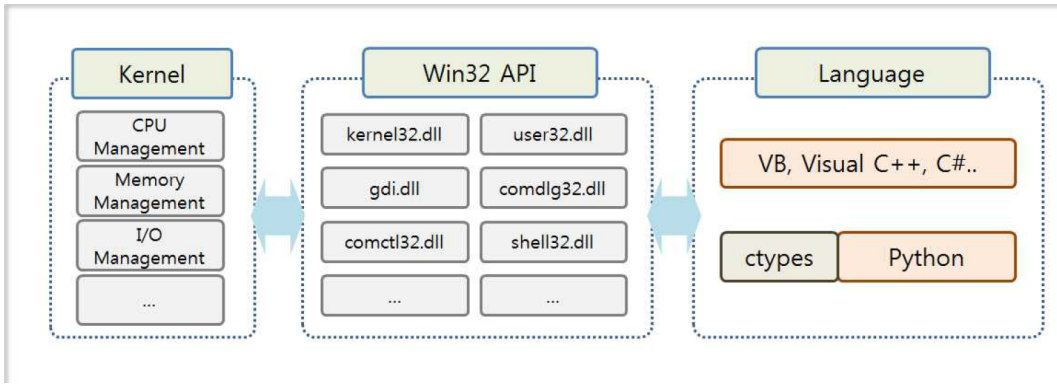
LABORATOR 1-- Application Hacking cu Python.

1 Introducere

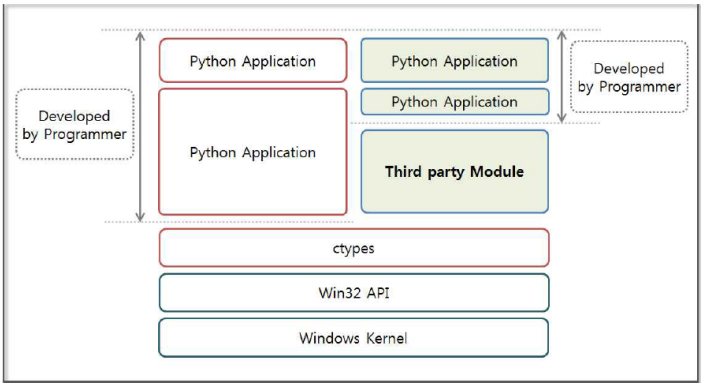
Pentru a pirata o aplicație Windows folosind Python, este necesar pentru a avea cunoștințe de bază despre API-ul Windows. API Windows constă dintr-un set de interfețe de programare a aplicațiilor (API) furnizate de Microsoft. Pentru a dezvolta o aplicație folosind Windows API, este necesar să utilizați diverse funcții care sunt suportate de sistemul de operare (kernel). Pentru utilizare în mod obișnuit Mediu Windows pe 32 de biți, API-ul Windows numit Win32 API este suportat.



Folosim biblioteci precum "lib" și "DLL" atunci când o aplicație Windows este dezvoltată. "Lib" este o bibliotecă statică care este inclusă atunci când este creat fișierul executabil Windows. "DLL" (dynamical lynked library) oferă o bibliotecă dinamică care este apelată în timpul execuției aplicației. Putem folosi la maximum Win32 API sub formă de DLL, unde sunt de obicei următoarele DLL-uri sunt folosite:

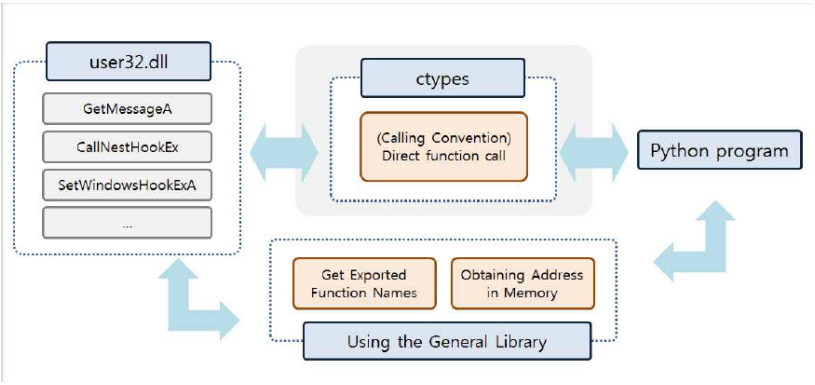
Type	Characteristics
kernel32.dll	Provides the ability to access basic resources, such as threads, file system, devices, processes
user32.dll	Provides the ability to change the user interface, including creating and managing windows, receiving window messages, displaying text on the screen, and presenting a message box
advapi32.dll	Provides the ability to modify the registry, shutdown and restart the system, also provides support functions to start / end / generate Windows services, account management
gdi32.dll	Manages functions for the printer, monitor and other output devices
comdlg32.dll	Open a file, save a file, manage the standard dialog window associated with the selected color and font
comctl32.dll	Status bar, progress bar, access to applications that are supported by the operating system, such as the toolbar
shell32.dll	Provides the functionality of the shell of the operating system so that the applications can have access
netapi32.dll	Provides a variety of communication features that are supported by the operating system to the applications

Exemplu de biblioteca externa folosita de python:



La început, atunci când utilizați API-ul Win32 și ctypes, poate fi dificil de utilizat apelurile API Win32 prin utilizarea ctypes. Există o cantitate extinsă de cunoștințe care este necesară în prealabil, de exemplu un mecanism de apelare a

funcției, valori returnate și tipuri de date. Cu toate acestea, ctypes pot fi folosite pentru biblioteci native care sunt susținute de o varietate de sisteme de operare, care oferă o unealtă puternică. Pentru a implementa tehnici sofisticate de hacking, ar trebui înțeles conceptul de bază al c-tipurilor (ctypes). C-typeurile simplifică procedura de a efectua apeluri dinamice de biblioteci, iar acestea suportă tipuri complexe de date in C și au avantajul oferind funcționalitate de nivel scăzut. Dacă respectați convențiile la apelarea funcțiilor pentru a profita de ctypes, puteți apela API-ul care este furnizat direct de MSDN. Conceptul de baza a unui Ctype se poate obseva mai jos



2 Funcții in Python

Ca și în cazul altor limbaje, Python folosește funcții pentru a fi îmbunătățit structural și pentru a elimina codul duplicat. Python suportă o varietate de funcții încorporate care pot fi utilizate prin includerea a unui apel de funcție sau importul unui modul. Cel mai frecvent utilizată este funcția ”print” și poate fi folosită fără instrucțiuni de import, dar funcțiile matematice pot fi utilizate numai după importarea modulului ”math”.

```
import math
print “value of cos 30:”, math.cos(30)

>>>>>cos value of 30: 0.154251449888
```

Este posibil să dezvoltam programe cu Python atât în mod procedural cat şi orientat pe obiecte. Pentru a dezvolta simplu programe de hacking, este convenabil să folosiți o manieră procedurală. Cu toate acestea, pentru a dezvolta programe complexe care sunt necesare pentru funcționarea într-un mediu diferit,

este necesară structurarea programului. Un limbaj orientat pe obiect poate fi folosit pentru a îmbunătăți productivitatea în timpul dezvoltării permițând reutilizarea. Dacă utilizați un limbaj orientat pe obiecte, este posibil a dezvolta un program care este construit logic. Structura de bază pentru declararea unei clase este următoarea.

```
class name:                          #(1)
  def __init__(self, argument):     #(2)
  def functioin(argument):           #(3)

class name(inherited class ame):     #(4)
  def functioin (argument):
```

(1) Creați o clasă: dacă specificați un nume de clasă după ce ați folosit cuvânt rezervat "clasă", clasa este declarată.
(2) Constructor: Funcția "init" este un constructor care este apelat implicit atunci când clasa este creată.
(3) Funcție: Este posibil să se declare o funcție în clasă. O instanța este apoi generată pentru a apela funcția. **(4) Moștenire**: Pentru a moșteni dintr-o altă clasă un nume, numele de clasa moştenită trebuie folosită ca argument când clasa este declarată. Moștenirea acceptă utilizarea variabilelor membre și funcțiile clasei superioare așa cum sunt.

3 Crearea unei clase:

```
class Hero:                                    #(1)
   def __init__(self, name, age, weight):      #(2)
      self.name = name                         #(3)
      self.age = age
      self.weight = weight
   def printHero(self):                        #(4)
      print "\n"
      print "---------------------------------------"
      print "1.name:" , self.name                #(5)
      print "2.age:" , self.age
      print "3.weight:" , self.weight
```

```
    def __init__(self, inSkill, inPower, idx):
        Hero.__init__(self, "hong gil dong", 18, 69.3)  #(7)
        self.skill = inSkill
        self.power = inPower
        self.idx = idx
    def printSkill(self):
        print "4.armed weapon:" , self.skill + "[ power:" ,
self.power[self.idx], "]"

skill = ["sword","spear","bow","axe"]
power = [98.5, 89.2, 100, 79.2]

querySkill = raw_input("select weapon: ")

i=0

for each_item in skill:
    if(each_item == querySkill):
        myHero = MyHero(querySkill, power, i)        #(8)
        myHero.printHero()                           #(9)
        myHero.printSkill()
    i = i+1

print "---------------------------------------"
print "\n"
```

Aici avem:
(1) Declarație de clasă: Declarați clasa ”Hero”.
(2) Declarația constructorului: Declarați constructorul care ia trei argumente și ”self” reprezentând clasa în sine.
(3) Inițializarea variabilelor: Inițializați variabilele clasei prin atribuirea argumentelor.
(4) Declarație de funcție: Declarați funcția ”printHero” în clasa.
(5) Utilizarea variabilelor: Folosiți variabile de clasă în formatul de ”self.nume variabilă”.
(6) Moștenirea clasei: Declarați clasa ”MyHero” care moștenește clasa ”Erou”.
(7) Apelarea constructorului: Generați și inițializați obiectul prin apelând constructorul clasei superioare.
(8) Crearea unei clase: Generați o clasă ”MyHero”. Treceți argumentele cerute constructorului.
(9) Funcția de clasă de apelare: sarcinile sunt executate prin apelarea funcțiilor care sunt declarate pentru obiectul ”myHero”.

4 Șiruri în Python (String)

Formatul șir este o tehnică care poate fi folosită pentru a introduce un anumit valoare în șirul pe care doriți să-l imprimați. Tipul valorii inserat este determinat de un cod de format șir. Formatul șirului este folosit în felul următor.

$$print(output\ string1\ \%s\ outputstring2\ \%\ inserted\ string)$$

Introduceți codul formatului șirului în mijlocul șirului de ieșire. Puneți caracterele pe care doriți să le introduceți cu simbolul % după șir. Codurile de formatare a unui șir în Python:

%s	String
%c	Character
%d	Integer
%f	Floating Pointer
%o	Octal Number
%x	Hexadecimal Number

Putem observa formatarea unui șir în Python prin exemplul următor:

```
print("print string: [%s]" % "test")
print("print string: [%10s]" % "test")          #(1)
print("print character: [%c]" % "t")
print("print character: [%5c]" % "t")           #(2)
print("print Integer: [%d]" % 17)
print("print Float: [%f]" % 17)                 #(3)
print("print Octal: [%o]" % 17)                 #(4)
print("print Hexadecimal: [%x]" % 17)           #(5)
>>>
print string: [test]
print string: [      test]
print character: [t]
print character: [    t]
print Integer: [17]
print Float: [17.000000]
print Octal: [21]
print Hexadecimal: [11]
```

Dacă utilizați codurile de formatare a șirurilor și numerele împreună, caracterele pot fi folosite pentru a asigura un spațiu în funcție de dimensiunea numerelor care sunt printate pe ecran. (1) Imprimarea unui șir de caractere cu lungime fixă: dacă se folosește $\%s$. cu un număr, se asigură spațiul cu o sumă corespunzătoare numarul. În exemplu, test este tipărit folosind 4 cifre și spațiile sunt tipărite pentru celelalte șase cifre, deci toate cele 10 caractere sunt tipărite.
(2) Imprimarea unui caracter fix care conține spații de o anumita Lungime: Dacă $\%c$ este folosit cu un număr, suma corespunzătoare numărului care este aceeasi

ca pentru $\%s$.
Prin urmare, sunt imprimate un caracter și patru spații libere.
(3) Șirul este același cu cel folosit cu numărul $\%c$, care poate fi extras numai ca număr. Din caracterul lui este extras un spațiu liber de 4 cifre.
(3) Număr real: 17 este convertit într-un număr real.
(4) Octal: 17 este convertit într-un număr octal, iar 21 este printat.
(5) Hexazecimal: 17 este convertit într-un număr hexazecimal, iar 11 este printat.

5 Spargerea unei parole cu Python

Spargerea unei parole NU este etică și reprezinta o Infracțiune informatică. Prezentăm doar în scop EDUCAȚIONAL (în Python) acest aspect. În cele ce urmeaza, vom programa in Python pentru a sparge parola unui fișier zip folosind metoda forței brute.

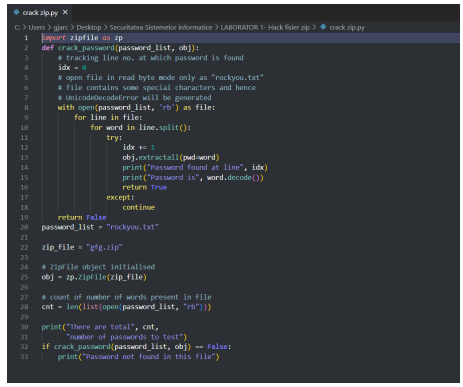
Formatul de fișier ZIP este un standard comun de arhivă și compresie. Este folosit pentru a comprima fișiere. Uneori, fișierele comprimate sunt confidențiale și proprietarul nu dorește să ofere acces fiecărui individ. Prin urmare, fișierul zip este protejat cu o parolă. Dacă parola este obișnuită, atunci este ușor de spart. Aici, vom folosi metoda forței brute pentru a sparge parola fișierului zip.

Cum să forțați parolele fișierelor ZIP în Python?
O metodă de forță brută este o metodă în care un set de valori predefinite sunt folosite pentru a sparge o parolă până la succes. Aceasta este practic o metodă de "loviți și încercați". Această metodă poate dura mult timp dacă setul de valori este mare, dar rata de succes este mare. Cu cât numărul de valori este mai mare, cu atât sunt mai mari șansele de spargere a parolelor. Aici vom folosi fișierul text "rockyou", cu cateva parole de încercat.

Abordare:

Mai întâi, importați modulul zipfile.
Inițializați obiectul ZipFile care ajută la extragerea conținutului fișierului zip.
Numărați numărul de cuvinte prezente în fișierul „rockyou.txt" și afișați-l pe terminal.
Apelați funcția crack password care returnează adevărat dacă este găsită o parolă, altfel returnează false. Treceți numele fișierului text și obiectul ZipFile ca parametri.
Variabila idx este folosită pentru a ține evidența numerelor de linii. Deschideți fișierul text rockyou.txt în modul rb pentru a gestiona conținutul fișierului în formă binară. Acest lucru se datorează faptului că fișierul conține unele simboluri speciale care nu pot fi gestionate dacă fișierul este deschis în modul r și va genera UnicodeDecodeError. După deschiderea fișierului, extrageți linia din fișier și apoi împărțiți cuvântul din acesta. În blocul try, extrageți conținutul fișierului

zip dând parola către câmpul pwd al metodei extractall. Metoda extractall() va extrage tot conținutul fișierului zip în directorul de lucru curent. Programul de mai sus extrage un fișier zip numit gfg.zip în același director cu acest script Python. Dacă parola este incorectă, va fi generată o excepție. În except block, continuați bucla pentru a verifica alte cuvinte din fișier. Dacă parola este găsită, returnează adevărat, altfel returnează false și afișează mesajul dorit.
A se implementa la laborator.



6 Concluzie

În acest laborator, am explicat cateva concepte de baza pentru Python si cum poate fi folosit pentru aplicatii de piratare -ceea ce POATE sa reprezinte o mare vulnerabilitate a PC-urilor.

Nu utilizați aceste informatii pentru a obține acces nedorit la orice computer fără permisiune, deoarece este **ilegal**. Nu utilizați Python pentru infractiuni informatice deoarece este Ilegal!

Pentru bibliografie mai detaliata lecturati capitolul I din cartea de mai jos:

7 Bibliografie

Python Hacking Essentials Paperback 2015 by Earnest Wish,
https://www.amazon.com/Python-Hacking-Essentials-Earnest-Wish/dp/1511797568