Computer network intrusion detection

KDD CUP 1999

INTRODUCTION

the KDD Cup 1999 dataset is a famous dataset used for the task of computer network intrusion detection. It was created as part of the Third International Knowledge Discovery and Data Mining Tools Competition held in conjunction with the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99).

The dataset contains network traffic data that simulates a typical U.S. Air Force LAN. It includes a variety of features extracted from network packets, such as protocol types, service types, flag values, and more. Each data instance is labeled as either normal or an attack (with multiple attack types present in the dataset).

The goal of the competition was to develop models that could accurately classify network connections as either normal or attacks, and if an attack, classify the type of attack. This task is crucial for building effective intrusion detection systems to protect computer networks from malicious activities.

The dataset has been widely used in research and benchmarking of intrusion detection algorithms and machine learning models. However, it's worth noting that the dataset is quite dated, and network intrusion detection methods have evolved since then. Therefore, it's often used for benchmarking purposes rather than as a representation of current network traffic.

This is the data set used for The Third International Knowledge Discovery and Data Mining Tools Competition, which was held in conjunction with KDD-99 The Fifth International Conference on Knowledge Discovery and Data Mining. The competition task was to build a network intrusion detector, a predictive model capable of distinguishing between bad'' connections, called intrusions or attacks, andgood'' normal connections. This database contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment.

the header of data set

on the first the data have one header on the original data set but there is a file with the data came with separated from the data

and the content of the header is :

1. **duration** : the length of time how long something lasts, from beginning to end
2. **protocol_type** : a protocol is a standardized set of rules for formatting and processing data
3. **service :** Network Service – a capability that facilitates a network operation
   and the service type on this data is:
   - 3.0 **HTTP** (Hypertext Transfer Protocol): Used for transferring hypertext requests and information on the World Wide Web.
   - 3.1 **SMTP** (Simple Mail Transfer Protocol): Used for sending and receiving email messages between servers.
   - 3.2 **Domain_u**: This might refer to domain users or user authentication within a domain, but without more context, it's difficult to specify.
   - 3.3 **Auth**: Short for authentication, typically refers to the process of verifying the identity of a user or system.

3.4 **Finger**: An outdated protocol used to retrieve information about users on a network.

3.5 **Telnet**: A protocol used for remote terminal access or control of computers over a network.

3.6 **Eco_i**: This is not a standard protocol. It might be a specific service or protocol related to eco-friendly initiatives or systems, but without context, it's unclear.

3.7 **FTP** (File Transfer Protocol): Used for transferring files between a client and a server on a network.

3.8 **NTP_u**: Network Time Protocol (NTP) is used for clock synchronization between computer systems. The "_u" may refer to "unprivileged" mode.

3.9 **Ecr_i**: Similar to "Eco_i", this is not a standard protocol. It might be a specific service or protocol related to eco-friendly initiatives or systems, but without context, it's unclear.

3.10 **Other**: This likely refers to miscellaneous or unspecified network protocols or services.

3.11 **Urp_i**: This is not a standard protocol. Without more context, it's unclear what it refers to.

3.12 **Private**: Likely indicates private or internal network traffic that isn't explicitly defined by standard protocols.

3.13 **POP_3**: Post Office Protocol version 3, used for retrieving email from a remote server.

3.14 **FTP_data**: Specifically refers to FTP data transfer, separate from the control channel.

3.15 **Netstat**: A command-line tool used to display network connections, routing tables, interface statistics, masquerade connections, and multicast memberships.

3.16 **Daytime**: A service that provides the current date and time.

3.17 **SSH** (Secure Shell): A cryptographic network protocol for secure data communication, remote command-line login, remote command execution, and other secure network services between two networked computers.

3.18 **Echo**: A service used to test the reachability of a host on an IP network and to measure the round-trip time for messages sent from the originating host to a destination computer.

3.19 **Time**: A service that provides the current time.

3.20 **Name**: This could refer to various naming services or protocols without more context.

3.21 **Whois**: A protocol and database for retrieving registration information about domain names, IP addresses, and autonomous systems.

3.22 **Domain**: This could refer to the Domain Name System (DNS) or other domain-related protocols or services.

3.23 **MTP** (Message Transfer Protocol): A protocol used for transferring electronic mail messages between computers.

3.24 **Gopher**: An early protocol designed for distributing, searching, and retrieving documents over the internet.

3.25 **Remote_job**: This is not a standard protocol. Without more context, it's unclear what it refers to.

3.26 **Rje**: Remote Job Entry, a protocol used to submit batch jobs to remote batch processing systems.

3.27 **Ctf**: This is not a standard protocol. Without more context, it's unclear what it refers to.

3.28 **Supdup**: A protocol used for remote login and execution on other machines.

3.29 **Link**: This could refer to various link-layer protocols or services without more context.

3.30 **Systat**: A command-line tool used to display information about current users on a system.

3.31 **Discard**: A service that discards all data sent to it without responding.

3.32 **X11**: A protocol for remote graphical user interface (GUI) access.

3.33 **Shell**: This could refer to various shell-related protocols or services without more context.

3.34 **Login**: This could refer to various login-related protocols or services without more context.

3.35 **IMAP4** (Internet Message Access Protocol version 4): A protocol used for retrieving email from a remote server.

3.36 **NNTP** (Network News Transfer Protocol): A protocol used for reading and posting Usenet articles.

3.37 **UUCP** (Unix-to-Unix Copy Protocol): A protocol used to send and receive files between Unix systems.

3.38 **Pm_dump**: This is not a standard protocol. Without more context, it's unclear what it refers to.

3.39 **IRC** (Internet Relay Chat): A protocol for real-time text messaging and communication in channels or groups.

3.40 **Z39_50**: A protocol for searching and retrieving information from library databases.

3.41 **Netbios_dgm**: NetBIOS Datagram Service, used for connectionless communication between computers in a LAN.

3.42 **LDAP** (Lightweight Directory Access Protocol): A protocol for accessing and maintaining directory services over a network.

3.43 **Sunrpc** (Sun Remote Procedure Call): A protocol used for interprocess communication between networked systems.

3.44 **Courier**: This could refer to various courier-related protocols or services without more context.

3.45 **Exec**: A service that allows a user to execute commands on a remote system.

3.46 **BGP** (Border Gateway Protocol): A protocol used to exchange routing information between different autonomous systems on the internet.

3.47 **Csnet_ns**: This is not a standard protocol. Without more context, it's unclear what it refers to.

3.48 **HTTP_443**: HTTP protocol over port 443, typically used for secure web browsing (HTTPS).

3.49 **Klogin**: Kerberized login, a protocol for securely logging into a remote system using Kerberos authentication.

3.50 **Printer**: This likely refers to printing services or protocols used for network printing.

3.51 **Netbios_ssn**: NetBIOS Session Service, used for establishing and managing sessions between computers in a LAN.

3.52 **POP_2**: Post Office Protocol version 2, an older protocol used for retrieving email from a remote server.

3.53 **NNTP** (Network News Transfer Protocol): A protocol used for reading and posting Usenet articles.

3.54 **Efs**: This is not a standard protocol. Without more context, it's unclear what it refers to.

3.55 **Hostnames**: This could refer to various hostname-related protocols or services without more context.

3.56 **UUCP_path**: This is not a standard protocol. Without more context, it's unclear what it refers to.

3.57 **SQL_net**: This is not a standard protocol. Without more context, it's unclear what it refers to.

3.58 **Vmnet**: This is not a standard protocol. Without more context, it's unclear what it refers to.

3.59 **Iso_tsap**: This is not a standard protocol. Without more context, it's unclear what it refers to.

3.60 **Netbios_ns**: NetBIOS Name Service, used for name registration and resolution in a LAN.

3.61 **Kshell**: Kerberized shell, a protocol for securely executing commands on a remote system using Kerberos authentication.

3.62 **Urh_i**: This is not a standard protocol. Without more context, it's unclear what it refers to.

3.63 **HTTP_2784**: HTTP protocol over port 2784, which might be a non-standard port used for web traffic.

3.64 **Harvest**: This is not a standard protocol. Without more context, it's unclear what it refers to.

3.65 **Aol**: This could refer to America Online (AOL) services, but without context, it's unclear.

3.66 **Tftp_u**: Trivial File Transfer Protocol (TFTP) in unprivileged mode, typically used for simple file transfers.

3.67 **HTTP_8001**: HTTP protocol over port 8001, which might be a non-standard port used for web traffic.

3.68 **Tim_i**: This is not a standard protocol. Without more context, it's unclear what it refers to.

3.69 **Red_i**: This is not a standard protocol. Without more context, it's unclear what it refers to.

4. **flag:** refers to a parameter or indicator within a network packet or frame that carries additional information about the packet's characteristics or handling instructions.
   Type of flag is :

   4.0 **SF** (Syn/FIN): This state indicates that a TCP connection was successfully established and then closed gracefully, typically indicating an open port. It means that the client sent a SYN packet to initiate the connection and then sent a FIN packet to gracefully close the connection after completing the communication.

   4.1 **S2**: This state typically indicates that a TCP connection attempt was received by the target system but didn't progress to a complete connection. It may indicate that the connection attempt was blocked or filtered by a firewall or intrusion prevention system.

   4.2 **S1**: Similar to S2, S1 typically indicates that the connection attempt was received by the target system, but it didn't progress to a complete connection. The exact meaning may vary depending on the tool or context.

   4.3 **S3**: Similar to S2 and S1, S3 indicates that a TCP connection attempt was received by the target system but didn't progress to a complete connection. The exact interpretation may vary depending on the tool or context.

   4.4 **OTH** (Other): This state indicates that the TCP connection attempt encountered an unexpected or unusual response from the target system, which doesn't fit into any of the standard connection states.

4.5 **REJ** (Reject): This state indicates that the TCP connection attempt was rejected by the target system. It typically means that the port is closed, and the target system actively refused the connection.

4.6 **RSTO** (Reset Originator): This state indicates that the TCP connection attempt was rejected by the target system with a TCP reset (RST) packet. It usually means that the port is closed, and the target system sent a reset packet to terminate the connection attempt.

4.7 **S0**: This state indicates that the TCP connection attempt was not acknowledged by the target system. It typically means that the port is closed, but it didn't respond at all to the connection attempt.

4.8 **RSTR** (Reset Response): This state indicates that the TCP connection attempt was received by the target system, but the connection was reset by the target system with a TCP reset (RST) packet.

4.9 **RSTOS0** (Reset Originator and Stealth): This state indicates that the TCP connection attempt was rejected by the target system with a TCP reset (RST) packet, and the target system didn't send back any response.

4.10 **SH** (Stealth Scan): This state indicates that the TCP connection attempt was received by the target system, but the connection was closed immediately without completing the handshake process. It's often associated with stealth scanning techniques that attempt to avoid detection by not completing the connection establishment process.

5. **src_bytes**: The number of bytes sent by the source (sender) in a network communication.

6.**dst_bytes**: The number of bytes received by the destination (receiver) in a network communication.

7. **land**: Indicates whether the connection is from/to the same host/port (1 if true, 0 otherwise). It's used to detect the "Land" denial-of-service attack.

8. **wrong_fragment**: The number of "wrong" fragments (fragments with a bad checksum) in the packet.

9. **urgent**: Indicates whether the Urgent pointer field is set in the TCP header.

10. **hot**: The number of "hot" indicators detected.

11. **num_failed_logins**: The number of failed login attempts.

12. **logged_in**: Indicates whether the user is logged in (1 if true, 0 otherwise).

13. **num_compromised**: The number of compromised conditions.

14. **root_shell**: Indicates whether a root shell session is present (1 if true, 0 otherwise).

15. **su_attempted**: Indicates whether the "su root" command was attempted (1 if true, 0 otherwise).

16. **num_root**: The number of "root" accesses.

17. **num_file_creations**: The number of file creation operations.

18. **num_shells**: The number of shell prompts.

19. **num_access_files**: The number of access control files.

20. **num_outbound_cmds**: The number of outbound commands in an ftp session.

21. **is_host_login**: Indicates whether the login belongs to the "host" category (1 if true, 0 otherwise).

22. **is_guest_login**: Indicates whether the login is a "guest" login (1 if true, 0 otherwise).

23. **count**: The number of connections to the same host as the current connection in the past 2 seconds.

24. **srv_count**: The number of connections to the same service (port) as the current connection in the past 2 seconds.

25. **serror_rate**: The percentage of connections that have "SYN" errors.

26. **srv_serror_rate**: The percentage of connections to the same service (port) that have "SYN" errors.

27. **rerror_rate**: The percentage of connections that have "REJ" errors.

28. **srv_rerror_rate**: The percentage of connections to the same service (port) that have "REJ" errors.

29. **same_srv_rate**: The percentage of connections to the same service (port).

30. **diff_srv_rate**: The percentage of connections to different services (ports).

31. **srv_diff_host_rate**: The percentage of connections to different hosts among the same service (port).

32. **dst_host_count**: The number of connections to the same destination host as the current connection in the past 2 seconds.

33. **dst_host_srv_count**: The number of connections to the same service (port) on the destination host as the current connection in the past 2 seconds.

34. **dst_host_same_srv_rate**: The percentage of connections to the same service (port) on the destination host.

35. **dst_host_diff_srv_rate**: The percentage of connections to different services (ports) on the destination host.

36. **dst_host_same_src_port_rate:** The percentage of connections from the same source port.

37. **dst_host_srv_diff_host_rate**: The percentage of connections to the same service (port) from different hosts.

38. **dst_host_serror_rate**: The percentage of connections to the destination host that have "SYN" errors.

39. **dst_host_srv_serror_rate**: The percentage of connections to the destination host and service (port) that have "SYN" errors.

40. **dst_host_rerror_rate**: The percentage of connections to the destination host that have "REJ" errors.

41. **dst_host_srv_rerror_rate**: The percentage of connections to the destination host and service (port) that have "REJ" errors.

42. **outcome :**

42.0 normal.: Represents normal network traffic or activity without any malicious intent or behavior. This serves as a baseline for comparison against potentially malicious activities.

42.1 buffer_overflow.: An attack where a program writes more data to a buffer than it can hold, potentially leading to the execution of arbitrary code or a system crash.

42.2 loadmodule.: An attempt to load a kernel module or extension into a system, which could be used to gain unauthorized access or escalate privileges.

42.3 perl.: Indicates the use of Perl scripts or Perl-based attacks to exploit vulnerabilities or execute malicious actions on a system.

42.4 neptune.: Refers to a denial-of-service (DoS) attack designed to overwhelm a target system or network with a flood of TCP/IP packets, rendering it inaccessible to legitimate users.

42.5 smurf.: Another type of DoS attack that relies on sending spoofed ICMP echo request (ping) packets to a large number of hosts, causing them to reply to the victim's IP address and flood it with traffic.

42.6 guess_passwd.: An attack where an attacker attempts to gain unauthorized access to a system by guessing usernames and passwords.

42.7 pod.: Stands for "Ping of Death," a type of DoS attack that sends oversized or malformed packets to a target system, causing it to crash or become unresponsive.

42.8 teardrop.: Similar to the "pod" attack, this involves sending fragmented packets with overlapping payloads to confuse or crash the target system's network stack.

42.9 portsweep.: An attempt to scan a range of ports on a target system to identify open ports and potential vulnerabilities.

42.10 ipsweep.: A reconnaissance attack where an attacker scans a range of IP addresses to identify active hosts on a network.

42.11 land.: Indicates a "Land" attack, where a malicious party sends spoofed packets with the same source and destination IP addresses, causing the target system to crash or hang.

42.12 ftp_write.: An attack where an unauthorized user gains write access to an FTP server, allowing them to upload or modify files.

42.13 back.: Refers to a backdoor or unauthorized access point left by an attacker on a compromised system for future exploitation.

42.14 imap.: Indicates an attack targeting the Internet Message Access Protocol (IMAP), which could involve unauthorized access to email accounts or sensitive information.

42.15 satan.: Refers to the SATAN (Security Administrator Tool for Analyzing Networks) vulnerability scanner, which could be used by attackers to identify weaknesses in a network.

42.16 phf.: Stands for "Process HTTP Forbidden," an attack where an attacker attempts to exploit vulnerabilities in the CGI-BIN directory of a web server to gain unauthorized access.

42.17 nmap.: Indicates the use of the Nmap network scanning tool, commonly used for network discovery, port scanning, and vulnerability assessment.

42.18 multihop.: An attack involving multiple compromised hosts used as intermediaries to hide the identity of the attacker or to launch further attacks.

42.19 warezmaster.: Refers to a user or system acting as a central repository or distributor of pirated software (warez).

42.20 warezclient.: Indicates a user or system downloading or accessing pirated software or content from a warez server.

42.21 spy.: Indicates espionage or unauthorized monitoring activities aimed at gathering sensitive information.

42.23 rootkit.: A type of malicious software designed to gain unauthorized access to a system and maintain persistent control while hiding its presence from detection.

Data problems

1.Data have to be encoded

On this data set there are a bunch of columns have to be encoded but there are some of the columns ready encoded on this table will explain

Those columns and the strategy to encode those columns

The columns	Encoded or not	Encoder strategy
protocol_type	NO	Target Encoder
service	NO	Target Encoder
flag	NO	Target Encoder
land	YES	Label Encoder
logged_in	YES	Label Encoder
root_shell	YES	Label Encoder
su_attempted	YES	Label Encoder
is_host_login	YES	Label Encoder
is_guest_login	YES	Label Encoder

2. missing data :this data came without any null value:

```
#########################################################################

#check if ther is null value
#pd.value_counts(data1[41]).sum()
null_count =data1.isnull().sum().sum() #if i want the totall
print("Number of null value is = \n",null_count)

#########################################################################
```

```
In [9]: runcell(0, 'C:/Users/LENOVO/Desktop/folders/python files/web/web 8.py')
Number of null value is =
 0
        duration protocol_type  ... dst_host_srv_rerror_rate  outcome
1769           0           tcp  ...                      0.0  normal.
1770           0           tcp  ...                      0.0  normal.
1771           0           tcp  ...                      0.0  normal.
```

3. duplicated row : this data have 41 column with 4896343 rows

After checking out there is a 3823439 row are duplicated so we drop it out :

```
#########################################################################
#check for duplicates

duplicates = data1.duplicated()

print(data1[duplicates])
num_duplicates = duplicates.sum()
print("Number of duplicated rows:", num_duplicates)
#########################################################################
#for first test drop the duplicated value
data2 = data1.drop_duplicates()
print(data2.info())
print(data2.head())
#########################################################################
```

4. mix data type : use the parameter on Readcsv (skip_bad_line) we solve this issue

5. the big size of the data: also with Readcsv parameter (chunksize) we read the file 10000 chunk at the time then using pd.concat(data) we reassemble it .

6. no header for the data : to solve this issue we give the name of the columns form outside reference and add it to the file also with Readcsv parameter (names)

On this pictures the code we use :

```
#################################################################
#read csv file
path='C:/Users/LENOVO/Desktop/folders/data2/data.csv'
data = pd.read_csv('C:/Users/LENOVO/Desktop/folders/data2/data.csv', chunksize=10000, on_bad_lines='skip', names=header)
data1 = pd.concat(data)


#################################################################
```

Summary

The KDD Cup 1999 dataset serves as a fundamental resource in the realm of computer network intrusion detection, originating from a renowned competition aimed at fostering advancements in knowledge discovery and data mining. This dataset emulates a standard U.S. Air Force LAN environment, encompassing a plethora of features derived from network packets, including protocol types, service types, flag values, and more. Each data instance is meticulously labeled as either "normal" or indicative of an attack, with diverse attack types represented within the dataset.

Participants in the competition were tasked with constructing predictive models capable of discerning between benign and malicious network connections, with a further requirement of

categorizing the nature of the attacks if detected. This endeavor bears profound significance, as it underscores the imperative of developing robust intrusion detection systems to safeguard computer networks from nefarious activities.

While the dataset has been extensively employed for research and benchmarking purposes, it is crucial to acknowledge its temporal context, as network intrusion detection methodologies have evolved since its inception. Nonetheless, its enduring utility as a benchmarking tool persists, facilitating ongoing investigations into intrusion detection algorithms and machine learning models.

However, the dataset is not without its challenges. Encoding categorical variables, addressing missing data, handling duplicated rows, managing mixed data types, and grappling with the dataset's substantial size are among the key obstacles encountered during preprocessing. Strategic approaches, such as target encoding for categorical variables and label encoding for boolean attributes, alongside systematic data cleaning procedures, play pivotal roles in mitigating these challenges and ensuring the dataset's suitability for analysis.

In essence, the KDD Cup 1999 dataset stands as a testament to the interdisciplinary synergy between data mining, cybersecurity, and network analysis. Its enduring relevance underscores its status as a foundational resource for advancing intrusion detection research and fortifying the resilience of computer networks against evolving threats.