FORTINET

I opened Pandora's box and it was full of obfuscation

~# whoami

Geri Revay 
Security Researcher at FortiGuard Labs

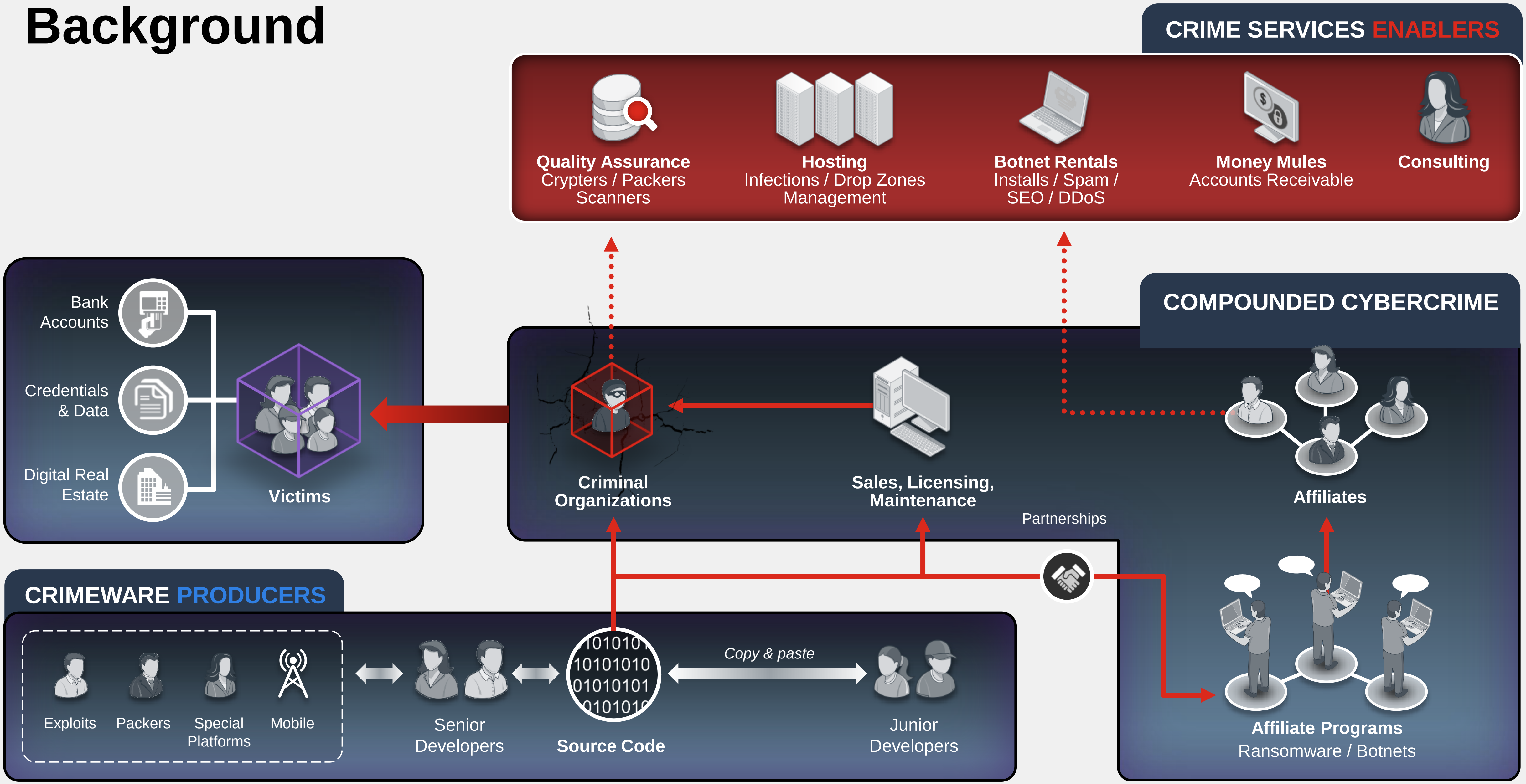
Ethical Hacking
Malware Research
Threat Intelligence
Twitter: @geri_revay

Agenda

- Introduction
- Obfuscation Techniques in Pandora
- Control-Flow Flattening
- Emulation
- The End

Introduction

Background



CRIME SERVICES ENABLERS
Quality Assurance
Crypters / Packers
Scanners
Hosting
Infections / Drop Zones
Management
Botnet Rentals
Installs / Spam /
SEO / DDoS
Money Mules
Accounts Receivable
Consulting
Bank Accounts
Credentials & Data
Digital Real Estate
Victims
Criminal Organizations
Sales, Licensing, Maintenance
Partnerships
COMPOUNDED CYBERCRIME
Affiliates
Affiliate Programs
Ransomware / Botnets
CRIMEWARE PRODUCERS
Exploits
Packers
Special Platforms
Mobile
Senior Developers
101010
10101010
01010101
0101010
Source Code
Copy & paste
Junior Developers

FortiEDR shows how malware is getting better

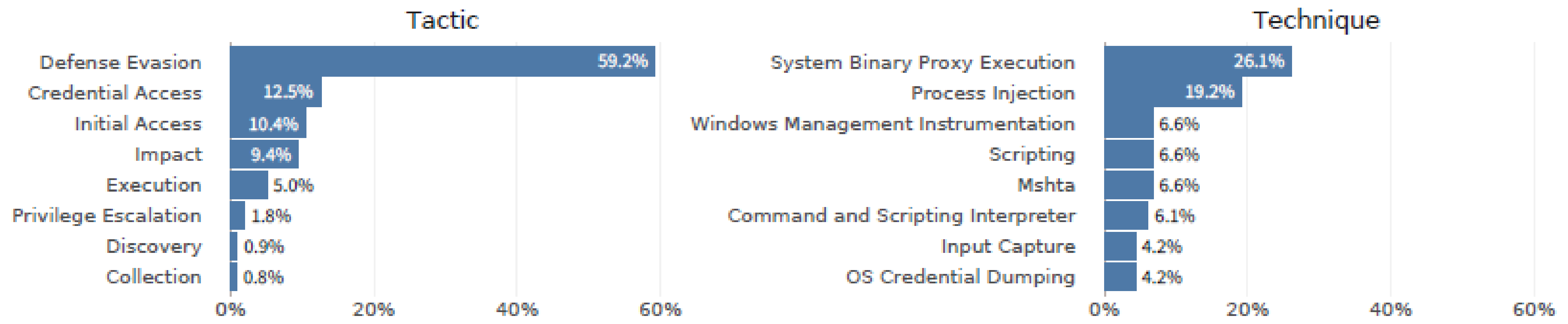
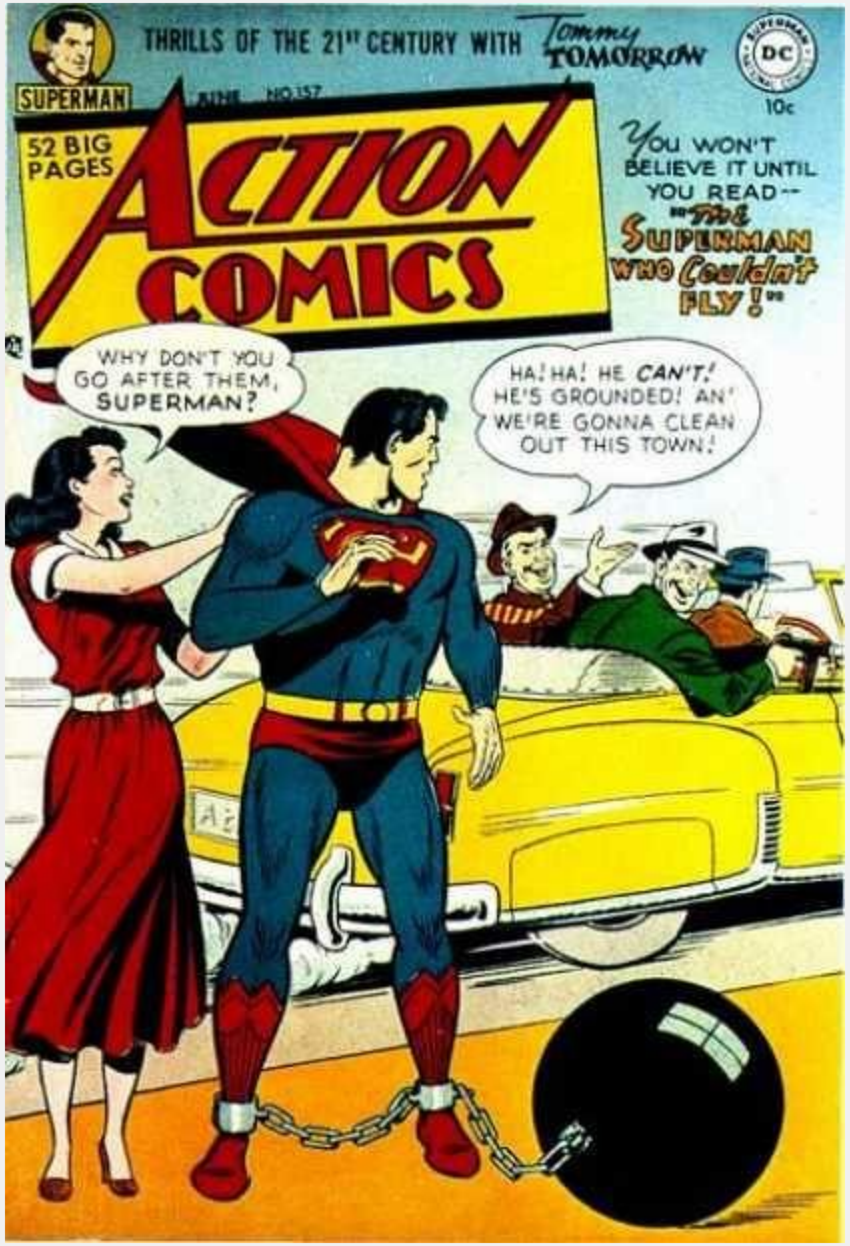


Figure 9 - Top malware tactics and techniques in EDR data for 2022-H1

Why Obfuscation?

https://www.coverbrowser.com/image/action-comics/157-1.jpg



- No Silver Bullet rather a Ball and Chain
- Cheap for the adversary
- Expensive for the analyst
- Different techniques and different levels of obfuscation
- There are obfuscators for most programming languages
- We will focus on C++

Use Case: Pandora Ransomware

- Analysis: https://www.fortinet.com/blog/threat-research/looking-inside-pandoras-box
- Contains everything a modern ransomware should
- Multi-Threading
- Strong Encryption
- Disable AMSI
- Disable Event Logging
- Unlocking files with Restart Manager
- **And all of the world's Evils…**

All of the World’s Evils

Obfuscation Techniques in Pandora

Overview

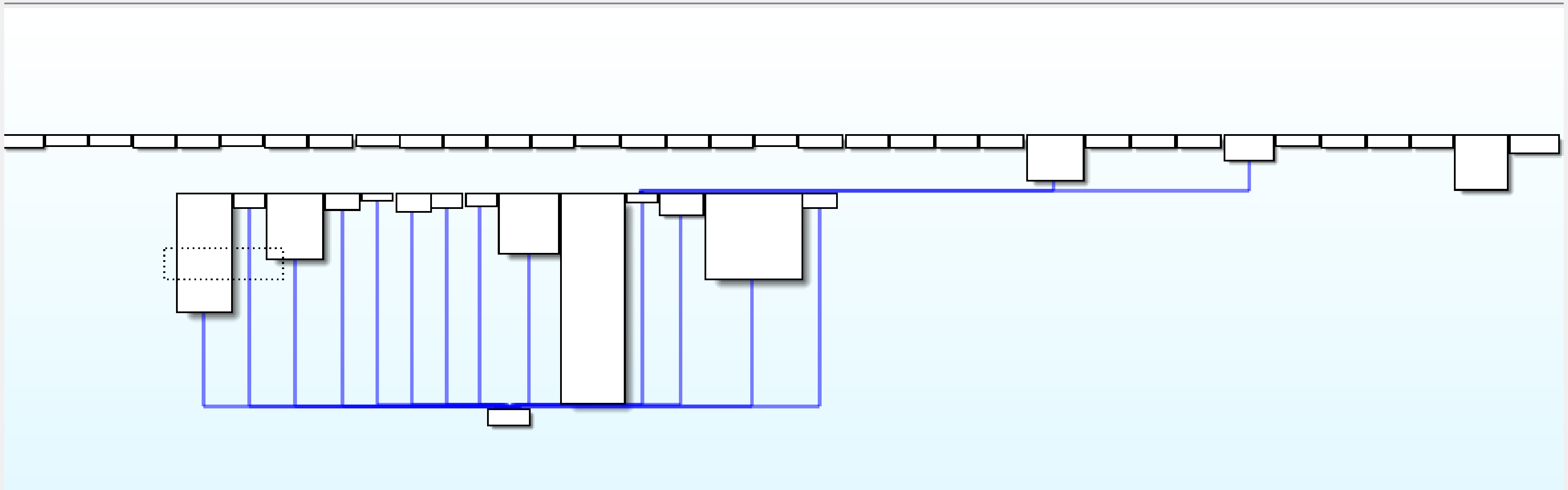
- Packed with custom UPX
- Strings encoding (14 different decoding functions)
- CALL addresses obfuscated with opaque predicates
- JMP addresses obfuscated with opaque predicates
- Control-Flow Flattening
- Windows API call obfuscation

Opaque Predicates for CALL and JMP addresses

```
rax = *(*address_table_base + 0x260BB2E4) + 0xFFFFFFFFAAF7CABC)
```

- Static data that still calculated in runtime
- Obfuscates connections between basic blocks

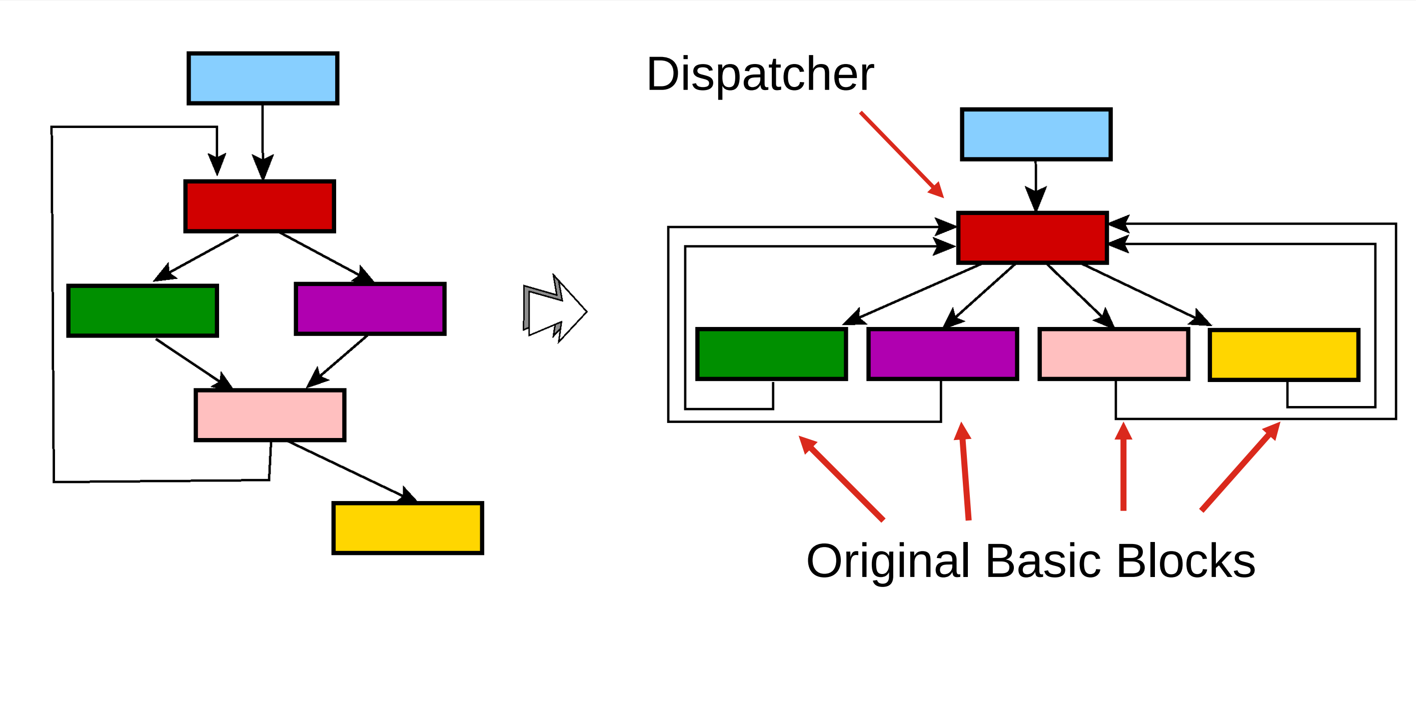
```
pppp:00007FF6B6F9673A mov     rax, cs:qword_7FF6B6FF9AB0
pppp:00007FF6B6F96741 mov     rdi, 0FFFFFFFFAAF7CABCh
pppp:00007FF6B6F96748 mov     rax, [rax+260BB2E4h]
pppp:00007FF6B6F9674F add     rax, rdi
pppp:00007FF6B6F96752 mov     esi, 260BB2E4h
pppp:00007FF6B6F96757 mov     rcx, cs:qword_7FF6B6FF9AB8
pppp:00007FF6B6F9675E add     rcx, rsi
pppp:00007FF6B6F96761 mov     ebp, 260BB8FDh
pppp:00007FF6B6F96766 mov     rdx, cs:qword_7FF6B6FF9AC0
pppp:00007FF6B6F9676D add     rdx, rbp
pppp:00007FF6B6F96770 call    rax
```



Control-Flow Flattening

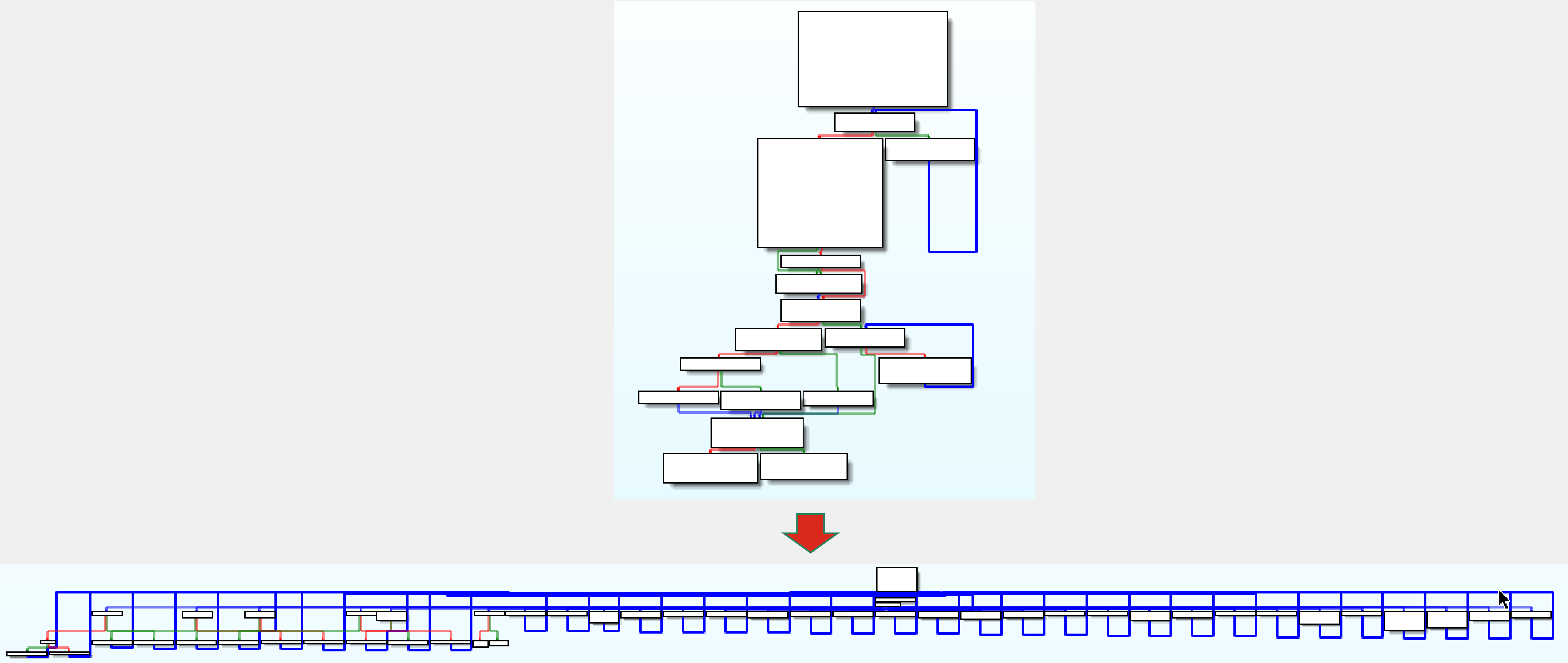
Control-Flow Flattening

- Obfuscation method
- Cheap for developer, expensive for reverse engineer
- Manipulates the control flow of functions
- Original Basic Block: contain the original logic of the function
- Dispatcher: decides which original basic block comes next

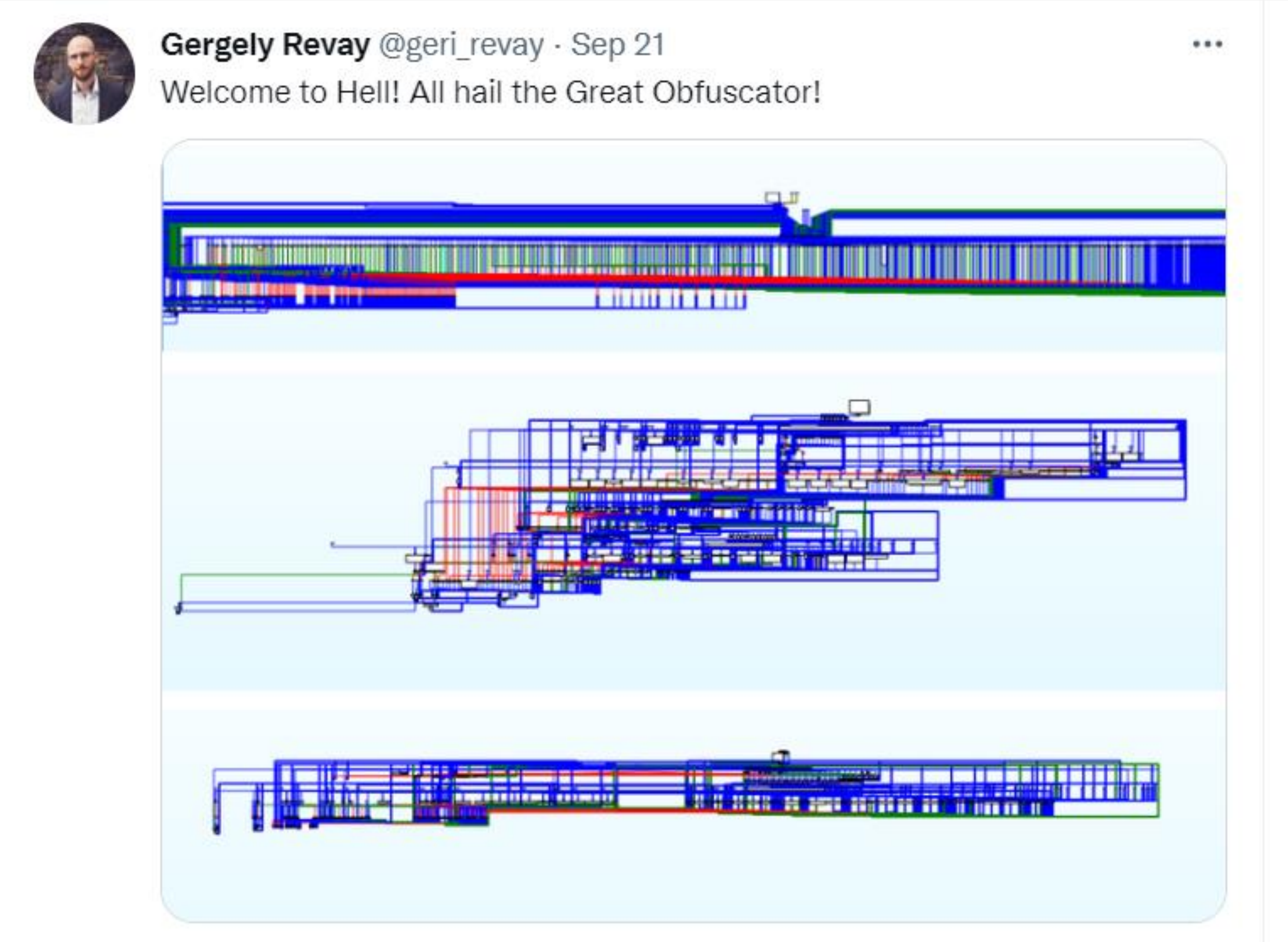


http://tigress.cs.arizona.edu/transformPage/docs/flatten/index.html

Control-Flow Flattening in Real Life



Control-Flow Flattening in Real Life



How to deal with CFF?

How to deal with CFF?

Pack your stuff and run!

How to deal with CFF?

Statically

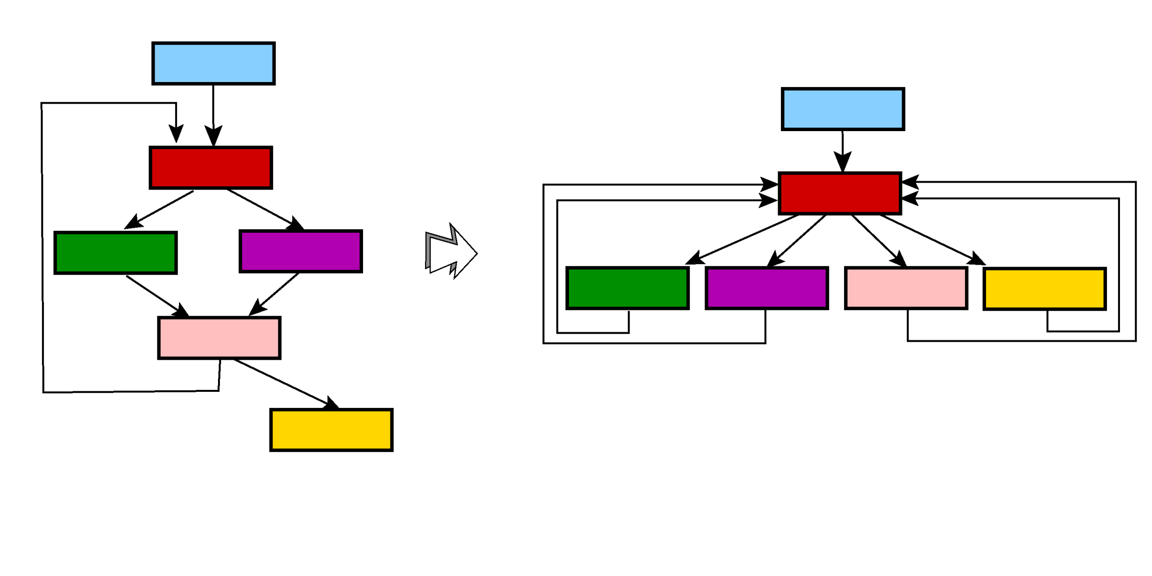
- Restore control-flow in IDA Pro
  - Emulation
  - Symbolic/Concolic Execution
  - Custom IDApython scripts
- .NET: Restore control-flow in MSIL
  - De4dot and other deobfuscators might be able to do it
  - Custom de4dot plugin

Dynamically

- Sandbox detonation
  - Finding IOCs
  - Next stage from memory/file dumps
- Debugging
  - Works but very tedious and slow
  - There might be other Anti-Analysis/Debugging measures in place

Restoring the Control-Flow

- Identify Dispatcher Basic Blocks
- Identify Original Basic Blocks
- Identify State variable
- Map States to OBBs
- Map Next States to OBBs
- Reconstruct code based on recovered paths

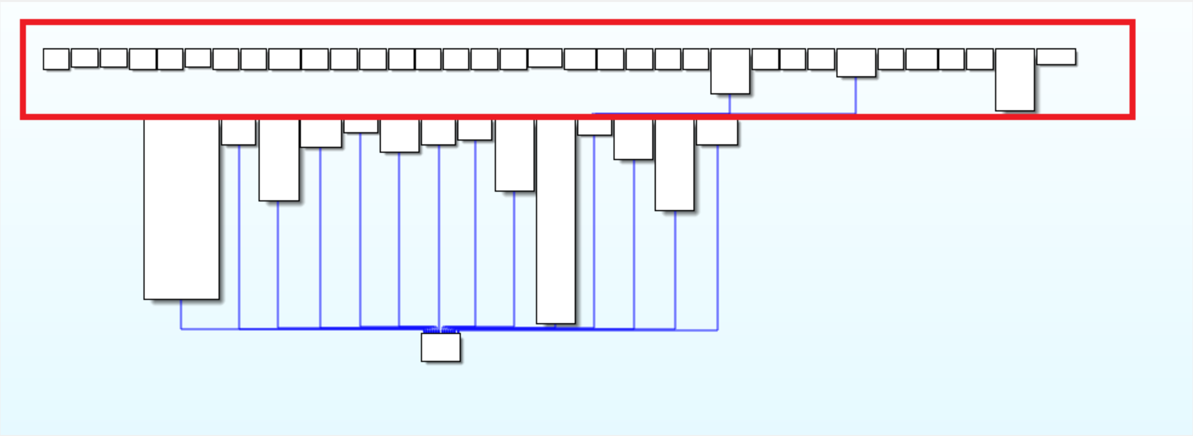


- Added fun in Pandora: Dispatcher is also spread around in multiple

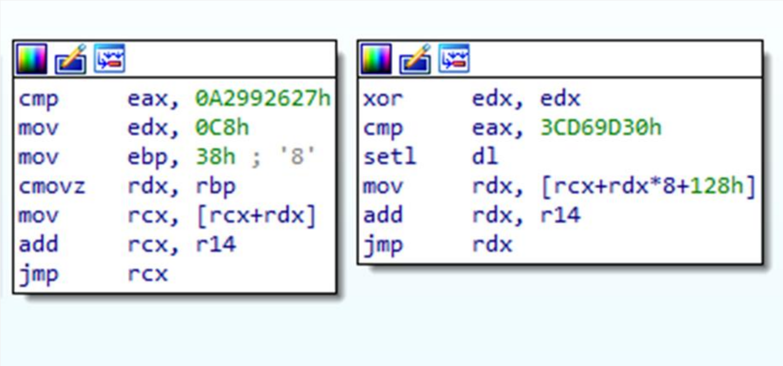
Pandora: Dispatcher

```
cmp     eax, 0A2992627h
mov     edx, 0C8h
mov     ebp, 38h ; '8'
cmovz   rdx, rbp
mov     rcx, [rcx+rdx]
add     rcx, r14
jmp     rcx
```

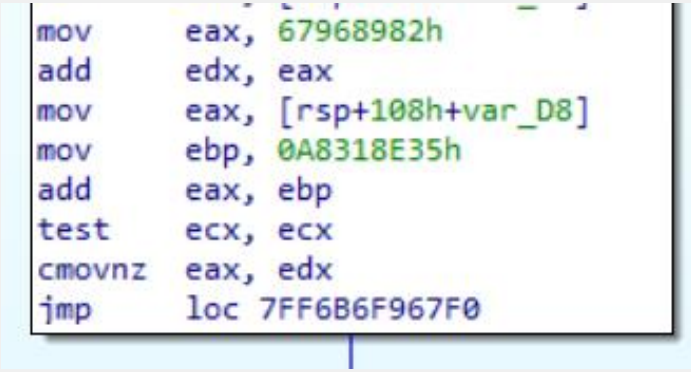
```
xor     edx, edx
cmp     eax, 3CD69D30h
setl    dl
mov     rdx, [rcx+rdx*8+128h]
add     rdx, r14
jmp     rdx
```



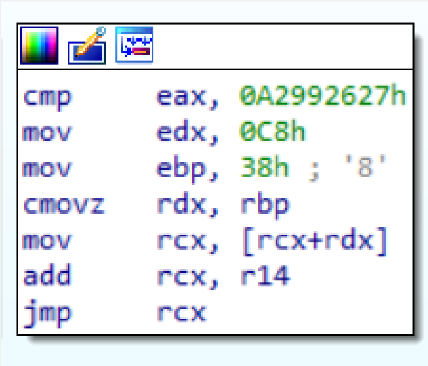
Pandora: Some Heuristics



- Manipulate state variable with cmovX or setlX
- Dispatcher BB starts with cmp or xor
- In case of xor a cmp follows
- The cmp instruction has the state value



Original BB or Code BB ends in relative jump



Dispatcher BB ends in jump to register

Decision in OBBs

```
pppp:00007FF6B6F96B3B mov     ecx, [rsp+108h+var_BC]
pppp:00007FF6B6F96B3F mov     edx, [rsp+108h+var_D8]
pppp:00007FF6B6F96B43 mov     eax, 0EB54C31Bh
pppp:00007FF6B6F96B48 add     edx, eax
pppp:00007FF6B6F96B4A mov     eax, [rsp+108h+var_D8]
pppp:00007FF6B6F96B4E mov     ebp, 0FEFF9F96h
pppp:00007FF6B6F96B53 add     eax, ebp
pppp:00007FF6B6F96B55 test    ecx, 1
pppp:00007FF6B6F96B5B cmovnz  eax, edx
pppp:00007FF6B6F96B5E jmp     loc_7FF6B6F967F0
```

- If OBB would end in a decision, that is moved to another BB
- Some comparison (here test ecx, 1) sets the next state
- These decisions needs to be tracked to learn potential next states

Emulation

Encouragement and cautionary tale

Emulation: the good and evil

- As many complex analysis technique, emulation can be a great help and an enormous time waster
- In practice, the goal is to find the places where it is useful
- Problems with emulation:
  - It does not really run
  - Dependency on other functions
  - Dependency on APIs and libraries

https://www.previewsworld.com/SiteImage/MainImage/STL120308.jpg

Pandora: where emulation worked well

- Opaque Predicates

‘Static’ calculated in run-time

```
pppp:00007FF6B6F9673A mov     rax, cs:qword_7FF6B6FF9AB0
pppp:00007FF6B6F96741 mov     rdi, 0FFFFFFFFAAF7CABCh
pppp:00007FF6B6F96748 mov     rax, [rax+260BB2E4h]
pppp:00007FF6B6F9674F add     rax, rdi
pppp:00007FF6B6F96752 mov     esi, 260BB2E4h
pppp:00007FF6B6F96757 mov     rcx, cs:qword_7FF6B6FF9AB8
pppp:00007FF6B6F9675E add     rcx, rsi
pppp:00007FF6B6F96761 mov     ebp, 260BB8FDh
pppp:00007FF6B6F96766 mov     rdx, cs:qword_7FF6B6FF9AC0
pppp:00007FF6B6F9676D add     rdx, rbp
pppp:00007FF6B6F96770 call    rax
```

Pandora: Opaque Predicates

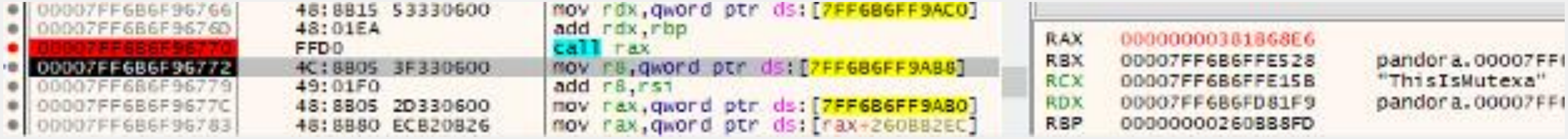
```python
import flare_emu
from ida_funcs import *

def call_hook(address, arguments, functionName, userData):
    print("[+] CALL at 0x{}".format(eh.hexString(address)))
    #check if call target a register
    if eh.analysisHelper.getOpndType(address, 0) != eh.analysisHelper.o_reg:
        return

    operand_name = eh.analysisHelper.getOperand(address, 0)
    operand_value = eh.getRegVal(operand_name)
    print("[+] {} = 0x{:x}".format(operand_name, operand_value))

if __name__ == '__main__':
    ea = get_screen_ea()
    print("[+] Staring emulation")
    eh = flare_emu.EmuHelper()
    function = get_func(ea)
    eh.emulateRange(function.start_ea, callHook=call_hook)
```

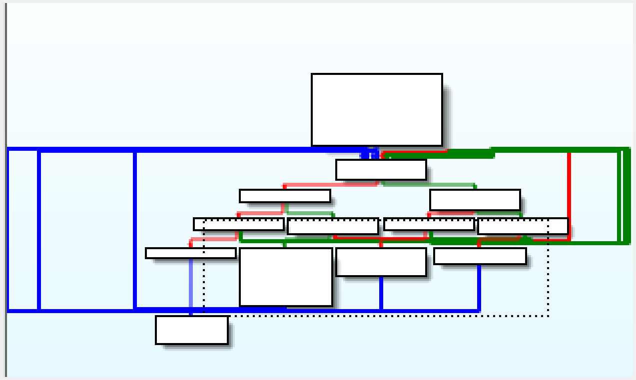
Output

```
[+] Staring emulation
[+] CALL at 0x00007FF6B6F96770
[+] rax = 0x7ff6b6f971e0
[+] CALL at 0x00007FF6B6F96794
[+] rax = 0x7ff6b6fc627c
[+] CALL at 0x00007FF6B6F970E8
[+] rdx = 0x7ff6b6fc629a
```

Pandora: where emulation worked well

- String decryption
- 14 different decryption function, same algorithm different constants
- Iterative process
  - First debugging, later ‘visual inspection’





Pandora: String decryption

```
def call_hook(address, arguments, functionName, userData):
    print("[+] CALL at 0x{}".format(eh.hexString(address)))
    #check if call target a register
    if eh.analysisHelper.getOpndType(address, 0) != eh.analysisHelper.o_reg:
        return

    #comment to call function: args, function addr
    operand_name = eh.analysisHelper.getOperand(address, 0)
    operand_value = eh.getRegVal(operand_name)

    fname = ""
    res = ""
    # check if points to the jump table
    if eh.analysisHelper.getMnem(operand_value).lower() == "jmp":
        fname = eh.analysisHelper.getName(eh.analysisHelper.getOpndValue(operand_value, 0))
        print("[+] API call found: {}".format(fname))
    else:
        fname = eh.analysisHelper.getName(operand_value)
        if "mw_decrypt_str" in fname:
            res = decrypt(arguments, fname)
            print('[+] Decrypted string: 0x{} {}'.format(eh.hexString(address), res))

    # if call target is not a start of a function then turn it to a function
    # 00007FF6B6F947A0
    if idaapi.get_func(operand_value) == None:
        print("[+] Creating function at 0x{:x}".format(operand_value))
        ida_funcs.add_func(operand_value)

```

Pandora: String decryption

```python
def decrypt(argv, fname):
    print("[+] Decrypting ...")
    myEH = flare_emu.EmuHelper()
    myEH.emulateRange(myEH.analysisHelper.getNameAddr(fname), registers = {"arg1":argv[0], "arg2":argv[1],
                        "arg3":argv[2], "arg4":argv[3]})
    return myEH.getEmuString(argv[0])
```

```
00007FF6B6F96766 mov     rdx, cs:qword_7FF6B6FF9AC0
00007FF6B6F9676D add     rdx, rbp
00007FF6B6F96770 call    rax                 ; Decrypted str: 'ThisIsMutexa'
00007FF6B6F96770                             ; rax = 0x7ff6b6f971e0 - mw_decrypt_str
00007FF6B6F96770                             ;      arg0 = 0x7ff6b6ffe15b
00007FF6B6F96770                             ;      arg1 = 0x7ff6b6fd81f9
00007FF6B6F96770                             ;      arg2 = 0x0
00007FF6B6F96770                             ;      arg3 = 0x0
00007FF6B6F96770                             ;      arg4 = 0x0
00007FF6B6F96770                             ;      arg5 = 0x0
00007FF6B6F96770                             ;      arg6 = 0xd54013ae
00007FF6B6F96770                             ;      arg7 = 0x0
100.00% (2032,1228) (3,399) 00005B70 00007FF6B6F96770: main+80 (Synchronized with Hex
```

Output

```
[+] Staring emulation
[+] CALL at 0x00007FF6B6F96770
[+] rax = 0x7ff6b6f971e0
[+] Decrypting ...
[+] Decrypted string: 0x00007FF6B6F96770 bytearray(b'ThisIsMutexa')
[+] CALL at 0x00007FF6B6F96794
[+] rax = 0x7ff6b6fc627c
[+] CALL at 0x00007FF6B6F970E8
[+] rdx = 0x7ff6b6fc629a
```

Pandora: I wasted my time so you don’t have to

- I worked on CFF resolution for pandora
- Problem:
  - Emulation was not able to recover next states from decision OBBs
  - Emulating all function calls is risky
  - Decisions might depend on these calls
  - Pandora has a complex way to calculate the values of next states
- Conclusion
  - In practice (where time is money) it is not worth the time
  - Analysis can be done in a debugger in less time
  - In other malware with less complex obfuscation might worth is

Thanks and Q’n’A

Geri Revay
Security Researcher at FortiGuard Labs

Ethical Hacking
Malware Research
Threat Intelligence
Twitter: @geri_revay

References

https://www.fortinet.com/blog/threat-research/looking-inside-pandoras-box

https://www.fortinet.com/blog/threat-research/Using-emulation-against-anti-reverse-engineering-techniques

https://research.openanalysis.net/pandora/ransomware/malware/unpacking/dumpulator/emulation/2022/03/19/pandora_ransomware.html

https://github.com/mandiant/flare-emu

FORTINET®