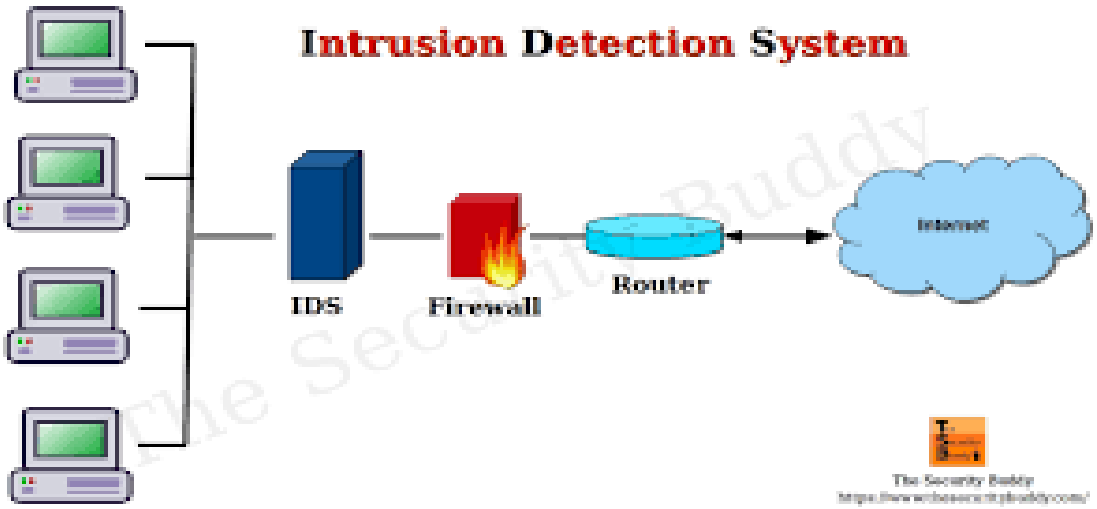
Name – Dhruva Kapadia
MIS- 111903142
Subject- Cyber Security

Assignment 5 : Implement any IDS/IPS system

An **Intrusion Detection System (IDS)** is a system that monitors network traffic for suspicious activity and issues alerts when such activity is discovered. It is a software application that scans a network or a system for harmful activity or policy breaching. Any malicious venture or violation is normally reported either to an administrator or collected centrally using a security information and event management (SIEM) system. A SIEM system integrates outputs from multiple sources and uses alarm filtering techniques to differentiate malicious activity from false alarms.



Classification of Intrusion Detection System:

IDS are classified into 5 types:

1. **Network Intrusion Detection System (NIDS):**

Network intrusion detection systems (NIDS) are set up at a planned point within the network to examine traffic from all devices on the network. It performs an observation of passing traffic on the entire subnet and matches the traffic that is passed on the subnets to the collection of known attacks. Once an attack is identified or abnormal behavior is observed, the alert can be sent to the administrator.

2. **Host Intrusion Detection System (HIDS):**

Host intrusion detection systems (HIDS) run on independent hosts or devices on the network. A HIDS monitors the incoming and outgoing packets from the device only and will alert the administrator if suspicious or malicious activity is detected. It takes a snapshot of existing system files and compares it with the previous snapshot.

3. **Protocol-based Intrusion Detection System (PIDS):**

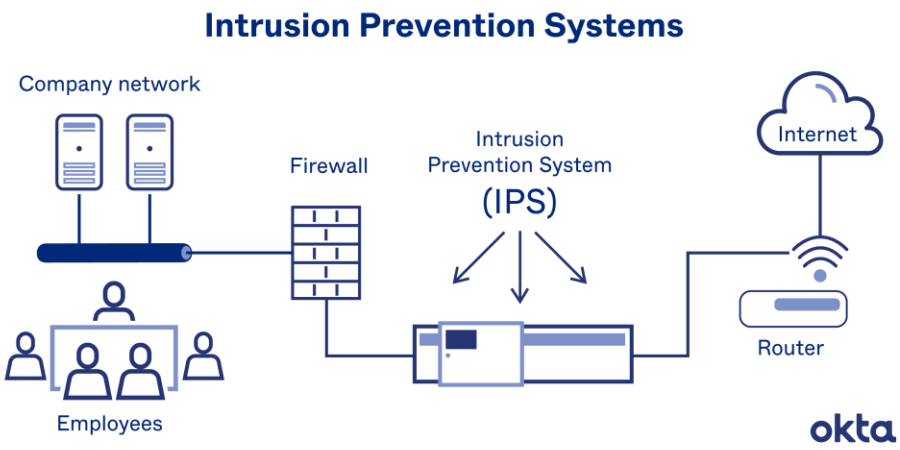
Protocol-based intrusion detection system (PIDS) comprises a system or agent that would consistently resides at the front end of a server, controlling and interpreting the protocol between a user/device and the server. It is trying to secure the web server by regularly monitoring the HTTPS protocol stream and accept the related HTTP protocol.

4. **Application Protocol-based Intrusion Detection System (APIDS):**

Application Protocol-based Intrusion Detection System (APIDS) is a system or agent that generally resides within a group of servers. It identifies the intrusions by monitoring and interpreting the communication on application-specific protocols.

5. **Hybrid Intrusion Detection System :**

Hybrid intrusion detection system is made by the combination of two or more approaches of the intrusion detection system. In the hybrid intrusion detection system, host agent or system data is combined with network information to develop a complete view of the network system.



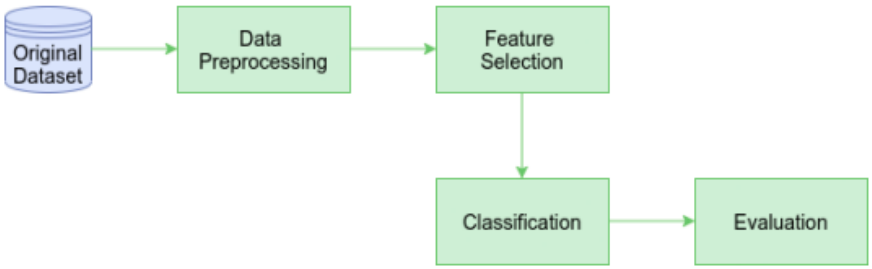
Detection Methods of IDS -

1. **Signature-based Method -**

   Signature-based IDS detects the attacks on the basis of the specific patterns such as number of bytes or number of 1's or number of 0's in the network traffic. The detected patterns in the IDS are known as signatures. Signature-based IDS can easily detect the attacks whose pattern (signature) already exists in the system but it is quite difficult to detect the new malware attacks as their pattern (signature) is not known.

2. **Anomaly-based Method -**

   Anomaly-based IDS was introduced to detect unknown malware attacks as new malware are developed rapidly. In anomaly-based IDS there is use of machine learning to create a trustful activity model and anything coming is compared with that model and it is declared suspicious if it is not found in the model. Machine learning-based methods have a better-generalized property in comparison to signature-based IDS as these models can be trained according to the applications and hardware configurations.



Implementation Details →

- The database contains a standard set of data to be audited, which includes a wide
- variety of intrusions simulated in a military network environment.

```
In [12]: df.head(10)
```

Out[12]:

	0	tcp	ftp_data	SF	491	0.1	0.2	0.3	0.4	0.5	...	25	0.17.1	0.03	0.17.2	0.24	0.25	0.26	0.05	0.27	normal
0	0	udp	other	SF	146	0	0	0	0	0	...	1	0.00	0.60	0.88	0.00	0.00	0.00	0.0	0.00	normal
1	0	tcp	private	S0	0	0	0	0	0	0	...	26	0.10	0.05	0.00	0.00	1.00	1.00	0.0	0.00	neptune
2	0	tcp	http	SF	232	8153	0	0	0	0	...	255	1.00	0.00	0.03	0.04	0.03	0.01	0.0	0.01	normal
3	0	tcp	http	SF	199	420	0	0	0	0	...	255	1.00	0.00	0.00	0.00	0.00	0.00	0.0	0.00	normal
4	0	tcp	private	REJ	0	0	0	0	0	0	...	19	0.07	0.07	0.00	0.00	0.00	0.00	1.0	1.00	neptune
5	0	tcp	private	S0	0	0	0	0	0	0	...	9	0.04	0.05	0.00	0.00	1.00	1.00	0.0	0.00	neptune
6	0	tcp	private	S0	0	0	0	0	0	0	...	15	0.06	0.07	0.00	0.00	1.00	1.00	0.0	0.00	neptune
7	0	tcp	remote_job	S0	0	0	0	0	0	0	...	23	0.09	0.05	0.00	0.00	1.00	1.00	0.0	0.00	neptune
8	0	tcp	private	S0	0	0	0	0	0	0	...	13	0.05	0.06	0.00	0.00	1.00	1.00	0.0	0.00	neptune
9	0	tcp	private	REJ	0	0	0	0	0	0	...	12	0.05	0.07	0.00	0.00	0.00	0.00	1.0	1.00	neptune

10 rows × 42 columns

Step 1 – Data Preprocessing

Step 2 – Modelling

Algorithms Applied: Gaussian Naive Bayes

Approach Used: I have applied various classification algorithms that are mentioned above on

the KDD dataset and compare there results to build a predictive model

Step 1 – Data Preprocessing:

Data preprocessing - Data preprocessing is a step in the data mining and data analysis process that takes raw data and transforms it into a format that can be understood and analyzed by computers and machine learning.

Calcuating shape of dataset

```
In [13]: df.shape
Out[13]: (125972, 42)

In [14]: df.columns
Out[14]: Index(['0', 'tcp', 'ftp_data', 'SF', '491', '0.1', '0.2', '0.3', '0.4', '0.5',
                '0.6', '0.7', '0.8', '0.9', '0.10', '0.11', '0.12', '0.13', '0.14',
                '0.15', '0.16', '0.17', '2', '2.1', '0.18', '0.19', '0.20', '0.21', '1',
                '0.22', '0.23', '150', '25', '0.17.1', '0.03', '0.17.2', '0.24', '0.25',
                '0.26', '0.05', '0.27', 'normal'],
               dtype='object')

In [ ]:
```

Calculating the types of attack present in dataset

Calculate how many different types of attack

In [18]:
```
# for each column in the dataframe
s = set()
ans = 0
for i in df["normal"]:
    if i not in s:
        ans += 1
        s.add(i)
print(ans)
```

```
23
```

Calculating values count for dataset

calculate values count for types of tools

In [21]:
```
df["normal"].value_counts()
```

Out[21]:
```
normal             67342
neptune            41214
satan               3633
ipsweep             3599
portsweep           2931
smurf               2646
nmap                1493
back                 956
teardrop             892
warezclient          890
pod                  201
guess_passwd          53
buffer_overflow       30
warezmaster           20
land                  18
imap                  11
rootkit               10
loadmodule             9
ftp_write              8
multihop               7
phf                    4
perl                   3
spy                    2
Name: normal, dtype: int64
```

Calculating Source type

calculate value count for data(source) type

```
[22]: df["ftp_data"].value_counts()
```

```
t[22]: http          40338
       private       21853
       domain_u       9043
       smtp           7313
       ftp_data       6859
                     ...
       tftp_u            3
       http_8001         2
       aol               2
       harvest           2
       http_2784         1
       Name: ftp_data, Length: 70, dtype: int64
```

Calculating dataframes datatypes

```
In [23]: df.dtypes
```

```
Out[23]: 0             int64
         tcp          object
         ftp_data     object
         SF           object
         491           int64
         0.1           int64
         0.2           int64
         0.3           int64
         0.4           int64
         0.5           int64
         0.6           int64
         0.7           int64
         0.8           int64
         0.9           int64
         0.10          int64
         0.11          int64
         0.12          int64
         0.13          int64
         0.14          int64
         0.15          int64
         0.16          int64
         0.17          int64
         2             int64
         2.1           int64
         0.18        float64
         0.19        float64
         0.20        float64
         0.21        float64
         1           float64
         0.22        float64
         0.23        float64
         150           int64
         25            int64
         0.17.1      float64
         0.03        float64
         0.17.2      float64
         0.24        float64
         0.25        float64
         0.26        float64
         0.05        float64
         0.27        float64
         normal       object
         dtype: object
```

```
In [24]: df.isnull().sum()
```

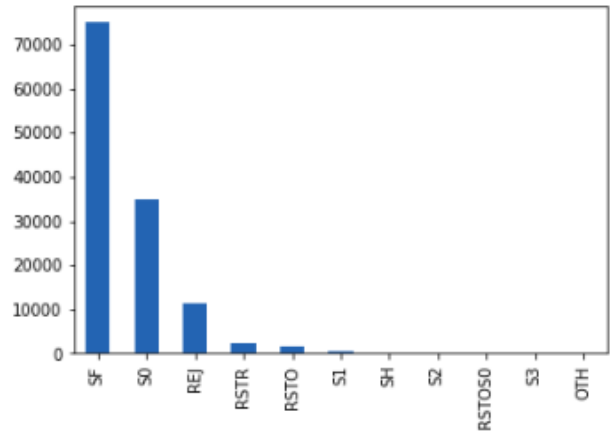
```
Out[24]: 0           0
         tcp         0
         ftp_data    0
         SF          0
         491         0
         0.1         0
         0.2         0
         0.3         0
         0.4         0
         0.5         0
         0.6         0
         0.7         0
         0.8         0
         0.9         0
         0.10        0
         0.11        0
         0.12        0
         0.13        0
         0.14        0
         0.15        0
         0.16        0
         0.17        0
         2           0
         2.1         0
         0.18        0
         0.19        0
         0.20        0
         0.21        0
         1           0
         0.22        0
         0.23        0
         150         0
         25          0
         0.17.1      0
         0.03        0
         0.17.2      0
         0.24        0
         0.25        0
         0.26        0
         0.05        0
         0.27        0
         normal      0
         dtype: int64
```

No missing values present

Visualizing the dataset through graphs

Bargraph visualization

In [38]:
```
bar_graph("SF")
```



Heatmap visualization

In [39]:
```
corr = df.corr()
```

In [40]:
```
plt.figure(figsize=(15,12))
```

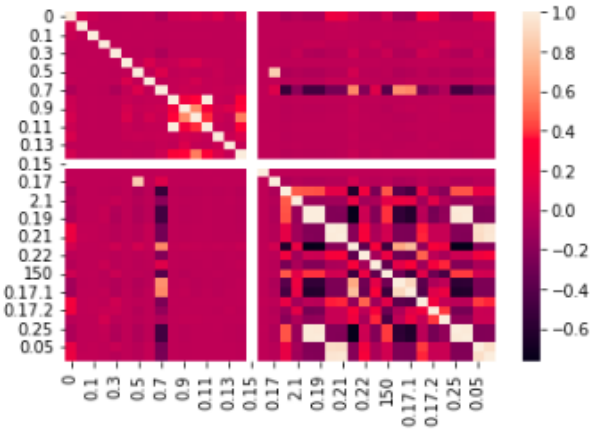
Out[40]:
```
<Figure size 1080x864 with 0 Axes>
```

```
<Figure size 1080x864 with 0 Axes>
```

Attack type or tool vs ftp_data

In [43]:
```
sns.heatmap(corr)
```

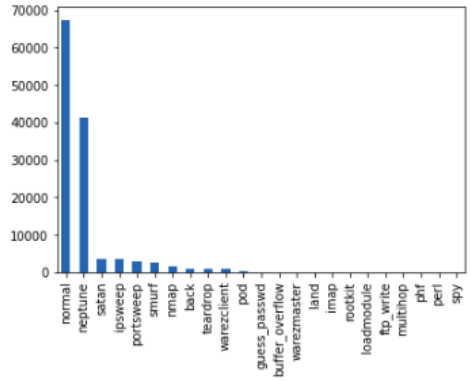
Out[43]:
```
<AxesSubplot:>
```



Visualizing categorical features using graph

In [27]:
```
def bar_graph(feature):
    df[feature].value_counts().plot(kind = "bar")
```

In [28]:
```
bar_graph("normal")
```



we can predict that most of the attacks are of normal type followed by neptune

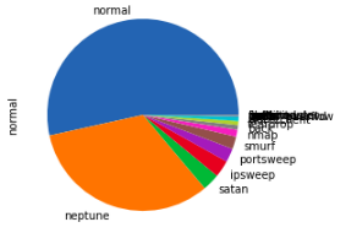
Pie chart

we can predict that most of the attacks are of normal type followed by neptune

In [ ]:
```
```

In [29]:
```
def pie_graph(feature):
    df[feature].value_counts().plot(kind = "pie")
```

In [30]:
```
pie_graph("normal")
```



Machine learning model

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score
```

```
df = df.drop(['target',], axis=1)
print(df.shape)

# Target variable and train set
Y = df[['Attack Type']]
X = df.drop(['Attack Type',], axis=1)

sc = MinMaxScaler()
X = sc.fit_transform(X)

# Split test and train data
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.33, random_state=42)
print(X_train.shape, X_test.shape)
print(Y_train.shape, Y_test.shape)
```

```
(494021, 31)
(330994, 30) (163027, 30)
(330994, 1) (163027, 1)
```

Code

In [ ]:
```
import @all modules
```

In [ ]:
```
df = pd.read_csv("Train.csv")
```

In [ ]:
```
df.head(10)
```

In [ ]:
```
df.shape
```

In [ ]:
```
df.columns
```

Calculate how many different types of attack

In [ ]:
```
# for each column in the dataframe
s = set()
ans = 0
for i in df["normal"]:
    if i not in s:
        ans += 1
        s.add(i)
print(ans)
```

calculate values count for types of tools

In [ ]:
```
df["normal"].value_counts()
```

calculate value count for data(source) type

In [ ]:
```
df["ftp_data"].value_counts()
```

In [ ]:
```
df.dtypes
```

In [ ]:
```
df.isnull().sum()
```

No missing values present

Visualizing categorical features using graph

In [ ]:
```
def bar_graph(feature):
    df[feature].value_counts().plot(kind = "bar")
```

In [ ]:
```
bar_graph("normal")
```

we can predict that most of the attacks are of normal type followed by neptune

In [ ]:
```
def pie_graph(feature):
    df[feature].value_counts().plot(kind = "pie")
```

In [ ]:
```
pie_graph("normal")
```

In [ ]:
```
bar_graph("SF")
```

In [ ]:
```
corr = df.corr()
```

In [ ]:
```
plt.figure(figsize=(15,12))
```

Attack type or tool vs ftp_data

In [ ]:
```
sns.heatmap(corr)
```

*Random Forest *

In [51]:
```
from sklearn.ensemble import RandomForestClassifier
model3 = RandomForestClassifier(n_estimators=30)
start_time = time.time()
model3.fit(X_train, Y_train.values.ravel())
end_time = time.time()
print("Training time: ",end_time-start_time)
```

```
Training time:  7.327942132949829
```

In [61]:
```
start_time = time.time()
Y_test_pred3 = model3.predict(X_test)
end_time = time.time()

Y_train_pred3 = model2.predict(X_train)

print("Testing time: ",end_time-start_time)
```

```
Testing time:  0.4444396495819092
```

In [66]:
```
print("Train score is:", model3.score(X_train, Y_train))
print("Test score is:",model3.score(X_test,Y_test))

print(accuracy_score(Y_test,Y_test_pred3))
```

```
Train score is: 0.9999728091747887
Test score is: 0.999662632570065
0.999662632570065
```

In [68]:
```
names = ['DT','RF']
values = [99.05230421954646,99.9662632570065]
f = plt.figure(figsize=(15,3),num=10)
plt.subplot(131)
plt.ylim(80,102)
plt.bar(names,values)
```

Out[68]:
```
<BarContainer object of 2 artists>
```