Can’t CrackMe

… The EULA said so

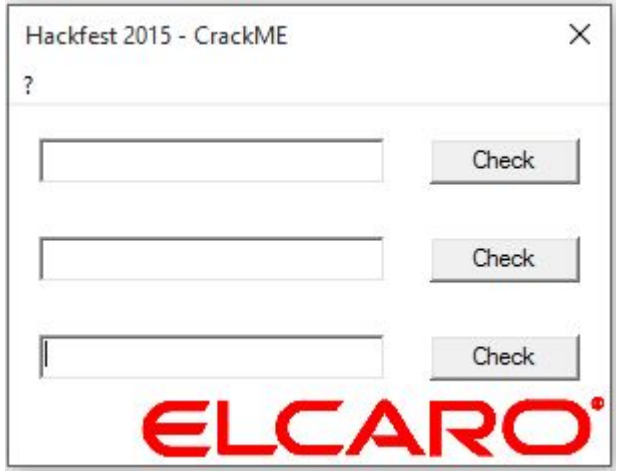
About Me

- Malware Researcher @ ESET
- Hackfest CTF since 2014
- CTF player since ???
- Twitter : https://twitter.com/__ek0

About the challenge

Find the 3 passwords.

URL : https://goo.gl/SMKBJE
Password : hackfest2015

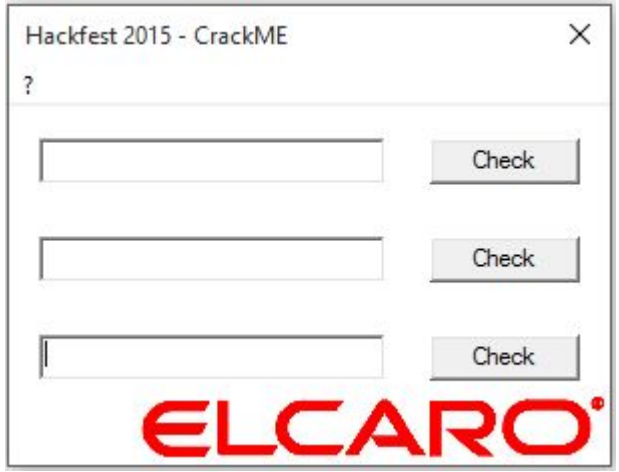


About the challenge

Find the 3 passwords.

URL : https://goo.gl/SMKBJE
Password : hackfest2015

Hint :
The AntiDebug is executed before the main.
The two first passwords are standards algorithms.
The first one is encoding.
The second one is crypto.



Solution

Solution

- AntiDebug : TlsCallback
- First password : Xored Base64 (Slightly modified)
- Second password : RC4
- Last password : Nanomites protection.

AntiDebug : TlsCallback

```
1  int __stdcall TlsCallback_0(int a1, int a2, int a3)
2  {
3    unsigned __int8 isDebuggerPresent; // ST0B_1@1
4    int result; // eax@1
5    int globalFlags; // [sp+0h] [bp-Ch]@1
6
7    globalFlags = *(_DWORD *)(__readfsdword(0x30) + 0x68);
8    isDebuggerPresent = *(_BYTE *)(__readfsdword(0x30) + 2);
9    result = isDebuggerPresent;
10   if ( isDebuggerPresent || globalFlags & 0x70 )
11   {
12     byte_417438 = 1;
13     ExitProcess(0);
14   }
15   return result;
16 }
```

AntiDebug : TlsCallback

```
1  int __stdcall TlsCallback_0(int a1, int a2, int a3)
2  {
3    unsigned __int8 isDebuggerPresent; // ST0B_1@1
4    int result; // eax@1
5    int globalFlags; // [sp+0h] [bp-Ch]@1
6
7    globalFlags = *(_DWORD *)(__readfsdword(0x30) + 0x68);
8    isDebuggerPresent = *(_BYTE *)(__readfsdword(0x30) + 2);
9    result = isDebuggerPresent;
10   if ( isDebuggerPresent || globalFlags & 0x70 )
11   {
12     byte_417438 = 1;
13     ExitProcess(0);
14   }
15   return result;
16 }
```

just need to patch this

First password : Modified Base64

```
hWnd = GetDlgItem(hDlg, 40000);
GetDlgItemTextW(hDlg, 40000, &String, 64);
b64encode((int)&String, (int)&v4, wcslen(&String), &v3);
xor_4e4e(&v4, v3);
result = wcslen((const unsigned __int16 *)encoded_password_array);
if ( result == v3 )
{
  result = wcmemcmp(encoded_password_array, &v4, 2 * v3);
  if ( !result )
    result = EnableWindow(hWnd, 0);
```

First password : Modified Base64

```
  *(_WORD *)(a2 + 2 * v11) = alphabet_array[(v4 >> 18) & 63];
  v5 = v11 + 1;
  *(_WORD *)(a2 + 2 * v5++) = alphabet_array[(v4 >> 12) & 63];
  *(_WORD *)(a2 + 2 * v5++) = alphabet_array[(v4 >> 6) & 63];
  *(_WORD *)(a2 + 2 * v5) = alphabet_array[v7 & 63];
  v11 = v5 + 1;
}
for ( i = 0; ; ++i )
{
  result = i;
  if ( i >= dword_416800[a3 % 3] )
    break;
  *(_WORD *)(a2 + 2 * (*a4 - 1 - i)) = '+';
}
```

First password : Modified Base64

```
  *(_WORD *)(a2 + 2 * v11) = alphabet_array[(v4 >> 18) & 63];
  v5 = v11 + 1;
  *(_WORD *)(a2 + 2 * v5++) = alphabet_array[(v4 >> 12) & 63];
  *(_WORD *)(a2 + 2 * v5++) = alphabet_array[(v4 >> 6) & 63];
  *(_WORD *)(a2 + 2 * v5) = alphabet_array[v7 & 63];
  v11 = v5 + 1;
}
for ( i = 0; ; ++i )
{
  result = i;
  if ( i >= dword_416800[a3 % 3] )
    break;
  *(_WORD *)(a2 + 2 * (*a4 - 1 - i)) = '+';
}
```

encoding

Normal alphabet :

ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/

Modified Alphabet :

QWERTYUIOPASDFGHJKLZXCVBNMqwertyuiopasdfghjklzxcvbnm9876543210_-

First password : Modified Base64

```
  *(_WORD *)(a2 + 2 * v11) = alphabet_array[(v4 >> 18) & 63];
  v5 = v11 + 1;
  *(_WORD *)(a2 + 2 * v5++) = alphabet_array[(v4 >> 12) & 63];
  *(_WORD *)(a2 + 2 * v5++) = alphabet_array[(v4 >> 6) & 63];
  *(_WORD *)(a2 + 2 * v5) = alphabet_array[v7 & 63];
  v11 = v5 + 1;
}
for ( i = 0; ; ++i )
{
  result = i;
  if ( i >= dword_416800[a3 % 3] )
    break;
  *(_WORD *)(a2 + 2 * (*a4 - 1 - i)) = '+';
}
```

encoding

padding ('=' in normal version)

Normal alphabet :

ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/

Modified Alphabet :

QWERTYUIOPASDFGHJKLZXCVBNMqwertyuiopasdfghjklzxcvbnm9876543210_-

First password : Modified Base64

```
  *(_WORD *)(a2 + 2 * v11) = alphabet_array[(v4 >> 18) & 63];
  v5 = v11 + 1;
  *(_WORD *)(a2 + 2 * v5++) = alphabet_array[(v4 >> 12) & 63];
  *(_WORD *)(a2 + 2 * v5++) = alphabet_array[(v4 >> 6) & 63];
  *(_WORD *)(a2 + 2 * v5) = alphabet_array[v7 & 63];
  v11 = v5 + 1;
}
for ( i = 0; ; ++i )
{
  result = i;
  if ( i >= dword_416800[a3 % 3] )
    break;
  *(_WORD *)(a2 + 2 * (*a4 - 1 - i)) = '+';
}
```

encoding

FLAG : ModifiedBase64IsStillBase64!!

padding ('=' in normal version)

Normal alphabet :

ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/

Modified Alphabet :

QWERTYUIOPASDFGHJKLZXCVBNMqwertyuiopasdfghjklzxcvbnm9876543210_-

Second Password : RC4

```
hWnd = GetDlgItem(hDlg, 40005);
GetWindowTextW(hWnd, &String, 64);
rc4_init(&v6, key_ptr, 12);
rc4_crypt(&v6, &String, &v7, 64);
v1 = wcmemcmp(encrypted_pass_ptr, &v7, strlen((const char *)encrypted_pass_ptr));
```

Second Password : RC4

```
for ( i = 0; i < 256; ++i )
  a1[i] = i;
```

```
while ( (signed int)v5 < 256 )
{
  v6 += key[v5 % a3] + table[v5];
  fn_swap(&table[v5], &table[v6]);
  result = v5++ + 1;
}
```

```
for ( i = 0; i < size; ++i )
{
  table[257] += table[++table[256]];
  fn_swap(&table[table[256]], &table[table[257]]);
  encrypted_string[i] = table[(unsigned __int8)(table[table[257]] + table[table[256]])] ^ password[i];
  result = i + 1;
}
```

Second Password : RC4

rc4_init

```
for ( i = 0; i < 256; ++i )
  a1[i] = i;
```

```
while ( (signed int)v5 < 256 )
{
  v6 += key[v5 % a3] + table[v5];
  fn_swap(&table[v5], &table[v6]);
  result = v5++ + 1;
}
```

```
for ( i = 0; i < size; ++i )
{
  table[257] += table[++table[256]];
  fn_swap(&table[table[256]], &table[table[257]]);
  encrypted_string[i] = table[(unsigned __int8)(table[table[257]] + table[table[256]])] ^ password[i];
  result = i + 1;
}
```

Second Password : RC4

rc4_init

```
for ( i = 0; i < 256; ++i )
  a1[i] = i;
```

```
while ( (signed int)v5 < 256 )
{
  v6 += key[v5 % a3] + table[v5];
  fn_swap(&table[v5], &table[v6]);
  result = v5++ + 1;
}
```

rc4_crypt

```
for ( i = 0; i < size; ++i )
{
  table[257] += table[++table[256]];
  fn_swap(&table[table[256]], &table[table[257]]);
  encrypted_string[i] = table[(unsigned __int8)(table[table[257]] + table[table[256]])] ^ password[i];
  result = i + 1;
}
```

Second Password : RC4

key is already in memory..

```
hWnd = GetDlgItem(hDlg, 40005);
GetWindowTextW(hWnd, &String, 64);
rc4_init(&v6, key_ptr, 12);
rc4_crypt(&v6, &String, &v7, 64);
v1 = wcmemcmp(encrypted_pass_ptr, &v7, strlen((const char *)encrypted_pass_ptr));
```

encrypted flag

Second Password : RC4

key is already in memory..

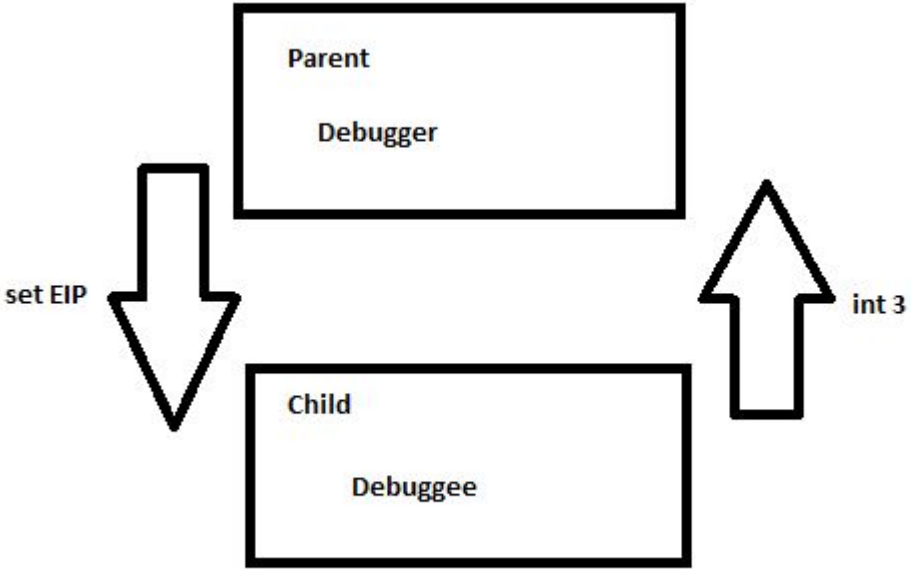
```
hWnd = GetDlgItem(hDlg, 40005);
GetWindowTextW(hWnd, &String, 64);
rc4_init(&v6, key_ptr, 12);
rc4_crypt(&v6, &String, &v7, 64);
v1 = wcmemcmp(encrypted_pass_ptr, &v7, strlen((const char *)encrypted_pass_ptr));
```

encrypted flag

FLAG : NoNeedToReverseWhenYouHaveTheKey

Third password : Nanomites

What’s nanomites ?



```
hWnd = GetDlgItem(hDlg, 40007);
GetWindowTextW(hWnd, &Buffer, 64);
v1 = (unsigned __int8)fn_enable_privileges(L'
CreateProcessW(0, CommandLine, 0, 0, 1, 0x80(
hProcess = ProcessInformation.hProcess;
DebugEvent.dwDebugEventCode = 0;
sub_403DC0(&DebugEvent.dwProcessId, 0, 92);
v8 = 0;
dword_4167FC = 1;
WriteProcessMemory(
  hProcess,
  L"ts of any Program benchmark tests without
   "-use any Elcaro name, trademark or logo.\
  &Buffer,
  0x80u,
  &NumberOfBytesWritten);
FlushInstructionCache(
  hProcess,
  L"ts of any Program benchmark tests without
   "-use any Elcaro name, trademark or logo.\
  0x80u);
ResumeThread(ProcessInformation.hThread);
while ( dword_4167FC )
{
  result = WaitForDebugEvent(&DebugEvent, 0xF
  if ( !result )
    return result;
  fn_process_debug_event((int)&DebugEvent, &
  ContinueDebugEvent(DebugEvent.dwProcessId,
}
WaitForSingleObject(ProcessInformation.hProce
```

```
mov     [ebp+var_C], 0
int     3                ; Trap to D
mov     cx, [ebp+var_8]
add     cx, 1
mov     [ebp+var_8], cx
int     3                ; Trap to D
mov     [ebp+var_C], 0
int     3                ; Trap to D
mov     edx, [ebp+var_C]
mov     ax, word_414220[edx*2]
mov     [ebp+var_4], ax
int     3                ; Trap to D
mov     ecx, [ebp+var_C]
add     ecx, 1
mov     [ebp+var_C], ecx
int     3                ; Trap to D
movzx   edx, [ebp+var_4]
not     edx
mov     [ebp+var_4], dx
int     3                ; Trap to D
mov     eax, [ebp+var_C]
mov     cx, [ebp+var_4]
mov     word_414220[eax*2], cx
int     3                ; Trap to D
movzx   edx, [ebp+var_4]
add     edx, 1
mov     [ebp+var_4], dx
int     3                ; Trap to D
movzx   eax, [ebp+var_4]
sub     eax, 1
mov     [ebp+var_4], ax
int     3                ; Trap to D
```

parent

```
hWnd = GetDlgItem(hDlg, 40007);
GetWindowTextW(hWnd, &Buffer, 64);
v1 = (unsigned __int8)fn_enable_privileges(L'
CreateProcessW(0, CommandLine, 0, 0, 1, 0x80(
hProcess = ProcessInformation.hProcess;
DebugEvent.dwDebugEventCode = 0;
sub_403DC0(&DebugEvent.dwProcessId, 0, 92);
v8 = 0;
dword_4167FC = 1;
WriteProcessMemory(
  hProcess,
  L"ts of any Program benchmark tests without
   "-use any Elcaro name, trademark or logo.\
  &Buffer,
  0x80u,
  &NumberOfBytesWritten);
FlushInstructionCache(
  hProcess,
  L"ts of any Program benchmark tests without
   "-use any Elcaro name, trademark or logo.\
  0x80u);
ResumeThread(ProcessInformation.hThread);
while ( dword_4167FC )
{
  result = WaitForDebugEvent(&DebugEvent, 0xF
  if ( !result )
    return result;
  fn_process_debug_event((int)&DebugEvent, &
  ContinueDebugEvent(DebugEvent.dwProcessId,
}
WaitForSingleObject(ProcessInformation.hProce
```

```
mov     [ebp+var_C], 0
int     3                ; Trap to D
mov     cx, [ebp+var_8]
add     cx, 1
mov     [ebp+var_8], cx
int     3                ; Trap to D
mov     [ebp+var_C], 0
int     3                ; Trap to D
mov     edx, [ebp+var_C]
mov     ax, word_414220[edx*2]
mov     [ebp+var_4], ax
int     3                ; Trap to D
mov     ecx, [ebp+var_C]
add     ecx, 1
mov     [ebp+var_C], ecx
int     3                ; Trap to D
movzx   edx, [ebp+var_4]
not     edx
mov     [ebp+var_4], dx
int     3                ; Trap to D
mov     eax, [ebp+var_C]
mov     cx, [ebp+var_4]
mov     word_414220[eax*2], cx
int     3                ; Trap to D
movzx   edx, [ebp+var_4]
add     edx, 1
mov     [ebp+var_4], dx
int     3                ; Trap to D
movzx   eax, [ebp+var_4]
sub     eax, 1
mov     [ebp+var_4], ax
int     3                ; Trap to D
```

parent

```
hWnd = GetDlgItem(hDlg, 40007);
GetWindowTextW(hWnd, &Buffer, 64);
v1 = (unsigned __int8)fn_enable_privileges(L'
CreateProcessW(0, CommandLine, 0, 0, 1, 0x80(
hProcess = ProcessInformation.hProcess;
DebugEvent.dwDebugEventCode = 0;
sub_403DC0(&DebugEvent.dwProcessId, 0, 92);
v8 = 0;
dword_4167FC = 1;
WriteProcessMemory(
  hProcess,
  L"ts of any Program benchmark tests without
   "-use any Elcaro name, trademark or logo.\
  &Buffer,
  0x80u,
  &NumberOfBytesWritten);
FlushInstructionCache(
  hProcess,
  L"ts of any Program benchmark tests without
   "-use any Elcaro name, trademark or logo.\
  0x80u);
ResumeThread(ProcessInformation.hThread);
while ( dword_4167FC )
{
  result = WaitForDebugEvent(&DebugEvent, 0xF
  if ( !result )
    return result;
  fn_process_debug_event((int)&DebugEvent, &
  ContinueDebugEvent(DebugEvent.dwProcessId,
}
WaitForSingleObject(ProcessInformation.hProce
```

child

```
mov     [ebp+var_C], 0
int     3                ; Trap to D
mov     cx, [ebp+var_8]
add     cx, 1
mov     [ebp+var_8], cx
int     3                ; Trap to D
mov     [ebp+var_C], 0
int     3                ; Trap to D
mov     edx, [ebp+var_C]
mov     ax, word_414220[edx*2]
mov     [ebp+var_4], ax
int     3                ; Trap to D
mov     ecx, [ebp+var_C]
add     ecx, 1
mov     [ebp+var_C], ecx
int     3                ; Trap to D
movzx   edx, [ebp+var_4]
not     edx
mov     [ebp+var_4], dx
int     3                ; Trap to D
mov     eax, [ebp+var_C]
mov     cx, [ebp+var_4]
mov     word_414220[eax*2], cx
int     3                ; Trap to D
movzx   edx, [ebp+var_4]
add     edx, 1
mov     [ebp+var_4], dx
int     3                ; Trap to D
movzx   eax, [ebp+var_4]
sub     eax, 1
mov     [ebp+var_4], ax
int     3                ; Trap to D
```

parent

```
hWnd = GetDlgItem(hDlg, 40007);
GetWindowTextW(hWnd, &Buffer, 64);
v1 = (unsigned __int8)fn_enable_privileges(L'
CreateProcessW(0, CommandLine, 0, 0, 1, 0x80
hProcess = ProcessInformation.hProcess;
DebugEvent.dwDebugEventCode = 0;
sub_403DC0(&DebugEvent.dwProcessId, 0, 92);
v8 = 0;
dword_4167FC = 1;
WriteProcessMemory(
  hProcess,
  L"ts of any Program benchmark tests without
   "-use any Elcaro name, trademark or logo.\
  &Buffer,
  0x80u,
  &NumberOfBytesWritten);
FlushInstructionCache(
  hProcess,
  L"ts of any Program benchmark tests without
   "-use any Elcaro name, trademark or logo.\
  0x80u);
ResumeThread(ProcessInformation.hThread);
while ( dword_4167FC )
{
  result = WaitForDebugEvent(&DebugEvent, 0xF
  if ( !result )
    return result;
  fn_process_debug_event((int)&DebugEvent, &
  ContinueDebugEvent(DebugEvent.dwProcessId,
}
WaitForSingleObject(ProcessInformation.hProce
```

password !

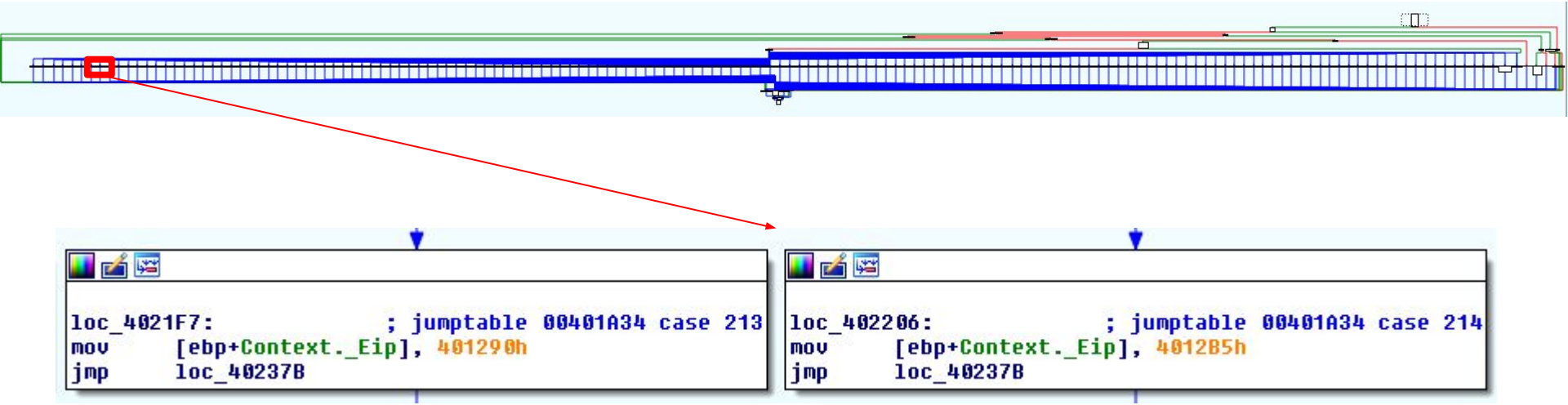
child

```
mov     [ebp+var_C], 0
int     3                ; Trap to D
mov     cx, [ebp+var_8]
add     cx, 1
mov     [ebp+var_8], cx
int     3                ; Trap to D
mov     [ebp+var_C], 0
int     3                ; Trap to D
mov     edx, [ebp+var_C]
mov     ax, word_414220[edx*2]
mov     [ebp+var_4], ax
int     3                ; Trap to D
mov     ecx, [ebp+var_C]
add     ecx, 1
mov     [ebp+var_C], ecx
int     3                ; Trap to D
movzx   edx, [ebp+var_4]
not     edx
mov     [ebp+var_4], dx
int     3                ; Trap to D
mov     eax, [ebp+var_C]
mov     cx, [ebp+var_4]
mov     word_414220[eax*2], cx
int     3                ; Trap to D
movzx   edx, [ebp+var_4]
add     edx, 1
mov     [ebp+var_4], dx
int     3                ; Trap to D
movzx   eax, [ebp+var_4]
sub     eax, 1
mov     [ebp+var_4], ax
int     3                ; Trap to D
```

Third password : Nanomites

Event processing routine.

When a debug event happens, the parent set the new EIP in the child context.



Nanomites source code

```
case EXCEPTION_BREAKPOINT:
    // Get thread context
    // Already suspended bc dbg event.
    hThread = OpenThread(THREAD_SET_CONTEXT | THREAD_GET_CONTEXT, FALSE, event->dwThreadId);
    GetThreadContext(hThread, &ctx);
    //wsprintf(dbg_string, L"CODE : %#x, EIP : %#x\n", exception.ExceptionRecord.ExceptionCode, ctx.Eip);
    //OutputDebugString(dbg_string);

    switch(*exception_cnt) {
    case 0:
        ctx.Eip = 0x00401260; // First exception happen in the RT
        break;
    case 1:
    case 2:
    case 3:
    case 4:
    case 5:
    case 6:
    case 7:
    case 8:
    case 9:
    case 10:
    case 11:
    case 12:
    case 13:
    case 14:
    case 15:
    case 17:
    case 18:
    case 19:
    case 20:
        ctx.Eip = INC_KEY;
        break;
    case 21:
        ctx.Eip = GET_CH;
        break;
    case 22:
        ctx.Eip = DEC_CH;
        break;
    case 23:
        ctx.Eip = DEC_CH;
        break;
    case 24:
        ctx.Eip = DEC_CH;
        break;
```

Nanomites source code

```
case EXCEPTION_BREAKPOINT:
    // Get thread context
    // Already suspended bc dbg event.
    hThread = OpenThread(THREAD_SET_CONTEXT | THREAD_GET_CONTEXT, FALSE, event->dwThreadId);
    GetThreadContext(hThread, &ctx);
    //wsprintf(dbg_string, L"CODE : %#x, EIP : %#x\n", exception.ExceptionRecord.ExceptionCode, ctx.Eip);
    //OutputDebugString(dbg_string);

    switch(*exception_cnt) {
    case 0:
        ctx.Eip = 0x00401260; // First exception happen in the RT
        break;
    case 1:
    case 2:
    case 3:
    case 4:
    case 5:
    case 6:
    case 7:
    case 8:
    case 9:
    case 10:
    case 11:
    case 12:
    case 13:
    case 14:
    case 15:
    case 17:
    case 18:
    case 19:
    case 20:
        ctx.Eip = INC_KEY;
        break;
    case 21:
        ctx.Eip = GET_CH;
        break;
    case 22:
        ctx.Eip = DEC_CH;
        break;
    case 23:
        ctx.Eip = DEC_CH;
        break;
    case 24:
        ctx.Eip = DEC_CH;
        break;
```

```
cmp     [ebp+var_3A8], EXCEPTION_BREAKPOINT
jz      loc_4019E1
```

```
loc_4021F7:                 ; jumptable 00401A34 case 213
mov     [ebp+Context._Eip], 401290h
jmp     loc_40237B
```

```
loc_402206:                 ; jumptable 00401A34 case 214
mov     [ebp+Context._Eip], 4012B5h
jmp     loc_40237B
```

```
loc_4019E1:
mov     edx, [ebp+arg_0]
mov     eax, [edx+8]
push    eax             ; dwThreadId
push    0               ; bInheritHandle
push    18h             ; dwDesiredAccess
call    ds:OpenThread
mov     [ebp+hThread], eax
lea     ecx, [ebp+Context]
push    ecx             ; lpContext
mov     edx, [ebp+hThread]
push    edx             ; hThread
call    ds:GetThreadContext
mov     eax, [ebp+int3_cnt_ptr]
mov     ecx, [eax]
mov     [ebp+int3_cnt], ecx
cmp     [ebp+int3_cnt], 239 ; switch 240 cases
ja      loc_40237A      ; jumptable 00401A34 default case
```

```
cmp     [ebp+var_3A8], EXCEPTION_BREAKPOINT
jz      loc_4019E1
```

```
loc_4021F7:             ; jumptable 00401A34 case 213
mov     [ebp+Context._Eip], 401290h
jmp     loc_40237B
```

```
loc_402206:             ; jumptable 00401A34 case 214
mov     [ebp+Context._Eip], 4012B5h
jmp     loc_40237B
```

```
loc_4019E1:
mov     edx, [ebp+arg_0]
mov     eax, [edx+8]
push    eax             ; dwThreadId
push    0               ; bInheritHandle
push    18h             ; dwDesiredAccess
call    ds:OpenThread
mov     [ebp+hThread], eax
lea     ecx, [ebp+Context]
push    ecx             ; lpContext
mov     edx, [ebp+hThread]
push    edx             ; hThread
call    ds:GetThreadContext
mov     eax, [ebp+int3_cnt_ptr]
mov     ecx, [eax]
mov     [ebp+int3_cnt], ecx
cmp     [ebp+int3_cnt], 239 ; switch 240 cases
ja      loc_40237A      ; jumptable 00401A34 default case
```

```
cmp     [ebp+var_3A8], EXCEPTION_BREAKPOINT
jz      loc_4019E1
```

```
loc_4021F7:                     ; jumptable 00401A34 case 213
mov     [ebp+Context._Eip], 401290h
jmp     loc_40237B
```

```
loc_402206:                     ; jumptable 00401A34 case 214
mov     [ebp+Context._Eip], 4012B5h
jmp     loc_40237B
```

```
loc_4019E1:
mov     edx, [ebp+arg_0]
mov     eax, [edx+8]
push    eax             ; dwThreadId
push    0               ; bInheritHandle
push    18h             ; dwDesiredAccess
call    ds:OpenThread
mov     [ebp+hThread], eax
lea     ecx, [ebp+Context]
push    ecx             ; lpContext
mov     edx, [ebp+hThread]
push    edx             ; hThread
call    ds:GetThreadContext
mov     eax, [ebp+int3_cnt_ptr]
mov     ecx, [eax]
mov     [ebp+int3_cnt], ecx
cmp     [ebp+int3_cnt], 239 ; switch 240 cases
ja      loc_40237A      ; jumptable 00401A34 default case
```

```
cmp     [ebp+var_3A8], EXCEPTION_BREAKPOINT
jz      loc_4019E1
```

```
loc_4021F7:                     ; jumptable 00401A34 case 213
mov     [ebp+Context._Eip], 401290h
jmp     loc_40237B
```

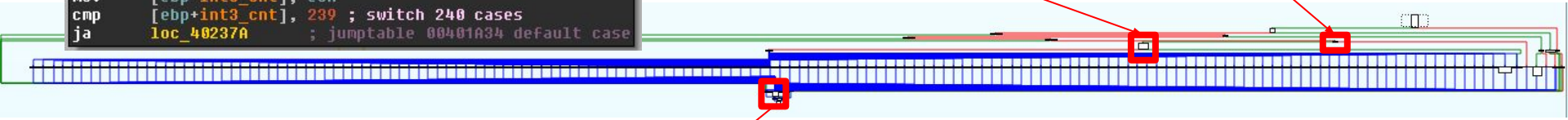
operation address in the child process

```
loc_402206:                     ; jumptable 00401A34 case 214
mov     [ebp+Context._Eip], 4012B5h
jmp     loc_40237B
```

another operation address

```
loc_4019E1:
mov     edx, [ebp+arg_0]
mov     eax, [edx+8]
push    eax             ; dwThreadId
push    0               ; bInheritHandle
push    18h             ; dwDesiredAccess
call    ds:OpenThread
mov     [ebp+hThread], eax
lea     ecx, [ebp+Context]
push    ecx             ; lpContext
mov     edx, [ebp+hThread]
push    edx             ; hThread
call    ds:GetThreadContext
mov     eax, [ebp+int3_cnt_ptr]
mov     ecx, [eax]
mov     [ebp+int3_cnt], ecx
cmp     [ebp+int3_cnt], 239 ; switch 240 cases
ja      loc_40237A      ; jumptable 00401A34 default case
```

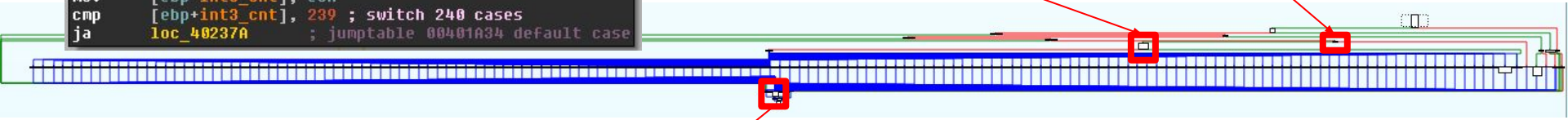
```
cmp     [ebp+var_3A8], EXCEPTION_BREAKPOINT
jz      loc_4019E1
```



```
loc_40237B:
mov     eax, [ebp+int3_cnt_ptr]
mov     ecx, [eax]
add     ecx, 1
mov     edx, [ebp+int3_cnt_ptr]
mov     [edx], ecx
lea     eax, [ebp+Context]
push    eax             ; lpContext
mov     ecx, [ebp+hThread]
push    ecx             ; hThread
call    ds:SetThreadContext
mov     edx, [ebp+hThread]
push    edx             ; hObject
call    ds:CloseHandle
jmp     short loc_4023B6
```

```
loc_4019E1:
mov     edx, [ebp+arg_0]
mov     eax, [edx+8]
push    eax             ; dwThreadId
push    0               ; bInheritHandle
push    18h             ; dwDesiredAccess
call    ds:OpenThread
mov     [ebp+hThread], eax
lea     ecx, [ebp+Context]
push    ecx             ; lpContext
mov     edx, [ebp+hThread]
push    edx             ; hThread
call    ds:GetThreadContext
mov     eax, [ebp+int3_cnt_ptr]
mov     ecx, [eax]
mov     [ebp+int3_cnt], ecx
cmp     [ebp+int3_cnt], 239 ; switch 240 cases
ja      loc_40237A      ; jumptable 00401A34 default case
```

```
cmp     [ebp+var_3A8], EXCEPTION_BREAKPOINT
jz      loc_4019E1
```



```
loc_40237B:
mov     eax, [ebp+int3_cnt_ptr]
mov     ecx, [eax]
add     ecx, 1
mov     edx, [ebp+int3_cnt_ptr]
mov     [edx], ecx
lea     eax, [ebp+Context]
push    eax             ; lpContext
mov     ecx, [ebp+hThread]
push    ecx             ; hThread
call    ds:SetThreadContext
mov     edx, [ebp+hThread]
push    edx             ; hObject
call    ds:CloseHandle
jmp     short loc_4023B6
```

```
loc_4019E1:
mov     edx, [ebp+arg_0]
mov     eax, [edx+8]
push    eax             ; dwThreadId
push    0               ; bInheritHandle
push    18h             ; dwDesiredAccess
call    ds:OpenThread
mov     [ebp+hThread], eax
lea     ecx, [ebp+Context]
push    ecx             ; lpContext
mov     edx, [ebp+hThread]
push    edx             ; hThread
call    ds:GetThreadContext
mov     eax, [ebp+int3_cnt_ptr]
mov     ecx, [eax]
mov     [ebp+int3_cnt], ecx
cmp     [ebp+int3_cnt], 239 ; switch 240 cases
ja      loc_40237A      ; jumptable 00401A34 default case
```

```
cmp     [ebp+var_3A8], EXCEPTION_BREAKPOINT
jz      loc_4019E1
```

```
loc_40237B:
mov     eax, [ebp+int3_cnt_ptr]
mov     ecx, [eax]
add     ecx, 1
mov     edx, [ebp+int3_cnt_ptr]
mov     [edx], ecx
lea     eax, [ebp+Context]
push    eax             ; lpContext
mov     ecx, [ebp+hThread]
push    ecx             ; hThread
call    ds:SetThreadContext
mov     edx, [ebp+hThread]
push    edx             ; hObject
call    ds:CloseHandle
jmp     short loc_4023B6
```

```
child_end:              ; jumptable 00401A34 case 239
push    0
push    80h             ; nSize
push    offset unk_4172A8 ; lpBuffer
push    offset password ; "ts of any Program benchmark
mov     ecx, hProcess
push    ecx             ; hProcess
call    ds:ReadProcessMemory
push    0               ; lpNumberOfBytesRead
push    80h             ; nSize
push    offset unk_4173B8 ; lpBuffer
push    offset flag     ; lpBaseAddress
mov     edx, hProcess
push    edx             ; hProcess
call    ds:ReadProcessMemory
mov     [ebp+Context._Eip], 4012FFh
jmp     short loc_40237B
```

Child operations

```
mov     cx, [ebp+key]
add     cx, 1
mov     [ebp+key], cx
```

key++

```
movzx   eax, [ebp+tmp]
sub     eax, 1
mov     [ebp+tmp], ax
```

char--

```
mov     edx, [ebp+i]
mov     ax, password[edx*2]
mov     [ebp+tmp], ax
```

char = password[i]

```
movzx   edx, [ebp+tmp]
add     edx, 1
mov     [ebp+tmp], dx
```

char++

```
mov     ecx, [ebp+i]
add     ecx, 1
mov     [ebp+i], ecx
```

i++

```
movzx   edx, [ebp+tmp]
not     edx
mov     [ebp+tmp], dx
```

char = ~char

```
mov     [ebp+i], 0
```

i = 0

```
mov     eax, [ebp+i]
mov     cx, [ebp+tmp]
mov     password[eax*2], cx
```

password[i] = char

```
movzx   ecx, [ebp+key]
movzx   edx, [ebp+tmp]
xor     edx, ecx
mov     [ebp+tmp], dx
```

char = char ^ key

Child operations, building the script

- Each character is encoded between a GET_CH and a PUT_CH op
- Each time we have another GET_CH, restart with original character.
- Each INC_I, increment index.
- The switch case condition is the EXCEPTION_BREAKPOINT counter.
- Don’t forget to do the reverse for each character.
- But sometimes …

Child operations, building the script

- Each character is encoded between a GET_CH and a PUT_CH op
- Each time we have another GET_CH, restart with original character.
- Each INC_I, increment index.
- The switch case condition is the EXCEPTION_BREAKPOINT counter.
- Don’t forget to do the reverse for each character.
- But sometimes … even software protections have flaws.
- Flag in the child memory.
- The validation is in the parent (we can debug !)
- What happens if we patch the child ?

Child operations

```
mov     cx, [ebp+key]
add     cx, 1
mov     [ebp+key], cx
```

key++

```
movzx   eax, [ebp+tmp]
sub     eax, 1
mov     [ebp+tmp], ax
```

char--

```
mov     edx, [ebp+i]
mov     ax, password[edx*2]
mov     [ebp+tmp], ax
```

char = password[i]

```
movzx   edx, [ebp+tmp]
add     edx, 1
mov     [ebp+tmp], dx
```

char++

```
mov     ecx, [ebp+i]
add     ecx, 1
mov     [ebp+i], ecx
```

i++

```
movzx   edx, [ebp+tmp]
not     edx
mov     [ebp+tmp], dx
```

char = ~char

```
mov     [ebp+i], 0
```

i = 0

```
mov     eax, [ebp+i]
mov     cx, [ebp+tmp]
mov     password[eax*2], cx
```

password[i] = char

```
movzx   ecx, [ebp+key]
movzx   edx, [ebp+tmp]
xor     edx, ecx
mov     [ebp+tmp], dx
```

char = char ^ key

Child operations : Reloaded

```
mov     cx, [ebp+key]
add     cx, 1
mov     [ebp+key], cx
```

key++

```
movzx   eax, [ebp+tmp]
sub     eax, 1
mov     [ebp+tmp], ax
```

add eax, 1

char--

```
mov     edx, [ebp+i]
mov     ax, password[edx*2]
mov     [ebp+tmp], ax
```

flag

char = password[i]

```
movzx   edx, [ebp+tmp]
add     edx, 1
mov     [ebp+tmp], dx
```

sub edx, 1

char++

```
mov     ecx, [ebp+i]
add     ecx, 1
mov     [ebp+i], ecx
```

i++

```
movzx   edx, [ebp+tmp]
not     edx
mov     [ebp+tmp], dx
```

char = ~char

```
mov     [ebp+i], 0
```

i = 0

```
mov     eax, [ebp+i]
mov     cx, [ebp+tmp]
mov     password[eax*2], cx
```

flag

password[i] = char

```
movzx   ecx, [ebp+key]
movzx   edx, [ebp+tmp]
xor     edx, ecx
mov     [ebp+tmp], dx
```

char = char ^ key

Result once patched.

Not perfect, but i see a pattern ;) (and took only 2 minutes to test !)

```
00F12302  .   68 80000000   PUSH 80                          Size = 128.
00F12307  .   68 B873F200   PUSH OFFSET 00F273B8              Buffer = HF_crackme_patched.0F273B8 -> 49
00F1230C  .   68 C0374100   PUSH 4137C0                       BaseAddress = 4137C0
00F12311  .   8B15 2873F20( MOV EDX,DWORD PTR DS:[0F27328]
00F12317  .   52            PUSH EDX                          hProcess
00F12318  .   FF15 38F0F10( CALL DWORD PTR DS:[<&KERNEL32.ReadProce- KERNEL32.ReadProcessMemory
00F1231E  .   C785 68FDFFF( MOV DWORD PTR SS:[EBP-298],4012FF
00F12328  .-  EB 01         JMP SHORT 00F1232B
00F1232A  >   90            NOP                               Case D6 of switch HF_crackme_patched.0F119
00F1232B  >   8B45 0C       MOV EAX,DWORD PTR SS:[EBP+0C]
00F1232E  .   8B08          MOV ECX,DWORD PTR DS:[EAX]
00F12330  .   83C1 01       ADD ECX,1
00F12333  .   8B55 0C       MOV EDX,DWORD PTR SS:[EBP+0C]
00F12336  .   890A          MOV DWORD PTR DS:[EDX],ECX
00F12338  .   8D85 B0FCFFF( LEA EAX,[EBP-350]
00F1233E  .   50            PUSH EAX                          pContext
00F1233F  .   8B8D 54FCFFF( MOV ECX,DWORD PTR SS:[EBP-3AC]
00F12345  .   51            PUSH ECX                          hThread
00F12346  .   FF15 7CF0F10( CALL DWORD PTR DS:[<&KERNEL32.SetThread( KERNEL32.SetThreadContext
00F1234C  .   8B95 54FCFFF( MOV EDX,DWORD PTR SS:[EBP-3AC]
00F12352  .   52            PUSH EDX                          hObject
00F12353  .   FF15 30F0F10( CALL DWORD PTR DS:[<&KERNEL32.CloseHand KERNEL32.CloseHandle
00F12359  .-  EB 0B         JMP SHORT 00F12366
Stack [003CF16C]=0 (current registers)
Imm=HF_crackme_patched.00F273B8, UNICODE "IoohJkbBreaiingThisPanomitesProtectPon!!"
```

Result once patched.

Not perfect, but i see a pattern ;) (and took only 2 minutes to test !)

```
00F12302  .   68 80000000   PUSH 80                                  Size = 128.
00F12307  .   68 B873F200   PUSH OFFSET 00F273B8                     Buffer = HF_crackme_patched.0F273B8 -> 49
00F1230C  .   68 C0374100   PUSH 4137C0                              BaseAddress = 4137C0
00F12311  .   8B15 2873F20( MOV EDX,DWORD PTR DS:[0F27328]
00F12317  .   52            PUSH EDX                                 hProcess
00F12318  .   FF15 38F0F10( CALL DWORD PTR DS:[<&KERNEL32.ReadProce- KERNEL32.ReadProcessMemory
00F1231E  .   C785 68FDFFF( MOV DWORD PTR SS:[EBP-298],4012FF
00F12328  .-  EB 01         JMP SHORT 00F1232B
00F1232A  >   90            NOP                                      Case D6 of switch HF_crackme_patched.0F119
00F1232B  >   8B45 0C       MOV EAX,DWORD PTR SS:[EBP+0C]
00F1232E  .   8B08          MOV ECX,DWORD PTR DS:[EAX]
00F12330  .   83C1 01       ADD ECX,1
00F12333  .   8B55 0C       MOV EDX,DWORD PTR SS:[EBP+0C]
00F12336  .   890A          MOV DWORD PTR DS:[EDX],ECX
00F12338  .   8D85 B0FCFFF( LEA EAX,[EBP-350]
00F1233E  .   50            PUSH EAX                                 pContext
00F1233F  .   8B8D 54FCFFF( MOV ECX,DWORD PTR SS:[EBP-3AC]
00F12345  .   51            PUSH ECX                                 hThread
00F12346  .   FF15 7CF0F10( CALL DWORD PTR DS:[<&KERNEL32.SetThread( KERNEL32.SetThreadContext
00F1234C  .   8B95 54FCFFF( MOV EDX,DWORD PTR SS:[EBP-3AC]
00F12352  .   52            PUSH EDX                                 hObject
00F12353  .   FF15 30F0F10( CALL DWORD PTR DS:[<&KERNEL32.CloseHand( KERNEL32.CloseHandle
00F12359  .-  EB 0B         JMP SHORT 00F12366
Stack [003CF16C]=0 (current registers)
Imm=HF_crackme_patched.00F273B8, UNICODE "IoohJkbBreaiingThisPanomitesProtectPon!!"
```

NOTE : The flag is “GoodJobBreakingThisNanomitesProtection!!”

Thank you