# Latrodectus

research.openanalysis.net/latrodectus/config/emulation/2024/09/30/latrodectus.html

OALABS Research September 30, 2024



## Overview

Latrodectus is a RAT that has been linked to the development of IcedID. It provides host information collection capabilities as well as the ability to download and launch additional payloads.

### Sample

`5cecb26a3f33c24b92a0c8f6f5175da0664b21d7c4216a41694e4a4cad233ca8` UnpacMe

### References

## Analysis

The focus of our analysis will be to extract the new AES encrypted strings.

We are going to use the pemulator.py lib to assist with emulation.

```
from pemulator import Emulator

file_path = '/tmp/5cecb26a3f33c24b92a0c8f6f5175da0664b21d7c4216a41694e4a4cad233ca8'
file_data = open(file_path, 'rb').read()
```

```
from unicorn import *
from unicorn.x86_const import *
import struct

emu = Emulator(file_data, mode=64, trace=False)

scratch_start = emu.get_memory_upper_bound() + 0x1000
emu.emu.mem_map(scratch_start, 0x1000)

buff = scratch_start

image_base = emu._pe.OPTIONAL_HEADER.ImageBase
data_addr = 0x01400110A8
dec_fn = 0x014000C1E4

# Move buff into rdx
emu.emu.reg_write(UC_X86_REG_RDX, buff)

# Move enc string into
emu.emu.reg_write(UC_X86_REG_RCX, data_addr)

# address to end emulation
kill_addr = 0xdeadbeef
kill_addr_data = struct.pack('<I', kill_addr)

esp = emu.emu.reg_read(UC_X86_REG_RSP)
emu.emu.mem_write(esp, kill_addr_data)

emu.emu.emu_start(dec_fn, kill_addr)

tmp_string = emu.emu.mem_read(buff, 0x1000).replace(b'\x00',b'')
print(tmp_string)

bytearray(b'POST')
```

```
def decrypt(data_addr):
    emu = Emulator(file_data, mode=64, trace=False)

    scratch_start = emu.get_memory_upper_bound() + 0x1000
    emu.emu.mem_map(scratch_start, 0x1000)

    buff = scratch_start

    image_base = emu._pe.OPTIONAL_HEADER.ImageBase
    #data_addr = 0x01400110A8
    dec_fn = 0x014000C1E4

    # Move buff into rdx
    emu.emu.reg_write(UC_X86_REG_RDX, buff)

    # Move enc string into
    emu.emu.reg_write(UC_X86_REG_RCX, data_addr)

    # address to end emulation
    kill_addr = 0xdeadbeef
    kill_addr_data = struct.pack('<I', kill_addr)

    esp = emu.emu.reg_read(UC_X86_REG_RSP)
    emu.emu.mem_write(esp, kill_addr_data)

    emu.emu.emu_start(dec_fn, kill_addr)

    tmp_string = emu.emu.mem_read(buff, 0x1000).replace(b'\x00',b'')
    return tmp_string

decrypt(0x01400110A8)

bytearray(b'POST')
```

```
import re

tmp_emu = Emulator(file_data, mode=64, trace=False)

# Find all refs to the enc strings
#
# .text:0000000140005E19 48 8D 94 24 80 00 00 00                lea     rdx,
[rsp+188h+var_108]
# .text:0000000140005E21 48 8D 0D 80 B2 00 00                   lea     rcx,
word_1400110A8
# .text:0000000140005E28 E8 B7 63 00 00                         call
latro_str_decryption_main
#
# .text:0000000140007A83 48 8D 54 24 40                         lea     rdx,
[rsp+88h+var_48]
# .text:0000000140007A88 48 8D 0D 91 A1 00 00                   lea     rcx,
word_140011C20
# .text:0000000140007A8F E8 50 47 00 00                         call
latro_str_decryption_main

egg = rb'\x48\x8D.{3,6}\x48\x8D\x0D(....)\xE8'

for match in re.finditer(egg, file_data):
    match_len = len(match.group())
    if match_len == 13:
        # Smaller offset
        image_base = tmp_emu._pe.OPTIONAL_HEADER.ImageBase
        start_address = tmp_emu._pe.get_rva_from_offset(match.start()) + image_base
        data_addr = struct.unpack('<I', match.group(1))[0] + 12 + start_address
    else:
        image_base = tmp_emu._pe.OPTIONAL_HEADER.ImageBase
        start_address = tmp_emu._pe.get_rva_from_offset(match.start()) + image_base
        data_addr = struct.unpack('<I', match.group(1))[0] + 15 + start_address
    #print(f"{hex(start_address)}: STRING: {hex(data_addr)}")
    try:
        tmp_string = decrypt(data_addr)
        if tmp_string.isascii():
            out = tmp_string.decode('utf-8')
            print(f"{hex(data_addr)}: {out}")
        else:
            print(f"{hex(data_addr)}: -------> {tmp_string}")
    except:
        print(f"{hex(data_addr)}: ***ERROR - no string")
```

```
0x1400109f8: {
0x140010a10: "pid":
0x140010a30: "%d",
0x140010a50: "proc":
0x140010a70: "%s",
0x140010a90: "subproc": [
0x140010ab8: ]
0x140010ad0: }
0x140010c00: &desklinks=[
0x140010c28: *.*
0x140010c48: "%s"
0x140010c68: ]
0x140010ae8: &proclist=[
0x140010b10: {
0x140010b28: "pid":
0x140010b48: "%d",
0x140010b68: "proc":
0x140010b88: "%s",
0x140010ba8: "subproc": [
0x140010bd0: ]
0x140010be8: }
0x140010000: /c ipconfig /all
0x140010070: C:\Windows\System32\cmd.exe
0x140010038: /c systeminfo
0x1400100c0: C:\Windows\System32\cmd.exe
0x140010110: /c nltest /domain_trusts
0x140010190: C:\Windows\System32\cmd.exe
0x1400101e0: /c nltest /domain_trusts /all_trusts
0x140010240: C:\Windows\System32\cmd.exe
0x140010290: /c net view /all /domain
0x140010300: C:\Windows\System32\cmd.exe
0x140010158: /c net view /all
0x140010350: C:\Windows\System32\cmd.exe
0x1400103a0: /c net group "Domain Admins" /domain
0x140010400: C:\Windows\System32\cmd.exe
0x140010450: /Node:localhost /Namespace:\\root\SecurityCenter2 Path AntiVirusProduct
Get * /Format:List
0x140010520: C:\Windows\System32\wbem\wmic.exe
0x140010580: /c net config workstation
0x1400105d0: C:\Windows\System32\cmd.exe
0x140010620: /c wmic.exe /node:localhost /namespace:\\root\SecurityCenter2 path
AntiVirusProduct Get DisplayName | findstr /V /B /C:displayName || echo No Antivirus
installed
0x140010780: C:\Windows\System32\cmd.exe
0x1400107d0: /c whoami /groups
0x140010810: C:\Windows\System32\cmd.exe
0x1400102d8: &ipconfig=
0x140010860: &systeminfo=
0x140010888: &domain_trusts=
0x1400108b0: &domain_trusts_all=
0x1400108e0: &net_view_all_domain=
0x140010910: &net_view_all=
```

```
0x140010938: &net_group=
0x140010960: &wmic=
0x140010980: &net_config_ws=
0x1400109a8: &net_wmic_av=
0x1400109d0: &whoami_group=
0x140010cb0: Custom_update
0x140010c80: Update_%x
0x140010ce8: .dll
0x140010d08: .exe
0x140010d28: Updater
0x140010d50: "%s"
0x140010d70:
0x140010d88: rundll32.exe
0x140010db8: "%s", %s %s
0x140010df0: runnung
0x140010e18: :wtfbbq
0x140010f98: front
0x140010fb8: /files/
0x140010fd8: .exe
0x140010e70: %d
0x140010e90: %s%s
0x140010eb0: files/bp.dat
0x140010ed8: %s\%d.dll
0x140010f08: %d.dat
0x140010f30: %s\%s
0x140010f58: init -zzzz="%s\%s"
0x140010e48: %s/%s
0x140010ff8: Wiski
0x140011018: .exe
0x140011120: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Tob 1.1)
0x1400111b0: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Tob 1.1)
0x140011060: Content-Type: application/x-www-form-urlencoded
0x1400110a8: POST
0x1400110c8: GET
0x140011250: CLEARURL
0x140011270: URLS
0x140011290: COMMAND
0x1400112b0: ERROR
0x1400112d0: 2sDbsEUXvhgLOO4Irt8AF6el3jJ0M1MowXyao00Nn6ZUjtjXwb
0x140011320:
counter=%d&type=%d&guid=%s&os=%d&arch=%d&username=%s&group=%lu&ver=%d.%d&up=%d&directi

0x1400113a0:
counter=%d&type=%d&guid=%s&os=%d&arch=%d&username=%s&group=%lu&ver=%d.%d&up=%d&directi

0x140011420:
counter=%d&type=%d&guid=%s&os=%d&arch=%d&username=%s&group=%lu&ver=%d.%d&up=%d&directi

0x140011498: &dpost=[{"data":"
0x1400114c0: "}]
0x140011b30: ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
0x140011be8: https://minrezviko.com/test/
```

```
0x140011c20: https://agrahusrat.com/test/
0x140011748: %s%d.dll
0x1400118e0: %s%d.exe
0x1400117d0: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Tob 1.1)
0x140011860: <html>
0x140011728: <!DOCTYPE
0x140011508: AppData
0x140011530: Desktop
0x140011558: Startup
0x140011580: Personal
0x1400115a8: Local AppData
0x140011620: Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
0x140011a18: &mac=
0x140011a38: %02x
0x140011a98: ;
0x140011ab0: &computername=%s
0x140011ad8: &domain=%s
0x1400115e0: \Registry\Machine\
0x140011bb8: %04X%04X%04X%04X%08X%04X
0x140011b00: \*.dll
0x14001189c: ------->
bytearray(b'\xd9\x80I\x9c\xf8\xa5\xbbCr\xb9B\xd8k\xc0j\xfa3\x9d\xdcR3\xc8\xe9\xc8\x04\
A\x9b\xd8^f\x86\x12\x17\x89W\xefB]\x9b\xb9\x81\x12\xbaI{\xe0q\x1b\xba\x08Bsl=\x16P"\xa

0x1400116c0: C:\WINDOWS\SYSTEM32\rundll32.exe %s,%s
0x140011770: C:\WINDOWS\SYSTEM32\rundll32.exe %s
0x1400118a0: 12345
0x1400118c0: &stiller=
0x140011880: 12345
0x140011960: TimeTrigger
0x140011990: PT0H%02dM
0x1400119c0: %04d-%02d-%02dT%02d:%02d:%02d
0x140011a78: PT0S
0x140011c58: \update_data.dat
0x140011ca0: URLS
0x140011cc0: URLS|%d|%s


hex(0x0140007A8F + 0xa191)

'0x140011c20'
```