KDD 1999 generation faults: a review and analysis

Amjad M. Al Tobi[a,b] and Ishbel Duncan[b]
[a]Centre of Information Systems, Sultan Qaboos University, Al-Khoud, Sultanate of Oman; [b]School of

Computer Science, University of St. Andrews, Fife, Scotland, UK

ABSTRACT

DARPA 1998 was one of the first Intrusion Detection datasets that was made publicly available. The KDD 1999 dataset was derived from DARPA 1998 to be used by researchers in developing machine learning (ML), classification and clustering algorithms with a security focus. DARPA 1998 has been criticised in literature due to raised concerns of problems in the dataset. Many research- ers have accused KDD 1999 of having similar concerns but insuffi- cient published evidence has been found. In this paper, we review the KDD 1999 generation process and present new proofs of existing inconsistencies in KDD 1999. We then present the process used to link some of the KDD 1999 (TELNET) records back to their origins in DARPA 1998 and discuss the interesting results and findings of this experiment.

KEYWORDS

DARPA 1998 dataset; KDD Cup 1999 dataset; KDD review; KDD generation process; KDD generation faults

1. Introduction

The computer security field, in general, lacks open and easy access to standardised and up-to-date datasets. This might be because of privacy and confidentiality restrictions, which complicates any attempt to create a publicly available dataset. As a result of the vast variations of services and newly developed tools that emerge every day, simulation of different network traffic is a difficult task. Therefore, DARPA 1998 and KDD 1999 datasets have been widely used in research and there have been no alternatives until recently, when new datasets such as the UNB ISCX Intrusion Detection Evaluation DataSet 2012 [1] started to emerge. The DARPA 1998 and KDD 1999 datasets are also used as benchmarks to compare newly developed systems and techniques with old analysis and results. In this paper, the DARPA term refers to the DARPA 1998 dataset, KDD refers to the KDD 1999 dataset and ML refers to machine learning.

1.1. DARPA 1998 dataset

The DARPA dataset [2] was created in a joint project between the Defence Advanced Research Project Agency (DARPA), Lincoln Laboratory at Massachusetts Institute of Technology (MIT) and the Air Force Research Laboratory (AFRL/SNHS). The project (Intrusion Detection Evaluation Project) aimed to evaluate different IDS available at that time. As a result the DARPA dataset was an attempt to be the first standard corpora of its kind in the security field.

The DARPA project produced two sets of data. The first set is used for training purposes and is composed of 35 days of simulation (7 weeks, 5 days a week). The second set was aimed at testing systems under evaluation and contains 10 days of simulation (2 weeks, 5 days a week). For every simulation day, network traffic traces (TCPDUMP) and host audit data and traces (BSM [Basic Security Module]) are provided. Every trace is accompanied by a list file containing the attack details such as time, machines, ports, attack name, etc. These attacks are basically different instances of four main classes of attacks, namely: Denial of Service (DoS), Remote to Local (R2L), User to Root (U2R) and Probing. Also there are some connections, which are labelled as Anomaly.

Details of the network setup of the simulation environment is provided on the DARPA project's website [2]. All datasets and files are available for download from the project's archive. In general, every

simulation day archive will contain the following important files, which include all the required information and traces of that day (see Table 1).

Table 1. Files provided in every simulation day in the DARPA project.

File Name	Description
tcpdump.data.gz	The raw tcpdump data from the sniffer in this simulation
tcpdump.list	The list file for the tcpdump data. This file lists connections and their status
pascal.bsm.gz	The actual raw BSM data from this simulation
bsm.list	The list file for the BSM data.
pascal.praudit.gz	ASCII version of pascal's BSM audit data obtained by passing pascal.bsm through praudit
ps_monitor.log.gz	The results of running the UNIX command 'ps -elf' every 60 s on the machine which was audited

Almost all network-based security research focussed on 'tcpdump.data.gz' and 'tcpdump.list' in the DARPA dataset, whereas host-based security research concentrated on the other files. In this paper, the reported experiment is focused on these two files from every simulation day of the training dataset only, as they are the ones used to generate the KDD dataset.

The 'tcpdump.list' files provide the label of every connection in DARPA's PCAP files ('tcpdump.data.gz') as the above example illustrates (Table 2). Every connection with Score zero or Name '-' is normal traffic. These labels were used in the generation and labelling of KDD 1999 dataset as well as used in the analysis presented in this paper.

Table 2. Example of DARPA 1998 'tcpdump.list' file content.

No.	Start Date	Start Time	Duration	Service	src_port	dst_port	src_ip	dst_ip	Attack Score	Name
1	01/23/1998	16:56:12	00:01:26	telnet	1754	23	192.168.1.30	192.168.0.20	0	-
2	01/23/1998	16:56:15	00:00:13	ftp	1755	21	192.168.1.30	192.168.0.20	0	-
3	01/23/1998	16:56:17	00:00:01	smtp	43493	25	192.168.0.40	192.168.1.30	0	-
⋮										
15	01/23/1998	16:56:45	00:00:00	http	1784	80	192.168.1.30	192.168.0.40	1	phf
16	01/23/1998	16:56:49	00:00:14	ftp	43504	21	192.168.0.40	192.168.1.30	0	-
⋮										
89	01/23/1998	16:58:44	00:00:02	http	1866	80	192.168.1.30	192.168.0.40	0	-
90	01/23/1998	16:58:45	00:00:22	telnet	1867	23	192.168.1.30	192.168.0.20	1	guess
91	01/23/1998	16:58:47	00:00:02	http	1868	80	192.168.1.30	192.168.0.40	0	-
⋮										
124	01/23/1998	17:00:01	00:00:00	ftp-data	20	43548	192.168.1.30	192.168.0.40	0	-
125	01/23/1998	17:00:02	00:00:02	rsh	1023	514	192.168.1.30	192.168.0.20	1	rcp
126	01/23/1998	17:00:03	00:00:22	telnet	1906	23	192.168.1.30	192.168.0.20	1	guess
127	01/23/1998	17:00:04	00:00:01	ftp-data	20	43550	192.168.1.30	192.168.0.40	0	-
128	01/23/1998	17:00:05	00:00:14	rlogin	1022	513	192.168.1.30	192.168.0.20	1	rlogin
129	01/23/1998	17:00:07	00:00:00	ftp-data	20	43552	192.168.1.30	192.168.0.40	0	-
⋮										
135	01/23/1998	17:00:20	00:00:00	ftp-data	20	43564	192.168.1.30	192.168.0.40	0	-
136	01/23/1998	17:00:21	00:00:02	rsh	1022	514	192.168.1.30	192.168.0.20	1	rsh
137	01/23/1998	17:00:22	00:00:01	rsh	1022	1021	192.168.0.20	192.168.1.30	1	rsh
138	01/23/1998	17:00:25	00:00:02	http	1908	80	192.168.1.30	192.168.0.40	0	-
⋮										

1.2. KDD 1999 dataset

KDD [3] is basically a transformation of DARPA's network traces into a collection of connections' features, 41 features in total. Every connection between two hosts will represent a single record in KDD, and every record will provide a high level view of this connection based on the feature value. The 41 features are divided into four groups as shown in Table 3.

Stolfo et al. [4] and Lee et al [5–7]. have provided some details of the KDD's generation process. They have used Bro (Network Analysis Framework) [8] to process the network traces ('tcpdump.data.gz' and 'tcpdump.list') from the two DARPA datasets to generate the equivalent KDD sets (training 743MB and testing 430MB). The KDD process has stripped five important pieces of information (start time of connection, source IP, source port, destination IP and destination port) that made it nearly impossible to link any single connection record to its source in the DARPA traces. Without this information, it was

Table 3. KDD 1999 features.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41
duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_compromised	root_shell	su_attempted	num_root	num_file_creations	num_shells	num_access_files	num_outbound_cmds	is_host_login	is_guest_login	count	srv_count	serror_rate	srv_serror_rate	rerror_rate	srv_rerror_rate	same_srv_rate	diff_srv_rate	srv_diff_host_rate	dst_host_count	dst_host_srv_count	dst_host_same_srv_rate	dst_host_diff_srv_rate	dst_host_same_src_port_rate	dst_host_srv_diff_host_rate	dst_host_serror_rate	dst_host_srv_serror_rate	dst_host_rerror_rate	dst_host_srv_rerror_rate
Basic Features (Total: 9)									**Content Features (Total: 13)**													**Time-based Features (Total: 9)**								**Connection-based Features (Total: 10)**									

impossible, until now, for researchers to validate the KDD creation process. Also, all scripts used and details of tools used along with their versions and parameter settings were not publicly available, which has made any attempt to validate KDD a difficult quest. Given these concerns it is surprising that KDD was accepted as a defacto standard for researchers to test their algorithms and detection systems. Since then no published paper has ever analysed or validated this creation process, although, Perona et al. [9], have replicated the generation process of the KDD1999. However, their replication was not intended to address the limitations of the KDD1999, instead it aimed to extend the connections profiling by including their contents (payloads). Their study showed that analysis of the connections' content can be used to detect attacks in network traffic. However, this finding will be restricted when such traffic is encrypted, as is the case with the modern traffic. This study has resulted in the generation of a new dataset called gureKddcup database [9] that is similar to the KDD1999, however, much cleaner with all payloads extracted.

It is worth noting our analysis of KDD has revealed that the definition of a connection for ICMP is different from UDP and TCP. We noted that every ICMP packet is treated as a single connection (state-less), whereas every connection in UDP and TCP consists of a sequence of exchanged packets between two machines (state-full). This was evident by the fact that the destination bytes field (Feature 6: dst_bytes) in all ICMP connections is always set to zero. That would mean there was no response from the other end of communicating parties. This was not the case for UDP and TCP connections.

In this paper, we present important discoveries that indicate serious flaws in KDD and we question its generation process through comparative analysis and identification of mismatches. We reveal many inconsistencies that affect many of KDD's features, which question its reliability and suitability.

2. Existing criticism

In this section, we provide the main criticisms discussed in IDS literature about both datasets; DARPA and KDD.

2.1. DARPA

Almost all criticism in the literature is directed against DARPA and mainly concern its representation of reality. All DARPA criticism is inherited by KDD as it is a transformation of the original dataset. However, there was no secondary investigation into how it was generated and the possible faults it might have, independent from DARPA concerns. Brugger [10], has called for stopping the use of KDD in research and provided some criticism against DARPA as the source of KDD, but not the KDD dataset itself. Brugger [10], has criticised the DARPA dataset for including a very limited number of attacks which can be detected with a fixed signature.

Many have investigated the credibility of the DARPA dataset and many others have questioned its validity. For example, McHugh's [11,12] criticism was focused on the generation methods of DARPA. He has mainly questioned the process of evaluating a real world system's performance by using experimental data. He has questioned the level to which the use of synthetic traffic is suitable for the

evaluation task. He also raised some concerns about the possible effects potentially caused by the architecture of the simulation environment on the overall evaluation.

McHugh [11,12] also questioned the use of ROC (Receiver Operating Characteristic) curve as a method for presenting the evaluation results. He has raised doubts about some unexpected problems that might take place by using this form of analysis in the IDS evaluation domain. The main problems that he has highlighted are, 'determining appropriate units of analysis, bias towards possibly unrealistic detection approaches, and questionable presentations of false alarm data'.

Mahoney and Chan [13,14] criticised DARPA on the data level. They have noted that most of the attacks in DARPA can be detected using the TTL (Time-to-live) field in packets headers. They have noted that the DARPA training set has eight distinct TTL values, which is an unrealistic setup. This might be caused by the scripts and software used to generate traffic in the testing environment.

2.2. KDD

Although many researchers believe that KDD has inherited DARPA's problems, no evidence is provided to backup this claim. There were a number of attempts to analyse KDD but no actual validation or firm evaluation, especially on its generation process, was ever conducted.

Tavallaee et al. [15], have attempted to analyse KDD and they have argued that the large number of redundant records in KDD is one of the main deficiencies of this dataset. Therefore, they suggested removing duplicate records and as a result they produced a shorter version called NSL-KDD. They assumed that records with exact feature values were duplicates. However, removing these records from KDD – based on that assumption – was not reasonable, as it is not logical to identify redundant records without having access to their timestamp, source and destination addresses and ports. That is because DARPA was composed of synthetic traffic, which was mostly produced using scripts or software by varying parameters. This could have generated identical traffic but between different hosts at different times, which, when this informa- tion is stripped, will produce the illusion of having identical traffic. Also, as noted in the Introduction, KDD treated every ICMP packet as a single connection. These cases produced traffic that looked identical in many cases, resulting in identical records in the KDD dataset. Based on the discussion in the next section (3. DARPA vs. KDD), this claim of redundancy was confirmed by comparing DARPA and KDD, but identifying the duplicate records is not easy task. Identifying these duplications the way Tavallaee et al. [15], have done has similarly raised concerns.

The production of NSL-KDD has introduced some confusion between researchers, as many believe KDD and NSL-KDD are the same. For example, Wutyi and Thwin [16], used KDD in their experiment, according to their references, but in their paper they referred to the dataset as NSL-KDD. Strangely enough, some researchers get confused between DARPA and KDD. For example, Creech and Hu [17], listed criticism of KDD and its weak performance under classification and ML algorithms, but careful examination of their references demonstrates that they were referring to the DARPA dataset.

In another attempt to analyse KDD, Portnoy et al. [18], have partitioned KDD's training data into 10 subsets, where each subset is about 10% of the original dataset. They have observed that the attacks in the KDD dataset were not evenly distributed, which has made their attempt to cross-validate very difficult. It seems that they have based their work on an assumption which neglects a crucial fact about KDD. The dataset aggregates 35 days of traffic into a single collection where the record order and distribution were not taken into consideration. KDD's documentation provides no hint about the order of these records. Therefore, it is possible to find a connection from the 35th simulation day at the beginning of the KDD dataset, as our experiment has revealed. In addition, a quick analysis on DARPA revealed that traffic and attack distributions between simulation days are not equally distributed; there are days that have far higher attack traffic than others. This was done deliberately to reflect real network traffic as much as possible. Therefore, splitting KDD into ten distinct subsets was neither justified nor reasonable.

Leung and Leckie [19] pointed out that smurf and neptune attacks affect evaluations as they form over 71% of the KDD dataset. Therefore, they have suggested the use of other means to detect these types of attacks and avoid the use of anomaly detection systems. This claim neglects the fact that the KDD dataset

is meant to provide a high level view of every connection in the network from which ML algorithms or tools were to test their capabilities in distinguishing anomalous traffic from benign ones. The high number of counts of these two attacks could be a result of the way KDD is treating ICMP traffic -as explained in the Introduction- or due to the duplication presented in the next section (3. DARPA vs. KDD).

Bouzida et al. [20,21] ran statistical analysis on KDD and concluded that KDD's feature set has serious limitations, rendering ML algorithms inappropriate for the task. They have reasoned the failure to detect the Remote to Local (R2L) attacks are due to problems in the transformation of DARPA into KDD. They also considered the training and test datasets of KDD to be incoherent.

Sabhnani et al. [22,23] have suggested that training and testing datasets of KDD are addressing non similar hypothesis for Remote to Local (R2L) and Local to Remote (L2R) attacks, which has resulted in a low detection rate of these types of attacks. They have also pointed out that the KDD dataset fails to provide an acceptable framework for training as the training dataset contains diminished and distorted attack signatures.

Engen et al. [24] have addressed the problem of misclassification of R2L intrusions, where they have concluded that there are lots of normal connections identical to R2L attacks. This was evident by Bouzida's [20,21] findings, for example they have shown that there are 8054 normal connections identical to snmpgetattack intrusions. As a result, Engen et al. [24] have listed four possible causes to this; 'flaws in the data collection for the DARPA evaluation, limitations in Tcpdump, mislabelling or limitations in the transformation of DARPA Tcpdump to the KDD Cup '99 data'. The latter is confirmed by the work presented in this paper.

All presented criticism have analysed one dataset independently from the other and no solid evidence for their conclusions and claims have ever been produced. This work provides solid evidence to the claims raised by Mahoney and Chan [13,14], Tavallaee et al. [15], Engen et al. [24] and Brugger's [10]. It is unique in terms of linking connections between DARPA and KDD and identifying many serious flaws in the generation process itself. Table 4 provides a map of the analysis focus of every publication listed above.

Table 4. Publication's analysis focus.

DARPA	KDD
• Brugger [10], • McHugh [11,12], • Mahoney and Chan [13,14]	• Tavallaee et al. [15], • Portnoy et al. [18], • Leung and Leckie [19], • Bouzida et al. [20], • Bouzida [21] • Sabhnani et al. [22,23]

Our focus and contributions in this paper. We:

- **Compare DARAP and KDD together**
- **Identify the mismatches between the number of attack connections between DARPA and KDD as well as the differences in the total exchanged payload (two way analysis),**
- **Determine a linking key to match records between DARPA and KDD,**
- **Link Telnet connections from KDD back to their sources in DARPA and extract their content,**
- **Analyse the content-based features' construction process, based on the matched records,**
- **Identify – with evidence – problems in feature generation and computation.**

The main contribution of this work is the analysis and the identification of the problems in the generation process of the KDD dataset. The original process was neither evaluated nor analysed in any way to test its validity in comparison to the original data. If this process was perfected and made available for the research community, we could have had various KDD-like datasets. These could have evolved through time, introducing new attacks in new ways or domains, with new feature sets or specific features for certain attack classes.

3. DARPA vs. KDD

3.1. General comparison between DARPA and KDD

In our experiment we have started our analysis by comparing the total size of exchanged packets of every protocol between DARPA and KDD. This analysis – as the following Table 5 shows – has revealed a major flaw in KDD dataset as it contains more TCP traffic than what actually existed in DARPA. It also migrated less of ICMP and UDP traffic with no standard proportion sizes. This table was constructed by summing the src_bytes and dst_bytes values in KDD for every protocol (ICMP, TCP and UDP). In DARPA, every PCAP file has been processed to compute the payload size of every packet and sum their total by the traffic type – ICMP, TCP and UDP.

Table 5. DARPA and KDD total exchanged bytes of protocols' payloads.

Protocol	DARPA (Bytes)	KDD (Bytes)	Percentage (KDD÷ DARPA)
ICMP	7,892,338,710	2,629,222,857	33.31%
TCP	6,978,605,268	11,678,578,061	**167.35%**
UDP	70,507,806	36,000,213	51.06%

Table 6. DARPA and KDD attacks mapping.

	DARPA		KDD		
Attack Class	Counts	Attack	Counts	Attack	Percentage KDD/DARPA
DOS	2,281	back	2,203	back	96.58%
	35	land	21	land	60.00%
	1,526,643	neptune	1,072,017	neptune	70.22%
	10,498	pod	264	pod	2.51%
	250,133	smurf	2,807,886	smurf	**1,122.56%**
	2,173	teardrop	979	teardrop	45.05%
	4	syslog			0.00%
R2L	881	dict	53	guess_passwd	5.69%
	1	dict_simple			
	50	guest			
	8	ftp-write	8	ftp_write	100.00%
	8	imap	12	imap	**150.00%**
	9	multihop	7	multihop	77.78%
	5	phf	4	phf	80.00%
	2	spy	2	spy	100.00%
	1,766	warezclient	1,020	warezclient	57.72%
	1	warzclient			
	1	warez	20	warezmaster	100.00%
	19	warezmaster			
U2R	11	eject	30	buffer_overflow	96.77%
	1	eject-fail			
	10	ffb			
	1	ffb_clear			
	6	format			
	1	format_clear			
	1	format-fail			
	1	load_clear	9	loadmodule	75.00%
	11	loadmodule			
	1	perl_clear	3	perl	60.00%
	4	perlmagic			
	254	rootkit	10	rootkit	3.94%
PROBING	16,336	ipsweep	12,481	ipsweep	76.40%
	2,357	nmap	2,316	nmap	98.26%
	10,617	portsweep	10,413	portsweep	98.08%
	32,632	satan	15,892	satan	48.70%
ANOMALY	9	anomaly			0.00%
TOTAL	**1,856,771**		**3,925,650**		**211.42%**

Our initial attempt to understand the logic used to compute the Content Features in KDD has led to the analysis discussed in this paper. We linked some KDD connections to DARPA in order to extract the

content of these connections and analysed them to reverse engineer the rules used to compute these features. This analysis has led to a deeper exploration to determine the solid linking mechanisms between the parent (DARPA) and the child (KDD) datasets.

In our first experiment to link the two datasets together, we mapped the attack types between DARPA and KDD. We counted the number of every attack type in DARPA as provided in the 'tcpdump.list' files of all the 35 simulation days of the training set and matched them with their equivalents in the KDD dataset. These 'tcpdump.list' files provide a connection view of the DARPA dataset – as illustrated in Table 5 – and they form the original ground truth used to construct the KDD.

The following table (Table 6) shows the mapping of ALL attacks in KDD to their sources in DARPA along with the percentage of those attack connections in KDD with respect to DARPA. This table has revealed an important discovery. It clearly shows that KDD has more smurf and imap attacks than occur in DARPA. This would mean either normal connections were incorrectly classified as attacks in KDD, as pointed out by Engen et al. [24], or duplication has taken place in the generation process, as argued by Tavallaee et al. [15]. This has resulted in KDD having more than double the total attacks in DARPA.

Moreover, it is also not clear if the selection of traffic was random or not, as the percentages, also shown in Table 6, demonstrate a great deal of variations between attack types. For instance, we found that only 2.51% of pod attack were added to KDD, whereas 100% of other attacks (ftp-write, spy, etc. . .) were migrated. In addition, it was not stated anywhere in the documentation of KDD how anomaly connections were treated and whether they were added to KDD as normal traffic, or ignored. An anomaly connection is one where there is deliberate uncommon behaviour by users, such as logging in at night time, to test the system.

Table 7. Content features for services.

		Content Feature Number												
Service	Count	10	11	12	13	14	15	16	17	18	19	20	21	22
auth	1814			1	1			1						
daytime	1			1										
discard	1			1										
domain	37	1		1										
echo	1			1										
finger	27	3												
ftp	4125	30	4	1					21		1			1
ftp_data	30,464	7		1	9	1		9	2	1	1			
gopher	6	1		1										
http	567,498	21		1	21	1		1		1	1			
imap4	7	2		1	16			16	1					
IRC	363	6		1										
login	3			1	1			1			1			
netbios_ssn	1			1										
nntp	5			1										
pop_2	1			1										
pop_3	923	1		1	1	1		3						
printer	1			1										
private	8			1										
shell	2			1				4						
smtp	95,157	3		1	1	1	1	4	2		2			
ssh	21	1		1										
sunrpc	2	1												
telnet	2218	77	5	1	7479	1	2	7468	43	2	9		1	1
time	13	1		1										
uucp	3			1										
X11	84	1		1		1								
other	434	5		1				9	2	1	1			

As our attempt was to reverse engineer the Content Features construction process, we analysed KDD to discover what type of traffic affects these features. Therefore, we extracted all connection records that have at least one of their Content Features (10–22) set to a value greater than zero. After that, we analysed the services within those connections. Table 7 shows the services and their respective Content Features. Values of those features represent the maximum value found with respect to the service under consideration. For example, the maximum value of feature 16 (num_root) in the Telnet service is 7468.

The minimum values of these features is always zero. It is also clear that feature 2 (num_outbound_cmds) is always zero. This raises a serious question about its contribution to the dataset.

The Telnet service contributes to 12 out of 13 features as clearly shown in Table 7. Therefore, the experiments described in this paper were focused on Telnet connections.

Table 7 presents another important observation, which is the generalization of features to all services. This generalisation could be considered as one of the main drawbacks of KDD. The presence of these features in connections, such as ICMP connections, will affect the learning abilities of ML algorithms and tools. This could provide an explanation of the failure in detecting attacks based on these features, such as R2L and U2R attacks.

4. Matching connections between DARPA and KDD

The second presented experiment has a number of phases, where every phase was aimed at understanding one part of the KDD's construction process. Initially basic statistical analysis of the data (KDD and DARPA) was generated and compared. Then subsets of both datasets were selected for further analysis and matching up.

4.1. Phase 1 – matching telnet attacks

We conducted another general attack type mapping between DARPA and KDD, but this time focused on Telnet connections only. This was to determine the possibility and feasibility of conducting manual matching of these attack connections between the two datasets. The following table – Table 8 – shows that a manageable number of cases exists for the manual matching attempt.

Table 8. Mapped attacks of telnet connections in DARPA and KDD.

	DARPA – TELNET		KDD – TELNET		
Attack Class	Counts	Attack	Counts	Attack	Percentage KDD/DARPA
DOS	1	land	1	land	100.00%
	2,373	neptune	1,923	neptune	81.04%
	1,773	teardrop			0.00%
R2L	881	dict	53	guess_passwd	5.69%
	1	dict_simple			
	50	guest			
	4	multihop	2	multihop	50.00%
	2	spy	2	spy	100.00%
U2R	11	eject	21	buffer_overflow	100.00%
	1	eject-fail			
	5	ffb			
	1	ffb_clear			
	1	format			
	1	format_clear			
	1	format-fail			
	1	load_clear	5	loadmodule	62.50%
	7	loadmodule			
	1	perl_clear	3	perl	60.00%
	4	perlmagic			
	6	rootkit	8	rootkit	83.33%
PROBING	14	ipsweep	14	ipsweep	100.00%
	3	nmap	1	nmap	33.33%
	18	portsweep	13	portsweep	72.22%
	13	satan	7	satan	53.85%
ANOMALY	9	anomaly			0.00%
TOTAL	**5,182**		**2,050**		**39.56%**

The manual analysis of the actual content of the Telnet connections was not avoid- able. This was because of the nature of dataset and its poor documentation and lack of its original generation tools. This was a result of the fact that, these content-based features in KDD, were generated based on the actual payloads

of these connections. Hence, as KDD's generation scripts are not available, we have had to focus on a subset of the dataset to evaluate this generation process. The approach presented in this paper can be easily extended to other services as well.

In order to be able to match records from KDD to connections in DARPA, we had to identify matching keys using the existing KDD features. Therefore, manual analysis and matching has introduced the use of duration, source bytes and destination bytes (features 1, 5 and 6) as promising keys to match Telnet connections. In the coming sections, the term 'Successfully mapped' means that we mapped a KDD record to its original connection in DARPA using all three features (duration, src_bytes, dst_bytes). The term 'partially mapped' means we have used two features (src_bytes, dst_bytes) in the mapping process.

The first row of Table 9 shows that there are over 50% of Telnet records in KDD with a unique key. That means every record within that 50% has a distinct key that identifies this record alone. This gives us a good probability for one-to-one matching of Telnet connections between KDD and DARPA.

The last row of Table 9 shows that there are 45% of Telnet connections with the same key. Most of these are the neptune and portsweep attack connections, where they share the same matching key value. Appendix A presents an example of some successful mapping of the Telnet attack connections.

Table 9. Key-collision of telnet connections in KDD.

No. of Keys	Collision Degree	Coverage
2173	1	50.81%
23	2	1.08%
3	3	0.21%
3	4	0.28%
1	5	0.12%
1	7	0.16%
1	9	0.21%
1	11	0.26%
1	40	0.94%
1	1965	45.94%

We have used the term collision to determine the number of records each key will map to. For example, a key with a collision degree of 3, would mean there are three records which have the same key (duration, src_bytes, dst_bytes).

Throughout our analysis, two out of the 53 guess_passwd attacks in KDD did not map to any of DARPA's attacks. The mapping key (0,126,179) aggregated 40/53 guess_passwd KDD connections which were successfully mapped to 38 connections in DARPA with the same key. The remaining two connections did not even partially map to any connection.

Table 10 presents the number of connections for every TCP service in KDD. It also shows the percentage of the uniquely identifiable connections among all connections of a certain service. As it can be seen, only X11, IRC, smtp and telnet services have more than 50% of their connections are uniquely identifiable using the Key[dura- tion, src_bytes, dst_bytes]. Services like http_2784 and pm_dump have very small number of connections to be attractive for this investigation.

As a result, the selection of the Telnet service to be the focus of this paper was due to the following main two reasons:

1. (1) The number of connections is very large for manual analysis. Therefore, focused cases with good diversity of attack scenarios criteria has led to the selection of the Telnet service (as shown in Tables 6 and 9),
2. (2) The Telnet service contributed to 12 out of 13 content-based features. Therefore, it was a reasonable service to focus on to determine the logic of the generation process of these features (see Table 7),

Table 10. KDD TCP services and their connections counts along with their uniquely identifiable proportions.

Service	Number of Connections	Percentage of Uniquely Identifiable Connections
http_2784	1	100.00%
X11	135	74.07%
IRC	521	68.71%
pm_dump	5	60.00%
smtp	96,554	59.80%
telnet	4277	50.81%
ftp	5214	41.10%
pop_3	1981	24.03%
http	623,091	19.88%
finger	6891	8.42%
other	16,498	3.15%
auth	3382	1.98%
domain	1113	1.17%
imap4	1069	0.75%
time	1579	0.70%
gopher	1077	0.65%
ssh	1075	0.65%
shell	1051	0.57%
bgp	1047	0.38%
discard	1059	0.38%
echo	1059	0.38%
login	1045	0.38%
nntp	1059	0.38%
uucp	1041	0.38%
csnet_ns	1051	0.29%
exec	1045	0.29%
rje	1070	0.28%
uucp_path	1057	0.28%
whois	1073	0.28%
Z39_50	1078	0.19%
ctf	1068	0.19%
hostnames	1050	0.19%
iso_tsap	1052	0.19%
link	1069	0.19%
mtp	1076	0.19%
name	1067	0.19%
netbios_ssn	1055	0.19%
netstat	1056	0.19%
pop_2	1055	0.19%
printer	1045	0.19%
remote_job	1073	0.19%
sql_net	1052	0.19%
supdup	1060	0.19%
systat	1056	0.19%
vmnet	1053	0.19%
courier	1021	0.10%
efs	1042	0.10%
http_443	1044	0.10%
klogin	1050	0.10%
kshell	1040	0.10%
nnsp	1038	0.10%
sunrpc	1056	0.09%
private	1,024,316	0.05%
aol	2	0.00%
ldap	1041	0.00%

Note that the main aim of this study is not to find problems with Telnet connections, rather it investigates the process adopted in the KDD generation that was used to engineer the features set. Telnet connections were used as an example to illustrate many problems in the adopted extraction process and the engineering of many of the KDD's features. Choosing connections with their Content-based features set to values greater than zero was the main selection criteria. As these small, randomly picked connections have shown so many problems, there are no guarantees that similar ones do not exist in many other connections types as well, especially services that trigger these content-based features. The approach and method presented in this study can be extended for other connections of other services to link them to their original sources in DARPA and be investigated, evaluated and analysed.

```
Require: Files_D ← {DARPA FILES}      // List of all DARPA PCAP files

  function PARSE_PCAPS(Files_D)
  MK_D   ← {}                         // Initialize set of DARPA mapping keys
  CONN_D ← {}                         // Initialize set of DARPA connection keys
  CONT_D ← {}                         // Initialize set of connection content
    for F_L ∈ Files_D do
      for conn_i ∈ F_L do
        if conn_i == TELNET then
          Conn.Key_i ← [start_timestamp, src_ip:port, dst_ip:port, simulation_day]
          Key_i      ← [duration, src_bytes, dst_byts]

          CONN_D ← CONN_D ∪ { Conn.Key_i }
          MK_D   ← MK_D   ∪ { Key_i }
          CONT_D ← CONT_D ∪ { content of conn_i }
        end if
      end for
    end for
    return CONN_D, MK_D, CONT_D
  end function
```

Figure 1. Algorithm 1 – DARPA pre-processing.

The matching process is in two steps. The first pass will match records between DARPA and KDD using the Key[duration, src_bytes, dst_bytes]. Therefore, all records between the two datasets that have the same combination of values for these fields will be added to the matched set (Successfully mapped). The second pass will attempt to match the remaining (none-matched) connections using Key[src_bytes, dst_bytes] (partial matching). The matched records at this stage will be added to the possible matched set (partially mapped). After the completion of the matching phase, selected cases were used for manual analysis. The details of the matching process is as follows:

Step 1. Process every PCAP file in DARPA training dataset using Bro to produce a list of mapping keys (Key[duration, src bytes, dst bytes]) for every Telnet connection and extract their payloads (Algorithm 1 in Figure 1).

Step 2. Process KDD training dataset to produce a list of mapping keys (Key[duration, src_bytes, dst_bytes]) for every Telnet connection (Algorithm 2 in Figure 2).

Step 3. Compare the sets of mapping keys between the two datasets as detailed in Algorithm 3 (Figure 3), by performing the following two phases:

- **Phase 1** Compare all mapping keys between the two lists (MK_D and MK_K), if any two keys match each other, then they will be added to the match list and be removed from their respective lists

- **Phase 2** Perform a partial comparison for the remaining mapping keys after the removal of the matched ones. This comparison will compare the mapping keys partially (Key[src_bytes, dst_bytes]). If any two keys match each other, then they will be added to the partially matched list and be removed from their respective lists.

Step 4. Perform manual analysis of the matched connections by comparing the payload of multiple connections – from DARPA PCAPs – and identifying possible patterns that contributed to the setting of the features – as pro- vided by KDD. This was done by selecting a number of connections that have their content-based feature (F_i in $\{F_{10}...F_{22}\}$) is set to a value greater than 0. Then, based on the definition of that feature, we performed a manual and rigorous analysis to identify all possible patterns from those connections that have contributed to this feature. Based on this analysis we were able to identify many irregularities in feature values between similar – in terms of payload – connections.

```
Require: File_kdd ← {KDD FILES}

  function PARSE_KDD(File_kdd)
  MK_K   ← {}                          // Initialize set of KDD mapping keys
  CONN_K ← {}                          // Initialize set of KDD connection keys
    for conn_i ∈ File_kdd do
      if conn_i == TELNET then
        Conn.Key_i ← [record_number of conn_i]
        Key_i      ← [duration, src_bytes, dst_byts]

        CONN_K ← CONN_K ∪ { Conn.Key_i }
        MK_K   ← MK_K   ∪ { Key_i }
      end if
    end for
    return CONN_K, MK_K
  end function
```

Figure 2. Algorithm 2 – KDD pre-processing.

4.2. Phase 2 – analysing connection contents

Applying this matching technique, we extracted the actual content (command and response sequences) of these connections and analysed them manually. We attempted to extract the relevant patterns depending on the available documentation that, we believed, will have an effect on the content features (10–22). Based on this analysis, as will be shown in the coming sections, we were able to detect a great deal of inconsistencies in the feature generation process in the original KDD.

5. Feature analysis

In this section we present all the inconsistencies that have been detected in every feature examined in these two experiments. Also we provide, where appropriate, a better explanation of those features that were poorly described in the KDD documentation based on our observations of the linked connections with DARPA.

5.1. Basic features

5.1.1. duration (feature 1)

We have identified many irregularities affecting this feature in KDD. It represents the duration of a connection, in seconds, from start to end. In a TCP connection, this represents the period between the first and last packet within a single connection. But in ICMP connections, all durations are zero, because KDD has treated every ICMP packet as a single connection.

Studying this feature within the Telnet connections, we were able to detect that some Telnet connections in KDD have a shorter duration than their equivalent connection in DARPA. After investigating this issue, we have found that if a TCP connection is attempted, where multiple SYN packets were sent at the handshake stage, only the last SYN packet was used to mark the start time of this connection. This makes the total duration shorter by the difference between the first and last SYN packets in this phase.

Another observation was that some Telnet sessions, especially the long ones, were assigned a duration of zero in KDD. There was no explanation in the KDD documentation of these cases. The following table (Table 11) shows some examples, out of over 100 mapped cases between KDD and DARPA, with a difference in durations. The features values of src_bytes and dst_bytes in both datasets were the same for these cases, and allowed us to match the records partially. This will affect the learning capability of ML algorithms.

```
Require:
  MK_K   ← {}              // Set of KDD's TELNET mapping keys
  CONN_K ← {}              // Set of KDD's TELNET connection keys
  MK_D   ← {}              // Set of DARPA's TELNET mapping keys
  CONN_D ← {}              // Set of DARPA's TELNET connection keys

  function COMPARE(MK_K, CONN_K, MK_D, CONN_D)
  MATCH ← {}               // Initialize set of matched keys between DARPA and KDD
    for Key_K^i ∈ MK_K do
      for Key_D^j ∈ MK_D do
        if Key_K^i == Key_D^j then
          MATCH ← MATCH ∪ {[CONN_K^i, CONN_D^j]}

          MK_K   ← MK_K   - {Key_K^i}        // Remove matched keys and connections
          CONN_K ← CONN_K - {CONN_K^i}
          MK_D   ← MK_D   - {Key_D^j}
          CONN_D ← CONN_D - {CONN_D^j}

        end if
      end for
    end for

  PART.MATCH ← {}        // Initialize set of partially matched keys
                         // between DARPA and KDD
    for Key_K^i ∈ MK_K do
      for Key_D^j ∈ MK_D do
        Key_K^i = Key_K^i[src_bytes, dst_bytes]
        Key_D^j = Key_D^j[src_bytes, dst_bytes]
        if Key_K^i == Key_D^j then
          PART.MATCH ← PART.MATCH ∪ {[CONN_K^i, CONN_D^j]}

          MK_K   ← MK_K   - {Key_K^i}        // Remove matched keys and connections
          CONN_K ← CONN_K - {CONN_K^i}
          MK_D   ← MK_D   - {Key_D^j}
          CONN_D ← CONN_D - {CONN_D^j}

        end if
      end for
    end for

   return MATCH, PART.MATCH
  end function
```

Figure 3. Algorithm 3 – Match records using mapping keys between KDD and DARPA.

5.1.2. wrong_fragments (feature 8)

Bro does not detect all fragmented packets as it does packet reassembling before passing them onto it's event handler. However, Bro documentation [25], does note what it terms 'weird' connections whenever a fragment with certain criteria is detected, such as an excessively small fragment or fragment overlap. There are eight cases that Bro regards as fragment problems and raises an event (conn_weird) to report them [Bro 25]. It is not clear how KDD has dealt with this problem and whether the developers have used Bro or some other tool to report wrong fragments.

5.2. Content features

5.2.1. hot (feature 10)

According to the KDD documentation [3] and Stolfo et al. [4], this feature represents the number of hot indicators. Lee et al. [5–7] has explained this as an access to a system directory or a creation or execution of a program. It is not clear what folders are considered as system directories. In our manual checking for what patterns could increment this feature, we found many inconsistencies. For instance, one of these irregularities were caused by – but not limited to- Program compilation with the GCC command, which sometimes increments this feature count and sometimes does not. Therefore, we could not accurately determine the logic for this feature's computation.

Table 11. Mapped telnet connections with wrong duration values.

	KDD				DARPA					
	Connection No.	Duration (sec)	Src. Bytes	Dest. Bytes	Duration (sec)	Src. Port	Dest. Port	Src. IP	Dest. IP	Simulation Day
Duration Difference	314,400	10	269	264,946	13	2620	23	194.7.248.153	172.16.112.194	week2_wednesday
	805,041	11,042	75	1,423	11,063	30,169	23	135.13.216.191	172.16.112.50	week7_tuesday
	891,338	11,177	417	9,716	11,198	9432	23	135.13.216.191	172.16.112.50	week7_thursday
	1,402,812	111	190	22,102	114	18,569	23	135.13.216.191	172.16.114.168	week4_wednesday
	4,828,062	11,271	2,429	20,456	11,280	11,519	23	135.13.216.191	172.16.112.50	week6_friday
Zero Duration	1,447,033	0	1,151	8,390	60,756	1024	23	205.160.208.190	172.16.112.50	week4_friday
	76,079	0	142	26,316	74,258	2514	23	135.13.216.191	172.16.112.50	week1_wednesday
	1,448,857	0	1,680	16,934	6,920	11,779	23	153.10.8.174	172.16.114.50	week4_friday
	406,573	0	302	31,786	303,780	4578	23	135.13.216.191	172.16.112.50	week3_monday
	264,455	0	303	31,907	229,525	11,340	23	135.13.216.191	172.16.112.50	week2_tuesday

For example, in the following mapped connection (1,446,934), in Table 12, there were four calls to '/bin/gcc' command, but KDD shows only three hot actions. Although connection (1,446,934) has almost the same sequence of commands and the same number of '/bin/gcc' command calls as connection (3,429,269), the former failed to detect one of the '/bin/gcc' command calls for no clear reason. (See Appendixes B and C for actual content.) Based on the fact that the DARPA dataset was generated using

Table 12. Two mapped telnet connections with similar command sequences and different counts of hot features.

KDD					DARPA					
Connection No.	Duration (sec)	Src. Bytes	Dest. Bytes	F (10)	Duration (sec)	Src. Port	Dest. Port	Src. IP	Dest. IP	Simulation Day
1,446,934	53	2628	3860	**3**	53	23,070	23	192.168.1.10	172.16.112.50	week4_friday
3,429,269	49	2402	3939	**4**	49	15,374	23	197.182.91.233	172.16.112.50	week5_wednesday

synthetic traffic and readymade scripts that include predefined attack scenarios, it will not be surprising to see lots of connection between different hosts at different times with nearly the exact same content and payload, which was evident throughout our analysis. Much of the compared traffic will have nearly same payload, but with inconsistently different feature values.

5.2.2. num_failed_logins (feature 11)

This feature counts the number of failed login attempts. In our investigation, we found that it is incremented whenever the 'incorrect login' string is present in the response message from server to the client. This means that if this message appeared in a normal response, such as an email message or SQL result, then this feature will be incremented.

Strangely enough, we identified the following case where KDD has failed to increment this feature (Table 13), even though there were two failed login attempts in this connection, as shown in Figure 4. That is because the response was 'Login incorrect' instead of 'incorrect login'.

Table 13. Mapped telnet connection with wrong num_failed_logins values.

KDD					DARPA					
Connection No.	Duration (sec)	Src. Bytes	Dest. Bytes	F (11)	Duration (sec)	Src. Port	Dest. Port	Src. IP	Dest. IP	Simulation Day
1,391,134	60	90	233	**0**	60	1026	23	207.230.54.203	172.16.114.50	week4_tuesday

```
Red Hat Linux release 4.1 (Vanderbilt)
Kernel 2.0.27 on an i486

0z
marx login: 0z

Password:

Login incorrect

v0z
marx login: v0z

Password:

Login incorrect

marx login:
```

Figure 4. Content of telnet connection (KDD No. 1,391,134) with two failed login attempts.

5.2.3. logged_in (feature 12)

This feature is used to indicate if a login to service was successful or not. This feature was evaluated by a blind search for the 'last login' pattern. Failure to detect this pattern will result in this feature not being set.

Table 14 presents some examples; connection (7449) detected a login even though the 'last login' pattern is missing, whereas in connection (3,927,225) login was not detected because the pattern was missing, even though a successful login took place (see Appendixes D and E for actual content).

Table 14. Mapped telnet connections with wrong calculation of logged_in feature.

KDD					DARPA					
Connection No.	Duration (sec)	Src. Bytes	Dest. Bytes	F (12)	Duration (sec)	Src. Port	Dest. Port	Src. IP	Dest. IP	Simulation Day
7449	184	1511	2957	**1**	184	1941	23	135.8.60.182	172.16.112.50	week1_monday
3,927,225	85	277	693	**0**	85	20,504	23	197.218.177.69	172.16.113.50	week5_friday

5.2.4. lnum_compromised (feature 13)

According to Lee et al. [5–7], this feature represents counts of error messages, such as, 'not found' or 'Jump to' instructions. Our analysis has found that some connections have 'Command not found' and 'No such file or directory' errors, but they did not contribute to this feature's calculation.

Table 15 shows that in connection (4,810,953), KDD has counted the 'no such file or directory' pattern as a compromising case. Whereas in connection (805,010) the system has failed to detect this pattern but it was incremented for some other pattern.

Table 15. Mapped telnet connections with wrong lnum_compromised values.

KDD					DARPA					
Connection No.	Duration (sec)	Src. Bytes	Dest. Bytes	F (13)	Duration (sec)	Src. Port	Dest. Port	Src. IP	Dest. IP	Simulation Day
4,810,953	61	2,336	4,194	1	61	4728	23	199.174.194.16	172.16.112.50	week6_thursday
805,010	60	2,328	4,551	1	60	23,147	23	206.47.98.151	172.16.112.50	week7_tuesday

Table 16. Mapped telnet connections with wrong lroot_shell values.

KDD					DARPA					
Connection No.	Duration (sec)	Src. Bytes	Dest. Bytes	F (14)	Duration (sec)	Src. Port	Dest. Port	Src. IP	Dest. IP	Simulation Day
406,471	150	1,587	6,707	**0**	150	25,134	23	202.247.224.89	172.16.112.50	week3_monday

5.2.5. lroot_shell (feature 14)

According the KDD documentation, this feature is set when a user achieved a root access to the shell. This is triggered when the terminal's prompt settings change. For example, a change from 'user@systemName ~ $' to 'systemName ~ #' indicates the user has escalated privilege to root level. This can be useful to detect user-to-root (U2R) attacks. In our observations we found that KDD has used a very simple check to detect this pattern. The KDD generation process checked that if a server response line starts with a '#' character then it considers that a root shell has been obtained. However, the detection of such a pattern was not always consistent as many flaws has been detected throughout our analysis.

In the following connection (406,471) in Table 16, a root shell was obtained, but the KDD generation script has failed to detect it (see Appendix F for actual content).

These inconsistences in detecting patterns might indicate that KDD has used multiple scripts with different logic in their generation process. That is because – as presented throughout this paper- a pattern will increment a feature in one connection, but it fails to do so in another one, even though same command was executed.

5.2.6. lnum_root (feature 16)

This feature counts the number of occurrences of a 'root' term within a connection's payload. Our investigation has exposed many inconsistencies with this. We have identified connections that count the number of occurrences of the word 'root', while other connections would count the number of lines that contained 'root'. In some other cases, we were not able to match the counts with any of those choices.

Treating this feature as a count is not sensible, as some connections will contain a large number of words whereas the number of occurrences of this term does not present a good indicator of anything. For example, Table 17 lists a single connection (1,401,653) which maps to a connection in DARPA with over 7468 'root' terms. However, after checking the payload, we found that all these terms appeared as a response to a directory content listing command (ls). In these responses, the 'root' term was identifying the user and group of those listed files. Using a frequency measure could have been a better indicator.

In two other cases shown in Table 17, connection (3,682,786) used the number of lines, four lines in this case, where the 'root' pattern has appeared although there were actually six patterns. Connection (841,798) used the number of occurrences of the pattern, three 'root' patterns, instead of the number of lines (two lines). We were also able to identify cases where neither pattern occurrences nor the number of lines were used to set this feature's value. Connection (1,401,653) is an example of this; it had 8332 repetitions of 'root' and 7476 lines with this pattern, but the value it had is 7468 that matches neither of these figures.

Table 17. Mapped telnet connections with wrong lnum_root values.

KDD					DARPA					
Connection No.	Duration (sec)	Src. Bytes	Dest. Bytes	F(16)	Duration (sec)	Src. Port	Dest. Port	Src. IP	Dest. IP	Simulation Day
3,682,786	290	415	70,529	**4**	290	32,612	23	206.48.44.18	172.16.112.50	week5_friday
841,798	103	302	8,876	**3**	103	9850	23	209.12.13.144	172.16.113.50	week7_wednesday
1,401,653	5328	1,604	920,608	**7468**	5328	2636	23	209.12.13.144	172.16.112.50	week4_wednesday

Table 18. Mapped telnet connection with wrong lnum_file_creations values.

KDD					DARPA					
Connection No.	Duration (sec)	Src. Bytes	Dest. Bytes	F (17)	Duration (sec)	Src. Port	Dest. Port	Src. IP	Dest. IP	Simulation Day
841,798	103	302	8,876	**4**	103	9850	23	209.12.13.144	172.16.113.50	week7_wednesday

5.2.7. lnum_file_creations (feature 17)

Lee et al. [5–7], Stolfo et al. [4], and KDD [3] have described this feature as the number of file creation attempts. Our examination has linked commands, such as, 'vi', 'cp', 'chmod', 'rm' and 'cat >' to this feature. But there is no sign of addressing any other commands, such as, 'gcc', 'mv', 'mkdir', 'touch', etc. . ..

For example, connection (841,798) in Table 18 has two 'chmod', one 'cat >', one 'cp' and one 'rm' commands, but only four of them where counted. In some other cases, some of these commands are not considered, while in others they are. This presents the vagueness of this transformation process, as it is not clear which measure was used for this metric; number of different commands or number of their calls.

5.2.8. lnum_shells (feature 18)

Our analysis has revealed that this feature is incremented by the detection of any of the shell command patterns, such as, '/bin/sh', '/bin/bash', '/bin/csh' or '/bin/ tcsh'. It is not clear if other patterns are considered or not, as further investigation is required. Even with this, connections in KDD have failed to count the occurrences of such patterns correctly. For example, connection (3,927,225) in Table 19 has two occurrences of '/bin/sh' pattern, but none of them were detected.

5.2.9. lnum_outbound_cmds (feature 20)

Although, Lee et al. [5–7], Stolfo et al. [4] and KDD [3] have described this feature as the number of outbound commands or connections within an FTP session, this feature has never been set as it is always a zero value in KDD with all services on all protocols. Therefore, we were not able to determine any link to patterns that might affect it. This also raises more concerns about the contribution of this feature to the learning capability of ML algorithms.

Table 19. Mapped telnet connection with wrong lnum_shells values.

KDD					DARPA					
Connection No.	Duration (sec)	Src. Bytes	Dest. Bytes	F (18)	Duration (sec)	Src. Port	Dest. Port	Src. IP	Dest. IP	Simulation Day
3,927,225	85	277	693	**0**	85	20,504	23	197.218.177.69	172.16.113.50	week5_friday

Table 20. Mapped telnet connection with wrong is_host_login values.

KDD					DARPA					
Connection No.	Duration (sec)	Src. Bytes	Dest. Bytes	F (21)	Duration (sec)	Src. Port	Dest. Port	Src. IP	Dest. IP	Simulation Day
1,381,226	337	237	1,540	**0**	337	13,114	23	208.254.251.132	172.16.112.50	week4_tuesday

5.2.10. is_host_login (feature 21)

This feature in Lee et al. [5–7], and KDD [3] is named as 'is_hot_login', but in the machine readable form file on KDD website it was labelled as 'is_host_login'. It is basically set to one if a 'root' username is used to login into the system. This feature is not affected, strangely enough, by switch user (su -) attempts using root accounts.

For example, the following connection (1,381,226) in Table 20 contains a 'su' call, where a root privilege is gained, but this feature has not been incremented.

5.3. Connection based attributes [features 32–41)

The calculation process of these features was not explained in a clear way, which resulted in misconceptions among researchers. Such confusion was evident by 15, when they explained the slow probing attack and how the KDD's connection features were constructed to solve this problem. 'To solve this problem, the "same host" and "same service" features are re-calculated but based on the connection window of 100 connections rather than a time window of 2 s.' (15]

Looking carefully at the explanation of Stolfo et al. [4], and Lee et al. [5–7] a separate and more confusing process will emerge. Their explanation of the construction process was as follows:

"We sorted these connection records by the destination hosts, and applied the same pattern mining and feature construction process. [Rather than|Instead of] using a time window of 2 seconds, we now used a "connection" window of 100 connections, and constructed a mirror set of "host-based traffic" features as the (time-based) "traffic" features."

The following figure (Figure 5) presents this latter description. Where two flaws can be seen very clearly in this process.

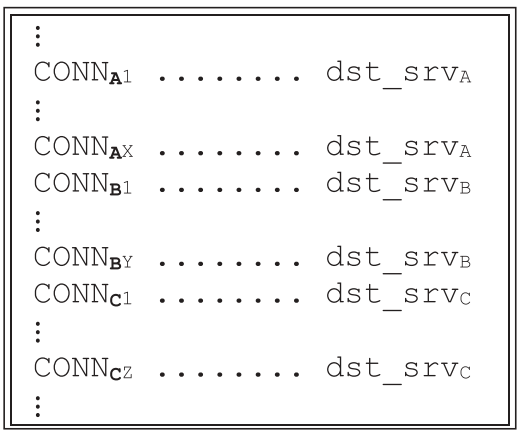

Figure 5. Representation of sorted connections by their destination hosts.

The first puzzle in this process is the purpose of sorting connections by destination and no reasonable answer can be provided. For example, what does an early connection to server C or D have to do with later connections to server A and/or B depending on where the earliest 100th connection lies? Also, how does the order of destination servers determine the relationship between connections? This is because the main purpose of constructing these features is to detect a slow probing attack, but as the explanation of this process shows, the time sequence of these connections is distorted by this sorting step.

If we assumed that the authors were looking at connections to each destination server separately, then a second puzzle will arise. For instance, if we had to analyse the last 100 connections to server B alone, then why would we compute the features that are related to the same destination host, (dst_host_count (32), dst_host_same_srv_rate (34), dst_host_- diff_srv_rate (35), dst_host_same_src_port_rate (36), dst_host_srv_diff_host_rate (37), dst_host_- serror_rate (38), dst_host_rerror_rate(40)), as the current connection? In this situation all the 100 connections are going to same server and it is not reasonable to compute the rate of connections that are going to this host.

These Connection-based attributes in KDD already present counts and rates of con- nections to different destination hosts. This means that these counts were crossing between server groups. So, organising connections into server blocks is illogical.

If these scepticisms are justified, there is still one major flaw here. Stolfo et al. [4] and Lee et al. [5–7] have claimed that the last 100 connections were examined in the construction of these features. Therefore, in the worst case scenario, all these connections will go to the same destination host and/or the same service, which will result in having counts of features 32 and 33 (dst_host_count and dst_host_srv_count) to be of value 100 at most. Our analysis has showed that, the vast majority of these counts exceeds this value, up to a recorded value of 255. Figure 6 shows the counts and percentages of such connections in KDD.

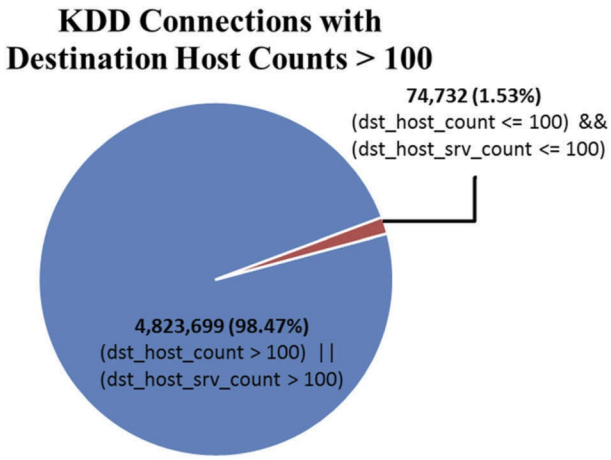

Figure 6. Connection counts based on features 32 and 33, with counts ≤ or >100.

6. Experiment design and tools

In these experiments, we have used a server with the following hardware specifications, 2U Supermicro chassis, 8× host-swap 2.5" SAS/SATA disk bays, Supermicro X8DTU-LN4F + motherboard, Dual Intel Xeon E5620 (quad core), 24GB RAM (6 × 4GB DDR3 ECC RDIMM), 4× 1TB SATA (RAID10) and 4× 1Gb Ethernet. This machine has Windows Server 2012 R2 Datacentre (64-bit) Operating System.

A virtual machine was created on Hyper-V with 4 Virtual Processors and 4 GB RAM. This VM was used to host the SecurityOnion (12.04.5.1–20,150,205) operating system, which has Bro (2.4), curl (7.22.0), wget (1.13.4) and TShark (1.6.7) installed to run these experiments.

The reason to select Bro to process the PCAP files was because it was the same tool used to process these files to generate KDD. We also used TShark to validate Bro's results of TCP connections, which has led to the configurations discussed below.

We have used bash scripting to download DARPA files (tcpdump.gz and tcpdump.list. gz) for every simulation day from the training dataset. Every tcpdump.gz file was processed by Bro to extract the basic features of every Telnet connection along with their actual content. Every extracted connection was mapped to its corresponding attack class as it appears in the tcpdump.list file. After that, all extracted connections were mapped to KDD connections using a Perl script. Duration, source bytes and destination bytes were used as matching keys.

In the development of the Bro scripts, we had to set two main parameters to be able to extract long connections as well as those connections with multiple SYN packets. For every Telnet connection a [set_inactivity_timeout(conn_id, 24.0 hr)] function is called to extend the timeout. That is because Bro by default will timeout any TCP connection after five minutes if there was no interaction between hosts. DARPA has many Telnet connections that last for long time where keep-alive packets are exchanged every 2 hours or so.

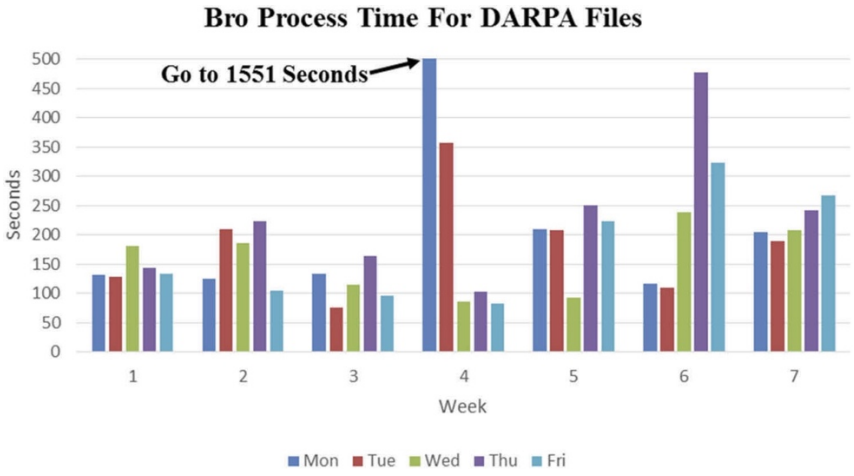

Figure 7. Run time – in seconds – of Bro scripts.

Another issue is with Telnet connections which start with multiple SYN packets. This traffic caused Bro to compute the start time of these connections in an incorrect way that caused a miscalculation of the total connection duration. Bro calculated the connection's start time for TCP connections based on the timestamp of the last SYN packet sent in the hand-shake phase before the server responded, rather than the first request. Therefore, we solved this challenge by setting this variable as this [redef tcp_attempt_delay = 1 min] because the longest delay between SYN packets was 50s.

The above figure (Figure 7) shows the run time of Bro, in seconds, to process every tcpdump file in DARPA.

7. Conclusion

In this paper, we provide evidence of existing problems in the generation process of the KDD 1999 dataset. This paper demonstrated that the KDD dataset has more attacks than what actually exists in the DARPA dataset. This suggests that a mislabelling or duplication of connections has taken place. The paper also discussed an experiment used to determine, for the first time, a linking technique between KDD and DARPA Telnet connections. This experiment has resulted in noting miscalculations on more than one quarter of the existing features, which mainly concentrated in the content-based features. This group of features are aimed at profiling connections to detect attacks related to connection payloads such as U2R and R2L. Many ML algorithms were failing in detecting those two types of attacks. Our work might shed some light on the reality of the generation process of these features, which could be one of many potential causes of the poor performance of these algorithms on these types of attacks. As a result we support

Brugger's [10],request to stop using the KDD dataset in any experiment due to these evident flaws. Researchers are encouraged to use more recent datasets for their experiments, such as the Kyoto datasets [26], Sperotto's (Twente) dataset [27] and the UNSW-NB15 dataset [28], etc.

As discussed in the literature, the main reason for researchers to use this dataset is the lack of an alternative. Therefore, a regeneration of this dataset, or any newer one, requires six main conditions to be satisfied to guarantee experimental reproducibility, which is a core requirement of science.

- Normal experimental practices:

  - First, a full documentation of the raw dataset (network traces) should be pro- vided, including environment settings, tools versions and parameters, etc.
  - Second, all scripts, programs and tools used in such transformation should be made available and well documented.
  - Third, the raw dataset (network traces) should also be made publicly available, to make the analysis of any proposed transformation easier and reproducible.
  - Fourth, count analysis of input (packets, bytes, protocols, attacks, etc. . .) from raw files should match the output (processed data) results.

- IDS practices:

  - Fifth, transformations of network traffic should focus on network level data that can be extracted from headers and avoid any processing at the payload level. This is to avoid any concern or limitation with the security and encrypted nature of traffic. Content analysis should be processed on host level, where better controls and enough information can be found for such tasks.
  - Generalisedfeaturesforallprotocolsandservicesshouldbeprohibited.Everyfeature should reflect a known computational process. For example, duration feature will be computed as the time period of connection with certain start and end criteria, where this criteria might differ from one protocol to another. Having a different computation process or different measuring criteria for the same features should be prohibited. Such inconsistences were evident in KDD, especially in Content-based features, where we can see that no one clear and unified process is used to calculate these features, even within the same protocol (TCP) for the same service (Telnet). For example, the logged_in feature in Telnet connection is incremented when a login action is detected in a connection. It is not clear how this feature contributed to services such as finger, echo, etc. that requires no login activity.

    A more recent network traffic dataset, such as the UNB ISCX Intrusion Detection Evaluation DataSet 2012 [1] should be transformed into a suitable form for ML and knowl- edge discovery applications. This leads us to suggest a collaboration project between leading research institutes and research communities to generate and share similar datasets. A new project could release the latest versions of datasets at time intervals to address changes in network behaviour and take into account changes in attack strategies or types of malware. Having a transparently created and updateable dataset for use by researchers is a core requirement for analysis and comparison. Finally, this paper urges that DARPA and KDD should no longer be used as a testbed for IDS training and evaluation.

Acknowledgments

We thank the editor and the anonymous reviewers for their constructive comments and suggestions, which provided a great help in improving this paper.

Availability of source codes

All source codes used in the analysis during this study are available from the corresponding author on request. After tidying the code, it might be placed in a public repository to be accessible for researchers.

Disclosure statement

No potential conflict of interest was reported by the authors.

Notes on contributors

Amjad M. Al Tobi received his BSc degree in Information Systems and Management Studies from Leeds University, Leeds, UK, in 2003. He completed his MSc in Computer Science from Sultan Qaboos University, Muscat, Oman, in 2008. He has completed his PhD in Network Security at the School of Computer Science, University of St Andrews, UK. He was a Deputy Director for Operations and Infrastructure at the Centre of Information Systems in Sultan Qaboos University. His experience values the importance and the need of highly intelligent IDS. This experience has planted the passion to pursue a PhD degree in the field of network anomaly intrusion detection. His current research interests include intrusion detection systems, machine learning and data mining.

Mr. Amjad M. Al Tobi (corresponding author) Centre of Information Systems,
Sultan Qaboos University,
Al-Khoud,
P.O. Box 40,
P.C. 123,
Sultanate of Oman
amjad@squ.edu.om; amhat@st-andrews.ac.uk

Ishbel Duncan gained her BSc Hons in Computational Science at the University of St Andrews in 1983. She then worked as a teacher and lecturer before several years spent in IT Services at the University of Durham led her into studying for a PhD on large scale software testing. She completed the PhD in 1994 and then moved to Australia to work with Professor Brian Henderson-Sellers on OO metrics under a research fellowship funded by AT&T at the University of Technology, Sydney. Returning to the United Kingdom, she worked with BT under a research fellowship scheme before taking a post as a Senior Lecturer at Anglia Polytechnic in Cambridge (now Anglia Ruskin) for three years. A post at Abertay University in Dundee teaching software engineering and games led to a lectureship at St Andrews where she has taught and researched in software engineering and security. Her work focuses on testing and evaluation of security systems using empirical exploration. As such, her current research is focussed on finding good datasets to define and test models, vulnerabilities or adequacy criteria for aspects of security such as intrusion or threat detection, virtualisation problems, BYOD and IoT.

Dr. Ishbel Duncan
School of Computer Science, University of St. Andrews,
North Haugh,
St. Andrews,
Fife KY16 9SX,
Scotland, UK Ishbel.Duncan@st-andrews.ac.uk

References

1. [1] Shiravi A, Shiravi H, Tavallaee M, et al. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. Computers & Security, 2012;31(3):357–374. DOI:10.1016/j.cose.2011.12.012
2. [2] 1998 DARPA Intrusion Detection Evaluation Data Set [Internet]. Massachusetts (MA): MIT Lincoln Laboratory; [cited 2016 Oct 9]. Available from: https://www.ll.mit.edu/r-d/datasets/ 1998-darpa-intrusion-detection-evaluation-data-set [dataset].
3. [3] UCI KDD Archive. KDD Cup 1999 data. Third Int Knowl Discov Data Min Tools Compet Fifth Int Conf Knowl Discov Data Min (KDD 99) ; 1999 [cited 2016 Oct 9]. Available from: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html [dataset]
4. [4] Stolfo SJ, Fan W, Lee W, et al. Cost-based Modeling for Fraud and Intrusion Detection: Results from the JAM Project. The 2000 DARPA Information Survivability Conference and Exposition (DISCEX 2000); Hilton Head, SC, USA; 2000. p. 130–144. Vol. 2. DOI:10.1109/ DISCEX.2000.821515
5. [5] Lee W, Stolfo SJ, Mok KW. Mining in a data-flow environment: experience in network intrusion detection. Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; New York, NY, USA; 1999. p. 114–124. DOI:10.1145/312129.312212
6. [6] Lee W, Stolfo SJ. A Framework for Constructing Features and Models for Intrusion Detection Systems. ACM Transactions on Information and System Security, 2000;3(4):227–261. DOI:10.1145/382912.382914
7. [7] Lee W. A Data Mining Framework for Constructing Features and Models for Intrusion Detection Systems [dissertation]. Upper Manhattan (NY): Columbia University; 1999.
8. [8] Paxson V. Bro: a system for detecting network intruders in real-time. Computer Networks, 1999;31(23):2435–2463.DOI:10.1016/S1389-1286(99)00112-7
9. [9] Perona I, Gurrutxaga I, Arbelaitz O, et al. Service-independent payload analysis to improve intrusion detection in network traffic. In Proceedings of the 7th Australasian Data Mining Conference-Volume87; Australian Computer Society, Inc., 2008. p. 171–178. URI:https://dl. acm.org/citation.cfm?id=2449315
10. [10] Brugger ST. KDD Cup '99 dataset (Network Intrusion) considered harmful. KDnuggets; 2007 [cited 2016 Oct 9]. Available from: http://www.kdnuggets.com/news/2007/n18/4i.html
11. [11] McHugh J. 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. 2000;3(4):262–294.DOI:10.1145/ 382912.382923
12. [12] McHugh J. The 1998 Lincoln laboratory IDS evaluation: a critique. In: Debar H, Mé L, Wu SF, editors. Proceedings of the Third International Workshop of Recent Advances in Intrusion Detection (RAID 2000) Vol. 1907. Toulouse, France: Springer Berlin Heidelberg; 2000. p. 145– 161. DOI:10.1007/3-540-39945-3_10
13. [13] Mahoney MV, Chan PK. An analysis of the 1999 DARPA/Lincoln laboratory evaluation data for network anomaly detection. In: Vigna G, Kruegel C, Jonsson E, editors. Recent Advances in Intrusion Detection (RAID 2003); Pittsburgh, PA, USA: Springer Berlin Heidelberg; 2003. p. 220–237. DOI:10.1007/978-3-540-45248-5_13
14. [14] Mahoney MV, Chan PK. PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffic. Melbourne, Florida (US): Florida Institute of Technology; 2001. URI:http:// hdl.handle.net/11141/94
15. [15] Tavallaee M, Bagheri E, Lu W, et al. A detailed analysis of the KDD CUP 99 data set. IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA 2009); Ottawa, Canada: IEEE Press Piscataway; 2009. DOI:10.1109/CISDA.2009.5356528.
16. [16] Wutyi KS, Thwin MMS. Heuristic Rules for Attack Detection Charged by NSL KDD Dataset. In: Zin T, Lin JW, Pan JS, et al. editors. Proceedings of the 9th International Conference of Genetic and Evolutionary Computing. Advances in Intelligent Systems and Computing (Vol. 387). Springer International Publishing, Cham; 2015. p. 137–153. DOI:10.1007/978-3-319-23204- 1_15
17. [17] Creech G, Hu J Generation of a new IDS test dataset: time to retire the KDD collection. IEEE Wireless Communications and Networking Conference (WCNC 2013; Shanghai, China; 2013. p. 4487–4492. DOI:10.1109/WCNC.2013.6555301
18. [18] Portnoy L, Eskin E, Stolfo SJ. Intrusion Detetion with Unlabeled Data Using Clustering. In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001). Philadelphia, PA, USA; 2001. p. 5–8. URI:http://citeseerx.ist.psu.edu/viewdoc/summary?doi= 10.1.1.126.2131
19. [19] Leung K, Leckie C. Unsupervised anomaly detection in network intrusion detection using clusters. In: Estivill-Castro V, editors. Proceedings of the 28th Australasian conference on Computer Science (ACSC '05) Vol. 38. Australian Computer Society, Inc. Darlinghurst, Australia; 2005. p. 333–342. URI:https://dl.acm.org/citation.cfm?id=1082198

20. [20] Bouzida Y. Principal Component Analysis for Intrusion Detection and Supervised Learning for New Attack Detection [dissertation]. Plouzané (France): Ecole Nationale Supérieure des Télécommunications de Bretagne; 2006.
21. [21] Bouzida Y, Cuppens F. Detecting known and novel network intrusions. In: Fischer-Hübner S, Rannenberg K, Yngström L, et al. editors. Security and Privacy in Dynamic Environments (SEC2006). Proceedings of the International Federation for Information Processing (IFIP) Vol. 201. Karlstad, Sweden: Springer US; 2006. p 258–270. DOI:10.1007/0-387-33406-8_22
22. [22] Sabhnani M, Serpen G. Why machine learning algorithms fail in misuse detection on KDD intrusion detection data set. Intelligent Data Analysis, 2004;8(4):403–415. URI:https://dl.acm. org/citation.cfm?id=1293811
23. [23] Sabhnani M, Serpen G. Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context. In Proceedings of the International Conference on Machine Learning: Models, Technologies and Applications (MLMTA 2003); June 23-26, 2003, Las Vegas, Nevada, USA; CSREA Press, 2003. p. 209–215. ISBN 1-932415-11-4
24. [24] Engen V, Vincent J, Phalp K. Exploring discrepancies in findings obtained with the KDD Cup '99 data set. Intelligent Data Analysis, 2011;15(2):251–276. DOI:10.3233/IDA-2010-0466
25. [25] Documentation B. Script reference: base/frameworks/notice/weird.bro; 2016 [cited 2016 Oct 9] Available from: https://www.bro.org/sphinx-git/scripts/base/frameworks/notice/weird.bro. html
26. [26] Song J, Takakura H, Okabe Y. Description of kyoto university benchmark data; 2006 [cited 2018 Jul 22]. Available from: http://www.takakura.com/Kyoto_data/BenchmarkData- Description-v5.pdf [dataset]
27. [27] Sperotto A, Sadre R, Van Vliet F, et al. A Labeled Data Set for Flow-Based Intrusion Detection. In: Nunzi G, Scoglio C, Li X, editors. International Workshop on IP Operations and Management (IPOM 2009) Vol. 5843; Berlin, Heidelberg: Springer, 2009. p. 39–50. DOI:10.1007/978-3-642-04968-2_4
28. [28] Moustafa N, Slay J. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Military Communications and Information Systems Conference (MilCIS); Canberra, ACT, Australia; IEEE, 2015. p. 1–6. DOI:10.1109/ MilCIS.2015.7348942

The following table shows some of the successfully mapped DARPA to KDD Telnet attack connections.

Attack Class	KDD					DARPA					
	Attack	Connection No.	Duration	Source Bytes	Destination Bytes	Source Port	Destination Port	Source IP	Destination IP	Simulation Day	Attack
R2L	multihop	721502	198	562	9139	8858	23	206.229.221.82	172.16.113.50	w3_thu	multihop
		1447587	718	1412	25260	1026	23	206.229.221.82	172.16.113.50	w4_fri	
	spy	1381226	337	237	1540	13114	23	208.254.251.132	172.16.112.50	w4_tue	spy
		1381227	299	112	847	13424	23	208.254.251.132	172.16.112.50	w4_tue	
U2R	buffer_overflow	805011	158	1567	3095	25722	23	209.74.60.168	172.16.112.50	w7_tue	eject
		3410716	169	1567	2857	4782	23	209.12.13.144	172.16.112.50	w5_tue	
		3410717	179	1559	2855	15646	23	209.17.189.98	172.16.112.50	w5_tue	
		3429269	49	2402	3939	15374	23	197.182.91.233	172.16.112.50	w5_wed	
		3682786	290	415	70529	32612	23	206.48.44.18	172.16.112.50	w5_fri	
		4810946	31	2402	3814	12354	23	195.73.151.50	172.16.112.50	w6_thu	
		4810947	33	2402	3815	13243	23	135.8.60.182	172.16.112.50	w6_thu	
		4810948	162	1567	2738	3563	23	202.247.224.89	172.16.112.50	w6_thu	
		4810949	127	1567	2736	14619	23	209.12.13.144	172.16.112.50	w6_thu	
		4810952	176	1559	2732	10294	23	207.253.84.13	172.16.112.50	w6_thu	
		4810954	47	2402	3816	8256	23	135.8.60.182	172.16.112.50	w6_thu	
		4810950	321	1506	1887	19020	23	199.227.99.125	172.16.112.50	w6_thu	eject-fail
		406471	150	1587	6707	25134	23	202.247.224.89	172.16.112.50	w3_mon	ffb
		805010	60	2328	4551	23147	23	206.47.98.151	172.16.112.50	w7_tue	
		1446933	113	6274	16771	9902	23	135.13.216.191	172.16.112.50	w4_fri	
		4810951	45	2336	4201	24707	23	209.154.98.104	172.16.112.50	w6_thu	
		4810953	61	2336	4194	4728	23	199.174.194.16	172.16.112.50	w6_thu	
		7450	305	1735	2766	2064	23	135.8.60.182	172.16.112.50	w1_mon	ffb_clear
		1446934	53	2628	3860	23070	23	192.168.1.10	172.16.112.50	w4_fri	format
		7449	184	1511	2957	1941	23	135.8.60.182	172.16.112.50	w1_mon	format_clear
		3682787	31	137	1351	5751	23	199.227.99.125	172.16.112.50	w5_fri	format-fail
	loadmodule	40482	79	281	1301	2619	23	135.8.60.182	172.16.113.50	w1_tue	load_clear
		841798	103	302	8876	9850	23	209.12.13.144	172.16.113.50	w7_wed	loadmodule
		1446937	31	142	1278	11910	23	135.13.216.191	172.16.112.50	w4_fri	
		1446938	21	135	1290	13745	23	135.13.216.191	172.16.112.50	w4_fri	
		3927225	85	277	693	20504	23	197.218.177.69	172.16.113.50	w5_fri	
	perl	41115	25	269	2333	17038	23	135.8.60.182	172.16.114.50	w1_tue	perl_clear
		894854	54	260	2635	8985	23	135.8.60.182	172.16.114.50	w7_thu	perlmagic
		3414775	45	268	2364	20553	23	195.115.218.108	172.16.114.50	w5_tue	
	rootkit	1391133	60	86	183	1025	23	207.230.54.203	172.16.114.50	w4_tue	rootkit
		1391134	60	90	233	1026	23	207.230.54.203	172.16.114.50	w4_tue	
		1391135	708	1727	24080	1027	23	207.230.54.203	172.16.114.50	w4_tue	
		1391137	98	621	8356	1049	23	207.230.54.203	172.16.114.50	w4_tue	
		1448890	61	294	3929	8558	23	207.230.54.203	172.16.114.50	w4_fri	

Appendix B. Connection No. 1,446,934

The following is the KDD values of connection number '1,446,934' and Telnet session content.

KDD Connection No.	**1446934**
DARPA	**Week4 Friday (192.168.1.10:23070 172.16.112.50:23)**

KDD Features [1 … 21]:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	21	…
duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_compromised	root_shell	su_attempted	num_root	num_file_creations	num_shells	num_access_files	num_outbound_cmds	is_host_login	is_guest_login	…
53	tcp	telnet	SF	2628	3860	0	0	0	3	0	1	1	1	0	0	0	0	0	0	0	0	…
Basic Features (Total: 9)									**Content Features (Total: 13)**													…

Telnet content as present in DARPA: (Shaded lines are the requests from client to server and unshaded ones are the **responses** from server to client)

```
UNIX(r) System V Release 4.0 (pascal)
alie
login: alie
ali7
Password:

Last login: Fri Jun 26 07:49:08 from listserv.com
Sun Microsystems Inc.   SunOS 5.5       Generic November 1995

Official U.S. government system for authorized use only. Do not discuss, enter, transfer, process or transmit
classified/sensitive national security information of greater sensitivity than that for which this system is
authorized. Use of the system constitutes consent to security testing and monitoring.  Unauthorized use could
result in criminal prosecution. Unauthorized use and misuse of government equipment includes, but is not limited
to, playing computer games (hack,doom), sending chain letters, gambling (sporting pools), personal business,
pornography, or anything that can offend or be construed as sexual harassment.
which gcc
pascal>

pascal> which gcc

/bin/gcc
uudecode<<XX898872970XX\`
pascal> uudecode<<XX898872970XX\`
⋮
... TRUNCATED  ...  Loading file content  ...
⋮
end
? end
XX898872970XX\`
? XX898872970XX\`
/bin/gcc -o /tmp/224332 /tmp/22433.c
pascal> /bin/gcc -o /tmp/224332 /tmp/22433.c
which gcc
pascal> which gcc

/bin/gcc
uudecode<<XX898872979XX\`
pascal> uudecode<<XX898872979XX\`
⋮
... TRUNCATED  ...  Loading file content  ...
⋮
end
? end
XX898872979XX\`
```

```
? XX898872979XX\`

/bin/gcc -o /tmp/224333 /tmp/22433.c
pascal> /bin/gcc -o /tmp/224333 /tmp/22433.c

/tmp/224332
pascal> /tmp/224332

Jumping to address 0xeffff6b8 B[364] E[400] SO[704]

/tmp/224333
#

/tmp/224333

# # # #

# ^D..exit
.
#
^D..#
.
pascal>

pascal> ^D..logout
.
```

Appendix C. Connection No. 3,429,269

The following is the KDD values of connection number '3,429,269' and Telnet session content.

KDD Connection No.	**3429269**
DARPA	**Week5 Wednesday (197.182.91.233:15374 172.16.112.50:23)**

KDD Features [1 … 21]:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	21	…
duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_compromised	root_shell	su_attempted	num_root	num_file_creations	num_shells	num_access_files	num_outbound_cmds	is_host_login	is_guest_login	…
49	tcp	telnet	SF	2402	3939	0	0	0	4	0	1	2	1	0	0	0	0	0	0	0	0	…
Basic Features (Total: 9)									**Content Features (Total: 13)**													…

Telnet content as present in DARPA: (Shaded lines are the requests from client to server and unshaded ones are the **responses** from server to client)

```
UNIX(r) System V Release 4.0 (pascal)
alie
login: alie
ali7
Password:

Last login: Wed Jul  1 16:12:34 from 194.27.251.21
Sun Microsystems Inc.   SunOS 5.5       Generic November 1995

Official U.S. government system for authorized use only. Do not discuss, enter, transfer, process or transmit
classified/sensitive national security information of greater sensitivity than that for which this system is
authorized.  Use of the system constitutes consent to security testing and monitoring.  Unauthorized use could
result in criminal prosecution. Unauthorized use and misuse of government equipment includes, but is not limited
to, playing computer games (hack,doom), sending chain letters, gambling (sporting pools), personal business,
pornography, or anything that can offend or be construed as sexual harassment.

June 25'th 1998:
Due to severe thunderstorms over Eyrie AFB, many of the systems on the EAFB network were knocked out, and network
traffic to the rest of the internet was disrupted.  Most systems are now back on line, and we expect all remaining
services to be restored shortly.
which gcc
pascal>

pascal> which gcc

/bin/gcc
uudecode<<XX899347368XX\`
pascal> uudecode<<XX899347368XX\`
┆
... TRUNCATED  ...  Loading file content  ...
┆
end
? end
XX899347368XX\`
? XX899347368XX\`
/bin/gcc -o /tmp/178572 /tmp/17857.c
pascal> /bin/gcc -o /tmp/178572 /tmp/17857.c
which gcc
pascal> which gcc

/bin/gcc
uudecode<<XX899347375XX\`
pascal> uudecode<<XX899347375XX\`
┆
... TRUNCATED  ...  Loading file content  ...
┆
```

```
end
? end

XX899347375XX\`
? XX899347375XX\`

/bin/gcc -o /tmp/178573 /tmp/17857.c
pascal> /bin/gcc -o /tmp/178573 /tmp/17857.c

/tmp/178572
pascal> /tmp/178572

Jumping to address 0xeffff7e0
Jumping to address 0xeffff7e0 B[364] E[400] SO[400]



/tmp/178573
#


/tmp/178573


# # #
^D..#

# exit
.
#
^D..#
.
pascal>

pascal> ^D..logout
.
```

Appendix D. Connection No. 7449

The following is the KDD values of connection number '7449' and Telnet session content.

KDD Connection No.	7449
DARPA	**Week1 Monday (135.8.60.182:1941 172.16.112.50:23)**

KDD Features [1 … 21]:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	21	…
duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_compromised	root_shell	su_attempted	num_root	num_file_creations	num_shells	num_access_files	num_outbound_cmds	is_host_login	is_guest_login	…
184	tcp	telnet	SF	1511	2957	0	0	0	3	0	1	2	1	0	0	1	0	0	0	0	0	…
Basic Features (Total: 9)									Content Features (Total: 13)													…

Telnet content as present in DARPA: (Shaded lines are the requests from client to server and unshaded ones are the **responses** from server to client)

```
UNIX(r) System V Release 4.0 (pascal)

tristank
login: tristank
AqEwoPJN
Password:
Sun Microsystems Inc.   SunOS 5.5       Generic November 1995

Official U.S. government system for authorized use only. Do not discuss, enter, transfer, process or transmit
classified/sensitive national security information of greater sensitivity than that for which this system is
authorized.  Use of the system constitutes consent to security testing and monitoring.  Unauthorized use could
result in criminal prosecution. Unauthorized use and misuse of government equipment includes, but is not limited
to, playing computer games (hack,doom), sending chain letters, gambling (sporting pools), personal business,
pornography, or anything that can offend or be construed as sexual harassment.

tcsh: using dumb terminal settings.

which gcc
pascal> which gcc

/bin/gcc
cat > ffbexploit.c
pascal> cat > ffbexploit.c
⋮
... TRUNCATED  ...  Loading file content  ...
⋮
./bin/gcc -o ffbexploit ffbexploit.c
^D..pascal> /bin/gcc -o ffbexploit ffbexploit.c

./ffbexploit
pascal> ./ffbexploit

ffbconfig: ............................................................-......n/......h...............;...#...#..
.#.... ;.. ....... ... ....`...`...`...`...`...`...`...`...`...`...`...`...`...`...`...`...`...`...`...`...`..
.`...`...`...`...`...`...`...`...`...`...`...`...`...`...`...`...`...`...`...`...`...`...`...`...`...`...`...`
...`...`...`...`...`...`...`...`...`...`...`...`...`: No such file or directory


#
^D..#
.
pascal>

pascal> ^D..logout
.
```

Appendix E. Connection No. 3,927,225

The following is the KDD values of connection number '3,927,225' and Telnet session content.

KDD Connection No.	**3927225**
DARPA	**Week5 Friday (197.218.177.69:20504 172.16.113.50:23)**

KDD Features [1 … 21]:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	21	…
duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_compromised	root_shell	su_attempted	num_root	num_file_creations	num_shells	num_access_files	num_outbound_cmds	is_host_login	is_guest_login	…
85	tcp	telnet	SF	277	693	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	…
Basic Features (Total: 9)									**Content Features (Total: 13)**													…

Telnet content as present in DARPA: (Shaded lines are the requests from client to server and unshaded ones are the **responses** from server to client)

```
SunOS UNIX (zeno)

wardc
login: wardc
nqXiqjyA
Password:
SunOS Release 4.1.4 (LUGH) #1: Tue Oct 10 12:12:31 EDT 1995
which loadmodule
zeno> which loadmodule

loadmodule: Command not found.
whereis loadmodule
zeno> whereis loadmodule

loadmodule:
pushd /usr/openwin/bin
zeno> pushd /usr/openwin/bin

/usr/openwin/bin ~
ls -F loadmodule
zeno> ls -F loadmodule

loadmodule*
popd
zeno> popd

~
cat > bin
zeno> cat > bin
⋮
... TRUNCATED  ...  Loading file content  ...
⋮
.chmod 755 bin
^D..zeno> chmod 755 bin

sh -c "IFS=/ /usr/openwin/bin/loadmodule a b"
zeno> sh -c "IFS=/ /usr/openwin/bin/loadmodule a b"

/usr/openwin/bin/loadmodule: /usr/sys/sun4/OBJ/a file does not exist.
Check your OpenWindows installation.
rm bin
zeno> rm bin

./sh
zeno> ./sh


#
.
^D..# zeno>

zeno> ^D..logout
.
```

Appendix F. Connection No. 406,471

The following is the KDD values of connection number '406,471' and Telnet session content.

KDD Connection No.	**406471**
DARPA	**Week3 Monday (202.247.224.89:25134 172.16.112.50:23)**

KDD Features [1 … 21]:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	21	…
duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_compromised	root_shell	su_attempted	num_root	num_file_creations	num_shells	num_access_files	num_outbound_cmds	is_host_login	is_guest_login	…
150	tcp	telnet	SF	1587	6707	0	0	0	1	0	1	3	0	0	1	1	0	0	0	0	0	…
Basic Features (Total: 9)									**Content Features (Total: 13)**													…

Telnet content as present in DARPA: (Shaded lines are the requests from client to server and unshaded ones are the **responses** from server to client)

```
UNIX(r) System V Release 4.0 (pascal)

clintonl
login: clintonl
clinton7
Password:
Last login: Mon Jun 15 11:26:30 from 194.27.251.21
Sun Microsystems Inc.   SunOS 5.5       Generic November 1995

Official U.S. government system for authorized use only. Do not discuss, enter, transfer, process or transmit
classified/sensitive national security information of greater sensitivity than that for which this system is
authorized.  Use of the system constitutes consent to security testing and monitoring.  Unauthorized use could
result in criminal prosecution. Unauthorized use and misuse of government equipment includes, but is not limited
to, playing computer games (hack,doom), sending chain letters, gambling (sporting pools), personal business,
pornography, or anything that can offend or be construed as sexual harassment.

who
pascal> who

bramy      console      Jun 15 07:50
williamf   pts/0        Jun 15 08:18    (alpha.apple.edu)
clintonl   pts/1        Jun 15 11:32    (202.247.224.89)
janes      pts/2        Jun 15 08:20    (alpha.apple.edu)
georgeb    pts/3        Jun 15 09:07    (alpha.apple.edu)
donaldh    pts/5        Jun 15 08:39    (alpha.apple.edu)
fredd      pts/6        Jun 15 08:54    (alpha.apple.edu)
ls
pascal> ls

src
cat > ffb.c << EOF
pascal> cat > ffb.c << EOF
⋮
... TRUNCATED  ...  Loading file content  ...
⋮
gcc -o ffb ffb.c
pascal> gcc -o ffb ffb.c

./ffb
pascal> ./ffb

Jumping to address 0xeffffa68 B[128] E[256] SO[128]
ffbconfig: ..............................................................-......n/......h..............;...#...#..
.#.... ;.. ....... ... ....h...h...h...h...h...h...h...h...h...h...h...h...h...h...h...h...h...h...h...h...h...h..
.h...h...h...h...h...h...h...h...h...h...h...h...h...h...h...h...h...h...h...h...h...h...h...h...h...h...h...h...h
...h...h...h...h...h...h...h...h...h...h...h...h...h...h: No such file or directory
id
# id
uid=2050(clintonl) gid=100(users) euid=0(root)
ls /home
```

```
# ls /home
TT_DB       desmonds    gwendolv    lanaa       orionc      triav
abramh      doireano    haraldl     lavernel    parkerm     tristank
adrieni     donaldh     henningm    leandere    quintond    ulandusm
alie        elmoc       henriker    liliana     rachaelc    valeskad
ansgarz     emonc       huws        lost+found  raeburnt    victors
avrap       erink       hyacintl    lucyj       randip      violetp
bedeliaa    felinai     inghami     lupitam     rexn        virginil
bellej      finnm       ingolfk     margarej    reynaldv    wardc
bramy       franko      jackj       mariaht     roderica    williamf
camronw     fredd       janes       mariel      romeob      wojciecd
cartert     ftp         janinee     marilenc    selmam      yannisb
charlab     galeo       jaroslan    marlenag    soniac      yuvalt
charlotk    geoffp      jennifed    marlync     sumikop     yvonnea
christim    georgeb     joelo       marlyy      suser       yvonnej
cliffu      georgind    jouniw      mistyd      suzannac    zephyro
clintonl    giovanng    katinas     operator    suzannas
darleent    grzegors    kiaraa      orindag     tonyae
```

```
cd /var
```

```
# cd /var
```

```
cd log
```

```
# cd log
```

```
ls
```

```
# ls
authlog          dead.letter      sysidconfig.log  syslog
```

```
tail syslog
```

```
# tail syslog
Jun 15 11:30:21 pascal sendmail[3149]: LAA03148:
to=wardc@pascal.eyrie.af.mil,emonc@pascal.eyrie.af.mil,geoffp@pascal.eyrie.af.mil,janinee@pascal.eyrie.af.mil,
ctladdr=geoffp (2065/100), delay=00:00:01, mailer=local, stat=Sent
Jun 15 11:31:12 pascal sendmail[3156]: LAA03156: from=<denises@epsilon.pear.com>, size=2384, class=0, pri=32384,
nrcpts=1, msgid=<19980615113112.CAA989>, proto=ESMTP, relay=epsilon.pear.com [195.115.218.108]
Jun 15 11:31:13 pascal sendmail[3157]: LAA03156: to=<geoffp@pascal.eyrie.af.mil>, delay=00:00:01, mailer=local,
stat=Sent
Jun 15 11:32:16 pascal sendmail[3149]: LAA03148: to=lolaa@delta.peach.mil, ctladdr=geoffp (2065/100),
delay=00:01:56, mailer=ether, relay=delta.peach.mil [194.7.248.153], stat=Sent (Mail accepted)
Jun 15 11:32:17 pascal sendmail[3149]: LAA03148: to=joyh@gamma.grape.mil,georginl@gamma.grape.mil, ctladdr=geoffp
(2065/100), delay=00:01:57, mailer=ether, relay=gamma.grape.mil [194.27.251.21], stat=Sent (Mail accepted)
Jun 15 11:32:17 pascal sendmail[3149]: LAA03148: to=wilburs@finch.eyrie.af.mil, ctladdr=geoffp (2065/100),
delay=00:01:57, mailer=ether, relay=finch.eyrie.af.mil. [172.16.114.168], stat=Sent (Mail accepted)
Jun 15 11:32:18 pascal sendmail[3149]: LAA03148: to=shirralm@saturn.kiwi.org, ctladdr=geoffp (2065/100),
delay=00:01:58, mailer=ether, relay=saturn.kiwi.org [196.227.33.189], stat=Sent (Mail accepted)
Jun 15 11:32:19 pascal sendmail[3149]: LAA03148: to=denises@epsilon.pear.com, ctladdr=geoffp (2065/100),
delay=00:01:59, mailer=ether, relay=epsilon.pear.com [195.115.218.108], stat=Sent (Mail accepted)
Jun 15 11:33:09 pascal sendmail[3178]: LAA03178: from=fredd, size=53, class=0, pri=30053, nrcpts=1,
msgid=<199806151533.LAA03178@pascal.>, relay=fredd@localhost
Jun 15 11:33:09 pascal sendmail[3179]: LAA03178: to=williamf@pascal.eyrie.af.mil, ctladdr=fredd (2139/100),
delay=00:00:00, mailer=local, stat=Sent
```

```
exit
```

```
# exit
```

```
logout
```

```
pascal> logout
```