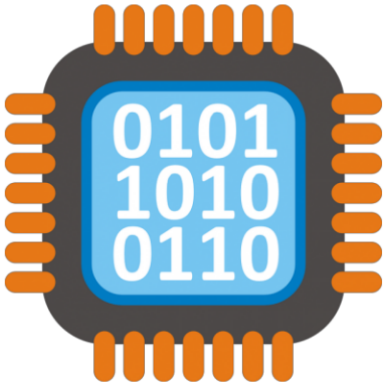
Secure Assembly Coding

Week # 13 Lectures



Dr. Qasem Abu Al-Haija,
Department of Cybersecurity,

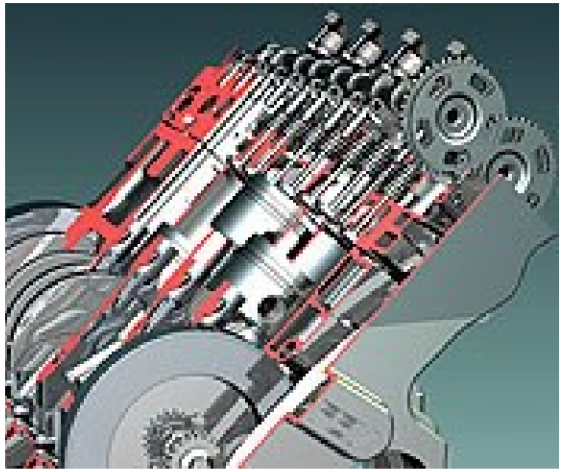
Reverse Engineering

Reverse Engineering

- Process of analyzing a subject system to create representations of the system at a higher level of abstraction"
- "Going backward through the development cycle."
- Discovering how a device usually works by taking it apart.
- Generally considered lawful if the system was obtained legitimately.

REing Mechanical Devices

- Not what you may think.
- Actually the reverse of the engineering process, going from a finished product to design.
- Used to “digitize” old parts and systems.

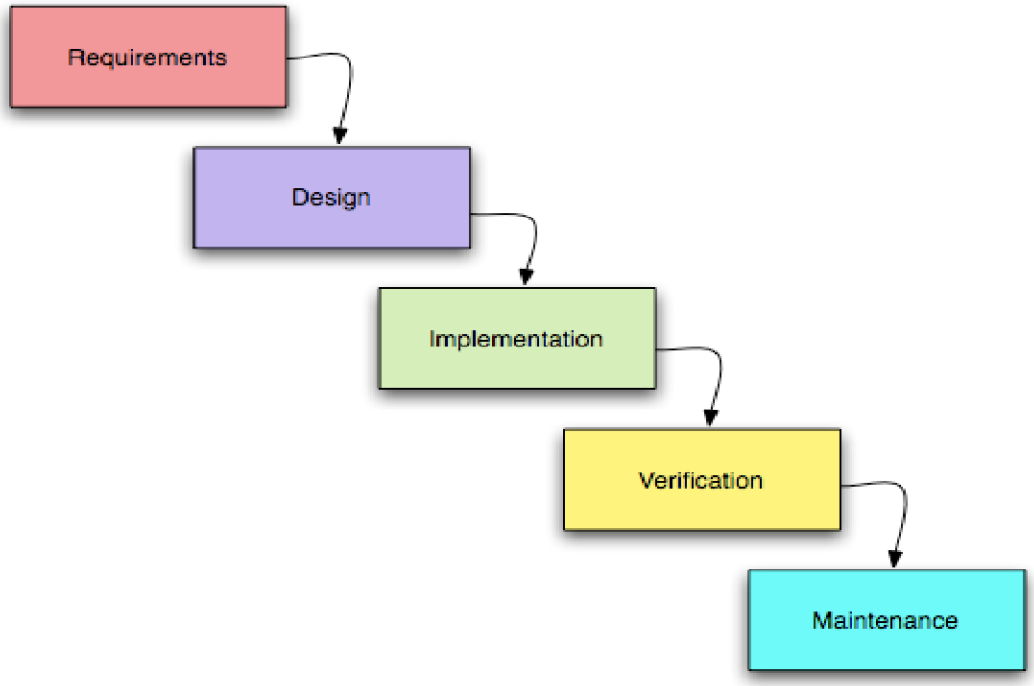


Antikythera mechanism

- A famous example of reverse engineering
- Ancient mechanical computer
- Discovered in a wreck in 1900, dated around 150–100 BC

Development Cycle

- The waterfall model
- Reverse Engineering moves through this process in reverse.
- May not end up with the same implementation.

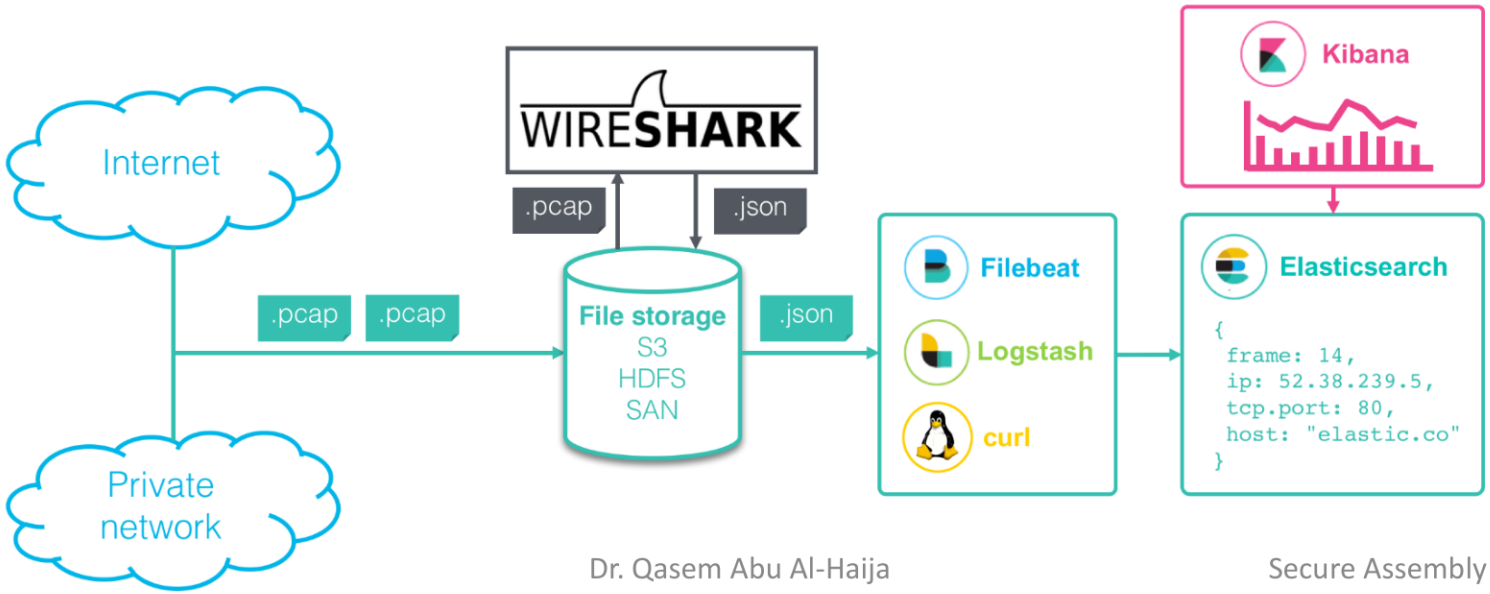


Software Techniques

- Analysis through observation of information exchange
- Disassembly
- Decompilation

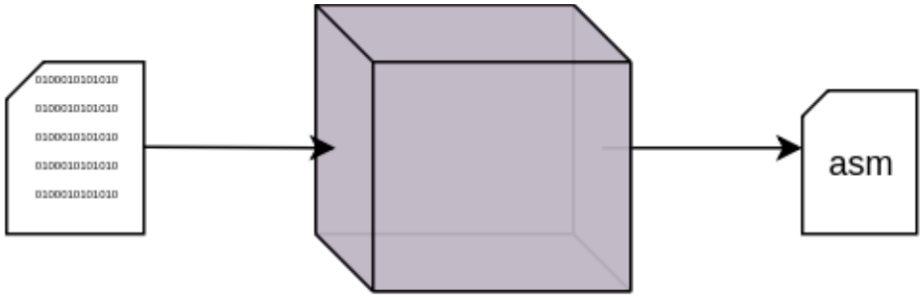
Analysis Through Observation

- Very common for protocol reverse engineering.
- Usually use a bus analyzer and or packet sniffers.
- Can be assisted through the use of low-level debuggers
- Example of tools: SoftICE, WireShark, …



Disassembly

- Most programs, when compiled, are turned into architecture-specific machine code.
- Disassemblers take the binary executable and display its assembly code.
- Need a good understanding of assembly and usually a hex editor.
- Example of tools: W32Dasm, IDA Pro, ...

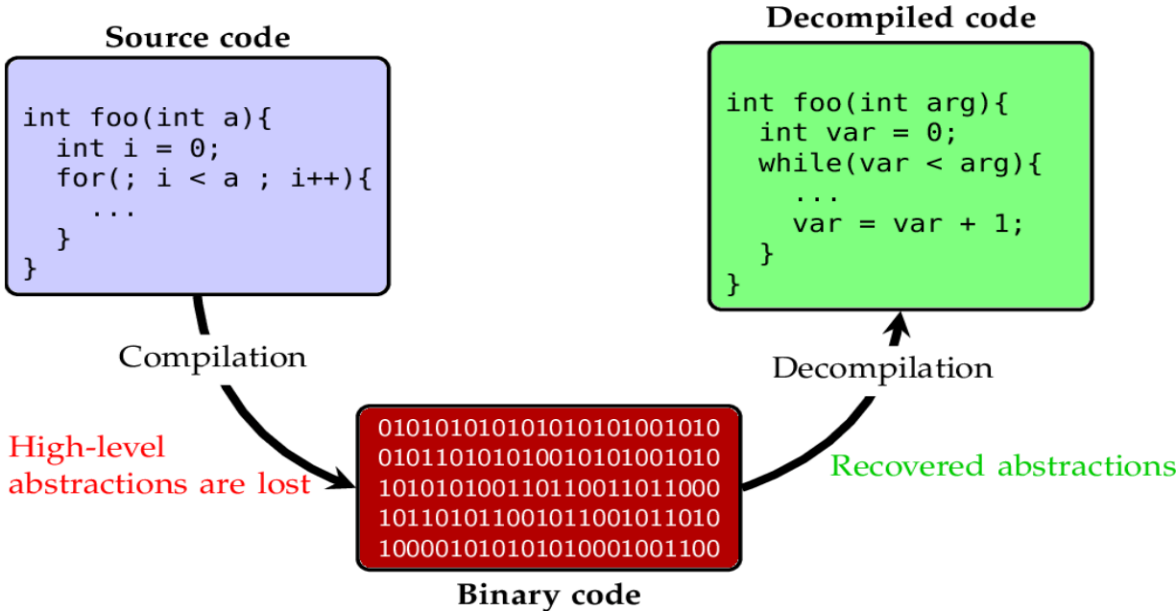


(Executable Binary) (Disassembler) (Assembly Code)

Decompilation

- A decompiler is a computer program that translates an executable file to a high-level source file that can be recompiled successfully.
- It is the opposite of a typical compiler, which translates a high-level language to a low-level language.
- Example of tools:

  Mocha, JAD,...



Motivations of Reverse Engineering

Motivation of RE

- Interoperability
- Lost documentation
- Product analysis
- Security auditing
- Removal of access restrictions
- Creation of duplicates
- Fraud

Interoperability

- Getting a device/piece of software to work on another platform.

- Example: Reversing systems developed for windows to work over Unix environment

Lost Documentation

- Need to re-learn how the device operates, how the device communicates
- Usually only done on antiquated devices or integrated circuits

Product Analysis

- To determine how the product works
- Can be used to estimate product costs
- Check product legalities: Determine if a product infringes on patent rights.

Security Auditing

- An audit determines if systems safeguard assets, maintain data integrity, and operate effectively.
- The company usually knows about its own products.
- Used to evaluate the risk of new products it may create or use from other companies.

Access Restriction Removal

- Possible legal issues
- Usually done to demo programs, the full version released as warez
- Sometimes, it becomes legal when a program or game becomes very old.

Create Duplicates

- This can be very difficult, trying to reproduce the entire system.
- Reverse engineering of copy restrictions on CDs and other media.
- In certain cases, the user is allowed a duplicate.

Fraud

- Any system (usually embedded or integrated) that stores critical information
- Most common example is credit cards / smart cards
- Passwords and other information are often stored on the card

Reverse Engineering Tools of Software Systems

Topics

- Basic background on assembly language
- Types of reverse engineering tools and demonstrations of these tools:
  - **Hex editors**: WinHex, Tsearch
  - **Decompilers**: REC, DJ
  - **Disassemblers/Debuggers**: IDAPro, OllyDbg, Win32Dasm, BORG

Program Abstractions

Computers understand binary code

Binary code can be written in hexadecimal

Hexadecimal code can be encoded in assembly language

Assembly language is human-readable but not as intuitive as source code

Decompilers convert assembly into an easier-to-read source code

11001111 10101 == CD21 == int 21

Assembly language is an abstraction of hexadecimal code

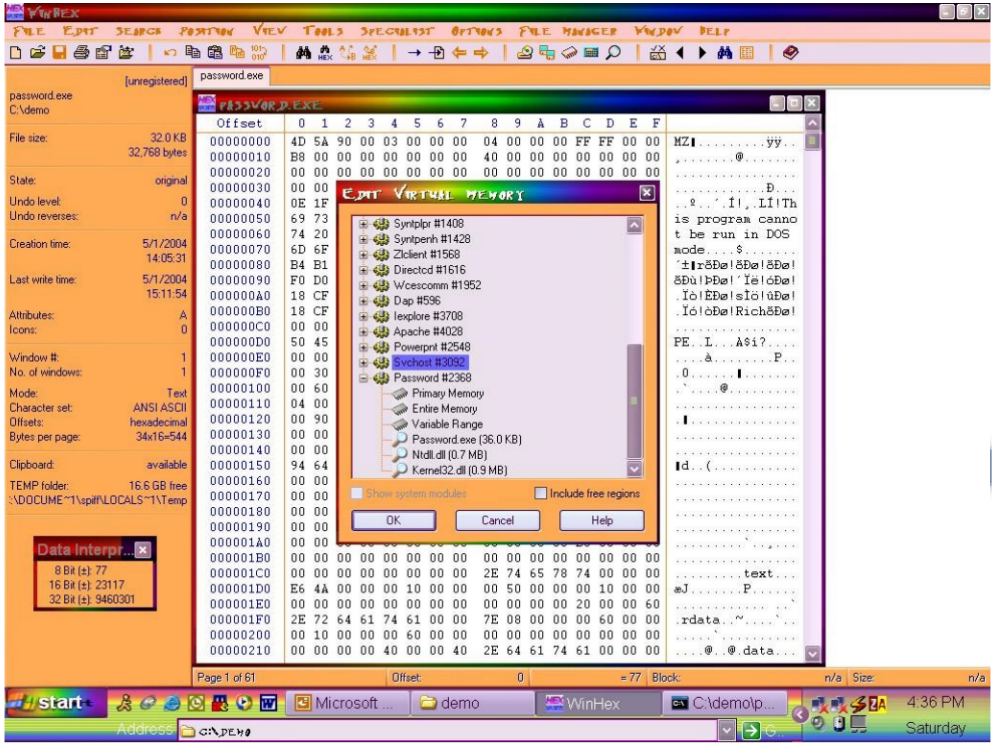
```
C:\>debug
-a
0B0C:0100 mov ah,9
0B0C:0102 mov dx,109
0B0C:0105 int 21
0B0C:0107 int 20
0B0C:0109 db "Hello World$"
0B0C:0115
-nhello.com
-rcx
CX 0000
:115
-w
Writing 00115 bytes
-q

C:\>hello
Hello World
C:\>debug hello.com
-d
0B69:0100  B4 09 BA 09 01 CD 21 CD-20 48 65 6C 6C 6F 20 57   ......!. Hello W
0B69:0110  6F 72 6C 64 24 75 30 80-CF 02 80 3E 34 00 FB 0A   orld$u0....>4...
0B69:0120  04 EB 24 EB 78 B4 07 80-3E 35 99 00 74 02 B4 02   ..$.x...>5..t...
0B69:0130  B0 3F 2A 26 34 99 72 EB-86 E1 E3 09 86 E1 E8 C8   .?*&4.r.........
0B69:0140  00 86 E1 E2 F7 86 E1 E8-15 E3 75 21 80 CF 04 80   ..........u!....
0B69:0150  3E 6E 99 00 74 05 F6 C7-02 75 48 89 3E 32 99 FF   >n..t....uH.>2..
0B69:0160  06 32 99 C6 06 34 99 FF-C6 06 35 99 00 E8 99 00   .2...4....5.....
0B69:0170  AC E8 61 E2 74 38 3C 0D-74 34 3A 06 02 96 74 2E   ..a.t8<.t4:...t.
-u
0B69:0100 B409          MOV     AH,09
0B69:0102 BA0901        MOV     DX,0109
0B69:0105 CD21          INT     21
0B69:0107 CD20          INT     20
0B69:0109 48            DEC     AX
0B69:010A 65            DB      65
0B69:010B 6C            DB      6C
0B69:010C 6C            DB      6C
0B69:010D 6F            DB      6F
0B69:010E 20576F        AND     [BX+6F],DL
0B69:0111 726C          JB      017F
0B69:0113 64            DB      64
0B69:0114 2475          AND     AL,75
0B69:0116 3080CF02      XOR     [BX+SI+02CF],AL
0B69:011A 803E3400FB    CMP     BYTE PTR [0034],FB
0B69:011F 0A04          OR      AL,[SI]
-_
```

Hex Editors

- Hex editors read executing programs from RAM.
- Display their contents in hexadecimal code.
- Enable the editing of the running hexadecimal code.

Example: WinHex

(http://www.sf-soft.de/)



Decompilers

- Decompile binary programs into readable source code.
- Replace all binary code that could not be decompiled with assembly code.

Example: REC

(http://www.backerstreet.com/rec)

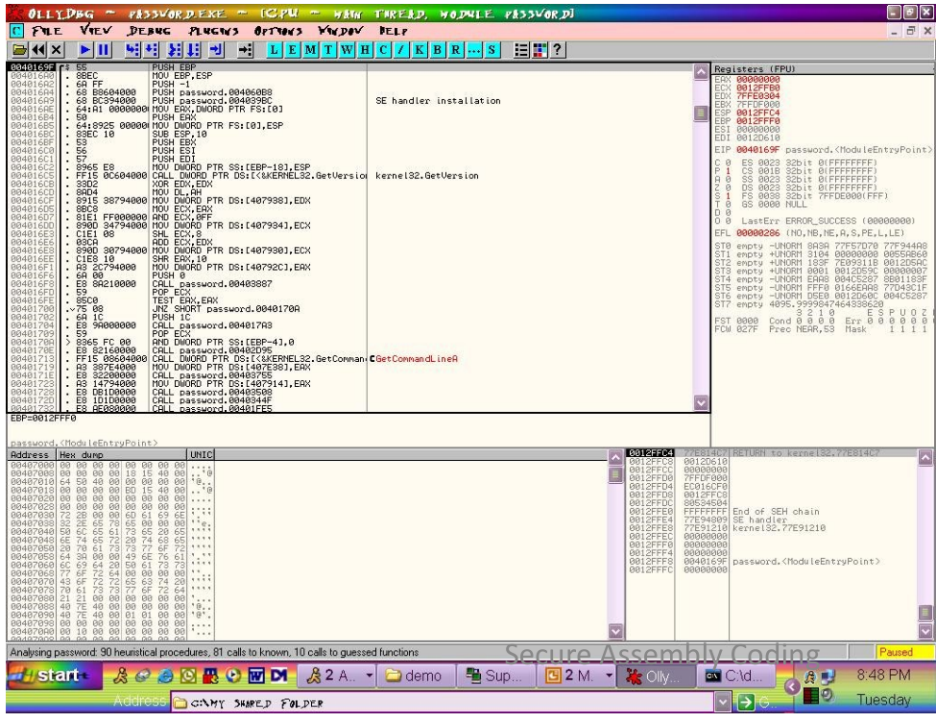
- Decompiles a program from binary code to C pseudo-code.
- Translates any binary it cannot decompile into assembly code.
- Typically generates about 60-70% of the program source code.

Disassemblers/Debuggers

- Convert binary code into its assembly equivalent.
- Extract ASCII strings and used libraries.
- View memory, stack, and CPU registers.
- Run the program (with breakpoints).
- Edit the assembly code at runtime.

Example: OllyDbg

http://home.t-online.de/home/Ollydbg/



Disassemblers/Debuggers Programs & Features chart

Product	Dis-Assembly	Processor options	Debugger	String extraction	Disk Hex editor	Memory Hex editor	Memory Dumper	Library's used	Decryptor
IDAPro	x	x		x	x			x	
OllyDbg	x	x	x	x	x	x	x	x	
W32Dasm	x	x	x	x	x	x	x	x	
BORG	x	x					x		x

Reverse Engineering Prevention Tools

"Code Obfuscators"

Such as Y0da's Cryptor, NFO

Code Obfuscation

- The process of modifying an executable so that it is no longer useful to a hacker but remains fully functional.
  - Modify actual method instructions or metadata
  - Does not alter the program's output.

- However, with enough time and effort, almost all code can be reverse-engineered.

- The goal is to distract the reader with the complicated syntax of what they are reading and make it difficult for them to determine the true content of the message.

Code Obfuscation can be done in several ways.

❑ Example#1: Rename Obfuscation

- ➢ Use naming that make the code difficult for the reader to understand.

Original Source Code Before Rename Obfuscation	Reverse-Engineered Source Code After Rename Obfuscation

private void
CalculatePayroll (SpecialList employee-
Group) {
while (employeeGroup.HasMore()) {
employee =
employeeGroup.GetNext(true) ;
employee.UpdateSalary() ;
Distribute Check(employee) ;
}
}
	private void a(a b) {
	while (b.a()) {
	a = b.a(true);
	a.a () ;
	a.(a) ;
	}
	}

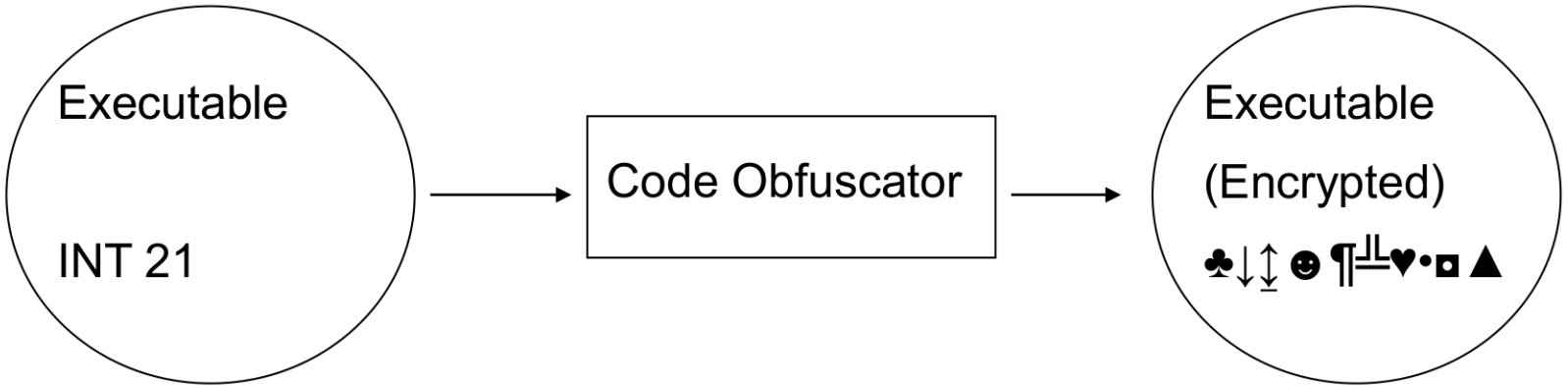
❑ Example#2: String Encryptions

- ➢ Encrypting the code of a program so you cannot view it in assembly.

Original Source Code Before String Encryption	Reverse-Engineered Source Code After String Encryption
... MessageBox.show("Invalid Authentication - Try Again") ...	... MessageBox.show(a.b("¥∑¸†§ƒжнжк•₡")) ...

Code Obfuscators/Encryption tools

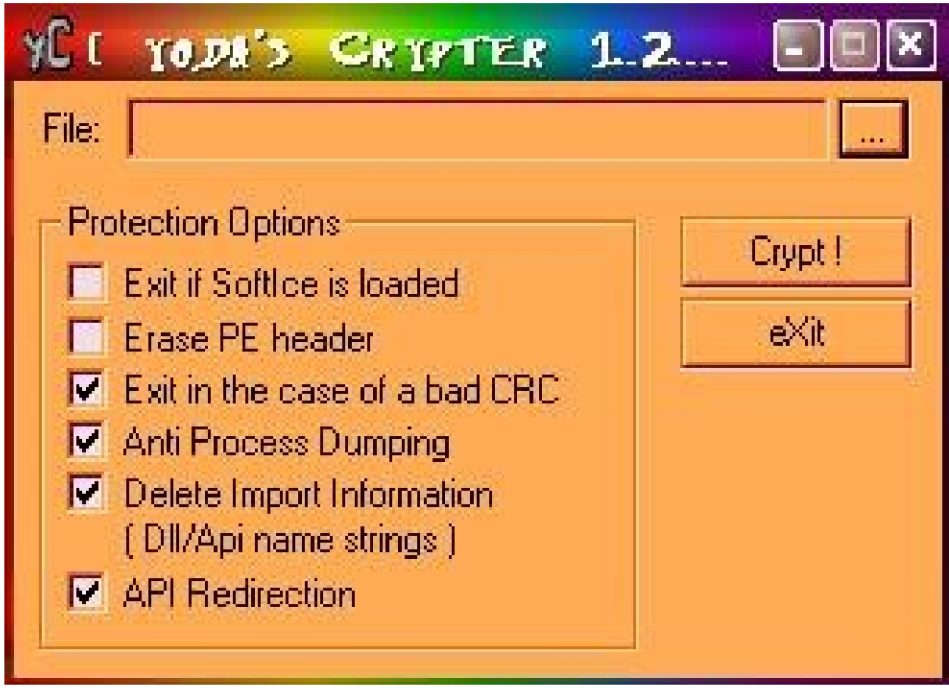
- Encrypts the code of a program so you cannot view it in assembly.



Obfuscators	Obfuscation	Anti-debugging techniques	GUI
Y0da's Cryptor	x	x	x
NFO	x	x	

Example: YOda's Cryptor

- Code Obfuscation.
  - Encrypts the code of a program.
- Anti-Debugging.
  - Detects all major debuggers and disassemblers.
- GUI platform.
  - Graphical user interface.



y0da.cjb.net

Main Sources for these slides

- *K. R. Irvine. Assembly Language for x86 Processors, 8th edition, Prentice-Hall (Pearson Education), June 2019. ISBN: 978-0135381656.*

- *B. Dang, A. Gazet, E. Bachaalany. Practical Reverse Engineering: x86, x64, ARM, Windows® Kernel, Reversing Tools, and Obfuscation. John Wiley & Sons, June 2014. ISBN: 978-1-118-78731-1*

- *Qasem Abu Al-Haija, "Microprocessor Systems", King Faisal University, Saudi Arabia*

- *Ghassan Issa, "Computer Organization", Petra University, Jordan.*

Thank you