Towards Generating Real-life Datasets for Network Intrusion Detection

Monowar H. Bhuyan[1], Dhruba K. Bhattacharyya[2], and Jugal K. Kalita[3]
(Corresponding author: Monowar H. Bhuyan)
Department of Computer Science and Engineering, Kaziranga University, Jorhat-785006, Assam, India[1]
(Email: monowar.tezu@gmail.com)
Department of Computer Science and Engineering, Tezpur University, Tezpur-784028, Assam, India[2]
(Email: dkb@tezu.ernet.in)
Department of Computer Science, University of Colorado at Colorado Springs, CO 80918, USA[3]
(Email: jkalita@uccs.edu)

(Received February 5, 2015; revised and accepted Apr. 20 & May 9, 2015)

Abstract

With exponential growth in the number of computer applications and the sizes of networks, the potential damage that can be caused by attacks launched over the Internet keeps increasing dramatically. A number of network intrusion detection methods have been developed with respective strengths and weaknesses. The majority of network intrusion detection research and development is still based on simulated datasets due to non-availability of real datasets. A simulated dataset cannot represent a real network intrusion scenario. It is important to generate real and timely datasets to ensure accurate and consistent evaluation of detection methods. In this paper, we propose a systematic approach to generate unbiased full-feature real-life network intrusion datasets to compensate for the crucial shortcomings of existing datasets. We establish the importance of an intrusion dataset in the development and validation process of detection mechanisms, identify a set of requirements for effective dataset generation, and discuss several attack scenarios and their incorporation in generating datasets. We also establish the effectiveness of the generated dataset in the context of several existing datasets.

Keywords: Dataset, intrusion detection, NetFlow, network traffic

1 Introduction

In network intrusion detection, particularly when using anomaly based detection, it is difficult to accurately evaluate, compare and deploy a system that is expected to detect novel attacks due to scarcity of adequate datasets. Before deploying in any real world environment, an anomaly based network intrusion detection system (ANIDS) must be trained, tested and evaluated using real labelled network traffic traces with a intensive set of intrusions or attacks. This is a significant challenge, since not many such datasets are available. Therefore the detection methods and systems are evaluated only with a few publicly available datasets that lack comprehensiveness and completeness [2, 17] or are outdated. For example, Cooperative Association for Internet Data Analysis (CAIDA) Distributed Denial of Service (DDoS) 2007, Lawrence Berkeley National Laboratory (LBNL), and ICSI datasets are heavily anonymized without payload information, decreasing research utility. Researchers also frequently use a single NetFlow based intrusion dataset found at [25, 40] with a limited number of attacks.

1.1 Importance of Datasets

In network traffic anomaly detection, it is always important to test and evaluate detection methods and systems using datasets as network scenarios evolve. We enumerate the following reasons to justify the importance of a dataset.

- *Repeatability of experiments*: Researchers should be able to repeat experiments with the dataset and get similar results, when using the same approach. This is important because the proposed method should cope with the evolving nature of attacks and network scenarios.

- *Validation of new approaches*: New methods and algorithms are being continuously developed to detect network anomalies. It is necessary that every new approach be validated.

- *Comparison of different approaches*: State-of-the-art network anomaly detection methods must not only be validated, but also show improvements over older methods in performance in a quantifiable manner. For example, the

DARPA 1998 dataset [26] is commonly used for performance evaluation of anomaly detection systems [24]. So that one method can be compared against others.

- *Parameters tuning*: To properly obtain the model to classify the normal from malicious traffic, it is necessary to tune model parameters. Network anomaly detection assumes the normality model to identify malicious traffic. For example, Cemerlic et al. [9] and Thomas et al. [44] use the attack-free part of the DARPA 1999 dataset for training to estimate parameter values.

- *Dimensionality or the number of features*: An optimal set of features or attributes should be used to represent normal as well as all possible attack instances.

1.2 Requirements

Although good datasets are necessary for validating and evaluating IDSs, generating such datasets is a time consuming task. A dataset generation approach should meet the following requirements.

- *Real world*: A dataset should be generated by monitoring the daily situation in a realistic way, such as the daily network traffic of an organization.

- *Completeness in labelling*: The labelling of traffic as benign or malicious must be backed by proper evidence for each instance. The aim these days should be to provide labelled datasets at both packet and flow levels for each piece of benign and malicious traffic.

- *Correctness in labelling*: Given a dataset, labelling of each traffic instance must be correct. This means that our knowledge of security events represented by the data has to be certain.

- *Sufficient trace size*: The generated dataset should be unbiased in terms of size in both benign and malicious traffic instances.

- *Concrete feature extraction*: Extraction of an optimal set of concrete features when generating a dataset is important because such features play an important role when validating a detection mechanisms.

- *Diverse attack scenarios*: With the increasing frequency, size, variety and complexity of attacks, intrusion threats have become more complex including the selection of targeted services and applications. When contemplating attack scenarios for dataset generation, it is important to tilt toward a diverse set of multi-step attacks that are recent.

- *Ratio between normal and attack traffic*: Most benchmark datasets are biased because the proportion of normal and attack traffic are not the same. This is because normal traffic is usually much more common than anomalous traffic. However, the evaluation of an intrusion detection method or system using biased datasets may not be fit for real-time deployment in certain situations. Most existing datasets have been created based on the following assumptions.

  – Anomalous traffic is statistically different from normal traffic [13].

  – The majority of network traffic instances is normal [36].

  However, unlike most traditional intrusions, DDoS attacks do not follow these assumptions because they change network traffic rate dynamically and employ multi-stage attacks. A DDoS dataset must reflect this fact.

1.3 Motivation and Contributions

By considering the aforementioned requirements, we propose a systematic approach for generating real-life network intrusion dataset at both packet and flow levels with a view to analyzing, testing and evaluating network intrusion detection methods and systems with a clear focus on anomaly based detectors. The following are the major contributions of this paper.

- We present guidelines for real-life intrusion dataset generation.

- We discuss systematic generation of both normal and attack traffic.

- We extract features from the captured network traffic such as *basic*, *content*-based, *time*-based, and *connection*-based features using a distributed feature extraction framework.

- We generate three categories of real-life intrusion datasets, viz., (i) TUIDS (Tezpur University Intrusion Detection System) intrusion dataset, (ii) TUIDS coordinated scan dataset, and (iii) TUIDS DDoS dataset. These datasets are available for the research community to download for free.

1.4 Organization of the Paper

The remainder of the paper is organized as follows. Section 2 discusses prior datasets and their characteristics. Section 3 is dedicated to the discussion of a systematic approach to generate real-life datasets for intrusion detection with a focus on network anomaly detectors. Finally, Section 4 presents observations and concluding remarks.

2 Existing Datasets

As discussed earlier, datasets play an important role in the testing and validation of network anomaly detection methods or systems. A good quality dataset not only allows us to identify the ability of a method or a system to detect anomalous behavior, but also allows us to provide potential effectiveness when deployed in real operating environments. Several

datasets are publicly available for testing and evaluation of network anomaly detection methods and systems. A taxonomy of network intrusion datasets is shown in Figure 1. We briefly discuss each of them below.

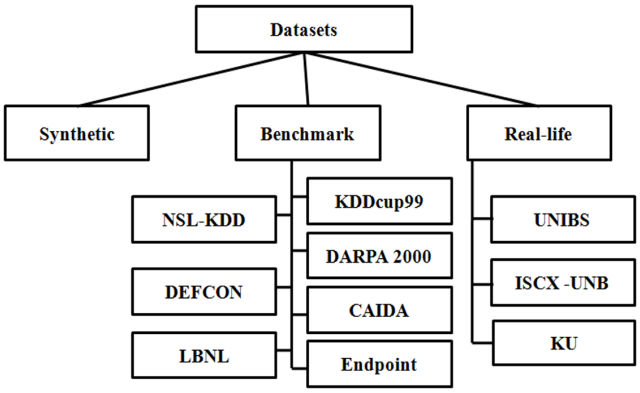

Figure 1: A taxonomy of network intrusion datasets [2]

2.1 Synthetic Datasets

Synthetic datasets are generated to meet specific needs or certain conditions or tests that real data satisfy. Such datasets are useful when designing any prototype system for theoretical analysis so that the design can be refined. A synthetic dataset can be used to test and create many different types of test scenarios. This enables designers to build realistic behavior profiles for normal users and attackers based on the dataset to test a proposed system. This provides initial validation of a specific method or a system; if the results prove to be satisfactory, the developers then continue to evaluate a method or a system in a specific domain real-life data.

2.2 Benchmark Datasets

We discuss seven publicly available benchmark datasets generated using simulated environments in large networks. Different attack scenarios were simulated during the generation of these datasets.

2.2.1 KDDcup99 Dataset

Since 1999, the KDDcup99 dataset [21] has been the most widely used dataset for evaluation of network based anomaly detection methods and systems. This dataset was prepared by Stolfo et al. [41] and is built upon the data captured in the DARPA98 IDS evaluation program. The KDD training dataset consists of approximately $4,900,000$ single connection vectors, each of which contains 41 features and is labelled as either normal or attack of a specific attack type. The test dataset contains about $300,000$ samples with a total 24 training types, with an additional 14 attack types in the test dataset only [14]. The represented attacks are mainly four types: denial of service, remote-to-local, user-to-root, and surveillance or probing.

- *Denial of Service (DoS)*: An attacker attempts to prevent valid users from using a service provided by a system. Examples include SYN flood, smurf and teardrop attacks.
- *Remote to Local (r2l)*: Attackers try to gain entrance to a victim machine without having an account on it. An example is the password guessing attack.
- *User to Root (u2r)*: Attackers have access to a local victim machine and attempt to gain privilege of a superuser. Examples include buffer overflow attacks.
- *Probe*: Attackers attempt to acquire information about the target host. Some examples of probe attacks are portscans and ping-sweep attacks.

Background traffic was simulated and the attacks were all known. The training set, consisting of seven weeks of labelled data, is available to the developers of intrusion detection systems. The testing set also consists of simulated background traffic and known attacks, including some attacks that are not present in the training set. The distribution of normal and attack traffic for this dataset is reported in Table 1. We also identify the services associated with each category of attacks [12, 22] and summarize them in Table 2.

2.2.2 NSL-KDD Dataset

Analysis of the KDD dataset showed that there were two important issues with the dataset, which highly affect the performance of evaluated systems often resulting in poor evaluation of anomaly detection methods [43]. To address these issues, a new dataset known as NSL-KDD [32], consisting of selected records of the complete KDD dataset was introduced. This dataset is also publicly available for researchers[1] and has the following advantages over the original KDD dataset.

- This dataset doesn't contain superfluous and repeated records in the training set, so classifiers or detection methods will not be biased towards more frequent records.
- There are no duplicate records in the test set. Therefore, the performance of learners is not biased by the methods which have better detection rates on frequent records.
- The number of selected records from each difficulty level is inversely proportional to the percentage of records in the original KDD dataset. As a result, the classification rates of various machine learning methods vary in a wider range, which makes it more efficient to have an accurate evaluation of various learning techniques.
- The number of records in the training and testing sets is reasonable, which makes it practical to run experiments on the complete set without the need to randomly select a small portion. Consequently, evaluation results of different research groups are consistent and comparable.

[1] http://www.iscx.ca/NSL-KDD/

Table 1: Distribution of normal and attack traffic instances in KDDCup99 dataset

Dataset	*DoS*		*Probe*		*u2r*		*r2l*		Normal
	Total instances	Attacks	Total instances	Attacks	Total instances	Attacks	Total instances	Attacks	
10% KDD	391458	smurf, neptune, back, teardrop, pod, land	4107	satan, ipsweep, portsweep, nmap	52	buffer_overflow, rootkit, loadmodule, perl	1126	warezclient, guess_passwd, warezmaster, imap, ftp_write, multihop, phf, spy	97277
Corrected KDD	229853		4107		52		1126		97277
Whole KDD	229853		4107		52		1126		97277

Table 2: List of attacks and corresponding services in KDDcup99 dataset

Dataset	*DoS*		*Probe*		*u2r*		*r2l*	
	Attack name	Service(s)	Attack name	Service(s)	Attack name	Service(s)	Attack name	Service(s)
KDD99	apache2	http	ipsweep	icmp	eject	Any user session	dictionary	telnet, rlogin, pop, imap, ftp
	back	http	mscan	many	ffbconfig	Any user session	ftp-write	ftp
	land	N/A	nmap	many	fdformat	Any user session	guest	telnet, rlogin
	mailbomb	smtp	saint	many	loadmodule	Any user session	imap	imap
	SYN flood	Any TCP	satan	many	perl	Any user session	named	dns
	ping of death	icmp			ps	Any user session	named	dns
	process table	Any TCP			Xterm	Any user session	sendmail	smtp
	smurf	icmp	-		-		xlock	X
	syslogd	syslog					xsnoop	X
	teardrop	N/A					-	-
	udpstorm	echo/chargen						

The NSL-KDD dataset consists of two parts: (i) KDDTrain$^+$ and (ii) KDDTest$^+$. The KDDTrain$^+$ part of the NSL-KDD dataset is used to train a detection method or system to detect network intrusions. It contains four classes of attacks and a normal class dataset. The KDDTest$^+$ part of NSL-KDD dataset is used for testing a detection method or a system when it is evaluated for performance. It also contains the same classes of traffic present in the training set. The distribution of attack and normal instances in the NSL-KDD dataset is shown in Table 3.

Table 3: Distribution of normal and attack traffic instances in NSL-KDD dataset

Dataset	*DoS*	*u2r*	*r2l*	*Probe*	*Normal*	*Total*
KDDTrain$^+$	45927	52	995	11656	67343	125973
KDDTest$^+$	7458	67	2887	2422	9710	22544

2.2.3 DARPA 2000 Dataset

A DARPA[2] evaluation project [18] targeted the detection of complex attacks that contain multiple steps. Two attack scenarios were simulated in the DARPA 2000 evaluation contest, namely Lincoln Laboratory scenario DDoS (LLDOS) 1.0 and LLDOS 2.0. To achieve variations, these two attack scenarios were carried out over several network and audit scenarios. These sessions were grouped into four attack phases: (a) probing, (b) breaking into the system by exploiting vulnerability, (c) installing DDoS software for the compromised system, and (d) launching DDoS attack against another target. LLDOS 2.0 is different from LLDOS 1.0 in that attacks are more stealthy and thus harder to detect. Since this dataset contains multi-stage attack scenarios, it is also commonly used for evaluation of alert correlation techniques.

2.2.4 DEFCON Dataset

The DEFCON[3] dataset is another commonly used dataset for evaluation of IDSs [11]. It contains network traffic captured during a hacker competition called Capture The Flag (CTF), in which competing teams are divided into two groups: *attackers* and *defenders*. The traffic produced during CTF is very different from real world network traffic since it contains only intrusive traffic without any normal background traffic. Due to this limitation, DEFCON dataset has been found useful only in evaluating alert correlation techniques.

2.2.5 CAIDA Dataset

CAIDA[4] collects many different types of data and makes them available to the research community. CAIDA datasets [8] are very specific to particular events or attacks. Most of its longer

[2] http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html

[3] http://cctf.shmoo.com/data/

[4] http://www.caida.org/home/

traces are anonymized backbone traces without their payload. The CAIDA DDoS 2007 attack dataset contains one hour of anonymized traffic traces from DDoS attacks on August 4, 2007, which attempted to consume a large amount of network resources when connecting to Internet servers. The traffic traces contain only attack traffic to the victim and responses from the victim with 5 minutes split form. All traffic traces are in pcap (tcpdump) format. The creators removed non-attack traffic as much as possible when creating the CAIDA DDoS 2007 dataset.

2.2.6 LBNL Dataset

LBNL's internal enterprise traffic traces are full header network traces without payload [23]. This dataset suffers from heavy anonymization to the extent that scanning traffic was extracted and separately anonymized to remove any information which could identify individual IPs. The background and attack traffic in the LBNL dataset are described below.

- *LBNL background traffic:* This dataset can be obtained from the Lawrence Berkeley National Laboratory (LBNL) in the US. Traffic in this dataset is comprised of packet level incoming, outgoing and internally routed traffic streams at the LBNL edge routers. Traffic was anonymized using the $tcpmkpub$ tool [35]. The main applications observed in the internal and external traffic are Web, email and name services. Other applications like Windows services, network file services and backup were used by internal hosts. The details of each service and information on each packet and other relevant description are given in [34]. The background network traffic statistics of the LBNL dataset are given in Table 4.

- *LBNL attack traffic:* This dataset identifies attack traffic by isolating scans in aggregate traffic traces. Scans are identified by flagging those hosts which unsuccessfully probe more than 20 hosts, out of which 16 hosts are probed in ascending or descending IP order [35]. Malicious traffic mostly consists of failed incoming TCP SYN requests, i.e., TCP port scans targeted towards LBNL hosts. However, there are also some outgoing TCP scans in the dataset. Most UDP traffic observed in the data (incoming and outgoing) is comprised of successful connections, i.e., host replies for the received UDP flows. Clearly, the attack rate is significantly lower than the background traffic rate. Details of the attack traffic in this dataset are shown in Table 4. Complexity and privacy were two main reservations of the participants of the endpoint data collection study. To address these reservations, the dataset creators developed a custom multi-threaded MS Windows tool using the $Winpcap$ API [7] for data collection. To reduce packet logging complexity at the endpoints, they only logged very elementary session-level information (bidirectional communication between two IP addresses on different ports) for the TCP and UDP packets. To ensure user privacy, an anonymization policy was used to anonymize all traffic instances.

2.2.7 Endpoint Dataset

The background and attack traffic for the endpoint datasets are described below.

- *Endpoint background traffic:* In the endpoint context, we see in Table 5 that home computers generate significantly higher traffic volumes than office and university computers because: (*i*) they are generally shared between multiple users, and (*ii*) they run peer-to-peer and multimedia applications. The large traffic volumes of home computers are also evident from their high mean number of sessions per second. To generate attack traffic, developers on infect Virtual Machines (VMs) at the endpoints with different malware, viz., Zotob.G, Forbot-FU, Sdbot-AFR, Dloader-NY, So-Big.E@mm, MyDoom.A@mm, Blaster, Rbot-AQJ, and RBOT.CCC. Details of the malware can be found in [42]. Characteristics of the attack traffic in this dataset are given in Table 6. These malwares have diverse scanning rates and attack ports or applications.

- *Endpoint attack traffic:* The attack traffic logged at the endpoints is mostly comprised of outgoing port scans. Note that this is the opposite of the LBNL dataset, in which most attack traffic is inbound. Moreover, the attack traffic rates at the endpoints are generally much higher than the background traffic rates of the LBNL datasets. This diversity in attack direction and rates provides a sound basis for performance comparison among scan detectors. For each malware, attack traffic of 15 minute duration was inserted in the background traffic for each endpoint at a random time instance. This operation was repeated to insert 100 non-overlapping attacks of each worm inside each endpoint's background traffic.

2.3 Real-life Datasets

We discuss three real-life datasets created by collecting network traffic on several consecutive days. The details include both normal as well as attack traffic in appropriate proportions in the authors' respective campus networks (i.e., testbeds).

2.3.1 UNIBS Dataset

The UNIBS packet traces [45] were collected on the edge router of the campus network of the University of Brescia in Italy, on three consecutive working days. The dataset includes traffic captured or collected and stored using 20 workstations, each running the GT (Ground Truth) client daemon. The dataset creators collected the traffic by running tcpdump on the faculty router, which was a dual Xeon Linux box that connected the local network to the Internet through a dedicated 100Mb/s uplink. They captured and stored the traces on a dedicated disk of a workstation connected to the router through a dedicated ATA controller.

2.3.2 ISCX-UNB Dataset

The ISCX-UNB dataset [37] is built on the concept of profiles that include the details of intrusions. The datasets were col-

Table 4: Background and attack traffic information for the LBNL datasets

Date	*Duration (mins)*	*LBNL hosts*	*Remote hosts*	*Background traffic rate (packet/sec)*	*Attack traffic rate (packet/sec)*
10/04/2004	10 min	4,767	4,342	8.47	0.41
12/15/2004	60 min	5,761	10,478	3.5	0.061
12/16/2004	60 min	5,210	7,138	243.83	72

Table 5: Background traffic information for four endpoints with high and low rates

Endpoint ID	*Endpoint type*	*Duration (months)*	*Total sessions*	*Mean session rate (/sec)*
3	Home	3	3,73,009	1.92
4	home	2	4,44,345	5.28
6	University	9	60,979	0.19
10	University	13	1,52,048	0.21

Table 6: Endpoint attack traffic for two high and two low-rate worms

Malware	*Release Date*	*Avg. Scan rate (/sec)*	*Port (s) Used*
Dloader-NY	Jul 2005	46.84 sps	TCP 1,35,139
Forbot-FU	Sept 2005	32.53 sps	TCP 445
Rbot-AQJ	Oct 2005	0.68 sps	TCP 1,39,769
MyDoom-A	Jan 2006	0.14 sps	TCP 3127-3198

lected using a real-time testbed by incorporating multi-stage attacks. It uses two profiles - α and β - during the generation of the datasets. α profiles are constructed using the knowledge of specific attacks and β profiles are built using the filtered traffic traces. Real packet traces were analyzed to create α and β profiles for agents that generate real-time traffic for HTTP, SMTP, SSH, IMAP, POP3 and FTP protocols. Various multistage attack scenarios were explored to generate malicious traffic.

2.3.3 KU Dataset

The Kyoto University dataset[5] is a collection of network traffic data obtained from honeypots. The raw dataset obtained from the honeypot system consisted of 24 statistical features, out of which 14 significant features were extracted [38]. The dataset developers extracted 10 additional features that could be used to investigate network events inside the university more effectively. The initial 14 features extracted are similar to those in the KDDcup99 datasets. Only 14 conventional features were used during training and testing.

2.4 Discussion

The datasets described above are valuable assets for the intrusion detection community. However, the benchmark datasets suffer from the fact that they are not good representatives of real world traffic. For example, the DARPA dataset has been questioned about the realism of the background traffic [27, 29] because it is synthetically generated. In addition to the difficulty of simulating real time network traffic, there are additional challenges in IDS evaluation [30]. These include difficulties in collecting attack scripts and victim software, differing requirements for testing signature based vs. anomaly based IDSs, and host-based vs. network based IDSs. In addition to these, we make the following observations based on our analysis.

- Most datasets are not labelled properly due to non-availability of actual attack information. These include KDDcup99, UNIBS, Endpoint and LBNL datasets.
- The proportion of normal and attack ratios are different in different datasets [21, 38, 45].
- Several existing datasets [21, 23, 38] have not been maintained or updated to reflect recent trends in network traffic by incorporating evolved network attacks.
- Most existing datasets are annonymized [8, 18] due to potential security risks to an organization. They do not share their raw data with researchers.
- Several datasets [8, 23, 18, 45] lack in traffic features. They have only raw traffic traces but it is important to extract relevant traffic features for individual attack identification.

3 Real-life Datasets Generation

As noted above, the generation of an unbiased real-life intrusion dataset incorporating a large number of real world attacks is important to evaluate network anomaly detection methods and systems. In this paper, we describe the generation of three real-life network intrusion datasets[6] including (a) a TUIDS (Tezpur University Intrusion Detection System) intrusion dataset, (b) a TUIDS coordinated scan dataset, and (c) a TUIDS DDoS dataset at both packet and flow levels [16]. The resulting details and supporting infrastructure is discussed in the following subsections.

3.1 Testbed Network Architecture

The TUIDS testbed network consists of 250 hosts, 15 L2 switches, 8 L3 switches, 3 wireless controllers, and 4 routers that compose 5 different networks inside the Tezpur University campus. The architectures of the TUIDS testbed and TUIDS testbed for DDoS dataset generation are given in Figures 2 and

[5] http://www.takakura.com/kyoto_data

[6] http://agnigarh.tezu.ernet.in/∼dkb/resources.html

3, respectively. The hosts are divided into several VLANs, each VLAN belonging to an L3 switch or an L2 switch inside the network. All servers are installed inside a DMZ[7] to provide an additional layer of protection in the security system of an organization.

3.2 Network Traffic Generation

To generate real time normal and attack traffic, we configured several hosts, workstations, and servers in the TUIDS testbed network. The network consists of 6 interconnected Ubuntu 10.10 workstations. On each workstation, we have installed several severs including a network file server (Samba), a mail sever (Dovecot), a telnet server, an FTP server, a Web server, and an SQL sever with PHP compatibility. We also installed and configured 4 Windows Servers 2003 to exploit a diverse set of known vulnerabilities against the testbed environment. Servers and their services running on our testbed are summarized in Table 7.

Table 7: Servers and their services running on the testbed network

Server	*Operating system*	*Services*	*Provider*
Main Server	Ubuntu 10.10	Web, eMail	Apache 2.4.3, Dovecot 2.1.14
Network File Server	Ubuntu 10.10	Samba	Samba 4.0.2
Telnet Server	Ubuntu 10.10	Telnet	telnet-0.17-36bulid1
FTP Server	Ubuntu 10.10	ftp	vsFTPd 2.3.0
Windows Server	Windows Server 2003	Web	IIS v7.5
MySQL Server	Ubuntu 10.10	database	MySQL 5.5.30

The normal network traffic is generated based on the day-to-day activities of users and especially generated traffic from configured servers. It is important to generate different types of normal traffic. So, we capture traffic from students, faculty members, system administrators, and office staff on different days within the University. The attack traffic is generated by launching attacks within the testbed network in three different subsets, viz., a TUIDS intrusion dataset, a coordinated scan dataset and a DDoS dataset. The attacks launched in the generation of these real-life datasets are summarized in Table 8.

As seen in the table above, 22 distinct attack types (1-22 in Table 8) were used to generate the attack traffic for the TUIDS intrusion dataset; six attacks (17-22 in Table 8) were used to generate the attack traffic for the coordinated scan dataset and finally six attacks (23-28 in Table 8) were used to generate the attack traffic for a DDoS dataset with combination of TCP, UDP and ICMP protocols.

[7]Demilitarized zone is a network segment located between a secured local network and unsecured external networks (Internet). A DMZ usually contains servers that provide services to users on the external network, such as Web, mail and DNS servers that are hardened systems. Typically, two firewalls are installed to form the DMZ.

Table 8: List of real time attacks and their generation tools

Attack name	*Generation tool*	*Attack name*	*Generation tool*
1.bonk	targa2.c	15.linux-icmp	linux-icmp.c
2.jolt	targa2.c	16.syn-flood	synflood.c
3.land	targa2.c	17.window-scan	nmap/rnmap
4.saihyousen	targa2.c	18.syn-scan	nmap/rnmap
5.teardrop	targa2.c	19.xmasstree-scan	nmap/rnmap
6.newtear	targa2.c	20.fin-scan	nmap/rnmap
7.1234	targa2.c	21.null-scan	nmap/rnmap
8.winnuke	targa2.c	22.udp-scan	nmap/rnmap
9.oshare	targa2.c	23.syn-flood(DDoS)	LOIC
10.nestea	targa2.c	24.rst-flood(DDoS)	Trinity v3
11.syndrop	targa2.c	25.udp-flood(DDoS)	LOIC
12.smurf	smurf4.c	26.ping-flood(DDoS)	DDoS ping v2.0
13.opentear	opentear.c	27.fraggle udp-flood(DDoS)	Trinoo
14.fraggle	fraggle.c	28.smurf icmp-flood(DDoS)	TFN2K

3.3 Attack Scenarios

The attack scenarios start with information gathering techniques collecting target network IP ranges, identities of name servers, mail servers and user e-mail accounts, etc. This is achieved by querying the DNS for resource records using network administrative tools like nslookup and dig. We consider six attack scenarios when collecting real time network traffic for dataset generation.

3.3.1 *Scenario 1*: Denial of Service Using Targa

This attack scenario is designed to perform attacks on a target using the targa[8] tool until it is successful. Targa is a very powerful tool to quickly damage a particular network belonging to an organization. We ran targa by specifying different parameter values such as IP ranges, attacks to run and number of times to repeat the attack.

3.3.2 *Scenario 2*: Probing Using nmap

In this scenario, we attempt to acquire information about the target host and then launch the attack by exploiting the vulnerabilities found using the nmap[9] tool. Examples of attacks that can be launched by this method are syn-scan and ping-sweep.

3.3.3 *Scenario 3*: Coordinated Scan Using rnmap

This scenario starts with a goal to perform coordinated port scans to single and multiple targets. Tasks are distributed

[8]http://packetstormsecurity.com/
[9]http://nmap.org/

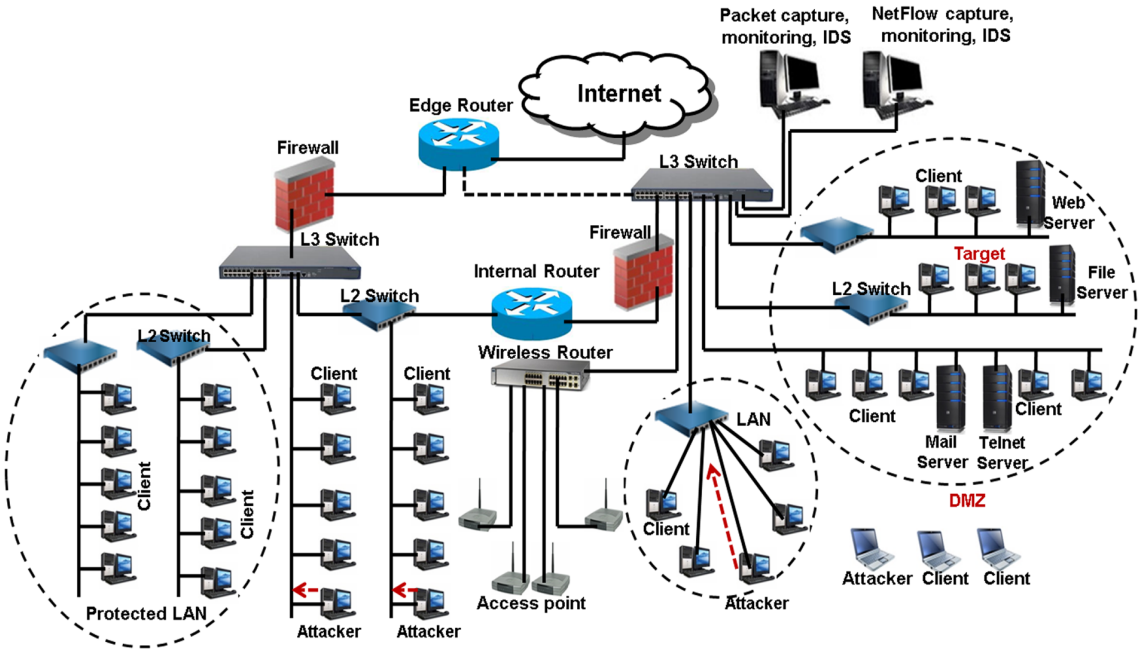

Figure 2: Testbed network architecture used during TUIDS dataset generation

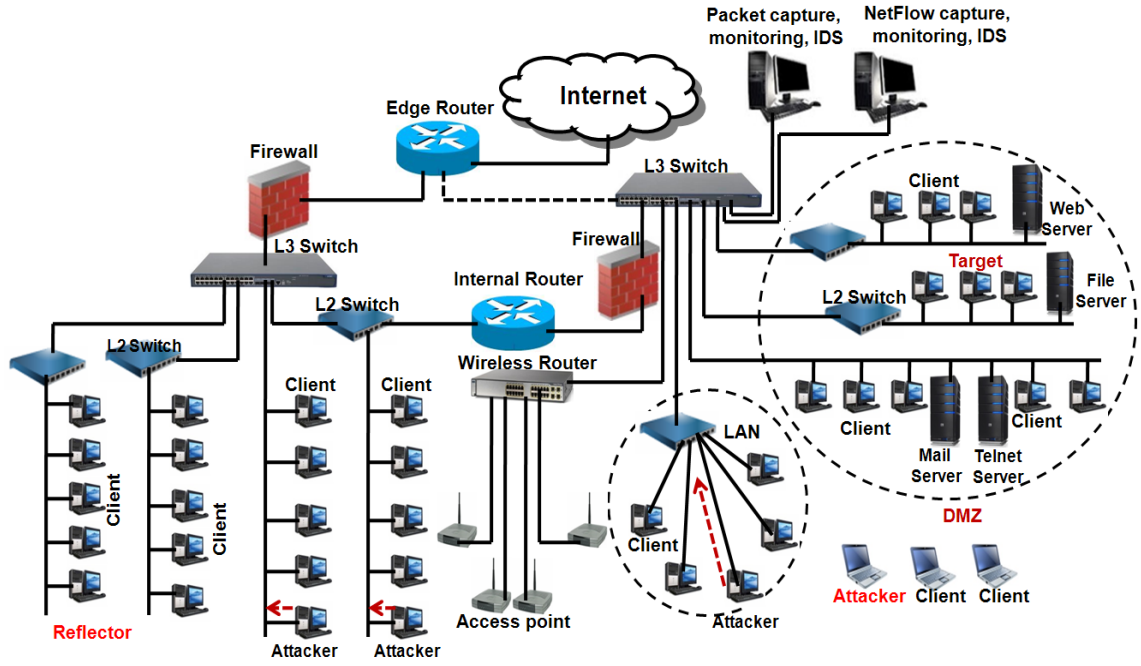

Figure 3: Testbed network architecture used during DDoS dataset generation

among multiple hosts for individual actions which may be synchronized. We use the rnmap[10] tool to launch coordinated scans in our testbed network during the collection of traffic.

3.3.4 *Scenario 4*: User to Root Using Brute Force ssh

These attacks are very common against networks as they tend to break into accounts with weak username and password combinations. This attack has been designed with the goal of acquiring an SSH account by running a brute force dictionary attack against our central server. We use the brutessh[11] tool and a customized dictionary list. The dictionary consists of over 6100 alphanumeric entries of varying length. We executed the attack for 60 minutes, during which superuser credentials were returned from the server. This ID and password combination was used to download other users' credentials immediately.

[10]http://rnmap.sourceforge.net/

[11]http://www.securitytube-tools.net/

3.3.5 *Scenario 5*: Distributed Denial of Service Using Agent-handler Network

This scenario mainly attempts to exploit an agent handler network to launch the DDoS attack in the TUIDS testbed network. The agent-handler network consists of clients, handlers and agents. The handlers are software packages that are used by the attacker to communicate indirectly with the agents. The agent software exists in compromised systems that will eventually carry out the attack on the victim system. The attacker may communicate with any number of handlers, thus making sure that the agents are up and running. We use Trinity v3, TFN2K, Trinoo, and DDoS ping 2.0 to launch the attacks in our testbed.

3.3.6 *Scenario 6*: Distributed Denial of Service Using IRC Botnet

Botnets are an emerging threat to all organizations because they can compromise a network and steal important information and distribute malware. Botnets combine individual malicious behaviors into a single platform by simplifying the actions needed to be performed by users to initiate sophisticated attacks against computers or networks around the world. These behaviors include coordinated scanning, DDoS activities, direct attacks, indirect attacks and other deceitful activities taking place across the Internet.

The main goal of this scenario is to perform distributed attacks using infected hosts on the testbed. An Internet Relay Chat (IRC) bot network allow users to create public, private and secret channels. For this, we use a LOIC[12], an IRC-based DDoS attack generation tool. The IRC systems have several other significant advantages for launching DDoS attacks. Among the three important benefits are (i) they afford a high degree of anonymity, (ii) they are difficult to detect, and (iii) they provide a strong, guaranteed delivery system. Furthermore, the attacker no longer needs to maintain a list of agents, since he can simply log on to the IRC server and see a list of all available agents. The IRC channels receive communications from the agent software regarding the status of the agents (i.e., up or down) and participate in notifying the attackers regarding the status of the agents.

3.4 Capturing Traffic

The key tasks in network traffic monitoring are lossless packet capturing and precise timestamping. Therefore, software or hardware is required with a guarantee that all traffic is captured and stored. The real network traffic is captured using the *Libpcap* [19, 20] library, an open source C library offering an interface for capturing link-layer frames over a wide range of system architectures. It provides a high-level common Application Programming Interface (API) to the different packet capture frameworks of various operating systems. The offered abstraction layer allows programmers to rapidly develop highly portable applications. A hierarchy of network traffic capturing components is given in Figure 4 [10].

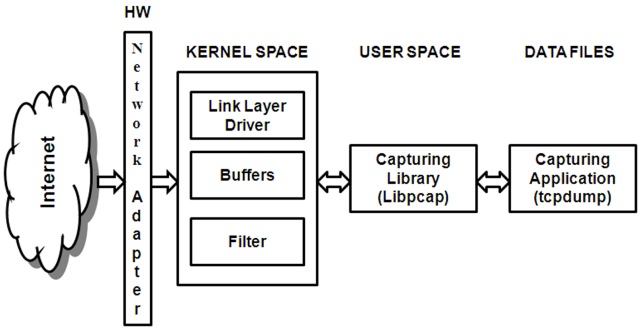

Figure 4: Hierarchy of Network Traffic Capturing Components

Libpcap defines a common standard format for files in which captured frames are stored, also known as the *tcpdump* format, currently a de facto standard used widely in public network traffic archives. Modern kernel-level capture frameworks on UNIX operating systems are mostly based on the BSD (or Berkeley) Packet Filter (BPF) [28]. The BPF is a software device that taps network interfaces, copying packets into kernel buffers and filtering out unwanted packets directly in interrupt context. Definitions of packets to be filtered can be written in a simple human readable format using Boolean operators and can be compiled into a pseudo-code to be passed to the BPF device driver by a system call. The pseudo-code is interpreted by the BPF Pseudo-Machine, a lightweight, high-performance, state machine specifically designed for packet filtering. *Libpcap* also allows programmers to write applications that transparently support a rich set of constructs to build detailed filtering expressions for most network protocols. A few *Libpcap* system calls can be read directly from user's command line, compile into pseudo-code and passed it to the Berkeley Packet Filter. *Libpcap* and the BPF interact to allow network packet data to traverse several layers to finally be processed and transformed into capture files (i.e., *tcpdump* format) or samples for statistical analysis.

With the goal of preparing both packet and flow level datasets, we capture both packet and NetFlow traffic from different locations in the TUIDS testbed. The capturing period started at 08:00:05 am on Monday February 21, 2011 and continuously ran for an exact duration of seven days, ending at 08:00:05 am on Sunday February 27th. Attacks were executed during this period for the TUIDS intrusion and the coordinated scan datasets. DDoS traffic was also collected for the same amount of time but during October, 2012 with several variations of real time DDoS attacks. Figure 5 illustrates the protocol composition and the average throughput during the last hour of data capture for the TUIDS intrusion dataset.

We use a tool known as lossless gigabit remote packet capture with Linux (Gulp[13]) for capturing packet level traffic in a mirror port as shown in the TUIDS testbed architecture. Gulp reads packets directly from the network card and writes to the disk at a high rate of packet capture without dropping packets. For low-rate packets, Gulp flushes the ring buffer if it has not written anything in the last second. Gulp writes into

[12]http://sourceforge.net/projects/loic/

[13]http://staff.washington.edu/corey/gulp/

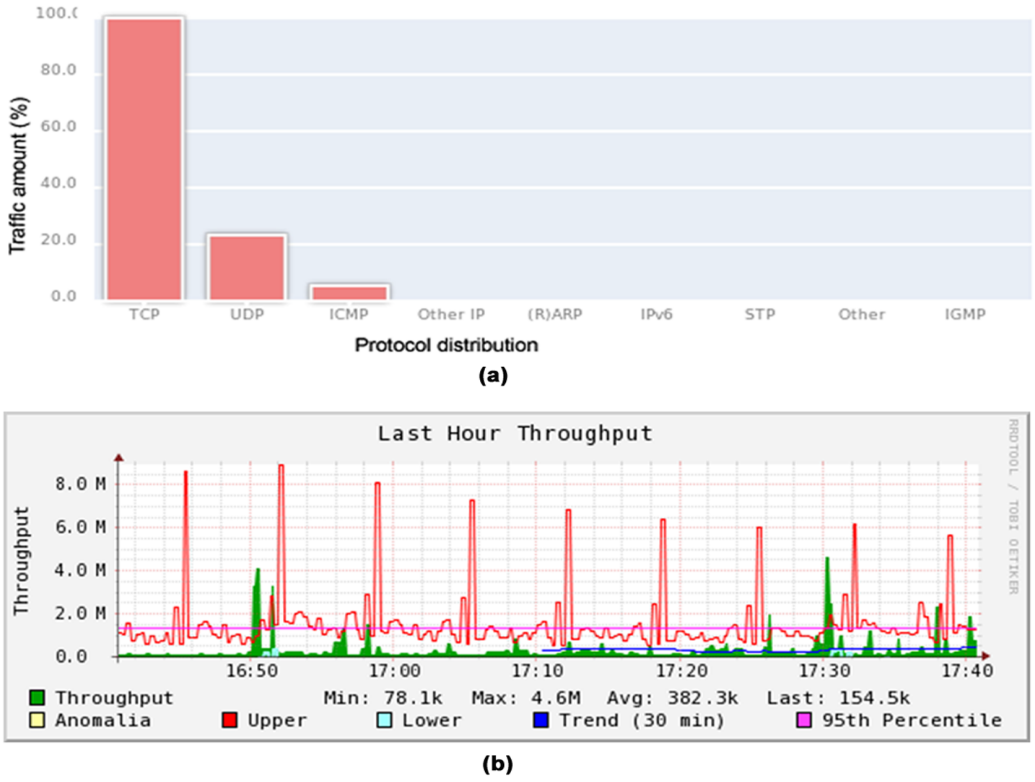

Figure 5: (a) composition of protocols and (b) average throughput during last hour of data capture for the TUIDS intrusion dataset seen in our lab's traffic

even block boundaries for excellent writing performance when the data rate increases. It stops filling the ring buffer after receiving an interrupt but it would write into the disk whatever remains in the ring buffer.

In the last few years, NetFlow has become the most popular approach for IP network monitoring, since it helps cope with scalability issues introduced by increasing network speeds. Now major vendors offer flow-enabled devices, such as Cisco routers with NetFlow. A NetFlow is a stream of packets that arrives on a source interface with the key values shown in Figure 6. A key is an identified value for a field within the packet. Cisco routers have NetFlow features that can be enabled to generate NetFlow records. The principle of NetFlow is as follows: When the router receives a packet, its NetFlow module scans the source IP address, the destination IP address, the source port number, the destination port number, the protocol type, the type of service (ToS) bit in the IP header, and the input or output interface number on the router of the IP packet to judge whether it belongs to a NetFlow record that already exists in the cache. If so, it updates the NetFlow record; otherwise, a new NetFlow record is generated in the cache. The expired NetFlow records in the cache are exported periodically to a destination IP address using a UDP port.

For capturing the NetFlow traffic, we need a NetFlow collector that can listen to a specific UDP port for getting traffic. The NetFlow collector captures exported traffic from multiple routers and periodically stores it in summarized or aggregated format into a round robin database (RRD). The following tools are used to capture and visualize the NetFlow traffic.

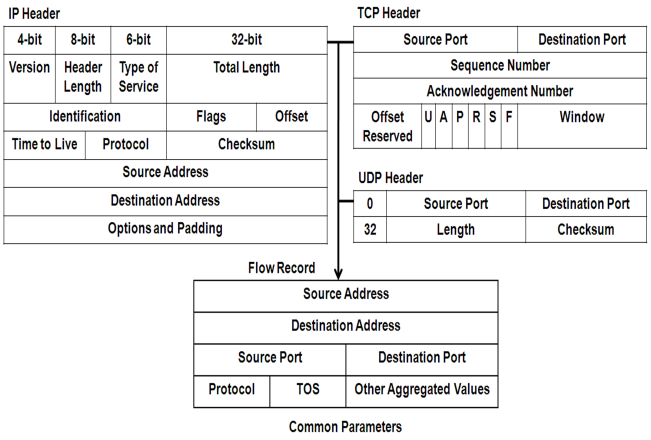

Figure 6: Common NetFlow parameters

(a) *NFDUMP*: This tool captures and displays NetFlow traffic. All versions of nfdump support NetFlow v5, v7, and v9. nfcapd is a NetFlow capture daemon that reads the NetFlow data from the routers and stores the data into files periodically. It automatically rotates files every n minutes (by default it is 5 minutes). We need one nfcapd process for each NetFlow stream. Nfdump reads the NetFlow data from the files stored by nfcapd. The syntax is similar to that of tcpdump. Nfdump displays NetFlow data and can create top N statistics for flows based on the parameters selected. The main goal is to analyze NetFlow data from the past as well as to track interesting traffic

Table 9: Parameters identified for packet level data

Sl. No.	*Parameter name*	*Description*
1	Time	Time since occurrence of first frame
2	Frame No	Frame number
3	Frame Len	Length of a frame
4	Capture Len	Capture length
5	TTL	Time to live
6	Protocol	Protocols (such as, TCP, UDP, ICMP etc.)
7	Src IP	Source IP address
8	Dst IP	Destination IP address
9	Src port	Source port
10	Dst port	Destination port
11	Len	Data length
12	Seq No	Sequence number
13	Header Len	Header length
14	CWR	Congestion window record
15	ECN	Explicit congestion notification
16	URG	Urgent TCP flag
17	ACK	Acknowledgement flag
18	PSH	Push flag
19	RST	Reset flag
20	SYN	TCP syn flag
21	FIN	TCP fin flag
22	Win Size	Window Size
23	MSS	Maximum segment size

patterns continuously from high speed networks. The amount of time from the past is limited only by the disk space available for all NetFlow data.

Nfdump has four fixed output formats: *raw*, *line*, *long* and *extended*. In addition, the user may specify any desired output format by customizing it. The default format is line, unless specified. The raw format displays each record in multiple lines and prints any available information in the traffic record. (b) *NFSEN*: NfSen is a graphical Web based front end tool for visualization of NetFlow traffic. NfSen facilitates the visualization of several traffic statistics, e.g., flow-wise statistics for various features, navigation through the NetFlow traffic, processes within a time span and continuous profiles. It can also add own plugins to process NetFlow traffic in a customized manner at a regular time interval.

Normal traffic is captured by restricting it to the internal networks, where 80% of the hosts are connected to the router, including wireless networks. We assume that normal traffic follows the normal probability distribution. Attack traffic is captured as we launch various attacks in the testbed for a week. For DDoS attacks, we used packet-craft[14] to generate customized packets. Figures 7 and 8 show the number of flows per second and also the protocol-wise distribution of flows during the capturing period, respectively.

3.5 Feature Extraction

We use wireshark and Java routines for filtering unwanted packets (such as packets with routing protocols, and packets with application layer protocols) as well as irrelevant information from captured packets. Finally, we retrieve all relevant information from each packet using Java routines and store it in comma separated form in a text file. The details of parameters identified for packet level data are shown in Table 9.

We developed several C routines and used them for filtering NetFlow data and for extracting features from the captured data. A detailed list of parameters identified for flow level data

[14]http://www.packet-craft.net/

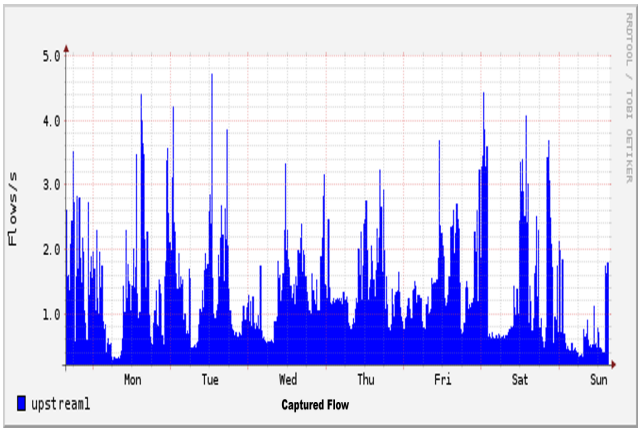

Figure 7: Number of flows per second in TUIDS intrusion datasets during the capture period

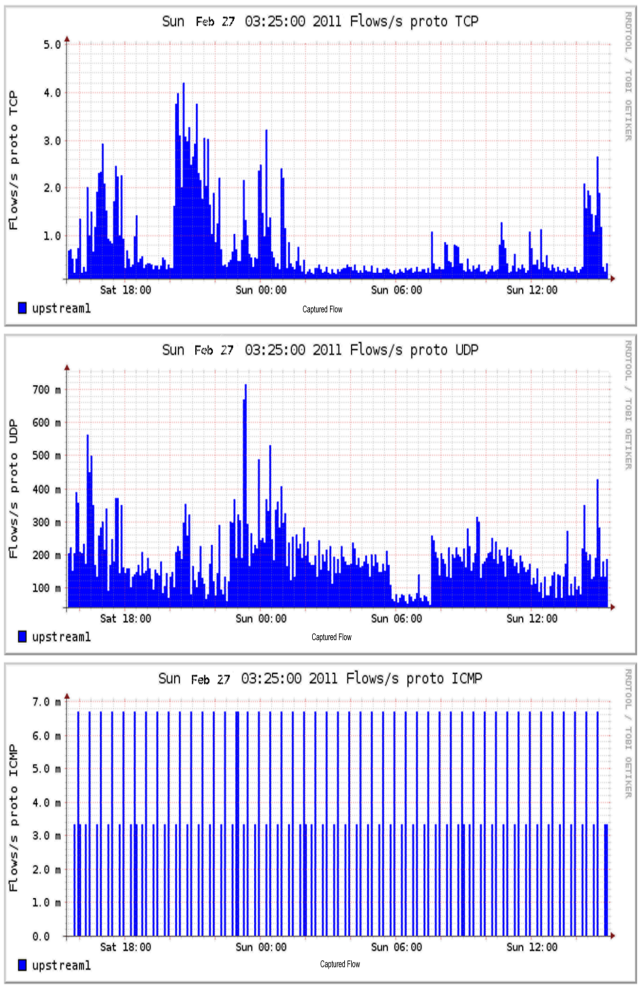

Figure 8: Protocol-wise distribution of flow per second in TUIDS intrusion dataset during the capture period

is given in Table 10.

We capture, preprocess and extract various features in both packet and flow level network traffic. We introduce a framework for fast distributed feature extraction from raw network traffic, correlation computation and data labelling, as shown in Figure 9. We extract four types of features: *basic*, *content* based, *time* based and *connection* based, from the raw network traffic. We use $T = 5\ seconds$ as the time window for extraction of both time based and connection based traffic features. S_1 and S_2 are servers used for preprocessing, attack labelling and profile generation. WS_1 and WS_2 are high-end workstations used for basic feature extraction and merging packet and NetFlow traffic. $N_1, N_2, \cdots N_6$ are independent nodes used for protocol specific feature extraction. The lists of extracted features at both packet and flow levels for the intrusion datasets are presented in Table 11 and Table 12, respectively. The list of features available in the KDDcup99 intrusion dataset is also shown in Table 13.

Table 10: Parameters identified for flow level data

Sl. No.	*Parameter name*	*Description*
1	flow-start	Starting of flow
2	Duration	Total life time of a flow
3	Proto	Protocol, i.e., TCP, UDP, ICMP etc.
3	Src-IP	Source IP address
4	Src-port	Source port
5	Dest-IP	Destination IP address
6	Dest-port	Destination port
7	Flags	TCP flags
8	ToS	Type of Service
9	Packets	Packets per flow
10	Bytes	Bytes per flow
11	Pps	Packet per second
12	Bps	Bit per second
13	Bpp	Byte per packet

3.6 Data Processing and Labelling

As reported in the previous section, traffic features are extracted separately (within a time interval). So, it is important to correlate each feature (i.e., basic, content based, time based, and connection based) to a time interval. Once correlation is performed for both packet and flow level traffic, labelling of each feature data as normal or anomalous is important. The labelling process enriches the feature data with information such as (i) the type and structure of malicious or anomalous data, and (ii) dependencies among different isolated malicious activities. The correlation and labelling of each feature traffic as normal or anomalous is made using Algorithm 1. $F = \{\alpha, \beta, \gamma, \delta\}$ is the set of extracted features, where α is the set of basic features, β is the set of content-based features, γ is the set of time-based features and δ is the set of connection-based features. Both normal and anomalous traffic are collected separately in several sessions within a week. We remove normal traffic from anomalous traces as much as possible.

The overall traffic composition with protocol distribution in the generated datasets is summarized in Table 14. The traffic includes the TUIDS intrusion dataset, the TUIDS coordinated scan dataset and the TUIDS DDoS dataset. The final labelled feature datasets for each category with the distribution of normal and attack information are summarized in Table 15. All datasets are prepared at both packet and flow levels and presented in terms of training and testing in Table 15.

Algorithm 1 : FC and labelling (F)

Input: extracted feature set, $F = \{\alpha, \beta, \gamma, \delta\}$
Output: correlated and labelled feature data, D

initialize D
call $FeatureExtraction()$, $F \leftarrow \{\alpha, \beta, \gamma, \delta\}$, ▷ the procedure FeatureExtraction() extracts the features separately for all cases
for $i \leftarrow 1$ to $|N|$ **do** ▷ N is the total traffic instances
  for $i \leftarrow 1$ to $|F|$ **do** ▷ F is the total traffic features
    if $(unique(src.ip \wedge dst.ip))$ **then**
      store $D[ij] \leftarrow \alpha_{ij}, \beta_{ij}$
    end if
    if $((T == 5s) \wedge (LnP == 100))$ **then** ▷ T is the time window, LnP is the last n packets
      Store $D[ij] \leftarrow \gamma_{ij}, \delta ij$
    end if
  end for
  $D[ij] \leftarrow \{normal, attack\}$ ▷ label each traffic feature instance based on the duration of the collected traffic
end for

3.7 Comparison with Other Public Datasets

Several real network traffic traces are readily available to the research community as reported in Section 2. Although these traffic traces are invaluable to the research community most if not all, fail to satisfy one or more requirements described in Section 1. This paper is mostly distinguished by the fact that the issue of data generation is approached from what other datasets have been unable to provide, for the network security community. It attempts to resolve the issues seen in other datasets by presenting a systematic approach to generate real-life network intrusion datasets. Table 16 summarizes a comparison between the prior datasets and the dataset generated through the application of our systematic approach to fulfill the principal objectives outlined for qualifying dataset.

Most datasets are unlabelled as labelling is labor-intensive and requires a comprehensive search to tag anomalous traffic. Although an IDS helps by reducing the work, there is no guarantee that all anomalous activity is labelled. This has been a major issue with all datasets and one of the reasons behind the post insertion of attack traffic in the DARPA 1999 dataset, so that anomalous traffic can be labelled in a deterministic manner. Having seen the inconsistencies produced by traffic merging, this paper has adopted a different approach to provide the same level of deterministic behavior with respect to anomalous traffic by conducting anomalous activity within the capturing period using available network resources. Through the use of logging, all ill-intended activity can be effectively labelled.

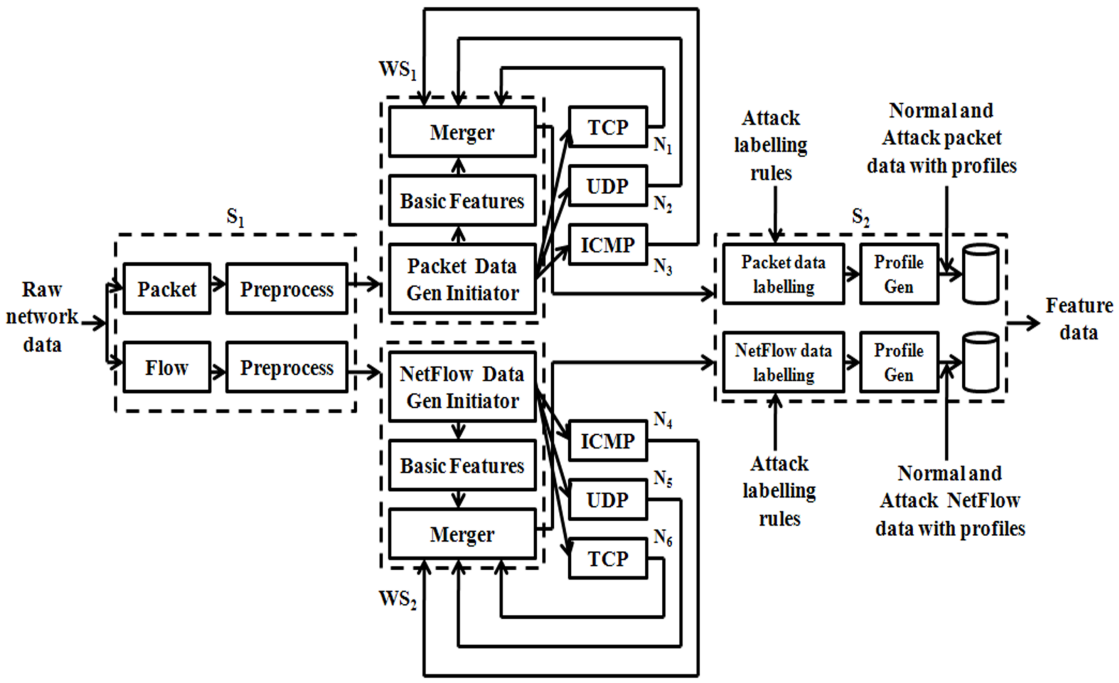

Figure 9: Fast distributed feature extraction, correlation and labelling framework

The extent and scope of network traffic capture become relevant in situations where the information contained in the traces may breach the privacy of individuals or organizations. In order to prevent privacy issues, almost all publicly available datasets remove any identifying information such as payload, protocol, destination and flags. In addition, the data is anonymized where necessary header information is cropped or flows are just summarized.

In addition to anomalous traffic, traces must contain background traffic. Most captured datasets have little control over the anomalous activities included in the traces. However, a major concern with evaluating anomaly based detection approaches is the requirement that anomalous traffic must be present at a certain scale. Anomalous traffic also tends to become outdated with the introduction of more sophisticated attacks. So, we have generated more up-to-date datasets that reflect the current trends and are tailored to evaluate certain characteristics of detection mechanisms which are unique to themselves.

As discussed earlier, several datasets are available for evaluating an IDS. Network intrusion detection researchers evaluate detection methods using intrusion datasets to demonstrate how their methods can handle recent attacks and network environments. We have used our datasets to evaluate several network intrusion detection methods. Some of them are outlier-based network anomaly detection approach (NADO) [4], an unsupervised method [3, 6], an adaptive outlier-based coordinated scan detection approach (AOCD) [5], and a multi-level hybrid IDS (MLH-IDS) [15]. We found better results in almost all the experiments when we used TUIDS dataset in terms of false positive rate, true positive rate and F-measure.

3.8 Comparison with Other Relevant Work

Our approach differs from other works as follows.

- The NSL-KDD [32] dataset is an enhanced version of the KDDcup99 intrusion dataset prepared by Tavallaee et al. [43]. This dataset is too old to evaluate a modern detection method or a system that has been developed recently. It removes repeated traffic records from the old KDDcup99 dataset. In contrast, our datasets are prepared using diverse attack scenarios incorporating recent attacks. Our datasets contain both packet and flow level information that help detect attacks more effectively in high speed networks.

- Song et al. [39] prepared the KU dataset and used the dataset to evaluate an unsupervised network anomaly detection method. This dataset contains 17 different features at packet level only. In contrast, we present a systematic approach to generate real-life network intrusion datasets and prepared three different categories of datasets at both packet and flow levels.

- Like Shiravi et al. [37], our approach considers recently developed attacks and attacks on network layers when generating the datasets. Shiravi et al. concentrate mostly on application-layer attacks. They build profiles for different real-world attack scenarios and use them to generate traffic that follows the same behavior while generating the dataset at packet level. In comparison, we generate three different categories of datasets at both packet and flow levels for the research community to evaluate detection methods or systems. Since we have extracted more number of features at both packet and flow levels. Our

Table 11: List of packet level features in TUIDS intrusion dataset

Label/feature name	*Type*	*Description*
Basic features		
1. Duration	C	Length (number of seconds) of the connection
2. Protocol-type	D	Type of protocol, e.g., tcp, udp, etc.
3. Src-ip	C	Source host IP address
4. Dest-ip	C	Destination IP address
5. Src-port	C	Source host port number
6. Dest-port	C	Destination host port number
7. Service	D	Network service at the destination, e.g., http, telnet, etc.
8. num-bytes-src-dst	C	The number of data bytes flowing from source to destination
9. num-bytes-dst-src	C	The number of data bytes flowing from destination to source
10. Fr-no	C	Frame number
11. Fr-len	C	Frame length
12. Cap-len	C	Captured frame length
13. Head-len	C	Header length of the packet
14. Frag-off	D	Fragment offset: '1' for the second packet overwrite everything, '0' otherwise
15. TTL	C	Time to live: '0' discards the packet
16. Seq-no	C	Sequence number of the packet
17. CWR	D	Congestion window record
18. ECN	D	Explicit congestion notification
19. URG	D	Urgent TCP flag
20. ACK	D	Acknowledgement flag value
21. PSH	D	Push TCP flag
22. RST	D	Reset TCP flag
23. SYN	D	Syn TCP flag
24. FIN	D	Fin TCP flag
25. Land	D	1 if connection is from/to the same host/port; 0 otherwise
Content-based features		
26. Mss-src-dest-requested	C	Maximum segment size from source to destination requested
27. Mss-dest-src-requested	C	Maximum segment size from destination to source requested
28. Ttt-len-src-dst	C	Time to live length from source to destination
29. Ttt-len-dst-src	C	Time to live length from destination to source
30. Conn-status	C	Status of the connection (e.g., '1' for complete, '0' for reset)
Time-based features		
31. count-fr-dest	C	Number of frames received by unique destinations in the last T seconds from the same source
32. count-fr-src	C	Number of frames received from unique sources in the last T seconds from the same destination
33. count-serv-src	C	Number of frames from the source to the same destination port in the last T seconds
34. count-serv-dest	C	Number of frames from destination to the same source port in the last T seconds
35. num-pushed-src-dst	C	The number of pushed packets flowing from source to destination
36. num-pushed-dst-src	C	The number of pushed packets flowing from destination to source
37. num-SYN-FIN-src-dst	C	The number of SYN/FIN packets flowing from source to destination
38. num-SYN-FIN-dst-src	C	The number of SYN/FIN packets flowing from destination to source
39. num-FIN-src-dst	C	The number of FIN packets flowing from source to destination
40. num-FIN-dst-src	C	The number of FIN packets flowing from destination to source
Connection-based features		
41. count-dest-conn	C	Number of frames to unique destinations in the last N packets from the same source
42. count-src-conn	C	Number of frames from unique sources in the last N packets to the same destination
43. count-serv-srcconn	C	Number of frames from the source to the same destination port in the last N packets
44. count-serv-destconn	C	Number of frames from the destination to the same source port in the last N packets
45. num-packets-src-dst	C	The number of packets flowing from source to destination
46. num-packets-dst-src	C	The number of packets flowing from destination to source
47. num-acks-src-dst	C	The number of acknowledgement packets flowing from source to destination
48. num-acks-dst-src	C	The number of acknowledgement packets flowing from destination to source
49. num-retransmit-src-dst	C	The number of retransmitted packets flowing from source to destination
50. num-retransmit-dst-src	C	The number of retransmitted packets flowing from destination to source
C-Continuous, D-Discrete		

datasets will help to identify individual attacks in more effectively in high speed networks.

4 Observations and Conclusion

Several questions may be raised with respect to what constitutes a perfect dataset when dealing with the datasets generation task. These include qualities of normal, anomalous or realistic traffic included in the dataset. We provide a path and a template to generate a dataset that simultaneously exhibits the appropriate levels of normality, anomalousness and realism while avoiding the various weak points of currently available datasets, pointed out earlier. Quantitative measurements can be obtained only when specific methods are applied to the dataset.

The following are the major observations and requirements when generating an unbiased real-life dataset for intrusion detection.

- The dataset should not exhibit any unintended property in both normal and anomalous traffic.
- The dataset should be labelled properly.
- The dataset should cover all possible current network scenarios.
- The dataset should be entirely nonanonymized.
- In most benchmark datasets, the two basic assumptions described in Section 1 are valid but this bias should be avoided as much as possible.

Table 12: List of flow level features in TUIDS intrusion dataset

Label/feature name	*Type*	*Description*
Basic features		
1. Duration	C	Length (number of seconds) of the flow
2. Protocol-type	D	Type of protocol, e.g., TCP, UDP, ICMP
3. Src-ip	C	Source host IP address
4. Dest-ip	C	Destination IP address
5. Src-port	C	Source host port number
6. Dest-port	C	Destination host port number
7. ToS	D	Type of service
8. URG	D	TCP urgent flag
9. ACK	D	TCP acknowledgement flag
10. PSH	D	TCP push flag
11. RST	D	TCP reset flag
12. SYN	D	TCP SYN flag
13. FIN	D	TCP FIN flag
14. Src-bytes	C	Number of data bytes transfered from source to destination
15. Dest-bytes	C	Number of data bytes transfered from destination to source
16. Land	D	1 if connection is from/to the same host/port; 0 otherwise
Content-based features		
17. Conn-status	C	Status of the connection (e.g., '1' for complete, '0' for reset)
Time-based features		
18. count-dest	C	Number of flows to unique destination IPs in the last T seconds from the same source
19. count-src	C	Number of flows from unique source IPs in the last T seconds to the same destination
20. count-serv-src	C	Number of flows from the source to the same destination port in the last T seconds
21. count-serv-dest	C	Number of flows from the destination to the same source port in the last T seconds
Connection-based features		
22. count-dest-conn	C	Number of flows to unique destination IPs in the last N flows from the same source
23. count-src-conn	C	Number of flows from unique source IPs in the last N flows to the same destination
24. count-serv-srcconn	C	Number of flows from the source IP to the same destination port in the last N flows
25. count-serv-destconn	C	Number of flows to the destination IP to the same source port in the last N flows
C-Continuous, D-Discrete		

- Several datasets lack traffic features, although it is important to extract traffic features with their relevancy for a particular attack.

Despite the effort needed to create unbiased datasets, there will always be deficiencies in any one particular dataset. Therefore, it is very important to generate dynamic datasets which not only reflect the traffic compositions and intrusions types of the time, but are also modifiable, extensible, and reproducible. Therefore, new datasets must be generated from time to time for the purpose of analysis, testing and evaluation of network intrusion detection methods and systems from multiple perspectives.

In this paper, we provide a systematic approach to generate real-life network intrusion datasets using both packet and flow level traffic information. Three different types of datasets has been generated using the TUIDS testbed. They are (i) the TUIDS intrusion dataset, (ii) the TUIDS coordinated scan dataset, and (iii) the TUIDS DDoS dataset. We incorporate the maximum number of possible attacks and scenarios when generating the datasets on our testbed network.

Acknowledgments

This work is partially supported by Department of Information Technology (DIT) and Council of Scientific & Industrial Research (CSIR), Government of India. The authors are thankful to the funding agencies and also gratefully acknowledge the anonymous reviewers for their valuable comments.

References

[1] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "RODD: An effective reference-based outlier detection technique for large datasets," in *Proceedings of First International Conference on Computer Science and Information Technology*, pp. 76–84, Bangalore, India, 2011.

[2] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 1, pp. 303–336, 2014.

[3] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Towards an unsupervised method for network anomaly detection in large datasets," *Computing and Informatics*, vol. 33, no. 1, pp. 1–34, 2014.

[4] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "NADO: Network anomaly detection using outlier approach," in *Proceedings of ACM International Conference on Communication, Computing & Security*, pp. 531–536, New York, USA, 2011.

[5] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "AOCD: An adaptive outlier based coordinated scan detection approach," *International Journal of Network Security*, vol. 14, no. 6, pp. 339–351, 2012.

[6] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "An effective unsupervised network anomaly detection method," in *Proceedings of ACM International Conference on Advances in Computing, Communications and Informatics*, pp. 533–539, New York, USA, 2012.

[7] CACE Technologies, *WinPcap*, June 2015. (http://www.winpcap.org)

[8] CAIDA, *The Cooperative Analysis for Internet Data Analysis*, 2011. (http://www.caida.org)

Table 13: List of features in the KDDcup99 intrusion dataset

Label/feature name	Type	Description
Basic features		
1. Duration	C	Length (number of seconds) of the connection
2. Protocol-type	D	Type of protocol, e.g., tcp, udp, etc.
3. Service	D	Network service at the destination, e.g., http, telnet, etc.
4. Flag	D	Normal or error status of the connection
5. Src-bytes	C	Number of data bytes from source to destination
6. Dst-bytes	C	Number of data bytes from destination to source
7. Land	D	1 if connection is from/to the same host/port; 0 otherwise
8. Wrong-fragment	C	Number of "wrong" fragments
9. Urgen	C	Number of urgent packets
Content-based features		
10. Hot	C	Number of "hot" indicators (hot: number of directory accesses, create and execute program)
11. Num-failed-logins	C	Number of failed login attempts
12. Logged-in	D	1 if successfully logged-in; 0 otherwise
13. Num-compromised	C	Number of "compromised" conditions (compromised condition: number of file/path not found errors and jumping commands)
14. Root-shell	D	1 if root-shell is obtained; 0 otherwise
15. Su-attempted	D	1 if "su root" command attempted; 0 otherwise
16. Num-root	C	Number of "root" accesses
17. Num-file-creations	C	Number of file creation operations
18. Num-shells	C	Number of shell prompts
19. Num-access-files	C	Number of operations on access control files
20. Num-outbound-cmds	C	Number of outbound commands in an ftp session
21. Is-host-login	D	1 if login belongs to the "hot" list; 0 otherwise
22. Is-guest-login	D	1 if the login is a "guest" login; 0 otherwise
Time-based features		
23. Count	C	Number of connections to the same host as the current connection in the past 2 seconds
24. Srv-count	C	Number of connections to the same service as the current connection in the past 2 seconds (same-host connections)
25. Serror-rate	C	% of connections that have "SYN" errors (same-host connections)
26. Srv-serror-rate	C	% of connections that have "SYN" errors (same-service connections)
27. Rerror-rate	C	% of connections that have "REJ" errors (same-host connections)
28. Srv-rerror-rate	C	% of connections that have "REJ" errors (same-service connections)
29. Same-srv-rate	C	% of connections to the same service (same-host connections)
30. Diff-srv-rate	C	% of connections to different services (same-host connections)
31. Srv-diff-host-rate	C	% of connections to different hosts (same-service connections)
Connection-based features		
32. Dst-host-count	C	Count of destination hosts
33. Dst-host-srv-count	C	Srv_count for destination host
34. Dst-host-same-srv-rate	C	Same_srv_rate for destination host
35. Dst-host-diff-srv-rate	C	Diff_srv_rate for destination host
36. Dst-host-same-src-port-rate	C	Same_src_port_rate for destination host
37. Dst-host-srv-diff-host-rate	C	Diff_host_rate for destination host
38. Dst-host-serror-rate	C	Serror_rate for destination host
39. Dst-host-srv-serror-rate	C	Srv_serror_rate for destination host
40. Dst-host-rerror-rate	C	Rerror_rate for destination host
41. Dst-host-srv-rerror-rate	C	Srv_rerror_rate for destination host
C-Continuous, D-Discrete		

Table 14: TUIDS dataset traffic composition

Protocol	*Size (MB)*	*(%)*
(a) Total traffic composition		
IP	66784.29	99.99
ARP	3.96	0.005
IPv6	0.00	0.00
IPX	0.00	0.00
STP	0.00	0.00
Other	0.00	0.00
(b) TCP/UDP/ICMP traffic composition		
TCP	49049.29	73.44%
UDP	14940.53	22.37%
ICMP	2798.43	4.19%
ICMPv6	0.00	0.00
Other	0.00	0.00

[9] A. Cemerlic, L. Yang, and J.M. Kizza, "Network intrusion detection based on bayesian networks," in *Proceedings of the 20th International Conference on Software Engineering and Knowledge Engineering*, pp. 791–794, San Francisco, USA, 2008.

[10] A. Dainotti and A. Pescape, "PLAB: A packet capture and analysis architecture," 2004. (http://traffic.comics.unina.it/software/ITG/D-ITGpublications/TR-DIS-122004.pdf)

[11] DEFCON, *The SHMOO Group*, 2011. (http://cctf.shmoo.com/)

[12] L. Delooze, *Applying Soft-Computing Techniques to Intrusion Detection*, Ph.D. Thesis, Computer Science Department, University of Colorado, Colorado Springs, 2005.

[13] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on Software Engineering*, vol. 13, pp. 222–232, Feb. 1987.

[14] A. A. Ghorbani, W. Lu, and M. Tavallaee, "Network attacks," in *Network Intrusion Detection and Prevention*, pp. 1–25, Springer-verlag, 2010.

[15] P. Gogoi, D. K. Bhattacharyya, B. Bora, and J. K. Kalita, "MLH-IDS: A multi-level hybrid intrusion detection method," *The Computer Journal*, vol. 57, pp. 602–623, May 2014.

[16] P. Gogoi, M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Packet and flow-based network intrusion dataset," in *Proceedings of the 5th International Conference on Contemporary Computing*, LNCS-CCIS 306, pp. 322–334, Springer, 2012.

Table 15: Distribution of normal and attack connection instances in real time packet and flow level TUIDS datasets

Connection type	Dataset type			
	Training dataset		Testing dataset	
(a) TUIDS intrusion dataset				
Packet level				
Normal	71785	58.87%	47895	55.52%
DoS	42592	34.93%	30613	35.49%
Probe	7550	6.19%	7757	8.99%
Total	121927	-	86265	-
Flow level				
Normal	23120	43.75%	16770	41.17%
DoS	21441	40.57%	14475	35.54%
Probe	8282	15.67%	9480	23.28%
Total	52843	-	40725	-
(b) TUIDS coordinated scan dataset				
Packet level				
Normal	65285	90.14%	41095	84.95%
Probe	7140	9.86%	7283	15.05%
Total	72425	-	48378	-
Flow level				
Normal	20180	73.44%	15853	65.52%
Probe	7297	26.56%	8357	34.52%
Total	27477	-	24210	-
(c) TUIDS DDoS dataset				
Packet level				
Normal	46513	68.62%	44328	60.50%
Flooding attacks	21273	31.38%	28936	39.49%
Total	67786	-	73264	-
Flow level				
Normal	27411	57.67%	28841	61.38%
Flooding attacks	20117	42.33%	18150	38.62%
Total	47528	-	46991	-

Table 16: Comparison of existing datasets and their characteristics

Dataset	*u*	*v*	*w*	*No. of instances*	*No. of attributes*	*x*	*y*	*z*	*Some references*
Synthetic	No	No	Yes	user dependent	user dependent	Not known	any	user dependent	[4, 1]
KDDcup99	Yes	No	Yes	805050	41	BCTW	P	C_1	[48, 33, 47, 31]
NSL-KDD	Yes	No	Yes	148517	41	BCTW	P	C_1	[43]
DARPA 2000	Yes	No	No	Huge	Not known	Raw	Raw	C_2	[37]
DEFCON	No	No	No	Huge	Not known	Raw	P	C_2	[37]
CAIDA	Yes	Yes	No	Huge	Not known	Raw	P	C_1	[37]
LBNL	Yes	Yes	No	Huge	Not known	Raw	P	C_2	[46]
Endpoint	Yes	Yes	No	Huge	Not known	Raw	P	C_2, C_3	[46]
UNIBS	Yes	Yes	No	Huge	Not known	Raw	P	C_2	[46]
ISCX-UNB	Yes	Yes	Yes	Huge	Not known	Raw	P	A	[37]
KU	Yes	Yes	No	Huge	24	BTW	P	C_1	[39]
TUIDS	Yes	Yes	Yes	Huge	50,24	BCTW	P,F	C_1	[4, 1]

u-realistic network configuration
v-indicates realistic traffic
w-describes the label information
x-types of features extracted as basic features (B), content based features (C), time based features(T) and window based features(W)
y-explains the types of data as packet based (P) or flow based (F) or hybrid (H) or others (O)
z-represents the attack category as C_1-all attacks, C_2-denial of service, C_3-probe, C_4-user to root, C_5-remote to local, and A-application layer attacks

[17] D. Hoplaros, Z Tari, and I. Khalil, "Data summarization for network traffic monitoring," *Journal of Network and Computer Applications*, vol. 37, pp. 194–205, 2014.

[18] Information Systems Technology Group MIT Lincoln Lab, *DARPA Intrusion Detection Data Sets*, Mar. 2000. (http://www.ll.mit.edu/mission/ communications/ist/corpora/ideval/data/2000data.html)

[19] V. Jacobson, C. Leres, and S. McCanne, "The tcpdump manual page," Lawrence Berkeley Laboratory, Berkeley, CA, 1989.

[20] V. Jacobson, C. Leres, and S. McCanne, "Libpcap," Lawrence Berkeley Laboratory, Berkeley, CA, Initial public release, June 1994.

[21] KDDcup99, "Knowledge discovery in databases DARPA archive," 1999. (https://archive.ics.uci.edu/ml/databases/kddcup99/)

[22] K. Kendall, *A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems*, Master's Thesis, MIT, 1999.

[23] Lawrence Berkeley National Laboratory (LBNL), *ICSI, LBNL/ICSI Enterprise Tracing Project*, 2005. (http://www.icir.org/enterprise-tracing/)

[24] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava, "A comparative study of anomaly detection schemes in network intrusion detection," in *Proceedings of the 3rd SIAM International Conference on Data Mining*, pp. 25–36, 2003.

[25] B. Li, J. Springer, G. Bebis, and M. H. Gunes, "A survey of network flow applications," *Journal of Network*

and Computer Applications, vol. 36, no. 2, pp. 567–581, 2013.

[26] R. P. Lippmann, D. J. Fried, I. Graf, et al., "Evaluating intrusion detection systems: The 1998 DARPA offline intrusion detection evaluation," in *Proceedings of the DARPA Information Survivability Conference and Exposition*, pp. 12–26, 2000.

[27] M. V. Mahoney and P. K. Chan, "An analysis of the 1999 DARPA/Lincoln laboratory evaluation data for network anomaly detection," in *Proceedings of the 6th International Symposium on Recent Advances in Intrusion Detection*, pp. 220–237, 2003.

[28] S. McCanne and V. Jacobson, "The BSD packet filter: A new architecture for user level packet capture," in *Proceedings of the Winter 1993 USENIX Conference*, pp. 259–269, 1993.

[29] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by lincoln laboratory," *ACM Transactions on Information and System Security*, vol. 3, pp. 262–294, Nov. 2000.

[30] P. Mell, V. Hu, R. Lippmann, J. Haines, and M. Zissman, *An Overview of Issues in Testing Intrusion Detection Systems*, 2003. (http://citeseer.ist.psu.edu/621355.html)

[31] Z. Muda, W. Yassin, M. N. Sulaiman, and N. I. Udzir, "A K-means and naive bayes learning approach for better intrusion detection," *Information Technology Journal*, vol. 10, no. 3, pp. 648–655, 2011.

[32] NSL-KDD, *NSL-KDD Data Set for Network-based Intrusion Detection Systems*, Mar. 2009. (http://iscx.cs.unb.ca/NSL-KDD/)

[33] M. E. Otey, A. Ghoting, and S. Parthasarathy, "Fast distributed outlier detection in mixed-attribute data sets," *Data Mining and Knowledge Discovery*, vol. 12, no. 2-3, pp. 203–228, 2006.

[34] R. Pang, M. Allman, M. Bennett, J. Lee, V. Paxson, and B. Tierney, "A first look at modern enterprise traffic," in *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*, pp. 2, Berkeley, USA, 2005.

[35] R. Pang, M. Allman, V. Paxson, and J. Lee, "The devil and packet trace anonymization," *SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 29–38, 2006.

[36] L. Portnoy, E. Eskin, and S. Stolfo, "Intrusion detection with unlabeled data using clustering," in *Proceedings of ACM CSS Workshop on Data Mining Applied to Security*, pp. 5–8, 2001.

[37] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Towards developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers & Security*, vol. 31, no. 3, pp. 357–374, 2012.

[38] J. Song, H. Takakura, and Y. Okabe, "Description of kyoto university benchmark data,". pp. 1–3. 2006. (http://www.takakura.com/Kyoto_data/BenchmarkData-Description-v5.pdf)

[39] J. Song, H. Takakura, Y. Okabe, and K. Nakao, "Toward a more practical unsupervised anomaly detection system," *Information Sciences*, vol. 231, pp. 4–14, Aug. 2013.

[40] A. Sperotto, R. Sadre, F. Vliet, and A. Pras, "A labeled data set for flow-based intrusion detection," in *Proceedings of the 9th IEEE International Workshop on IP Operations and Management*, pp. 39–50, Venice, Italy, 2009.

[41] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, "Cost-based modeling for fraud and intrusion detection: Results from the JAM project," in *Proceedings of the IEEE DARPA Information Survivability Conference and Exposition*, vol. 2, pp. 130–144, USA, 2000.

[42] symantec.com, *Symantec Security Response*, June 2015. (http://securityresponse.symantec.com/avcenter)

[43] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proceedings of the 2nd IEEE International Conference on Computational Intelligence for Security and Defense Applications*, pp. 53–58, USA, 2009.

[44] C. Thomas, V. Sharma, and N. Balakrishnan, "Usefulness of DARPA dataset for intrusion detection system evaluation," in *Proceedings of the Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security*, SPIE 6973, Orlando, FL, 2008.

[45] UNIBS, *University of Brescia Dataset*, 2009. (http://www.ing.unibs.it/ntw/tools/traces/)

[46] J. Xu and C. R. Shelton, "Intrusion detection using continuous time bayesian networks," *Journal of Artificial Intelligence Research*, vol. 39, pp. 745–774, 2010.

[47] G. Zhang, S. Jiang, G. Wei, and Q. Guan, "A prediction-based detection algorithm against distributed denial-of-service attacks," in *Proceedings of the ACM International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly*, pp. 106–110, Leipzig, Germany, 2009.

[48] Y. F. Zhang, Z. Y. Xiong, and X. Q. Wang, "Distributed intrusion detection based on clustering," in *Proceeding of the International Conference on Machine Learning and Cybernetics*, vol. 4, pp. 2379–2383, Aug. 2005.

Monowar H. Bhuyan is an assistant professor in the Department of Computer Science and Engineering at Kaziranga University, Jorhat, Assam, India. He received his Ph.D. in Computer Science & Engineering from Tezpur University (a Central University) in February 2014. He is a life member of IETE, India. His research areas include data mining, cloud security, computer and network security. He has published 20 papers in international journals and referred conference proceedings. He is on the programme committee members/referees of several international conferences/journals.

Dhruba K. Bhattacharyya received his Ph.D. in Computer Science from Tezpur University in 1999. Currently, he is a Professor in the Computer Science & Engineering Department at Tezpur University. His research areas include data mining, network security and bioinformatics. Prof. Bhattacharyya has published more than 220 research papers in leading international journals and conference proceedings. Dr. Bhattacharyya also has written/edited 10 books. He is on the editorial boards of several international journals and also on the programme committees/advisory bodies of several international conferences/workshops.

Jugal K. Kalita is a professor of Computer Science at the University of Colorado at Colorado Springs. He received his Ph.D. from the University of Pennsylvania in 1990. His research interests are in natural language processing, machine learning, artificial intelligence, bioinformatics and applications of AI techniques to computer and network security. He has published more than 150 papers in international journals and referred conference proceedings and has written two technical books. Professor Kalita is a frequent visitor of Tezpur University where he collaborates on research projects with faculty and students.