How did the WannaCry ransomworm spread?

blog.malwarebytes.com/cybercrime/2017/05/how-did-wannacry-ransomworm-spread/

Adam McNeil May 19, 2017

Security researchers have had a busy week since the WannaCry ransomware outbreak that wreaked havoc on computers worldwide. News of the infection and the subsequent viral images showing everything from large display terminals to kiosks being affected created pandemonium in ways that haven't been seen since possibly the MyDoom worm circa 2004.

News organizations and other publications were inundating security companies for information to provide to the general public – and some were all too happy to oblige. Information quickly spread that a malicious spam campaign had been responsible for circulating the malware. This claim will usually be a safe bet, as ransomware is often spread via malicious spam campaigns. Admittedly, we also first thought the campaign may have been spread by spam and subsequently spent the entire weekend pouring through emails within the Malwarebytes Email Telemetry system searching for the culprit. But like many others, our traps came up empty.

Claims of WannaCry being distributed via email may have been an easy mistake to make. Not only was the malware outbreak occurring on a Friday afternoon, but around the same time a new ransomware campaign was being heavily distributed via malicious email and the popular Necurs botnet. We recently wrote about the Jaff ransomware family and the spam campaign that was delivering it.

Some may have seen the rash of news occurring on their feeds, an uptick in ransomware-themed document malware in their honeypots, and then jumped to conclusions as a way to be first with the news.

But here at Malwarebytes we try not to do that. And now after a thorough review of the collected information, on behalf of the entire Malwarebytes Threat Intelligence team, we feel confident in saying those speculations were incorrect.

Indeed, the 'ransomworm' that took the world by storm was not distributed via an email malspam campaign. Rather, our research shows this nasty worm was spread via an operation that hunts down vulnerable public facing SMB ports and then uses the alleged NSA-leaked EternalBlue exploit to get on the network and then the (also NSA alleged) DoublePulsar exploit to establish persistence and allow for the installation of the WannaCry Ransomware.

We will present information to support this claim by analyzing the available packet captures, binary files, and content from within the information contained in The Shadow Brokers dump, and correlating what we know thus far regarding the malware infection vector.

Here's what we know

EternalBlue

EternalBlue is an SMB exploit affecting various Windows operating systems from XP to Windows 7 and various flavors of Windows Server 2003 & 2008. The exploit technique is known as heap spraying and is used to inject shellcode into vulnerable systems allowing for the exploitation of the system. The code is capable of targeting vulnerable machine by IP address and attempting exploitation via SMB port 445. The EternalBlue code is closely tied with the DoublePulsar backdoor and even checks for the existence of the malware during the installation routine.

```
<outputparameters>
  <parameter xdevmap='ETERNALBLUE_DOUBLEPULSAR_PRESENT' type='Boolean' name=
  'DoublePulsarPresent' description='Set to true if the DOUBLEPULSAR backdoor was already
  installed and the exploit did not have to be thrown'/>
</outputparameters>
<redirection>
  <local protocol='TCP' listenaddr='TargetIp' listenport='TargetPort' closeoncompletion=
```

EternalBlue checks for DoublePulsar

```
[*] Sending SMB Echo request
[*] Good reply from SMB Echo request
[-] Error doing SMB setup 0x%08X
    [+] Closing SMBv1 connection creating free hole adjacent to SMBv2 buffer.
[-] Error sending data for final SMBv2 buffer.
    [+] Sending final SMBv2 buffers.
    [+] Sending large SMBv1 buffer.
[-] Unable to allocate buffer for shellcode
    [+] Ping returned Target architecture: x64 (64-bit)
    [+] Ping returned Target architecture: x86 (32-bit)
        [+] Backdoor returned code: 70 - Error: ExAllocate/Free not found - Backdoor removed
        [+] Backdoor returned code: 60 - Error: Allocation Failed
        [+] Backdoor returned code: 50 - Error: Invalid Params
        [+] Backdoor returned code: 40 - Error: Invalid Transaction Params
        [+] Backdoor returned code: 30 - Error: Bad Transaction
        [+] Backdoor returned code: 20 - Error: Bad Opcode
        [+] Backdoor returned code: 10 - Success!
        [+] Backdoor returned code: %X - Error: Unknown error
[-] No response received from exploit packet. Not good.
    [+] ETERNALBLUE overwrite completed successfully (0x%08X)!
[*] Fingerprinting SMB non-paged pool quota
[*] Trying again with %d Groom Allocations
    [+] Backdoor installed
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-WIN-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
[*] Triggering free of corrupted buffer.
[*] Sending egg to corrupted connection.

DONE.
[*] Sending all but last fragment of exploit packet
[*] Building exploit buffer
[*] Target OS selected valid for OS indicated by SMB reply
[*] Auto target successful based on SMB reply
[*] Auto targeted based on SMB string
    [+] Backdoor not installed, game on.
    [+] Backdoor is already installed -- nothing to be done.
[*] Pinging backdoor...
```

EternalBlue strings

Bits of information obtained by reviewing the EternalBlue-2.2.0.exe file help demonstrate the expected behavior of the software. The screenshot above shows that the malware:

- Sends an SMB Echo request to the targeted machine
- Sets up the exploit for the target architecture
- Performs SMB fingerprinting
- Attempts exploit
- If successful exploitation occurs, WIN
- Pings the backdoor to get an SMB reply
- And if the backdoor is not installed, it’s game on!

The ability of this code to beacon out to other potential SMB targets allows for propagation of the malicious code to other vulnerable machines on connected networks. This is what made the WannaCry ransomware so dangerous. The ability to spread and self-propagate causes widespread infection without any user interaction.

DoublePulsar

DoublePulsar is the backdoor malware that EternalBlue checks to determine the existence and they are closely tied together.

This particular malware uses an APC (Asynchronous Procedure Call) to inject a DLL into the user mode process of lsass.exe. Once injected, exploit shellcode is installed to help maintain persistence on the target machine. After verifying a successful installation, the backdoor code can be removed from the system.

```
<paramchoice name="Function" xdevmap="DOUBLEPULSAR_FUNCTION_TYPE" description="Operation for backdoor to perform">
    <default>OutputInstall</default>
    <paramgroup name="OutputInstall" description="Only output the install shellcode to a binary file on disk.">
        <parameter name="OutputFile" description="Full path to the output file" type="String"/>
    </paramgroup>

    <paramgroup name="Ping" description="Test for presence of backdoor">
    </paramgroup>

    <paramgroup name="RunDLL" description="Use an APC to inject a DLL into a user mode process.">
        <parameter name="DllPayload" xdevmap="DOUBLEPULSAR_DLL_PAYLOAD" description="DLL to inject into user mode" type="LocalFile" />
        <parameter name="DllOrdinal" xdevmap="DOUBLEPULSAR_DLL_ORDINAL" description="The exported ordinal number of the DLL being
        injected to call" type="U32" >
            <default>1</default>
        </parameter>
        <parameter name="ProcessName" xdevmap="DOUBLEPULSAR_PROCESS_NAME" description="Name of process to inject into" type="String">
            <default>lsass.exe</default>
        </parameter>
        <parameter name="ProcessCommandLine" xdevmap="DOUBLEPULSAR_COMMAND_LINE" description="Command line of process to inject into"
        type="String">
            <default></default>
        </parameter>

    </paramgroup>

    <paramgroup name="RunShellcode" description="Run raw shellcode">
        <parameter name="ShellcodeFile" description="Full path to the file containing shellcode" type="LocalFile"/>
        <parameter name="ShellcodeData" xdevmap="EXPLOIT_SHELLCODE" description="Full path to the file containing shellcode to run" type=
        "LocalFile" />
    </paramgroup>

    <paramgroup name="Uninstall" description="Remove's backdoor from system">
    </paramgroup>
</paramchoice>
```

DoublePulsar Parameters

The purpose of the DoublePulsar malware is to establish a connection allowing the attacker to exfiltrate information and/or install additional malware (such as WannaCry) to the system. These connections allow an attacker to establish a Ring 0 level connection via SMB (TCP port 445) and or RDP (TCP port 3389) protocols.

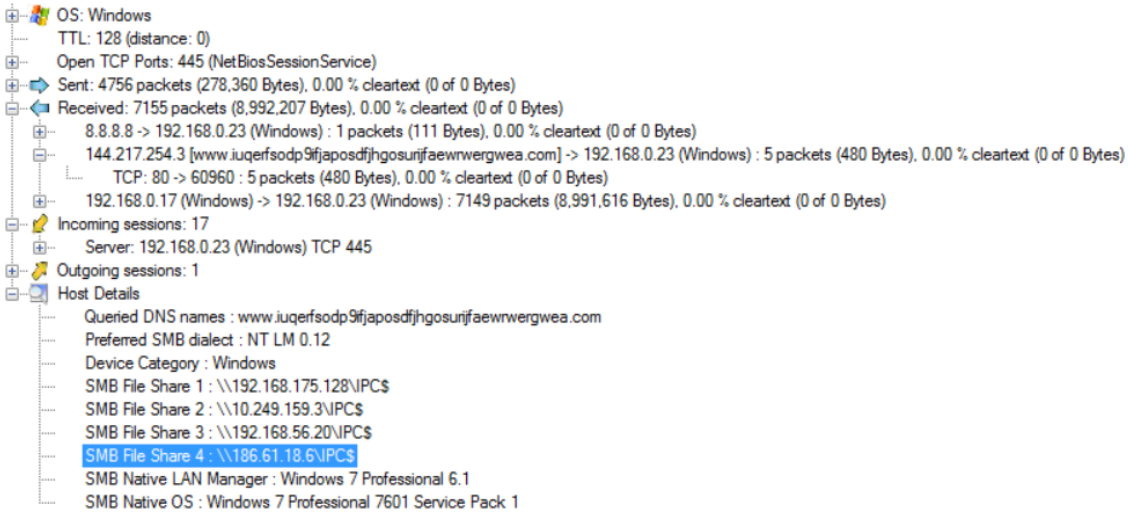
```
<paramchoice name="Protocol" xdevmap="DOUBLEPULSAR_PROTOCOL_TYPE" description="Protocol for the backdoor to speak">
    <default>SMB</default>
    <paramgroup name="SMB" description="Ring 0 SMB (TCP 445) backdoor">
    </paramgroup>
    <paramgroup name="RDP" description="Ring 0 RDP (TCP 3389) backdoor">
    </paramgroup>
</paramchoice>
```

DoublePulsar Ring0 Connections

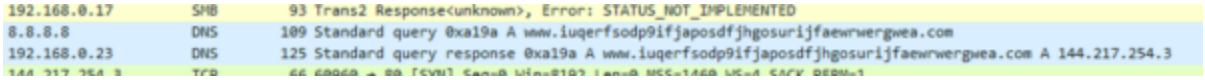
Network analysis

Taking a look at the wannacry.pcap file shared to VirusTotal by @benkow_ helps us attribute the previously discussed code as the infection vector via the initial calls of the attack cycle.

A high-level view of a compromised machine in Argentina (*186.61.18.6*) that attacked the honeypot:

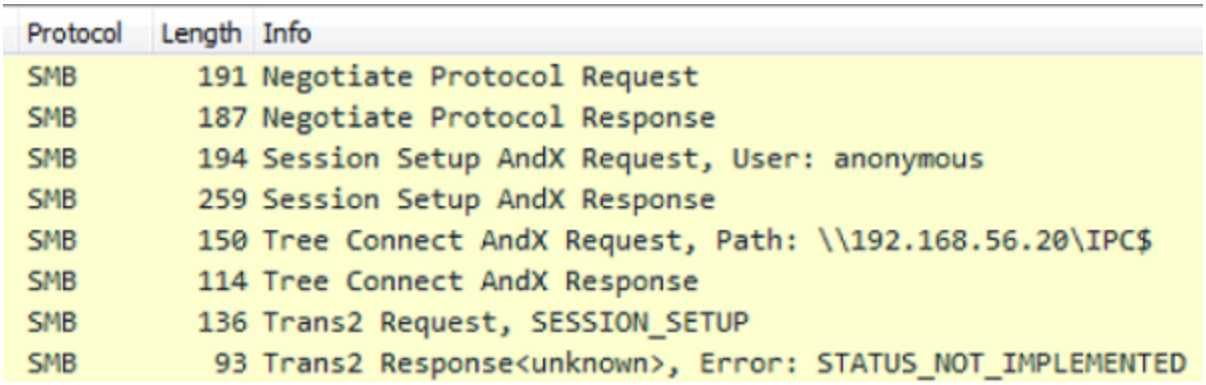


The widely publicized kill-switch domain is present in the pcap file. As was reported, the malware made a DNS request to this site. Until @MalwareTech inadvertently shut down the campaign by registering the domain, the malware would use this as a mechanism to determine if it should run.



DNS lookup to Sinkhole

The SMB traffic is also clearly visible in the capture. These SMB requests are checking for vulnerable machines using the exploit code above.

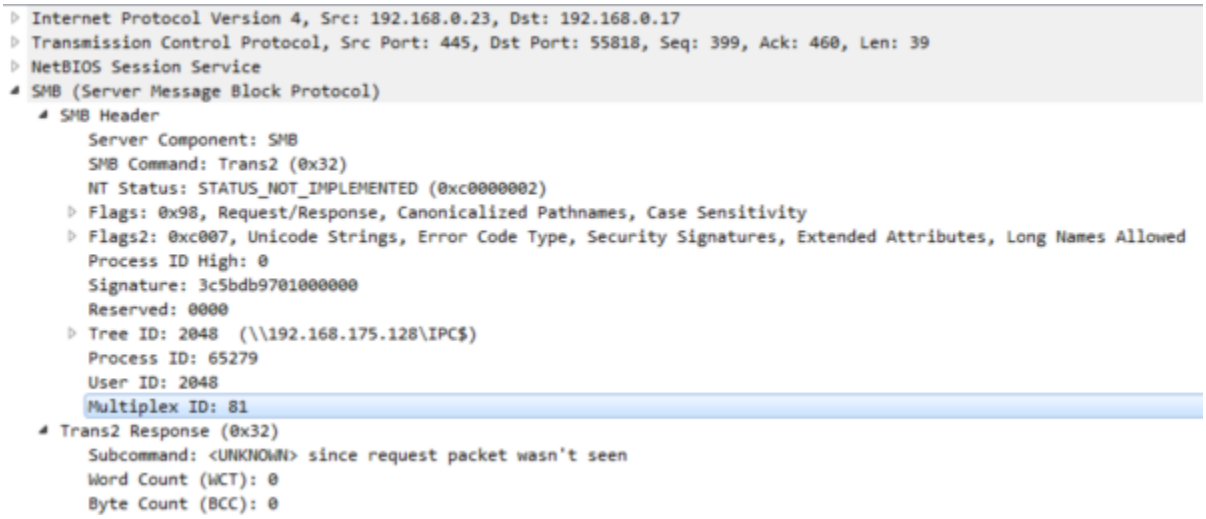


Protocol	Length	Info
SMB	191	Negotiate Protocol Request
SMB	187	Negotiate Protocol Response
SMB	194	Session Setup AndX Request, User: anonymous
SMB	259	Session Setup AndX Response
SMB	150	Tree Connect AndX Request, Path: \\192.168.56.20\IPC$
SMB	114	Tree Connect AndX Response
SMB	136	Trans2 Request, SESSION_SETUP
SMB	93	Trans2 Response<unknown>, Error: STATUS_NOT_IMPLEMENTED

SMB Requests

The exploit sends an SMB ‘trans2 SESSION_SETUP’ request to the infected machine. According to SANS, this is short for *Transaction 2 Subcommand Extension* and is a function of the exploit. This request can determine if a system is already compromised and will issue different response codes to the attacker indicating ‘normal’ or ‘infected’ machines.

Diving into the .pcap a bit more, we can indeed see this SMB Trans2 command and the subsequent response code of 81 which indicates an infected system. If the attacker receives this code in response, then the SMB exploits can be used as a means to covertly exfiltrate data or install software such as WannaCry.



Trans2 Multiplex ID

Putting it all together

The information we have gathered by studying the DoublePulsar backdoor capabilities allows us to link this SMB exploit to the EternalBlue SMB exploit. It's really not hard to do so as both were patched as part of the MS17-017 Security Bulletin prior to this event, and as previously mentioned, were both released in the well-publicized ShadowBrokers-NSA dumps.

Without otherwise definitive proof of the infection vector via user-provided captures or logs, and based on the user reports stating that machines were infected when employees arrived for work, we're left to conclude that the attackers initiated an operation to hunt down vulnerable public facing SMB ports, and once located, using the newly available SMB exploits to deploy malware and propagate to other vulnerable machines within connected networks.

Developing a well-crafted campaign to identify just as little as a few thousand vulnerable machines would allow for the widespread distribution of this malware on the scale and speed that we saw with this particular ransomware variant.

So what did we learn?

Don't jump to conclusions. Malware analysis is difficult and it can take some time to determine attribution to a specific group, and/or to assess the functionality of a particular campaign – especially late on a Friday (which BTW, can all you hackers quit making releases on Fridays!!). First, comes stopping the attack, second comes analyzing the attack. Remember, patience is a virtue.

Update, update, UPDATE! Microsoft released patches for these exploits prior to their weaponization. Granted, patches weren't available for all Operating Systems, but the patch was available for the vast majority of machines. This event even forced Microsoft to release

a patch for the long-ago EOL Windows XP – which gets back to the first thing that was said. UPDATE! Why are there still machines on XP!? These machines are vulnerable (beyond this attack) to the ransomware functionality of this attack and they need to be updated.

Disable unnecessary protocols. SMB is used to transfer files between computers. The setting is enabled on many machines but is not needed by the majority. Disable SMB and other communications protocols if not in use.

Network Segmentation is also a valuable suggestion as such precautions can prevent such outbreaks from spreading to other systems and networks, thus reducing exposure of important systems.

And finally, don't horde exploits. Microsoft president Brad Smith used this event to call out the '*nations of the world*' to not stockpile flaws in computer code that could be used to craft digital weapons.

That reminds me of an article I wrote a few years ago (and which was substantially cut for length) about Hacking Team and the government sanctioned use of exploits.

Hack Me: A Geopolitical Analysis of the Government Use of Surveillance Software

I guess things haven't changed…