

A decorative graphic on the left side of the slide, consisting of a network of white lines and small circles on a dark blue background, resembling a circuit board or a neural network.

ANTI-REVERSING TECHNIQUES

WHAT ARE WE CONCERNED ABOUT??

- If it's not open source, there's probably a reason why
 - Copyright protection
 - Intellectual property
 - Digital Right Management
 - Malware authors don't want others defeating their code
- Without anti-reversing techniques, software is essentially open source
- What about hardware reversing??
- Never say never – cannot entirely prevent reversing. Can only hope to contain it!!

BASIC APPROACHES TO ANTI-REVERSING

- Eliminate Symbolic Information
 - Eliminate obvious textual info
- Obfuscate Program
 - Prevent static analysis
- Embedded anti-debugging code
 - Hinder live analysis



ELIMINATE CLUES

- Eliminate symbolic information like names of functions and global variables
 - Rename to meaningless things
- Hide the true contents of strings
 - Encryption can help here



OBFUSCATION OF CODE

- Encryption after compilation
 - Hinders static analysis
- However a skillful reverser could capture the code during runtime when the code has to be decrypted



A golden key is positioned diagonally across the center of a grid of hexadecimal code. The key has a classic notched bit and a simple bow. The background is a dark blue gradient with faint circuit-like patterns in the corners.

03003802	996CB7BA	0EG0161B	G0021C06
BA7CE203	G0030200	01208600	37D14D00
1B7125G0	024FG002	53D03C00	AD722500
BD03C00	887525C1	01A07700	37D14D00
B7125G0	024FG002	53D03C00	AD722500
BD03C00	887525C1	4F553F	53414242
F4F3D41	4242434E	3D4A6	2 6469204
6C2F4F	553D4553	414	7 4F3D414
425604	00312E30	424	1 0003424
003042	4CC	8 024E4E4F	00B1D3
2254F1	21	509 8833B0CC	2957EE
3ECAA	CB3EE8EF	DF038D7F	A14217
2AA4D	04143B75	4F571C83	535C04
7DED9	B57C659E	C820EE07	FA49F
96DB	7D7F743D	9A36DD29	454E0
014D	410800C8	9A54E072	5A14C

PACKERS (GO GREEN BAY)

- A *packer* compresses or encrypts the original contents of a binary, and replaces the main program with a stub that unpacks it
- Originally designed to save space in small systems, now used to hide the contents of a program.
- More complicated packers will only unpack small portions of the original program at a time.
- Often used by malware

STATIC VS DYNAMIC ANALYSIS

- Everything on the previous slides thwarts static analysis
 - The instructions and the data have to be available at runtime
 - What tool did we work with that performs dynamic analysis?
 - Just attach your favorite debugger!

ANTI-DEBUGGING TECHNIQUES

- Detecting and countering debuggers is tricky
 - Very platform and OS specific
 - Both failure modes are undesirable – failing to detect an attached debugger and falsely determining that a debugger is attached
- You can usually just ask an OS (via a well documented API) whether a debugger is present
 - This API call is very easy for reverse engineers to find

DEBUGGING TECHNIQUES

- How does a debugger pause your program?
- When a breakpoint is hit, this causes an interrupt 3 to occur
- The current instruction that is occurring is replaced by an INT 3
- Once this occurs the program is frozen so state can be inspected
- When the breakpoint is hit, the debugger can inspect RIP to see where the interrupt occurred, look up and restore the overwritten byte, and have execution continue with the original instruction

PREVENTING CODE MODIFICATION

- Examine your program's text at runtime for new 0xCC bytes – these are bytes that indicate an INT 3 occurred
- Compute checksums of your program's text at runtime
- Frequently compare timestamps
- If normal execution times are on the order of milliseconds, a several second delay probably means that somebody is inspecting program state in a debugger

YOU CANNOT PREVENT REVERSING

- You can only hope to slow it down
- ASLR – Address Space Layout Randomization
 - Randomizes the location where executable functions are loaded into memory
 - Helps prevent buffer overflow attacks
- DEP – Data Execution Prevention
 - Locks down what sections of the stack are executable



SEPARATION OF CODE

- Apple's Secure Enclave
 - Separate processor for specialized code
 - Runs it's own microprocessor which is not accessible by the OS
- Code segmentation
 - What barriers are in place to prevent access to code

CONTINUED PREVENTION

- Checksums
- Confusing disassemblers
 - There are many ways programming can program loops, lists, and others common structures that can confuse dissemblers.
 - Reversing aware programmers can sufficiently confuse reversers
 - Inject arbitrary code