vmware by Broadcom VMware Security Blog

VMware Blogs Communities Tech Zone

Featured Categories VMware Security Get A Demo

Threat Analysis Unit

# Carbon Black Threat Research Technical Analysis: Petya / NotPetya Ransomware

June 28, 2017 / 24 min read

Share on:

On June 27, public announcements were made about a large-scale campaign of ransomware attacks across Europe. The ransomware impacted notable industries such as Maersk, the world's largest container shipping company. The initial infection vector appears to be the exploitation of a Ukrainian tax software called MEDoc. The sample also spreads on the internal network via exploitation of the EternalBlue SMB vulnerability, PsExec, WMI, and Admin$ shares.

## Technical Analysis

The metadata for the analyzed sample is noted below:

**File Size :** `362,360`

**MD5:** `71b6a493388e7d0b40c83ce903bc6b04`

**SHA1:** `34f917aaba5684fbe56d3c57d48ef2a1aa7cf06d`

**SHA256:** `027cc450ef5f8c5f653329641ec1fed91f694e0d229928963b30f6b0d7d3a745`

**Fuzzy:**
`6144:y/Bt80VmNTBo/x95ZjAetGDN3VFNq7pC+9OqFoK30b3ni5rdQY/CdUOs2:y/X4NTS/x9jNG+w+9OqFoK323qdQYKUG`

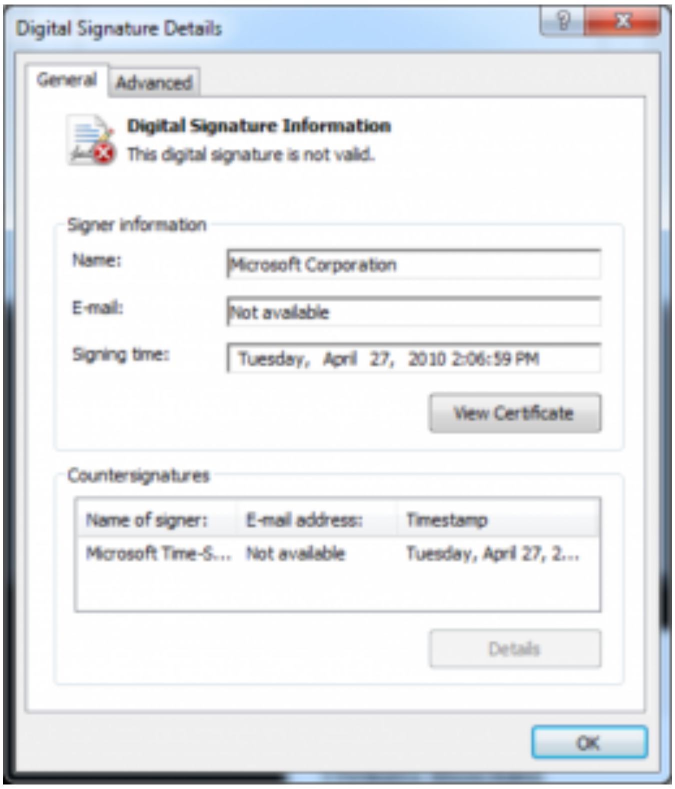
**Magic:** `PE32 executable for MS Windows (DLL) (console) Intel 80386 32-bit`

**Import Hash:** `52dd60b5f3c9e2f17c2e303e8c8d4eab`

**Compiled Time:** `Sun Jun 18 07:14:36 2017 UTC`

**PE Sections (5):**
```
Name        Size        MD5

   .text      48,640     c5bd3bb710ae377938b17980692b785b

   .rdata     34,304     46418e52b546c1f696eb8a524f18c56e

   .data      20,992     5216f0c62d1fd41b1d558e129e18d0fe

   .rsrc      247,808    f07e68575f50a62382d99e182baa05d5

   .reloc     3,584      c5d1d4cdade7dcfbe14ec10dcf66cfb1

 + 0x57000    6,008      da2b0b17905e8afae0eaca35e831be9e
(Authenticode Signature)
```

This sample was compiled recently (18 June 2017 at 07:14:36 UTC) and contains an expired digital signature signed by Microsoft. This sample utilizes SysInternals PsExec v1.98 for remote execution, with the PsExec digital signature copied and applied to the malware.



**Digital Signature Information**
This digital signature is not valid.

Signer information

Name: Microsoft Corporation
E-mail: Not available
Signing time: Tuesday, April 27, 2010 2:06:59 PM

Countersignatures

| Name of signer: | E-mail address: | Timestamp |
|---|---|---|
| Microsoft Time-S... | Not available | Tuesday, April 27, 2... |

SysInternals Digital Signature Found on Malware

Initial analysis suggests that the malware is a new variant of the Petya family of ransomware. This recent sample follows the encryption and ransom note functionality seen in Petya samples. However, further analysis suggests otherwise, leading to industry concerns of actual attribution and naming, with the colloquial names given of “NotPetya” and “Schrodinger’s Petya.” To understand this, let’s explore some of the more technical components of the ransomware.

The malware itself, a Windows DLL library, is typically saved to the local Windows folder as perfc.dat if it runs with SeDebugPrivilege permission. If it does not have this privilege, it saves perfc.dat to the All Users application data directory (CSIDL_COMMON_APPDATA) which is resolved by Windows to either C:\ProgramData or “C:\Documents and Settings\All Users\Application Data”.

The malware must be executed with a direct call to its export of ordinal 1, seen as “#1” in the command line. There are various reasons why malware tends to use this style of execution, with the most common being to prevent execution within a sandbox. A call to the standard DLL export function simply exits.

## Ransomware Data Encryption

Upon execution, the malware will begin to perform privilege checking to determine what permissions the running account has, primarily checking for the SeDebugPrivilege permission, as detailed later in *User Rights Checking and Malware Logic*. Based upon these privileges the malware will encrypt the victim system in one of two methods.

If the running user has the SeDebugPrivilege permission, the malware will assume it has administrative privileges, it will then attempt to encrypt the drive using the known Petya code. This is performed by overwriting the Master Boot Record (MBR) of the hard drive (PhysicalDrive0) with malicious code that displays a fake “chkdsk.exe” partition repair screen as it encrypts the hard drive, shown below. This is a technique that is known to exist in Petya, using nearly identical text and code. This is the reason why the malware was initially considered to be Petya upon dynamic analysis, although static analysis showed vastly different code for the remainder of the malware’s functionality.

False CHKDSK Output from Malware MBR Code

MBR-based Ransom Note

Alternatively, if the user is not running with administrative privileges, as determined by a lack of SeDebugPrivilege, the malware will use a user-space encryption routine, which is a different routine from the one detailed above.

After the malware is finished obtaining privileges on the system, it looks for specific processes running on the system.  The malware authors use a proprietary XOR encryption routine to mask what process names they are looking for.  Carbon Black created a brute force routine to determine what those process names were:

| **Encrypted Value** | **AV Filename** | **AV Product** |
|---|---|---|
| 0x2E214B44 | avp.exe | Kaspersky |
| 0x651B3005 | NS.exe | Norton Security |
| 0x6403527E | ccSvcHst.exe | Symantec |

A global variable is modified if these process names are found running on the system.  This global variable affects how the malware runs on the system.  If none of these AV processes are found on the system and the malware obtains the required privileges, the credential stealer is dropped and executed.  The presence of these AV's on the system also affect if the malware runs PsExec, writes to the MBR, the type of encryption used, and if the anti-forensic log clearing is executed.

Process Name Hashing Function

Depending on the return values from this routine and the values SeDebugPrivilege check, if the process context does not have SeDebugPrivilege permissions then the variant will employ a technique common in ransomware variants. Where files' data is loaded by using the FindFile and MapViewofFile APIs before encrypting the file's data. The file is encrypted in memory using AES encryption and then written to disk when the FlushViewOfFile is called.

After the encryption routine completes, the sample will attempt to remove forensic artifacts of its activities by utilizing cmd.exe to run the command listed in the table below. This is done after performing additional operating system checks which include ensuring that the system is not version 5.1/5.2 (Windows XP) or 6.1 (Windows 7).

```
wevtutil cl Setup & wevtutil cl System & wevtutil cl Security & wevtutil
cl Application & fsutil usn deletejournal /D %c:”
```

Anti-forensics Commands

The command will use the wevtutil command to clear (cl) the logs for the Setup, System, Security, and Application event logs. It then uses the fsutil command to delete (/D) the USN change journal from the current drive (noted as %c character type).

Before the sample exits it will force a reboot of the infected system using one of three methods.  The method chosen is dependent upon the context of the process privileges, which are outlined in the table below. The sample will schedule a reboot early in its execution but, depending on the user privileges, will attempt to force a reboot earlier.

| Condition | Reboot Method |
|---|---|
| If SeShutDown<br>and SeDebug | The malware will initiate a shutdown using the InitiateSystemShutdown() Api |
| If only SeDebug | The malware will initiate a shutdown using a call to NtDll.NtRaiseHardError() API |
| If not SeDebug | The malware will rely upon the previously created scheduled task (sctasks.exe) to restart (shutdown.exe) the system. |

System Shutdown Conditions

## Delivery and Exploitation

While the initial attack vector of a patient zero is currently unknown, what is known is that the malware is propagated across local environments using several methods. Like other recent malware the ransomware utilizes the highly effective EternalBlue exploit for Windows SMB vulnerabilities to copy itself to other systems and execute, which is detailed later in this post.

In addition to the SMB exploit propagation method, the malware also attempts to establish default administrative network shares (Admin$) with a call to WNetAddConnection2() using a null username and password. By using null for these two values, the network connection is made using the current user’s credentials, which directly affect organizations with shared local administrator accounts. Additionally, if the malware was somehow executed by a domain administrator, these connections would be made even without a shared local administrator account. An example of this additional lateral movement attempt is below:

Lateral Movement to Admin Shares

If a connection is made to the remote system, the malware will create a copy of itself directly on the other system using standard CreateFile() and WriteFile() API routines. The malware also has the ability to extract an embedded and compressed resource (resource 3) containing the SysInternals PsExec.exe utility. The psexec.exe file is written to disk as dllhost.dat.  This utility is then used to execute this file on the remote system.

PsExec Remote Execution Command Line Construction

Failing that, the malware will then leverage the existing Windows WMI utility to execute the file on the remote system, which is displayed in the image below:

WMIC Remote Execution Command Line Construction

As noted earlier, upon initial execution the sample will check several system privileges to determine the appropriate execution path.  If the process has SeDebugPrivilege it will, among other things, access the physical drive (highlighted in red below) and then initiate a scan that will enumerate other systems on the domain (area highlighted in green in the image below).  The area highlighted in red below shows where a function (labeled here as filename_check) is called, where the sample will check to see if the file C:\Windows\perfc is present on the system.

If present, the process will exit, which is leading many to call this a “kill switch” in the ransomware.  This was most likely done to ensure that ransomware was not installed and executed twice on a system, much in the same way a mutex or a file lock (Linux based), is used to prevent multiple instances from running on a system.  Code responsible for this process is shown below.

Reinfection Check

The second function (labeled Check_access_and_physicaldisk_Modify) is responsible for attempting to access the physical disk, and then gather drive geometry about the disk itself.  The sample will then make modifications to the MBR.  After the alterations have completed the sample will remove forensic artifacts in the manner detailed in Table 1 above, and then shut down as described in Table 2 above.

If file already exists, exit the process

Privilege and Drive Access Checks Prior to Scanning

The function (highlighted in green in the image above) will gather system information and then enumerate the other systems on the domain.  This enumeration is highlighted in the red box below.

Grabbing Local System Info and Enumerating Systems

## SMB propagation

As previously referenced, this sample is also capable of spreading through SMB kernel exploitation (MS17-010, commonly referred to as EternalBlue).  The function displayed in the image below shows the sample building the SMB header, which will be used in the exploitation process.

Building Exploit's SMB Header

The sample will then decode several strings used in the SMB phase of the  attack (which are displayed in a later table).  The area highlighted in red below shows the call to create the SMB packet.  The area, highlighted in green, shows where the sample will send SMB packets and then wait for a response from the targeted system.

The table below shows the strings that were decoded in the previous image.  These strings were encoded using a simple XOR key of 0x72.

PC NETWORK PROGRAM 1.0

LANMAN1.0

Windows for Workgroups 3.1a

LM1.2X002

LANMAN2.1

NT LM 0.12

Decoded Strings used in SMB Exploitation

If the sample finds an exploitable system, it will perform in the same manner that existing ransomware acted by passing a shellcode payload to the target system.  The function highlighted in the image below shows where the shellcode is first decoded with a simple XOR value of 0xCC.

Initial De-obfuscation of Shellcode Payload

The image below shows a portion of the decoded shellcode, which contains additional layers of obfuscation and is still being analyzed.  Resource 4 (once inflated and decoded using a simple XOR of 0x86) is additional shell code that is used in the SMB exploitation process.

Continued De-obfuscation of Shellcode Payload

## User Rights Checking and Effect on Malware Logic

This sample looks for three different types of privileges to perform its actions:

1. SeShutdownPrivilege
   – Required to shutdown the system
2. SeDebugPrivilege
   – A token field that allows the owning process to adjust the memory of other processes on the computer.  This is a very powerful privilege that allows the malware to perform near system level tasks.
3. SeTcbPrivilege
   – This is another very powerful privilege that allows the owning process to act as part of the operating system.

Prior to any major function being executed, the malware attempts to gain any/all of the above privileges. The values are added together using Boolean OR commands which can result in the following results:

- 0 = No privileges granted
- 1 = SeShutdownPrivilege
- 2 = SeDebugPrivilege
- 3 = SeShutdownPrivilege/SeDebugPrivilege
- 4 = SeTcbPrivilege
- 5 = SeShutdownPrivilege/SeTcbPrivilege
- 6 = SeDebugPrivilege/SeTcbPrivilege
- 7 = All privileges

Based upon the overall values constructed from these checks, the malware will perform varying sets of routines. Most notably are the methods for which to encrypt files on the system and in rebooting the system after infection.

## Credential Harvesting

If the variant detects that its process is running with SeDebugPrivilege privileges it will call a function that is highlighted in red in the image below.

Calling Credential Harvesters

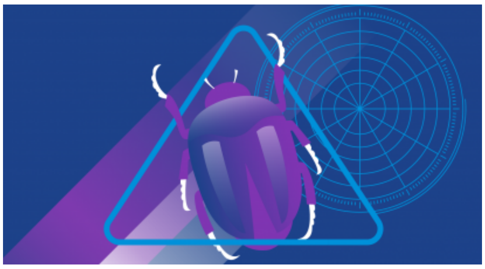
The malware contains four separate, compressed resources that are used as additional commands to execute during operation. This function will initially attempt to determine if the operating system is either x86 or 64-bit and then it will either load a password dumper from either Resource “1” or “2” (which ultimately are x86 and 64-bit executable files respectively). It should be noted that all of the four resources are stored as compressed buffers that can be inflated with zlib. Each resource contains a DWORD value at offset 0 that is the length of the data once it is inflated.

Once the OS type has been determined and the appropriate resource loaded, the sample will write the executable to the system %TEMP% path with a OS-generated temporary file name (highlighted in red in the image below). The sample will then create a named pipe to the temporary file (highlighted in blue), and execute the file as a new process, with a parameter of the named pipe (highlighted in green). This will allow the sample to get the results of the credential harvester.  Once completed the sample will delete the temporary file from the disk.

(Editor’s Note: This post was updated on 30 June to clarify the reboot sequence and adjust filename case for one process hash.)

:redential Harvesting

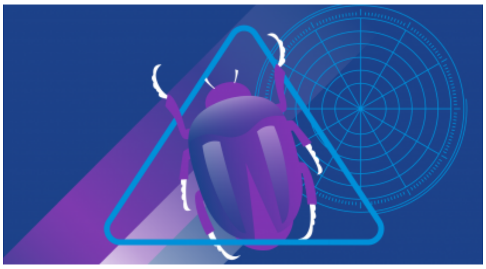
## Related Articles



Misc

NetSupport RAT: The RAT King Returns

Alan Ngo, Abe Schneider, Fae Carlisle / November 20, 2023 / 18 min read



Misc

Jupyter Rising: An Update on Jupyter Infostealer

Swee Lai Lee, Bria Beathley, Abe Schneider, Alan N... / November 6, 2023 / 18 min read

Misc

Hunting Vulnerable Kernel Drivers

Takahiro Haruyama / October 31, 2023 / 34 min read

Resources

Blogs

Careers

Communities

News and Stories

Topics

Trust Center

Support

Broadcom Support

Documentation

Hands-On Labs

Twitter

YouTube

Facebook