Unpacking: od sztuki do rzemiosła

Secure 2015

CERT.PL

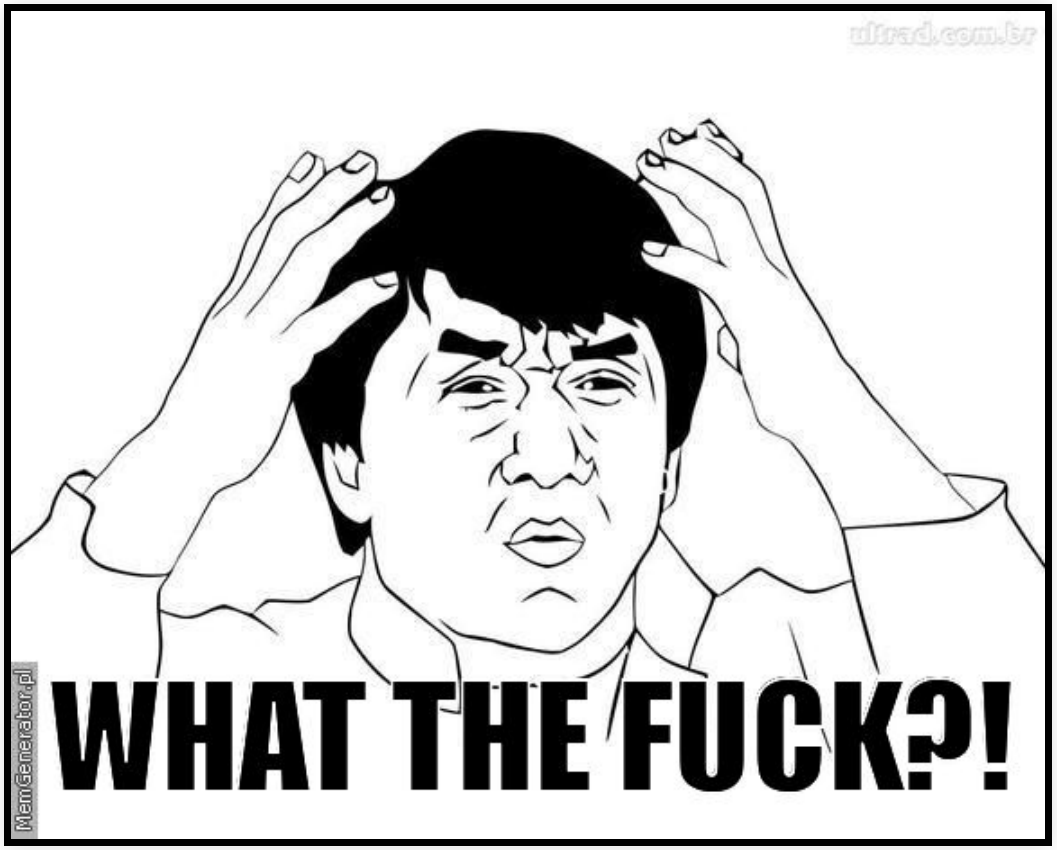
Warszawa

@maciekkotowicz

$ whoami

- IRT@CERT.pl
- DragonSector CTF
- RE/Exploit dev
- Formal methods

(Un)packery -WTF

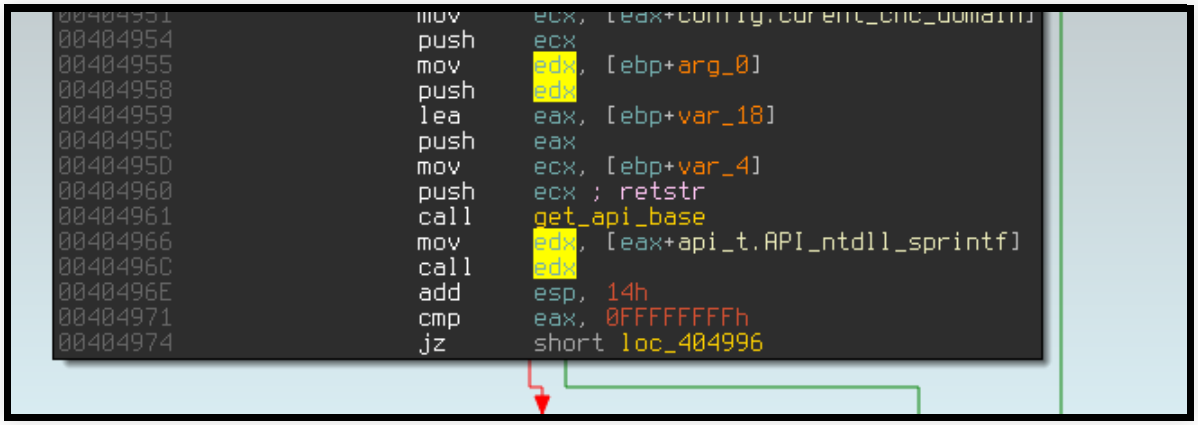


Pakowanie - Wprowadzenie

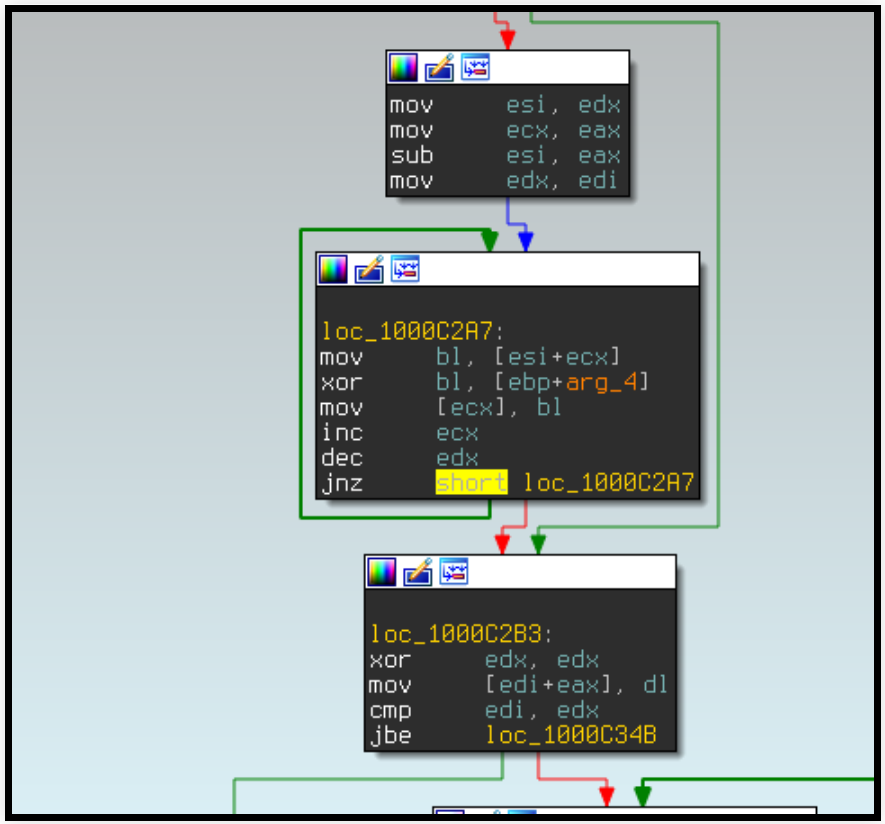
- Obfuskacja
- Statyczne kodowanie
- Silniki mutujace
- Wirtualizacja

Obfuskacja

- calls
- api-calls
- strings
- junk-code/indirection

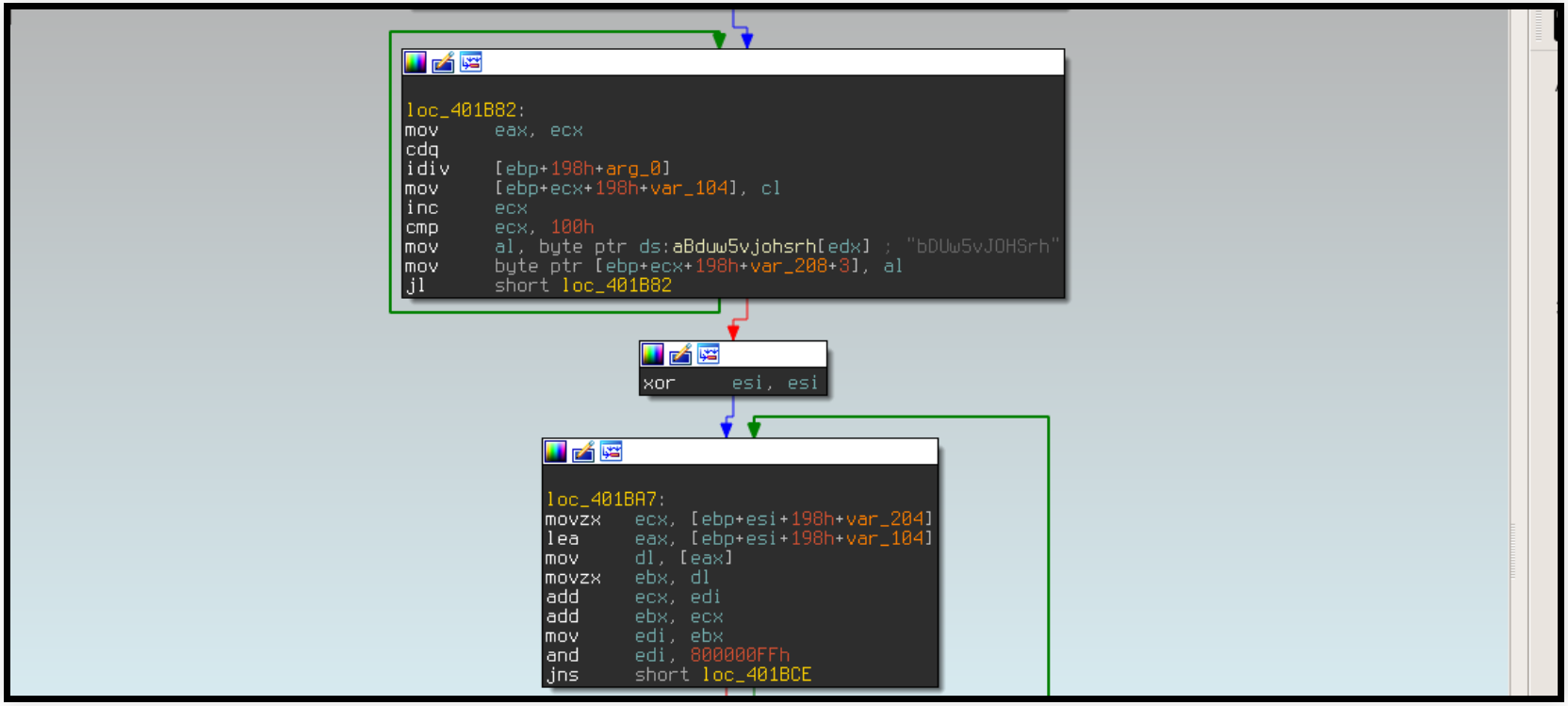



5C84 push ebp
5C85 mov ebp, esp
5C87 push eax
5C88 mov eax, [ebp+4]
5C8B mov [ebp+arg_8], eax
5C8E mov eax, [ebp+arg_4]
5C91 add eax, [ebp+arg_0]
5C94 jmp loc_B5D9
5C94 sub_15C84 endp
5C99 ; ---
5C99
5C99 loc_15C99: ; CODE XREF: seg000:00008DE4↑j
5C99 add [ebp+4], eax
5C9C pop eax
5C9D leave
5C9E retn 8
5CA1 ; ---


mov esi, edx
mov ecx, eax
sub esi, eax
mov edx, edi
loc_1000C2A7:
mov bl, [esi+ecx]
xor bl, [ebp+arg_4]
mov [ecx], bl
inc ecx
dec edx
jnz short loc_1000C2A7
loc_1000C2B3:
xor edx, edx
mov [edi+eax], dl
cmp edi, edx
jbe loc_1000C34B

Statyczne kodownie

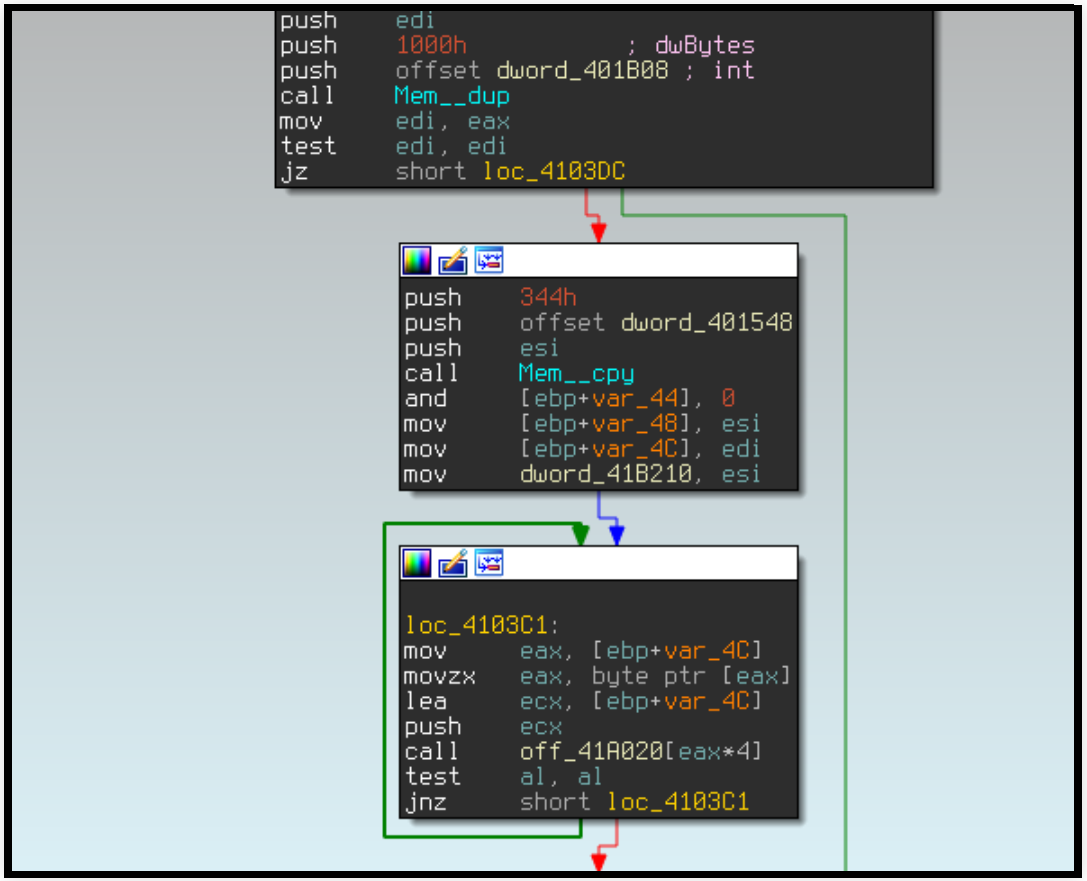
- kompresja
- szyfrowanie
- xor/base64/hex/etc



Silniki mutujace

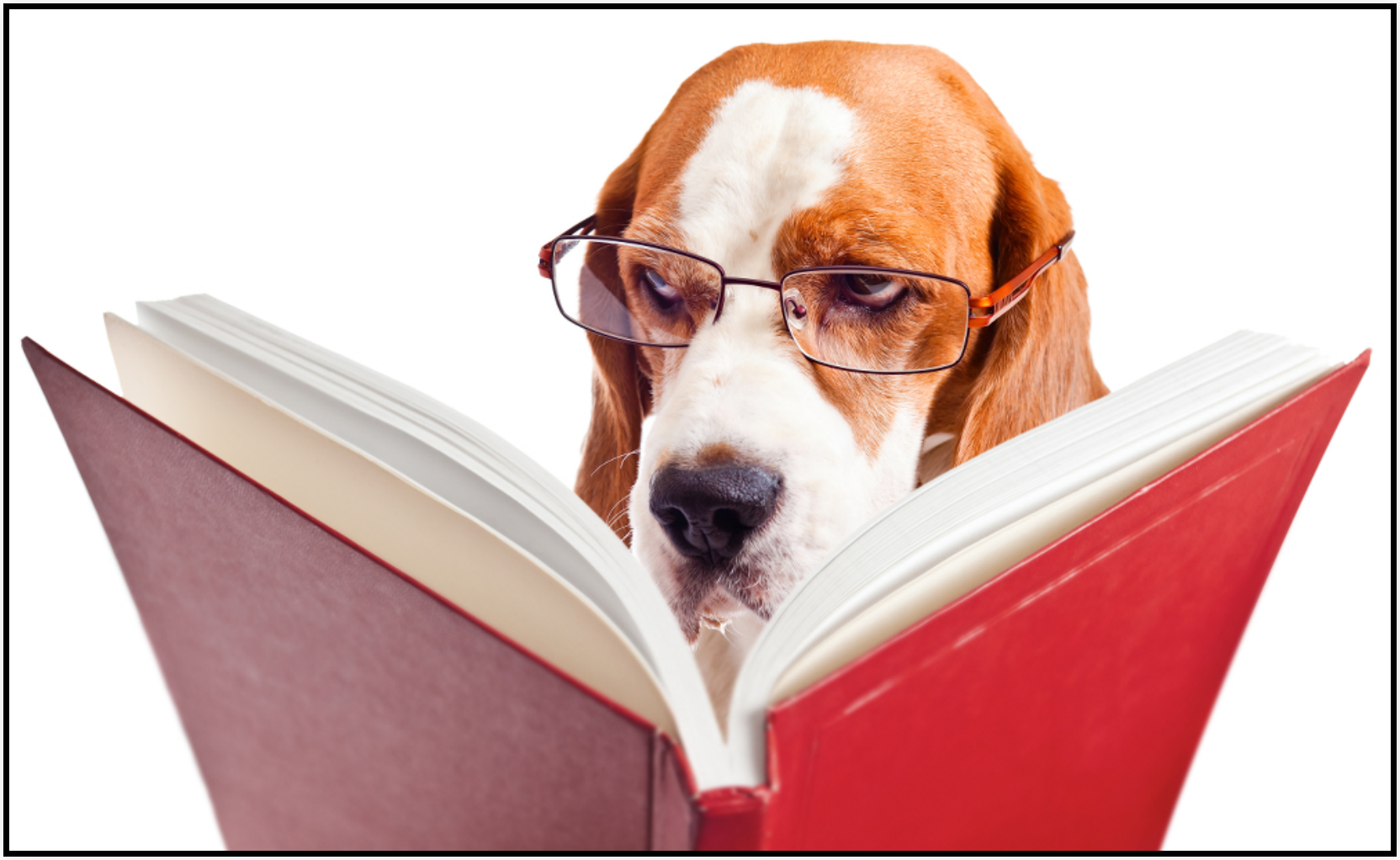
- junk-code
- zmiana rejstrów
- zmiana kolejności operacji
- losowy porządek zmiennych
- wieele innych

Wirtualizacja

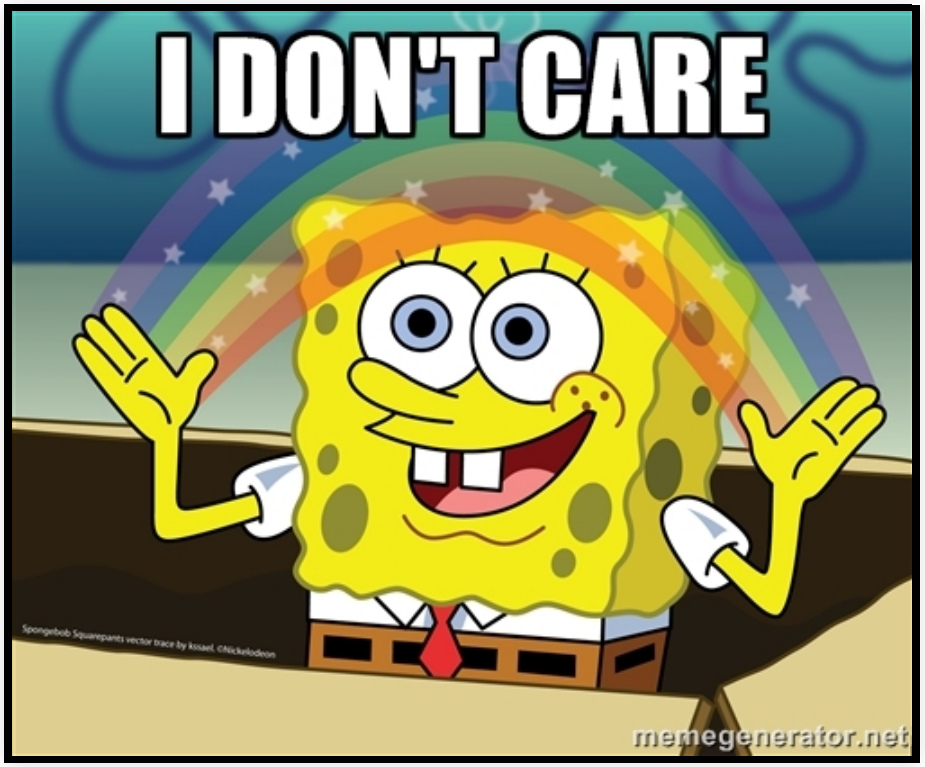


Smutne życie

Teoria



Obfuskacja/Szyfrownie/etc



Sledzenie kodu

Śledzenie kodu

Śledzenie kodu

- Wstrzykiwanie
- shellcode
- RunPe

Wstrzykiwanie kodu

- OpenProcess
- VirtualAlloc
- WriteProcessMemory / LoadLibrary
- CreateRemoteThread / QueueUserAPC

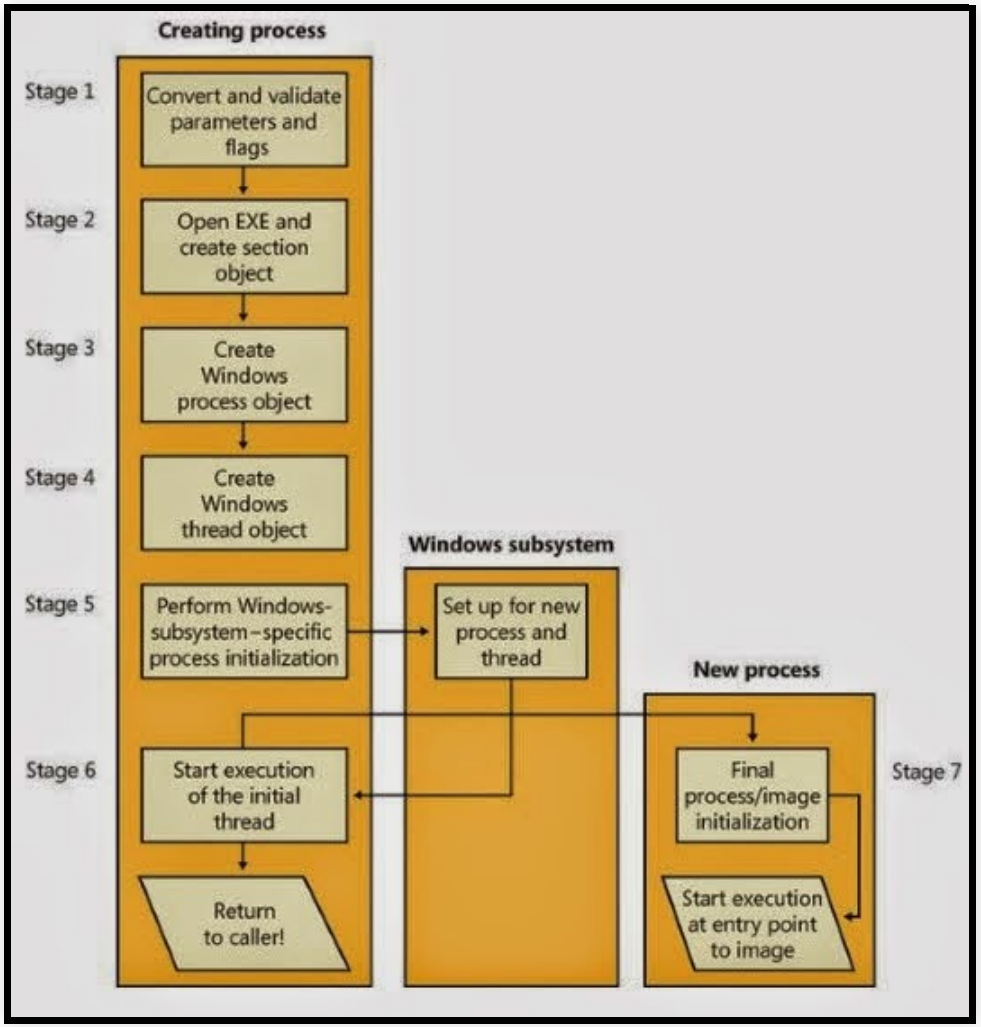
Wstrzykiwanie kodu

- OpenProcess
- SupendThread
- MapViewOfSection
- VirtualProtect
- SetThreadContext
- ResumeThread

Wstrzykiwanie kodu

- CreateSection // explorer.exe shared memory
- MapViewOfSection
- FindWindow("Shell_TrayWnd")
- SetWindowLong
- SendNotifyMessage

RunPE



- DecodeStage
- CreateProcess*
- InjectCode
- Exit

Praktyka

- debugger
- hook-dll
- DBI
- hypervisor

Skoczek

```
    def post_NtResumeThread(self,ev,retval):
        tHandle,_ = self.get_funcargs(ev)
        pid = self.pid_from_handle(tHandle)
        self.log(ev,'called[post] NtResumeThread(%d) - %s '%(tHandle,pid))
        if hasattr(self,'pid') and self.pid:
#           pname = self.proc_name(pid)
            self.log(ev,'jumping to %d'%self.pid)
            ev.debug.attach(self.pid)
```

```
    def post_CreateProcessInternalW(self,ev,retval):
        args  = self.get_funcargs(ev)
        proc = ev.get_process()
        name = proc.peek_string( args[1], fUnicode = True )
        cmdline = proc.peek_string( args[2], fUnicode=True)
        pi = proc.read_structure(args[-2], PROCESS_INFORMATION )

        self.phandles[pi.hProcess] = {'tid': pi.dwThreadId,'pid': pi.dwProce
        self.phandles[pi.hThread] = {'tid': pi.dwThreadId,'pid': pi.dwProces
        if args[6] & CREATE_SUSPENDED:
            self.log(ev,'called[post][SUSPENDED] CreateProcessInternalW(%s,%
            self.pid = pi.dwProcessId
        else:
            self.log(ev,'called[post][%d] CreateProcessInternalW(%s,%s) - %d
            ev.debug.attach(pi.dwProcessId)
```

Śledzenie pamieci

```
def page_break(self,ev,proc,rv,ba,size,prot):
    if prot == PAGE_EXECUTE_READWRITE:
            baddr =self.get_guard_addr(ba)
            try:
                ## unwatch whole thing, just use small ones
                if baddr :
                   proc.mprotect(baddr,self.WATCH_PAGES[baddr]['s']-1,PAGE_E
##                   ev.debug.dont_watch_buffer(proc.get_pid(),baddr,self.WA
                   del self.WATCH_PAGES[baddr]

                _end = MemoryAddresses.align_address_to_page_end(ba+size)
                self.WATCH_PAGES[ba] = {'s': size, 'r':0,'x':0,'w':0,'prot':
                proc.mprotect(ba,size-1,PAGE_READONLY)

##                ev.debug.watch_buffer(proc.get_pid(),ba,size-1)
                self.log(ev,'[%d] [VirtProt] Set GuardPage @ %x - %s' % (rv,
            except Exception as e:
                import traceback
                self.log(ev,'err: %s'% `e`)
                self.log(ev,traceback.format_exc()
```

Heurystyki

```
def pre_WSAStartup(self,ev,ra,*args):
    self.__dump_from_entrypoint(ev,ra,'WSAStartup')

def pre_NtTerminateProcess(self,ev,ra,*args):
    self.handle_exit(ev,ra)

def pre_ExitProcess(self,ev,ra,*args):
    self.handle_exit(ev,ra)

def pre_HeapCreate(self,ev,ra,*args):
    self.log(ev,'create_heap')
    self.__dump_from_entrypoint(ev,ra,'HeapCreate')

def pre_CreateRemoteThread(self,ev,ra,*args):
    self.log(ev,'crt: %s'%repr(args))
    p = self.proc_name(ev,self.pid_from_handle(args[0]))
....
    self.log(ev,'crt')

def post_GetCommandLineW(self,ev,retval):
    ....
    mbi3 = proc.mquery(mbi2.AllocationBase)
    if (mbi.AllocationBase== mbi2.AllocationBase) or mbi.is_executable_a
        addr,size = self.find_whole_page(ev,proc,mbi.AllocationBase)
        self.dump_memory(ev,addr,size)
```

Parę Hookow

```
    apiHooks = {
        'kernel32.dll': [
            ('ExitProcess', (DWORD,)),
#            ('VirtualAllocEx', (HANDLE,LPVOID,DWORD,DWORD,DWORD)),
            ('CreateProcessInternalW', ( LPVOID,LPVOID, LPVOID,LPVOID,LPVOID
            ('CreateRemoteThread',(HANDLE,DWORD,DWORD,DWORD,DWORD,DWORD,DWOR
            ('HeapCreate',(DWORD,DWORD,DWORD)),
            ('GetCommandLineW', ( )),
            ('VirtualProtect',(LPVOID,DWORD,DWORD,LPVOID)),
            ('VirtualFree', (LPVOID,DWORD,DWORD)),
            ( 'CreateFileA'     , (PVOID, DWORD, DWORD, PVOID, DWORD, DWORD,
            ( 'CreateFileW'     , (PVOID, DWORD, DWORD, PVOID, DWORD, DWORD,
##
         ],
         'ntdll.dll': [
            ('NtAllocateVirtualMemory',(HANDLE,LPVOID,DWORD,DWORD,DWORD,DWOR
            ('NtReadVirtualMemory',(HANDLE,LPVOID,LPVOID, DWORD,LPVOID) ),
            ('NtWriteVirtualMemory',(HANDLE,LPVOID,LPVOID, DWORD) ),
            ('NtTerminateProcess', (DWORD,DWORD)),
            ('NtResumeThread', ( HANDLE, PVOID  )),
            ('NtOpenProcess', (LPVOID,LPVOID,LPVOID,LPVOID)),
            ('NtMapViewOfSection',(HANDLE,HANDLE,PVOID,DWORD,DWORD,LPVOID,DW
                ('NtUnmapViewOfSection',(HANDLE,PVOID)),
                ('NtProtectVirtualMemory',(HANDLE,PVOID,PVOID,ULONG,PVOID)),

         ],
        'ws2_32' : [
            ('WSAStartup',(DWORD,DWORD))
        ]
    }
```


NOTHING TO
DO HERE

Problemy

- anti-vm
- anti-dbg
- anit-fuuu...*
- roznice miedzy userland/kerneland
- narzut czasowy
- reszta problemow sandboxowych

Q & A

@maciekkotowicz lokalhost.pl/talks/zosia2014 CERT Polska @CERT_Polska_en

CERT.PL

rekrutacja@nask.pl