Course Guidelines

REVERSE ENGINEERING

deti universidade de aveiro
departamento de eletrónica, telecomunicações e informática

João Paulo Barraca, Bernardo Cunha, José Luis Azevedo

Faculty

- João Paulo Barraca – jpbarraca@ua.pt
  - IT – Telecommunications and Networks - Aveiro
- Bernardo Cunha – mbc@det.ua.pt
  - IEETA - Intelligent Robotics and Systems
- José Luis Azevedo - jla@ua.pt
  - IEETA - Intelligent Robotics and Systems

Operational aspects

- Lectures in a mixed format: remote + in place (if possible)
  - According to the pandemic situation and actual lecture contents
- Contents: everything available in the Teams Channel
- Languages:
  - Classes may be lectured in English, but will default to Portuguese.
  - Contents will be available in English
- Communication:
  - Announcements will be made through Teams (and or elearning)
  - Direct communication through the Teams Channels. Participation is mandatory!
  - Email if required: jpbarraca@ua.pt, mbc@det.ua.pt, jla@ua.pt

Objectives

- Know the techniques to **identify the components of a system**
- Know the techniques to **observe the behavior** of systems and components
- Know the **methodologies for reverse engineering**
- Know the relevant protocols and technologies **to build systems, applications and devices**
- Understand the techniques, processes and tools for **decomposition of applications**

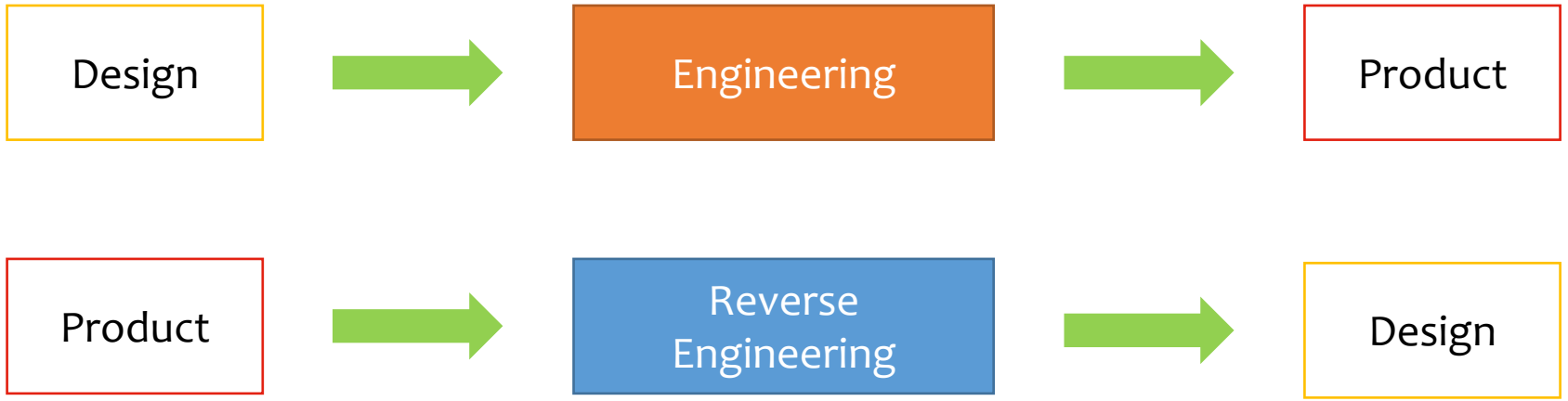
Objectives

- Understand the techniques, processes and tools for **decomposition of devices and systems**
- Understand the techniques, processes and tools for **decomposition of mobile applications**
- Capability to **perform tasks of reverse engineering**
- Capability of **documenting the process of reversing engineering**
- Capability to **replicate components** analyzed through reverse engineering

Objectives

- This will not be a course about hacking, malware analysis, or cracking

- This will be a course about reconstructing software/systems from products



Syllabus

Intro, plus 3 main modules

0. Introduction ~1 week

1. Mobile Applications ~3-4 weeks

2. Binary Applications ~5-6 weeks

3. Devices ~5-6 weeks

Evaluation

- 3 assignments, to be implemented by groups of 2 students:
  - Android – 20%
  - Applications – 25%
  - Devices – 25%
  - Assignments should be returned ~2 weeks after the last lecture on the topic

- 1 final exam – 30%
  - In June/July

- Some variations may be required

Bibliography

- Will be provided in every lecture:
  - Books, papers, reports, videos

- Available on the O'Reily library:
  - A. P. David, Ghidra Software Reverse Engineering for Beginners, Packt Publishing, 2021, ISBN: 9781800207974
  - Bruce Dang, Alexandre Gazet, Elias Bachaalany, Practical Reverse Engineering: x86, x64, ARM, Windows Kernel, Reversing Tools, and Obfuscation, 2014, ISBN: 9781118787311
  - Philip Polstra, Reverse Engineering and Exploit Development, Infinite Skills 2015 (Video)
  - Eldad Eilam, Reversing: Secrets of Reverse Engineering, Willey, 2005, 9780764574818
  - Dennis Andriesse, Practical Binary Analysis, ISBN-13: 9781593279127, 2018

- Relevant website (links): https://beginners.re/main.html

Introduction to Reverse Engineering

REVERSE ENGINEERING

João Paulo Barraca

deti universidade de aveiro
departamento de eletrónica, telecomunicações e informática

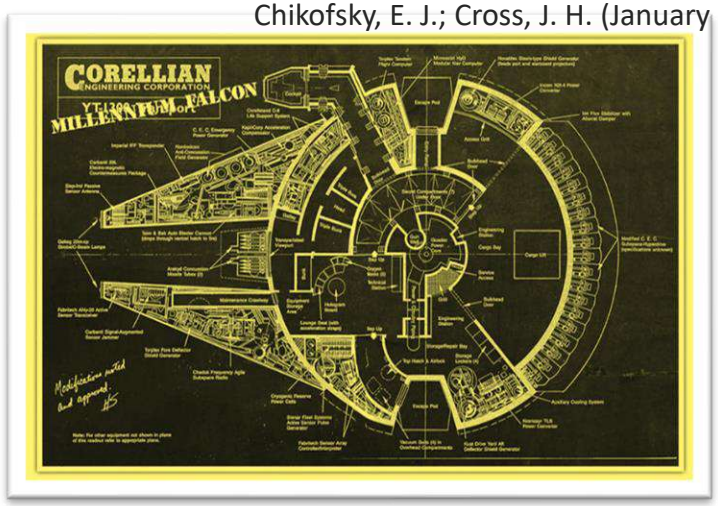
What is Reverse Engineering (RE)

- 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



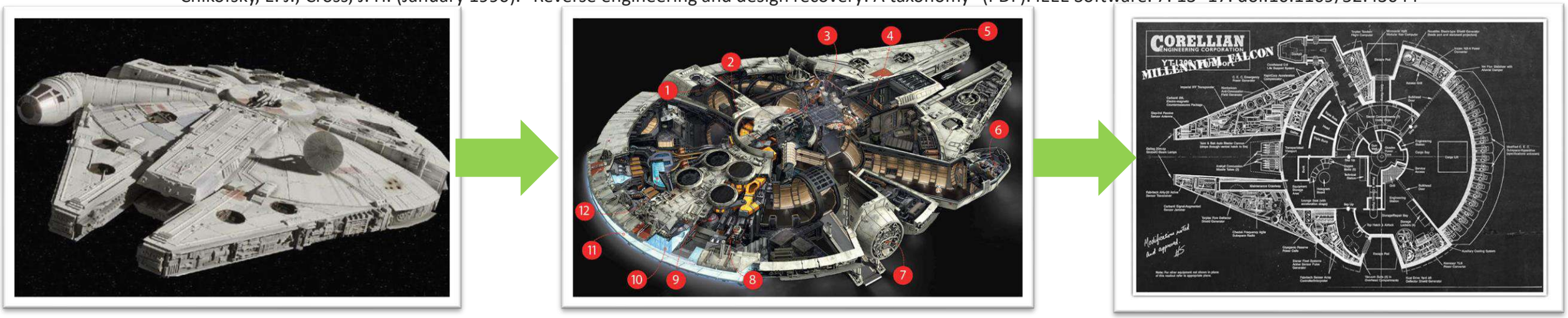
Forward Engineering

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



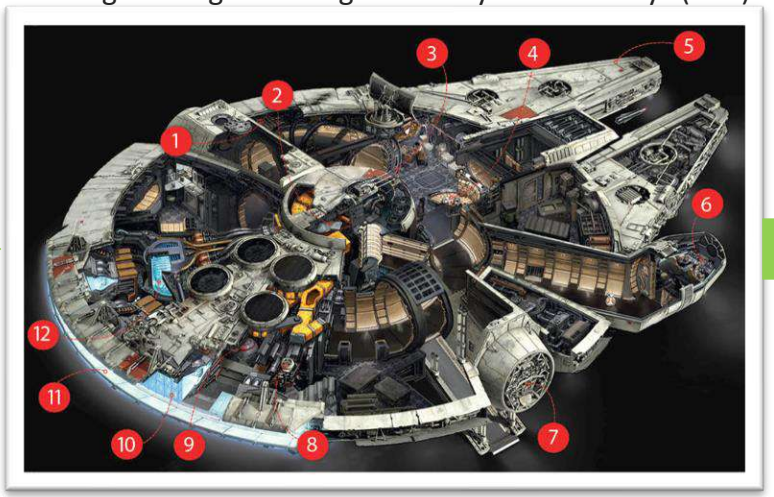
Reverse Engineering

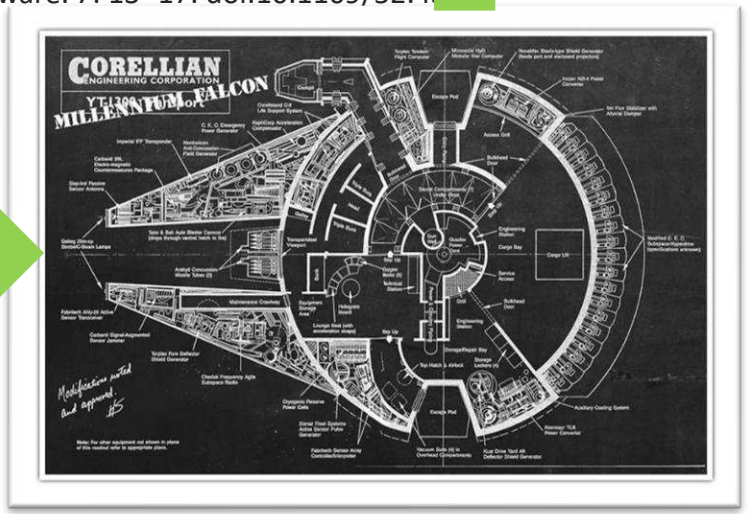
Images belong to their respective owners

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

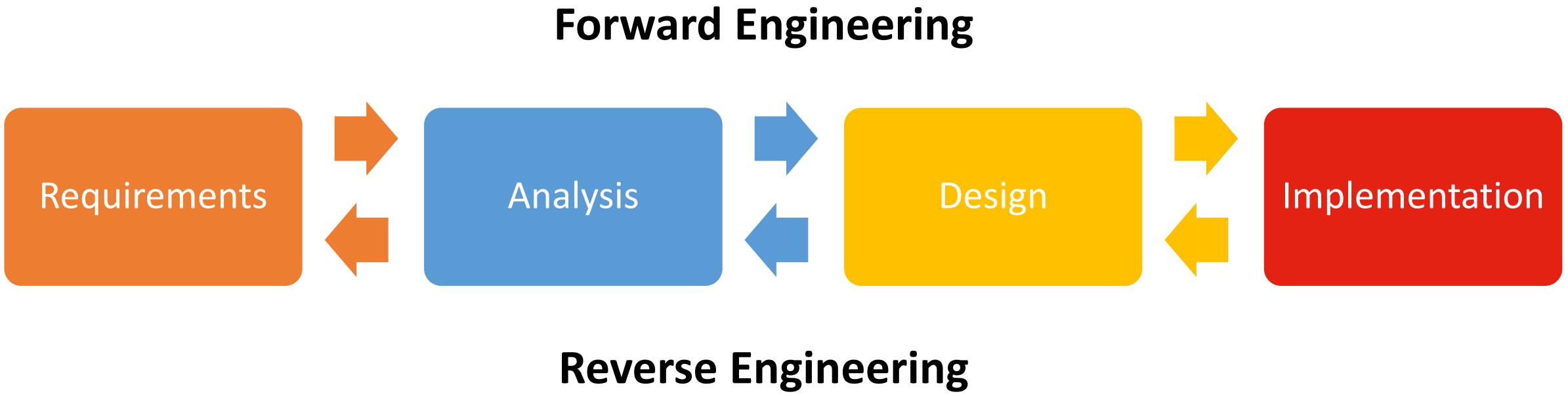




Reverse Engineering

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?**

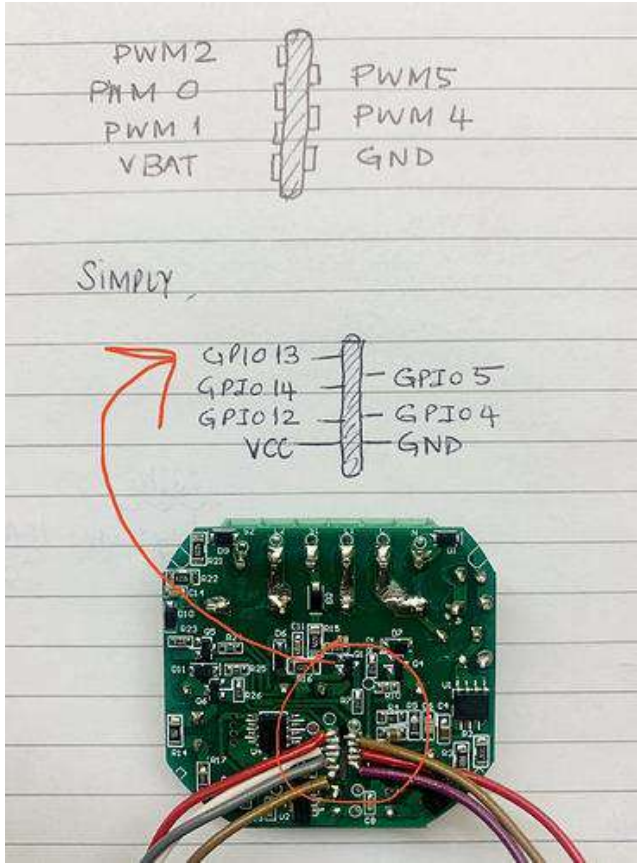
- RE allows obtaining **relevant design/operation information**
  - To modify the product to fit the new environment
    - Some components may be reconstructed
  - To build adapters integrating the product

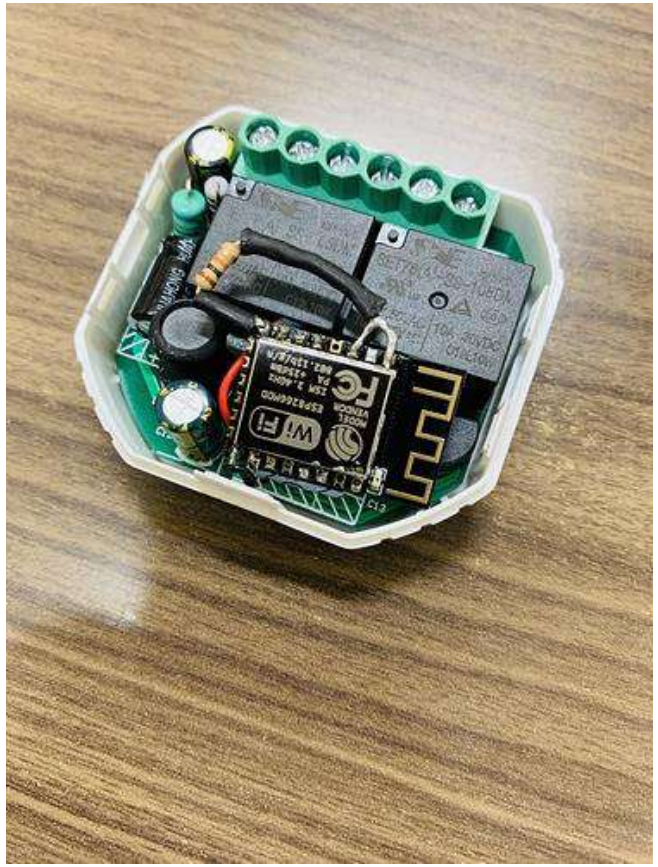
- In corporate world it's standard to have products adapted to a specific use case
  - Process takes a long time, and is expensive
  - RE may provide a simpler route
  - Especially relevant if the manufacturer doesn't provide that service
    - Or simply doesn't exist

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
- Frequently without cooperation from manufacturers
  - Alarms: ParadoxAlarmInterface/pai
  - Sports bracelets: Gadgetbridge
- Sometimes with some collaboration
  - Magic Lantern





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

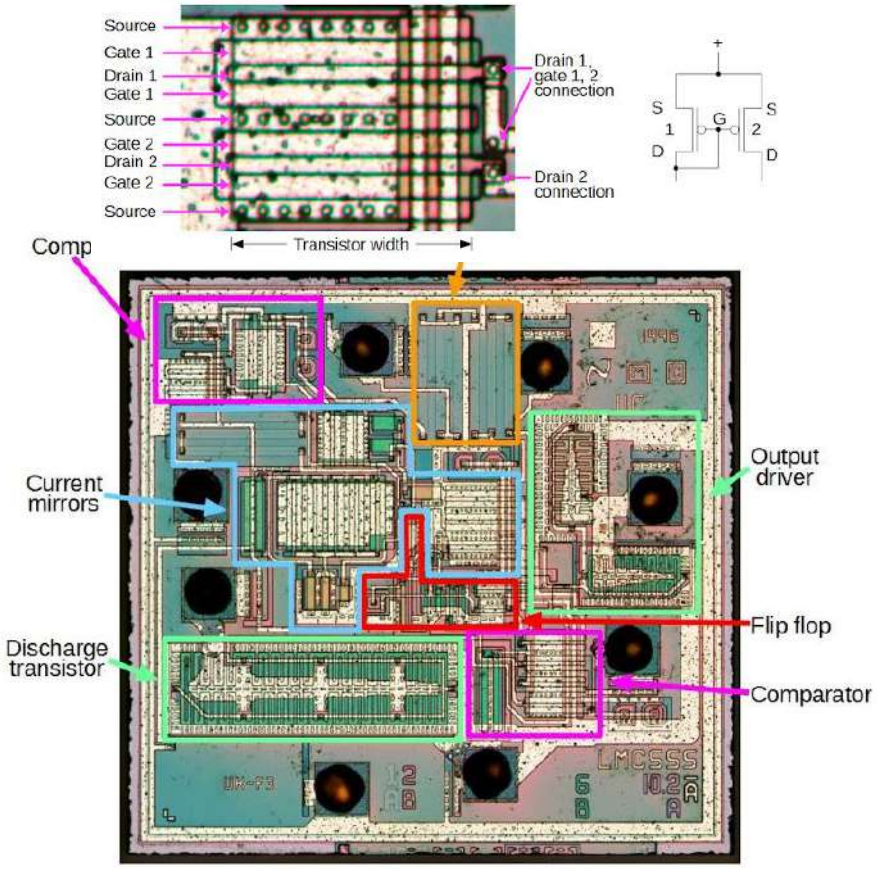
Why RE is Relevant and Required

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

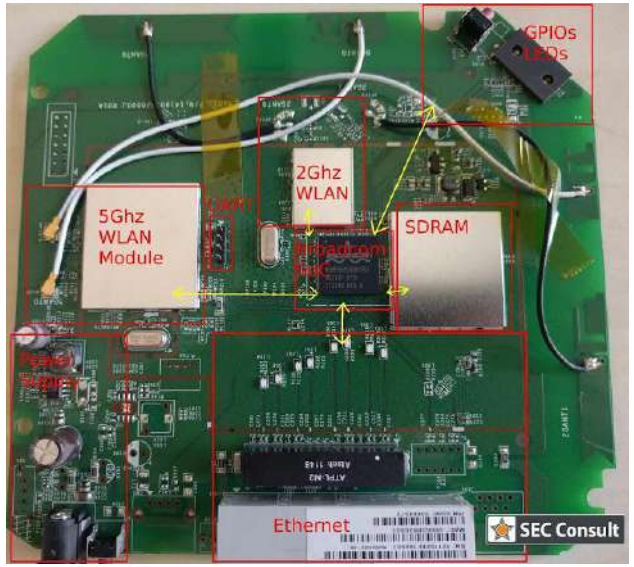
- Companies must determine the values/weaknesses of products in competing markets
  - What strategies/materials/methods/technology are used by competitors
  - Helps segmenting market and setting prices
  - Helps acquiring knowledge to develop new product
- Also: does a certain product violates a patent 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

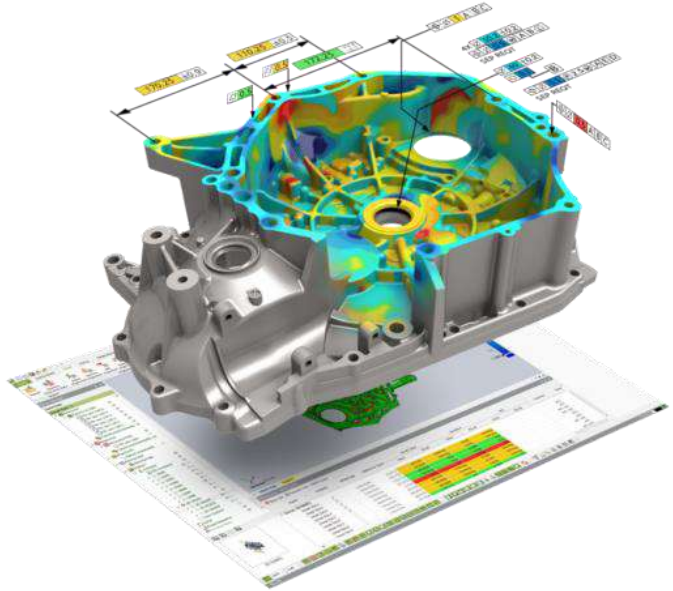
Learn from other's products or from products of other domains



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

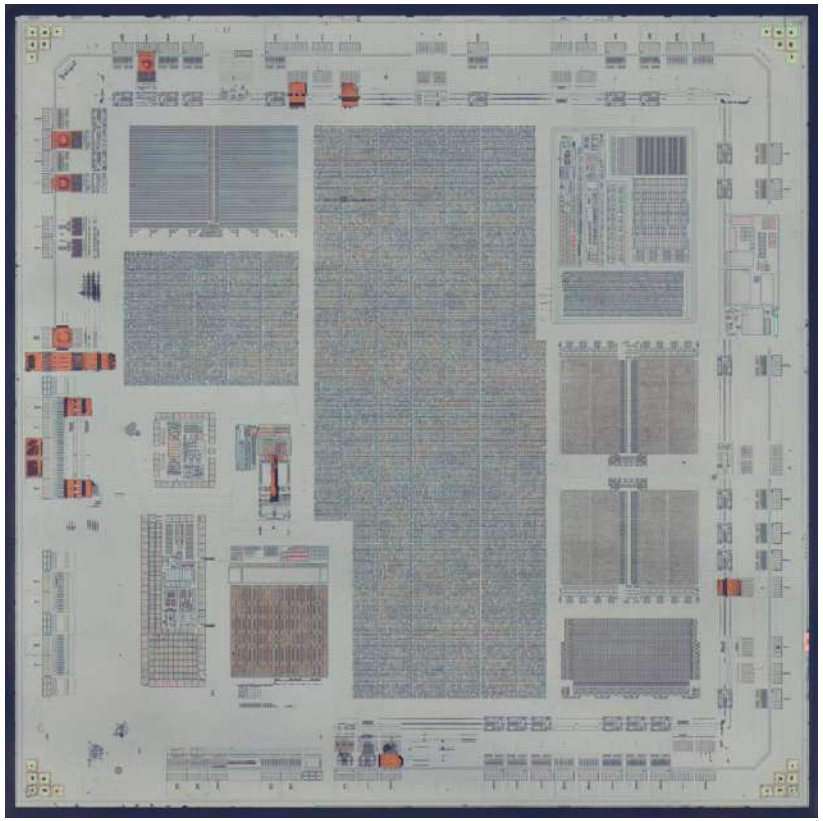
- Engineers frequently assume that an engineered entity is known (They trust dependencies)
  - That is… if you develop something, you know what it does
  - Also assume (or wish) that documentation exists

- 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?

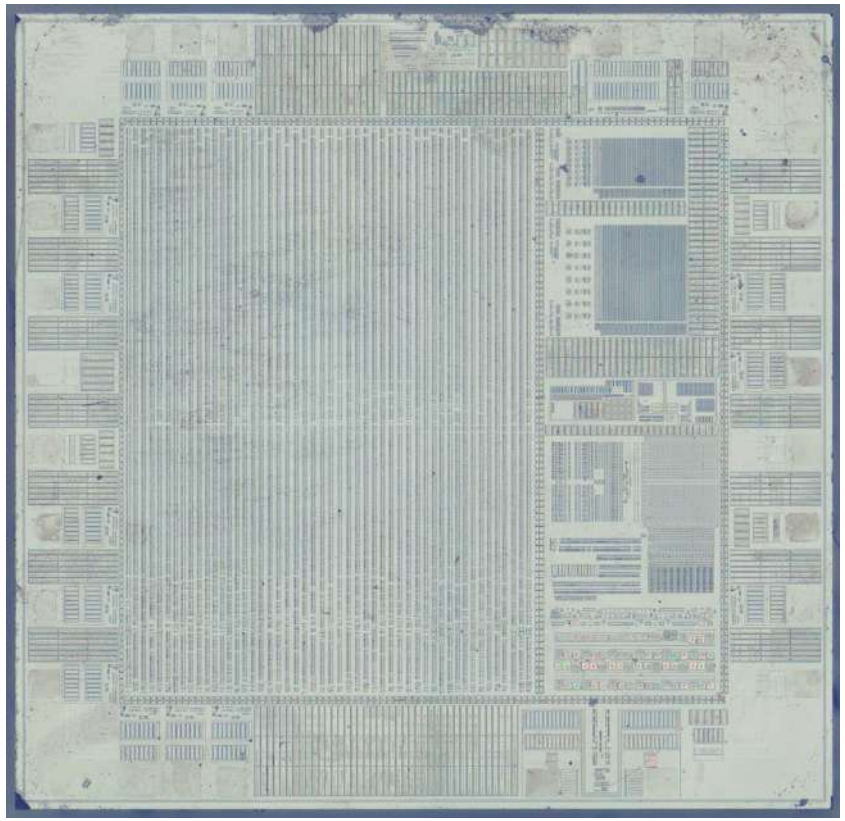
- RE can recover a similar design from the implementation, independently of the documentation, or the original design

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

Legal Framework

Allowed situations (USA, DMCA)

- **Interoperability**: even circumventing DRM
- **Encryption research**: if the protection prevents the evaluation of the technology
- **Security testing**: determine if a software is secure and to improve it
- **Regulation**: to limit what information is presented to minors
- **Government Investigation**: government agencies are not affected
- **Privacy protection**: users may reverse and circumvent data gathering technologies
- EULAs may restrict RE actions, although this is not guaranteed by law

Eldad Eilam, 2005

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)

Files and Filetypes

REVERSE ENGINEERING

João Paulo Barraca

deti universidade de aveiro
departamento de eletrónica, telecomunicações e informática

Files

- Files are containers that are parsed according to a schema
  - Parsing implies knowing the file content
- How to select the adequate parser?
  - Using the file extension
  - Using magic headers
  - Using rules provided by configuration
  - Previous knowledge
- What if the parser is wrong?

File extensions

- File extensions are words appended to the filename, after a dot

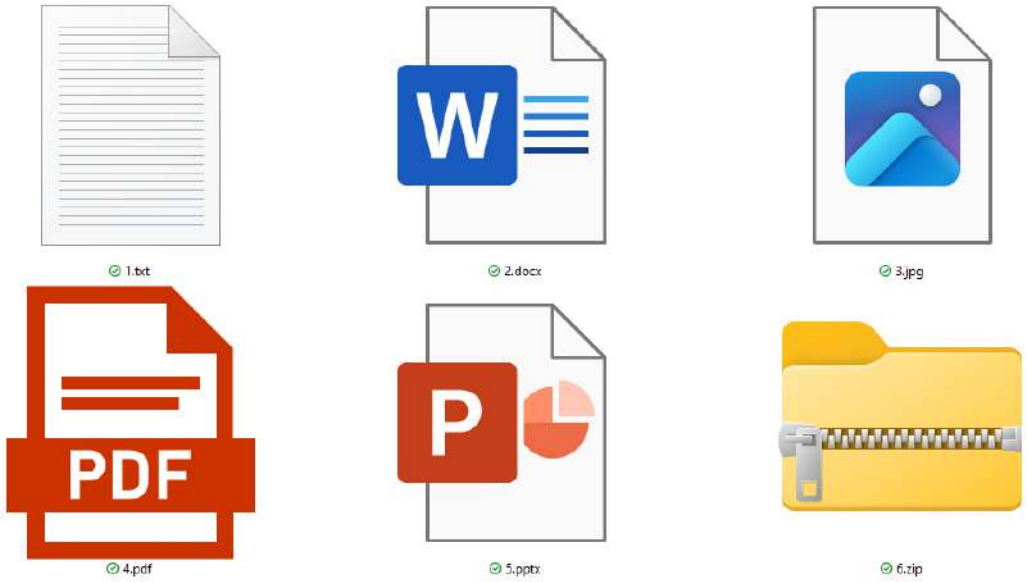
`lecture.pptx`

- File extensions are a basic mechanism to know how to handle a file
  - Operating systems uses extensions to select the correct process
  - Applications use it to filter which files are adequate (.e.g images). Mostly an usability aspect
  - Humans use extensions to differentiate files

- Popular file extensions:
  - zip, rar, bz2, gz, 7z: compressed files
  - exe, dll, so, com: executable files
  - jpg, tiff, bmp, fits, png: images

File extensions

- Knowing the file extension is important to apply the correct analysis process
  - Analyzing a JPG is different from analyzing an EXE, or even a PNG



File extensions

Extensions are misleading!

- Windows hides extension of known file types
  - **`Sample.pptx`** becomes only **`Sample`**

- Executable files may have an embedded icon
  - Freely defined by the developer
  - Explorer will show that icon

- A file named **`Sample.pptx.exe`** will be shown as **`Sample.pptx`**
  - Users recognize the extension and may think the file is safe

- In a RE task, consider that a file may have bogus extensions

File Signature

Also known as Magic Bytes/Header

- Most files can also be recognized by a magic value in the file start/end
  - Manipulating headers can lead to incorrect detection and maybe processing
  - Some OS use the magic headers instead of the file extension
  - Also known as File Signatures

- Some magic values:
  - Office Documents: `D0 CF 11 E0`
  - ELF: `7F E L F`
  - JPG: `FF D8`
  - PNG: `89 P N G 0D 0A 1A 0A`
  - Java class: `CA FE BA BE`

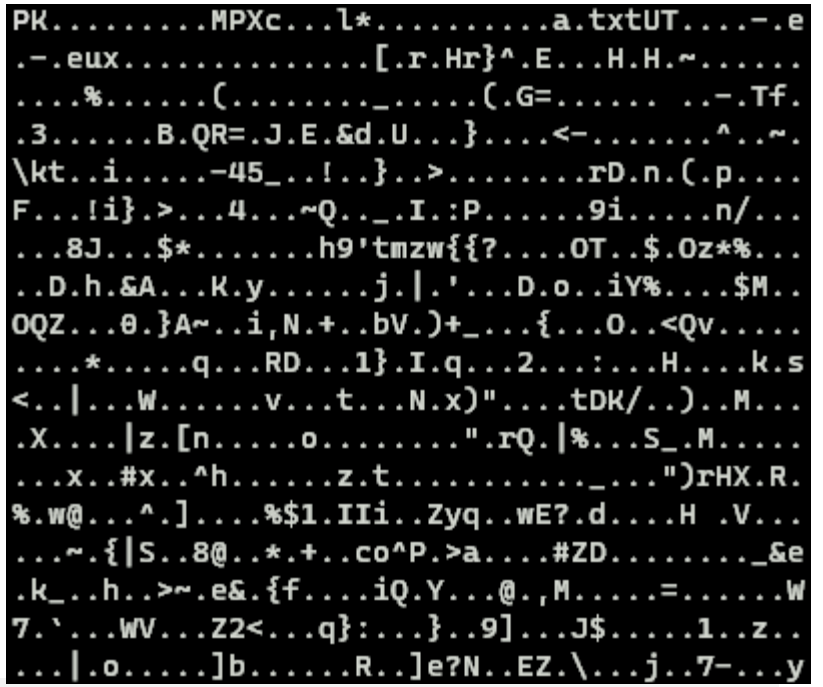
File Signature

Sometimes, magic headers are reused

- PK.. (`50 4B 03 04`) is the magic for ZIP files

```
└─$ file 8\ -\ Obfuscation.pptx
8 - Obfuscation.pptx: Microsoft PowerPoint 2007+
```


```
└─$ file sample.zip
sample.zip: Zip archive data, at least v2.0 to extract
```



File Signature

Sometimes, magic headers are reused

- Actually, pptx are zip files

```
 $ unzip -l 8\ -\ Obfuscation.pptx
Archive:  8 - Obfuscation.pptx
  Length      Date    Time    Name
---------  ---------- -----   ----
     5179  1980-01-01 00:00   ppt/presentation.xml
    12041  1980-01-01 00:00   customXml/item1.xml
     1203  1980-01-01 00:00   customXml/itemProps1.xml
      219  1980-01-01 00:00   customXml/item2.xml
      335  1980-01-01 00:00   customXml/itemProps2.xml
      394  1980-01-01 00:00   customXml/item3.xml
      606  1980-01-01 00:00   customXml/itemProps3.xml
    33895  1980-01-01 00:00   ppt/slideMasters/slideMaster1.xml
     2477  1980-01-01 00:00   ppt/slides/slide1.xml
     4665  1980-01-01 00:00   ppt/slides/slide2.xml
     4384  1980-01-01 00:00   ppt/slides/slide3.xml
     4003  1980-01-01 00:00   ppt/slides/slide4.xml
     4719  1980-01-01 00:00   ppt/slides/slide5.xml
```

```
PK.........!.x.......;.......ppt/presentati
on.xml...n.8.....;..-............@P...Jt"T'.
t.t...$-.CS(z.w..........x.....W>..k.>..|..E
WV.......2....%.../.w..O......~....I5.SaZ.^K
.E.~...x...7].[....aR....r^?T...q.%a.......3
......w..Q.....6>.K..)Z.;.^..%.^...Mp..Z)...
...u.7......B^...r.cDS...*v.B.Q....m87..$..z
w...1..........[.kr4?O........)..l...\..! ..
....#'pv..).Q....r..pu..=..n...C{...u....R.u
..N0.]z....>k..~..x....]~i.u._..a._..a._....
......,........ ?...a~.....G\~..~d..5.f...Kf
.Y../....R....r..../....?....r.8u......?.A..
.G"k}..AV|... ...~.....G"..:.H...u...:..~$..
+........^.:....o..b..R....K?..L.1.Mi..M...#
JO.J.g.Z>.7...5_.2...q...<.^.t........C....j
```

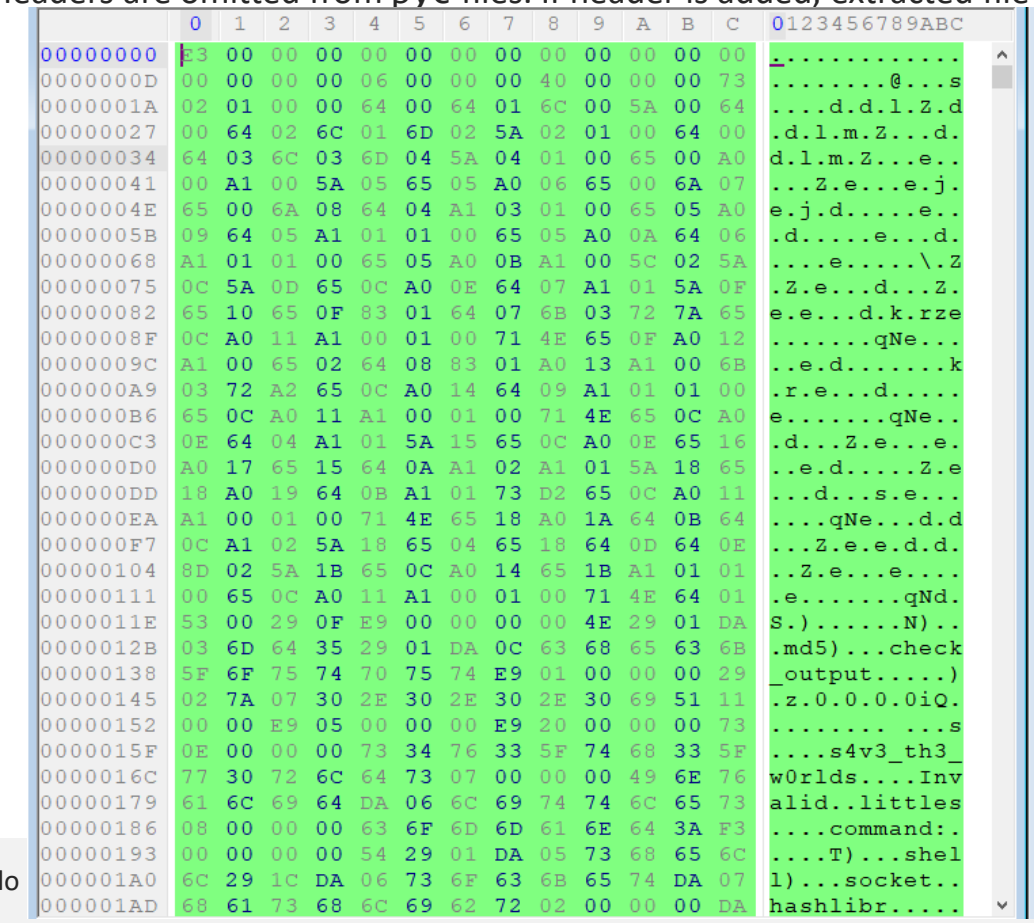
File Signature

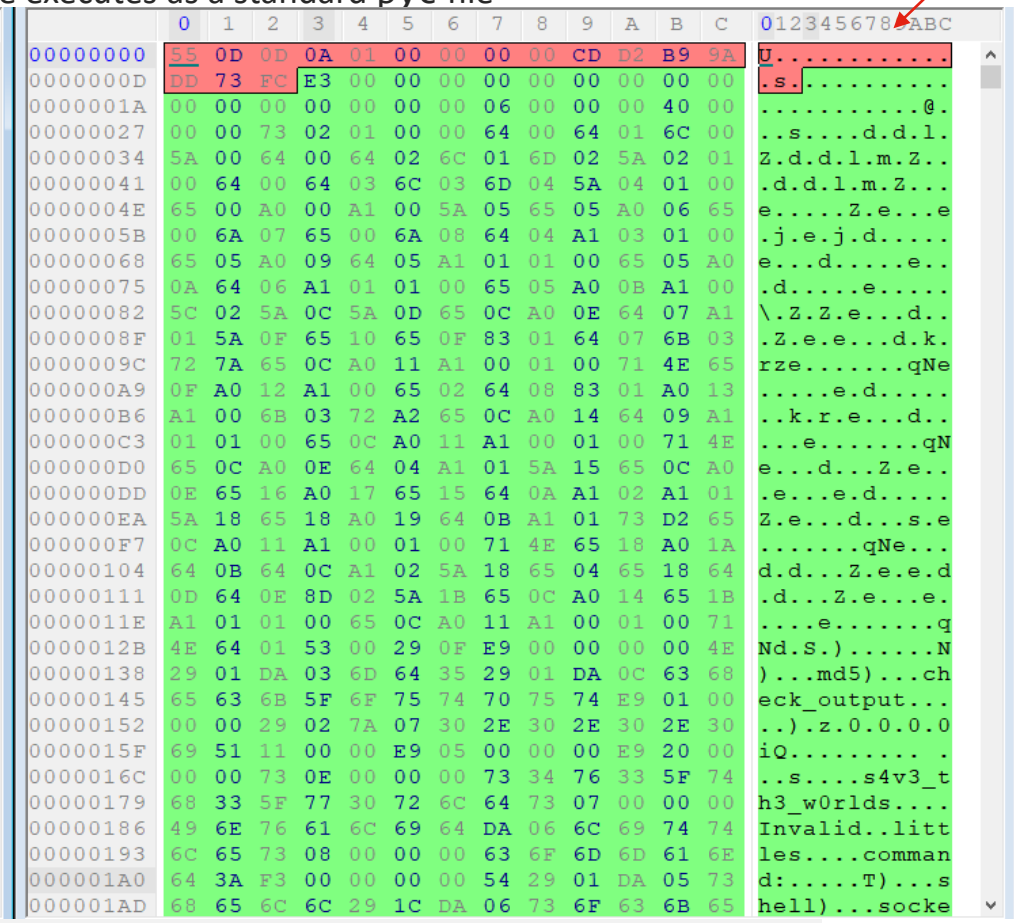
Magic Headers can be manipulated if the content is known

- `PyInstaller` allows conversion of a Python application to an executable application
  - It packs the `pyc` files into a container. Container is extracted on runtime and compiled python code is executed
  - Headers are omitted from pyc files. If header is added, extracted file executes as a standard pyc file

Added header

Extracted



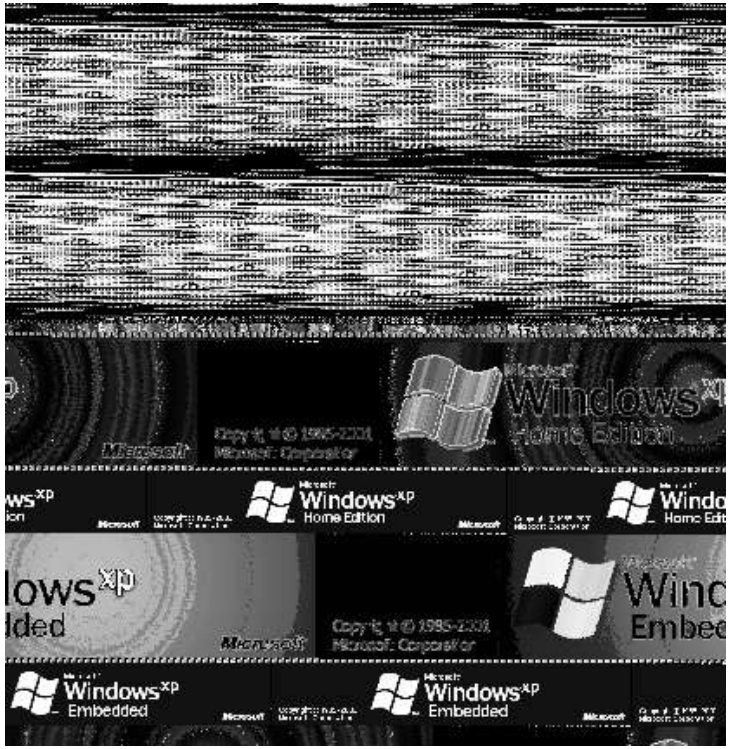


Reconstructed

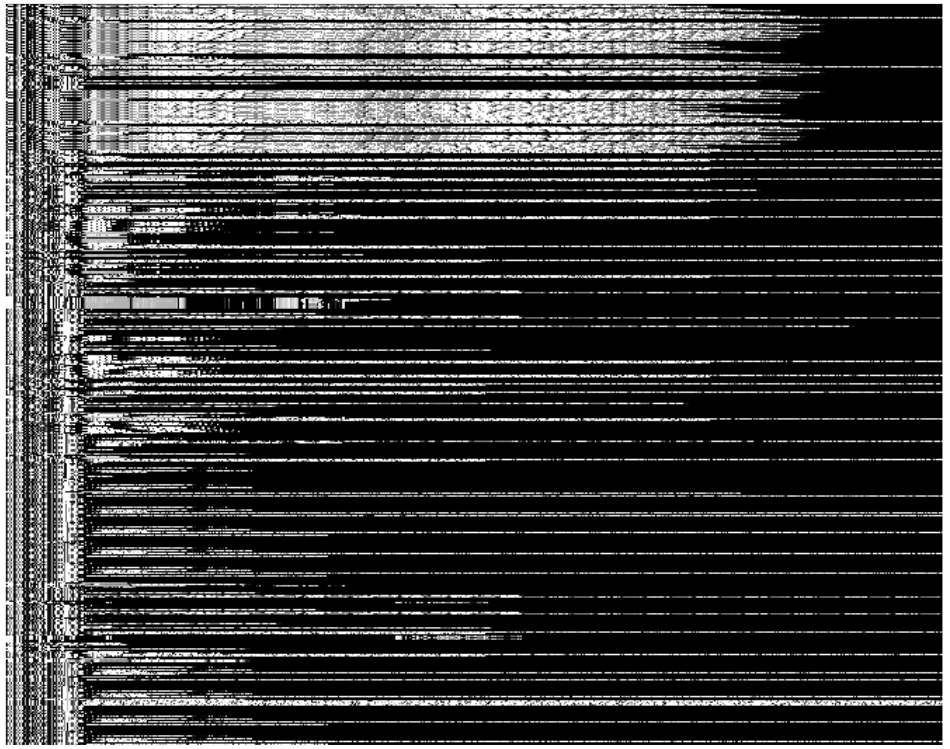
File Signature

Magic Headers can be manipulated if the content is known

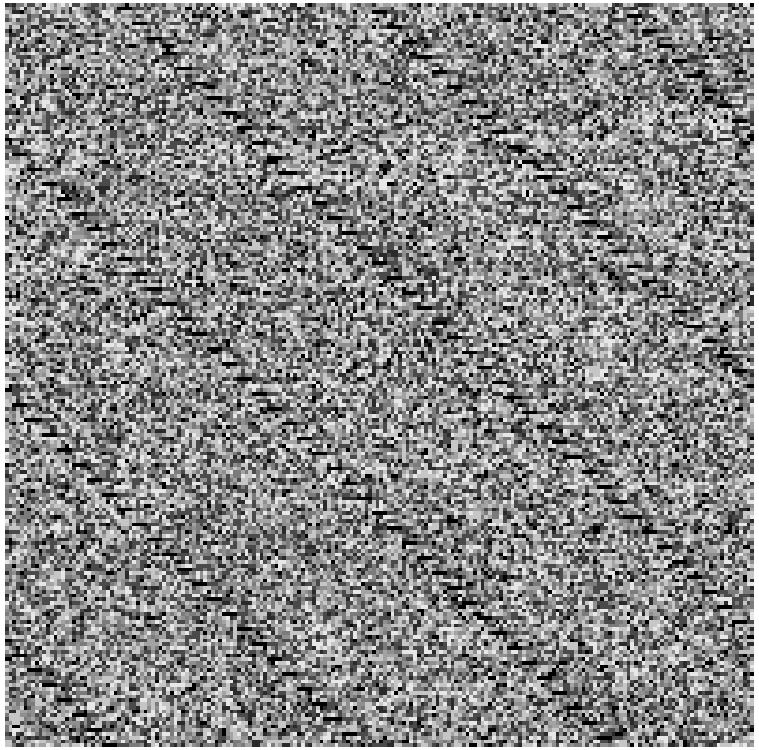
- Direct Visualization may help
  - Direct byte visualization, Mapping to an image, Entropy Analysis, Tuples



shell32.dll



Network traffic



Compressed data

Greg Conti, Sergei Bratus, “Voyage of the Reverser A Visual Study of Binary Species”

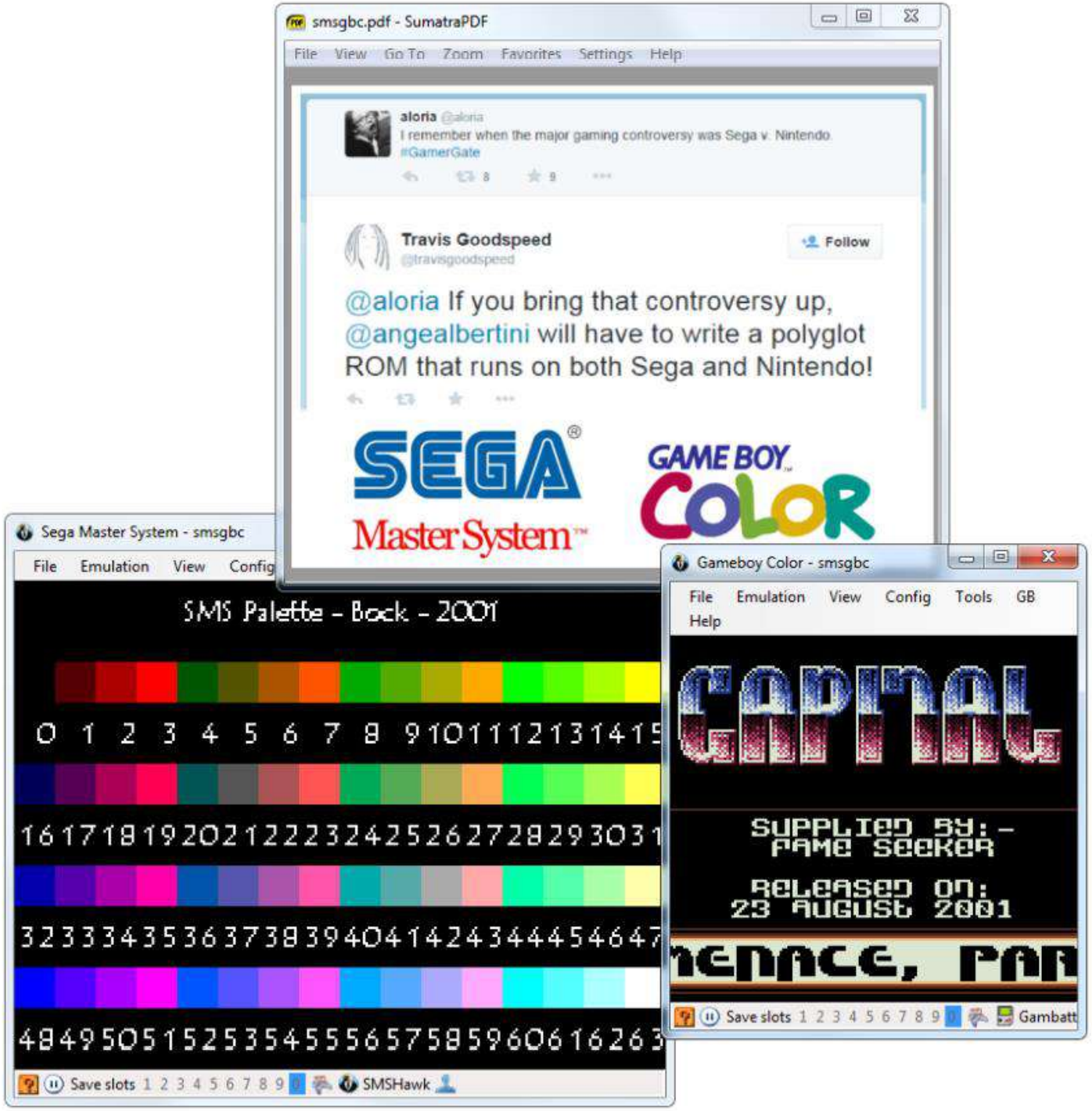
Content Type Obfuscation

Polyglots

A file that has different types simultaneously, which may bypass filters and avoid security counter-measures.

pocorgtfo19.pdf (alchemistowl.org)

Technical Note: This file, pocorgtfo19.pdf, ***is valid as a PDF*** *document,* ***a ZIP archive****, and* ***a HTML page****. It is also available as a* ***Windows PE executable****, a* ***PNG image*** *and an* ***MP4 video****, all of which have the same MD5 as this PDF*


smsgbc.pdf - SumatraPDF
File View Go To Zoom Favorites Settings Help
aloria @aloria
I remember when the major gaming controversy was Sega v. Nintendo.
#GamerGate
Travis Goodspeed
@travisgoodspeed
Follow
@aloria If you bring that controversy up, @angealbertini will have to write a polyglot ROM that runs on both Sega and Nintendo!
SEGA
Master System
GAME BOY COLOR
Sega Master System - smsgbc
File Emulation View Config
SMS Palette - Back - 2001
Save slots 1 2 3 4 5 6 7 8 9
SMSHawk
Gameboy Color - smsgbc
File Emulation View Config Tools GB Help
CAPTAIN
SUPPLIED BY:-
PAME SEEKER
RELEASED ON:
23 AUGUST 2001
Save slots 1 2 3 4 5 6 7 8 9
Gambatt

Content Type Obfuscation - Polyglots

Types

- **Simple Polyglot file**: file **has** different types, accessed depending on how it is handled

- **Ambiguous file**: is one that **is interpreted** differently depending on the parser. One parser may crash or fail to process it, while other may return a valid file.

- **Chimera file**: file has some data that is interpreted as different types

Content Type Obfuscation - Polyglots

Use in Malware

https://nvd.nist.gov/vuln/detail/CVE-2009-1862

…allows remote attackers to **execute arbitrary code** or cause a **denial of service** (memory corruption) via (1) a **crafted Flash application in a .pdf file** or (2) **a crafted .swf file**, related to authplay.dll, as exploited in the wild in July 2009.

Content Type Obfuscation - Polyglots

Strategies

- Stacks: Data is appended to the file
- Cavities: Uses blank (non used space) in the file
- Parasites: Uses comments or metadata fields that allow content to be written
- Zippers: mutual comments

Content Type Obfuscation - Polyglots

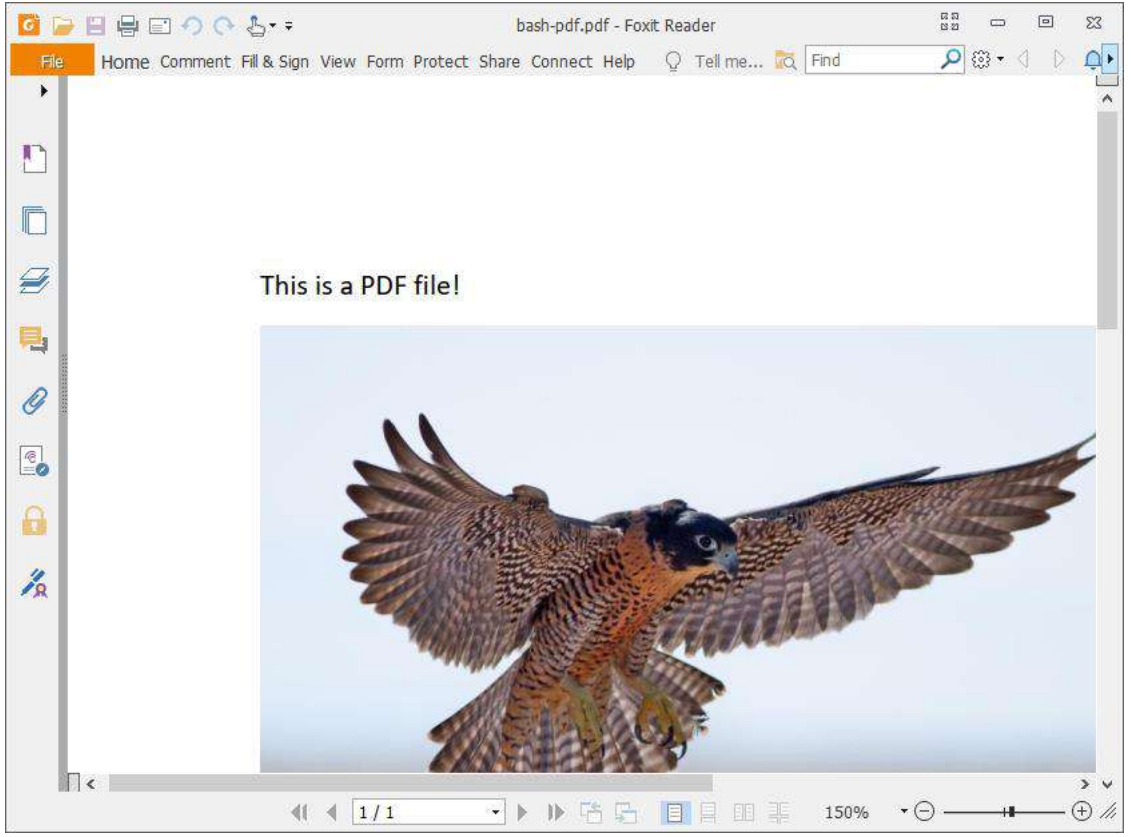
Empty Space

- Files sometimes allow empty or unused space
  - Before, in the middle or after actual content (appended)
  - Most common in Block formats (ISO and ROM dumps, TAR archives)
    - NAND dumps, ROM dumps, ISOs are directly mapped to sectors
  - Some formats allow arbitrary bytes before file start (e.g. PDF)
    - PDFs are processed from the end
- “Empty space” can be abused to inject crafted content

A simple bash-pdf polyglot

bash-pdf.pdf

```
$ file bash-pdf.pdf
bash-pdf.pdf: POSIX shell script executable (binary data)
$ ./bash-pdf.pdf
Hello World
```




```
1   #!/bin/bash
2   echo "Hello World"; exit
3   %PDF-1.7
4   %µµµµ
5   1 0 obj
6   <</Type/Catalog/Pages 2 0 R/Lang(en-US) /StructTreeRoot 11 0 R/MarkInfo<</Marked true>>/Metadata 23 0 R/ViewerPreferences 24 0 R>>
7   endobj
8   2 0 obj
9   <</Type/Pages/Count 1/Kids[ 3 0 R] >>
10  endobj
11  3 0 obj
12  <</Type/Page/Parent 2 0 R/Resources<</Font<</F1 5 0 R>>/ExtGState<</GS7 7 0 R/GS8 8 0 R>>/XObject<</Image9 9 0 R>>/ProcSet[/PDF/Text/ImageB/ImageC/ImageI]
    >>/MediaBox[ 0 0 612 792] /Contents 4 0 R/Group<</Type/Group/S/Transparency/CS/DeviceRGB>>/Tabs/S/StructParents 0>>
13  endobj
14  4 0 obj
15  <</Filter/FlateDecode/Length 245>>
16  stream
```

A simple bash-pdf polyglot

Why?

- PDF is a collection of objects
  - Objects are dictionaries of properties with a named type
  - Called “CosObjects” or Carousel Object System
  - Simply added to file. New revisions will create new objects that are appended
  - A PDF can have unused object
  - Objects can contain executable code (the code is not executed by the pdf reader!)
    - Objects can contain anything!
    - Well…. There is the LAUNCH action, and Javascript is a valid object type…

A simple bash-pdf polyglot

A simple object

```
1 0 obj
<</length 100>>
stream

…100 bytes..



endstream
endobj
```

A simple bash-pdf polyglot

Two objects

```
1 0 obj
<</length 100>>
stream
…100 bytes..
endstream
Endobj
2 0 obj
<</length 100>>
stream
…100 bytes..
endstream
endobj
```

A simple bash-pdf polyglot

Two objects and something else that is not parsed

```
1 0 obj
<</length 100>>
stream
…100 bytes..
endstream
Endobj
I should not be here, but who cares. And I could be anywhere
2 0 obj
<</length 100>>
stream
…100 bytes..
endstream
endobj
```

A simple bash-pdf polyglot

The XREF Table

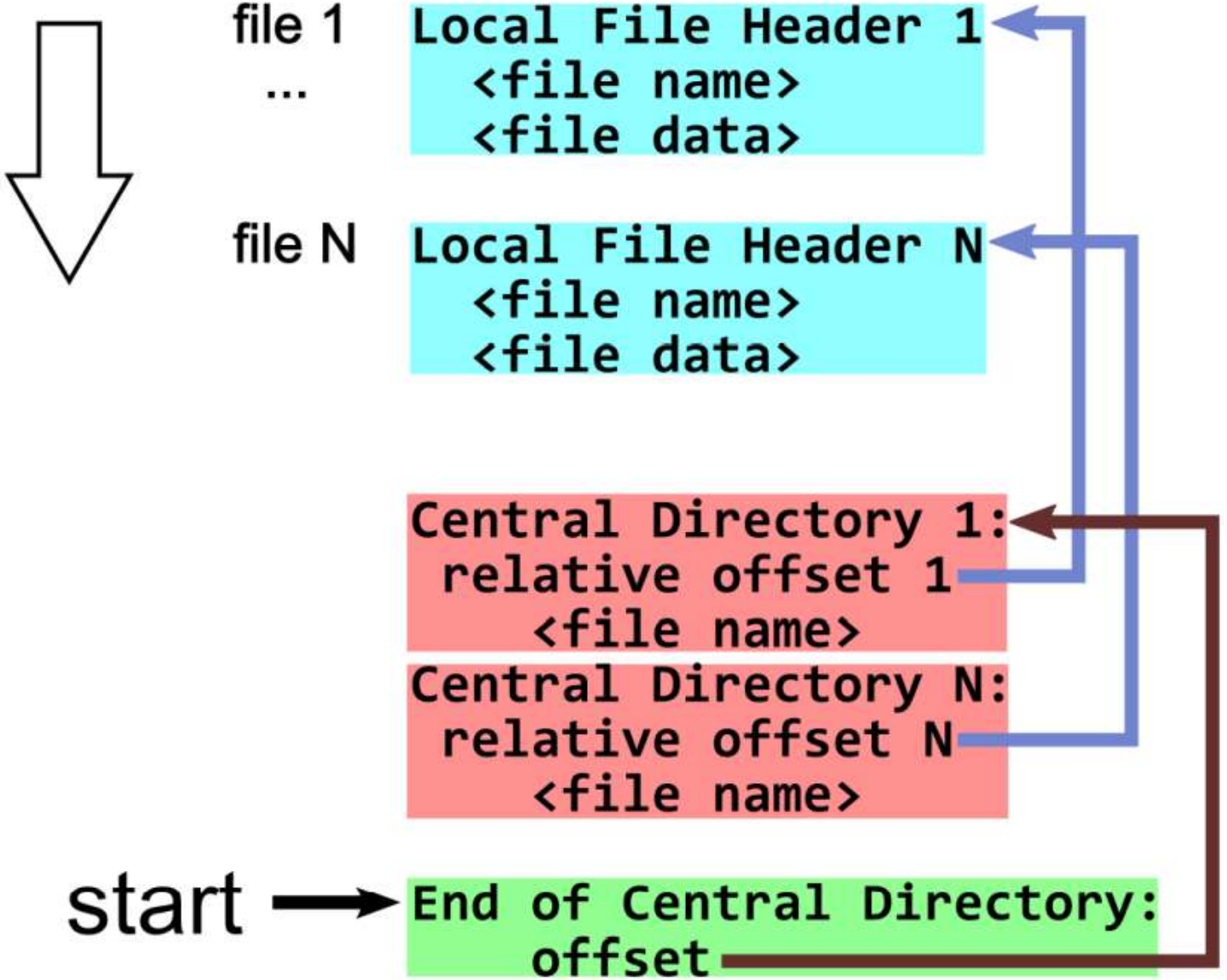
- PDF have a table with the offset of every object
  - In the end!
  - Reader skips to the end of the file, reads the table and parses the objects
    - That's one reason why it ignores garbage between objects

- XREF table also defines where the file magic (`%PDF-1.5\n\n`) is
  - There may be some bytes before the magic
  - Actually, 1024 random bytes are allowed

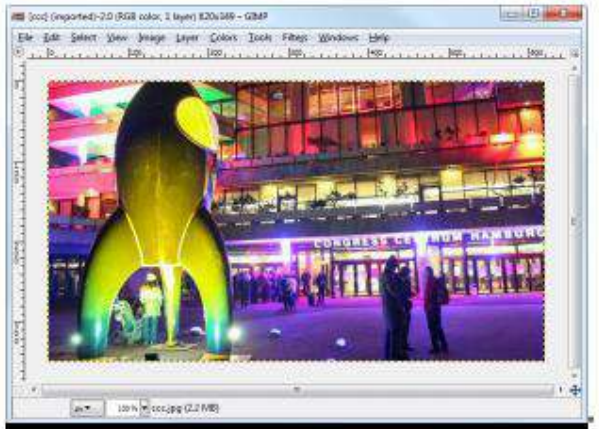
```
1   xref
2   0 26
3   0000000011 65535 f
4   0000000017 00000 n
5   0000000166 00000 n
6   0000000222 00000 n
7   0000000511 00000 n
8   0000000830 00000 n
9   0000000998 00000 n
10  0000001237 00000 n
11  0000001290 00000 n
12  0000001343 00000 n
13  0000055720 00000 n
14  0000000012 65535 f
15  0000000013 65535 f
16  0000000014 65535 f
17  0000000015 65535 f
18  0000000016 65535 f
19  0000000017 65535 f
20  0000000018 65535 f
21  0000000019 65535 f
22  0000000020 65535 f
23  0000000000 65535 f
24  0000056466 00000 n
25  0000056683 00000 n
26  0000083140 00000 n
27  0000086318 00000 n
28  0000086363 00000 n
29  trailer
30  <</Size 26/Root 1 0 R/Info 10 0 R/ID[<85F88F67066D2E4AAB78E636585E887B><85F88F67066D2E4AAB78E636585E887B>] >>
31  startxref
32  86664
33  %%EOF
34  xref
35  0 0
36  trailer
37  <</Size 26/Root 1 0 R/Info 10 0 R/ID[<85F88F67066D2E4AAB78E636585E887B><85F88F67066D2E4AAB78E636585E887B>] /Prev 86664/XRefStm 86363>>
38  startxref
39  87341
40  %%EOF
```

Offsets of object locations

ZIP


file 1
...
Local File Header 1
<file name>
<file data>
file N
Local File Header N
<file name>
<file data>
Central Directory 1:
relative offset 1
<file name>
Central Directory N:
relative offset N
<file name>
start
End of Central Directory:
offset

JPG

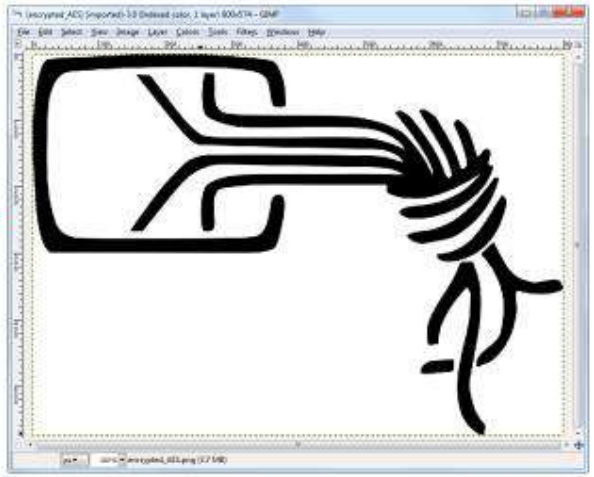


JAR
(ZIP + CLASS)

```
>java -jar ccc.jpg
Hello World! [Java]

>
```

AES_{K_1}



PNG

AES_{K_2}

FLV

3DES

PDF

Content Type Obfuscation - Polyglots

Practical application

- Malware makes use of polyglots are means to circumvent filters
  - A Packet/Email/Web application firewall will block executables, but will it block JPGs?
    - If it does, can it be done with a low rate of false positives?

- General process involves download a polyglot and a decoder
  - Polyglot contains malicious code
  - Decode is implemented in a less suspicious manner (e.g., Javascript)

- From a Reversing Perspective: how much effort will we spend analyzing a JPG?
  - Automated tools such as `binwalk`, `TrId` and file can help (but are limited)