La Rétroconception

Application à l'analyse logicielle

Serge LEFRANC

La Rétroconception

Introduction

La Rétroconception

Introduction

La rétroconception peut se définir comme l'action d'extraire une connaissance ou un savoir d'une réalisation.

- Bien que son application à l'informatique soit récente, cela fait longtemps qu'elle est utilisée dans des domaines industriels comme l'électronique ou l'automobile (« analyse de la concurrence »…).
- Elle permet l'accession à un savoir ou à une technologie à moindre frais…
- … C'est donc le monde du logiciel commercial qui a le plus été touché par ce phénomène, ceci explique la forte présence de rétroconcepteur dans le monde Windows.
- Une évolution commence à se faire sentir car des compétences ont été acquises dans le monde du logiciel libre afin d'identifier les mécanismes de fonctionnement et de propagation des différents vers qui ont touchés cette communauté.

La Rétroconception

Introduction

- Le but de cette présentation est de faire un **tour d'horizon** de la rétroconception.
- Cet exposé va donc tenter de montrer, après avoir rappelé brièvement le contexte juridique, que les objectifs de la rétroconception sont nombreux et dépendent des acteurs qui vont la réaliser.
- Nous verrons également qu'il est difficile de définir une méthodologie car le spectre des objectifs est trop large (du plus simple au plus compliqué…).
- Ceci nous conduiras à présenter quelques schémas de protections et les méthodes de contournement associées.
- Pour finir nous illustrerons notre propos avec un exemple simple.

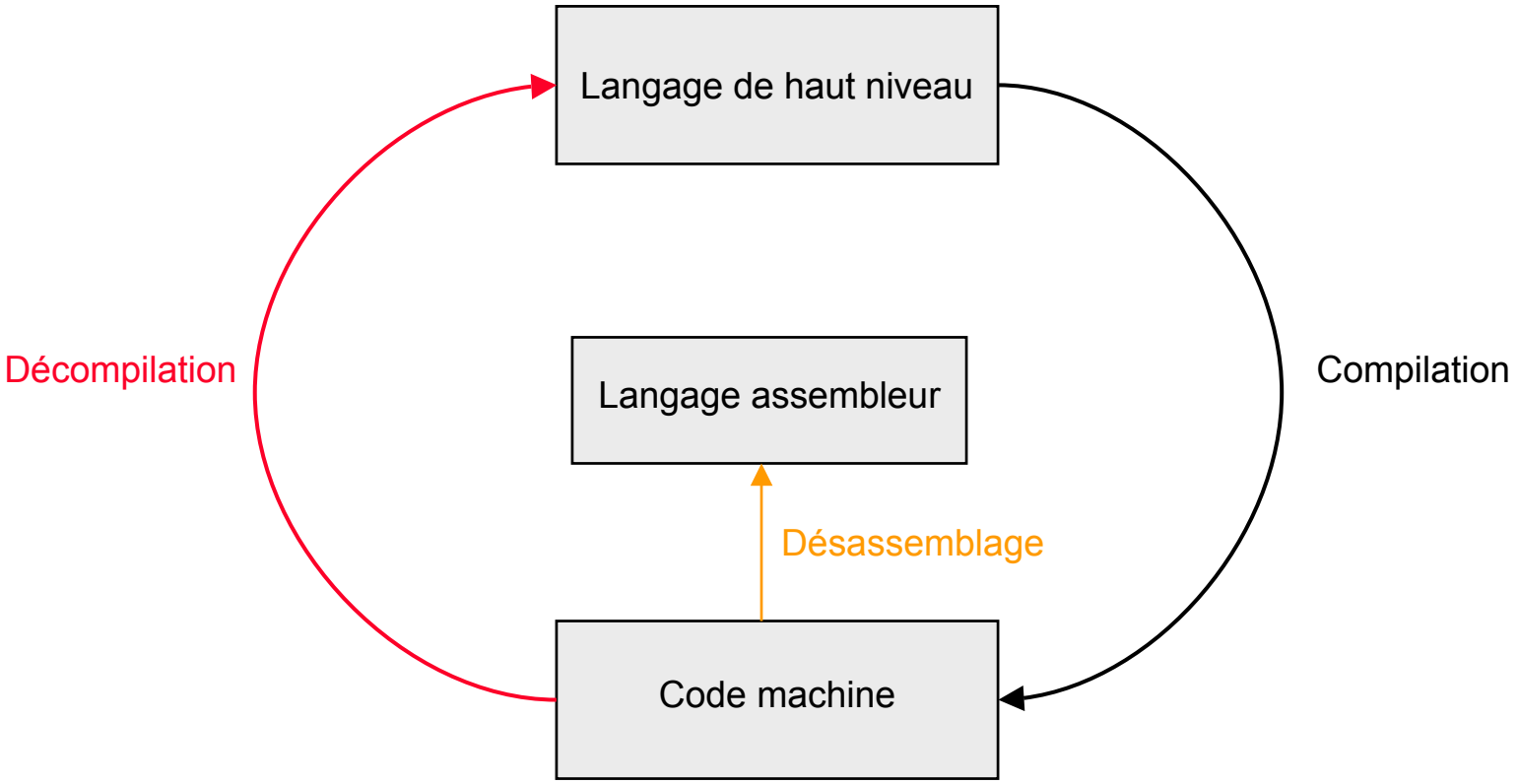
La Rétroconception

Plan de l'exposé

- Introduction
- Aspect juridique
- Les objectifs de la rétroconception
- Méthodologie
- Les outils
- Quelques protections
- Les mesures de contournements associées
- Exemple
- Conclusion

La Rétroconception

Principe



La Rétroconception

Aspects juridiques

basé sur le cours de Me Gérard Haas:
« L'univers numérique et le droit » ENSTA 2001

La Rétroconception

Aspects juridiques (cours de Me Gérard Haas) 1/4

Droit de décompilation 1/3

- *« L'utilisateur régulier d'un logiciel peut, sans autorisation de l'auteur, procéder à **la reproduction et à la traduction** de la forme de son code, c'est à dire sa « décompilation », lorsque ces opérations sont indispensables pour obtenir les informations nécessaires à **l'interopérabilité** de ce logiciel avec d'autres logiciels »* (CPI art L.122-6-1-IV*).*

- La notion de décompilation a été introduite par la directive européenne du 14 mai 1991.

- **L'objectif** clairement affirmé du législateur est de permettre l'interopérabilité des logiciels, définie par la directive comme la *« capacité des programmes d'ordinateurs d'échanger des données et d'utiliser des données qui ont été échangées ».*

Aspects juridiques (cours de Me Gérard Haas) 2/4

Droit de décompilation 2/3

- Le droit de décompiler demeure cependant enfermé dans des conditions très strictes:
  - Il doit s'agir d'un **logiciel créé de façon indépendante** et qui donc n'est pas déjà conçu pour être compatible avec d'autres logiciels.
  - Seuls sont **habilités à procéder** à des actes de **décompilation** les licenciés ou utilisateurs réguliers du logiciel ou les tiers agissant pour leur compte.
  - Les **informations** nécessaires à l'interopérabilité ne sont pas déjà facilement et rapidement **accessibles**.
  - Les **actes de décompilation** sont **limités** aux parties du logiciel d'origine nécessaires à l'interopérabilité.

Il convient d'être réservé sur la portée pratique de cette dernière limitation. En effet il est impossible de savoir la plupart du temps où se trouvent les interfaces dans un programme… La preuve d'un abus s'annonce délicate, sauf si le concepteur a donné des indications suffisantes.

La Rétroconception

Aspects juridiques (cours de Me Gérard Haas) 3/4

Droit de décompilation 3/3

- De plus, les dispositions permettant la décompilation n'autorisent pas pour autant que les **informations obtenues**:
  - Soient utilisées à des fins autres que l'interopérabilité des logiciels avec d'autres logiciels.
  - Soient communiquées à des tiers sauf si cela est nécessaire pour permettre l'interopérabilité avec le logiciel crée de façon indépendante.
  - Soient utilisés pour la mise au point, la production ou la commercialisation d'un logiciel dont l'expression est fondamentalement similaire ou pour tout autre acte portant atteinte au droit d'auteur.

Aspects juridiques (cours de Me Gérard Haas) 4/4

Droit d'analyse

- *« L'utilisateur régulier d'un logiciel se voit reconnaître expressément le droit, sans avoir à demander l'autorisation à l'auteur,* ***d'observer, étudier ou tester*** *le fonctionnement de ce logiciel afin de déterminer* ***les idées et principes*** *qui sont à la base de n'importe quel élément du programme, lorsqu'il effectue des opérations de chargement, d'affichage, de passage d'informations, de transmission ou de stockage du programme »* (CPI art. L.122-6-1-111).

- Le droit d'analyse ne peut être exercé que dans le cadre de l'utilisation normale du logiciel.

- A la différence du droit de décompilation, le droit d'analyse est réservé à **l'utilisateur** du logiciel, qui ne peut autoriser un tiers à l'exercer à sa place.

La Rétroconception

Objectifs de la rétroconception

Objectifs de la rétroconception

- **Ils sont nombreux…**
  - Interopérabilité.
  - Vérification de la présence de fonctions cachées.
  - Piratage logiciel.
  - Espionnage industriel.
  - …
- **… Mais vont dépendre des acteurs qui la réalise**
  - Une entreprise.
  - Un particulier.
  - Un groupe de pirates.
  - Une organisation criminelle.
  - …

Objectifs de la rétroconception

Une entreprise

- Permettre l'interopérabilité entre plusieurs logiciels différents.
- Vérifier l'absence de vulnérabilités dans un produit.
- Accéder aux technologies mises au point par des concurrents…

Un particulier

- Enlever les mécanismes de protection lui interdisant l'utilisation de ce logiciel sans s'être acquitté des droits de licence.
- Modifier certaines fonctions pour un usage particulier.

Objectifs de la rétroconception

Une organisation criminelle

- Permettre l'utilisation du logiciel sans l'avoir acquis et de distribuer à grande échelle, et de façon payante, la version piratée du logiciel.
- A des fins idéologiques ou terroristes.

Un groupe de pirates

- Enlever les protections logicielles d'un produit et le distribuer.
- Découvrir des vulnérabilités pour les exploiter et les distribuer.

Contrairement à l'organisation criminelle, le groupe de pirate ne travaille que pour la renommée. Son profit n'est pas financier.

La Rétroconception

Méthodologie

Méthodologie

- **Il n'existe pas réellement de méthodologie!**
  - Sinon il existerait des logiciels tout faits « clé en main »…
  - Le panel des objectifs visées étant très large, les divers acteurs ne cherchent pas forcément la même chose. Cela a un impact sur la façon dont il faut travailler.
  - Il existe un spectre large de difficulté:
    - Du plus simple… : on sait ce que l'on cherche (enlever une fenêtre pop-up).
    - … Au plus dur : on ne sait pas ce que l'on cherche (vérifier l'absence de porte cachées, identifier un mécanisme de sécurité non documenté…).

- **Il est cependant possible d'isoler un semblant de méthode…**
  - Il existe des logiciels particuliers qui vont nous permettre d'accéder à l'information recherchée.
    - Un désassembleur.
    - Un debugger.
    - Des logiciels de surveillance système.

La Rétroconception

Les outils

Les outils

- **Il existe deux catégories principales:**
  - Les outils de rétroconception proprement dit.
  - Les outils de surveillance système.

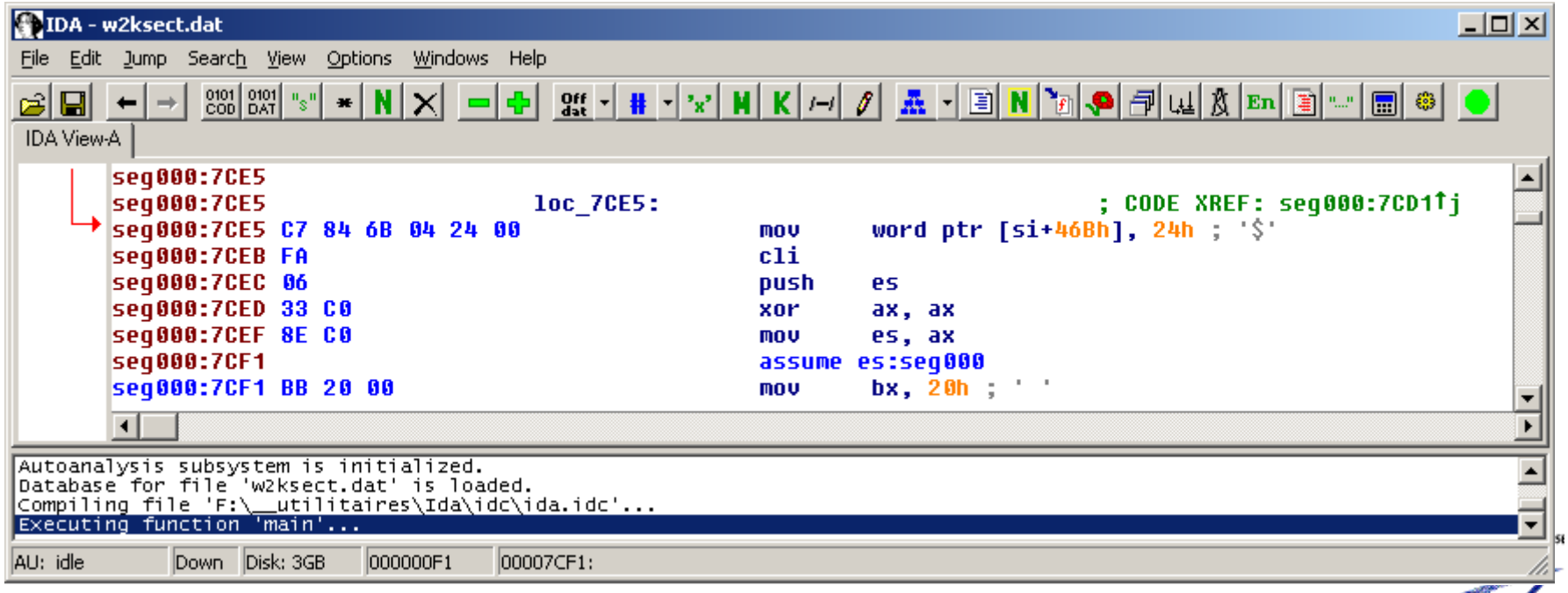
Les outils

Les outils de rétroconception

- Ils permettent d'avoir une vision du fonctionnement du programme.
- Il devient alors possible d'étudier les processus internes du logiciel, de comprendre ses faiblesses.
- Il en existe deux catégories:
  - Les **désassembleurs**.
  - Les **débuggeurs**.

Les outils

- **Les outils de rétroconception : les désassembleurs**
  - Ils permettent d'étudier de façon statique ce que réalise le programme en traduisant l'exécutable en suite d'instructions assembleur (IDA Pro, W32DASM…).



Les outils

- **Les outils de rétroconception : les débuggeurs**
  - Ils permettent d'analyser ce que fait le programme au moment de son exécution.
  - Le débuggeur est capable de tracer l'exécution du programme pas à pas, de connaître la valeur des registres, de placer des points d'arrêts à certains endroits ou en fonction de certaines conditions (breakpoint: pause dans l'exécution de l'application).
  - Il existe deux types de débuggeur sous Windows: les débuggeurs de niveau **applicatif** ou ceux de niveau **système**.
  - Les débuggeurs de niveau applicatif résident entre l'OS et le programme et que ceux de niveau système résident entre le processeur et l'OS (ces derniers sont plus puissants car ils peuvent débugger au niveau noyau).

Les outils

Les outils de surveillance système

- Ils vont nous permettre de connaître les interactions du programme avec son environnement de fonctionnement.
- Deux outils sont particulièrement utiles: **FileMon** et **RegMon** de la société SysInternals.
- **FileMon** permet de surveiller l'activité du système de fichiers en temps réel.
- **RegMon** permet de connaître quel processus est en train d'accéder à la base des registres et ce qui y est fait.
- Il est également possible, dans certains cas, d'utiliser un sniffer afin de surveiller le trafic réseau généré par une application…

La Rétroconception

Les protections et les mesures de contournement associées

Les protections et les mesures de contournement associées

Quelques protections

Quelques protections

- A l'origine faite pour protéger le logiciel de la copie ou de l'utilisation non-licencié, la plupart des mesures de protections ont rapidement été contournées par l'utilisation de la rétroconception.

- Il est donc apparu nécessaire de les faire évoluer en leur incorporant des mécanismes anti-rétroconception…

Quelques protections anti-copie

Les tokens logiciels

C'est une des méthodes les plus utilisées de protection logicielle. Ils peuvent prendre différentes formes.

- Clé d'enregistrement.
- Serial multiple.
- Serial/Nom.
- Fichier de clé.

Les tokens matériels

Étant donné le caractère volatile de l'information numérique et la facilité de sa reproduction, les tokens matériels ont été inventés pour pallier ces problèmes en misant sur le fait qu'une « clé » physique est plus difficile à dupliquer…

- Disquette clé.
- Dongle.
- Carte à puce.

Quelques protections anti-copie

- **Les NAGs screen**
  - Fenêtres « pop-up » qui annoncent à l'utilisateur qu'il peut utiliser le programme mais qu'il n'en possède pas la licence.
- **Les limites**
  - Limites dans l'utilisation du logiciel: sur le nombre de fois ou le programme peut être utilisé, sur la durée d'utilisation… Le programme ne pourra plus se lancer une fois la limite atteinte.
- **Les cripplewares**
  - Limitations dans les fonctionnalités auxquelles l'utilisateur a accès. Elles sont débloquées lorsqu'il a acquis la licence d'utilisation du logiciel.

Quelques protections anti-retro

Le chiffrement de code

- C'est la façon la plus simple de protéger le code ASM d'un programme.
- Le même type de technique est utilisée par les virus chiffrés.

```
   mov EAX, count
   mov EBX, address
   mov ECX, offset
_LOOP:
   xor [ECX], EBX
   DEC EAX
   ja _LOOP
   ; ensuite on trouve le code et les
   ; données chiffrés  du programme
```

Quelques protections anti-retro

Le packing d'exécutable

A l'origine utilisé pour compresser le programme tout en lui permettant d'être exécutable (décompression en mémoire).

- Permet de « brouiller » l'exécutable tout en réduisant sa taille.

Quelques protections anti-retro

Le camouflage de code (obfuscation)

Permet de rendre le code du logiciel incompréhensible, tout en le laissant fonctionnel.

- Le but est de rendre vaine toute tentative de décompilation en ne permettant pas à un utilisateur de récupérer de l'information pertinente sur le programme de façon rapide.

- Il existe différentes façons de rendre cela possible:
  - **Transformation lexicale**: on remplace le nom des classes et des fonctions par d'autres qui ne veulent rien dire (à priori…).
  - **Transformation de contrôle**: on ajoute des conditions opaques. Par exemple, on change les instructions suivantes : par :

```
instruction_a
instruction_b
```

```
instruction_a
if (p == TRUE)
    instruction_b
else
    instruction_b
```

Quelques protections anti-retro 7/7

Les fonctions anti-debugging

- Méthodes ayant pour but de rendre l'utilisation du débuggeur plus compliquée (elle n'est jamais impossible…).

- Deux principales méthodes:
  - **Actions préventives**: ce sont des actions qui sont effectuées par le programme afin de ne pas permettre à l'utilisateur de le tracer durant son exécution (masquage des interruptions, vérification de la présence d'un débuggeur…).
  - **Code auto-modifiant**: algorithme de chiffrement/déchiffrement...

La Rétroconception

Les protections et les mesures de contournement associées

Les mesures de contournement

Les mesures de contournement 1/5

Un exemple simple (énoncé) 1/2

- Les schémas de protection présentés sont très simples à mettre en échec si ils n'incorporent pas des mécanismes de chiffrement/obfuscation car ils font souvent appel au modèle suivant:

```
result = security_check(condition1, condition2)
if (result == TRUE)
  then <autorise et exécute le programme>
else <essaye encore>
```

- La `condition1` peut être le numéro de série entré par l'utilisateur et la `condition2` le numéro de série qui est valide.

- La fonction `security_check` peut aller de la simple comparaison de bits à une requête I/O…

Les mesures de contournement

Un exemple simple (énoncé)

- Le bout de code précédent se traduirait en ASM:

```
push condition2
push condition1
call security_check
test EAX, EAX
jnz address1 ; adresse du début du programme
<essaye encore...>
```

- Le résultat de l'appel à la fonction est stocké dans le registre `EAX`. L'opération `TEST` réalise un ET logique. Si le résultat est 0 (FALSE), l'opération de ET logique va positionner le flag Z à 1 dans le registre de flags et le saut ne va pas avoir lieu.

Les mesures de contournement

Utilisation d'un débuggeur

- En positionnant certains points d'arrêt (appel de fonctions de comparaison de chaînes de caractères par exemple), il va par exemple être possible à l'utilisateur:
  - De récupérer le serial (stocké en dur dans le programme: `condition2`).
  - De modifier l'instruction JNZ en JMP, le saut vers le début du programme aura toujours lieu.
  - En modifiant l'appel à la fonction `security_check` par des NOP, il ne va rien se passer.
  - Si la clé n'est pas stockée dans le programme, il est possible d'extraire l'algorithme de génération de clé afin de fabriquer un générateur de clés (keygen).

Les mesures de contournement

- **Utilisation d'un désassembleur**
  - L' appel au débuggeur n'est pas systématique, parfois une analyse statique du programme peut se révéler suffisante.
  - En regardant les références à des chaînes de caractères dans le code désassemblé, l'utilisateur peut remonter jusqu'a la routine `security_check` et la « corriger ».

Les mesures de contournement

- **Unpacking d'exécutable**
  - De la même façon qu'il existe des programmes pour « packer » des exécutables, il en existe d'autres pour les « unpacker »!
  - Au moment de l'exécution, le programme est dépacké/déchiffré en mémoire…
- **Modification de l'exécutable en mémoire**
  - Si l'unpacking est difficile ou la routine de chiffrement est compliquée et si l'utilisateur a pu localiser en mémoire les instructions en clair, il sera possible de les modifier directement.

La Rétroconception

Exemple

Modification du programme `netstat.exe` de Windows 2000

Étude de cas

But

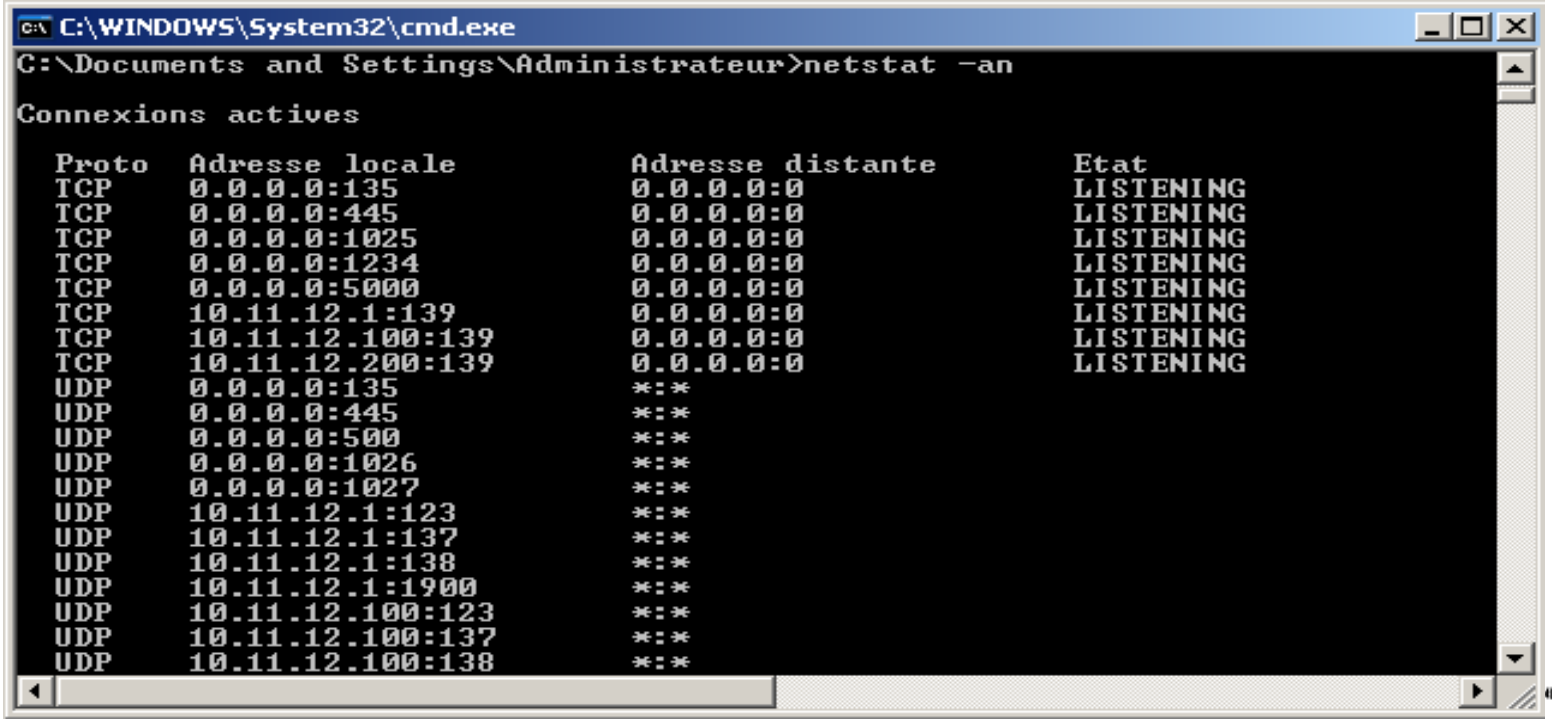
Dissimuler les connexions réseau établies sur une machine Windows 2000.

- Pour cela nous allons modifier le programme `netstat.exe` de Win2K car c'est lui (notamment…) qui permet à un administrateur de connaître l'état des connexions de sa machine.
- Nous allons analyser le fonctionnement de ce programme et son comportement lorsqu'une connexion est ouverte sur la machine, puis nous allons essayer d'identifier la ou les fonctions responsables de l'affichage.
- Nous réfléchirons à une façon de modifier le code du programme qui nous permettra de sélectionner les lignes à afficher.
- Cette analyse s'inspire en partie du travail effectué par `threat` sur `nethide`.

La Rétroconception

Analyse de netstat.exe

- On lance une fenêtre `cmd.exe` sous Windows et on tape la commande `netstat.exe -an` pour avoir la liste des connexions ouvertes.



Analyse de netstat.exe

- On sélectionne un port sur lequel un service est dans l'état « listenning ».
- On se connecte sur ce service via le client telnet:
  `telnet localhost port_selectionne`
- On relance la commande netstat pour voir la présence d'une nouvelle ligne indiquant l'établissement d'une connexion sur la machine locale.

Méthodologie

- On recherche des fonctions permettant l'affichage et/ou le formatage d'une chaîne de caractères.
  - On désassemble l'exécutable avec IDA pro (ou DASM32) et on regarde les fonctions « intéressantes » qui sont importées par le programme.
    - `FormatMessageA`
    - `CharToOemBuffA`
    - `fprintf`
    - `sprintf`

Méthodologie

- Identification de celle responsable de l'affichage via le MSDN Library.
  - `FormatMessageA`
- Prototype de cette fonction:

```
DWORD FormatMessage ( DWORD dwFlags,
                      LPCVOID lpSource,
                      DWORD dwMessageId,
                      DWORD dwLanguageId,
                      LPTSTR lpBuffer,
                      DWORD nSize,
                      va_list *Arguments);
```

Méthodologie

- La fonction `FormatMessageA` est appelée par différentes procédures, il faut déterminer celle qui est responsable de l'affichage.
  - On utilise SoftICE et on place un point d'arrêt sur la fonction `FormatMessageA`.
  - On quitte SoftICE et on relance netstat. SoftICE apparaît et on trace le programme avec F12 jusqu'à voir les lignes correspondant au programme `netstat.exe` apparaître dans notre fenêtre `cmd.exe`.
    - On remarque un `CALL` récurrent à l'adresse `0100305B` chaque fois qu'une nouvelle ligne s'affiche…
    - C'est donc l'adresse de la fonction d'affichage!

Analyse de la fonction d'affichage 1/2

- On va à l'adresse `0100305B` sous IDA pro afin de regarder ce que fait la fonction d'affichage.
- On la parcourt et on remarque différents appels à des fonctions d'impression et de manipulation de chaînes de caractères que nous avons identifiées comme étant intéressantes pour réaliser nos modifications.
- On essaye de lire et de comprendre le code afin de trouver un endroit que nous pourrons modifier…
- **Remarques**: Il était possible de ne pas utiliser le débuggeur pour ce travail car la fonction `FormatMessageA` n'est appelée que par deux procédures différentes. Il était très simple d'identifier celle responsable de l'affichage de la ligne souhaitée. Cependant ce n'est pas toujours le cas…

La Rétroconception

```
.text:0100305B
.text:0100305B                 Arguments       = dword ptr -8
.text:0100305B                 lpszDst         = dword ptr -4
.text:0100305B                 arg_0           = dword ptr  8
.text:0100305B                 dwMessageId     = dword ptr  0Ch
.text:0100305B                 arg_8           = byte ptr  10h
.text:0100305B
.text:0100305B 55                              push    ebp
.text:0100305C 8B EC                           mov     ebp, esp
.text:0100305E 51                              push    ecx
.text:0100305F 51                              push    ecx
.text:01003060 8D 45 10                        lea     eax, [ebp+arg_8] ; Load Effective Address
.text:01003063 53                              push    ebx
.text:01003064 89 45 F8                        mov     [ebp+Arguments], eax
.text:01003067 8D 45 F8                        lea     eax, [ebp+Arguments] ; Load Effective Address
.text:0100306A 56                              push    esi
.text:0100306B 50                              push    eax             ; Arguments
.text:0100306C 33 F6                           xor     esi, esi        ; Logical Exclusive OR
.text:0100306E 8D 45 FC                        lea     eax, [ebp+lpszDst] ; Load Effective Address
.text:01003071 56                              push    esi             ; nSize
.text:01003072 50                              push    eax             ; lpBuffer
.text:01003073 56                              push    esi             ; dwLanguageId
.text:01003074 FF 75 0C                        push    [ebp+dwMessageId] ; dwMessageId
.text:01003077 56                              push    esi             ; lpSource
.text:01003078 68 00 09 00 00                  push    900h            ; dwFlags
.text:0100307D FF 15 04 10 00+                 call    ds:FormatMessageA ; formate la ligne qui va etre plasse a stdout
.text:01003083 8B D8                           mov     ebx, eax
.text:01003085 3B DE                           cmp     ebx, esi        ; Compare Two Operands
.text:01003087 75 04                           jnz     short loc_100308D ; Jump if Not Zero (ZF=0)
.text:01003089 33 C0                           xor     eax, eax        ; Logical Exclusive OR
.text:0100308B EB 5A                           jmp     short loc_10030E7 ; Jump
.text:0100308D                 ; ---------------------------------------------------------------------------
.text:0100308D
```

La Rétroconception

```
.text:0100308D
.text:0100308D                          loc_100308D:                            ; CODE XREF: sub_100305B+2C↑j
.text:0100308D A1 3C 10 00 01                       mov     eax, ds:_iob
.text:01003092 83 7D 08 02                          cmp     [ebp+arg_0], 2      ; Compare Two Operands
.text:01003096 57                                   push    edi
.text:01003097 8D 70 40                             lea     esi, [eax+40h]      ; Load Effective Address
.text:0100309A 74 03                                jz      short loc_100309F ; Jump if Zero (ZF=1)
.text:0100309C 8D 70 20                             lea     esi, [eax+20h]      ; Load Effective Address
.text:0100309F
.text:0100309F                          loc_100309F:                            ; CODE XREF: sub_100305B+3F↑j
.text:0100309F 68 00 80 00 00                       push    8000h
.text:010030A4 FF 76 10                             push    dword ptr [esi+10h]
.text:010030A7 FF 15 5C 10 00+                      call    ds:_setmode         ; Indirect Call Near Procedure
.text:010030AD 8B 7D FC                             mov     edi, [ebp+lpszDst]
.text:010030B0 59                                   pop     ecx
.text:010030B1 59                                   pop     ecx
.text:010030B2 33 C0                                xor     eax, eax            ; Logical Exclusive OR
.text:010030B4 83 C9 FF                             or      ecx, 0FFFFFFFFh     ; Logical Inclusive OR
.text:010030B7 F2 AE                                repne scasb                 ; Compare String
.text:010030B9 F7 D1                                not     ecx                 ; One's Complement Negation
.text:010030BB 49                                   dec     ecx                 ; Decrement by 1
.text:010030BC 51                                   push    ecx                 ; cchDstLength
.text:010030BD FF 75 FC                             push    [ebp+lpszDst]       ; lpszDst
.text:010030C0 FF 75 FC                             push    [ebp+lpszDst]       ; lpszSrc
.text:010030C3 FF 15 7C 10 00+                      call    ds:CharToOemBuffA   ; Indirect Call Near Procedure
.text:010030C9 FF 75 FC                             push    [ebp+lpszDst]
.text:010030CC 68 3C 12 00 01                       push    offset aS           ; "%s"
.text:010030D1 56                                   push    esi
.text:010030D2 FF 15 38 10 00+                      call    ds:fprintf          ; Indirect Call Near Procedure
.text:010030D8 83 C4 0C                             add     esp, 0Ch            ; Add
.text:010030DB FF 75 FC                             push    [ebp+lpszDst]
.text:010030DE FF 15 00 10 00+                      call    ds:LocalFree        ; Indirect Call Near Procedure
.text:010030E4 8B C3                                mov     eax, ebx
.text:010030E6 5F                                   pop     edi
.text:010030E7
.text:010030E7                          loc_10030E7:                            ; CODE XREF: sub_100305B+30↑j
.text:010030E7 5E                                   pop     esi
.text:010030E8 5B                                   pop     ebx
.text:010030E9 C9                                   leave                       ; High Level Procedure Exit
.text:010030EA C3                                   retn                        ; Return Near from Procedure
.text:010030EA                          sub_100305B endp
.text:010030EA
```

Analyse de la fonction d'affichage

- Après l'appel à cette fonction, il y a une suite de 5 instructions très importante:

```
.text:01003083    mov  ebx, eax
.text:01003085    cmp  ebx, esi
.text:01003087    jnz  short loc_100308D
.text:01003089    xor  eax, eax
.text:0100308B    jmp  short loc_10030E7
```

- Copie le nombre de caractères écrit dans le buffer dans EBX
- Regarde si la chaîne de caractères est vide (valeur renvoyé par la fonction `FormatMessageA`).
- Si le nombre de caractères de la chaîne n'est pas nul, le flot d'instructions se déplace à l'adresse `100308D` (affichage de la chaîne de caractères…).
- Sinon le registre EAX est mis à 0.
- Le flot d'instruction se déplace à la fin du programme (rien à afficher).

Modification de la fonction d'affichage 1/2

- Donc si il n'y a rien à afficher, on va directement à la fin du programme, adresse `010030E7`.
- Sinon on se déplace à l'adresse `0100308D`, 4 octets plus loin dans le flot d'instructions.
- Afin de modifier les instructions pour que plus rien ne soit affiché par le programme, et cela, même si il y a quelque chose dans la chaîne de caractères, il suffit de modifier l'adresse du saut conditionnel afin qu'il désigne le saut qui amène à la fin du programme!
- Il suffit de modifier:

```
.text:01003087 75 04   jnz  short loc_100308D
```

par:

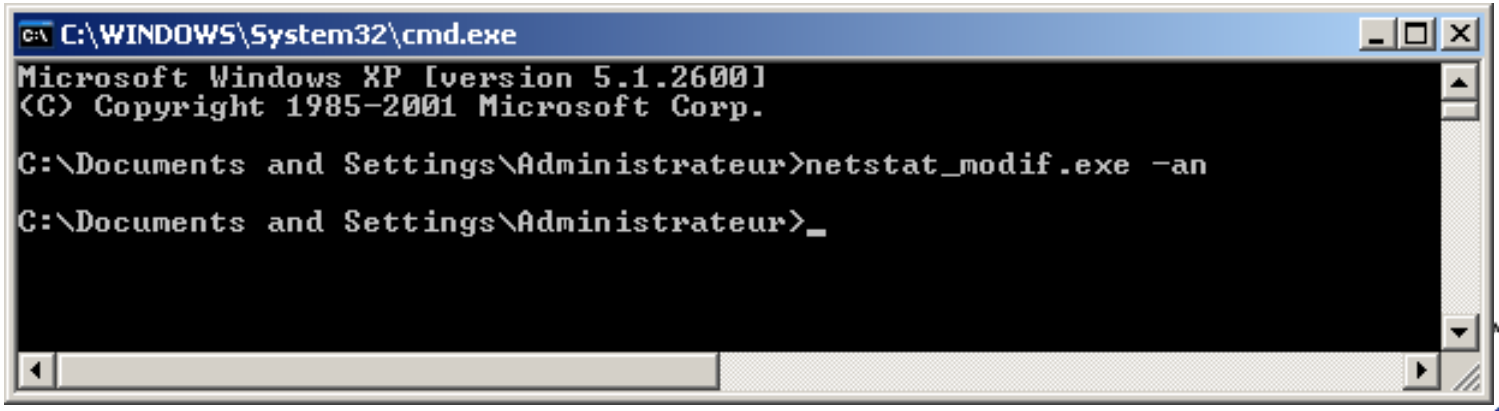
```
.text:01003087 75 02   jnz  short loc_100308B
```

- On change la valeur d'un seul octet et le programme ne va plus rien afficher!

La Rétroconception

Modification de la fonction d'affichage 2/2

- Il ne reste plus qu'à utiliser un éditeur hexadécimal (HEXedit fera parfaitement l'affaire…).
- On recherche la chaîne octale `3B DE 75 04 33 C0. 75 04` qui représente notre saut conditionnel ainsi que l'instruction suivante et précédente.
- Il suffit de modifier `75 04` par `75 02` (2 octets moins loin dans le flot d'instructions)…
- On vérifie que plus rien n'est affiché.



La Rétroconception

Conclusion

Conclusion

- Il n'existe pas de méthode miracle pour faire de la rétroconception… Le processus de décompilation ne peut pas être complètement automatisé car ce problème est équivalent a celui « d'arrêt des machines » (halting problem) dont Alan Turing a démontré en 1936 qu'il n'existait pas de solutions ou d'algorithmes qui le résolvent, et ce, pour tous les types de conditions initiales…
- Il est également illusoire de penser mettre au point des techniques de protection contre la rétroconception: une partie du programme sera toujours, à un moment donné, en clair dans la mémoire…
- La modification d'un seul octet sur un total de 28432 permet de changer de façon radicale le fonctionnement d'un programme (cela le rend même complètement inutile dans ce cas précis…).
- Nous avons expérimenté UNE façon de faire… Il était possible de faire autrement en modifiant le programme de façon plus fine en sélectionnant les lignes à ne pas afficher…

Conclusion

- Le domaine de la rétroconception informatique est encore assez jeune comparé à d'autres secteurs industriels (électronique, automobile…).
- Ces techniques sont essentiellement pratiquées par les « crackers » de logiciels avec un fort taux de réussite (à l'heure actuelle, aucune protection n'a été en mesure de résister longtemps à cette communauté d'informaticiens…).
- Cependant, le spectre de connaissances nécessaires est très limité comparé à ce qui est nécessaire pour faire de la recherche de vulnérabilités.
- Le monde du logiciel libre n'a pas facilité le développement d'une expertise internationale comme cela peut être le cas dans d'autres domaines de la sécurité informatique, mais les choses semblent évoluer...