Exploiting a Vulnerability to Deepen Our Knowledge of an OS

Jada Bryant - b63j795

Teyler Halama - b56k732

Justin Guerrero - j87n896

Arnold Smithson - q46t616

Operating Systems Fall 2020

Professor Travis Peters

Montana State University

Table of Contents

Introduction

With every upgrade in technology and security comes an upgrade of malware and malicious strategies. Over the decades, malware has surfaced in many forms, the most common of which are usually trojans, worms, or ransomware. And, much like biological viruses, malware can evolve from existing malware. One of the most famous base exploits that created several malware descendants was EternalBlue.

Discovered and used by the NSA around 2012, EternalBlue was an exploit that took advantage of a security vulnerability in several Windows operating systems. While an exploit itself does not inject any malicious code, it's a highway of sorts that expedites that injection of malware, which is why so many attacks, such as WannaCry and EternalRock, are based off of EternalBlue. The goal of this technical document is to apply our knowledge of the operating system and examine the EternalBlue exploit and what it takes advantage of, deepening our understanding of how exploits get discovered and developed along the way. To understand the EternalBlue exploit and its subsequent use in malware, however, an understanding of what fileless malware and ransomware are is necessary, as they are the most common types of malware that rise out of this exploit.

Background & Related Work

A Tale of Two Malwares: The History of Fileless Malware and Ransomware

Fileless Malware

Between the two malwares, technically, fileless malware has been around longer (that we know of) before ransomware. The first fileless malware program, called the Lehigh Virus, was detected in 1987 (Saad, 2019). At a high level, the malware would keep track of the number of times the computer has been booted since its installation by executing a file upon boot. At 4 times, the malware corrupts the boot sector and File Allocation Table, rendering the computer useless.

Fileless malware, by definition, is malware that hides within memory instead of putting a persistent file in the file system. The malware achieves this by using non- ASCII characters in the memory registers it lives within, making it more difficult for antivirus software to examine the registries. This form of writing makes fileless malware substantially more difficult for antivirus software to detect them, as antivirus software simply scans file systems for similar malware patterns to find and quarantine certain files (Sanjay, 2018). To spread and use fileless malware through a network, hackers can attempt to seize control of PowerShell or Windows Management Instrumentation.

Malware has been making more and more use of Microsoft Powershell to execute their malicious payload. Powershell itself is a system administration tool that allows users to look at the services of the OS, reconfigure systems and servers, automate tasks, generate a series of

scripts and schedule them to run in the background, and much more (“Security 101,” 2017). As evident, this versatility as well as the intimacy with the OS and its networks made this tool desirable for malicious users. Fileless malware uses Powershell as its main method to allocate memory for the script using “VirtualAlloc” and “VirtualProtect.” These two commands alone allow the malicious code to save a pre-loaded script to memory addresses and protect them from even the OS, accentuating that “notoriously-hidden” quality of fileless malware.

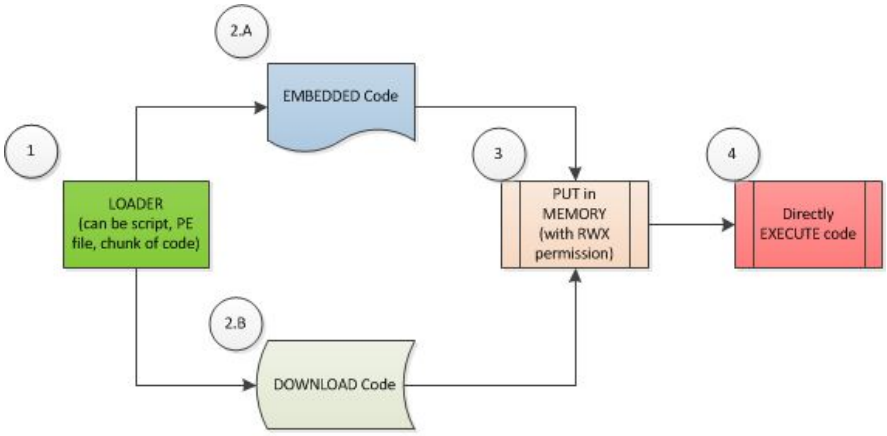


Figure 1: The process of infection for fileless malware (“Security 101,” 2017).

Figure 1 shows how fileless malware gets loaded into the system. The initial script is loaded into the computer, and executes code that either uses embedded code or downloads code from the network to put into memory. Then, to bypass the user restrictions put into Powershell, the malicious code is run via command line. By doing this, fileless malware can make use of other types of malware that have the potential to do even more harm to the computer.

Fileless malware, besides being its own class of malware, is also an attribute that can be combined with other types–namely, ransomware and worms. One example of a fileless worm is Code Red, a worm that attacked in increasingly severe cycles before terminating itself after a predetermined amount of time (Smelcer, 2017). Another famous fileless attack is the Poison Ivy Remote Access Trojan. Its main advantage over other malware is that instead of its memory being stored as a single chunk, it takes advantage of the external fragmentation of memory and splits itself across several sections, making the entire program hard to delete all at once (Smelcer, 2017).

While it is hard to detect, that doesn’t mean fileless malware is undetectable. Antivirus software, in order to improve detection, has started looking at abnormal behavior. Any abnormal behavior would include accessing resources not usually accessed or suspicious login activity (possibly at hours computer is not in use). However, this leads to more false positives, because behavior is subjective (Smelcer, 2017).

Ransomware

According to Trend Micro, the first Ransomware detection—unnamed in the article—was in 2005-06, in Russia (Trend Micro Incorporation, 2017). It started as locking commonly-used files and leaving a note asking for money, and quickly evolved into preventing the OS from booting by infecting the OS’s “Master Boot Record” (ibid.). Even worse, the ransomware grew to

delete files if payment wasn't issued in a timely manner, or the encryption wasn't cracked in time. The most common files that get locked in companies are database files and website files.

The ease of use of ransomware has made the malware skyrocket in popularity, with a business model named after it known as "Ransomware as a service." In this business model, malicious hackers can offer their services for a fee to users who want to extort other users or businesses. Either that, or "starter kits" can be sold underground to people who want to whip up a quick malware program to do what they want. However, as with most programs having an open-source alternative, so too does ransomware, in the form of "Hidden Tear." "Hidden Tear" is an open-source ransomware sample that can be modified for specific purposes. A public repository for it has been on Github since 2015. Recently, though, the variants of ransomware have exploded in number (ibid).

As recent as 2016, the amount of "ransomware families" jumped to almost nine times the number of families in 2015! With this surge came a wave of ransomware that takes advantage of the social engineering employed in a lot of the apps and sites users explore daily (ibid). This means ransomware can be delivered in the form of emails, pop-ups, downloadables, and anything else that humans have been engineered to click on and get information from them. This makes humans very susceptible to getting attacked by malware (ibid). Through this rapid growth of malware, a certain famous exploit was discovered, leaked, and subsequently abused: EternalBlue.

EternalBlue Exploit in the OS

The Discovery (And "Theft") of EternalBlue

EternalBlue is an exploit developed by the NSA that takes advantage of a software vulnerability in the Windows OS Server Message Block version 1 (SMBv1), a file sharing protocol. It allows attackers to remotely run arbitrary code and send packets to gain access to a network (MS-ISAC, 2019). Malware that utilizes this exploit is able to spread itself across the entire network and compromise all devices connected to that network.

While EternalBlue was only just revealed to the public in 2017, the NSA kept the secret far longer than that. The Washington Post released an article in 2017 detailing the NSAs involvement and their comment following the breach of data (Nakashima and Timberg, 2017). Prior to the infamous WannaCry attack of 2017, the NSA opted to keep this vulnerability secret so they could continue to collect foreign data from other countries. What the NSA didn't seem to expect, however, was that data would be collected from them.

A notable hacker group by the name of the Shadow Brokers managed to steal knowledge of the exploit from the NSA and developed the first major malware to use it, naming it WannaCry. Upon learning this was stolen, the NSA informed Microsoft of the vulnerability, allowing Microsoft to release a quick patch that covered up this exploit for all Microsoft operating systems still in service. However, post-WannaCry, the severity of this exploit was measured, and Microsoft issued patches even for operating systems out of service, such as Windows XP. While the patch protected against most attacks using this exploit, that didn't stop people from developing other programs that use it.

EternalBlue's Descendants

There were several new malware programs that made use of this exploit, some more popular than others. While a select few have been covered by mass media due to the damage they've done, several more malware have been created as recently as 2019 using the exploit. The following figure shows a chart of the many branches EternalBlue has sprouted:

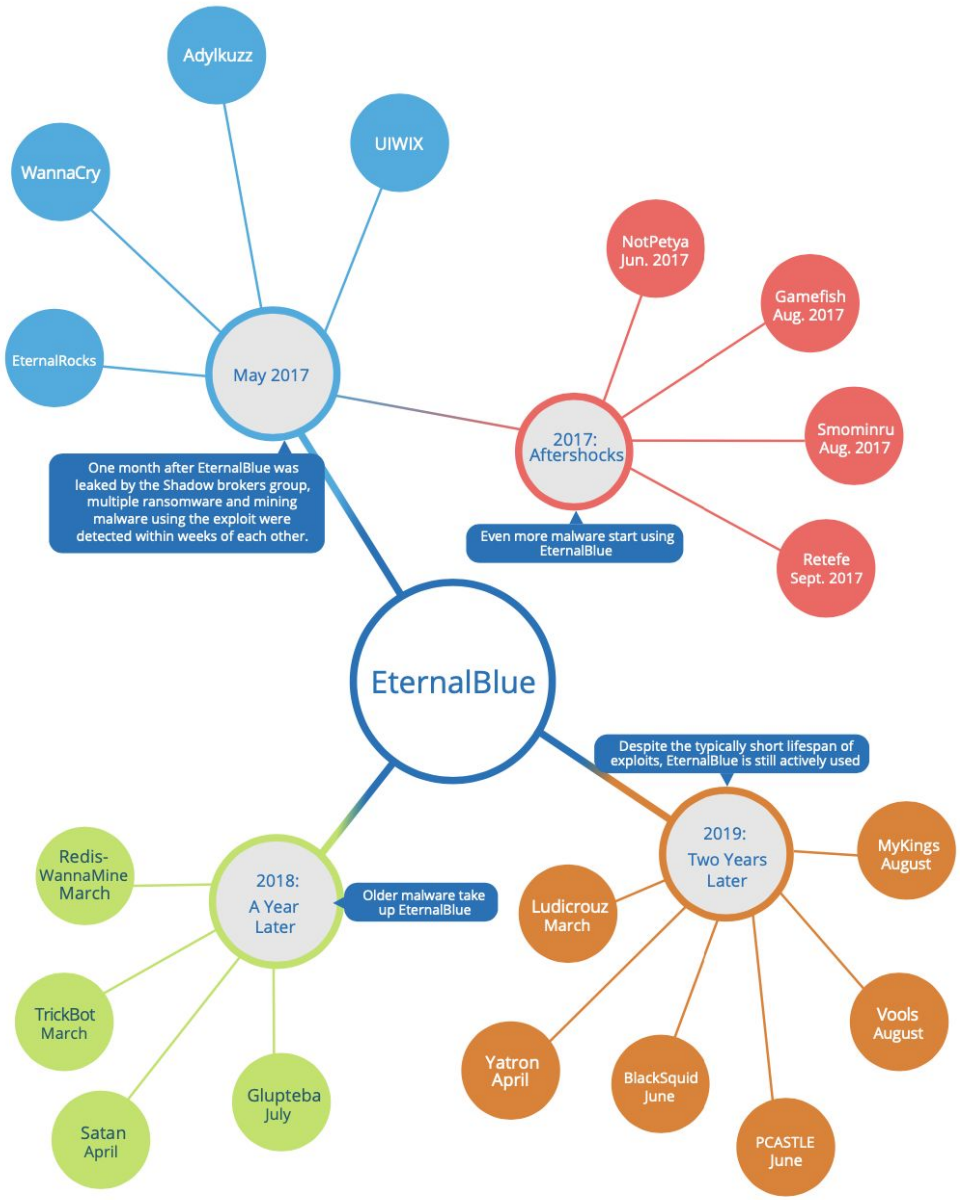


Figure 2: A chart of malware born from the EternalBlue exploit (Trend Micro Corporation, 2019).

The first malware mentioned already was WannaCry, the famous attack that first put EternalBlue in the spotlight. WannaCry was a ransomware that took thousands of files hostage and demanded exorbitant amounts of currency from the infected users and companies. It ended

up infecting hundreds of thousands of computers, putting a lot of people on edge about where the next attack might come from.

The next prolific attack that made use of EternalBlue was NotPetya that same year. Similar to WannaCry, NotPetya is a ransomware that encrypts files and demands sums of bitcoin to unlock them. What the victim doesn't realize is that NotPetya never actually unlocked or decrypted those files, resulting in a complete loss of data and property for the affected companies (Greenburg, 2018). NotPetya, however, targeted a major shipping company that moves ⅛ of the world's shipping capacity via boat. This caused the company to stop its entire network to halt the spread of the attack, also halting that shipping process. It caused quite a bit of damage, rendering many computers useless. According to an article from Wired, it was considered to be the fastest-spreading piece of malware ever (Greenburg, 2018). Other pieces of malware have used this exploit, but NotPetya and WannaCry are by far two of the most notable programs.

Wannacry

Wannacry takes advantage of EternalBlue to compromise Windows machines, load malware, and self-propagate to other devices on the network. An attack begins with a SMB handshake, the request and response for the protocol, and it connects to the IPC$ share that allows subsequent named pipe connections on the remote machine. It then sends an initial NT Trans request which moves the SMB server to a state where the vulnerability is exposed and the exploit can take place. A specially crafted packet is created large enough to require multiple secondary Trans2 Requests which in turn are malformed to accommodate for the size of the request (FireEye, 2017). This is when the malware makes its debut on the remote machine via an encrypted payload containing shell code. After successfully triggering the vulnerability, the payload launches the service "mssecsvc" which contains the local network and internet for vulnerable and connected machines. The service then uses the exposed SMB ports to remote into one of these machines and deliver the payload. This attack can penetrate all machines in a typical LAN in just a few minutes.

The ransomware contains an executable file that scans the network and triggers the vulnerability on machines in the network. In the same resource section, there is a section named "R" that contains the code for the ransomware that has a password protected, encrypted zip file resource named "XIA". This file contains two other executable files, the encrypted keys, image files, and a Tor client. This zip can be extracted with the password that is in the malware code "Wncry@2ol7" (FireEye, 2017) .

NotPetya

NotPetya was originally thought to be ransomware because it displays a ransomware message after infecting the machine, however it has been determined that it is more of a destructive disk wiping malware. NotPetya harvests all SMB and user credentials from the machine and uses those credentials to compromise other machines on the network. Because NotPetya uses these connection credentials, it only needs to infect one machine to be able to infect other machines on the network even if they have been patched for SMBv1 vulnerabilities. Once a system has been infected, NotPetyas obtains read write access to the hard drive through Windows API DeviceIoControl (LogRhythm, 2017). This allows the malware the ability

to unmount mounted volumes, determine the amount of disks and sectors in the machine and how much memory each disk or sector has. The malware is then able to overwrite sectors of the hard drive and because of this it is unable to restore files even if the ransom has been paid.

History Summary

Ransomware and Fileless malware have been around for decades, with increasing complexity to match the growing cybersecurity industry. Because of fileless malware's adeptness at hiding itself within memory, it became a favored tactic among hackers to inject malicious code without being detected, allowing for other malware to more easily infect other computers. The EternalBlue exploit seemed perfect for this type of malware because it allowed that malware to gain easy access to what it needed in order to inject the code, causing hundreds of thousands of computer infections. Even today, some computers that run older operating systems for a myriad of reasons will now find themselves vulnerable to this exploit. These reasons and more are why our goal is to examine this exploit with a fine level of detail. We're hoping our knowledge of how operating systems work will help us understand what this exploit is and how malware have come to take advantage of it.

Methods

How knowledge of OS helps understand an exploit: An Overview of EternalBlue

This section is dedicated to an in-depth walk-through for the reader to gain knowledge and a deeper understanding of the details and topics discussed in this document. The reader will also gain an understanding of how fileless malware and other various forms of malware affect an operating system.

Eternal Blue is an attack that exploits the Operating System by taking advantage of some bugs in the Microsoft SMB code. The utilization of the bugs help change the execution path of the program, triggering a response that damages files or exposes private information. This section should show not only the importance of security, but demonstrate the importance of testing an Operating System thoroughly so it cannot be creatively exploited.

The goal of Eternal Blue is to gain access to the root level user and remotely control a terminal from the attacking computer. The end result is normally an attacker with elevated privileges capable of executing programs, adding files, and deleting files. Worst case scenario, modify files on the network since SMB is connected to a local network. With the I/O (Input / Output System) being a network connection and having SMB in play, this allows for multiple possibilities to gain access to a user's files by traversing infected personal computers and working its way to a network server.

What is SMBv1?

SMBv1 was developed in 1983, with the goal of file sharing on a local network. It is a network communication protocol that enables shared access to files, printers, and ports. It also allows machines to communicate and provides remote services. SMB is an inefficient protocol as it uses a lot of networking resources, even more so when transmitting over expensive WAN links. It comprises about 10% of all network traffic in an Enterprise network (Stealthbits, 2020). Originally, SMB ran over NetBIOS (Network Basic Input/Output System) but after being renamed in 1996 to Common Internet File System, it ran over direct connections on TCP port 445.

Where the vulnerability lies is in packet transmission. Hackers are able to create specially crafted packets that SMBv1 is not capable of handling, allowing arbitrary code to be run on the machine. It is now an outdated protocol that has been replaced by faster, more secure versions that don’t hog network resources (Ned Pyle, Microsoft, 2016).

SMBv1 is enabled by default on Windows 8 and older versions of Windows 10. Operating systems older than Windows 8.1 and Windows Server 2012 R2 do not have the ability to remove or disable SMBv1. Some older applications haven’t updated to using SMBv2 or

SMBv3. In the case of an older operating system being used or a SMBv1 application is being used, it may be necessary to continue using SMBv1, thus it needs to be implemented as securely as possible when possible. EternalBlue is just one of the exploits that came from SMBv1. Other exploits the NSA developed were EternalRomance and EternalChampion which became part of the Metasploit Project (Stealthbits, 2020).

A Combination of Bugs in SMB to Gain Kernel Access

A great attack of any kind starts by studying a target enough to know more than one weakness. In this case, there were a total of 3 major flaws within the Microsoft SMB code that allowed for an exploit gaining RCE (Remote Code Execution) via kernel level access. These techniques used to gain control of an OS are nothing new, they were simply possible through some inconsistencies in the code. The succeeding sections will walk through how each individual bug allowed for this outcome and how they can be used together to obtain the goal attackers were looking for.

Bug 1: Improper Casting Bug

In order to understand some of the parsing errors that Eternal Blue took advantage of, it's worth having a basic understanding of File Extended Attributes (FEA). Regardless of which operating system you are on, there are attributes on files that the specified OS File System understands. These attributes the the operating system understand are mainly security and record related (i.e. File permissions when doing ls -l` and the last time a file was modified). These attributes help for managing the file system as a user or administration. It can be thought of like a barcode on a grocery store product. There are some important things as a buyer that need to be presented to them in order to understand what to purchase. However, the store needs the additional information about that product to do some behind the scenes logistics in the store.

The OS uses FEA to track computer file metadata that the file system does not know about. The Check Point Research article (Check Point Research, 2019) does a great job of writing up some pseudo code to show how this works, however a greater understanding of FEA's can be found in the TechRepublic article (TechRepublic, 2009) even though it is teaching FEA in the through the eyes of a linux user.

It is crucial to know that File Extended Attributes contain file metadata that have to do with low level OS related functionality such memory allocation and even buffer overflows. These include things such as author, flags, and most importantly, byte length info.

If there is a way to trick something into reading a file as bigger than it actually is, you would get a buffer overflow. Buffer overflows exceed a buffer region of memory in order to intentionally feed input that a buffer can't store. The way this exploit is tricked into by creating a buffer overflow is through a bug in the function that converts an FEA format Os2 to the windows format Nt (Check Point Research, 2019) .

What should happen in the function SrvOs2FeaListSizeToNT is a simple conversion of FEA format from the generic Os2 FEA format to the Windows NT format. If the

SizeOfListInBytes attribute the FEA is less than 2^16, the expected result will occur as donated in Figure 3:

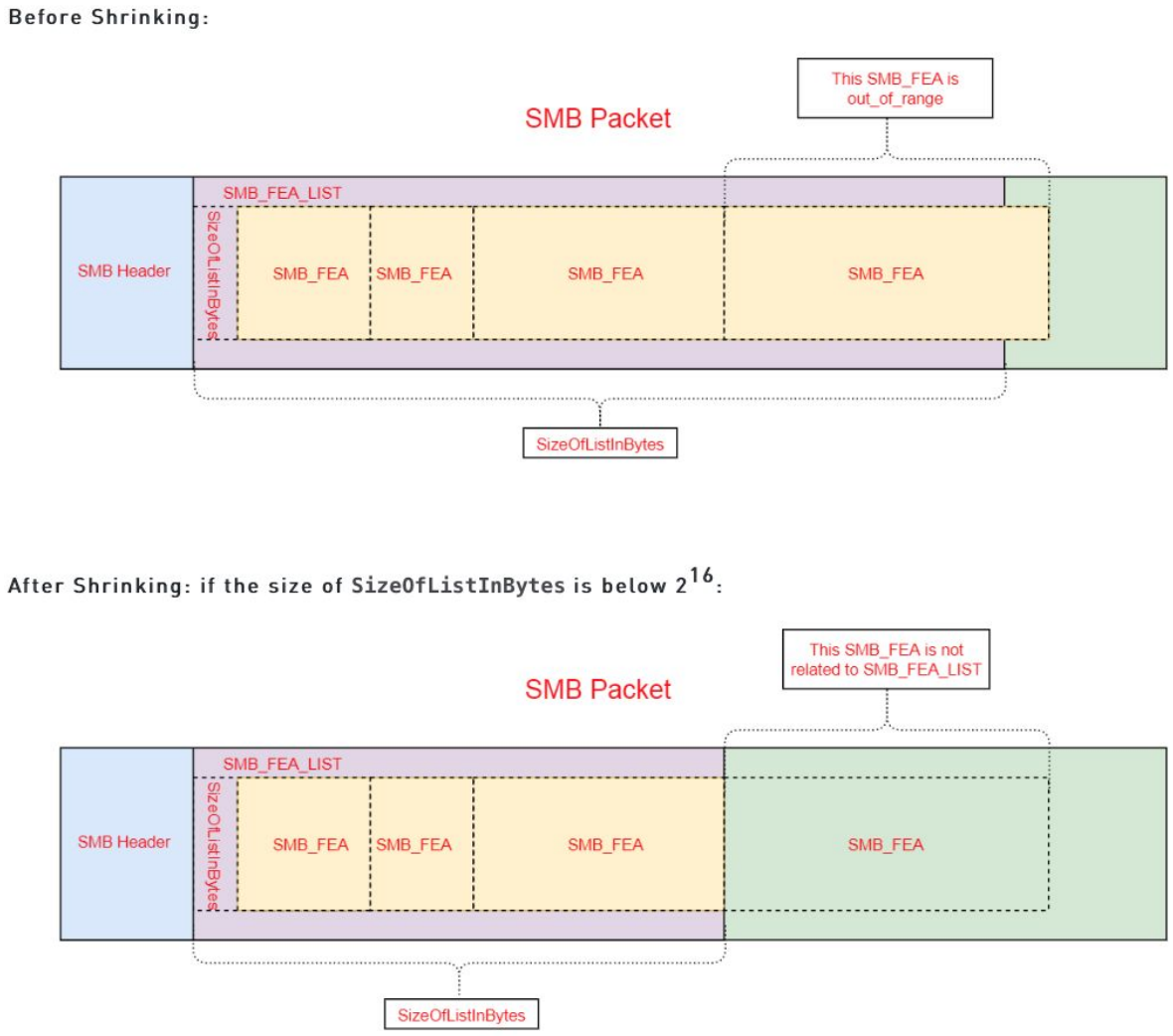


Figure 3: SMB packet model describing basic allocation (Check Point Research, 2019).

However, if the SMB_FEA that is being added to the SizeOfListInBytes is above the range of 2^16, it would be enlarged instead of shrunk. This error causes an out of bounds write as denoted by the UnRelated Data in Figure 4:

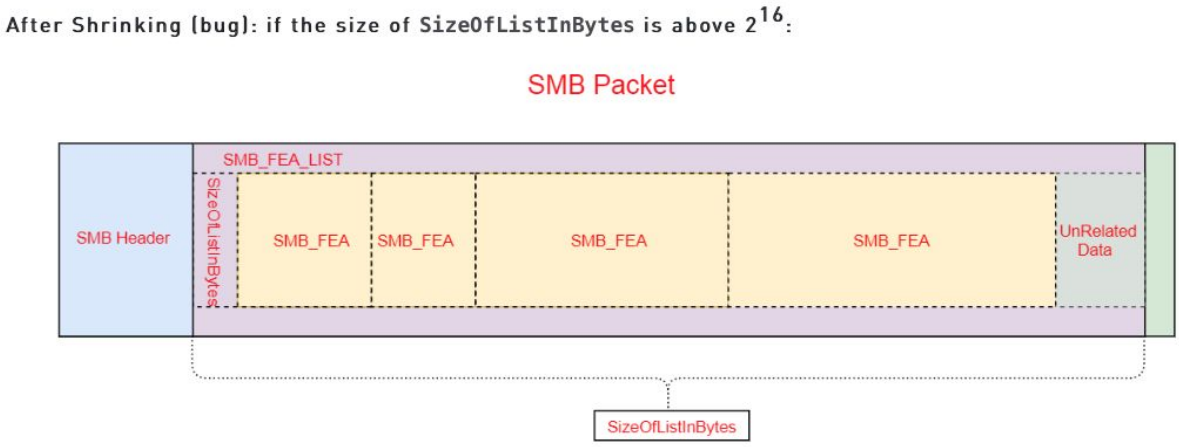


Figure 4: Bug 1 in the SMB packet (Check Point Research, 2019).

Bug 2: Improper Parsing Bug

This bug at its core is caused by a lack of validation. At its core, there are two SMB transaction functions that handle transmitting files over an SMB protocol. As denoted by the pseudocode naming convention (Check Point Research, 2019), SMB_COM_NT_TRANSACT handles data represented by a parameter that has a max size of DWord (Double Word 32 bits). However, SMB_COM_NT_TRANSACTION2 handles data represented by a parameter in the header at a max size of Word (16 bits).

In order to handle the edge cases where the data is too large for SMB to send over a single packet, there is a sub-command _SECONDARY for both of these functions. Examples of the original command followed by its proper sub command would resemble:

SMB_COM_NT_TRANSACT => SMB_COM_NT_TRANSACT_SECONDARY
SMB_COM_TRANSACTION2 => SMB_COM_TRANSACTION2_SECONDARY

This issue for this bug again is due to the lack of validation between the original command and its proper subcommand. A comparison to this would be if you were to validate a form on the front end of a website, but didn't validate on the backend. You could theoretically find out what the front end was looking for and still send something malicious to the back end.

With that in mind, this means it is entirely possible to have a case where SMB_COM_NT_TRANSACT has a following sub command of SMB_COM_TRANSACTION2. This would mean that the first half would read the split packet as a DWord while the second half is read as a Word (Check Point Research, 2019).

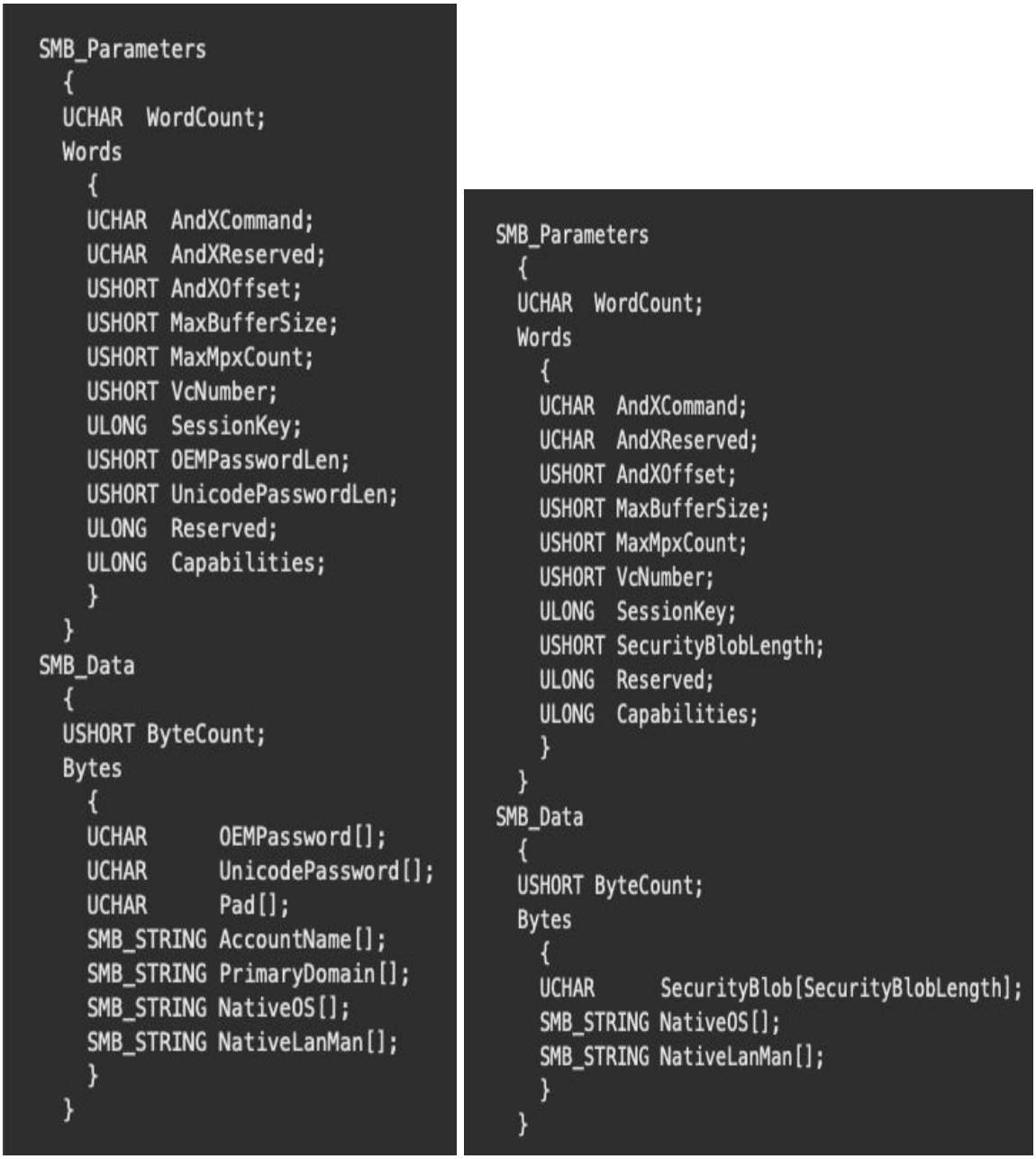
Bug 3: Create a Hole

This bug has a lot going on. The goal of this section is to create a hole with a specified size in the kernel non-paged pool. We are then going to use Bug 1 and 2 to fill that hole we create with this bug to cause an out of bound write to the next chunk.

A great metaphor for this bug is comparing it to a DMV form you have to fill out to get your driver's license. Think about how the form has sections that you as the applicant have to fill out. There are check boxes, paragraphs, and other information that the DMV allows you to fill. At the bottom, there is usually a “Employee use only” section which is only allowed to be used by an employee working at the DMV.

A SMB packet is basically like a form. There are parts left up to the client sending the packet to be filled. Let’s say that the employee that works at the DMV is the Busy Chunk of our SMB packet. This bug basically moves the line of the “Employee use only” down the form a little so you as the applicant can fill in things only meant to be written by employees. That way, the malicious user can input data that the employees don’t expect or can’t accommodate, causing confusion. Then, when all the paperwork gets settled, the DMV treats that info that was only meant to be filled out by employees with higher priority. With the SMB packed, this bug is going to trick the OS executing code that the client wrote on the Busy Chunk.

Upon an SMB connection, in order to establish a session a SMB_COM_SESSION_SETUP_ANDX request must be sent to the client to establish authentication (Check Point Research, 2019). There are 2 formats for this request. The first is LM and NTLM authentication and NTLMv2 or NTLM SSP authentication (More documentation on this format: [MS-SMB]: Client Request Extensions). Both of these authentications have their own struct formats:



```
SMB_Parameters
  {
  UCHAR  WordCount;
  Words
    {
    UCHAR  AndXCommand;
    UCHAR  AndXReserved;
    USHORT AndXOffset;
    USHORT MaxBufferSize;
    USHORT MaxMpxCount;
    USHORT VcNumber;
    ULONG  SessionKey;
    USHORT OEMPasswordLen;
    USHORT UnicodePasswordLen;
    ULONG  Reserved;
    ULONG  Capabilities;
    }
  }
SMB_Data
  {
  USHORT ByteCount;
  Bytes
    {
    UCHAR      OEMPassword[];
    UCHAR      UnicodePassword[];
    UCHAR      Pad[];
    SMB_STRING AccountName[];
    SMB_STRING PrimaryDomain[];
    SMB_STRING NativeOS[];
    SMB_STRING NativeLanMan[];
    }
  }
```

```
SMB_Parameters
  {
  UCHAR  WordCount;
  Words
    {
    UCHAR  AndXCommand;
    UCHAR  AndXReserved;
    USHORT AndXOffset;
    USHORT MaxBufferSize;
    USHORT MaxMpxCount;
    USHORT VcNumber;
    ULONG  SessionKey;
    USHORT SecurityBlobLength;
    ULONG  Reserved;
    ULONG  Capabilities;
    }
  }
SMB_Data
  {
  USHORT ByteCount;
  Bytes
    {
    UCHAR      SecurityBlob[SecurityBlobLength];
    SMB_STRING NativeOS[];
    SMB_STRING NativeLanMan[];
    }
  }
```

Figure 5: LM and NTLM on Left. NTLMv2 or NTLM SSP on the right. (Check Point Research, 2019).

As you can see, they are broken into two main sections: SMB_Parameters and SMB_Data. Specifically, the USHORT ByteCount property “represents the length of the SMB_Data struct members section in bytes” (Check Point Research, 2019). We can also denote that the UCHAR WordCount property represents the length of SMB_Parameters in size Word (Check Point Research, 2019).

The bug exists inside of a validation check that happens named SrvValidateSmb. There is nothing inherently wrong with this function, but there is an issue in a child function named BlockingSessionSetupAndX. When BlockingSessionSetupAndX is called, it wrongly calculates

the ByteCount. This is because the ByteCounts are different. Listed below are the summed totals of the WordCount from each authentication struct:

- LM and NTLM Auth: WordCount == 13
- Extended Security Format (NTLM SSP): WordCount == 12

```
BlockingSessionSetupAndX(request, smbHeader)
{
    // ...

    // check word count
    if (! (request->WordCount == 13 || (request->WordCount == 12 && (request->Capablilities & CAP_EXTENDED_SECURITY))) ) {
        // error and return
    }

    // ...

    if ((request->Capablilities & CAP_EXTENDED_SECURITY) && (smbHeader->Flags2 & FLAGS2_EXTENDED_SECURITY)) {
        // this request is Extend Security request
        GetExtendSecurityParameters(request);  // extract parameters and data to variables (allocation)
        SrvValidateSecurityBuffer(request);  // do authentication
    }
    else {
        // this request is NT Security request
        GetNtSecurityParameters(request);  // extract parameters and data to variables (allocation)
        SrvValidateUser(request);  // do authentication
    }

    // ...
}
```

Figure 6: Bug 3 validation error (Check Point Research, 2019)

From Figure 6, you can see that it is entirely possible to send a SMB_COM_SETUP_ANDX as Extended Security with CAP_Extended_Security without adding FLAGS2_EXTENDED_SECURITY (Check Point Research, 2019). That means the example request will bypass the 2 proper if statements and go to the else statement which will incorrectly process the request as an NT Security request, which has a WordCount of 13 instead of the correct 12.

As a result of this bug, it allocates space in the non-paged kernel pool for additional properties like NativeOs and NativeLanMan unicode strings that don't exist in an Extended Security request (Check Point Research, 2019). This allocation can act as a placeholder for data. In other words, we just made a new form field in the "Only for use of the DMV" that the DMV will think was written by a privileged employee.

Putting the Bugs Together (to gain RCE)

All of this knowledge comes into an eight step process that Checkpoint Research has laid out perfectly. If you wish to understand this in great depth, check out the "Exploitation Technique" from (Checkpoint Research Point, 2019).

The really boiled explanation of how Eternal Blue gains REC is by using bugs 1-3 as tools against the SMB packet.

- Bug 1: Create a buffer overflow in memory. Allows for overwrite of the FEA header characteristics (Creating a Ushort 16 bit in place of a Long 32bit)
- Bug 2: Parse the packet incorrectly so that the heap gets shrunk and the desired code is overlaid into memory
- Bug 3: Grooming. Send data down to the overlaid memory and delete creating a hole. Rinse and repeat until that hole in memory is big enough to place the SMB data packet to be placed in memory

With these tools, the result is RCE that allows the attacker to have a reverse shell with execute privileges. The execute privileges come from the HAL (Hardware Access Layer) being the memory that this exploit overwrites. RCE is achieved through utilizing small bugs combined creatively to manipulate the operating system to that attackers desire.

Discussion

How Do We Avoid the Threat of EternalBlue?

Lucky for most of the world, EternalBlue was patched in March of 2017, finally giving some sense of security in the cyber world. However, that does not mean that EternalBlue is dead. The patch from Microsoft fixed the three major vulnerabilities discussed earlier in this document. The first of the three was one that was found inside the SMB client, which will be referred to as Bug A. A flaw in the process of converting FEA (File Extended Attributes) from the Os2 structure to the NT structure by the Windows SMB implementation (srv.sys driver) leads to a buffer overflow in the non-paged kernel pool that was discussed earlier in this document. The patch—simple, yet effective—changed the function SrvOsFeaListSizeToNt() from using the primitive value Unsigned Short (Ushort) to an Unsigned Long (Ulong)(Check Point Research, 4 Feb. 2019). An Unsigned short has a bit value of 2 where the Long has a bit value of 4. This gives quite a bit more range to values, which is one of the main reasons the Buffer Overflow attack was possible. Ushorts have a range value of 0-65,535 whereas the Ulong values have a range of 0-4,294,967,295, allowing much larger values to be passed through the protocol. This change allowed the function in question to work properly and prevent the Buffer Overflow; the patch can be seen below:

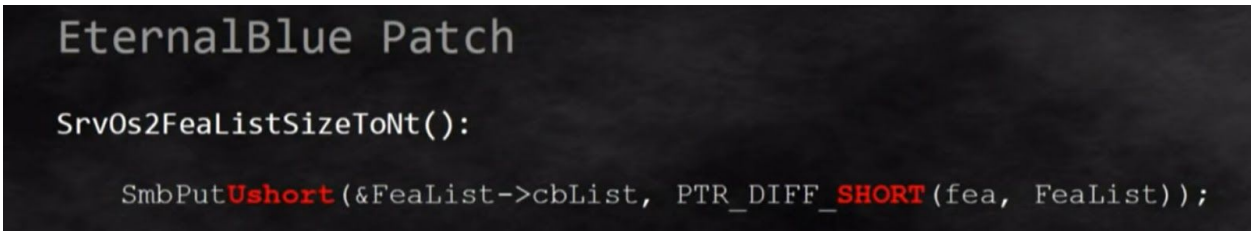


Figure 8: The vulnerable function(Check Point Research, 4 Feb. 2019).

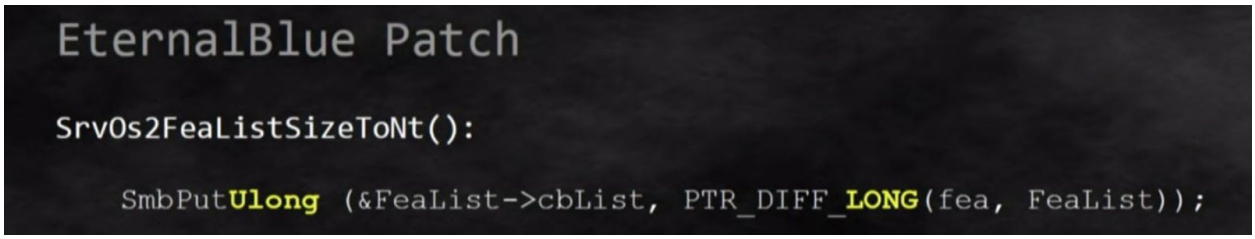


Figure 9: The patched function(Check Point Research, 4 Feb. 2019).

While this may not seem like a big difference, when it comes to an entire protocol, changing the variable type now disallows the buffer overflow in the system.

The second major bug that was found in EternalBlue as described earlier was the ‘Oversized Trans/Trans2 Request’ and will be referred to as Bug 2. This exploit as noted above, was what allowed the user to place the backdoor inside of the system via the SMB block. Through the creation of a successful buffer overflow attack in phase 1 of the 3 part exploit the user was able to gain access to the targeted system. The way that the transactions occur is not the problem, but rather how the information was sent. As we covered above, the lack of validation in the functions (SMB_COM_NT_TRANSACT and SMB_COM_NT_TRANSACTION2) allowed for the packages to be sent in arbitrary order, allowing Bug 1 to be triggered(Check Point Research, 4 Feb. 2019).

This was primarily fixed by the patch created for Bug 1. Another major fix that was done to prevent Bug 2 from continuing to be an issue was the correction of the DWORD and WORD values. The correction allowed for DWORD to DWORD subtraction, and WORD to WORD subtraction as well. As noted before, DWORD (32 bit) values are twice the size as WORD(16 bit) values, allowing the buffer to be overwritten(Check Point Research, 4 Feb. 2019).

This leaves the last of the three bugs in this exploit, henceforth labeled as Bug 3. This bug was not really a bug in itself, but rather something that filled a gap—quite literally. Bug 3 will not work without the help of Bugs 1 and 2; however, it did leave the system awfully vulnerable as discovered earlier in this document. The major fixes that happened with this bug were changing a bit of code in the BlockingSessionSetupAndX(request, smbHeader) function. The fix was again related to the patch created for Bug 1(Check Point Research, 4 Feb. 2019). This allowed the GetNtSecurityParameters(request) function to be processed correctly. The patch also prevented any further instances of allowing the Extended Security with a word count of 12 to be interpreted as the Cap Ended Security function parsed as an NT with a word count of 13 (the buffer overflow).

So what was learned here? Small changes within a code can make the difference between tens of billions of dollars worth of damage worldwide and a safe and secure operating system. Just as what was learned in the Patriot Missile Control System many years ago, code is fragile. Powerful, but fragile.

Perpetuating Damages

But what’s next? Everything’s all safe now, right? Well, not entirely. The speculation of EternalBlue is that it still poses a threat today. Which, long after the devastating attacks that took place in 2017, seems far fetched. However, EternalBlue is still one of the most common exploits used today. To this day, the number of unpatched systems in the world is still over 1 million worldwide, thus allowing the threat of being exploited by EternalBlue to remain prevalent all over(SentinelOne, 4 May 2020). This raises the question: why have these systems not been subject to the readily available patch?

Microsoft has stated that their patch completely closes the risk of exploitation through the SMB Protocol as discussed throughout this document. These updates could potentially stop the deployment of ransomware, malware, and worms, amongst other various types of cyber attacks that happen every day.

The issue that is speculated for most users is that many systems online will need a mandatory Windows software update installed in order to provide protection, and, while these software updates may be important, many users ignore them for various reasons.

This is the primary reason that EternalBlue has had such a lengthy and destructive life in the wild: the failure of businesses, corporations, and everyday users to update their systems to a newer version of Windows OS has caused the exploit to live on. Below shows an image that gives an estimated amount of systems still unpatched by Microsoft's security update MS17-010.

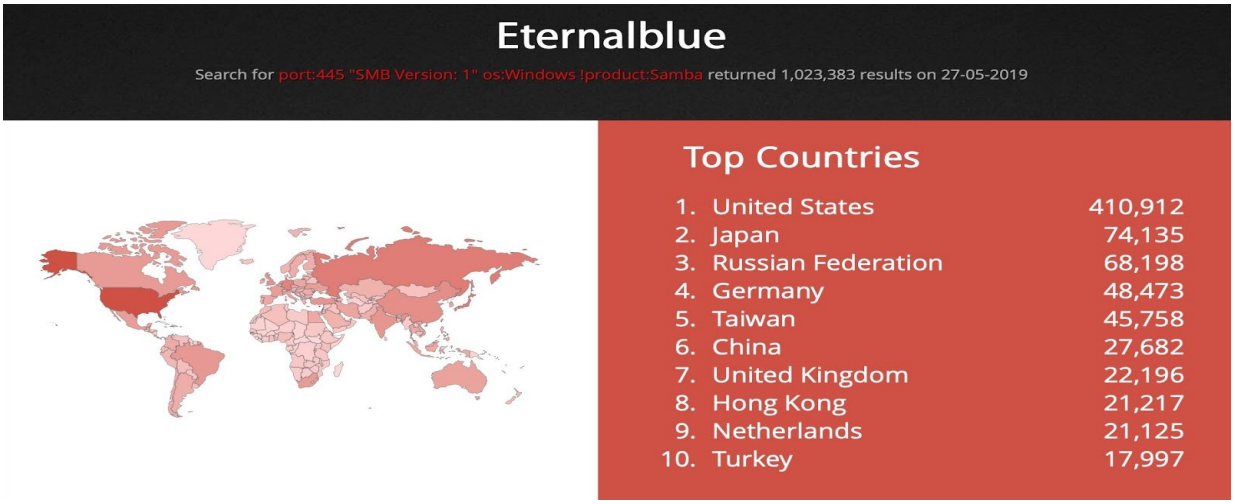


Figure 10: The number of machines still unpatched(SentinelOne, 4 May 2020)

The damages discussed in the previous section of this document revealed that the EternalBlue exploit accumulated over 1 billion dollars of loss and damage in over 65 countries throughout the course of its early life in the wild(SentinelOne, 4 May 2020) Furthermore, with the amount of unpatched systems discussed above still vulnerable to the exploit, it's fairly safe to say that there are still damages that are happening right now. Thus, it proves necessary to discuss the possible damages that could still be occurring from the utilization of the EternalBlue exploit.

As noted earlier with the abundance of computers that are still vulnerable to this particular exploit, it is indeed possible that one could potentially target these systems. Provided the attacker has the right set of skills and knowledge to run the script, it's fairly easy.
To elaborate on the skill set needed to utilize the EternalBlue exploit, one must know in depth how the exploit works, right?

Well not exactly. Today, there are a multitude of websites that are designed to practice and play with Penetration Testing all around the web. These websites, while designed to be fun and safe space environments, assist with teaching the techniques and skills involved to go from a "script kittie" (an inexperienced hacker who relies of previously written exploits and code to gain access into systems) to an elitist such as those who are known as the masterminds of devastating attacks such as those discussed throughout this document.

For example, the creators of the website HackTheBox have gone to the lengths of creating "boxes" (a practice environment such as a windows or linux machine) that are subject

to the EternalBlue exploit, essentially giving step by step instructions on how to use this exploit with no prior knowledge of how SMB or Buffer Overflow attacks work.

This marks dangerous territory for those who are still subject to the exploit due to the amount of resources and simplistic methods that have become public knowledge throughout the last 2 ½ years. For better or for worse, these websites exist and give aspiring hackers an environment to grow and learn from. However, this still poses a threat to those who are still using unpatched systems since there are still an estimated blocking of around 20 million attempts to utilize the EternalBlue exploit to attack systems every month, as reported in June 2020(avast). Part of the reason why this exploit still has so many attempts to attack systems daily is because of how easy it is to check to see if a system is still vulnerable. With penetration testing tools such as Nmap (network scanning tool) it makes it relatively simple to find a target (see figure below).

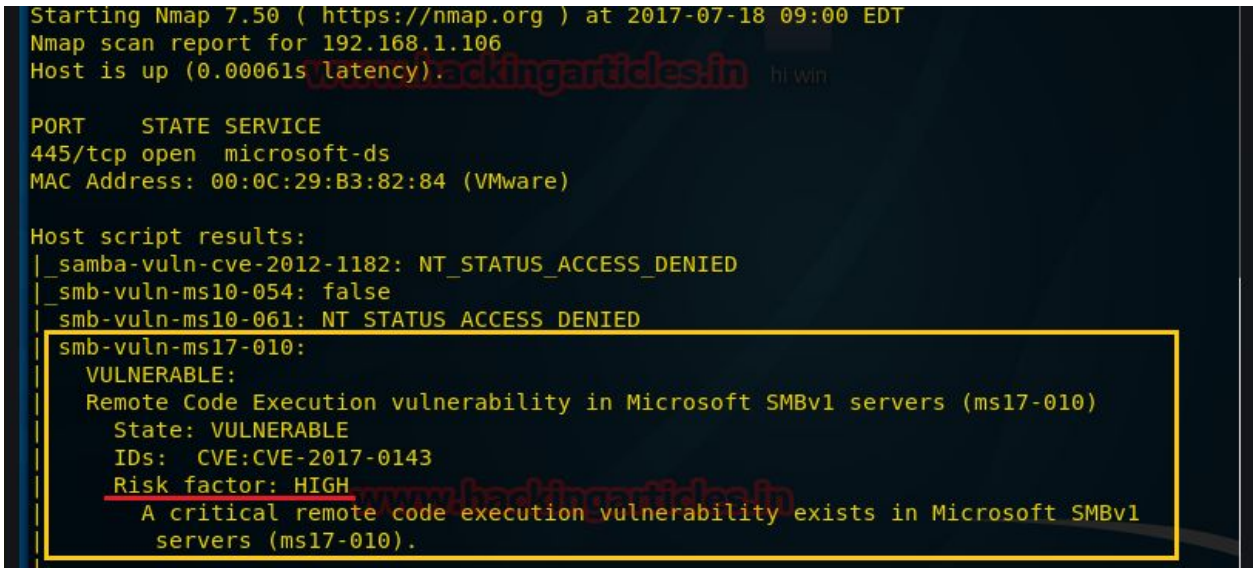


Figure 11(HackTheBox)

With the amount of enthusiasm around the Cyber Security field and Computer Science as a whole and the amount of outdated systems, there will always be utilization of the threat known eternally as, EternalBlue.

Could There Be Something More Devastating?

EternalBlue was just the tip of the iceberg when it came to the leak of exploits from the ShadowBrokers hacking group. Along with EternalBlue, there was an entire Eternal family of exploits that targeted similar vulnerabilities within the Windows SMB, six exploits to be exact(Security News - Trend Micro USA, Oct. 2019). With the way SMB packets were crafted, the developers at Windows would have had no idea that anything was going to happen until it was too late. The problems that were associated with the SMB were indeed eventually fixed, but at what cost? Are they still a threat?

However, exploits such as EternalBlue unfortunately are not the only concern when it comes to Cyber Security threats and exploits. With so many various techniques used to exploit systems

such as Buffer Overflows and DLL Injections, systems are not the only thing that can be exploited. There are also a number of exploitation techniques that use indirect methods to exploit systems through web servers, databases, and other various components on a network that can be linked back to the source system and compromise the host.

As of now, it's hard to say for sure if there will ever be something more devastating than what the EternalBlue exploit did throughout the world and it's even more difficult to know if there are vulnerabilities that exist in systems that we still do not know about. The problem with the Operating System of a computer, or the in and outs of a Protocol such as the SMB client is that they are so detailed and so complex that there is bound to be something, somewhere, that will break.

Conclusion

Prior knowledge of the OS came mainly from the Operating Systems class. Because the class is only 15 weeks long, this means each section of the OS had to be covered at a high level in order for students to gain a good understanding of the OS as a whole. As such, one of the learning goals while exploring this exploit was to deepen knowledge of certain parts of the OS—specifically, security, files,I/O, virtual memory, and memory management. Those sections were chosen because the group showed the most interest in security, and the exploit that dipped into all three of these while being the most prolific happened to be EternalBlue. In the brainstorming period for this project, the article about NotPetya was brought into discussion, which is how the investigation of EternalBlue came to be. However, researching an exploit proved to be challenging at times, especially when source code was hard to find.

Because this was an exploit developed by the NSA, source code was relatively difficult to find. Thankfully, a github repo as well as other articles exploring EternalBlue were found, making exploring EternalBlue and explaining it in a more accessible context easier to accomplish. Logistical challenges arose when delegating work for each team member, and when making the paper more cohesive as a whole. To cater both to productivity as well as interest, team members made their interest known in what they wanted to contribute to the paper, and negotiations for sections were made if more than one person was interested in a particular section. Once it was all parsed out to everyone, the researching and writing began, making notes to reorganize things once the bulk was written. After the first draft was completed, a meeting was held where organization and cohesion were addressed, making the draft flow better overall.

EternalBlue deepened the foundational knowledge gained from class by delving deeper into the specific topics listed above. A knowledge of what the exploit does, how it infects computers, how to stop it, and what other descendants have cropped up were helpful as an introductory look into computer security, and it proved to be an interesting exercise for those in the group already well-versed in security. The dive into files began with the initial research, contrarily, into fileless malware, as it was interesting to see how a malware that doesn't leave a file can continue to corrupt and encrypt the files from local memory registers. By hiding in memory registers and causing trouble for anti-virus, this was also a dip into memory management. Afterwards, the research of how ransomware works was useful because it explained in further detail how those files get decrypted, but it also showcased the world of merciless ransomware that doesn't unlock even when the victim pays the money, causing total data loss.

Overall, this research over types of Malware while using EternalBlue as an example to explore security, memory management, files, I/O, and virtual memory proved to be a worthwhile research project to pursue, not only cementing prior knowledge, but showing how to apply that knowledge in the context of security. Future work done could include a comparison of the descendants of EternalBlue, especially the ones not covered but shown in Figure 2. Highlighting and isolating these differences could be the main strategy in protecting against the idiosyncrasies of these attacks in the future. Is it possible that EternalBlue can be patched

against completely, squashing any attempt at using the exploit in the future? Unfortunately, probably not, but that's no reason to stop trying. Education like this is the best way to protect against exploits like this, because one has to learn how the intricacies of the OS build itself up before one can examine how an exploit tears it back down.

Bibliography

02, B. W. G. S. J., Authors, Researcher, W. G. S. S. T., Sanchez, W. G., Researcher, S. T., & Us, C. (2017, June 2). *MS17-010: EternalBlue's Buffer Overflow in SRV Driver*. https://www.trendmicro.com/en_us/research/17/f/ms17-010-eternalblue.html.

Boyanov, P. (2018). EDUCATIONAL EXPLOITING THE INFORMATION RESOURCES AND INVADING THE SECURITY MECHANISMS OF THE OPERATING SYSTEM WINDOWS 7 WITH THE EXPLOIT ETERNALBLUE AND BACKDOOR DOUBLEPULSAR. *Association Scientific and Applied Research*, *14*, 34.

Danen, Vincent. “Learn to Use Extended File Attributes in Linux to Boost Security.” TechRepublic, TechRepublic, 15 Dec. 2009, www.techrepublic.com/blog/linux-and-open-source/learn-to-use-extended-file-attributes-in-linux-to-boost-security/.

Deland-Han. (2020, September 08). IPC$ share and null session behavior - Windows Server. Retrieved November 15, 2020, from https://docs.microsoft.com/en-us/troubleshoot/windows-server/networking/inter-process-communication-share-null-session

“Detecting Petya/NotPetya Ransomware.” LogRhythm, 19 Feb. 2020, logrhythm.com/blog/detecting-petya-notpetya-ransomware/.

“EternalBlue - Everything There Is To Know.” Check Point Research, 4 Feb. 2019, research.checkpoint.com/2017/eternalblue-everything-know/.

“EternalBlue: The NSA-developed Exploit That Just Won’t DIe.” SentinelOne, 4 May 2020, https://www.sentinelone.com/blog/eternalblue-nsa-developed-exploit-just-wont-die/.

Greenberg, A. (2018, August 22). The Untold Story of NotPetya, the Most Devastating Cyberattack in History. Retrieved November 15, 2020, from https://www.wired.com/story/notpetya-cyberattack-ukraine-russia-code-crashed-the-world/

Islam, Ali. “SMB Exploited: WannaCry Use of ‘EternalBlue.’” FireEye, 26 May 2017, www.fireeye.com/blog/threat-research/2017/05/smb-exploited-wannacry-use-of-eternalblue.html.

Nakashima, E., & Timberg, C. (2017). NSA officials worried about the day its potent hacking tool would get loose. Then it did. *The Washington Post*.

"NotPetya Technical Analysis." LogRhythm, 19 Feb. 2020, logrhythm.com/blog/notpetya-technical-analysis/.

Openspecs-Office. "[MS-SMB]: Client Request Extensions." [MS-SMB]: Client Request Extensions | Microsoft Docs, docs.microsoft.com/en-us/openspecs/windows_protocols/ms-smb/a00d0361-3544-4845-96ab-309b4bb7705d?redirectedfrom=MSDN.

"Putting the Eternal in EternalBlue: Mapping the Use of the Infamous Exploit." Security News - Trend Micro USA, Oct. 2019, www.trendmicro.com/vinfo/us/security/news/vulnerabilities-and-exploits/putting-the-eternal-in-eternalblue-mapping-the-use-of-the-infamous-exploit.

Ren, A., Liang, C., Hyug, I., Broh, S., & Jhanjhi, N. Z. (2020). A Three-Level Ransomware Detection and Prevention Mechanism. *EAI Endorsed Transactions on Energy Web*, *7*(26).

Saad, S. (2019). "JSLess: A Tale of a Fileless Javascript Memory-Resident Malware." *Information Security Practice and Experience. 15th International Conference (ISPEC 2019)*.

Sanjay, B. (2018). "An Approach to Detect Fileless Malware and Defend its Evasive mechanisms." *2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS)*.

Security 101: The Rise of Fileless Threats that Abuse PowerShell. (2017, June 1). Retrieved November, 2020, from https://www.trendmicro.com/vinfo/pl/security/news/security-technology/security-101-the-rise-of-fileless-threats-that-abuse-powershell

"Security Primer EternalBlue." MS-ISAC, Jan. 2019.

Smelcer, J. (2017). Rise of fileless malware (Order No. 10642524). Available from ProQuest Dissertations & Theses Global. (1990643022). Retrieved from https://search-proquest-com.proxybz.lib.montana.edu/docview/1990643022?accountid=28148

Tedhudek. "_MDL (Wdm.h) - Windows Drivers." _MDL (Wdm.h) - Windows Drivers | Microsoft Docs, docs.microsoft.com/en-us/windows-hardware/drivers/ddi/wdm/ns-wdm-_mdl.

Trend Micro Incorporated (2017). "Ransomware: Past, Present, and Future." Last accessed on November 3, 2020 from

https://documents.trendmicro.com/assets/wp/wp-ransomware-past-present-and-future.pdf

Trend Micro Incorporated. (2020, October 18). Putting the Eternal in EternalBlue: Mapping the Use of the Infamous Exploit. Retrieved November 15, 2020, from https://www.trendmicro.com/vinfo/us/security/news/vulnerabilities-and-exploits/putting-the-eternal-in-eternalblue-mapping-the-use-of-the-infamous-exploit

Vulnerability Details : CVE-2017-0147. https://www.cvedetails.com/cve/CVE-2017-0147/.

“What Is a Buffer Overflow: Attack Types and Prevention Methods: Imperva.” Learning Center, Imperva, 29 Dec. 2019, www.imperva.com/learn/application-security/buffer-overflow/.

“What Is SMBv1 and Why You Should Disable It: STEALTHbits.” Stealthbits Technologies, 3 Sept. 2020, stealthbits.com/blog/what-is-smbv1-and-why-you-should-disable-it/.

Wikimedia Foundation. (2020, September 22). *EternalBlue*. Wikipedia. https://en.wikipedia.org/wiki/EternalBlue.