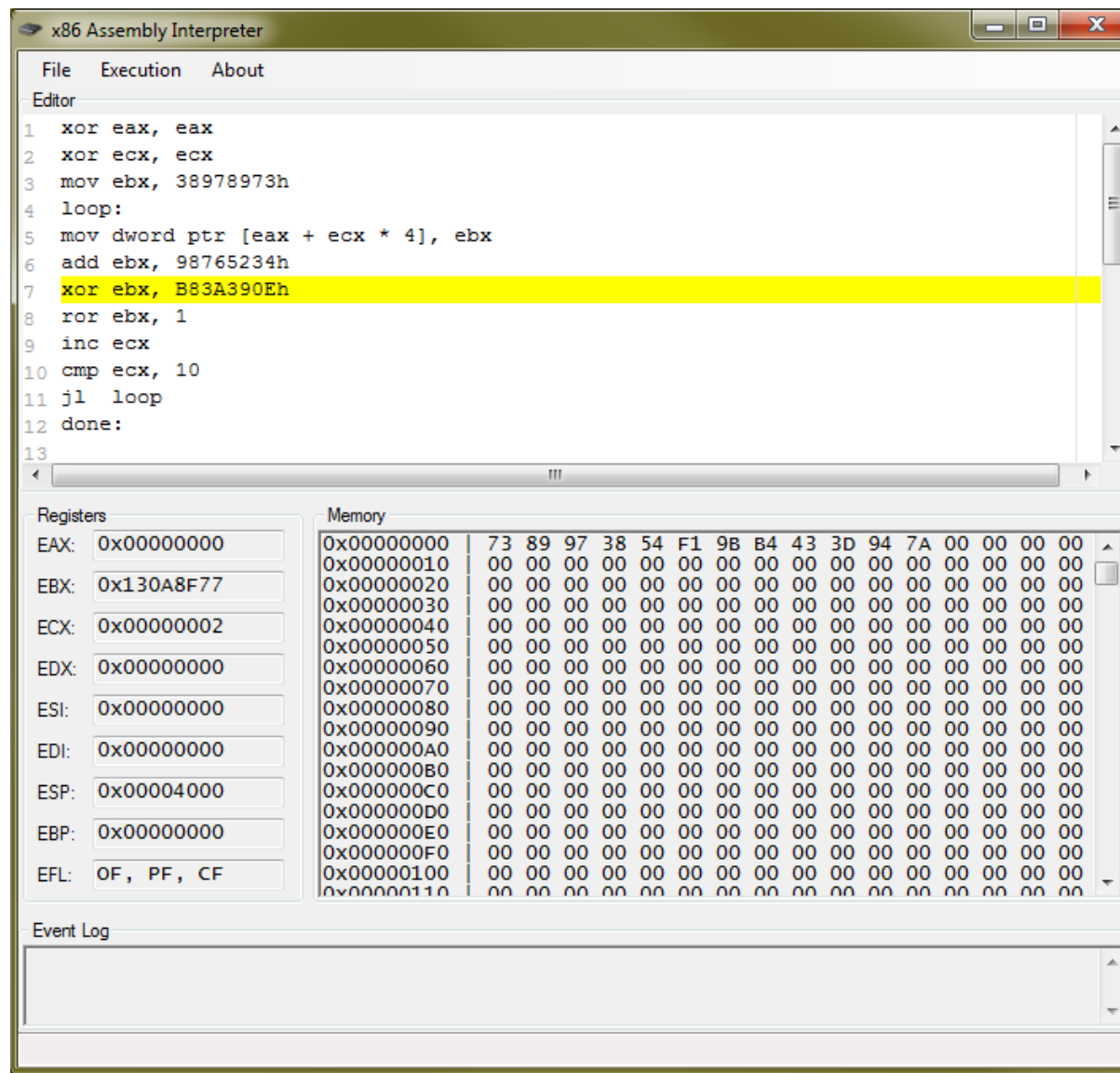


Introduction à la rétro-ingénierie (re)



Ian-Kyle Wagner

Table de contenu

- Décompilation “high level” (Java,.Net,Android)
- Décompilation “Middle level” (C,C++)
- Intro à l’Assembleur x86 et ARM
- Décompilation “Low level” (ASM)
- Outils et techniques pour chaque niveau

Qu'est-ce que la rétro-ingénierie

- Analyser le code source à partir d'un binaire

```
Hopper Disassembler v4
File Edit Find Modify Navigate Scripts Window Help
D A C P U
*xormePPC
1000052c db 0x94 ; DATA XREF=_start+64
1000052d db 0x21 ;
1000052e db 0xff ;
1000052f db 0xd0 ;
; ===== BEGINNING OF PROCEDURE =====
; Variables:
; arg_34: 52
; arg_2C: 44
sub_10000530:
10000530 mflr r0
10000534 stw r0, 52(r1)
10000538 stw r31, 44(r1)
1000053c mr r31, r1
10000540 lis r9, 0x1000 ; 0x10000000
10000544 addi r3, r9, 0x644 ; "Insert key: "
10000548 crclr r6
1000054c bl _edata+72
10000550 lwz r4, 8(r31)
10000554 lis r9, 0x1000 ; 0x10000000
10000558 addi r3, r9, 0x654 ; 0x10000654
1000055c crclr r6
10000560 bl _edata+96
10000564 lwz r3, 8(r31)
10000568 bl verifyFlag ; verifyFlag
1000056c mr r9, r3
10000570 cmplwi cr7, r9, 0x2675
10000574 bne cr7, loc_1000058c
10000578 lis r9, 0x1000 ; 0x10000000
1000057c addi r3, r9, 0x658 ; "Good Job"
10000580 crclr r6
10000584 bl _edata+72
10000588 b loc_1000059c
loc_1000058c:
1000058c lis r9, 0x1000 ; 0x10000000, CODE XREF=sub_10000530+68
10000590 addi r3, r9, 0x664 ; "Wrong Password"
10000594 crclr r6
10000598 bl _edata+72
>>>
```

Pourquoi Re

https://images.duckduckgo.com/iu/?u=http%3A%2F%2Fwww.framboise314.fr%2Fwp-content%2Fuploads%2F2016%2F02%2Fsamba_linux_windoq.jpg&f=1



https://images.duckduckgo.com/iu/?u=http%3A%2F%2Fwww.redeszone.net%2Fapp%2Fuploads%2F2014%2F12%2Fvirlock_foto.png&f=1

This computer was automatically blocked. Reason: Pirated software has been detected.



Willful copyright infringement is a federal crime that carries penalties of up to five years in federal prison, a \$250,000 fine, forfeiture and restitution (17 U.S.C s.506, 18 U.S.C s.2319)

As a first-time offender you are required by law to pay a fine of 250 USD
Hard drive contents, network files have been encrypted and disabled. [View encrypted files](#)
Hard drive contents will be permanently removed from this computer if the fine is not paid.
There are two ways to pay a fine:

- 1.You can pay your fine online through BitCoin. BitCoin is available nationwide. Click the tabs below to find the nearest vendor. Your computer will be unlocked after you make your payment.
- 2.You can come to your local courthouse and pay your fine at the Cashiers window. Your computer will be unlocked within 4-5 working days.

To regain access transfer bitcoins to the following address (click to copy):
1N43vMz9qBlxcBFFzCGnENSmbRE3eXifr

After the payment is finalized enter Transfer ID below.

 Online fine payments are processed by Chase Paymentech.

If the fine is not paid within three days, a warrant will be issued for your arrest, which will be forwarded to your local authorities. You will be charged, fined, convicted for up to 5 years.

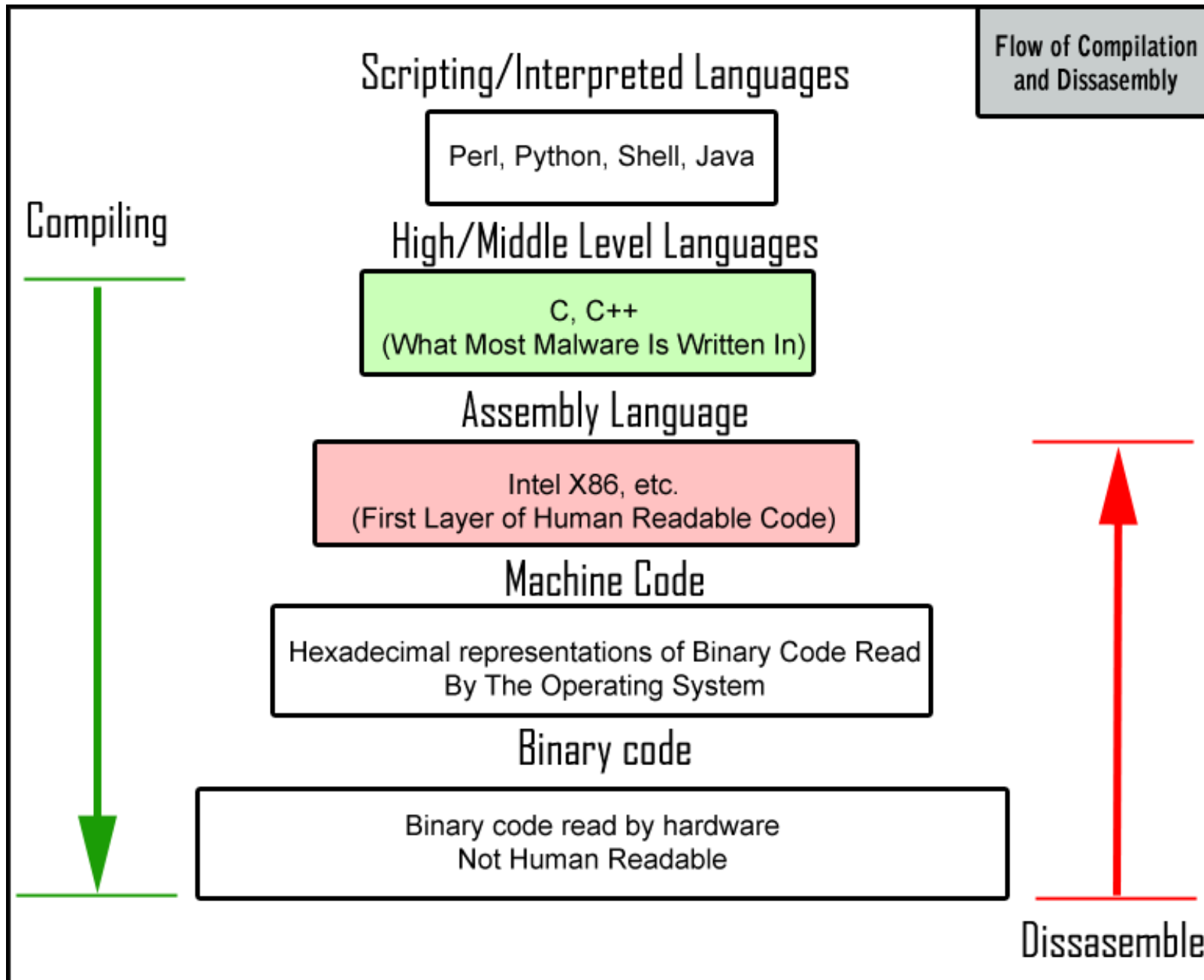
Payment [BitCoin Information](#) [BitCoin Exchanges](#) [BitCoin ATMs](#) [Internet Browser](#) [Notepad](#)

Operation Global III is a coordinated effort by U.S., Canadian, European, Australian, New Zealand and other law enforcement agencies across the globe targeting computers with pirated content.



https://vignette.wikia.nocookie.net/ogoped/ia/images/e/e1/Intel_Itanium_Logo_alpha.svg-1--0.png/revision/latest?cb=20170131230417

Les différents niveaux



Décompilation haut niveau

- Utile pour le java, C#
- Évite les problèmes de désassemblage de plusieurs niveaux .net -> gcc/llvm -> arch
- Plus facile à analyser



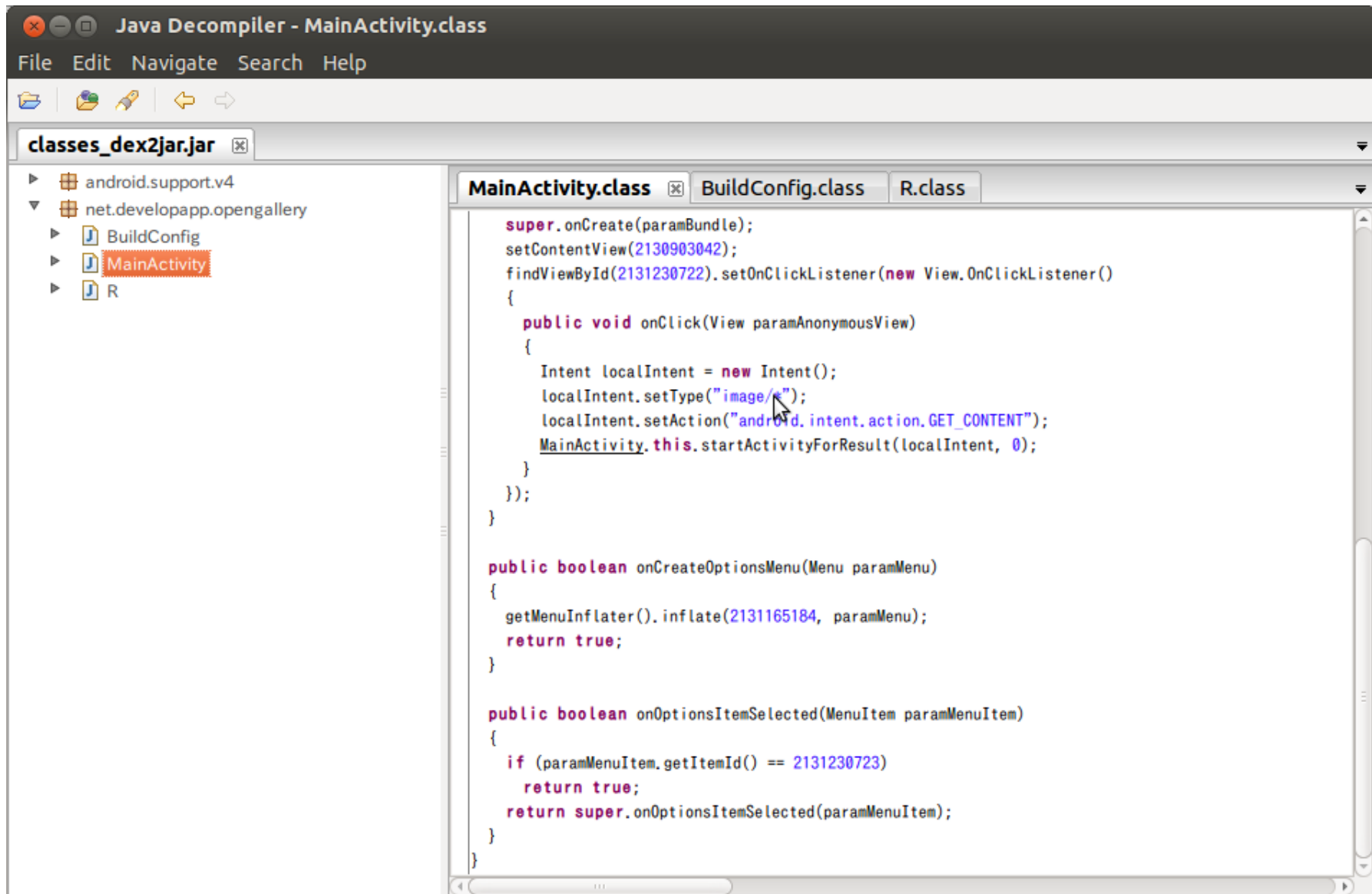
https://images.duckduckgo.com/iu/?u=http%3A%2F%2Fresources.jetbrains.com%2Fstorage%2Fproducts%2Fdotpeek%2Fimg%2Fmeta%2Fdotpeek_250x250.png&f=1



<https://images.duckduckgo.com/iu/?u=https%3A%2F%2Favatars1.githubusercontent.com%2Fuser%2F11619605%3Fv%3D3%26s%3D400&f=1>

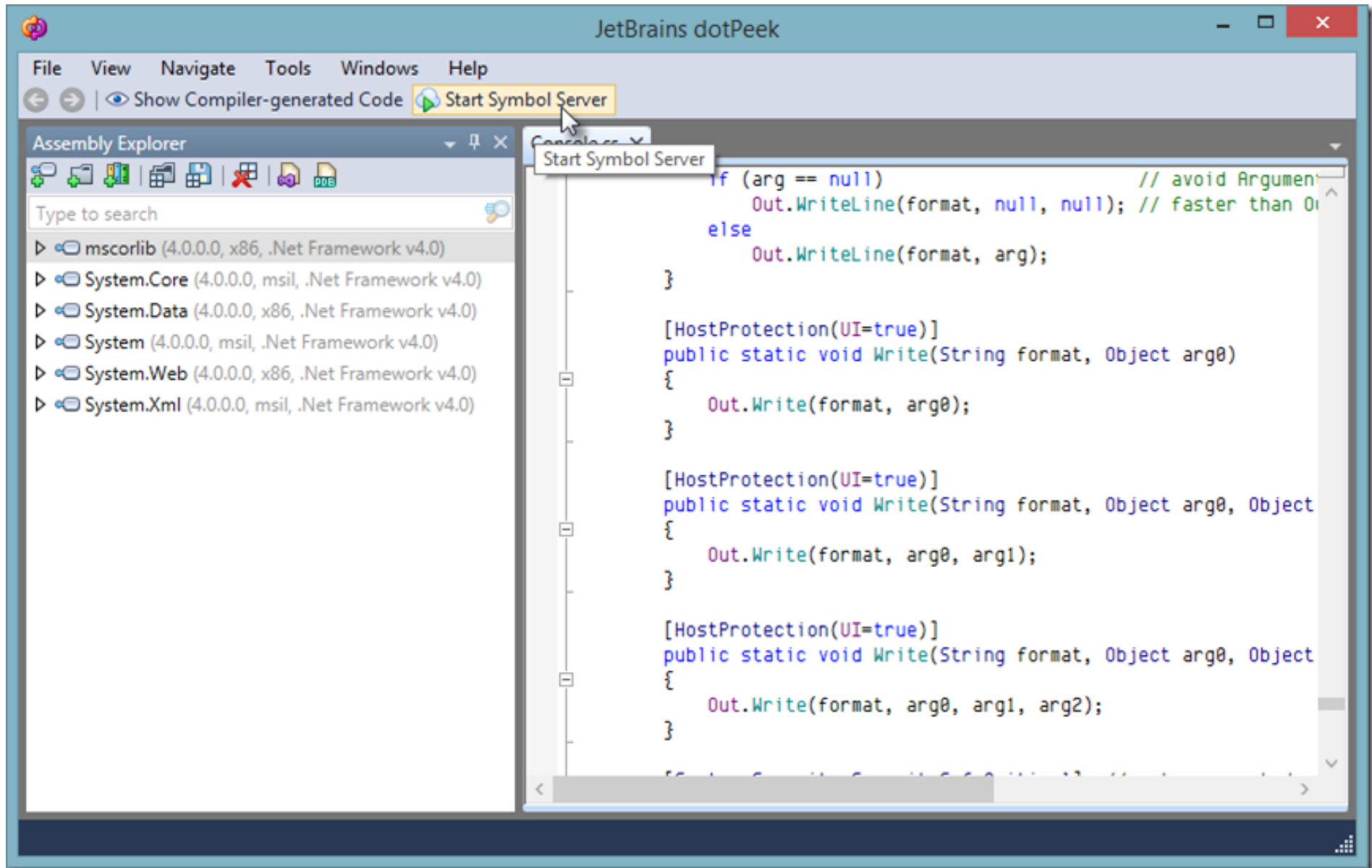
Décompilation en java

Site <http://www.javadecompilers.com/>



Décompilation en .Net

- Désassembleur syscalls (PE, C# -> C inconnu)



Démonstration (Haut Niveau)

Haut Niveau exercices

- Décompiler les fichiers .jar sur le site pour les challenges (github)
- Analyser le fichier InsecureBankingv2.apk et trouver l'encryption vulnérable (optionnel)
- Décompiler les fichier PE de type .net (.exe) et trouver le mot de passe ou la clé du programme

Moyen Niveau C/C++

- Possible d'éviter l'apprentissage de multiples assembleurs (NASM, GAS (ARM,PPC),etc.)
- Portable, facile à comprendre et à analyser
- C++ meilleure compréhension OOP, pas de transition directe dans l'assembleur
- Permet d'avoir une meilleure compréhension du code qui est traduit de façon ambiguë en ASM

Analyser le C++

- Utiliser Ida Pro ARM et x86 seulement
- Plugin snowman (F3)

The screenshot displays the IDA Pro interface with the following components:

- Menu Bar:** File, Analyse, View, Help
- Instructions Window (Left):**

Address	Disassembly
8048094:	push ebp
8048095:	mov ebp, esp
8048097:	sub esp, 0x18
804809a:	cmp [ebp + 0xc]:32, 0x0
804809e:	jnz 0x80480a5
80480a0:	mov eax, [ebp + 0x8]:32
80480a3:	jmp 0x80480c1
80480a5:	mov eax, [ebp + 0x8]:32
80480a8:	mov edx, eax
80480aa:	sar edx, 0x1f
80480ad:	idiv [ebp + 0xc]:32
80480b0:	mov eax, edx
80480b2:	mov [esp + 0x4]:32, eax
80480b6:	mov eax, [ebp + 0xc]:32
80480b9:	mov [esp]:32, eax
80480bc:	call 0x8048094
80480c1:	leave
80480c2:	ret
- C++ Window (Right):**

```
int32_t gcd(int32_t arg1, int32_t arg2) {  
    int32_t eax1;  
  
    if (arg2 != 0) {  
        eax1 = gcd(arg2, arg1 % arg2);  
    } else {  
        eax1 = arg1;  
    }  
    return eax1;  
}
```
- Status Bar:** Line 6, Column 27

Analyser le C

- Utiliser le site internet:
<https://retdec.com/decompilation/>

The screenshot shows the RetDec web interface. At the top is a navigation bar with links: Home, News, Publications, Contact, Try Decompile! (highlighted), API, and IDA Plugin. Below the navigation bar is a heading "Try Out Decompile In Your Browser". A subtext says "Select a file to be decompiled and press the decompilation button." The main form is divided into sections: "Input" with "Type of Input" (C Code, Executable File, Raw Machine Code), "Input File" (none, max. 10 MB), "Supported formats" (ELF, PE, COFF, AR (archive), Intel HEX), "Supported architectures (32b)" (Intel x86, ARM, ARM+Thumb, MIPS, PIC32, PowerPC), and "PDB File" (none, optional). Below this is "Input File Settings" with a "Show" link. Then "Decompilation Settings" with a "Show" link. At the bottom is a "Decompile!" button and a note "(max. decompilation time: 5 minutes)".

Home News Publications Contact Try Decompile! API IDA Plugin

Try Out Decompile In Your Browser

Select a file to be decompiled and press the decompilation button.

Input

Type of Input: C Code Executable File Raw Machine Code

Input File: (none) (max. 10 MB)

Supported formats: ELF, PE, COFF, AR (archive), Intel HEX

Supported architectures (32b): Intel x86, ARM, ARM+Thumb, MIPS, PIC32, PowerPC

PDB File: (none) (optional)

Input File Settings

[Show](#)

Decompilation Settings

[Show](#)

Decompile!

(max. decompilation time: 5 minutes)

Assembleur point clés

- Encodage (little/big endian)
- RISC vs CISC
- Les instructions les plus communes et utiles

```
.section ".text"
.global _start
_start:
    mov 4,%g1          ! 4 is SYS_write
    mov 1,%o0          ! 1 is stdout
    set .msg,%o1       ! pointer to buffer
    mov (.msgend-.msg),%o2 ! length
    ta 8

    mov 1,%g1          ! 1 is SYS_exit
    clr %o0            ! return status is 0
    ta 8

.msg:
    .ascii "Hello world!\n"
.msgend:
```

Endianness

- Big Endian (PowerPc,SPARC)

le bit le plus significatif est à l'adresse la moins significative. Ex: 0x7370617263 -> "sparc"

- Little Endian(x86,(arch)el)

le bit le moins significatif est à l'adresse la moins significative. Ex: 0x6372617073 -> "craps"

- En python `mot[::-1]` pour changer l'endianness d'un mot

CISC

- CISC (Complex Instructions Set)
- Peu de registres (généraux et spéciaux)
- Exemple x86: eax, ebx, ecx, edx, esp, ebp
- Instructions complexes par exemple
- Inc edx -> i++ in C/C++

RISC

- RISC (reduced instruction set)
- Beaucoup de registres
- Example ARMv7: r0-r12 (généraux), r13 (stack), r14 (link) pour fonctions, r15 (horloge)
- Instructions simples (logiques, arith, etc.)
- Ex ARM :
 - MOV R2, #0
 - ADD R2, R2, #1
 - i++ in C/C++

CISC VS RISC examples

- CISC

x86 (amd64,i386)

6502 (nes)

68k (m68k)



https://images.duckduckgo.com/iu/?u=http%3A%2F%2Fimg3.wikia.nocookie.net%2F__cb20140207125257%2Flogopedia%2Fimages%2Fd%2Fd6%2FIntel_Xeon.png&f=1

- RISC

ARM (armhf, armel)

SPARC (sparc(64))

PowerPc (ppc(64(el)))



https://images.duckduckgo.com/iu/?u=http%3A%2F%2Fwww.tug.ca%2Fblast%2FV24%2Flibrary%2Flogo_IBM_POWER6_BoP.gif&f=1

Instructions en ASM (ARM & x86)

- Arithmétiques
- Logiques
- Accès données et mémoire
- Les boucles et conditions

Pour les architectures RISC
(ARM, powerpc, sparc, etc.) {opt}.

Pour l'x86 (opt).

Instructions arithmétiques

ASM

C/C++

- ADD dest, src/val, {val2/src2} -> $z = x + y$;
- SUB dest, src/val, {val2/src2} -> $z = x - y$;
- (i)MUL dest(ecx), src/val, val2/src2 -> $z = x * y$;
- (i)div {dest}(ecx),{src/val},{src2/val2} -> $z = x / y$;
- DIV pas dans ARM seulement x86, ppc, sparc, etc.
- imul et idiv pour opérations signés en x86

Instructions Logiques

ASM

C/C++

- AND dest, src/value,{src2/value2}-> $z = x \& y;$
- OR dest, src/value,{src2/value2} -> $z = x | y;$
- XOR dest,src/value,{src2/value2} -> $z = x \wedge y;$

Accès Mémoire et Données

- MOV dest,src/val ;move data to register (all)
- lea dest,src/val ;load calculated address x86
- LDR dest, [src]/=label ;loads at label/src address ARM
- STR dest, [src] ;stores at src address ARM
- PUSH {{reg}}/(val/reg) ;puts data on stack
- POP{{reg}}/(reg) ;restores registers(ARM)/removes data from stack into register (x86)

Les boucles et conditions

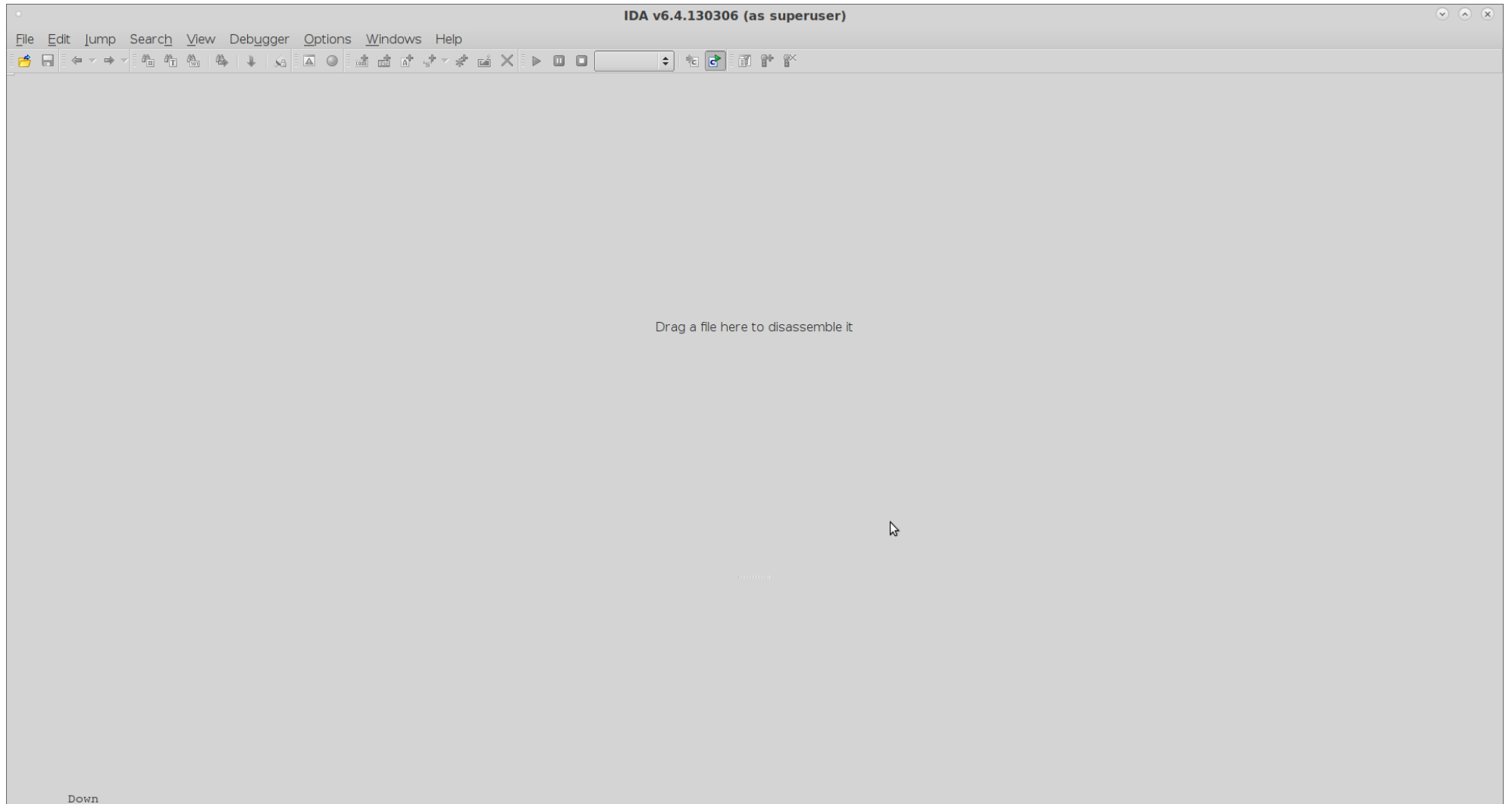
- `cmp reg, reg/val` ;vérifier la valeur d'un registre pour vérifier une condition (tous)
- `B(cond) label` ;sauter (in)conditionnellement à une fonction/procédure défini par un label (RISC)
- `J(cond) label` ;sauter (in)conditionnellement à une fonction/procédure défini par un label (x86)
- Définir le label dans le code l'équivalent d'un bloc if/else ou le code l'intérieur d'une boucle
- Loop en x86 sert à sauter à une boucle à partir du registre `ecx`

Conditions en assembleur

x86	ppc	sparc	ARM	C/C++
e	eq	e	EQ	==
ne	ne	ne	NE	!=
g/(a)	gt	g	GT	>
ge/(ae)	ge	ge	GE	>=
l/(b)	lt	l	LT	<
le/(be)	le	le	LE	<=

(cond) pour opérations non signés x86

Démo Ida pro



Exercices (ASM et C/C++)

- Télécharger les applications crackme de github folder C(++) and ASM binaries
- Utiliser Ida Pro ou Hopper pour les résoudre
- Ne pas debug aucune aide sera accordé pour ceux utilisant un debugger

Pour plus d'information (ASM)

- http://en.wikibooks.org/wiki/X86_Assembly
- <http://www.ds.ewi.tudelft.nl/vakken/in101/labcourse/instruction-set/>
- https://www.tutorialspoint.com/assembly_programming/index.htm
- https://en.wikibooks.org/wiki/SPARC_Assembly
- Guide to RISC Processors: for Programmers and Engineers

Outils pour la rétro-ingénierie

- Disassemblers:

Ida Pro:

<https://www.hex-rays.com/products/ida/index.shtml>

Hopper: <https://www.hopperapp.com/download.html>

Binary Ninja: <https://binary.ninja/faq/>

Radare2: <http://www.radare.org/r/>

- Decompilers C/C++:

retargetable: <https://retdec.com/home/>

snowman: <https://derevenets.com/>

Outils pour la rétro-ingénierie(suite)

- Decompilers .Net:

<https://github.com/0xd4d/dnSpy/releases>

<https://www.jetbrains.com/decompiler/>

- Decompilers Java:

JD-GUI(out of date): <http://jd.benow.ca/>

<http://www.javadecompilers.com/>

Conclusion

- Utiliser un décompilateur pour des .jar, .apk et .exe (.net)
- Pour des elf, mach-o ou PE utiliser un désassembleur et analyser le code source de l'application (ASM)