CS620 Advanced Computer Networks

Lab 1

Course Mentor: Snehal Shetty
TA: Seshagiri Prabhu

Amrita Center for Cyber Security
Amritapuri

July 26, 2013

Socket Programming in C

Contact info

Snehal Shetty

Office

103

email

snehal@am.amrita.edu

Seshagiri Prabhu

Office

110

email

seshagiriprabhu@am.amrita.edu

Outline

1 Socket

2 Problem description

3 Problem Dissection

4 Inference

Sockets

Sockets are a protocol independent method of creating a connection between processes. Sockets can be either

- connection based or connectionless: Is a connection established before communication or does each packet describe the destination?
- packet based or streams based: Are there message boundaries or is it one stream?
- reliable or unreliable. Can messages be lost, duplicated, reordered, or corrupted?

Socket Characteristics

Socket are characterized by their domain, type and transport protocol. Common domains are:

- `AF_UNIX`: address format is UNIX pathname
- `AF_INET`: address format is host and port number

Common types are:

- Virtual Circuits: received in order transmitted and reliably
- datagram: arbitrary order, unreliable

Socket Characteristics (cont'd)

Each socket type has one or more protocols. Ex:

- TCP/IP (virtual circuits)
- UDP (datagram)

Use of sockets:

- Connection–based sockets communicate client-server: the server waits for a connection from the client.
- Connectionless sockets are peer-to-peer: each process is symmetric.

Socket APIs

- `socket`: creates a socket of a given domain, type, protocol (buy a phone)
- `listen`: assigns a name to the socket (get a telephone number)
- `accept`: specifies the number of pending connections that can be queued for a server socket. (call waiting allowance)
- `connect`: client requests a connection request to a server (call)
- `send, sendto`: write to connection (speak)
- `recv, recvfrom`: read from connection (listen)
- `shutdown`: end the call

Problem Description
Compile the program


```
1  # Make file
2  CC=gcc
3  CFLAG= -c -Wall
4
5  EXECS := circle
6  PROG  := circle.c
7  OBJS  := $(addprefix OBJECTS/, $(addsuffix .o, $(EXECS)))
8
9  all      :$(EXECS)
10
11 OBJECTS :
12       mkdir -p $@
13
14 $(OBJS) :| OBJECTS
15
16 $(EXECS):$(OBJS)
17              $(CC) $^ -o $@
18
19 OBJECTS/circle.o :$(PROG)
20                  $(CC) $(CFLAG) $^ -o $@
21
22 clean    :
23       rm -rf $(EXECS) OBJECTS
24
25 .PHONY   : clean all
```

Problem Description
Compile and run the program

Compile the program

```
seshagiri@ACCS:~$ make
mkdir -p OBJECTS
gcc -c -Wall circle.c -o OBJECTS/circle.o
gcc OBJECTS/circle.o -o circle
```

Execute the binary file - *circle*

```
seshagiri@ACCS:~$ ./circle
0.250
```

Problem Description

What can we understand after executing the binary file?

What can we understand after executing the binary file?

$< burp >$ NOTHING! $< /burp >$

Problem Description
What should we do next?

What should we do next?

Inspect the source code

Problem Description

Intended the code properly for better understanding


```
1 #include <stdio.h>
2 #define _ -F<00||--F-OO--;
3
4 int F=00,OO=00;
5
6 main() {
7         F_OO();
8         printf("%1.3f\n", 4.*-F/OO/OO);
9 }
10
11 F_OO() {
12               _-_-_-_
13           _-_-_-_-_-_-_-_-_
14       _-_-_-_-_-_-_-_-_-_-_-_-_
15     _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
16   _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
17   _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
18 _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
19 _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
20 _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
21 _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
22   _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
23   _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
24     _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
25       _-_-_-_-_-_-_-_-_-_-_-_-_-_-_
26           _-_-_-_-_-_-_-_-_-_
27               _-_-_-_-_
28 }
```

Problem Dissection

Macro... Macro..

Unnoticed macro _

```
#define _ -F<00||--F-OO--;
```

What macro does?

- Replaces '_' with `'-F<00||--F-OO--;'`
- for eg: `'_-_-_'` becomes `'-F<00||--F-OO--;- -F<00||--F-OO--;- -F<00||--F-OO--;'`
- One of the important property of `||` operator is that if the first condition is true, it will NOT check for the second condition unlike `&&` operator.

Problem Dissection

Values of variables inside F_OO()

Values of F and OO

- F and OO are initially set as *zero*.
- In the first line of F_OO function, -F<00 becomes false hence checks for the second condition.
- --F becomes -1 and OO-- remains as 0.
- After the execution of the first line, value of F becomes -1 and OO becomes -1.

Problem Dissection

Values of F and OO inside F_OO()

- **Line 1:** texttt-0¡0 (false) —— (-1)-(0–) (executed); -1¡0 (true) —— (-2)-(-1)(not executed); -1 ¡ 0 (true) —— (-3)-(-2)(not executed). `F = -1, OO = -1`
- **Line 2:** `1<0 (false) || (-2)-(1--) (executed); -2<0 (true) || (-3)-(-2)(not executed); -3 < 0 (true) || (-4)-(-3)(not executed) -4<0 (true) || (-5)-(-4)(not executed); -5 < 0 (true) || (-6)-(-5)(not executed).` `F = -2, OO = -2`

  .

  .

  .

- **Line 16:** `F = -16, OO = -16`

Problem Dissection

Lets go back to `main()` where `F_OO()` was called

Inside `main()`

```
printf("%1.3f\n",4.*-F/OO/OO);
```

This line calculates the value 0.250

Problem Dissection

What was the learning from this program?

What was the learning by inspecting the code

NIL

Problem Dissection
What should be done next?

What should be done next?
Lets

Problem Dissection
What should be done next?

Bingo! Wiki rocks

http://en.wikipedia.org/wiki/International_Obfuscated_C_Code_Contest#Examples

The same example is mentioned in the above mentioned link and it says that the program works by calculating its own area and diameter, and then doing a division to approximate pi!!

They call it as an Obfuscated C code

Problem Dissection

How does it calculates the value of Π?

Lets do reverse engineering!

Using the hint in Wiki, we found that program works by calculating its own area and diameter. So we need to make the Π, Area, radius/diameter looks like `4.*-F/OO/OO.`

So lets do reverse Engineering!

Problem Dissection
Calculating the value of Π

Calculating the value of Π from Area and radius/diameter

Area = Π * $radius^2$

Π = Area / $radius^2$

As we need 4 in the numerator we may have to substitute radius with diameter.

Π = Area / $(diameter/2)^2$

Π = 4 * Area / (diameter * diameter)

Π = 4 * Area / diameter / diameter

Inference

What can we conclude from this?

Conclusion

Variable F is the area of the circle used in the program
The Diameter of the circle is the variable OO

Learning
What's new I have learned from this?

Learning something new

Previously, I have tried to de-obfuscating PHP code (in which they used base64 encoding, rot 13 transformation etc) files during Capture The Flag contests. This is the first time I am hearing about C code obfuscation. That was a nice learning experience!

Thank you!

Seshagiri Prabhu
seshagiriprabhu@gmail.com
AM.EN.P2CSN12028

Assignment repo