Increasing Diversity in Network Intrusion Detection Systems Evaluation

Victor C. Valgenti and Min Sik Kim
Petabi, Inc.
Irvine, CA 92612
USA
Email: vvalgenti@petabi.com

Abstract—The performance of Network Intrusion Detection Systems (NIDS) depends heavily on the inputs to the system (rules and network traffic). A common trend in the evaluation of NIDS is to use a narrow selection of publicly or privately available rule-sets and traffic. Private rule-sets and traffic make the repeatability of experiments difficult while publicly available rule-sets and traffic often lack the diversity to explore the NIDS’s true operating range. This can cause misleading results in the face of inputs that do not adequately test the NIDS. To improve diversity and provide better context for evaluations it is necessary to employ synthesized traffic and rules in addition to the use of public or private traffic and rule-sets. This research expands on previous models and tools to provide systematic means for increasing the diversity and context of any evaluation providing for a broader perspective from which to view NIDS performance and compare results.

I. INTRODUCTION

Network Intrusion Detection Systems (NIDS) typically require a minimum of two inputs: The network traffic to examine, and a set of rules to use during examination. For the case of anomaly-detection, the set of rules is replaced by the algorithm used for detection. Regardless, the inputs into the system significantly impact the results of any experiment or evaluation. The use of public, or local, packet captures in NIDS evaluations provides value in demonstrating the performance of the NIDS under very specific conditions (i.e. the state of the network that generated the trace). However, generalizations made from these results do not necessarily translate to other scenarios as the traffic captures may not contain sufficient diversity to explore the performance of the system under test. Further, when proprietary traffic or rule-sets are used in evaluation there are a host of privacy and business issues that may prevent the release of those rule-sets or traffic traces making the repeatability of experiments difficult.

The problem extends from the lack of a uniform method to produce test data which, in turn, extends from the wide diversity of traffic features utilized by NIDS to arrive at a decision. There exists no official base-line definition of network traffic, nor is there any attempt to examine the boundaries of input into a system. Instead, evaluations tend to use the means closest at hand—often assuming that one particular input is sufficient. Certainly, the use of publicly available rule-sets or traffic can eliminate many concerns over fairness, but often results in myopic evaluations.

In this work we offer both the tools and consistent models for establishing greater diversity in evaluations of NIDS. First, we define the extremes of inputs for NIDS evaluation testing. Second, we provide tools for the generation of traffic and rules to effectively create these inputs. Finally, we demonstrate case-study usage of these tools and techniques for the evaluation of NIDS. The ultimate goal of this work is to offer not only a model for NIDS evaluation but the support for that model as well. Our hope is that these tools and this paper will provide for more diverse and consistent NIDS evaluation.

II. RELATED WORK

There have been several papers exploring the impact on performance to NIDS dependent on the rule-set and the traffic involved [1]–[5]. Adjusting either the traffic or rule-set for the NIDS can significantly change results from evaluation. Traffic that matches deeply with the rule-set will cause greater burden on the NIDS [1], [6], [7] while traffic that matches shallowly will be processed at maximum efficiency. This idea holds to anomaly detection as well as the background traffic for a particular experiment can greatly affect the outcome for anomaly-detection algorithms.

The work by Michella Becchi et al. [2] provided a foundation for workload evaluation of regular expression matching in deep-packet inspection. The core of this research is a traffic generation model that uses a tunable variable that provides for generating traffic that will match a set of compiled regular expressions deeply or shallowly depending on the setting. This provides a means to examine the workload of a large matching automata compiled from many regular expressions against traffic that intersects with that matching automata at varied depths. We extended this model to better include other features typically targeted by NIDS [3]. However, we have since realized that the work outlined earlier is very specific to deep-packet inspection and not necessarily relevant to the larger audience of NIDS evaluation. In this work, we provide a means to explore the range of inputs given to the NIDS so as to provide broader perspective when examining the results from specific network traffic or rule-sets.

III. DEFINING INPUTS

The two primary inputs into a NIDS are the traffic and the rule-set (or algorithm) used. These two inputs define the test
space as illustrated in Figure 1. As rule complexity grows, the NIDS has more difficulty in arriving at a decision. Similarly, as the traffic begins to match more heavily with the rules, the NIDS typically must work harder to process the traffic. These extremes illustrate the range of input for NIDS and can cause significant variance in results if not considered during evaluation. One can expect simple rules to perform better than complex rules, and traffic with little intersection into the investigated features of the NIDS to offer little difficulty in processing. The question, however, is how to define the inputs into the system.

A. Defining Traffic Inputs

The primary technique for evaluating NIDS is to expose the NIDS to either a live network or to use packet captures derived from some network. This tests the state of a network and can prove an admirable indicator of fitness for specific network environments and conditions but does not explore the range of potential inputs to the system. The problem is that such network traffic may simply lack the features necessary to properly examine a NIDS. Further, public data sets soon become outdated and not necessarily relevant to modern trends [8], [9] while private data sets make repeatability of experiments difficult. This motivates our desire to define inputs to the system that will evaluate the range of possible traffic input and ultimately provide better context for quantifying results.

For a NIDS, traffic is decomposed into sets of data, as illustrated in Figure 2, that are examined to arrive at a decision. The large circles in Figure 2 represent the smallest set of data used by a NIDS to arrive at a decision. This set of data could be a single datagram, a series of communications, or even a series of events. We term this element the quantifiable unit and note 3 such elements in Figure 2 designated $Q_1$ through $Q_3$ and define the quantifiable unit as per Definition 1.

**Definition 1. Quantifiable Unit:** For any NIDS there exists a quantifiable unit, $Q$, representing the smallest set of data upon which the NIDS may make a decision.

**Definition 2. Traffic-set:** For any set of network traffic $T$ there exists $m$ quantifiable units such that $(Q_1, Q_2, ..., Q_m \in T)$ for some $m$.

Within each quantifiable unit the NIDS examines specific features of that data. These features may be fixed elements, such as header or data values, or statistics aggregated from a series of communications. We term these traffic features and they represent the set of meaningful data that might occur for that quantifiable unit and are labeled $F_1$ through $F_i$ in Figure 2. We formally define traffic features in Definition 3.

The NIDS examines the traffic features of each quantifiable unit and throws an alert if a sufficient number of features match the patterns, values, or thresholds for which the NIDS is searching.

**Definition 3. Traffic Features:** Within any quantifiable Unit, $Q$, there exists a set of $n$ traffic features, $(F_1, F_2, ..., F_n \in Q)$, where each traffic feature $F_i \in Q$ is the set of specific values or states possible for that particular traffic feature where $0 < i \leq n$.

Under these constraints, the NIDS attempts to detect patterns in the traffic. From this we derive the extremes of traffic input. First, traffic that approximates random noise across all traffic features of interests offers a ‘Ground Truth’ evaluation. This is due to the fact that such traffic will be devoid of overt patterns, diverse in that many varied values should appear in any feature examined, and ultimately should cause few or no alerts. Such Random traffic offers the closest facsimile of purely innocuous traffic while maintaining a diversity of values within the traffic. Conversely, traffic where every quantifiable element matches to a pattern illustrates the exact opposite of Random traffic. In this instance, the ‘Matching’ traffic demonstrates pandemic circumstances where all traffic is of interest. While Random and Matching traffic define the primary extremes of input for NIDS, we note that there is still one more factor to consider: diversity. Random traffic remains the most diverse as given a large enough sample then all possible values for a particular feature are likely to occur in the traffic. Diversity of matching traffic depends on the rules or algorithm used, but can be considered less than that for Random. However, traffic with limited diversity can create strange results during evaluation. For example, a set of traffic that exhibits very little change between datagrams can become either very easy for the NIDS to evaluate or very difficult depending on how the values in the features align with the rule-set or algorithm used by the NIDS. The extreme case is what we term ‘Homogeneous’ traffic where the same value is used for every feature for every quantifiable unit. Figure 3 illustrates these extremes of traffic input and we formally define Random traffic.
traffic in Definition 4, then Matching traffic in Definition 6, and finally Homogeneous traffic in Definition 7.

**Definition 4. Random Traffic:** a set of network traffic, \( T_{\text{rand}} \), where the distribution of any specific value, \( v \in F_i \), for any traffic feature \( F_i \in Q_j \), for any \( Q_j \in T_{\text{rand}} \) is approximately \( \frac{1}{|F_i|} \) where \( |F_i| \) represents the cardinality of \( F_i \) for \( 0 < j \leq m \) and \( 0 < i \leq n \).

For example, assume that the quantifiable unit is the network datagram. Further, assume that one of the features, \( F_1 \), examined by the NIDS is the destination port of the Transport Control Protocol (TCP) header. Then for the set of random traffic \( T_{\text{rand}} \), the probability of any specific value occurring in the destination port field of the TCP header of any packet should approximate \( \frac{1}{65536} \) as there are 65536 possible values for that particular field.

**Definition 5. Matching Features:** For any traffic feature \( F_i \) in a quantifiable unit \( Q \) there exists a set of specific values, \( F'_i \), known to the NIDS that define suspicious behavior such that \( F'_i \subseteq F_i \) and \( 0 < i \leq n \).

**Definition 6. Matching Traffic:** a set of network traffic, \( T_{\text{matching}} \), where any specific value, \( v \in F_i \), for traffic feature \( F_i \in Q_j \) must also meet \( v \in F'_i \) for any \( Q_j \in T_{\text{matching}} \) and \( 0 < j \leq m \) and \( 0 < i \leq n \).

Continuing the example from Definition 4, assume that the specific values of interest to the NIDS for the TCP destination port, traffic feature \( F_1 \), are 8080 and 8888 making \( F'_1 = (8080, 8888) \). Thus, in \( T_{\text{matching}} \) the value for the destination port for any TCP packet will be either 8080 or 8888 with equal probability.

**Definition 7. Homogeneous Traffic:** a set of network traffic, \( T_{\text{homogeneous}} \), where there exists a single value \( v \in F_i \) such that for all \( F_i \) in all \( Q \in T_{\text{homogeneous}} \), \( F_i = v \).

In similar form to the example for Definition 6, assume that a single value, \( v \), has been randomly chosen from the possible range of TCP ports (0-65535) and \( v = 8008 \). Thus, in \( T_{\text{homogeneous}} \) the value for any destination port for any TCP packet will be exactly 8008.

These traffic extremes provide a broader context from which to compare the results from specific network environments. Random traffic offers innocuous, diverse traffic while Matching traffic will illustrate the impact of pandemical situations. Homogeneous traffic will offer insight into the impact of homogeneity on the NIDS—demonstrating likely either worst-case or best-case results. None of these traffic extremes are considered ‘Real-world’ inputs, but they offer more context for understanding the results from specific network conditions.

**B. Defining Rule-sets**

Regardless whether or not a NIDS uses rules, all NIDS examine specific features of the network traffic to aid in their decision making as illustrated in Figure 2. Rule-sets define the sets of specific values for traffic features during matching. Those rule-sets that define larger ranges of values and/or employ ambiguous matching are more complex than rule-sets that employ fixed values for matching. The common approach for using rules in evaluation of NIDS is to employ publicly available rule-sets or to use proprietary rule-sets. Proprietary rule-sets suffer many of the same privacy and Intellectual Property concerns that traffic traces from proprietary networks suffer. As such, they are often not provided to the public making the repeatability of any test difficult. Publicly available rule-sets used in research are often out-dated and may not represent the current state of the rules. Publicly available and proprietary rule-sets offer valuable insight into how a NIDS operates when using specific rule-sets but it remains difficult to quantify that performance.

**Definition 8. Simple Rule-set:** for any rule \( r_1, r_2, ..., r_l \in R_{\text{simple}} \), where \( R_{\text{simple}} \) is the total set of \( l \) rules, there is a set of traffic features targeted by any given rule, \( r_j(r_{j1}, r_{j2}, ..., r_{jl} \in r_j) \) where \( 0 < j \leq l \). For every rule traffic feature, \( r_{jl} \), a finite fixed value, \( v \), may be chosen from the set of all possible values for feature \( F_i \) for \( 0 < i \leq n \) and \( n \) is the number of traffic features. As such, for any rule traffic feature \( r_{jl} \), a specific value \( v \in F_i \) is assigned with probability \( \frac{1}{|F_i|} \) where \( |F_i| \) represents the cardinality of \( F_i \).

For example, assume that \( r_{jl} \) examines the destination port in the TCP header of the network traffic. There are 65536 possible exact matches for a port, thus \( v \) is randomly chosen from \( F_i \) with the probability of any specific value chosen approximately \( \frac{1}{65536} \) for any rule in the rule-set.

**Definition 9. Complex Rule-set:** for any rule \( r_1, r_2, ..., r_l \in R_{\text{complex}} \), where \( R_{\text{complex}} \) is the total set of \( l \) rules, there is a set of traffic features targeted by any given rule, \( r_j(r_{j1}, r_{j2}, ..., r_{jl} \in r_j) \) where \( 0 < j \leq l \). For some features \( r_{jl} \), there exists a possible set of ambiguous notation, \( a_1, a_2, ..., a_N \in A \) where \( 0 < k \leq N \) and there are \( N \) such ambiguous notations. From these ambiguous notations, a single ambiguous notation is randomly selected with probability \( \frac{1}{|A|} \) where \( |A| \) represents the total possible ambiguous notations for that feature. Given an ambiguous notation, \( a_k \), a set of values \( V \) is selected dependent on the notation such that \( V \subseteq F_i \) and \( V \) utilizes the notation defined in \( a_k \).

For example, assume that \( r_{jl} \) examines the destination port in the TCP header of the network traffic. Assume there are three possible ambiguous notations: a list \([a,b,c]\), a range \([a,c]\), or ‘*’ for any possible value. A singe ambiguous notation is
selected from the set of possible notations with equal probability. Given the specific notation selected, the set \( V \) is then determined dependent on the particular notation. For example, for \( ^* \) \( V \) would be equal to the range \([0:65535]\), a list would contain a number of randomly selected unique values (less than the total possible for \( F_i \)), and a range would randomly define a \( \text{start} \) and \( \text{end} \) such that \( 0 \leq \text{start} < \text{end} < |F_i| \).

Under these definitions, a simple rule is a rule that examines only finite elements within the evaluated feature set, while a complex rule uses ranges or other dynamic approaches to evaluating a particular traffic feature. Algorithmic-based detectors do not have specific rules, though they still examine traffic features in quantifiable units of traffic. As such, the idea of simple and complex rule-sets are mostly irrelevant for such detectors. Regardless, similar to traffic, these extremes of inputs offer a broader perspective to quantifying the results from private or public rule-sets.

IV. Design and Implementation

To meet the definitions of traffic and rules outlined in Section III we designed two tools: Sniffles (for traffic generation) and the Rule Generator for generating random rules. These tools are available for download [10], [11] and use in evaluation of NIDS or other network devices. We use them frequently in the evaluation of our own products. These tools are designed to handle common cases for testing and are extensible so that new cases can easily be added.

A. Sniffles Traffic Generator

A very basic diagram of Sniffles can be found in Figure 4. Essentially, Sniffles takes as an input a set of rules used by a NIDS. This set of rules is parsed into a set of traffic stream objects that describe the potential traffic that might match to a particular rule. The traffic stream objects are wrapped in a conversation that may include one or many such traffic streams. The primary benefit of the conversation is that it allows multiple communications synchronously, or asynchronously, between the same two endpoints. Regardless, once the set of rules has been parsed into the set of conversations the Traffic Generator takes over. The traffic generator randomly selects a conversation and generates an actual instance of the possible traffic defined by the traffic streams within a conversation. In this manner, network traffic is generated such that it will ultimately match rules defined in the rule-set, or completely random traffic if the traffic stream defines random traffic.

Currently, Sniffles supports many of the rule features of Snort rules as well as a rule format specifically designed for Sniffles. New rule formats can be added by extending the parser and traffic stream objects. Sniffles has two major traffic stream types: normal traffic and scan traffic. Traffic content is generated using methods outlined in previous work [2], [3] if a regular expression or fixed string is defined for a rule. Otherwise, content is generated such that each byte is randomly chosen. Sniffles ultimately generates a pcap file that can be used as a direct input to most NIDS.

B. Random Rule Generator

The Random Rule Generator takes a set of rule feature definitions and then uses those definitions to create rule-sets that meet the requirements as illustrated in Figure 5. Given one set of \( n \) such features and a specified number, \( l \), then \( l \) rules can be generated each with \( n \) features. Each feature defines the boundaries for that rule and any ambiguous notation while the feature parser defines the file output format of the rule. When simple rules are generated, a rule is generated such that a fixed value for any feature is generated with the expectation of any specific value as \( \frac{1}{\text{range of feature}} \). When complex rules are generated, that feature is generated such that the rule will select from a list of ambiguous notations (if they exits for that rule) and use one such notation for the rule. If a particular feature has no ambiguous notation then a value is chosen as per simple rule generation. The Rule Generator also includes a random regular expression generator that can build a regular expression dependent on a variety of distributions. The set of rules generated can then be used as an input to Sniffles assuming that a parser has been generated for that specific rule format.

V. Case Studies

We employ three example Case Studies to illustrate the use of the models from Section III and the tools from Section IV. These Case Studies illustrate the potential variance among different traffic and rule-sets as well as how looking at the extremes of input can help provide better context for any results. The first case study evaluates the deep packet inspection engine of a NIDS. The second case study examines the performance of header processing for a NIDS and the final case study looks at scan detection.

A. Deep-packet Inspection

The first case study illustrates the use of the philosophy outlined in this work to examine the performance of a Deep-packet Inspection (DPI) engine called gpp-grep [12] given a variety of inputs. For this evaluation we developed a set of features for what we term simple and complex features respectively. Gpp-grep is primarily a regular expression matching engine, thus the rules used in this evaluation are all
Fig. 6. Matching Throughput with the Snort and Simple rule-sets. Error bars mark first standard deviation from 10 test runs.

Fig. 7. Matching Throughput with the Complex rule-set.

Fig. 8. Diversity of data byte values.

TABLE I

<table>
<thead>
<tr>
<th>RE Set</th>
<th>Total RE</th>
<th>Size (MB)</th>
<th>Build-time (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>simple</td>
<td>250</td>
<td>13.1</td>
<td>10.6</td>
</tr>
<tr>
<td>complex</td>
<td>250</td>
<td>176.7</td>
<td>1692.3</td>
</tr>
<tr>
<td>snort</td>
<td>250</td>
<td>71.4</td>
<td>46.1</td>
</tr>
</tbody>
</table>

regular expressions. Simple regular expressions are comprised of regular expressions where each rule is a fixed string (i.e. there is no ambiguous notation within the regular expressions). Complex regular expression are created making extensive use of ambiguous notation. In particular, the Rule Generator will develop a regular expression based on a particular set of distributions where the distribution probabilities help to determine how likely ambiguity is used for any point in a regular expression. The current distribution is built into the random rule generator for complex rules and creates regular expressions approximating the dotstar of 0.5 (as per Becchi et al. [2]) as well as using most of the other regular expression ambiguous notation such as character types (i.e. \d for [0-9]), character classes (i.e. [abc]), alternation (i.e. (a|b|c)), repetition (i.e. a*b+c?), and counting (i.e. ab{1,5}c). We created rule-sets of 250 regular expressions for each type: simple and complex.

We also randomly selected 250 regular expressions from the Sourcefire Vulnerability Research Team [13] Snort rule-set for January 22, 2015. The number 250 was chosen to be large enough to grant a good idea of performance but not requiring too much time in compilation of rules. For this Snort rule-set 250 is slightly more than \( \frac{1}{4} \) of the total unique regular expressions.

We used Sniffles to generate packet captures each with 100,000 packets and 79 bytes of data per packet for each rule-set. The number of bytes of data was was derived from the average data size for the non-synthetic packet captures described later. One packet capture was created for Random traffic, where the value for any byte is universally distributed, one packet capture was Homogeneous where the data is a lower-case ‘x’ for all bytes for all packets, and the final packet captures matched to a given rule-set (Simple, Complex, or Snort).

In addition to the synthetic traffic-sets, we also used three packet captures pulled from live networks. One packet capture represents traffic gathered from a computer war game event held by the Computer Security Group (CSG) at Washington State University in the spring of 2009 [14]. This packet capture is not publicly available. The second packet capture was selected from the army side of the 2009 Inter-Service Academy Cyber Defense Competition [15] (ISACDC) put on by the Information Technology Operations Center (ITOC) of the US Army. The final packet capture was the first packet capture from the 2012 National CyberWatch Mid-Atlantic Collegiate Cyber Defense Competition [16] (MACCDC).

Figure 6 and Figure 7 illustrate the performance under the given traffic-sets and rule-sets. Table I illustrates the build-time and compiled size of each rule-set. As expected, more complex rule-sets create larger matching automata and reduce throughput. In fact, the difference in throughput between the Simple rule-set and the Snort rule-set is nearly an order of 10, which is also roughly the difference in throughput between the Snort rule-set and the Complex rule-set. Next, we note that Random and Matching traffic illustrate the expected trend where the matching traffic demonstrates significantly
lower throughput over Random traffic for the Simple and Snort rule-sets. However, for the Complex rule-set the two demonstrate nearly identical throughput. The complex rules have created many cycles in the matching automata such that any sufficiently diverse input is likely to match at nearly the same speed. Finally, we note the impact of homogeneity on the traffic-sets. For the Simple rule-set the Homogeneous traffic demonstrates the best throughput, while it demonstrates nearly the worst for Snort and the worst for the Complex rule-set. This illustrates the bi-polar nature of Homogeneous data such that if the homogeneous data does not intersect with the target rules, then it will likely cause best-case performance while it will cause worst-case, or near worst-case performance when it intersects with process intensive regions of the rule-sets.

In fact, the performance of the CSG, ISASC and MACCDC traffic is also due partially to this reduced diversity in data. Figure 8 illustrates the distribution of data in the packets for each of the traffic-sets. In these captures more than 40% of the data has a byte value of less than 32 or greater than 128 (i.e. targeting non-printable ASCII values). The rules, on the other hand, target primarily printable ASCII values (byte values of 32-128). As such, less than 60% of each of these traffic-sets actually intersects with the rule-set. This makes processing easier and illustrates the relatively close average performance between these sets.

B. Packet Classification

For the second case study we examine the packet classification of the open source NIDS Snort [17]. The goal of this test is to get an idea of how Snort’s packet header processing handles various inputs. For this test, we created a feature set for a simple rule-set, complex rule-set, and a homogeneous rule-set. The feature set included the network flow quintuple (protocol, source address, source port, destination address, and destination port) used for classifying the header of a packet. We used these feature sets and created 5892 rules for each rule-set. We also took all of the default enabled rules for the Snort rule-set for January 22, 2015 [13]. The number 5892 happens to be the number of default enabled rules in the Snort rule-set. We then created packet captures of 2 million packets each, one for random traffic, and one to match to each of the rule-sets (Homogeneous, Simple, Snort and Complex). We used these feature sets and created 5892 rules for each rule-set. We also took all of the default enabled rules for the Snort rule-set for January 22, 2015 [13]. The number 5892 happens to be the number of default enabled rules in the Snort rule-set. We then created packet captures of 2 million packets each, one for random traffic, and one to match to each of the rule-sets (Homogeneous, Simple, Snort and Complex).

First, we note the differences in the size of the structures required by Snort to accommodate these rule-sets in Table II. These results follow expectations and mirror those from Case Study 1 except that the Snort rule-set requires less space than the Simple rule-set. This stems from the fact that all rules in the Snort rule-set use the script values for internal vs external address spaces, which are not set in our environment. Thus, the IP addresses for these rules are converted to ‘any’ meaning that any IP address will do. Given that, there is really no need to store any IP address information. Also similar to Case Study 1 the rule-set employed greatly affects the overall performance regardless the traffic. Also, similar to Case Study 1 we see that the non-synthetic traffic-sets have quite similar performance, and that performance is nearer to best-case performance rather than worst-case. This comes from the lack of diversity in those files and the fact that the traffic that does exist intersects with shallower regions of the rule-set. In general, the more diverse
in IP addresses (Table III) and in port values (Figure 10) the more difficult the matching with the notable exception being Random traffic for the Snort rules. In that case, the Snort rule-set looks at a smaller set of ports, but the Random traffic examines a diverse set of ports meaning that less traffic intersects with the rules which greatly improves throughput. Similarly, the Homogeneous traffic intersects with a single region in the rule-set and is processed at near maximum throughput in all cases. Testing in this manner provides a clearer view of both how the NIDS is impacted by traffic and by rule-sets.

C. Scan Detector

As a final case study, we chose to examine the scan detection analyzer for the bro [18] NIDS. Bro is a NIDS capable of a wide variety of traffic analysis and detection. For the purpose of this case study, we examine the scan detector policy included with the default installation of bro. This policy does not require a rule-set. Thus, there is no need to concern ourselves with creating simple or complex sets of rules. The detector targets Destination IP addresses and ports over time to identify scans. To test this we created one traffic capture with 10,000 packets each for Random, Homogeneous, and Matching traffic (where every packet is part of a scan) as well as using 10,000 packets from a publicly available packet capture (termed Honeynet) from Azusa Pacific University’s Honeynet project [19]. The results are displayed in Figure 11.

Diversity among the IP addresses and ports once again dominates as the primary factor in performance. The lack of diversity in the Homogeneous and Honeynet traffic captures led to higher throughput since most of the traffic was not involved in any scans and easily disregarded. The Matching traffic lacked diversity but the fact that every packet was part of a scan attack required tracking of all packets and led to greater effort. The diversity in the Random traffic increased strain on the detector as a wider array of port and IP address combinations needed tracking in order to detect scans. This occurred despite the absence of scans in the traffic. Ultimately, the scan detector performance is not only sensitive to scans in the traffic, but to the number of distinct IP address and port combinations as well.

VI. Conclusion

In this paper we showcase methodologies and tools for creating a larger perspective when evaluating NIDS. The goal of this research is to better account for variance when evaluating NIDS and when comparing results between research or working products. These tools are provided to help foster such evaluation, but are by no means the only possible solution. We look forward to an evolutionary process in providing better, consistent comparison methodologies for Network Intrusion Detection Systems.

REFERENCES