



EPOCHE & ESPRI

Applying Reverse Engineering in Common Criteria Evaluations Below EAL 4

Content



E P O C H E & E S P R I

- Current Situation
 - Assurance level, Attack Potential and Vulnerabilities Analysis
- Reverse engineering
 - C/C++, .Net and Java
 - Obfuscation?
- What happen with current attack potential?

Current Situation



E P O C H E & E S P R I

- Assurance level implies attack potential

EAL 1	BASIC
EAL 2	
EAL 3	
EAL 4	ENHANCED BASIC

- Lower levels less information available.
- Higher level puts the source code on the table

Current Situation



E P O C H E & E S P R I

- Information provided by sources
 - Critical Functions
 - Direct Buffer Overflows Detection
 - Cryptography analysis
 - Execution flow
 - Communication between subsystems
- The source code is a high detail level evidence of the TOE.

Current Situation



E P O C H E & E S P R I

- Source code availability enables deeper analysis and usually new vulnerabilities.
- But it is necessary a higher assurance level
 - Bigger attack potential is considered
- How ever, reverse engineering is out there!

Reverse engineering



E P O C H E & E S P R I

- What?
 - Binaries analysis to figure out the execution flow and the internals of a program.

- Available tools
 - Easy to use
 - Decompile
 - Debug

Reverse engineering



E P O C H E & E S P R I

- We can make a technology classification

Binary type	Technologies	De-compilation
Machine Code	C, C++, Delphi...	Quite confident <ul style="list-style-type: none">■ Operation codes■ Standard libraries
Byte Code	Java, .Net ...	Not very confident <ul style="list-style-type: none">■ Class taxonomy OK■ Problems with java

R.E. IDA Pro



E P O C H E & E S P R I

- IDA Pro
 - Input
 - Binary file

 - De-compilation results
 - Assembly code
 - Execution flow
 - Functions and strings recognition

R.E. IDA Pro



E P O C H E & E S P R I

- IDA Pro execution flow analysis

IDA - C:\Documents and Settings\levaluator\Desktop\ejecutables\ejemplo\ejemplo

File Edit Jump Search View Debugger Options Windows Help

100% (-112,418) (907,290) 00004F9 080484F9: main+5A

```
mov [esp+130h+5], offset aCorrectPasswor ; "Correct Password!\n"
call _scanf
lea eax, [ebp-107h]
mov [esp+130h+5], eax
call check
test eax, eax
jnz short loc_8048509
```

```
loc_8048509: ; "Wrong Password!\n"
mov [esp+130h+5], offset aWrongPassword
call _puts
```

```
loc_8048515:
mov eax, 1
mov edx, [ebp-8]
xor edx, large gs:14h
jz short loc_804852B
```

```
loc_804852B:
add esp, 124h
pop ecx
pop ebp
lea esp, [ecx-4]
retn
main endp
```

Name	Address	P
__init_proc	08048328	P
__gmon_start__	08048368	
__libc_start_main	08048378	
_scanf	08048388	
__stack_chk_fail	08048398	
_puts	080483A8	
_strcmp	080483B8	
__start	080483D0	P
__do_global_ctors_aux	08048400	
frame_dummy	08048460	
check	08048484	P
main	0804849F	P
__libc_csu_fini	08048540	P
__libc_csu_init	08048550	P
__CSC_ctors_aux	080485A8	P

Copyright (c) 1990-2009 Python Software Foundation - <http://www.python.org/>

IDA Python version 1.1.0 final (serial 0)
Copyright (c) 2004-2009 Gergely Erdelyi - <http://d-dome.net/idadpython/>

Propagating type information...
Function argument information has been propagated
The initial autoanalysis has been finished.

R.E. IDA Pro



EPOCHE & ESPRI

■ IDA Pro features and views

The screenshot displays the IDA Pro interface with several windows open:

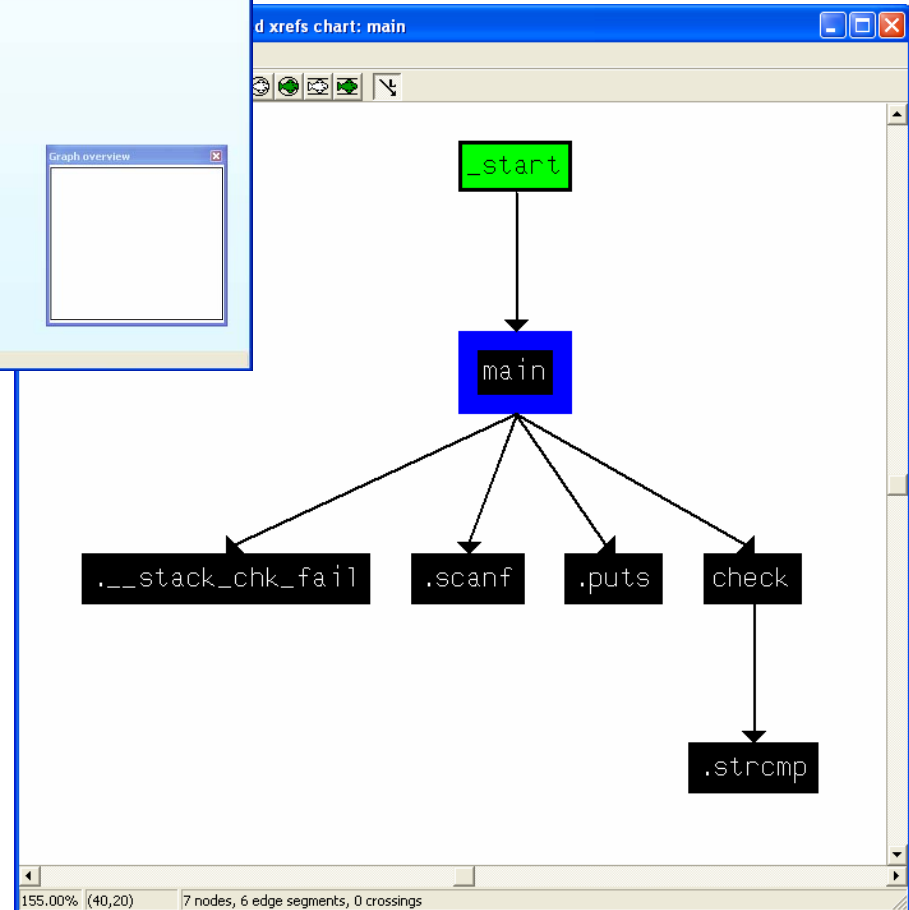
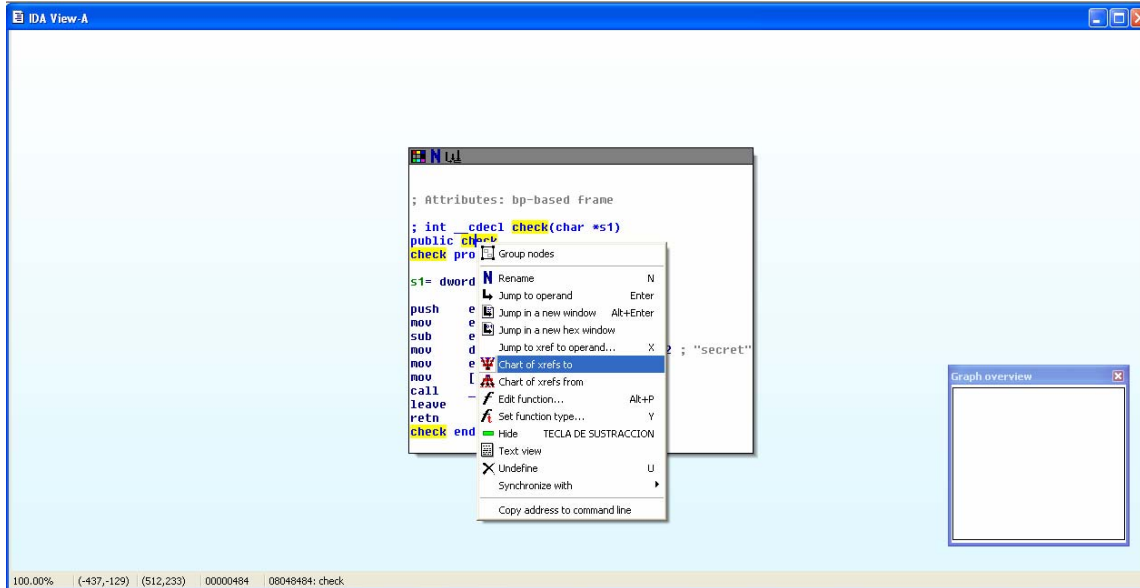
- Hex View-A:** Shows assembly code with hex and ASCII values. Example: `.text:00401110 87 00 00 50 E8 4B 07 00 00 FF 35 1B D1 44`
- Functions window:** Lists functions with their names, segments, start addresses, lengths, and flags. Example: `start .text 00401000 00000059`
- Structures:** Shows structure definitions. Example: `Ins/Del : create/delete structure`
- Names window:** Lists symbols with their names and addresses. Example: `@_virt_commit 004012A0`
- Exports:** Lists exported functions with their names, addresses, and ordinals. Example: `__GetExcepDllInfo 00401059 1`
- Imports:** Lists imported functions with their addresses, ordinals, names, and libraries. Example: `0041318C LoadLibraryA KERNEL32`
- Strings window:** Lists strings with their addresses, lengths, and contents. Example: `data:0040... 00000007 C secret`

At the bottom, the status bar shows: `Python version 1.1.0 final (serial 0) Copyright (C) 2004-2009 Gergely Erdelyi - http://d-dome.net/idadpython/ Using FLIRT signature: BCC v4.X/5.X & BCB v1.0/v7.0 BDS2006 win32 runtime Python AU: idle Down Disk: 350MB`

R.E. IDA Pro



EPOCHE&ESPRI



- IDA Pro graph drawing

R.E. Pai Mei



E P O C H E & E S P R I

- Pai Mei
 - Input
 - PIDA file (Python IDA)

 - De-compilation results
 - Process Stalking
 - Execution flow (from IDA PRO)

R.E. Pai Mei



E P O C H E & E S P R I

■ Pai Mei Process Stalking

The screenshot displays the PAIMEIconsole interface with the following components:

- Data Sources:** Shows available targets including 'C sample' and 'tag'.
- Data Exploration:** A table listing process instances with columns for #, Time, EIP, TID, Module, Func?, and Tag. The table shows several instances of 'ejemplo.exe' running at different times and EIP addresses.
- Data Capture:** Includes a 'Refresh Process List' button and a table for capturing data. The 'Load' field is set to 'C:\Documents and Settings\evaluator\'. Coverage depth options include 'Functions' and 'Basic Blocks'. Checkboxes for 'Restore BPs', 'Heavy', and 'Unhandled Only' are visible.
- PAIMEI Modules:** A table showing 376 functions and 436 basic blocks for the 'ejemplo.exe' module.
- CODE! PAIMEI.[EXT]:** A log window showing the configuration of 91 breakpoints on functions in the main module of 'ejemplo.exe'. The log includes the following entries:

```
[*] Exporting 36 hits to MySQL.
[*] Stalking module ejemplo.exe
[*] Setting 91 breakpoints on functions in main module
[*] Loading 0x7c900000 \WINDOWS\system32\ntdll.dll
[*] Loading 0x7c800000 \WINDOWS\system32\kernel32.dll
[*] Loading 0x77d40000 \WINDOWS\system32\user32.dll
[*] Loading 0x77f10000 \WINDOWS\system32\gdi32.dll
[*] debugger hit 0040c810 cc #1
[*] debugger hit 00402114 cc #2
[*] debugger hit 004014b0 cc #3
[*] debugger hit 00402118 cc #4
[*] debugger hit 0040c7ec cc #5
[*] debugger hit 0040c7e0 cc #6
[*] debugger hit 0040a8dc cc #7
[*] debugger hit 0040a8e4 cc #8
[*] debugger hit 00402da8 cc #9
[*] debugger hit 0040c81c cc #10
[*] debugger hit 0040ac18 cc #11
[*] debugger hit 0040c7d4 cc #12
[*] debugger hit 0040c7da cc #13
[*] debugger hit 0040c852 cc #14
[*] debugger hit 0040c858 cc #15
[*] debugger hit 0040c8d6 cc #16
[*] debugger hit 0040c80a cc #17
[*] debugger hit 00409f2c cc #18
[*] debugger hit 0040a5b4 cc #19
[*] debugger hit 0040c8a6 cc #20
[*] debugger hit 0040c828 cc #21
[*] debugger hit 0040c82e cc #22
[*] debugger hit 0040c7f2 cc #23
[*] debugger hit 00408d40 cc #24
[*] debugger hit 00401165 cc #25
[*] debugger hit 0040c8e8 cc #26
[*] debugger hit 0040c88e cc #27
```
- Process Stalker:** A window titled 'C:\Documents and Settings\evaluator\Desktop\ejecutables\ejemplo\ejemplo.exe' showing a password prompt: 'Please, enter password:'.

The status bar at the bottom indicates: 'Successfully connected to MySQL server at localhost. Process Stalker MySQL: localhost PyDbg: localhost uDraw: NONE User: evaluator'.

R.E. Pai Mei



E P O C H E & E S P R I

■ Pai Mei Process Stalking

The screenshot displays the PAIMEIconsole application interface. On the left sidebar, there are icons for PAIMEI diff, PAIMEI docs, PAIMEI explore, PAIMEI [EX-T], PAIMEI fuzzer, PAIMEI peek, and PAIMEI stalker. The main window is divided into several panes:

- Data Sources:** Shows available targets like 'C sample' and 'tag'.
- Data Exploration:** A table listing process data:

#	Time	EIP	TID	Module	Func?	Tag
21	+0s	0040c828	3340	ejemplo.exe	Y	tag
22	+0s	0040c82e	3340	ejemplo.exe	Y	tag
23	+0s	0040c7f2	3340	ejemplo.exe	Y	tag
24	+0s	00408d40	3340	ejemplo.exe	Y	tag
25	+0s	00401165	3340	ejemplo.exe	Y	tag

Below the table, it shows 'Functions: 36 / 376' and 'Basic Blocks: 36 / 436'. The 'Dereferenced Data' pane shows a password prompt: 'secret. Please, enter password: .%.s. Correct Password!!... Wrong Password!!...'. The 'Data Capture' pane on the right shows a 'Refresh Process List' button and a table for capturing data, with options for 'Coverage Depth' (Functions selected), 'Restore BPs', 'Heavy', and 'Unhandled Only'. The bottom status bar shows 'Successfully connected to MySQL server at localhost.', 'Process Stalker', 'MySQL: localhost', 'PyDbg: localhost', 'uDraw: NONE', and 'User: evaluador'.

■ Pai Mei De-compilation

PAIMEIconsole
Connections Advanced Help

Browser

# Func	# BB	PIDA Module
376	436	ejemploc.exe

Add Module(s)

Modules

- ejemploc.exe
 - 0040106c - sub_40106C
 - 004010f3 - sub_4010f3
 - 00401150 - sub_401150
 - 00401150
 - 00401165 - _main
 - 00401165
 - 0040119e
 - 004011ab
 - 004011b6
 - 00401494 - sub_401494
 - 004014b0 - nullsub_1
 - 00402114 - nullsub_2
 - 00402118 - nullsub_3
 - 0040270b - sub_40270B
 - 00402da8 - sub_402da8
 - 0040391c - sub_40391C
 - 00407b90 - j_unknown_libname_9
 - 00407ddc - sub_407ddc
 - 00407de2 - sub_407de2
 - 00407de8 - sub_407de8
 - 00407dee - sub_407dee
 - 00407e18 - sub_407e18
 - 00408288 - sub_408288

Disassembly

```
_main
00401165  push ebp
00401166  mov  ebp, esp
00401168  add  esp, 0FFFFFF00h
0040116e  push offset format
00401173  call _printf
00401178  pop  ecx
00401179  lea  eax, [ebp+si]
0040117f  push eax
00401180  push offset aS
00401185  call _scanf
0040118a  add  esp, 8
0040118d  lea  edx, [ebp+si]
00401193  push edx
00401194  call sub_401150
00401199  pop  ecx
0040119a  test eax, eax
0040119c  jnz  short loc_4011AB

0040119e  push offset aCorrectPasswor
004011a3  call _printf
004011a8  pop  ecx
004011a9  jmp  short loc_4011B6

004011ab  push offset aWrongPasswor
004011b0  call _printf
004011b5  pop  ecx

004011b6  mov  eax, 1
004011bb  mov  esp, ebp
004011bd  pop  ebp
```

Special

[*] PaiMei Explorer
[*] Module by Pedram Amini
[*] Loaded PIDA module 'ejemploc.exe' in 0.25 seconds.

Successfully connected to MySQL server at localhost. PaiMei ... Hayati MySQL: localhost PyDbg: localhost uDraw: NONE User: evaluador

R.E. Pai Mei



EPOCHE & ESPRI

- Pai Mei graph drawing

The image shows a control flow graph (CFG) in uDraw(Graph) 3.1.1 and the PAIME! console. The CFG consists of three nodes:

```
00401165 push ebp
00401166 mov ebp, esp
00401168 add esp, 0FFFFFF00h
0040116e push offset format
00401173 call _printf
00401178 pop ecx
00401179 lea eax, [ebp+5]
0040117f push eax
00401180 push offset aS
00401185 call _scanf
0040118a add esp, 8
0040118d lea edx, [ebp+5]
00401193 push edx
00401194 call sub_401150
00401199 pop ecx
0040119a test eax, eax
0040119c jnz short loc_4011B6
```

```
004011a9 push offset aWrongPassword
004011ab call _printf
004011b0 pop ecx
```

```
0040119e push offset aCorrectPassword
004011a3 call _printf
004011a8 pop ecx
004011a9 jmp short loc_4011B6
```

```
004011b6 mov eax, 1
004011bb mov esp, ebp
004011bd pop ebp
004011be ret
```

The PAIME! console shows the loaded module 'ejemploc.exe' and the assembly code for the '_main' function, which matches the code in the CFG nodes. The console also displays connection logs for uDraw(Graph) server and MySQL, PyDbg, and uDraw.

R.E. Reflector



E P O C H E & E S P R I

- Reflector
 - Input
 - Binary file (Byte code)

 - De-compilation results
 - Class taxonomy
 - Source code (C#, Visual Basic, Delphi...)
 - Execution flow

R.E. Reflector



E P O C H E & E S P R I

- Reflector class taxonomy

Red Gate's .NET Reflector

File View Tools Help

C#

- ejemplosharp
 - ejemplosharp.exe
 - References
 - { } -
 - <Module>
 - Derived Types
 - { } ejemplosharp
 - checkPassword
 - Program
 - Base Types
 - Derived Types
 - .ctor()
 - Main(String[]) : Void

```
private static void Main(string[] args);
```

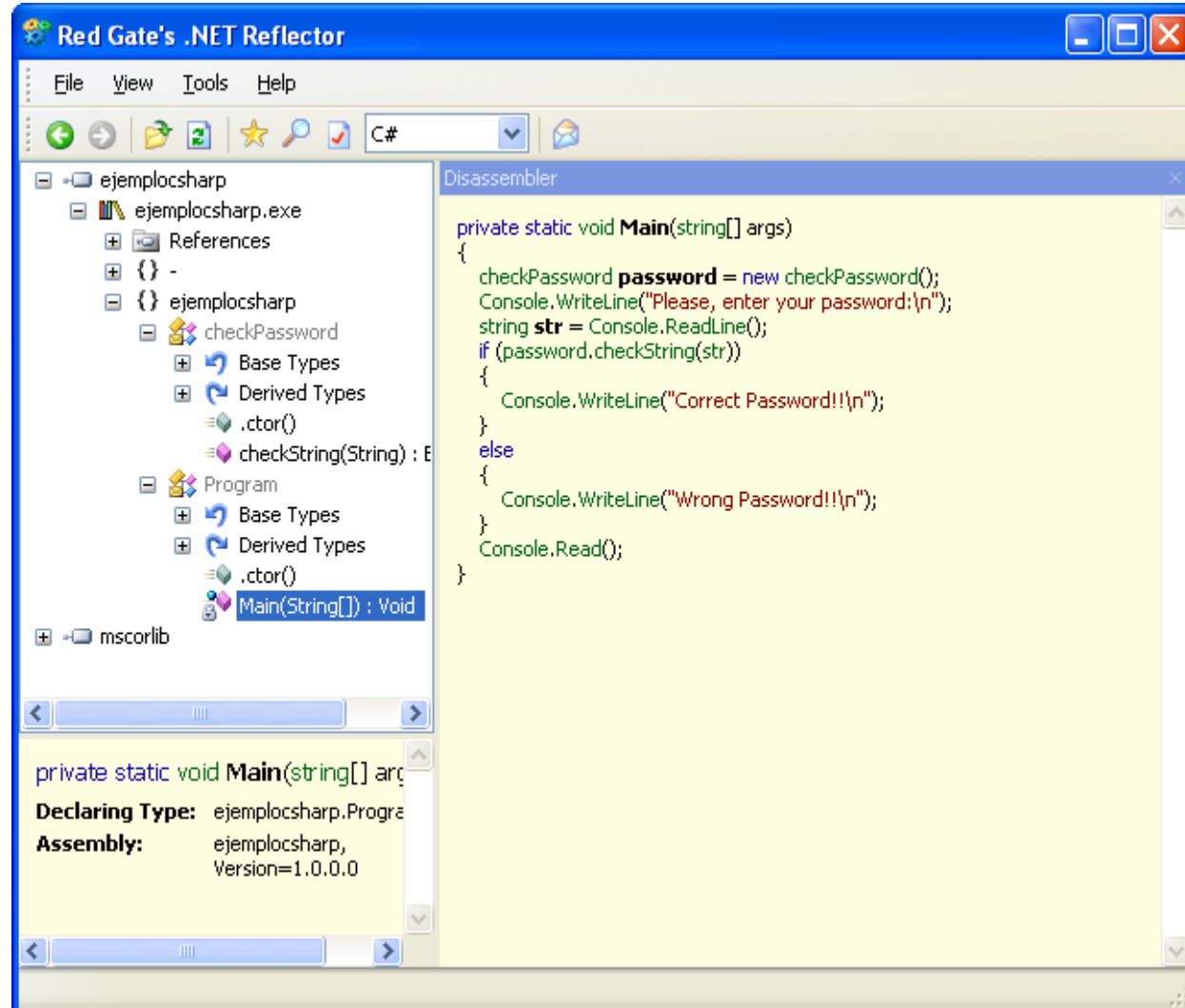
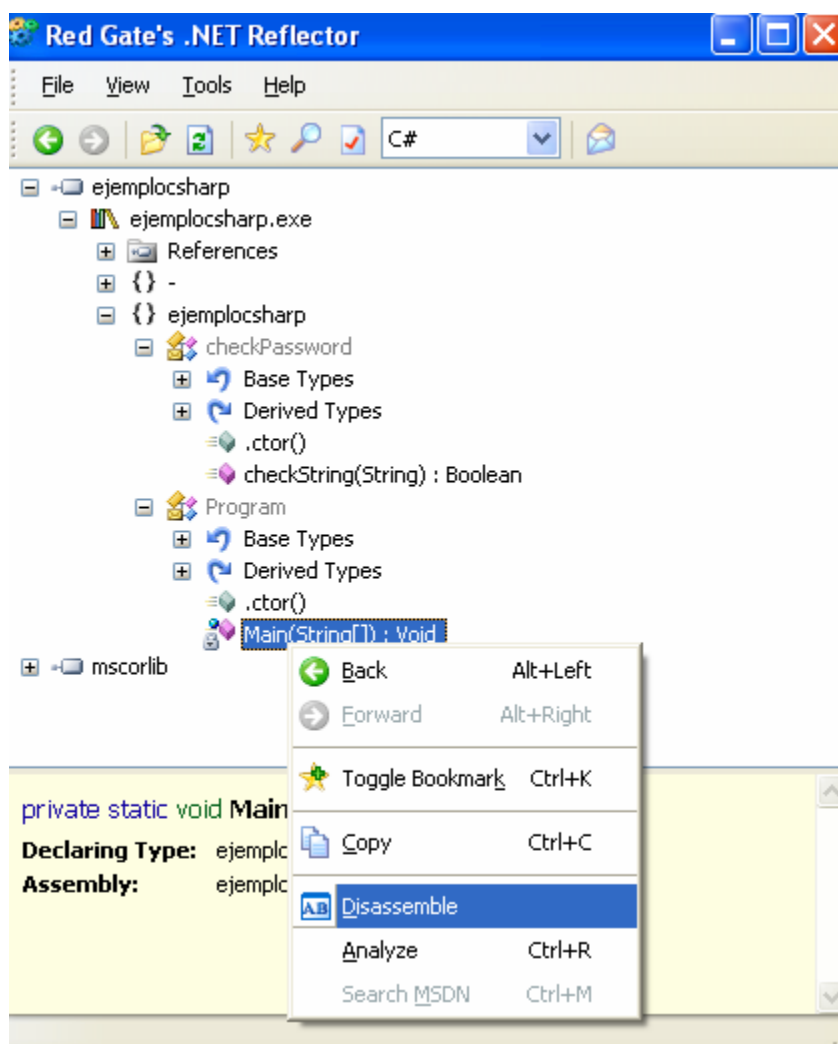
Declaring Type: ejemplosharp.Program
Assembly: ejemplosharp, Version=1.0.0.0

R.E. Reflector



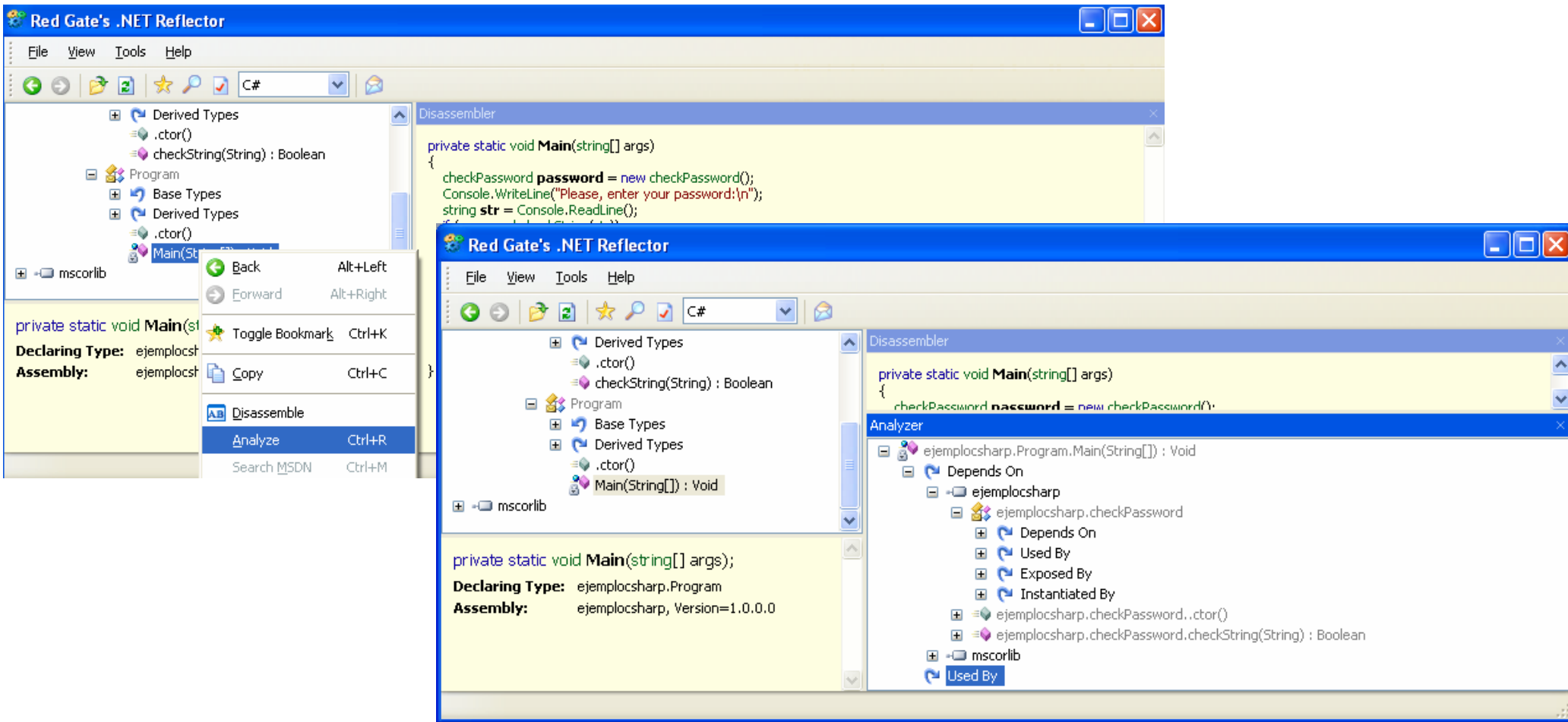
EPOCHESPRI

- Reflector Disassembler



R.E. Reflector

- Reflector Analysis



The screenshot displays two instances of Red Gate's .NET Reflector. The top instance shows the disassembled code for the `Main` method, which includes a password check loop. The bottom instance shows the same code with the `Analyze` menu option selected, resulting in an `Analyzer` window that displays a dependency graph for the `Main` method. The graph shows dependencies on `ejemplosharp.checkPassword` and `microsoft.mscorlib`.

Top Instance (Disassembler):

```
private static void Main(string[] args)
{
    checkPassword password = new checkPassword();
    Console.WriteLine("Please, enter your password:\n");
    string str = Console.ReadLine();
}
```

Bottom Instance (Analyzer):

`ejemplosharp.Program.Main(String[]) : Void`

- Depends On
 - ejemplosharp
 - ejemplosharp.checkPassword
 - Depends On
 - Used By
 - Exposed By
 - Instantiated By
 - ejemplosharp.checkPassword..ctor()
 - ejemplosharp.checkPassword.checkString(String) : Boolean
 - microsoft.mscorlib
- Used By

R.E. DJ Java Decompiler



E P O C H E & E S P R I

- DJ Java Decompiler
 - Input
 - Java .class / jar

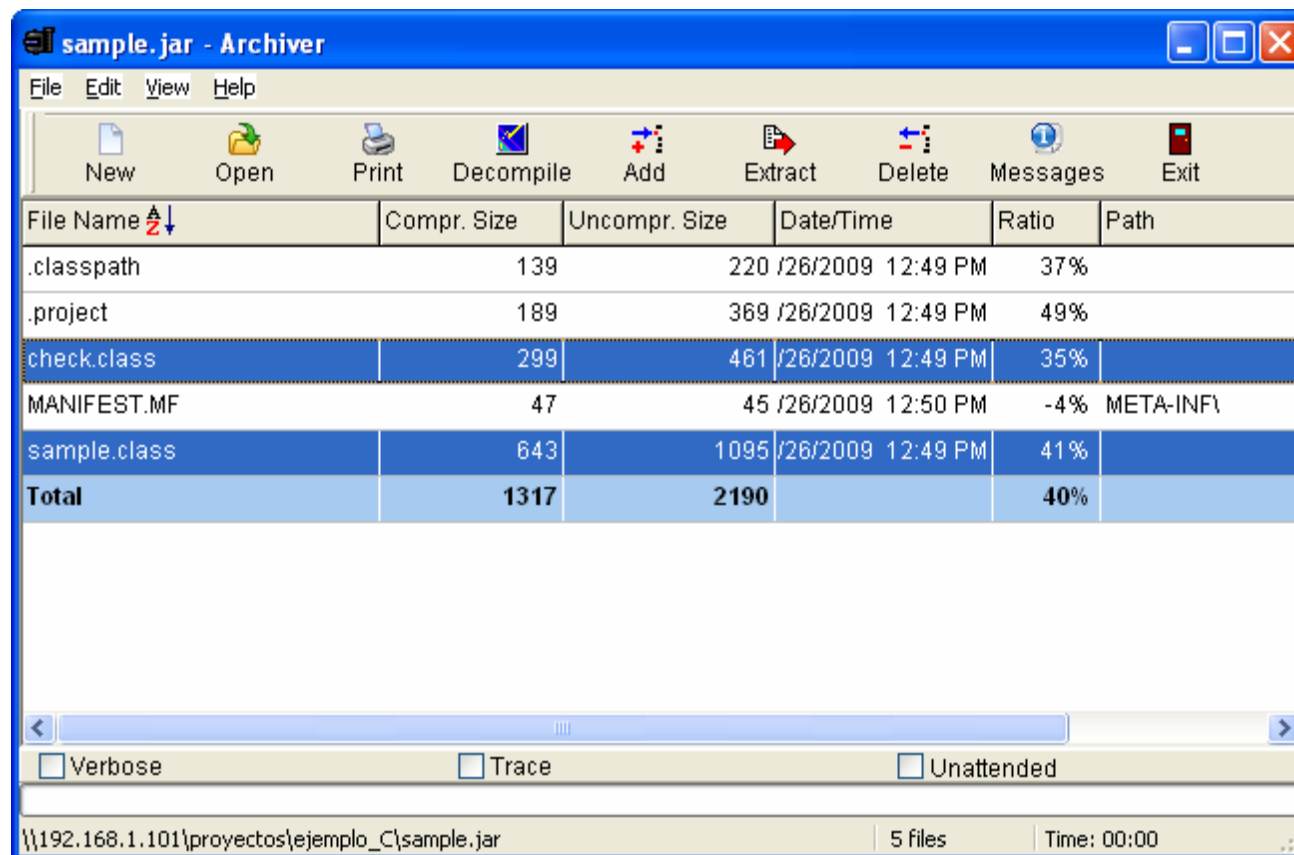
 - De-compilation results
 - Class taxonomy
 - Source Code

R.E. DJ Java Decompiler



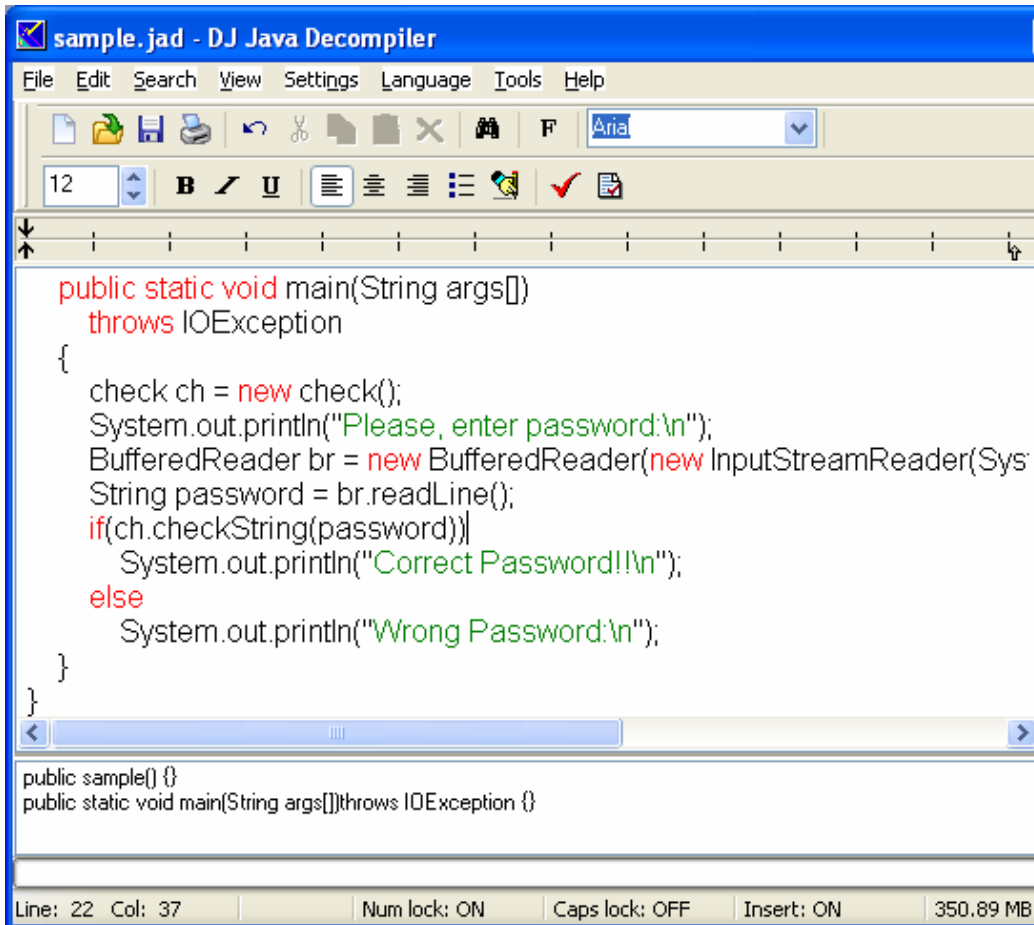
E P O C H E & E S P R I

- DJ Java Decompiler features



R.E. DJ Java Decompiler

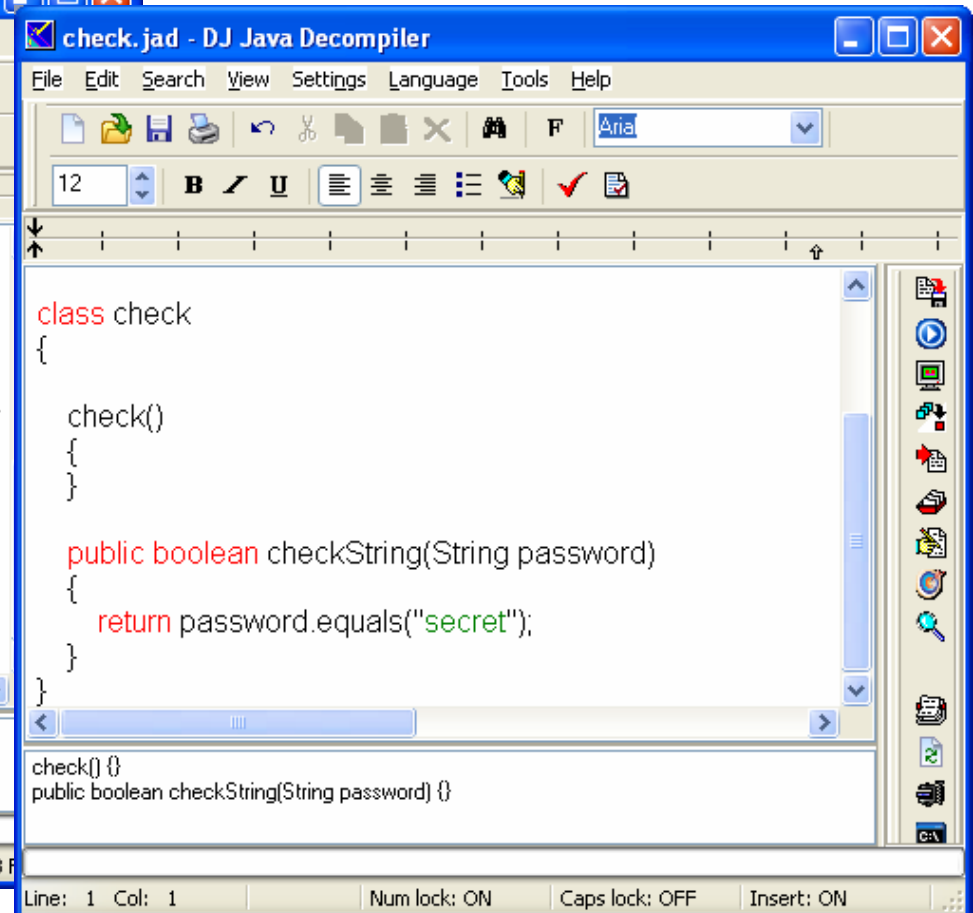
- DJ Java Decompiler features



```
public static void main(String args[])
throws IOException
{
    check ch = new check();
    System.out.println("Please, enter password:\n");
    BufferedReader br = new BufferedReader(new InputStreamReader(Sys
String password = br.readLine();
    if(ch.checkString(password))
        System.out.println("Correct Password!!\n");
    else
        System.out.println("Wrong Password:\n");
}
}

public sample() {}
public static void main(String args[])throws IOException {}
```

Line: 22 Col: 37 Num lock: ON Caps lock: OFF Insert: ON 350.89 MB



```
class check
{
    check()
    {
    }

    public boolean checkString(String password)
    {
        return password.equals("secret");
    }
}

check() {}
public boolean checkString(String password) {}
```

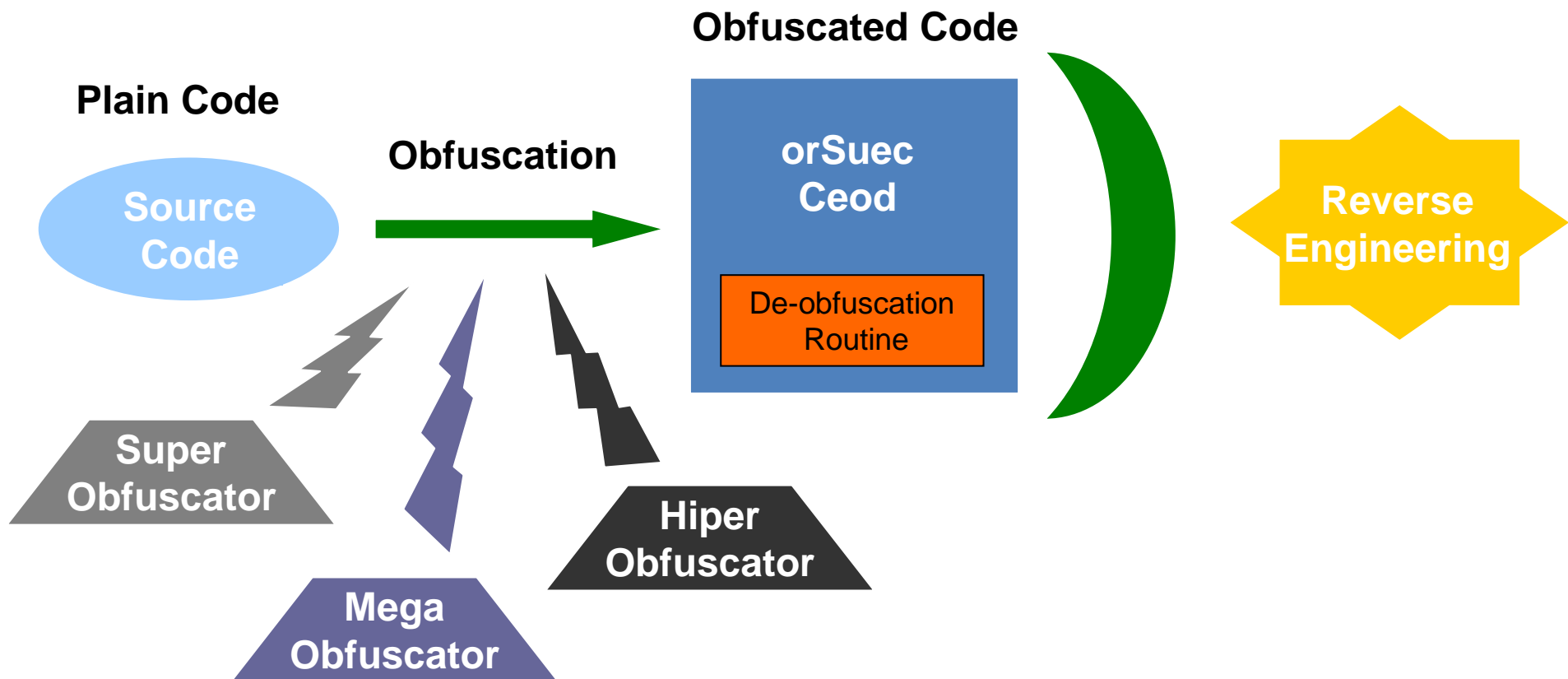
Line: 1 Col: 1 Num lock: ON Caps lock: OFF Insert: ON

Obfuscation?



E P O C H E & E S P R I

- Obfuscation

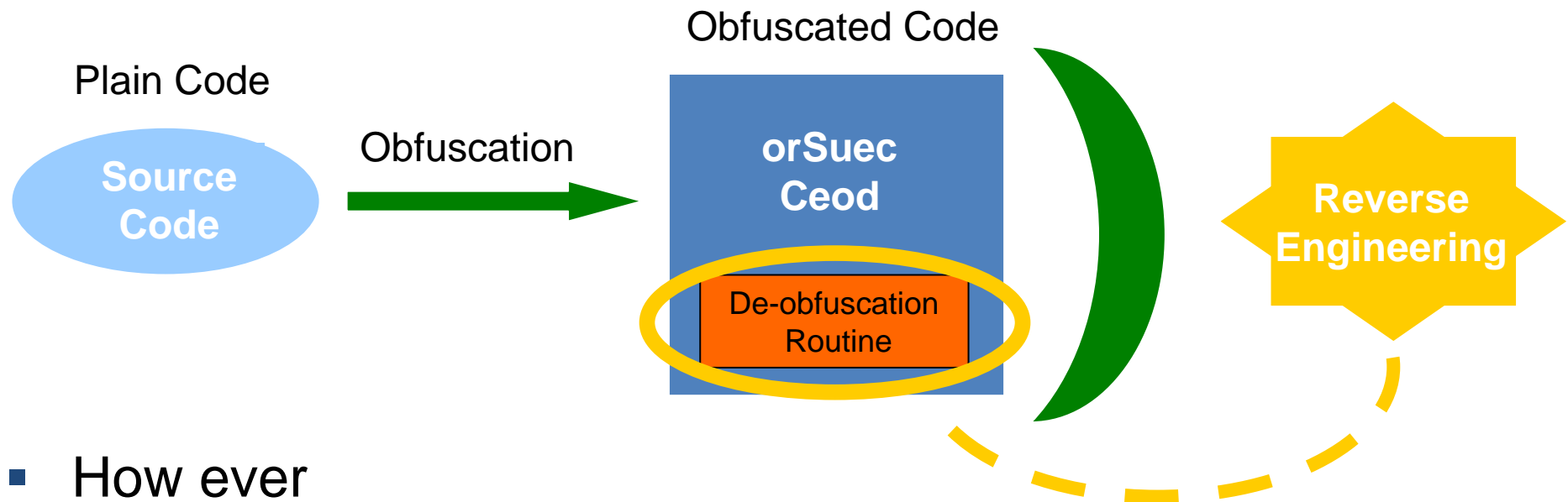


Obfuscation?



E P O C H E & E S P R I

- Obfuscation



- How ever

- Usually the de-obfuscation routine is included
- Many obfuscation methods are public
 - De-obfuscation is public too!
- De-compilation tools integrate known de-obfuscation routines

Profit?



E P O C H E & E S P R I

- Does reverse engineering applies in lower assurance levels?

- It is simple

- It is public



It applies

- New situation: EAL 1 to EAL 3

Sources available

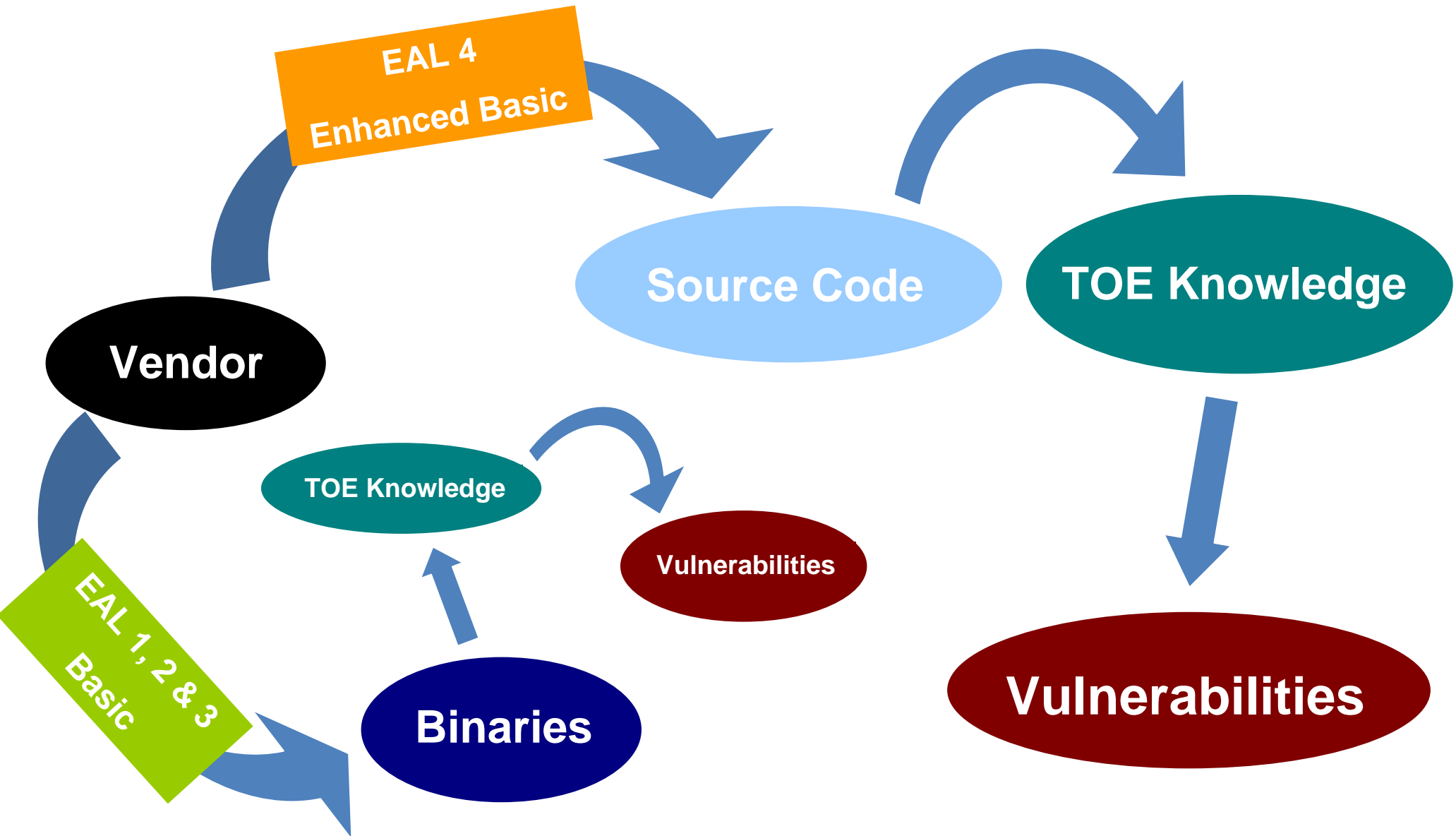


Deeper vulnerabilities Analysis

Conclusions



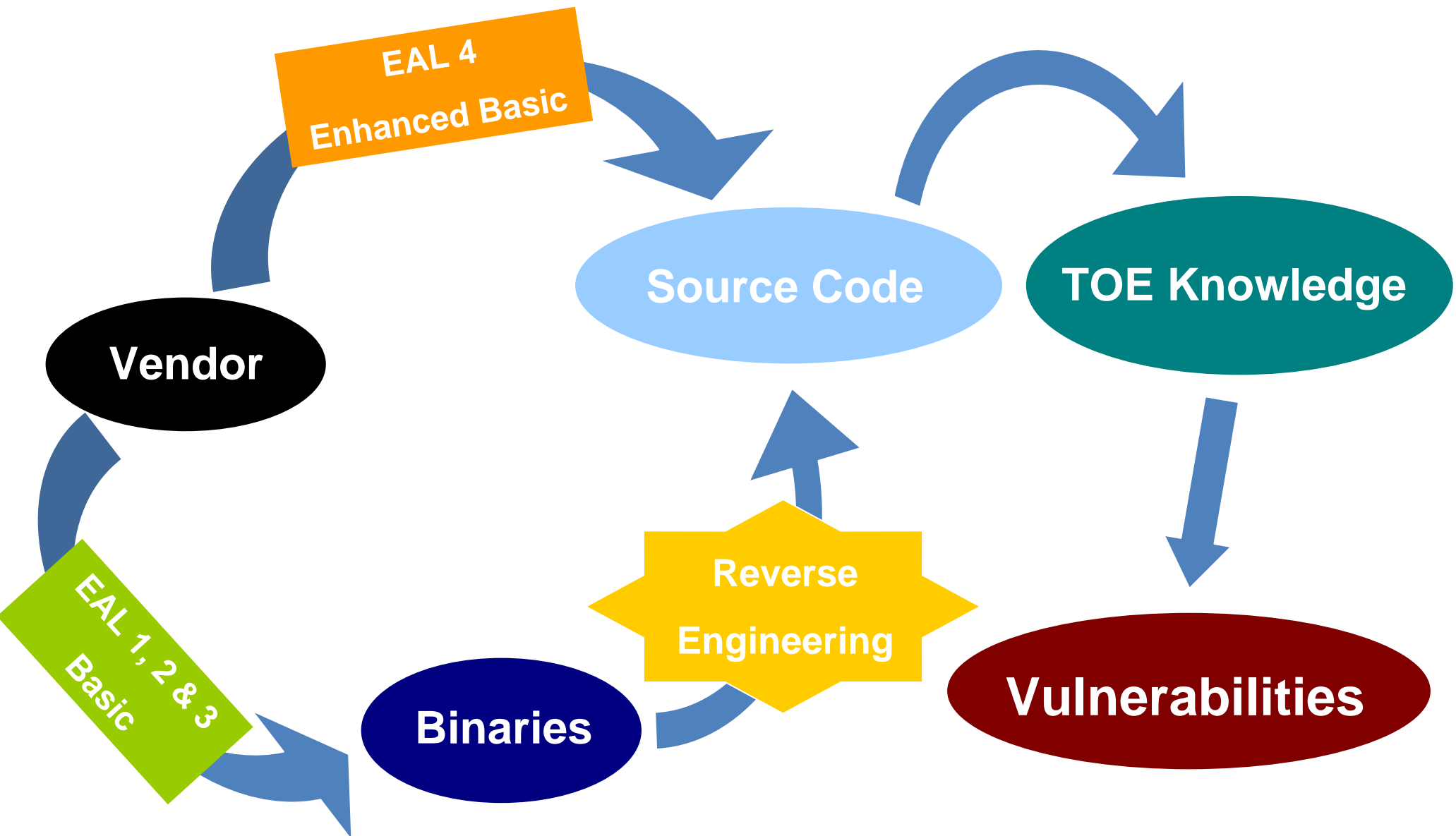
E P O C H E & E S P R I



Conclusions



E P O C H E & E S P R I



The end...



E P O C H E & E S P R I

Thanks! Questions?

Trifón Giménez
eval@epoche.es