value, compresses the data set with the LINTS, and feeds the compressed data set to Snort. The output of LINTS and Snort are written into the directory created for that value. Once the run is complete, data is extracted and stored. The *runit.sh* script determines the data set and rule set from the directory structure and reads the independent variable from the *runlints.config* script; this allows a skeleton directory structure to be copied. The name of the top level is changed to the short name of the data set, the name of the second level is changed to the name of the rule set, and then the *runit.sh* script may be added to the Linux *batch*[31] system from each run directory.

4. Results

This section is divided into three parts. The first part compares the efficiency and effectiveness of LINTS against Netcomp.[7,8] The second part explores the various strategies available in LINTS against several data sets. The third part uses the lessons learned from the second part to configure LINTS to compress two different data sets and examine the results.

In the first part, the results from using Netcomp are compared both for efficiency and effectiveness against the results from LINTS. In this comparison, the 15 June 2012 data from the ISCX 2012 data set and 5 July 2017 data in the CIC-IDS 2017 data set are used.

In the second part, LINTS trials are conducted against the CDX 2009, MACCDC 2010, MACCDC 2011, MACCDC 2012, ISCX 2012, UNSW-NB 2015, ARL 2016, and CIC-IDS 2017 data sets. For each data set, five trials were completed: Trial 1 employs the *Compress only TCP* strategy; This strategy is employed in the rest of the trials. Trial 2 employs the *Entropy* strategy; Trial 3 employs the *Flows* strategy capped-by-packets; Trial 4 employs the *Flows* strategy capped-by-bytes; and Trial

5 employs the *Tainted Flows* strategy keeping malicious flows. Trial 5 has four runs: Run 1 caps flows by packets; Run 2 caps flows by bytes; Run 3 caps flows by packets and excludes high-entropy TCP packets; and Run 4 caps flows by bytes and excludes high-entropy TCP packets.

It is one thing to conduct repeated iterations of LINTS against a data set to discover which settings provide the best compression with an acceptable alert loss rate (ALR). However, this option does not work in the field. A more realistic example would be to configure LINTS based upon experience and use that configuration to compress unknown traffic and examine the result. In the third part, LINTS is configured based upon the lessons learned from the second part; it will then be used to compress the February data from the UNSW-NB 2015 data set and the ARL 2017 data set.

4.1 Comparison against Netcomp

4.1.1 ISCX 2012 Data Set

In this section, the effectiveness and efficiency of Netcomp is compared to the effectiveness and efficiency of LINTS using the file *testbed-15jun.pcap* renamed *iscx120615.pcap* from the Information Security Centre of Excellence Intrusion Detection System (ISCX-IDS) 2012 data set and registered Snort rule set from August 2013.

4.1.1.1 Netcomp ISCX 2012

Figure 4 shows the effect of dropping packets based upon a tainted flow packet threshold on the ISCX 15 June 2012 data set. The ALR is plotted using blue dots, and the compression expressed as the percentage of the original size of the data is plotted in red squares. Table C-1 shows the details for that run. When the tainted flow packet threshold is set to 0.00, the data set is compressed to 88.00% of its original size, and 0.04% of the alerts have been lost. When the tainted flow packet threshold is set to 0.99, the data set is compressed to 31.00% of the original size, and 0.89% of the alerts have been lost.

This run was conducted on 4 April 2019, and the duration or run time for thresholds 0.96–0.99 has been lost. The average run time for each iteration of this run is 13927.644396 s, or 3 h, 52 min, and 7.644396 s. The standard deviation for the run time is 6288.978043 s, or 1h 44 min, and 48.978043 s.

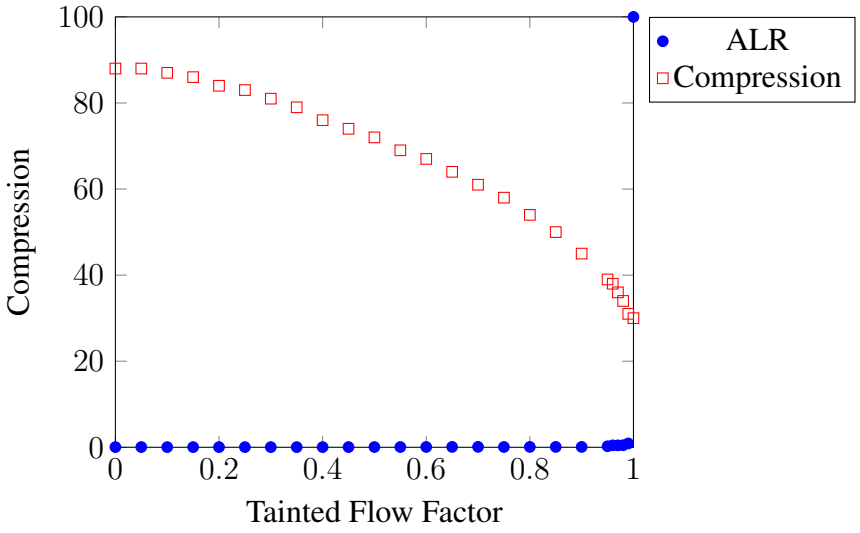

Fig. 4: Netcomp compression using the ISCX 15 June 2012 data set and the RegAug2013 rule set

4.1.1.2 LINTS ISCX 2012

Figure 5 shows the compression and ALR for June 15 of the ISCX 2012 data set. In this run, LINTS is using the tainted session strategy with a packet threshold and an entropy threshold of 7.7. The ALR is plotted using blue dots, and the compression expressed as the percentage of the original size of the data is plotted in red squares. Table C-2 shows the details for that run. When the packet threshold is set to 5, the data set is compressed to 8% of its original size, and 0.28% of the alerts have been lost. When the tainted flow packet threshold is set to 100, the data set is compressed to 42.00% of the original size, and 0.14% of the alerts have been lost.

The average run time for each iteration of this run is 1246.58 s, or 20 min and 46.58 s. The standard deviation for the run time is 156.049 s, or 2 min and 36.049 s.

4.1.2 CIC-IDS 2017

In this section, the effectiveness and efficiency of Netcomp is compared to LINTS using the file *Wednesday-WorkingHours.pcap* renamed *cic170705.pcap* from the CIC-IDS 2017 data set and registered Snort rule set from September 2018.

4.1.2.1 Netcomp Day 3 CIC-IDS 2017

Figure 6 shows the effect of dropping packets based upon a tainted flow packet threshold on the CIC 5 July 2018 data set. The ALR is plotted using blue dots,

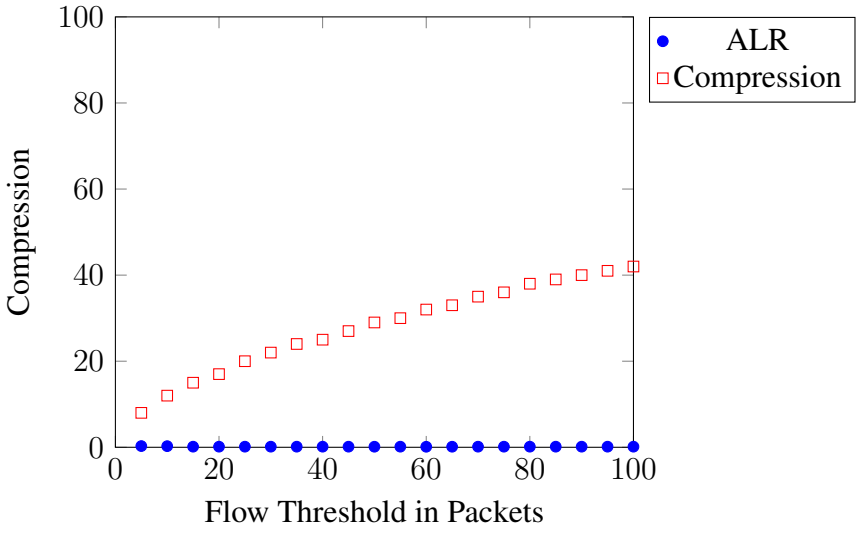

Fig. 5: LINTS compression using the ISCX 15 June 2012 data set and the RegAug2013 rule set

and the compression expressed as the percentage of the original size of the data is plotted in red squares. Table C-3 displays the details from that run. When the tainted flow packet threshold is set to 0.80, the data set is compressed to 36.00% of its original size, and 0.00% of the alerts have been lost. When the tainted flows packet threshold is set to 0.90, the data set is compressed to 34.00% of the original size, and 0.94% of the alerts have been lost.

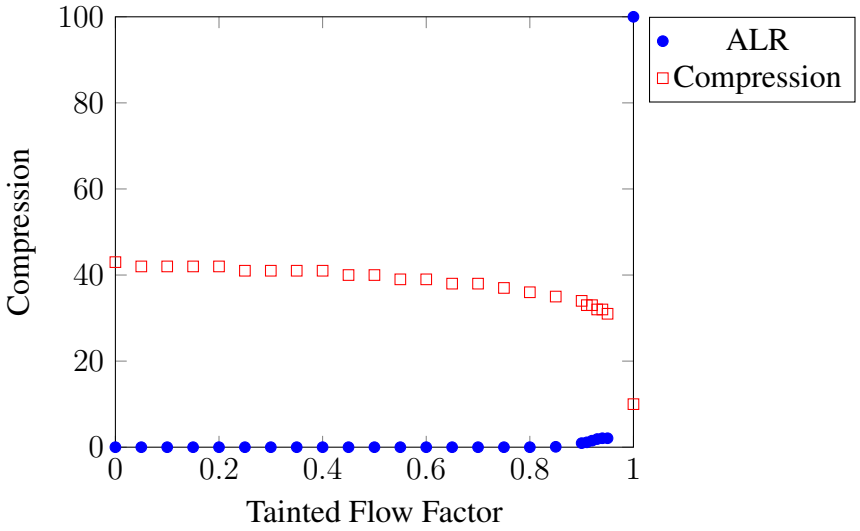

Fig. 6: Netcomp compression using the CIC 5 July 2018 data set and the RegSep2018 rule set

The average run time for each iteration of this run is 4581.432512 s, or 1 h, 16 min, and 21.432512 s. The standard deviation for the run time is 127.364286 s, or 2 min and 7.364286 s.

4.1.2.2 LINTS Day 3 CIC-IDS 2017

Figure 7 shows the compression and ALR for day 3 of the CIC-IDS 2017 data set. In this run, LINTS is using the tainted session strategy with a packet threshold and an entropy threshold of 7.7. The ALR is plotted using blue dots, and the compression expressed as the percentage of the original size of the data is plotted in red squares. Table C-4 shows the details for that run. When the packet threshold is set to 10, the data set is compressed to 20% of its original size, and 0.00% of the alerts have been lost. When the tainted flows packet threshold is set to 100, the data set is compressed to 22.00% of the original size, and 0.00% of the alerts have been lost.

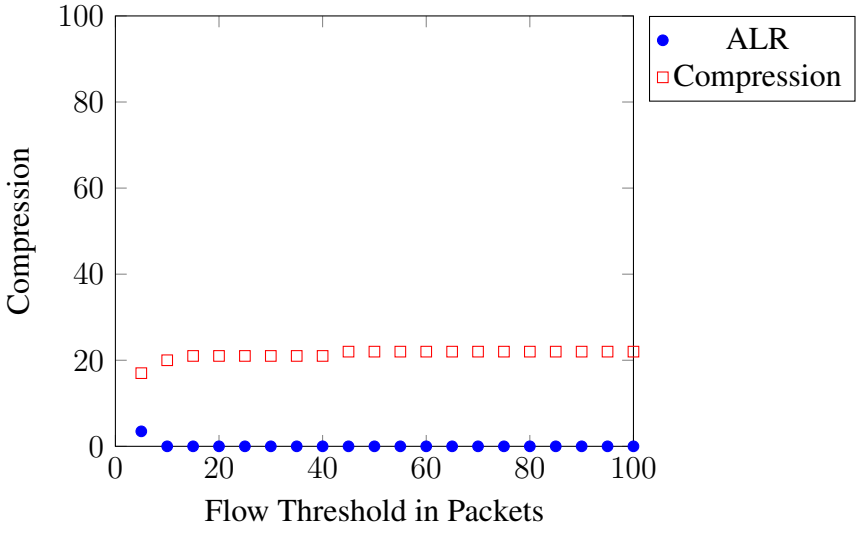

Fig. 7: LINTS compression using day 3 of the CIC-IDS 2017 data set and the RegSep2018 rule set

The average run time for each iteration of this run is 336.029 s, or 5 min and 36.029 s. The standard deviation for the run time is 156.049 s, or 2 min and 19.7468 s.

4.1.3 Netcomp vs. LINTS Observations

This section compares the effectiveness and efficiency of Netcomp and LINTS. The effectiveness is measured in the compression and ALR. The efficiency is measured in duration in seconds.

The best compression that Netcomp was able to obtain against the 15 June day from the ISCX 2012 data set was 31% of the original size at 0.89% ALR. LINTS was able to compress this data set to 8% of the original size with only 0.28% ALR. LINTS is able to compress the 5 June file from ISCX-IDS 2012 data set 3.9 times more than Netcomp while losing 3.2 time fewer alerts. It took Netcomp an average of 3 h, 52 min, and 8 s to process this data set. It took LINTS an average of 20 min and 47 s to process this data set. LINTS is also able to process this data set in 9% of the time required for Netcomp, or a little more than 10 times less.

The best compression that Netcomp was able to obtain against day 3 of the CIC-IDS 2017 data set was 43% of the original size at 0.00% ALR. LINTS was able to compress this data set to 20% of the original size with only 0.00% ALR. LINTS was able to compress day 3 of the CIC-IDS 2017 data set a little less than half the time required by Netcomp. It took Netcomp an average of 1 h, 16 min, and 21 s to process this data set. It took LINTS an average of 2 min and 20 s to process this data set. LINTS was also able to process this data set in 7% of the time required for Netcomp, or a little more than 13 times less.

LINTS outperforms Netcomp in the compression of traffic without the loss of alerts by 4 and 2 times. LINTS outperforms Netcomp in efficiency being 10 and 13 times faster.

4.2 LINTS Trials

This section describes the LINTS trials that were conducted against each data set. These trials were conducted to better understand how the various compression strategies perform against the various data sets. This information will be used to configure LINTS when performing the blind experiments reported in Section 4.2.9.

4.2.1 Compress Only TCP

In these trials, all of the TCP data was compressed out of the network stream. This was done to better understand how much of the network stream consisted of packets outside of TCP and how many alerts were contained in that traffic.

Listing 1 displays the *lints.conf* file for these trials. Line #1, *elapsedtime_format*, is set to display the elapsed time in seconds and microseconds. This simplifies the extraction and processing of duration data. Line #2 tells LINTS to add the Compress only TCP strategy. Since no other strategies have been added, all TCP traffic will

be compressed out of the network stream.

Listing 1: The *lints.conf* file for Trial 1, Run 1

```
1 elapsedtime_format = %S
2 keepnontcp
```

Listing 2 displays the *runlints.conf* file for these trials. The *runlints.sh* script sources this configuration file, executing the command here. The comments illustrate that this strategy does not have an independent variable because traffic either is TCP or it is not.

Listing 2: The *runlints.conf* file for Trial 1, Run 1

```
1 #
2 # This trial tests the keeptcp strategy.
3 # This strategy does not have an independent
4 # variable.
5 # This also explains why there is only one
6 # interation.
7 #
```

Listing 3 displays the *sequence.conf* file for these trials. Line #1 sets the algorithm to *arithmetic*, which implements Eq. 3 where n moves from *start* to *start* plus *limit*, and each s_n is equal to some constant a, which is set to zero by default plus the *difference*, d, in this case set to 0.5 times $n - 1$. When using this configuration file, the *sequence* program will produce a single number being 0.5.

$$s_n = a + d(n - 1) \quad . \tag{3}$$

Listing 3: The *sequence.conf* file for Trial 1 Run 1

```
1 type = arithmetic
2 difference = 0.5
3 limit = 1
4 start = 2
```

As seen in Table 1, the traffic other than TCP seldom accounts for more than 15% of the network stream. Often it accounts for very few alerts, but occasionally it

Table 1: Results from the Compress only TCP strategy

Data Set	Size (%)	ALR (%)
cdx09u010	11.00	98.22
cdx09u020	15.00	19.77
maccdc10	1.00	74.15
maccdc11	5.00	99.99
maccdc12	1.00	99.99
iscx12	0.00	100.00
unsw15jan	0.00	100.00
arl16	9.00	33.50
cic17te	0.00	100.00

can account for a large percentage of the alerts. Based upon this assessment, it is considered reasonable to focus on compressing the TCP traffic and keeping the rest.

4.2.2 Entropy

In these trials, the entropy strategy is applied to various data sets against relevant rule sets. This was done to explore the amount of compressed and encrypted data in the network stream and the number of alerts contained in that portion of the network stream. The ALR verse the compression is plotted for several data sets against relevant rule sets in Fig. 8–Fig. 16. The compression is reported as the percentage of the original size of the network traffic. The raw data from these runs may be seen in Table C-5–Table C-12.

Listing 4 displays the *lints.conf* file for these trials. Line #1, *elapsedtime_format*, is set to display the elapsed time in seconds and microseconds. This simplifies the extraction and processing of duration data. Line #2 tells LINTS to add the Compress only TCP strategy. Line #3 tells LINTS to add the *keep* strategy, which instructs it to keep any traffic that has not already been marked as benign. Line #4 sets the threshold to 0.5.

Listing 4: The *lints.conf* file for Trial 2, Run 1

```
1 elapsedtime_format = %S
2 keepnontcp
3 keep
4 threshold = 0.5
```

Listing 5 displays the *runlints.conf* file for these trials. The *runlints.sh* script sources this configuration file, executing the command here. This file sets the independent variable to be *-e*. This means that the values generated by the *sequence* program will be placed after the *-e* option in *lints*, setting the entropy threshold.

Listing 5: The *runlints.conf* file for Trial 2 Run 1

```
1 #
2 # This trial tests the entropy strategy.
3 # This strategy used the entropy threshold
4 # as the independent variable.
5 #
6
7 IV=-e
```

Listing 6 displays the *sequence.conf* file for these trials. Line #1 sets the algorithm to *arithmetic*, which implements Eq. 3. It configures *sequence* to generate numbers from 5.0 to 8.0 in increments of 0.1.

Listing 6: The *sequence.conf* file for Trial 2, Run 1

```
1 type = arithmetic
2 difference = 0.1
3 limit = 31
4 start = 51
```

Sometimes removing the high-entropy packets from the network data stream can increase the number of Snort alerts. This was especially true for the MACCDC 2011, MACCDC 2012, and the ISCX 2012 data sets. With the exception of the ISCX 2012 data set, this dip below zero resolves itself by an entropy value of 7.7. The alerts generated at 7.7 for these data sets have been compared against the control to ensure that they are the same or at least a very similar set of alerts.

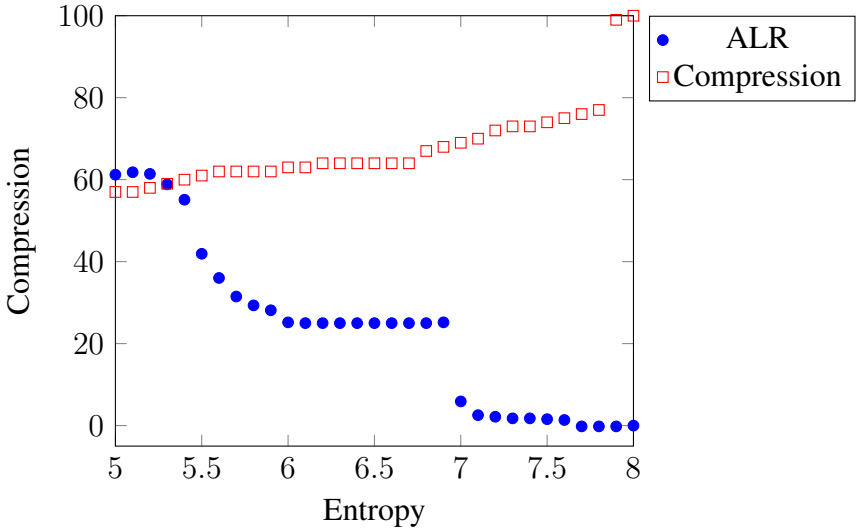

Fig. 8: LINTS compression using the entropy strategy on the CDX 2009 gator-usama010 data set as analyzed by the Circa2009 rule set

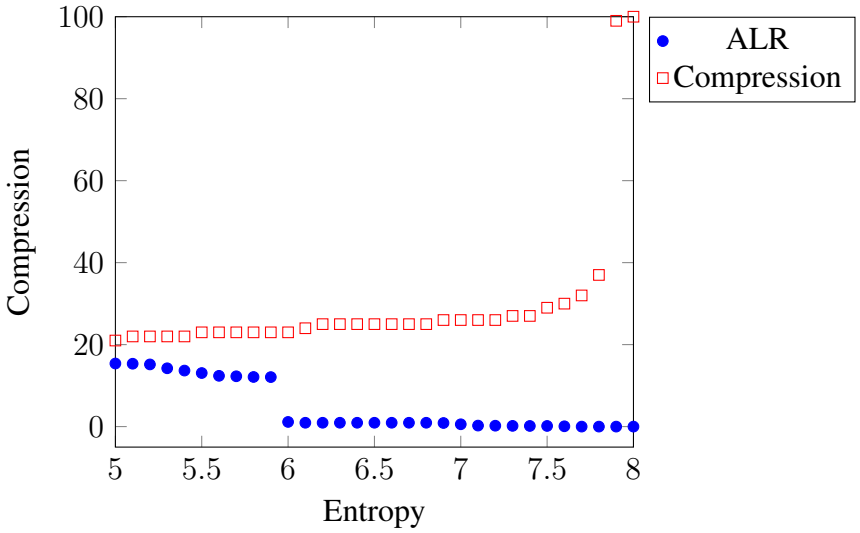

Fig. 9: LINTS compression using the entropy strategy on the CDX 2009 gator-usama020 data set as analyzed by the Circa2009 rule set

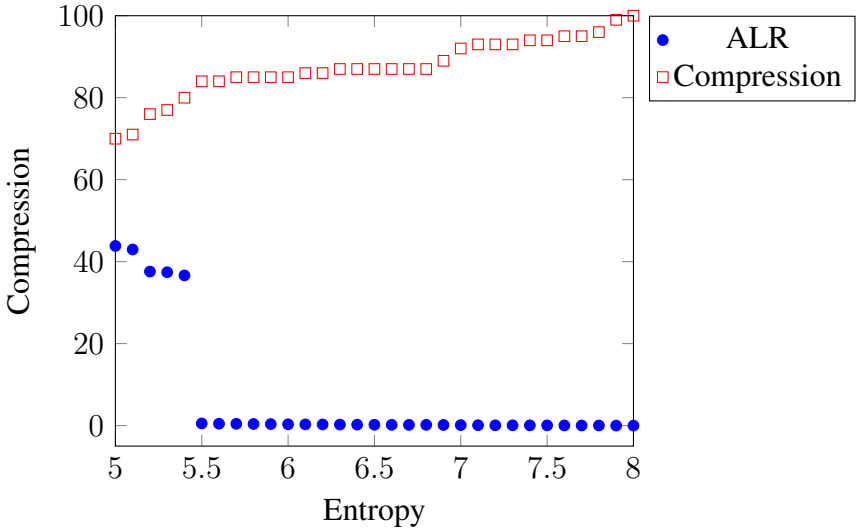

Fig. 10: LINTS compression using the entropy strategy on the MACCDC 2010 testing data set as analyzed by the Circa 2009 rule set

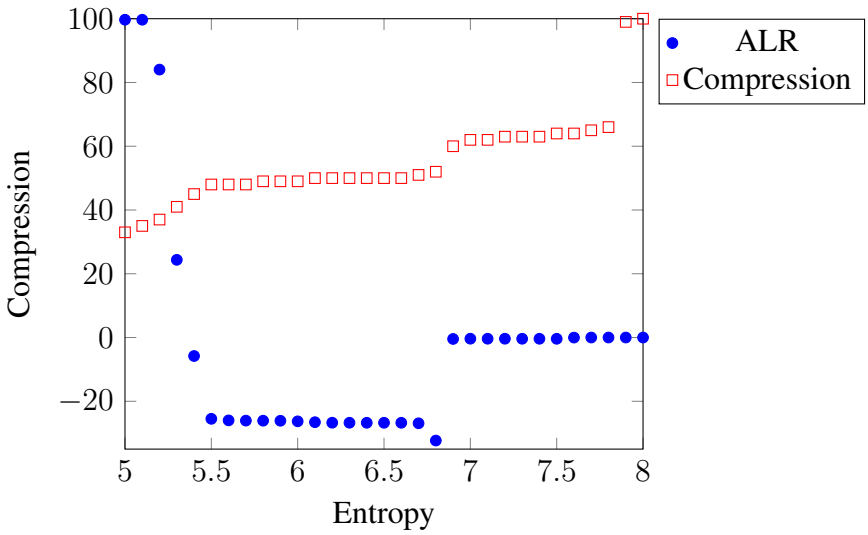

Fig. 11: LINTS compression using the entropy strategy on the MACCDC 2011 testing data set as analyzed by the RegAug2013 rule set

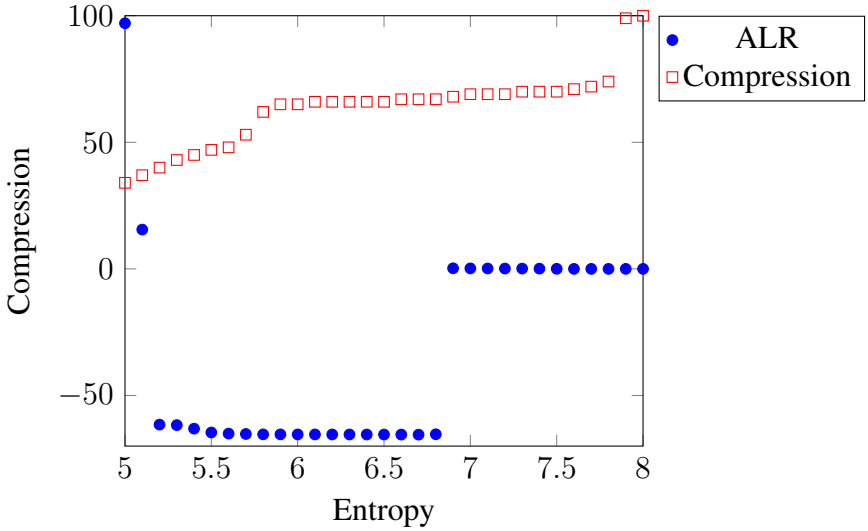

Fig. 12: LINTS compression using the entropy strategy on the MACCDC 2012 testing data set as analyzed by the RegAug2013 rule set

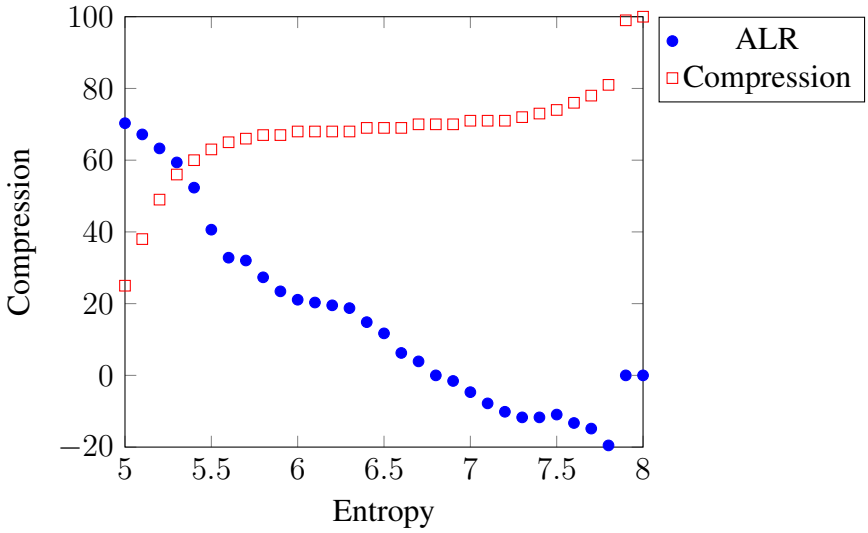

Fig. 13: LINTS compression using the entropy strategy on the ISCX 2012 testing data set as analyzed by the RegAug2013 rule set

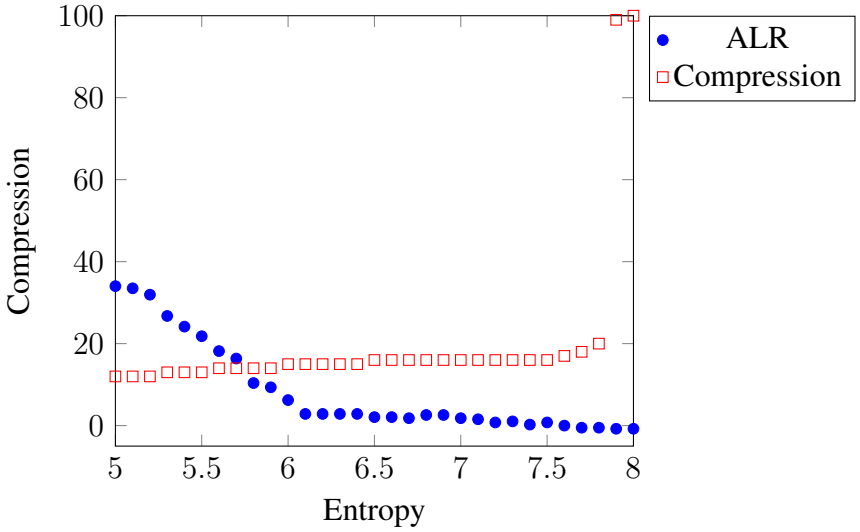

Fig. 14: LINTS compression using the entropy strategy on the UNSW-NB 2015 January data set as analyzed by the RegSep2018 rule set

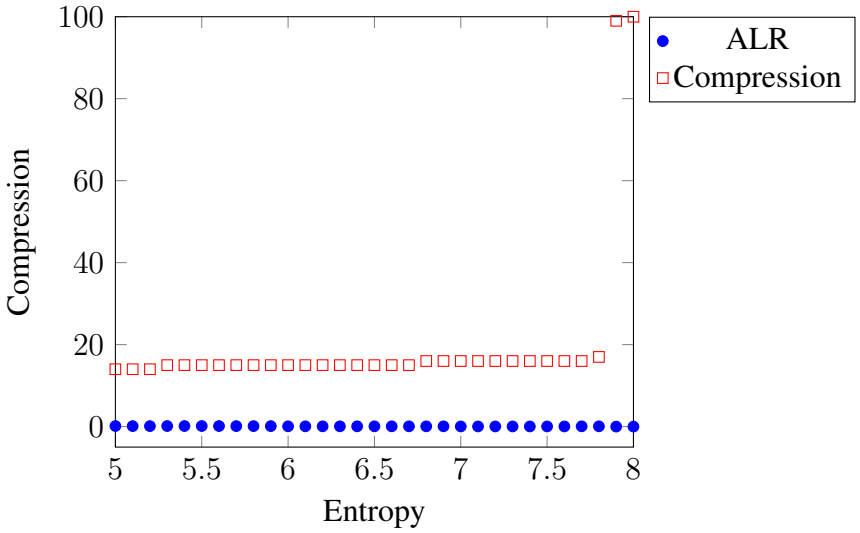

Fig. 15: LINTS compression using the entropy strategy on the ARL2016 data set as analyzed by the RegSep2018 rule set

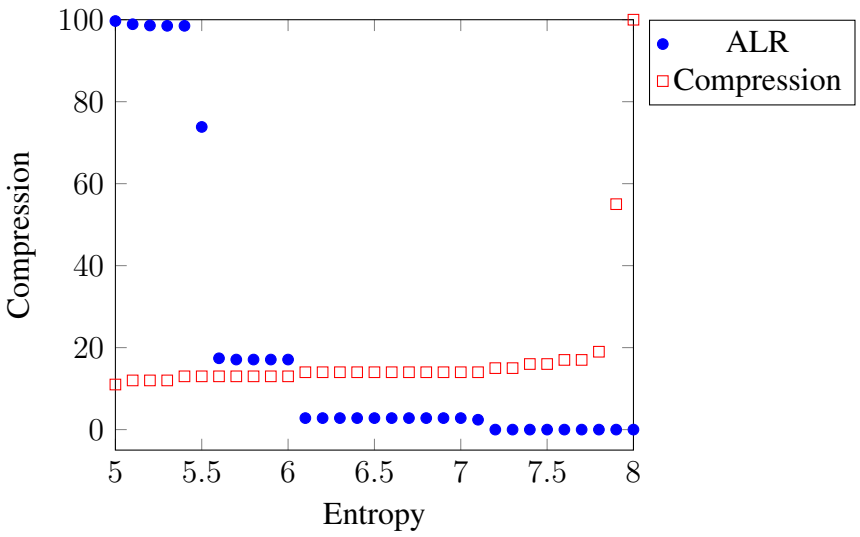

Fig. 16: LINTS compression using the entropy strategy on the CIC 2017 testing data set as analyzed by the RegJul2018 rule set

These experiments demonstrate that although eliminating high-entropy data may not always provide a significant amount of compression, it rarely introduces a significant amount of alert loss. Smith et al.[3] discovered that files with an entropy of over 7.0 were either compressed or encrypted. Since bringing compressed or encrypted data from the sensor to the CAS is usually not very helpful, entropy may be used to eliminate this traffic. Table 2 shows that at an entropy of 7.7, there is little alert loss but there can be significant compression.

Table 2: Results from the entropy strategy

Data Set	Size (%)	ALR (%)
cdx09u010	76.00	-0.19
cdx09u020	32.00	0.00
maccdc10	95.00	0.03
maccdc11	65.00	0.00
maccdc12	72.00	0.00
iscx12	78.00	-14.84
unsw15jan	18.00	-0.51
arl16	16.00	0.07
cic17te	17.00	0.00

4.2.3 Capped by Packets

Previously Smith and Hammell[6] theorized that malicious flows would show themselves to be malicious early. In these trials, the capped-by-packet strategy is applied to various data sets against relevant rule sets. This was done to explore the depth of the flows in the network stream and the positions of the alerts inside the flows in the network stream. The ALR verse the compression is plotted for several data sets against relevant rule sets in Figs. 17–25. The raw data may be found in Tables C-13–C-18.

Listing 7 displays the *lints.conf* file for these trials. Line #1, *elapsedtime_format*, is set to display the elapsed time in seconds and microseconds. This simplifies the extraction and processing of duration data. Line #2 sets the *flow_feature* to *packets*. This instructs LINTS to cap flow by the number of packets. Line #3 tells LINTS to add the Compress only TCP strategy.

Listing 8 displays the *runlints.conf* file for these trials. The *runlints.sh* script sources this configuration file, executing the command here. This file sets the independent

Listing 7: The *lints.conf* file for Trial 3, Run 1

```
1 elapsedtime_format = %S
2 flow_feature = packets
3 keepnontcp
```

variable to be *-C*, meanining that the values generated by the *sequence* program will be placed after the *-C* option in *lints*, setting the flow threshold.

Listing 8: The *runlints.conf* file for Trial 3, Run 1

```
1 #
2 # This trial used the flow threshold as the
3 # independent variable.
4 #
5
6 IV=-C
```

Listing 9 displays the *sequence.conf* file for these trials. Line #1 sets the algorithm to *arithmetic*, which implements Eq. 3. It configures *sequence* to generate numbers from 5 to 100 in increments of 5. For most of the data set, 100 packets was enough to get to a reasonable level of packets loss; however, for the trials with MACCDC 2011 and MACCDC 2012, the limit was increased to 40 to allow for 200 packets.

Listing 9: The *sequence.conf* file for Trial 3, Run 1

```
1 type = arithmetic
2 difference = 5
3 limit = 20
4 start = 2
```

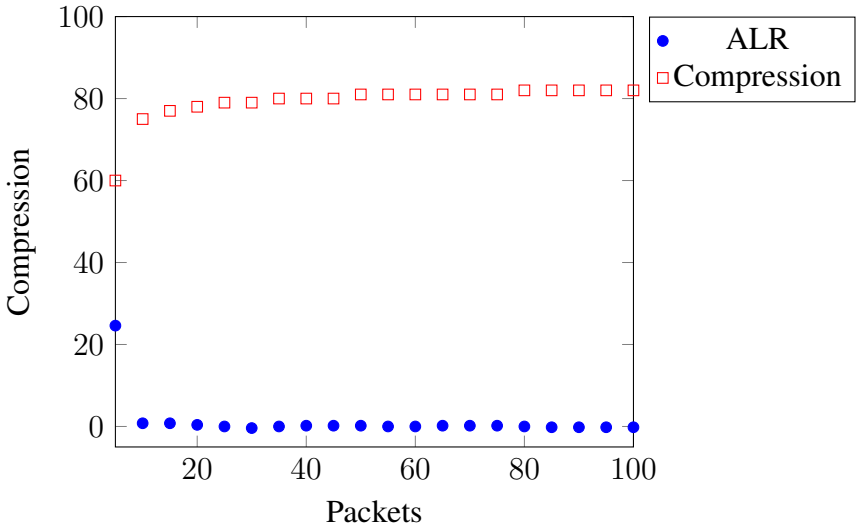

Fig. 17: LINTS compression using the capped-by-packets strategy on the CDX 2009 gator-usama010 data set as analyzed by the Circa2009 rule set

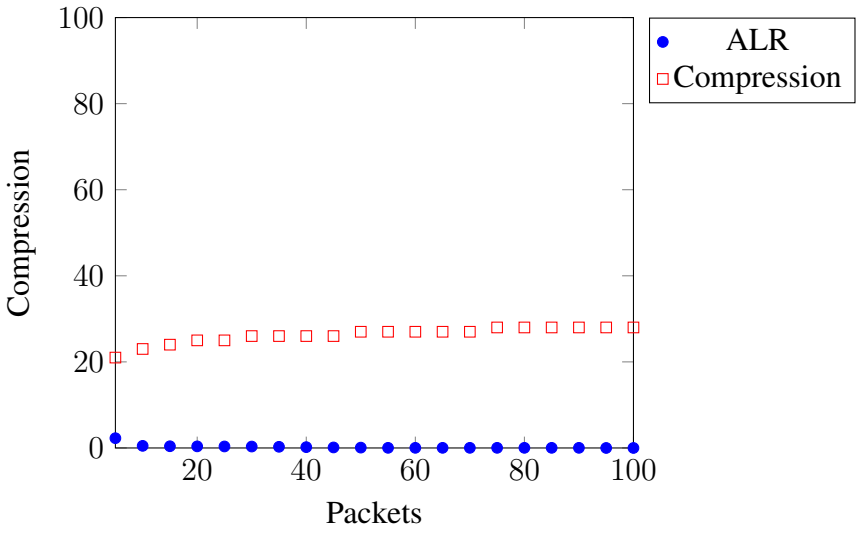

Fig. 18: LINTS compression using the capped-by-packets strategy on the CDX 2009 gator-usama020 data set as analyzed by the Circa2009 rule set

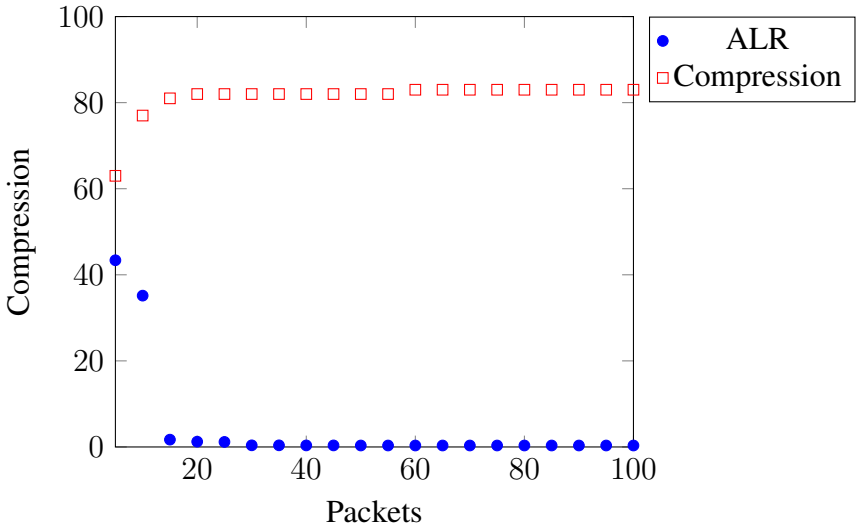

Fig. 19: LINTS compression using the capped-by-packets strategy on the MACCDC 2010 data set as analyzed by the Circa 2009 rule set

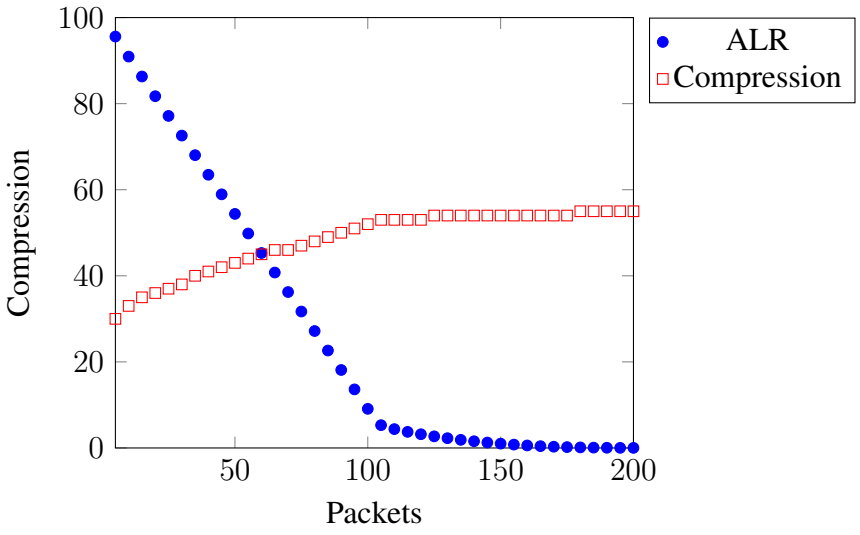

Fig. 20: LINTS compression using the capped-by-packets strategy on the MACCDC 2011 data set as analyzed by the RegAug2013 rule set

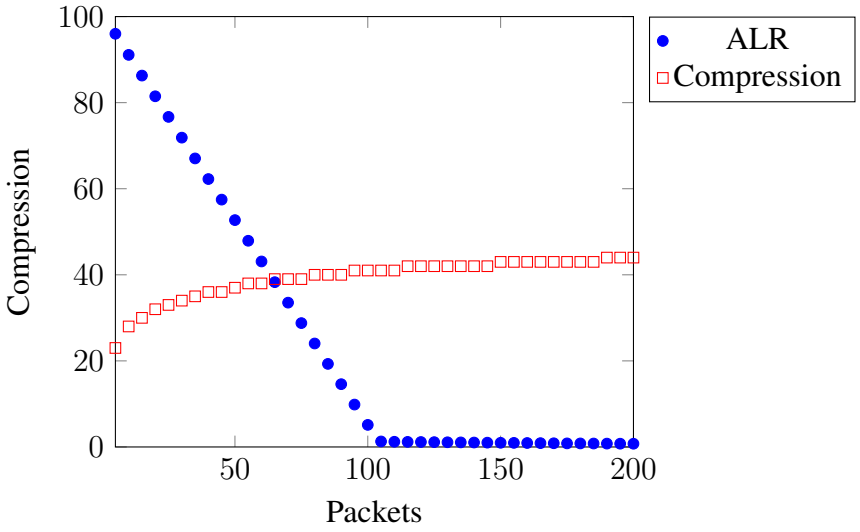

Fig. 21: LINTS compression using the capped-by-packets strategy on the MACCDC 2012 data set as analyzed by the RegAug2013 rule set

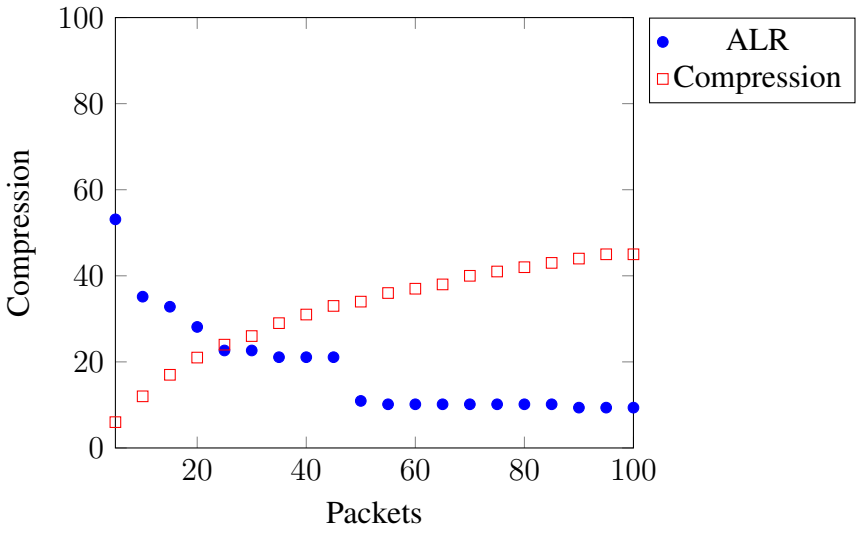

Fig. 22: LINTS compression using the capped-by-packets strategy on the ISCX 2012 testing data set as analyzed by the RegAug2013 rule set

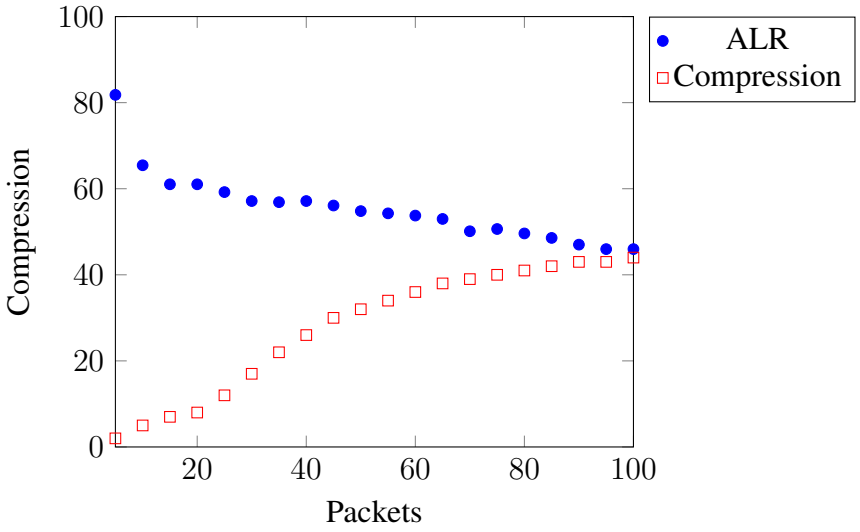

Fig. 23: LINTS compression using the capped-by-packets strategy on the UNSW-NB 2015 January data set as analyzed by the RegSep2018 rule set

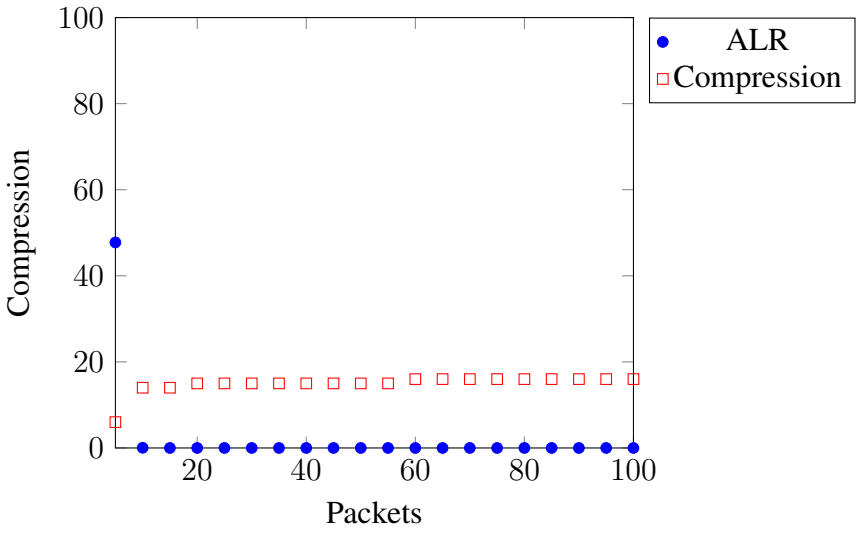

Fig. 24: LINTS compression using the capped-by-packets strategy on the ARL 2016 data set as analyzed by the RegSep2018 rule set

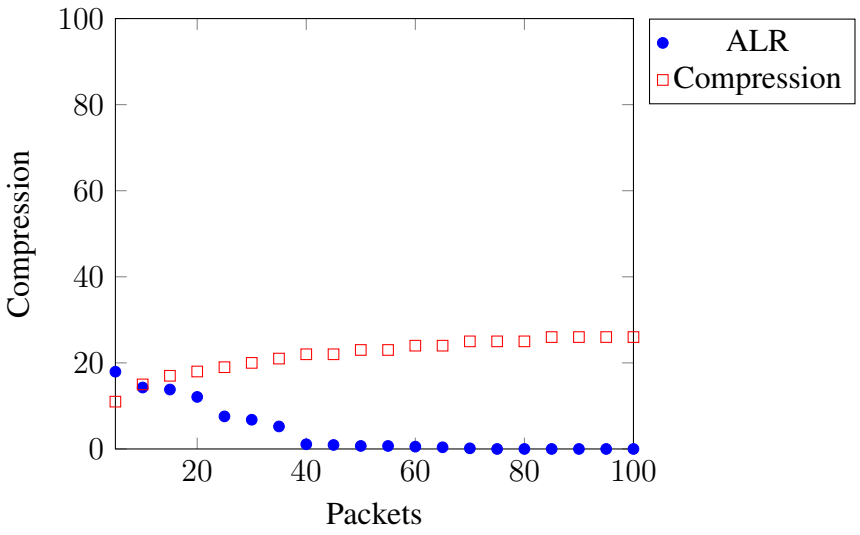

Fig. 25: LINTS compression using the capped-by-packets strategy on the CIC 2017 testing data set as analyzed by the RegJul2018 rule set

Assuming that a 1% ALR is acceptable, Table 3 shows the depth in packets, percent of original size, and ALR when this first crosses below 1%. These results would disprove the theory that malicious packets happen early in a network flow. However, for three of the data sets, this strategy was very effective. Of particular importance is that the one real-world data set is one where this strategy was effective. Unlike the entropy strategy, there does not seem to be a single packet cap that will provide an acceptable ALR for all data sets.

Table 3: Results from the capped-by-packets strategy

Data Set	Packet Count	Size (%)	ALR (%)
cdx09u010	10	75.00	0.78
cdx09u020	10	23.00	0.49
maccdc10	30	82.00	0.38
maccdc11	155	54.00	0.78
maccdc12	100	41.00	5.13
iscx12	100	45.00	9.37
unsw15jan	100	44.00	45.97
arl16	10	14.00	0.04
cic17te	45	22.00	0.93

4.2.4 Capped by Bytes

In these trials, the capped-by-bytes strategy is applied to various data sets against relevant rule sets. This was done to explore the size of the flows in the network stream and the positions of the alerts inside the flows in the network stream. The ALR verse the compress is plotted for several data sets against relevant rule sets in Figs. 26–34. The raw data may be found in Tables C-19–C-24.

Listing 10 displays the *lints.conf* file for these trials. Line #1, *elapsedtime_format* is set to display the elapsed time in seconds and microseconds. This simplifies the extraction and processing of duration data. Line #2 sets the *flow_feature* to *bytes*. This instructs LINTS to cap flow by the number of bytes. Line #3 tells LINTS to add the Compress only TCP strategy.

Listing 11 displays the *runlints.conf* file for these trials. The *runlints.sh* script sources this configuration file, executing the command here. This file sets the independent variable to be *-C*, meaning that the values generated by the *sequence* program will be placed after the *-C* option in *lints* setting the flow threshold.

Listing 10: The *lints.conf* file for Trial 4, Run 1

```
1 elapsedtime_format = %S
2 flow_feature = bytes
3 keepnontcp
```

Listing 11: The *runlints.conf* file for Trial 4, Run 1

```
1 #
2 # This trial used the flow threshold as the
3 # independent variable.
4 #
5
6 IV=-C
```

Listing 12 displays the *sequence.conf* file for these trials. Line #1 sets the algorithm to *arithmetic*, which implements Eq. 3. It configures *sequence* to generate numbers from 500 to 10,000 in increments of 500. For most of the data sets, 10,000 bytes was enough to get to a reasonable level of alert loss; however, for the trials with MACCDC 2011 and MACCDC 2012, the limit was increased to allow for 40,000 bytes.

Listing 12: The *sequence.conf* file for Trial 4, Run 1

```
1 type = arithmetic
2 difference = 500
3 limit = 20
4 start = 2
```

Assuming that a 1% ALR is acceptable, Table 4 shows the depth in packets, percent of original size, and ALR when this first crosses below 1%. As expected, these results mirror the results from capping by packets. This strategy seems to work very well for some data sets and poorly for others. Also, as with the capped by packet strategy, there does not seem to be a single data cap that will give an acceptable ALR for all data sets.

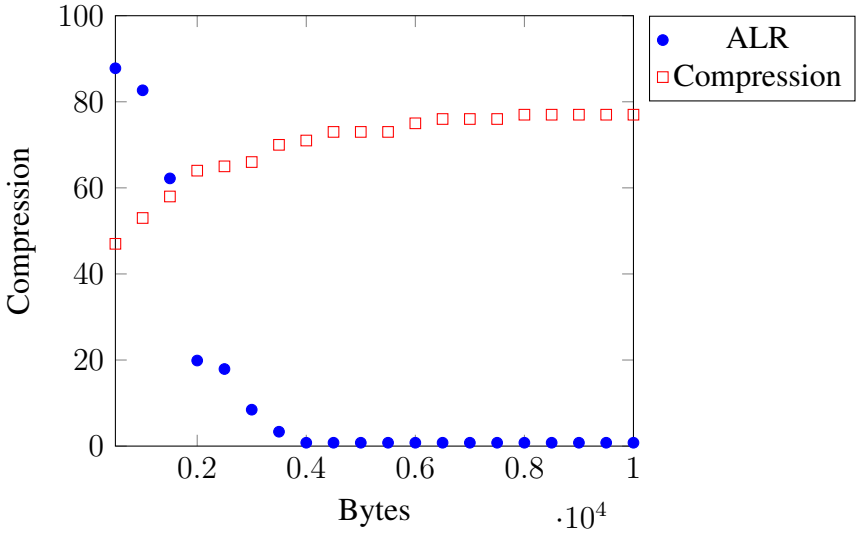

Fig. 26: LINTS compression using the capped-by-bytes strategy on the CDX 2009 gator-usama010 data set as analyzed by the Circa2009 rule set

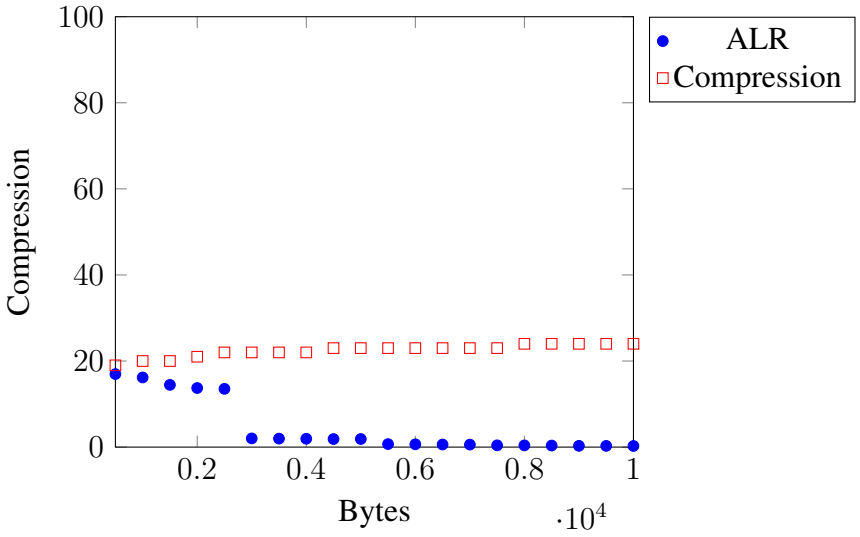

Fig. 27: LINTS compression using the capped-by-bytes strategy on the CDX 2009 gator-usama020 data set as analyzed by the Circa2009 rule set

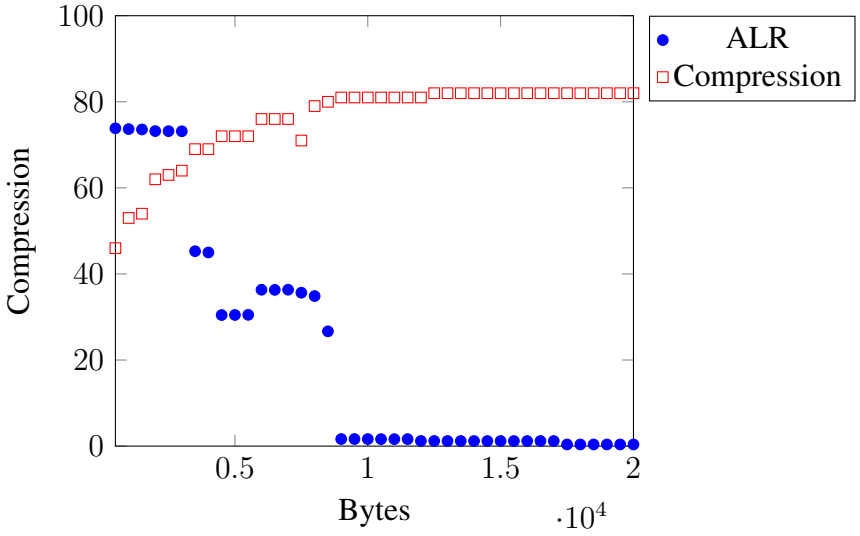

Fig. 28: LINTS compression using the capped-by-bytes strategy on the MACCDC 2010 data set as analyzed by the Circa2009 rule set

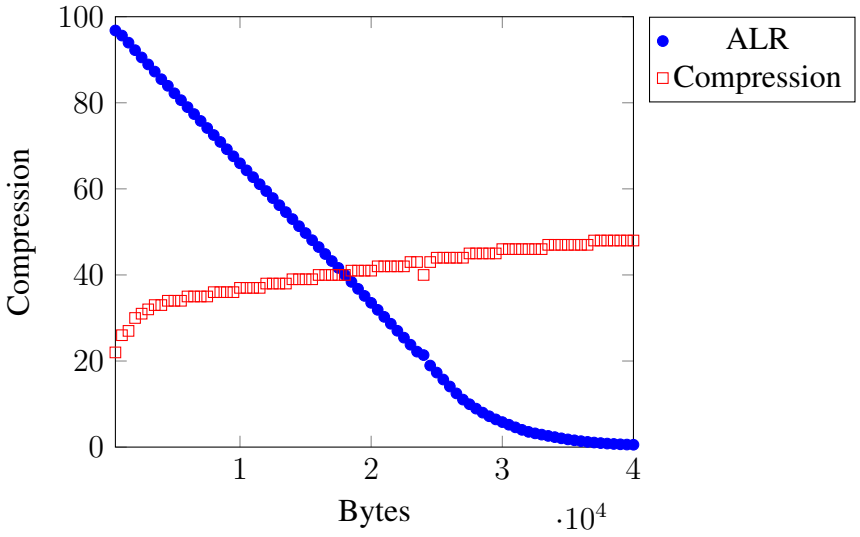

Fig. 29: LINTS compression using the capped-by-bytes strategy on the MACCDC 2011 data set as analyzed by the RegAug2013 rule set

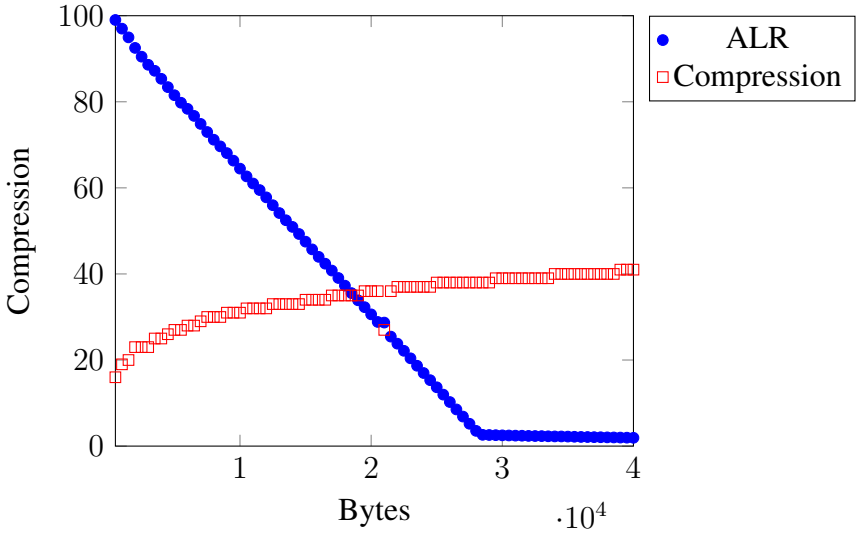

Fig. 30: LINTS compression using the capped-by-bytes strategy on the MACCDC 2012 data set as analyzed by the RegAug2013 rule set

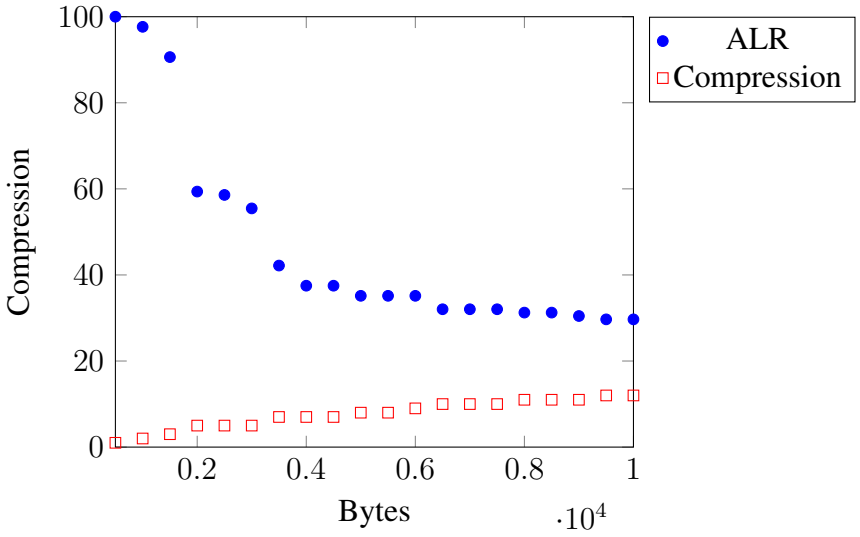

Fig. 31: LINTS compression using the capped-by-bytes strategy on the ISCX 2012 testing data set as analyzed by the RegJul2018 rule set

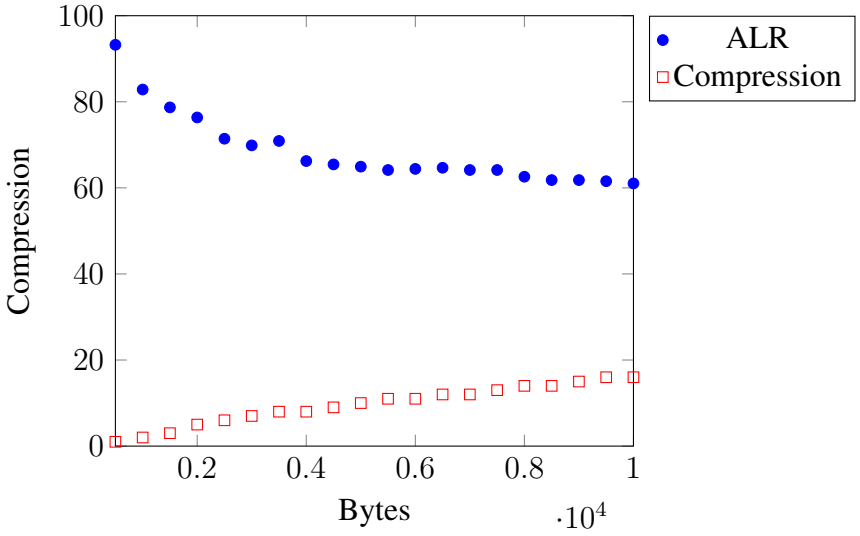

Fig. 32: LINTS compression using the capped-by-bytes strategy on the UNSW-NB 2015 January data set as analyzed by the RegSep2018 rule set

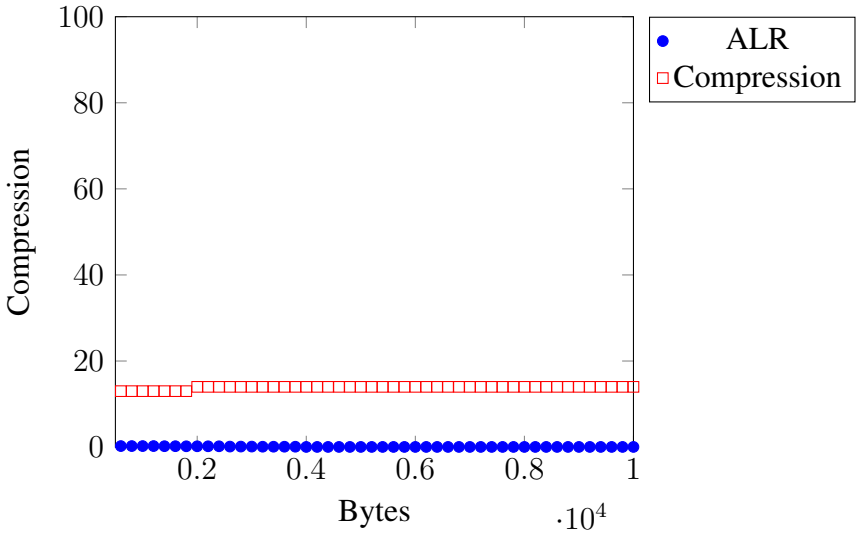

Fig. 33: LINTS compression using the capped-by-bytes strategy on the ARL 2016 data set as analyzed by the RegJul2018 rule set

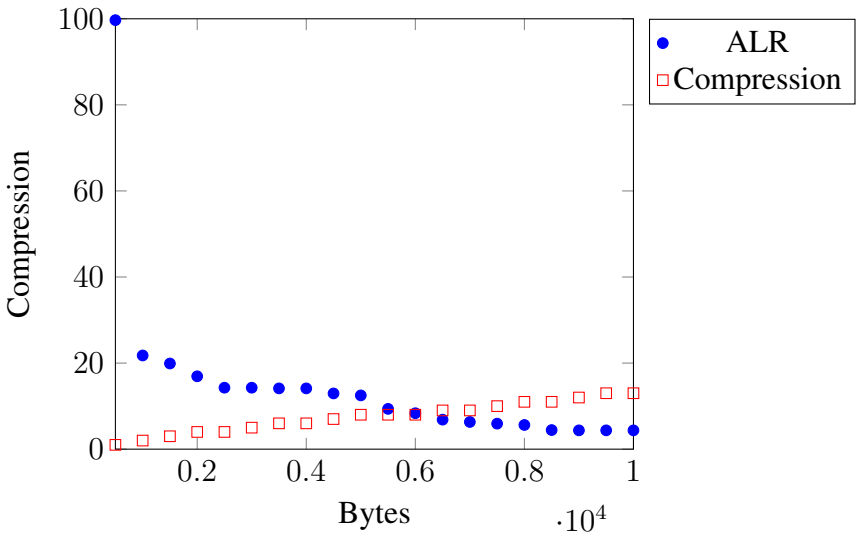

Fig. 34: LINTS compression using the capped-by-bytes strategy on the CIC 2017 testing data set as analyzed by the RegJul2018 rule set

Table 4: Results from the capped-by-bytes strategy

Data Set	Packet Count	Size (%)	ALR (%)
cdx09u010	4500	71.00	0.78
cdx09u020	5500	23.00	0.70
maccdc10	17500	82.00	0.38
maccdc11	20000	41.00	33.52
maccdc12	10000	31.00	64.47
iscx12	10000	12.00	29.68
unsw15jan	10000	16.00	61.03
arl16	400	6.00	0.24
cic17te	10000	13.00	4.37

4.2.5 Tainted Flow Capped by Packets

In these trials, the combination of the tainted flow capped-by-bytes strategy is applied to various data sets against relevant rule sets. This was done to explore effectiveness of these strategies in compressing network traffic for network intrusion detection applications. The ALR verse the compress is plotted for several data sets against relevant rule sets in Figs. 35–43. The raw data may be found in Tables C-25–C-30.

Listing 13 displays the *lints.conf* file for these trials. Line #1, *elapsedtime_format* is set to display the elapsed time in seconds and microseconds. This simplifies the extraction and processing of duration data. Line #2 set the Bloom filter to *malicious.bflt* in the parent directory. Line #3 sets the *flow_feature* to *packets*. This instructs LINTS to stop comparing the N-grams in the payload to the Bloom filter and stop keeping packets in untainted flows when the packet count is greater than the threshold. Line #3 tells LINTS to add the Compress onlyTCP strategy.

Listing 13: The *lints.conf* file for Trial 5, Run 1

```
1 elapsedtime_format = %S
2 filter = ../malicious.bflt
3 flow_feature = packets
4 keepnontcp
```

Listing 14 displays the *runlints.conf* file for these trials. The *runlints.sh* script sources this configuration file, executing the command here. This file sets the independent variable to be *-C*, meaning that the values generated by the *sequence* program will be placed after the *-C* option in *lints* setting the flow threshold.

Listing 14: The *runlints.conf* file for Trial 5, Run 1

```
1 #
2 # This trial used the flow threshold as the
3 # independent variable.
4 #
5
6 IV=-C
```

Listing 15 displays the *sequence.conf* file for these trials. Line #1 sets the algorithm to *arithmetic*, which implements Eq. 3. It configures *sequence* to generate numbers

from 5 to 100 in increments of 5. For most of the data sets, 100 bytes was enough to get to a reasonable level of alert loss. The number of iterations was increased to include 730 packets for the UNSW-NB 2015 January data set. Only three iterations were captured for the ARL 2016 data set; however, that was more than enough trials necessary to get an ALR less than one.

Listing 15: The *sequence.conf* file for Trial 5, Run 1

```
1 type = arithmetic
2 difference = 5
3 limit = 20
4 start = 2
```

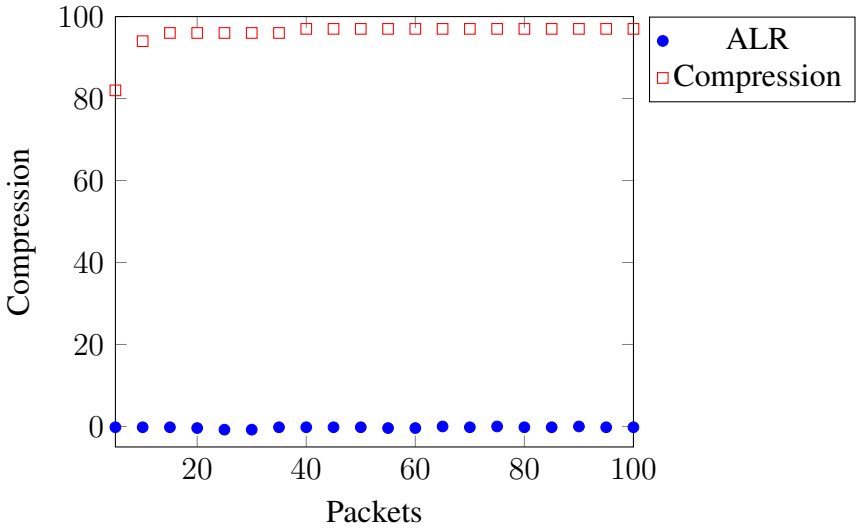

Fig. 35: LINTS compression using the tainted flow capped-by-packets strategy on the CDX 2009 gator-usama010 data set as analyzed by the Circa2009 rule set

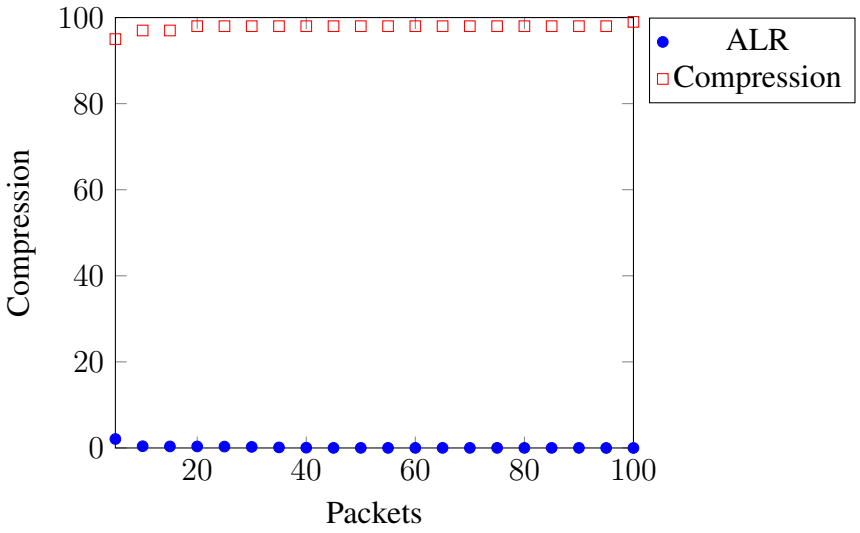

Fig. 36: LINTS compression using the tainted flow capped-by-packets strategy on the CDX 2009 gator-usama020 data set as analyzed by the Circa2009 rule set

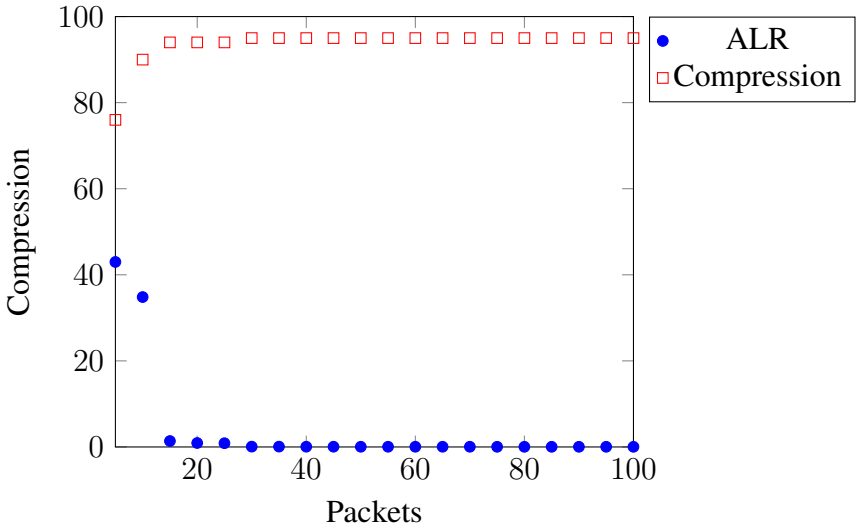

Fig. 37: LINTS compression using the tainted flow capped-by-packets strategy on the MACCDC 2010 data set as analyzed by the Circa 2009 rule set

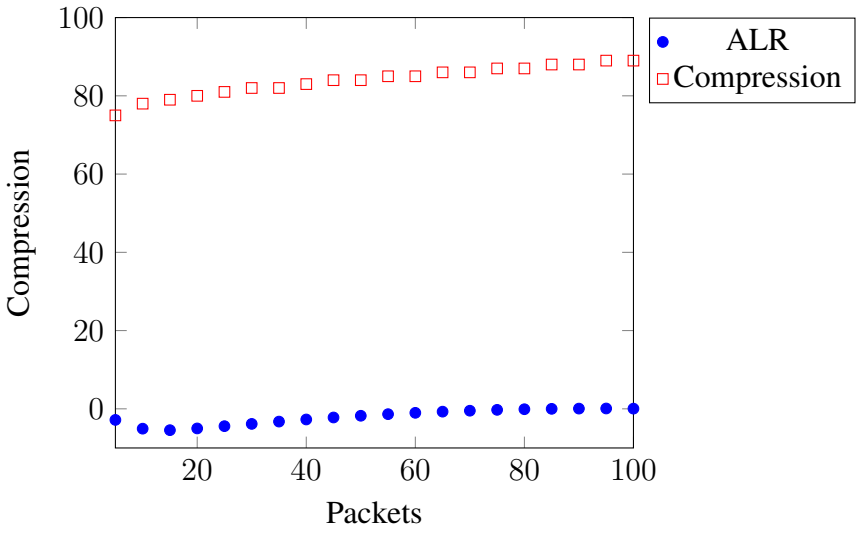

Fig. 38: LINTS compression using the tainted flow capped-by-packets strategy on the MACCDC 2011 data set as analyzed by the RegAug2013 rule set

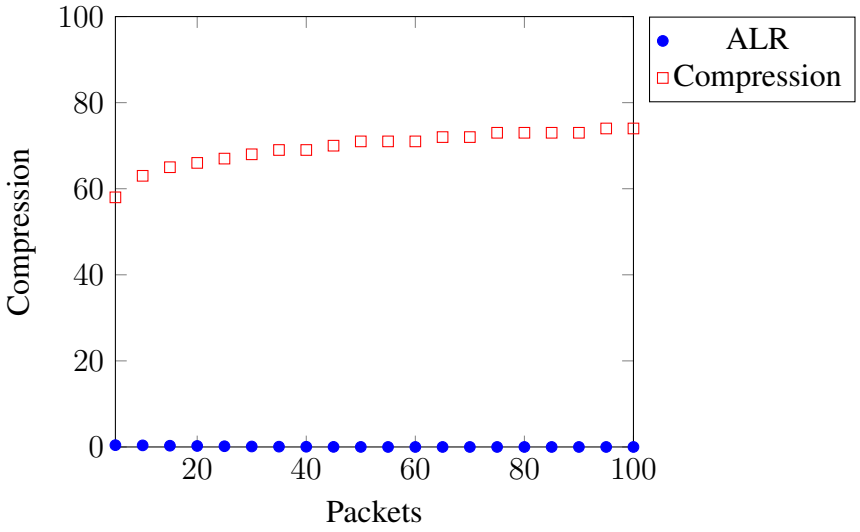

Fig. 39: LINTS compression using the tainted flow capped-by-packets strategy on the MACCDC 2012 data set as analyzed by the RegAug2013 rule set

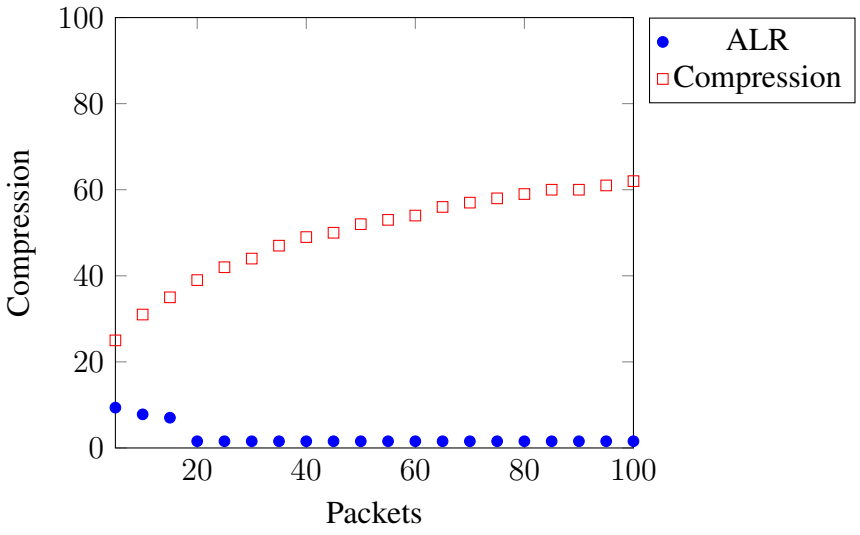

Fig. 40: LINTS compression using the tainted flow capped-by-packets strategy on the ISCX 2012 testing data set as analyzed by the RegAug2013 rule set

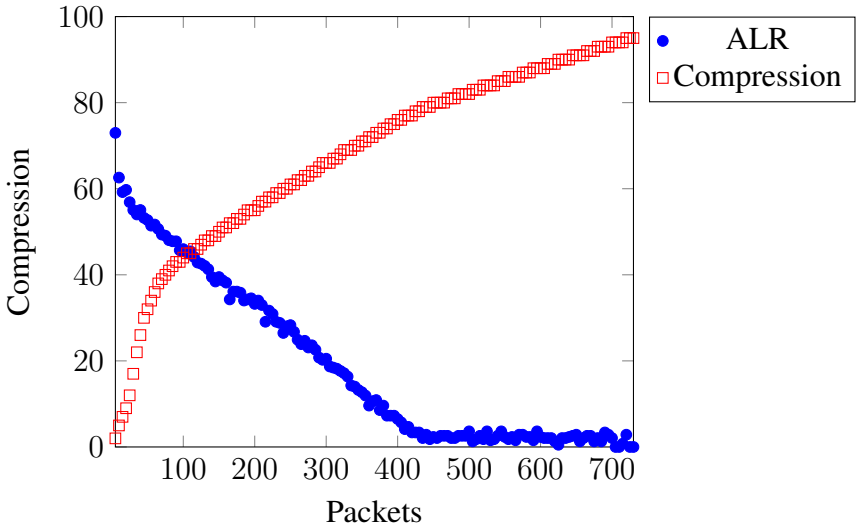

Fig. 41: LINTS compression using the tainted flow capped-by-bytes strategy on the UNSW-NB 2015 January data set as analyzed by the RegSep2018 rule set

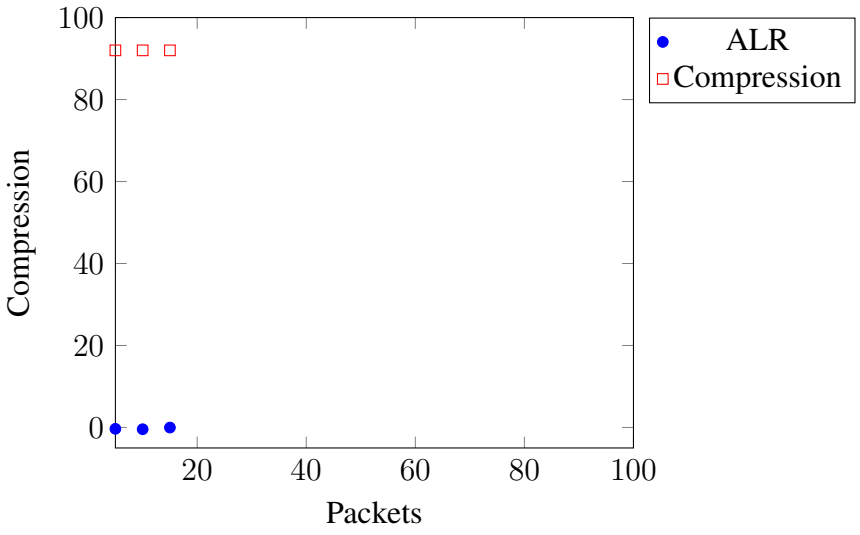

Fig. 42: LINTS compression using the tainted flow capped-by-packets strategy on the ARL 2016 data set as analyzed by the RegJul2018 rule set

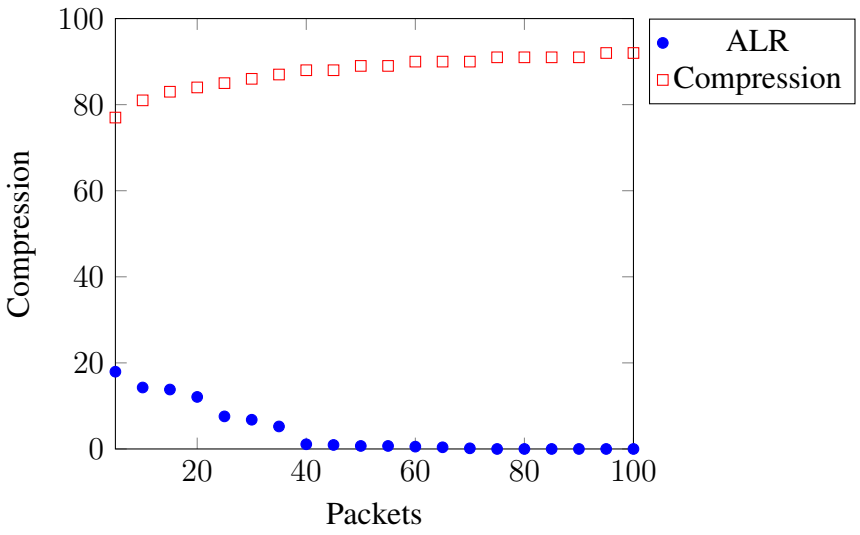

Fig. 43: LINTS compression using the tainted flow capped-by-packets strategy on the CIC 2017 testing data set as analyzed by the RegJul2018 rule set

Assuming that a 1% ALR is acceptable, Table 5 shows the depth in packets, percent of original size, and ALR when this first crosses below 1%. The theory behind this strategy is that malicious flows will manifest themselves as malicious early. The hope is that being able to detect a malicious flow early would allow malicious flows to be retained and benign flows dropped after a small number of packets. In practice, this strategy retains too much traffic to be useful for compression.

Table 5: Results from the tainted flow capped-by-packets strategy

Data Set	Packet Count	Size (%)	ALR (%)
cdx09u010	5	82.00	-0.19
cdx09u020	10	97.00	0.40
maccdc10	20	94.00	0.92
maccdc11	5	75.00	-2.83
maccdc12	5	58.00	0.42
iscx12	100	62.00	1.56
unsw15jan	625	90.00	0.51
arl16	5	92.00	-0.33
cic17te	45	88.00	0.93

4.2.6 Tainted Flow Capped by Bytes

In these trials, the combination of the tainted flow capped-by-bytes strategy is applied to various data sets against relevant rule sets. This was done to explore the effectiveness of these strategies in compressing network traffic for network intrusion dection applications. The ALR verse the compress is plotted for several data sets against relevant rule sets in Figs. 44–52. The raw data may be found in Tables C-31–C-36.

Listing 16 displays the *lints.conf* file for these trials. Line #1 sets the *elapsed-time_format* to display the elapsed time in seconds and microseconds. This simplifies the extraction and processing of duration data. Line #2 sets the Bloom filter to *malicious.bflt* in the parent directory. Line #3 sets the *flow_feature* to *bytes*. This instructs LINTS to stop comparing the N-grams in the payload to the Bloom filter and stop keeping packets in untainted flows when the byte count is greater than the threshold. Line #3 tells LINTS to add the Compress only TCP strategy.

Listing 17 displays the *runlints.conf* file for these trials. The *runlints.sh* script sources this configuration file, executing the command here. This file sets the in-

Listing 16: The *lints.conf* file for Trial 5, Run 2

```
1 elapsedtime_format = %S
2 filter = ../malicious.bflt
3 flow_feature = bytes
4 keepnontcp
```

dependent variable to be *-C*, meanint that the values generated by the *sequence* program will be placed after the *-C* option in *lints* setting the flow threshold.

Listing 17: The *runlints.conf* file for Trial 5, Run 2

```
1 #
2 # This trial used the flow threshold as the
3 # independent variable.
4 #
5
6 IV=-C
```

Listing 18 displays the *sequence.conf* file for these trials. Line #1 sets the algorithm to *arithmetic*, which implements Eq. 3. It configures *sequence* to generate numbers from 200 to 4,000 in increments of 200. For most of the data sets, 4,000 bytes was enough to get to a reasonable level of alert loss. The data sets MACCDC 2012 and UNSW-NB 2015 January did not produce an ALR by 4,000 bytes. The results from using the tainted flow capped-by-bytes strategy show that these data sets will not produce less than 1% ALR until the threshold is deep into the flow. Only three interations were completed for the ARL 2016 data set; however, that was enough to obtain an ALR less than 1%.

Listing 18: The *sequence.conf* file for Trial5, Run 2

```
1 type = arithmetic
2 difference = 200
3 limit = 20
4 start = 2
```

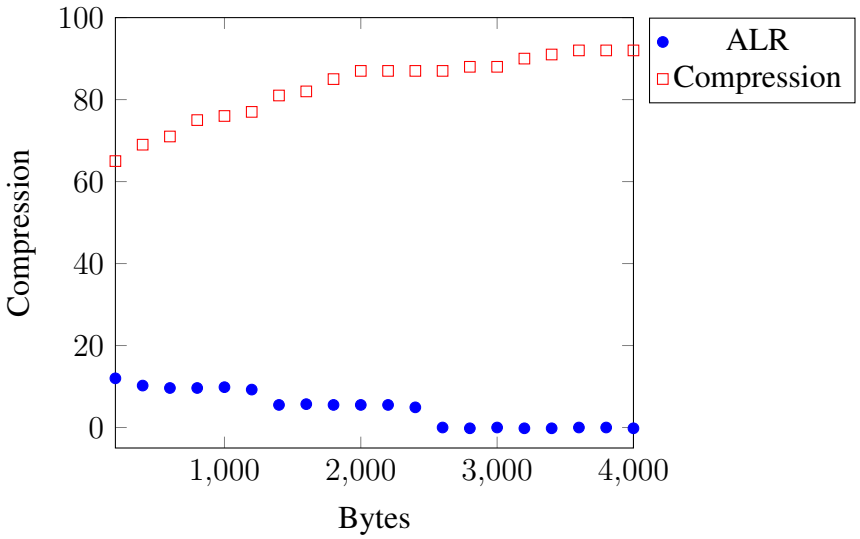

Fig. 44: LINTS compression using the tainted flow capped-by-bytes strategy on the CDX 2009 gator-usama010 data set as analyzed by the Circa2009 rule set

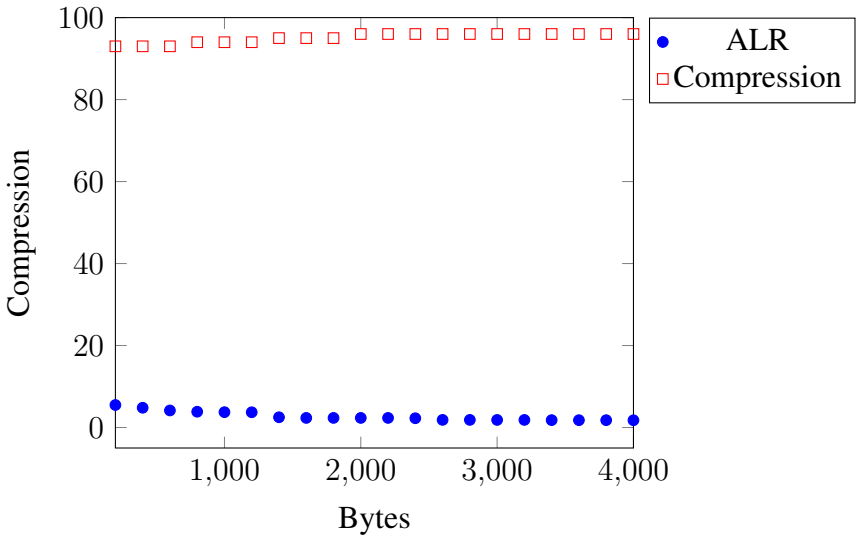

Fig. 45: LINTS compression using the tainted flow capped-by-bytes strategy on the CDX 2009 gator-usama020 data set as analyzed by the Circa2009 rule set

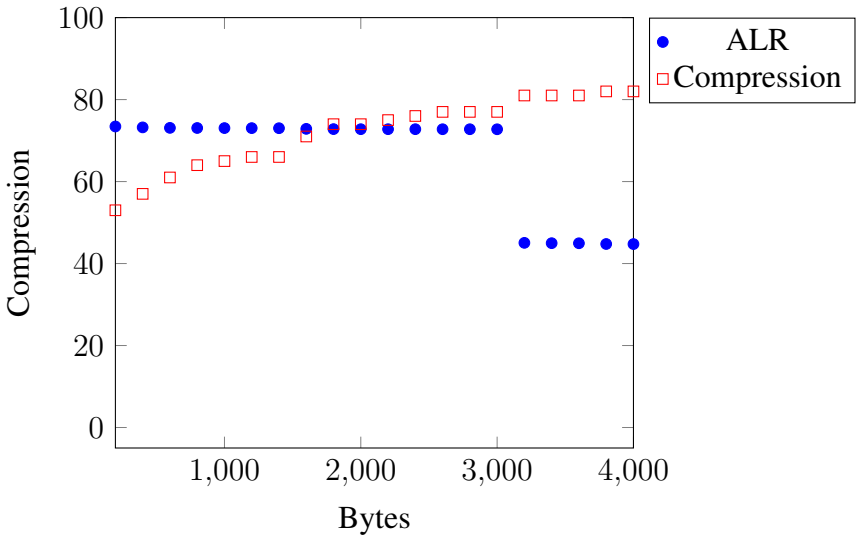

Fig. 46: LINTS compression using the tainted flow capped-by-bytes strategy on the MACCDC 2010 data set as analyzed by the Circa 2009 rule set

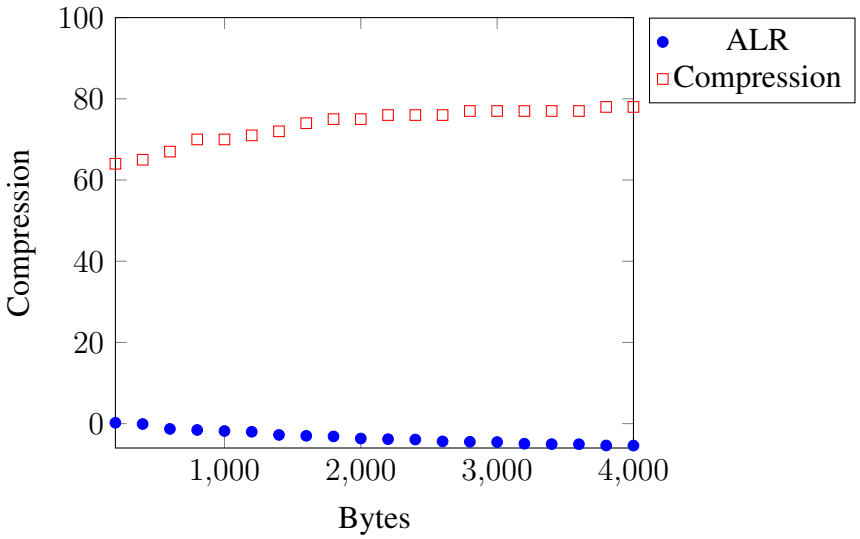

Fig. 47: LINTS compression using the tainted flow capped-by-bytes strategy on the MACCDC 2011 data set as analyzed by the RegAug2013 rule set

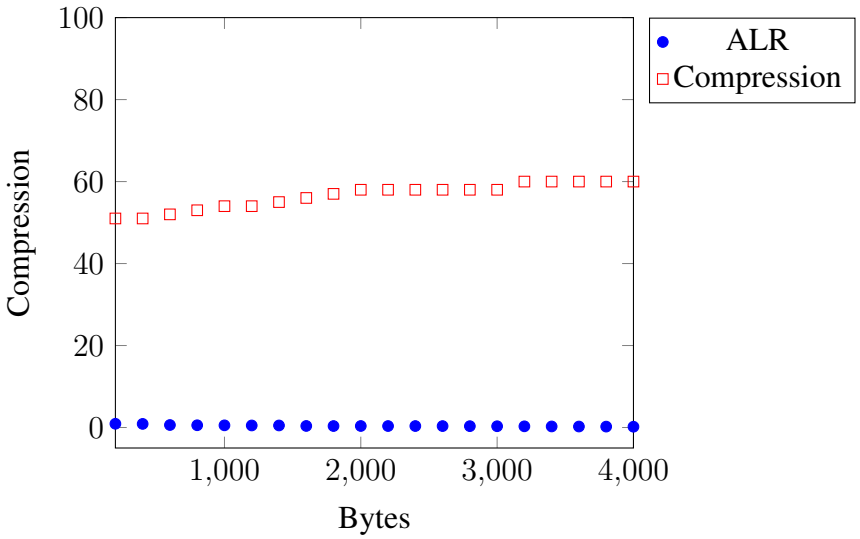

Fig. 48: LINTS compression using the tainted flow capped-by-bytes strategy on the MACCDC 2012 data set as analyzed by the RegAug2013 rule set

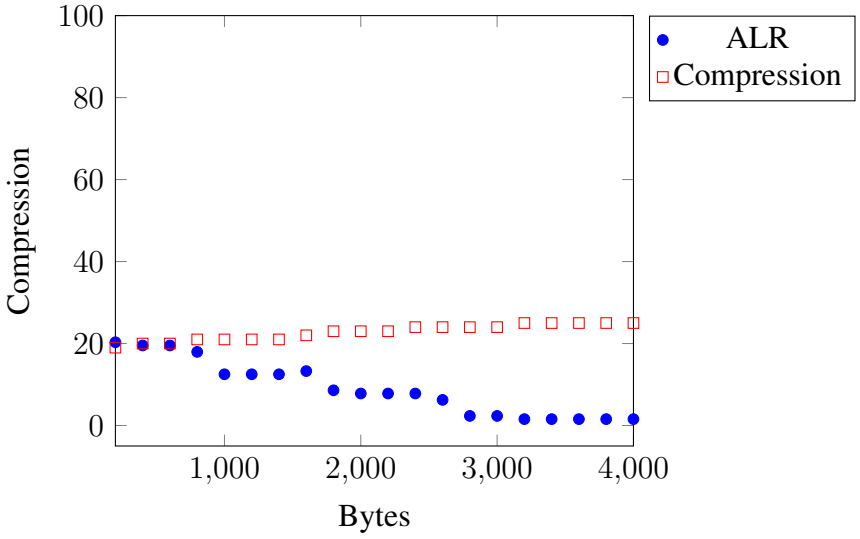

Fig. 49: LINTS compression using the tainted flow capped-by-bytes strategy on the ISCX 2012 testing data set as analyzed by the RegJul2018 rule set

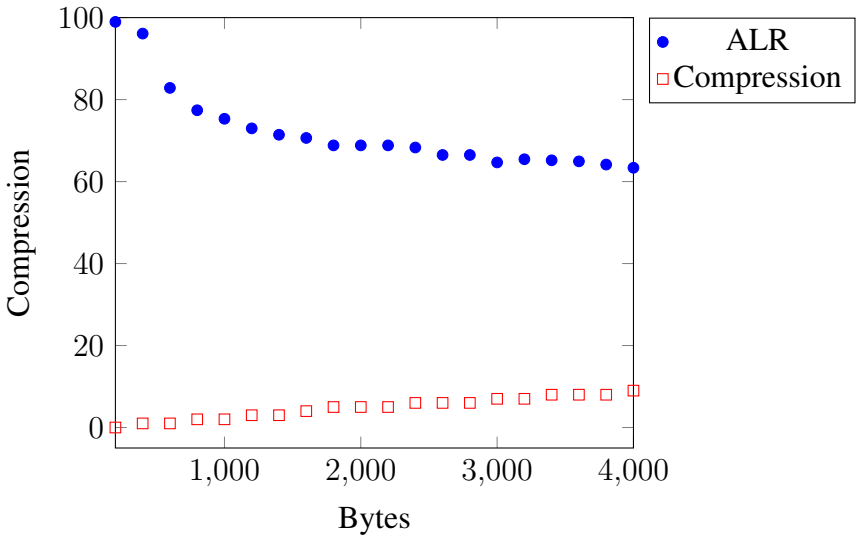

Fig. 50: LINTS compression using the tainted flow capped-by-bytes strategy on the UNSW-NB 2015 January data set as analyzed by the RegSep2018 rule set

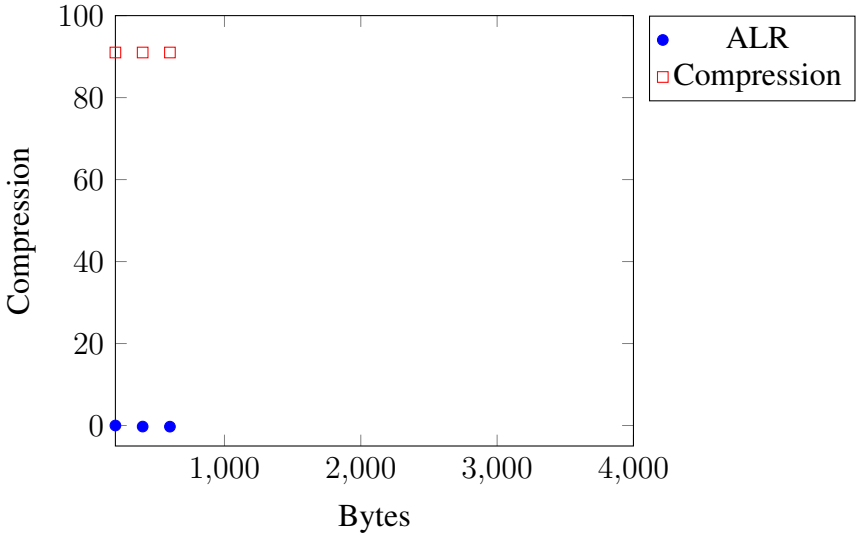

Fig. 51: LINTS compression using the tainted flow capped-by-bytes strategy on the ARL 2016 data set as analyzed by the RegJul2018 rule set

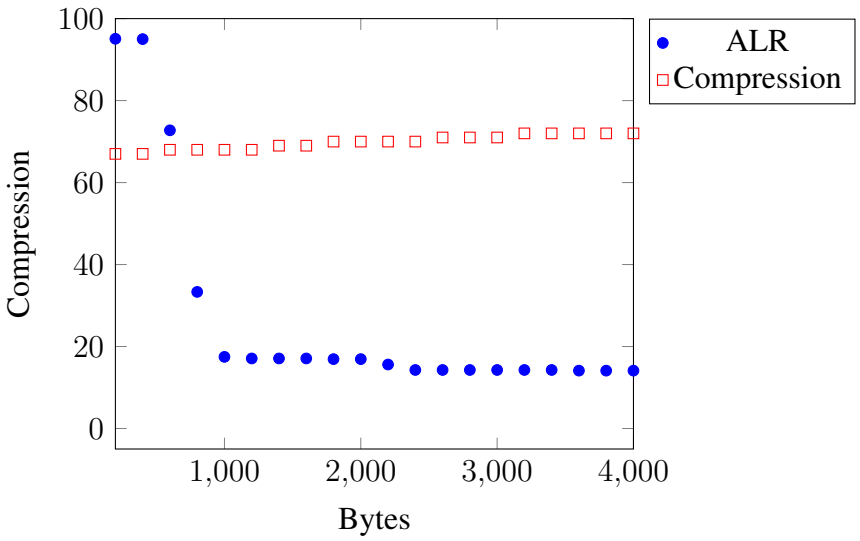

Fig. 52: LINTS compression using the tainted flow capped-by-bytes strategy on the CIC 2017 testing data set as analyzed by the RegJul2018 rule set

Assuming that a 1% ALR is acceptable, Table 6 shows the depth in packets, percent of original size, and ALR when this first crosses below 1%. The theory behind this strategy is that malicious flows will manifest themselves as malicious early. The hope is that being able to detect a malicious flow early would allow malicious flows to be retained and benign flows dropped after a small number of bytes. In practice, this strategy retains too much traffic to be useful for compression.

Table 6: Results from the tainted flow capped-by-bytes strategy

Data Set	Packet Count	Size (%)	ALR (%)
cdx09u010	2600	87.00	0.00
cdx09u020	4000	96.00	1.77
maccdc10	4000	82.00	44.75
maccdc11	200	64.00	0.21
maccdc12	200	51.00	0.91
iscx12	4000	25.00	1.56
unsw15jan	4000	9.00	63.37
arl16	200	91.00	0.00
cic17te	4000	72.00	14.12

4.2.7 Tainted Flow Capped by Packets with Entropy

In these trials, the combination of the tainted flow capped-by-packets strategy is applied to various data sets against relevant rule sets. In addition, the Entropy strategy has been employed with the entropy threshold set to 7.7. This was done to explore the effectiveness of these strategies in compressing network traffic for network intrusion dection applications. The ALR verse the compress is plotted for several data sets against relevant rule sets in Figs. 53–61. The raw data may be found in Tables C-37–C-42.

Listing 19 displays the *lints.conf* file for these trials. Line #1, *elapsedtime_format*, is set to display the elapsed time in seconds and microseconds. This simplifies the extraction and processing of duration data. Line #2 sets the entropy threshold to 7.7, meaning that packets with a payload entropy of 7.7 or greater will be marked as benign and compressed out of the network stream. Line #3 sets the Bloom filter to *malicious.bflt* in the parent directory. Line #4 sets the *flow_feature* to *packets*. This instructs LINTS to stop comparing the N-grams in the payload to the Bloom filter and stop keeping packets in untainted flows when the packet count is greater than the threshold. Line #5 tells LINTS to add the Compress only TCP strategy.

Listing 19: The *lints.conf* file for Trial 5, Run 3

```
1 elapsedtime_format = %S
2 entropy = 7.7
3 filter = ../malicious.bflt
4 flow_feature = packets
5 keepnontcp
```

Listing 20 displays the *runlints.conf* file for these trials. The *runlints.sh* script sources this configuration file, executing the command here. This file sets the independent variable to be *-C*, meaning that the values generated by the *sequence* program will be placed after the *-C* option in *lints* setting the flow threshold.

Listing 20: The *runlints.conf* file for Trial 5, Run 3

```
1 #
2 # This trial used the flow threshold as the
3 # independent variable.
4 #
5
6 IV=-C
```

Listing 21 displays the *sequence.conf* file for these trials. Line #1 sets the algorithm to *arithmetic*, which implements Eq. 3. It configures *sequence* to generate numbers from 5 to 100 in increments of 5. For most of the data sets, 100 bytes was enough to get to a reasonable level of alert loss. The UNSW-NB 2015 January data set does not produce an ALR of less than 1% after 100 hundred packets. This is consistent with the previous results. Only 10 iterations were captured for the ARL 2016 data set; however, that was enough to get an ALR less than 1%.

Listing 21: The *sequence.conf* file for Trial 5, Run 3

```
1 type = arithmetic
2 difference = 5
3 limit = 20
4 start = 2
```

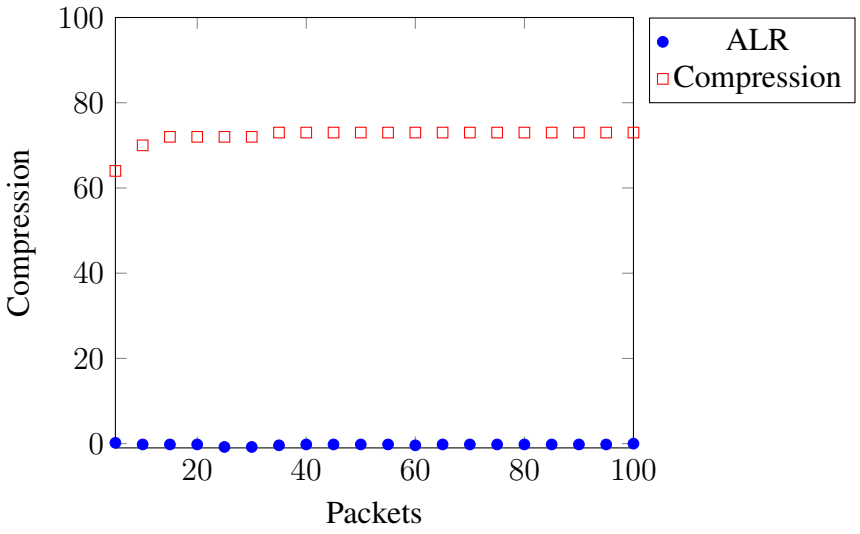

Fig. 53: LINTS compression using the tainted flow capped-by-packets with entropy strategy on the CDX 2009 gator-usama010 data set as analyzed by the Circa2009 rule set

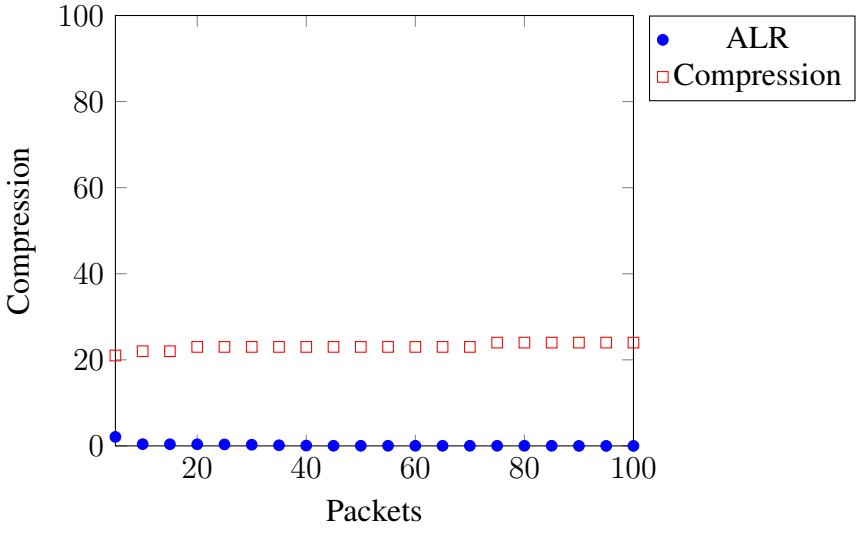

Fig. 54: LINTS compression using the tainted flow capped-by-packets with entropy strategy on the CDX 2009 gator-usama020 data set as analyzed by the Circa2009 rule set

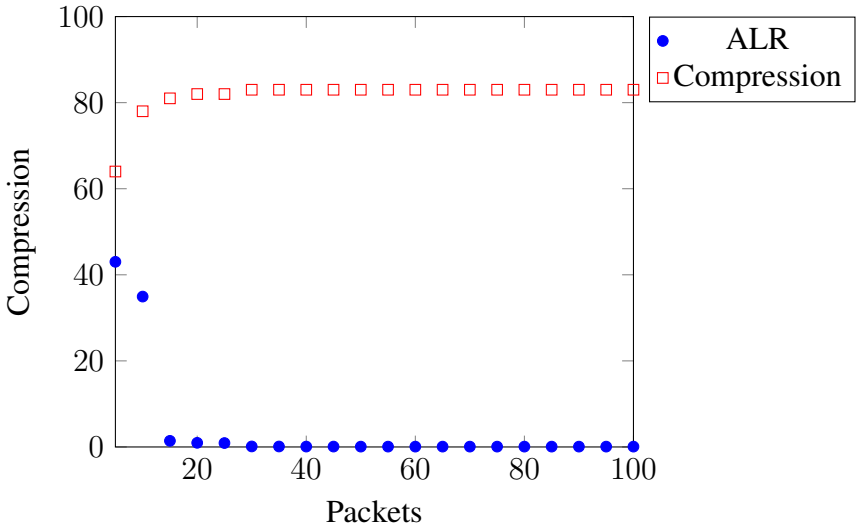

Fig. 55: LINTS compression using the tainted flow capped-by-packets with entropy strategy on the MACCDC 2010 data set as analyzed by the Circa2009 rule set

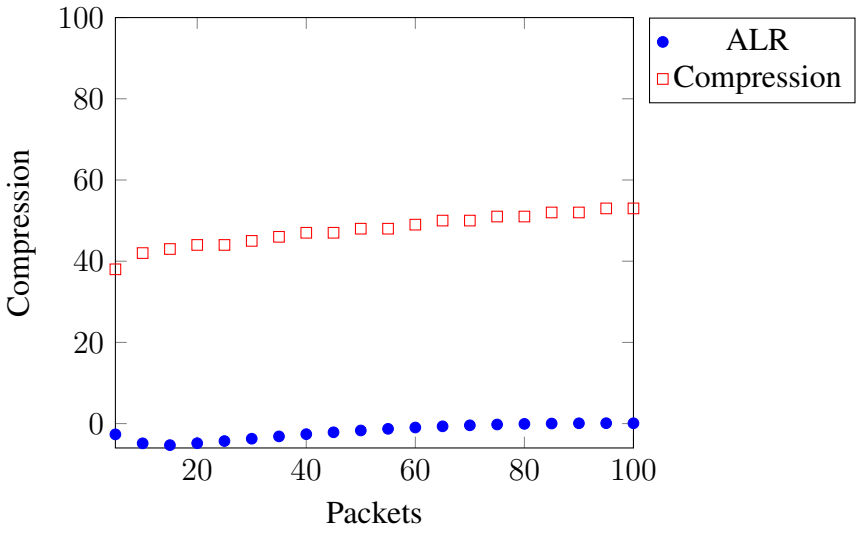

Fig. 56: LINTS compression using the tainted flow capped-by-packets with entropy strategy on the MACCDC 2011 data set as analyzed by the RegAug2013 rule set

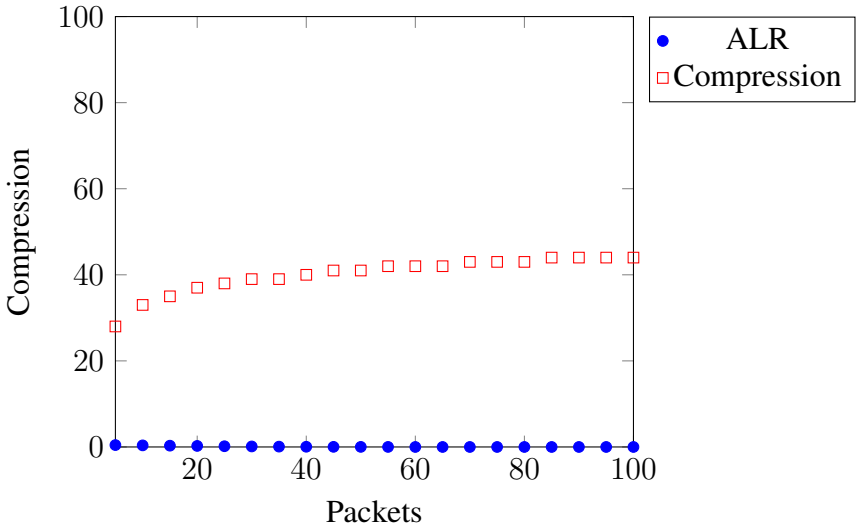

Fig. 57: LINTS compression using the tainted flow capped-by-packets with entropy strategy on the MACCDC 2012 data set as analyzed by the RegAug2013 rule set

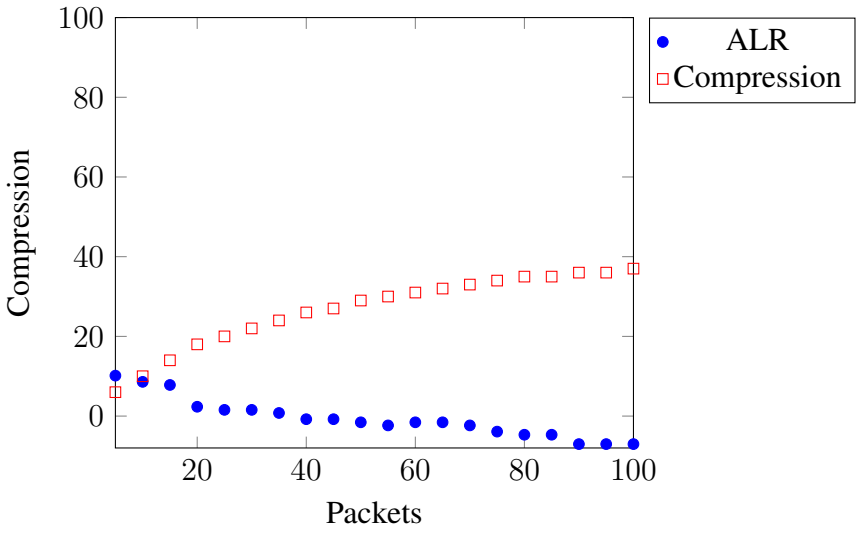

Fig. 58: LINTS compression using the tainted flow capped-by-packets with entropy strategy on the ISCX 2012 testing data set as analyzed by the RegAug2013 rule set

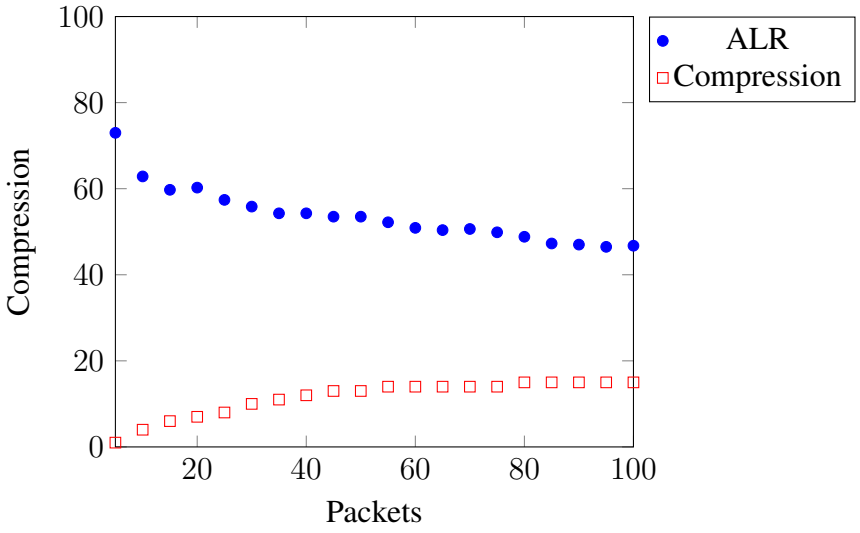

Fig. 59: LINTS compression using the tainted flow capped-by-packets with entropy strategy on the UNSW-NB 2015 January data set as analyzed by the RegSep2018 rule set

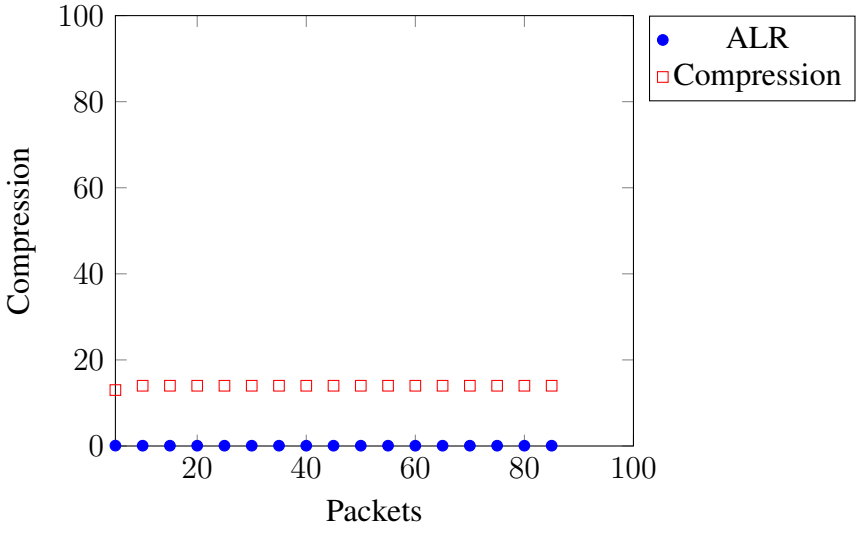

Fig. 60: LINTS compression using the tainted flow capped-by-packets strategy with entropy on the ARL 2016 data set as analyzed by the RegSep2018 rule set

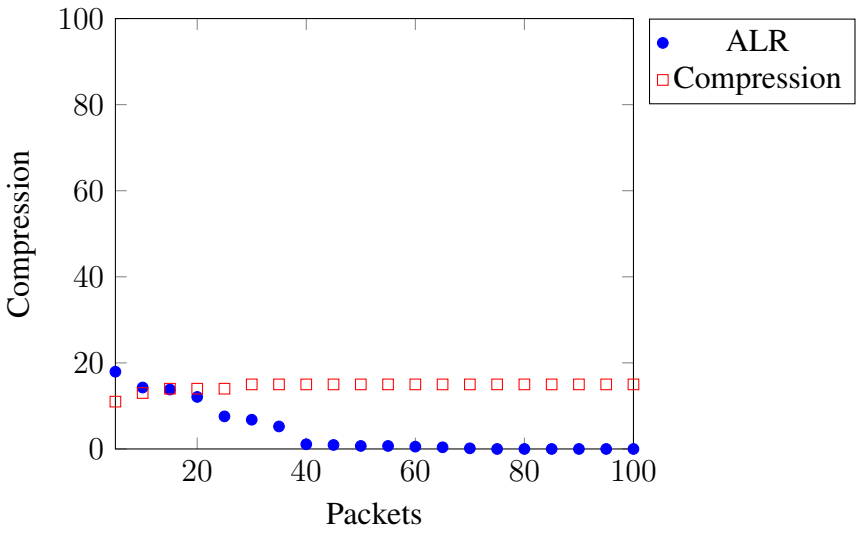

Fig. 61: LINTS compression using the tainted flow capped-by-packets with entropy strategy on the CIC 2017 testing data set as analyzed by the RegJul2018 rule set

Assuming that a 1% ALR is acceptable, Table 7 shows the depth in packets, percent of original size, and ALR when this first crosses below 1%. High-entropy packets only comprise compressed and encrypted traffic, which are of little use at the CAS, but they are more likely to contain spurious N-grams. As seen in Table 7, this combination is effective in compressing most of the data sets we examine. The most notable exception is the UNSW January 2015 data set. This data set contains several examples of flows that do not manifest their maliciousness until very deep into the flow.

Table 7: Results from the tainted flow capped-by-packets with entropy strategy

Data Set	Packet Count	Size (%)	ALR (%)
cdx09u010	5	64.00	0.19
cdx09u020	10	22.00	0.40
maccdc10	20	82.00	0.96
maccdc11	5	38.00	-2.65
maccdc12	5	28.00	0.44
iscx12	35	24.00	0.78
unsw15jan	100	15.00	46.75
arl16	5	13.00	0.09
cic17te	45	15.00	0.93

4.2.8 Tainted Flow Capped by Bytes with Entropy

In these trials, the combination of the tainted flow capped-by-bytes strategy is applied to various data sets against relevant rule sets. In addition, the Entropy strategy has been employed with the entropy threshold set to 7.7. This was done to explore the effectiveness of these strategies in compressing network traffic for network intrusion dection applications. The ALR verse the compress is plotted for several data sets against relevant rule sets in Fig. 62–Fig. 70. The raw data may be found in Tables C-43–C-48.

Listing 22 displays the *lints.conf* file for these trials. Line #1, the *elapsedtime_format* to display the elapsed time in seconds and microseconds. This simplifies the extraction and processing of duration data. Line #2 sets the entropy threshold to 7.7. Packets with a payload entropy of 7.7 or greater will be marked as benign and compressed out of the network stream. Line #3 sets the Bloom filter to *malicious.bflt* in the parent directory. Line #4 sets the *flow_feature* to *bytes*. This instructs LINTS to stop comparing the N-grams in the payload to the Bloom filter and stop keeping

packets in untainted flows when the byte count is greater than the threshold. Line #5 tells LINTS to add the Compress only TCP strategy.

Listing 22: The *lints.conf* file for Trial 5, Run 4

```
1 elapsedtime_format = %S
2 entropy = 7.7
3 filter = ../malicious.bflt
4 flow_feature = bytes
5 keepnontcp
```

Listing 23 displays the *runlints.conf* file for these trials. The *runlints.sh* script sources this configuration file, executing the command here. This file sets the independent variable to be *-C*, meaning that the values generated by the *sequence* program will be placed after the *-C* option in *lints* setting the flow threshold.

Listing 23: The *runlints.conf* file for Trial 5, Run 4

```
1 #
2 # This trial used the flow threshold as the
3 # independent variable.
4 #
5
6 IV=-C
```

Listing 24 displays the *sequence.conf* file for these trials. Line #1 sets the algorithm to *arithmetic*, which implements Eq. 3. It configures *sequence* to generate numbers from 200 to 4,000 in increments of 200. For most of the data sets, 4,000 bytes was enough to get to a reasonable level of alert loss.

The data sets MACCDC 2012 and UNSW-NB 2015 did not produce an acceptable ALR by 4,000 bytes. The results from using the tainted flow capped-by-packets strategy show that these data set will not produce less than 1% ALR until the threshold is deep into the flow. Only 10 iterations were completed for the ARL 2016 data set; however, that was enough to obtain an ALR less than 1%.

Listing 24: The *sequence.conf* file for Trial 5, Run 4

```
type = arithmetic
difference = 200
limit = 20
start = 2
```

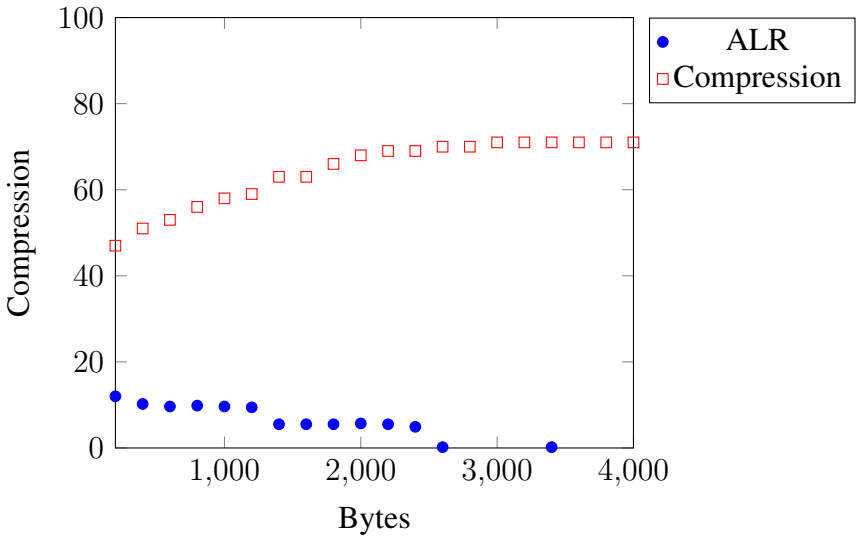

Fig. 62: LINTS compression using the tainted flow capped-by-bytes with entropy strategy on the CDX 2009 gator-usama010 data set as analyzed by the Circa2009 rule set

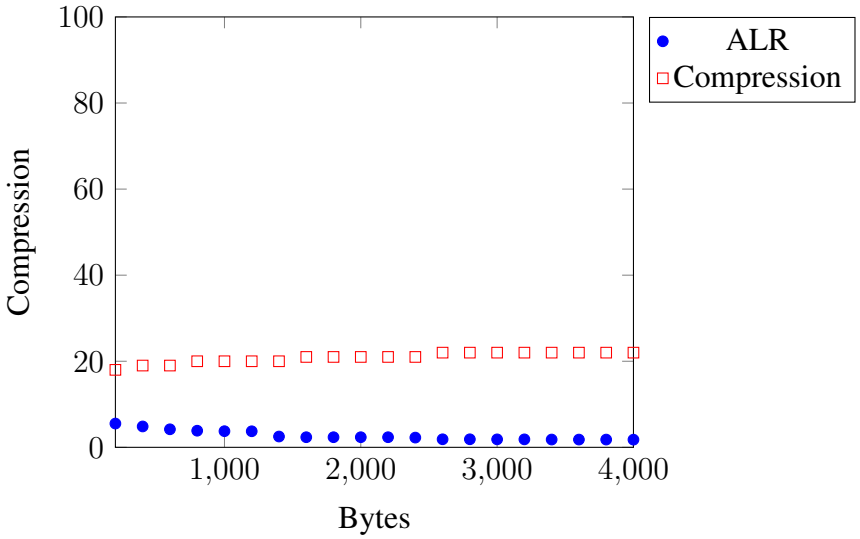

Fig. 63: LINTS compression using the tainted flow capped-by-bytes with entropy strategy on the CDX 2009 gator-usama020 data set as analyzed by the Circa2009 rule set

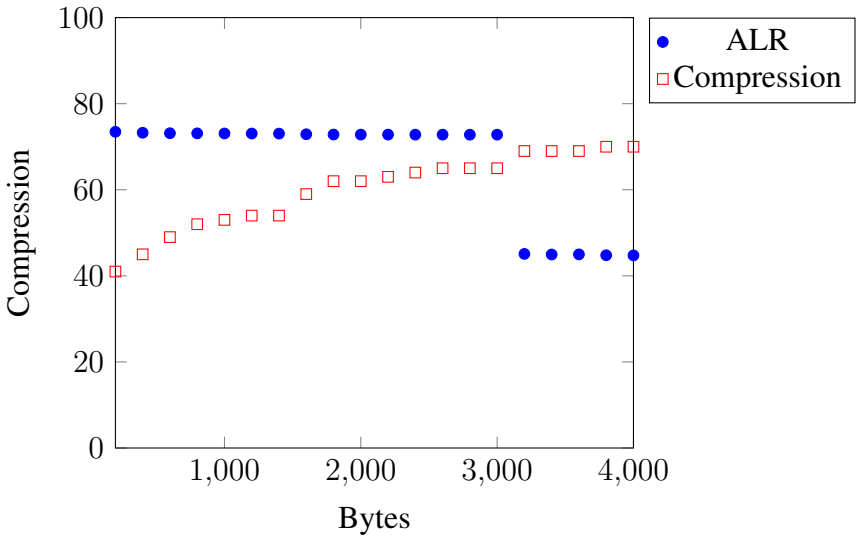

Fig. 64: LINTS compression using the tainted flow capped-by-bytes with entropy strategy on the MACCDC 2010 data set as analyzed by the Circa2009 rule set

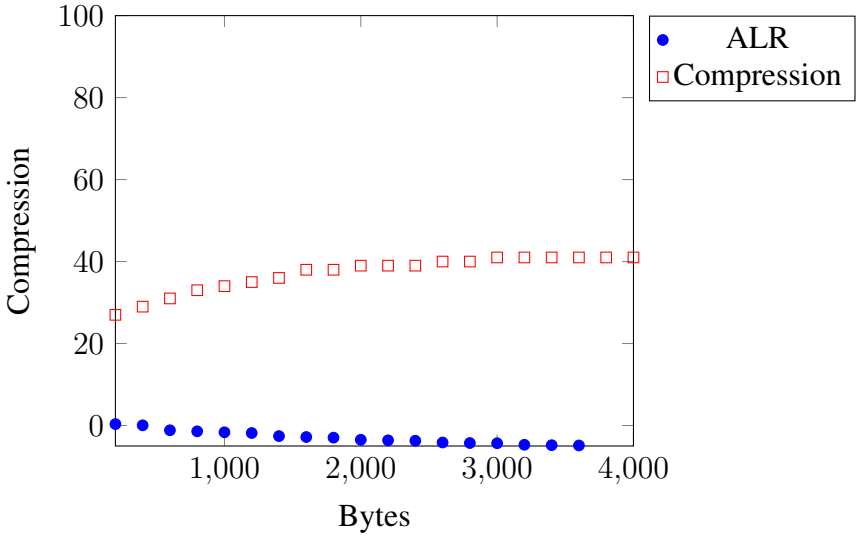

Fig. 65: LINTS compression using the tainted flow capped-by-bytes with entropy strategy on the MACCDC 2011 data set as analyzed by the RegAug2013 rule set

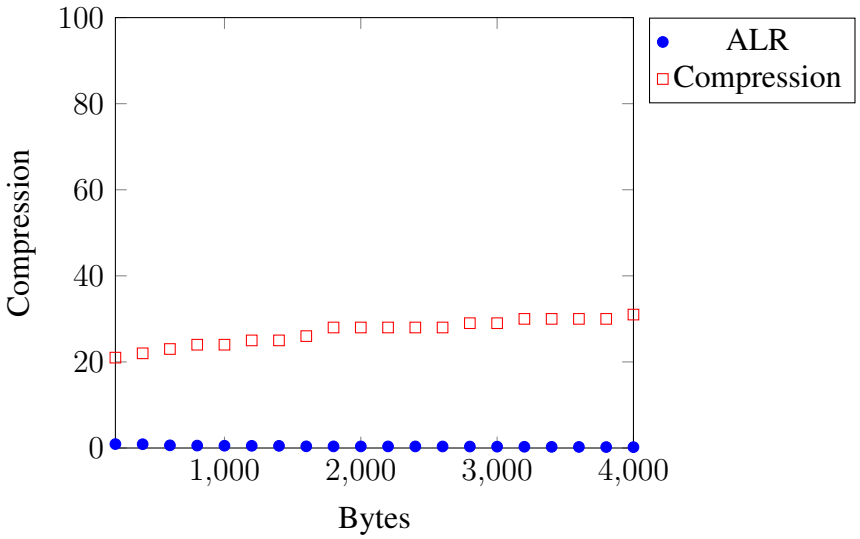

Fig. 66: LINTS compression using the tainted flow capped-by-bytes with entropy strategy on the MACCDC 2012 data set as analyzed by the RegAug2013 rule set

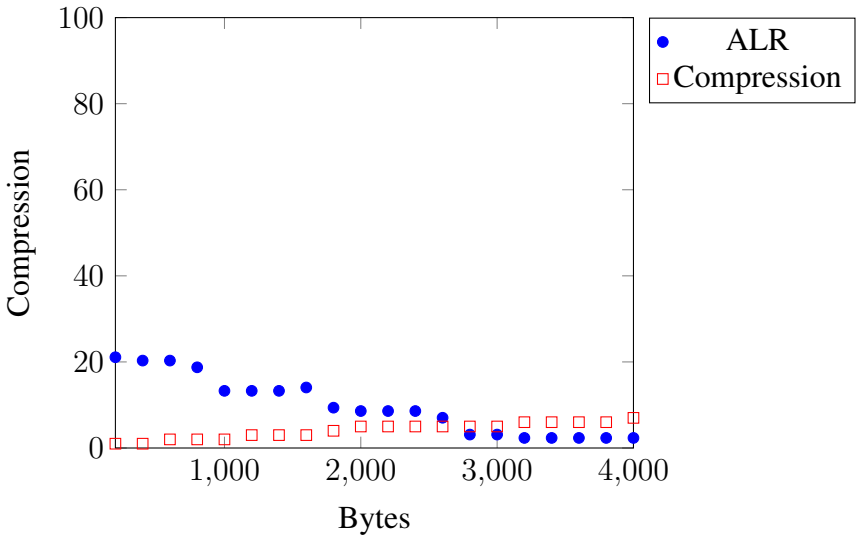

Fig. 67: LINTS compression using the tainted flow capped-by-bytes with entropy strategy on the ISCX 2012 testing data set as analyzed by the RegAug2013 rule set

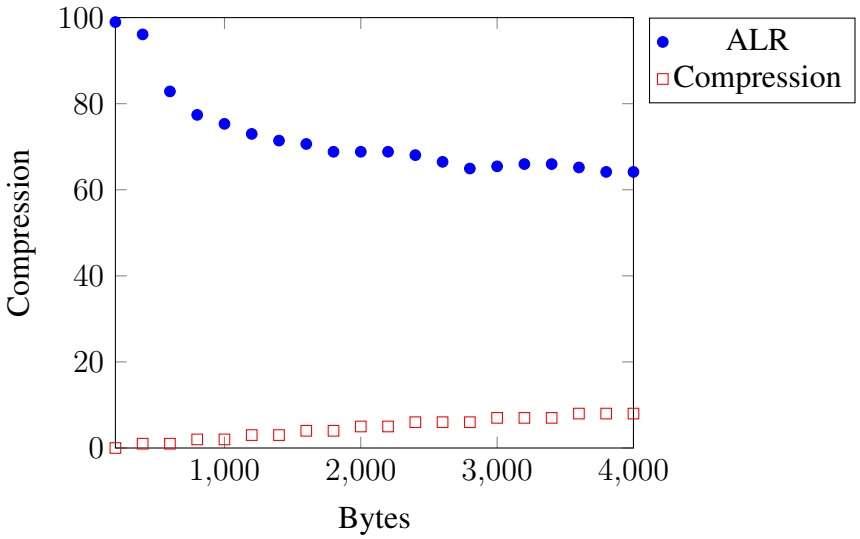

Fig. 68: LINTS compression using the tainted flow capped-by-bytes with entropy strategy on the UNSW-NB 2015 January data set as analyzed by the RegSep2018 rule set

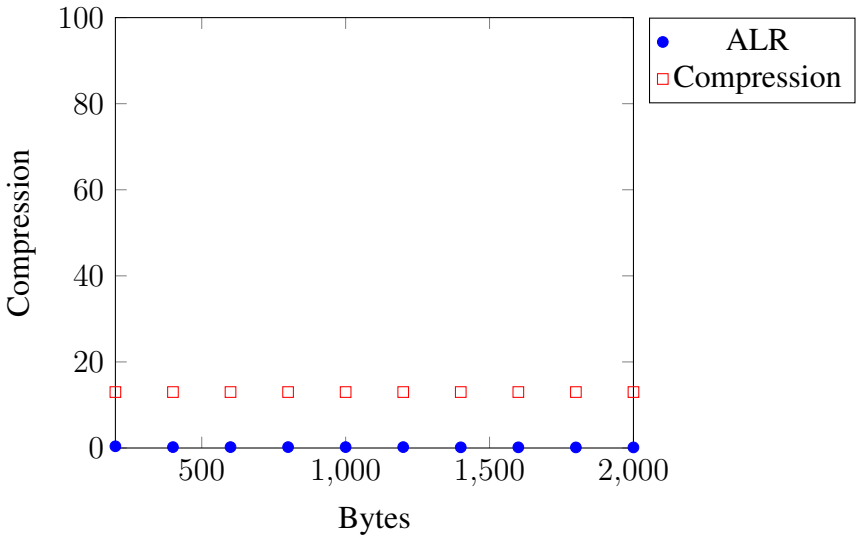

Fig. 69: LINTS compression using the tainted flow capped-by-bytes with entropy strategy on the ARL 2016 data set as analyzed by the RegSep2018 rule set

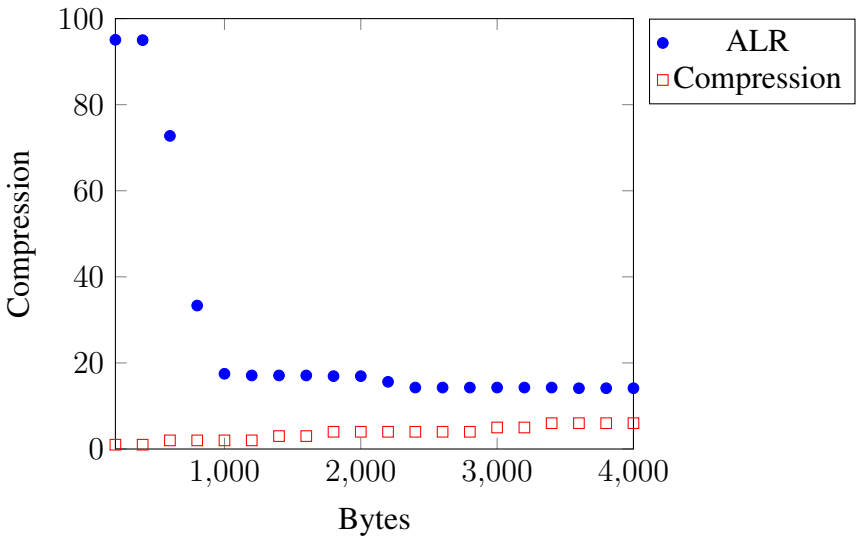

Fig. 70: LINTS compression using the tainted flow capped-by-bytes with entropy strategy on the CIC 2017 testing data set as analyzed by the RegJul2018 rule set

Assuming that a 1% ALR is acceptable, Table 8 shows the depth in packets, percent of original size, and ALR when this first crosses below 1%.

Table 8: Results from the tainted flow capped-by-bytes with entropy strategy

Data Set	Packet Count	Size (%)	ALR (%)
cdx09u010	2600	70.00	0.19
cdx09u020	4000	22.00	1.78
maccdc10	4000	70.00	44.75
maccdc11	200	27.00	0.35
maccdc12	200	21.00	0.91
iscx12	4000	7.00	2.34
unsw15jan	100	15.00	46.75
arl16	200	13.00	0.28
cic17te	4000	6.00	14.12

4.2.9 LINTS Trial Observations

Conducting several trials with different data sets, compression strategies, and thresholds demonstrates that most of the data sets may be compressed to an acceptable level with an acceptable ALR; however, no single combination of strategies and thresholds will compress all of the data sets to an acceptable level with an acceptable ALR.

4.3 LINTS Compression Based upon Lessons Learned

It is one thing to conduct several trials with different combinations of strategies and thresholds to find the ultimate level of compression with acceptable packet loss; however, this would not be useful in real-world situations. A working knowledge of the typical traffic seen at a particular installation may allow settings to be discovered that would provide the desired amount of compression with an acceptable ALR. This diversity of strategies and thresholds has the advantage of complicating the adversaries' ability to adjust the placement of the attack to avoid detection by being compressed out of the traffic being sent to the CAS for further analysis.

4.3.1 UNSW-NB15 February

The UNSW-NB15 January data set was resistent to compression by either the capped-by-packets or capped-by-bytes strategy. Even the tainted flow strategy did not improve the compression because the malicious traffic first appears deep into the net-

work flows. The large amount of compressed and encrypted traffic made it compressable by employing the entropy strategy. In an effort to compress the UNSW-NB15 February data set based only upon knowledge of the UNSW-NB15 January data set, the entropy strategy was selected with a threshold of 7.7. Given these settings, LINTS was able to compress the UNSW-NB15 February data set to 21% of its original size with an ALR 0.08% in under 8 mins and 57 secs. This is small enough that standard lossless compression would be able to further reduce it to less than 10% of the original size, which is typically acceptable for network intrusion detection applications.

4.3.2 ARL 2017

The ARL 2016 data set did not compress well using the tainted flow methods. It did contain a large amount of compressed and encrypted traffic and compressed well to the entropy strategy. However, it compressed even better using the capped strategies without any tainting. In an effort to compress the ARL 2017 data set based only upon knowledge of the ARL 2016 data set, the capped-by-packets strategy was selected with a threshold of 10. Given these settings, LINTS was able to compress the ALR 2017 data set to 14% of its original size with an ALR of -0.21% in 6 h, 47 mins, and 51 secs. This is small enough that standard lossless compression would be able to further reduce it to less than 10% of the original size, which is typically acceptable for network intrusion detection applications.

The results of these “blind” tests demonstrate that based upon the settings learned from similar data sets, LINTS was able to compress both the UNSW-NB15 February and the ARL 2017 data set to an acceptable level of compression with an acceptable ALR.

5. Conclusion

The improvements to the way the various strategies are aggregated allowed LINTS to compress network traffic to a greater degree and with a smaller ALR than the previous Netcomp program. The improvements to the flow-tracking algorithm allowed LINTS to perform this compression with significantly fewer resources than the previous Netcomp program. The compression of several data sets with multiple compression strategies and thresholds reveals that although most network traffic may be compressed small enough with an acceptable ALR, the combination of strategies and thresholds can vary greatly between data sets. A single combination

of strategies and thresholds that would acceptably compress every data set could not be found. By examining related data sets, it was possible to select a strategy and threshold combination that would blindly compress a data set to acceptable levels of compression and alert loss. With local tuning, LINTS appears to be able to compress network traffic enough to allow transmission to the CAS without undue impact on the sites bandwidth with an acceptable ALR. This local tuning may be an advantage because it should make it more difficult for adversaries to ensure their attacks are compressed out of the network traffic.

When this project began, the thinking was that anomaly detection methods that might be too noisy for network intrusion detection purposes may be useful in tainting network traffic for compression purposes. However, suitable open source anomaly detection methods were not found.[4] It is certainly possible that such methods do exist or that the proper application of artificial intelligence will provide a useful tainting method in the future. LINTS is designed to facilitate the addition of new strategies. Currently, it uses a Bloom filter of malicious N-grams to detect if a packet is likely to be malicious and taint the network traffic. The collection of malicious N-grams and creation of malicious Bloom filters could be greatly improved. The N-gram size of 5 and the use of the MurmurHash were both derived from previous work.[21,22] It is possible that different N-gram sizes and hashes may provide better or less resource-intensive results. The key for the unordered map is an ASCII representation of the source and destination addresses and ports. This key is then hashed with the standard hash algorithm. It is possible that a more efficient hashing strategy would improve the performance.

6. References

1. Smith SC, Hammell II RJ. Proposal for kelly criterion-inspired lossy network compression for network intrusion applications. Journal of Information Systems Applied Research. 2017;10(2):43.

2. Kelly JL. A new interpretation of information rate. Information Theory, IRE Transactions on. 1956;2(3):185–189.

3. Smith S, Neyens S, Hammell R. The use of entropy in lossy network traffic compression for network intrusion detection applications. In: ICMLG 2017 5th International Conference on Management Leadership and Governance; p. 352.

4. Smith SC, II RJH. The use of packet header anomaly detection in lossy network traffic compression for network intrusion detection applications. Army Research Laboratory (US); 2018 Nov. Report No.: ARL-TR-8578.

5. Smith SC, Hammell II RJ. The use of snap length in lossy network traffic compression for network intrusion detection applications. Journal of Information Systems Applied Research. 2019;12(1):17.

6. Smith SC, II RJH. The use of flow features in lossy network traffic compression for network intrusion detection applications. Journal of Systemics, Cybernetics and Informatics. 2019;17(2):40–48.

7. Smith SC. Lossy Network Compression for Distributed Network Intrusion Detection Applications [thesis]. Towson University; 2019.

8. Smith SC, Hammell II RJ. The use of tainted flows in network compression for distributed network intrusion detection. In: Proceedings of the Conference on Information Systems Applied Research ISSN; Vol. 2167; p. 1508.

9. Bloom BH. Space/time trade-offs in hash coding with allowable errors. Commun. ACM. 1970;13(7):422–426.

10. Roesch M. Snort: lightweight intrusion detection for networks. In: Proceedings of the 13th System Administration Conference (LISA '99); Vol. 99; 1999 Nov 7–12; Seattle, WA. p. 229–238.

11. Paxson V. Bro: A system for detecting network intruders in real-time. Computer networks. 1999;31(23-24):2435–2463.

12. Smith SC, Hammell RJ, Wong KW, Mateo CJ. An experimental exploration of the impact of host-level packet loss on network intrusion detection. In: 2016 Cybersecurity Symposium (CYBERSEC); p. 13–19.

13. Smith SC, Hammell RJ, Wong KW, Mateo CJ. An experimental exploration of the impact of multi-level packet loss on network intrusion detection. In: 2016 IEEE 14th International Conference on Software Engineering Research, Management and Applications (SERA); p. 23–30.

14. Kemmerer RA, Vigna G. Intrusion detection: a brief history and overview (supplement to Computer magazine). Computer. 2002;35(4):27–30.

15. Sangster B, O'Connor T, Cook T, Fanelli R, Dean E, Adams WJ, Morrell C, Conti G. Toward instrumenting network warfare competitions to generate labeled datasets. In: Proc. of the 2nd Workshop on Cyber Security Experimentation and Test (CSET09); 2009 Aug; Montreal, Canada.

16. Ierace N, Urrutia C, Bassett R. Intrusion prevention systems. Ubiquity. 2005;2005(June):2–2.

17. Long KS. Catching the cyber spy: ARL's interrogator. Army Research Laboratory; 2004. DTIC Document No.: ADA432198.

18. Long KS, Morgan JB. Using data mining to improve the efficiency of intrusion detection analsysis. Army Research Laboratory (US); 2007. Report No.: ARL-TR-4211.

19. Smith SC, Wong KW, Hammell RJ, Mateo CJ. An experimental exploration of the impact of network-level packet loss on network intrusion detection. Army Research Laboratory (US); 2015 Aug. Report No.: ARL-TR-7371.

20. Wang K, Parekh JJ, Stolfo SJ. Anagram: A content anomaly detector resistant to mimicry attack. In: International workshop on recent advances in intrusion detection; p. 226–248.

21. Chang RJ, Harang RE, Payer GS. Extremely lightweight intrusion detection (elide). Army Research Laboratory (US); 2013 Dec. Report No.: ARL-TR-7371.

22. Yu K, Leslie NO. Fast-d: malware and intrusion detection for mobile ad hoc networks (manets). In: NATO Specialist Meeting IST-145 on Predictive Analytics and Analysis in the Cyber Domain; p. 10–11.

23. Shannon CE. A mathematical theory of communication. ACM SIGMOBILE mobile computing and communications review. 2001;5(1):3–55.

24. Sharafaldin I, Lashkari AH, Ghorbani AA. Toward generating a new intrusion detection dataset and intrusion traffic characterization.. In: ICISSP; p. 108–116.

25. Shiravi A, Shiravi H, Tavallaee M, Ghorbani AA. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. Computers & Security. 2012;31(3):357–374.

26. Moustafa N, Slay J. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In: 2015 Military Communications and Information Systems Conference (MilCIS); p. 1–6.

27. Moustafa N, Slay J. The evaluation of network anomaly detection systems: Statistical analysis of the unsw-nb15 data set and the comparison with the kdd99 data set. Information Security Journal: A Global Perspective. 2016;25(1-3):18-31.

28. Turner A, Bing M. Tcpreplay: Pcap editing and replay tools for *nix. 2005 [accessed 2015 Mar]. http://tcpreplay.synfin.net.

29. Appleby A. MurmurHash, final version.. LiveJournal; 2008.

30. Roesch M, Green C. Snort users manual 2.9.16. The Snort Project; 2020.

31. Koenig T. at, batch, atq, atrm - queue, examine or delete jobs for later execution. In: Linux Online Manual; 7 ed.; Red Hat, Inc.; 1996.

Appendix A. Online Manual Pages

NAME

lints – the Lossy Intelligent Network Traffic Squeezer.

SYNOPSIS

lints [**options**] [**readfile**] ...

DESCRIPTION

The *lints* program is used to compress network traffic by keep only those packets that are most likely to be of interest to network intrusion detection applications. It does this by employing a number analytics which return a score between negative one and positive one. A score of negative one indicates that a packet is known to be benign. A score of positive one indicates that a packet is known to be malicious. The following analytics are implemented:

BenignBPF

This analytic is engaged by defining the *benign_bpf* configuration item which contains the text of a Berkeley packet filter (BPF) or the name of a file containing the text of a BPF. If the packet matches the filter then the analytic returns negative one because the packet is known to be benign. It returns zero otherwise.

Entropy

This analytic is engaged by setting the entropy threshold with the *entropy* configuration item. This analytic computes the Shannon entropy of the first *depth* characters of the packet pay load. It compares this entropy to the entropy threshold contained in the *entropy* configuration item. If the entropy is greater than the threshold this analytic returns negative one. It returns zero otherwise.

Keep

This analytic is engaged by setting the *keep* configuration item. It is used to keep any packet that is not known to be benign.

MaliciousBPF

This analytic is engaged by defining the *malicious_bpf* configuration item which contains the text of a BPF or a file contain the text of a BPF. If the packet matches the filter then the analytic returns one because the packet is known to be malicious. It returns zero otherwise.

NotTCP

This analytic is engaged by setting the *keepnontcp* configuration item. It returns one if the packet does not use the Transport Control Protocol (TCP) and zero otherwise.

TaintedFlow

This analytic is engaged by setting the *flow_threshold* configuration item. It tracks TCP flows and compares the number of packets or bytes in the flow to the *flow_threshold*. If the threshold has not been exceeded, then the analytic returns one. If the threshold has been exceeded, then the analytic return zero unless the flow has been tainted. If the *filter* configuration items has been specified, it is the name of a file containing a Bloom filter of malicious n-grams. While the threshold has not be met the packet pay load is divided into a n-grams and compared against the n-grams in the Bloom filter. If there is a match, the flow is tainted. Packets from tainted flows are always scored as one regardless flow threshold.

CONFIGURATION ITEMS

The *lints* program leverages the **Configuration** class which defines a list of configuration items (CIs) which may be set through a configuration file, the environment, and the command line. The class also supports some standard command line options such as *help* and *version* which should not appear in a configuration file or the environment.

The default configuration file is *.lintsrc* in the user's home directory. Each line of the configuration file contains the name of the CI a separator and the value of the configuration item. Separators may be a colon, the equals sign, or white space. CIs are first read from the default configuration file, then imported from the environment, and finally processed from the command line. CIs set in the environment will overwrite items set in a the default configuration file, and items set on the command line will overwrite items set in the environment. However, configuration files my specified in configuration file, the environment, and on the command line. Configuration files are processes as they are encountered. This means that items defined in a configuration file specified in a configuration file will over write previous items but be overwritten by

subsequent items. This is also true in the environment and command line. There is one important caveat. To prevent eternal loops, configuration files may only be read once. If a same file is called again, it will ignored.

The following table displays the name of the CI, the argument specification of the CI, the name of the environment variable, the short option, and the long option. The **Configuration** class leverages the **getopt_long** function from the standard library, and the argument specifications correspond to the values of the *has_arg* element of the **option** structure. These are **no_argument**, **required_argument**, or **optional_argument**. If the specification is **no_argument**, then the argument is a flag that is either defined or undefined. Values may be assigned in configuration files or the environment, but these values are ignored. If the specification is **required_argument**,**then**a**value**must**be**defined**and**the**value**is**used**in some way. If the specification is **optional_argument**, then the argument may be act like a flags, but the value is used. These options are specified on the command line by the lack of space for short options and by the equals sign for long options. (e.g. the option foo which a short option of "f" and long option of "foo" taking the optional argument "bar" would be specified as either "-fbar or --foo=bar.) Required options may be specified this way as well, but a space between the option and argument of the option is also allowed. The **Configuration** class adds **list_argument** to this list. The **list_argument** is an extension of the **required_argument**. There exists one and only one value for required, optional, or no argument CIs and each subsequent definition of the option over writes the previous definition. List argument CIs may have several arguments and each subsequent definition add a value to the list.

Name	Argument	Environment	Short	Long
alarm	no	n/a	−a	−−alarm
benign_bpf	required	n/a	−B	−−benign_bpf
config	list	LINTS_CONFIG	−c	−−config
depth	required	LINTS_DEPTH	−d	−−depth
elapsedtime	required	ELAPSEDTIME_FORMAT	−E	−−etmfmt
entropy	required	LINTS_ENTROPY	−e	−−entropy
filter	required	LINTS_FILTER	−f	−−filter
flow_feature	required	n/a	−s	−−flow_feature
flow_threshold	requiredq	n/a	−C	−−flow_threshold
help	no	n/a	−h	−−help
keep	no	n/a	−K	−−keep
keepnontcp	no	n/a	−k	−−keepnontcp
logman_ident	required	LINTS_IDENT	−I	−−ident
logman_facility	required	LINTS_FACILITY	−F	−−facility
logman_logfile	required	LINTS_LOGFILE	−l	−−logfile
logman_loglevel	required	LINTS_LOGLEVEL	−L	−−loglevel
logman_logtype	required	LINTS_LOGTYPE	−M	−−logtype
logman_options	list	LINTS_OPTIONS	−O	−−options
logman_quite	no	LINTS_QUIET	−q	−−quiet
malicious_bpf	required	n/a	−m	−−malicious_bpf
procpacap_bpf	required	n/a	−b	−−bpf
procpcap_count	required	n/a	−n	−−number
procpcap_interface	required	n/a	−i	−−interface
procpcap_promisc	required	n/a	−p	−−promisc
procpcap_readfile	list	n/a	−r	−−readfile
procpcap_snaplen	required	n/a	−S	−−snaplen
procpcap_to_ms	required	n/a	−t	−−timeout
rusage_format	required	RUSAGE_FORMAT	−u	−−rusagefmt
threshold	required	LINTS_THRESHOLD	−t	−−threshold
version	no	n/a	−v	−−version
writefile	required	LINTS_WRITEFILE	−w	−−whitefile

alarm
The **alarm** configuration item requires an argument which specifies the timer in seconds. This item sets an alarm to terminate the program after the specified number of seconds. Some programs may run a very long time and others like the one that sniff the network traffic have no way of know when to stop. This configuration item allows the user to set a timer. At the end of the _timer_ specified in seconds, the program will generate an alarm signal or **SIGALRM** which will instruct the program to clean up and terminate.

benign_bpf
The B benign_bpf configuration item specifies a BPF for known benign packets or the name of a file containing a BPF for known benign packets. Packets that match this filter will be excluded from the output.

config
The **config** configuration item is a list of arguments which specifies configuration files to read.

depth
The **depth** configuration item specifies how many character from pay load should be included in the entropy computation or in comparing n-grams to the malicious Bloom filter.

elapsedtime_format
The *lints* program leverages the **RUsage** class to log resource usage information. One component of this is the elapsed time. The argument to this CI contains an elapsed time format string which is inspired by format used in the **strftime** function. See **strfetime**(3) for the details about the format string.

entropy
The **entropy** configuration item sets the entropy threshold for packets. Packets that have a pay load entropy greater than the threshold will be excluded from the output.

filter
The **filter** configuration item specifies the name of a file containing the Bloom filter of malicious n-grams.

flow_feature
The **flow_feature** configuration item specifies which of the flow's features to compare against the threshold. The features "bytes" and "packets" are supported.

flow_threshold
The **flow_threshold** configuration item specifies the threshold in either bytes or packets. Once this threshold has been exceeded packets will be excluded from the output unless the flow is tainted.

help
The **help** configuration items is a flag which does not take an argument on the command line. It could be set to "yes" or "true" in a configuration file, but this would not be very useful since it causes the program print usage information and exit.

keep
The **keep** configuration item instructs *lints* to keep all packets that are not known to be benign.

keepnontcp
The **keepnontcp** configuration item instructs *lints* to keep all packets that do not employ TCP.

logman_ident
The **logman_ident** configuration item is inspired by the *ident* option to the **openlog** function. It is prepended to every log message to allow one to determine which program sent each message. If the **logman_logtype** is set to *SYS_LOG*, then this is passed to the **syslog** system using the **openlog** function. If the **logman_logtype** is set to **LOG_FILE**, then this will be prepended to every log message printed to the log file.

logman_facility
The **logman_facility** configuration item has a required argument. It is inspired by the *facility* option to the **openlog** function. It is used to specify what type of program is logging the message. This allows the **syslog** system to handle message from different facilities differently. See **syslog**(3) for a list of facilities, their macros, and their meanings.

logman_logfile
The **logman_logfile** configuration item has a required argument. It is used in the *LOG_FILE* log type and specify the name of the log file. The default is system error.

logman_loglevel
The **logman_loglevel** configuration item has a required argument. It is used to determine which log messages are written. The concept is based upon the priority level in the **syslog** system. In this system message is assigned a priority when it is logged. The function **syslogmask** is used to set the mask. The priority of each message is checked against the mask to determine if the message will be written.

logman_options
The **logman_options** configuration item contains a list of options. Each item in the list may be specified by a separate configuration file entry or command line option. The options may also be separated by a vertical bar or the pipe symbol which is also used as a unary **OR** operator.

logman_quiet
The **logman_quiet** configuration item is a flag which does not take an argument on the command line. It could be set to "yes" or "true" in a configuration file. It suppresses all but the most critical log messages by setting the *logman__loglevel* to *error*.

logman_logtype
The **logman_logtype** configuration item has a required argument that specifies the type of logging system to be used for this invocation of the program. Currently there are 2 supported logging systems *LOG_FILE* and *SYS_LOG*. The *LOG_FILE* system writes log messages to a file specified by the **logman_logfile** configuration item. It does not support the facility feature, and cannot support all of the options. The *SYS_LOG* system leverages the **syslog** and supporting functions from the **UNIX** standard library.

malicious_bpf
The **malicious_bpf** contains the text of a BPF for known malicious traffic or the name of a file containing the text. Packets matching this filter will be included in the output.

procpcap_bpf
The argument of this CI is a file containing a Berkeley packet filter (BPF) or text of BPF.

procpcap_count
This argument of this CI is the maximum number of packets to read.

procpcap_interface
The argument of this CI is the network interface from which to read packets.

procpcap_promisc
The argument of this CI is set as the promiscuous mode for the network interface.

procpcap_readfile
This CI adds another file to the list of network capture files to read.

procpcap_snaplen
This CI contains the snap length or the maximum length of a packet to read.

procpcap_to_ms
This CI set the timeout in microseconds.

rusage_format
The **rusage_format** configuration item has a required argument that specifies the format in which resource usage information will be displayed. The format is inspired by format used by the GNU **time** command. See the **strfrusage** function for the details about the format string.

threshold
The **threshold** configuration item is compared against the score of each analytic. If the score is greater or equal than the threshold, the packet is included in the output. If the score is less than the threshold, the packet is excluded from the the output.

version
The **version** configuration item is a flag which does not take an argument on the command line. It could be set to "yes" or "true" in a configuration file, but this would not be very useful since it causes the program to print version information and exit.

writefile
The name of the file where the interesting packets are written.

DIAGNOSTICS

Lints leverages the **LogManager** class for all diagnostic messages. At the *information* level *lints* logs the file name, PCAP version, data link name, data link description and the time it began processing. Upon completion it logs the number of packets and bytes read and the number of packets and bytes written. It also logs a resource usage message.

FILES

~/.lintsrc
is the default configuration file.

EXAMPLE

The following examples will use the Information Security Centre of Excellence data set from 2012. This data set contained 5 days. The first day was for training and contained no malicious activity. These examples use the following four days combined into a single network capture file called *iscx2012te.pcap*. This file was analyzed by **Snort** and the output was used with the **pcappurge** program to separate the malicious traffic from the benign traffic which were placed into separate network capture files. These network capture files were used by the **mkbf** program to create the *iscx2012te-mal.bflt* file which contains a Bloom filter of malicous n-grams. Since the command lines for these examples can become quite long, a configuration file will be used.

The following example uses *lints* to remove all of the TCP traffic from the network capture.

```
$ cat lints1.conf
keepnontcp
procpcap_readfile = iscx2012te.pcap
writefile = /dev/null
$ lints -c lints1.conf
lints: Processing iscx2012te.pcap version 2.4 EN10MB (Ethernet) at Thu Dec 17 14:25:48 2020
lints: 16198584 packets and 11918089119 bytes read.
        412890 packets and 71010235 bytes written.
        97.45% packet savings 99.40% byte savings.
lints: 0:0:6.88user 0:0:4.16system 0:1:2.63elapsed 17.63CPU (0text+0data) 3180max)k
        22203272inputs+0output (0major+597minor) pagefaults 0swaps
```

The following example will remove all TCP packets that have a pay load with an entropy score score of 7.0 or more.

```
$ cat lints2.conf
entropy = 7.0
keep
keepnontcp
procpcap_readfile = iscx2012te.pcap
writefile = /dev/null
$ lints -c lints2.conf
lints: Processing iscx2012te.pcap version 2.4 EN10MB (Ethernet) at Thu Dec 17 14:43:43 2020
lints: 16198584 packets and 11918089119 bytes read.
        13783302 packets and 8481402874 bytes written.
        14.91% packet savings 28.84% byte savings.
lints: 0:0:49.19user 0:0:4.91system 0:1:5.44elapsed 82.67CPU (0text+0data) 3736max)k
        22288024inputs+0output (2major+521minor) pagefaults 0swaps
```

The following example will stop writing TCP packets ater 10 packets have been read in that flow.

```
$ cat lints3.conf
```

flow_feature = packets
flow_threshold = 10
keepnontcp
procpcap_readfile = iscx2012te.pcap
writefile = /dev/null
$ *lints -c lints3.conf*
lints: Processing iscx2012te.pcap version 2.4 EN10MB (Ethernet) at Thu Dec 17 14:52:02 2020
lints: 16198584 packets and 11918089119 bytes read.
4027520 packets and 1373164498 bytes written.
75.14% packet savings 88.48% byte savings.
lints: 0:0:16.00user 0:0:4.83system 0:1:4.17elapsed 32.46CPU (0text+0data 14476max)k
22287792inputs+0output (1major+3208minor) pagefaults 0swaps

The following example will stop writing TCP packets ater 1550 bytes have been read in that flow.

$ *cat lints4.conf*
flow_feature = bytes
flow_threshold = 1500
keepnontcp
procpcap_readfile = iscx2012te.pcap
writefile = /dev/null
$ *lints -c lints4.conf*
lints: Processing iscx2012te.pcap version 2.4 EN10MB (Ethernet) at Thu Dec 17 15:13:59 2020
lints: 16198584 packets and 11918089119 bytes read.
3977400 packets and 415412026 bytes written.
75.45% packet savings 96.51% byte savings.
lints: 0:0:15.76user 0:0:4.74system 0:1:3.58elapsed 32.25CPU (0text+0data 14584max)k
22287792inputs+0output (1major+3214minor) pagefaults 0swaps

The following example will stop writing TCP packets ater 1550 bytes have been read in that flow unless the flow has been tainted by comparing n-grams in the packet pay load with malicious n-grams in a Bloom filter.

$ *cat lints5.conf*
filter = iscx2012te-mal.bflt
flow_feature = bytes
flow_threshold = 1500
keepnontcp
procpcap_readfile = iscx2012te.pcap
writefile = /dev/null
$ *lints -c lints4.conf*
lints: Processing iscx2012te.pcap version 2.4 EN10MB (Ethernet) at Thu Dec 17 15:18:08 2020
lints: 16198584 packets and 11918089119 bytes read.
4019671 packets and 479398865 bytes written.
75.19% packet savings 95.98% byte savings.
lints: 0:7:55.27user 0:0:6.94system 0:8:4.02elapsed 99.63CPU (0text+0data 539028max)k
23344464inputs+0output (2major+132244minor) pagefaults 0swaps

SEE ALSO

bfltmerge(1), **dispbflt**(1), **mkbf**(1), **pcappurge**(1), **pcap**(3pcap). **Configuration**(3), **LogManager**(3), **ProcessPacket**(3), **ProcessPcap**(3), **RUsage**(3), **strfetime**(3), **strfrusage**(3), **syslog**(3).

AUTHOR(S)

Sidney C. Smith, U.S. Army Combat Capabilities Development Command, Army Research Laboratory, Army Research Directorate.

NAME

run_lints_exp – conduct trials using the lints and snort.

SYNOPSIS

run_lints_exp [**options**] *dataset*

DESCRIPTION

Run_lints_exp conducts a trial of an experiment using the Lossy Intelligent Network Traffic Squeezer (LINTS). This trial uses *lints*(1) to compress the traffic at various thresholds controlled by the *sequence*(1) program. It then feeds these compressed file to *snort*(1) for analysis. A directory for each threshold is created where the output of *lints* and *snort* are stored. Attributes from the output are extracted and a table is printed in a space delimited value format. The name of this file is constructed using the name of the compressor, the short name of the data set, the short name of the rule set, a 't', the trial number, an 'r', and the run number. For example the first run of the sixth trial of an experiment using *lints* to compress the small sample file from the DARPA 98 data set analyzed using the Circa2000 *snort* rule set would have the following name: **lintsd98ssc2000t6r1.dat**. This table contains the threshold, the number of packet received by *snort*, the number of alerts detected by *snort*, the number of bytes written by *lints*, the compression rate, and the alert loss rate (ALR). An example of this table is provided below.

Threshold	Received	Alerts	Bytes	Compression	ALR
5	3296	230	444182	19.00	0.00
10	4846	230	842280	36.00	0.00

OPTIONS

–b ––basedir
Base directory for the Cybersecurity Research Data Set Infrastructure.

–c ––config
the configuration file for *lints*.

–h ––help
print a help message and exit successfully.

–i ––iv_switch
switch or command line option for the independent variable. This option determines which of lints options will be given the threshold as an argument.

–r ––ruleset
the key in the *rule set* table of the Data Set Database for the *snort* rule set.

–s ––seqconf
set the configuration file for the *sequence* program.

–v ––version
print a version message and exit successfully.

CAVEAT

The *run_lints_exp* program tests for the existence of the threshold directory. If the directory exists, *run_lints_exp* assumes that the iteration has been completed in an earlier run. It attempts to extract the information out of the *lints.out* and *snort.out* files and moves on to the next iteration.

VERSION

0.1.0 Initial development.

COPYRIGHT

SEE ALSO

dsdb(1), **lints**(1), **sequence**(1), **snort**(1).

DIAGNOSTICS

Diagnostic messages are written to standard error.

AUTHOR(S)

Sidney C. Smith, U.S. Army, Combat Capabilities Development Command, Army Research Laboratory, Army Research Directorate.

Appendix B. Shell Scripts

Listing 25: Code for the runlints.sh script

```
1  #!/bin/sh
2
3  BASEDIR=/app/csrdsi
4  LOGFILE=runlints.log
5  REPORTSRC="${BASEDIR}/templates/lints/lints_plots.tex"
6  source ${BASEDIR}/etc/profile
7
8  printf "starting␣run␣at␣%s\n" "`date`" >> "${LOGFILE}"
9
10 #
11 # load configuration.
12 #
13 if [ -f ./runlints.conf ]; then
14     echo pulling configuration from ./runlints.conf >>
15         "$LOGFILE"
16     source ./runlints.conf
17 fi
18
19 #
20 # pull configuration from directory structure.
21 #
22 PWD=`pwd`
23 RUN=`basename $PWD`
24 RUNNUM=`echo "${RUN}" | sed s/run//`
25 REMAINDER=`echo "${PWD}" | sed s/"${RUN}$"//`
26 TRIAL=`basename $REMAINDER`
27 TRIALNUM=`echo "$TRIAL" | sed s/trial//`
28 REMAINDER=`echo "$REMAINDER" | sed s/"${TRIAL}\/"//`
29 RULESETSN=`basename "${REMAINDER}"`
30 REMAINDER=`echo ${REMAINDER} | sed s/"${RULESETSN}\/"//`
31 DATASETSN=`basename ${REMAINDER}`
32 REMAINDER=`echo ${REMAINDER} | sed s/"${DATASETSN}\/"//`
33 TOOL=`basename ${REMAINDER}`
34 BASE=`printf "%s%s%st%sr%s"\
```

```
35      $TOOL $DATASETSN $RULESETSN $TRIALNUM $RUNNUM‘
36  echo BASE = ${BASE} >> "${LOGFILE}"
37
38  #
39  # Test the data set.
40  #
41  if [ -z "${DATASET}" -o "${DATASET}" = "dataset" ]; then
42      echo setting data set from short name ${DATASETSN} >>
43          "$LOGFILE"
44      DATASET=‘dsdb -q -s -d "${DATASETSN}" -o name‘
45  fi
46  dsdb -q -d "${DATASET}" -o location > /dev/null 2>&1
47  if [ $? -ne 0 ]; then
48      printf "data␣set␣\"%s\"␣not␣found.\n" \
49          "${DATASET}" >> "${LOGFILE}"
50      exit -1
51  fi
52  echo DATASET = ${DATASET} >> "${LOGFILE}"
53
54  #
55  # Test the rule set.
56  #
57  if [ -z "${RULESET}" -o "${RULESET}" = "ruleset" ]; then
58      echo setting rule set from short name ${RULESETSN} >>
59          "$LOGFILE"
60      RULESET=‘dsdb -q -s -r "${RULESETSN}" -o name‘
61  fi
62  dsdb -q -r "${RULESET}" -o location > /dev/null 2>&1
63  if [ $? -ne 0 ]; then
64      printf "rule␣set␣\"%s\"␣not␣found.\n" \
65          "${RULESET}" >> "${LOGFILE}"
66      exit -1
67  fi
68  echo RULESET = ${RULESET}  >> "${LOGFILE}"
69
```

```
70 #
71 # Test the alerts.
72 #
73 dsdb -q -d "${DATASET}" -r "${RULESET}" -o alerts \
74     > /dev/null 2>&1
75 if [ $? -ne 0 ]; then
76     printf "alerts␣for␣\"%s\"␣and␣\"%s\"␣not␣found.\n" \
77         "${DATASET}" "${RULESET}" >> "${LOGFILE}"
78     exit -1
79 fi
80
81
82 #
83 # Test the Bloom filter is any.
84 #
85 FILTER=`grep '^filter' lints.conf |
86     sed 's/filter[[:space:]:=]*//'`
87 if [ "${FILTER}" ]; then
88     if [ ! -f "${FILTER}" ]; then
89         printf "filter␣\"%s\"␣does␣not␣exist.\n" \
90             "${FILTER}" >> "${LOGFILE}"
91         exit -1
92     fi
93 fi
94 echo FILTER = ${FILTER}  >> "${LOGFILE}"
95
96 #
97 # Test independent variable.
98 #
99 if [ -z "${IV}" ]; then
100         printf "Setting␣the␣Independent␣variable␣to" >>
101                 "${LOGFILE}"
102     if [ "${TRIALNUM}" = "2" ]; then
103
104         echo "␣an␣entropy␣threshold." >> "${LOGFILE}"
```

```
105             IV="-e"
106         else
107             echo "␣a␣flow␣threshold." >> "${LOGFILE}"
108             IV="-C"
109         fi
110 fi
111 echo IV = ${IV}  >> "${LOGFILE}"
112
113
114 #
115 # Run run_lint_exp
116 #
117 echo "run_lints_exp␣starting␣at␣`date`" \
118     > run_lints_exp.out
119 run_lints_exp -c lints.conf -s sequence.conf -i "${IV}" \
120     -r "${RULESET}" "${DATASET}" \
121     >> run_lints_exp.out 2>&1
122 echo "run_lints_exp␣complete␣at␣`date`" \
123     >> run_lints_exp.out
124
125 #
126 # Generate the report.
127 #
128 printf "\\def␣\\\\experiment␣{%s}\n" "${BASE}" >
129             tailor.tex
130 printf "\\def␣\\dataset␣{%s}\n" "${DATASET}" >>
131             tailor.tex
132 printf "\\def␣\\\\ruleset␣{%s}\n" "${RULESET}" >>
133             tailor.tex
134 printf "\\def␣\\datafile␣{%s}\n" "${BASE}.dat" >>
135             tailor.tex
136 printf "\\def␣\\script␣{%s}\n" "runlints.conf" >>
137             tailor.tex
138 pdflatex ${REPORTSRC} >> "${LOGFILE}" 2>&1
139 pdflatex ${REPORTSRC} >> "${LOGFILE}" 2>&1
```

```
140
141 printf "ending␣run␣at␣%s\n" "`date`" >> "${LOGFILE}"
```

Appendix C. Raw Results

This appendix contains the raw results for the various runs conducted in this report.

C.1 Netcomp vs. LINTS comparisons

Table C-1: Results from Netcomp compression using the ISCX 15 June 2012 data set and the RegAug2013 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
0.00	27,035.626	8,822	88	0.04
0.05	25,716.796	8,821	88	0.05
0.10	23,436.601	8,822	87	0.04
0.15	20,937.896	8,822	86	0.04
0.20	19,580.141	8,822	84	0.04
0.25	18,644.113	8,822	83	0.04
0.30	16,435.880	8,822	81	0.04
0.35	16,974.986	8,822	79	0.04
0.40	9,734.898	8,822	76	0.04
0.45	10,613.656	8,821	74	0.05
0.50	10,791.436	8,820	72	0.06
0.55	10,650.147	8,820	69	0.06
0.60	10,685.992	8,819	67	0.07
0.65	10,571.559	8,818	64	0.09
0.70	9,556.099	8,817	61	0.10
0.75	9,663.522	8,819	58	0.07
0.80	9,610.537	8,818	54	0.09
0.85	8,556.455	8,818	50	0.09
0.90	6,990.071	8,818	45	0.09
0.95	8,488.316	8,804	39	0.24
0.96	0.000	8,786	38	0.45
0.97	0.000	8,786	36	0.45
0.98	0.000	8,783	34	0.48
0.99	0.000	8,747	31	0.89
1.00	7,805.806	0	30	100.00
		Concluded		

Table C-2: Results from LINTS compression using the ISCX 15 Jun 2012 data set and the RegAug2013 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
5	969.850	8,801	8	0.28
10	1,017.160	8,803	12	0.26
15	1,051.145	8,813	15	0.14
20	1,219.606	8,813	17	0.14
25	1,220.665	8,813	20	0.14
30	1,120.684	8,813	22	0.14
35	1,329.771	8,813	24	0.14
40	1,324.526	8,813	25	0.14
45	1,265.646	8,813	27	0.14
50	1,444.709	8,814	29	0.13
55	1,287.354	8,813	30	0.14
60	1,175.424	8,815	32	0.12
65	1,180.434	8,814	33	0.13
70	1,190.598	8,813	35	0.14
75	1,191.841	8,813	36	0.14
80	1,217.281	8,813	38	0.14
85	1,345.991	8,813	39	0.14
90	1,269.771	8,813	40	0.14
95	1,614.368	8,813	41	0.14
100	1,494.821	8,813	42	0.14

Concluded

Table C-3: Results from Netcomp compression using the CIC 5 July 2017 data set and the RegSep2018 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
0.00	4,233.673	1,056	43	0.00
0.05	4,547.427	1,056	42	0.00
0.10	4,516.364	1,056	42	0.00
0.15	4,499.578	1,056	42	0.00
0.20	4,806.736	1,056	42	0.00
0.25	4,686.023	1,056	41	0.00
0.30	4,484.712	1,056	41	0.00
0.35	4,426.602	1,056	41	0.00
0.40	4,565.711	1,056	41	0.00
0.45	4,551.515	1,056	40	0.00
0.50	4,633.092	1,056	40	0.00
0.55	4,679.335	1,056	39	0.00
0.60	4,612.341	1,056	39	0.00
0.65	4,657.358	1,056	38	0.00
0.70	4,607.975	1,056	38	0.00
0.75	4,641.972	1,056	37	0.00
0.80	4,409.616	1,056	36	0.00
0.85	4,574.932	1,055	35	0.09
0.90	4,466.019	1,046	34	0.94
0.91	4,896.321	1,044	33	1.13
0.92	4,624.850	1,040	33	1.51
0.93	4,624.850	1,036	32	1.89
0.94	4,717.414	1,034	32	2.08
0.95	4,537.750	1,034	31	2.08
1.00	4,590.866	0	10	100.00
Concluded				

Table C-4: Results from LINTS compression using day 3 of the CIC-IDS data set and the RegSep2018 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
5	327.106	998	17	3.48
10	348.088	1,034	20	0.00
15	355.335	1,034	21	0.00
20	350.046	1,034	21	0.00
25	312.415	1,034	21	0.00
30	313.235	1,034	21	0.00
35	320.679	1,034	21	0.00
40	342.128	1,034	21	0.00
45	363.251	1,034	22	0.00
50	337.001	1,034	22	0.00
55	344.973	1,034	22	0.00
60	339.448	1,034	22	0.00
65	348.364	1,034	22	0.00
70	355.381	1,034	22	0.00
75	337.791	1,034	22	0.00
80	292.283	1,034	22	0.00
85	313.286	1,034	22	0.00
90	309.487	1,034	22	0.00
95	359.338	1,034	22	0.00
100	350.941	1,034	22	0.00

Concluded

C.2 LINTS Trials

C.2.1 Entropy

Table C-5: Results using the LINTS entropy strategy to compress the CDX 2009 usama-gator010 data set and the Circa 2009 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
5.0	22.695	197	57	61.22
5.1	23.349	194	57	61.81
5.2	24.020	196	58	61.41
5.3	24.330	209	59	58.85
5.4	26.045	228	60	55.11
5.5	27.874	295	61	41.92
5.6	27.559	325	62	36.02
5.7	27.782	348	62	31.49
5.8	27.872	359	62	29.33
5.9	27.893	365	62	28.14
6.0	28.390	380	63	25.19
6.1	28.988	381	63	25.00
6.2	29.139	381	64	25.00
6.3	29.928	381	64	25.00
6.4	29.636	381	64	25.00
6.5	29.120	381	64	25.00
6.6	28.957	381	64	25.00
6.7	29.170	381	64	25.00
6.8	31.719	381	67	25.00
6.9	31.191	380	68	25.19
7.0	31.459	478	69	5.90
7.1	31.937	495	70	2.55
7.2	32.668	497	72	2.16
7.3	32.961	499	73	1.77
7.4	32.449	499	73	1.77
7.5	33.716	500	74	1.57
7.6	34.760	501	75	1.37

Continued on next page

Table C-5 Continued from previous page

Threshold	Duration	Alerts	Size (%)	ALR (%)
7.7	34.318	509	76	−0.19
7.8	35.186	509	77	−0.19
7.9	34.517	509	99	−0.19
8.0	34.920	508	100	0.00

Concluded

Table C-6: Results using the LINTS entropy strategy to compress the CDX 2009 usama-gator020 data set and the Circa 2009 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
5.0	340.590	9,507	21	15.39
5.1	597.984	9,513	22	15.34
5.2	546.528	9,532	22	15.17
5.3	556.986	9,637	22	14.23
5.4	517.012	9,700	22	13.67
5.5	528.214	9,768	23	13.07
5.6	520.137	9,844	23	12.39
5.7	501.649	9,856	23	12.28
5.8	523.413	9,876	23	12.11
5.9	514.428	9,879	23	12.08
6.0	374.724	11,108	23	1.14
6.1	353.936	11,131	24	0.94
6.2	397.822	11,131	25	0.94
6.3	388.032	11,131	25	0.94
6.4	384.008	11,131	25	0.94
6.5	375.663	11,131	25	0.94
6.6	356.177	11,131	25	0.94
6.7	331.088	11,131	25	0.94
6.8	372.974	11,131	25	0.94
6.9	344.849	11,137	26	0.88
7.0	269.891	11,172	26	0.57
7.1	271.680	11,208	26	0.25
7.2	272.660	11,212	26	0.22
7.3	273.113	11,216	27	0.18
7.4	275.660	11,218	27	0.16
7.5	279.875	11,218	29	0.16
7.6	281.699	11,227	30	0.08
7.7	286.054	11,236	32	0.00
7.8	302.149	11,236	37	0.00

Continued on next page

Table C-6 Continued from previous page

Threshold	Duration	Alerts	Size (%)	ALR (%)
7.9	396.225	11,237	99	0.00
8.0	392.186	11,237	100	0.00

Concluded

Table C-7: Results using the LINTS entropy strategy to compress the ”’ 2010 data set and the Circa 2009 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
5.0	1,212.050	460,600	70	43.83
5.1	1,296.171	467,695	71	42.97
5.2	1,403.198	511,873	76	37.58
5.3	1,394.257	513,161	77	37.42
5.4	1,488.519	519,572	80	36.64
5.5	1,837.299	815,709	84	0.53
5.6	1,903.091	816,327	84	0.46
5.7	1,896.140	816,486	85	0.44
5.8	1,903.166	816,761	85	0.40
5.9	1,927.904	817,029	85	0.37
6.0	1,906.461	817,472	85	0.32
6.1	1,970.198	817,592	86	0.30
6.2	1,931.063	817,749	86	0.28
6.3	1,926.560	818,012	87	0.25
6.4	1,963.295	818,225	87	0.23
6.5	1,988.425	818,340	87	0.21
6.6	1,927.735	818,555	87	0.19
6.7	1,925.859	818,589	87	0.18
6.8	1,930.790	818,632	87	0.18
6.9	1,948.246	818,726	89	0.16
7.0	1,931.965	818,980	92	0.13
7.1	2,008.120	819,279	93	0.10
7.2	1,966.819	819,470	93	0.07
7.3	1,974.725	819,493	93	0.07
7.4	1,943.942	819,542	94	0.06
7.5	1,930.679	819,586	94	0.06
7.6	2,044.432	819,715	95	0.04
7.7	1,975.896	819,839	95	0.03
7.8	1,955.192	819,838	96	0.03

Continued on next page

Table C-7 Continued from previous page

Threshold	Duration	Alerts	Size (%)	ALR (%)
7.9	2,032.868	820,092	99	0.00
8.0	2,062.673	820,112	100	0.00

Concluded

Table C-8: Results using the LINTS entropy strategy to compress the MACCDC 2011 data set and the RegAug2013 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
5.0	507.566	20,078	33	99.68
5.1	523.560	20,150	35	99.68
5.2	548.038	1,009,460	37	84.04
5.3	660.393	4,785,277	41	24.37
5.4	735.725	6,694,894	45	−5.80
5.5	817.887	7,941,500	48	−25.50
5.6	828.771	7,973,003	48	−25.99
5.7	783.691	7,977,525	48	−26.07
5.8	812.378	7,978,843	49	−26.09
5.9	811.671	7,980,230	49	−26.11
6.0	911.911	7,991,417	49	−26.29
6.1	843.013	8,008,998	50	−26.56
6.2	822.590	8,018,930	50	−26.72
6.3	815.374	8,019,527	50	−26.73
6.4	829.664	8,019,989	50	−26.74
6.5	809.663	8,020,158	50	−26.74
6.6	812.486	8,020,051	50	−26.74
6.7	845.395	8,030,449	51	−26.90
6.8	883.753	8,373,250	52	−32.32
6.9	949.853	6,356,437	60	−0.45
7.0	1,216.919	6,351,787	62	−0.37
7.1	1,162.874	6,352,138	62	−0.38
7.2	1,171.252	6,352,457	63	−0.38
7.3	1,170.462	6,352,645	63	−0.39
7.4	1,170.445	6,353,058	63	−0.39
7.5	1,173.756	6,353,640	64	−0.40
7.6	1,176.763	6,329,179	64	−0.02
7.7	1,192.231	6,327,803	65	0.00
7.8	1,205.270	6,327,803	66	0.00

Continued on next page

Table C-8 Continued from previous page

Threshold	Duration	Alerts	Size (%)	ALR (%)
7.9	1,583.413	6,327,805	99	0.00
8.0	1,498.903	6,327,802	100	0.00
			Concluded	

Table C-9: Results using the LINTS entropy strategy to compress the MACCDC 2012 data set and the RegAug2013 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
5.0	250.565	6,375	34	97.03
5.1	296.658	181,924	37	15.52
5.2	296.089	347,818	40	−61.51
5.3	308.354	348,260	43	−61.71
5.4	317.453	351,258	45	−63.11
5.5	348.754	354,528	47	−64.62
5.6	345.630	355,456	48	−65.06
5.7	410.758	355,821	53	−65.22
5.8	460.851	356,116	62	−65.36
5.9	445.745	356,124	65	−65.37
6.0	453.295	356,171	65	−65.39
6.1	447.038	356,172	66	−65.39
6.2	454.496	356,199	66	−65.40
6.3	453.377	356,222	66	−65.41
6.4	457.377	356,235	66	−65.42
6.5	464.535	356,272	66	−65.43
6.6	475.912	356,298	67	−65.45
6.7	475.299	356,359	67	−65.47
6.8	471.018	356,026	67	−65.32
6.9	486.933	214,829	68	0.24
7.0	500.204	214,933	69	0.19
7.1	505.283	214,986	69	0.16
7.2	507.367	215,059	69	0.13
7.3	489.397	215,090	70	0.12
7.4	493.498	215,131	70	0.10
7.5	478.124	215,277	70	0.03
7.6	469.101	215,345	71	0.00
7.7	464.585	215,346	72	0.00
7.8	471.424	215,349	74	0.00

Continued on next page

Table C-9 Continued from previous page

Threshold	Duration	Alerts	Size (%)	ALR (%)
7.9	530.604	215,349	99	0.00
8.0	531.081	215,349	100	0.00

Concluded

Table C-10: Results using the LINTS entropy strategy to compress the ISCX 2012 testing data set and the RegAug2013 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
5.0	191.718	38	25	70.31
5.1	281.972	42	38	67.18
5.2	369.234	47	49	63.28
5.3	425.408	52	56	59.37
5.4	469.350	61	60	52.34
5.5	476.899	76	63	40.62
5.6	519.570	86	65	32.81
5.7	530.536	87	66	32.03
5.8	530.824	93	67	27.34
5.9	553.995	98	67	23.43
6.0	540.887	101	68	21.09
6.1	552.273	102	68	20.31
6.2	543.891	103	68	19.53
6.3	623.726	104	68	18.75
6.4	582.566	109	69	14.84
6.5	570.425	113	69	11.71
6.6	573.439	120	69	6.25
6.7	537.278	123	70	3.90
6.8	563.220	128	70	0.00
6.9	602.567	130	70	−1.56
7.0	581.810	134	71	−4.68
7.1	579.677	138	71	−7.81
7.2	616.284	141	71	−10.15
7.3	611.500	143	72	−11.71
7.4	599.903	143	73	−11.71
7.5	632.971	142	74	−10.93
7.6	633.820	145	76	−13.28
7.7	629.835	147	78	−14.84
7.8	646.545	153	81	−19.53

Continued on next page

Table C-10 Continued from previous page

Threshold	Duration	Alerts	Size (%)	ALR (%)
7.9	770.823	128	99	0.00
8.0	758.113	128	100	0.00

Concluded

Table C-11: Results using the LINTS entropy strategy to compress the UNSW-NB 2015 January data set and the RegSep2018 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
5.0	1,485.723	254	12	34.02
5.1	846.912	256	12	33.50
5.2	891.603	262	12	31.94
5.3	900.096	282	13	26.75
5.4	903.532	292	13	24.15
5.5	1,084.796	301	13	21.81
5.6	1,093.063	315	14	18.18
5.7	1,240.872	322	14	16.36
5.8	1,528.023	345	14	10.38
5.9	1,550.045	349	14	9.35
6.0	1,544.412	361	15	6.23
6.1	1,570.311	374	15	2.85
6.2	1,394.557	374	15	2.85
6.3	1,235.795	374	15	2.85
6.4	1,344.497	374	15	2.85
6.5	1,140.738	377	16	2.07
6.6	1,741.022	377	16	2.07
6.7	1,719.044	378	16	1.81
6.8	1,689.104	375	16	2.59
6.9	1,649.628	375	16	2.59
7.0	1,074.290	378	16	1.81
7.1	654.115	379	16	1.55
7.2	723.399	382	16	0.77
7.3	714.187	381	16	1.03
7.4	715.462	384	16	0.25
7.5	721.133	382	16	0.77
7.6	699.105	385	17	0.00
7.7	740.285	387	18	−0.51
7.8	732.223	387	20	−0.51

Continued on next page

Table C-11 Continued from previous page

Threshold	Duration	Alerts	Size (%)	ALR (%)
7.9	1,363.024	388	99	−0.77
8.0	1,368.271	388	100	−0.77

Concluded

Table C-12: Results using the LINTS entropy strategy to compress the CIC 2017 testing data set and the registered Snort rule set from July 2018

Threshold	Duration	Alerts	Size (%)	ALR (%)
5.0	779.076	4	11	99.68
5.1	643.156	14	12	98.90
5.2	756.562	18	12	98.59
5.3	483.528	19	12	98.51
5.4	497.865	19	13	98.51
5.5	502.112	335	13	73.84
5.6	682.514	1,058	13	17.40
5.7	837.165	1,062	13	17.09
5.8	829.322	1,062	13	17.09
5.9	559.359	1,062	13	17.09
6.0	867.988	1,062	13	17.09
6.1	853.187	1,245	14	2.81
6.2	864.617	1,245	14	2.81
6.3	844.277	1,245	14	2.81
6.4	907.579	1,245	14	2.81
6.5	896.159	1,245	14	2.81
6.6	881.254	1,245	14	2.81
6.7	880.248	1,245	14	2.81
6.8	850.708	1,245	14	2.81
6.9	726.985	1,245	14	2.81
7.0	573.831	1,245	14	2.81
7.1	551.751	1,250	14	2.41
7.2	518.321	1,281	15	0.00
7.3	569.787	1,281	15	0.00
7.4	602.006	1,281	16	0.00
7.5	571.183	1,281	16	0.00
7.6	633.823	1,281	17	0.00
7.7	583.040	1,281	17	0.00
7.8	620.928	1,281	19	0.00

Continued on next page

Table C-12 Continued from previous page

Threshold	Duration	Alerts	Size (%)	ALR (%)
7.9	1,264.774	1,281	55	0.00
8.0	1,324.921	1,281	100	0.00

Concluded

C.2.2 Capped by Packets

Table C-13: Results using the LINTS capped-by-packets strategy to compress the CDX 2009 usama-gator010 data set and the Circa2009 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
5	28.251	383	60	24.60
10	30.967	504	75	0.78
15	31.623	504	77	0.78
20	31.627	506	78	0.39
25	31.650	508	79	0.00
30	32.003	510	79	−0.39
35	31.038	508	80	0.00
40	31.074	507	80	0.19
45	30.848	507	80	0.19
50	30.906	507	81	0.19
55	30.955	508	81	0.00
60	30.997	508	81	0.00
65	30.832	507	81	0.19
70	31.221	507	81	0.19
75	30.788	507	81	0.19
80	31.569	508	82	0.00
85	30.845	509	82	−0.19
90	30.654	509	82	−0.19
95	31.005	509	82	−0.19
100	30.956	509	82	−0.19
			Concluded	

Table C-14: Results using the LINTS capped-by-packets strategy to compress the CDX 2009 usama-gator020 data set and the Circa2009 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
5	434.760	10,981	21	2.27
10	506.521	11,181	23	0.49
15	492.058	11,192	24	0.40
20	510.562	11,195	25	0.37
25	509.365	11,197	25	0.35
30	498.222	11,199	26	0.33
35	518.169	11,206	26	0.27
40	505.593	11,215	26	0.19
45	518.376	11,223	26	0.12
50	488.547	11,227	27	0.08
55	343.585	11,233	27	0.03
60	324.548	11,233	27	0.03
65	391.284	11,233	27	0.03
70	388.032	11,233	27	0.03
75	384.070	11,233	28	0.03
80	379.312	11,233	28	0.03
85	363.510	11,233	28	0.03
90	318.695	11,233	28	0.03
95	378.135	11,236	28	0.00
100	313.099	11,236	28	0.00
			Concluded	

Table C-15: Results using the LINTS capped-by-packets strategy to compress the MACCDC 2012 data set and the RegAug2013 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
5	14,970.799	8,600	23	96.00
10	15,369.097	19,161	28	91.10
15	14,901.636	29,504	30	86.29
20	15,130.876	39,873	32	81.48
25	14,664.731	50,181	33	76.69
30	15,157.994	60,577	34	71.87
35	14,491.715	70,976	35	67.04
40	14,906.160	81,288	36	62.25
45	14,665.500	91,575	36	57.47
50	14,970.390	101,836	37	52.71
55	14,164.498	112,159	38	47.91
60	14,688.997	122,533	38	43.10
65	14,946.348	132,873	39	38.29
70	14,225.873	143,133	39	33.53
75	16,180.215	153,364	39	28.78
80	15,546.665	163,572	40	24.04
85	14,662.237	173,770	40	19.30
90	14,836.894	183,926	40	14.59
95	14,596.802	194,124	41	9.85
100	14,209.269	204,300	41	5.13
			Concluded	

Table C-16: Results using the LINTS capped-by-packets strategy to compress the ISCX 2012 testing data set and the RegAug2013 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
5	104.633	60	6	53.12
10	145.411	83	12	35.15
15	185.971	86	17	32.81
20	211.973	92	21	28.12
25	239.423	99	24	22.65
30	251.495	99	26	22.65
35	274.533	101	29	21.09
40	286.070	101	31	21.09
45	292.841	101	33	21.09
50	314.126	114	34	10.93
55	309.759	115	36	10.15
60	336.327	115	37	10.15
65	344.408	115	38	10.15
70	344.585	115	40	10.15
75	363.572	115	41	10.15
80	369.631	115	42	10.15
85	356.523	115	43	10.15
90	388.412	116	44	9.37
95	389.690	116	45	9.37
100	370.286	116	45	9.37
			Concluded	

Table C-17: Results using the LINTS capped-by-packets strategy to compress the UNSW-NB 2015 January data set and the RegSep2018 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
5	1,457.467	70	2	81.81
10	1,159.504	133	5	65.45
15	1,237.389	150	7	61.03
20	1,112.152	150	8	61.03
25	926.805	157	12	59.22
30	907.265	165	17	57.14
35	957.466	166	22	56.88
40	1,036.465	165	26	57.14
45	1,703.273	169	30	56.10
50	1,700.112	174	32	54.80
55	1,678.996	176	34	54.28
60	1,694.598	178	36	53.76
65	974.363	181	38	52.98
70	1,581.628	192	39	50.12
75	1,024.577	190	40	50.64
80	1,476.501	194	41	49.61
85	1,900.231	198	42	48.57
90	1,866.847	204	43	47.01
95	1,867.504	208	43	45.97
100	1,260.996	208	44	45.97
105	977.742	208	45	45.97
110	799.796	212	45	44.93
115	842.980	215	46	44.15
120	821.231	220	46	42.85
125	865.301	221	47	42.59
130	830.780	223	48	42.07
135	861.097	226	48	41.29
140	841.528	230	49	40.25
145	858.776	237	49	38.44

Continued on next page

Table C-17 Continued from previous page

Threshold	Duration	Alerts	Size (%)	ALR (%)
150	850.844	233	50	39.48
155	856.636	236	51	38.70
160	831.878	235	51	38.96
165	832.886	253	52	34.28
170	872.452	246	52	36.10
175	868.948	246	53	36.10
180	898.601	247	53	35.84
185	867.618	254	54	34.02
190	865.853	254	55	34.02
195	843.123	255	55	33.76
200	885.985	254	55	34.02
205	1,412.477	254	56	34.02
210	1,461.697	262	57	31.94
215	1,951.953	273	57	29.09
220	1,653.496	266	58	30.90
225	1,598.850	269	58	30.12
230	1,923.028	273	59	29.09
235	1,809.868	274	59	28.83
240	1,646.972	286	60	25.71
245	1,667.807	274	60	28.83
250	1,639.429	279	61	27.53
255	1,841.807	279	61	27.53
260	1,688.885	287	62	25.45
265	1,637.435	295	62	23.37
270	1,687.103	290	63	24.67
275	1,747.881	296	63	23.11
280	1,636.369	294	64	23.63
285	1,805.885	302	64	21.55
290	1,770.569	305	65	20.77
295	1,674.530	304	65	21.03
300	1,625.469	306	66	20.51

Continued on next page

Table C-17 Continued from previous page

Threshold	Duration	Alerts	Size (%)	ALR (%)
305	1,781.178	313	66	18.70
310	1,850.711	314	67	18.44
315	1,603.436	315	67	18.18
320	1,731.248	317	68	17.66
325	1,757.728	319	69	17.14
330	1,721.443	322	69	16.36
335	1,646.570	333	69	13.50
340	1,763.896	328	70	14.80
345	1,789.137	334	70	13.24
350	1,797.697	333	71	13.50
355	1,580.112	336	71	12.72
360	1,798.867	345	72	10.38
365	1,717.325	344	72	10.64
370	1,779.528	343	73	10.90
375	1,610.741	349	73	9.35
380	1,715.068	351	74	8.83
385	1,731.640	360	74	6.49
390	1,842.495	357	75	7.27
395	1,948.158	360	75	6.49
400	1,788.559	360	76	6.49
405	1,690.136	360	76	6.49
410	1,729.078	369	76	4.15
415	1,582.802	370	77	3.89
420	1,974.284	369	77	4.15
425	1,955.599	372	78	3.37
430	1,869.270	375	78	2.59
435	1,985.458	377	79	2.07
440	1,988.543	374	79	2.85
445	1,928.372	375	79	2.59
450	1,840.931	372	79	3.37
455	2,053.819	373	80	3.11

Continued on next page

Table C-17 Continued from previous page

Threshold	Duration	Alerts	Size (%)	ALR (%)
460	2,061.651	378	80	1.81
465	1,908.106	375	80	2.59
470	1,895.828	375	81	2.59
475	1,931.174	376	81	2.33
480	1,893.223	377	81	2.07
485	2,026.720	378	82	1.81
490	1,919.564	375	82	2.59
495	2,044.002	378	82	1.81
500	2,022.187	371	82	3.63
505	1,874.726	380	83	1.29
510	1,945.556	375	83	2.59
515	1,930.186	375	83	2.59
520	2,042.655	375	84	2.59
525	1,983.472	374	84	2.85
530	1,885.467	379	84	1.55
535	1,628.996	375	84	2.59
540	2,111.850	378	85	1.81
545	2,046.760	373	85	3.11
550	1,906.988	376	85	2.33
555	1,939.615	378	86	1.81
560	2,053.438	376	86	2.33
565	1,977.419	376	86	2.33
570	1,921.981	374	86	2.85
575	1,994.493	377	87	2.07
580	1,948.223	379	87	1.55
585	1,951.010	376	87	2.33
590	1,961.483	376	88	2.33
595	2,055.956	379	88	1.55
600	1,951.046	373	88	3.11
605	2,079.334	380	88	1.29
610	1,965.641	377	89	2.07

Continued on next page

Table C-17 Continued from previous page

Threshold	Duration	Alerts	Size (%)	ALR (%)
615	2,091.758	377	89	2.07
620	1,999.960	377	89	2.07
625	2,058.221	383	90	0.51
630	2,127.676	377	90	2.07
635	2,046.671	380	90	1.29
640	2,016.774	375	90	2.59
645	1,986.561	378	91	1.81
650	1,972.470	374	91	2.85
655	2,027.771	380	91	1.29
660	2,033.070	377	91	2.07
665	1,983.771	372	92	3.37
670	2,124.100	375	92	2.59
675	2,023.679	380	92	1.29
680	2,066.898	377	93	2.07
685	2,008.875	377	93	2.07
690	2,075.087	372	93	3.37
695	2,051.922	374	93	2.85
700	2,066.322	377	94	2.07
705	1,980.602	385	94	0.00
710	2,085.297	382	94	0.77
715	2,023.614	385	94	0.00
720	2,060.294	375	95	2.59
725	1,511.191	385	95	0.00
730	1,538.946	382	95	0.77
735	1,498.467	375	95	2.59
740	1,577.820	376	96	2.33
745	1,589.241	385	96	0.00
750	1,580.726	385	96	0.00
755	1,525.512	385	96	0.00
760	1,598.957	385	96	0.00
765	1,579.000	382	96	0.77

Continued on next page

Table C-17 Continued from previous page

Threshold	Duration	Alerts	Size (%)	ALR (%)
770	1,603.538	382	96	0.77
775	1,527.277	382	96	0.77
780	1,590.635	385	96	0.00
785	1,593.138	385	96	0.00
790	1,614.175	382	97	0.77
795	1,596.588	385	97	0.00
800	1,547.725	385	97	0.00

Concluded

Table C-18: Results using the LINTS capped-by-packets strategy to compress the CIC 2017 testing data set and the RegJul2018 rule

Threshold	Duration	Alerts	Size (%)	ALR (%)
5	759.931	1,051	11	17.95
10	734.435	1,098	15	14.28
15	804.996	1,104	17	13.81
20	577.189	1,126	18	12.09
25	612.385	1,184	19	7.57
30	639.180	1,194	20	6.79
35	897.722	1,214	21	5.23
40	881.574	1,267	22	1.09
45	648.050	1,269	22	0.93
50	864.092	1,272	23	0.70
55	871.759	1,272	23	0.70
60	887.676	1,274	24	0.54
65	911.727	1,276	24	0.39
70	882.712	1,279	25	0.15
75	879.190	1,281	25	0.00
80	881.849	1,281	25	0.00
85	880.534	1,281	26	0.00
90	879.658	1,281	26	0.00
95	784.310	1,281	26	0.00
100	691.019	1,281	26	0.00
			Concluded	

C.2.3 Capped by Bytes

Table C-19: Results using the LINTS capped-by-bytes strategy to compress the CDX 2009 usama-gator010 data set and the Circa 2009 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
500	20.733	62	47	87.79
1000	24.183	88	53	82.67
1500	26.692	192	58	62.20
2000	28.657	407	64	19.88
2500	28.419	417	65	17.91
3000	28.941	465	66	8.46
3500	29.028	491	70	3.34
4000	29.961	504	71	0.78
4500	29.857	504	73	0.78
5000	30.141	504	73	0.78
5500	29.618	504	73	0.78
6000	30.204	504	75	0.78
6500	29.825	504	76	0.78
7000	30.235	504	76	0.78
7500	30.235	504	76	0.78
8000	30.572	504	77	0.78
8500	29.985	504	77	0.78
9000	30.579	504	77	0.78
9500	30.313	504	77	0.78
10000	30.725	504	77	0.78
			Concluded	

Table C-20: Results using the LINTS capped-by-bytes strategy to compress the CDX 2009 usama-gator020 data set and the Circa 2009 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
500	461.893	9,330	19	16.97
1000	466.506	9,418	20	16.18
1500	478.026	9,612	20	14.46
2000	503.012	9,694	21	13.73
2500	511.715	9,715	22	13.54
3000	507.297	11,010	22	2.02
3500	492.256	11,015	22	1.97
4000	537.892	11,018	22	1.94
4500	510.662	11,026	23	1.87
5000	450.559	11,025	23	1.88
5500	324.306	11,158	23	0.70
6000	290.156	11,163	23	0.65
6500	331.979	11,169	23	0.60
7000	341.289	11,171	23	0.58
7500	339.591	11,190	23	0.41
8000	355.919	11,192	24	0.40
8500	356.302	11,196	24	0.36
9000	320.489	11,204	24	0.29
9500	260.715	11,205	24	0.28
10000	337.699	11,207	24	0.26
			Concluded	

Table C-21: Results using the LINTS capped-by-bytes strategy to compress the MAC-CDC 2012 data set and the RegAug2013 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
500	14,560.738	2,119	16	99.01
1000	14,532.975	6,479	19	96.99
1500	14,227.888	10,822	20	94.97
2000	14,811.920	16,090	23	92.52
2500	15,035.873	20,485	23	90.48
3000	15,235.632	24,539	23	88.60
3500	15,332.346	27,560	25	87.20
4000	15,060.131	31,562	25	85.34
4500	15,244.125	35,719	26	83.41
5000	14,927.273	39,754	27	81.53
5500	14,977.979	43,536	27	79.78
6000	14,860.720	46,594	28	78.36
6500	14,492.387	50,125	28	76.72
7000	15,262.400	54,134	29	74.86
7500	15,044.431	58,183	30	72.98
8000	14,543.036	62,022	30	71.19
8500	15,293.260	65,379	30	69.64
9000	14,606.947	68,722	31	68.08
9500	14,965.133	72,521	31	66.32
10000	14,767.614	76,499	31	64.47
			Concluded	

Table C-22: Results using the LINTS capped-by-bytes strategy to compress the ISCX 2012 testing data set and the RegAug2013 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
500	69.593	0	1	100.00
1000	70.910	3	2	97.65
1500	68.304	12	3	90.62
2000	74.650	52	5	59.37
2500	81.979	53	5	58.59
3000	82.320	57	5	55.46
3500	86.278	74	7	42.18
4000	89.188	80	7	37.50
4500	90.011	80	7	37.50
5000	99.822	83	8	35.15
5500	104.490	83	8	35.15
6000	105.523	83	9	35.15
6500	108.908	87	10	32.03
7000	114.240	87	10	32.03
7500	114.798	87	10	32.03
8000	122.135	88	11	31.25
8500	127.800	88	11	31.25
9000	127.159	89	11	30.46
9500	129.636	90	12	29.68
10000	132.353	90	12	29.68
			Concluded	

Table C-23: Results using the LINTS capped-by-bytes strategy to compress the UNSW-NB 2015 January data set and the RegSep2018 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
500	1,404.570	26	1	93.24
1000	1,159.506	66	2	82.85
1500	1,237.389	82	3	78.70
2000	1,112.152	91	5	76.36
2500	924.848	110	6	71.42
3000	895.927	116	7	69.87
3500	1,020.872	112	8	70.90
4000	990.087	130	8	66.23
4500	1,726.610	133	9	65.45
5000	1,716.041	135	10	64.93
5500	1,691.846	138	11	64.15
6000	1,706.485	137	11	64.41
6500	953.881	136	12	64.67
7000	1,627.096	138	12	64.15
7500	979.703	138	13	64.15
8000	1,244.393	144	14	62.59
8500	1,487.596	147	14	61.81
9000	1,535.072	147	15	61.81
9500	1,557.791	148	16	61.55
10000	1,565.311	150	16	61.03
			Concluded	

Table C-24: Results using the LINTS capped-by-bytes strategy to compress the CIC 2017 testing data set and the RegJul2018 rule

Threshold	Duration	Alerts	Size (%)	ALR (%)
500	698.434	4	1	99.68
1000	656.196	1,002	2	21.77
1500	742.534	1,026	3	19.90
2000	463.707	1,064	4	16.93
2500	493.340	1,098	4	14.28
3000	507.081	1,098	5	14.28
3500	558.363	1,100	6	14.12
4000	795.976	1,100	6	14.12
4500	850.519	1,115	7	12.95
5000	597.517	1,121	8	12.49
5500	726.279	1,161	8	9.36
6000	827.495	1,174	8	8.35
6500	808.563	1,193	9	6.86
7000	824.731	1,200	9	6.32
7500	815.777	1,205	10	5.93
8000	839.822	1,209	11	5.62
8500	835.358	1,224	11	4.44
9000	817.328	1,225	12	4.37
9500	862.297	1,225	13	4.37
10000	851.618	1,225	13	4.37
			Concluded	

C.2.4 Tainted Flow Capped by Packets

Table C-25: Results using the LINTS tainted flow capped-by-packets strategy to compress the CDX 2009 usama-gator010 data set and the Circa 2009 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
5	38.076	509	82	−0.19
10	40.632	509	94	−0.19
15	42.566	509	96	−0.19
20	49.110	510	96	−0.39
25	48.389	512	96	−0.78
30	48.719	512	96	−0.78
35	49.220	509	96	−0.19
40	50.287	509	97	−0.19
45	48.618	509	97	−0.19
50	48.940	509	97	−0.19
55	48.648	510	97	−0.39
60	48.476	510	97	−0.39
65	48.486	508	97	0.00
70	48.827	509	97	−0.19
75	48.490	508	97	0.00
80	49.126	509	97	−0.19
85	48.157	509	97	−0.19
90	48.505	508	97	0.00
95	48.997	509	97	−0.19
100	48.399	509	97	−0.19
			Concluded	

Table C-26: Results using the LINTS tainted flow capped-by-packets strategy to compress the CDX 2009 usama-gator020 data set and the Circa 2009 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
5	342.757	11,003	95	2.08
		Continued on next page		

Table C-26 Continued from previous page

Threshold	Duration	Alerts	Size (%)	ALR (%)
10	368.030	11,192	97	0.40
15	377.670	11,196	97	0.36
20	385.944	11,197	98	0.35
25	387.162	11,200	98	0.32
30	389.328	11,208	98	0.25
35	395.027	11,222	98	0.13
40	387.703	11,230	98	0.06
45	396.094	11,234	98	0.02
50	398.770	11,234	98	0.02
55	393.995	11,234	98	0.02
60	394.044	11,234	98	0.02
65	397.212	11,234	98	0.02
70	397.269	11,234	98	0.02
75	396.611	11,234	98	0.02
80	396.103	11,234	98	0.02
85	400.894	11,234	98	0.02
90	396.343	11,234	98	0.02
95	400.668	11,237	98	0.00
100	402.818	11,237	99	0.00

Concluded

Table C-27: Results using the LINTS tainted flow capped-by-packets strategy to compress the MACCDC 2012 data set and the RegAug2013 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
5	459.480	214,428	58	0.42
10	452.623	214,543	63	0.37
15	457.666	214,721	65	0.29
20	472.720	214,838	66	0.23
25	470.676	214,977	67	0.17
30	488.360	215,093	68	0.11
35	480.403	215,185	69	0.07
40	491.971	215,237	69	0.05
45	501.596	215,299	70	0.02
50	501.127	215,309	71	0.01
55	534.831	215,336	71	0.00
60	514.007	215,353	71	0.00
65	517.393	215,350	72	0.00
70	517.786	215,352	72	0.00
75	519.196	215,352	73	0.00
80	489.078	215,359	73	0.00
85	528.719	215,351	73	0.00
90	521.860	215,335	73	0.00
95	551.039	215,342	74	0.00
100	544.449	215,342	74	0.00
			Concluded	

Table C-28: Results using the LINTS tainted flow capped-by-packets strategy to compress the ISCX 2012 testing data set and the RegAug2013 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
5	432.700	116	25	9.37
10	435.523	118	31	7.81
15	431.529	119	35	7.03
20	436.214	126	39	1.56
25	446.120	126	42	1.56
30	451.082	126	44	1.56
35	460.398	126	47	1.56
40	465.139	126	49	1.56
45	473.820	126	50	1.56
50	484.829	126	52	1.56
55	465.480	126	53	1.56
60	489.047	126	54	1.56
65	487.176	126	56	1.56
70	506.235	126	57	1.56
75	510.906	126	58	1.56
80	505.995	126	59	1.56
85	509.939	126	60	1.56
90	529.104	126	60	1.56
95	528.266	126	61	1.56
100	535.404	126	62	1.56
			Concluded	

Table C-29: Results using the LINTS tainted flow capped-by-packets strategy to compress the UNSW-NB 2015 January data set and the RegSep2018 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
5	2,247.710	104	2	72.98
10	2,227.973	144	5	62.59
15	1,921.553	157	7	59.22
20	1,626.515	155	9	59.74
25	1,577.055	166	12	56.88
30	1,583.682	173	17	55.06
35	1,552.924	177	22	54.02
40	1,558.337	173	26	55.06
45	1,529.835	180	30	53.24
50	1,508.895	182	32	52.72
55	1,772.992	187	34	51.42
60	1,785.159	186	36	51.68
65	1,776.470	190	38	50.64
70	1,804.159	195	39	49.35
75	1,756.914	196	40	49.09
80	1,766.372	200	41	48.05
85	1,763.331	201	42	47.79
90	1,789.593	201	43	47.79
95	1,809.527	209	43	45.71
100	1,801.578	208	44	45.97
105	1,952.424	211	45	45.19
110	1,817.742	210	45	45.45
115	1,810.539	215	46	44.15
120	1,771.247	220	46	42.85
125	1,768.032	221	47	42.59
130	1,879.062	223	48	42.07
135	1,755.731	226	48	41.29
140	1,716.833	233	49	39.48
145	1,861.980	237	49	38.44

Continued on next page

Table C-29 Continued from previous page

Threshold	Duration	Alerts	Size (%)	ALR (%)
150	1,703.215	233	50	39.48
155	1,670.300	236	51	38.70
160	1,603.225	238	51	38.18
165	1,598.836	253	52	34.28
170	1,584.453	246	52	36.10
175	1,587.071	246	53	36.10
180	1,590.781	247	53	35.84
185	1,597.793	254	54	34.02
190	1,845.544	253	55	34.28
195	1,837.121	252	55	34.54
200	1,843.102	257	55	33.24
205	2,488.764	254	56	34.02
210	2,538.284	258	57	32.98
215	2,447.756	273	57	29.09
220	2,544.399	263	58	31.68
225	2,461.332	266	58	30.90
230	2,565.719	273	59	29.09
235	2,291.477	274	59	28.83
240	2,644.625	283	60	26.49
245	2,464.431	277	60	28.05
250	2,486.451	276	61	28.31
255	2,507.496	282	61	26.75
260	2,414.061	289	62	24.93
265	2,370.185	293	62	23.89
270	2,198.984	290	63	24.67
275	2,553.860	296	63	23.11
280	2,278.718	294	64	23.63
285	2,189.928	298	64	22.59
290	2,377.365	305	65	20.77
295	2,359.793	307	66	20.25
300	2,524.132	306	66	20.51

Continued on next page

Table C-29 Continued from previous page

Threshold	Duration	Alerts	Size (%)	ALR (%)
305	2,345.426	313	66	18.70
310	2,186.672	314	67	18.44
315	2,449.067	315	67	18.18
320	2,444.782	317	68	17.66
325	2,151.444	319	69	17.14
330	2,461.047	322	69	16.36
335	2,408.650	330	69	14.28
340	2,330.966	331	70	14.02
345	2,331.646	334	70	13.24
350	1,953.204	336	71	12.72
355	1,845.028	339	71	11.94
360	1,944.162	348	72	9.61
365	1,532.773	344	72	10.64
370	1,533.233	343	73	10.90
375	1,523.741	352	73	8.57
380	1,536.110	348	74	9.61
385	1,535.922	357	74	7.27
390	1,528.552	357	75	7.27
395	1,528.195	357	75	7.27
400	1,529.403	360	76	6.49
405	2,503.278	363	76	5.71
410	1,682.599	369	77	4.15
415	1,545.262	367	77	4.67
420	1,547.974	372	77	3.37
425	1,546.102	372	78	3.37
430	1,867.256	372	78	3.37
435	1,541.348	377	79	2.07
440	1,540.468	374	79	2.85
445	1,538.196	378	79	1.81
450	1,539.209	376	80	2.33
455	1,537.389	377	80	2.07

Continued on next page

Table C-29 Continued from previous page

Threshold	Duration	Alerts	Size (%)	ALR (%)
460	1,545.339	375	80	2.59
465	1,531.582	375	80	2.59
470	1,543.494	375	81	2.59
475	1,553.849	377	81	2.07
480	1,554.793	377	81	2.07
485	1,544.969	375	82	2.59
490	1,534.223	375	82	2.59
495	1,548.452	375	82	2.59
500	1,602.182	371	82	3.63
505	1,562.435	380	83	1.29
510	1,573.342	378	83	1.81
515	1,564.061	375	83	2.59
520	1,561.184	378	84	1.81
525	1,567.294	371	84	3.63
530	1,562.138	379	84	1.55
535	1,544.015	378	84	1.81
540	1,581.897	375	85	2.59
545	1,571.057	371	85	3.63
550	1,952.278	376	85	2.33
555	2,250.584	378	86	1.81
560	2,129.566	376	86	2.33
565	2,381.389	379	86	1.55
570	2,442.378	374	86	2.85
575	1,934.926	374	87	2.85
580	1,885.309	376	87	2.33
585	2,051.666	376	87	2.33
590	2,283.906	379	88	1.55
595	1,911.606	371	88	3.63
600	1,700.939	376	88	2.33
605	1,573.906	377	88	2.07
610	1,581.943	377	89	2.07

Continued on next page

Table C-29 Continued from previous page

Threshold	Duration	Alerts	Size (%)	ALR (%)
615	1,596.665	377	89	2.07
620	1,561.467	380	89	1.29
625	1,589.657	383	90	0.51
630	1,563.552	377	90	2.07
635	1,567.517	377	90	2.07
640	1,611.732	376	90	2.33
645	1,568.810	375	91	2.59
650	1,595.057	374	91	2.85
655	1,899.480	380	91	1.29
660	1,656.027	377	91	2.07
665	1,908.284	375	92	2.59
670	1,745.258	375	92	2.59
675	1,612.091	380	92	1.29
680	1,590.288	377	93	2.07
685	1,582.369	380	93	1.29
690	1,587.762	372	93	3.37
695	1,574.717	374	93	2.85
700	1,602.442	377	94	2.07
705	1,579.921	385	94	0.00
710	1,606.266	385	94	0.00
715	1,591.327	382	94	0.77
720	1,617.370	374	95	2.85
725	1,574.609	385	95	0.00
730	1,603.860	385	95	0.00
735	1,580.334	375	95	2.59

Concluded

Table C-30: Results using the LINTS tainted flow capped-by-packets strategy to compress the CICCIC 2017 testing data set and the RegJul2018 rule

Threshold	Duration	Alerts	Size (%)	ALR (%)
5	1,378.812	1,051	77	17.95
10	1,406.062	1,098	81	14.28
15	1,435.281	1,104	83	13.81
20	1,383.563	1,126	84	12.09
25	1,323.924	1,184	85	7.57
30	1,220.955	1,194	86	6.79
35	1,231.789	1,214	87	5.23
40	1,424.342	1,267	88	1.09
45	1,457.301	1,269	88	0.93
50	1,409.821	1,272	89	0.70
55	1,314.869	1,272	89	0.70
60	1,460.379	1,274	90	0.54
65	1,398.631	1,276	90	0.39
70	1,440.951	1,279	90	0.15
75	1,341.009	1,281	91	0.00
80	1,446.666	1,281	91	0.00
85	1,352.205	1,281	91	0.00
90	1,318.523	1,281	91	0.00
95	1,321.930	1,281	92	0.00
100	1,392.349	1,281	92	0.00
			Concluded	

C.2.5 Tainted Flow Capped by Bytes

Table C-31: Results using the LINTS tainted flow capped-by-bytes strategy to compress the CDX 2009 usama-gator010 data set and the Circa 2009 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
200	32.822	447	65	12.00
400	35.721	456	69	10.23
600	40.954	459	71	9.64
800	43.353	459	75	9.64
1000	43.085	458	76	9.84
1200	44.112	461	77	9.25
1400	45.636	480	81	5.51
1600	45.187	479	82	5.70
1800	45.925	480	85	5.51
2000	46.613	480	87	5.51
2200	46.855	480	87	5.51
2400	46.890	483	87	4.92
2600	46.438	508	87	0.00
2800	46.916	509	88	−0.19
3000	47.204	508	88	0.00
3200	46.973	509	90	−0.19
3400	47.729	509	91	−0.19
3600	47.657	508	92	0.00
3800	48.414	508	92	0.00
4000	47.487	509	92	−0.19
			Concluded	

Table C-32: Results using the LINTS tainted flow capped-by-bytes strategy to compress the CDX 2009 usama-gator020 data set and the Circa 2009 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
200	312.275	10,620	93	5.49
		Continued on next page		

Table C-32 Continued from previous page

Threshold	Duration	Alerts	Size (%)	ALR (%)
400	307.716	10,695	93	4.82
600	312.521	10,769	93	4.16
800	325.877	10,804	94	3.85
1000	331.514	10,816	94	3.74
1200	337.418	10,818	94	3.72
1400	338.607	10,956	95	2.50
1600	338.370	10,974	95	2.34
1800	349.907	10,974	95	2.34
2000	349.778	10,974	96	2.34
2200	349.174	10,974	96	2.34
2400	349.846	10,983	96	2.26
2600	352.792	11,027	96	1.86
2800	350.595	11,027	96	1.86
3000	351.938	11,029	96	1.85
3200	354.544	11,029	96	1.85
3400	351.773	11,034	96	1.80
3600	357.149	11,035	96	1.79
3800	357.763	11,035	96	1.79
4000	358.153	11,037	96	1.77

Concluded

Table C-33: Results using the LINTS tainted flow capped-by-bytes strategy to compress the MACCDC 2012 data set and the RegAug2013 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
200	442.765	213,375	51	0.91
400	436.299	213,448	51	0.88
600	440.560	214,017	52	0.61
800	423.495	214,155	53	0.55
1000	424.800	214,206	54	0.53
1200	424.305	214,257	54	0.50
1400	434.143	214,270	55	0.50
1600	438.413	214,526	56	0.38
1800	445.555	214,554	57	0.36
2000	469.592	214,549	58	0.37
2200	465.565	214,558	58	0.36
2400	462.145	214,558	58	0.36
2600	453.142	214,580	58	0.35
2800	435.039	214,632	58	0.33
3000	459.988	214,675	58	0.31
3200	449.425	214,716	60	0.29
3400	452.856	214,774	60	0.26
3600	448.072	214,811	60	0.24
3800	456.188	214,864	60	0.22
4000	452.927	214,925	60	0.19
			Concluded	

Table C-34: Results using the LINTS tainted flow capped-by-bytes strategy to compress the ISCX 2012 testing data set and the RegAug2013 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
200	439.898	102	19	20.31
400	446.585	103	20	19.53
600	425.769	103	20	19.53
800	417.639	105	21	17.96
1000	408.065	112	21	12.50
1200	399.411	112	21	12.50
1400	383.751	112	21	12.50
1600	377.300	111	22	13.28
1800	375.233	117	23	8.59
2000	376.164	118	23	7.81
2200	379.617	118	23	7.81
2400	500.147	118	24	7.81
2600	380.628	120	24	6.25
2800	372.015	125	24	2.34
3000	387.044	125	24	2.34
3200	396.771	126	25	1.56
3400	382.782	126	25	1.56
3600	381.750	126	25	1.56
3800	378.412	126	25	1.56
4000	372.932	126	25	1.56
			Concluded	

Table C-35: Results using the LINTS tainted flow capped-by-bytes strategy to compress the UNSW-NB 2015 January data set and the RegSep2018 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
200	1,778.111	4	0	98.96
400	1,765.164	15	1	96.10
600	1,766.285	66	1	82.85
800	1,757.391	87	2	77.40
1000	1,766.500	95	2	75.32
1200	1,753.335	104	3	72.98
1400	1,744.156	110	3	71.42
1600	1,599.889	113	4	70.64
1800	1,523.936	120	5	68.83
2000	1,450.644	120	5	68.83
2200	1,458.102	120	5	68.83
2400	1,458.196	122	6	68.31
2600	1,466.026	129	6	66.49
2800	1,641.490	129	6	66.49
3000	1,600.774	136	7	64.67
3200	1,589.220	133	7	65.45
3400	1,977.670	134	8	65.19
3600	2,316.592	135	8	64.93
3800	2,502.570	138	8	64.15
4000	2,437.329	141	9	63.37

Concluded

Table C-36: Results using the LINTS tainted flow capped-by-bytes strategy to compress the CIC 2017 testing data set and the RegJul2018 rule

Threshold	Duration	Alerts	Size (%)	ALR (%)
200	1,352.412	63	67	95.08
400	1,282.959	64	67	95.00

Continued on next page

Table C-36 Continued from previous page

Threshold	Duration	Alerts	Size (%)	ALR (%)
600	1,291.519	349	68	72.75
800	1,187.805	854	68	33.33
1000	1,123.331	1,057	68	17.48
1200	1,296.911	1,062	68	17.09
1400	1,334.330	1,062	69	17.09
1600	1,359.556	1,062	69	17.09
1800	1,322.793	1,064	70	16.93
2000	1,374.772	1,064	70	16.93
2200	1,343.672	1,081	70	15.61
2400	1,324.778	1,098	70	14.28
2600	1,413.200	1,098	71	14.28
2800	1,416.683	1,098	71	14.28
3000	1,403.979	1,098	71	14.28
3200	1,400.648	1,098	72	14.28
3400	1,271.900	1,098	72	14.28
3600	1,389.497	1,100	72	14.12
3800	1,358.279	1,100	72	14.12
4000	1,322.544	1,100	72	14.12

Concluded

C.2.6 Tainted Flow Capped by Packets with Entropy

Table C-37: Results using the LINTS tainted flow capped by packets with entropy strategy to compress the CDX 2009 usama-gator010 data set and the Circa 2009 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
5	44.242	507	64	0.19
10	45.755	509	70	−0.19
15	47.815	509	72	−0.19
20	47.474	509	72	−0.19
25	48.482	512	72	−0.78
30	47.604	512	72	−0.78
35	47.675	510	73	−0.39
40	48.076	509	73	−0.19
45	47.648	509	73	−0.19
50	48.229	509	73	−0.19
55	48.075	509	73	−0.19
60	47.963	510	73	−0.39
65	48.511	509	73	−0.19
70	48.282	509	73	−0.19
75	47.941	509	73	−0.19
80	48.932	509	73	−0.19
85	49.039	509	73	−0.19
90	44.188	509	73	−0.19
95	39.587	509	73	−0.19
100	39.806	508	73	0.00
			Concluded	

Table C-38: Results using the LINTS tainted flow capped by packets with strategy to compress the CDX 2009 usama-gator020 data set and the Circa 2009 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
5	395.917	11,002	21	2.09
10	431.290	11,191	22	0.40
15	429.841	11,194	22	0.38
20	424.418	11,195	23	0.37
25	410.136	11,198	23	0.34
30	442.380	11,206	23	0.27
35	441.695	11,220	23	0.15
40	472.419	11,229	23	0.07
45	430.322	11,233	23	0.03
50	429.382	11,233	23	0.03
55	433.050	11,233	23	0.03
60	434.526	11,233	23	0.03
65	431.170	11,233	23	0.03
70	445.329	11,233	23	0.03
75	439.195	11,233	24	0.03
80	443.880	11,233	24	0.03
85	437.851	11,233	24	0.03
90	408.352	11,236	24	0.00
95	399.106	11,236	24	0.00
100	369.172	11,236	24	0.00
			Concluded	

Table C-39: Results using the LINTS tainted flow capped by packets with entropy strategy to compress the MACCDC 2012 data set and the RegAug2013 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
5	488.321	214,394	28	0.44
10	494.186	214,534	33	0.37
15	510.281	214,694	35	0.30
20	512.306	214,819	37	0.24
25	546.857	214,964	38	0.17
30	541.144	215,088	39	0.12
35	534.531	215,171	39	0.08
40	532.828	215,224	40	0.05
45	529.783	215,281	41	0.03
50	541.451	215,304	41	0.02
55	547.651	215,326	42	0.01
60	547.219	215,342	42	0.00
65	540.199	215,347	42	0.00
70	544.639	215,353	43	0.00
75	539.338	215,346	43	0.00
80	558.309	215,348	43	0.00
85	559.835	215,348	44	0.00
90	560.133	215,343	44	0.00
95	557.259	215,348	44	0.00
100	582.792	215,345	44	0.00
			Concluded	

Table C-40: Results using the LINTS tainted flow capped by packets with entropy strategy to compress the ISCX 2012 testing data set and the RegAug2013 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
5	421.388	115	6	10.15
10	438.582	117	10	8.59
15	415.894	118	14	7.81
20	412.920	125	18	2.34
25	418.154	126	20	1.56
30	417.099	126	22	1.56
35	412.039	127	24	0.78
40	405.320	129	26	−0.78
45	406.122	129	27	−0.78
50	412.008	130	29	−1.56
55	417.943	131	30	−2.34
60	423.111	130	31	−1.56
65	424.463	130	32	−1.56
70	431.645	131	33	−2.34
75	444.107	133	34	−3.90
80	451.505	134	35	−4.68
85	435.056	134	35	−4.68
90	448.812	137	36	−7.03
95	448.966	137	36	−7.03
100	449.818	137	37	−7.03
			Concluded	

Table C-41: Results using the LINTS tainted flow capped by packets with entropy strategy to compress the UNSW-NB 2015 January data set and the RegSep2018 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
5	1,520.721	104	1	72.98
10	1,433.394	143	4	62.85
15	1,515.807	155	6	59.74
20	1,557.135	153	7	60.25
25	1,446.741	164	8	57.40
30	1,375.109	170	10	55.84
35	1,092.540	176	11	54.28
40	1,121.198	176	12	54.28
45	1,100.093	179	13	53.50
50	1,122.642	179	13	53.50
55	1,088.509	184	14	52.20
60	1,136.444	189	14	50.90
65	1,107.095	191	14	50.38
70	1,112.404	190	14	50.64
75	1,102.005	193	14	49.87
80	1,094.167	197	15	48.83
85	1,149.586	203	15	47.27
90	1,136.941	204	15	47.01
95	1,130.697	206	15	46.49
100	1,147.525	205	15	46.75
			Concluded	

Table C-42: Results using the LINTS tainted flow capped by packets with entropy strategy to compress the CIC 2017 testing data set and the RegJul2018 rule

Threshold	Duration	Alerts	Size (%)	ALR (%)
5	705.441	1,051	11	17.95
10	697.768	1,098	13	14.28
15	685.079	1,104	14	13.81
20	708.630	1,126	14	12.09
25	729.203	1,184	14	7.57
30	702.013	1,194	15	6.79
35	706.559	1,214	15	5.23
40	704.797	1,267	15	1.09
45	727.802	1,269	15	0.93
50	643.738	1,272	15	0.70
55	730.056	1,272	15	0.70
60	703.876	1,274	15	0.54
65	744.823	1,276	15	0.39
70	681.312	1,279	15	0.15
75	746.393	1,281	15	0.00
80	686.369	1,281	15	0.00
85	728.953	1,281	15	0.00
90	654.376	1,281	15	0.00
95	729.268	1,281	15	0.00
100	682.401	1,281	15	0.00
			Concluded	

C.2.7 Tainted Flow Capped by Bytes with Entropy

Table C-43: Results using the LINTS tainted flow capped by bytes with entropy strategy to compress the CDX 2009 usama-gator010 data set and the Circa 2009 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
200	31.141	447	47	12.00
400	32.737	456	51	10.23
600	33.325	459	53	9.64
800	35.143	458	56	9.84
1000	35.235	459	58	9.64
1200	35.579	460	59	9.44
1400	36.876	480	63	5.51
1600	36.880	480	63	5.51
1800	37.682	480	66	5.51
2000	38.102	479	68	5.70
2200	38.239	480	69	5.51
2400	38.504	483	69	4.92
2600	39.095	507	70	0.19
2800	38.998	509	70	−0.19
3000	39.852	509	71	−0.19
3200	39.183	509	71	−0.19
3400	39.624	507	71	0.19
3600	39.286	509	71	−0.19
3800	39.125	509	71	−0.19
4000	39.285	509	71	−0.19

Concluded

Table C-44: Results using the LINTS tainted flow capped by bytes with entropy strategy to compress the CDX 2009 usama-gator020 data set and the Circa 2009 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
200	403.044	10,616	18	5.52

Continued on next page

Table C-44 Continued from previous page

Threshold	Duration	Alerts	Size (%)	ALR (%)
400	405.685	10,692	19	4.85
600	413.748	10,766	19	4.19
800	409.441	10,803	20	3.86
1000	384.265	10,815	20	3.75
1200	406.122	10,817	20	3.73
1400	409.183	10,954	20	2.51
1600	400.269	10,972	21	2.35
1800	398.555	10,972	21	2.35
2000	402.882	10,972	21	2.35
2200	403.702	10,972	21	2.35
2400	406.237	10,982	21	2.26
2600	413.939	11,026	22	1.87
2800	407.800	11,026	22	1.87
3000	412.000	11,028	22	1.85
3200	414.988	11,028	22	1.85
3400	416.254	11,033	22	1.81
3600	416.157	11,034	22	1.80
3800	399.902	11,034	22	1.80
4000	378.856	11,036	22	1.78

Concluded

Table C-45: Results using the LINTS tainted flow capped by bytes with entropy strategy to compress the MACCDC 2012 data set and the RegAug2013 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
200	487.965	213,386	21	0.91
400	492.121	213,438	22	0.88
600	486.253	214,002	23	0.62
800	488.165	214,143	24	0.56
1000	488.145	214,188	24	0.53
1200	486.279	214,244	25	0.51
1400	482.017	214,269	25	0.50
1600	490.650	214,509	26	0.39
1800	482.404	214,516	28	0.38
2000	486.082	214,538	28	0.37
2200	494.368	214,536	28	0.37
2400	489.694	214,546	28	0.37
2600	484.069	214,559	28	0.36
2800	490.741	214,606	29	0.34
3000	490.415	214,661	29	0.31
3200	486.666	214,711	30	0.29
3400	499.200	214,759	30	0.27
3600	489.902	214,800	30	0.25
3800	491.136	214,859	30	0.22
4000	482.348	214,899	31	0.20
			Concluded	

Table C-46: Results using the LINTS tainted flow capped by bytes with entropy strategy to compress the ISCX 2012 testing data set and the RegAug2013 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
200	399.062	101	1	21.09
400	375.067	102	1	20.31
600	369.787	102	2	20.31
800	360.718	104	2	18.75
1000	330.633	111	2	13.28
1200	323.451	111	3	13.28
1400	322.507	111	3	13.28
1600	323.231	110	3	14.06
1800	324.778	116	4	9.37
2000	324.446	117	5	8.59
2200	323.955	117	5	8.59
2400	324.907	117	5	8.59
2600	322.440	119	5	7.03
2800	324.192	124	5	3.12
3000	322.455	124	5	3.12
3200	322.464	125	6	2.34
3400	325.669	125	6	2.34
3600	323.971	125	6	2.34
3800	325.659	125	6	2.34
4000	324.876	125	7	2.34
			Concluded	

Table C-47: Results using the LINTS tainted flow capped by bytes with entropy strategy to compress the UNSW-NB 2015 January data set and the RegSep2018 rule set

Threshold	Duration	Alerts	Size (%)	ALR (%)
200	1,552.471	4	0	98.96
400	1,392.976	15	1	96.10
600	1,551.300	66	1	82.85
800	1,497.163	87	2	77.40
1000	1,471.885	95	2	75.32
1200	1,377.935	104	3	72.98
1400	1,129.808	110	3	71.42
1600	1,095.043	113	4	70.64
1800	1,117.488	120	4	68.83
2000	1,107.694	120	5	68.83
2200	1,116.288	120	5	68.83
2400	1,109.725	123	6	68.05
2600	1,100.099	129	6	66.49
2800	1,118.925	135	6	64.93
3000	1,129.523	133	7	65.45
3200	1,122.058	131	7	65.97
3400	1,126.451	131	7	65.97
3600	1,120.569	134	8	65.19
3800	1,126.394	138	8	64.15
4000	1,106.823	138	8	64.15
4200	748.221	140	8	63.63
4400	746.321	141	8	63.37
4600	725.957	140	8	63.63
4800	587.197	146	9	62.07
5000	569.124	145	9	62.33
5200	574.771	142	9	63.11
5400	574.852	143	9	62.85
5600	571.971	144	9	62.59
5800	570.460	146	9	62.07

Continued on next page

Table C-47 Continued from previous page

Threshold	Duration	Alerts	Size (%)	ALR (%)
6000	569.749	143	10	62.85
6200	567.244	145	10	62.33
6400	572.849	145	10	62.33
6600	609.186	145	10	62.33
6800	570.850	145	10	62.33
7000	570.929	143	10	62.85
7200	568.242	143	10	62.85
7400	569.207	145	10	62.33
7600	571.620	145	10	62.33
7800	566.139	145	11	62.33
8000	569.985	149	11	61.29
8200	878.191	153	11	60.25
8400	810.570	149	11	61.29
8600	869.030	152	11	60.51
8800	825.801	153	11	60.25
9000	880.418	149	11	61.29
9200	815.798	149	11	61.29
9400	872.541	151	12	60.77
9600	815.861	152	12	60.51
9800	891.337	154	12	60.00
10000	821.050	153	12	60.25
10200	877.253	150	12	61.03
10400	820.065	150	12	61.03
10600	858.901	153	12	60.25
10800	809.430	154	12	60.00
11000	915.546	152	12	60.51
11200	803.968	152	12	60.51
11400	897.758	155	12	59.74
11600	798.242	152	13	60.51
11800	906.955	152	13	60.51
12000	804.861	151	13	60.77

Continued on next page

Table C-47 Continued from previous page

Threshold	Duration	Alerts	Size (%)	ALR (%)
12200	861.882	154	13	60.00
12400	784.896	151	13	60.77
12600	803.717	153	13	60.25
12800	846.310	155	13	59.74
13000	829.617	152	13	60.51
13200	649.887	155	13	59.74
13400	556.851	155	13	59.74
13600	562.738	153	13	60.25
13800	552.563	157	13	59.22
14000	553.838	159	13	58.70
14200	553.388	159	13	58.70
14400	556.486	156	13	59.48
14600	557.998	161	13	58.18
14800	554.878	157	13	59.22
15000	558.644	159	13	58.70
15200	560.699	157	13	59.22
15400	553.785	157	14	59.22
15600	556.918	157	14	59.22
15800	560.173	160	14	58.44
16000	554.884	158	14	58.96
16200	929.099	160	14	58.44
16400	904.095	160	14	58.44
16600	928.516	157	14	59.22
16800	848.858	161	14	58.18
17000	897.396	158	14	58.96
17200	820.995	161	14	58.18
17400	893.859	158	14	58.96
17600	883.868	162	14	57.92
17800	877.818	159	14	58.70
18000	923.504	160	14	58.44
18200	861.835	159	14	58.70

Continued on next page

Table C-47 Continued from previous page

Threshold	Duration	Alerts	Size (%)	ALR (%)
18400	928.422	159	14	58.70
18600	837.831	159	14	58.70
18800	898.177	162	14	57.92
19000	871.790	162	14	57.92
19200	861.870	159	14	58.70
19400	915.091	162	14	57.92
19600	844.196	162	14	57.92
19800	932.610	163	14	57.66
20000	833.606	161	15	58.18
20200	898.911	164	15	57.40
20400	829.592	161	15	58.18
20600	904.299	165	15	57.14
20800	892.342	163	15	57.66
21000	865.645	160	15	58.44
21200	800.422	163	15	57.66
21400	562.203	164	15	57.40
21600	561.490	164	15	57.40
21800	560.712	163	15	57.66
22000	557.376	160	15	58.44
22200	560.105	164	15	57.40
22400	560.508	160	15	58.44
22600	557.212	165	15	57.14
22800	556.652	164	15	57.40
23000	561.001	161	15	58.18
23200	563.022	162	15	57.92
23400	555.765	162	15	57.92
23600	557.363	162	15	57.92
23800	559.610	161	15	58.18
24000	559.009	161	15	58.18
24200	560.039	165	15	57.14
24400	561.390	164	15	57.40

Continued on next page

Table C-47 Continued from previous page

Threshold	Duration	Alerts	Size (%)	ALR (%)
24600	560.709	165	15	57.14
24800	565.126	164	15	57.40
25000	560.777	164	15	57.40
25200	562.177	164	15	57.40
25400	562.357	164	15	57.40
25600	561.479	161	15	58.18
25800	556.735	165	15	57.14
26000	559.664	166	16	56.88
26200	563.303	163	16	57.66
26400	556.274	162	16	57.92
26600	557.613	163	16	57.66
26800	557.035	163	16	57.66
27000	562.024	166	16	56.88
27200	559.405	166	16	56.88
27400	559.996	166	16	56.88
27600	558.556	166	16	56.88
27800	560.412	163	16	57.66
28000	564.058	164	16	57.40
28200	562.936	166	16	56.88
28400	559.352	163	16	57.66
28600	565.333	168	16	56.36
28800	560.847	166	16	56.88
29000	559.821	171	16	55.58
29200	561.644	168	16	56.36
29400	565.257	167	16	56.62
29600	559.354	170	16	55.84
29800	562.859	170	16	55.84
30000	559.858	165	16	57.14
30200	562.012	165	16	57.14
30400	560.701	165	16	57.14
30600	561.006	166	16	56.88

Continued on next page

Table C-47 Continued from previous page

Threshold	Duration	Alerts	Size (%)	ALR (%)
30800	565.952	166	16	56.88
31000	557.635	166	16	56.88
31200	559.030	168	16	56.36
31400	561.976	169	16	56.10
31600	559.401	169	16	56.10
31800	565.154	165	16	57.14
32000	560.780	169	16	56.10

Concluded

Table C-48: Results using the LINTS tainted flow capped by bytes with entropy strategy to compress the CIC 2017 testing data set and the RegJul2018 rule

Threshold	Duration	Alerts	Size (%)	ALR (%)
200	642.912	63	1	95.08
400	664.382	64	1	95.00
600	642.721	349	2	72.75
800	665.581	854	2	33.33
1000	645.878	1,057	2	17.48
1200	646.817	1,062	2	17.09
1400	648.007	1,062	3	17.09
1600	531.993	1,062	3	17.09
1800	446.287	1,064	4	16.93
2000	416.743	1,064	4	16.93
2200	408.136	1,081	4	15.61
2400	401.514	1,098	4	14.28
2600	401.534	1,098	4	14.28
2800	401.833	1,098	4	14.28
3000	401.604	1,098	5	14.28
3200	400.062	1,098	5	14.28
3400	401.648	1,098	6	14.28

Continued on next page

Table C-48 Continued from previous page

Threshold	Duration	Alerts	Size (%)	ALR (%)
3600	401.886	1,100	6	14.12
3800	403.656	1,100	6	14.12
4000	400.816	1,100	6	14.12

Concluded

List of Symbols, Abbreviations, and Acronyms

ALR	alert loss rate
ARL	Army Research Laboratory
ACCS	Australian Centre for Cyber Security
BPF	Berkeley Packet Filter
CAS	central analysis system
CDX	Cyber Defense Exercise
CIC	Canadian Institute for Cybersecurity
CIC-IDS	Canadian Institute for Cybersecurity Intrusion Detection System
CPU	central processing unit
CSE	Communications Security Establishment
CSRDSI	Cybersecurity Research Data Set Infrastructure
CSV	comma-separated value
DARPA	Defense Advanced Research Projects Agency
DDOS	distributed denial of service
DEVCOM	US Army Combat Capabilities Development Command
DNS	Domain Name Server
DOS	denial of service
DREN	Defense Research Engineering Network
DSDB	Data Set Database
FTP	File Transfer Protocol
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure

ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IMAP	Internet Message Access Protocol
ISCX	Information Security Centre of Excellence
ISCX-IDS	Information Security Centre of Excellence Intrusion Detection System
LINTS	Lossy Intelligent Network Traffic Squeezer
MACCDC	Mid-Atlantic Collegiate Cyber Defense Competition
NIDS	network intrusion detection system
NSA/CSS	National Security Agency/Central Security Service
PCAP	packet capture
PLS	Packet Loss Study
POP3	Post Office Protocol 3
SMTP	Simple Mail Transport Protocol
SNMP	Simple Network Management Protocol
SSH	Secure Shell
TCP	Transmission Control Protocol
UDP	User Datagram Procotol
UNB	University of New Brunswick
UNSW	University of New South Wales