MACHINE LEARNING TECHNIQUE FOR DETECTING PHISHING WEBSITE

A PROJECT

REPORT IN

NETWORKSECURITY (CSE2008)

PROJECT MEMBERS: -

J.K VISHWAJEET -18BCE0831
NABEEL KHAN-18BCE0847
SANIDHYA SEHGAL -18BCE0877

Acknowledgments

We take immense pleasure in thanking our ma’am, Prof. Umadevi K S for having permitted us to carry out the project.

We express gratitude to my guide, for guidance and suggestions that helped us to complete the project on time. Words are inadequate to express our gratitude towards her who encouraged and supported us during the project.

Abstract

Phishing is a cybercrime in which a target or targets are contacted by email, telephone or text message by someone posing as a legitimate institution to lure individuals into providing sensitive data such as personally identifiable information, banking and credit card details, and passwords. Phishing is a type of social engineering attack often used to steal user data, including login credentials and credit card numbers. It occurs when an attacker, masquerading as a trusted entity, dupes a victim into opening an email, instant message, or text message. The recipient is then tricked into clicking a malicious link, which can lead to the installation of malware, the freezing of the system as part of a ransomware attack or the revealing of sensitive information. The user is redirected to a bogus page which looks legitimate and is compelled to share its sensitive data without realizing that the page is a malicious one designed to steal information.

Malware is any software intentionally designed to cause damage to a computer, server, client, or computer network (by contrast, software that causes unintentional harm due to some deficiency is typically described as a software bug). A wide variety of types of malware exist, including computer viruses, worms, Trojan horses, ransomware, spyware, adware, rogue software, and scareware.
We will create a machine learning project where the user will enter the URL of the website and the application will be able to tell whether the website is legitimate or malicious. If malicious, we will try to detect if it contains any malware in it so that we can warn the user before he/she downloads the files from it that can cause a damage to the system.

We will use Python and Scikit learning framework for this project.

Introduction

Phishing is a serious cybercrime and most common of all. According to Avanan’s phishing statistics, 1 in every 99 emails is a phishing attack. And this amounts to 4.8 emails per employee in a five-day work week. Considering close to a third or 30% phishing emails make it past default security, the threat is very much present. The success rate of these attacks has emboldened scammers to launch more of them. Avanan reports an increase of 65% in phishing attacks from 2016 to 2017. And this is a global phenomenon affecting every region and economy. In 2018 83% of people received phishing attacks worldwide resulting in a range of disruptions and damages.

This includes decreased productivity (67%), loss of proprietary data (54%), and damage to reputation (50%). The biggest damage comes from spear phishing at $7.2 million, malware at $2.4 million, extortion at $5,000, and credential harvesting at $400 per account. The above statistics clearly shows that phishing is a serious issue.

With evolving phishing techniques, machine learning is the weapon which has the ability to reduce this attack to a great extent. Using machine learning, we will train our model to the data set containing the features of the phishing websites. We will then save this model in a pickle file. We will then ask input in the form of URL and parse through the different features of that website. We will then test this data point and display whether the URL is legitimate or malicious.

Features:

- IP Address
- URL length
- @ symbol in URL
- ‘//’ in URL
- ‘-‘ in URL
- Number of domains and subdomains
- Use of HTTPS
- Domain Registration Length and Request URL
- HTML tags and anchor tag
- SSL final state

Other features are there as well.

Malware is any piece of software which is intended to cause harm to your system or network. Malware is different from normal programs in a way that they most of them can spread itself in the network, remain undetectable, cause changes/damage to the infected system or network, persistence. They can bring down the machine’s performance to knees and can cause a destruction of the network. Consider the case when the computer becomes infected and is no longer usable, the data inside becomes unavailable – these are some of the malware damage scenarios. Malware attacks can be traced back to the time, even before the internet became widespread.

The best-known types of malware, viruses, and worms are known because they spread, rather than any specific types of behavior. A computer virus is software that embeds itself in some other executable software (including the operating system itself) on the target system without the user's knowledge and consent and when it is run, the virus is spread to other executables. On the other hand, a worm is a stand-alone malware software that actively transmits itself over a network to infect other computers. These definitions lead to the observation that a virus requires the user to run an infected software or operating system for the virus to spread, whereas a worm spreads itself.

Characteristics of phishing websites

An average phishing site will have the accompanying Attributes:

- It utilizes real looking substance, for example, pictures,
- Writings, logos or even mirrors the real site to allure guests to enter their records or money related data.
- It might contain real connects to web substance of the genuine site, for example, get in touch with us, security or disclaimer to trap the guests.
- It might utilize a comparative area name or sub-space name as that of the genuine site.
- It might utilize structures to gather guests' data where these structures are like that in the real site.
- It might in type of spring up window that is opened in the closer view with the certified page out of sight to deceive and confound the guest believing that he/she is as yet visiting the real site.
- It might show the IP address or the phony location on the guests' location bar accepting that guests may not mindful of that. Some fraudsters may perform URL parodying by utilizing contents or HTML directions to develop counterfeit location bar instead of the first location.

Characteristics of malware files

An average malware detecting software will have the accompanying Attributes:

- Increased processor load,
- Reduced performance
- Greater network latency
- Slow loading of web pages
- Lots of ad popups
- Increased temp file storage on hard drive.
- Odd network traffic
- Nom-typical ‘user’ activity
- Your personal details appearing for sale on the dark web as your bank account is hacked

Method and Algorithm Used

Random forest algorithm is a supervised classification algorithm. As the name suggest, this algorithm creates the forest with several trees.
In general, the more trees in the more robust the forest looks like. In the same forest way in the random forest higher the number of trees in the forest gives the classifier, the high accuracy results.

Basic decision tree concept

Decision tree concept is more to the rule-based system. Given the training dataset with targets and features, the decision tree algorithm will come up with some set of rules.

The same set rules can be used to perform the prediction on the test dataset.
Suppose you would like to predict that your daughter will like the newly released animation movie or not. To model the decision tree, you will use the training dataset like the animated cartoon characters your daughter liked in the past movies.

So, once you pass the dataset with the target as your daughter will like the movie or not to the decision tree classifier. The decision tree will start building the rules with the characters your daughter like as nodes and the targets like or not as the leaf nodes. By considering the path from the root node to the leaf node. You can get the rules.

The simple rule could be if some x character is playing the leading role then your daughter will like the movie. You can think few more rules based on this example.

Then to predict whether your daughter will like the movie or not. You just need to check the rules which are created by the decision tree to predict whether your daughter will like the newly released movie or not.
In decision tree algorithm calculating these nodes and forming the rules will happen using the information gain and gini index calculations.
In random forest algorithm, instead of using information gain or gini index for calculating the root node, the process of finding the root node and splitting the feature nodes will happen randomly.

Why Random forest algorithm

To address why random forest algorithm, we have listed the advantages below:

- The same random forest algorithm or the random forest classifier can use for both classification and the regression task.
- Random forest classifier will handle the missing values.

- When we have more trees in the forest, random forest classifier will not overfit the model.
- Can model the random forest classifier for categorical values also.

Code

Random_forest.py – This script creates our machine learning model and using Random Forest algorithm, we train our data. We get a 97% accuracy on the test data using this algorithm.

```
#importing libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.externals import joblib


#importing the dataset
dataset = pd.read_csv("phishcoop.csv")
dataset = dataset.drop('id', 1) #removing unwanted column
x = dataset.iloc[ : , :-1].values
y = dataset.iloc[:, -1:].values


#spliting the dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.25, random_state = 42 )


#applying grid search to find best performing parameters
from sklearn.model_selection import GridSearchCV
parameters = [{'n_estimators': [100,700], 'max_features': ['sqrt', 'log2']}]


# 'criterion' :['gini', 'entropy']}]
grid_search = GridSearchCV(RandomForestClassifier(), parameters,cv =5,n_jobs= -1)
grid_search.fit(x_train, y_train.ravel())

#printing best parameters
print("Best Accurancy =" +str( grid_search.best_score_))
print("best parameters =" + str(grid_search.best_params_))

#fitting RandomForest regression with best params
```

```
classifier = RandomForestClassifier(n_estimators = 100, max_features = 'log2',random_state = 0)
classifier.fit(x_train, y_train.ravel())

#predicting the tests set result
from sklearn.metrics import mean_squared_error
y_pred = classifier.predict(x_test)

#confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

#Creating the RandomForest Model
from sklearn.ensemble import RandomForestRegressor
forest_regressor = RandomForestRegressor()
forest_reg = RandomForestRegressor()
forest_reg.fit(x_train,y_train)
prediction = forest_reg.predict(x_test)
rmse = mean_squared_error(y_pred,y_test.ravel())



#pickle file joblib

joblib.dump(classifier, 'rf_final.pkl')
#Features Importance random forest
names = dataset.iloc[:,:-1].columns
importances = classifier.feature_importances_
sorted_importances = sorted(importances, reverse=True)
indices = np.argsort(-importances)
var_imp = pd.DataFrame(sorted_importances,names[indices], columns=['importance'])


#plotting variable importance
plt.title("Variable Importances")
plt.barh(np.arange(len(names)), sorted_importances, height = 0.7)
plt.yticks(np.arange(len(names)), names[indices], fontsize=7)
plt.xlabel('Relative Importance')
plt.show()
```

Inputscript.py – This script is used to extract all the features from a url and feed it into the machine learning model inside a pickle file.

```
import re
from tldextract import extract
import ssl
import socket
from bs4 import BeautifulSoup
import urllib.request
import whois
import datetime


#Try to get response from text
```

```
try:
    response = requests.get(url)
except:
    response = ""

#check wether URL has IP
def url_having_ip(url):
#using regular function
    symbol = re.findall(r'(http((s)?)://)((((\d)+).)*)((\w)+)(/((\w)+))?',url)
    if(len(symbol)!=0):
        having_ip = 1 #phishing
    else:
        having_ip = -1 #legitimate
    return 0

#Find the length of URL
def url_length(url):
    length=len(url)
    if(length<54):
        return -1
    elif(54<=length<=75):
        return 0
    else:
        return 1

#check if URL is short
def url_short(url):
    if re.findall("goo.gl|bit.ly", url):
        return 1
    else:
        return -1

#Check for @ Symbol
def having_at_symbol(url):
    symbol=re.findall(r'@',url)
    if(len(symbol)==0):
        return -1
    else:
        return 1

#Check for //
def doubleSlash(url):
    if(url.count('//') > 1):
        return 1
    else:
        return -1

#Check for Prefix and Suffix
def prefix_suffix(url):
    subDomain, domain, suffix = extract(url)
    if(domain.count('-')):
        return 1
    else:
        return -1

#Check for Sub domains
```

```
def sub_domain(url):
    subDomain, domain, suffix = extract(url)
    if(subDomain.count('.')==0):
        return -1
    elif(subDomain.count('.')==1):
        return 0
    else:
        return 1

#Find SSL final state
def SSLfinal_State(url):
    try:
#check wheather contains https
        if(re.search('^https',url)):
            usehttps = 1
        else:
            usehttps = 0
#getting the certificate issuer to later compare with trusted issuer
#getting host name
        subDomain, domain, suffix = extract(url)
        host_name = domain + "." + suffix
        context = ssl.create_default_context()
        sct = context.wrap_socket(socket.socket(), server_hostname = host_name)
        sct.connect((host_name, 443))
        certificate = sct.getpeercert()
        issuer = dict(x[0] for x in certificate['issuer'])
        certificate_Auth = str(issuer['commonName'])
        certificate_Auth = certificate_Auth.split()
        if(certificate_Auth[0] == "Network" or certificate_Auth == "Deutsche"):
            certificate_Auth = certificate_Auth[0] + " " + certificate_Auth[1]
        else:
            certificate_Auth = certificate_Auth[0]
        trusted_Auth                                                                                   =
['Comodo','Symantec','GoDaddy','GlobalSign','DigiCert','StartCom','Entrust','Verizon','Trustwave','Unizeto','Buypass
','QuoVadis','Deutsche                                                                          Telekom','Network
Solutions','SwissSign','IdenTrust','Secom','TWCA','GeoTrust','Thawte','Doster','VeriSign']
#getting age of certificate
        startingDate = str(certificate['notBefore'])
        endingDate = str(certificate['notAfter'])
        startingYear = int(startingDate.split()[3])
        endingYear = int(endingDate.split()[3])
        Age_of_certificate = endingYear-startingYear

#checking final conditions
        if((usehttps==1) and (certificate_Auth in trusted_Auth) and (Age_of_certificate>=1) ):
            return -1 #legitimate
        elif((usehttps==1) and (certificate_Auth not in trusted_Auth)):
            return 0 #suspicious
        else:
            return 1 #phishing

    except Exception as e:

        return 1

#Check if domain is registered
```

```
def domain_registration(url):
    try:
        w = whois.whois(url)
        updated = w.updated_date
        exp = w.expiration_date
        length = (exp[0]-updated[0]).days
        if(length<=365):
            return 1
        else:
            return -1
    except:
        return 0

#Check for Fevicon
def favicon(url):
    return 0#Cannot be determined only with URL

#Check for Port
def port(url):
    return 0#Cannot be determined only with URL

#Check if http token exists
def https_token(url):
    subDomain, domain, suffix = extract(url)
    host =subDomain +'.' + domain + '.' + suffix
    if(host.count('https')):
        return 1
    else:
        return -1

#Request for URL
def request_url(url):
    try:
        subDomain, domain, suffix = extract(url)
        websiteDomain = domain

        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        imgs = soup.findAll('img', src=True)
        total = len(imgs)

        linked_to_same = 0
        avg =0
        for image in imgs:
            subDomain, domain, suffix = extract(image['src'])
            imageDomain = domain
            if(websiteDomain==imageDomain or imageDomain==''):
                linked_to_same = linked_to_same + 1
        vids = soup.findAll('video', src=True)
        total = total + len(vids)

        for video in vids:
            subDomain, domain, suffix = extract(video['src'])
            vidDomain = domain
            if(websiteDomain==vidDomain or vidDomain==''):
                linked_to_same = linked_to_same + 1
        linked_outside = total-linked_to_same
        if(total!=0):
            avg = linked_outside/total

        if(avg<0.22):
            return -1
        elif(0.22<=avg<=0.61):
            return 0
        else:
            return 1
    except:
        return 0

#Check for anchor element
def url_of_anchor(url):
    try:
        subDomain, domain, suffix = extract(url)
        websiteDomain = domain

        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        anchors = soup.findAll('a', href=True)
        total = len(anchors)
        linked_to_same = 0
        avg = 0
        for anchor in anchors:
            subDomain, domain, suffix = extract(anchor['href'])
            anchorDomain = domain
            if(websiteDomain==anchorDomain or anchorDomain==''):
                linked_to_same = linked_to_same + 1
        linked_outside = total-linked_to_same
        if(total!=0):
            avg = linked_outside/total

        if(avg<0.31):
            return -1
        elif(0.31<=avg<=0.67):
            return 0
        else:
            return 1
    except:
        return 0

#Check for links in tags
def Links_in_tags(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')

        no_of_meta =0
        no_of_link =0
        no_of_script =0
        anchors=0
        avg =0
        for meta in soup.find_all('meta'):
            no_of_meta = no_of_meta+1
        for link in soup.find_all('link'):
            no_of_link = no_of_link +1
        for script in soup.find_all('script'):
            no_of_script = no_of_script+1
        for anchor in soup.find_all('a'):
            anchors = anchors+1
        total = no_of_meta + no_of_link + no_of_script+anchors
        tags = no_of_meta + no_of_link + no_of_script
        if(total!=0):
            avg = tags/total

        if(avg<0.25):
            return -1
        elif(0.25<=avg<=0.81):
            return 0
        else:
            return 1
    except:
        return 0


def sfh(url):
    return 0

def email_submit(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        if(soup.find('mailto:')):
            return 1
        else:
            return -1
    except:
        return 0

def abnormal_url(url):
    return 0

def redirect(url):
    return 0

def on_mouseover(url):
    if re.findall("<script>.+onmouseover.+</script>", response):
        return 1
    else:
        return -1
def rightClick(url):
    if re.findall(r"event.button ?== ?2", response):
        return 1
    else:
        return -1
def popup(url):
    if re.findall(r"alert\(", response):
        return 1
    else:
        return -1
```

```
def iframe(url):
    if re.findall(r"[<iframe>|<frameBorder>]", response):
        return 1
    else:
        return -1


def age_of_domain(url):
    try:
        w = whois.query(url)
        start_date = w.creation_date
        current_date = datetime.datetime.now()
        age =(current_date-start_date[0]).days
        if(age>=180):
            return -1
        else:
            return 1
    except Exception as e:
        return 0

#Not possible to find DNS of URL
def dns(url):
    #ongoing
    return 0

#Not possible to find web traffic
def web_traffic(url):
    #ongoing
    return 0

#Not possible to find page rank as Google has blocked API's
def page_rank(url):
    return 0

#Not possible to google index find as Google has blocked API's
def google_index(url):
    #ongoing
    return 0

#Not possible to find with URL only
def links_pointing(url):
    return 0

#Not possible to find with URL only
def statistical(url):
    return 0

def main(url):
    check = [[url_having_ip(url),url_length(url),url_short(url),having_at_symbol(url),
             doubleSlash(url),prefix_suffix(url),sub_domain(url),SSLfinal_State(url),
              domain_registration(url),favicon(url),port(url),https_token(url),request_url(url),
              url_of_anchor(url),Links_in_tags(url),sfh(url),email_submit(url),abnormal_url(url),
              redirect(url),on_mouseover(url),rightClick(url),popup(url),iframe(url),
              age_of_domain(url),dns(url),web_traffic(url),page_rank(url),google_index(url),
              links_pointing(url),statistical(url)]]
    return check
```

Index.py – This script is used to run all the other scripts. It calls the other scripts and tells whether the site is phishing or not.

```
#importing libraries
from sklearn.externals import joblib
import Inputscript

#load the pickle file
classifier = joblib.load('rf_final.pkl')

#input url
print("\n\n\n\nWelcome to the phishing website detection software !\n")

url = input("Enter the Website URL: ")

#checking and predicting
checkprediction = Inputscript.main(url)
prediction = classifier.predict(checkprediction)

#Printing the result
if(prediction == 1):
   print("The site is a phishing site")
else:
   print("The site is not a phishing site")
```

MalwareDetecor.py – This This script creates our machine learning model and using Random Forest algorithm, we train our data. We get a 99% accuracy on the test data using this algorithm

```
#Importing relevant libraries
import pandas as pd
import numpy as np

#Importing and preprocessing the data
malData = pd.read_csv("MalwareData.csv",sep = '|')
legit = malData[0:41323].drop(['legitimate'],axis = 1)
mal = malData[41323::].drop(['legitimate'],axis = 1)


#Importing the libraries
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.feature_selection import SelectFromModel
from sklearn.model_selection import train_test_split

#Extra Tree classifier to eliminate unwanted feature
data_in = malData.drop(['Name','md5','legitimate'], axis = 1).values
labels = malData['legitimate'].values
extratrees = ExtraTreesClassifier().fit(data_in,labels)
select = SelectFromModel(extratrees,prefit = True)
```

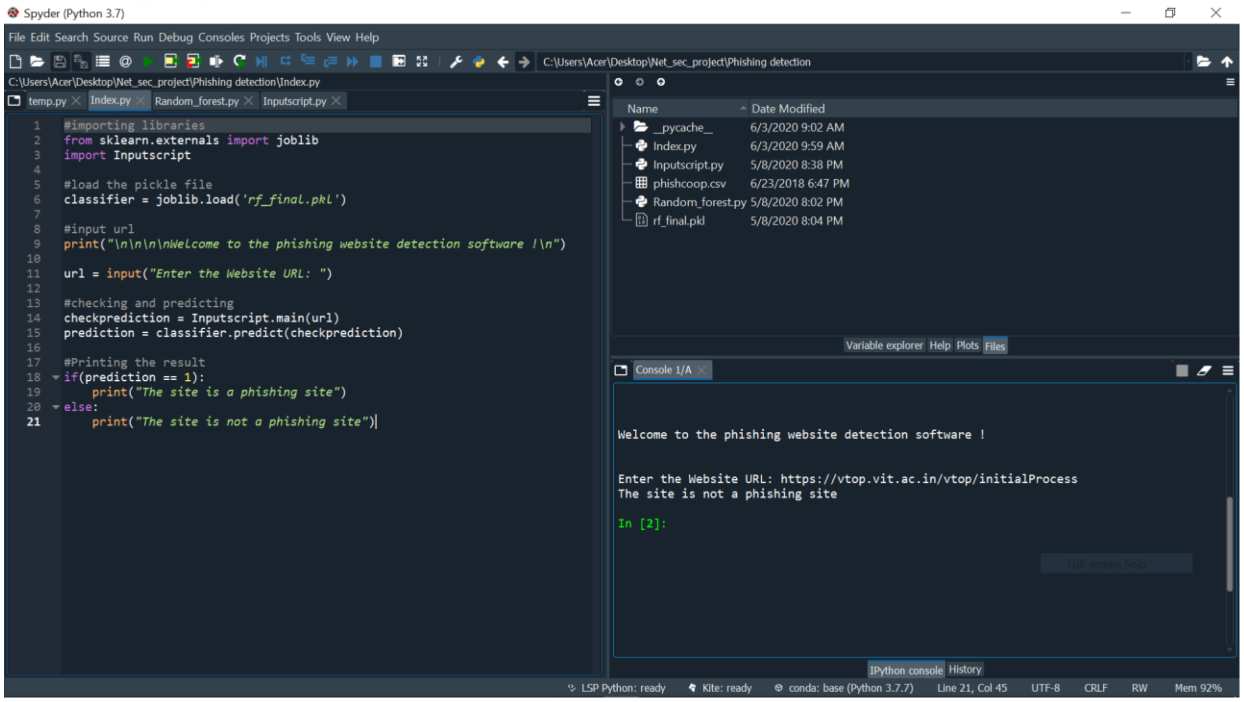
```
data_in_new = select.transform(data_in)

#Important feature columns
features = data_in_new.shape[1]
importances = extratrees.feature_importances_
indices = np.argsort(importances)[::-1]
feature_importance = []
for i in range(features):
   feature_importance.append([malData.columns[2+indices[i]],importances[indices[i]]])

#Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier
legit_train, legit_test, mal_train, mal_test = train_test_split(data_in_new,labels,test_size = 0.2)
classifier = RandomForestClassifier(n_estimators = 50)
classifier.fit(legit_train,mal_train)

#Result of test dataset
from sklearn.metrics import confusion_matrix
result = classifier.predict(legit_test)
conf_mat = confusion_matrix(mal_test,result)
```

#Note:

The main difference between random forests and extra trees also known as extreme random forests is that, instead of computing the locally optimal feature/split combination (in random forest), for each feature under consideration, a random value is selected for the split (in extra trees).

Output

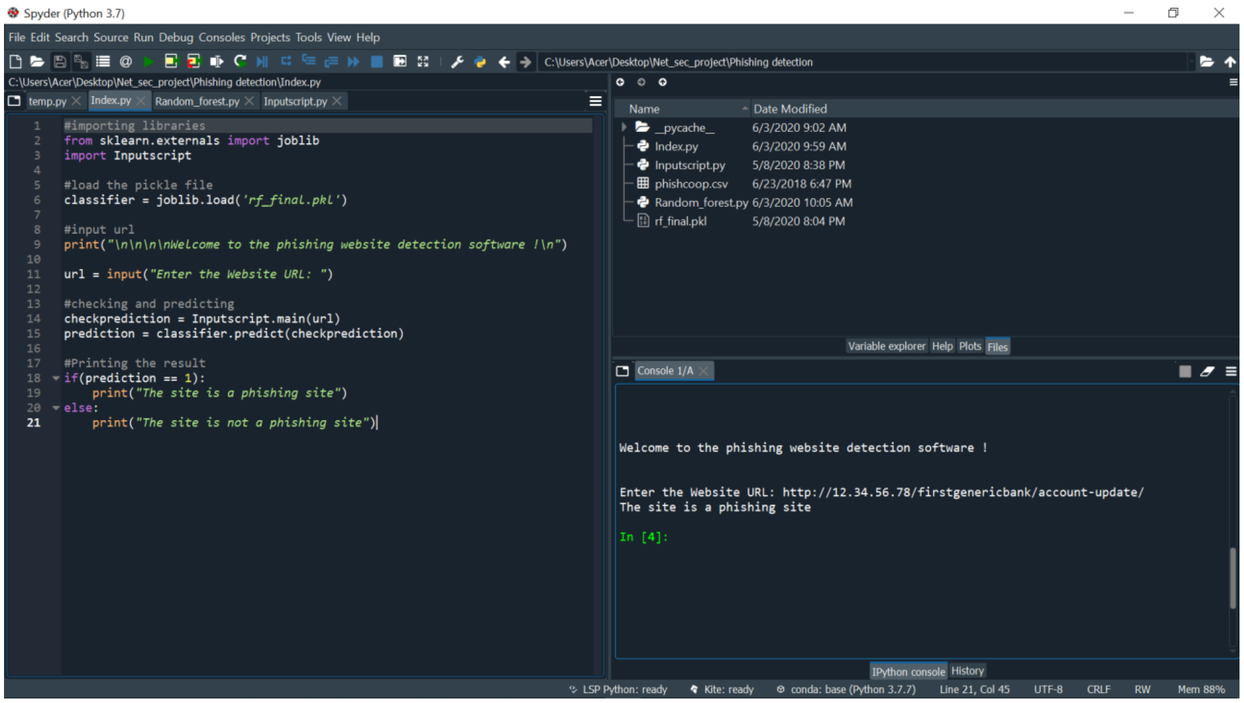
Phishing Software:


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Acer\Desktop\Net_sec_project\Phishing detection
C:\Users\Acer\Desktop\Net_sec_project\Phishing detection\Index.py
temp.py
Index.py
Random_forest.py
Inputscript.py
Variable explorer
Help
Plots
Files
Console 1/A
IPython console
History
LSP Python: ready
Kite: ready
conda: base (Python 3.7.7)
Line 21, Col 45
UTF-8
CRLF
RW
Mem 92%

```python
#importing libraries
from sklearn.externals import joblib
import Inputscript

#load the pickle file
classifier = joblib.load('rf_final.pkl')

#input url
print("\n\n\n\nWelcome to the phishing website detection software !\n")

url = input("Enter the Website URL: ")

#checking and predicting
checkprediction = Inputscript.main(url)
prediction = classifier.predict(checkprediction)

#Printing the result
if(prediction == 1):
    print("The site is a phishing site")
else:
    print("The site is not a phishing site")
```

Name	Date Modified
__pycache__	6/3/2020 9:02 AM
Index.py	6/3/2020 9:59 AM
Inputscript.py	5/8/2020 8:38 PM
phishcoop.csv	6/23/2018 6:47 PM
Random_forest.py	5/8/2020 8:02 PM
rf_final.pkl	5/8/2020 8:04 PM

```
Welcome to the phishing website detection software !


Enter the Website URL: https://vtop.vit.ac.in/vtop/initialProcess
The site is not a phishing site

In [2]:
```

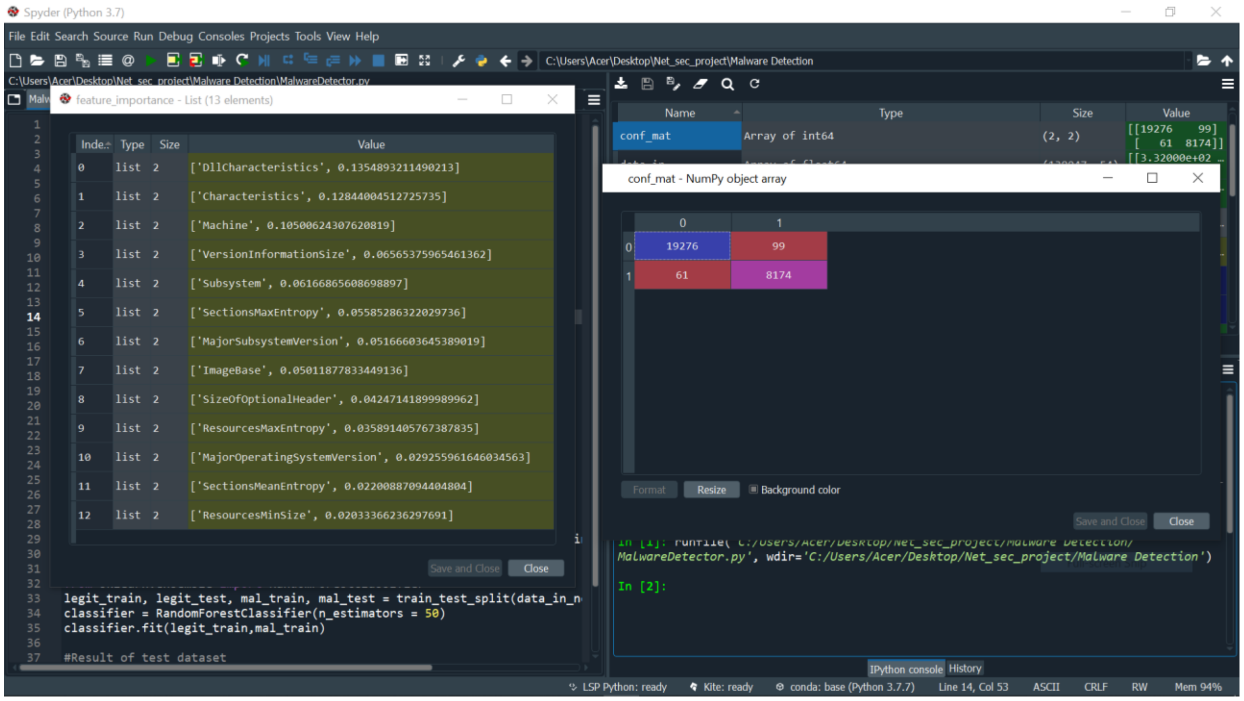

Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Acer\Desktop\Net_sec_project\Phishing detection
C:\Users\Acer\Desktop\Net_sec_project\Phishing detection\Index.py
temp.py
Index.py
Random_forest.py
Inputscript.py
Variable explorer
Help
Plots
Files
Console 1/A
IPython console
History
LSP Python: ready
Kite: ready
conda: base (Python 3.7.7)
Line 21, Col 45
UTF-8
CRLF
RW
Mem 88%

```python
#importing libraries
from sklearn.externals import joblib
import Inputscript

#load the pickle file
classifier = joblib.load('rf_final.pkl')

#input url
print("\n\n\n\nWelcome to the phishing website detection software !\n")

url = input("Enter the Website URL: ")

#checking and predicting
checkprediction = Inputscript.main(url)
prediction = classifier.predict(checkprediction)

#Printing the result
if(prediction == 1):
    print("The site is a phishing site")
else:
    print("The site is not a phishing site")
```

Name	Date Modified
__pycache__	6/3/2020 9:02 AM
Index.py	6/3/2020 9:59 AM
Inputscript.py	5/8/2020 8:38 PM
phishcoop.csv	6/23/2018 6:47 PM
Random_forest.py	6/3/2020 10:05 AM
rf_final.pkl	5/8/2020 8:04 PM

```
Welcome to the phishing website detection software !


Enter the Website URL: http://12.34.56.78/firstgenericbank/account-update/
The site is a phishing site

In [4]:
```

Malware detection Software:



feature_importance - List (13 elements)

Index	Type	Size	Value
0	list	2	['DllCharacteristics', 0.1354893211490213]
1	list	2	['Characteristics', 0.12844004512725735]
2	list	2	['Machine', 0.10500624307620819]
3	list	2	['VersionInformationSize', 0.06565375965461362]
4	list	2	['Subsystem', 0.06166865608698897]
5	list	2	['SectionsMaxEntropy', 0.05585286322029736]
6	list	2	['MajorSubsystemVersion', 0.05166603645389019]
7	list	2	['ImageBase', 0.05011877833449136]
8	list	2	['SizeOfOptionalHeader', 0.04247141899989962]
9	list	2	['ResourcesMaxEntropy', 0.035891405767387835]
10	list	2	['MajorOperatingSystemVersion', 0.029255961646034563]
11	list	2	['SectionsMeanEntropy', 0.02200887094404804]
12	list	2	['ResourcesMinSize', 0.02033366236297691]

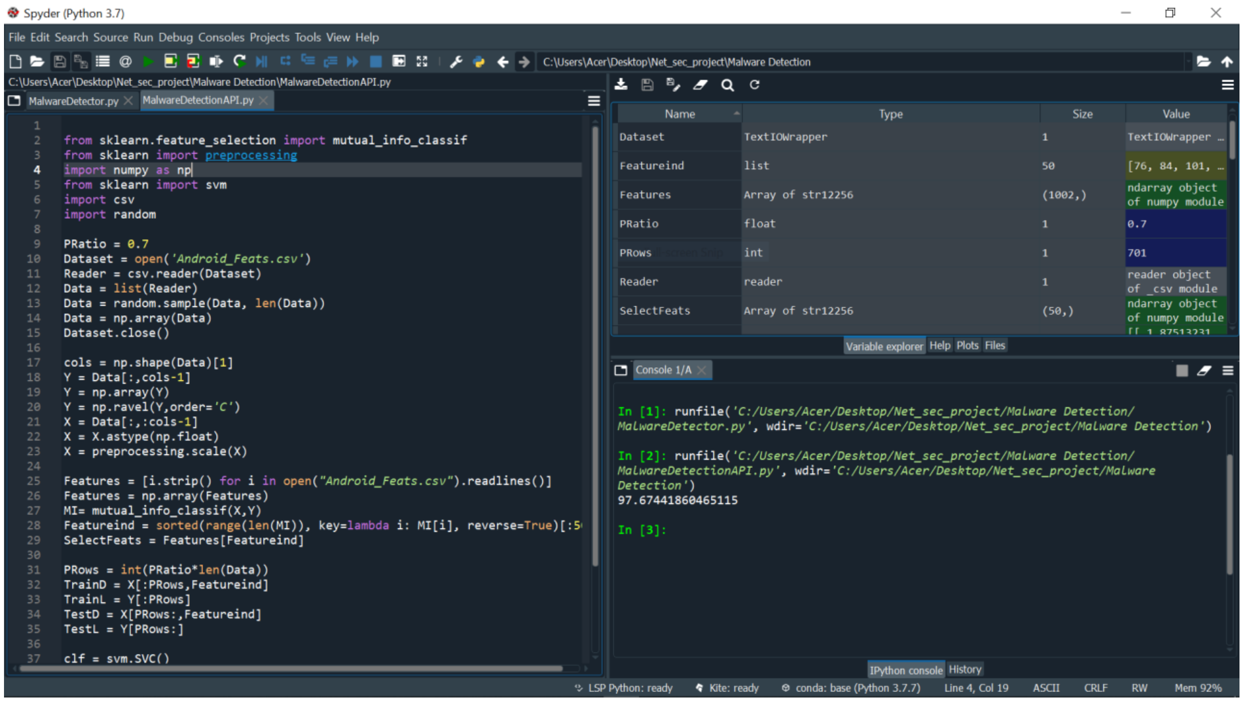
```python
legit_train, legit_test, mal_train, mal_test = train_test_split(data_in_n
classifier = RandomForestClassifier(n_estimators = 50)
classifier.fit(legit_train,mal_train)

#Result of test dataset
```

Name	Type	Size	Value
conf_mat	Array of int64	(2, 2)	[[19276 99] [61 8174]]

conf_mat - NumPy object array

	0	1
0	19276	99
1	61	8174

```
In [1]: runfile('C:/Users/Acer/Desktop/Net_sec_project/Malware Detection/
MalwareDetector.py', wdir='C:/Users/Acer/Desktop/Net_sec_project/Malware Detection')

In [2]:
```



C:\Users\Acer\Desktop\Net_sec_project\Malware Detection\MalwareDetectionAPI.py

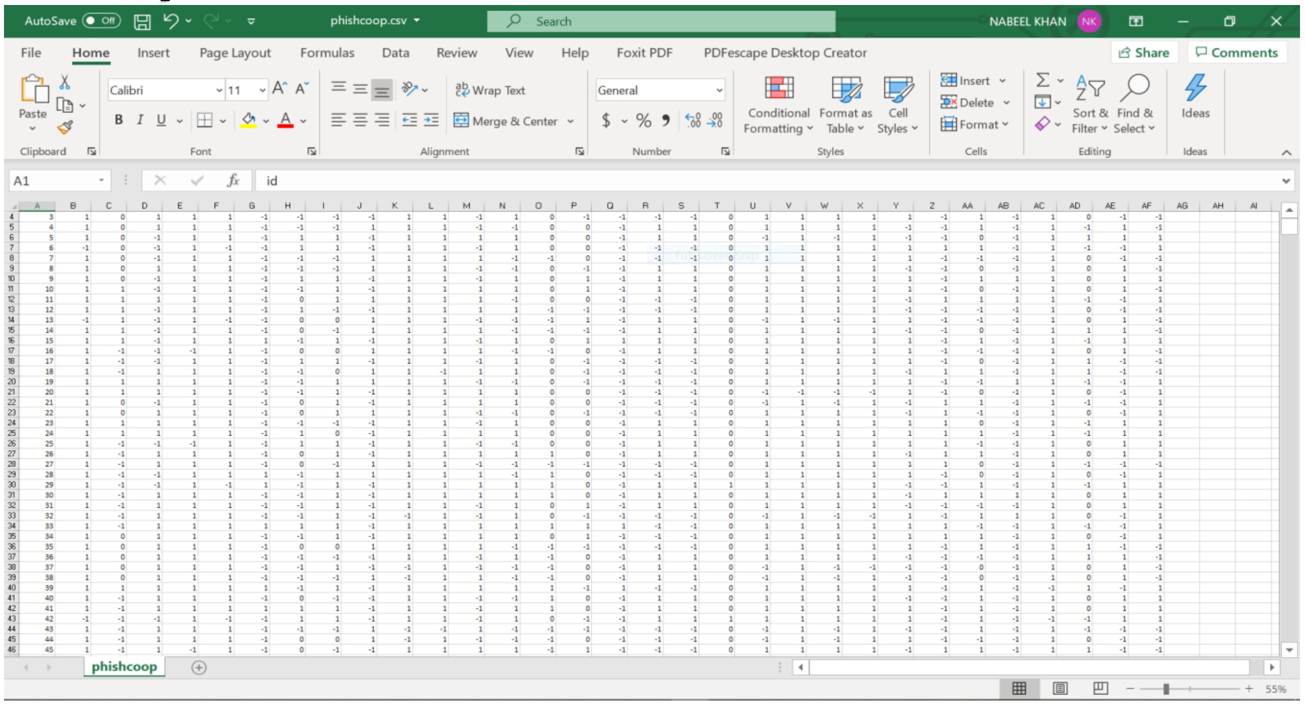
```python

from sklearn.feature_selection import mutual_info_classif
from sklearn import preprocessing
import numpy as np
from sklearn import svm
import csv
import random

PRatio = 0.7
Dataset = open('Android_Feats.csv')
Reader = csv.reader(Dataset)
Data = list(Reader)
Data = random.sample(Data, len(Data))
Data = np.array(Data)
Dataset.close()

cols = np.shape(Data)[1]
Y = Data[:,cols-1]
Y = np.array(Y)
Y = np.ravel(Y,order='C')
X = Data[:,:cols-1]
X = X.astype(np.float)
X = preprocessing.scale(X)

Features = [i.strip() for i in open("Android_Feats.csv").readlines()]
Features = np.array(Features)
MI= mutual_info_classif(X,Y)
Featureind = sorted(range(len(MI)), key=lambda i: MI[i], reverse=True)[:5
SelectFeats = Features[Featureind]

PRows = int(PRatio*len(Data))
TrainD = X[:PRows,Featureind]
TrainL = Y[:PRows]
TestD = X[PRows:,Featureind]
TestL = Y[PRows:]

clf = svm.SVC()
```

Name	Type	Size	Value
Dataset	TextIOWrapper	1	TextIOWrapper …
Featureind	list	50	[76, 84, 101, …
Features	Array of str12256	(1002,)	ndarray object of numpy module
PRatio	float	1	0.7
PRows	int	1	701
Reader	reader	1	reader object of _csv module
SelectFeats	Array of str12256	(50,)	ndarray object of numpy module

```
In [1]: runfile('C:/Users/Acer/Desktop/Net_sec_project/Malware Detection/
MalwareDetector.py', wdir='C:/Users/Acer/Desktop/Net_sec_project/Malware Detection')

In [2]: runfile('C:/Users/Acer/Desktop/Net_sec_project/Malware Detection/
MalwareDetectionAPI.py', wdir='C:/Users/Acer/Desktop/Net_sec_project/Malware
Detection')
97.67441860465115

In [3]:
```

Dataset

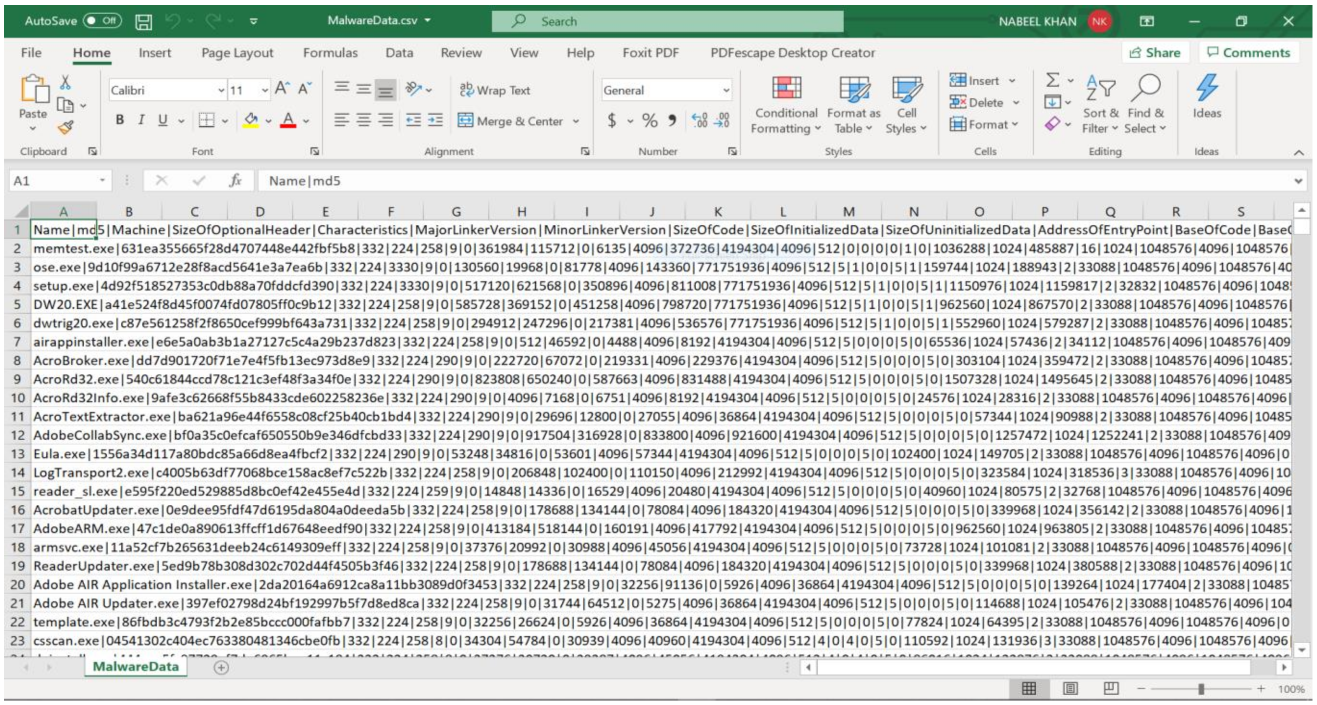
Phishcoop.csv:



A1: id

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF
4	3	1	0	1	1	1	-1	-1	-1	-1	1	1	-1	1	0	-1	-1	-1	-1	0	1	1	1	1	1	-1	1	-1	1	0	-1	-1
5	4	1	0	1	1	1	-1	-1	-1	1	1	1	-1	-1	0	0	-1	1	1	0	1	1	1	1	-1	-1	1	-1	1	-1	1	-1
6	5	1	0	-1	1	1	-1	1	1	-1	1	1	1	1	0	0	-1	1	1	0	-1	1	-1	1	-1	-1	0	-1	1	1	1	1
7	6	-1	0	-1	1	-1	-1	1	1	-1	1	1	-1	1	0	0	-1	-1	-1	0	1	1	1	1	1	1	1	-1	1	-1	-1	1
8	7	1	0	-1	1	1	-1	-1	-1	1	1	1	1	-1	-1	0	-1	-1	-1	0	1	1	1	1	1	-1	-1	-1	1	0	-1	-1
9	8	1	0	1	1	1	-1	-1	-1	1	1	1	-1	-1	0	-1	-1	1	1	0	1	1	1	1	-1	-1	0	-1	1	0	1	-1
10	9	1	0	-1	1	1	-1	1	1	-1	1	1	-1	1	0	1	-1	1	1	0	1	1	1	1	1	-1	1	1	1	0	1	1
11	10	1	1	-1	1	1	-1	-1	1	-1	1	1	1	1	0	1	-1	1	1	0	1	1	1	1	1	-1	0	-1	1	0	1	-1
12	11	1	1	1	1	1	-1	0	1	1	1	1	1	-1	0	0	-1	-1	-1	0	1	1	1	1	-1	1	1	1	1	-1	-1	1
13	12	1	1	-1	1	1	-1	1	-1	-1	1	1	1	1	-1	-1	-1	-1	-1	0	1	1	1	1	-1	-1	-1	-1	1	0	-1	-1
14	13	-1	1	-1	1	-1	-1	0	0	1	1	1	-1	-1	-1	1	-1	1	1	0	-1	1	-1	1	1	-1	-1	-1	1	0	1	-1
15	14	1	1	-1	1	1	-1	0	-1	1	1	1	1	-1	-1	-1	-1	1	1	0	1	1	1	1	-1	-1	0	-1	1	1	1	-1
16	15	1	1	-1	1	1	1	-1	1	-1	1	1	-1	1	0	1	1	1	1	0	1	1	1	1	1	-1	1	-1	1	-1	1	1
17	16	1	-1	-1	-1	1	-1	0	0	1	1	1	1	-1	-1	0	-1	1	1	0	1	1	1	1	1	-1	-1	-1	1	0	1	-1
18	17	1	-1	-1	1	1	-1	1	1	-1	1	1	-1	1	0	-1	-1	-1	-1	0	1	1	1	1	1	-1	0	-1	1	1	-1	-1
19	18	1	-1	1	1	1	-1	-1	0	1	1	-1	1	1	0	-1	-1	-1	-1	0	1	1	1	1	-1	1	1	-1	1	1	-1	-1
20	19	1	1	1	1	1	-1	-1	1	1	1	1	-1	-1	0	-1	-1	-1	-1	0	1	1	1	1	1	-1	-1	1	1	-1	-1	1
21	20	1	1	1	1	1	-1	-1	1	-1	1	1	1	1	0	0	-1	-1	-1	0	-1	-1	-1	-1	1	-1	0	-1	1	0	-1	1
22	21	1	0	-1	1	1	-1	0	1	-1	1	1	1	1	0	0	-1	-1	-1	0	-1	1	-1	1	-1	1	1	-1	1	-1	-1	1
23	22	1	0	1	1	1	-1	0	1	1	1	1	-1	-1	0	-1	-1	-1	-1	0	1	1	1	1	-1	1	-1	-1	1	0	-1	1
24	23	1	1	1	1	1	-1	-1	-1	-1	1	1	-1	1	0	0	-1	1	1	0	1	1	1	1	1	1	0	-1	1	-1	1	1
25	24	1	1	1	1	1	-1	1	0	-1	1	1	1	1	0	0	-1	1	1	0	1	1	1	1	1	1	1	-1	1	-1	1	1
26	25	1	-1	-1	-1	1	-1	1	1	-1	1	1	-1	-1	0	0	-1	1	1	0	1	1	1	1	1	1	-1	-1	1	0	1	1
27	26	1	-1	1	1	1	-1	0	1	-1	1	1	1	1	1	0	-1	1	1	0	1	1	1	1	-1	1	1	-1	1	0	1	1
28	27	1	-1	1	1	1	-1	0	-1	1	1	1	-1	-1	-1	-1	-1	-1	-1	0	1	1	1	1	1	1	0	-1	1	-1	-1	-1
29	28	1	-1	-1	1	1	1	-1	1	1	1	1	1	-1	1	0	-1	-1	-1	0	1	1	1	1	1	-1	0	-1	1	0	-1	1
30	29	1	-1	-1	1	-1	1	-1	1	-1	1	1	1	1	1	0	-1	1	1	1	1	1	1	1	-1	-1	1	-1	1	-1	1	1
31	30	1	-1	1	1	1	-1	-1	1	-1	1	1	1	1	1	0	-1	1	1	0	1	1	1	1	-1	1	1	1	1	0	1	1
32	31	1	-1	1	1	1	-1	-1	1	-1	1	1	-1	1	0	1	-1	1	1	0	1	1	1	1	-1	-1	-1	-1	1	0	1	1
33	32	1	-1	1	1	1	-1	-1	1	-1	-1	1	-1	1	0	-1	-1	-1	-1	0	-1	1	-1	-1	1	-1	1	1	1	0	-1	1
34	33	1	-1	1	1	1	1	1	1	-1	1	1	1	1	1	1	1	-1	-1	0	1	1	1	1	1	1	-1	-1	1	-1	-1	1
35	34	1	0	1	1	1	-1	-1	1	-1	1	1	1	1	0	1	-1	-1	-1	0	1	1	1	1	1	1	1	-1	1	0	-1	1
36	35	1	0	1	1	1	-1	0	0	1	1	1	1	-1	-1	-1	-1	-1	-1	0	1	1	1	1	1	-1	1	-1	1	1	-1	-1
37	36	1	0	1	1	1	-1	-1	-1	-1	1	1	-1	1	-1	0	-1	1	1	0	1	1	1	1	-1	-1	-1	-1	1	1	1	-1
38	37	1	0	1	1	1	-1	-1	1	-1	-1	1	-1	-1	-1	0	-1	1	1	0	-1	1	-1	-1	-1	-1	0	-1	1	0	1	-1
39	38	1	0	1	1	1	-1	-1	-1	1	-1	1	1	-1	-1	0	-1	1	1	0	-1	1	-1	1	-1	-1	0	-1	1	0	1	-1
40	39	1	1	1	1	1	1	-1	1	-1	1	1	1	1	1	-1	1	-1	-1	0	1	1	1	1	1	-1	1	-1	-1	1	-1	1
41	40	1	-1	1	1	1	-1	0	-1	-1	1	1	-1	-1	1	0	-1	1	1	0	1	1	1	1	-1	-1	1	-1	1	0	1	1
42	41	1	-1	1	1	1	1	1	1	-1	1	1	-1	1	1	0	-1	1	1	0	1	1	1	1	1	-1	1	-1	1	0	1	1
43	42	-1	-1	-1	1	-1	-1	1	1	-1	1	1	-1	1	0	-1	-1	1	1	1	1	1	1	1	1	-1	1	-1	-1	-1	1	1
44	43	1	-1	1	1	1	-1	-1	-1	1	-1	-1	1	-1	-1	-1	-1	-1	-1	0	-1	1	-1	1	-1	-1	1	-1	1	-1	-1	-1
45	44	1	-1	1	1	1	-1	0	0	1	-1	1	-1	-1	-1	0	-1	-1	-1	0	-1	1	-1	1	1	-1	-1	-1	1	0	-1	-1
46	45	1	-1	1	-1	1	-1	0	-1	-1	1	1	1	1	-1	1	-1	-1	-1	0	1	1	1	1	-1	1	1	-1	1	1	-1	-1

MalwareData:



A1: Name|md5

```
1  Name|md5|Machine|SizeOfOptionalHeader|Characteristics|MajorLinkerVersion|MinorLinkerVersion|SizeOfCode|SizeOfInitializedData|SizeOfUninitializedData|AddressOfEntryPoint|BaseOfCode|Base
2  memtest.exe|631ea355665f28d4707448e442fbf5b8|332|224|258|9|0|361984|115712|0|6135|4096|372736|4194304|4096|512|0|0|0|0|1|0|1036288|1024|485887|16|1024|1048576|4096|1048576|
3  ose.exe|9d10f99a6712e28f8acd5641e3a7ea6b|332|224|3330|9|0|130560|19968|0|81778|4096|143360|771751936|4096|512|5|1|0|0|5|1|159744|1024|188943|2|33088|1048576|4096|1048576|40
4  setup.exe|4d92f518527353c0db88a70fddcfd390|332|224|3330|9|0|517120|621568|0|350896|4096|811008|771751936|4096|512|5|1|0|0|5|1|1150976|1024|1159817|2|32832|1048576|4096|1048
5  DW20.EXE|a41e524f8d45f0074fd07805ff0c9b12|332|224|258|9|0|585728|369152|0|451258|4096|798720|771751936|4096|512|5|1|0|0|5|1|962560|1024|867570|2|33088|1048576|4096|1048576|
6  dwtrig20.exe|c87e561258f2f8650cef999bf643a731|332|224|258|9|0|294912|247296|0|217381|4096|536576|771751936|4096|512|5|1|0|0|5|1|552960|1024|579287|2|33088|1048576|4096|10485
7  airappinstaller.exe|e6e5a0ab3b1a27127c5c4a29b237d823|332|224|258|9|0|512|46592|0|4488|4096|8192|4194304|4096|512|5|0|0|0|5|0|65536|1024|57436|2|34112|1048576|4096|1048576|409
8  AcroBroker.exe|dd7d901720f71e7e4f5fb13ec973d8e9|332|224|290|9|0|222720|67072|0|219331|4096|229376|4194304|4096|512|5|0|0|0|5|0|303104|1024|359472|2|33088|1048576|4096|10485
9  AcroRd32.exe|540c61844ccd78c121c3ef48f3a34f0e|332|224|290|9|0|823808|650240|0|587663|4096|831488|4194304|4096|512|5|0|0|0|5|0|1507328|1024|1495645|2|33088|1048576|4096|10485
10 AcroRd32Info.exe|9afe3c62668f55b8433cde602258236e|332|224|290|9|0|4096|7168|0|6751|4096|8192|4194304|4096|512|5|0|0|0|5|0|24576|1024|28316|2|33088|1048576|4096|1048576|4096|
11 AcroTextExtractor.exe|ba621a96e44f6558c08cf25b40cb1bd4|332|224|290|9|0|29696|12800|0|27055|4096|36864|4194304|4096|512|5|0|0|0|5|0|57344|1024|90988|2|33088|1048576|4096|10485
12 AdobeCollabSync.exe|bf0a35c0efcaf650550b9e346dfcbd33|332|224|290|9|0|917504|316928|0|833800|4096|921600|4194304|4096|512|5|0|0|0|5|0|1257472|1024|1252241|2|33088|1048576|409
13 Eula.exe|1556a34d117a80bdc85a66d8ea4fbcf2|332|224|290|9|0|53248|34816|0|53601|4096|57344|4194304|4096|512|5|0|0|0|5|0|102400|1024|149705|2|33088|1048576|4096|1048576|4096|0
14 LogTransport2.exe|c4005b63df77068bce158ac8ef7c522b|332|224|258|9|0|206848|102400|0|110150|4096|212992|4194304|4096|512|5|0|0|0|5|0|323584|1024|318536|3|33088|1048576|4096|10
15 reader_sl.exe|e595f220ed529885d8bc0ef42e455e4d|332|224|259|9|0|14848|14336|0|16529|4096|20480|4194304|4096|512|5|0|0|0|5|0|40960|1024|80575|2|32768|1048576|4096|1048576|4096
16 AcrobatUpdater.exe|0e9dee95fdf47d6195da804a0deeda5b|332|224|258|9|0|178688|134144|0|78084|4096|184320|4194304|4096|512|5|0|0|0|5|0|339968|1024|356142|2|33088|1048576|4096|1
17 AdobeARM.exe|47c1de0a890613ffcff1d67648eedf90|332|224|258|9|0|413184|518144|0|160191|4096|417792|4194304|4096|512|5|0|0|0|5|0|962560|1024|963805|2|33088|1048576|4096|10485
18 armsvc.exe|11a52cf7b265631deeb24c6149309eff|332|224|258|9|0|37376|20992|0|30988|4096|45056|4194304|4096|512|5|0|0|0|5|0|73728|1024|101081|2|33088|1048576|4096|1048576|4096|
19 ReaderUpdater.exe|5ed9b78b308d302c702d44f4505b3f46|332|224|258|9|0|178688|134144|0|78084|4096|184320|4194304|4096|512|5|0|0|0|5|0|339968|1024|380588|2|33088|1048576|4096|10
20 Adobe AIR Application Installer.exe|2da20164a6912ca8a11bb3089d0f3453|332|224|258|9|0|32256|91136|0|5926|4096|36864|4194304|4096|512|5|0|0|0|5|0|139264|1024|177404|2|33088|10485
21 Adobe AIR Updater.exe|397ef02798d24bf192997b5f7d8ed8ca|332|224|258|9|0|31744|64512|0|5275|4096|36864|4194304|4096|512|5|0|0|0|5|0|114688|1024|105476|2|33088|1048576|4096|104
22 template.exe|86fbdb3c4793f2b2e85bccc000fafbb7|332|224|258|9|0|32256|26624|0|5926|4096|36864|4194304|4096|512|5|0|0|0|5|0|77824|1024|64395|2|33088|1048576|4096|1048576|4096|0
23 csscan.exe|04541302c404ec763380481346cbe0fb|332|224|258|8|0|34304|54784|0|30939|4096|40960|4194304|4096|512|4|0|4|0|5|0|110592|1024|131936|3|33088|1048576|4096|1048576|4096|
```