



Reverse Engineering and Obfuscation

Fengwei Zhang



What is Reverse Engineering?

- Reverse Engineering is a popular hacking approach that extracts the knowledge and design of a system and reproduces its behavior based on the extracted information
- Reverse Engineering is the process of analyzing a subject system to create representations of the system at a higher level of abstract
- Going backwards through the development cycle
 - Disassembly
- Broader explanation: Nuclear weapons

What is Obfuscation

- To prevent the reverse engineering, we often use the obfuscation to raise the bar of the difficulty of this process
- Examples of Obfuscation
 - Encryption, dynamic loading, reflection
 - More packing techniques
 - Commercial packers including 360 [11], Alibaba [7], Tencent [12], Baidu [8], and Bangcle [48] (References are from DexLego work)
 - DexLego: <https://fengweiz.github.io/paper/dexlego-dsn18.pdf>



RE Applications

- White hats
 - Malware analysis on x86 and ARM
 - Vulnerability discovery: Intel Management Engine
- Black hats
 - Cracking, hacking
 - Malware re-engineering (shell code injection/reuse)
 - Lab 5 covers Android applications
 - Vulnerability discovery (exploitation)

Obfuscation Applications

- White hats
 - Copyright issues
 - Protect the intellectual properties of commercial products
 - Android and iOS apps, Windows OS/Office, etc
- Black hats
 - Malware writers use obfuscation to create advanced malware that cannot be analyzed by security experts
 - Multiple layers of obfuscation

RE Approaches

- Behavior analysis
 - Executing the target program within an isolated execution environment (VM)
 - Registry (RegShot)
 - Files (DiskMon, FileMon)
 - Network (Wireshark)
 - API and system services/calls (EasyHook)
- Code Analysis
 - Studying the code of the target program
 - Identify the packing technique and then unpack the target
 - Disassemble, analyze call/data flow graph
 - Debug, decrypt, and reveal actual values
 - Patch binary to traverse hidden code branches



Malware RE Scheme

1. Create isolated experimental environment (a VM)
2. Submit malware to existing sandboxes (e.g., Anubis)
 - Inspect its high-level behavior
 - Watch file droppers and created processes
3. Examine its imports, exports and strings
 - Based on import API, guess malware type
4. Identify packer and unpack (manual or auto)
5. Disassemble/decompile malware
 - Trace API usage (context, constant attributes)
6. Debug binary
 - Trigger conditions
 - Resolve implicit jumps (control flow)
7. Patch binary
 - Execute hidden payload

Case Study: DexLego

- What is DexLego?
 - It is a tool for unpacking
 - Working against packers including 360 [11], Alibaba [7], Tencent [12], Baidu [8], and Bangcle [45]
 - Reverting (Reverse Engineering) the Obfuscation techniques
 - Based on Android apps
 - Lab 5 is based on Android apps as well
 - <https://fengweiz.github.io/paper/dexlego-dsn18.pdf>