Introduction to Reverse Engineering

REVERSE ENGINEERING

João Paulo Barraca

deti universidade de aveiro
departamento de eletrónica, telecomunicações e informática

What is Reverse Engineering (RE)

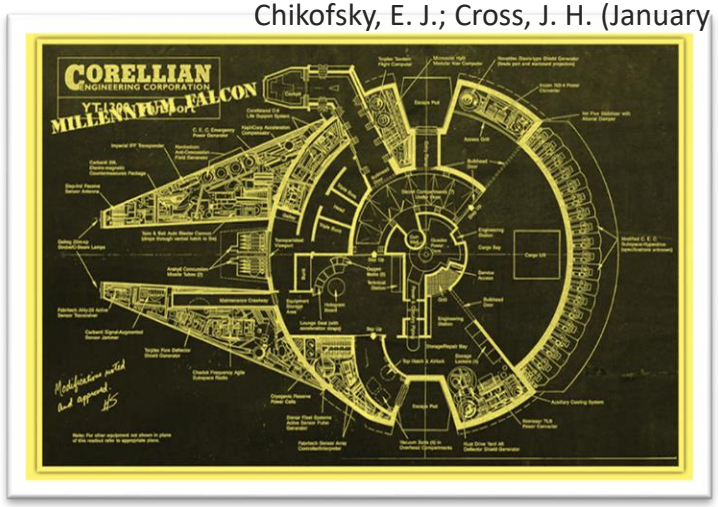
- Reverse Engineering (RE) is the process of extracting features from any man-made artifact (Engineered)
  - Knowledge
  - Design blueprints
  - Function

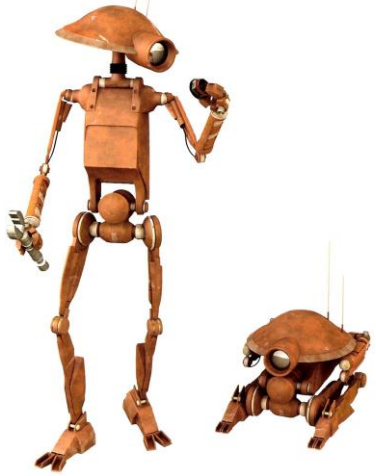
- It’s not purely scientific research: with RE the artifact was engineered
  - The scientific process doesn’t generically focus on a product
    - Focus is on mechanisms, processes, events, phenomena
    - … and we have no idea whether the universe was engineered or not ☺

What is Reverse Engineering (RE)

The process of **analyzing** a **subject system** to **identify** the system's **components** and their **interrelationships** and to **create representations** of the system in another form or at a higher level of abstraction

Chikofsky, E. J.; Cross, J. H. (January 1990). "Reverse engineering and design recovery: A taxonomy" (PDF). IEEE Software. 7: 13–17. doi:10.1109/52.43044





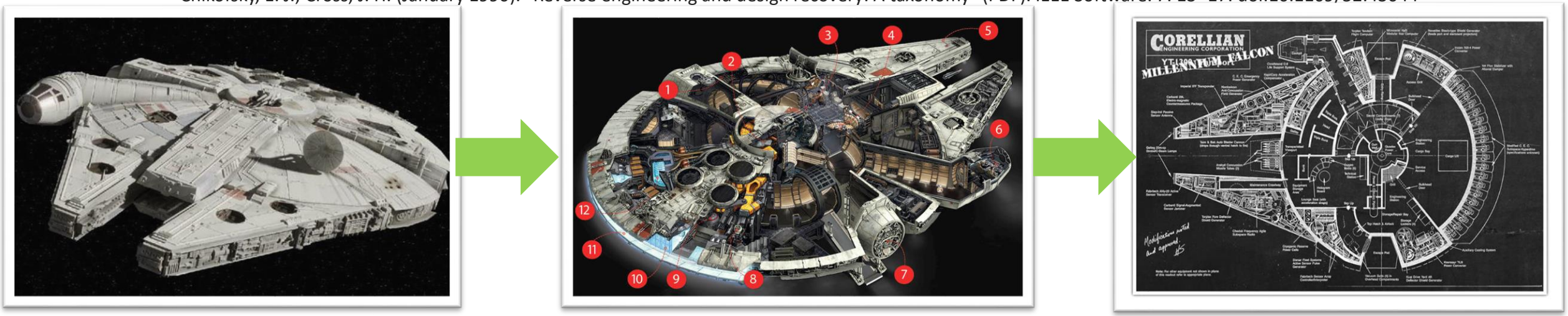
Forward Engineering

Images belong to their respective owners

What is Reverse Engineering (RE)

The process of **analyzing** a **subject system** to **identify** the system's **components** and their **interrelationships** and to **create representations** of the system in another form or at a higher level of abstraction

Chikofsky, E. J.; Cross, J. H. (January 1990). "Reverse engineering and design recovery: A taxonomy" (PDF). IEEE Software. 7: 13–17. doi:10.1109/52.43044



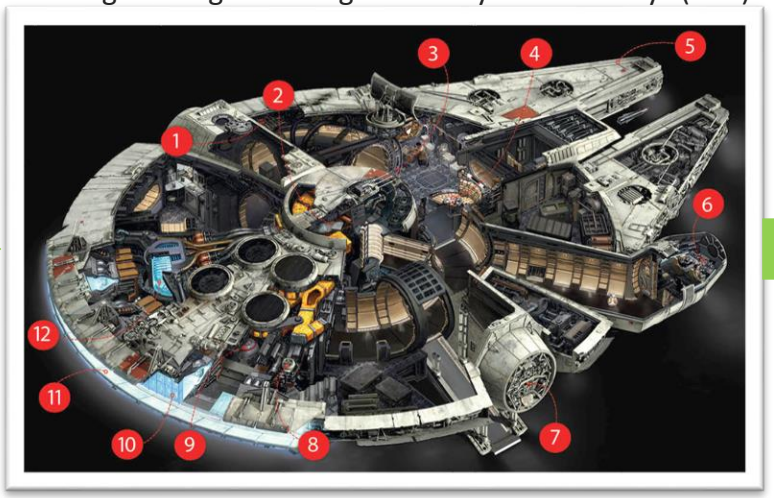
Reverse Engineering

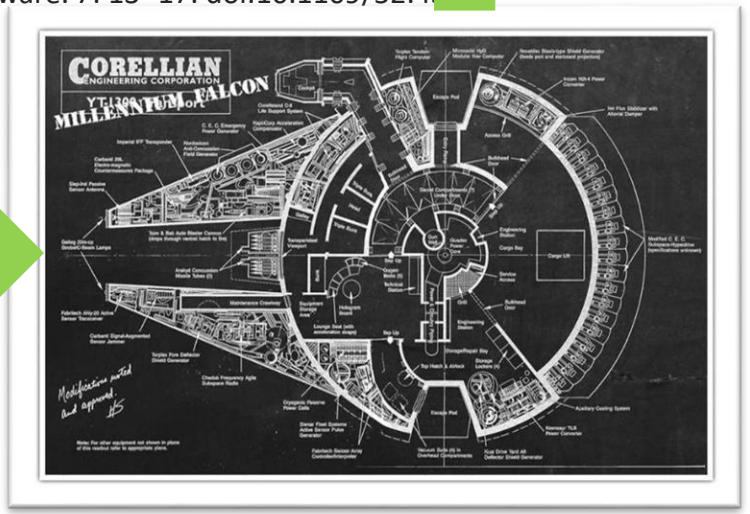
Images belong to their respective owners

What is Reverse Engineering (RE)

The process of **analyzing** a **subject system** to **identify** the system's **interrelationships** and to **create representations** of the system i higher level of abstraction

Chikofsky, E. J.; Cross, J. H. (January 1990). "Reverse engineering and design recovery: A taxonomy" (PDF). IEEE Software. 7: 13–17. doi:10.1109/52.4

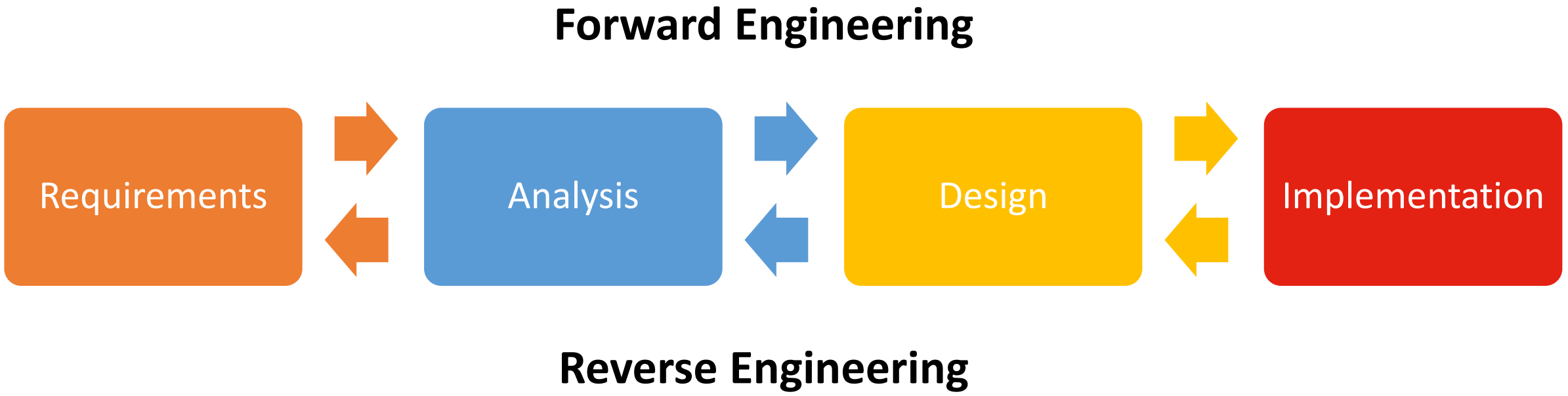




Reverse Engineering

Images belong to their respective owners

What is Reverse Engineering (RE)



- Processes are not perfect, in either direction.
- Implementation may not fully comply with requirements, while reversed engineered analysis may not fully represent the implementation design, and design will be limited

RE Concepts

- **Abstraction Level**
  - The result of a RE process will produce a design at a given abstraction level.
  - The higher the better
- **Completeness**
  - Level of detail at the abstraction level.
  - The greater the better
- **Interactivity**
  - How much humans are required for RE.
  - The lesser the better (higher automation)

When do we have RE activities?

- RE **always evolved with engineering** and existed since its dawn
  - It is frequently done informally by everyone in their daily lives
- Every time **we look at a software/device/system** and try to understand how it works, or understand any aspect of its behavior and structure
  - Because we want to make a better one
  - Because we wish to estimate if it suits a purpose…
- Every time we **look at our code** and try to find what it was supposed to do
  - Especially when there is no documentation

Why RE is Relevant and Required

Personal Education

- Observing a product allows anyone to learn from its characteristics.
  - Why it behaves that way
  - What it does
  - How it does something
  - Why something doesn't happen

- One can **complement engineering** education by observing code/products made by others
  - Open-source software plays an important role here
    - Because it the source is available, it doesn't mean that structure, components, etc… are readily available or understood
  - Actually… instead of learning from patterns, **why not learn from its application as implemented by other professionals**?
    - There are a lot of "hidden" subtleties due to the experience of their authors

Why RE is Relevant and Required

Work around limitations

- Products are engineered in order to **provide some value**, and **turn profit**
  - Some value = value perceived by the buyers, in relation to other products
  - Profit = max price for the minimal cost
- Products are frequently built to promote further revenue
  - Support contracts, build an ecosystem, help sell other products
  - Closed in their interfaces and limited in their feature set
- Reverse engineering can be used **to increase the feature set**
  - After the product is made, and without cooperation from manufacturer

Why RE is Relevant and Required

Work around limitations

Magic Lantern extends existing Cameras with a huge amount of extra features

https://magiclantern.fm/

3D scanning vehicles enables aftermarket variants to produce alternative parts

https://www.creaform3d.com/

Observing existing parts allows new parts to be designed to improve reliability, performance, design..

Why RE is Relevant and Required

Make a product compatible

- A product is developed for a set of scenarios. **What if we want it to operate on another, unexpected, environment?**

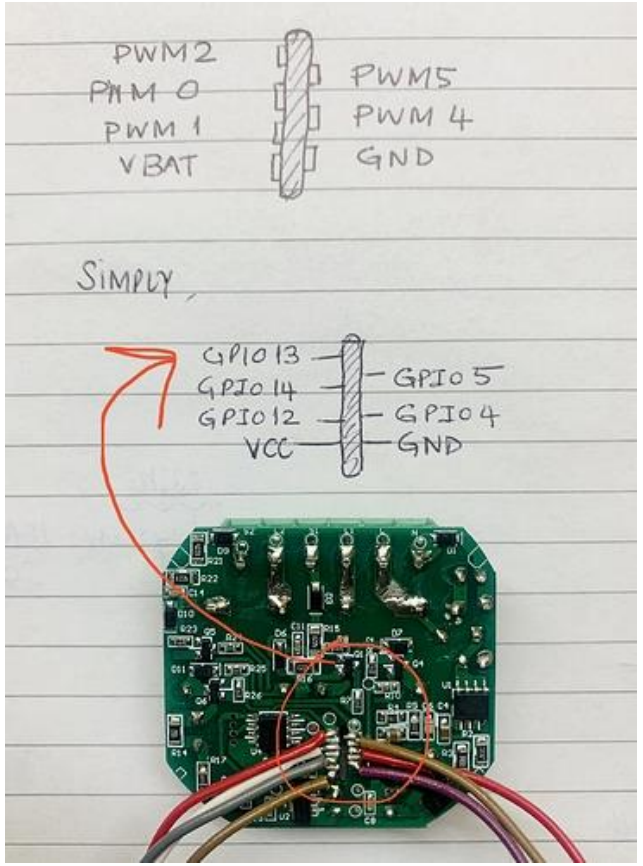
- RE allows obtaining **relevant design/operation information**
  - To modify the product to fit the new environment
    - Some components may be reconstructed
  - To build adapters integrating the product

- In corporate world it's standard to have products adapted to a specific use case
  - Process takes a long time, and is expensive
  - RE may provide a simpler route
  - Especially relevant if the manufacturer doesn't provide that service
    - Or simply doesn't exist

Why RE is Relevant and Required

Make a product compatible

- Make/DIY movements are keen on RE
- Driven by integrating and enhancement
  - Mostly for personal use
  - Community driven
- Sometimes with some collaboration
  - Magic Lantern



Unkown tuya chip - Hardware - Home Assistant Community (home-assistant.io)

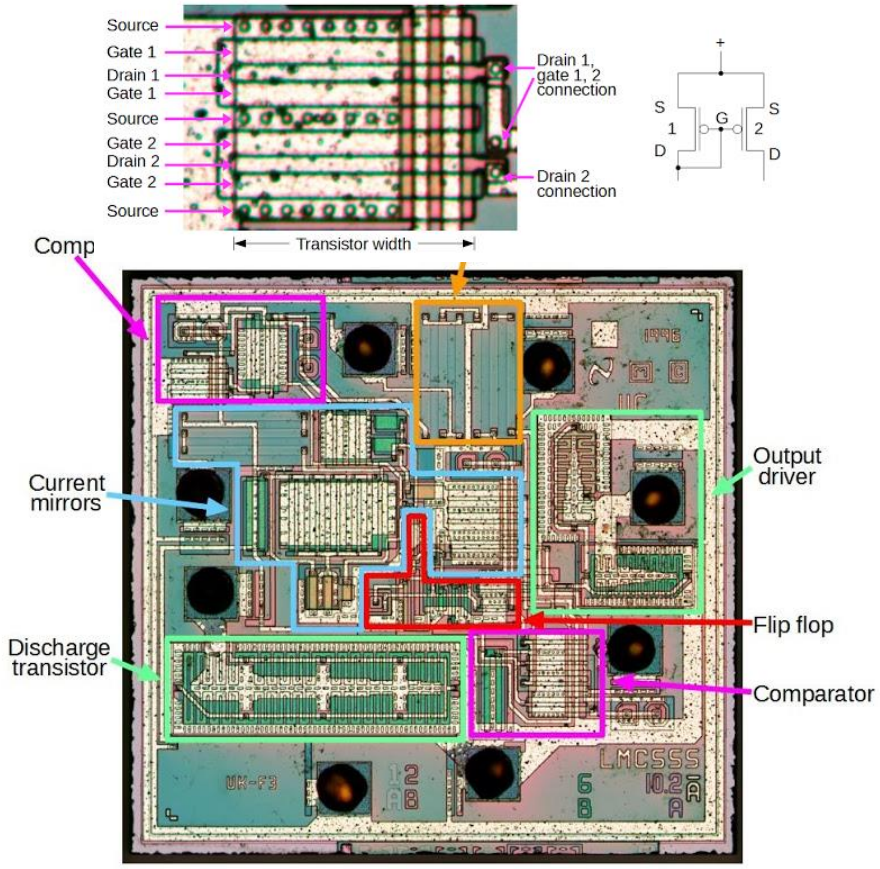
Why RE is Relevant and Required

Learn from other's products or from products of other domains

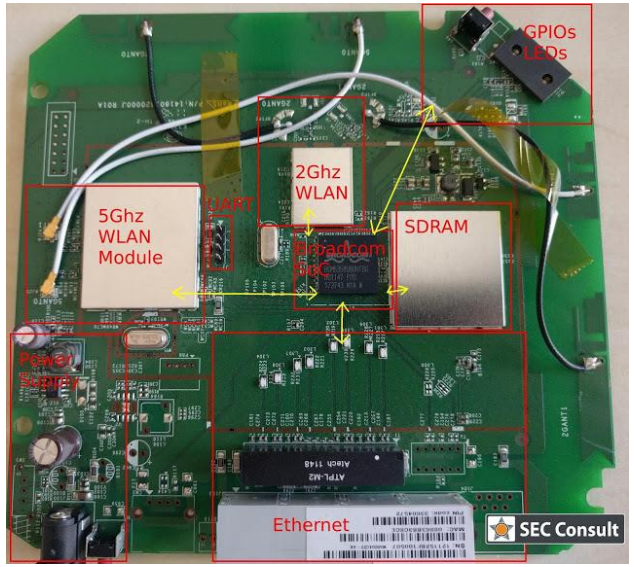
- Companies must determine the values/weaknesses of products in competing markets
  - What strategies/materials/methods/technology are used by competitors
  - Helps segmenting market and setting prices
  - Helps acquiring knowledge to develop new product
- Also: does a certain product violates a patent of ours?
  - Includes patented designs
- RE can be used for that purpose
  - and can feed information to engineering
  - determine the need for judicial actions protecting Intellectual Property

Why RE is Relevant and Required

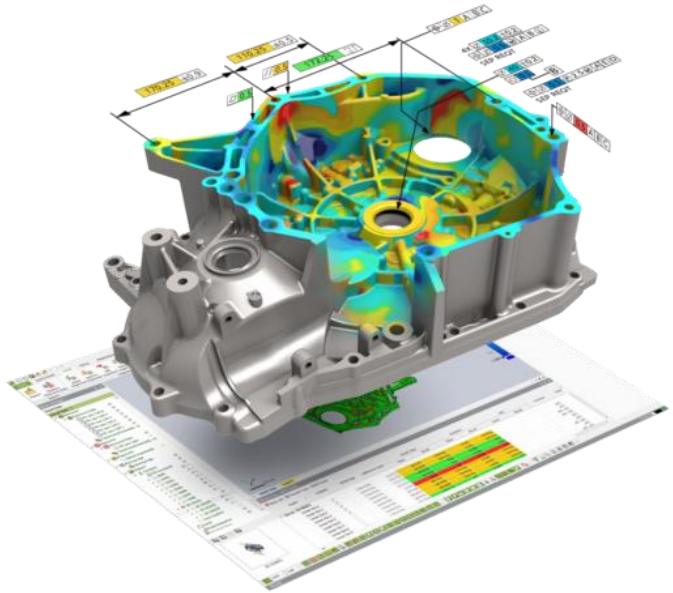
Learn from other's products or from products of other domains



http://www.righto.com/2016/04/teardown-of-cmos-555-timer-chip-how.html



https://sec-consult.com/



https://dewyseng.com/

Why RE is Relevant and Required

Finding the purpose of a certain code/binary blob or part

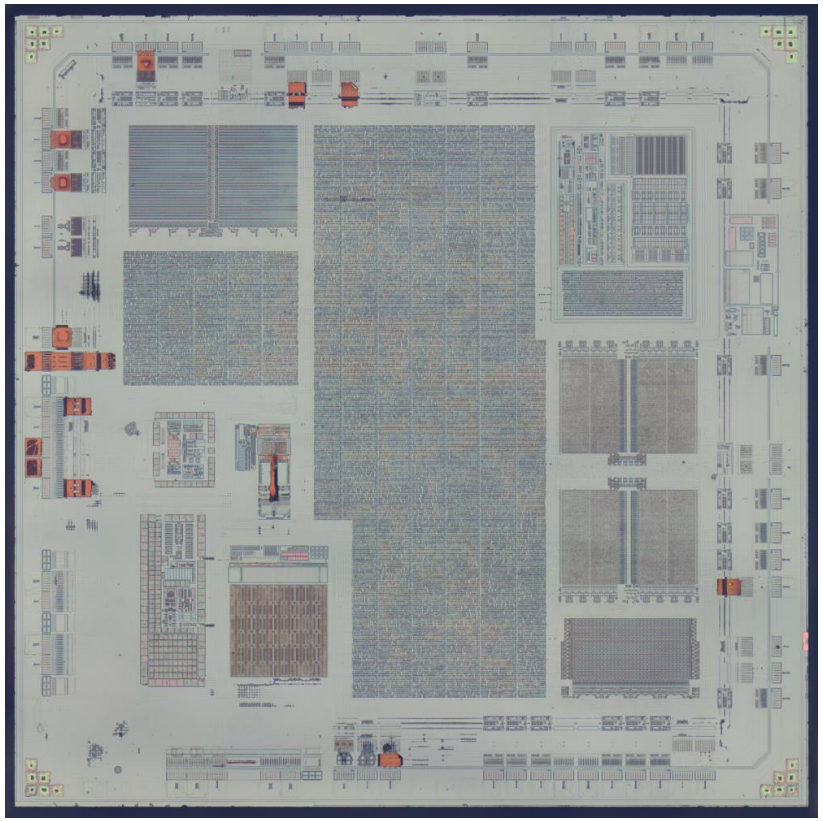
- Engineers frequently assume that an engineered entity is known (They trust dependencies)
  - That is… if you develop something, you know what it does
  - Also assume (or wish) that documentation exists

- What if:
  - documentation is lost?
  - the blob is external to the company?
  - the blob is misbehaving?
  - the blob was modified?
  - the engineer/supplier is not trusted?
  - the part is fake?
  - the company needs to validate the design process?

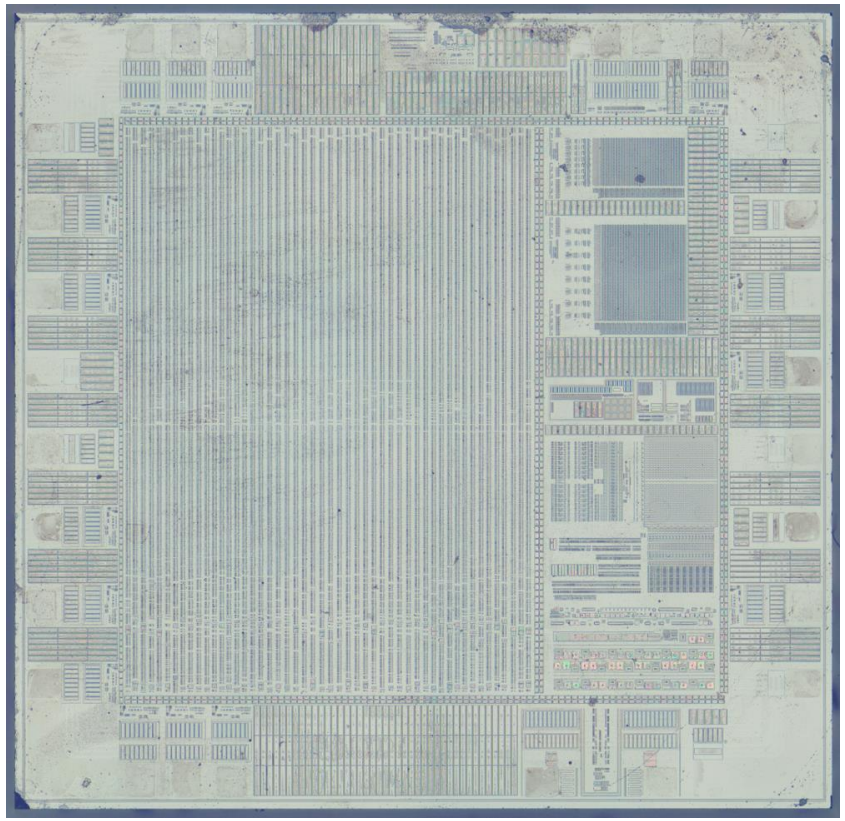
- RE can recover a similar design from the implementation, independently of the documentation, or the original design

Why RE is Relevant and Required

Finding the purpose of a certain code/binary blob or part



Fake FT232RL



Genuine FT232RL

https://zeptobars.com/en/read/FTDI-FT232RL-real-vs-fake-supereal

Why RE is Relevant and Required

Discovering flaws and faults

- Implementation may deviate from design
  - … it always deviates
- Implementation may present flaws due to unseen aspects
  - Processes used
  - Technology used
  - Interaction with additional components
  - Manufacturing flaws
  - Knowledge and experience
- RE is used in the scope of software testing to validate systems
  - Symbolic execution and Fuzzy testing are ways of helping the reverse engineering
  - Characterize if a given implementation reproduces the expected design
  - Identify additional modes

Why RE is Relevant and Required

Find and analyze malicious code

- For Anti-Virus, and Malware researchers, source code is not available
  - Or for offensive/red teams in black box scenarios

- **Malware detection relies on reverse engineering** to understand programs
  - RE allows the identification of patterns of malicious code
  - May rely on:
    - Interaction patterns
    - Bytecode structure
    - Communication with external hosts
    - Binary structure
    - Text contents
    - …

- Some RE is done in real time to find unknown malware
  - Or at least to identify suspect code, triggering further inspection

Limitations

- May be illegal in some cases, or lead to ambiguous situations
  - Higher risk of jeopardizing products developed
- Requires trained and experienced staff
  - Which is not abundant
- It’s costly in terms of time, resources and money
  - Expensive tools, scarce number of researchers, lengthy process
- May lead to incomplete or incorrect designs.
  - No guaranteed result!
  - An RE activity may be a complete waste of resources (time, staff, money)

Legal Framework

- The legality of RE is not assured a priori
  - varies with jurisdiction
  - varies with what is being reversed
  - varies with the purpose of the RE activity
  - varies with the impact to the product owner

WARNING

I AM NOT A LAWYER

- Applicable legislation:
  - EU: EU Directive 2009/24
  - USA: Digital Millennium Copyright Act

- This only applies to third parties
  - Product owners are free to use their own products as they seem fit
  - RE for the purpose of Software Quality Control

Legal Framework

Allowed situations (Europe, Directive 2009/24/EC)

The unauthorized reproduction, **translation**, **adaptation** or **transformation** of the form of the code in which a copy of a computer program has been made available **constitutes an infringement of the exclusive rights of the author**.

- .. circumstances may exist when such a reproduction of the code and translation of its form are indispensable to obtain the necessary information **to achieve the interoperability of an independently created program with other programs**.

- .. in these limited circumstances only, performance of the acts of reproduction and translation by or on behalf of **a person having a right to use a copy of the program** is legitimate and compatible with fair practice…

Legal Framework

Allowed situations (Europe, Directive 2009/24/EC)

- Article 5 b): To learn

The person **having a right to use a copy of a computer program** shall be entitled, without the authorisation of the rightholder, to **observe, study or test the functioning** of the program **in order to determine the ideas and principles** which underlie any element of the program **if he does so while performing any of the acts** of loading, displaying, running, transmitting or storing the program **which he is entitled to do**.

- **Broad Interpretation**: if you own a legitimate copy of the software, and are able to load it/run it/etc… you may analyze it for the purpose of learning

Legal Framework

Allowed situations (Europe, Directive 2009/24/EC)

- Article 5 b): To learn

- Caveats:
  - Replicating an algorithm may not be allowed, as a copy of the work infringes the copyright
  - Copy protection mechanism cannot be overcome
    - If there is a copy protection and you cannot freely execute the program, you do not have authorization to use it
  - Methods for bypassing protections are not legal
    - Crackers, keygens

- EULAs cannot restrict RE tasks

Legal Framework

Allowed situations (Europe, Directive 2009/24/EC)

- Article 6: Decompilation is generally allowed for the purposes listed in this directive, but mostly focusing on interoperability

- (allowed when) indispensable to obtain the information necessary to achieve the interoperability of an independently created computer program with other programs

- Provided that the following conditions are met:
  - those acts are performed by the licensee or by another **person having a right to use a copy of a program**, or on their behalf by a person authorized to do so
  - the information necessary to achieve interoperability **has not previously been readily available** to the persons referred to in point (a); and
  - those **acts are confined** to the parts of the original program which are necessary in order to achieve interoperability.

What RE Recovers?

- **System structure**: its components and their interrelationships, as expressed by their interfaces
- **Functionality**: what operations are performed on what components
- **Dynamic behavior**: system understanding about how input is transformed to output
- **Rationale**: design involves decision making between a number of alternatives at each design step
- **Construction**: modules, documentation, test suites, etc.

Chih-Wei Lu et al, “Reverse Engineering”, Handbook of Software Engineering and Knowledge Engineering, Vol.2

Software Reversing Levels

System Level Reversing

- Observe how the software is provided and how it operates
  - Involves analyzing the environment, packaging, dependencies, and then observed behavior
  - May require tools to intercept traffic, system calls, input/output

- End goal: collect information to direct further analysis
  - Important in order to select tools, processes, and overall strategy
    - Language use, packaging algorithms, encryption
  - Important to characterize behavior and identify external dependencies
    - Remote servers involved, files accessed, communication channels used

Software Reversing Process

Code Level Reversing

- Extract design concepts and algorithms from binaries
  - Compiled to binary code or bytecode.

- It’s a complex, architecture dependent process
  - Some say “an art form”
  - Expensive enough that competitive RE is not usually pursued
    - To fully reverse and reassemble a given competing software (except in some cases)

- Makes use of tools capable of representing the low-level language in something “human compatible”
  - Compiler optimization and obfuscation make this process uncertain
  - Perfect reconstruction is frequently impossible as low-level languages do not use the same constructs as higher-level ones

Software Reversing Activities

- Understanding the processes
  - Large scale observation of the program at a process level
  - Identification of major components and their functionality
- Understanding the Data
  - Understand data structures used
- Understanding Interfaces
  - Which interfaces exist and how the process reacts to them

Software Reversing

- Programs are developed in a high-level programming languages
  - C, C++, C#, Java, Python, Go…
- A compiler converts the high-level instructions to low level instructions
  - Machine Code: instructions that are executed directly by the CPU
  - Bytecode: instructions that are executed by a middleware, VM or Interpreter
- Reverse Engineering involves understanding low level instructions
  - Which is not easy and is costly
  - Requires knowledge of the specific target being analyzed (the VM, the CPU)
    - Different CPUs have different opcodes and execution behavior

Low level languages

Machine Code

- Each CPU has a specific instruction set
  - Associated to rules regarding structure, execution flow,

- When a program is compiled to “binary”, the high-level logic is converted to a sequence of instructions
  - This sequence may be executed by a family of CPUs or a single model
  - Running this sequence on another CPU may involve binary translation (conversion)

- Humans are typically not capable of reading binary instructions, but instructions are always able to be translated to Assembly
  - Good: We can read binary code
  - Bad: each CPU has a specific variant of Assembly. Also, assembly is not simple.

Low level language

Machine Code

```
// Original C
int square(int num) {
    return num * num;
}
```

```
//ARM64 GCC 5.4
square(int):
        sub     sp, sp, #16
        str     w0, [sp, 12]
        ldr     w1, [sp, 12]
        ldr     w0, [sp, 12]
        mul     w0, w1, w0
        add     sp, sp, 16
        ret
```

```
//MIPS64 GCC 5.4
square(int):
        daddiu  $sp,$sp,-32
        sd      $fp,24($sp)
        move    $fp,$sp
        move    $2,$4
        sll     $2,$2,0
        sw      $2,0($fp)
        lw      $3,0($fp)
        lw      $2,0($fp)
        mult    $3,$2
        mflo    $2
        move    $sp,$fp
        ld      $fp,24($sp)
        daddiu  $sp,$sp,32
        j       $31
        nop
```

Compiler Explorer (godbolt.org)

```
//PowerPC GCC 4.8.5
square(int):
        stwu     1,-32(1)
        stw     31,28(1)
        mr      31,1
        stw     3,8(31)
        lwz     10,8(31)
        lwz     9,8(31)
        mullw   9,10,9
        mr      3,9
        addi    11,31,32
        lwz     31,-4(11)
        mr      1,11
        blr
```

```
//x86_64 gcc 5.4
square(int):
        push    rbp
        mov     rbp, rsp
        mov     DWORD PTR [rbp-4], edi
        mov     eax, DWORD PTR [rbp-4]
        imul    eax, DWORD PTR [rbp-4]
        pop     rbp
        ret
```

Low level languages

Machine Code

- For compiled programs, the RE tasks involves extracting information from the sequence of Assembly instructions
  - Disassembly is automatic, the rest frequently it isn’t

- Reconstruction is never perfect!
  - Different level of abstraction: e.g., it is not trivial to recover C++ class structure and OOP relations from Assembly code
  - **Different compilers generate different assembly** for the same source code
  - **Same compiler may generate different assembly** for the same source code
    - Optimization flags, CPU matching, protection mechanisms, target object type…

Low level languages

Bytecode

- Some languages are compiled to a bytecode (!= machine code)
  - Intermediate language that is processed by a VM or framework
  - .NET, Java, Python, JS, LISP, LUA, Ocaml, Tcl, FoxPro, WebAssembly

- Bytecode contains a compact (optimized) representation of the higher layer structures
  - Framework/VM will execute bytecode in the target CPU
  - Same bytecode usually can be executed in multiple CPUs, provided there is a native VM implementation
    - The Java moto: Write Once, Run Anywhere

- Bytecode allows easier extraction of information, provided there is such route
  - May recover classes, function names, and even comments (but not always)
  - Traditional decompiling tools will not process bytecode (that easily)