Python Scripting Continued

Stuck? Join our Discord for help! https://discord.gg/wvfe3XJ

In this document, we'll cover the functionality of different libraries. While these functions may not be individually relevant, they are used as part of larger scripts to perform a combination of actions.

OS Library

The OS library provides different functionality that interfaces with an operating system. Examples include viewing file permissions, interacting with processes and various other operating system functions. Today we'll look at using this library to list files in a particular directory

```
1  import os
2
3  listOfFiles = os.listdir("/")
4  print(type(listOfFiles))
5  for l in listOfFiles:
6      print(l)
```

To start using a library, we need to import it as in line 1. Line 3 shows the *listdir* function taking the path as a parameter. Line 4 is printing the output of the variable and line 5 is a loop that iterates through the variable and prints the values in the variable. Running this python script gives this output:

```
<class 'list'>
cdrom
tmp
swapfile
lib64
dev
proc
.autorelabel
run
etc
data
vmlinuz
srv
initrd.img.old
mnt
home
initrd.img
sys
lost+found
usr
lib
root
media
lib32
var
boot
opt
bin
snap
```

The first line prints the type of the variable, which is a list. This makes sense as the following lines in the code iterate through the list and print the output.

There are various other useful functions in the OS library.

Zipfile Library

From time to time, we'll encounter compressed files. A common compressed file format is the ".zip" format. We would want to extract these files for future use. We can do this using the zipfile library.

```
1  import zipfile
2  with zipfile.ZipFile('/path-of-file', 'r') as zip_ref:
3      zip_ref.extractall('/path-to-extract')
```

Like every library, we start out by importing it. Line 2 starts with the with keyword. The with keyword follows the following format

with some-code as variable:
    do action

In the above, the zipfile library is taking the file provided by the path in read mode and creating a zipfile object with it, and referencing it as the variable zip_ref. On line 3, it's referencing the variable to extract the contents of the zip file. The *with* directive is useful because it is used to manipulate files; all actions inside the directive are done while the file is open, which means that a programmer does not explicitly have to close a file after carrying out file manipulation.

The above piece of code taking a zip file at a particular path and extracts the contents of the file to a particular directory.

Exiftool

Exiftool is a command line tool that extracts metadata from files. Metadata is usually provides a trove of information such a file owner, software used to create the file, and more. This data is used as part of the enumeration step when carrying out attacks. For example, if a user finds a particular software name and version used to create a file, they can find exploits against these particular softwares.

The output of the command line tool usually looks as follows:

```
ashu@ashu-Inspiron-5379 ~/D/t/christmas-challenges> exiftool exploiting-services

ExifTool Version Number         : 10.80
File Name                       : exploiting-services
Directory                       : .
File Size                       : 8.3 kB
File Modification Date/Time     : 2019:12:10 23:47:25+00:00
File Access Date/Time           : 2019:12:16 07:35:05+00:00
File Inode Change Date/Time     : 2019:12:10 23:47:25+00:00
File Permissions                : rw-rw-r--
Error                           : Unknown file type
```

While the command line tool works well for single files, it can be difficult to do for multiple files at the same time. Here's where the python library comes in.

```
1  import exiftool
2
3  files = ["/exploiting-services"]
4  with exiftool.ExifTool() as et:
5      metadata = et.get_metadata_batch(files)
6  for d in metadata:
7    print(d)
8
```

After cloning the repository(as shown on the documentation page attached above), we use the same with directive to call exiftool and print the metadata(as shown below).

{u'File:FilePermissions': 664, u'File:FileSize': 8471, u'SourceFile': u'/home/ashu/Documents/thm-rooms/christmas-challenges/exploiting-services', u'File:FileInodeChangeDate': u'2019:12:10 23:47:25+00:00', u'File:FileAccessDate': u'2019:12:16 07:35:05+00:00', u'ExifTool:Error': u'Unknown file type', u'File:FileModifyDate': u'2019:12:10 23:47:25+00:00', u'ExifTool:ExifToolVersion': 10.8, u'File:FileName': u'exploiting-services'}

Please note that the script needs to be run from the same location as the cloned folder.

Reading Files

The most common way to read a file is using the same with directive with the open statement

```
1  with open('example.txt', 'r') as reader:
2      f = reader.readlines()
3  print(f)
```

In this case, we want to open the *example.txt* file in **r**ead mode. Once we open it, we read all the lines in the file until the end and store each line in a list. Running this will output the following:

```
ashu@ashu-Inspiron-5379 ~/D/t/christmas-challenges> python3 other-scripting/read-file.py
['one\n', 'two\n', 'three\n']
```

Since python is reading each line, each entry in the list contains a new line character that needs to be stripped when individually accessed. For example:

```
5  for line in f:
6      new_line = line.strip('\n')
7      print(new_line)
```

Line 6 removes the new line character using the strip function. Alternatively, we can also use:
reader.read().splitlines()
Which also removes the new line character.