Digital Forensics
Lecture 01- Reverse Engineering

An Introduction to Reverse Engineering

Akbar S. Namin
Texas Tech University
Spring 2018

Reverse Engineering

- A process where an engineered artifact (e.g., car, jet engine, or a software program) is deconstructed in a way that reveals its inner-most details, such as its design and architecture.
- The process of extracting the knowledge or design blueprint from anything man-made
- In software domain, it refers to “binary reverse engineering”
  - Extracting valuable information from programs for which source code is unavailable

What We Can Do with RE?

- Answers to questions like …
  - What is causing a program to crash?
  - What is causing a program behave differently than expected?
  - Where is the root cause of the misbehaved program?
  - Is there anything interesting in a given file?
  - What does a file do?
  - How can we execute a file?
  - What happens to an input arrived through a user, file, or even network?
  - How do I get a program to do action X?
  - How can we use paths of a given program and adopt them in our own program?
  - Can we make this more efficient?

What We Can Do with RE?

- RE is particularly useful in modern software analysis for:
  - Finding malicious code.
    - E.g. virus and malware detection techniques
  - Discovering unexpected flaw and faults.
    - Detect bugs and defects introduced by “forward engineering”
  - Finding the (other) use of others’ code.
    - How and where protected code are used in applications
    - Enables the detection of code replication issues
  - Learning from others’ products.
    - Enables the the study of advanced software approaches
  - Discovering features or opportunities that the original developers did not realize.
    - Code complexity can foster new innovation
  - Deciphering undocumented machine code
    - Understating the meaning and purpose of undocumented binary code
    - Understand low-level software, and learn techniques to dig into program’s binaries

Definition of RE

- Reverse Engineering
  - A critical set of techniques and tools for understanding what software is really all about.
  - “The process of analyzing a subject system to identify the system’s components and their interrelations and to create representations of the system in another form or at a higher level of abstraction” (IEEE 1990)
  - Allow us to visualize the software’s structure, its ways of operation, and the features that drive its behavior.
  - Gives us a reasonable way to comprehend the complexity of the software

Common Applications of RE in the Software World

- Security-related RE
  - Employed in encryption research
  - Used in connection with malicious software
  - Is very popular with cracking copy protection schemes
- Software development-related
  - Employed to discover how to interoperate with undocumented software
  - Used to determine the quality of third-party code
  - Used for extracting valuable information from a competitor’s product for the purpose of improving technologies

Security-Related RE

- Malicious software
  - RE is used in detecting and analyzing malicious software
  - Locate vulnerabilities in operating systems, and also other software
  - Dissect and analyze malicious programs
  - Trace and debug every step the program takes and assess the damage it could cause
- Reversing cryptographic algorithms
  - Grouped into two groups:
    - Restricted algorithms
      - No key is used
      - Rather algorithms are used to shift letters
      - Very poor security . Once the algorithm is exposed nothing is secure
    - Key-based algorithms
      - The use of key that is kept private; whereas, the algorithm is made public

Security-Related RE

- Digital rights management
  - Products can be duplicated and piracy has become common practice (e.g., digital medias such as music, software applications,
  - Digital Rights Management (DRM) technologies.
    - Controlling the distribution of digital content
- Auditing program binaries
  - Open sources are safer, since they have been inspected by many
    - Vulnerabilities can be discovered and fixed
  - Binaries can be reversed in a limited way, but it can be done by professionals

Software-Development Related RE

- Achieving interoperability with binaries
  - When working with DLL or API libraries, documentation is almost always insufficient
    - Some developers may contact the vendors or may keep trying to get things work
    - Some others with RE skills will find it easier to reverse engineer the libraries
- Developing competing software
  - Most popular application of RE
  - Developing a software from scratch is easier,
  - However, developing some complex algorithms might need some RE skills
- Evaluating software quality and robustness
  - Estimate the general quality of the coding practices used in the binaries and not open sources
  - Try to assess the quality of programs as “Just trust my binaries”

Low-Level Software (System Software)

- A generic name for infrastructure
  - E.g., Compilers, linkers, debuggers, operating systems, assembly languages, etc.
- Understanding of low-level software and programming are needed for RE
- Reversing tools such as disassemblers and de-compilers never actually provide the answers
  - They only present the information
  - Extracting information is up to the reverser
- Low-level software
  - The bottom layer: millions of microscopic transistors
  - The top layer: some elegant looking graphics, a keyboard, and a mouse
- Reverses must be aware of anything that comes between the program source code and the CPU
  - Assembly languages, compilers, VM and byte-codes, and operating systems

Low-Level Software (System Software)

- Assembly languages
  - The lowest level in the software chain
  - Very suitable for reversing: the language of reversing
  - Machine code or binary code, or object code
    - Different representation of the same thing is represented by assembly language
    - A bits representation versus a textual representation by assembly language
  - Each command is represented by a number: operation code (opcode)
  - Each object is a sequence of opcodes
  - CPUs read object code from memory, decode it, and act based on the instruction embedded in it.
  - Assembler programs are used to translate the textual assembly language code into binary code
  - Disassembler reads object code and generates the textual mapping of each instruction in it.
    - Disassemblers are a key tool for reversers

Low-Level Software (System Software)

- Compilers
  - A program that takes a source file and generates a corresponding machine code file
    - Can be either a standard platform-specific object
    - Or platform-independent format (i.e., bytecode)
  - The biggest problem in deciphering compiler-generated code is the optimization
  - Optimization may be applied for minimization of the code size or improving execution performance
  - The resulting optimized code is difficult to read and comprehend

Low-Level Software (System Software)

- Virtual machines and bytecodes
  - Decoded by a program (VM) instead of a CPU
  - Advantage:
    - Platform independence

- Operating systems
  - A program that manages the computer
  - A kind of coordinator between the different elements in a computer

Reversing Process

- Generally done in two phases:
  - System-level reversing
    - A large-scale observation
    - Help determine the general structure of a program
    - Helps locate areas of interest within a program
    - Involves with:
      1. Running various tools and utilities on a program to obtain information
      2. Inspecting program executable
      3. Track program input and output
  - Code-level reversing
    - A more in-depth and artistic work
    - Observe the code from a very low-level
    - Offer detailed information on a selected code chunk
    - Extract design concepts and algorithms from a program binary

RE Tools

- System-monitoring tools
  - Tools for sniffing, monitoring, exploring, etc.
  - Monitor networking activity, file accesses, registery access, etc.
  - Display information gathered by the OS about the application
  - Display the use of mutexes, pipes, events, etc.
- Disassemblers
  - Programs that take a program’s executable binary as input and generate textual files containing the assembly language code
  - A processor-specific process

RE Tools

- Debuggers
  - A program that allows software developers to observe their program while it is running
  - Ability to 1) set breakpoints, 2) to trace through code
  - Disassembly mode: A debugger uses a built-in disassembler to disassemble object code on the fly
- De-compilers
  - Takes an executable binary file and attempts to produce readable high-level language code from it
  - Actual recovery of the original source code is not really possible

Is Reversing Legal?

- High-risk reversing projects need attention:
  - Interoperability
    - Exposing sensitive information through interfaces (to publish or not to publish)
  - Competition
    - Opponents of RE are worried about their technologies being stolen
    - E.g. stealing a segment of code and embedding into your own
    - More complex: apply a de-compiler to obtain a variation of the original code
  - Copyright law
    - Protect software and other intellectual property from unauthorized duplication
    - RE may violate copyright law because of the creation of “intermediate copies”
    - Creating intermediate copies is a violation of copyright law

Is Reversing Legal?

- Trade secrets and patents
  - Options for protecting an invention
  - “trade-secret misappropriation” – having a rough employee sell the secret to a competitor
- DMCA (The Digital Millennium Copyright Act)
  - Protect the copyright protection technologies (they are vulnerable)
  - It protects copyright protection systems from circumvention
  - What prohibited under DMCA?
    - Circumvention of copyright protection systems
    - The development of circumvention technologies
  - Exemptions in DMCA
    - Interoperability
    - Security testing
    - Educational institutions and public libraries
    - Government investigation
    - Regulation
    - Protection of privacy