Extra Reading – Week 06 – Networks and System Security

Part 1: What is Reverse Engineering?

Definition

Reverse engineering is the act of dismantling an object to understand how it works. It involves analysing a finished product to gain knowledge about its design, function, and implementation.

Core Purpose

- Analyse and gain knowledge about how something works.
- Duplicate or enhance existing objects.
- Learn from competitors' products.
- Recreate obsolete or unavailable products.
- Address compatibility issues.
- Conduct security analysis.
- Educational purposes

Applications

- **Software:** Machine code analysis, source code recovery
- **Hardware:** Circuit analysis, chip reverse engineering
- **Physical machines:** Mechanical component analysis
- **Military technology:** Weapons systems analysis
- **Biological functions:** Gene operation understanding.

Part 2: The Reverse Engineering Process

Three General Steps

1. Information Extraction

- Study the object being reverse engineered.
- Extract design information.
- Examine how components fit together.
- **In software:** Gather source code, design documents, use disassemblers.
- Break program into constituent parts.

2. Modelling

- Abstract collected information into conceptual model.

- Explain each component's function in overall structure.
- Create generalized model from specific information.
- **In software:** Data flow diagrams, structure charts
- Guide design of new objects/systems.

3. Review

- Review and evaluate the model.
- Ensure realistic abstraction of original.
- **In software:** Software testing procedures
- Validate before implementing to reengineer original.

Key Tools & Techniques

Software RE Tools

- **Hexadecimal Dumper:** Displays binary numbers in hexadecimal format.
- **Disassembler:** Reads binary code, displays executable instructions as text
- **Debugger:** Prevents disassembling data portions of programs
- **CAD Software:** For physical object recreation (3D imaging)

Part 3: Software Reverse Engineering Examples

Common Use Cases

1. Platform Adaptation

- Adapt programs from one microprocessor to another.
- Create cross-platform compatibility.

2. Source Code Recovery

- Reconstruct lost source code.
- Work with legacy systems.

3. Performance Analysis

- Study how programs perform operations.
- Improve performance and efficiency.
- Fix bugs when source unavailable.

4. Compatibility Creation

- **Example: Phoenix BIOS**
    - Created BIOS compatible with IBM's proprietary version.
    - Used clean room technique to avoid copyright issues.

    - Documented steps without referencing proprietary code.

5. Malware Analysis

- Identify obfuscated malicious code.
- Understand virus and malware operations.
- Create defensive or offensive cyberweapons.
- **Example: NSA's Ghidorah** used to analyze WannaCry ransomware

6. Hardware Analysis

- Analyse competitor processors
- Create replacement parts for legacy equipment.
- Enhance security (e.g., Google Project Zero identifying CPU vulnerabilities)

7. Network Security Assessments

- Red team vs. blue team exercises
- Simulate attacks and reverse engineer them.
- Strengthen corporate network defences.

Part 4: Legal & Ethical Considerations

Legal Framework (U.S.)

Generally Legal When:

- Original version obtained legally
- No contractual agreements broken.
- Intended to improve product or create interoperability.
- Protected under Defend Trade Secrets Act (with exceptions)

Potentially Illegal When:

- Copying patented products without authorization
- Violating copyright law through duplication
- Breaking software license agreements
- Circumventing technological protection measures (TPM)
- Violating nondisclosure agreements

Relevant Laws

1. Patent law
2. Copyright and fair use law
3. Trade secret law

4. Digital Millennium Copyright Act (anti-circumvention provisions)
5. Electronic Communications Privacy Act
6. Specific contract law

Clean Room Technique

- Two separate programmer groups
- Ensures original is not directly copied.
- Avoids patent/copyright infringement.
- Maintains ethical wall between teams.

Part 5: Essential Books for Malware Analysis & RE

Foundational Texts

1. **Practical Malware Analysis** - Michael Sikorski & Andrew Honig
    - Definitive guide for all levels
    - Covers static and dynamic analysis.
2. **The IDA Pro Book** - Chris Eagle
    - Master the industry-standard disassembler.
    - Essential for professional malware analysis
3. **Malware Analyst's Cookbook** - Michael Ligh et al.
    - Practical techniques and recipes
    - Firsthand approach to malware dissection
4. **Rootkits: Subverting the Windows Kernel** - Greg Hoglund
    - Kernel-level malware understanding
    - OS manipulation techniques
5. **Practical Reverse Engineering** - Bruce Dang
    - Real-world problem solving
    - Firsthand guidance

Advanced Resources

6. **Mastering Malware Analysis** - Alexey Kleymenov
    - Advanced dissection techniques
    - Real-world sample analysis
7. **The Art of Memory Forensics** - Michael Hale Ligh
    - Memory-based malware detection

    - System memory analysis
8. **Windows Internals, Parts 1 & 2** - Mark Russinovich
    - Essential OS understanding
    - Windows-targeting malware comprehension
9. **The Art of Computer Virus Research and Defence** - Peter Szor
    - Virus foundations
    - Défense mechanisms

Specialized Topics

10. **Gray Hat Python** - Justin Seitz
    - Python automation for RE
    - Custom tool scripting
11. **Reversing: Secrets of Reverse Engineering** - Eldad Eilam
    - Fundamental RE concepts
    - Classic text
12. **The Shell coder's Handbook** - Chris Anley
    - Exploit understanding
    - Malware exploitation techniques
13. **Malware Data Science** - Joshua Saxe
    - Machine learning integration
    - Data science for threat detection

Part 6: Essential Tools

Disassemblers & Debuggers

Industry Standards

- **IDA Pro:** Gold standard for binary disassembly
- **Ghidra:** Free NSA-developed suite, growing popularity.
- **Binary Ninja:** Modern, automation-focused
- **Radare2:** Powerful open-source framework

Debuggers

- **x64dbg:** Open-source, 32/64-bit Windows debugging
- **OllyDbg:** Classic 32-bit Windows debugger
- **Immunity Debugger:** Python scripting support

- **Hopper:** macOS/Linux binary analysis

Static Analysis Tools

- **Detect It Easy (DIE):** Packer/compiler identification.
- **PE-sieve:** Memory injection detection
- **APKiD:** Android obfuscation detection
- **YARA:** Malware signature creation and matching

Dynamic Analysis Tools

- **Cuckoo Sandbox:** Automated malware analysis
- **CAPE Sandbox:** Unpacking and ransomware analysis.
- **Any. Run:** Interactive real-time sandbox
- **Hybrid Analysis:** Free online analysis service
- **Joe Sandbox:** Advanced automated analysis

Memory & Process Analysis

- **Volatility:** Memory dump analysis (essential)
- **Process Hacker:** System process monitoring.
- **Procmon:** Real-time Windows activity monitoring
- **Fakenet-NG:** Network service simulation

Specialized Tools

- **Shellcode_Emulator:** Safe shellcode analysis
- **dnSpy:** .NET reverse engineering
- **Wireshark:** Network traffic analysis

Part 7: Practice Platforms

Malware Analysis Platforms

1. **MalwareBazaar:** Real-world malware samples
2. **Any.Run:** Interactive sandbox with real-time feedback
3. **Hybrid Analysis:** Free detailed analysis reports
4. **Joe Sandbox:** Professional-grade automated analysis
5. **VirusShare:** Extensive malware sample archive
6. **Flare-On Challenge:** Annual FireEye RE competition.

Reverse Engineering Challenges

1. **Crackmes.one:** RE challenge collection

2. **Pwnable.kr:** Binary exploitation focus
3. **Root-Me:** Multi-area cybersecurity challenges
4. **TryHackMe:** Firsthand malware analysis labs
5. **Hack The Box:** RE and exploitation labs.

Part 8: YouTube Learning Resources

Live Malware Analysis

- **OALabs:** Malware breakdowns, CTF challenges
- **HackerSploit:** Comprehensive security tutorials
- **Jaybailey216:** Practical RE examples
- **Amr Thabet (MalTrak):** Malware analysis techniques
- **LiveOverflow:** CTF challenge solutions
- **BlacKSilence:** Arabic malware analysis content

Reverse Engineering Tutorials

- **Reverse Engineering with Peter:** In-depth crackme solutions
- **stacksmashing:** Hardware and software RE
- **Almond Force:** Real-world RE challenges
- **John Hammond:** CTF walkthroughs
- **HEXORCIST:** Advanced binary analysis
- **All Things IDA:** IDA Pro mastery

Beginner-Friendly Channels

- **Null Byte:** Beginner tutorials
- **VirginiaCyberRange:** Free cybersecurity content
- **LetsDefend:** Blue team skills
- **HackerEnv:** Broad security topics
- **Low Level Learning:** Low-level programming focus

Part 9: Why This Matters

The Evolving Threat Landscape

- Malware becomes more intricate daily.
- Ransomware holds data hostage.
- Advanced Persistent Threats (APTs) steal sensitive information.
- New attack vectors constantly emerge.

Critical Skills Needed

For Cyber Defenders:

- Understand threat capabilities.
- Develop effective countermeasures.
- Rapid incident response

For Ethical Hackers:

- Identify vulnerabilities.
- Evaluate security controls.
- Develop exploits responsibly.

For Forensic Analysts:

- Investigate security incidents.
- Recover evidence.
- Attribute attacks

For Researchers:

- Stay ahead of threat actors.
- Develop new detection methods.
- Advance security science