NIST Technical Note 2142

An Evaluation Design for Comparing Netflow Based Network Anomaly Detection Systems Using Synthetic Malicious Traffic

Shuvo Bardhan
Mitsuhiro Hatada
James Filliben
Douglas Montgomery
Alexander Jia

This publication is available free of charge from:
https://doi.org/10.6028/NIST.TN.2142

National Institute of
Standards and Technology
U.S. Department of Commerce

NIST Technical Note 2142

An Evaluation Design for Comparing Netflow Based Network Anomaly Detection Systems Using Synthetic Malicious Traffic

Shuvo Bardhan
Douglas Montgomery
Alexander Jia
Advanced Networks Technology Division
Information Technology Laboratory

Mitsuhiro Hatada
Computer Security Division
Information Technology Laboratory

James Filliben
Statistical Engineering Division
Information Technology Laboratory

This publication is available free of charge from:
https://doi.org/10.6028/NIST.TN.2142

March 2021

U.S. Department of Commerce
Wynn Coggins, Acting Secretary

National Institute of Standards and Technology
James K. Olthoff, Performing the Non-Exclusive Functions and Duties of the Under Secretary of Commerce for Standards and Technology & Director, National Institute of Standards and Technology

National Institute of Standards and Technology Technical Note 2142
Natl. Inst. Stand. Technol. Tech. Note 2142, 27 pages (March 2021)
CODEN: NTNOEF

This publication is available free of charge from:
https://doi.org/10.6028/NIST.TN.2142

Abstract

In this paper, we present a procedure to evaluate and compare multiple netflow based network anomaly detection (NF-NAD) systems based on accuracy of detection and mean time of detection. Conventionally, different variations of benign or normal traffic have been used to evaluate NF-NAD systems. Here we showcase a methodology where benign traffic is constant through the entirety of the experiment. We create different variations of synthetic malicious traffic to evaluate and compare NF-NAD systems. A two-phase approach is used to measure the accuracy and learning capability of the NF-NAD system. We have created a designed experiment (having factors, levels and design points) to showcase our methodology.

Key words

Network Anomaly Detection Systems, Experiment Design, Netflows and Computer Worms.

Table of Contents

List of Tables

List of Figures

1. Introduction

An Intrusion Detection System (*IDS*) is a software or hardware system that monitors network traffic for malicious activity and signs of system compromise. An *IDS* can be of two types based on its detection methods:

1. Behavior Based (*Anomaly Based Detection*)–detection based on identifying deviation from normal behavior of network traffic [1]; and
2. Knowledge Based (*Signature Based Detection*)–detection based on identifying traffic patterns of known attacks (i.e., attack signatures) [2].

With networks growing larger and larger everyday, the prominence of netflow-based `IDS`'s have grown significantly [3]. Our work focuses on netflow-level anomaly based `IDS's` placed between the internal network and the internet–which is the most important IDS as it acts as prime line of defense (shown in figure 1). There have been significant contributions in this area; however, researchers find it hard to get datasets to evaluate on such systems. Most datasets cannot be shared due to privacy issues and the ones that are available are old and do not represent current attack trends [4].

In this paper, we propose a novel methodology to compare netflow-based network anomaly detection systems using synthetically generated malicious traffic modeling a propagating computer worm [5]. The novelty of our work lies in the fact that we have used malicious traffic as the focus instead of normal traffic to evaluate `NF-NAD` systems [6]. For this methodology, we have restricted the malicious traffic to common attacks–distributed denial of service attacks (DDoS) [7] and scanning attacks.

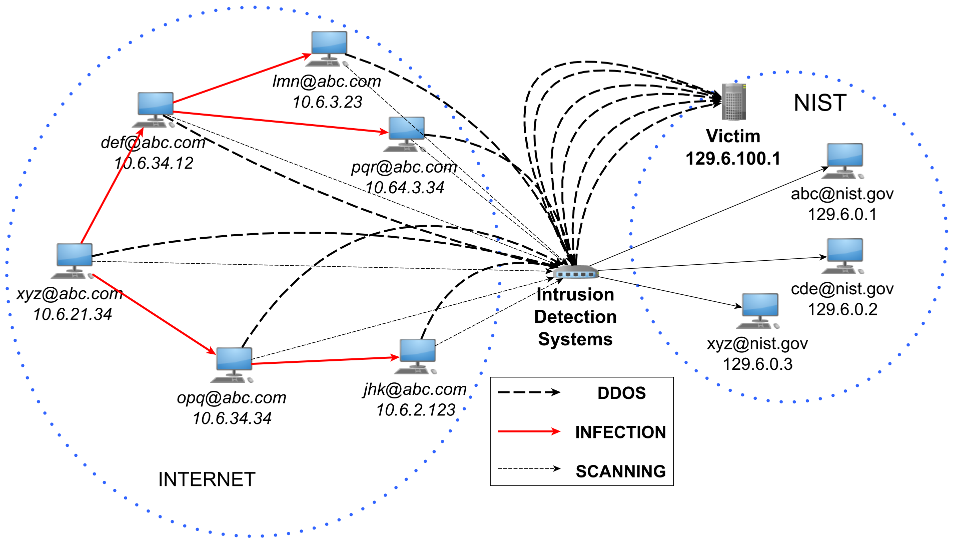

Fig. 1. Computer Worm–propagation and attacks.

The rest of the paper is divided into several sections. The next section is *Literature Survey*, where we provide the motivation behind our work through a brief description of the related work and the datasets that are currently in existence. The following section is *Experiment Setup*. Here we describe how we create the benign and the synthetic malicious traffic that can be used to evaluate and compare multiple `NF-NAD` systems. We then move on to explain the experiment design–factors and levels that we have chosen to evaluate and compare multiple `NF-NAD` systems. The next section is *Evaluation Metric for Comparing*, where we describe our comparing methodology. We have used self created values to illustrate our methodology in this section. We then test our datasets through our evaluation methodology on a `NF-NAD` system and discuss the results in the *Case Study* section. Lastly, we conclude our paper in the *Conclusion* section.

2. Literature Survey

In this section, we review previous work done in the field. We do so by initially discussing the main public datasets that have been used to evaluate network anomaly detection (`NAD`) systems. We then move onto discussing procedural issues and metrics used to compare such systems.

2.1 Anomaly-Based IDS Datasets

In 2018, Sharafaldin et al. did an extensive survey of 276 studies on anomaly-based intrusion detection systems [4]. The survey mainly focused on two aspects:

1. *Reliability*–consistency of measurements; and
2. *Validity*–accuracy and quality of measurements.

Some of the most popular datasets used to evaluate Anomaly-based IDS's were described and are as follows:

1. `DARPA'99` (*Lincoln Laboratory MIT, 1998-99*)–the dataset has email, browsing, FTP, Telnet, IRC, and SNMP traffic. The attacks include DoS, Guess password, Buffer overflow, remote FTP, Syn flood, Nmap, and Rootkit. The dataset is outdated and it does not represent the current trends [8].
2. `KDD'99` (*Univ. of California Irvine, 1999*)–the dataset is an improved version of `DARPA`. The dataset contains Neptune-DoS, pod-DoS, SmurfDoS and buffer-overflow attacks. The benign and attack traffic are combined via a simulated environment. The dataset contains redundant data which leads to skewed results [9].
3. `DEFCON` (*The Shmoo Group, 2000-2002*)–the DEFCON-8 dataset (2000) has port scanning and buffer overflow attacks. DEFCON-10 dataset (2002) has port scan and sweeps, bad packets, administrative privilege and FTP by Telnet protocol attacks. This dataset is mainly used for evaluation of alert correlation techniques.

4. CDX (*United States Military Academy, 2009*)–the dataset has traffic such as Web, email, DNS lookups and other required services. The attack traffic is generated by using attack tools such as Nikto, Nessus and WebScarab. This dataset is mainly used to check IDS alert rules but it lacks diversity.
5. ISCX2012 (*University of New Brunswick, 2012*)–this dataset contains–(i) Alpha-profile, which represents various multi-stage attacks and (ii) Beta-profile, which is the benign traffic generator. The dataset has network traffic for HTTP, SMTP, SSH, IMAP, POP3 and FTP protocols.
6. ADFA (*University of New South Wales, 2013*)–this dataset contains FTP and SSH password brute force, Java based meterpreter, Add new Superuser, Linux meterpreter payload and C100 Webshel attacks. The normal traffic and the attack traffic are combined here.

Sharafaldin et al. found 194 out of 276 ($\approx 70\%$) studies used public datasets (e.g., KDD, DARPA, etc.) and among them 50% of the studies used KDD and DARPA which were 19 years old. Most of the datasets apart from KDD and DARPA were also pretty old and all of them did not represent the current trends. There was no standard dataset that could be used for evaluation of NAD systems. The overall summary was that due to lack of publicly available robust datasets, the majority of research in the future would depended upon creation of synthetic datasets.

2.2 Anomaly-Based IDS Evaluation

Here we will be discussing the procedural issues and the evaluation metrics (mentioned in [4]), which were used to evaluate Anomaly-based IDS's. In [4], they found most published research in literature have used training sets having more than 80% malicious traffic in them and this becomes a serious problem. If the training sets have too much abnormal activity in it, even a properly functioning NAD system maybe unable to detect anomalies. This is because the NAD system considers the abnormal activities in the training set as normal traffic and does not raise any alarms for the same or similar abnormal activities found in the testing set.

The effectiveness of an anomaly-based detection mechanism is evaluated by–(a) ability to distinguish between normal vs abnormal behavior; and (b) time: (i) required to train the model; and (ii) detection delay. In literature, the most commonly used metrics to evaluate are as follows:

1. Detection Rate *(DR)*–ratio of the number of correctly identified attacks to the total number of attacks; and
2. False Positive Rate *(FPR)*–ratio of the number of events classified as abnormal to the total number of abnormal events.

2.3 Motivation

The motivation behind our work is–(a) lack of standard training datasets having extremely low malicious activities; and (b) non-existence of a standard methodology to evaluate an anomaly-based IDS.

3. Experiment Setup

In this section, we will show how we have created datasets that can be used for the evaluation of NF-NAD systems. We will then proceed to the evaluation methodology itself. The datasets created represent the traffic (netflow) at an observable point between the Enterprise 1[1] internal network and the internet (as shown in figure 1). The datasets created are broadly divided into the following:

1. Benign Traffic (*normal traffic*); and
2. Malicious Traffic (*attack traffic*).

3.1 Benign Traffic

The benign traffic comes from Enterprise 1 packet traces converted into netflows using Silk[10]. Post conversion, a thorough cleansing of the dataset was made by using Netflow Analyzer–a signature based detection system[11] to remove as much abnormal or malicious traffic as possible [12]. The benign traffic used was constant through the entirety of the experiment.

3.2 Malicious Traffic

The malicious traffic was created using [12]. This is the code which generates the synthetic malicious traffic in netflows. The malicious traffic contains attack traffic culminating from infected hosts due to a propagating random-scanning computer worm (as shown in figure 1). The attacks that we have modelled are of 4 types–

1. Distributed Denial of Service Attack.
2. Horizontal Scan Attack.
3. Vertical Scan Attack.
4. Fin Scan Attack.

Our synthetic attack traffic generator used the scanning worm propagation model that we had previously developed [5] (shown in Algorithm 1 using Table 1). Our synthetic attack traffic generator can scale the attack traffic as desired by setting Population Size, Scanning Rate and Susceptible Proportion (discussed in detail in section 3.3.1). These are parameters of the attack traffic. The duration of each malicious dataset is also configurable.

[1] A medium sized enterprise network (5K users, 30K networked devices).

Table 1. Algorithm 1–Lists, Variables and Functions

Type	**Name**	**Description**
`Lists`	`List_IP`	IP Address Space.
	`List_SP`	List of Susceptible IP Addresses.
	`List_IIP`	List of Infected IP Addresses.
	`Rand_IP`	List of Random IP Addresses.
	`List_INF`	List of Newly Infected Hosts (IP Addresses) in each iteration.
Variables	N	Number of hosts in a network.
	p	Proportion of hosts susceptible to the computer worm.
	r	Scan Rate of the worm.
	n	Number of Initial Infected Hosts.
	d	Death Rate of the worm.
	P	Patching Rate of the worm.
	i	Number of newly infected hosts (per iteration).
Functions	Random (`L`, n)	Function which returns n random IP addresses from a list of IP Addresses (`L`) in the form of a `List`.
	Time ( )	Function which returns the current timestamp.

Algorithm 1 Scanning Worm Propagation Model

Input: List_IP, N, p, r, n, d, P
Output: List_INF

```
 1: Rand_IP ← $\phi$
 2: List_INF ← $\phi$
 3: $i \leftarrow n$
 4: List_SP = Random ( List_IP , $|\,(p \times N)\,|$ )
 5: while $i \; < \; |\,(p \times N)\,|$ do
 6:     $k \leftarrow 0$
 7:     while $k < (i$ - $(d + P) \times i)$ do
 8:         Rand_IP $\uplus$ { Random ( List_IP, $p$ ) }
 9:         $k \leftarrow k + 1$
10:     for $ip_i$ in Random_IP do
11:         for $ip_j$ in List_SP do
12:             if $ip_i == ip_j$ and $ip_i \notin$ List_IIP then
13:                 List_IIP $\uplus$ { $ip_i$, Time()}
14:                 $i \leftarrow i + 1$
15:     List_INF $\uplus$ { $i$ }
16:     Rand_IP ← $\phi$
17: return List_INF
```

The scanning worm propagation model (Algorithm 1) generates the list of newly infected hosts per iteration of a scanning worm till saturation. Each newly infected host is assigned a unique random IP address. In this model, the newly set of infected hosts at each iteration launches an attack–DDoS or scanning attack and then at the next iteration, the next set of newly infected hosts launches an attack, while the previous set of infected hosts do not. Initially, the attack intensity increases with each iteration, as it moves towards saturation, it flattens out [13].

3.3 Experiment Design

As mentioned before, we evaluate a `NF-NAD` system through a designed experiment [14]. To illustrate we have taken the simplest experiment design comprising of $k = 5$ factors and $l = 4$, 2 and 1 levels (Table 2). A factor of an experiment is a controlled independent variable; a variable whose levels are set by the experimenter. This makes the number of runs $n = 4 \times 2 \times 2 \times 2 \times 1 = 32$ (i.e., all possible combinations of the levels in Table 2) .

Table 2. Experiment Design ($k = 5,\ l = 4$, 2 and 1).

Factors	**Levels**	**Design Points**
X1 Attacks	*Type 1*	DDoS
	Type 2	Vertical Scan
	Type 3	Horizontal Scan
	Type 4	Fin Scan
X2 Population Size	*Low*	64000 hosts
	High	128000 hosts
X3 Scanning Rate	*Low*	10 scans per second
	High	50 scans per second
X4 Susceptible Size	*Low*	0.25 of $X2$ hosts
	High	0.75 of $X2$ hosts
X5 Benign Traffic	*Static*	1 hour

3.3.1 Factors and Levels

Factor $X1$ represents the attack types emanating from the hosts infected by a propagating computer worm. Malicious traffic coming from a computer worm was chosen because they are one of the most dangerous threats network administrators face today. DDoS and scanning attacks are the most common attacks seen in them. The other factors $X2$ Population Size, $X3$ Scanning Rate and $X4$ Susceptible Size represent the factors affecting the worm propagation. Values for these factors are taken from [5]. They produce different variations of computer worms (as shown in Figure 2). For our experiment design, we kept the factor $X5$ Benign Traffic constant throughout the entirety of the experiment. We do so because we have a two-phase evaluation approach that obviates the need of different variations of normal traffic to evaluate.

For each design point in factor $X1$ (for e.g., DDoS), there are 8 ($= 2 \times 2 \times 2$) malicious datasets created (as shown in Table 3 and Figure 2). Since there are 4 attacks, the number of synthetic malicious datasets created are 32 ($= 4 \times 8$). Figure 2 is the graphical representation of runs 1-8 (shown in Table 3). Each iteration (on X-axis) represents 5 minutes on the dataset.

Table 3. Experiment Design Matrix–DDoS.

Run ID i	**Infection Behavior** I	**Pop. Size** N	**Susc. Prop.** p	**Scan. Rate** r
1	DDoS	64000	0.25	10
2	DDoS	128000	0.25	10
3	DDoS	64000	0.75	10
4	DDoS	128000	0.75	10
5	DDoS	64000	0.25	50
6	DDoS	128000	0.25	50
7	DDoS	64000	0.75	50
8	DDoS	128000	0.75	50

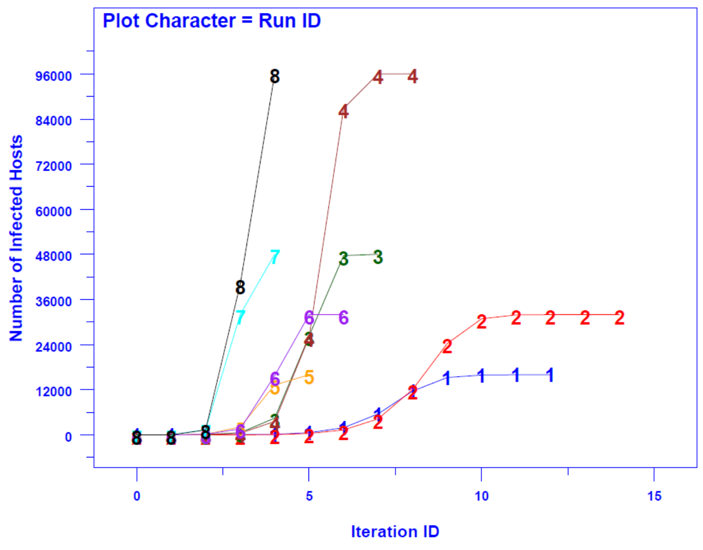

Fig. 2. Graphical representation each run in Table 3.

The objective of our approach was to create different variations of the same attack (using Algorithm 1) and test if the NF-NAD system can detect each one of them. From Figure 2, each run of the DDoS attack (to be detected by the NF-NAD) can be visualized. We designed the experiments in such a way that the difficulty of detection varies with each run (shown in Table 3). We can rank the difficult of detection by measuring the slope of the curve, whichever is lower is ranked higher. The notion here is that Run 4 is comparatively more difficult to be detected (by the NF-NAD) than Run 8 (because of the difference in slope). The same approach is used for the remaining attack types in $X1$–Vertical Scan, Horizontal Scan and Fin Scan.

3.3.2 Responses

As mentioned before, we have a two-phase evaluation approach to evaluate a NF-NAD systems–*Phase I* and *Phase II*. *Phase I* is specifically designed to see how well the NF-NAD system can classify each attack, while *Phase II* is designed to see whether the NF-NAD system has actually learnt and classified the attack correctly. In *Phase I*, we attain response $Y1$ and in *Phase II*, we attain response $Y2$ (as shown in figure 3):

1. $Y1$ (in *Phase I*)–number of alarms raised when training set is benign traffic and testing set is benign traffic + malicious traffic; and
2. $Y2$ (in *Phase II*)–number of alarms raised when training set is benign traffic + malicious traffic and testing set is also benign traffic + malicious traffic.

For each malicious dataset, there is *Phase I* and possibly a *Phase II* testing. In *Phase I*, we place the benign traffic in the training set and benign traffic + malicious traffic in the testing set. This is to see if the NF-NAD system can detect the abnormality and classify the attack in the malicious dataset correctly. If it succeeds in classifying the attack on the particular malicious dataset, we then go onto *Phase II*.

In *Phase II*, we place the benign traffic + malicious traffic in the training set and label them as benign traffic. We keep the testing set the same–benign traffic + malicious traffic. We test it again on the NF-NAD system and since this is a learning based system, the ideal outcome should be $Y2 = 0$ (as it should not raise any alarm). This is true, because the

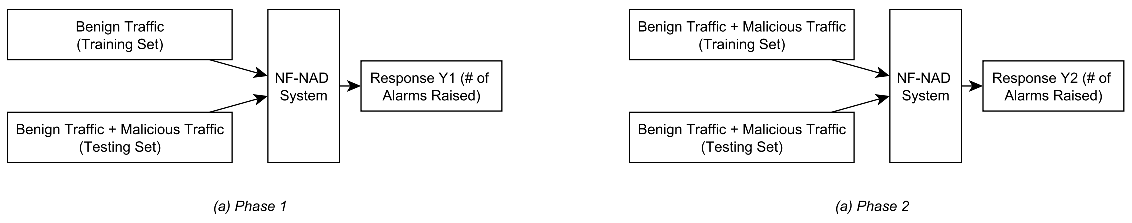
Benign Traffic (Training Set)
Benign Traffic + Malicious Traffic (Testing Set)
NF-NAD System
Response Y1 (# of Alarms Raised)
(a) Phase 1
Benign Traffic + Malicious Traffic (Training Set)
Benign Traffic + Malicious Traffic (Testing Set)
NF-NAD System
Response Y2 (# of Alarms Raised)
(a) Phase 2

Fig. 3. Responses of the two-phase approach for the evaluation of NF-NAD systems.

testing set–(benign traffic + malicious traffic, although labeled as benign traffic) is the same as training set–(benign traffic + malicious traffic) and the `NF-NAD` system should consider it as normal and not raise any alarm. This two-phase approach for evaluation holds true for any `NAD` system.

In reality, each `NF-NAD` system raises a number of alarms for a particular attack. For testing purposes, we set threshold values for *Phase I* and *Phase II* – T_{Y1} and T_{Y2} respectively (for each `NF-NAD` system). These values are taken from empirical evidence. If $Y1 > T_{Y1}$ and $Y2 < T_{Y2}$, then the `NF-NAD` system has identified the attack correctly, otherwise it has not identified it correctly.

4. Evaluation Metric for Comparing

For each attack in $X1$, we provide a methodology to compare `NF-NAD` systems based on–(i) accuracy of detection; and (ii) mean detection time.

4.1 Accuracy of Detection

Initially, we generate the confusion matrix [15] (for each attack in $X1$) from the following responses:

1. Correct Response (*CR*)–correct alarms.
2. `NF-NAD` Response (*NR*)–alarms generated by `NF-NAD`.

For each malicious dataset, if the `NF-NAD` system raises an alarm and identifies the attack correctly during *Phase I* and *Phase II*, the `True Positives` (*TP*) value is incremented by 1. If the `NF-NAD` system raises an alarm for a different attack (thereby not identifying it correctly), `False Positives` (*FP*) value is incremented by 1. If the `NF-NAD` system does not raise any alarm, but it should have, `False Negatives` (*FN*) value is incremented by 1. If the `NF-NAD` system does not raise any alarm and it should not have raised any alarm at all, then the `True Negatives` (*TN*) value is incremented by 1. In order to generate `True Negatives`, we need to have one test, which we call Phase 0 (as shown in figure 4) where the training set and the testing is the same–benign traffic (with no malicious traffic).

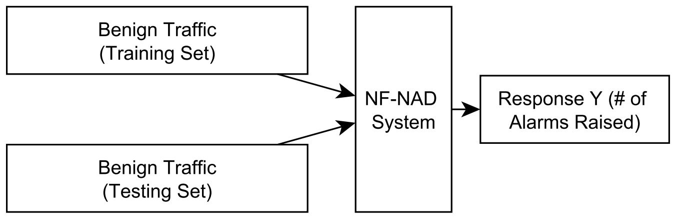

Fig. 4. Phase 0 (to calculate True Negative value)

We can calculate the Detection Rate (or Accuracy) and False Postive Rate from the confusion matrix. Figure 5 is an example confusion matrix for DDoS attack. For each attack in $X1$–(i) DDoS, (ii) Vertical Scan, (ii) Horizontal Scan and (iv) Fin Scan, a separate confusion matrix needs to be created. We compare the `NLFNAD` systems based on the detection rate, whichever is higher is ranked higher. If there are two `NF-NAD` systems having the same detection rate, we resolve this issue on the basis of the false positive rates both `NFLAND` systems have, whichever has lower is ranked higher. While ranking, it is possible to have two `NF-NAD` systems having the same confusion matrix (for an attack in $X1$). In order to resolve this issue, we generate the Receiver Operating Characteristic (*ROC*) [15] curves under the assumption that it would be extremely rare to have two `NF-NAD` systems generating the same *ROC* curve for the same attack. In order to do so, we vary the threshold values-T_{Y1} for the attack to generate the *ROC* curves. We calculate the area under the *ROC* curves (*AUC*) and whichever is higher, is more accurate and ranked higher.

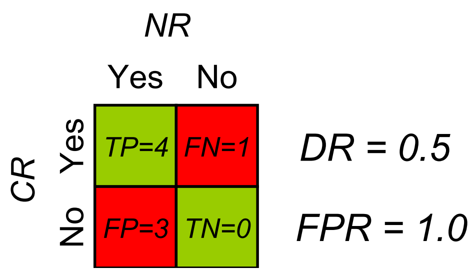

Fig. 5. Confusion Matrix–DDoS (Example)

For illustration (as to how to calculate the *AUC*), we have used the threshold values $T_{Y1} = \{10, 20, 30\}$ alarms for (say) DDoS to generate the *ROC* curve (as shown in Figure 6) from Table 4. We have used a different set of self-created values of `True Positive Rates` and `False Positive Rates` for illustration. The *AUC* turns out to be 0.859–which is also an estimate of the accuracy of detecting DDoS by the `NF-NAD` system. In general, $AUC = 1.0$ is the best case and $AUC = 0.5$ is the worst case (also shown in Figure 6). This approach helps us comparing based on accuracy.

Table 4. *ROC* curve–DDoS (Example)

Srl. No.	Threshold (# of Alarms)	True Positive Rate	False Positive Rate
1.	10	0.90	0.60
2.	20	0.78	0.19
3.	30	0.56	0.01

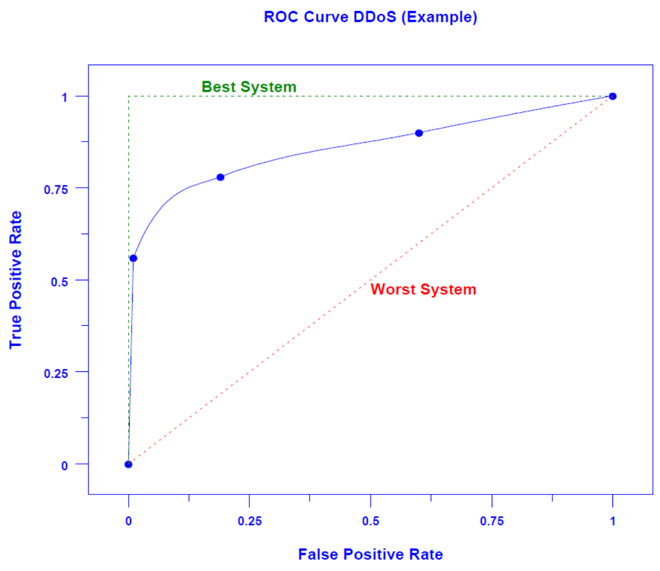

Fig. 6. ROC Curve–DDoS (Example) from Table 4

4.2 Mean Detection Time

Secondly, we compare NF-NAD systems based on the mean detection time for each attack in $X1$. We clock the detection time (t_i) taken by the NF-NAD system to detect each malicious dataset of an attack in $X1$ (say DDoS) and calculate the mean. For illustration, we have used self-created detection times in seconds (t_i) for DDoS and the mean detection time calculated is 17.5 secs (as shown in Table 5).

Table 5. Mean Detection Time–DDoS (Example).

Run ID i	**Run Details–DDoS** N	p	r	**Slope**	**Rank**	**Detection Time** t_i
1	64000	0.25	10	6128	1	30
2	128000	0.25	10	12133	3	25
3	64000	0.75	10	21855	5	27
4	128000	0.75	10	55916	7	15
5	64000	0.25	50	11213	2	19
6	128000	0.25	50	16207	4	22
7	64000	0.75	50	30927	6	10
8	128000	0.75	50	60791	8	5

$$\texttt{Mean Detection Time (DDoS)} = \left(\sum_{i=1}^{8} t_i\right)/8 = 17.5 \text{ secs}$$

While comparing NF-NAD systems based on mean detection time, it is possible for two NF-NAD systems having the same or very close mean detection times. Our approach allows us to resolve this issue as well. We do so by ranking the runs based on difficulty of detection by measuring the slope of the curve (as shown in Table 5). In this case, the mean detection time for DDoS (top 4) is 24 seconds (=(30+19+25+22)/4). If the values still remain the same or close (which is quite rare), we then need to calculate the mean detection time (DDoS) of the top 2 most difficult runs. In this case it is 24.5(=(30+19)/2) seconds. If this still does not resolve then we have to increase the levels in $X2$, $X3$ and $X4$. Using this approach, we can now compare based on mean detection time.

5. Case Study

Using our evaluation methodology, here we present the results of testing our dataset on a NF-NAD system, which uses k-nearest neighbor (k-NN). The k-NN is a simple and yet most efficient classification algorithm and is widely used in practice. Given training data is plotted on a vector space, and when unknown data is obtained, arbitrary k data are obtained from it in order of closest distance, and the class to which the data belongs is estimated by majority vote. We used various statistics of the flow data in a time window as features, such as the number of flows, the average number of packets, and the standard deviation of the number of bytes. The features are calculated in each 10-second time window by sliding in 5-second increments. We generated attack data by different simulation runs for both training and testing. We collected legitimate data for one hour on a certain day for training, and at the same time on the following day for testing. The experimental results are the average of the data from several simulations with different parameters. Training is performed with k=3, and the testing considers an anomaly when the average distance to k samples in the neighborhood is greater than the threshold Th = 0.1.

Table 6. Results of evaluating k-NN.

Attack Type	**Mean Detection Time (in secs.)**	**Accuracy (or Detection Rate)**
DDoS	0.033	0.774
Horizontal Scan	0.034	0.785
Vertical Scan	0.037	0.782
FIN Scan	0.035	0.829

The NF-NAD system has detected each attack in X1. There is no significant difference in Mean Time Detection, but the Accuracy of FIN Scan is higher than that of DDoS, Horizontal Scan, and Vertical Scan. We also have generated *ROC* curves and calculated the *AUC* for each attack in X1 (as shown in figure 7).

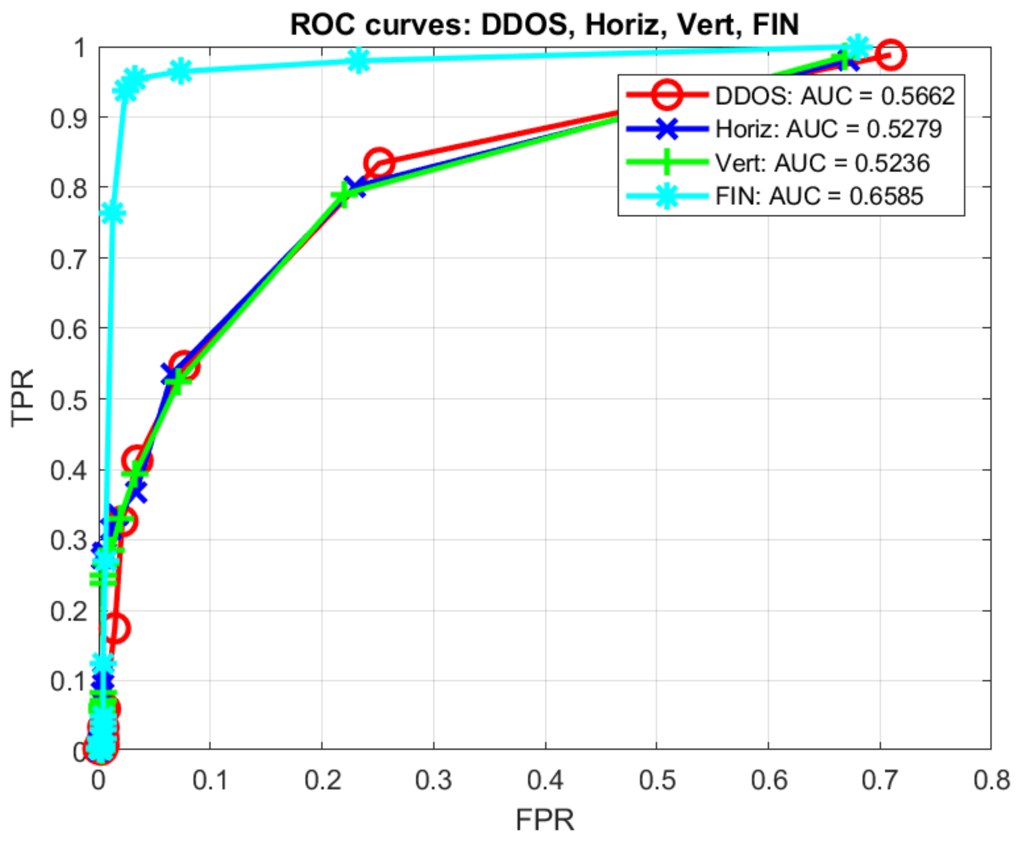

Fig. 7. *ROC* curves and calculated *AUC* for each attack in X1.

6. Conclusion

In this paper, we have presented a novel methodology to evaluate and compare multiple NF-NAD systems. We have initially shown how we have created the benign traffic and the synthetic malicious traffic (culminating from a propagating scanning worm). We later on showcased our methodology through an experiment design. We went onto explain in details as to why we chose these factors, levels and responses for this evaluation methodology. We illustrated our evaluation methodology through self created values. The evaluation methodology provides us an insight about the existing flaws of a NF-NAD system based on–(i) accuracy in detection ; and (ii) mean time of detection. We evaluated a NF-NAD system using our evaluation methodology and we presented the results. For more rigorous evaluation, one needs to increase the levels of the factors in the experiment design.

References

[1] Jyothsna V (2011) A review of anomaly based intrusion detection systems. *International Journal of Computer Applications* 28:26–35.

[2] Masdari M, Khezri H (2020) A survey and taxonomy of the fuzzy signature-based intrusion detection systems. *Applied Soft Computing* 92:106301. https://doi.org/https://doi.org/10.1016/j.asoc.2020.106301. Available at http://www.sciencedirect.com/science/article/pii/S1568494620302416

[3] Sperotto A, Schaffrath G, Sadre R, Morariu C, Pras A, Stiller B (2010) An overview of ip flow-based intrusion detection. *IEEE Communications Surveys Tutorials* 12(3):343–356. https://doi.org/10.1109/SURV.2010.032210.00054

[4] Sharafaldin I, Lashkari AH, Ghorbani AA (2018) Toward generating a new intrusion detection dataset and intrusion traffic characterization. *Proceedings of the 4th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,* INSTICC (SciTePress), , pp 108–116. https://doi.org/10.5220/0006639801080116

[5] Bardhan S, Montgomery D, Filliben J, Heckert A (2019) A general methodology for deriving network propagation models of computer worms. *Technical Note, NIST*, , . https://doi.org/https://doi.org/10.6028/NIST.TN.2035

[6] Miller N (2018) *Benchmarks for Evaluating Anomaly-Based Intrusion Detection Solutions* (ProQuest Dissertations Publishing, California State University, Long Beach), .

[7] Mirkovic J, Reiher P (2004) A taxonomy of ddos attack and ddos defense mechanisms. *SIGCOMM Comput Commun Rev* 34(2):39–53. https://doi.org/10.1145/997150.997156. Available at https://doi.org/10.1145/997150.997156

[8] Lippmann R, Haines JW, Fried DJ, Korba J, Das K (2000) Analysis and results of the 1999 darpa off-line intrusion detection evaluation. *Recent Advances in Intrusion Detection*, eds Debar H, Mé L, Wu SF (Springer Berlin Heidelberg, Berlin, Heidelberg), , pp 162–182.

[9] Kayacik HG, Zincir-Heywood AN, Heywood MI (2005) Selecting features for intrusion detection: A feature relevance analysis on kdd 99. *PST*, , .

[10] Silk-https://tools.netsa.cert.org/silk/docs.html. Available at https://tools.netsa.cert.org/silk/docs.html.

[11] Netflow-analyzer–flow-level signature based system detection system. Available at https://www.manageengine.com/products/netflow/.

[12] https://github.com/shuvobardhan/proj-nad. Available at https://github.com/shuvobardhan/Proj-NAD.

[13] Chen Z, Gao L, Kwiat K (2003) Modeling the spread of active worms. *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, Vol. 3, pp 1890–1900 vol.3.

[14] Montgomery DC (2006) *Design and Analysis of Experiments* (John Wiley Sons, Inc., Hoboken, NJ, USA), .

[15] Fawcett T (2006) An introduction to roc analysis. *Pattern Recognition Letters* 27(8):861 – 874. https://doi.org/https://doi.org/10.1016/j.patrec.2005.10.010. ROC Analysis in Pattern Recognition Available at http://www.sciencedirect.com/science/article/pii/S016786550500303X

Appendix A: *ROC* curve tables for each attack in X1.

Table 7. *ROC* curve–DDoS.

Srl. No.	Threshold Values	True Positive Rates	False Positive Rates
0	0.05	0.988122	0.709847
1	0.10	0.834449	0.251521
2	0.15	0.546347	0.076897
3	0.20	0.412281	0.034715
4	0.25	0.326240	0.021181
5	0.30	0.174500	0.014498
6	0.35	0.060071	0.005598
7	0.40	0.034046	0.004247
8	0.45	0.018236	0.004242
9	0.50	0.012029	0.004062
10	0.55	0.008375	0.004230
11	0.60	0.007922	0.003610
12	0.65	0.011325	0.002390
13	0.70	0.003834	0.003239
14	0.75	0.007604	0.002011
15	0.80	0.002325	0.002228
16	0.85	0.003303	0.002019
17	0.90	0.002869	0.001219
18	0.95	0.003391	0.000000

Table 8. *ROC* curve–Horizontal Scan.

Srl. No.	Threshold Values	True Positive Rates	False Positive Rates
0	0.05	0.980923	0.671746
1	0.10	0.801310	0.229635
2	0.15	0.535047	0.065944
3	0.20	0.367494	0.033359
4	0.25	0.336297	0.016855
5	0.30	0.315267	0.011514
6	0.35	0.280511	0.006681
7	0.40	0.281735	0.004052
8	0.45	0.273902	0.003465
9	0.50	0.103106	0.004453
10	0.55	0.094414	0.003869
11	0.60	0.082550	0.003490
12	0.65	0.069647	0.002606
13	0.70	0.013252	0.003019
14	0.75	0.009700	0.002226
15	0.80	0.013984	0.001007
16	0.85	0.003567	0.002220
17	0.90	0.001900	0.001006
18	0.95	0.001448	0.000000

Table 9. *ROC* curve–Vertical Scan.

Srl. No.	**Threshold Values**	**True Positive Rates**	**False Positive Rates**
0	0.05	0.986140	0.668930
1	0.10	0.790630	0.219837
2	0.15	0.525050	0.071871
3	0.20	0.392624	0.032600
4	0.25	0.329005	0.018950
5	0.30	0.284869	0.011569
6	0.35	0.266143	0.006464
7	0.40	0.238666	0.004662
8	0.45	0.249302	0.004065
9	0.50	0.083409	0.004022
10	0.55	0.072817	0.004292
11	0.60	0.063875	0.003433
12	0.65	0.056931	0.002813
13	0.70	0.017280	0.002606
14	0.75	0.004793	0.002437
15	0.80	0.005289	0.001592
16	0.85	0.001315	0.002435
17	0.90	0.003129	0.000798
18	0.95	0.002638	0.000000

Table 10. *ROC* curve–FIN Scan.

Srl. No.	Threshold Values	True Positive Rates	False Positive Rates
0	0.05	0.999781	0.680154
1	0.10	0.979723	0.232652
2	0.15	0.964641	0.073529
3	0.20	0.954043	0.032693
4	0.25	0.937245	0.023962
5	0.30	0.763616	0.012672
6	0.35	0.269501	0.006014
7	0.40	0.124583	0.004035
8	0.45	0.049377	0.004029
9	0.50	0.040828	0.004243
10	0.55	0.029921	0.003633
11	0.60	0.017016	0.003829
12	0.65	0.018248	0.003024
13	0.70	0.008998	0.002821
14	0.75	0.018137	0.001615
15	0.80	0.003573	0.002430
16	0.85	0.002320	0.002220
17	0.90	0.003873	0.000797
18	0.95	0.003418	0.000000