Received: 25 July 2019 | Revised: 15 August 2019 | Accepted: 21 August 2019 | Published on: 10 October 2019

DOI: 10.1002/spy2.91

WILEY

RESEARCH ARTICLE

Using machine learning techniques to identify rare cyber-attacks on the UNSW-NB15 dataset

Sikha Bagui[1] | **Ezhil Kalaimannan**[1] | **Subhash Bagui**[2] | **Debarghya Nandi**[3] | **Anthony Pinto**[1]

[1]Department of Computer Science, The University of West Florida, Pensacola, Florida

[2]Department of Mathematics and Statistics, The University of West Florida, Pensacola, Florida

[3]Division of Epidemiology and Biostatistics, University of Illinois at Chicago, Chicago, Illinois

Correspondence
Ezhil Kalaimannan, Department of Computer Science, The University of West Florida, Pensacola, FL.
Email: ekalaimannan@uwf.edu

Funding information
Askew Institute of the University of West Florida

Abstract
This paper uses a hybrid feature selection process and classification techniques to classify cyber-attacks in the UNSW-NB15 dataset. A combination of *k*-means clustering, and a correlation-based feature selection, were used to come up with an optimum subset of features and then two classification techniques, one probabilistic, Naïve Bayes (NB), and a second, based on decision trees (J48), were employed. Our results show that this hybrid feature selection method in combination with the NB model was able to improve the classification accuracy of most attacks, especially the rare attacks. The false alarm rates were lower for most of the attacks, and particularly the rare attacks, with this combination of feature selection and the NB model. The J48 decision tree model, however, did not perform any better with the feature selection, but its classification rate for all attack families was already very high, with or without feature selection.

KEYWORDS
correlation based subset evaluation, feature selection, J48, *k*-means clustering, machine learning, Naïve Bayes classifier, UNSW-NB15

1 | INTRODUCTION

In today's dynamically changing and vastly interconnected world, the increase in network traffic has led to the significant growth in data and network security breaches. Network intrusions or attacks can be defined as a set of events, transmitted through network packets, undetectable, and unanalyzable by present-day firewalls, which are able to compromise confidentiality, integrity, and availability of computer systems.

Intrusion detection systems (IDSs) are the watchdogs of information systems.[1] IDSs are software and hardware systems designed and programmed to automate the process of monitoring events happening in a computer system or network and analyzing them for potential security issues.[2] The goal of an IDS is to detect cyberattacks by analyzing the signature of data packets as they traverse the network.

A firewall lacks the ability to detect and block modern cybersecurity attacks and hence is not an ideal candidate to perform deep packet analysis or inspection in such environments. However, implementing an IDS is a protection strategy to closely inspect network traffic and raise alerts or alarms in the event of a suspected attack pattern. Tuning signatures to eliminate false positives (FPs) or keeping up with the changing attack vectors used by cybercriminals is exhausting. IDSs have proven to be more efficient in detecting and classifying modern cybersecurity attacks and have also proven efficient in detecting zero-day attacks.

One way of classifying an IDS is based on its deployment position in a network and determining how its activities affect the entire network. A network-based intrusion detection system (NIDS) is a set of specialized components which is intended to monitor suspicious network traffic and to protect a system from network-based threats. By default, it monitors inbound and outbound network traffic for any rare activities. NIDSs monitor and inspect network traffic for segments or devices and analyze network, transport, and application level protocol packet headers to generate alerts on suspicious activity.

A host-based intrusion detection system (HIDS) is deployed on internal hosts in a network. Their main functionality is to monitor and analyze the internals of a single host system as it relates to the network packets on the network interfaces. HIDSs mainly contribute to inspecting internal machine state elements such as memory, operating system processes, libraries, and system calls. They are usually installed on mission-critical hosts of a network to monitor rare activities with respect to host attributes. NIDSs can also monitor and inspect multiple hosts from a single network point of view.

In this paper, we use machine learning techniques to classify the UNSW-NB15 dataset.[3] The UNSW-NB15 dataset,[3] developed using IXIA PerfectStorm, is a comprehensive network-based dataset which reflects modern network traffic scenarios and a variety of low footprint intrusions.[4] The KDD98,[5] KDDCUP99,[6] and NSL-KDD[7] were benchmark datasets generated and used a decade ago; however, recent studies have shown that these datasets have aged, meaning that these datasets no longer inclusively reflect modern day network traffic,[4,8] hence this work uses UNSW-NB15.

In this work we use a hybrid feature selection process for feature selection and two classifiers, one probabilistic, Naïve Bayes (NB), and the second, tree-based, J48, for classification. For the hybrid feature selection process, a combination of *k*-means clustering and a Correlation Based Subset evaluation, CFS, was used to come up with an optimum subset of features and then two classification algorithms were employed. This work was done using Weka.[9]

The rest of this paper is organized as follows. Section 2 presents the related works; Section 3 presents a brief description of the dataset; Section 4 discusses the feature selection process; Section 5 presents the classification methods employed; Section 6 presents the experimental design, results and discussion of results; and finally, Section 7 presents the conclusions.

2 | RELATED WORKS

Majority of the prior works on anomaly detection using machine learning techniques were employed on the KDD CUP99 dataset.[6,10–12]

Some of the more recent works were on other datasets. Chowdhury et al[13] employed a combination of two machine learning algorithms, simulated annealing and support vector machines, to improve the detection accuracy and false alarm rates (FARs) on a dataset from the Cyber Range Lab of the Australian Center for Cyber Security (ACCS). Karslig et al[14] worked on the NSL-KDD dataset. They implemented an anomaly based detection system using the *k*-means algorithm to distinguish between normal and abnormal samples and a threshold value was calculated. New samples that were far from the cluster's centers, more than the threshold value, were detected as anomalies. Their work yielded an accuracy of 80.119%.

Few works have been done on applying classification techniques to UNSW-NB15. Moustafa and Slay[15] presents a statistical analysis of the observations and attributes in UNSW-NB15[3] and five different classifiers are used to determine the accuracy and FARs. Moustafa et al[16] proposed an ensemble intrusion detection technique, AdaBoost, to detect malicious events. The AdaBoost ensemble learning method was developed using three machine learning techniques, decision tree, NB, and artificial neural networks on the UNSW-NB15 and another dataset, NIMS botnet. The ensemble technique provided higher detection rates and lower FP rates. Koroniotis et al[17] used machine learning techniques on flow identifiers on UNSW-NB15[3] to efficiently detect botnets and their tracks.

Mogal et al[18] used the UNSW-NB15 dataset for NIDSs. In this paper, preprocessing was done using Central Points of attribute values with the a priori algorithm, and the machine learning classifiers, NB and logistic regression were used. The results showed an improvement after preprocessing.

Moustafa and Slay[19] focused on classifying the families of attacks in the UNSW-NB15 dataset.[3] This work focused on identifying significant features of the UNSW-NB15[3] using Association Rule Mining Techniques and then they used the NB as well as the EM algorithms. However, for both these algorithms, their classification accuracy for the rare attacks was not so high (eg, 20% for BackDoor).

From a brief review of the related literature, it is evident that more work is needed for the identification of features for the families of attacks. This work focuses at identifying features for attack families. We present features which classify attacks and report on achieving improved classification accuracy and FAR. Also, there is not much work done on the classification of rare attacks (attack with a low number of instances).

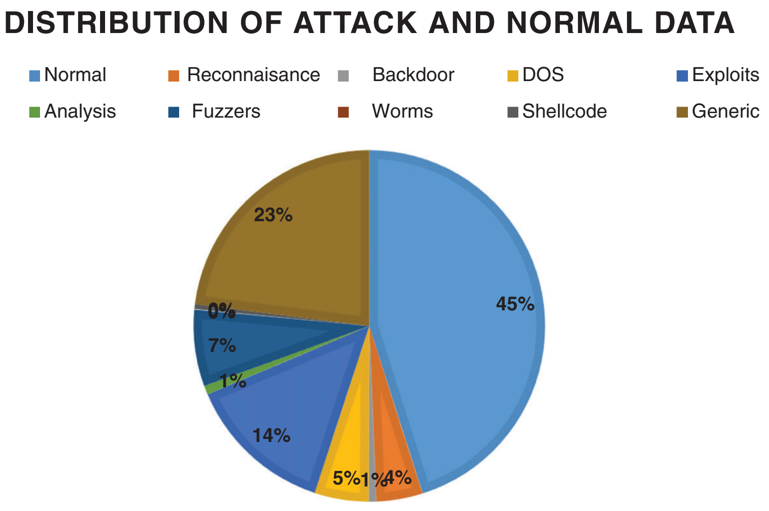

FIGURE 1 Distribution of attack and normal data[3,15,19]

3 | DESCRIPTION OF THE UNSW-NB15 DATASET

Based on the nature of the attack activities, the *UNSW-NB15* dataset[3] contains nine families of attacks, described in Moustafa and Slay[19] and Moustafa and Slay,[15] as outlined below:

1. *Fuzzers*: An attack which uses massive amounts of randomized data known as "Fuzz" to trigger a failure of a network or make an attempt to crash important servers on a network.
2. *Analysis*: This class contains various forms of attacks based on port scans, vulnerability scans, spam files and footprinting. It is also referred to as Active Reconnaissance where the network is scanned in some manner but not exploited.
3. *Backdoors*: This category utilizes a technique through which attackers use a legitimate system portal to gain illicit access. Backdoors use malicious software to install themselves in a computer system and to provide remote access to attackers as a part of an exploit.
4. *Denial of service (DoS)*: A well-known cyber-attack where the perpetrator seeks to compromise a machine with several illegitimate connection requests to make the network resources unavailable to its intended users temporarily or permanently. These can be difficult to distinguish from a legitimate network activity; however, there are some indicators to detect such intrusive activities in progress.
5. *Exploits*: Exploit attacks are generally achieved by targeting and compromising known vulnerabilities which exist in operating systems. Exploit software can also be used to automate these attacks once a potential vulnerability is detected in a network.
6. *Generic*: An attack based on ciphers, which is essentially a collision attack on the secret key generated by the cryptographic principles. This analysis can be applied to block, stream and message authentication code ciphers. It is often referred as a collision attack as most of these families are susceptible to the birthday attack. The attack depends on the higher likelihood of collisions found between random attack attempts.
7. *Reconnaissance*: This collects preliminary information about any public network or target host and is further utilized by exploit techniques to penetrate target hosts or networks by leveraging the gathered information. This class utilizes information freely available to the public "Whois" service, ARIN records and Shodan. Social media searches aid in reconnaissance attacks. They are also referred to as passive reconnaissance.
8. *Shellcode*: This is a subset of the class exploit. This attack utilizes a small piece of code as a payload of an exploit. The malicious code is injected into an active application to compromise and gain remote access to a victim's computer. It typically starts a command shell from which the attacker can control the compromised machine.
9. *Worms*: A worm is a malicious attack which spreads through network propagation and infects a much larger network rather quickly. A worm can also infect computers and then turn them into zombies or bots, with the intent of using them in distributed attacks through the formation of botnets.

Figure 1 presents the distribution of the nine families of attacks as well as the normal network traffic in the UNSW-NB15 dataset. Since 45% of the data in this dataset is normal network traffic, making it difficult to graphically see all the attack data, in Figure 2, we show just the attack data (without the normal traffic data).

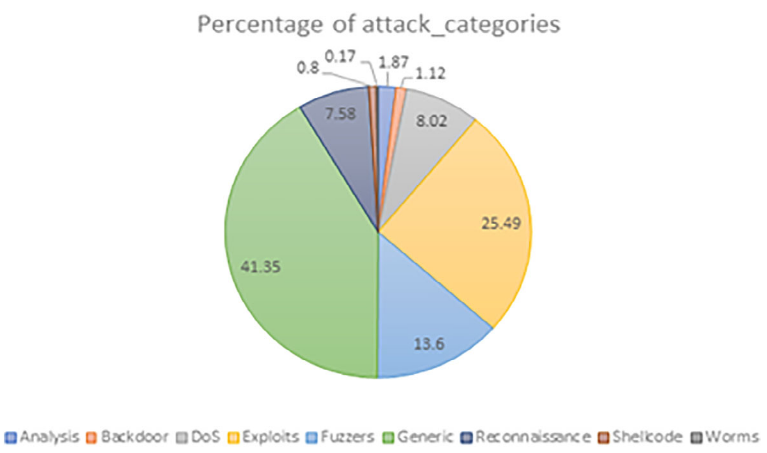

FIGURE 2 Distribution of attack data[3,15,19]

By the following attacks make up a very low percentage of the data/attacks, we refer to following attack families as *rare attacks*: Analysis (0.8%), BackDoor (1%), Shellcode (0.5%), and Worms (0%).

As shown in UNSW-NB15,[3] this dataset, UNSW-NB15, consists of 49 features. Table 1 presents the result with brevity in future sections, presents the features that were recorded in UNSW-NB15 dataset.[3,15,19] The last two features, 48 and 49, were not presented as these are the label features.

Flow features include identifying features which describe network traffic flow between connected packets of hosts; basic features include features which represent basic protocol header information; content features include attributes of TCP header information pertaining to content size, source IP address and packet flow; time features include features of time, for example, inter-arrival time between packets; and additionally generated features include general purpose features and connection features. The label features are not included in this table.

4 | FEATURE SELECTION

To use machine learning algorithms efficiently, it is becoming increasingly important to perform feature selection.[20] Feature selection, also referred to as attribute selection or variable selection, is a process of selecting more relevant attributes, and removing irrelevant or less relevant attributes or noisy data or features that do not add additional value to a machine learning algorithm.[21] Using only relevant features for machine learning algorithms allows for faster processing and more accurate predictability.[20] A lot of work has been done on feature selection using traditional techniques like Information Gain, the Gini Index, uncertainty, and correlation coefficients.[22] In this paper, we used a combination of two algorithms for feature selection: *K*-means clustering and correlation-based feature selection (CFS). *K*-means clustering and CFS are described briefly next.

4.1 | *k*-Means clustering

The *k*-means clustering algorithm takes the input parameter, *k*, and partitions a set of *n* objects into *k* clusters so that the intra-cluster similarity is high by the intercluster similarity is low. Cluster similarity is measured by the mean value of the objects in a cluster.[23] Below we present the standard *k*-means clustering algorithm.

```
Input:
k: the number of clusters
D: the dataset

Output: Set of k clusters

Method:
Arbitrarily choose k objects from D as the initial cluster centers;
Repeat
        Calculate the distance between each data object and all cluster centers and assign each object to the nearest cluster
        Recalculate the cluster center for each cluster
Until no change in the center of the clusters.
```

TABLE 1 Features present in the UNSW-NB15 dataset[3,15,19]

Feature #	Name	Description
	1	Flow features
1	srcip	Source IP address
2	sport	Source port number
3	dstip	Destinations IP address
4	dsport	Destination port number
5	proto	Protocol type, such as TCP, UDP
	2	Basic features
6	state	The states and its dependent protocol (e.g., CON)
7	dur	Row total duration
8	sbytes	Source to destination bytes
9	dbytes	Destination to source bytes
10	sttl	Source to destination time to live
11	dttl	Destination to source time to live
12	sloss	Source packets retransmitted or dropped
13	dloss	Destination packets retransmitted or dropped
14	service	Such as http, ftp, smtp, ssh, dns and ftp-data
15	sload	Source bits per second
16	dload	Destination bits per second
17	spkts	Source to destination packet count
18	dpkts	Destination to source packet count
	3	Content features
19	swin	Source TCP Window advertisement value
20	dwin	Destination TCP Window advertisement value
21	Stcpb	Source TCP base sequence number
22	dtcpb	Destination TCP base sequence number
23	smeansz	Mean of the packet size transmitted by srcip
24	dmeansz	Mean of the packet size transmitted by dstip
25	trans-depth	The connection of http request/response transaction
26	res_bdy_len	The content size of data transferred from http
	4	Time features
27	sjit	Source jitter
28	djit	Destination jitter
29	stime	Row start time
30	ltime	Row last time
31	sintpkt	Source interpacket arrival time
32	dintpkt	Destination interpacket arrival time
33	tcprtt	Setup round-trip-time, the sum of synack and ackdat
34	synack	The time between the SYN and the SYN_ACK packets
35	ackdat	The time between the ACK and the ACK_DAT packets
36	is_sm_ips_ports	If srcip(1) = dstip(3) and sportC(2) = dsport(4), assign 1 else 0.

TABLE 1 Continued

Feature #	Name	Description
	5	Additional generated features
37	ct_state_ttl	No. of each state(6) according to values of sttl(10) and dttl(11)
38	ct_flw_http_method	No. of methods such as Get and Post in http service
39	is_ftp_login	If ftp session is accessed by userid and password, then 1 else 0
40	ct_ftp_cmd	No. of flows that have command in ftp session
41[a]	ct_srv_src	No. of rows of the same service(14) and srcip(1) in 100 rows
42[a]	ct_srv_dst	No. of rows of the same service(14) and dstip(3) in 100 rows
43[a]	ct_dst_ltm	No. of rows of the same dstip(3) in 100 rows
44[a]	ct_src_ltm	No. of rows of the same srcip(1) in 100 rows
45[a]	ct_src_dport_ltm	No. of rows of the same srcip(1) and dsport(4) in 100 rows
46[a]	ct_dst_sport_ltm	No. of rows of the same dstip(3) and sport(2) in 100 rows
47[a]	ct_dst_src_ltm	No. of rows of the same srcip(1) and dstip(3) in 100 rows

[a]Connection features.

The k-means clustering algorithm is relatively scalable in processing large datasets because the computational complexity of the algorithm is $O(nkt)$, where n is the total number of objects, k is the number of clusters, and t is the number of iterations.[23]

4.2 | Correlation-based feature selection

CFS is a relatively fast scheme-independent attribute subset evaluator that ranks feature subsets according to a correlation based heuristic evaluation function.[24] CFS considers the predictive value of each attribute, along with the degree of inter-redundancy. An attribute subset is considered good if the attributes it contains are: (a) highly correlated with the class attribute; and (b) not strongly correlated with one another. Irrelevant features are ignored because they have a low correlation with the class and redundant features are removed since they will be highly correlated with one or more of the features. CFS's feature subset evaluation function is[24]:

$$M_s = \frac{kr_{cf}}{\sqrt{k + k(k-1)r_{ff}}}$$

M_s is the heuristic merit for a feature subset S containing k features, r_{cf} is the mean feature-class correlation ($f \in S$) and r_{ff} is the average feature intercorrelation.

We used CFS in this experimental process since, like the NB classifier, CFS assumes that features are conditionally independent given the class, thus improving the NB result with this feature selection measure.[24]

5 | CLASSIFICATION

Two types of classification models were used in this work: a probabilistic classifier, NB, and a tree-based classifier, the decision tree model, J48.

5.1 | NB classifier

The NB classifier is a probability-based machine learning algorithm based on Bayes theorem of posterior probability. It assumes class-conditional independence, that is, the effect of an attribute value on a given class is independent of the values of the other attributes.[23] Despite the independence assumption and its simplicity, the NB classifier is considered a competent classifier that scales well and works well with classification tasks,[25] and there are many recent works on trying to improve the performance of the NB classifier.[26,27] NB does not perform well in the presence of redundant and/or irrelevant attributes.[28] Redundant or highly correlated attributes bias the NB classifier,[29] hence feature selection is very important for this process.

The NB classifier works as follows[23]:

1. Let D be a training set of tuples and their associated class labels. Each tuple is represented by an n-dimensional attribute vector, $X = (x_1, x_2, \ldots, x_n)$.
2. Suppose there are m classes, $C_1, C_2, \ldots, C_m$. Given a tuple X, the classifier will predict that X belongs to the class with the highest posterior probability, conditioned on X. That is, the NB classifier predicts that the tuple X belongs to the class C_i if and only if:

$$P(C_i|X) > P(C_j|X) \text{ for } 1 \leq j \leq m, j \neq i.$$

Thus, we maximize $P(C_i|X)$.

5.2 | J48 decision tree

The decision tree algorithm, also known as the ID3 (iterative dichotometer), is a tree like structure where the internal nodes test an attribute and each branch represents an outcome of the test, and each leaf node holds a class label. The nodes of the decision tree are determined by an attribute selection method, in this case, information gain. The attribute with the highest information gain is selected as the splitting attribute for a node, N. This attribute minimizes the information needed to classify the tuples in the resulting partitions and reflects the least randomness or "impurity" in these partitions. This approach minimizes the expected number of tests needed to classify a given tuple. The expected information needed to classify a tuple D is given by[23]:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

where p_i is the probability that an arbitrary tuple in D belongs to class C_i. Since the information is encoded in bits, a log function of base 2 is used. $Info(D)$ is also known as the entropy of D.

J48, an extension of ID3, is able to handle missing values and allows for pruning. When decision trees are built, many of the branches may reflect anomalies in training data due to noise or outliers. In tree pruning, statistical measures are used to remove the least reliable branches.[23]

6 | EXPERIMENTAL DESIGN, RESULTS AND DISCUSSION

Figure 3 presents our experimental design process. The first step was feature selection. Then, classification algorithms were used to determine the accuracy, attack detection rate (ADR), and FAR.

A sample of 8000 records was randomly selected from the UNSW-NB15 dataset,[3] and we performed feature selection using k-means clustering and CFS. With k-means clustering, features were individually selected, based on their means, and CFS evaluated the benefits of each candidate subset. The limitations of both approaches used individually for feature selection clearly suggest the need for a hybrid model,[30] hence the use of k-means and CFS. This hybrid model was successfully used by Chormunge and Jena[31] to reduce the dimensionality of data.

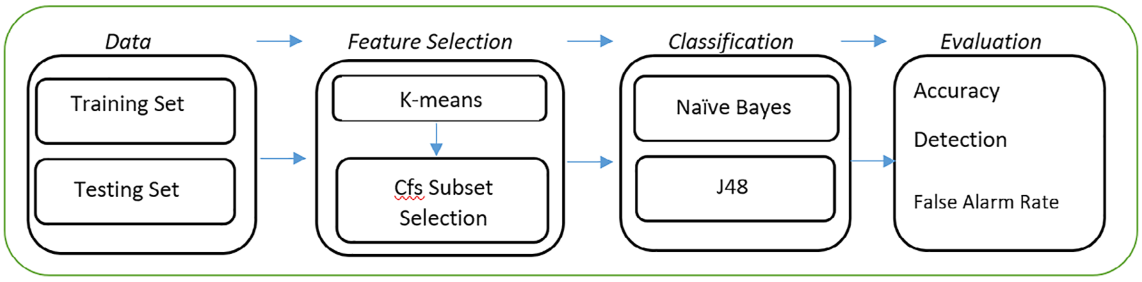

FIGURE 3 Experimental design for our network intrusion system

TABLE 2 Selected final cluster centroids for $k = 2$

	Cluster 0 (nonattack)	Cluster 1 (attack)
Protocol	TCP	UDP
Service	—	DNS
State	FIN	INT
Label	0.36	0.76
sbytes	14 781.7794	538.9787
dbytes	25 230.7427	57.8624
dinpkt	195.3254	40.842
ct_srv_src	4.66	14.9134
ct_state_ttl	1.0013	1.7734
ct_src_dport_ltm	1.4577	8.7412
is_sm_ips_port	0.0001	0.0232

6.1 | The feature selection process

Our feature selection process was:

- Perform *k*-means clustering with different *k*s.
- Select the feature characteristics unique in the cluster which account for a major population of a specific attack family.
- Run subsetEval on it and select the best features.
- Try to preserve as many of the connection features as possible.
- Select the top 7-8 features based on the union of the above two results. In case of exceedance, give preference to connection features and clustering results.

6.2 | *K*-means clustering

Several runs were performed with different *k*s. Below we present results for the *k*'s that had some meaningful results.

6.2.1 | $k = 2$

Running *k*-means for two clusters gave us some interesting results. The label mean for all points in cluster 1 (attack) was 0.7667, while the mean for all points in cluster 0 (nonattack) was 0.3538. Table 2 presents the results of the final cluster centroids for $k = 2$, for some features that we found a distinction in.

From Table 2, we can infer that the attack data mostly occurs with the following characteristics:

- Protocol—UDP
- Service—DNS
- Low values for sbytes, dbytes, dinpkt
- High values for connection features like ct_srv_src, ct_state_ttl, ct_src_dport_ltm, and so on.
- High value for is_sm_ips_port

6.2.2 | $k = 10$

For $k = 10$, the results were not so distinct. Based on the 10 cluster means, the following attributes were selected: protocol, ser, state, sload, sloss, ct_flw_http. Table 3 presents their cluster centroids.

TABLE 3 Final cluster centroids for $k = 10$

Attr	Cl. 0	Cl.1	Cl.2	Cl.3	Cl.4	Cl.5	Cl.6	Cl.7	Cl.8	Cl.9
Protocol	Tcp	Unas	Tcp	tcp	udp	Udp	Udp	Tcp	udp	tcp
Ser	http	—	—	—	dns	Dns	—	—	dns	http
State	FIN	INT	FIN	FIN	INT	INT	INT	FIN	INT	FIN
Sload	111 971	131 279 837	39 425	1 215 775	97 174 286	97 144 922	224 665 772	57 642	59 723 747	97 916
Sloss	13	0.56	5	8	0	0	0	8	0	14
ct_flw_http	1	0	0.007	0.0002	0	0	0	0.008	0	0.7

TABLE 4 Selected features

Attack family	Features
Fuzzers	7, 10, 11, 14
Analysis[a]	5, 6, 7, 10, 11, 14, 19, 37
BackDoor[a]	5, 6, 7, 14, 25, 33, 34, 35, 37, 39
DOS	5, 7, 10, 12, 13, 18, 19, 20, 24, 28, 37, 41, 42
Exploits	7, 11, 25, 36, 38, 39, 40, 41, 42, 43, 47
Generics	5, 6, 7, 14, 23, 33, 36, 37, 46
Reconnaisance	7, 10, 11, 14, 41, 43, 44, 47, 42
Shellcode[a]	7, 10, 11, 14, 41, 42, 44
Worms[a]	10, 11, 13, 14, 25, 38,41

[a]Rare attacks.

6.3 | Correlation-based feature selection

Based on CFS subsetEval, we obtained the following results: Service, Sttl, Dttl,sloss,dloss, Trans_depth, Ct_srv_src, and Ct_ftw_http.

The final set of features that were extracted for each attack family, based on our feature selection process, is presented in Table 4. On the average, our work reduced the number of features from what was previously reported by Moustafa and Slay.[19] Although the number of features for the DoS attacks was 13, all other attack families had less than 11 features. Excluding BackDoor, which had 10 features, the other rare attacks, Shellcode and Worms had 7 features each, and Analysis had eight features.

6.3.1 | Analysis of the selected features

In Table 4, we can come up with the following analysis (refer to Table 1 for feature numbers).

Analysis is identifiable by one flow feature, protocol type (5), five basic features, state, dur, sttl, dttl and service (6, 7, 10, 11, and 14 respectively), one content feature, swin (19), and one additionally generated feature, cd_state_ttl (37), which is not a connection feature.

BackDoor is identifiable by one flow feature, protocol type (5), three basic features (6, 7, 14), one content feature (25), three time features, tcprtt, synack, and ackdat (33, 34 and respectively) and two additionally generate features, ct_state_ttl and is_ftp_login (37 and 39 respectively).

Shellcode is identifiable by basic features, dur, sttl, dttl and service (7, 10, 11, and 14 respectively) and three additionally generated features, ct_srv_src, cd_srv_dst, and ct_src_ltm (41, 42, and 44 respectively).

Worms are identifiable by basic features, sstl, dttl, dloss, and servicw (10, 11, 13, and 14 respectively), one content feature, trans-depth (25), and two additionally generated features, ct_flw_http_method (38), and ct_srv_src (41), of which feature 41 is a connection feature.

Of the 47 features (as per Table 1), only 27 of the features were useful. Of the flow features, only protocol type (5) was useful. Of the basic features, sbytes, dbytes, sload, dload spkts, and dpkts were not useful. Of the content features, stcpb, dtcpb, and res_bdy_len were not useful. Of the time features, sjit, stme, ltime, sintpkt, and dintpkt were not useful. Of the additional generated features, ct_ftp_cmd, and ct_src_dport_ltm were not useful.

The most important features were: dur (7), which occurred in all the attack families except Worms; service (14), which occurred in all the attack families except DoS and Exploits; sttl and dttl (10 and 11 respectively) that occurred in six of the nine attack families; and ct_srv_src (41) which occurred in five of the attack families.

6.4 Classification

After feature selection, two classification algorithms were used: NB and J48 decision trees. The two classification algorithms were compared based on classification accuracy, ADR and FAR. The classification accuracy, ADR and FAR were determined using the classification metrics {TP, TN, FP, FN}. True positive (TP) is the number of correctly classified attacks for each attack family; true negative (TN) is the number of correctly classified nonattacks or normal rows; FP is the number of misclassified attacks for each attack family; and false negative (FN) is the number of misclassified normal rows.

- Classification accuracy is the TP plus TN divided by the total number of instances:

$$\text{Accuracy} = (\text{TP} + \text{TN})/\text{total no.of instances}$$

- ADR is the TP divided by the TP plus FN:

$$\text{ADR} = (\text{TP})/(\text{TP} + \text{FN})$$

- FAR is the FP plus FN divided by the total number of instances:

$$\text{FAR} = (\text{FP} + \text{FN})/\text{total no.of instances}$$

The best detection is achieved by getting the accuracy as close to 100% as possible and the FAR as close to 0% as possible.

Tables 5 presents the results of the NB classification and Table 6 presents the results of the J48 algorithm. The first two columns of both tables show the percentage of correct classification (classification accuracy) without and with feature selection respectively, columns 3 and 4 present the ADR without and with feature selection respectively, and columns 5 and 6 present the FAR without and with feature selection respectively.

From Table 5 we can see that the NB classifier achieved a much higher classification accuracy rate with feature selection. This is very important, especially for rare attacks like Analysis, where the classification rate jumped from 74% to 90.66%, BackDoor, where the classification rate jumped from 66% to 90.02%, Shellcode, where the classification rate with feature selection jumped from 72% to 75.24%, and Worms, where the classification rate with feature selection jumped from 84% to

TABLE 5 Results of the NB classifier

	Accuracy w/o FS (%)	Accuracy w FS (%)	ADR w/o FS (%)	ADR w FS (%)	FAR w/o FS (%)	FAR w FS (%)
Fuzzers	57	80.03	96.70	58.90	43	19.97
Analysis[a]	74	90.66	90.99	90.10	26	9.34
BackDoor[a]	66	90.02	92.96	89.50	34.00	9.98
DOS	66	92.97	84.17	42.18	34.00	7.03
Exploits	56.96	46.70	90.67	88.45	43.04	53.30
Generics	83	92.61	96.61	96.36	17.00	7.39
Reconnaisance	65	71.42	97.59	93.59	35.00	28.58
Shellcode[a]	72	75.24	98.15	91.53	28.00	24.76
Worms[a]	84	99.00	93.18	29.54	16.00	1.00

[a]Rare attacks.

TABLE 6 Results of J48 classifier

	Accuracy w/o FS (%)	% Accuracy with FS (%)	ADR w/o FS (%)	ADR w FS (%)	FAR w/o FS (%)	FAR w FS (%)
Fuzzers	97.83	92.70	75.18	2.42	2.17	7.30
Analysis[a]	99.23	99.18	6.64	0	0.77	0.82
BackDoor[a]	99.29	99.29	2.22	0	0.71	0.71
DOS	95.07	95.06	5.99	1.76	4.93	4.94
Exploits	93.32	89	64.27	52.25	6.68	11.00
Generics	99.35	99.13	97.80	96.20	0.65	0.87
Reconnaisance	98.83	96.30	79.50	41.47	1.17	3.70
Shellcode[a]	99.59	99.54	43.12	0	0.41	0.46
Worms[a]	99.59	99.94	40.90	0	0.41	0.06

[a]Rare attacks.

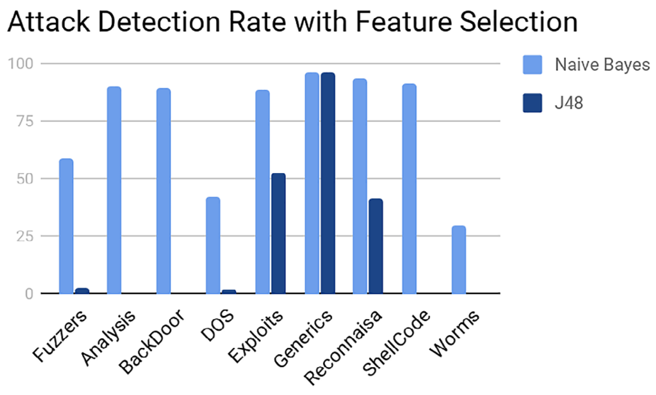

FIGURE 4 ADR with feature selection

99%. So, based on accuracy, the NB classifier did a good job of classifying rare attacks with feature selection. There was not much of an improvement in the ADR based on feature selection. However, there appeared to be in improvement in the FAR with feature selection, especially for the rare attacks. The FAR rate for Analysis dropped from 26% to 9.34%, BackDoor dropped from 34% to 9.98%, Shellcode dropped from 28% to 24.76%, and Worms dropped from 16% to 1%. Most other attack families also showed significant drops in FARs except for Exploits.

From Table 6, we can see that feature selection did not have much of an impact on the classification accuracy rate for J48, but the classification rate, without or with feature selection was very high. From this we conclude that not all the features are needed to correctly classify the attacks in the J48 algorithm, since a subset of features is achieving the same rate of accuracy. This also implies that there are a lot of redundant features in the data. The ADR, however, went down with feature selection, and for the rare attacks it was at 0%. The FARs, on the contrary, went up in some cases and down in some cases (and stayed the same in some cases) with feature selection. But overall, the FARs for the rare attacks without or with feature selection was low, performing better than the NB algorithm with feature selection.

Figure 4 shows the ADR with feature selection was much higher with NB. J48 had 0% ADR for some of the rare attacks like Analysis, BackDoor, Shellcode, and Worms, but NB was able to detect them.

7 | CONCLUSION

We used a hybrid feature selection process to identify the important features for each family of attacks. Overall, the most important features were: dur (7), which occurred in all the attack families except Worms; service (14), which occurred in all the attack families except DoS and Exploits; sttl and dttl (10 and 11 respectively) that occurred in six of the nine attack families; and ct_srv_src (41) which occurred in five of the attack families.

Using the subset of features identified through the hybrid feature selection process, we achieved a higher classification rate as well as lower FAR rate for most families of attacks using the NB classifier. We were able to better identify rare attacks such as Analysis, BackDoor, Shellcode, and Worms using the NB classification model. The high detection rate of worms can be noted as worthy since its classification accuracy was 99% and FAR was 1%. The feature selection process, however, did not have any impact on the J48 algorithm. From the results of the J48 classification, which performed almost as well with feature selection as without feature selection, we can deduce that this dataset contained a lot of redundant features.

ACKNOWLEDGMENT

This work has been partially supported by the Askew Institute of the University of West Florida.

ORCID

Ezhil Kalaimannan https://orcid.org/0000-0003-2080-7007

REFERENCES

1. Axelsson S. The base-rate fallacy and the difficulty of intrusion detection. *ACM Trans Inform Syst Secur*. 2000;3(3):186-205.
2. Gupta JND, Kalaimannan E, Patnayakuni R. *IDS Alarms investigation with limited resources. Proceedings of the Pre-ICIS Workshop on Information Security and Privacy (WISP'12)*; 2012. https://aisel.aisnet.org/wisp2012/15/.
3. UNSW-NB15. *UNSW-NB15 Dataset Description*; 2015. https://www.unsw.adfa.edu.au/australian-centre-for-cyber-security/cybersecurity/ADFA-NB15-Datasets/. Accessed May, 2015.
4. Moustafa N, Slay J. *UNSW-NB15: a comprehensive dataset for network intrusion detection systems. Military Communications and Information Systems Conference (MilCIS). Canberra, Australia: IEEE*; 2015. https://doi.org/10.1109/MilCIS.2015.7348942.
5. McHugh J. Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Trans Inform Syst Secur*. 2000;3:262-294.
6. KDD Cup 1999. *KDD Cup 1999 Data*; 2007. http://kdd.ics.uci.edu/databases/kddcup99. Accessed August, 2019.
7. NSL-KDD dataset for network-based intrusion detection systems. http://nsl.cs.unb.ca/NSL-KDD/. Accessed February 12, 2019.
8. Creech G, Hu J. *Generation of a new IDS test dataset: time to retire the KDD collection. Proceedings of IEEE Wireless Communications and Networking Conference: Services and Applications. Shanghai, China: IEEE*; 2013:4487-4492.
9. Weka 3. Machine Learning Software in Java. https://www.cs.waikato.ac.nz/ml/weka/. Accessed April 18, 2019.
10. Leung K, Leckie C. *Unsupervised anomaly detection in network intrusion detection using clusters. Proceedings of 28th Australian Computer Science Conference*. Newcastle, Australia: University of Newcastle; 2005.
11. Sabhani M, Serpen G. *Application of machine learning algorithms to KDD intrusion detection dataset within misuse detection context. Proceedings of the International Conference on Machine Learning Models, Technologies and Applications (MLMTA), Las Vegas, NV*; 2003.
12. Volden HH. Anomaly detection using machine learning techniques [master's thesis]; Spring 2016.
13. Chowdhury MN, Ferens K, Ferens M. *Network intrusion detection using machine learning. Proceedings of International Conference on Security and Management. Athens: CSREA Press*; 2016.
14. KarsligEl ME, Yavuz AG, Guvensan MA, Hanifi K, Bank H. *Network intrusion detection using machine learning anomaly detection algorithms. Proceedings of Signal Processing and Communications Applications Conference, Antalya, Turkey: IEEE*; 2017. https://doi.org/10.1109/SIU.2017.7960616.
15. Moustafa N, Slay J. The evaluation of network anomaly detection systems: statistical analysis of the UNSW-NB15 dataset and the comparison with the KDD99 dataset. *Inform Secur J Glob Persp*. 2016;25(1-3):18-31.
16. Moustafa N, Turnbull BP, Choo KR. An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things. *IEEE Internet Things J*. 2018;6:4815-4830. https://doi.org/10.1109/JIOT.2018.2871719.
17. Koroniotis N, Moustafa N, Sitnikova E, Slay J. *Towards developing network forensic mechanism for botnet activities in the IoT based on machine learning techniques. International Conference on Mobile Networks and Management*. Cham: Springer; 2017.
18. Mogal DG, Ghungrad SR, Bhusare BB. NIDS using machine learning classifiers on UNSW-NB15 and KDDCUP99 datasets. *Int J Adv Res Comput Commun Eng*. 2017;6(4):533-537.
19. Moustafa N, Slay J. *The significant features of UNSW-NB15 and the KDD99 datasets for network intrusion detection systems. 4th International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security. Kyoto, Japan: IEEE*; 2016:25-31.
20. Kumar V, Minz S. Feature selection: a literature review. *Smart Comput Rev*. 2014;4(3):211-229.
21. Yang P, Zhu Q. Finding key attribute subset in dataset for outlier detection. *Knowl Based Syst*. 2011;24(2):269-274.
22. Forman G. An extensive empirical study of feature selection metrics for text classification. *J Mach Learn Res*. 2003;3:1289-1305.
23. Han J, Kamber M, Pei J. *Data Mining: Concepts and Techniques*. Burlington, MA: Morgan Kaufmann Publishers; 2012.
24. Hall M. Correlation-Based Feature Selection for Machine Learning [mater's thesis]; 1999. https://www.cs.waikato.ac.nz/~mhall/thesis.pdf. Accessed May 2019.
25. Fang X. Inference-based Naïve Bayes: turning Naïve Bayes cost-sensitive. *IEEE Trans Knowl Data Eng*. 2013;25(10):2302-2313.
26. Wong TT. A hybrid discretization method for Naïve Bayesian classifiers. *Pattern Recogn*. 2012;45(6):2321-2325.
27. Wong TT, Chang LH. Individual attribute prior setting methods for Naïve Bayesian classifiers. *Pattern Recogn*. 2011;44(5):1041-1047.

28. Bermejo P, Gamez JA, Puerta JM. Speeding up incremental wrapper feature subset selection, with Naïve Bayes classifier. *Knowl Based Syst*. 2014;55:140-147.
29. Inza I, Larranga P, Etxeberria R, Sierra B. Feature subset selection by Bayesian network-based optimization. *Artif Intell*. 2000;123(1-2):157-184.
30. Ruiz R, Ruiz JSA, Riquelme JC. *Best agglomerative ranked subset for feature selection*. *JMLR Workshop and Conference Proceedings*. Vol 4; 2008:148-162. http://proceedings.mlr.press/v4/ruiz08a.html.
31. Chormunge S, Jena S. Correlation based feature selection with clustering for high dimensional data. *J Electr Syst Inform Technol*. 2018;5:542-549. https://doi.org/10.1016/j.jesit.2017.06.004.

How to cite this article: Bagui S, Kalaimannan E, Bagui S, Nandi D, Pinto A. Using machine learning techniques to identify rare cyber-attacks on the UNSW-NB15 dataset. *Security and Privacy*. 2019;2:e91. https://doi.org/10.1002/spy2.91