VE482 Lab Notes

Yihao LIU, Hang SU, Yue YAO

VE482 AU18 TA Group

October 23, 2018

Source code available at https://github.com/tonyfloatersu/VE482-AU2018-LAB-NOTE

Table Of Contents I

RC Week 7

Something about compilation

What if we cannot distribute the source code?

Examples of Absolute Dumb Solution

Obfuscation Nice try, but far from enough. A lot of code formatters are on their way.

Encryption Decipher takes time, safety is not guaranteed, become useless if someone successfully cracked it.

Blabla ... Don't really figure out any other dumb solutions.

Good News: We don't have naïve solutions.
Bad News: We don't have solutions. (Just joking, of course we have.)

Some example of obfuscation

```
                                           #define r(R) R"()"
                          /*[*/#include  /**/<stdio.h>
                      #include<math.h>/*!![crc=0f527cd2]*/
                   float I,bu,k,i,F,u,U,K,O;char o[5200];int
              #define R(U) (sizeof('U')==1||sizeof(U"1"[0])==1)
            h=0,t=-1,m=80,n=26,d,g,p=0,q=0,v=0,y=112,x=40;  float
           N(float/*x*/_){g=1<<30;d=-~d*1103515245&--g;return  d*_
          /g;}void/**/w(int/**/_){if(t<0){for(g=0;g<5200;o[g++    ]=
          0);for(;g;o[g+79]=10)g-=80;for(t=37;g<62;o[80+g++]=32)   ;
         }if(m&&o[h*80+m-1]==10){for(g=0;g<79;o[t*80+g++]=0){}o[t
         ++*80+g]=10;t%=64;n+=2;I=N(70)+5;if(n>30&&(I-x)*(I-x)+n*
        n>1600&&R()){O=0;F=(x=0x1!=sizeof(' '))?k=1+N(2),i=12-k+N(
        8),N(4):(k=17+N(5),i=0,r()[0]?O=.1:  0);for(u=U=-.05;u<32;
        U=k+i+i*.5*sin((u+=.05)+F))for( K=0   ;K< U;K+=.1)if((bu=K*
       sin(u/5),g=I+cos( u/5)*K)>=0&&g   <     79  )o[g+(int)(t+44+
       bu*(.5-(bu>0?3*O:  O)   ) )%64*  80      ]  =32;x*=02//* */2
      -1;n=O+x?n=I+(x?0   :N     (k)-   k           /2),g=(t+42  )%
      64,m=-~g%64,x?g=m          =-~        m%64:0  ,n>5?o[g*80   +
     n-3]=o[m*80+n-3]=       0:   0              ,n <75?o[g*80+n
     +2]=o[m*80+n+2]=0   :0:0;                      x=I;}h=-~h%64
    ;m=0;}putchar((g=o [h*                          80+m++])?g:_);
   if(g){w(_);}}void W                               (const char*_
  ){for(;*_;w(*_++));}                               int main(int a
  ,char**_){while(a--)d              +=_[a          ]-(char*)0;W( \
 "#include<stdio.h>typed"             "e"         "f\40int\40O;v"
 "oid o(O _){putchar(_);}O"                    "\40main(){O"  ""
"*_[512],**p=_,**d,b,q;for(b=0;b"      "++<512;p=_+q)_[q"    \
"=(p-_+1)*9%512]=(O*)p;") ;      for(;(g= getchar())-EOF;p=
q){q=p;for(v=512;p-q-g&&q-p-              g;  v--)q=-~q*9%512
;W("o(");if(p>q)w(y),w(45);w(                      40);w(y^=20
);w(075);for(a=0;a<v;a++)w(42);                      for(W("(O**"
 );a--;w(42)){}w(41);w(y^024);w(                      41);if(p<=q)w(
   45),w(y^20);W(");");}for(a=7;a-6                      ;W(a<6?"{;}":""
      ))for(a  =0;a  <6 &&   !o[h*80+m                       +a];a++){}W("r"
         "etu"  /*J   */       "rn+0;}\n"                             );return
             /*                      "#*/0                                   ;}
```

Source: https://www.ioccc.org/2015/yang/prog.c

Library is a current solution

Definition

A pack of non-volatile code meant to be reused by the program. For C or C++ library, it comes in two parts:

Function Interface A header file that exposes functionality of the library to whoever / whatever uses it.

Implementation Pre-compiled binary file that contains machine code for source code implementation.

Tools Related

- Compiler
- Linker

Static / Dynamic

There are 2 types of library: static library and dynamic library.
The biggest difference lies in the runtime of the target exec file.

Static Lib: The machine code is applied to target exec file, so the exec file do not need these static lib file in runtime.

Dynamic Lib: Target exec file still requires dynamic lib.

The different behavior lies in a fact that dynamic lib implementation is PIC, position-independent code.
It can be executed / called regardless of absolute address in RAM.

Static / Dynamic

Make such libraries

(For convenience and WLOG, we use c++ and g++ to illustrate.)

Static Lib

Build lib

```
g++ -c dumb.c
ar rcs libdumb.a dumb.o
```

Make exec

```
g++ -o dumb main.c -L./ -ldumb
```

Dynamic Lib

Build lib

```
g++ dumb.c -fPIC -shared -o libdumb.so
```

Make exec

```
g++ -o dumb main.c -L./ -ldumb
```

Experiment: size of executable file

Some knowledge you need to know

Standalone Exec File The executable that doesn't require any external module / library / program. It is boot itself (bootstrap) and runs on bare ground of OS.

Lib dependency detect `ldd` can find shared libraries of exec file.

Lib linking order The order of lib linking should make a DAG directing the target source file. If inter-dependent situation happens, some fixes are needed.

Experiment: size of executable file

Standalone Exec file

For c++, we use the following commandline parameters to make a standalone exec file:

```
-static -static-libgcc -static-libstdc++
```

These parameters guarantee that all the static version of dependencies are linked to the exec file.

Experiment: size of executable file

Lib dependency detect

An example of `ldd` usage:

```
sh-4.4$ ldd /bin/bash
linux-vdso.so.1 (0x00007ffcec7d9000)
libreadline.so.7 => /usr/lib/libreadline.so.7 (0x00007f003e00b000)
libdl.so.2 => /usr/lib/libdl.so.2 (0x00007f003e006000)
libc.so.6 => /usr/lib/libc.so.6 (0x00007f003de42000)
libncursesw.so.6 => /usr/lib/libncursesw.so.6 (0x00007f003dbd5000)
/lib64/ld-linux-x86-64.so.2 => /usr/lib64/ld-linux-x86-64.so.2 (0x00007f003e584000)
```

Dependencies among dynamic libs exist. Here's another example:

```
sh-4.4$ readelf -d /usr/lib/libreadline.so.7 | grep 'NEEDED'
 0x0000000000000001 (NEEDED)             Shared library: [libncursesw.so.6]
 0x0000000000000001 (NEEDED)             Shared library: [libc.so.6]
sh-4.4$ ldd /usr/lib/libreadline.so.7
        linux-vdso.so.1 (0x00007ffef8ce7000)
        libncursesw.so.6 => /usr/lib/libncursesw.so.6 (0x00007fab7b330000)
        libc.so.6 => /usr/lib/libc.so.6 (0x00007fab7b16c000)
        /usr/lib64/ld-linux-x86-64.so.2 (0x00007fab7b835000)
```

Related commands like `objdump` can also be used.
Better RTFM and use `man`.

Experiment: size of executable file

Lib linking order (static acyclic)

```
sh-4.4$ cat dumb1.cpp
int var_0 = 19;
sh-4.4$ cat dumb2.cpp
extern int var_0;
int soudayo() {
        return var_0;
}
sh-4.4$ cat dumb_main.cpp
extern int soudayo();
int main() {
        return soudayo();
}
sh-4.4$ g++ -c dumb1.cpp -o dumb1.o
sh-4.4$ g++ -c dumb2.cpp -o dumb2.o
sh-4.4$ ar rcs libdumb1.a dumb1.o
sh-4.4$ ar rcs libdumb2.a dumb2.o
sh-4.4$ g++ -L. -ldumb1 -ldumb2 dumb_main.cpp -o a.out
/usr/bin/ld: cannot find -ldumb2
collect2: error: ld returned 1 exit status
sh-4.4$ g++ -L. -ldumb2 -ldumb1 dumb_main.cpp -o a.out
/usr/bin/ld: cannot find -ldumb1
collect2: error: ld returned 1 exit status
sh-4.4$ g++ dumb_main.cpp -L. -ldumb2 -ldumb1
sh-4.4$ ./a.out
sh-4.4$ echo $?
19
```

```
+----------------------+
| +-----------+        |
| | dumb_main | <---+  |
| +-----------+     |  |
|       +-dependent-+  |
|       +              |
| +------------+       |
| | dumb2      | <--+  |
| +------------+    |  |
|       +-dependent-+  |
|       +              |
| +------------+       |
| | dumb1      |       |
| +------------+       |
+----------------------+
```

Experiment: size of executable file

Lib linking order (dynamic acyclic)

```
sh-4.4$ g++ -fPIC -shared dumb1.cpp -o libdumb1.so
sh-4.4$ g++ -fPIC -shared dumb2.cpp -L. -ldumb1 -o libdumb2.so
sh-4.4$ export LD_LIBRARY_PATH=.:$LD_LIBRARY_PATH
sh-4.4$ g++ -Wl,--as-needed dumb_main.cpp -L. -ldumb2
sh-4.4$ ./a.out
sh-4.4$ echo $?
19
```

It can be noticed that LD_LIBRARY_PATH has been modified with current dir included, so dependency is automatically established.

```
+-------------------------------------------+
| +-----------+     +-------+     +-------+ |
| | dumb_main |<-+--| dumb2 |<-+--| dumb1 | |
| +-----------+  |  +-------+  |  +-------+ |
|                |             +            |
|                +- dynamically dependent   |
+-------------------------------------------+
```

Experiment: size of executable file

Lib linking order (circular dependency)

```
  libfo.a         libbar.a
+-----------+   +-----------+
| +-------+ |   | +-------+ |
| | fo1.o +------>| bar.a | |
| |       |<------+       | |
| +-------+ |   | +---+---+ |
|           |   |     |     |
|           |   +-----|-----+
| +-------+ |         |
| | fo2.o |<---------+
| +-------+ |
+-----------+
```

The solution for static circular dependency is that:

`gcc main.c -L. -lfo -lbar -lfo`

Experiment: size of executable file

Experiment and Observes

```cpp
#include <iostream>

int main() {
        std::cout << "Hello, your name? " << std::endl;
        std::string temp; std::cin >> temp;
        std::cout << "Hello! " + temp << std::endl;
        return 0;
}
```

Then we get the following result:

```
sh-4.4$ g++ dumb.cpp -o dynamic_dumb
sh-4.4$ ldd ./dynamic_dumb
        linux-vdso.so.1 (0x00007fffa19a6000)
        libstdc++.so.6 => /usr/lib/libstdc++.so.6 (0x00007f0c8b64a000)
        libm.so.6 => /usr/lib/libm.so.6 (0x00007f0c8b4c5000)
        libgcc_s.so.1 => /usr/lib/libgcc_s.so.1 (0x00007f0c8b4ab000)
        libc.so.6 => /usr/lib/libc.so.6 (0x00007f0c8b2e7000)
        /lib64/ld-linux-x86-64.so.2 => /usr/lib64/ld-linux-x86-64.so.2 (0x00007f0c8b829000)
sh-4.4$ g++ -static -static-libgcc -static-libstdc++ dumb.cpp -o standalone_dumb
sh-4.4$ ldd ./standalone_dumb
        not a dynamic executable
sh-4.4$ ls -al | grep -E '(dynamic|stand)'
-rwxr-xr-x 1 anthonysu anthonysu   18712 Oct 22 19:19 dynamic_dumb
-rwxr-xr-x 1 anthonysu anthonysu 2181936 Oct 22 19:19 standalone_dumb
```

Experiment: size of executable file

Analysis

Dynamic Linking

- Size: 18172
- `readelf -d ./dynamic_dumb | grep 'NEEDED' | wc -l` is 4. Indicating 4 shared libraries are used.

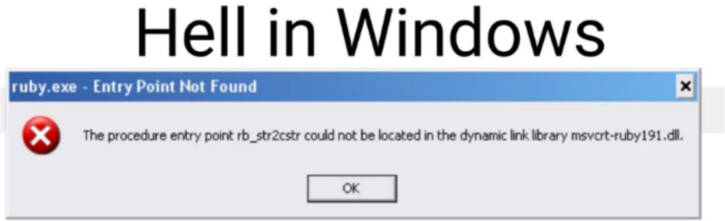
Static Standalone

- Size: 2181936
- No shared library.

Dependency Hell

Dll Hell (Windows)

Problem arises when dll file version conflict between PC and program requirement. Dll file do not have the ability of backward compatibility, so minor changes in dll render internal structure different from previous version.



ruby.exe - Entry Point Not Found

The procedure entry point rb_str2cstr could not be located in the dynamic link library msvcrt-ruby191.dll.

some meme

Always be careful when creating libs!

I'VE SEEN THINGS YOU PEOPLE WOULDN'T BELIEVE
ALL THESE CODES WILL BE LOST IN LIB, LIKE MEM IN RAM
TIME TO COMPILE !
imgflip.com

You can (not) redo.

RC Week 7

OK, what is multi-thread?

Threads in Process

RC Week 7

Strategies about Multi-threading

RC Week 7

C++ Programming about Multi-threading