MSDS 7349 Data and Network Security Homework Basic Security

Due Week 8

Name: Andrew Abbott

What to Submit

Your final submission shall be a single pdf document that includes this document, screen captures of your exercises plus your answers to each of the written questions (if any). Note that you are expected to clearly label each section so as to make it clear to the instructor what files and data belong to which exercise/question.

Collaboration is expected and encouraged; however, each student must hand in their own homework assignment. To the greatest extent possible, answers should not be copied but, instead, should be written in your own words. Copying answers from anywhere is plagiarism, this includes copying text directly from the textbook. Do not copy answers. Always use your own words. For each question list all persons with whom you collaborated and list all resources used in arriving at your answer. Resources include but are not limited to the textbook used for this course, papers read on the topic, and Google search results. Note that 'Google' is not a resource. Don't forget to place your name on the document.

Exercise 1 : UNIX Password Cracker

The goal of this exercise is to write a password cracker for the UNIX file system. UNIX stores all passwords in the file /etc/passwd. Well, it doesn't store the password itself. Instead, it stores a *signature* of the password by using the password to encrypt a block of zero bits prepended by a *salt* value with a one-way function called crypt( ). The result of the crypt( ) function is stored in the /etc/passwd file. For example, for a password of "*egg*" and salt equal to "*HX*", the function crypt('egg','HX') returns HX9LLTdc/jiDE.

When you try to log in, the program /bin/login takes the password that you typed, uses crypt( ) to encrypt a block of zero bits, and compares the result of this function with the value stored in the /etc/passwd file.

The security of this approach rests on both the strength of the crypt( ) function and the difficulty in guessing a user's password. The crypt( ) algorithm has proven to be highly resistant to attacks. Conversely, the user's choices for passwords have been found to be relatively easy to guess, with many passwords being words contained in the dictionary.

To write our UNIX password cracker, we will need to use the crypt( ) algorithm that hashes UNIX passwords. Fortunately, the crypt library already exists in the Python 2.79 standard library (on UNIX-based operating systems). (Note: for Windowsbased operating systems, you will need to find the correct way to import the UNIX crypt() algorithm.) To calculate the encrypted UNIX password signature, we simply call the function crypt.cypt( ) and pass it the password and salt as parameters. This function returns the signature as a string.

A simple dictionary attack involves computing the possible signatures generated for each word in the dictionary with a range of salt values.

Let's create our first password cracker using a dictionary attack.

1) Create a file called cracker.py. Start your program by reading in the HW2-passwords.txt file and, for each password found in the file, iterate through each dictionary word found in the HW2-dictionary.txt file and appropriate salt value.
   Report out the password found, if any, for each user. If no password is found, indicate that no password was found.

In [27]:
```
import crypt

filename = 'HW2passwords.txt' # save the password filename in a variable
dictfile = 'HW2dictionary.txt' # save the dictionary filename in a variable

with open(filename, 'r') as f: # using with open() will close the file when not in use.
    users = f.readlines() # reads the password file into the users variable
with open(dictfile, 'r') as d:
    dict = d.read().splitlines() # creates a list of the words in the dictionary file
users = [user.split(': ') for user in users] # Creates a list of lists

print(users)
print(dict)
```

```
[['victim', 'HX9LLTdc/jiDE', '503:100:Iama Victim:/home/victim:/bin/sh\n'], ['root', 'DFNFxgW7C05fo', '504:100', 'Markus Hess:/r
oot:/bin/bash\n']]
['apple', 'orange', 'egg', 'lemon', 'grapes', 'secret', 'strawberry', 'password']
```

In [34]:
```
for user in users: # for each user
    for pw in dict: #check each word in the dictionary file
        if crypt.crypt(pw,user[1][:2]) == user[1]: # using crypt.crypt and the first 2 characters of the password as the salt
            print(user[0], ': ', pw)               # compare the decrypted password to the dictionary word, print if a match
```

```
victim :  egg
```

2) Using literature review, identify from where you can retrieve the salt value used in generating the signature.

I was able to determine the salt value based on the information in the exercise description.

Exercise 2 : Zip File Password Cracker

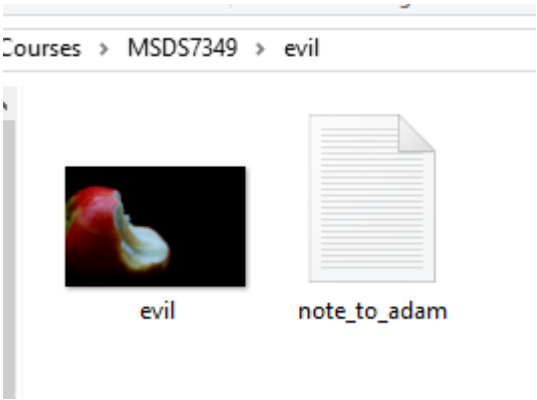
The goal of this exercise is to write a zip file extractor and password cracker. For this exercise, we will use the zipfile library. You may view information about the zipfile library in Python 2.79 by issuing the command help('zipfile') to learn more about the library. Pay close attention to the extractall( ) method. You may use this method to extract the contents from a zip file.

Let's begin the process of writing a zip file password cracker.

1) Write a quick script to test the use of the zipfile library. After importing the library, instantiate a new ZipFile class by specifying the filename of the password-protected zip file (*evil.zip*). utilize the extractall( ) method and specify the optional parameter for the password (*secret*). Execute your script and turn in the code and output.

In [1]:
```
import zipfile

pwd = b'secret'
zipname = 'evil.zip'
zipfile.is_zipfile(zipname)
with zipfile.ZipFile(zipname) as zf:
    zf.extractall(pwd = pwd)
```



2) Use the except Exception exception handler to catch exceptions and print them out when an incorrect password is used. Execute your script with an incorrect password and exception handler and turn in the code and output.
3) Write a script that performs a dictionary attack on the password protected zip file. Execute your script and turn in the code and output. Be sure to provide user feedback on exceptions thrown.

In [1]:
```
import zipfile

pwd = b'secret'
zipname = 'evil.zip'
zipfile.is_zipfile(zipname)
with zipfile.ZipFile(zipname) as zf:
    try:
        zf.extractall(pwd = pwd)
    except Exception:
        print('Wrong!')
```

In [2]:
```
pwd = b'secret2' #The password is incorrect
zipname = 'evil.zip'
zipfile.is_zipfile(zipname)
with zipfile.ZipFile(zipname) as zf:
    try:
        zf.extractall(pwd = pwd)
    except Exception:
        print('Wrong!')
```

```
Wrong!
```

In [77]:
```
for pw in dict: #Trying each word in the dictionary
    pwd = bytes(pw, 'utf-8')
    try:
        zf.extractall(pwd=pwd)
    except Exception:
        print('Wrong!')
```

```
Wrong!
Wrong!
Wrong!
Wrong!
Wrong!
Wrong!
Wrong!
Wrong!
```

Exercise 3 : Port Scanner

The goal of this exercise is to learn about port scanners for networked systems.

First, create a simple Python-based port scanner. Using the socket library, you will create a script that iterates through a range of IP addresses, and, for each IP address, will identify the active ports available for that IP address. At least ports corresponding to telnet, ftp SSH, smtp, http, imap, and https services should be scanned and identified.

Second, download and install the nmap port scanning software from nmap.org. Utilize nmap to identify the operating system and the open ports of devices on a range of IP addresses.

In [4]:

```
import socket

socket.setdefaulttimeout(0.5)
portlist = [20, 21, 22, 23, 25, 80, 143, 443] #telnet, ftp ssh, smtp, http, imap, https ports
for i in range(116,120):                       #source www.pearsoncertification.com/articles/article.aspx?p=1868080
    for j in range(242,245):  #scans a range of ips, this ip was suggested by Joe as open
        ip = "162.144.%d.%d" % (i, j)
        print('Scanning', ip)
        for k in portlist:
            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            result = s.connect_ex((ip, k))
            if(result==0):
                print('Port %d: OPEN' % (k,))
            s.close()
```

```
Scanning 162.144.116.242
Port 21: OPEN
Port 22: OPEN
Port 80: OPEN
Port 143: OPEN
Scanning 162.144.116.243
Port 21: OPEN
Port 22: OPEN
Port 80: OPEN
Port 143: OPEN
Port 443: OPEN
Scanning 162.144.116.244
Port 21: OPEN
Port 22: OPEN
Port 80: OPEN
Port 143: OPEN
Port 443: OPEN
Scanning 162.144.117.242
Scanning 162.144.117.243
Port 80: OPEN
Scanning 162.144.117.244
Port 21: OPEN
Port 80: OPEN
Port 443: OPEN
Scanning 162.144.118.242
Port 21: OPEN
Port 22: OPEN
Port 80: OPEN
Port 143: OPEN
Port 443: OPEN
Scanning 162.144.118.243
Port 21: OPEN
Port 22: OPEN
Port 80: OPEN
Port 143: OPEN
Port 443: OPEN
Scanning 162.144.118.244
Port 21: OPEN
Port 22: OPEN
Port 80: OPEN
Port 143: OPEN
Port 443: OPEN
Scanning 162.144.119.242
Port 21: OPEN
Port 22: OPEN
Port 80: OPEN
Port 143: OPEN
Port 443: OPEN
Scanning 162.144.119.243
Port 21: OPEN
Port 22: OPEN
Port 80: OPEN
Port 143: OPEN
Port 443: OPEN
Scanning 162.144.119.244
Port 21: OPEN
Port 22: OPEN
Port 80: OPEN
Port 143: OPEN
Port 443: OPEN
```

Nmap Output | Ports / Hosts | Topology | Host Details | Scans

server.artdefehr.com (162.144.116.242)

Host Status

State:	up
Open ports:	14
Filtered ports:	4
Closed ports:	982
Scanned ports:	1000
Up time:	2915144
Last boot:	Thu Jan 26 02:15:12 2017

Addresses

IPv4:	162.144.116.242
IPv6:	Not available
MAC:	Not available

Hostnames

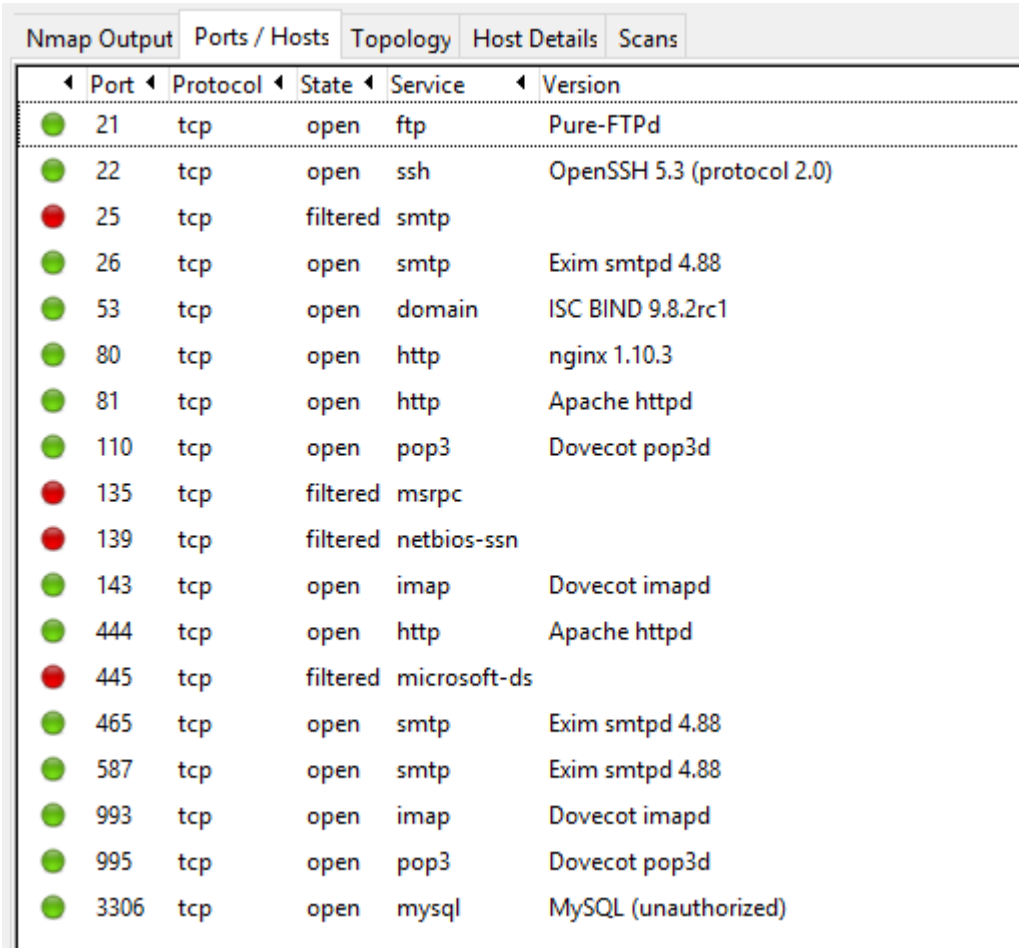
Name - Type: server.artdefehr.com - PTR

Operating System

Name: Linux 2.6.32 - 3.13

Accuracy: 100%

- **Ports used**
- **OS Classes**
- **TCP Sequence**
- **IP ID Sequence**
- **TCP TS Sequence**
- **Comments**



Nmap Output | Ports / Hosts | Topology | Host Details | Scans

Port	Protocol	State	Service	Version
21	tcp	open	ftp	Pure-FTPd
22	tcp	open	ssh	OpenSSH 5.3 (protocol 2.0)
25	tcp	filtered	smtp	
26	tcp	open	smtp	Exim smtpd 4.88
53	tcp	open	domain	ISC BIND 9.8.2rc1
80	tcp	open	http	nginx 1.10.3
81	tcp	open	http	Apache httpd
110	tcp	open	pop3	Dovecot pop3d
135	tcp	filtered	msrpc	
139	tcp	filtered	netbios-ssn	
143	tcp	open	imap	Dovecot imapd
444	tcp	open	http	Apache httpd
445	tcp	filtered	microsoft-ds	
465	tcp	open	smtp	Exim smtpd 4.88
587	tcp	open	smtp	Exim smtpd 4.88
993	tcp	open	imap	Dovecot imapd
995	tcp	open	pop3	Dovecot pop3d
3306	tcp	open	mysql	MySQL (unauthorized)