Characterizing the Effectiveness of Network-based Intrusion Detection Systems

Eric Ficke*, Kristin M. Schweitzer†, Raymond M. Bateman† and Shouhuai Xu*
*Department of Computer Science
University of Texas at San Antonio
†U.S. Army Research Laboratory South - Cyber

***Abstract*—Network-based Intrusion Detection Systems (NIDSs) must detect and defend against many kinds of attacks. These defenses are certainly limited in their capabilities; however, there is a lack of precise understanding of their strengths and weaknesses. In particular, there are two kinds of NIDSs, *flow-based* vs. *packet-based*, whose effectiveness needs to be systematically characterized using real, or as real as possible, datasets with known ground truth. In this paper, we report our empirical study on using a modern dataset, with known ground truth about the attacks it contains, to evaluate the effectiveness of flow-based vs. packet-based NIDSs. This allows us to draw initial insights towards the ultimate characterization of the gap between flow-based and packet-based NIDSs.**

***Index Terms*—Intrusion Detection, Intrusion Detection Systems, Security Metrics, Snort, Suricata, Flow-based, Packet-based**

I. Introduction

Network-based Intrusion Detection Systems (NIDSs) are an integral part of cyber defense. These systems work to identify malicious network traffic and alert cyber defenders to such activities. When compared with Host-based Intrusion Detection Systems (HIDSs), NIDSs have a potential advantage when they can detect attacks before the attack reaches the victim computers. However, NIDSs have their own limitations. One such limitation is a high false-positive rate, which overwhelms cyber defenders because it is infeasible to respond to all the alerts with human analysts. Another limitation is that constantly-improving network infrastructure has created network throughput that NIDSs cannot match [1].

A candidate solution to the aforementioned problem is to use *flow-based* NIDS, which is in contrast to the previous solution of *packet-based* NIDS. This flow vs. packet distinction is important because a cyber attack may consist of one or multiple IP packets. More specifically, a flow is described by a tuple of (at least) the following elements: the source IP address, the destination IP address, the source port number, and the destination port number. They often also include timestamps for the beginning and end of the interaction. As a result, a flow containing one malicious packet should be considered an attack because each packet in a flow is intrinsically related (i.e., if an attacker initiates a flow, the attacker is responsible for its entirety, not just a single or subset of its packets). This explains why flow-based NIDS may reduce the number of unnecessary alerts that would be triggered by packet-based NIDS.

In this paper, we aim to characterize the gap between flow-based and packet-based NIDSs while using a modern dataset.

A. Contributions

We make the following contributions. First, we initiate the study on characterizing the gap between flow-based NIDSs and packet-based NIDSs through appropriate security metrics, such as the *true-positive* and *false-positive* rates. We propose and advocate the following objective standard for NIDS effectiveness: a NIDS should create exactly one alert for a flow *if and only if* the flow is malicious. Given that this practice may have the effect that NIDS developers choose to block alerts on a given flow after one alert has been produced, we stress that the correct practice would be to instead combine multi-alerts into a union, which contains descriptors for each attack behavior observed. These metrics can be used to characterize the capability of one kind of NIDS in detecting the attacks that cannot be detected by its competitors.

Second, we validate our methodology using two NIDSs (i.e., Snort [2] and Suricata [3]) and a modern dataset. These NIDSs use packet-based detection by default, but we additionally enable Snort's Stream Preprocessor in order to allow flow-based detection. Thus, we consider Suricata a packet-based NIDS and Snort a hybrid (flow/packet-based) NIDS.

Third, we analyze the difference between the NIDSs' detection capabilities by types of attacks so as to understand the differences in capability of packet-based vs. flow-based NIDS.

Fourth, we observe a pattern in the behavior of the attack traffic with respect to protocols (e.g., SSH vs. HTTP), namely that many protocols change in flow size when attacks are present. The direction and magnitude of the change varies by protocol, but such changes are likely indicators of attacks.

B. Related Work

From a technical point of view, modern NIDSs can be classified into two categories: *packet-based* vs. *flow-based*. The former approach inspects every network packet, including full or partial payload information, and therefore has a great potential in detecting malicious traffic. The latter approach considers the information available in packet headers and aggregates data such as flow volume and rate statistics, with the advantage of minimizing the per-packet evaluation time, enabling greater throughput, and being better able to fully monitor high-bandwidth networks [1]. Operating either at the

packet-level or flow-level, NIDSs employ one or more analysis techniques to determine the nature of the traffic [4]. The most widely used technique is to compare traffic data with known signatures of attacks, but this has very limited success in coping with new or zero-day attacks. Anomaly detection (e.g., [5], [6], [7], [8]) aims to go beyond this limitation, but often incurs the alternative cost of high false-positives.

From a security metrics point of view, *relative strength* of cyber defense tools has been investigated in [9], [10]. Intuitively, a defense has no extra value with respect to an employed set of defenses if it cannot detect any attack that cannot be detected by the employed defenses already. Naturally, the *collective strength* of defenses can be measured correspondingly [9], [10], [11], [12], [13], [14], [15]. It is worth mentioning that a collective use of multiple defense tools or mechanisms (e.g., anti-malware tools [11], [12], [13], [14]) may still be far from perfect or even acceptable.

From an evaluation point of view, datasets are important to measure the effectiveness of NIDS via appropriate security metrics [16]. However, high quality datasets are extremely hard to obtain. For example, the popular DARPA'99 dataset [17] has many shortcomings, especially its age (19 years), which means that it does not reflect current cyber threats. Another dataset contains traffic labeled on the flow level, enabling the comparison of flow-based and packet-based NIDS systems [18], but its contents are primarily malicious traffic collected from a single honeypot. Because of this, it may not be reflective of cyber attacks against production enterprise networks. These shortcomings lead us to utilize the 2012 dataset from the University of New Brunswick's Information Security Center of Excellence (ISCX), which contains sufficient size (~92G), testbed architecture (including several realistic subnets and DMZ), and tagging (which is flow-based and includes 24 hours of benign training data) [19].

C. Paper Outline

The paper is organized as follows. We describe the problem and methodology for solving the problem in Section II. We conduct a case study in Section III. We discuss future research in Section IV. We conclude the paper in Section V.

II. Problem Statement and Methodology

A. Problem Statement

The problem is to characterize the strength and weakness of flow-based vs. packet-based NIDSs. More specifically, we want to answer the following Research Questions (RQs):

- RQ1: How can we characterize the detection capabilities of flow-based vs. packet-based NIDSs?
- RQ2: How can we characterize the relative detection capabilities of flow-based vs. packet-based NIDSs?
- RQ3: How can we characterize the detection capabilities of NIDSs with respect to different types of attacks?
- RQ4: How can we characterize potential indicators of attacks?

RQ1. This question is to investigate the detection capabilities of flow-based vs. packet-based NIDSs. Detection capabilities can be measured by standard metrics such as True-Positive Rates (TPR) and False-Positive Rates (FPR) [16].

RQ2. This question is to investigate the relative capabilities of flow-based vs. packet-based NIDSs. We propose using the following metrics to measure the relative detection capabilities.

For an NIDS, let $Alerts(NIDS)$ denote the set of alerts it generates with respect to a network during a period of time. For a set of alerts A, let $TP(A)$ denote the set of True-Positive alerts belonging to A, and $FP(A)$ denote the set of False-Positive alerts belonging to A.

Definition 1 (SecurityGain): Suppose the defender already employed one NIDS, denoted by $NIDS_1$ (e.g., flow-based NIDS). The security gain that can be incurred by the employment of another NIDS, denoted by $NIDS_2$ (e.g., packet-based NIDS), can be defined as:

$$\begin{aligned} & SecurityGain(NIDS_1, NIDS_2) \\ = \quad & \frac{TP(Alerts(NIDS_1) \cup Alerts(NIDS_2))}{TP(Alerts(NIDS_1))} - 1, \end{aligned}$$

where without loss of generality $TP(Alerts(NIDS_1)) > 0$.

Note that security gain is in the range of $[0, \#Attacks]$. Security gain will be zero if $Alerts(NIDS_2) \subseteq Alerts(NIDS_1)$, meaning that the alerts from $NIDS_2$ are a subset of the alerts from $NIDS_1$.

Definition 2 (FPIncrease): Suppose the defender already employed one NIDS, denoted by $NIDS_1$. The side-effect of employing another NIDS, denoted by $NIDS_2$, can be measured by the increase to the False-Positives as:

$$\begin{aligned} & FPIncrease(NIDS_1, NIDS_2) \\ = \quad & \frac{FP(Alerts(NIDS_1) \cup Alerts(NIDS_2))}{FP(Alerts(NIDS_1))} - 1, \end{aligned}$$

where without loss of generality $FP(Alerts(NIDS_1)) > 0$.

Definition 3 (RelativeValue): Suppose the defender already employed one NIDS, denoted by $NIDS_1$. The relative value for employing another NIDS, denoted by $NIDS_2$, can be measured by combining the preceding metrics as:

$$\begin{aligned} & RelativeValue(NIDS_1, NIDS_2) \\ = \quad & \frac{SecurityGain(NIDS_1, NIDS_2)}{FPIncrease(NIDS_1, NIDS_2)}, \end{aligned}$$

assuming $FPIncrease(NIDS_1, NIDS_2) \neq 0$.

These metrics can be adapted to support alert combinations besides the union which we use. Depending on the function substituted, a metric may return negative values indicating a security loss or false positive decrease.

RQ3. This question is to investigate the strength and weakness of each NIDS in terms of the types of attacks it can and cannot detect. Understanding these aspects will shed light on enhancing the detection of NIDSs and understanding what attack strategies may be intentionally crafted against an NIDS.

RQ4. This question is to identify indicators that may be robustly used to detect cyber attacks. Ideally, these indicators should be hard to evade. For example, flow size (e.g., the number of packets in a flow) may be an indicator because it can suggest substantial changes in network behaviors.

B. Data Description

The dataset comes from a testbed network of user-generated attack traffic from the University of New Brunswick's Information Security Centre of Excellence (ISCX) [19]. The dataset contains seven days of mixed malicious and benign traffic from a like-real network architecture, including flow descriptors and ground-truth tags. The dataset contains benign traffic generated from some statistical models of real traffic, as well as manually conducted attacks. Each day's traffic scenario is summarized in Table I.

TABLE I
UNB ISCX DATASET

Date	# Flows	# Attacks	Description
6/11/2012	474,278	0	Normal Activity
6/12/2012	133,193	2,086	SSH brute-force
6/13/2012	275,528	20,358	Internal infiltration
6/14/2012	171,380	3,776	HTTP DoS
6/15/2012	571,698	37,460	IRC Botnet DDoS
6/16/2012	522,263	11	SSH brute force
6/17/2012	397,595	5,219	SSH brute force
Total	2,545,935	68,910	2.71% attack traffic

Because the dataset includes over a full day of entirely benign traffic, it provides a sufficient baseline for training anomaly-based NIDSs. The NIDS examples used in this paper do not make use of the baseline traffic, so we omit it from our results. We also exclude the capture on 6/16 because it contains too few attacks. Thus, our measurements correspond to 6/12-6/15 and 6/17 in the original dataset.

We classify the attacks in each day according to the following: if one detector is capable of detecting an attack in a certain group, it is capable of detecting all attacks within that group. For example, SQL injection and buffer overflows are grouped into "input validation". That is, a monitor that is able to detect an SQL injection should also be capable of detecting the buffer overflow attack.

First, for the traffic on the days with the *SSH brute force* attack (6/12, 6/17), all of the attacks are encompassed by this description (i.e., "SSH brute force").

Second, the traffic corresponding to 6/13 contains a mix of various types of attacks. The scenario begins with a malicious pdf-based *input validation* exploit transmitted via email across port 25. This opens a *reverse shell* on port 5555. Botnets are installed, using ports 6660-9 for *command & control* channels. These bots use *input validation* attacks against an SQL server on port 80 and conduct Nmap *network scanning* for new targets using a multitude of registered and unregistered ports.

Third, the HTTP DoS on 6/14 attack contains multiple attack types as well. We know based on the ISCX paper that this scenario contains an SMB (port 445) attack which includes a stack exploit. We consider this an *input validation* attack. Another known attack in this scenario is of course the *DoS attack* on HTTP port 80. The remainder of the attacks cover over 600 other registered and unregistered ports, so we consider them *network scanning*.

Fourth, the traffic on 6/15 contains several of the attacks identified in the internal infiltration scenario, namely the pdf-based *input validation*, *reverse shell*, and *command & control*. This scenario also contains a *DDoS attack* on HTTP port 80. The remainder of the attacks land on unregistered ports and are grouped into *network scanning*.

C. Data Processing

To prepare our data processing, we create two databases, one for flow-based analysis and the other for packet-based analysis. We use Python's MongoDB module, *pymongo*, to manage the databases. We divide the databases into one collection per day of traffic. We link the packet database to the flow database in order to create the ground-truth tags at the packet level. The 5-tuple used to match packets to flows includes: *source IP address*, *destination IP address*, *source port*, *destination port*, and *time*. We confirm that they reside between the correct flow's start time and stop time, accounting for time zone shift. Note that not all timing tags may precisely encompass the flow window. In this case, we assign the packet to all flows matching the remainder of the 5-tuple (excluding *time*).

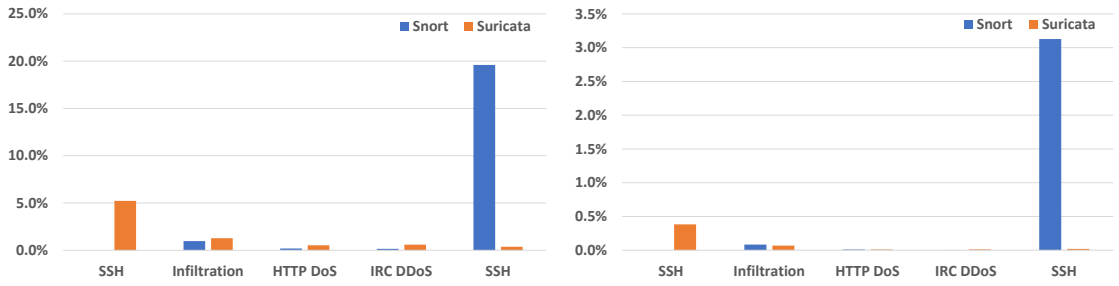
The data analysis methodology is described as follows.

1) We replay the network traffic through each NIDS tested. This produces alerts according to the rule sets defined in the device configuration. For these experiments, we use only the rule sets directly available from the publishers at the time of installation, with no modification. The configurations of each product are modified minimally with changes such as configuring the home network and enabling the Snort preprocessor.
2) We add the alerts as fields to the packet database. Each alert can be uniquely mapped to a packet using the timestamp, which has nanosecond precision. When timestamp precision may not be enough to uniquely identify each packet, verification of source/destination IP address or port number should be sufficient.
3) Once alerts have been added to the packet database, they can be mapped to flows using the associations created earlier. This allows us to semantically convert the packet-based output of our NIDSs into flow-based alerts. This step is important because it enables the comparison of packet-based and flow-based NIDSs.
4) The final step is a simple query of the flow database to find flow tags and whether an alert was produced for each. We measure True-Positive Rate (TPR) and False-Positive Rate (FPR) for each NIDS as:

$$TPR = \frac{\text{\# of correct alerts}}{\text{\# of "Attack" tags}}, \tag{1}$$

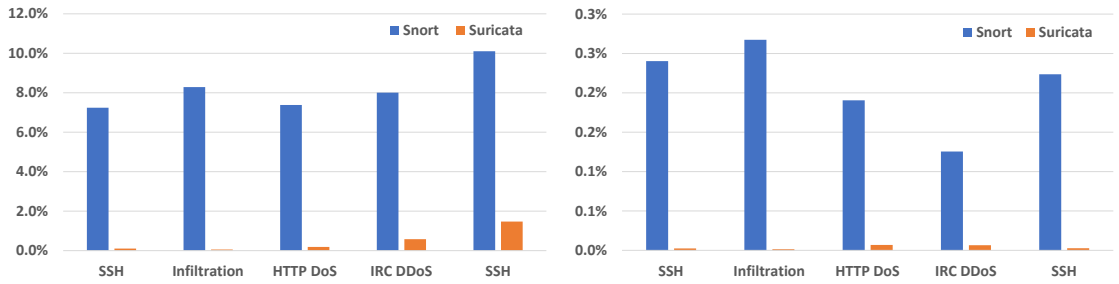
$$FPR = \frac{\text{\# of incorrect alerts}}{\text{\# of "Normal" tags}} \tag{2}$$

In this step, we also query the database to find the measurement of the combined effectiveness of the two NIDSs (i.e., $Alerts(NIDS_1) \cup Alerts(NIDS_2)$).



(a) Flow-based Measurement (b) Packet-based Measurement

Fig. 1. True-Positive Rates (TPR)



(a) Flow-based Measurement (b) Packet-based Measurement

Fig. 2. False-Positive Rates (FPR)

We also directly compare this TPR for each NIDS to understand how its approach performs for each attack type relative to the other NIDSs.

III. Results

A. RQ1: Characterizing Detection Capabilities of Flow-based vs. Packet-based NIDSs

Figure 1 shows the TPR for Snort and Suricata, both for the flow-based perspective (1a) and packet-based perspective (1b). This figure demonstrates that flow-based NIDS has comparable results to packet-based NIDS, despite the scale being higher.

Figure 2 demonstrates that for FPR, flow-based NIDSs lead to similar results as packet-based NIDSs. In both of these cases, the change in scale is coherent with the intuition that each flow contains many packets and that a majority of the packets within a malicious flow are *not* independently identifiable as attacks.

These figures also reveal the effectiveness of these NIDSs, from which we draw the following:

Insight 1: Suricata outperforms Snort in terms of the True-Positive Rate and the False-Positive Rate. Despite Snort's higher net True-Positive Rate (from its 6/17 output), its excessive False-Positive Rate far outweighs its benefits. For these attacks, it is clear that Suricata is the better NIDS.

B. RQ2: Characterizing Relative Detection Capabilities of Flow-based vs. Packet-based NIDSs

Table II summarizes the measurement of $SecurityGain(Suricata, Snort)$, denoted "Snort", and $SecurityGain(Snort, Suricata)$, denoted "Suricata", for both flow and packet metrics. We observe that the flow-based NIDS generates results with similar scale to the packet-based NIDS. The result on 6/12 is reflective of the fact that snort generated 0 True-Positives. The result on 6/13 is inconsistent with our assertion that these NIDSs produce similar results (i.e., Snort performs better in the packet-based measurement, but not in the flow-based measurement). In this case, the discrepancy is caused by imprecision in the labeling of flow windows (as explained in II-C).

TABLE II
FLOW-BASED $SecurityGain(X, Y)$, DENOTED BY Y (FL) AND PACKET-BASED $SecurityGain(X, Y)$, DENOTED BY Y (PK)

Date	Snort (Fl)	Suricata (Fl)	Snort (Pk)	Suricata (Pk)
6/12	0%	Undefined	0%	Undefined
6/13	42%	86%	117%	78%
6/14	35%	286%	100%	100%
6/15	25%	402%	15%	667%
6/17	5,374%	2%	16,516%	1%

Table III summarizes $FPIncrease(Suricata, Snort)$, denoted "Snort", and $FPIncrease(Snort, Suricata)$, denoted

"Suricata". These results are very clear: Snort introduces many more false positives than Suricata in all cases tested. One reason for this observation is that many of Snort's rules are low priority, diagnostic or are related to misuses that may have been improperly tagged as normal in this dataset. Specifically, we find 26 signatures which, for one or more days, create exclusively false positives. Several examples of these include: "[128:4:1] (spp_ssh) Protocol mismatch", "[119:31:1] (http_inspect) UNKNOWN METHOD" and "[122:25:1] (portscan) ICMP Sweep".

TABLE III
FLOW-BASED $FPIncrease(X, Y)$, DENOTED BY Y (FL) AND PACKET-BASED $FPIncrease(X, Y)$, DENOTED BY Y (PK)

Date	Snort (Fl)	Suricata (Fl)	Snort (Pk)	Suricata (Pk)
6/12	6,905%	1%	9,926%	1%
6/13	14,647%	0%	19,510%	1%
6/14	3,900%	2%	2,768%	4%
6/15	1,367%	6%	1,914%	5%
6/17	684%	14%	8,303%	1%

From a simple glance at Tables II and III, it is obvious that $RelativeValue(Snort, Suricata)$, denoted "Suricata" is higher for days 6/12-6/15, and that $RelativeValue(Suricata, Snort)$ is higher for 6/17. Table IV shows the relative value of each system compared to the other. Notice that "Suricata" is far more significant on 6/12-6/15 than "Snort" on 6/17. From this observation, we draw Insight 2.

Insight 2: For the results on 6/17 (SSH brute force scenario), Snort outperforms Suricata according to the $RelativeValue$ metric. One might assume that Snort's stream preprocessor enables detection of these attacks because of the stark difference from Suricata's result, but this is not the case here. In contrast, we observe that Snort fails to produce even one true positive on 6/12, with the same attack scenario. Further exploration into the data reveals that 68% of Snort's TP alerts on the 6/17 data is generated by a single signature: "[128:4:1] (spp_ssh) Protocol mismatch" and the remainder by another: "[129:15:1] Reset outside window". These alerts hardly describe the reality of the attack. In light of this, these results realistically represent a failure on Snort's part.

TABLE IV
SUMMARY OF FLOW-BASED $RelativeValue(X, Y)$, DENOTED BY Y (FL) AND PACKET-BASED $RelativeValue(X, Y)$, DENOTED BY Y (PK).

Date	Snort (Fl)	Suricata (Fl)	Snort (Pk)	Suricata (Pk)
6/12	0	Undefined	0	Undefined
6/13	353^{-1}	202	167^{-1}	152
6/14	111^{-1}	154	28^{-1}	28
6/15	55^{-1}	63	128^{-1}	131
6/17	8	9^{-1}	2	.5

In this case, it is clear that Suricata should be chosen over Snort. In other experiments where these results may not be immediately clear, one may also consider the geometric mean of the $RelativeValues$: in this case, these values are 22 and 23 for Suricata (by flow and packet) and 23^{-1} in both cases for Snort. Note that for this application, we weight each day equally because of the semantic grouping of the attacks in the ISCX dataset. In other cases, it may make more sense to weight each flow (rather than day) equally. In light of these observations, we draw Insight 3.

Insight 3: The $SecurityGain$ and $FPIncrease$ metrics provide an intuitive understanding of each NIDSs contribution to an existing security environment. The resultant $RelativeValue$ ratio gives a strong comparative indicator of the benefits associated with installing the additional device. This also gives a direct means of conducting a cost-benefit analysis of two systems, making the decision to adopt a new technology quantifiable.

C. RQ3: Characterizing Capabilities of NIDSs in Detecting Different Types of Attacks

In figure 3, we show the effectiveness of each NIDS with their outputs grouped by the attack types detailed in II-B. Because both Snort and Suricata have packet-signatures enabled but only Snort uses flow-signatures, we are only interested in the $SecurityGain$ of Snort. Clearly the only major gain in this case is the detection rate for *SSH brute force* attacks. This leads us to Insight 4.

Insight 4: In general, the addition of flow-based signature processing has not added coverage for any new attack types. However, this approach greatly increased the detection effectiveness again *SSH brute force* attacks. One possible reason for this is that the nature of SSH traffic (i.e., its encryption) prevents packet-based signatures from being effective against it. Indeed, it is possible that Snort's analysis was able to detect suspicious activity that Suricata didn't because of its flow-based Stream Preprocessor.

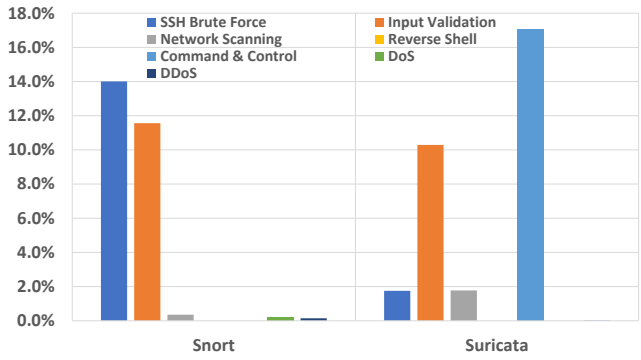


Fig. 3. True-Positive Rates by attack types

D. RQ4: Characterizing Potential Indicators of Attack

Figure 4 plots the average number of packets in normal or malicious flows, grouped by network protocol and day. We look at the overall averages, as well as those for HTTP and SSH, the two representative victims in the dataset. These ratios could be used to help identify changes in flow size, which may

be an early indicator of an attack. The majority of these attacks show that the flow size (in the number of packets) decreases by roughly 50% during attacks. The only exception to this is the IRC DDoS scenario on 6/15. In this case, we see an extreme spike in flow size for HTTP traffic, as a result of the excessive quantity of web requests induced by the attack. Because of these patterns, we conclude with:

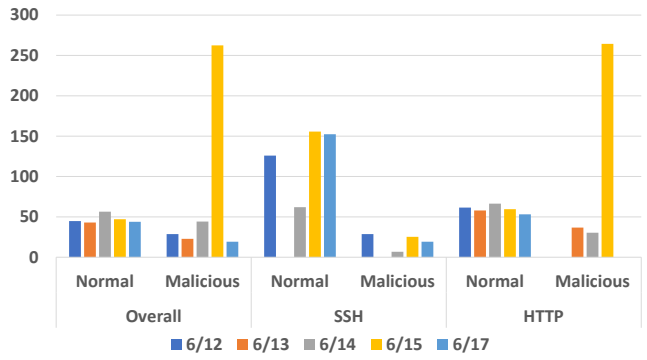


Fig. 4. Average flow size (in the number of packets) for some common protocols, grouped by protocol and day.

Insight 5: Flow size (in the number of packets) for SSH and HTTP traffic drops by roughly half during attacks. This observation could be used to improve behavioral analysis for detecting these types of attacks (specifically, HTTP DoS/DDoS and SSH brute force).

IV. Limitations

The present study has several limitations. (i) We only focused on two commercial NIDSs. In order to gain a deeper understanding of the strengths and weaknesses of the state-of-the-art NIDSs, future research needs to accommodate other (commercial) flow-based and/or anomaly-based NIDSs. Equally important, future research needs to consider other datasets of a similar nature, if available. (ii) Because of the imprecision in tagging the flows in the ISCX dataset (i.e., $startDateTime$ and $stopDateTime$), we relax our interpretation of matching flows to only the 4-tuple of (*source IP address, destination IP address, source port, destination port*). This means that both TPR and FPR may be inflated because one packet may have been associated to more than one flow. (iii) Our alert combination approach uses a simple union to compare the outputs of two NIDSs. This is a naive approach. Future research should consider alert mass or reliability in alert fusion [20], [21], which can improve the cost of False-Positives without reducing the TPR. (iv) Insight 5 represents flow behavior of a naive attacker. In reality, this observation may not be applicable to adversarially-crafted flows, which may be intentionally altered to deceive the detector [22].

V. Conclusion

We have empirically characterized the detection capabilities of flow-based vs. packet-based NIDSs. We offered some initial findings in terms of the strength and weakness of each kind of NIDS. We found that Suricata evidently outperforms Snort by a large margin. Moreover, we found that flow size can be an indicator of cyber attacks. Future work will include deeper analysis of the dataset, especially explaining how the attacks precisely evaded the NIDSs.

Acknowledgements. The authors would like to thank Dr. Arash Habibi Lashkari for providing us the access to the UNB ISCX 2012 Dataset that is used in the present study. This work was supported in part by ARL grant #W911NF-17-2-0127.

References

[1] A. Sperotto and A. Pras, "Flow-based intrusion detection," in *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, pp. 958–963, May 2011.
[2] "Snort - network intrusion detection & prevention system," Mar 2018.
[3] "Suricata | open source ids / ips / nsm engine," Mar 2018.
[4] S. Axelsson, "Intrusion detection systems: A survey and taxonomy," tech. rep., Technical report, 2000.
[5] J. Gao, G. Hu, X. Yao, and R. Chang, "Anomaly detection of network traffic based on wavelet packet," in *Proc. IEEE APCC'2006*, pp. 1–5, 2006.
[6] P. Parveen, Z. Weger, B. Thuraisingham, K. Hamlen, and L. Khan, "Supervised learning for insider threat detection using stream mining," in *Proc. IEEE ICTAI'2011*, pp. 1032–1039, 2011.
[7] N. Lu, S. Mabu, T. Wang, and K. Hirasawa, "Integrated fuzzy gnp rule mining with distance-based classification for intrusion detection system," in *Proc. IEEE SMC'2012*, pp. 1569–1574, 2012.
[8] E. Biermann, E. Cloete, and L. M. Venter, "A comparison of intrusion detection systems," *Computers & Security*, vol. 20, no. 8, pp. 676–683, 2001.
[9] N. G. Boggs and S. Stolfo, "Aldr: A new metric for measuring effective layering of defenses," *Fifth Layered Assurance Workshop (LAW'11)*, 2011.
[10] N. Boggs, S. Du, and S. Stolfo, "Measuring drive-by download defense in depth," in *Proc. RAID'14*, pp. 172–191.
[11] J. A. Morales, S. Xu, and R. Sandhu, "Analyzing malware detection eciency with multiple anti-malware programs," *ASE Science Journal*, vol. 1, no. 2, pp. 56–66, 2012.
[12] A. Mohaisen and O. Alrawi, "Av-meter: An evaluation of antivirus scans and labels," in *Proc. DIMVA'2014*, pp. 112–131, 2014.
[13] D. Yardon, "Symantec develops new attack on cyberhacking." http://www.wsj.com/articles/SB10001424052702303417104579542140235850578, May 4, 2014.
[14] P. Du, Z. Sun, H. Chen, J. Cho, and S. Xu, "Statistical estimation of malware detection metrics in the absence of ground truth," *IEEE Trans. Information Forensics and Security*, vol. 13, no. 12, pp. 2965–2980, 2018.
[15] H. Chen, J. Cho, and S. Xu, "Quantifying the security effectiveness of firewalls and dmzs," in *Proc. HoTSoS'2018*, pp. 9:1–9:11, 2018.
[16] M. Pendleton, R. Garcia-Lebron, J.-H. Cho, and S. Xu, "A survey on systems security metrics," *ACM Comput. Surv.*, vol. 49, pp. 62:1–62:35, Dec. 2016.
[17] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 darpa off-line intrusion detection evaluation," *Comput. Netw.*, vol. 34, pp. 579–595, Oct. 2000.
[18] A. Sperotto, R. Sadre, F. van Vliet, and A. Pras, *A Labeled Data Set For Flow-based Intrusion Detection*, pp. 39–50. Lecture Notes in Computer Science, Springer Verlag, 10 2009.
[19] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers & Security*, vol. 31, no. 3, pp. 357 – 374, 2012.
[20] C. Katar, "Combining multiple techniques for intrusion detection," in *International Journal of Computer Science and Network Security*, vol. 6, February 2006.
[21] V. M. Shah and A. K. Agarwal, "Reliable alert fusion of multiple intrusion detection systems," *I. J. Network Security*, vol. 19, pp. 182–192, 2017.
[22] I. Corona, G. Giacinto, and F. Roli, "Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues," *Information Sciences*, vol. 239, pp. 201 – 225, 2013.