Reverse engineering или как гадать по байтам

$ whoami

- Co-founder of R0Crew(reverse4you.org)
- Co-founder of BSides Kyiv
- Co-founder of DC8044(Kyiv)
- CTO at PushRet
- Malware analyst
- Security researcher
- Sofa councilor

“Возможно, все мы в детстве сломали немало игрушек пытаясь понять «как оно там устроено». С возрастом это проходит. Но не у всех.”

– lurkmore

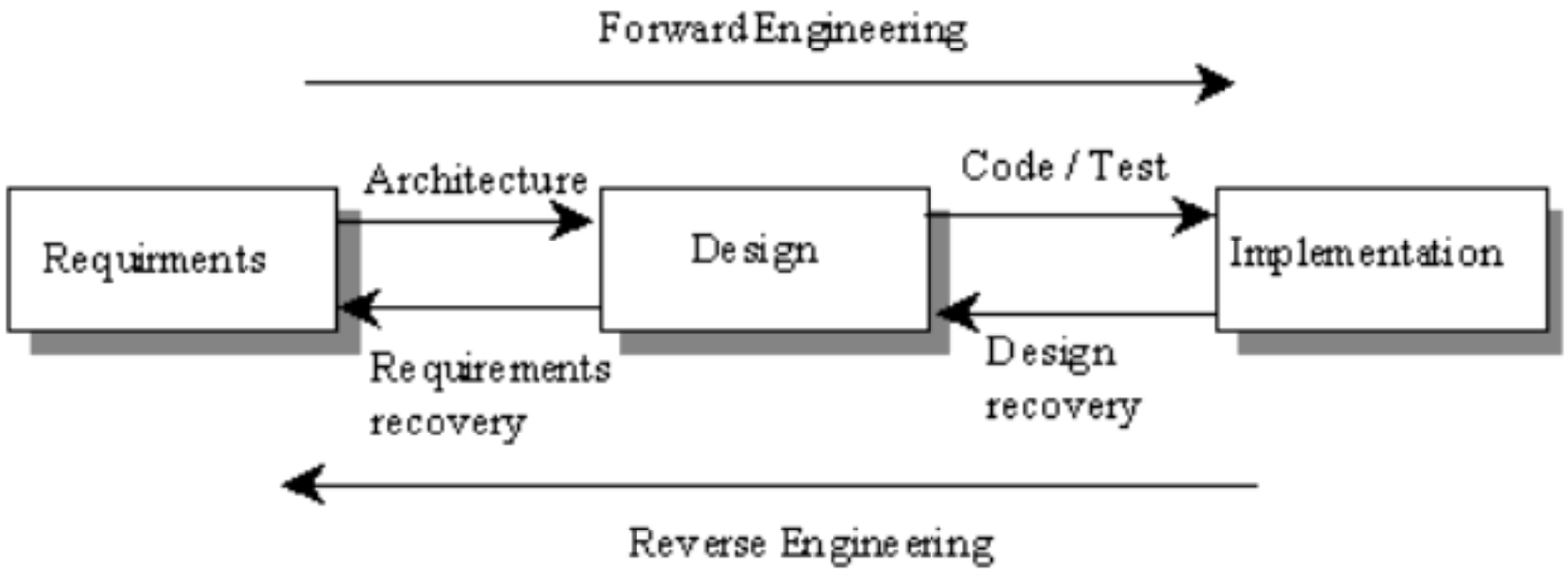
Выглядит это все примерно так

```c
#include <stdio.h>

int main(int argc, const char** argv) {
        printf("Hello world!\n");
        return 0;
}
~
~
```

```
; int __cdecl main(int argc, const char **argv, const char **envp)
                public _main
_main           proc near

var_14          = dword ptr -14h
var_10          = qword ptr -10h
var_8           = dword ptr -8
var_4           = dword ptr -4

                push    rbp
                mov     rbp, rsp
                sub     rsp, 20h
                lea     rax, aHelloWorld ; "Hello world!\n"
                mov     [rbp+var_4], 0
                mov     [rbp+var_8], edi
                mov     [rbp+var_10], rsi
                mov     rdi, rax        ; char *
                mov     al, 0
                call    _printf
                xor     ecx, ecx
                mov     [rbp+var_14], eax
                mov     eax, ecx
                add     rsp, 20h
                pop     rbp
                retn
_main           endp
```

А вот так еще лучше

```
; int __cdecl main(int argc, const char **argv, const char **envp)
                public _main
_main           proc near

var_14          = dword ptr -14h
var_10          = qword ptr -10h
var_8           = dword ptr -8
var_4           = dword ptr -4

55                              push    rbp
48 89 E5                        mov     rbp, rsp
48 83 EC 20                     sub     rsp, 20h
48 8D 05 47 00 00 00            lea     rax, aHelloWorld ; "Hello world!\n"
C7 45 FC 00 00 00 00            mov     [rbp+var_4], 0
89 7D F8                        mov     [rbp+var_8], edi
48 89 75 F0                     mov     [rbp+var_10], rsi
48 89 C7                        mov     rdi, rax        ; char *
B0 00                           mov     al, 0
E8 0D 00 00 00                  call    _printf
31 C9                           xor     ecx, ecx
89 45 EC                        mov     [rbp+var_14], eax
89 C8                           mov     eax, ecx
48 83 C4 20                     add     rsp, 20h
5D                              pop     rbp
C3                              retn
                _main           endp
```


Forward Engineering
Requirments
Architecture
Design
Code / Test
Implementation
Requirements recovery
Design recovery
Reverse Engineering

Процесс

Работать приходится вот так

Что из этого получается?

И даже такое...

Сложности

- Это не C++ за 21 день не выучишь
- Это не Java за 3 месяца не выучишь
- Это хардкорно

С чего начать

- Архитектура процессора
- Ассемблер
- OS Internals
- Организация памяти

Must read

- RE for beginners
- Kris Kaspersky books
- reverse4you.org
- exelab.ru
- wasm.in
- tuts4you.org

Что использовать

- x64dbg/OllyDbg/Immunity Debugger/gdb/lldb
- IDA Pro/Hopper/radare2
- 010 Editor/winhex
- Burp/mimtproxy/fiddler/wireshark

Плюсы

- Бесплатный софт
- Рут на сервере микрософта
- Немного денег на багбаунти
- Много денег на эксплойтах
- Деньги на пиво с лайков в социалочках

Из минусов

- Вы можете надолго пропасть из жизни
- Вас могут побаиваться и не понимать
- За вами могут приехать

Варианты работы

- Malware analyst
- Security researcher
- Freelancer
- Cracker
- Research anything for blackhats

To be continue...

Contacts

- tg/twitter: @ximerus

- fb/vk/linkedin: ximerus