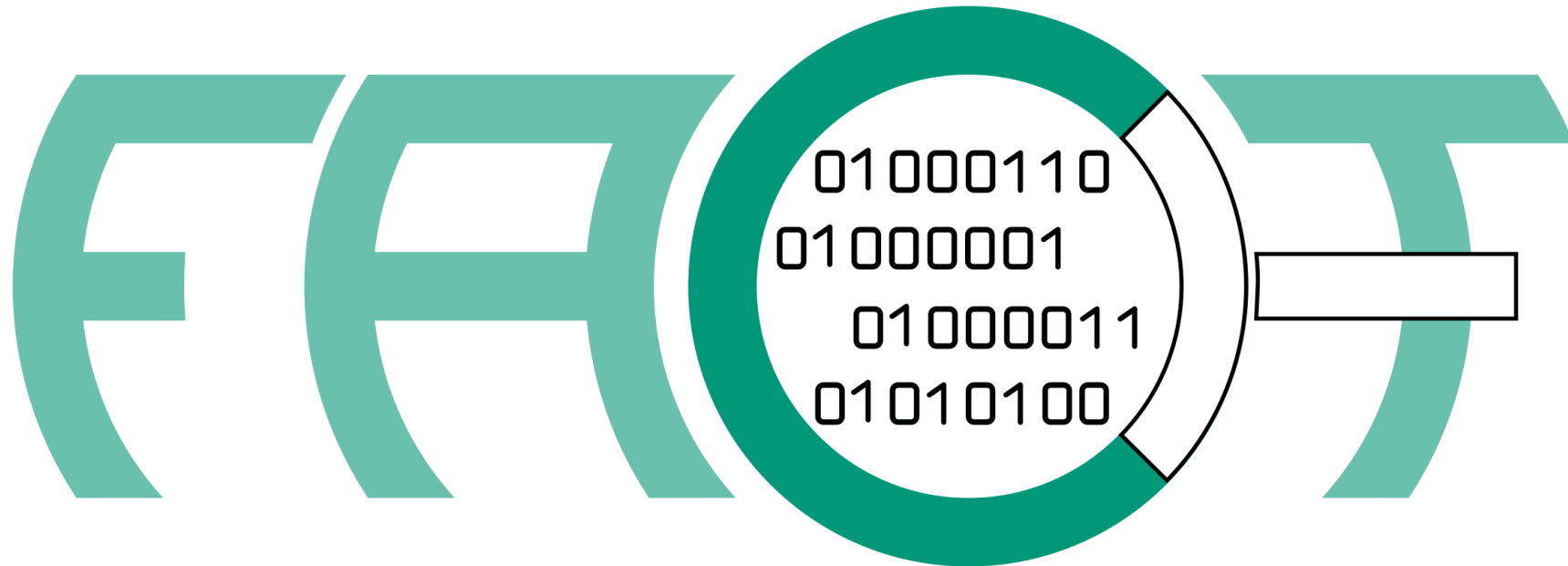

AUTOMATED UNPACKING, ANALYSIS AND COMPARISON OF ARBITRARY FIRMWARE IMAGES

The Firmware Analysis and Comparison Tool (FACT)



FIRMWARE **A**NALYSIS AND **C**OMPARISON **T**OOL

Who Are We?

- FKIE ~ Research institute for communication, information processing and ergonomics
 - Department CAD – Cyber Analysis and Defense
- Fellow developers: Peter Weidenbach, Jörg Stucke and Raphael Ernst

Some Practical Information

- GitHub Link for FACT
 - https://github.com/fkie-cad/FACT_core

WORKSHOP SCOPE

- Introduction to FACT
- Application of FACT in firmware / hardware analysis context
- Design details supporting applicability
- Demo, Demo and some more Demo

Why something new?

- Attacks on Firmware vulnerabilities are on the rise (see botnet, mirai)
- Firmware analysis offers unique challenges
 - Extracting firmware from containers, finding important parts
 - Handling multiple architectures

Firmware



Vulnerability



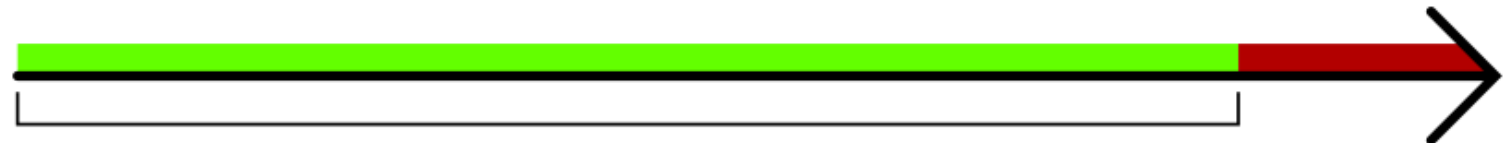
Advantage of FACT

- Bundle multiple steps to gain easily combine results
- Move manual responsibilities to machine

Firmware



Vulnerability



FACT

Introduction – Idea of FACT



- Automate as much as possible ..
 - .. and reasonable
- Includes
 - unpacking
 - keeping meta data
 - simple analysis
 - firmware comparison

Introduction – Idea of FACT

The screenshot displays the FACT web interface for a firmware analysis. The browser address bar shows the URL: localhost:5000/analysis/96733bad61b4f76e08b6a7fe76fc1d65cf47aba760dba3b5e3291318845badce_20439364. The navigation bar includes links for Home, Database, Compare, Upload, Statistic, System Health, and About.

Analysis for Asus RT-AC58U - 3.0.0.4.380.6516

UID: 96733bad61b4f76e08b6a7fe76fc1d65cf47aba760dba3b5e3291318845badce_20439364

General	
device name	RT-AC58U
vendor	Asus
device class	Router
version	3.0.0.4.380.6516
release date	2016-10-05
file name	FW_RT-AC58U_3.0.0.4_380_6516-g6772678.trx
virtual path	Asus RT-AC58U - 3.0.0.4.380.6516 (Router)
file size	19.49 MiB (20,439,364 bytes)
file type	ulimage header, header size: 64 bytes, header CRC: 0x7F1064C3, created: Fri Sep 9 07:37:40 2016, image size: 20439300 bytes, Data Address: 0x80208000, Entry Point: 0x80208000, data CRC: 0x204F2DB7, OS: Linux, CPU: ARM, image type: OS Kernel Image, compression type: lzma, image name: "l003"

Analysis Results	
binary analysis	
cpu architecture	
crypto material	
cve matching	
file hashes	
file type	
software components	
unpacker	

File Tree

- FW_RT-AC58U_3.0.0.4_380_6516-g6772678.trx (19.49 MiB)
 - uboot.lzma (19.49 MiB)
 - _25a2c3c4cbef366969627c17c5518aa5bc46d0fac8b4095ec3fd84df49792a52_20439300.extracted
 - 202D40.squashfs (17.45 MiB)
 - E0.7z (19.49 MiB)
 - uboot_header.bin (64.00 Byte)

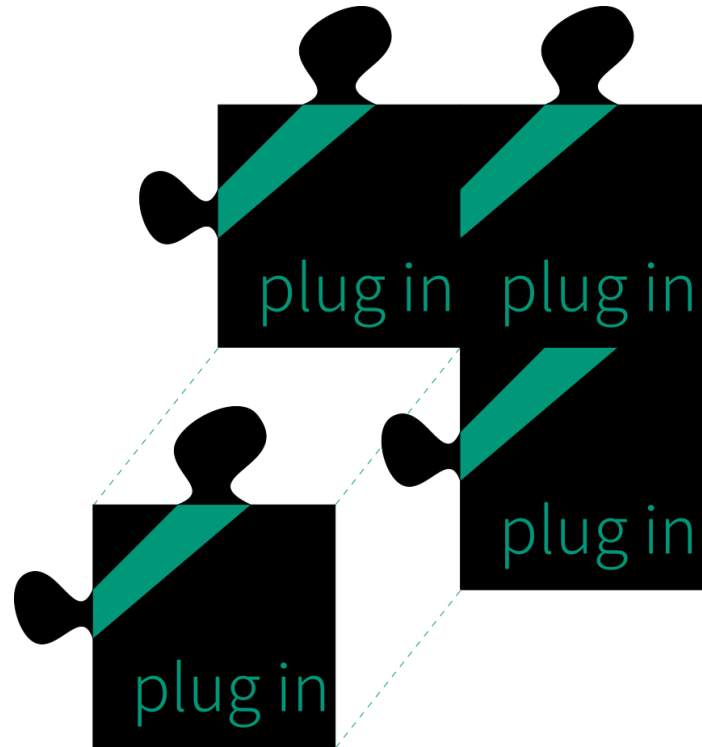
- Provide comprehensible GUI for novices and experts
 - Web-based GUI allows easy application in local and remote environments

Introduction – Idea of FACT



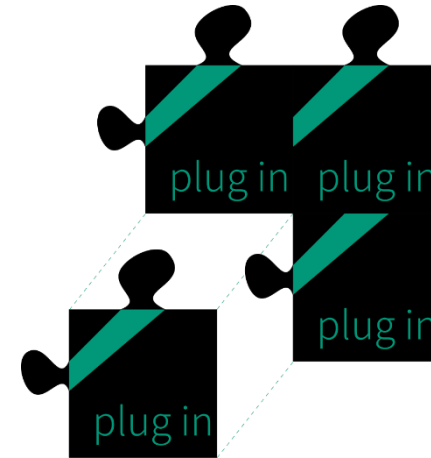
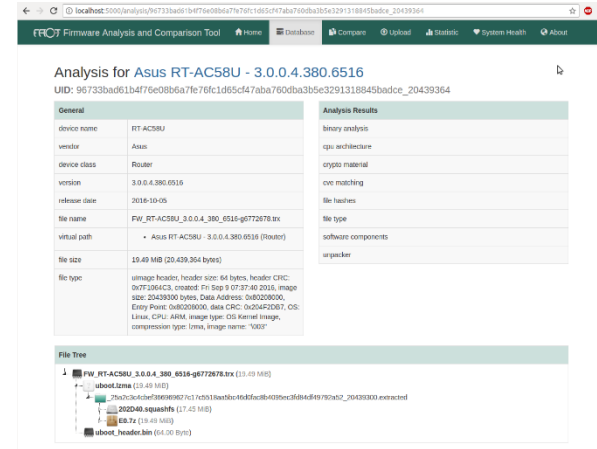
- Offer database that allows
 - archiving
 - searching
 - aggregating
 - ..

Introduction – Idea of FACT



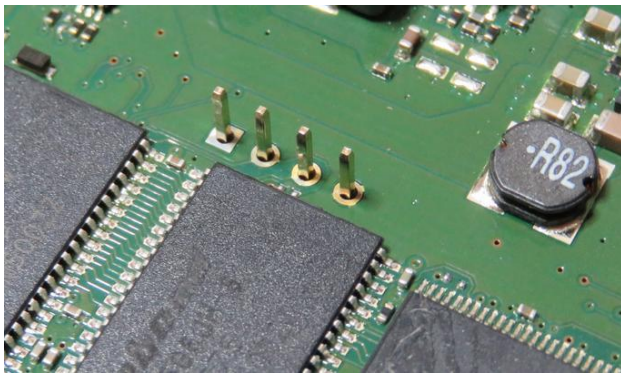
- Plugin architecture to allow extension of
 - Unpacking capabilities
 - Analysis functionality
- Plugins should have as few overhead as possible

Introduction – Idea of FACT



Application of FACT - What can be automated ?

- What tasks are there or
 - what is firmware analysis?
- Differences exist in
 - Starting points (Device, PCB, Firmware dump, Firmware update file / executable, ..)
 - Viewpoint of analysis (Manufacturer, White hat, Researcher, Black hat ..)
 - Aim of analysis (Detect components, Find vulnerabilities, Modify content / code ..)



```
[00000008] pt 1024
% offset      8 1 2 3 4 5 6 7 8 9 A B C D E F   01:50789ABCDEF
00000008 1698 4818 1799 4843 0c99 7043 4018 0c99  ..8...MC..PC...
00000009 C201 0098 1220 0000 20 0110 21 079a  ..8  ..  ..  ..
0000000a 1898 092 0a9a 23 1021 f7 ccf8 84  ..8  .1  ..  ..
0000000b 0120 01f0 17 2c 0500 0121 4420 01f0  ..  ..  ..  ..  ID
0000000c 1174 2000 0767 0121 20 01f0 0000 0000  ..  ..  ..  ..  ..
0000000d 0980 0408 1030 0818 9880 1998 1822 0149  ..8  ..  ..  ..  ..
0000000e 02f0 0000 1220 01f0 f5fc 3022 1021 92  ..  ..  ..  ..  ..
0000000f 199a 1898 0b  f7 a5fe 840 0120 01f0  ..  ..  ..  ..  ..
00000010 f1f0 02c 0400 0121 4020 01f0 05fc 2960  ..  ..  ..  ..  ..
00000011 0121 020 01f0 c0fc 2060 020 01f0 02fc  ..  ..  ..  ..  ..
00000012 1998 1022 3830 050 0100 03f0 9a88 079a  ..  ..  ..  ..  ..
00000013 1998 092 23 2a00 1021 f7 04fe 840  ..  ..  ..  ..  ..
00000014 0120 01f0 cffe 2c 0400 0121 4720 01f0  ..  ..  ..  ..  ..
00000015 01fc 0100 0121 20 01f0 c0fc 20 1887  ..  ..  ..  ..  ..
00000016 35 0700 109f 1c0 2c40 0120 2a49 3e04  ..  ..  ..  ..  ..  HA*16
00000017 0400 43c2 11a0 07f0 0100 097 07c0 0700  ..  ..  ..  ..  ..
00000018 07c0 2100 f7 aafe 85 20 01f0 aafe  ..  ..  ..  ..  ..
00000019 2240 2169 c0fc 04aa 43c2 11a0 07c0 23  ..  ..  ..  ..  ..  H*1
0000001a 010c 097 07c0 0700 0700 f7 07fe 28  ..  ..  ..  ..  ..
0000001b 0200 1d20 f0 c1fa 0120 01f0 03fe 28  ..  ..  ..  ..  ..
0000001c 0000 f000 85 0100 0000 1000 0000 32  ..  ..  ..  ..  ..
0000001d f432 0198 14 3100 f7 18fe 28 0701  ..  ..  ..  ..  ..  2
0000001e 22 092 0198 210 210 0021 f7 28fe  ..  ..  ..  ..  ..  # 2 1
0000001f 28 1701 022 092 23 2c0 0198 0021  ..  ..  ..  ..  ..  # 2
00000020 0000 000 2037 0100 8c0 8188 3130 0000  ..  ..  ..  ..  ..  7 10
00000021 0120 000 28 8188 020 8188 0111 0000  ..  ..  ..  ..  ..  00
00000022 f7 11fe 28 1d01 3520 f0 07fa 3e27  ..  ..  ..  ..  ..  15 5
00000023 120 0124 0400 0102 0102 0000 240c 0900  ..  ..  ..  ..  ..  0.50.10.5.0
00000024 23 0219 0198 1021 097 f7 cfd 28  ..  ..  ..  ..  ..  #
00000025 0801 3119 0e04 300c 1010 0704 1fc 2c1b  ..  ..  ..  ..  ..  11.10.0.20.0
00000026 13 02 0001 0100 0000 1000 02 0400 0001  ..  ..  ..  ..  ..  2
00000027 13 02 0102 07 28fb 00 2c 0200  ..  ..  ..  ..  ..  2
00000028 1302 0102 0102 1c00 0100 0000 0000  ..  ..  ..  ..  ..  4 10
00000029 0192 091 0e08 0748 0a30 f0 14fb 040  ..  ..  ..  ..  ..  .8
0000002a 2c 0500 3420 f0 01f0 0000 1000 1000  ..  ..  ..  ..  ..  1 12
0000002b 0260 c108 0192 091 0e08 0e48 1030 f0  ..  ..  ..  ..  ..  1.h.  H.0
0000002c 02f0 040 2c 0200 3420 f0 37fa 200  ..  ..  ..  ..  ..  .4 .7.
0000002d 1000 7030 0c 150 0700 0099 0102 91  ..  ..  ..  ..  ..  1
0000002e 2a 2100 f0 e7fa 040 2c 0200 3420 * 1  ..  ..  ..  ..  ..  1
0000002f 1f0 24fa 10 7000 0800 0000 0000  ..  ..  ..  ..  ..  .8.  10000.4
00000030 6481 d548 21 02f0 1608 0248 32 92 d.H  ..  ..  ..  ..  ..  H
```









```
; void init(void)
init_proc near




arg_0= dword ptr 4
arg_4= dword ptr 8
arg_8= dword ptr 0Ch

push    ebp
push    edi
push    esi
push    ebx
call    sub_8049E40
add     ebx, 33C7h
sub     esp, 0Ch
mov     ebp, [esp+ich+arg_0]
lea     esi, [off_8049EFC - 804F000h][ebx]
call   _init_proc
lea     eax, [off_8049EF8 - 804F000h][ebx]
sub     esi, eax
sar     esi, 2
test    esi, esi
jz     short loc_8049C85

xor     edi, edi
lea     esi, [esi+40]
```

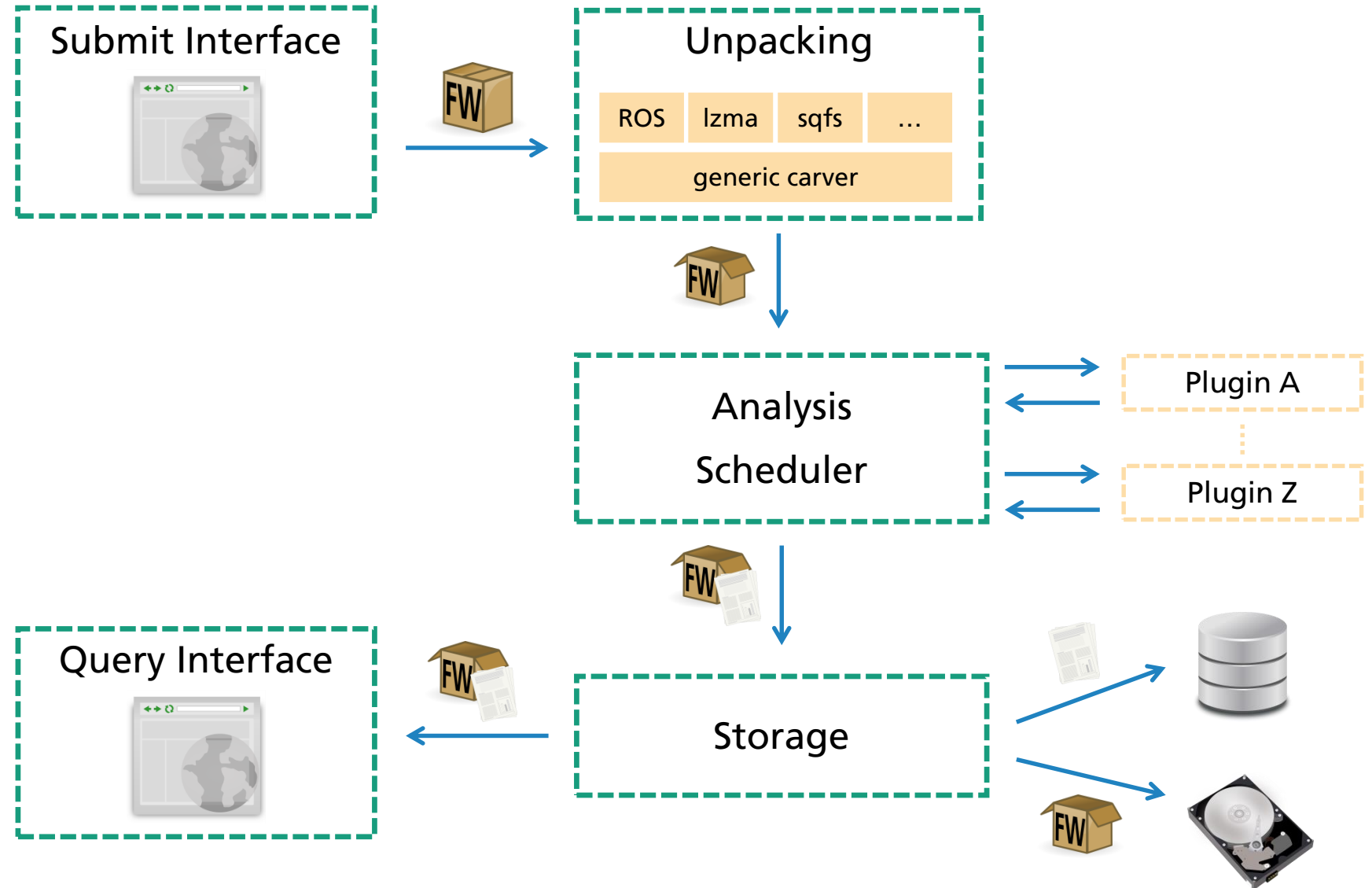
Application of FACT - What can be automated ?

- Getting firmware 
- Complex manual analysis 
- In between
 - Unpacking 
 - Feature extraction 
 - Archiving 
- As well
 - Generating statistics 
 - Keeping track of meta data 
 - Obtaining sample sets for evaluation / testing 

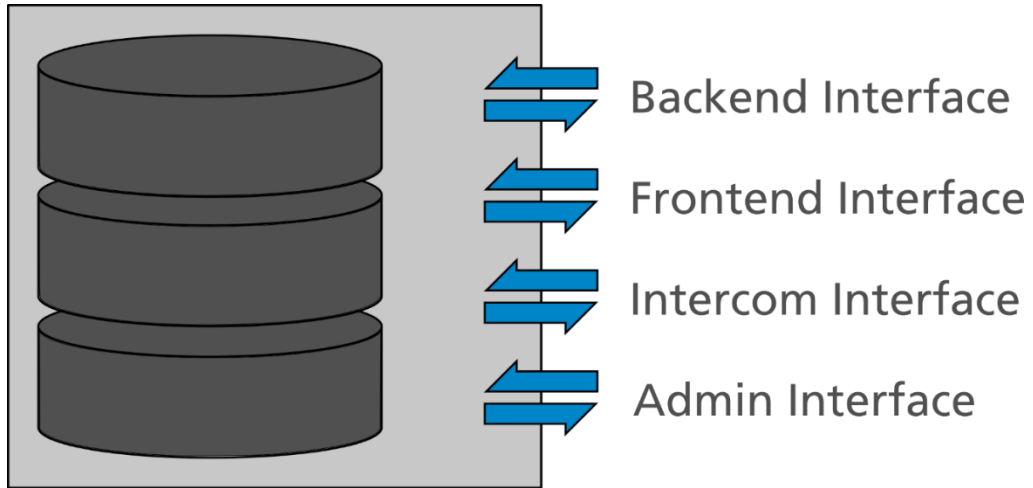
Automation		
possible	probably possible	not possible
		

Live Demo

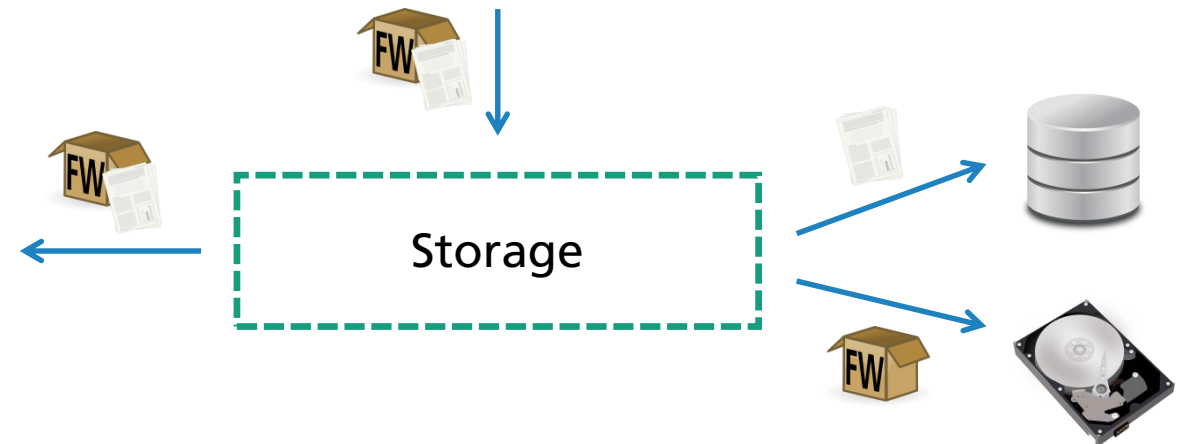
FACT Architecture



FACT Database

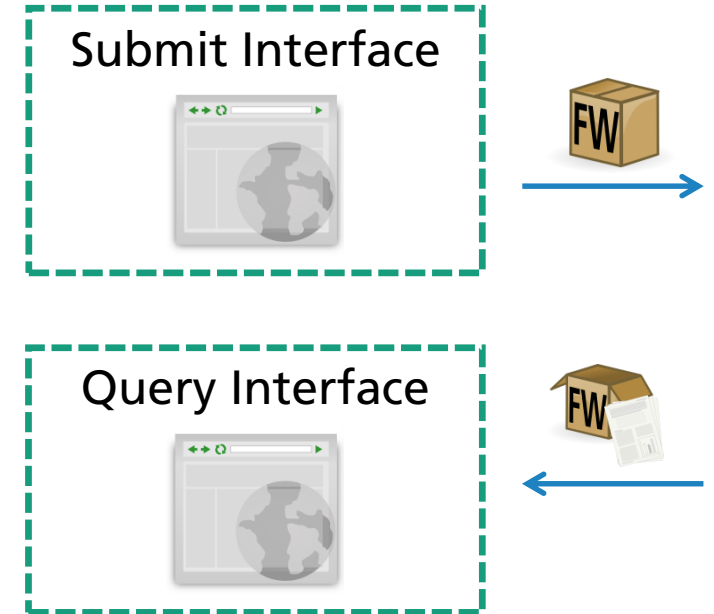


- Analysis results are stored in MongoDB
 - Multiple interfaces for better abstraction
- Firmware container and extracted files are stored directly on FS



FACT Frontend

- Python, HTML and JS
 - uWSGI as middleware, implementation via Flask incl. Jinja
 - Web-App written with Bootstrap for responsiveness
 - Load largely on server side (Jinja), Client side load minimal



uWSGI

[1]

 Flask
web development,
one drop at a time

[2]



[3]

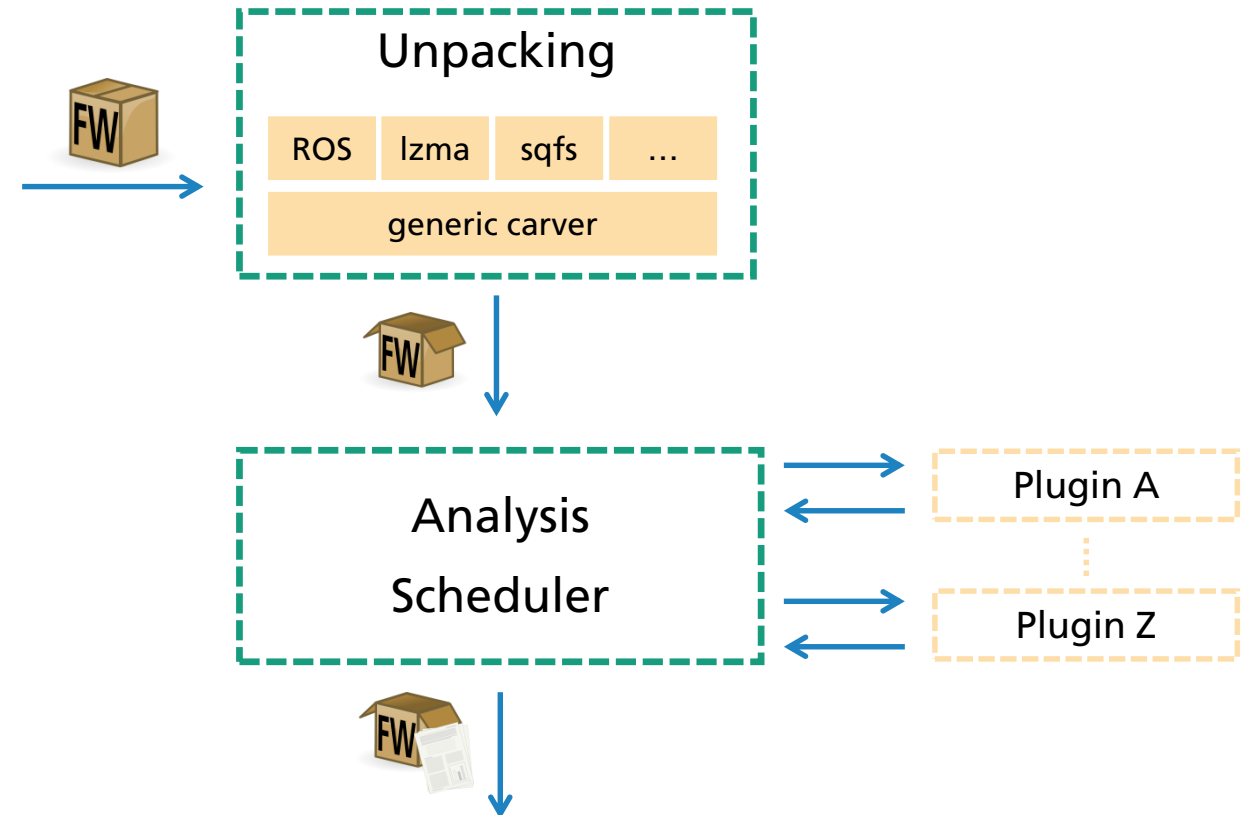
[1] <http://crazylinux.it/en/post/compile-uwsgi-php-en/>

[2] <http://flask.pocoo.org/>

[3] <http://jinja.pocoo.org/docs/2.9/>

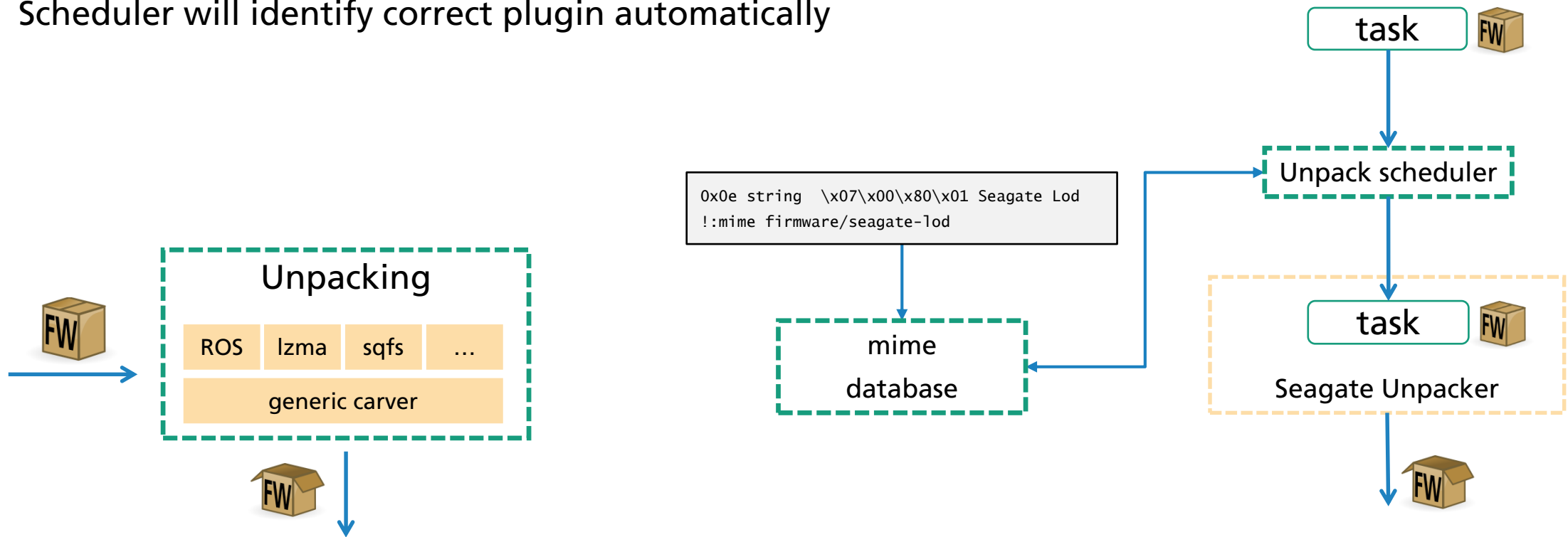
FACT Backend

- Three major parts of backend:
 - Unpacking
 - Analysis
 - Comparison
- Each part contains
 - Scheduler
 - Multiple plugins
- Each scheduler works independently



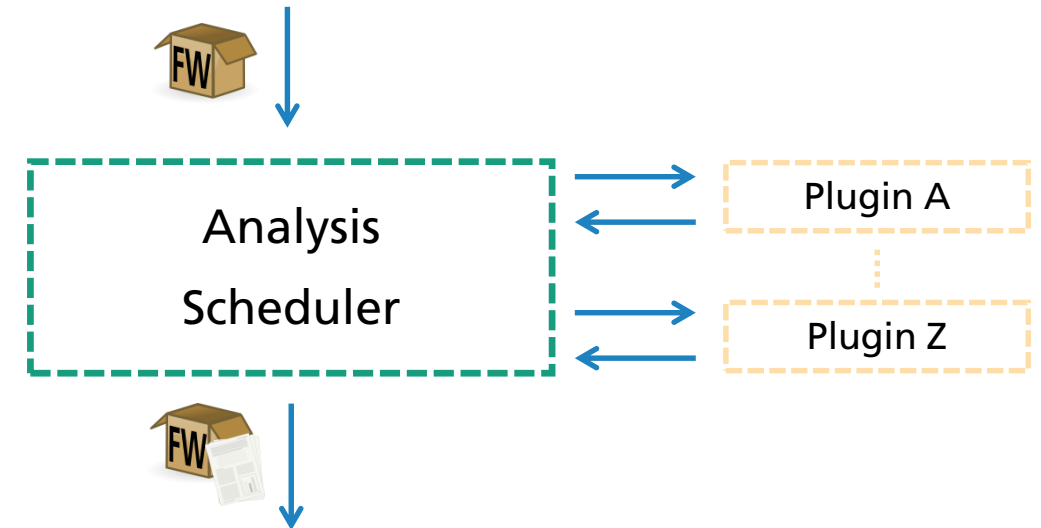
Unpack Cycle

- Identify file type using „file“ command with custom mime database
- Scheduler will identify correct plugin automatically



Analysis Cycle

- Scheduling handles plug-ins individually
 - Each plugin can have multiple worker processes
 - Dependencies allowed
- Plugins have access to binary and previous results
 - Incremental analysis possible
- Adding external tool is easy
 - python wrapper + output parser (+ html view)



Types of Analysis Plugins

Analysis plugins

Feature extraction

- [crypto code](#)
- crypto material
- ip and uri finder
- [manufacturer detection](#)
- printable strings
- string evaluator
- [version string finder](#)

Tool wrapper

- binwalk
- [firmadyne](#)
- malware scanner

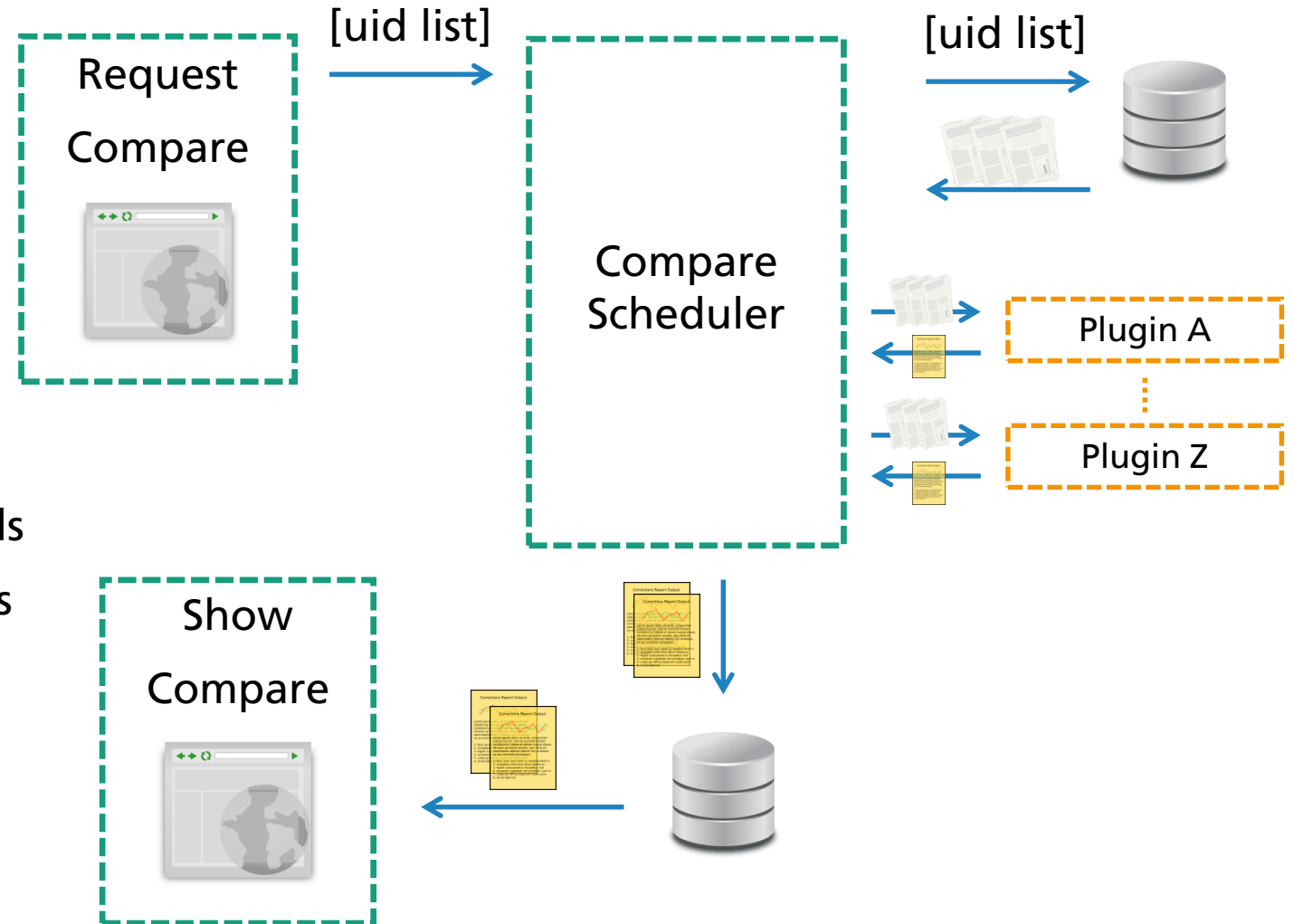
FS analysis

- init systems
- users and passwords

Research related

- [A²S²C](#)
- base64 decoder
- [binary analysis](#)
- cpu architecture
- [cve matching](#)
- software components

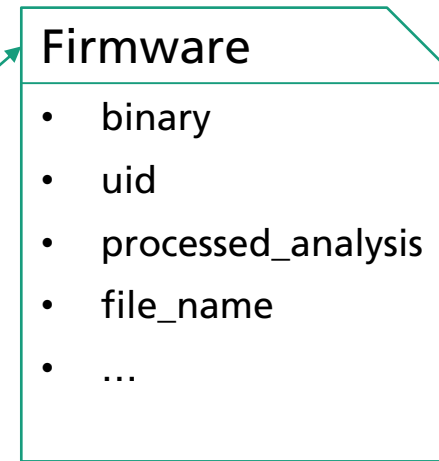
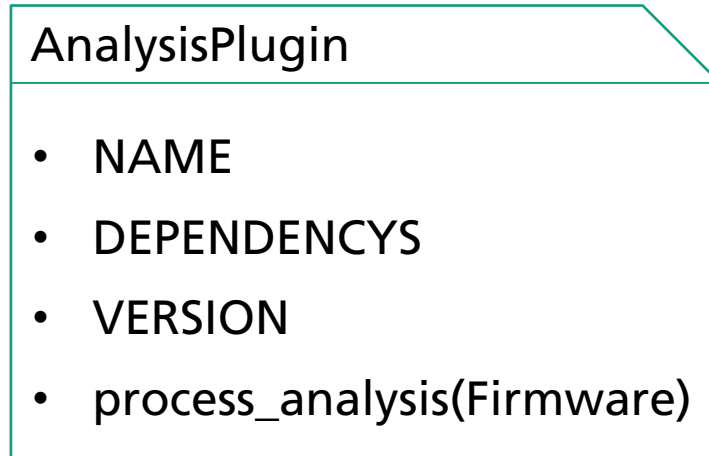
Comparison Cycle



- Separated from Analysis / Unpacking
- Triggered manually using firmware uids
 - Plugins use both binary and analysis results
 - Single threaded - Low Overhead

Live Demo

Analysis Plug-in design

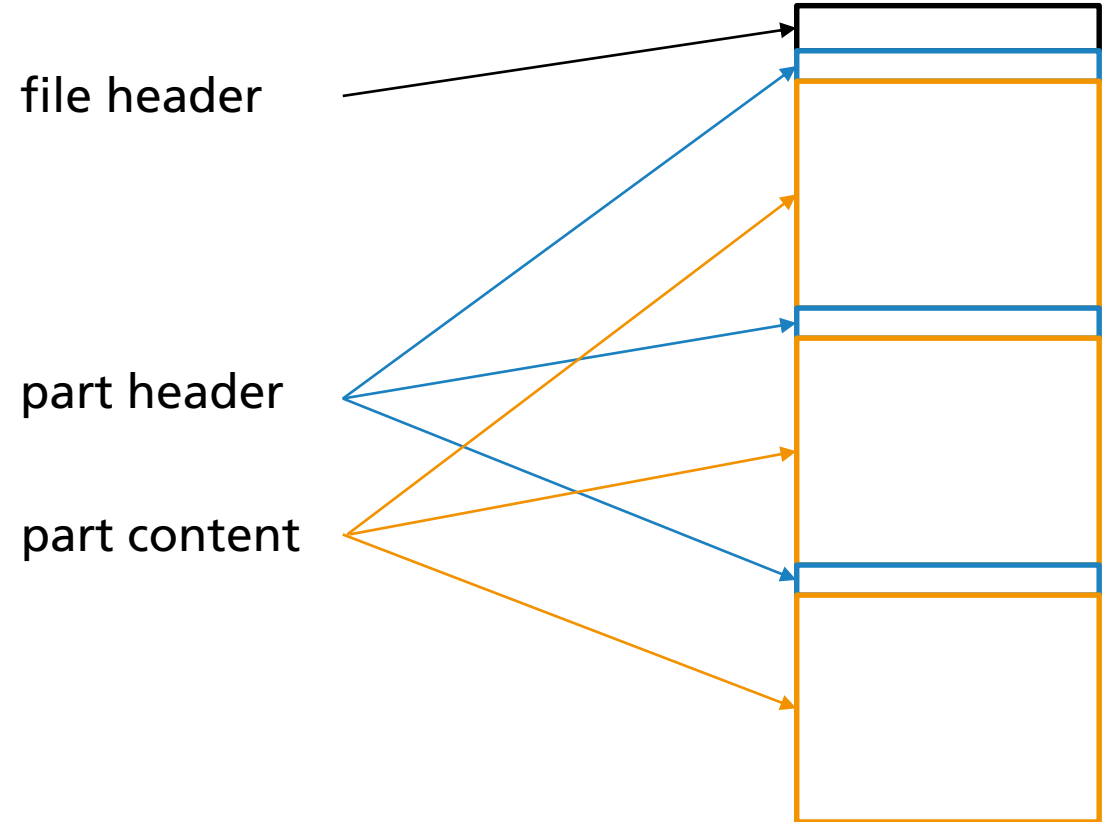
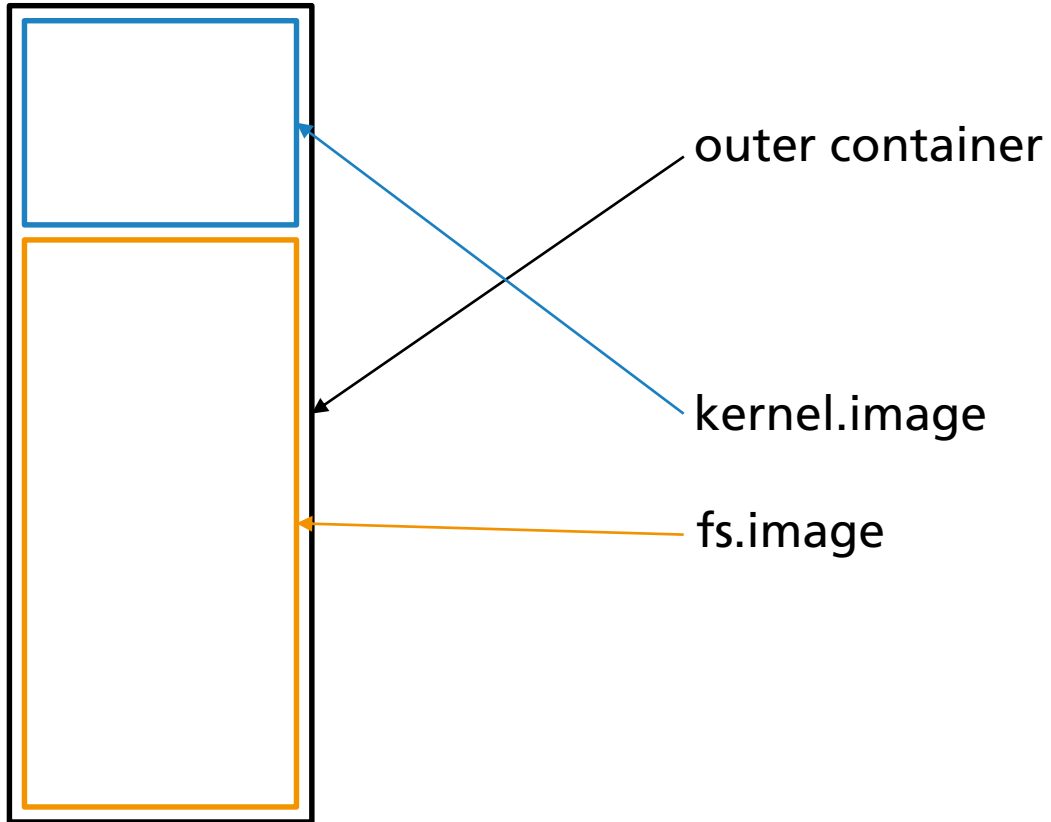


```
def process_analysis(self, file_object):
    analysis_result = analysis_function(file_object.binary)
    file_object.processed_analysis[self.NAME] = analysis_result
    return file_object
```


Typical Container Formats

Example I

Example II



Thank you for
your attention