Check Point
SOFTWARE TECHNOLOGIES LTD.

LABELESS – NO MORE

Aliaksandr Chailytko | Team Leader
Aliaksandr Trafimchuk | Malware Researcher

Thanks to:
Stanislav Skuratovich

Creation Purposes

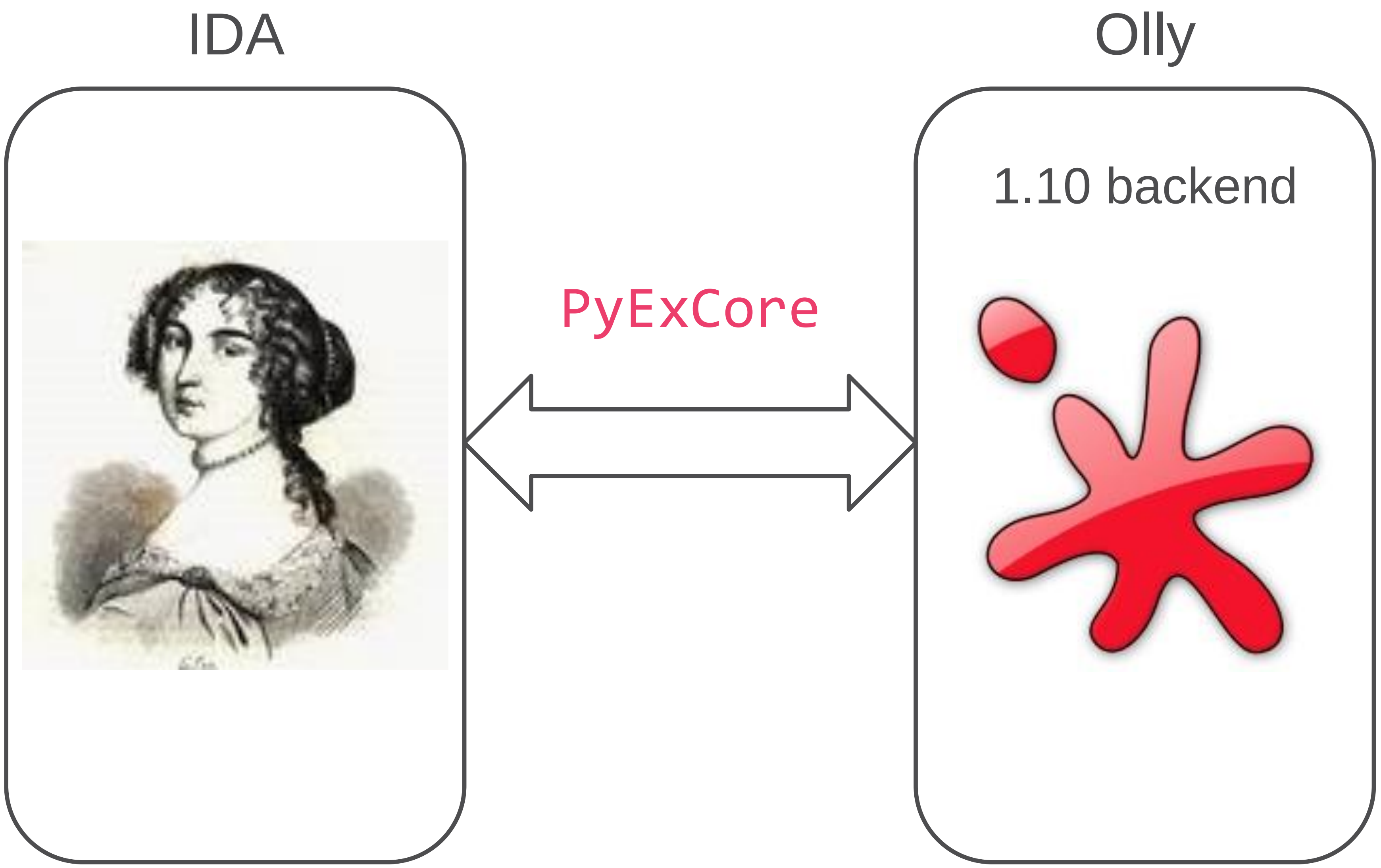
- Lack of tool for on-the-fly labels synchronization between IDA and Olly
- Lack of tool for automatic on-the-fly imports fixing
- Lack of tool for IDA-Olly scripting interaction

Main Features

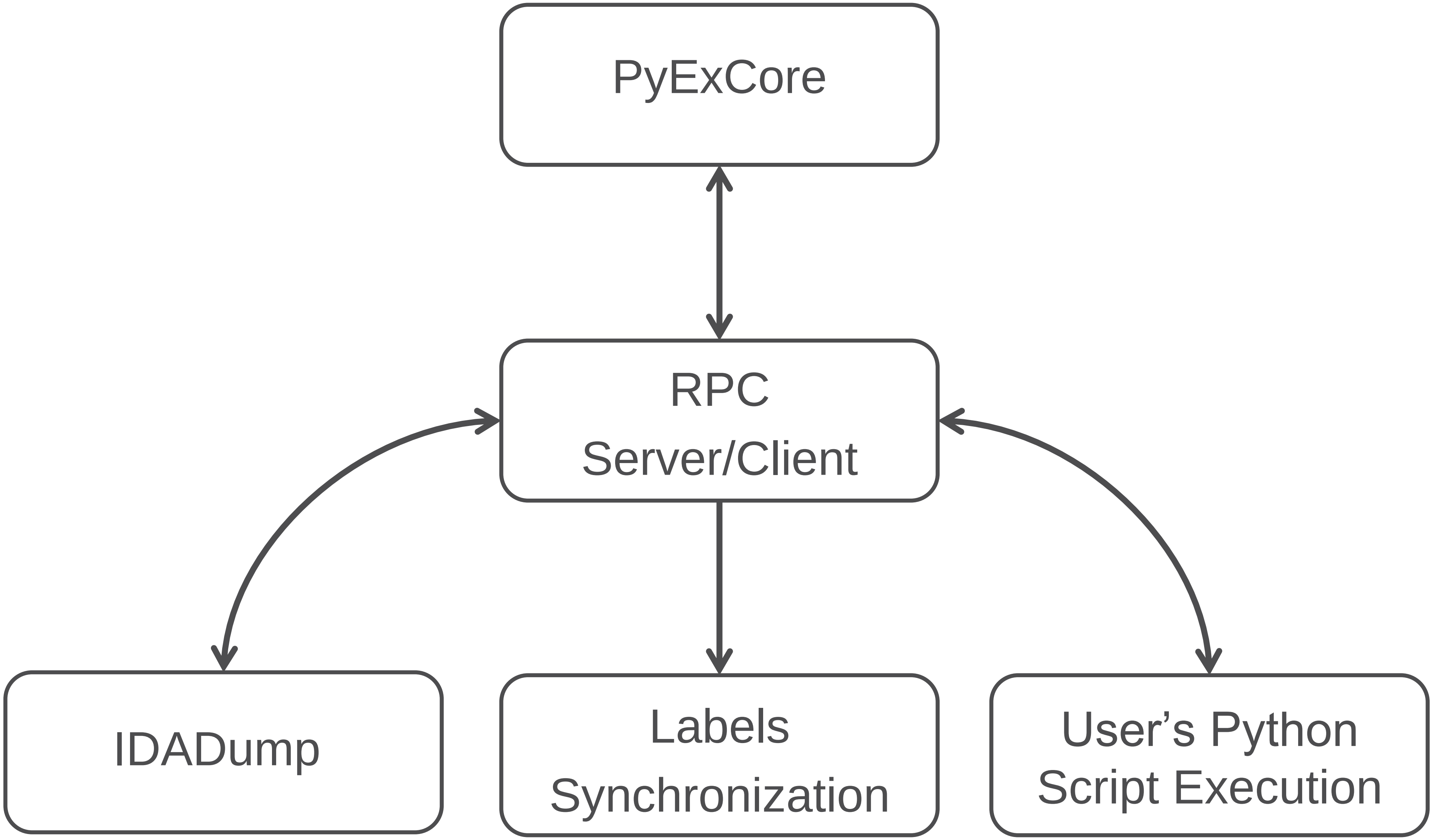
- Labels synchronization (image base-independent)
- Dumping code directly from Olly (IDADump)
- Automatic on-the-fly imports fixing
- Remote Olly Python scripts execution
- IDA-Olly scripting synergy

ARCHITECTURE

Labeless as Plugin System



PyExCore Structure



Under The Hood

- IDA API & Python for IDA side module
- Python for Olly side module
- Protobuf-based communication protocol

LABELS SYNCHRONIZATION

Synchronization of Labels

ID ...lly

ID ...cess

call fmfu_ ...A1060

...

call fmfu_ ...A3080

Existing Solutions Limitations

- Using .map file
  - Requires a lot manual work
  - Automatic updating of labels is not supported
  - Rebasing is not supported
- Applying manually
  - Time consuming
  - Error-prone

Labeless Solution

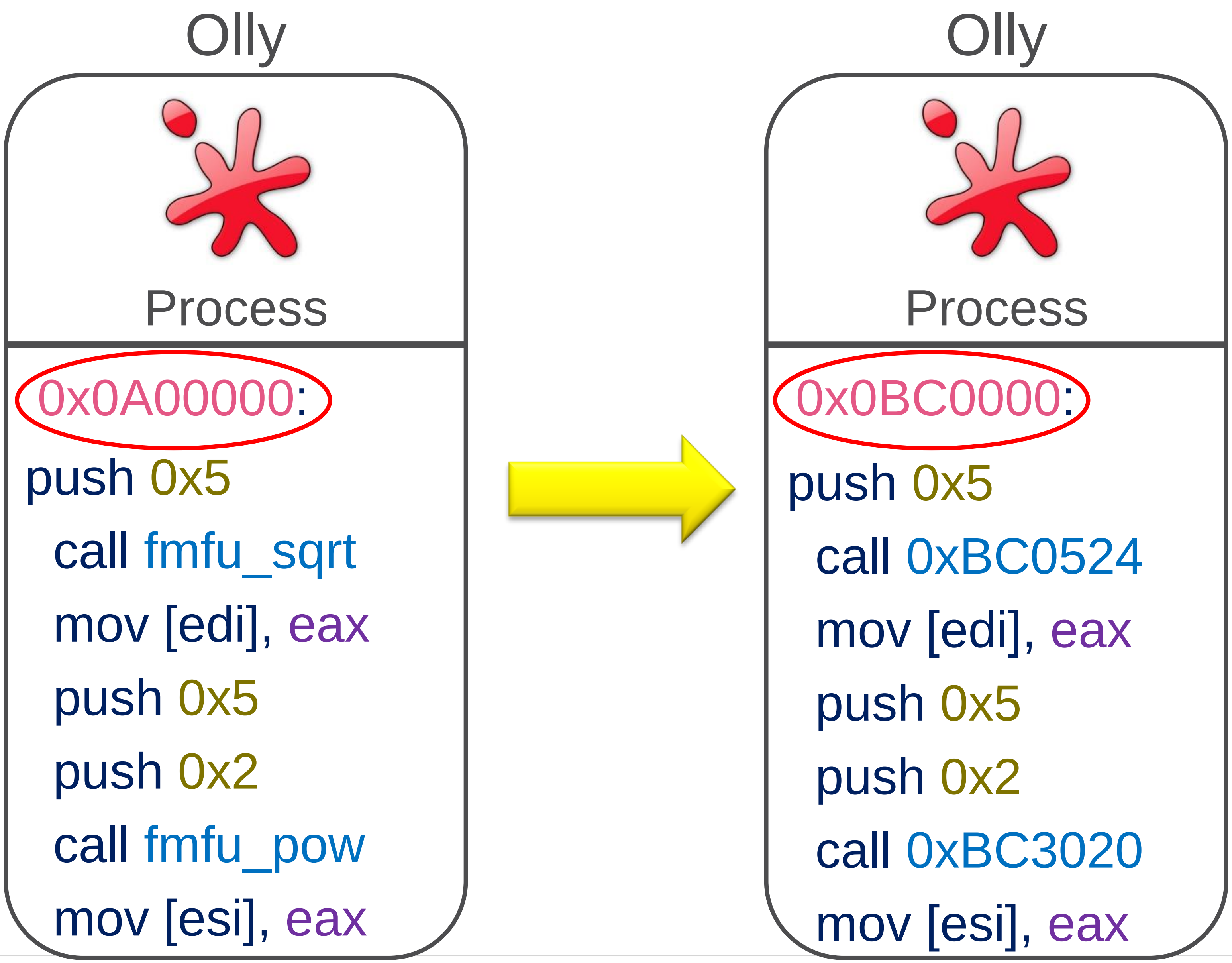
- Seamless synchronization of labels
  - Function names
  - Comments
  - Global variables syncing with demangling
- Synchronization modes
  - On demand
  - On rename (update on-the-fly)
- Supports image base-independent synchronization

Without & With Labeless

```
push 0x0
push 0x0
push 0x40D2AC winhcfna.arl_ThreadWaitForObjects
push 0x0
push 0x0
call dword ptr ds:[CreateThread]
push eax
call 0x402A82 winhcfna.arl_CloseHandleOperations
call 0x41563D winhcfna.arl_RegistryKeyOperations
push 0x1
call 0x411AEC winhcfna.arl_InitPeers
call 0x40488C winhcfna.arl_InitSecurityInterface
```

BASE-INDEPENDENT SYNCHRONIZATION

Rebasing Example



```
0x0A00000:
push 0x5
  call fmfu_sqrt
  mov [edi], eax
  push 0x5
  push 0x2
  call fmfu_pow
  mov [esi], eax
```

```
0x0BC0000:
push 0x5
  call 0xBC0524
  mov [edi], eax
  push 0x5
  push 0x2
  call 0xBC3020
  mov [esi], eax
```

Possible Solutions

- Fill all labels by hand
  - Takes too much time
  - Routine work that may lead to unobvious errors
- Rebase module in IDA (IDA native feature) and synchronize labels with Olly
  - May lead to problems with jumps, calls, offsets, etc.
- Force binary to allocate a memory on address we need
  - Manual work every time

Labeless Solution

Ima

0x8

arl_pow

push eb

mov eb

ess {1}

eBase:

00000

Olly

ocess {2}

BW0546:

h ebp

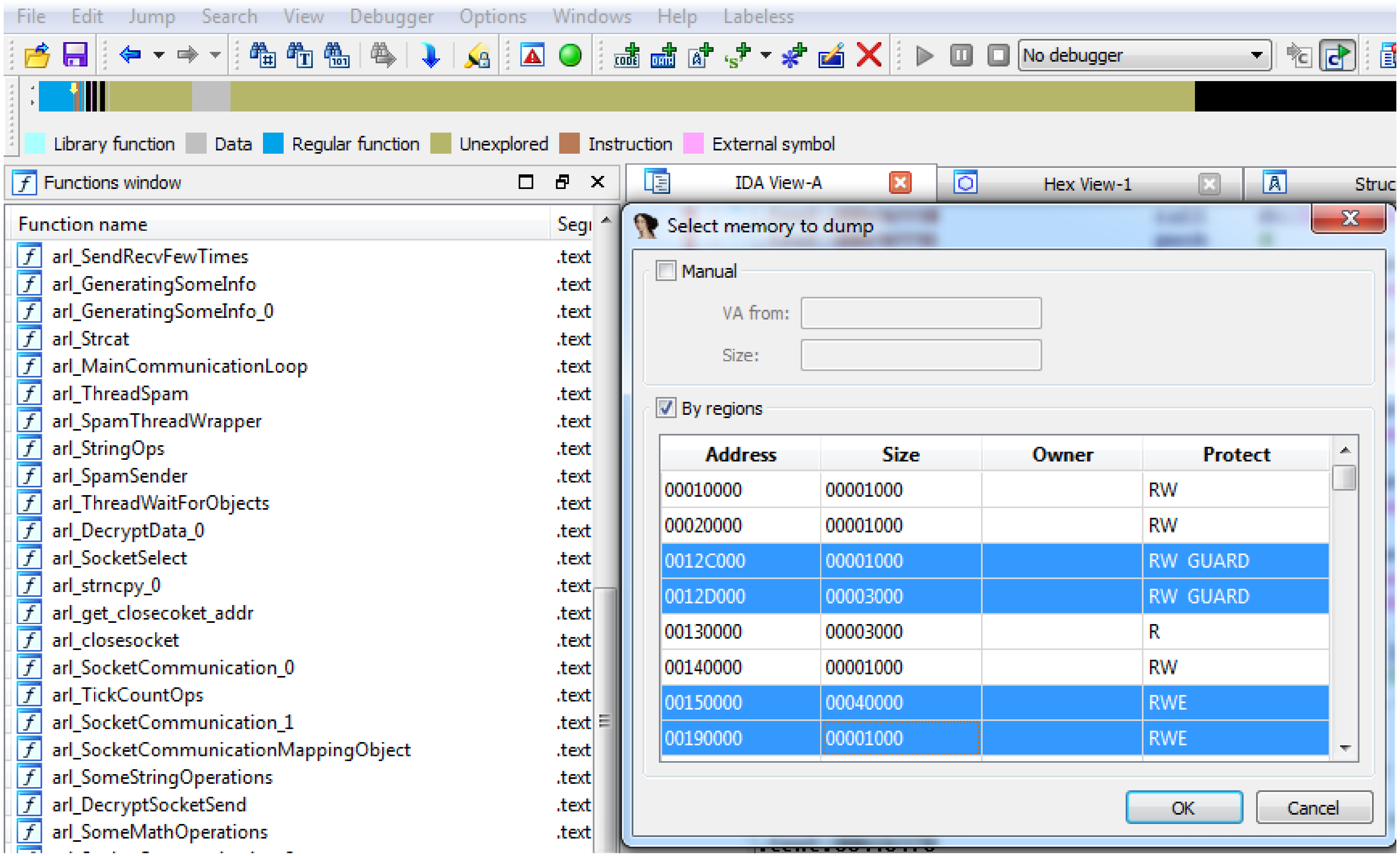
v ebp, esp

IDADUMP

IDADump Use Cases

- The debugged process has injected module which doesn’t appear in the modules list
- A heap spray was performed
- No valid PE header
- Corrupted import table
- The analyzed module contains stolen bytes
- Multiple injections

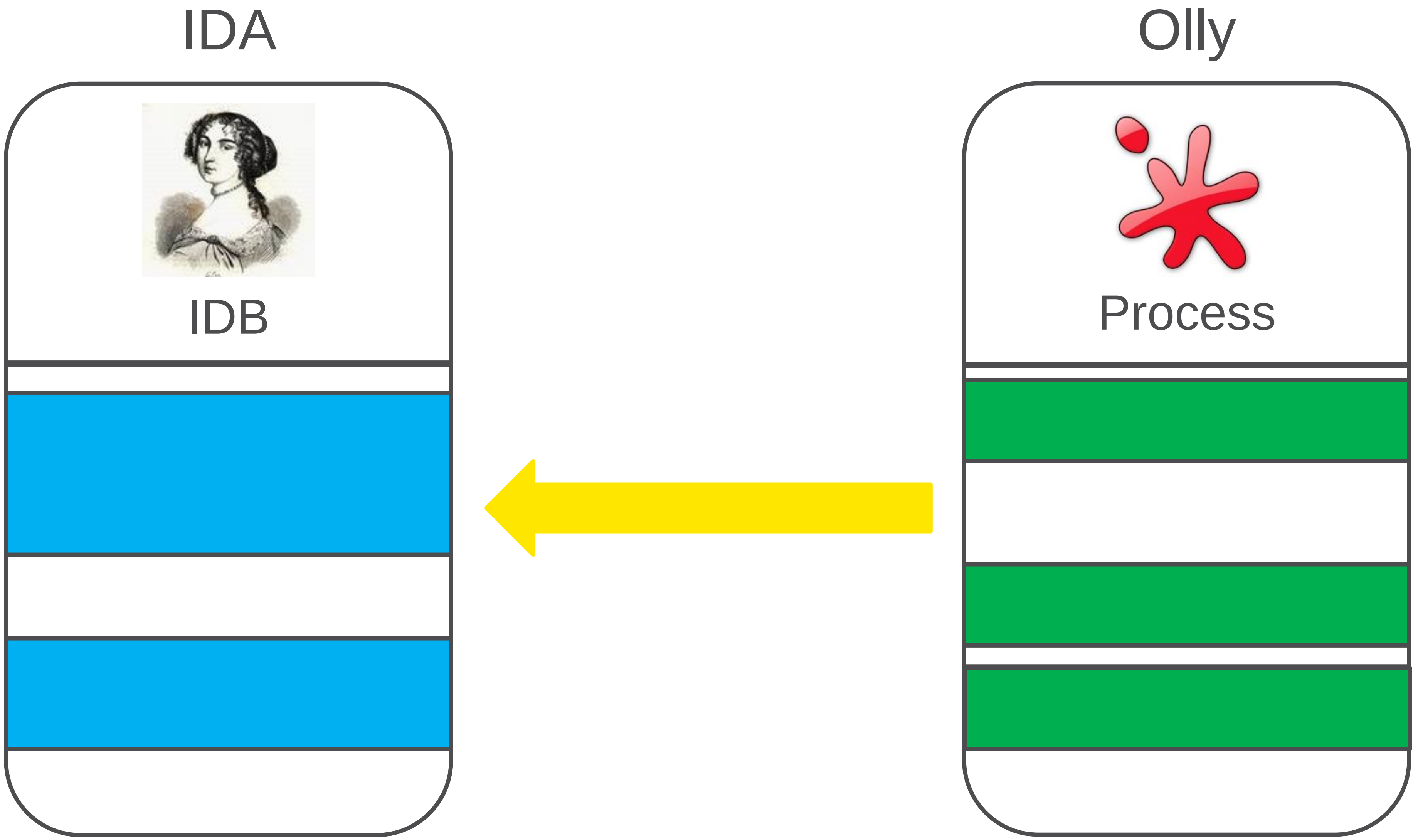
IDADump: Memory Map View



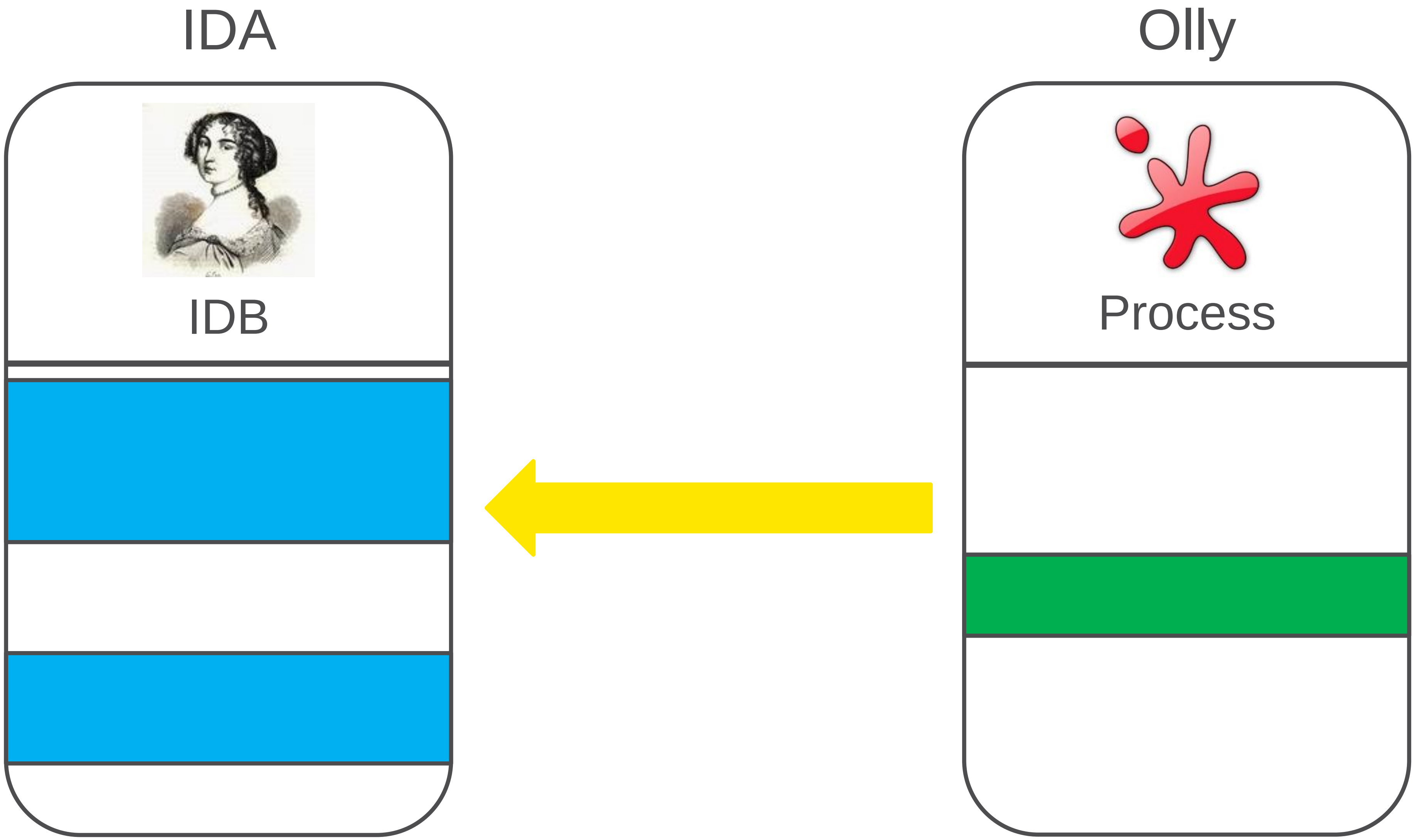
Function name	Segı
arl_SendRecvFewTimes	.text
arl_GeneratingSomeInfo	.text
arl_GeneratingSomeInfo_0	.text
arl_Strcat	.text
arl_MainCommunicationLoop	.text
arl_ThreadSpam	.text
arl_SpamThreadWrapper	.text
arl_StringOps	.text
arl_SpamSender	.text
arl_ThreadWaitForObjects	.text
arl_DecryptData_0	.text
arl_SocketSelect	.text
arl_strncpy_0	.text
arl_get_closecoket_addr	.text
arl_closesocket	.text
arl_SocketCommunication_0	.text
arl_TickCountOps	.text
arl_SocketCommunication_1	.text
arl_SocketCommunicationMappingObject	.text
arl_SomeStringOperations	.text
arl_DecryptSocketSend	.text
arl_SomeMathOperations	.text

Address	Size	Owner	Protect
00010000	00001000		RW
00020000	00001000		RW
0012C000	00001000		RW GUARD
0012D000	00003000		RW GUARD
00130000	00003000		R
00140000	00001000		RW
00150000	00040000		RWE
00190000	00001000		RWE

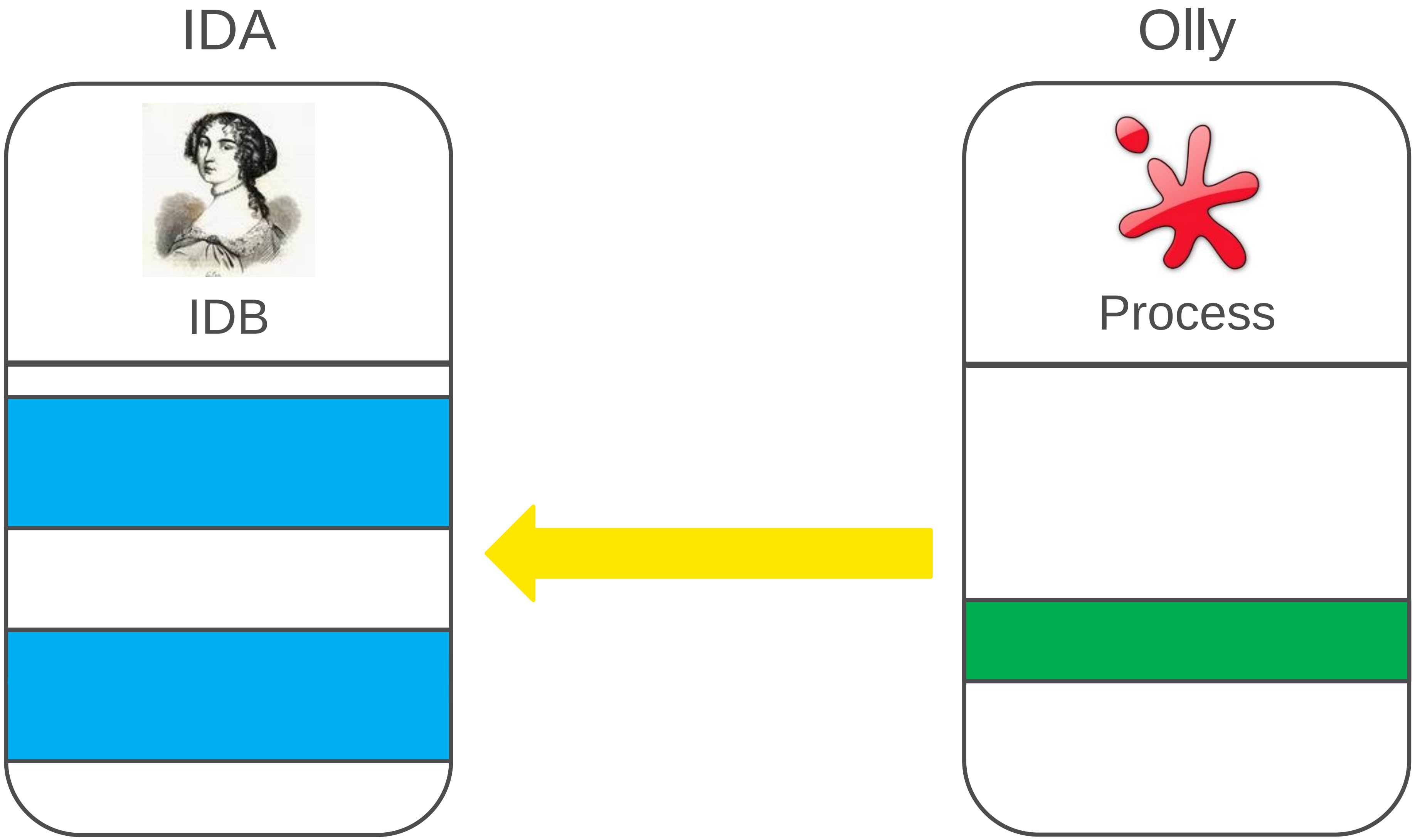
Working Modes: Wipe & Import



Working Modes: Add Segment



Working Modes: Overwrite segment



How Does IDADump Work?

- Performs safe memory read (avoiding PageGuard tricks) of chosen segments
- Performs exported functions scan in external modules. Fixes these functions references in target IDB
- PE header analysis (if valid, helps IDA to decide where data is and where code is)
- Performs post-processing analysis of dumped code

Post-processing Analysis

IDA

IDB

```
call dword_14C8 + ECh
...
jmp dword_14C8 + ABh
```

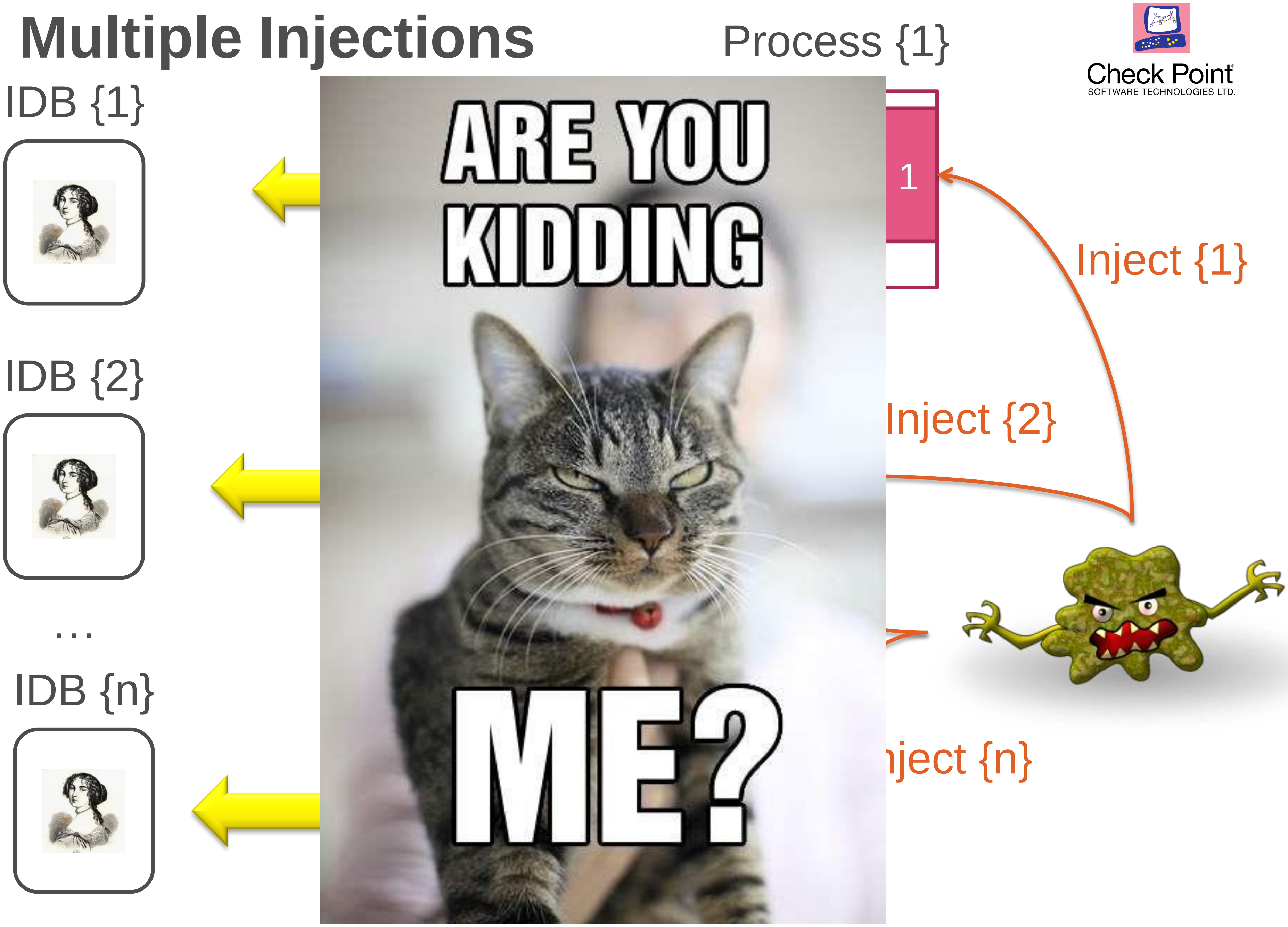
Disassemble code referenced by jmp & call instructions

IDA

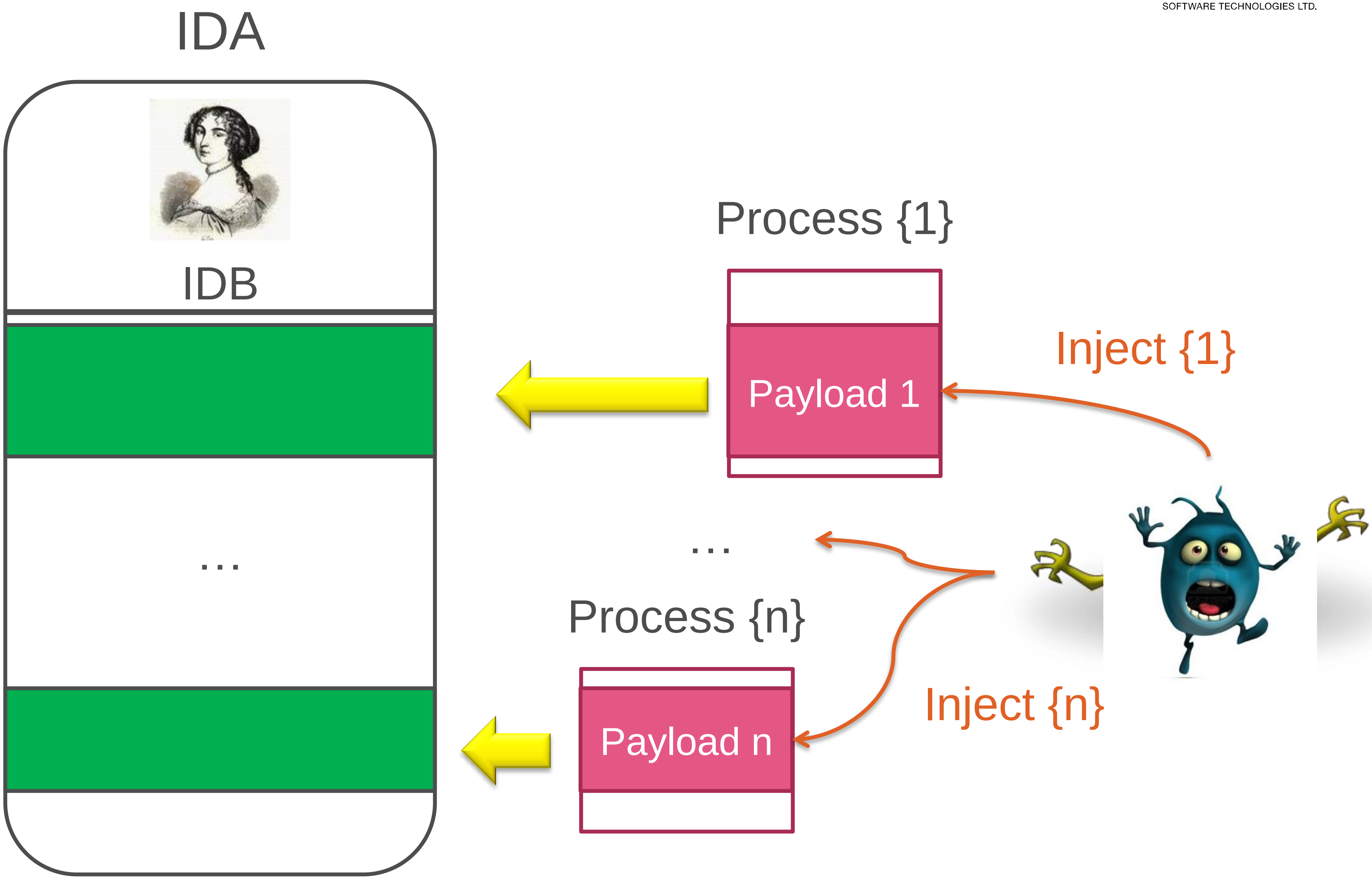
IDB

```
call sub_15B4
...
jmp sub_1573
```

Multiple Injections



Multiple Injections



Multiple Injections

Advantages

- All valuable information is located in one IDB file
- No need to handle multiple IDA instances & look for correlation with other databases
- The same EXTERN section for jmp & call operations
- Sections relations are preserved

Multiple Injections

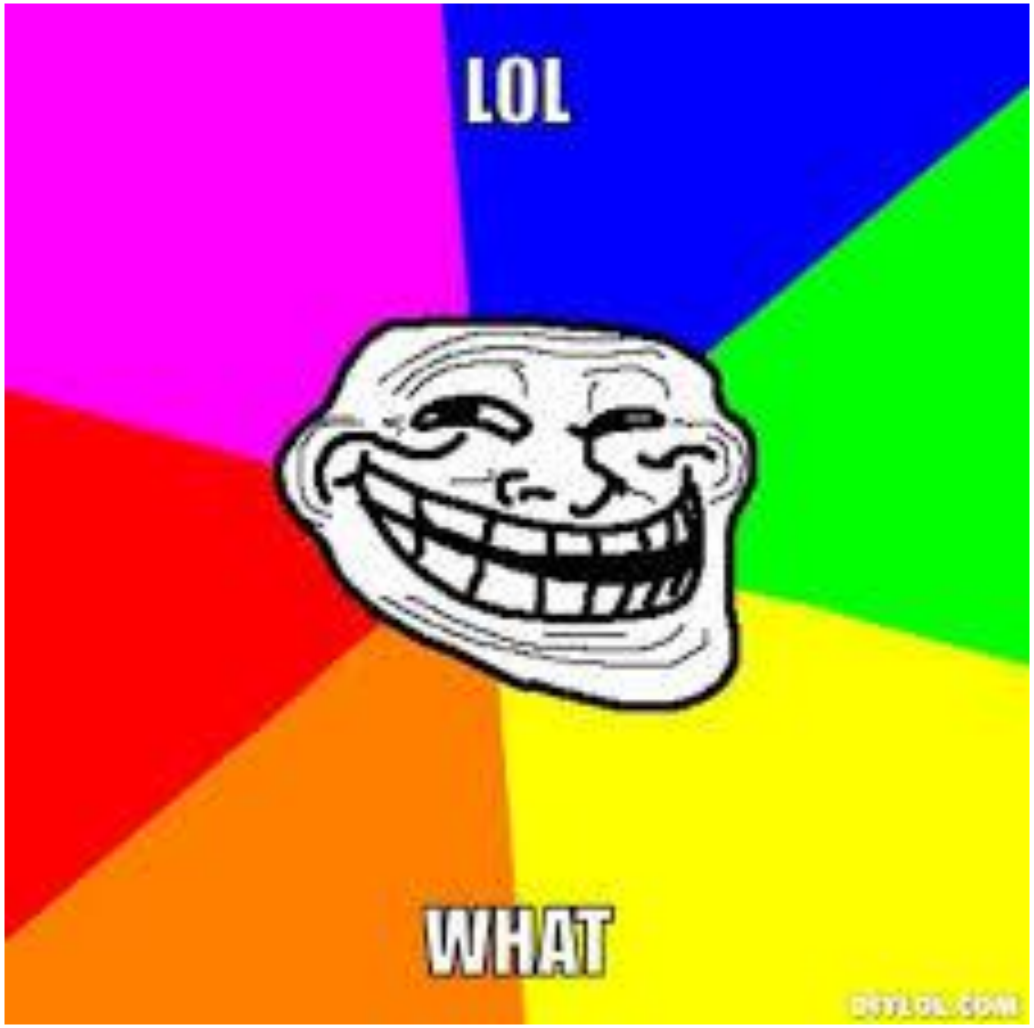
Disadvantages

- Mapping region of memory with address that is already used (Solution: may be solved using segments in IDA)

IMPORTS FIXING

How to Make jmps & calls in Dumped Memory Human-Readable?

```
call near ptr 7529916Dh
…
jmp near ptr 752A2F2Fh
…
call near ptr 752B1DF5h
```

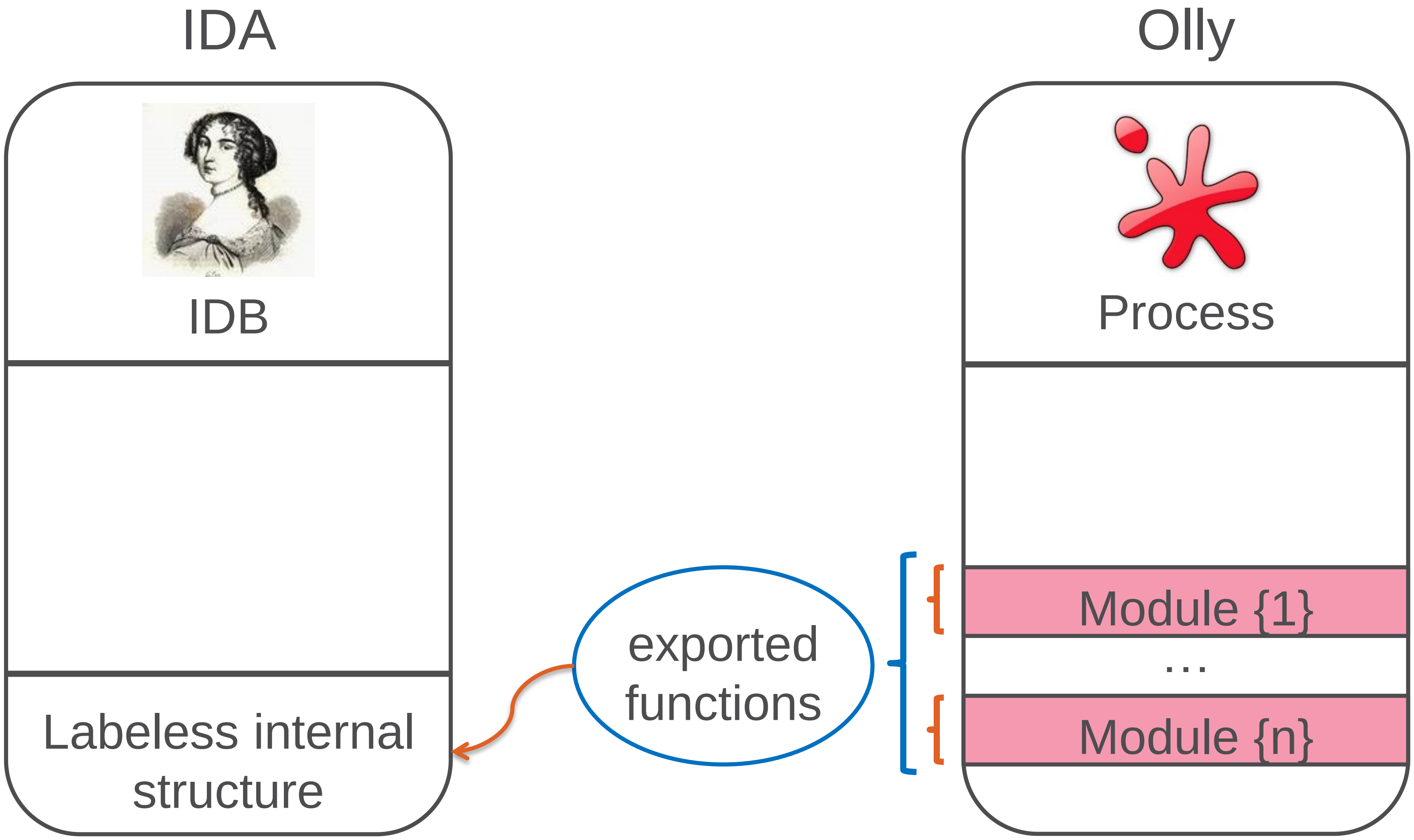


LordPE & ImpRec Solutions

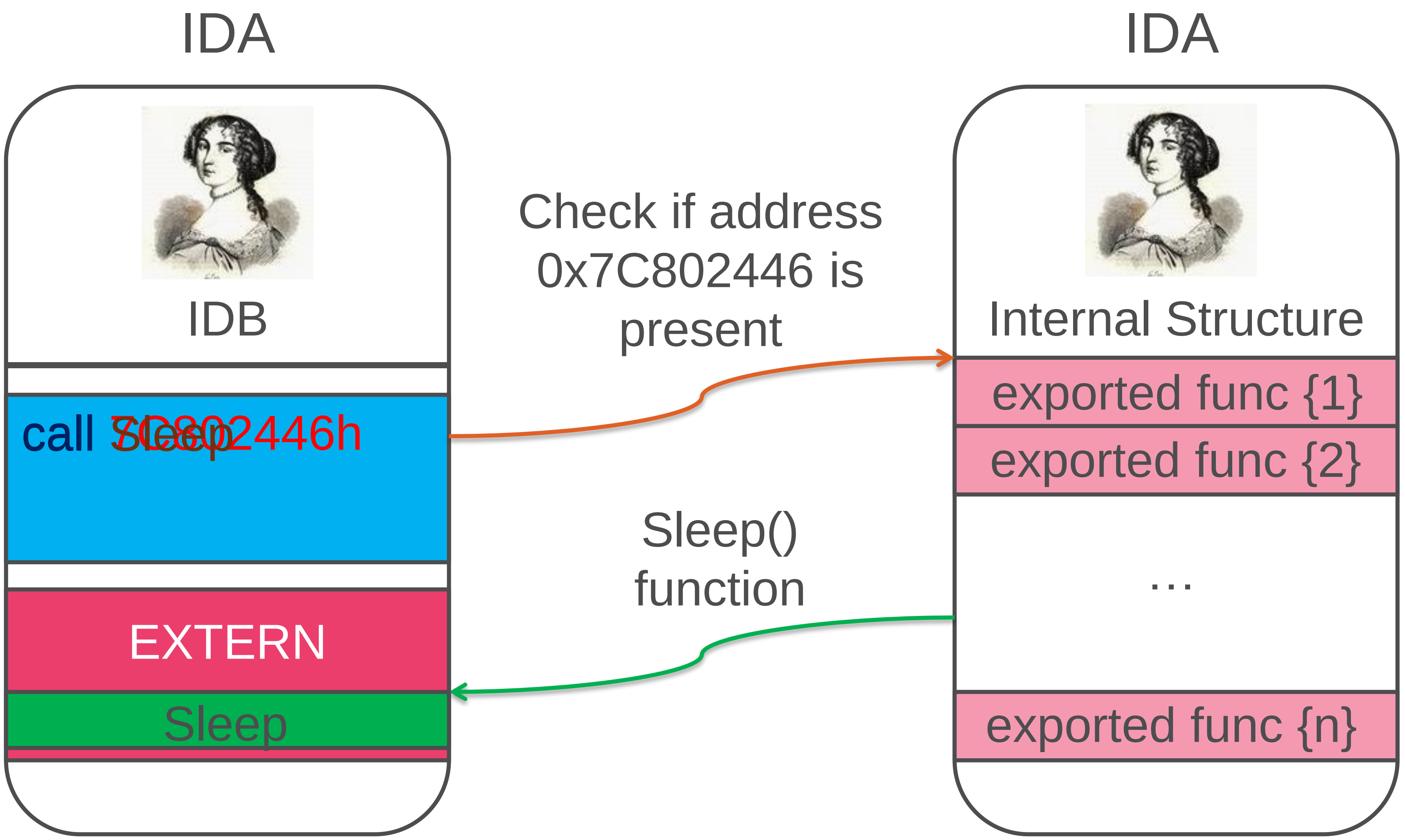
Disadvantages

- Manual search of import table(s) in already running binary
- Manual calculation of relevant offsets and addresses
- Import table may reside in multiple sections
- Time consuming
- Error-prone

Labeless Solution (Getting APIs)



Labeless Solution (Applying APIs)



Labeless Solution

Check Point
SOFTWARE TECHNOLOGIES LTD.

```
call near ptr 7529916Dh
…
jmp near ptr 752A2F2Fh
…
push 752B1DF5h
retn
```

```
call OpenProcess
…
jmp GetCommandLineW
…
push WriteProcessMemory
retn
```

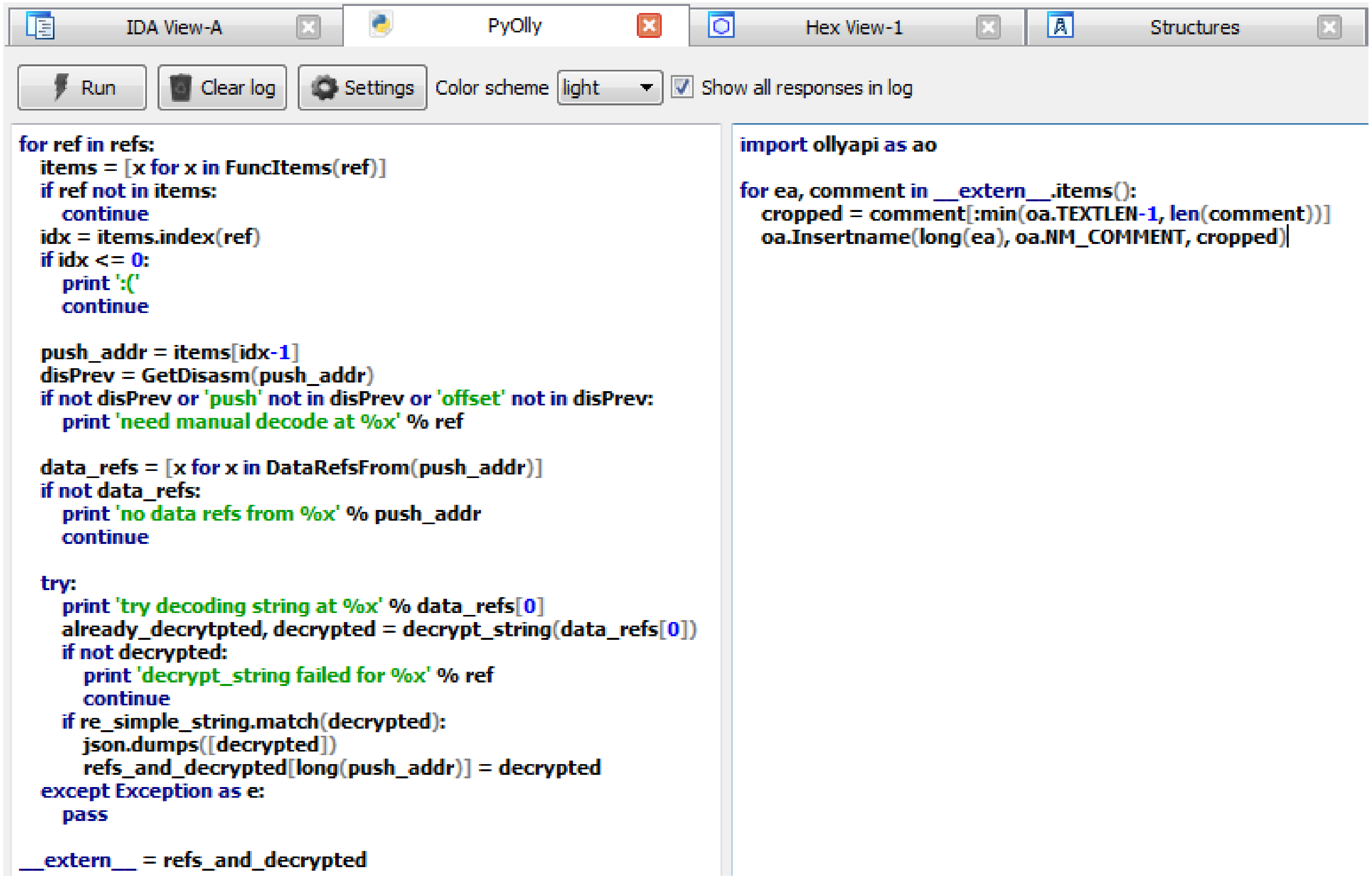
Labeless Solution

Advantages

- No need to perform error-prone tasks
  - Searching for sections with imports
  - Offsets and addresses calculation
- Automatic fixing of functions located in external modules
- Quicker than any existent solution
- Saves precious researcher’s time
- Can be performed on the fly

IDA-OLLY SCRIPTING

IDA-Olly Scripting: IDE View



```
for ref in refs:
    items = [x for x in FuncItems(ref)]
    if ref not in items:
        continue
    idx = items.index(ref)
    if idx <= 0:
        print ':('
        continue

    push_addr = items[idx-1]
    disPrev = GetDisasm(push_addr)
    if not disPrev or 'push' not in disPrev or 'offset' not in disPrev:
        print 'need manual decode at %x' % ref

    data_refs = [x for x in DataRefsFrom(push_addr)]
    if not data_refs:
        print 'no data refs from %x' % push_addr
        continue

    try:
        print 'try decoding string at %x' % data_refs[0]
        already_decrytpted, decrypted = decrypt_string(data_refs[0])
        if not decrypted:
            print 'decrypt_string failed for %x' % ref
            continue
        if re_simple_string.match(decrypted):
            json.dumps([decrypted])
            refs_and_decrypted[long(push_addr)] = decrypted
    except Exception as e:
        pass

__extern__ = refs_and_decrypted
```

```
import ollyapi as ao

for ea, comment in __extern__.items():
    cropped = comment[:min(oa.TEXTLEN-1, len(comment))]
    oa.Insertname(long(ea), oa.NM_COMMENT, cropped)
```

IDA-Olly Scripting: Common Data

Check Point
SOFTWARE TECHNOLOGIES LTD.

IDA

IDA-Olly
Scripting IDE

IDA Script

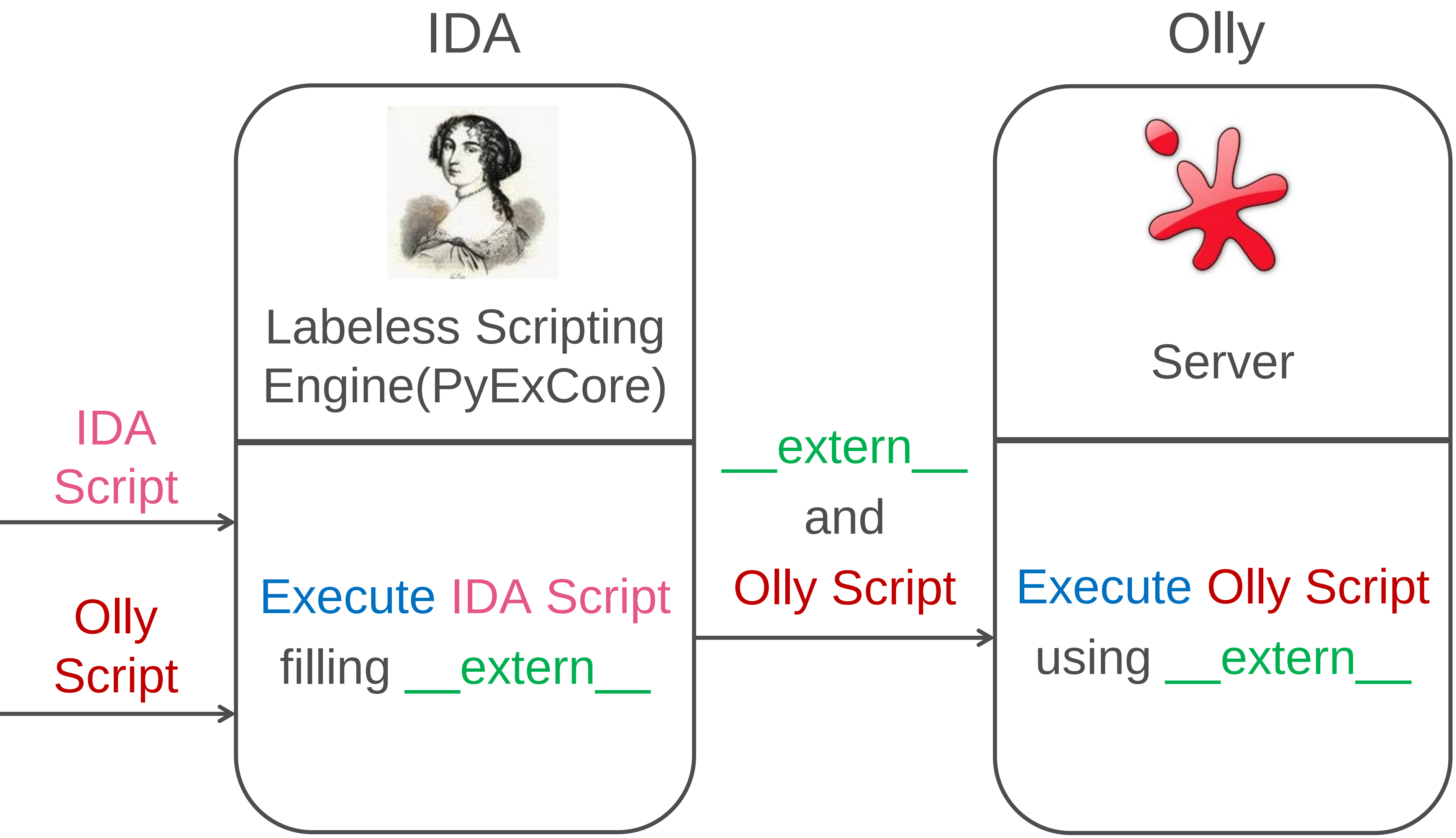
```
tp = dict()
for i in range(100):
    tp[i] = i * i

__extern__ = tp
```

Olly Script

```
for k in __extern__:
    print 'ext[%u] = %u' \
        % (k, __extern__[k])
```

Architecture


IDA
Olly
Labeless Scripting Engine(PyExCore)
Server
IDA Script
Olly Script
Execute IDA Script filling __extern__
__extern__ and Olly Script
Execute Olly Script using __extern__

CONCLUSION

Labeless

- ✓ Automatic tool for on-the-fly labels synchronization between IDA and Olly that supports module rebasing
- ✓ Automatic tool for on-the-fly imports fixing that is faster than any existing tool
- ✓ Convenient tool for IDA-Olly scripting synergy
- ✓ Easy to use memory dumping tool

Areas of Improvement

- WinDBG support
- Support of x64 architecture

Labeless Repository

- Labeless is available as open-source
- Source code is released under Creative Commons BY-NC 4.0
- Link to repository: https://bitbucket.org/a1ex_t/labeless

Check Point
SOFTWARE TECHNOLOGIES LTD.

QUESTIONS?

Aliaksandr Chailytko | Team Leader
Aliaksandr Trafimchuk | Malware Researcher

Contacts:
alexanderc@checkpoint.com

Thanks to:
Stanislav Skuratovich