Advanced Network Security

Lab 3 – Exploiting Vulnerabilities

Overview

During this lab session you will exploit a vulnerable system, extract the password hashes from the target system, and crack the passwords using a password cracking tool. In addition to this you will write a simple password cracker that carries out an offline dictionary attack.

Task 1 – Exploiting a Vulnerabilities

VSFTPD

1. Start your Kali and Metasploitable virtual machines
2. Refer to the lab sheet from lab week and run a port scan against metasploitable using nmap
3. Initiate the database for Armitage using:
    - `msfdb init`
4. Start Armitage from the terminal
5. Run an nmap scan against Metasploitable using Armitage
    - Hosts -> Nmap Scan -> Intense Scan
6. Find attacks for the Metasploitable virtual machine using Armitage
7. Execute the VSFTPD exploit, using the reverse connection option, access the shell via the interactive menu, and use the `uname -a` command to confirm the shell is on the Metasploitable machine.

Other Exploits

1. Use the `/multi/misc/java_rmi_server` exploit to gain a reverse shell on metasploitable
2. Connect to the backdoor running on port 1524 using the following command (replace RHOST with the IP address of metasploitable):
    - `nc RHOST 1524`

Task 2 – Exfiltration of information

Now we have access to the remote machine, we want to extract the passwords so they can be cracked on the attack machine. We can use the SCP command from the first lab session to transfer the passwords file;

1. Use one the reverse shells from task 1 to access the metasploitable server.
2. Copy the /etc/shadow file to the home directory of msfadmin
3. Research how to use SCP on Kali to retrieve the etc file from the metasploitable server
4. Research how to use both John the Ripper and Johnny to crack the hashes stored in the etc file
5. Explain why passwords are stored as hashes, and how this storage mechanism can be improved.

Task 4 – Practical Research (Part of Portfolio)

Research the vulnerabilities and exploits from task 1 and document your findings. Identify other vulnerabilities in the target system, using the reconnaissance you carried out in last week's lab session. Exploit some vulnerabilities in the target system and document both the vulnerabilities and the exploits you used to exploit the system.

Programming Tutorial – Password Cracking with Python

Tips & Tricks

- If you simply copy and paste the code from the examples, without carrying out the exercises, you will learn nothing. Please work through the exercises, and ask questions if you get stuck.
- This lab session may take more than one week to complete.
- Before tackling a task, consider the manual process you would carry out to complete the task. You may need to devise a metaphor for the task in order to do this. Check the metaphor with a friend.
- When tackling a programming problem, break the problem down into small sub tasks, and consider each task individually.
- When developing your code, use a small scale case study to simplify the initial phases of design and testing.

Task 1 – Manual Problem Solving Process – Plain Text Dictionary Attack

You have been given two files, one containing a list of user passwords (Passwords.txt), the other a dictionary containing potential passwords/phrases (Dictionary.txt). Each line in each of the files contains a new password or phrase.

Describe the steps you would take to figure out which of the passwords/phrases, from the dictionary, match those in the passwords file. Document your proposed solution in Libre Office.

Task 2 – Manual Problem Solving Process – Hashed Password Dictionary Attack

You have been given two files, one containing a list of hashed passwords (HashedPasswords.txt), the other a dictionary containing potential passwords/phrases (Dictionary.txt). Each line in each of the files contains a new password or phrase.

Describe the steps you would take to figure out which of the passwords/phrases, from the dictionary, match those in the passwords file. **Highlight the additional steps you would have to take due to the passwords being hashed.** Document your proposed solution in Libre Office.

Task 3 – File Input & Output

Part of the manual solutions you proposed for tasks 1 & 2, no doubt included opening the files to read the passwords and candidate passwords from.

Parse the information at the link below, to learn how to open a file, you do not have to read the entire page, the example code is probably the most important part for you to read.

https://docs.python.org/2/library/stdtypes.html#file-objects

The code sample below will open the file, read each line from the file and print it to the screen.

```
1   #!/usr/bin
2
3   import sys
4   import os
5
6   # Open the Dictionary File
7   DictionaryFile = open("Dictionary.txt", "r")
8
9   # Print each line from the file
10  for line in DictionaryFile:
11      print line
12
13  # Close the file
14  DictionaryFile.close()
```

Write a program that;

1. Opens the dictionary file
2. Displays each line on the screen.

Task 4 – Lists

Part of the solutions you proposed in tasks 1, likely involved comparing the passwords or phrases stored in the dictionary file and with those in the passwords file. In order to store the passwords from the password file in memory, a data structure is required. In this exercise we are going to use a list.

A list in Python is roughly equivalent to an ArrayList in Java, it can store a number of items, and is dynamic so you don't need to specify how many items it will hold.

A list is a useful place to store data items. Parse the information at the link below to learn how to add items to a list.

https://docs.python.org/2/tutorial/datastructures.html

The code sample below creates a list, and read the lines from the dictionary file into the list.

```
# Create a phrase list to store the lines from the dictionary in
PhraseList = []

# Store each line from the dictionary to to phrase list
for Phrase in DictionaryFile:
    PhraseList.append(Phrase)

# print the list
print PhraseList
```

Write a program that;

1. Opens the dictionary file.
2. Displays each line on the screen.
3. Adds each line to a list
4. Prints the list to the screen

Task 5 – Dictionaries

Part of your solution for task 2 will likely entail the creation of a list of passwords from the dictionary file, and a list of the hashes they map to. This will allow you to compare each of the hashes with the hashed password, then when a match occurs look up the password that corresponds to the hash.

A dictionary is a suitable data structure for this task, as it stores keys that map to values.

Think of a conventional dictionary that contains words and descriptions. The words are the keys, and the descriptions are the values.

To solve our problem, we need to store the hashes as key, so that we can search the dictionary, and the passwords as values, so when we find a matching key we can look up the value.

The following link describes how hash values can be generated using hashlib;

https://docs.python.org/2/library/hashlib.html

The following link describes how to use dictionaries in Python;

https://docs.python.org/2/tutorial/datastructures.html

The code sample below, shows the use of hashlib to generate hash values and add them to a dictionary along with their corresponding password.

```
#!/usr/bin

import sys
import os
import hashlib

# Open the Dictionary File
DictionaryFile = open("Dictionary.txt", "r")

# Create a phrase dictionary to store the lines from the dictionary and their hashes in
PhraseDict = {}

# Read each phrase from the dictionary file
for Phrase in DictionaryFile:
    #create a hash value for the phrase
    m = hashlib.md5()
    m.update(Phrase)
    hashvalue = m.hexdigest()
    #Add the key value pair to the dictionary (key=hash:value=phrase)
    PhraseDict [hashvalue] = Phrase

# Print the phrase dictionary, so you can see what it looks like
print PhraseDict
```

Write a program that:

1. Reads each line from the dictionary file.
2. Creates a hash for each password/line/phrase.
3. Stores the hash and its corresponding password/line/phrase in a dictionary.
4. Display the dictionary on the screen.

Task 6 – Putting it All Together

Write a program based on your solution for task 1. Your program should determine which of the passwords/phrases in the dictionary file match those in the passwords file.

The documentation on control flows at the link below, explains how to use a for loop and an if statement;

https://docs.python.org/2/tutorial/controlflow.html

The code sample below shows how comparison of the passwords can be carried out using a for loop and if statement;

```
# Compare each line in password file with the phrase list
for Password in PasswordFile:
    # If a match is found display the password that matches
    if Password in PhraseList:
        print Password.rstrip('\n')
```

Task 7 – Making a Hash of It

Write a program based on your solution for task 2. Your program should determine which of the passwords in the dictionary file, match the hashed password stored in the hashedpassword file.

Use the same control flow documentation as task 6.

The code sample below shows the use of rstrip to remove the newline character (line 3) from the hashedpassword before comparing it to the dictionary. If the new line character is not removed no keys in the dictionary will match the hashedpassword. (This drove me mad for a short time on Saturday afternoon ☺)

```
1  for HashedPassword in HashedPasswordFile:
2      # strip the new line character from the password
3      HashedPassword = HashedPassword.rstrip('\n')
4      # If the hashedpassword matches one in the dictionary, display the associated password
5      if HashedPassword in PhraseDict:
6          print "This password is", PhraseDict[HashedPassword]
```

Task 8 – Extended Task

Propose a solution using a salt to mitigate the attack on the hashed passwords. Write a program that implements your solution, and describe the caveats of your solution.