But all in all, the secret.txt file only contain a one line of txt.

[17:19, 1/3/2019]

But our task is to Exploit the BoF vulnerablity.

[17:31, 1/3/2019]

Btu Cs Zizo: I did exploit this Vulnerability and submitted the file and the Exploit code together.

Please spend some time understanding the Normal operation of this service.

You should install a third Virtual Machine to help you manga the two given Machines with Mouse.

[17:39, 1/3/2019] Btu Cs Zizo: I mean you will install Another VM and let's say you will give it an IP address 192.168.178.11

[17:39, 1/3/2019] Btu Cs Zizo: And then you should SSH to the attacker Machine by using:

ssh root@192.168.178.12

[17:40, 1/3/2019] Btu Cs Zizo: And now. You can use the attacker machine with the mouse to make every thing is usable and easy.

[17:40, 1/3/2019] Btu Cs Zizo: Then, Instead of running the timeservice on the Sever machine.

[17:41, 1/3/2019] Btu Cs Zizo: Run it on the attacker machine inside the debugger and debug it.

[17:42, 1/3/2019] Btu Cs Zizo: You will not only use this function to exploit.

[17:42, 1/3/2019] Btu Cs Zizo: So use this function and the memcpy function

[17:43, 1/3/2019] +49 1575 1083856: I have a question also. Did you use a null character to escape the if condition?

[17:44, 1/3/2019 you should also use memcpy.

[17:46, 1/3/2019] Btu Cs Zizo: So.

1. Use the Null character /x00 to escape from the If condition.

2. Calculate how many bytes you need to reach the RT address.

Don't forget the Endians.

Example:

A address 0x12345678

Use as a return address will be added to payload as 0x78563412 of \x78\x56\x34\x12

3. And then try to find a way to overcome the ASLR.

[17:47, 1/3/2019] Btu Cs Zizo: Don't use Netcat in delivering your Exploit.

[17:47, 1/3/2019] +49 1575 1083856: so u run the service locally and then test it using gdb?

[17:47, 1/3/2019] Btu Cs Zizo: And for sure you will use the Memcpy to deliver the shell code to the Memory.

[17:49, 1/3/2019] +49 1525 2065468: Zizo, thank you very much for guiding, but how we should properly run timesevice program in attacker machine, because it stucks in recvfrom function,

and doesn't wait for input and doesn't go inside the get\_time function?

[17:51, 1/3/2019] Btu Cs Zizo: 1. Install another VM as I mentioned and give it for example an address

192.168.178.11 with the command.

sudo ifconfig eth0 192.168.178.11 netmask 255.255.255.0

[17:52, 1/3/2019] Btu Cs Zizo: And make sure you set the Network configuration of the machine from virtual box to be Internal network.

[17:52, 1/3/2019] Btu Cs Zizo: 2. And now ssh to the attacker machine. In another Terminal.

**ssh root@192.168.178.12**

[17:53, 1/3/2019] Btu Cs Zizo: 3. When you are in the attacker machine shell, first recompile the timeservice.c application

[17:54, 1/3/2019] Btu Cs Zizo:

**($) gcc-3.4 -g timeservice.c -o timeservice**

[17:54, 1/3/2019] Btu Cs Zizo: 4. Now run it inside the debugger,

Btu Cs Zizo: Do the following.

[17:55, 1/3/2019] Btu Cs Zizo: 1. Make sure that the timeservice is not running by

**($) killall timeservice**

**($) $ gdb /root/timeservice**

[17:57, 1/3/2019] Btu Cs Zizo:

**(gdb) b 131 recvfrom() function**

**(gdb) b 20 //get\_time() function**

**(gdb) run 1234**

[17:58, 1/3/2019] Btu Cs Zizo: If the application still stuck at the recvfrom function

[17:59, 1/3/2019] Btu Cs Zizo: Please reboot the machine where this application is running, in our scenario the atttacker machine.

# Righ ctrl+r

[18:00, 1/3/2019] Btu Cs Zizo: What is the second option?

[18:00, 1/3/2019] Btu Cs Zizo: The second option that you are running it to deliver the exploit it.

[18:01, 1/3/2019] Btu Cs Zizo: So you will not use the netcat to deliver this exploit.

You need to type a simple python script which can open socket with the attacker machine on the port 1234

[18:01, 1/3/2019] Btu Cs Zizo: And instead of making a break point at get\_time only.

[18:01, 1/3/2019] Btu Cs Zizo: U will also make a break point at line 131 which is the recvfrm function.

[18:02, 1/3/2019] Btu Cs Zizo: So by the following.

**($) killall timeservice**

**($) gdb timeservice**

**(gdb) set follow-fork-mode child**

**($) b 131**

**($) b get\_time**

**($) r 1234**

[18:03, 1/3/2019] Btu Cs Zizo: you will find that the application stops at the recvfrom waiting for your packets from another machine.

[18:03, 1/3/2019] Btu Cs Zizo: And here you python script will help.

[18:03, 1/3/2019] Btu Cs Zizo: You will do the following inside the python script.

[18:04, 1/3/2019] Btu Cs Zizo:

1st create a file called bof\_exploit.py for example.

the following inside the script.

Code: nano bof\_exploit.py

import socke #import the socket module

ip='10.0.0.20'

port=1234

my\_client = socket.socket(socket.AF\_INET,socket.SOCK\_DGRAM)

# create an object from the socket class

#AF\_INET Means here you will be using the IPv4

# Dgram means you will be using UDP

#prepare your shell code. wrapping it with proper number of NOPs (/x90) and the return Address.

# you should start this code with the /x00

#To escape from the If condition.

# use the following statement to send this payload.

my\_client.sendto(payload, (ip, port))

#closing this file, save and running the script.

**($) python bof\_exploit.py**

[18:09, 1/3/2019] Btu Cs Zizo: Now you sent these bytes the the recvfrm function which was waiting for you inside the debuger.

[18:09, 1/3/2019] Btu Cs Zizo: Go for the gdb and type

**(gdb) continue**

**(gdb) continue**

[18:10, 1/3/2019] Btu Cs Zizo: It will continue without any problems.

[18:10, 1/3/2019] Btu Cs Zizo: And it will stop at the get\_time function as we made a breakpoint there by ( b get\_time)

Check the memory of the stack frame when the program stops at the breakpoint

**(gdb) info frame**

**(gdb)nexti 19**

[18:11, 1/3/2019] Btu Cs Zizo: Then at this time I need to tell some good notes about the gdb

[18:11, 1/3/2019] Btu Cs Zizo: 1. if you need to run the gdb with only one assembly instruction you should type

[18:11, 1/3/2019] Btu Cs Zizo:

**(gdb) nexti**

[18:11, 1/3/2019] Btu Cs Zizo: And if you need to run it with n instruction

[18:12, 1/3/2019] Btu Cs Zizo: nexti n

[18:12, 1/3/2019] Btu Cs Zizo: And if you need to check the the registers

**(gdb) info registers**

[18:12, 1/3/2019] Btu Cs Zizo: And if you need to check the eip to monitor it's behaviour

**(gdb) info register eip**

**or (gdb) i r eip (for short)**

[18:14, 1/3/2019] Btu Cs Zizo: And most important is if you need to know the local varibales of a function

**(gdb) info locals**

[18:14, 1/3/2019] Btu Cs Zizo: And of the arguments

**(gdb) info arguments**

[18:14, 1/3/2019] Btu Cs Zizo: And if you need to know the address of a specific variable just type the following

**(gdb) print &timebuf**

[18:15, 1/3/2019] Btu Cs Zizo: And now let's get back to the debugger and it was stopped by get\_time

[18:15, 1/3/2019] Btu Cs Zizo: so type

**(gdb) nexti 19**

[18:16, 1/3/2019] Btu Cs Zizo: Exmine the stack by

**(gdb) x/60 $esp**

[18:16, 1/3/2019] Btu Cs Zizo: At this time you will find that your code inside the stack

[18:17, 1/3/2019] Btu Cs Zizo: And now your challenge will move from How to deliver the exploit code

to

How to bypass the ASLR.

[18:17, 1/3/2019] Btu Cs Zizo: ASLR means = Address Space Layout Randomization

[18:18, 1/3/2019] Btu Cs Zizo: This means that every time you will run the application, the stack will take different addresses.

[18:21, 1/3/2019] Btu Cs Zizo: This how i can explain something. :] :D

[22:03, 1/3/2019] Btu Cs Zizo: \_\_\_\_\_\_\_

One side note)

- Don's use NetCat to deliver the Shell code as it will not interpret /x00.

- But if you need it in any other tasks.

Use it. Cause it will automate much work for u.

[22:05, 1/3/2019] Btu CS Kit:

**($) Printf “\x00”| nc -u IP port**

[22:06, 1/3/2019] Btu CS Kit: It works using with printf

[22:07, 1/3/2019] Btu Cs Zizo: OK. I mean. Netcat alone without any other techniques it will not work.

And at the same time we should use it, as it will automate much work.

[23:03, 1/3/2019] Btu Cs Zizo: Bash scripting is painful.

This will be clear if you tried to print for example "A" 90 times.

In python it is just: print "A" \* 90

But in Bash scripting you have to follow techniques like this:

https://stackoverflow.com/questions/5799303/print-a-character-repeatedly-in-bash

[23:04, 1/3/2019] Btu Cs Zizo: So, as a point of view.

You can invest two hours to know python basics from here.

https://www.youtube.com/watch?v=YYXdXT2l-Gg&list=PL-osiE80TeTskrapNbzXhwoFUiLCjGgY7

[10:13, 1/4/2019] Btu Cs Zizo: If you found that the debugger is behaving in a strange way like stucking at Recvfrm function or at any location.

Do the following.

1. Make sure you killed any other instances of the timeservice application by

**($) killall timeservice**

**($) ps aux | grep timeservice**

**($) gdb timeservice**

2. After running it inside the debugger, make sure you asked the debugger to follow the child of this application because the C code contains the fork function.

**(gdb) set follow-fork-mode child**

And to make sure that the debugger understands this

**(gdb) show follow-fork-mode**

It must give u a line as a result have the child at the end.

3. If you did all there steps, and still the debugger stuck at specific location, u should

Reset this Machine

If you need to reset a machine in Vbox in a fast way.

Just press

Right CTRL + r

[10:18, 1/4/2019] Btu Cs Zizo: In the point no. 2: You should execute these commands inside the gdb prompt (gdp)