SYNACKTIV

American Conquest

(et ses dingueries)

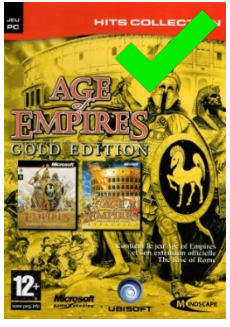
Bière Sécu Rennes

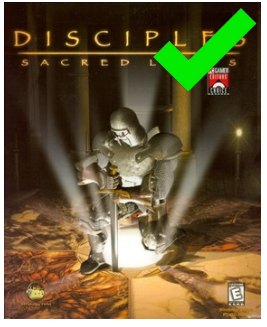
Motivations

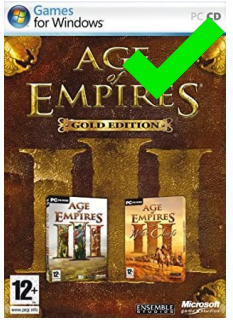
- **Pourquoi chercher des vulnérabilités dans les vieux jeux vidéo ?**
  - Pour le fun
  - Pour recycler sa collection de jeux vidéo
  - Intéressant lorsque les jeux sont réédités
  - De nombreux bugs, exploitation pas toujours évidente
- **Concentration uniquement sur les RCE (pas de technique de cheat)**

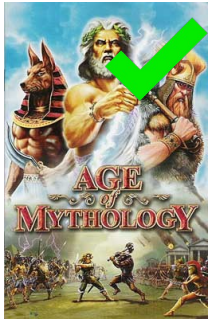
Tableau de chasse

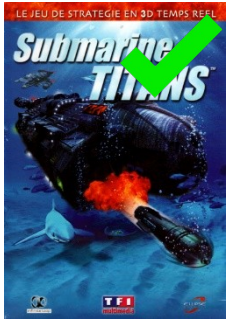
Windows XP

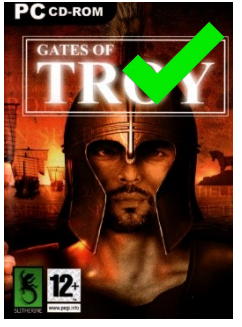






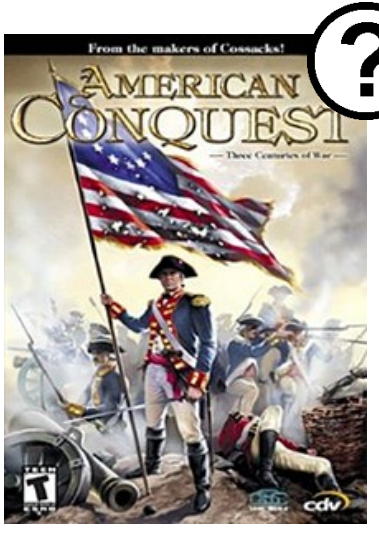






Windows 10 (re-released)

SYNACKTIV

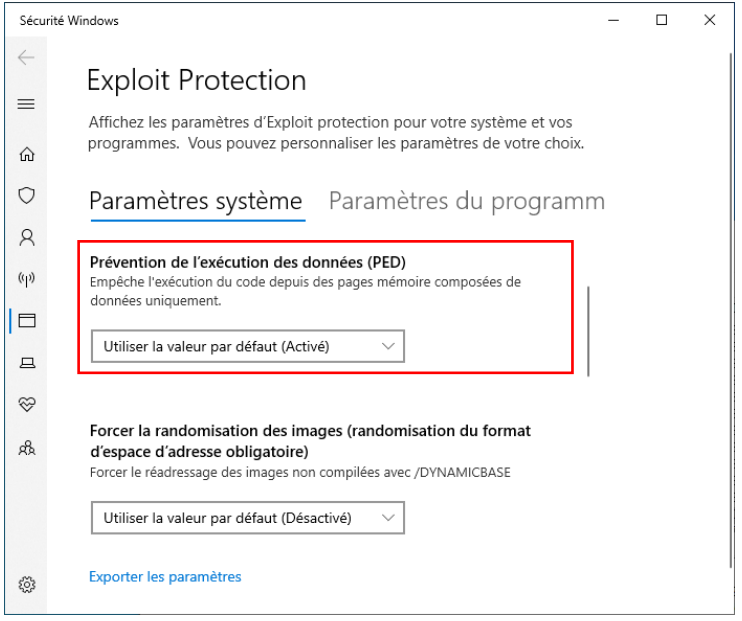


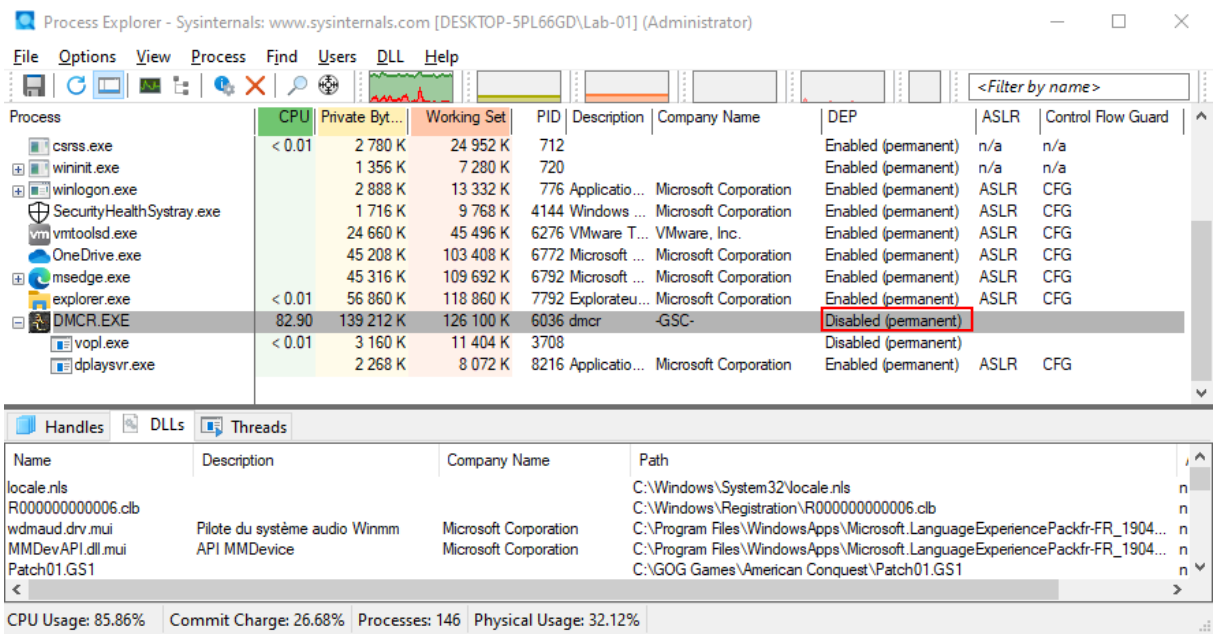
Où trouver des vieux jeux réédités ?

American Conquest

- Type - Stratégie
- Date de sortie - 7 Février 2003
- Editeur - GSC Game World

Windows 10: Mitigations





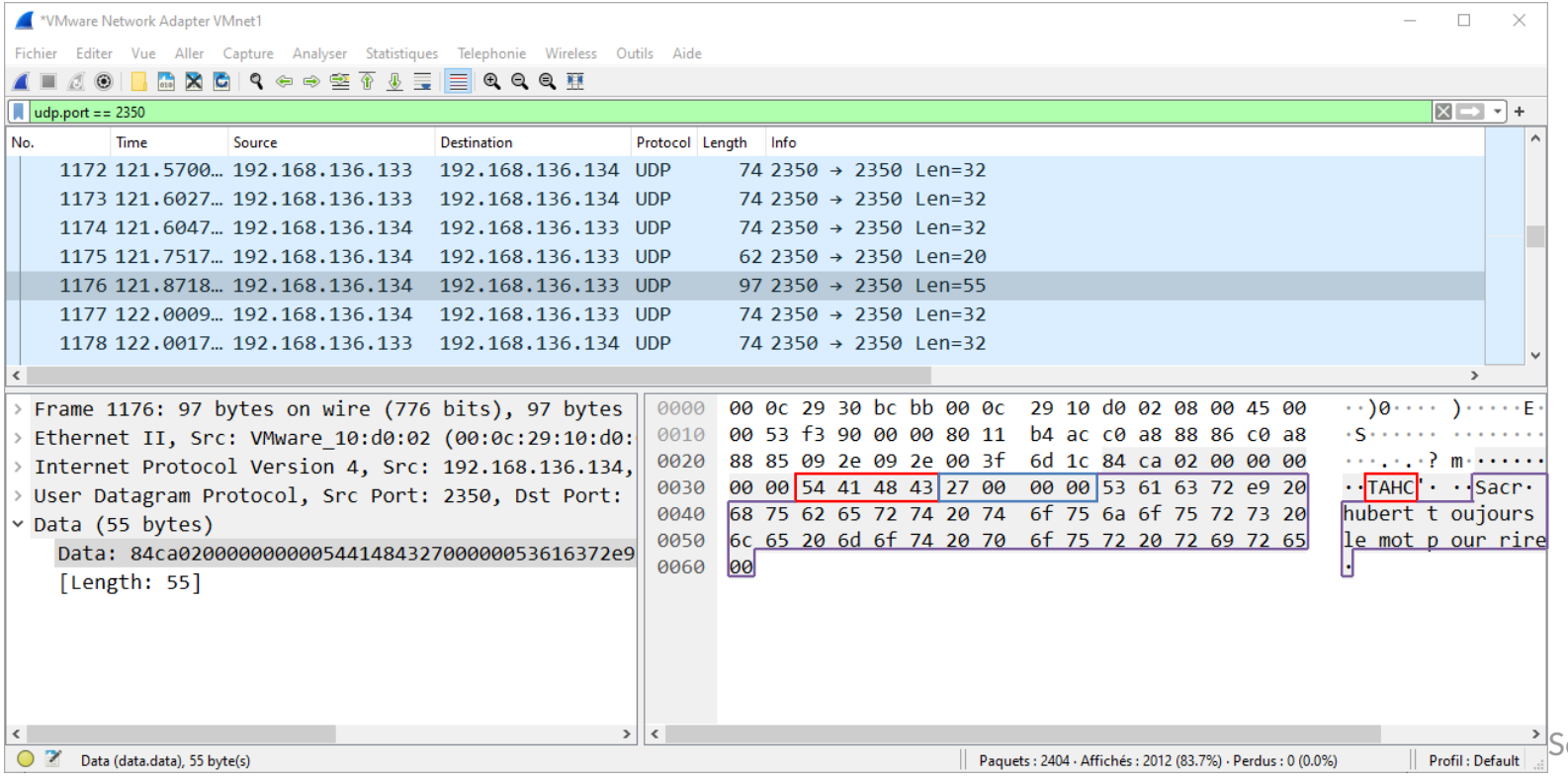
- Non applicable pour les programmes compilé en 32 bits

Quick Win

- Session multijoueur
- Ecrire un message dans le chat

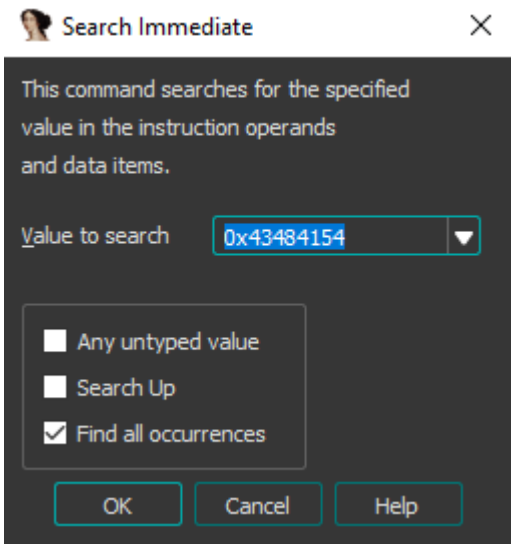
Quick Win

- Retrouver le message dans une capture réseau



Quick Win

- Retrouver l'identifiant de message
- Démarrer la rétro-ingénierie sur le gestionnaire de message




```
case 'CHAT':
  memcpy(ChatTempBuffer, &packet_1->data.ping.field_4, packet_1->data.chat.length);
  ChatTempBuffer[packet_1->data.chat.length] = 0;
  dword_6B89D8 = idTo;
  break;
```

Buffer overflow in .bss, charset without constraint

What is FIDN ?

```
else if ( packet_1->magic == 'FIDN' )
{
  packet_1->magic = 'FRPL';
  len = GetRessourceLength(packet_1->data.chat.data);
  packet_1->data.chat.length = len;
  _send_message(8u, packet_1, idTo);
}
```

Vulnérabilité

- **GetRessourceLength**
  - RReset
    - UnixToWindowsPathAndOpen

```
struct TGSCfile *__cdecl UnixToWindowsPathAndOpen(char *path)
{
  bool v2; // [esp+0h] [ebp-114h]
  char _path[256]; // [esp+4h] [ebp-110h] BYREF
  int pathlen; // [esp+104h] [ebp-10h]
  int i; // [esp+108h] [ebp-Ch]
  bool v6; // [esp+10Fh] [ebp-5h]
  struct TGSCfile *v7; // [esp+110h] [ebp-4h]

  strcpy(_path, path);
  pathlen = strlen(_path);
  for ( i = 0; i < pathlen; ++i )
  {
    if ( _path[i] == '/' )
      _path[i] = '\\';
  }
  v2 = g_CGSCset.next && byte_67DC31;
  v6 = v2;
  _search_unrar_dll();
  v7 = CGSCset::gOpenFile(&g_CGSCset, _path, v6);
  if ( v7 )
    return v7;
  else
    return (struct TGSCfile *)-1;
}
```

Vulnérabilités

- **GetRessourceLength**
  - RReset
    - UnixToWindowsPathAndOpen (stack bof)
      - CGSCset::gOpenFile
        - CGSCarch::GetFileHandle (stack bof)
          - *hash_set_func (stack bof)*

```
struct TGSCfile *__thiscall CGSCarch::GetFileHandle(CGSCarch *this, const char *path)
{
  int v2; // eax
  char _path[64]; // [esp+Ch] [ebp-50h] BYREF
  int v6; // [esp+4Ch] [ebp-10h]
  unsigned int i; // [esp+50h] [ebp-Ch]
  char *v8; // [esp+54h] [ebp-8h]
  unsigned int v9; // [esp+58h] [ebp-4h]

  i = 0;
  v8 = 0;
  v9 = 0;
  memset(_path, 0, sizeof(_path));
  strcpy(_path, path);
  _strupr(_path);
  v6 = hash_set_func(_path);
```

```
int __cdecl hash_set_func(char *String)
{
  char *v1; // eax
  int checksum; // edx
  int v3; // ecx
  char *v4; // ebx
  int v5; // eax
  int v6; // eax
  char v7; // t1
  char Destination[64]; // [esp+4h] [ebp-40h] BYREF

  memset(Destination, 0, sizeof(Destination));
  v1 = _strupr(String);
  strcpy(Destination, v1);
  checksum = 0;
  v3 = 0x10;
  v4 = Destination;
  do
  {
    v5 = *(_DWORD *)v4;
    BYTE1(v5) = *(_DWORD *)v4;
    LOBYTE(v5) = BYTE1(*(_DWORD *)v4);
    v6 = __ROL4__(v5, 0x10);
    v7 = BYTE1(v6);
    BYTE1(v6) = v6;
    LOBYTE(v6) = v7;
    checksum += v6;
    v4 += 4;
    --v3;
  }
  while ( v3 );
  return checksum;
}
```

Contrainte

- **Jeu de caractères**
  - Octets null en fin de chaîne
  - Conversion minuscule en majuscule
- **DMCR.exe base address 0x00400000**
  - Un seul gadget
  - Pas de jmp esp
- **DLLs ne sont pas chargée selon l'ImageBase**

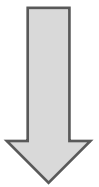
Fonction de hash

- Contrôle EAX en sortie de fonction via le « Path »

```
   mov ecx, 10h
   lea ebx, [ebp + source]
HashLoop:
    mov  eax, [ebx]
    xchg ah, al
    rol  eax, 10h
    xchg ah, al
    add  edx, eax
    add  ebx, 4
    loop HashLoop
    mov eax, edx
```

```
0x00462a36 (6*F ): jmp eax
```

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAA||!-]{)
```



hash_set_func

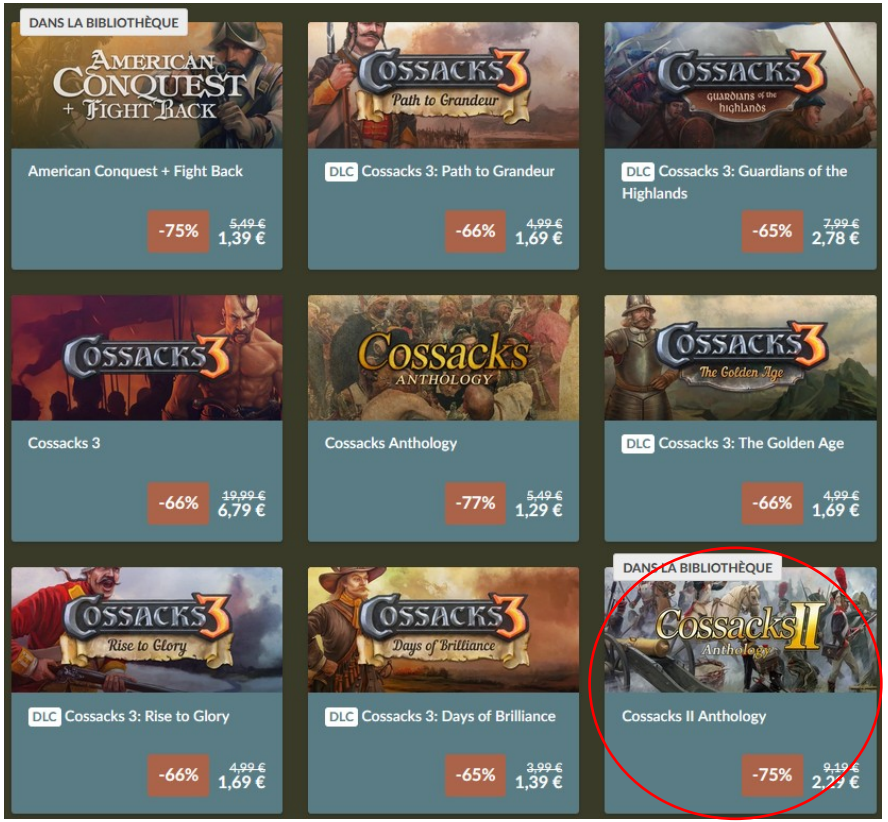
0x006B88D8
(ChatTempBuffer)

Demo

Timeline

- 17/12/2023 PoC
- 24/01/2024 Envoi d'un mail à l'éditeur
- 05/04/2024 Relance éditeur
- 05/04/2024 L'éditeur répond, le bug ne sera pas corrigé

Dommage collatéral



- **Cossacks II**
  - Type – Stratégie
  - Date de sortie - 26 Avril 2005
  - Editeur - GSC Game World

Dommage collatéral

- **Vulnérabilités toujours présentes mais**
  - pas d'archive .GSC
  - stack cookie
- **Cas d'école**

```
buffer          db 260 dup(?)
cookie          dd ?
 s              db 4 dup(?)
 r              db 4 dup(?)
pPath           dd ?
```

Stack View

```
push    ebp
mov     ebp, esp
sub     esp, 11Ch
mov     eax, cookie
mov     [ebp+cookie], eax
mov     eax, [ebp+pPath]
push    eax             ; Source
lea     ecx, [ebp+buffer]
push    ecx             ; Destination
call    _strcpy
add     esp, 8
```

...

```
mov     ecx, [ebp+cookie]
call    checkcookie
mov     esp, ebp
pop     ebp
retn
```

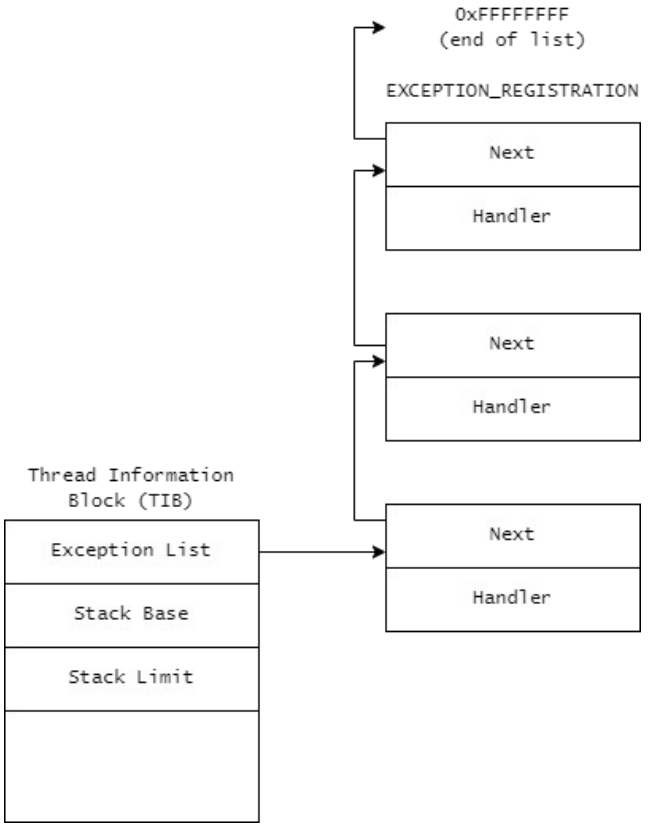
Did you know SEH ?

- Structured Exception Handler (SEH)
- Gestion des exceptions

IDA View-EIP | [18C0] - Structured exception handlers

Address	Name	Stack
01434469	.text:engine_?DecompressData@@YA_NPAPAEPAIP...	0019EF98
0148106D	.text:engine_?DecompressData@@YA_NPAPAEPAIP...	0019EFE4
01436E6D	.text:engine_?DecompressData@@YA_NPAPAEPAIP...	0019F238
0143EA9D	.text:engine_?DecompressData@@YA_NPAPAEPAIP...	0019F6EC
0143EA9D	.text:engine_?DecompressData@@YA_NPAPAEPAIP...	0019FB8C
01445A20	sub_898F00+BACB20	0019FD50
012ED35C	__except_handler3	0019FF60
7751AF30	ntdll.dll:ntdll_wcstombs+70	0019FFCC
77528C28	ntdll.dll:ntdll_RtlCaptureContext+F8	0019FFE4

Debugger > Debugger Windows > SEH List



SEH Exploitation

- Remplacer le *Handler* d'une structure SEH
- Générer une exception
  - Ecrire sur une page Read-Only

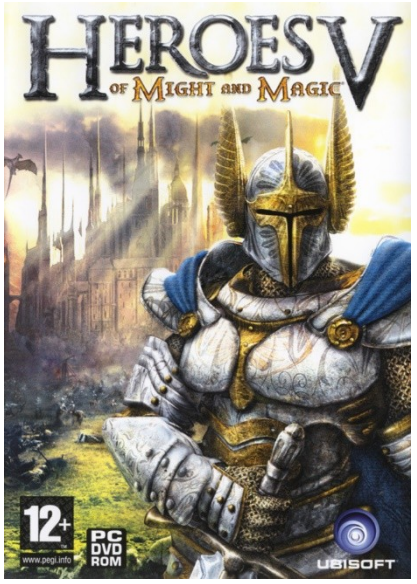
IDA View-EIP | [18C0] - Structured exception handlers | Segments

Name	Start	End	R	W	X	D	L	Align	Base
debug001	00010000	00020000	R	W	.	D	.	byte	0000
debug002	00020000	00022000	R	W	.	D	.	byte	0000
debug003	00022000	00028000	?	?	?	D	.	byte	0000
debug004	00030000	00031000	R	.	.	D	.	byte	0000
debug005	00040000	0005D000	R	.	.	D	.	byte	0000
debug006	00060000	00095000	?	?	?	D	.	byte	0000
debug007	00095000	00098000	R	W	.	D	.	byte	0000
debug008	00098000	000A0000	R	W	.	D	.	byte	0000
debug009	000A0000	0017E000	?	?	?	D	.	byte	0000
Stack_PAGE_GUARD[00...	0017E000	00180000	R	W	.	D	.	byte	0000
Stack[000018C0]	00180000	001A0000	R	W	.	D	.	byte	0000
debug010	001A0000	001A4000	R	.	.	D	.	byte	0000
debug011	001B0000	001B2000	R	W	.	D	.	byte	0000
debug012	001C0000	001C1000	R	.	.	D	.	byte	0000
debug013	001D0000	001D1000	R	.	.	D	.	byte	0000
debug014	001E0000	001EF000	R	W	.	D	.	byte	0000

Demo

Conclusion

- Next …



https://www.linkedin.com/company/synacktiv
https://twitter.com/synacktiv
Nos publications sur : https://synacktiv.com